

# Unveiling Principles of Neural Computations: From Biological to Artificial Intelligence, and Back

Dissertation

zur

Erlangung der Doktorwürde

(Dr. sc. UZH ETH Zürich)

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät der Universität Zürich

und der

Eidgenössischen Technischen Hochschule Zürich

von

Giorgia Dellaferrera

Aus

Italien

Promotionskommission

Giacomo Indiveri

Valerio Mante

Evangelos Eleftheriou

Angeliki Pantazi

Emre Neftci

Wulfram Gerstner

Zürich, 2023



University of  
Zurich <sup>UZH</sup>

**ETH** zürich



Giorgia Dellaferrera

# Unveiling Principles of Neural Computations: From Biological to Artificial Intelligence, and Back

**Doctoral Thesis**

Institute of Neuroinformatics  
Universität Zürich & ETH Zürich

**PhD committee members**

Prof. Dr. Giacomo Indiveri  
Prof. Dr. Valerio Mante  
Prof. Dr. Evangelos Eleftheriou  
Dr. Angeliki Pantazi  
Prof. Dr. Wulfram Gerstner  
Prof. Dr. Emre Neftci

January 2023

# Abstract

The algorithms of deep learning have their roots grounded in simplified models of brain circuits. With the advent of powerful computational resources, the design of artificial neural networks departed from their biological inspiration to focus on engineering powerful machines able to solve complex tasks with accuracy comparable to or even exceeding human performance. However, in the past decade, neuroscience and artificial intelligence have found again their initial symbiotic relationship. Neural mechanisms are used as a source of inspiration for the development of deep network architectures and training algorithms, to find computational models with improved robustness to noise, the ability to generalize, and to perform continual learning. On the other hand, biologically inspired networks offer a new perspective on understanding how the brain learns and processes information, potentially leading to innovative therapeutic applications. Additionally, standard deep learning models have been successfully applied to the analysis and understanding of neural data. A remarkable example is the possibility of using artificial networks as models of some areas of the brain, to predict, and even drive, neural activity from sensory stimuli. In this thesis, I delve into the interplay between deep learning and neuroscience. First, I introduce the basic concepts of artificial neural networks from a biological perspective, and I overview existing literature on biologically plausible learning and computational models of the cortex. Second, I present my contributions to the field of brain-inspired learning, including an optimizer inspired by synaptic integration, a learning rule solving the biologically unrealistic aspects of backpropagation, and the application of self-supervised spiking networks to the task of blind source separation. Third, I introduce a comparative analysis of the robustness of cortical neurons and artificial neural networks to adversarial noise, showing how deep learning can be used to manipulate neural activity toward prescribed patterns. Fourth, I present a framework to quantify the gap between the performance of humans and state-of-the-art artificial models, focusing on the field of language. Finally, I reflect on the future steps of artificial intelligence and more specifically on the debate on whether neuroscience is necessary to drive breakthrough progress for deep learning.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Part 1 Introduction</b>	<b>1</b>
<b>1 Introduction to the Thesis Structure</b>	<b>2</b>
1.1 The context . . . . .	2
1.2 Thesis contribution . . . . .	4
<b>2 Artificial Neural Networks: Methods, Success and Limitations from a biological perspective</b>	<b>5</b>
2.1 Biological roots of artificial neural networks . . . . .	5
2.2 Plasticity and learning . . . . .	6
2.2.1 Learning in biology . . . . .	7
2.2.2 Learning in artificial neural networks . . . . .	8
2.3 Network architectures under a biological lens . . . . .	13
2.3.1 Similarities between biological circuits and artificial networks . . . . .	13
2.3.2 Fully connected neural networks . . . . .	17
2.3.3 Convolutional neural networks . . . . .	17
2.3.4 Recurrent neural networks . . . . .	21
2.3.5 Transformers . . . . .	23
2.4 Representative examples of state-of-the-art achievements . . . . .	24
2.4.1 Computer vision benchmarks . . . . .	25
2.4.2 Playing games: AlphaGo . . . . .	26
2.4.3 Writing human-like text: GPT3, LaMDA . . . . .	26
2.4.4 Driving plasma fusion: Tokamak and reinforcement learning . . . . .	26
2.4.5 Predicting protein structure: AlphaFold . . . . .	27
2.4.6 Medical diagnosis: predicting pathologies before onset . . . . .	28
2.4.7 Generating art: image and music . . . . .	30
2.5 Longstanding challenges of artificial neural networks . . . . .	31
2.5.1 Catastrophic forgetting . . . . .	31
2.5.2 Lack of out-of-distribution generalization . . . . .	33
2.5.3 Susceptibility to adversarial attack . . . . .	35
2.5.4 Temporal and spatial credit assignment problem . . . . .	35
2.5.5 Energy and data-hungry models . . . . .	37
<b>3 Biologically Inspired Algorithms for Neural Networks</b>	<b>40</b>
3.1 Principles of biological computations . . . . .	40
3.2 Addressing catastrophic forgetting with biologically inspired strategies . . . . .	41
3.2.1 Approaches inspired by synaptic consolidation . . . . .	41
3.2.2 Approaches inspired by the replay mechanism . . . . .	42
3.3 Towards bio-plausible training for ANNs . . . . .	44
3.3.1 The neural gradient representation by activity differences (NGRAD) framework and predictive coding . . . . .	45

3.3.2	Supervised learning . . . . .	47
3.3.3	Self supervised learning . . . . .	58
3.4	Addressing energy consumption: spiking neural networks . . . . .	60
3.4.1	The potential in the efficiency of SNNs compared to ANNs . . . . .	62
3.4.2	Neuronal dynamics and elements of neuronal systems . . . . .	62
3.4.3	Leaky integrate and fire neuron models . . . . .	63
3.4.4	Spike encoding . . . . .	64
3.4.5	Power efficiency of spiking neural networks . . . . .	64
3.4.6	The spiking neural unit . . . . .	65
<b>4</b>	<b>Modeling the Mind</b> . . . . .	<b>66</b>
4.1	Digital twins of the brain . . . . .	66
4.1.1	The encoding and the decoding problems . . . . .	66
4.1.2	Examples of ANNs to study biological vision . . . . .	67
4.1.3	Brain-Score . . . . .	68
4.1.4	Substitute models to drive perception . . . . .	68
4.2	Limitations of deep learning applied to neuroscience . . . . .	69
4.2.1	No free lunch from deep learning in neuroscience . . . . .	69
4.2.2	A roadmap for neuroscience research . . . . .	70
4.2.3	The brain as a highly social network . . . . .	70
	<b>Part 2 Results</b> . . . . .	<b>72</b>
<b>5</b>	<b>GRAPES: a Biologically Inspired Optimizer</b> . . . . .	<b>73</b>
5.1	Introduction . . . . .	73
5.2	Results . . . . .	75
5.2.1	The GRAPES algorithm . . . . .	75
5.2.2	Convergence analysis of GRAPES applied to SGD . . . . .	82
5.2.3	Main steps of the convergence analysis . . . . .	84
5.2.4	Simulation results on handwritten digit classification . . . . .	86
5.2.5	Performance under various learning rules and data sets . . . . .	87
5.2.6	Mitigation of catastrophic forgetting . . . . .	91
5.2.7	Application of GRAPES to biologically inspired neural networks . . . . .	93
5.2.8	Behaviour of GRAPES under hardware constraints . . . . .	95
5.3	Discussion . . . . .	95
5.4	Methods . . . . .	99
<b>6</b>	<b>PEPITA: a Biologically Plausible Training Scheme</b> . . . . .	<b>105</b>
6.1	Introduction . . . . .	105
6.2	Background and related work . . . . .	106
6.2.1	Credit assignment in conventional networks . . . . .	106
6.2.2	Credit assignment without random feedback path . . . . .	107
6.3	Error-driven input modulation . . . . .	107
6.3.1	Overview of the proposed learning rule . . . . .	107
6.3.2	The learning rule . . . . .	108
6.3.3	Extension to convolutional layers . . . . .	109
6.4	Results . . . . .	110
6.4.1	Methods . . . . .	110
6.4.2	Experimental results . . . . .	110
6.4.3	Analytic results . . . . .	113
6.4.4	Final weight distribution . . . . .	118
6.5	Discussion . . . . .	119
<b>7</b>	<b>Dendritic Learning for Blind Source Separation</b> . . . . .	<b>121</b>

7.1	Introduction . . . . .	121
7.2	Results . . . . .	122
7.2.1	Learning of repeated input patterns by a two-compartment neuron model . . . . .	122
7.2.2	Synthesized and natural auditory stimuli . . . . .	124
7.2.3	Learning of mixture sounds in the network model . . . . .	126
7.2.4	Experiment 1: sound segregation with single and multiple mixtures of synthesized sounds . . . . .	127
7.2.5	Experiment 2: sound segregation with alternating multiple mixtures of synthesized sounds . . . . .	130
7.2.6	Experiment 3: effect of temporal delay in target presentation with synthesized sounds . . . . .	131
7.2.7	Experiment 4: sound segregation with single and multiple mixtures of real-world sounds . . . . .	133
7.2.8	Experiment 5: image segregation with single and multiple mixtures of real-world images . . . . .	133
7.3	Discussion . . . . .	135
7.4	Materials and methods . . . . .	138
7.4.1	Datasets . . . . .	138
7.4.2	Neuron model . . . . .	138
7.4.3	Network architecture . . . . .	139
7.4.4	Experimental settings and performance measure . . . . .	140
<b>8</b>	<b>Fooling the Primate Brain with Small, Targeted Image Manipulation</b>	<b>142</b>
8.1	Introduction . . . . .	142
8.2	Results . . . . .	143
8.2.1	Deceptive images of human faces evoked monkey face-like neuronal responses . . . . .	143
8.2.2	Deceptive images also led to target category-like responses in two other directions . . . . .	148
8.2.3	Deceptive images reveal mismatch between primate vision and ANNs . . . . .	149
8.3	Discussion . . . . .	153
<b>9</b>	<b>Human or Machine? Turing Tests for Vision and Language</b>	<b>156</b>
9.1	Introduction . . . . .	156
9.2	Related Works . . . . .	157
9.2.1	Glimpse of the 70-year history of Turing test . . . . .	157
9.2.2	AI versus humans in vision tasks . . . . .	158
9.2.3	AI versus humans in language tasks . . . . .	158
9.3	Experiments . . . . .	159
9.3.1	Color Estimation . . . . .	160
9.3.2	Object detection . . . . .	160
9.3.3	Attention prediction . . . . .	161
9.3.4	Image captioning . . . . .	161
9.3.5	Word associations . . . . .	161
9.3.6	Conversation . . . . .	162
9.4	Results . . . . .	162
9.4.1	Color estimation . . . . .	163
9.4.2	Object detection . . . . .	163
9.4.3	Attention prediction . . . . .	164
9.4.4	Image captioning . . . . .	164
9.4.5	Word associations . . . . .	164
9.4.6	Conversation . . . . .	165
9.5	Discussion . . . . .	165
9.6	Methods and Additional Results for the Conversation Task . . . . .	166
9.6.1	Results . . . . .	167

9.6.2	Results comparing AMT and in-person experiments . . . . .	169
9.6.3	Results for the human judge on different length of conversations . . . . .	169
9.6.4	Dataset collection: instructions to human participants . . . . .	175
9.6.5	Dataset collection: prompt and settings for GPT3text-davinci002 and GPT3text-curie-001 . . . . .	177
9.6.6	Human judges responses collection . . . . .	178
9.6.7	Example of collected conversations . . . . .	178
<b>Part 3 Discussion</b>		<b>185</b>
<b>10 Discussion</b>		<b>186</b>
10.1	Summary and Outlook . . . . .	186
10.2	A parallel quest: the design of neuromorphic hardware . . . . .	187
10.3	How crucial is neuroscience for the progress of AI . . . . .	188
10.4	How crucial is AI for the progress of neuroscience . . . . .	188
10.5	Some unanswered, and possibly unanswerable, questions . . . . .	189
<b>11 Author Contributions</b>		<b>191</b>
<b>Bibliography</b>		<b>192</b>

## Part 1

# Introduction



# Chapter 1

## Introduction to the Thesis Structure

This thesis addresses the complex and increasingly entangled interplay between artificial intelligence (AI) and biological networks. It is composed of two main topical areas: “from neuroscience to AI” and “from AI to neuroscience” (Figure 1.1). The first part focuses on biologically inspired algorithms for training both artificial and spiking neural networks. The second part focuses on neural prediction and control using artificial neural networks, mapped in particular to the visual cortex of rodents and primates. The organization of the introduction reflects the topical structure of the thesis. I first present an overview on renowned and recent biologically inspired algorithms to both improve the performance of BP and to train neural networks by avoiding the biologically unrealistic aspects of BP. Then, I present milestone works in which artificial network models have been successfully employed to predict and control neural activity, in particular in the visual cortex of rodents and primates. Before delving into the technical aspects of bio-inspired neural networks and neural networks for investigating neuroscience, I give a general introduction on the basic concepts of machine learning, outlining the connections with biological circuits. I also point out some of the impressive achievements obtained with artificial intelligence on very complex tasks. I emphasise these results and contrast them with the failure on seemingly easy tasks such as generalize to different input samples or remember sequences of tasks, related to longstanding limitations of neural networks, such as adversarial attack and catastrophic forgetting.

The introduction serves to give the basis for understanding the technical chapters of the thesis. Chapters 5, 6 and 7 introduce original ideas developed during my Ph.D. to train neural networks based on principles of synaptic integration, local computation, and dendritic computation, respectively. Next, Chapter 8 introduces how deep networks can be used to investigate robustness to adversarial noise in the primate brain. The quest for learning algorithms that can be implemented by the brain is undeniably motivated by trying to answer one of the greatest problems in science: understanding *how the brain computes*. Additionally, a deeper understanding of learning in the brain could help guide the treatment of human brain diseases. Furthermore, the development of biologically inspired algorithms is often motivated as crucial to narrow the gap in performance between AI and the animal brain, but how can we evaluate this gap under a general framework? Chapter 9 proposes a novel methodology resorting to the Turing test to quantify the gap in performance between humans and machines, focusing in particular in conversational tasks.

### 1.1 The context

Artificial neural networks (ANNs) are learning models inspired by the biological circuits that constitute the animal brain. The Backpropagation of the Error (BP) algorithm [Rumelhart et al., 1995] is the most commonly used training scheme for ANNs. Since the introduction of BP, neuroscientific principles have been proposed as a fertile source of ideas to push forward the design of ANNs by either enriching the dynamics of BP or devising alternative training schemes. Introducing principles of biological learning in the training of ANNs has been demonstrated to address issues commonly faced by BP, such as overfitting and catastrophic forgetting. A striking ex-

ample is dropout, the regularization process preventing overfitting, which was motivated by the theory of sexual reproduction [Srivastava et al., 2014]. A second example is replay, a mechanism inspired by the reactivation in the animal brain of neuronal activity patterns representing memories, which has been proven effective in mitigating the phenomenon of catastrophic forgetting [van de Ven et al., 2020]. In parallel, studies of underlying mechanisms of biological learning have motivated criticisms of BP as a biologically unrealistic strategy of training [Crick, 1989]. Indeed, several key aspects of BP cannot be implemented by biological neurons: symmetric and bidirectional connectivity paths, non-local parameter updates, freezing of network activity during plasticity, and lack of online learning. Motivated by these reasons, the community has proposed several biologically inspired learning schemes alternatives to BP to alleviate some of the above-mentioned issues [Lillicrap et al., 2016, Nokland, 2016, Frenkel et al., 2019]. In this context, this thesis presents both an original way to integrate principles of biological learning in the optimization process of BP – the GRAPES optimizer [DellaFerrera et al., 2022b], inspired by dendritic integration – and a biologically plausible training scheme for ANNs – the PEPITA algorithm [DellaFerrera and Kreiman, 2022].

On the other hand, although ANNs are extremely simplified models compared to biological networks, they offer an unprecedented tool to investigate the mechanisms underlying brain dynamics. ANNs are the current best models of cortical activity and the finest predictors of neural activity in the ventral stream [Yamins et al., 2014, Cadena et al., 2019]. Groundbreaking paradigms have demonstrated that ANN “substitute models” fitted to neural activity can be used to synthesize images that guide small neural populations toward a specific response [Bashivan et al., 2019, Walker et al., 2019]. In my Ph.D. work, I have contributed to strengthening this research line in three main directions: understanding the patterns of neural activity that drive perception responses in primates [Yuan et al., 2020], reconstructing visual stimuli from hidden activations, and predicting neural activity in the rodents’ visual cortex in absence of visual stimulation. In this thesis, I describe in particular how artificial models mapped to the monkey brain can be used to fool perception in the primate brain, and how this paradigm can be used to rigorously compare the robustness to adversarial noise between biological and artificial brains.

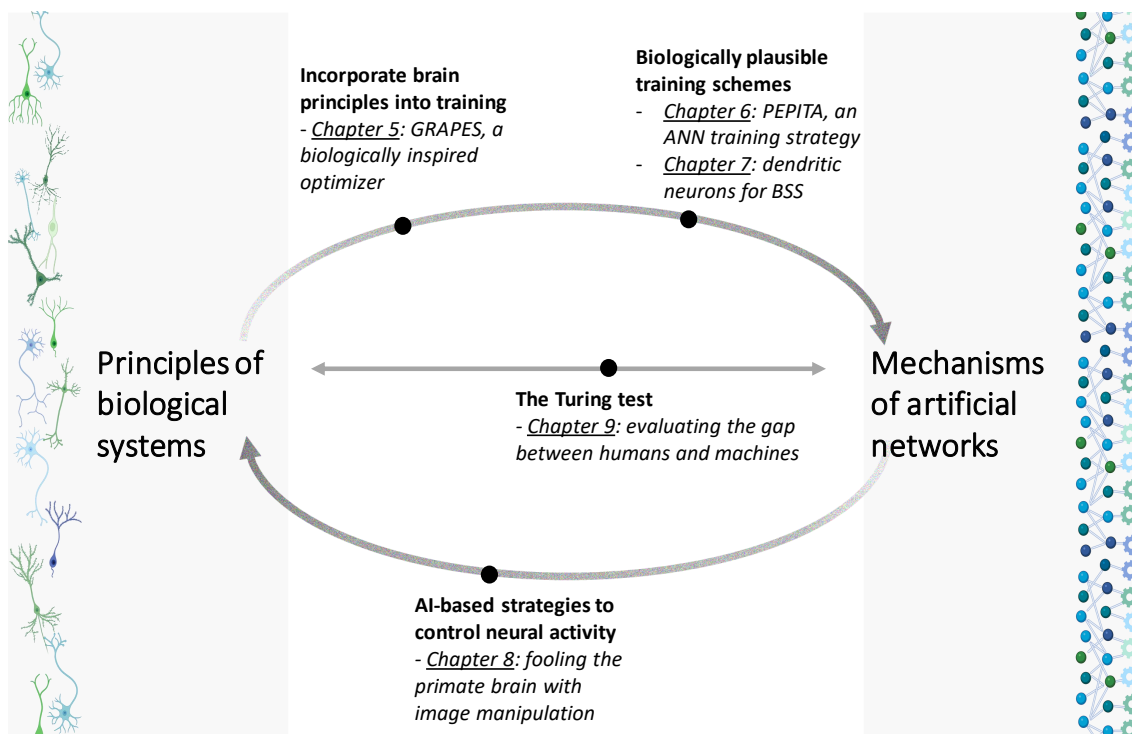


Figure 1.1: Biological intelligence to artificial intelligence, and back: the framework of this thesis.

## 1.2 Thesis contribution

This thesis summarizes the research I performed during my Ph.D. at the Institute of Neuroinformatics in the group of G. Indiveri (University of Zurich and ETH Zurich) and IBM Research Zurich in the group of E. Eleftheriou and A. Pantazi between 2020 and 2022. My doctoral studies also included a research visit at the Harvard Medical School and Boston Children’s Hospital, a summer school at the Brain Mind and Machine institute mentored by M. Zhang (Harvard and MIT), as well as the continuation of a collaboration with the group of T. Fukai at the Okinawa Institute for Science and Technology started in 2019. The ideas and results presented in this thesis are the fruit of my interaction and work with all these groups.

My Ph.D. research lies at the intersection between Deep Learning and Neuroscience. Specifically, I worked in two directions. On one hand, my work aimed at designing biologically inspired learning algorithms for neural networks, both spiking and non-spiking. On the other hand, I investigated how deep neural networks can be applied to neuroscience research, to gain a better understanding of the mechanisms governing information processing and learning in the brain.

In Chapter 1 I provide a general introduction to the interplay of Artificial Intelligence and Neuroscience. I explain the biological roots of artificial neural networks, give an overview of the basic concept of machine learning and state-of-the-art architectures providing links with neuroscience, and introduce existing biologically inspired training algorithms. Furthermore, I review existing literature on artificial models of the brain, their design choices, achievements and limitations.

The first part of the thesis contains three chapters on neuroscience-inspired artificial intelligence. In Chapter 5 I present a biologically inspired optimizer that can be applied to both artificial neural networks trained with Backpropagation or with other biologically inspired training schemes and to spiking neural networks. This optimizer is shown to lead to improved accuracy, convergence rate, mitigated catastrophic forgetting and improved robustness to noise and reduced precision. This work was published in Nature Communications in 2022 as a result of my work at IBM Research Zurich and the Institute of Neuroinformatics in Zurich [Dellaferrera et al., 2022b].

In Chapter 6 I propose a novel training scheme for artificial neural networks which solves the biologically nonrealistic aspects of Backpropagation. The algorithm relies on using only forward computations, leading to a learning rule which is local and not affected by the weight transport problem. This work was performed during my research stay in the Kreiman Lab and has been published at ICML 2022 [Dellaferrera and Kreiman, 2022].

In Chapter 7 I present an application of a self-supervised learning scheme for multi-compartmental spiking neurons to the task of blind source separation. Our results show that a simple learning model embedded in a simple architecture is able to reproduce some of the behaviors in audio experiments obtained with human subjects. This work was started as part of an internship in the Fukai unit and later completed during my Ph.D.. It has been published in Frontiers of Neuroscience in 2022 [Dellaferrera et al., 2022a].

In the second part of the thesis, I investigate neuroscientific open questions. Specifically, in Chapter 8 I describe how machine learning models can be used to investigate the robustness of neurons in the visual cortex with respect to minimal perturbation of the visual stimuli. We compare the vulnerability to adversarial attack between artificial and biological neural networks, and we find that much larger perturbation budgets are needed to change the perception in the primate brain compared to artificial circuits. This work has been started by L. Yuang and W. Xiao and I later worked on expanding the computational findings of the paper, which is currently unpublished [Yuan et al., 2020].

Finally, in Chapter 9 I focus on quantifying the gap between human and state-of-the-art deep learning models on the task of conversation. I propose a novel framework to implement the Turing test for chatbots, using Amazon Mechanical Turk to recruit human judges. This work started as a collaboration with M. Zhang and G. Kreiman at the BMM summer school in 2022 and continued as a larger collaboration project involving several tasks. The results are currently unpublished [Zhang et al., 2022b].

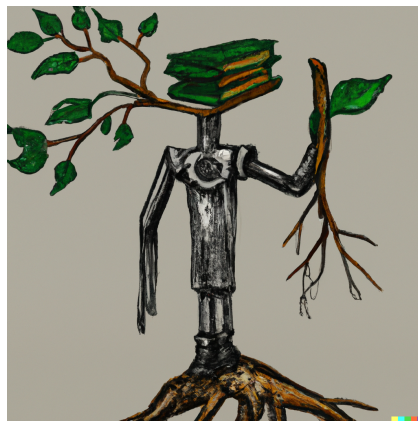
To conclude, in the Discussion I propose some considerations on the future of the interplay of Deep Learning and Neuroscience, and on the concept of intelligence applied to artificial neural networks.

## Chapter 2

# Artificial Neural Networks: Methods, Success and Limitations from a biological perspective

Here I review the impressive achievements of artificial networks and the current challenges of AI. I start with a brief introduction to the most commonly used strategy to train ANNs, the backpropagation of the error algorithm (BP) and common architectures in Machine Learning (ML), highlighting their connections to biological circuits. To enable reading of the core chapters of the thesis to readers unfamiliar with machine learning, I provide an overview of the basic concepts that will be used in the technical sections. I will then summarize some of the state-of-the-art results obtained with BP-trained networks in different domains, including computer vision and natural language processing. Finally, I will analyze the current main challenges of ML: catastrophic forgetting, out-of-distribution generalization, adversarial attack, high power consumption, and lack of biological plausibility. In contrast, the animal brain is able to perform continual learning, generalize to different distributions, and is robust to noise. Additionally, the brain operates using very limited energetic resources. This strongly motivates the community to get inspiration from how the brain learns and processes information to model and improve strategies to train neural networks.

**Biological roots of artificial intelligence.** DALL·E prompt: A painting of a robot that has green leaves on the head and tree roots as legs, standing on top of a book.



### 2.1 Biological roots of artificial neural networks

The field of computational neuroscience and neuromorphic computing lies at the intersection of biology, machine learning and neural network theory. While the research line focusing on the development of biologically plausible AI algorithms has been enhanced only in the past decade, the roots of neural networks are grounded in biology. Artificial neural networks were originally proposed as an abstract and simplified mathematical description of the mechanisms of learning and input processing taking part in biological circuits [Wythoff, 1993]. In 1943, McCulloch, a neurophysiologist, and Pitts, a logician, proposed a simplified mathematical model of a neuron working in discrete time and with boolean values. The basic assumption is that, as the nervous activity presents an ‘all-or-none’ behavior, neural events can be treated by means of propositional logic. The so-called MCP neuron model served as a basic cell of a wired network and was introduced

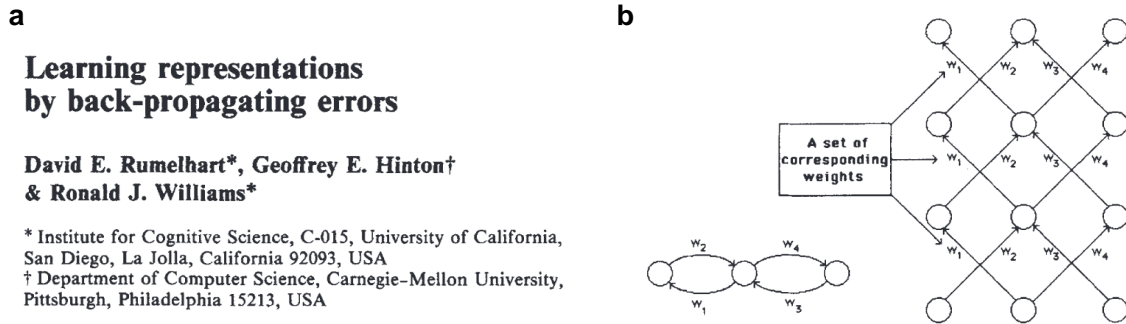


Figure 2.1: (a) Headline of the original backpropagation paper [Rumelhart et al., 1986]. (b) Schematics of the BP in [Rumelhart et al., 1986]

with the goal of understanding how the brain produces complex patterns by using many interconnected cells. [McCulloch and Pitts, 1943a]. The next step in modeling circuit learning was taken by Rosenblatt, a psychologist, in 1958, when he proposed the single-layer perceptron by building on the MCP neuron model. The perceptron, now considered the first modern neural network, is a hierarchical structure composed of layers of neurons in which the output of each neuron is given by the weighted sum of its input, compared to a fixed threshold [Rosenblatt, 1958]. With the perceptron, Rosenblatt demonstrated that mathematical models of neurons can learn from data by adjusting the synaptic weights. In particular, the perceptron could solve classification tasks for linearly separable classes. The interest in neural network learning escalated a couple of decades later when in 1986 Rumelhart proposed the backpropagation (BP) algorithm to train fully connected networks [Rumelhart et al., 1986]. Figure 2.1 shows (a) the headline of the original paper proposing the BP algorithm and (b) a sketch of the forward and backward computation as illustrated in the manuscript [Rumelhart et al., 1986]. Section 2.2.2 provides an introduction to the BP algorithm in supervised learning settings. In supervised settings, training consists in learning a mapping between an input  $x$  and an output  $y$ , so that the output of the network matches the target ( $t$ ) - or label- associated with the input. For example, supervised tasks include mapping an image or a video to a class, a caption, a region of interest (ROI), or a command, or mapping a sound sample to text or a picture.

In Rumelhart’s work, neural networks stopped serving as tools to model biological plausible properties, and their role shifted to working as a tool in pursuing the actual goal of optimizing the performance of a model for an application [Gregory Ashby and Helie, 2011]. Following the introduction of backpropagation, neural network theory rapidly diverged from rigorously modeling and explaining the operating principles of biological circuits and rather focused on designing highly accurate and specialized machines to solve increasingly complex and narrow tasks. The driving force of machine learning development quickly became the design of algorithms that could perform close to - or even better than - humans in highly specific tasks. As a consequence of this role shift, while neural networks present features that retain similarities with biological circuits - *e.g.*, learning occurs through modification of connection strength among neurons -, they rely on several requirements that are incompatible with the dynamics of neurons in the brain.

## 2.2 Plasticity and learning

Learning in the brain occurs through the adjustment of the connections between neurons, *i.e.* the synapses. Analogously, learning in artificial networks consists in updating the parameters connecting the nodes in order to minimize a loss function. Credit assignment refers to the process of modifying the synaptic strength between neurons - or nodes - in a network to minimize the

network's output error in a specific task. Here, I review learning rules governing the dynamics of weight change in biology, spiking neural networks, and artificial neural networks

### 2.2.1 Learning in biology

Unveiling the mechanisms through which the animal brain learns and adapts to new tasks and contexts is an intriguing open challenge in neuroscience. While a comprehensive understanding of how learning and the formation of new memories occur in complex biological systems remains elusive, both experimental and computational progress has been made in studying principles of biological computations. Two fundamental aspects of plasticity in biological circuits are the locality of modulatory signals and spike-based information propagation. The review in [Khacef et al., 2022] provides a comprehensive overview of spike-based learning models. In the following sections, I report a selection of the fundamental rules and requirements studied in computational neuroscience.

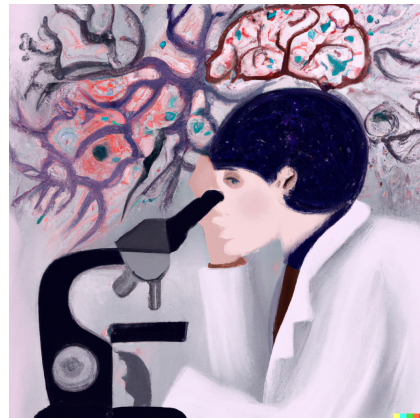
**Locality of learning rules** The principle of locality in biological systems requires that all the information necessary to compute the update of a synapse is directly accessible in space and in time to the synapse. The local nature of computations emerges from the physical constraints of biological circuits. The local information may be related to the spike time or spike frequency of the pre- and post-synaptic neurons to which the synapse is directly connected. When the synaptic update depends only on the activity of the pre- and post-synaptic neurons, the plasticity dynamics are referred to as 'two-factor learning rule'. Other plasticity schemes take into account an additional third factor which presents a feedback signal (*e.g.*, reward, punishment, or novelty) that modulates the weight update prescribed by the neuronal activity. The third factor, which hence is not local, plays the role of neuromodulators, such as dopamine, that modulate plasticity in the brain [Khacef et al., 2022].

**Spike-based computations and Hebbian learning** In the first half of the 20-th century, studies on the chemical synapses revealed that neural information processing occurs in the form of spikes. Synaptic inputs are integrated into the soma of the neurons, which emits an output spike once the membrane potential reaches a threshold value, propagating along the axon. The information flow among neurons goes from the pre-synaptic axons to the post-synaptic dendrites [Khacef et al., 2022].

In 1949 Hebb published his formal postulate for the neural mechanisms of learning and memory: *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased* [Hebb, 1949]. The principle of increasing synaptic strength between two co-active neurons was hence defined *Hebbian plasticity*. It is also referred to as *Long-Term Potentiation* (LTP). In Hebb's postulate, the time-dependency of the spike is taken into account in the directionality (*i.e.*, A fires before B).

Expanding on Hebb's postulate, other Hebbian-like rules were proposed to take into account mechanisms of synaptic weakening. Among these, the Oja's rule introduces a 'forgetting' parameter and solves the stability problem with a form of local multiplicative normalization for synaptic weights [Oja, 1982]. In this direction, the Bienenstock Cooper Munro (BCM) learning rule states that during pre-synaptic stimulation, the low-frequency activity of the post-synaptic neuron leads to *Long-Term Depression* (LTD) while high-frequency activity would lead to LTP [Bienenstock et al., 1982].

**Inspecting the brain.** DALL-E prompt: A painting of a scientist that looks at a microscope with many neurons, on a background of a painting with a brain.



Experimental and computational evidence supports the hypothesis that synapses have local access to the timings of pre-synaptic and postsynaptic neurons spikes, and that information is contained in the timing of the spikes rather than in their rate [Gerstner et al., 1993, Markram et al., 1995, Khacef et al., 2022]. Specifically, a pre/post pairing with a time difference of 10 ms leads to LTP, while using the same time difference of 10 ms in an inverted post/pre pairing leads to LTD [Markram et al., 1997]. This result led to the development of the *Spike-Timing Dependent Plasticity* (STDP) rule, which models long-term synaptic plasticity as dependent on the relative timing of pre- and postsynaptic action potentials [Song et al., 2000]. If the post-synaptic neuron spikes after the pre-synaptic one ( $\Delta t < 0$ ), the synapse is potentiated. If instead the pre-synaptic neuron spikes after the post-synaptic neuron ( $\Delta t \geq 0$ ), the connection is weakened. The time window in which the spike interaction leads to adjustments in synaptic weight is defined by the time constants  $\tau_+$  and  $\tau_-$ , respectively. Given these elements, the synaptic update of STDP is described by:

$$\Delta W_{STDP} = \begin{cases} A_+ \exp(\frac{\Delta t}{\tau_+}) & \text{if } \Delta t < 0 \\ -A_- \exp(\frac{-\Delta t}{\tau_-}) & \text{if } \Delta t \geq 0 \end{cases} \quad (2.1)$$

where  $A_+$  and  $A_-$  are the maximum amounts of synaptic change.

To take into account the dependence on the repetition frequency of the pairs of spike, Pfister and Gerstner [Pfister and Gerstner, 2006] later introduced the *Triplet rule*, in which LTP and LTD depend on a combination of three pre- and post-synaptic spikes. Two pre- and one post- lead to triplet depression, while one pre- and two post- lead to triplet potentiation. The Triplet-based STDP (T-STDP) is shown to fit experimental data from visual cortical slices as well as from hippocampal cultures and can be mapped to a BCM learning rule. Furthermore, the T-STDP is able to model the spike-rate dependence, which is correlated with the Calcium concentration of the postsynaptic neuron. In particular, the rate-dependency of learning leads LTP to dominate over LTD at high frequencies regardless of precise spike-timing. In biological circuits, the precise spike-timing dependence could be implemented through local processes in the synapses that have access to both the timing information of pre-synaptic spikes and to the postsynaptic spike times. The information on the spike-timing can be received through sensing the neuron’s local membrane voltage changes or by receiving large depolarizations caused by APs that are actively backpropagated in the dendritic tree [Stuart and Sakmann, 1994, Khacef et al., 2022]. A further extension of the STDP rule is the so-called Voltage-based STDP (V-STDP) [Clopath and Gerstner, 2010]. The V-STDP model unifies multiple experimental observations such as post-synaptic membrane voltage dependence, pre-post spike-timing dependence, and post-synaptic rate dependence. Specifically, synaptic changes depend on presynaptic spike arrival and the postsynaptic membrane potential, filtered with two different time constants. Hence, in the V-STDP rule, depression and potentiation are two independent mechanisms whose sum produces the total synaptic change. This plasticity rule leads both to the development of localized receptive fields and to connectivity patterns that reflect the neural code.

## 2.2.2 Learning in artificial neural networks

**Gradient-based learning** In 1998, LeCun and colleagues [Lecun et al., 1998a] wrote “There are several approaches to automatic machine learning but one of the most successful approaches popularized in recent years by the neural network community can be called numerical or gradient-based learning”. Today, more than 30 years later, gradient-based learning is still the most popular – and successful – method to train neural networks. Indeed, Artificial Neural Networks trained with the backpropagation algorithm are a successful example of a gradient-based learning technique. Given a network architecture, gradient-based learning algorithms synthesize a complex decision surface that can classify high dimensional patterns such as handwritten characters [LeCun and Cortes, 2010]. A neural network computes a function  $y^p = F(x^p, W)$ , where  $x^p$  is the  $p$ -th input pattern,  $y^p$  is the associated  $p$ -th network output, and  $W$  is the collection of adjustable parameters in the system, also called *weights*. In a classification task, the network’s output is the class predicted for the input pattern, or the vector of probabilities associated to each class. Given a *target* response

$t^p$ , the discrepancy between the target and the network's output is measured by a *loss function*  $L^p = \mathcal{D}(t^p, F(W, x^p))$ . The learning problem consists in finding the parameters  $W$  that minimize the loss function over a set of labeled examples held out from the training set, *i.e.*, the test set. In order to minimize the loss function, gradient-based learning relies on estimating the impact of small variations of the parameter values on the loss function, by computing the gradient of the loss function with respect to the parameters. The loss function is associated with an energy landscape curvature. In the case of deep neural networks, the such landscape is non-convex and features global minima, local minima and saddle points. At initialization, the network is initialized at one point of the landscape curvature. The goal of the training is to move the network's state towards the minimum points of the landscape. In the gradient descent algorithm, the network's parameters are iteratively adjusted by following the steepest path downward at each step. This can be described by the following update rule:

$$W_{ij,t} = W_{ij,t-1} - \eta \frac{\partial \mathcal{L}(W)}{\partial W_{ij,t}} \quad (2.2)$$

The gradient of the loss function is the first-order derivative of high-dimensional variables. The optimization involves finding a point where the gradients have a very small magnitude (ideally zero). The gradient gives the direction of the maximum increase of the function. By updating the network's parameters along that direction, the training steps decrease the magnitude of the gradient.

Several loss functions can be employed to train DNNs. In the context of regression tasks, some common loss functions are:

- the mean absolute error, *i.e.*, the difference between actual and predicted values  
 $\mathcal{L}(y, f(x)) = |y - f(x)|$ ;
- the mean square error or quadratic loss, *i.e.*, the mean of the square of the difference between actual and predicted values  
 $\mathcal{L}(y, f(x)) = \frac{1}{2}(y - f(x))^2$ ;
- the root mean squared error, *i.e.*, the square root of the mean square error  $\mathcal{L}(y, f(x)) = \sqrt{\frac{1}{2}(y - f(x))^2}$ ;
- absolute deviation
- $\epsilon$ -insensitive loss, in which the error is taken into account only if smaller than a threshold  $\epsilon$   
 $\max(|y - f(x)| - \epsilon, 0)$ .

In the context of binary classification, some standard losses are:

- 0-1 loss, which is related to the accuracy; it amplifies false negatives and is not differentiable  
 $\mathcal{L}(y, f(x)) = \frac{1}{N} \sum_{i=1}^N \delta_{f(x_i) \neq y_i}$ ;
- the square loss, *i.e.*, the mean of the square of the difference between actual and predicted values  
 $\mathcal{L}(y, f(x)) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2$ ;
- the binary cross-entropy loss, which evaluates a score that summarizes the average difference between the actual and predicted probability distribution for predicting the correct class  
 $\mathcal{L}(y, f(x)) = \frac{1}{N} \sum_{i=1}^N t_i \cdot \log(y_i) + (1 - t_i) \cdot \log(1 - y_i)$ ;
- the Hinge loss, which is used when the target values are in the set  $\{-1, 1\}$ ;  
 $\mathcal{L}(y) = \sum_{i=0}^N \max(0, 1 - f(x) \cdot y)$ ;
- the squared Hinge loss, *i.e.*, the square of the Hinge loss.

The parameter initialization is crucial to obtain good accuracy in training. The goal of initialization techniques is to initialize the networks in a place of the energy landscape which facilitates the optimization process. Some strategies to choose the initial weight matrices are:



- Constant initialization, in which all parameters are initialized to the same values. The drawback is that, if the weights are set all to the same values, in the deep layers all the gradients are the same, preventing learning.
- Random initialization, in which the parameters are initialized with a Normal distribution with a given mean and standard deviation. The choice of the standard deviation is key to ensuring learning. the drawback of this technique is that the output of the layer scales with the number of input neurons at each layer, therefore the variance of the activations becomes very large in the deeper layers.
- The Xavier initialization scheme [Glorot and Bengio, 2010], which was introduced to alleviate the issue of random initialization. It scales the variance of the initialization by the input image dimension in the first layer and by the dimension of the preceding layer in deeper layers. This allows keeping the same variance as that of the input at each hidden layer. This strategy is used with the tanh non-linearity.
- The He initialization strategy [He et al., 2015], which is similar to the Xavier initialization, but it also takes into account the fan-out of the hidden layers. It is used specifically with ReLU non-linearity.

**Delta rule** The delta rule is a gradient descent learning rule for supervised learning proposed by Widrow and Hoff [Widrow and Hoff, 1960]. It is also called the Least Mean Square (LMS) method. The delta rule is applied to adjust the weights of the inputs to artificial nodes in a single-layer neural network. For each input vector, the output vector is compared to the target response. If the difference is zero, the weights are not updated. If the output is different from the target, the weights are updated to minimize the difference. The change in weight from the input node  $u_i$  to the output node  $u_j$  is given by:

$$\Delta W_{ij} = \eta \cdot a_i \cdot e_j, \quad (2.3)$$

where  $\eta$  is the learning rate,  $a_i$  represents the activation of  $u_i$  and  $e_j$  is the difference between the target and the actual output of  $u_j$ . In practice, the delta rule implements gradient descent by moving the weight vector from the point on the surface of the paraboloid down toward the lowest point, the vertex.

The delta rule has been hypothesized to be implemented by pyramidal spiking neurons with dendritic prediction of somatic spiking [Urbanczik and Senn, 2014]. It is a special case of the more general backpropagation algorithm, which is applied to train multi-layer networks.

**BackPropagation and supervised learning** The backpropagation of the error (BP) algorithm is the most popular and commonly used strategy of credit assignment to train neural networks. BP was introduced in the framework of supervised learning, *i.e.*, the setting in which each training sample is accompanied by labels. The scope of BP-based learning is to approximate a unidirectional mapping from a  $n$ -dimensional input space, where  $n$  is the number of input variables, to an  $m$ -dimensional space, where  $m$  is the number of output variables. Such learning is implemented during a training phase by modifying the strength of the weights connecting the network's node based on error feedback on training samples. [Wythoff, 1993]

Figure 2.2 reports the schematics of a fully-connected network with the key terminology needed to understand the technical chapters that will follow.

- The *nodes* are point-like neuron models. organized in a multi-layer hierarchical structure.
- The *input layer* is the bottom layer of the network and encodes an input signal  $x$ , *e.g.*, an image. It does not perform any processing on the signal.
- The *hidden layers*  $h_i$  progressively extract more complex features from the input signal, that travels from the bottom to the top layers of the network. The term *hidden* indicates that such layers do not receive input from not deliver output to the 'outside world'. The subscript  $i$  indicates the layer index.

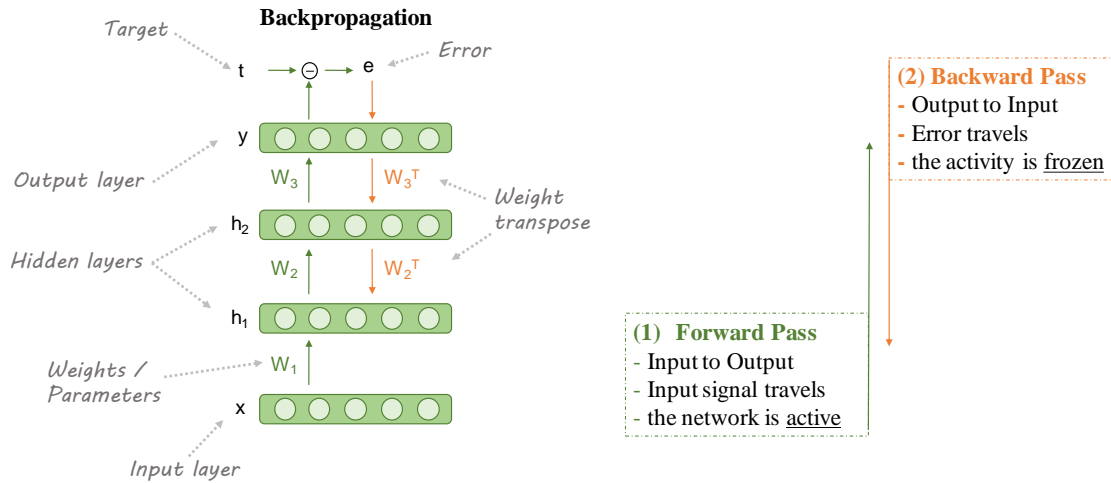


Figure 2.2: Schematics of a fully connected 2-hidden layer network reporting keywords. Circles denote nodes and arrows denote connections. Illustration of forward and backward pass.

- The *output layer* is at the top layer of the network  $y$  and delivers the network's *output* to the 'outside world' or the 'user'.
- The network's learnable *parameters*, *i.e.*, the *weights*  $W_i$ , connect neurons in contiguous layers.
- The *output*  $y$  of the network is compared to a *target* or *label* to compute an *error*.

The BP training scheme entails two alternating phases per input signal – or batch of input signals: the Forward pass, or activation flow, and the Backward pass, or error flow.

- In the Forward pass, the input signal  $x$  travels from the bottom to the top layers through the network's weights  $W_i$ . At each node an activation (floating value) is computed, based on the weighted sum of the node's input:  $a_i = W_i x_{i-1}$ . A non-linear monotonic function  $\sigma$  (*e.g.*, sigmoid, hyperbolic tangent, or REctified Linear Unit for hidden layers, softmax for output layers) is applied to the nodes activations:  $h_i = \sigma(a_i)$ . The role of the non-linearity is to allow a multilayer neural network to perform nonlinear functional mappings.
- In the Backward pass, the error signal computed at the output layer is propagated from the top to the bottom layers, through the transpose of the weights used in the forward pass. As the error travels through the network, the network's activity is frozen. This is the *learning phase* is the algorithm. The error signal at each layer is computed by applying the chain rule of calculus. This is also known as automatic differentiation - reverse mode differentiation. The update of each parameter is proportional to the derivative of the loss function with respect to the parameters. The updates are computed and applied in a backward fashion.

The equations describing the computations performed for the updates are:

- computing the weight change  $\Delta W_{ij,t} = \eta \frac{\partial \mathcal{L}}{\partial W_{ij}}$
- applying the update  $W_{ij,t} = W_{ij,t-1} - \Delta W_{ij,t}$

where  $t$  is the current time step,  $\eta$  is the learning rate (*i.e.*, the update step size) and  $\mathcal{L}$  is the loss function.

**Optimization strategies** The networks parameters can be updated;

- once per epoch, computing the gradient for all training samples. This is the gradient descent (GD) strategy, which is stable for small datasets, but too slow for big datasets. It is computationally inefficient.
- once per sample, computing and applying the gradient at each training sample. This is stochastic gradient descent (SGD). It requires less memory than GD and it is faster, however, it is more affected by noise and consequently is less stable. On the other hand, it helps with generalization.
- once per subset of training samples, the so-called *mini batch*. This is known as mini-batch gradient descent, which combines the advantages of GD and SGD in terms of memory requirement, convergence rate, and stability. In practice, this strategy updates the weights based on the average gradient over the mini-batch.

The gradient descent-based algorithms may converge to local minima when the loss landscape is non-convex. To overcome such a limitation, the optimization can take *momentum* into account. This technique modifies the basic gradient descent updates by retaining information about previous updates and combining it with the gradient computed at the current step. The mathematical description of the momentum optimizer is:

$$\Delta W_{ij,t} = \left( \eta \frac{\partial \mathcal{L}_t}{\partial W_{ij}} \right) + (\lambda \Delta W_{ij,t-1})$$

where  $\lambda$  is the momentum factor that weighs how much of the previous update should be retained. Typically,  $\lambda = 0.9$ . Introducing momentum in the optimization allows increasing the learning speed, avoid converging to local minima and smooth the update variation. This is possible as the introduction of momentum dampens the oscillations in GD/SGD. Indeed the current update mostly uses information from previous updates and only some information from the current update. The choice of the learning rate needs to take into account that, if the step size is

- too large, at each step the loss oscillates between opposite “sides” of the minimum on the loss landscape, possibly leading to divergence;
- too small, the learning requires an intractable number of steps and, additionally, it might get stuck in local minima.

The Learning Rate Schedule is a strategy to mitigate these issues. It makes use of decaying learning rate, through schemes such as step decay, cosine or exponential decay. Generally, this keeps the learning rate big for the first few epochs and reduces it afterwards. In practice, the choice of the learning rate depends inversely on the curvature of the convex function. The steeper the function, the lower the learning rate that can be used. Another scheme for the learning rate is the so-called *one-cycle* learning rate, in which the learning rate initially is small, then it is gradually increased, and finally it is gradually decreased. Interestingly, this strategy has a connection to how children learn, sometimes focusing on small details, and other times exploring a lot.

[New techniques for optimization in this thesis: Chapter 5](#)

**Preventing overfitting** One issue of training neural networks is *overfitting*, the phenomenon by which the model learns details and noise of the training samples, rather than the general features. This impairs the generalization to the testing samples, thereby leading to poor test accuracy. In practice, in the first phase of training, both the error on the training samples and the error on the testing samples decreases. However, in the second phase, the network learns spurious and misleading regularities on the training set, leading the test error to increase, while the training error continues to decrease [Reed, 1993]. Regularization schemes modify learning algorithms to prevent overfitting. Some popular regularization techniques are dropout, weight decay, early stopping and data augmentation can be applied.

Dropout [Hinton et al., 2012] is a training algorithm for neural networks, which is designed to regularize neural networks in order to improve their generalization performance. Dropout training is implemented as a modification to standard SGD training. During learning, at each presentation of a training sample, the input states and hidden unit states of the network are multiplied by a binary mask with values 0 or 1. The zeros in the mask cause some units to be removed from the network. This mask is generated randomly each time an example is presented. Each element of the mask is sampled independently of the others, using some fixed probability  $p$ , termed *dropout keep rate*. At test time, no units are dropped, and the weights are multiplied by the dropout keep rate to compensate for that unit being present more often than it was during training. The effect of dropout is equivalent to training exponentially many neural networks that share weights, then averaging together their outputs. Therefore, dropout acts as a very effective regularizer, thereby enabling the training of very large networks [Srivastava et al., 2013].

Weight decay is another effective approach to reducing overfitting. It relies on penalizing large weights, by adding extra terms to the loss function, for example through L1 or L2 regularization. Avoiding large weights prevents relying too much on the content of a few single pixels or nodes.

Furthermore, early stopping can be used to determine when overfitting is starting. In early stopping, the training samples are split into a training and validation set and, when the validation accuracy starts decreasing, indicating that overfitting has begun, training is stopped.

Finally, the approach of data augmentation is used to generate more data points from the existing data set. It entails the transformation of data through different methods including rotations, translations, reflections, scaling, cropping, adding Gaussian noise, and adding occlusion. The models trained with data augmentation are more robust to the deformations applied to the training images and generalize better to testing images.

## 2.3 Network architectures under a biological lens

The BP algorithm has been applied to train increasingly complex neural network models on progressively more challenging tasks. Here I review some common model architectures in ML and outline some connections with biological circuits in the brain. Among the wide zoology of models, I will focus on the architectures that are employed in the technical chapters (Chapters 5 to 9) of the thesis.

### 2.3.1 Similarities between biological circuits and artificial networks

The structure of several neural network architectures exhibits undeniable similarities with the organization of biological circuits. Five essential principles are shared between neuroscience and AI: hierarchical organization, plasticity at the level of connection strength, non-linearity, normalization, and sparsity.

**Hierarchy** The topological organization of the brain is characterized by the interplay of different levels each playing a specific and specialized functional role. In particular, the cerebral cortex of the mammalian brain exhibits connectivity patterns organized into a hierarchy, which encompasses the microscopic cellular level, the mesoscopic level of local neural circuits and columns, and the macroscopic level of nerve fiber projections between brain areas [Zhou et al., 2007]. The highest structural level consists of the systems level of corticocortical connections. The cortical areas process the sensory stimuli in a hierarchical fashion, which means that the response properties change systematically from lower- to higher-level areas. In the visual stream, for instance, receptive fields have been found experimentally to become larger and the topographic (retinotopic) organization less pronounced. Therefore, neurons at lower levels are activated by simple features, such as edge orientation and simple shapes, while neurons at higher levels respond to global, invariant, and semantic image features, such as faces and complex motion features. In particular, the ventral pathway of the visual cortex is composed of the following areas. The lateral geniculate nucleus (LGN) performs a sort of multi-scale high-pass filtering and contrast normalization, V1

contains pooled oriented edge detectors with local receptive fields, V2 and V4 detect more complex local motifs, and the infero-temporal cortex that encodes object categories [LeCun, 2012]. This multi-stage pathway of information flow is known as the visual cortical hierarchy (VCH) [Hilgetag and Goulas, 2020].

Similarly to biological circuits, ANNs exploit depth. Deep neural networks are formed by staking several layers or blocks of layers. An essential empirical feature of BP, is its ability to learn useful internal representations of inputs [Lillicrap et al., 2020]. Such representations take the form of low-level features in the early layers, such as edges orientation, colors and fragments of shapes, and are combined into semantic features at higher levels. Such intermediate representations allow the models to grasp several concepts as a combination of shared features [Illing et al., 2021, LeCun, 2012]. An essential feature of high-level representation is that they are invariant (or robust) to irrelevant variations of the input, such as translation, scaling, mild rotation, and illumination change in the case of images [LeCun, 2012].

In the next sections, I will outline the main types of layers and their connections with neuroscience.

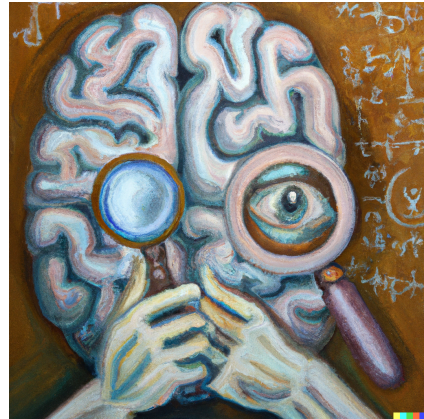
**Plasticity at the level of connection strength** Learning in the brain occurs through experience-driven adjustment of synaptic strength of connections among neurons. Neural plasticity refers to the ability of the nervous system to change its activity and structure in response to intrinsic or extrinsic stimuli by reorganizing its connections. Synaptic plasticity constitutes the premise of the development of the brain and to its ability to learn and store memory [Mateos-Aparicio and Rodríguez-Moreno, 2019].

Analogously, training neural networks consists of updating initially random parameters, *i.e.*, weights and biases, so that the models learn to perform a specific task. During training, as the weights are modified, the error on the training patterns is decreased. The process of designing the dynamics governing the weight updates is termed the *Credit Assignment* problem.

**Nonlinearity** As mentioned above, each level of a hierarchical model includes a fixed non-linear transformation layer, which can be a multinomial logistic or a winner-take-all or a point-wise non-linear mapping (e.g. logistic, tanh, shrinking function, or half-rectifier). The choice of introducing a non-linearity in the first computational models of brain mechanisms was originally inspired by the transfer functions of biological neurons. For instance, the perceptron model [Rosenblatt, 1958] relied on a step function as non-linearity. The threshold of a step function is a critical input boundary above which outputs are fully activated, which is reminiscent of ‘all-or-none’ properties of neurons [Hodgkin and Huxley, 1952]. However, to allow nodes to have analog values, sigmoidal tuning curves [Dayan and Abbott, 2005] and functions with a linear component in the positive polarity [Glorot et al., 2011] have substituted the step functions. Therefore, the most commonly used nonlinear functions in state-of-the-art ML models do not reflect the input-output relation in biological neurons. The most commonly used point-wise non-linear functions are:

- sigmoid function  $f(x) = \frac{1}{1+e^{-x}}$ , with output in the range  $[0,1]$ . The gradient is positive close to the zero, but decays to zero away from the zero.
- hyperbolic tangent (tanh)  $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ , with output in the range  $[-1,1]$ . As for the sigmoid, the gradient is positive close to the zero, but decays to zero away from the zero.
- rectified linear unit (ReLU)  $f(x) = \max(x, 0)$ , with output in the range  $[0, \text{inf})$ . The gradient is 0 for negative values of the input and 1 for positive values.

**Neural networks under a biological lens.** DALL-E prompt: An oil painting of a brain with eyes and arms that looks into a magnifying glass, on a background of mathematical formulas.



- Leaky ReLU,  $f(x) = \max(x, \alpha x)$ . The gradient is  $\alpha$  for negative values of the input and 1 for positive values.
- softmax, generally for the output layer,  $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$ , with output in the range  $[0,1]$

Recently, some biologically inspired activation functions have been proposed, however, they have not had a significant impact in the community [Bhumbra, 2018].

**Normalization** Homeostatic plasticity includes a set of mechanisms that stabilize and normalize network activity to lie within certain ranges or to have a certain distribution, independent of input statistics or network parameters. This ensures that normal brain function is maintained [Shen et al., 2020]. Among these mechanisms, divisive normalization is a canonical computation in various areas of the brain [Carandini and Heeger, 2011]. It consists of dividing the initial input-driven activity of a neuron by the summed activity of a large pool of neighboring neurons [Louie et al., 2013]. In the brain, normalization occurs across four spatial scales: normalization of a single neuron’s activity, which brings closer the firing rate of two neurons; normalization of synaptic weights of a neuron, which multiplicatively adjusts the strength based on firing rates; normalization of a layer of neurons, which ensures that the mean of the cumulative distribution for neurons in a layer is kept constant; normalization of an entire network of neurons, to maintain stable the global distributions of firing rates and synaptic weights for the network [Shen et al., 2020]. Normalization was originally proposed to explain nonlinear responses in the primary visual cortex. In the lower stage of the ventral pathway of the visual cortex, the lateral geniculate nucleus performs a contrast normalization operation on the input [LeCun, 2012]. Furthermore, divisive normalization has been found to play a role in other areas of sensory systems in mechanisms such as contrast gain control in the retina and thalamus, surround suppression in the middle temporal area, and gain control in the auditory cortex. Additionally, normalization explains neural responses underlying higher-order processes such as multisensory integration and visual attention [Louie et al., 2013].

Analogously, artificial neural networks heavily rely on normalization. ML normalization is functionally equivalent to normalization in the brain: both drive activation patterns of hidden units toward a homeostatic state, where all neurons are equally used [Shen et al., 2020]. Normalization is a standard component of state-of-the-art architectures. A general multi-stage architecture for hierarchical feature learning at each level contains a normalization layer, a linear filtering (*e.g.*, convolution), a non-linear transformation, and a pooling layer. The normalization layer may be a whitening operation, or in the case of spatial signals, a high-pass filtering with local energy normalization at a single scale or multiple scales [LeCun, 2012]. Normalization can be implemented through whitening activations at every training step or at some interval, either by modifying the network directly or by changing the parameters of the optimization algorithm to depend on the network activation values [Wiesler et al., 2014]. However, if these computations are combined with the optimization steps, the gradient descent step may not take into account the normalization [Ioffe and Szegedy, 2015]. To address this issue, some more sophisticated normalization techniques have been proposed, including batch normalization, layer normalization, and weight normalization [Lin, 2021]. These techniques and their variants help to stabilize hidden unit activity and accelerate network training [Shen et al., 2020].

The *batch normalization* (BN) performs a global normalization along the batch dimension such that for each neuron in a layer, the activation over all the mini-batch training cases follows a standard normal distribution. This approach ensures that, for any parameter values, the network always produces activations with the desired distribution. As a result, BN reduces the internal covariate shift, which is the change in the distribution of network activations due to the change in network parameters during training. From a biological standpoint, BN is related to the normalization of the activity of a neuron, which is recognized as an important stabilizing mechanism [Shen et al., 2020]. BN allows us to use much higher learning rates and be less careful about initialization. Furthermore, BN also acts as a regularizer and, in some cases, it can substitute Dropout [Ioffe and Szegedy, 2015]. Finally, BA significantly reduces the training time in feed-forward neural networks.

However, BA strongly depends on the mini-batch size and it is not obvious how to apply it to recurrent neural networks. [Ba et al., 2016]. Building on BN, the *layer normalization* (LN) com-

putes the mean and variance used for normalization from all of the summed inputs to the neurons in a layer on a single training case. Unlike BN, LN performs the same computation at training and test times. LN can be easily applied to recurrent neural networks by computing the normalization statistics separately at each time step. In recurrent networks, LN has shown an effective stabilizing effect in the hidden state dynamics [Ba et al., 2016]. In the brain, layer-wise normalization occurs in sensory systems such as the fruit fly olfactory system [Shen et al., 2020].

Finally, the *weight normalization* (WN) applies the normalization over the incoming weights. It relies on a reparameterization of the weight vectors that decouples the length of the weight vectors from their direction. WN does not introduce dependencies between the examples in a minibatch, therefore it can also be applied to recurrent models [Salimans and Kingma, 2016]. Biologically, the brain implements a mechanism that is related to weight normalization, known as *synaptic scaling*. If the firing rate of a neuron is on average greater than its target firing rate, then all of its incoming excitatory synapses are weakened to reduce its future activity, and vice versa [Turrigiano et al., 1998]. An additional mechanism of weight normalization in biology is the *dendritic normalization*, which occurs locally on individual branches of a neuron’s dendritic arbor. If one synapse is strengthened, then its neighboring synapses on the arbor are weakened [Royer and Paré, 2003a].

**Sparsity** The neocortex represents information using sparse distributed patterns of activity [Barth and Poulet, 2012]. The activity of the neurons is sparse since, at any point in time, only a small fraction of neurons are firing while the rest remain silent. Representations are distributed, which means that, despite each active neuron contributing information, it is the set of active neurons that determines what is being represented. Sparse representations have been experimentally observed in early auditory, visual and somatosensory areas [Ahmad and Hawkins, 2016]. Sparsity ensures that only a small percentage of the neurons consume energy at every instant, complying with the constrained energy resources of the brain. Furthermore, it reduces the memory requirements of the learning and information processing circuits. Experimentally, interplay between sparsity and scaling the size of biological brains has been observed, revealing that brains that have more neurons exhibit more sparse activity [Herculano-Houzel, 2012].

In contrast, most standard machine learning models are dense and over-parametrized, which implies heavy memory and computation effort during model training and inference. For energetic reasons, sparsity is a desirable property for artificial neural networks as well. Sparse activations can decrease the memory footprint of regular networks to fit mobile and battery-powered devices. Furthermore, sparsity alleviates the growing energy and performance costs of complex models. Sparsity also enables shorter training time, reducing the computational time resources. Finally, sparsification leads to improved generalization and robustness and to improved performance for inference and/or training [Hoeffler et al., 2021]. Therefore, an important branch of ML research focuses on methods to sparsify activity in neural networks. “Today’s sparsification methods can lead to a 10-100x reduction in model size, and to corresponding theoretical gains in computational, storage, and energy efficiency, all without significant loss of accuracy” [Hoeffler et al., 2021]. The most powerful technique to reduce memory requirements is known as *pruning* [Hoeffler et al., 2021, Reed, 1993]. Overall, pruning reduces the representational complexity by zeroing out subsets of the model parameters, hence using only a subset of the dimensions at a time. The underlying principle is to train networks larger than necessary and remove the parts not needed. The elements that can be sparsified are the neurons and the weights, as well as the filters in convolutional layers and the heads in attention layers. Furthermore, in gradient-based training, also errors and gradients can be sparsified to only update weights partially. Sparsification can be applied after training - often followed by fine-tuning- or during training - following a sparsification schedule, *i.e.*, how fast to prune how many elements. Several approaches to select the elements to be removed have been proposed, including evaluating the network with and without the elements in question (not scalable), removing elements randomly, removing weights with the smallest absolute magnitude, and merging neurons with similar output activations and rewiring the network accordingly [Hoeffler et al., 2021]. The review by Hoeffler and colleagues [Hoeffler et al., 2021] provides a more in-depth overview of the key techniques and ideas of network sparsification.

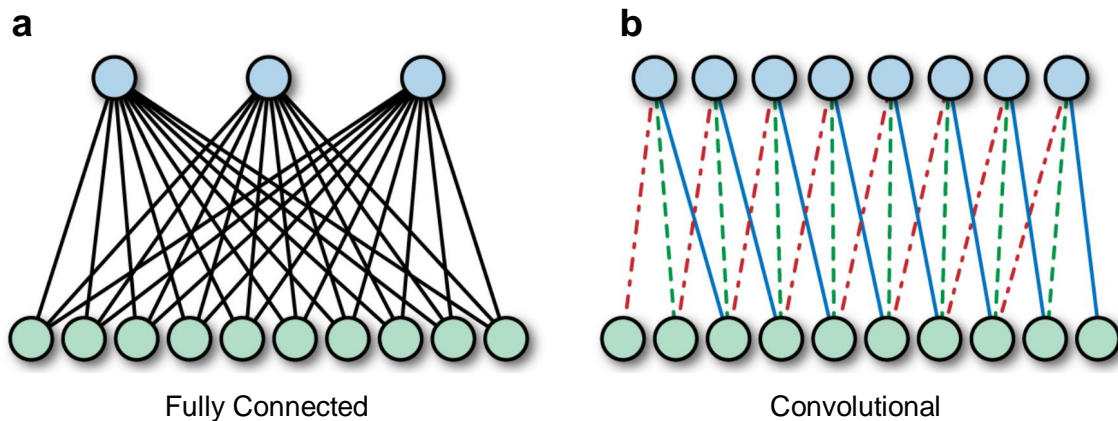


Figure 2.3: (a) Schematics of a fully connected layer. (b) Schematics of a convolutional layer with weight sharing. Adapted from <https://www.oreilly.com/library/view/learning-tensorflow/9781491978504/ch04.html>.

### 2.3.2 Fully connected neural networks

A fully connected neural network consists of a sequence of fully connected or dense layers that connect every node in a layer to every node in the following layer. A fully connected layer is a function from  $\mathbb{R}^n$  to  $\mathbb{R}^m$  where  $n$  is the input dimension and  $m$  is the output dimension. Figure 2.3(a) shows the diagram of a fully connected layer. If  $y_k$  is the  $k$ -th element of the output of the fully connected layer, it can be written as  $y_k = \sigma(\sum_{j=1}^n W_{kj}x_j)$ , where  $\sigma$  is a nonlinearity and is applied element-wise.

Fully connected layers are structure ‘agnostic’, which means that there are no specific assumptions or requirements for the input structure. This allows using fully connected models for a broad spectrum of applications. However, this flexibility comes at the cost of a lower performance compared to more input-specific layers. The following sections will present some examples of ML architectures tailored to specific tasks and applications, such as convolutional networks for computer vision, LSTM cells for time series, and Transformers for speech processing and computer vision.

### 2.3.3 Convolutional neural networks

Convolutional Neural Networks (CNNs) are architectures that have been specifically designed to process images. This specificity allows encoding of given features in the model, in particular by extracting two-dimensional features. CNN models consist of a series of layers of three types: convolutional layers, pooling layers and fully connected layers. The development of CNNs has been strongly inspired by the mechanisms governing the visual cortex. In 1959, Hubel and Wiesel carried out experiments on the cat’s striate cortex. They discovered that simple cells exhibit an enhanced response both to specific edge orientation and location in the receptive field, meaning that they have a higher firing rate when a bar of a particular orientation is shown in a particular location (Figure 2.4(a)). Complex cells receive input from many simple cells and thus have more spatially invariant responses. Hence, complex cells can be considered as a combination of several simple cells (Figure 2.4(b)). These operations are modeled in a convolutional neural network, where the role of simple cells is modeled by convolutional layers and that of complex cells by pooling layers (Figure 2.4(c)). The first computational model of the findings of Hubel and Wiesel was the Neocognitron, proposed in 1980 by Fukushima [Fukushima, 1980] which consists of a cascading model of the two types of cells, applied to pattern recognition tasks. This model was later extended to more complex architectures including the LeNet model by LeCun in 1998 [Lecun et al., 1998b], designed to solve the handwritten digit classification task MNIST (Figure 2.7(a)).



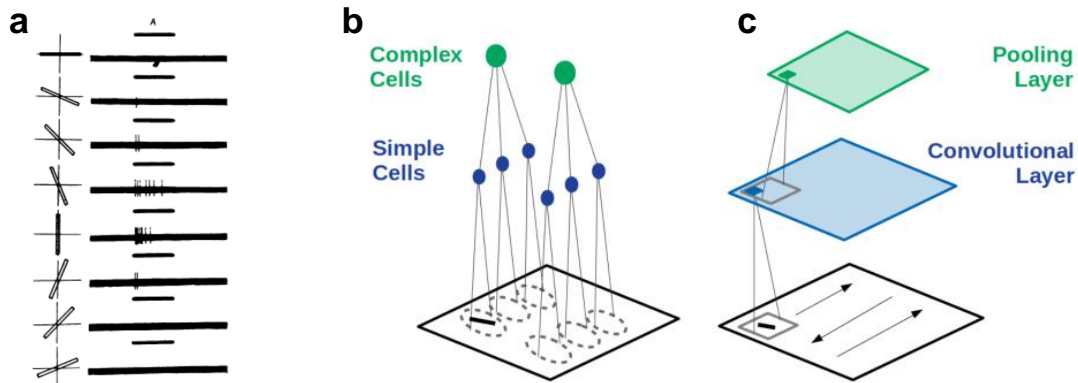


Figure 2.4: (a) Hubel and Wiesel's experiments on the cat's striate cortex. Response to shining a rectangular light spot with different spatial rotations. Reproduced from [Hubel and Wiesel, 1959]. (b,c) Relationship between (b) simple and complex cells (components of the visual system) and (c) convolutional and pooling layers (base operations of a convolutional neural network). Reproduced from [Lindsay, 2021].

**Convolutional layer** The basic operation of a convolutional layer is obtained by applying a convolution operation to the input image. The convolutional layer is a collection of convolutional filters, the *kernels* or *filters*. The input to each convolutional layer has three dimensions: height, width and depth, where the depth is the number of channels. For example, in an RGB image, the depth is equal to three. The input is convolved with these filters to generate the output feature map. The convolution involves the multiplication between the weights and the input. As each input channel to the convolution layer is two-dimensional, the multiplication between weights and input is usually done between a 2D array of weights – the *filter* or *kernel* – and an array of inputs. The multiplication between the filter and the input is a dot product, which involves element-wise multiplication and the results are summed to give the final output in the form of a scalar quantity. Therefore, the application of a small filter to every location in the image creates a *feature map*, *i.e.*, a 2D array containing the filtered values of the input. A convolutional layer has as many feature maps as filters applied. Each filtered value present in the feature map is then passed through a non-linearity. The distance between two consecutive receptive fields is known as a *stride*.

Several features are shared between the visual stream in the animal brain and the convolutional layers. First, both are characterized by translation invariance, which means that the same object elicits similar responses independently of its location in the visual field in the top layers encoding complex and semantic features. Second, convolutional networks resemble the hierarchical structure of the visual cortex: the filters at the lower layers extract information about edge orientation and simple shapes, while the layers above combine low-level information into more complex features.

[Applications of FC and Conv layers in this thesis: Chapter 5 and 6](#)

**Pooling layer** The pooling layers are introduced to reduce the risk of overfitting, by reducing the size of the convolved feature map. Pooling layers downsample the feature map obtained through the convolutional layer. Downsampling refers to the process of producing a lower-resolution image of input that still contains all the important features but does not address the finer details of the original input. They summarize the features present in the feature map by combining them in patches. Taking the maximal (resp. the average) activation in a small section of each feature map downsamples the image and leads to complex cell-like responses. This operation is denoted 'max-pooling' (resp. 'average-pooling') [Lindsay, 2021]. Furthermore, pooling helps in reducing the computational load, saves memory, and overall reduces the number of parameters.

**Alexnet** Since the introduction of LeNet, an increasingly large number of architectures have been proposed. Different architectures are defined by different combinations of layer types, depth,

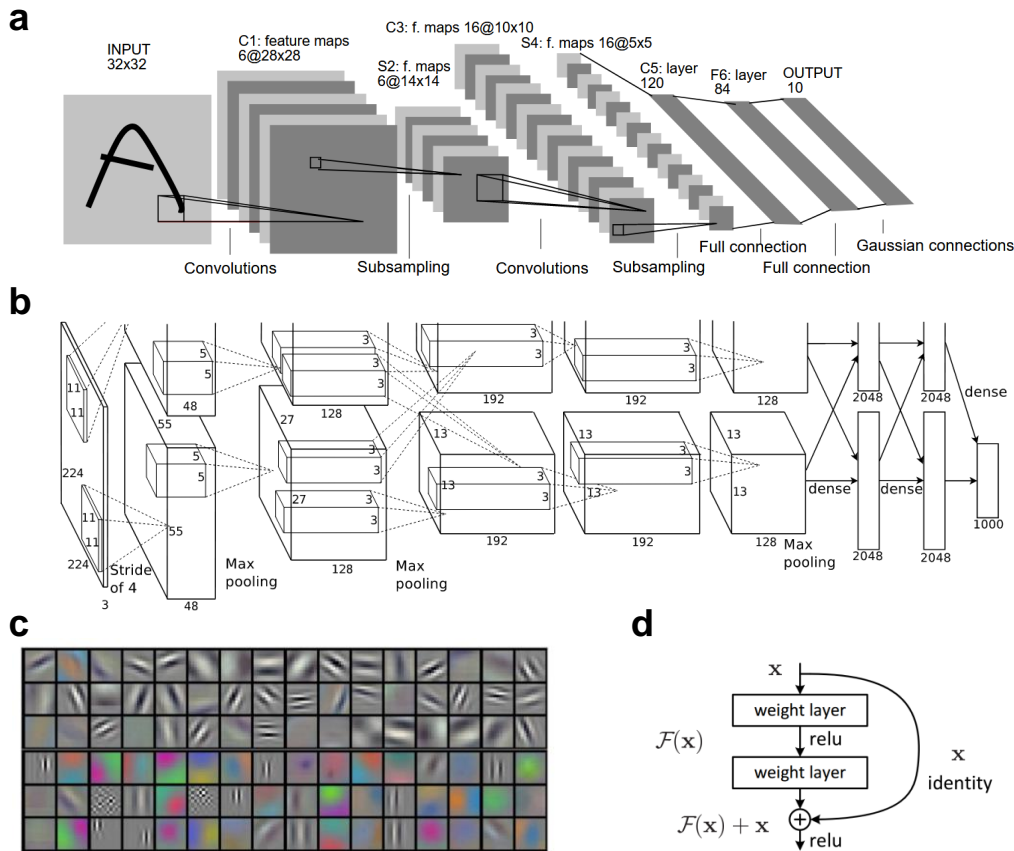


Figure 2.5: (a) LeNet-5 architecture. Reproduced from [Lecun et al., 1998b]. (b) Alexnet architecture. Reproduced from [Krizhevsky et al., 2012]. (c) 96 convolutional kernels of size  $11 \times 11 \times 3$  learned by the first convolutional layer on the  $224 \times 224 \times 3$  input images. Taken from [Krizhevsky et al., 2012]. (d) A building block of a residual network. Taken from [He et al., 2016a].

kernel sizes, and regularization techniques. In this context, one of the most influential papers in computer vision introduced AlexNet, a network composed of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. In AlexNet, the dense layers have most of the parameters of the model, many more compared to convolutions, and they perform the vast majority of the computation. Compared to the LeNet model, the AlexNet architecture increases the number of feature extraction layers from five to seven. Furthermore, the ReLU function is used as nonlinearity instead of the sigmoid. Finally, larger kernel sizes ( $5 \times 5$ ) are used in the earlier layers. The AlexNet architecture is shown in Figure 2.5(b), which illustrates how the computational image processing tasks are shared between two GPUs working in parallel for faster training on hardware. Figure 2.5(c) shows the convolutional kernels learned by the network's two data-connected layers. When trained on the ImageNet dataset [Deng et al., 2009a] (Figure 2.7(d)), the network achieved state-of-the-art performance, earning the title of ILSVRC Winner 2012.

The features learned by the network are general and can be applied to across datasets and tasks. For instance, by removing the last softmax layer, the features extracted by the last dense layer can be applied to tasks other than classification, for example, to decode information on which the network is not trained.

**Biological features and criticisms of CNNs** During training, the AlexNet model has been shown to learn various frequency- and orientation-selective kernels, and colored blobs. Such repre-

sentations are reminiscent of the features of orientation selectivity that emerge in the retinotopic map of the animal’s visual cortex. An additional similarity with biological vision is the hierarchical processing of the input signal: as the information travels from the bottom to the top layers of the network, the extracted features at each layer become increasingly more complex. However, CNNs present the main limitation in terms of biological plausibility, that is *weight sharing*. Weight sharing consists of passing the same filter over the entire input image. This strategy substantially reduces the number of trainable parameters and consequently the training time and cost. Furthermore, weight sharing allows feature search to be insensitive to feature location, hence increasing generalization and preventing overfitting. However, weight sharing cannot be implemented by biological neurons, as synaptic plasticity is local and neurons cannot share receptive fields. Furthermore, the filter values can be both positive and negative, thus the same neuron can be weighted by both inhibitory (negative) and excitatory (positive) weights. In the visual system, connections between areas tend to come only from excitatory cells [Lindsay, 2021]. Outside the domain of biological plausibility, deep CNNs suffer from another major criticism, the *vanishing gradient* problem, that is the very small values of the gradient preventing the weights from changing during the backward pass. This limitation is particularly disadvantageous because increasing the network depth has been demonstrated to greatly improve accuracy, especially for challenging tasks. An effective solution to the vanishing gradient is the introduction of residual neural networks.

**Residual networks** A Residual Neural Network (ResNet) is a model architecture that contains *skip connections* or *residual connections*. Skip connections entail adding the output of a previous layer to the output of a deeper layer, in a bypass pathway fashion. This strategy allows gradient information to pass through the layers, by creating ‘highways’ of information. As the information from the earlier parts of the network is passed directly to the deeper parts of the network, this mitigates the ‘vanishing’ magnitude of signal propagation, allowing to effectively optimize training in deeper networks. Additionally, by reducing the impact of the vanishing gradient, the learning speed increases.

The first proposed neural network containing cross connectivity was the *Highway network* [Srivastava et al., 2015] in 2015. In this model, the skip connections are modulated by learned gating mechanisms that, by merging the information of different layers, introduce regularization of the information flow. The shortcut gates can be opened and closed, in a data-dependent fashion. This strategy allows us to train networks 100 layers deep and to achieve a much higher convergence rate than the standard feed-forward networks. Concurrent with the Highway network paper, He and colleagues proposed the ResNet models, residual learning frameworks that empower training of very deep models by introducing residual blocks [He et al., 2016a]. Compared to the Highway network, the ResNet models present identity shortcuts, thus parameter-free and always operating (*i.e.*, data-independent). Figure 2.5(d) shows the diagram of a skip connection with double-layer skip, that uses ReLU as nonlinearity. The residual block consists of a conventional feedforward network plus a residual connection. The conventional feedforward network may contain nonlinearities and regularization operations. In the same research line of the Highway Network and ResNet, in 2017 Huang presented the Dense Convolutional Network (DenseNet) model [Huang et al., 2017a]. DenseNet relies on each layer to every other layer in a feedforward fashion. This technique allows alleviating the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and reduce the number of parameters.

**What can AI learn from neuroscience?** DALL-E prompt: An oil painting from Botero of a robot learning from a book on the biology of the brain.



Applications of AlexNet, ResNet and DenseNet in this thesis: Chapter 8

### 2.3.4 Recurrent neural networks

Recurrent Neural Networks (RNNs), compared to the described fully connected and convolutional models, are tailored to process sequential data as input. This feature makes RNNs particularly suited for applications in the field of natural language processing and time series analysis. Indeed, the context of an element, *e.g.*, a word in the case of speech processing, is key in understanding its role in the sequence, *e.g.*, its meaning in a sentence. An RNN is a neural network that simulates a discrete-time dynamical system that has an input  $x_t$ , an output  $y_t$ , and a hidden state  $h_t$ , where  $t$  represents time. The dynamical system is defined by:

$$h_t = f_h(x_t, h_{t-1}) \quad (2.4)$$

$$y_t = f_o(h_t) \quad (2.5)$$

where  $f_h$  and  $f_o$  are a state transition function and an output function, respectively.

A conventional RNN is constructed by defining the transition function and the output function as:

$$h_t = f_h(x_t, h_{t-1}) = \phi_h(W^T h_{t-1} + U^T x_t) \quad (2.6)$$

$$y_t = f_o(h_t) = \phi_o(V^T h_t) \quad (2.7)$$

where  $W$ ,  $U$  and  $V$  are the transition, input and output matrices respectively, and  $\phi_h$  and  $\phi_o$  are nonlinear functions, typically sigmoid or tanh.

When unfolded in time, as shown in Figure 2.6, an RNN can be seen as a deep network, as the input travels through several nonlinearities [Pascanu et al., 2013]. The schematics show that RNNs can be thought of as several copies of the same network, where each copy passes the information to the following one. The recurrent nature of RNNs, which implies the existence of loops within the architecture, allows information to be retained over multiple time steps. The possibility of connecting past information to the current tasks enables modeling and solving time-dependent and sequential tasks, such as language translation and speech processing. However, while theoretically RNNs could retain information about past inputs and discover temporal correlations between events separated by a long time delay, in practice RNNs are not able to learn long-term dependencies. Indeed, RNNs are trained with backpropagation-through-time (BPTT), which for long sequences, *i.e.*, long delay between inputs and related teacher signals, suffers from decaying error flowing back “in time”. In practice, long-term components go exponentially fast to norm 0, preventing the model from learning correlations between temporally distant events. This phenomenon, which leads to prohibitive training times or no learning at all of long data sequences, is known as *vanishing gradient* [Pascanu et al., 2012]. A solution to the vanishing gradient issue was proposed with the introduction of *Long-Short Term Memory Networks* [Hochreiter and Schmidhuber, 1997] and of *Gated Recurrent Units* [Cho et al., 2014].

**Long-Short Term Memory Networks** Long-Short Term Memory Networks (LSTMs) are neural network architectures designed to address the vanishing gradient problem of RNNs, allowing the networks to capture long-term dependencies of temporal sequences. LSTMs were proposed in [Hochreiter and Schmidhuber, 1997] and later refined with several variants including [Gers et al., 2000, Gers et al., 2003, Graves et al., 2005]. Training LSTMs relies on a gradient-based algorithm enforcing constant error flow through internal states of the network’s self-connected units. Such an enforcing is obtained by introducing in each unit structures called *gates*, whose role is that of removing or adding information to the cell state. The gates consist of a sigmoid layer and a pointwise multiplication operation. The LSTM units are called *memory cells* and contain

- a *forget gate*, which establishes what information can be removed from the current cell state,
- a multiplicative *input gate unit*, which decides which information needs to be stored in the cell state and which protects the memory information of the cell from perturbations by irrelevant inputs, and
- a multiplicative *output gate unit*, which protects other units from perturbation by current irrelevant information stored in the cell.

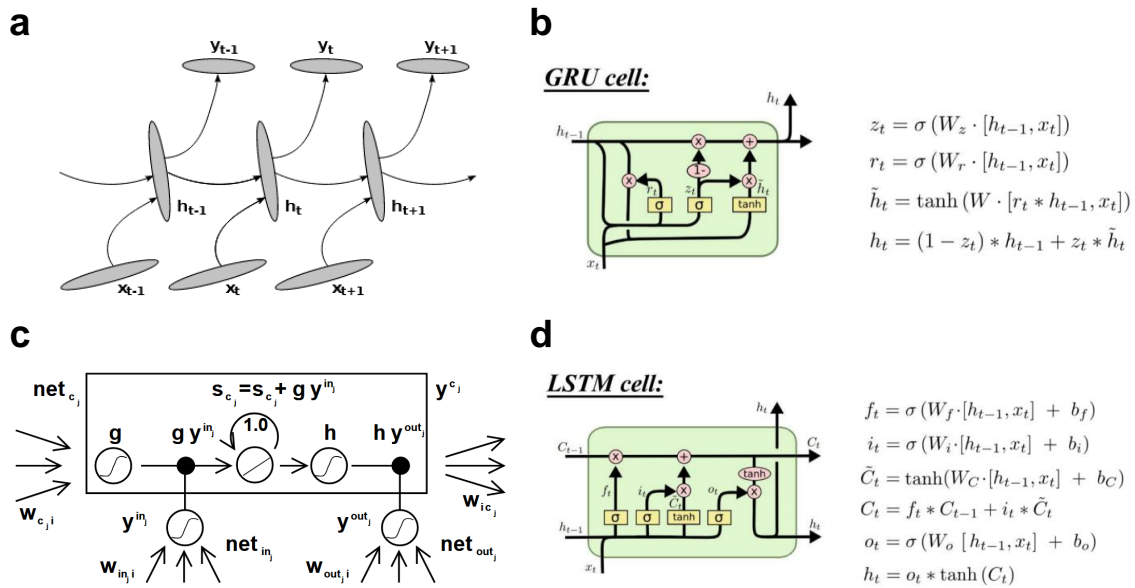


Figure 2.6: (a) A conventional neural network unfolded in time. Reproduced from [Pascanu et al., 2013]. (b) Architecture and equations of a GRU unit. Reproduced from [Daoulas et al., 2021]. (c,d) Architecture and equations of an LSTM memory cell and its gate units. (c) Reproduced from the original paper proposing LSTMs [Hochreiter and Schmidhuber, 1997]. (d) Reproduced from [Daoulas et al., 2021]. In panels (b) and (d), each line carries a vector from the output of one node to the inputs of others. The pink circles indicate pointwise operations (e.g., vector addition), while the yellow boxes are learned neural network layers. Lines merging represent concatenation, while a line forking denotes its content being copied and the copies going to different locations.

The output of the LSTM cell contains information about the three gates. The set of equations governing the dynamics of LSTMs is reported in Figure 2.6(d). These features enable LSTM models to learn long data sequences. Indeed, the long-term dependencies and relations are encoded in the cell state vectors and the cell state derivative can prevent the LSTM gradients from vanishing.

**Gated Recurrent Units** Gated Recurrent Units (GRU) [Cho et al., 2014] are a type of hidden unit for recurrent networks, which is motivated by the LSTM network, but is simpler to compute and implement than LSTM. Each GRU unit contains

- an *update gate*  $z$ , which selects whether the hidden state  $h$  is to be updated with a new hidden state  $\tilde{h}$ , *i.e.*, how much of the past information is passed for the future steps, and
- a *reset gate*  $r$ , which decides whether the previous hidden state is ignored, *i.e.*, how much of the previous information should be forgotten.

The final activation of the unit, *i.e.*, the vector which holds information for the current unit and passes it down to the network, is computed from both the update gate and the reset gate information. Mathematically, the gates  $z_t$  and  $r_t$  and the final activation of a GRU unit  $h_t$  at time  $t$  are described by the equations in Figure 2.6(b). In such equations,  $\sigma$  is the logistic sigmoid function and  $W_i$  are the learnable weight matrices. When the reset gate is close to 0, the hidden state ignores the previous hidden state and resets with the current input only. This allows the hidden state to drop any information that is irrelevant later in the next steps, hence, leading to a more compact representation than LSTMs. Thanks to the coexistence of separate reset and update gates, each hidden unit will learn to capture dependencies over different time scales. Those units that learn to capture short-term dependencies feature reset gates that are frequently active, while those that capture longer-term dependencies feature update gates that are mostly active.

### 2.3.5 Transformers

Transformers are deep feed-forward ANNs with a (self)attention mechanism. Compared to the models described so far, they do not have any recurrence or convolution. They have been proposed originally for machine translation tasks [Vaswani et al., 2017] and have later been extended to many other domains including visual processing [Dosovitskiy et al., 2021, Tolstikhin et al., 2021, Steiner et al., 2021, Chen et al., 2021a, Zhuang et al., 2022, Zhai et al., 2022], speech processing [Chen et al., 2021b, Dong et al., 2018a] and other fields [Lin et al., 2021] such as chemistry [Schwaller et al., 2019] and life science [Rives et al., 2021]. The workhorse of transformers is modeling sequential data, in the frameworks of both sequence-to-sequence and sequence modeling. The transformer architecture for NLP consists of the following building blocks [Phuong and Hutter, 2022]:

- token embedding, to represent a vocabulary element into a vector;
- position embedding, to represent a token's position in a sequence and make sense of word order; the values are multiplied by multiple sinusoidal functions so that the network can learn the position of its different inputs;
- attention, to take into account contextual information to predict the current token; the attention mechanism relies on a Query-Key-Value (QKV) model;
- multi-head attention, the operation by which transformers run multiple attention heads (with separate learnable parameters) in parallel and combine their outputs;
- layer normalization, which controls the mean and variance of individual neural network activations;
- unembedding, which converts a vector representation of a token and its context into a distribution over the vocabulary elements.

The attention block works as follows. The token  $t$  currently being predicted is mapped to a query vector  $q$ , and the tokens in the context are mapped to key vectors  $k_t$  and value vectors  $v_t$ . The inner products  $q^T k_t$  are interpreted as the degree to which token  $t$  is important for predicting the current token  $q$ . The products are used to derive a distribution over the context tokens, which is then used to combine the value vectors. In the concrete example of NLP and language translation, each word to be translated generally depends only on some of the words of the sentence, which are found through the attention operation. In this context, the attention results are human interpretable: it computes what you should attend at the current time step based on the previous time steps.

Compared to recurrent and convolutional layers, the self-attention layers bring three main benefits: lower computational complexity per layer, a larger amount of operations that can be parallelized, and the ability to learn long-range dependencies [Vaswani et al., 2017].

Based on different combinations of building blocks, the transformer model can be used in three different ways. In the *Encoder-Decoder* the entire architecture is employed and this is typically used in sequence-to-sequence modeling (e.g., neural machine translation). In the *Encoder only* way, only the encoder is used and the outputs of the encoder are utilized as a representation for the input sequence, which is usually applied to classification or sequence labeling problems. In the *Decoder only* mode, only the decoder is used and the encoder-decoder cross-attention module is removed. This is generally applied to sequence generation, such as language modeling. Several architectures have been proposed with these features, including Sequence-to-sequence (seq2seq) prediction (EDTransformer) [Vaswani et al., 2017], BERT (encoder-only transformer) [Devlin et al., 2018], and GPT (an instance of a decoder-only transformer) [Brown et al., 2020b]. Furthermore, several variants of the standard transformer architectures have been proposed, the so-called X-former variants, which lead to improvements in terms of model efficiency, generalization, and adaptation [Lin et al., 2021].

Transformers have achieved state-of-the-art performance on a wide variety of complex tasks, however, they suffer from some additional drawbacks compared to convolutional and recurrent networks. Transformers are challenging to scale up, they are more difficult to train as they require manually designed optimization strategies dependent on architectures and tasks, their performance can be incredibly brittle, and they suffer from biases and adversarial attacks. Furthermore, transformers have not been applied to reinforcement learning, because they need a lot of manual supervision. More specifically, self-attention suffers from two main challenges in practical applications. First, in terms of complexity, the attention module becomes a bottleneck when dealing with long sequences. Second, self-attention does not assume any structural bias over inputs, and the order information is learnt from training data. As a consequence, transformers are prone to overfit on small or moderate-size data [Lin et al., 2021].

[Transformer models to predict neural activity in this thesis: Chapter 8.](#)

**Self celebrating AI.** DALL·E prompt: A 3D rendering of a group of colorful robots partying with fireworks and partying caps and whistles.



## 2.4 Representative examples of state-of-the-art achievements

Artificial neural network applications span a wide range of domains, including automatized health-care diagnosis, social network analysis, audio and speech processing (e.g., recognition and enhancement), visual data processing methods (such as multimedia data analysis and computer vision), and natural language processing (NLP) (translation and sentence classification), playing games and controlling robotic devices (e.g., drones). These applications have been classified into five cat-

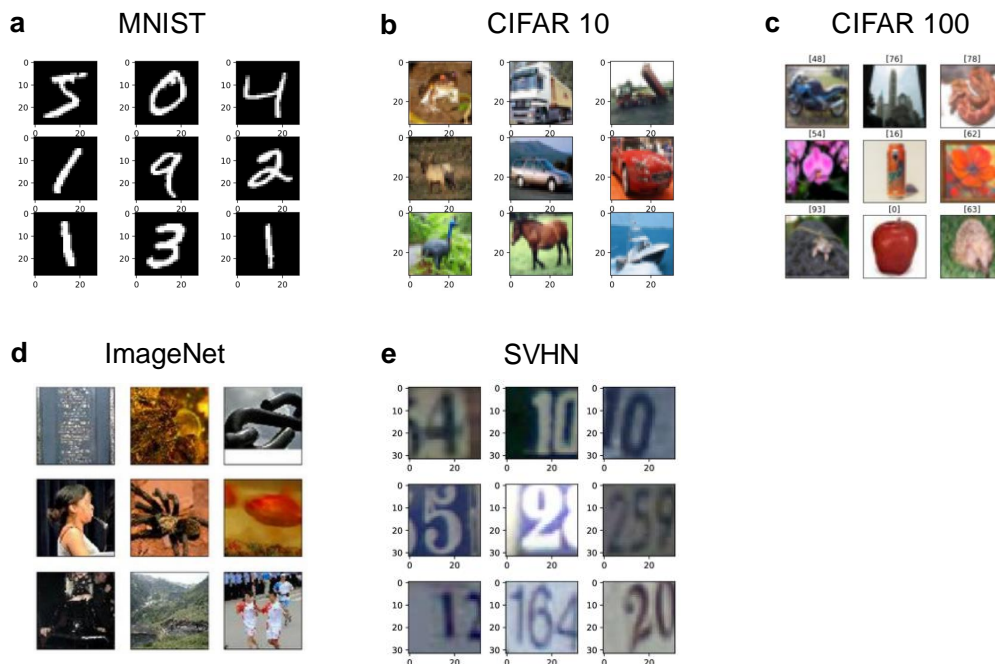


Figure 2.7: Illustration of five common datasets for image classification: (a) MNIST [LeCun and Cortes, 2010], (b) CIFAR-10 [Krizhevsky et al., a], (c) CIFAR-100 [Krizhevsky et al., b], (d) ImageNet [Deng et al., 2009a], and (e) Street View House Numbers (SVHN) [Netzer et al., 2011]

egories: classification, localization, detection, segmentation, and registration [Alz, 2020]. In this section, I will review some common datasets for image classification which will be mentioned later in the thesis, as well as some examples in which AI has achieved impressive results on some specific and creative tasks. This list is not meant to be exhaustive, but rather spark curiosity in the reader on the potential of AI. Furthermore, these examples aim to underline the contrast between the impressive performance of neural networks in challenging tasks and the dramatic failure in contexts that may seem simpler, such as dealing with noise in the case of adversarial attacks, as it will be described in following sections.

### 2.4.1 Computer vision benchmarks

Five common datasets to test models on image classification tasks are:

- MNIST [LeCun and Cortes, 2010] for handwritten digits classifications. The input image dimension is  $28 \times 28 \times 1$  and the number of classes is 10.
- CIFAR-10 [Krizhevsky et al., a] for naturalistic image classifications with low resolution. The input image dimension is  $32 \times 32 \times 2$  and the number of classes is 10.
- CIFAR-100 [Krizhevsky et al., b] for naturalistic image classifications with low resolution. The input image dimension is  $32 \times 32 \times 3$  and the number of classes is 100.
- ImageNet [Deng et al., 2009a] for naturalistic image classifications with high resolution. The input image dimension is  $469 \times 387 \times 3$  and the number of classes is 1000.
- Street View House Numbers (SVHN) [Netzer et al., 2011] for digits classification in natural images. The input image dimension is  $32 \times 32 \times 1$  (or original images with character-level bounding boxes) and the number of classes is 10.



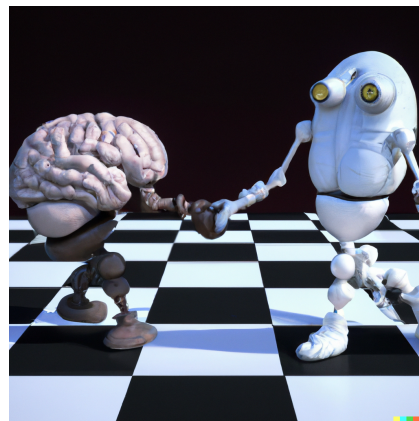
### 2.4.2 Playing games: AlphaGo

The Go game is a 3000 years old Chinese strategic board game in which two players compete to surround more territory than the opponent. Because of its complexity in evaluating board positions and moves, Go has been labeled as the most challenging classical game for AI. Indeed, the evaluation of all possible moves is computationally intractable. Several computer programs have been proposed in the past decades to play the game, including Monte Carlo tree search [Coulom, 2006, Kocsis and Szepesvari, 2006], combined with policies trained to predict human experts' moves [Coulom, 2007]. However, the performance of such algorithms never exceeded that of amateur players. AlphaGo [Silver et al., 2016] was the first model that was able to defeat a Go Europe champion. AlphaGo combines advanced search trees with deep neural networks trained with both supervised learning from human experts and reinforcement learning from games of self-play. The algorithm relies on a *value network* to evaluate board positions and on a *policy network* to select moves.

AlphaGo employs techniques of deep convolutional networks initially designed for computer vision tasks. The input of the network is the board position as a  $19 \times 19$  image, which convolutional layers process to construct a representation of the position. Such a representation effectively reduces the depth and breadth of the search tree: evaluating positions using the value network, and sampling actions using the policy network.

The world of Go was shocked to see AlphaGo implement new and novel strategies. First, AlphaGo learned from human Go players. Now, human players learn from AlphaGo. The student has become the master.

**Challenging human players with AI.** DALL-E prompt: A 3D rendering of a brain with eyes, mouth, arms and legs that walks hand in hand with a robot on a big chessboard.



### 2.4.3 Writing human-like text: GPT3, LaMDA

In recent years, wealthy tech firms including Google, Facebook and OpenAI have developed a new class of computer programs known as “large language models”.

The generative pre-trained transformer (GPT) is a language model from OpenAI [Brown et al., 2020b]. It has been trained on all existent public books and a large dump copy of the internet (as of October 2019).

Google’s response to GPT3 is LaMDA, which stands for Language Model for Dialogue Applications. The model relies on the Transformer architecture. Unlike GPT3, LaMDA was trained on dialogue, which allowed it to pick up on several of the nuances that distinguish open-ended conversation from other forms of language. Thanks to the open-ended nature of the dialogues, LaMDA can take part in conversations touching a wide number of topics with surprisingly nuanced language.

The GPT and LaMDA models support conversational agents, known as *chatbots*, such as BlenderBot [Shuster et al., 2022a]. Such chatbots, while engaging in conversations with users, can rely on long-term memory and search the internet. As a consequence, the produced dialogues are factually consistent and do not have any upper bound on the length. At this point, a question arises: are modern chatbots capable to pass the Turing test?

The Turing test for conversational AI in this thesis: Chapter 9.

### 2.4.4 Driving plasma fusion: Tokamak and reinforcement learning

Fusion power is a promising long-term candidate to supply the energy needs of humanity [Fus, 1993]. In 1920, Eddington proposed an explanation for the mechanism of nuclear fusion occurring in stars [EDDINGTON, 1920]. Nuclear fusion releases energy when two atomic nuclei combine to form a new atomic nucleus. This process requires fuels – deuterium and tritium – that are abundant and nearly inexhaustible on Earth. Furthermore, the process does not release harmful atmospheric emissions. In order to decelerate climate change and global warming observed since the mid-20th century, human activities need to reduce emissions of greenhouse gases and rely on environmentally sustainable energy sources. A promising candidate is the energy derived from nuclear fusion. The possibility of controlling the fusion process on Earth would offer a potential long-term energy source that uses abundant fuel supplies and does not produce greenhouse gases or long-lived radioactive waste.

Generating the process of nuclear fusion on Earth requires controlling a plasma core, which is a gas comprised of hydrogen isotopes, that needs to be brought to heating and pressure conditions able to initiate and sustain fusion reactions. This control is currently realized through magnetic confinement in devices with toroidal symmetry called *Tokamak*. The *tokamak magnetic control problem* is one of the many technological and theoretical problems of these machines. In particular, one of the most impelling questions of plasma physics is to investigate the effects of shaping the plasma distribution into different configurations to optimize the stability, confinement and energy exhaust, and to inform the first burning-plasma experiment, ITER. The target plasma current, position and shape, *i.e.* the plasma configurations, are obtained through a feedback controller that can manipulate the magnetic field. The desired magnetic field is driven through precise control of several coils that are magnetically coupled to the plasma.

The conventional feedback controllers rely on solving an inverse problem to precompute a set of feedforward coil configurations. This approach allows for a successful control, however, it requires demanding engineering effort, design effort and expertise whenever the target plasma configuration is changed, together with complex, real-time calculations for equilibrium estimation. A revolutionary alternative approach was introduced as part of a collaboration between the plasma physics researchers at the EPFL Tokamak and the artificial intelligence researchers at DeepMind [Degrave et al., 2022]. This work relies on Reinforcement Learning (RL) to drive high-dimensional, high-frequency, closed-loop control using magnetic actuator coils. The proposed RL-designed magnetic controller autonomously learns to command the full set of control coils through interactions via a tokamak simulator. Compared to the conventional approach, the RL-based model is less computationally intense, allowing to accelerate the study of alternative plasma configurations, as well as to predict impending plasma disruptions. Such an approach has proven successful in controlling a diverse set of plasma configurations, including elongated, conventional shapes, as well as advanced configurations, such as negative triangularity and ‘snowflake’ configurations. These results encourage a fundamental shift of plasma control design from engineering-driven control of a pre-designed state to artificial-intelligence-driven optimization of objectives specified by an operator. Once again, artificial intelligence-based approaches have proven to be a crucial tool for the solution of one of the most challenging problems for human researchers.

**GPT3: successful writer of the year.** DALL-E prompt: A cubist painting of a robot writing a book.



### 2.4.5 Predicting protein structure: AlphaFold

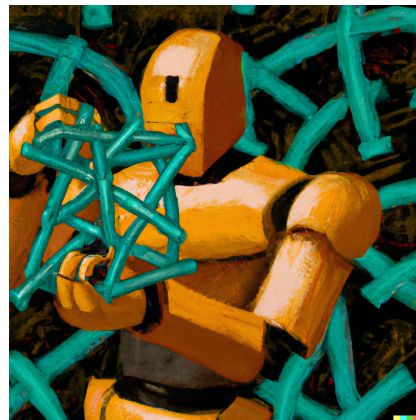
Protein folding is the physical process by which an amino acid sequence assumes its three-dimensional atomic structure, which endows the protein with its functional properties. Understanding which is the biologically active conformation of a protein given its amino acid sequence is known as the *protein folding problem* [Dill et al., 2008]. In 1972, in his Nobel Prize speech, Christian Anfinsen made a historic prediction: it should in principle be possible to determine a protein’s three-dimensional

shape based solely on the one-dimensional string of molecules that comprise it [Toews, 2021]. Finding a solution to this problem has remained an open challenge in the field of biology for fifty years, until in 2021 DeepMind designed the AlphaFold model. AlphaFold [Jumper et al., 2021] is an AI system that predicts a protein’s 3D structure from its amino acid sequence.

Algorithmically, AlphaFold incorporates novel neural network architectures and training procedures based on the evolutionary, physical and geometric constraints of protein structures. In particular, its architecture jointly embeds several components, including multiple sequence alignments and pairwise features, a new output representation and associated loss that enable accurate end-to-end structure prediction, and an equivariant attention architecture. An essential building block of the algorithm is the so-called Evoformer, which views the prediction of protein structures as a graph inference problem in 3D space in which the edges of the graph are defined by residues in proximity. Furthermore, the model contains a structure module which operates on a concrete 3D backbone structure using the pair representation and the original sequence row of the multiple sequence alignment representation from the trunk. The AlphaFold network achieves high accuracy already using only supervised learning on Protein Data Bank data, however, the accuracy can be further improved by incorporating an approach similar to noisy student self-distillation that makes effective use of the unlabeled sequence data. Through this architecture, AlphaFold predicts the 3D coordinates of all heavy atoms for a given protein using the primary amino acid sequence and aligned sequences of homologs as inputs.

AlphaFold is the first computational method able to predict protein structures with atomic accuracy to near experimental accuracy even for proteins for which no similar structure is known. All structures found with AlphaFold are collected in a database which is continuously updated with structures for newly discovered protein sequences. The latest database release contains over 200 million entries. AlphaFold has the immediate potential to help us understand the structure of proteins and advance biological research.

**Finding protein structures with AI.** DALL·E prompt: A painting in vangogh style of a robot that holds the structure of a 3D protein structure in its hands.



#### 2.4.6 Medical diagnosis: predicting pathologies before onset

In many of the previously described examples, artificial intelligence methods have proven to be successful alternatives to the approaches developed by humans to solve a wide variety of problems, such as driving cars, answering questions, playing games or designing controllers for plasma. But the potential of AI goes far beyond improving human capabilities, by increasing performance reliability and speed. AI has proven to succeed in tasks that humans cannot (currently) perform. A striking example is provided by the application of AI to medical diagnosis. The Lab of Regina Barzilay at MIT has demonstrated that neural networks can not only detect pathologies present in the patient from current scans but can also predict the onset of illnesses after 5 years from the current scans, predictions that doctors cannot make. Specifically, in the context of breast cancer research, mammographic breast density is currently assessed by radiologists to diagnose breast cancer. While human diagnosis is prone to errors, deep learning models have proven to be more accurate than radiologists in cancer diagnosis, and they can be used to predict whether a patient will develop cancer after 5 years from the date of the examination. This shows that mammograms contain informative cues that humans cannot detect, while deep learning models can discover such cues from the data and leverage them to assess the risk of breast cancer [Yala et al., 2019].

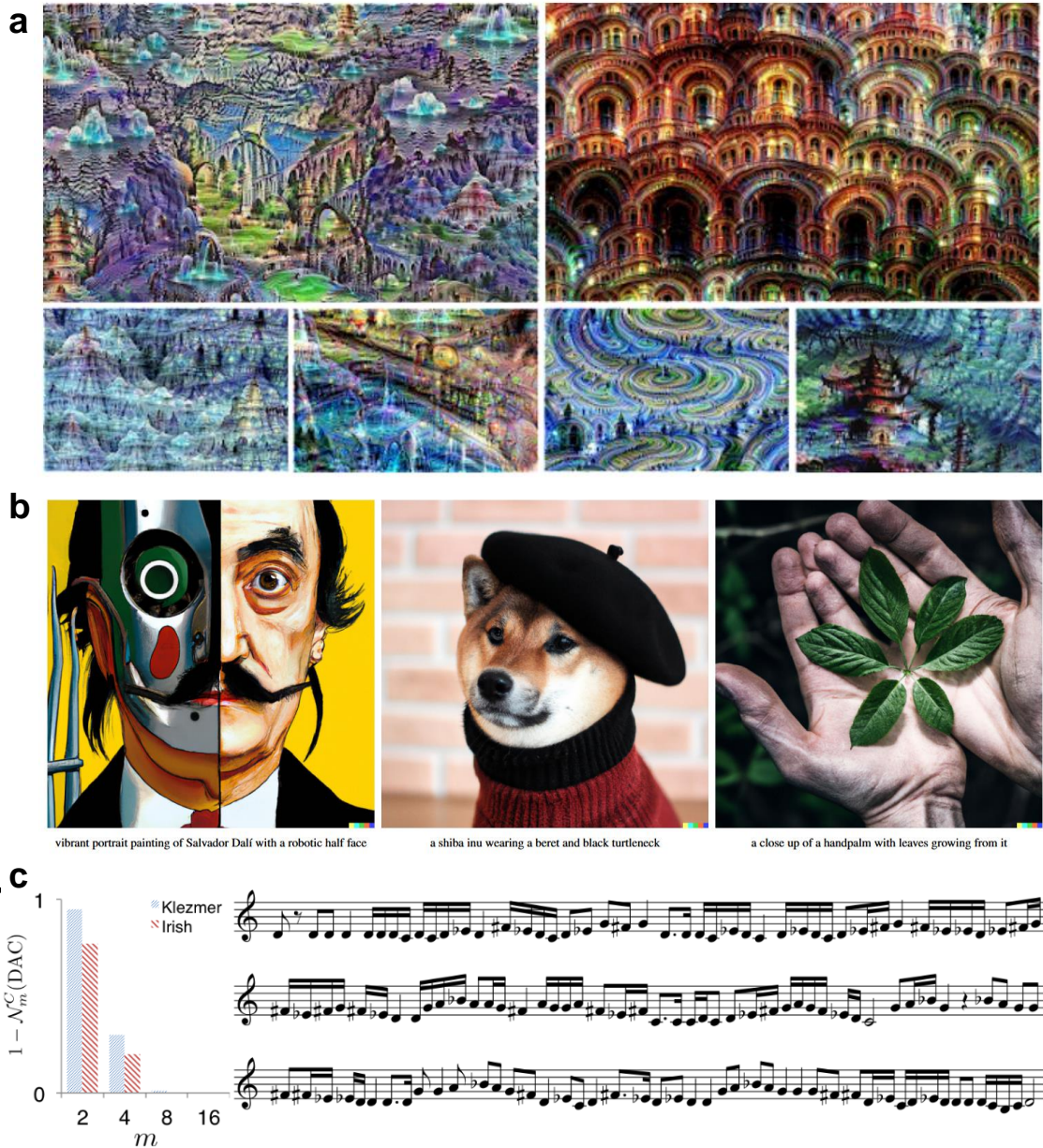


Figure 2.8: (a) Images generated with the Inception concept. The images are generated from a random-noise image, so that the result is purely the result of the neural network. Reproduced from [Mordvintsev et al., 2015]. (b) Images generated with Dall e. Reproduced from [Ramesh et al., 2022]. (c) Examples of a DAC-generated melody classified as Klezmer. Reproduced from [Colombo et al., 2017].

### 2.4.7 Generating art: image and music

In the previous paragraphs, I have described impressive applications of artificial intelligence on well-defined problems, which given some input data and instructions, lead to the desired output. Such tasks are very complex, however, they do not involve what we call *creativity*. In 1998 Margaret Boden in *Creativity and Artificial Intelligence - Why AI must try to model creativity* [Boden, 1998] wrote:

*Creativity is a fundamental feature of human intelligence, and a challenge for AI. AI techniques can be used to create new ideas in three ways: by producing novel combinations of familiar ideas; by exploring the potential of conceptual spaces; and by making transformations that enable the generation of previously impossible ideas. AI will have less difficulty in modeling the generation of new ideas than in automating their evaluation.*

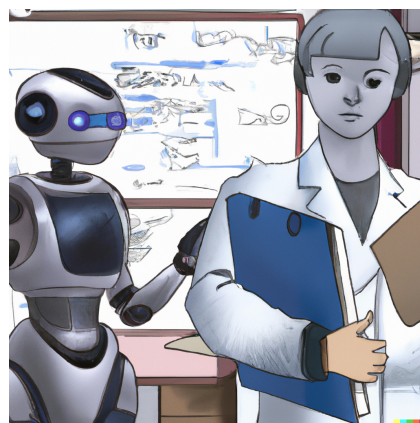
In the following decades, Boden’s suggestion to apply computational models to creative tasks was followed, and more and more sophisticated models have been developed to generate creative behaviors. This line of research has given rise to the field of *Computational creativity* [Colton and Wiggins, 2012], a subfield of Artificial Intelligence research that focuses on engineering software models able to produce artifacts and ideas. Countless examples of generative algorithms based on artificial neural networks have been proposed in creative domains [Colombo, 2021] such as painting [Mordvintsev et al., 2015, Gatys et al., 2016, Ramesh et al., 2022] and music composition [Sturm et al., 2016, Hadjeres et al., 2017, Oore et al., 2017, Oore et al., 2020, Colombo et al., 2017, Colombo et al., 2018]. In the context of figurative art created by AI, the DALL-E model [Ramesh et al., 2021] is able to generate an image from a text description. DALL-E is based on a transformer architecture that autoregressively models the text and image tokens as a single stream of data. The figures accompanying the text in this thesis have been generated with this impressive - and often surprising, model. The prompt text for each figure is reported in the image captions.

Despite the successes of AI models to generate art, one question still remains. Machines are now trained to perform creative tasks defined by their human programmers. They can generate new samples of music or visual art given human-prescribed rules, they are mimicking the human creation of art. Will they be able to come up with their own way of generating art? Or will creativity remain a feature of human intelligence? On this note, I conclude this section with the words of the early neuroscientist Sir Geoffrey Jefferson [Jefferson, 1949]:

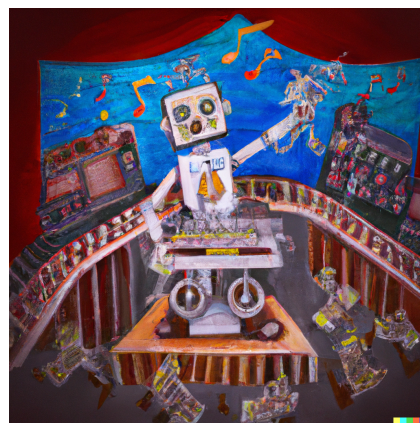
*Not until a machine can write a sonnet or compose a concerto because of thoughts and emotions felt, and not by the chance fall of symbols, could we agree that machine equals brain - that is, not only write it but know that it had written it. No mechanism could feel (and not merely artificially signal, an easy contrivance) pleasure at its successes, grief when its valves fuse, be warmed by flattery, be made miserable by its mistakes, be charmed by sex, be angry or depressed when it cannot get what it wants.*

#### Learning better diagnosis techniques through machines.

DALL-E prompt: A realistic painting of a person that teaches a robot how to read medical charts.



#### Creative AI. DALL-E prompt: A painting of a robot that composes music, on the background of a theatre.



## 2.5 Longstanding challenges of artificial neural networks

The previous section has described impressive applications of artificial intelligence techniques on disparate tasks. Deep networks have been demonstrated to perform equally well – or even better – than humans in playing complex games, driving cars, and recognizing objects, just to name a few. Furthermore, deep networks have been shown to be capable of tasks that humans cannot perform, such as predicting the onset of an illness on a long time horizon. However, such impressive results are in stark contrast with a number of limitations affecting the performance of neural networks, including the inability to perform continual learning (*i.e.*, catastrophic forgetting) and to generalize predictions to out-of-distribution test samples, the lack of robustness to targeted signal perturbations (*i.e.*, adversarial attack), the demanding time and energy consumption to train the networks. Achieving artificial intelligence requires being able to remember tasks learned sequentially, to quickly adapt to new noisy environments and changes in the tasks, and to learn with limited temporal and energy resources. Therefore, comprehensively addressing the mentioned aspects is crucial for closing the gap between artificial and biological intelligence and allowing machines to continuously adapt to changes in the environment with minimal human intervention, a process analogous to human learning [Nguyen et al., 2019]. In addition, the learning algorithms of deep neural networks are not biologically realistic. While this is not a problem per se, the discrepancies between training in AI and learning in the biological brain may be informative on the source of AI weaknesses that do not affect the brain. In the next sections, I will provide an overview of some instabilities of deep neural networks: catastrophic forgetting, lack of out-of-distribution generalization, susceptibility to adversarial attack, lack of biological plausibility, and high power consumption.

### 2.5.1 Catastrophic forgetting

The biological brain is able to learn, memorize and remember tasks learnt sequentially, an ability which has been termed *continual learning* or *life-long learning*. While it is true that humans tend to gradually forget previously learned information over long time periods, the learning of novel information generally does not catastrophically interfere with consolidated knowledge [French, 1999, Ratcliff, 1990, Parisi et al., 2018]. One of the few examples was reported in [Pallier et al., 2003], in which Pallier and colleagues studied forgetting in the context of language perception and comprehension. The behavioral experiment was performed on adult subjects, born in Korea and adopted by French families in childhood, who have become fluent in their second language and report no conscious recollection of their native language. Curiously, the Korean subjects did not exhibit different responses than a control group of native French subjects who have never been exposed to Korean. The subjects were evaluated both in behavioral tests assessing their memory for Korean and in event-related functional magnetic resonance imaging to monitor cortical activations as a response to exposure to the Korean language.

In contrast, when neural networks are trained on a sequence of tasks without being re-exposed to previously learnt tasks, they tend to forget how to perform the previous tasks. Indeed, the training focuses on acquiring information to solve the current task (*e.g.*, task B), while the network has no way of knowing that it needs to retain knowledge about the previous task (*e.g.*, task A). As a consequence, the weights in the network that are important for task A are changed to meet the objectives of task B. This leads to accuracy degradation on previous tasks while the network is learning new tasks. This phenomenon is known as *catastrophic forgetting* [Kirkpatrick et al., 2017]. Catastrophic forgetting occurs both when neural networks are trained with different but related tasks and when they are trained with non independent and identically distributed (iid) samples. This property was well-studied [Robins, 1995, French, 1999] before the deep learning renaissance that began in 2006 [Goodfellow et al., 2014a]. In 2013 Srivastava and colleagues [Srivastava et al., 2013] brought again attention to studying this aspect of modern deep neural nets, and, since then, countless attempts to address the phenomenon have been made. Due to its implications in autonomous learning agents and robots, lifelong learning is in the spotlight of machine learning research. Agents operating in the real world must deal with sensory uncertainty, efficiently process

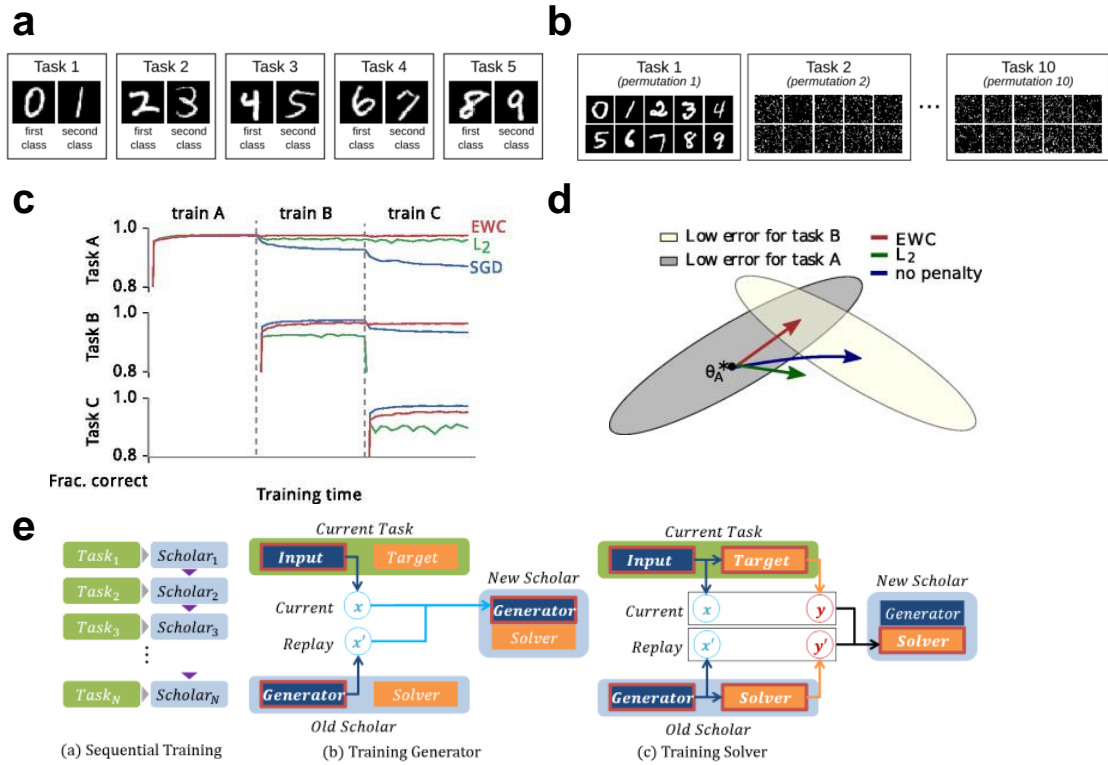


Figure 2.9: (a) Incremental MNIST. Reproduced from [van de Ven and Tolias, 2018]. (b) Permuted MNIST. Reproduced from [van de Ven and Tolias, 2018]. (c) Catastrophic forgetting on the MNIST dataset for SGD, L2 decay and EWC. Reproduced from [Kirkpatrick et al., 2017]. (d) Elastic Weight Consolidation schematics. Reproduced from [Kirkpatrick et al., 2017]. (e) Mitigating Catastrophic forgetting with Replay. Reproduced from [Shin et al., 2017]

continuous streams of multisensory information, and they effectively need to learn multiple tasks without interfering with consolidated knowledge [Parisi et al., 2018]. For example, a robot that constantly has to learn new categories as it explores a new environment may not have sufficient resources to store the training dataset nor to re-train an entire model from scratch.

In image classification, the continual learning framework can be studied in simplified settings by defining sequential tasks from standard benchmarks such as MNIST. For instance, in the incremental MNIST – or split MNIST – paradigm each task is defined as the presentation of two classes of digits (Figure 2.9a). In the permuted MNIST paradigm, instead, each task is defined as one pattern of pixel permutation that is applied identically to the images of all the 10 classes, while the labels are kept unchanged (Figure 2.9b). When a neural network is trained with SGD on the permuted MNIST protocol, the accuracy of the network on each task drops dramatically every time a new task is presented (blue line in Figure 2.9c).

Several attempts have been made to overcome catastrophic forgetting. For example, dropout regularization [Hinton et al., 2012] is able to mitigate the phenomenon to some extent in feedforward neural networks. Indeed, the dropout algorithm consistently improves the ability of standard SGD at adapting to the new task and, at the same time, remembering the old task. In some scenarios, dropout effectively increases the optimal size of the net, so the reduction of forgetting may be explained mostly by the larger nets having greater capacity. [Goodfellow et al., 2014a]. Additionally, L2 regularization, which consists in regularizing the network with a fixed quadratic constraint for each weight, is helpful in reducing the drop in accuracy (green line in Figure 2.9c). Also different activation functions have been analyzed in the context of catastrophic forgetting, with the conclusion that the choice of the function is strongly dependent on the problem [Goodfellow et al., 2014a].

Subsequently, several algorithms have been introduced specifically in an attempt to enable continual learning in neural networks. Some of these techniques are based on Bayesian inference, which retains a distribution over model parameters that indicates the plausibility of any setting given the observed data. When new data arrive, Bayesian inference is used to combine the information of previous data on the model parameters (the previous posterior) with the information given by the current data (the likelihood). Examples of Bayesian learning techniques include *incremental moment matching*, which introduces uncertainty on the parameters and rely on a Gaussian distribution to approximate the posterior distribution of parameters [Lee et al., 2017], and *Online Structured Laplace Approximations*, which combines Bayesian online learning with the Kronecker factored Laplace approximation to update a quadratic penalty for every new task [Ritter et al., 2018]. Other approaches to continual learning are based on episodic memory. For instance, Gradient Episodic Memory (GEM) is a model developed to learn over continuums of data that alleviates forgetting, while transferring beneficial knowledge to past tasks by storing a subset of the observed examples (*i.e.*, a small episodic memory) from previous tasks. [Lopez-Paz and Ranzato, 2017]. Its follow-up work presents Averaged GEM (A-GEM) [Chaudhry et al., 2018], which, thanks to a small change to the loss function, makes GEM orders of magnitude faster at training time while maintaining similar performance as GEM. Another algorithm exploits both Bayesian learning with episodic memory: the variational continual learning (VCL), which fuses online variational inference (VI), advances in Monte Carlo VI for neural networks, and a small episodic memory by combining VI with the coreset data summarization method [Bachem et al., 2015, Nguyen et al., 2017]. While the listed methods have been successful in partially preventing catastrophic forgetting, they are most likely not reflecting the mechanisms that the animal brain relies on to perform life-long learning. A number of other techniques have been proposed to mitigate catastrophic forgetting by taking inspiration from underlying mechanisms of the brain: synaptic consolidation, memory replay, and neuromodulated meta-learning. I will describe such algorithms in Section 3.2.

[A new strategy to mitigate catastrophic forgetting in this thesis: Chapter 5](#)

## 2.5.2 Lack of out-of-distribution generalization

A remarkable feature of the brain is its ability to generalize robustly. In particular, the performance of the primate visual stream is mostly unaffected by changes in the input distribution, such as illumination conditions, a shift of background, change of viewpoint, and partial occlusion due for example to raindrops in front of the object. Humans exhibit an impressive ability to acquire new skills and transfer knowledge across domains and tasks [Barnett and Ceci, 2002]. The animal visual system seems to exhibit a generic mechanism that enables generalization to distributions and distortions that are previously unseen. In this context, another crucial challenge of neural networks is building AI models that can learn robust, general concepts that remain valid outside the context of their training data, under distributional changes or domain shifts. Indeed, an unsolved weakness of neural networks is their lack of robustness when the test data are in different distributions from the training data. Specifically, neural network training relies on the *i.i.d.* assumption, that is that the training and testing data are identically and independently distributed. This is referred to as the problem of

**Vulnerabilities of artificial models.** DALL-E prompt: A 3D rendering of a robot that scratches his head on a background of a huge colorful oil painting of times square.



*out-of-distribution generalization* [Shen et al., 2021, Patel et al., 2015] (OOD generalization). Such phenomenon stems from the fact that machine learning models often rely on features that may be spuriously correlated with the label only in the training samples, leading to poor accuracy on the test samples. Figure 2.10 shows some types of out-of-distribution correlations: (a) correlation between digits and colors in the Colored MNIST dataset; (b) correlations between object entity



and image style (*e.g.*, photography, painting, cartoon, and drawing; (c) correlation between background features and object entity. Neural networks are also unstable with respect to noise and distortion patterns: for example, when the models are trained with images perturbed with white noise, they cannot generalize to testing images perturbed with salt-and-pepper noise, which is perceived similarly to white noise by humans [Geirhos, 2018].

Several works investigate one of these aspects in-depth, focusing on the divergent performance of DNN models and humans. Here I report some interesting recent examples from the literature. In [Geirhos, 2018], the authors test the ability of a classifier or human visual system to tolerate changes in the input distribution up to a certain degree. The artificial and biological visual systems are evaluated on various test distributions that differ to some degree from the training distribution (different types of image manipulations and realistic, non-i.i.d. conditions). The comparison of results of computational ANN experiments with the human psychophysical experiments revealed that the way that humans and ANNs process object information is substantially different, even when ANN training includes image distortions. Indeed, ANNs fail to generalise the distortion pattern learned during training to a different pattern applied during testing. Furthermore, the performance degradation of ANNs with decreasing signal-to-noise ratio under noise is more prominent than for the human behavioral performance. In [Bomatter et al., 2021a] the authors analyze OOD generalization in terms of the role of context in object recognition. In order to investigate the discrepancy in performance based on image background, they conduct psychophysics experiments to establish a human benchmark for in- and out-of-context recognition and compare it with both state-of-the-art computer vision models and a context-aware architecture for object recognition. The findings reveal that contextual cues can enhance visual recognition and that an unexpected context can impair visual recognition capabilities both for humans and models. In [Madan et al., 2020] the authors investigate whether generalization to out-of-distribution (OOD) category-viewpoint combinations. The work evaluates CNNs trained to classify both object category and 3D viewpoint on OOD combinations, and identifies the neural mechanisms that facilitate such OOD generalization. The results reveal that two key factors influence the generalization ability: data diversity, *i.e.*, *the number of distribution category-viewpoint combinations within a constant size dataset*, and the introduction of *Separate architecture*, *i.e.*, in which category and viewpoints are learnt in non-shared layers. Furthermore, the investigation reveals that OOD generalization is facilitated by the emergence of two types of neurons, one driving OOD generalization for category (selective to a category and invariant to a viewpoint), and the other for viewpoint (selective to viewpoint and invariant to a category). In a follow-up work [Madan et al., 2022], the authors study what aspects of domain shift make generalization challenging in the first place and how this compares with human generalization features. The investigation relies on a benchmark with various photo-realistic domains with the same geometry, scene layout and camera parameters as the popular 3D ScanNet dataset, and with disjoint and controlled domain shifts in lighting, materials, and viewpoints. The domain shifts are investigated independently, revealing that generalization is possible in different lighting conditions, but not on unseen materials and especially on viewpoints. A comparison with psychophysics experiments demonstrates that the generalization capabilities of human vision are similar on all the tested domain shifts, unlike the computational models, providing a further example of the discrepancy between the biological and the artificial vision system.

The lack of OOD generalization hinders the development of machine learning agents able to perform reliably in real cases, where often the test distribution on which the model has been deployed deviates from the training distribution. A key example is the application of vision models to autonomous driving. In this context, for safety reasons the models are required to generalize to situations that have not been presented during training in order to be reliably employed in real-world scenarios. Some examples include the cases of unseen weather, unseen lighting conditions, and presence on the road of objects that are not included in the training set.

Out-of-distribution applied to neural data in this thesis: Chapter 8



Figure 2.10: Examples of out-of-distribution correlations. (a) Correlation shift data from Colored MNIST. (b) Diversity shift data from the PACS dataset: Photo, Art Painting, Cartoon, Sketch. (c) Two dimensional out-of-distribution data from the NICO dataset: in cage, in water, on grass, and others. All panels reproduced from [Bai et al., 2021]

### 2.5.3 Susceptibility to adversarial attack

A particular case of lack of robustness to image perturbation is the longstanding challenge of deep learning known as *adversarial attack*. The such phenomenon consists in perturbing the input with minimal, carefully crafted and targeted manipulation, in order to change the output of the model toward a targeted response. In vision tasks, the change of only a few pixels, which is mostly imperceptible to the human eye, leads to a confident misclassification in network response [Szegedy et al., 2013, Goodfellow et al., 2014b]. The analysis in [Goodfellow et al., 2014b] demonstrates that adversarial attack can be explained as a property of high-dimensional dot products and that vulnerability to adversarial attack arises from linear behavior in high-dimensional spaces. Indeed, as the perturbation on the input can grow linearly with the dimensionality of the weight vectors, many infinitesimal changes to the input can add up to one large change to the output. Furthermore, adversarial samples designed for one specific network architecture are often effective in fooling the output of other models. This can be explained as a consequence of the fact that adversarial perturbations are highly aligned with the weight vectors of a model, and different models learn similar functions when trained on the same task [Goodfellow et al., 2014b].

A common technique to generate adversarial samples relies on adding to the input an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input [Goodfellow et al., 2014b]. Several other techniques have been proposed to build adversarial samples imperceptibly different from the original images, including rotating the image by a small angle in the direction of the gradient. Additionally, recent works release the constraint of imperceptibility to humans and shift the focus of adversarial attacks towards interpretable and physically-realizable adversarial features. For instance, [Casper et al., 2021] demonstrated that adversarial samples can be built not only as pixel-level perturbations but also by inserting a generated visual patch on top of natural images. Thanks to the “copy-paste” nature of the method, such adversarial attacks can be physically realized for example by printing the generated patch on paper and sticking it on top of a physical object. Such a strategy is inspired by the observation that in nature some animals such as butterflies use adversarial eyespots to confuse predators.

This counterintuitive property of neural networks reveals that models trained with backpropagation are exposed to intrinsic instabilities and blind spots. In a society where deep learning has been increasingly used in applications in safety-critical tasks such as autonomous driving, the vulnerability of models to small targeted perturbations represents a crucial safety issue [Xu et al., 2019]. Indeed, adversarial examples are inputs to machine learning models that an attacker intentionally designed to arbitrarily push the model to make the wrong prediction output. Physically realizable adversarial attack strategies prove therefore to be a threat to real-world applications of DNNs.

### 2.5.4 Temporal and spatial credit assignment problem

A pivotal open question in neuroscience is how upstream synapses are adjusted on the basis of downstream errors [Lillicrap et al., 2016]. In machine learning, the estimation of how much each

parameter has contributed to the output error is referred to as the problem of *synaptic credit assignment*. The backpropagation algorithm (BP) solves the credit assignment problem by computing the update of each parameter as its derivative with respect to the network’s loss. This is implemented by backpropagating the gradient of the error from the top to the bottom layers of the network through the transpose of the forward weights  $W$ .

While this strategy is highly effective on a broad range of tasks, it has been criticized for relying on biologically unrealistic mechanisms that cannot be implemented by the neurons in the brain [Crick, 1989, Whittington and Bogacz, 2019]. Some theories support the claim that the brain might be able to implement the underlying principles of BP [Lillicrap et al., 2020], however, a few aspects of BP appear to be at odds with neurobiology.

- The backpropagation algorithm [Rumelhart et al., 1995] relies on a forward and a backward pass. During the forward pass, the input signal is propagated from the input layer to the output layer, where the error is computed by comparing the network’s output with the target. During the backward pass, the error flows from the top layers to the bottom layers through the same weights used in the feedforward pathway (Figure 3.1a). Sharing the weights in the feedforward and in the feedback pathway leads to the weight symmetry or *weight transport problem* [Burbank and Kreiman, 2012]. This is not compatible with the architectural features of distributed systems such as the brain, in which elementary computing units do not have bidirectional connections with the same weight in each direction [Neftci et al., 2017]. Weight symmetry, furthermore, requires rapid information transfer back along axons from each of its synaptic outputs [Crick, 1989].
- For each synapse, the error gradient — and the associated synaptic update — is computed through the recursive application of the chain rule. Therefore, the parameter updates depend on the activity of all downstream nodes, which means that feedback neurons would need to have precise information about all the synaptic weights in the forward pathway units. The error derivative needs to be transported as a second signal through the network, requiring the derivative of the non-linearities to be known. Lillicrap described this aspect of BP as requiring a *highly orchestrated computation and delivery of third factors to individual neurons in the hidden layer*. On the contrary, biological synapses learn based on local signals related to the activity of the neurons they connect with [Whittington and Bogacz, 2019].
- The error gradients are stored separately from the activations [Liao et al., 2016] and do not influence the activities of the nodes produced in the forward pass. Hence, during the backward pass, the network activity is frozen. In the brain, instead, the neural activity is not frozen during plasticity changes and the signals traveling through feedback connections influence the neural activities produced by forward propagation, leading to their enhancement or suppression [Lillicrap et al., 2020]. Furthermore, BP requires distinct forward and backward (or ‘positive’ and ‘negative’) phases. How forward and backward pathways interact in the brain remains unclear, however, there is no evidence that supports separate alternating phases [Bartunov et al., 2018].
- Input signals cannot be processed in an online fashion, but each sample needs to wait for both the forward and backward computations to be completed for the previous sample. Indeed, since the update of the parameters of a given layer depends on downstream layers, parallelization of the backward pass is unfeasible [Launay et al., 2019, Jaderberg et al., 2016]. This is referred to as the *update locking problem* [Jaderberg et al., 2017, Czarnecki et al., 2017].

Further weaker criticisms are related to the lack of cell-type variability and to gradient signals being both positive and negative, which is unlikely to model a plausible neural computation [Bartunov et al., 2018]. Additionally, during training, the modulatory signals often vary across many orders of magnitude, which become extreme in the case of exploding and vanishing gradients [Lillicrap et al., 2020]. These considerations have motivated the development of several approaches to train neural networks alternative to BP. These methods demonstrate that exact backpropagated

gradients are not essential for learning deep representations. Section 3.3 reviews the most successful biologically inspired approaches to credit assignment.

Besides the non-locality in space, BP is also non-local in time when applied to recurrent neural networks in the form of the Backpropagation Through Time (BPTT) algorithm. In order to track all past activities, BPTT requires to unroll the network in time. Hence, it implies an offline computation of the gradients, which is non-local in time. Furthermore, the network unrolled in time becomes very deep as the input-sequence length increases [Bohnstingl et al., 2022]. Consequently, in settings with small networks that do not require online learning, BPTT is a powerful approach, but, for larger networks, the memory overhead becomes impractical [Zenke and Neftci, 2021].

Furthermore, some of the biologically unrealistic aspects of BP hinder its efficient applicability to specific hardware. The implementation of BP on a neural substrate is challenging as it involves employing symmetric weights, carrying out the operations such as multiplications with derivatives and activation functions, propagating error signals with high, floating-point precision, alternating between forward and backward passes, and changing the sign of synaptic weights. Neuromorphic hardware generally models dynamics closely related to that in the brain, hence the requirements listed above make the implementation of BP on neuromorphic chips difficult. Additionally, BP requires the immediate availability of network-wide information stored with high-precision memory during learning, and precise operations that are difficult to realize in neuromorphic hardware. [Neftci et al., 2017].

### 2.5.5 Energy and data-hungry models

The impressive progress performance of neural networks has been empowered by the increased availability of large computational resources that necessitate high energy consumption. Training an artificial neural network is costly both financially, due to the cost of hardware, electricity and cloud computing time, and environmentally, due to the CO<sub>2</sub> footprint caused by modern tensor processing hardware [Strubell et al., 2019]. The emissions associated with the training of cutting-edge models are comparable to, for example, the lifetime carbon emissions of a car [Kaack et al., 2022]. In 2018 OpenAI released an analysis showing that since 2012, the amount of computing used in the largest AI training runs has been increasing exponentially with a 3.4-month doubling time [Amodei and Hernandez, 2018]. The computational resources for network training are quantified in floating point operations (FLOPs), *i.e.*, the number of additions and multiplications of scalar values required to obtain a result. The precise mapping from FLOPs to energy consumption is hardware- and algorithm-dependent, however, more FLOPs generally correspond to higher energy use. Figure 2.11(a) shows that the trend of the total amount of computing, in petaFLOP/s-day, used to train different neural network models as a function of time is exponential. Figure 2.11(b) compares the estimated CO<sub>2</sub> emissions from training common natural language processing models, compared to the average CO<sub>2</sub> emission for 1 year of a person, 1 year of an American person, and the lifetime of a car. The elevated energy resources required to develop and deploy models are increasing, together with the associated carbon footprint, thereby raising interest in the impact of artificial intelligence on climate change. The ICT sector accounts for around 1.4% of GHG emissions today, of which ML probably accounts for a small, but unknown, share [Kaack et al., 2022]. On the other hand, Machine Learning is a versatile tool that can be used to build applications that alleviate bottlenecks in addressing climate change, such as tracking deforestation, forecasting renewable power production and transportation demands, and improving the efficiency of cooling and heating systems [Kaack et al., 2022].

#### Power hungry machines.

DALL·E prompt: A photorealistic image of a robot sitting at a table set with many light bulbs on the dishes and in the glasses.



The dual impact of Artificial Intelligence on climate change is increasingly becoming central in the focus of both industrial and academic research. Start-ups such as Axelera AI and Nebuly are working towards more sustainable AI, *i.e.*, the deployment of hardware and software that can minimize time and energy consumption of neural networks training, inference and deployment, with the final aim of contributing to reducing the global computing-related impacts of data centers. In a parallel research line, the field of neuromorphic computing strives to design and deploy algorithms and chips that are energy efficient by mimicking the same organizing principles of the biological nervous system [Douglas et al., 1995, Boahen, 2005]. In particular, neuromorphic circuits rely on spiking representations for communication, learning and memory, and asynchronous event-based computation, which are suitable for building modular systems and creating complex hierarchies of computation [Indiveri and Horiuchi, 2011]. I will discuss spiking neural networks, their dynamics and learning rules in Section 3.4.

Artificial intelligence models are not only power-hungry but also data-hungry. In particular, the performance of supervised classification algorithms is often dependent on the availability of a sufficient amount of training data. However, labeling samples is expensive and time-consuming due to the significant human effort involved [Jalilian et al., 2017]. Data labeling, therefore, represents a bottleneck in the development of supervised learning algorithms. As a result, it is desirable to have methods that learn a classifier with high accuracy from a limited number of labeled training data. As an alternative method to supervised learning, self supervised learning is a training framework for neural networks that addresses the scarcity of annotated data. In self supervised learning, the technique is to generate pseudo labels for data through data augmentation, such as random crop, rotation, geometric transformations, and color augmentation. The training phase updates the networks' parameters to embedding augmented versions of the same sample close to each other while trying to push away embeddings from different samples [Jaiswal et al., 2020]. This technique is successful in extracting input features. Extracted features can be evaluated for example by mapping with a read-out layer to classes in a classification task. I will discuss self supervised learning in Section 3.3.3.

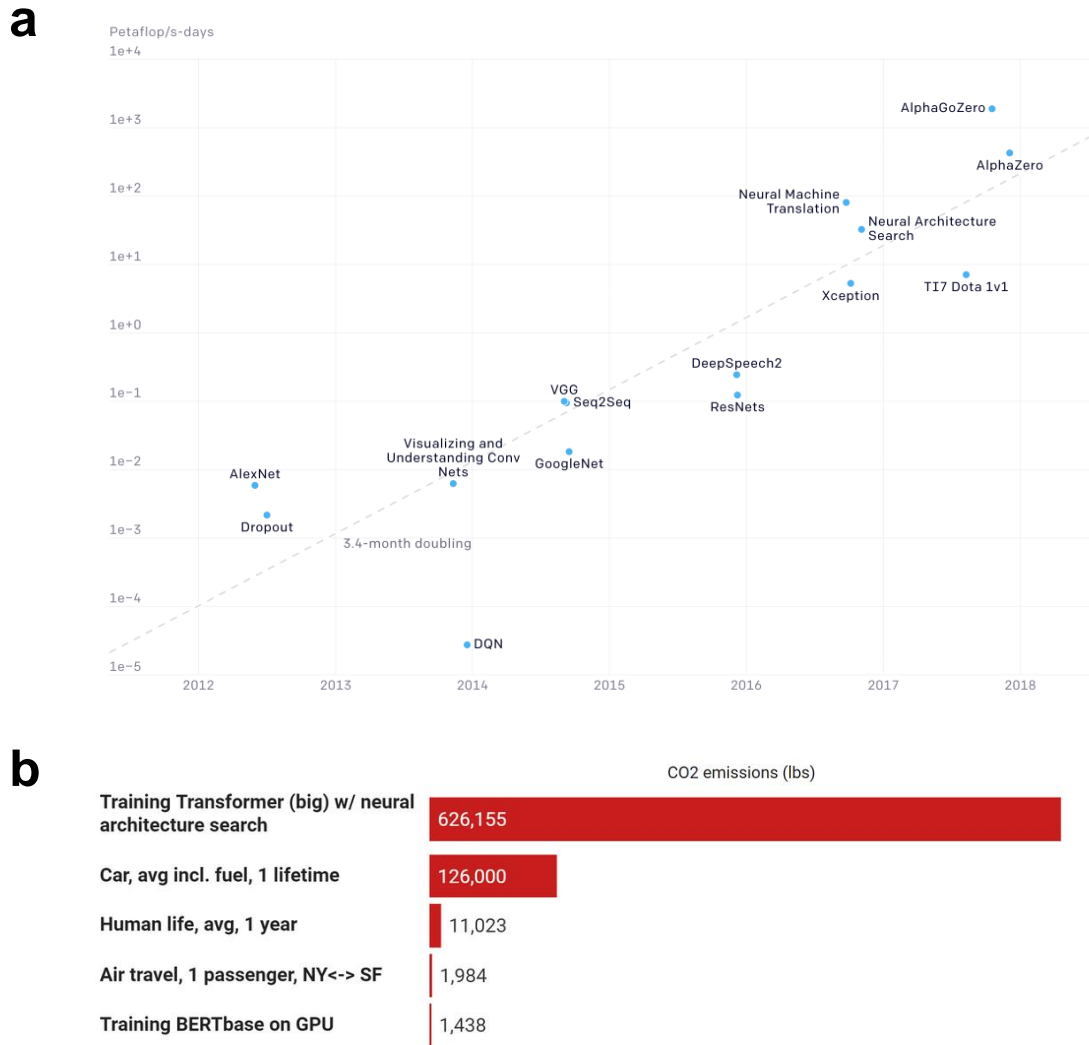


Figure 2.11: (a) The total amount of compute, in petaflop/s-days, used to train different neural network models. The trend is exponential. Reproduced from [Amodei and Hernandez, 2018] (b) Estimated CO2 emissions from training common NLP models, compared to familiar consumption. Reproduced from [Strubell et al., 2019, Kamperis, 2019]

## Chapter 3

# Biologically Inspired Algorithms for Neural Networks

While backpropagation-based learning has achieved impressive results on a broad range of tasks, neural networks invariably suffer from limitations such as catastrophic forgetting, adversarial attack, and lack of generalization to out-of-distribution samples. The animal brain is not affected by such issues, therefore designing training algorithms for neural networks more compatible with the learning mechanisms of the brain is a promising direction to address the mentioned limitations. This motivation has triggered a recent wave of research into training schemes that are more brain-like and which achieve performance competitive with backpropagation in perceptual tasks. In this section, I will discuss which principles of biology can be used to inform the development of neural networks. I will outline some practical examples of how brain-inspired computations have been successfully used to overcome weaknesses of neural network training, such as catastrophic forgetting. Then I will review biologically inspired learning rules which have been proposed as an alternative to backpropagation toward more brain-like training of ANNs. Finally, I will introduce the basic concepts of neuromorphic computing and spiking neural networks, developed with the goal of energy-efficient and fault-tolerant models.

### 3.1 Principles of biological computations

A multitude of experimental results in neurobiology hint that biological synapses and systems act in much more complex ways than the artificial scalar synapses at the basis of current deep neural networks [Zenke et al., 2017]. Current models overlook aspects of neurophysiology that might be relevant to improve their performance. Biological circuits may inspire the development of artificial models at different levels of complexity and from several perspectives. First, at a low level, the complex morphology of biological networks in terms of diversity of neuron types, synaptic plasticity, connectivity motives, and network topologies offers copious examples of mechanisms to enrich the dynamics of artificial networks. Learning in the brain is mediated by a rich set of neurophysiological principles that regulate changes in synaptic strength. As a prominent example, synaptic plasticity in the brain is driven mainly by local signals, such as the activity of neighboring neurons [Whittington and Bogacz, 2019]. The local interaction between synapses plays a crucial role in regulating weight changes during learning. For example, the mechanism of heterosynaptic competition controls synapse growth by limiting the total strength of synapses connected to the same pre- or postsynaptic neuron [Royer and Paré, 2003b]. This phenomenon occurs as a nonlinear competition across synapses at each neuron. Specifically, as the summed weight of synapses into (or out of) a neuron hits a threshold, all the incoming (or outgoing) synapses to that neuron undergo a slight heterosynaptic long-term depression (“summed-weight limit rule”) [Fiete et al., 2010]. Additionally, in the cortex, each neuron tends to target a specific firing rate, and synaptic strengths are regulated to keep such rates constant, despite input perturbation [Turrigiano et al., 1998]. Synaptic scaling acts

as a global negative feedback control of synaptic strength, regulating the weight changes based on the neural local activities [Turrigiano, 2008, Moulin et al., 2020, Ibata et al., 2008]. These homeostatic mechanisms are typically not modeled in the training of standard ANNs, which rely on global signals instead of local information [Nøkland and Eidnes, 2019, Bengio et al., 2016].

Second, the underlying architectural principles of the animal brain represent a powerful guide to inform the design of complex deep network models. A remarkable feature that is fundamental to most of the state-of-the-art AI models is the hierarchical organization of networks, which extract progressively higher-level features with increasing depth.

Third, at a systemic level, the evolution has led to the development in the animal brain of sophisticated neural circuits specialized for different tasks, including sensing, navigation, and central pattern generation for locomotion.

[Enriching the dynamics of Backpropagation with principles of synaptic plasticity in biology in this thesis: Chapter 5.](#)

## 3.2 Addressing catastrophic forgetting with biologically inspired strategies

The brain presents the remarkable ability to perform continual learning, which still constitutes an unsolved challenge for neural networks. Hence, implementing algorithms that mimic the underlying mechanisms of the brain may be a promising strategy to overcome such a fundamental limitation for neural networks. A large body of computational methods has been developed by taking inspiration from the biological learning mechanisms of the mammalian brain [Parisi et al., 2018]. In this section, I present two of the most common brain-inspired techniques to mitigate catastrophic forgetting in AI: elastic weight consolidation inspired by synaptic consolidation and replay inspired by the replay mechanisms in the brain. These methods reflect two main mechanisms for mitigating forgetting in neural networks identified by [Parisi et al., 2018], namely regularization mechanisms to constrain parameter updates and replay of previous knowledge.

### 3.2.1 Approaches inspired by synaptic consolidation

Experimental evidence shows that the learning of a new task in the animal brain encompasses the strengthening of a fraction of excitatory synapses, which consists of the increase in the volume of individual dendritic spines of neurons. Synaptic consolidation is a mechanism in the brain that reduces the plasticity of synapses that are essential to perform previously learned tasks. This acts as a protection mechanism in neocortical circuits that enables long term retention of memories. Therefore, synaptic consolidation accounts, at least in part, for the retention of the knowledge related to the learned skill, and consequently for the ability of the brain to perform continual learning. Such a mechanism is related to the so-called *stability-plasticity* dilemma, which regards the trade-off of a system between the plasticity needed to integrate and adapt to new knowledge and the stability needed to retain previously consolidated skills. The balance between the two aspects is essential to assimilate new knowledge in order to adapt to new environments, and at the same time prevent catastrophic forgetting [Grossberg, 1987, Mermillod et al., 2013, Benna and Fusi, 2016]. Several regularization methods towards continual learning have been inspired by synaptic consolidation. One of the earliest algorithms attempting to alleviate catastrophic forgetting by constraining the network's parameters similarly to synaptic consolidation is the Learning without Forgetting (LwF) method, which optimizes both for high accuracy for the new task and for the preservation of responses on the existing tasks from the original network. Memory retention from previous tasks is achieved by constraining the network for each original task to have output probabilities for each image to be close to the recorded output from the network trained on previous tasks [Li and Hoiem, 2016].

In the same direction, another approach proposed in [Kirkpatrick et al., 2017], known as Elastic Weight Consolidation (EWC), takes inspiration from synaptic consolidation. The method seeks to preserve knowledge by making a fraction of synaptic weights less plastic and therefore making



the encoded memories more stable over long timescales. This principle is implemented in EWC by constraining relevant weights to each task to remain close to the distribution learnt during training on the specific task. In practice, when a network is trained on a sequence of two tasks, task A and task B, during the training of task B, the parameters that are most relevant to perform task A are kept close to a region of low error for task A. Figure 2.9(d) gives a schematic representation of the described principle. The constraint is modeled as a quadratic penalty, thus it acts as a spring anchoring the parameters to the previous solution, where the stiffness of a parameter is higher for parameters more relevant to the task. The relevance of a parameter  $\theta$  on task A is computed based on the approximation of the posterior probability  $p(\theta, \mathcal{D}_A)$  to a Gaussian with mean given by the parameters  $\theta_A^*$  and a diagonal precision given by the diagonal of the Fisher information matrix  $F$ .  $F$  has three properties: (i) It is equivalent to the second derivative of the loss near a minimum, (ii) it can be computed from first-order derivatives alone and is thus easy to calculate, and (iii) it is guaranteed to be positive semidefinite. This technique can be formalized as minimizing the loss  $\mathcal{L}$ :

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta_{A,i}^*)^2, \quad (3.1)$$

where  $\mathcal{L}_B(\theta)$  is the loss related to task B,  $\lambda$  weights how strong the constraint on task A should be, and  $i$  is the index running on the network’s parameters. This method proves to substantially mitigate catastrophic by increasing the fraction of retained memories over a sequence of tasks both in supervised setting scenarios, such as binary classification of random patterns and supervised learning with the permuted MNIST task (Figure 2.9(c)). Additionally, EWC proves to be successful in enabling continual learning in reinforcement learning (RL), in particular on the Depp Q Networks (DQNs) architectures. In this context, EWC is applied to introduce constraints to protect previously acquired knowledge. Furthermore, it is combined with the additional high-level mechanisms to detect which task is being performed, to incorporate the new tasks and to switch among tasks. This allows EWC to know which quadratic constraints to update when the task context changes. The novel approach is evaluated on experiments which consist of 10 games chosen randomly from those that are played by DQNs at least at human-level performance. While with plain gradient descent methods the agent does not learn to play more than one game, with EWC the agents are able to learn to play multiple games.

Also, the method designed in [Zenke et al., 2017] relies on an estimate of the synapse’s *importance* toward solving problems encountered in the past. The approach exploits internal synaptic dynamics to enable ANNs to learn sequences of classification tasks so that each synapse accumulates task-relevant information over time. This leads to the computation of the importance measure efficiently and locally during training. The importance reflects past credit for improvements of the task objective and thus represents the local contribution of each synapse to the change in the global loss. Such information is then used to rapidly store new memories without forgetting old ones. New tasks are learned through synaptic changes occurring mostly in the synapses that were not important in previous tasks, while the important synapses are consolidated. This allows avoiding catastrophic forgetting of these past tasks.

### 3.2.2 Approaches inspired by the replay mechanism

Biological mechanisms are characterized by separate, interactive memory systems [O’Reilly and Norman, 2002]. Experimental evidence reported in [Gelbard-Sagiv et al., 2008] demonstrates that the human brain exhibits phases of free recall without external cues after being exposed to a stimulus such as a movie clip. In the entorhinal cortex, neurons that selectively fire when the subject is watching a specific video are found to be active also during the recall of the same stimulus. A similar observation has been made for the entorhinal cortex and hippocampal units at the population level. Also in the rodent brain, hippocampal neurons have been shown to generate a replay of previous firing sequences during sleep and rest states after locomotion [Wilson and McNaughton, 1994, Skaggs and McNaughton, 1996, Lee and Wilson, 2003]. Cells co-activated when the animal moved in specific locations of the space during subsequent sleep were more prone to fire than in sleep

episodes preceding the behavioral tasks [Wilson and McNaughton, 1994]. The activation pattern, furthermore, reflect the order in which the cells fired during spatial exploration [Skaggs and McNaughton, 1996]. Such phenomenon of sequential replay occurs also during the awake state immediately after the behavioral tasks. In the reactivation, the recent episodes of spatial experience are replayed in a temporally reversed order [Diba and Buzsáki, 2007, Foster and Wilson, 2006]. These observations lead to the hypothesis that the encoding and reactivation of sequential experience in the hippocampus is essential for spatial learning in rodents and the formation of long-term memories of events in time in humans [Lee and Wilson, 2003]. Additionally, the bidirectional re-enactment of temporal sequences may contribute to the establishment of higher-order associations in episodic memory [Diba and Buzsáki, 2007]. Therefore, episodic and spatial memories can arise from the encoding of complex associations in hippocampal neuronal circuits, whose consolidation involves the reactivation of the original network firing patterns during sleep and rest [O'Neill et al., 2010]. In this framework, the complementary learning systems theory [McClelland et al., 1995] postulates that the hippocampal system exhibits short-term adaptation and allows for the rapid learning of novel information which will, in turn, be played back over time to the neocortical system for its long-term retention. The hippocampus features a rapid learning rate and encodes sparse representations of events to minimize interference. Conversely, the neocortex employs a slow learning rate and builds overlapping representations of the learned knowledge. This may indicate that information is consolidated in the neocortex via the reactivation in the hippocampus of encoded experiences in terms of multiple internally generated replay [Ratcliff, 1990, Parisi et al., 2018].

Inspired by the neuroscientific evidence of replay in the mammalian brain, in order to address the phenomenon of catastrophic forgetting in neural networks several methods have been proposed which rely on replaying raw inputs or compressed representations during training. In practice, replay in neural networks is implemented as mixing new instances with old ones and fine-tuning the network with this mixture [Parisi et al., 2018]. Some methods realize this technique by storing and replaying raw pixels. This is known as *rehearsal regime* [Ratcliff, 1990, Robins, 1995]. Rehearsal is defined as the relearning of a subset of previously learned items (three in the original formulation) at the same time that each new item is introduced. The effect of replay is, similar to replay in the hippocampus, protecting the previously learned items from disruption by new items. In particular, one of the most successful methods is the *sweep rehearsal*, in which the training buffer is dynamic and the three previously learned items are chosen at random for each epoch, which leads the number of intervening trials to significantly increase. Several follow-up works have expanded on the replay framework. For instance, [Shin et al., 2017] introduced the deep generative replay framework, which allows sequential learning on multiple tasks by synthesizing and rehearsing fake data that mimics former training examples. The cooperative interplay between a deep generative model (generator) and a task-solving model (solver) allows one to sample training data for previous tasks and interleave them with those for a new task. Another interesting example of raw-pixel replay is the approach presented in [Castro et al., 2018] which trains DNNs incrementally through a combination of new samples and a small exemplar set corresponding to samples from the old classes. The parameter updates are driven by a loss composed of a distillation measure to retain the knowledge acquired from the old classes, and a cross-entropy loss to learn the new classes.

While it is able to alleviate catastrophic forgetting, raw-pixel replay involves redundant information and is memory-inefficient. As an alternative in the framework of replay, other methods were proposed in connection with the hippocampal indexing theory [Teyler and Rudy, 2007], which claims that the hippocampus stores compressed representations of neocortical activity patterns while awake. These patterns are replayed and then the corresponding neocortical neurons are re-activated via reciprocal connectivity with the effect of consolidating memories [Lewis and Durrant, 2011, O'Neill et al., 2010]. Approaches relying on this theory interleave compressed representations with samples from the new tasks during training. One of the first methods showing that interleaving information of new tasks with internally generated patterns of previous experiences helps consolidate existing knowledge without explicitly storing training samples was proposed already in 1995 [Robins, 1995]. This approach goes beyond the rehearsal regime, with the aim of addressing the limitation by which previously presented training information may not be available anymore. It introduces the *pseudorehearsal regime*, which does not need to access the initial training popula-

tion, but rather employs artificially constructed populations of *pseudoitems* instead of the original previously learned items. A pseudoitem is constructed by generating a new input vector (setting at random 50% of input elements to 0 and 50% to 1 as usual), and passing it forward through the network in the standard way. The output vector generated by this input is the associated target output. The resulting pseudoitems map the function appropriate to reproducing the actual population. More recently, [Draeos et al., 2016], inspired by the process of adult neurogenesis in the hippocampus, combined pseudorehearsal with autoencoders to add new neurons to deep layers of artificial neural networks in order to facilitate their acquisition of novel information while preserving previously trained data representations. Another approach using compressed representation is REMIND, which relies on the replay of latent features using memory indexing and on a codebook for constructing the memory, which is pre-computed and fixed [Hayes et al., ]. By removing the constraint of a fixed codebook and avoiding generative replay, [Zhang et al., 2021] implements a model called Hypotheses-driven Augmented Memory Network (HAMN), which efficiently consolidates previous knowledge using an augmented memory matrix of “hypotheses” and replays reconstructed image features to avoid catastrophic forgetting. The model learns a set of hypotheses in the augmented memory, based on which it constructs the latent representation of an image as a linear combination of the hypotheses. Through multi-head content-based attention, the model shares the augmented memory over multiple image features so that features can be reconstructed from hypotheses in parallel. Compared to pixel-level replay, HAMN is more efficient in terms of memory, as it avoids redundant information in the image pixel space.

### 3.3 Towards bio-plausible training for ANNs

As described in Section 2.5.4, the BP algorithm lacks biological realism in many regards, including non-locality, weight transport, freezing of activity, and backward locking. These considerations have motivated the development of alternative methods for credit assignment, each proposing solutions to some of these criticisms. Biologically plausible training strategies not only may give a better intuition for learning mechanisms in biological brains, but may also open new computational possibilities, such as updating layers asynchronously [Launay et al., 2019]. Furthermore, from a pure machine learning perspective, there is growing interest in neuron-like architectures with local learning rules, mainly driven by the progress in neuromorphic hardware [Illing et al., 2019].

In this section, I will first introduce the arguments supporting the hypothesis that the brain could implement updates relying on BP-like error signals. In this context, I will introduce the *neural gradient representation by activity differences* (NGRAD) framework [Lillicrap et al., 2020]. Then, I will provide an overview of the most competitive and promising biologically inspired training algorithms for ANNs alternative to BP, including sign symmetry [Xiao et al., 2018], feedback alignment [Lillicrap et al., 2016], weight mirror circuits [Akroud et al., 2019], the Kolen Pollac reciprocal network [Akroud et al., 2019], direct random target propagation [Frenkel et al., 2019], global error vector broadcasting [Clark et al., 2021], deep feedback control [Meulemans et al., 2021], the least control principle [Meulemans et al., 2022], and the contrastive, local and predictive plasticity [Illing et al., 2021]. I will analyze the achievements, requirements, and limitations of the proposed algorithms, outlining promising directions for extensions and applications of the current frameworks. With specific references and examples, I will outline best practices to consider when implementing and applying such techniques to various tasks, *e.g.*, in image classification. I will focus on presenting results on image recognition, as most of the recent publications benchmark the proposed algorithms on such a task. In particular, to provide a baseline, I remark that, according to the analysis in [Illing et al., 2019], biologically plausible training schemes are typically defined as successful if they reach around 98% test accuracy on the MNIST data set. Finally, I will indicate the still unsolved theoretical aspects of rigorously analyzing the convergence properties of some of the training methods.

The biologically inspired methods can be grouped into two main classes: *labeled learning* – including local classifier and supervised clustering – and *unlabeled learning* – including self supervised learning and predictive coding. We remark that despite their biological inspiration, all the de-

scribed algorithms have a substantial difference from learning in the brain: they train networks initialized from scratch according to different schemes. In contrast, before any type of learning, the brain is already endowed with prior knowledge that has been optimized with evolution. Such priors are expressed in the form of neural architectures and connectivity strengths. Consequently, the brain only needs a relatively small amount of error-driven learning in order to acquire even very complex skills [Lillicrap et al., 2020].

### 3.3.1 The neural gradient representation by activity differences (NGRAD) framework and predictive coding

Lillicrap and colleagues [Lillicrap et al., 2020] argue that in spite of the apparent differences between BP and observed principles of biological learning, the brain may be able to implement the core computations of BP. The quest for understanding if biological neurons may implement a BP-like mechanism is motivated by two main reasons. First, while BP may find a set of trained weights via a different pathway than that of the brain, in some settings the resulting receptive fields have been shown to be similar to those studied in the brain [Crick, 1989]. Striking examples show that neural networks trained with BP can simulate response properties in the posterior parietal cortex [Zipser and Andersen, 1988], in vestibulo-ocular reflexes [Anastasio and Robinson, 1989], and - more recently - in the primary motor cortex [Lillicrap and Scott, 2013]. Another example is provided by the so-called *substitute models of the brain* - or *digital twins*. These networks, which are trained with BP, are currently the best models to predict and explain neural responses. Milestone works have demonstrated that CNNs mapped to the ventral stream can predict and even drive the activity of small neuron populations in rodents' V1 [Walker et al., 2019] and in macaque V4 and IT [Bashivan et al., 2019, Yamins et al., 2014, Ponce et al., 2019]. Other non-BP-related approaches such as Gabor filters achieve a substantially lower performance in predicting response properties. Second, networks trained with BP can achieve impressive results in very complex tasks - as reviewed in Section 2.4 - outperforming any other training strategy in a broad range of applications. This successful result may indicate that the brain may exploit precise error-driven feedback for learning.

An early hypothesis was that BP might be implemented via the retrograde transmission of information along axons [Brandt and Lin, 1996, Lillicrap et al., 2016]. However, this is in contrast with experimental findings that retrograde transport operates on timescales significantly slower than forward propagating neural activity [Harris, 2008]. Another proposed explanation is that error information can be propagated through a second network, however, this requires that weights are symmetric [Kording and König, 2001]. As for the signed error signals, it has been suggested that, in the cortex, firing rates above a certain threshold convey positive error signals, and rates below this value convey negative errors. However, these computations would need elaborate book-keeping to integrate and propagate the signed error across multiple layers as signed information [Lillicrap et al., 2020].

The theory proposed in [Lillicrap et al., 2020] is that biological circuits could compute synaptic updates by using feedback connections to elicit neuron activities whose locally computed differences encode BP-like error signals. This relies on the principle that higher-level activities - dictated for example by targets or larger spatial or temporal contexts - can nudge lower-level activities towards values that are more consistent with the higher-level activity or the desired output. Consequently, the information on the nudged activity (*i.e.*, top-down-driven activity) can drive local synaptic adjustments. Interestingly, a number of training schemes propose to compute the weight updates based on differences in activity states, rather than on propagated gradients. Such approaches successfully avoid propagating activities and error derivatives. The neural gradient representation by activity differences (NGRAD) is a framework collecting the learning algorithms relying on this idea. We define the NGRAD hypothesis as the idea that the cortex uses an NGRAD to implement an approximation to gradient descent.

Some algorithms complying with the NGRAD principles take inspiration from the predictive coding framework [Rao and Ballard, 1999]. Predictive coding is a general learning framework that was originally proposed as a model of visual processing and, in particular, as a hierarchical and pre-

dictive strategy for encoding natural images. In predictive coding in a hierarchical model network, nodes at different levels of hierarchy are connected by both feedforward and feedback connections. Each level attempts to predict the responses at the next lower level via feedback connections, and the error between this prediction and the actual response is sent back to the higher level via feedforward connections. This model of visual processing explains the property of endstopping (also known as *extra-classical* receptive field) in visual cortical areas. In endstopping a vigorous response to an optimally oriented line segment is reduced or eliminated when the same stimulus extends beyond the neuron’s classical receptive field (RF). The predictive coding framework entails that neural networks learn the statistical regularities of the natural world, and signal deviations from such regularities to higher processing centers. This reduces redundancy by removing the predictable components of the input signal [Rao and Ballard, 1999]. In the NGRAD framework, Whittington and colleagues [Whittington and Bogacz, 2017] propose to apply predictive coding for supervised learning settings through a local Hebbian plasticity algorithm. The activity differences driving synaptic updates are between sets of neurons in a local circuit. In this scheme, a predictive coding network is shown to converge to BP in a certain limit of parameters, supporting the NGRAD hypothesis that local updates based on activity difference can, at least theoretically, approximate BP-like synaptic adjustments.

Another example of algorithms fitting in the NGRAD framework is the Generalized recirculation (GeneRec) scheme [O’Reilly, 1996], developed in the auto-encoder framework. GeneRec builds on the Recirculation algorithm [O’Reilly, 1996, Hinton and McClelland, 1987], a learning algorithm to train autoencoders. The original recirculation approach is based on comparing the activations of the network on the original input to the activations of the reconstructed input. The input from the hidden units changes the state of the visible units and the new state is fed through the system again. Finally, the difference in activations is used to compute the parameter updates. For monotonic activation functions, this gives an approximation of the error derivative, with a different magnitude but with the correct sign. GeneRec trains multilayer recurrent networks with sigmoidal units through two phases. In a *minus phase*, the network is presented with the input, and recurrent activities are allowed to settle to equilibrium. In a *plus phase*, input is provided to the network and simultaneously the environment provides the target output activation, which is propagated to the output and hidden layers. The output neurons are nudged towards their target values, and activities are again allowed to settle to equilibrium. The weight update is proportional to the difference between the product of the presynaptic and postsynaptic activities from the plus and minus phases [O’Reilly, 1996, Lillicrap et al., 2020]. Therefore the learning rule is local, however, the requirement for long positive and negative settling phases for computing the activities is not supported by neurophysiology findings [Bartunov et al., 2018].

A further possible implementation of the NGRAD backbone for learning is to drive updates based on activity differences between different compartments of multi-compartmental neurons. Several algorithms [Urbanczik and Senn, 2014, Guerguiev et al., 2017, Sacramento et al., 2018a, Asabuki and Fukai, 2020] alleviate the requirement of having two separate pathways for feedback signals [Lillicrap et al., 2016] and demonstrate that local error signal can be computed by having two distinct dendritic compartments in each hidden layer neuron: a *basal* compartment, strongly coupled to the soma for integrating bottom-up sensory information, and an *apical* compartment for integrating top-down feedback to drive plasticity [Guerguiev et al., 2017]. Biologically, in the neocortex, electrotonically segregated apical dendrites receive top-down inputs, a phenomenon known as *dendritic segregation*. A pioneer work in this direction by Urbanczik and Senn proposes a plasticity rule which is driven by the correlation of presynaptic activity with a postsynaptic, somatodendritic prediction error, rather than by the correlation between pre- and postsynaptic activity, as in Hebbian learning [Urbanczik and Senn, 2014]. In this framework, learning seeks to minimize discrepancies between somatic firings and a local dendritic potential, which may arise from stochastic fluctuations and synaptic input, which is received by the soma. This model can be applied to unsupervised, supervised, and reinforcement learning.

Extending on the principle of learning based on segregated dendritic compartments, Guerguiev and colleagues [Guerguiev et al., 2017] demonstrate that dendritic segregation can be exploited in multi-layer networks to learn image categorization. Neurons in different layers can coordinate

local synaptic weight updates via distinct basal and apical compartments that integrate feedback and feedforward signals separately. Similarly to DTP, feedback from higher layers is used to construct local targets at the hidden layers. The resulting algorithm takes advantage of multilayer architectures to identify useful higher-order representations, reproducing the hallmark of deep learning. Tested on the MNIST dataset, the scheme applied to one- and two-hidden layer networks achieves better accuracy than on single-layer networks. The algorithm is constrained to supervised learning settings with a teaching signal.

Sacramento and colleagues [Sacramento et al., 2018a] follow a similar research line of learning by reducing the prediction error between the somatic potential and their dendritic branch potential, but further integrate into the architecture a second population of cells, that of interneurons with both lateral and cross-layer connectivity (*e.g.*, somatostatin-positive, SST, cells). Therefore, each hidden area of the network consists of a microcircuit with pyramidal cells and lateral inhibitory interneurons. Lateral synaptic input from local interneurons learns to cancel top-down feedback from downstream brain areas. Both the lateral synapses from and onto interneurons are trained. Such an algorithm can be applied to learn non-linear regression tasks, MNIST classification task, input reconstruction and denoising. Compared to [Guerguiev et al., 2017] in which the error for learning the bottom-up synapses does not influence the somatic activity, here the apical dendrite has an explicit error representation at every moment in time by simultaneously integrating top-down excitation and lateral inhibition. This allows us to avoid two-phase training and to solve the credit assignment problem for time-continuous input streams.

Finally, Asabuki and Fukai proposed a plasticity rule to train networks based on somato-dendritic mismatch in unsupervised settings [Asabuki and Fukai, 2020]. Learning occurs as a self-supervising process in which a single neuron (more precisely, the soma) generates an appropriate supervision signal to learn the spatiotemporal firing patterns repeated in upstream neurons (projecting to the dendrites of the neuron). This enables efficient learning of clustered temporal events in a completely unsupervised manner, which can be applied to perform blind source separation of mutually correlated signals, temporal chunking, and reproduce experiments of audio sources separation through embedded repetition [Dellaferreira et al., 2022a].

The results obtained via the described NGRAD algorithms lead Lillicrap and colleagues to the conclusion that BP may offer a conceptual framework for understanding how the cortex learns, despite it is still unclear how some aspects of the NGRAD framework could be implemented by the neurons, in particular how to compute differences between neuronal activities and use this information to drive synaptic updates [Lillicrap et al., 2020].

### 3.3.2 Supervised learning

The following algorithms propose different solutions to the described biological limitations of BP. They all rely on labeled samples, by either using error or target information to compute the parameter updates.

To mathematically describe the proposed frameworks, I adopt the same notation used in [Nokland, 2016]. Given a mini-batch of input-output vectors  $(x, y)$  and a networks with two hidden layers (Figure 3.1a) with weights  $W_i$  and bias  $b_i$ , the activations of the networks are computed as:

$$a_1 = W_1x + b_1, h_1 = f(a_1), \quad (3.2)$$

$$a_2 = W_2h_1 + b_2, h_2 = f(a_2), \quad (3.3)$$

$$a_y = W_3h_2 + b_3, y = f_y(a_y), \quad (3.4)$$

where  $f(\cdot)$  is the hidden non-linearity and  $f_y(\cdot)$  is the output non-linearity. Given a loss function  $J$ , the error at the output layer is computed as the gradient of the loss function with respect to the output activations:

$$e = \delta a_y = \frac{\partial J}{\partial a_y} \quad (3.5)$$

The gradients for the hidden layers are computed with BP as:

$$\delta a_2 = \frac{\partial J}{\partial a_2} = (W_3^T e) \odot f'(a_2) \quad (3.6)$$

$$\delta a_1 = \frac{\partial J}{\partial a_1} = (W_3^T \delta a_2) \odot f'(a_1) \quad (3.7)$$

$$(3.8)$$

Finally, the weight updates are computed based on the gradients as:

$$\delta W_3 = -\eta e h_2^T \quad (3.9)$$

$$\delta W_2 = -\eta \delta a_2 h_1^T \quad (3.10)$$

$$\delta W_1 = -\eta \delta a_1 x^T \quad (3.11)$$

$$(3.12)$$

where  $\eta$  is the learning rate.

**Feedback Alignment (FA)** Lillicrap and colleagues [Lillicrap et al., 2016] proposed one of the first solutions to the weight transport problem. They introduced a deep learning algorithm derived from approximations of BP. Their method leverages the same connectivity structure as BP, but uses independent forward and backward pathways. Their findings demonstrate that fully connected neural networks can be effectively trained when in the feedback path the transpose of the forward weight matrix ( $W^T$ ) is substituted by a fixed random matrix ( $B$ ). Given the same notation and settings described for BP, in FA the gradients for the hidden layers are computed as:

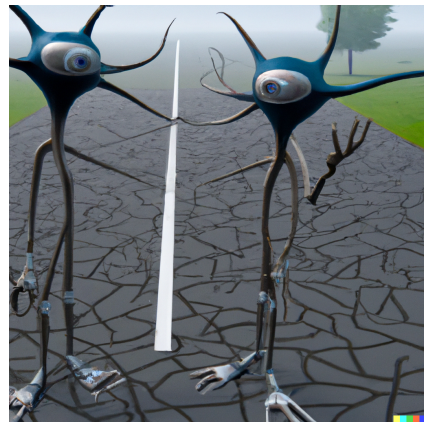
$$\delta a_2 = (B_2 e) \odot f'(a_2) \quad (3.13)$$

$$\delta a_1 = (B_1 \delta a_2) \odot f'(a_1) \quad (3.14)$$

$$(3.15)$$

Figure 3.1c shows schematics of the forward and backward connections in a fully connected network. This shows that the precise symmetric connectivity between forward and backward pathways is not needed to achieve learning. The only constraint in the choice of the random matrix is that  $B$  needs to satisfy  $e^T W B e > 0$ , where  $e$  is the network's error. During training, only the forward pathway ( $W$ ) is updated, by propagating the gradient of the loss function through the backward path ( $B$ ). The backward parameters ( $B$ ) are left unchanged. The dynamics of the training algorithm push the updates of the forward weights toward a soft alignment with the backward weights. This mechanism allows the error to flow backward through the network to deliver useful teaching signals for the updates of the forward weights. Indeed, since soft alignment between  $W$  and  $B^T$  is encouraged, the forward weights evolve towards a setting resembling that of backpropagation, with the backward weights being equal to  $W^T$ . In practice, the rough transposition makes the updates of FA similar enough to those of BP that the network is able to learn simple tasks with an accuracy comparable to BP. This weak symmetry, however, is not forced nor achieved through weight transport, as it emerges spontaneously via the dynamics of the training algorithm, allowing us to solve the weight symmetry problem. Such dynamics are guaranteed by the only constraint on the  $B$  matrix, *i.e.*,  $e^T W B e > 0$ . When the inequality is satisfied, the teaching signal (or modulator signal) backpropagated by the matrix,  $B e$ , lies within 90 degrees of the signal which would be prescribed by BP ( $W^T e$ ).

**Separate synaptic pathways: an illustration of Feedback Alignment.** DALL·E prompt: A 3D rendering of a humanized neuron with eyes and mouth that is standing on a road that forks in two.



This ensures that the updates prescribed by  $B$  push the network toward the same direction as those of BP. More specifically, analytical results suggest that  $W$  evolves towards alignment with the Moore-Penrose pseudoinverse of  $B$ . Under this condition, in networks with one hidden layer, feedback alignment is able to asymptotically reduce the error to zero.

The computation of modulatory signals in feedback alignment requires the decoupling of forward and backward activity and their interaction during learning. Biological circuits could support decoupling through multiple plasticity mechanisms. For instance, the modulator signals can be provided by inputs that arrive at different times or to different subcellular compartments [Clopath and Gerstner, 2010, Kwag and Paulsen, 2009, Sjostrom and Hausser, 2006], or via different types of synapse [Coemans et al., 2004, Yang and Lisberger, 2014]. Furthermore, forward and backward pathways could interact via the post-synaptic voltage, which would allow inference and learning processes to interact. Additionally, feedback alignment might be related to learning paradigms for multi-compartmental neuron models that update the synaptic strength based on forward and teaching signals reaching separately the dendritic and the somatic compartments [Urbanczik and Senn, 2014, Asabuki and Fukai, 2020]. The decoupling of forward and backward pathways allows simplifying the computation and transport of third factors, as the modulatory signals are delivered via random connections and are computed on the basis of random connections in the backward pathway.

However, while FA represents a pivotal step towards solving the weight transport problem, it cannot scale to complex models and tasks [Bartunov et al., 2018]. Furthermore, FA is still affected by the other issues of BP. First, the error signals do not affect the post-synaptic activities (*i.e.*, the network is frozen during the backward pass and the signed error vectors are delivered via a distinct path [Bartunov et al., 2018]). Second, the modulatory signals still require having precise information about all the random connections (*i.e.*, non-local). Third, the training is backward-locked. Forth, FA requires transporting signed error information in the backward pass [Bartunov et al., 2018]. Finally, while the disjoint forward and backward path solves the weight symmetry problem, it introduces a further biologically unrealistic aspect. The disjoint feedback pathway may exist in an architecture with a second set of ‘error’ neurons that act as the feedback variables. However, this would imply that there exist entirely segregated feedforward cortical neurons that are unaffected by feedback activity. This is at odds with experimental evidence showing that feedback in the cortex tends to be area-wise reciprocal and that feedforward and feedback pathways appear to be active and interacting, at least to some extent, simultaneously [Lillicrap et al., 2020].

**The sign symmetry algorithm (SS)** The sign symmetry approach, also known as sign-concordant feedback, [Liao et al., 2016] is another approach to training neural networks addressing the weight symmetry problem. Empirical results show that when the feedback matrix is different from the transpose of the forward matrix, the magnitude of feedback weights does not matter for the performance, while the sign of the feedback weights is crucial for the network to learn. This means that, if the feedback matrix is built by combining the sign of the forward weights with random magnitudes, training leads to performance comparable to BP. In practice, the sign-concordant feedback training is implemented by using feedback matrices that share the sign with the forward matrices, but use random magnitudes (Figure 3.1b). The sign symmetry algorithm along with several variations the feedback matrix is tested on several datasets: MNIST, CIFAR-10, CIFAR-100, and TIMIT. For many datasets, the performance obtained using the same sign but random magnitudes, together with batch normalization and batch manhattan, leads to perform better than standard BP (but not than BP with batch normalization and batch manhattan). We remark that in the sign symmetry approach the feedback connections are changing with training. This is a major difference from FA, where the feedback matrix is random but fixed. [Bartunov et al., 2018] investigated the scalability of FA to more complex methods and datasets, revealing that FA does not work in more complex scenarios. Building on these results, [Xiao et al., 2018] demonstrates that the sign symmetry training scales to more complex datasets such as ImageNet and MS COCO. It evaluates five configurations: (i) BP, (ii) FA on all layers, (iii) FA on convolutional layers and BP on fully connected layers, (iv) sign-symmetry on all layers, (v) sign-simmetry on convolutional layers and BP on fully connected layers. The findings show that BP achieves the



best performance and that the sign-symmetry models are only slightly worse than BP. FA exhibits the worst accuracy when applied to all layers, but its performance improves when SS is applied only to convolutional layers and the fully connected layers are trained with BP. We underline that the sign symmetry training implies that the feedback weights keep track of the evolution of the forward weights. This might explain the gap in performance with FA.

**Direct Feedback Alignment (DFA)** The feedback alignment training algorithm succeeds in training networks with a few hidden layers without weight transport. While precise symmetry between forward and backward pathways is not required in FA, the connectivity pattern needs to be the same, *i.e.*, all the neurons that are connected with a forward connection also need to be connected in the backward direction. Furthermore, in FA only connections between contiguous layers are contemplated. While neurophysiology insight support reciprocal connectivity between cortical areas [Gilbert and Li, 2013], the error signal may be relayed through areas not involved in the forward pathway to reach more distant layers. This condition requires the forward and connectivity pathways to be disconnected. Building on the FA results, Nokland [Nokland, 2016] proposed to train networks with disjoint connectivity patterns: the forward path is composed of connections between contiguous layers (as in BP and in FA), while the backward path is formed by connections that carry the error signal directly from the output layer to each hidden layer independently. Such circuitry is shown in Figure 3.1d. In DFA, the modulatory signals for the hidden layers are computed as:

$$\delta a_2 = (B_2 e) \odot f'(a_2) \tag{3.16}$$

$$\delta a_1 = (B_1 e) \odot f'(a_1) \tag{3.17}$$

$$\tag{3.18}$$

The empirical results in [Nokland, 2016], show that the performance obtained by DFA and FA is comparable and only slightly lower than BP for fully connected architectures. The accuracy gap increases in the case of convolutional models. Additionally, the direct error transportation pathway employed by DFA has the advantage that the modulatory signal at every hidden layer is non-zero for all non-zero error  $e$ . Therefore, DFA is not vulnerable to the vanishing gradient issue and enables training of very deep networks (*i.e.*, 100 hidden layer fully connected architectures), that neither BP or FA can train with simple initialization schemes. Furthermore, DFA has been shown to be effective in a broad range of applications in state-of-the-art settings and network types beyond fully connected models. Some successful applications are neural view synthesis, recommender systems, geometric learning with graph convolutions, and natural language processing with Transformers [Launay et al., 2020a]. Additionally, the asynchronous weight updates and the single operation at the center of the training stage enable new classes of training co-processors (*i.e.*, photonic co-processors [Launay et al., 2020b]), leveraging dedicated hardware to perform the direct error propagation with optical random projections. However, a systematic study on DFA [Launay et al., 2019] showed that DFA is unable to train deep convolutional architectures, as convolutional layers systematically fail to align. This phenomenon prevented DFA from scaling to complex architectures such as VGG-16 and complex computer vision tasks like ImageNet [Refinetti et al., 2021].

While there is no empirical evidence that a mechanism analogous to DFA is implemented in the brain, Nokland’s findings are essential to demonstrate that ANNs can be trained not only without weight transport but also with disconnected forward and feedback pathways, *i.e.*, in the framework of *synaptic asymmetry* [Launay et al., 2020a]. This result is important as the training does not require knowing the derivative of the non-linearity. Additionally, the parameter updates depend only on the pre-synaptic activity and the temporal derivative of the post-synaptic activity, allowing the learning rule to be local. The update phase can be considered to be not a separate phase, but as the final computation of an extended forward pass. Finally, the error signal can be considered to be a separate signal relayed by other neurons, addressing the issue of BP in which the error is a second signal, separate from the input, in the neurons participating in the forward pass.

**Indirect Feedback Alignment (IFA)** With the same motivation driving the design of DFA, Nokland designed a second feedback path configuration known as indirect Feedback Alignment. As in DFA, the forward and backward pathways are disconnected. However, the backward path presents a connectivity framework different from the direct output-hidden layer connections of DFA. In IFA, the error is propagated directly from the output layer to the first hidden layer. Then, the modulatory signal computed at the first hidden layer is propagated in a forward fashion through the same weights used for the forward computations.

In such schematics, the modulatory signals for the hidden layers are computed as:

$$\delta a_2 = (W_2 \delta a_1) \odot f'(a_2) \quad (3.19)$$

$$\delta a_1 = (B_1 e) \odot f'(a_1) \quad (3.20)$$

$$(3.21)$$

The IFA algorithm provides another demonstration that fully connected architectures can be trained without weight transport and with disconnected forward and feedback computations. Additionally, compared to DFA, IFA requires using a single random matrix instead of one matrix per layer, thereby saving memory. However, this comes at the cost of lower and more fragile performance. Empirical results on IFA are not reported in the original paper [Nokland, 2016], nor, to the best of my knowledge, have been reported in any follow-up work.

**Weight Mirrors (WM)** Bartunov and colleagues [Bartunov et al., 2018] evaluated the performance of biologically inspired training schemes, including FA, DFA and target propagation, on image classification datasets with increasing complexity (MNIST [LeCun and Cortes, 2010], CIFAR [Krizhevsky et al., a], ImageNet [Deng et al., 2009a]). The results revealed that FA has a performance only slightly lower than BP on MNIST and CIFAR-10 with fully connected architectures. However, FA struggles with locally connected networks and is not able to scale up to training large networks on ImageNet. Given these findings, the authors hint that the ‘research should continue into new physiologically realistic learning algorithms that can scale-up’. Akrouit and colleagues [Akrouit et al., 2019] take up the challenge and propose novel methods to address the scalability issues of FA outlined in [Bartunov et al., 2018]. The underlying idea is that, since FA leads to the learning of simple tasks through the soft alignment between the feedback and the forward weights, an improvement of such an alignment might empower FA to handle harder tasks. To this goal, they propose two approaches to learn appropriate feedback weights without weight transport. The first method, the *weight mirrors* (WM), proposes to adjust the initially random feedback matrix  $B$  so that it becomes proportional to the forward matrix  $W$ . By taking inspiration from the alternation of wakefulness and sleep states in the animal brain, the WM algorithm alternates two modes in training: engaged and mirror modes. During the *engaged mode*, the network receives input samples specific to the task (*i.e.*, ImageNet images) and learns the forward weights  $W$  through the standard FA algorithm. In the *mirror mode*, the feedback weights are adjusted to mimic the forward ones, *i.e.*, the initially random feedback matrix  $B$  is updated so that it becomes proportional to the transpose of the forward weight matrix  $W$  without weight transport. To avoid weight transport, a noise vector  $x_i$  is given as input to each layer  $i$  separately. At each layer,  $x_i$  passes through the forward weight matrix  $W_i$ , and the output of the layer  $y_i = W_i x_i$  is computed (Figure 3.1e). The noise vector  $x_i$  models a mechanism in which neurons discharge noisily, that could occur during sleep or during development in utero. The  $B_i$  matrix is adjusted relying on the locally available information on  $x_i$  and  $y_i$ , where  $y_i$  contains information on  $W_i$ . The update of  $B_i$  is based on the *transposing rule*  $\delta B_i = \eta x_i y_i^T$ . In practice,  $B_i$  integrates a signal, called the *teaching signal*, which is proportional to  $W_i^T$  on average. During training, the initial values of  $B_i$  shrink away, and  $B_i$  converges to a positive scalar multiple of  $W_i^T$ . Therefore, throughout training weight mirroring keeps the forward and feedback weights in agreement, achieving an alignment angle in the range of  $[6^\circ, 12^\circ]$ . The subscript  $i$  denotes the index of the layer. The computations in the mirror mode are performed layer by layer. Importantly, in this technique, weight decay or synaptic scaling are essential to keep the norm of the  $B$  weights small. When tested on a ResNet-18 and on a ResNet-50 architectures trained on ImageNet, WM outperformed FA and SS, and obtained an accuracy only slightly lower than BP.

**Kolen-Pollack (KP) reciprocal network** The second approach presented in [Akrou et al., 2019] to improve the agreement of the feedback and forward weights in FA builds on the Kolen-Pollack (KP) algorithm to alleviate the weight transport problem. The principle of the Kolen-Pollack (KP) algorithm is that parameter updates can rely on the transport of changes in weights rather than the transport of weights. As in WM, the forward and backward matrices ( $W$  and  $B$ ) are initialized differently, and then, during training, they are updated with identical weight adjustments. This technique, however, is not biologically plausible, as different synapses are unlikely to undergo the same adjustments. The novel algorithm proposed in [Akrou et al., 2019], ( called KP reciprocal network) builds on KP and proposes the following update rules.  $W$  is updated based on its input vector and a teaching vector  $\delta$  sent from the feedback path (as in feedback alignment).  $B$  is updated based on its input and a cell-specific scalar teaching signal from the forward path. The update of the feedback weights  $B_l$  at layer  $l$  is computed as:

$$\Delta B_{l+1} = -\eta y_l \delta_{l+1}^T, \quad (3.22)$$

where  $\delta_{l+1}^T$  is a scalar teaching signal from the forward path. If the updates of the forward and feedback weights have the same learning rate and weight decay, then the  $B$  matrix converges to the transpose of  $W$ . Compared to the WM algorithm, this approach only needs a single mode of operation and does not need to update the backward pathway layer by layer. Additionally, the KP network kept the alignment of the forward and feedback matrices in a range closer to zero than WM. When tested on a ResNet-18 and on a ResNet-50 architectures trained on ImageNet, KP outperformed FA and SS, and obtained an accuracy only slightly lower than BP, and slightly better than WM.

Similarly to DFA, both the WM and the KP algorithms rely on disjoint sets of neurons. Importantly, in WM and KP the distinct neurons are paired such that each forward-path cell is connected with a feedback-path cell. The authors speculate that such one-to-one wiring may appear during development. However, the WM and KP schemes are unlikely to be implemented exactly as they are in the brain, as they required the input-related activity and the teaching signal flowing through the cell without interfering.

**Node perturbation (NP)** [Lansdell et al., 2020] proposes a strategy to alleviate the weight symmetry problem that relies on combining three techniques: (i) approximating BP via a reinforcement learning scheme, (ii) adding noise to neural activity, and (iii) integrating global and local feedback signals. The training algorithm, which is called *Node Perturbation* (NP), trains a feedback system to provide synthetic gradients which approximate the modulatory signals that would be prescribed by BP. The algorithm can be described as a two-learner system, in which one network enables the other network to learn more efficiently. The such framework may be compatible with the dynamics of certain biological circuits, including the dendritic trees of pyramidal neurons containing an apical and a basal component, and the ensemble of climbing fibers and Purkinje cells in the cerebellum. The NP scheme is implemented as a network that learns to use feedback signals trained with reinforcement learning via a global reward signal. In practice, the loss gradient term is replaced with a synthetic estimate that is learned through noisy perturbation of the neural activity. At each layer, the loss is linearized and noise is introduced to perturb the layer’s output and the loss function. The estimate of the error used to update the weights is computed based on the correlation between the perturbed loss and the introduced noise. Importantly, such a technique can be applied to learn appropriate feedback weights  $B$  in an architecture analogous to that of FA. Theoretical and empirical results demonstrate that NP is able to train the  $B$  matrix to converge to the forward weight matrix  $W$ .

During training, the feedback network co-adapts with the evolving feedforward weights to continue to provide a useful error signal. The alignment angle between  $B$  and  $W$  and the sign congruence are better than the ones obtained with standard FA, supporting the hypothesis that NP communicates error signals between layers better than FA. If the feedback learning rate is zeroed, the scheme falls back to FA. In fully connected architectures trained on MNIST, NP achieves an accuracy comparable to BP, surpassing the performance of FA. In autoencoders trained on MNIST, while FA performs poorly, NP performs better than BP. Such an improvement may arise from the addition

of noise that encourages the learning of more robust latent factors. Finally, in convolutional models trained on CIFAR-10 and CIFAR-100, NP performs worse than BP but better than FA.

While NP outperforms FA in the tested tasks, in terms of biological plausibility NP only solves the weight transport problem, but is affected by the same issues of FA, namely backward locking, freezing of neural activity, and non-locality. Compared to the WM scheme [Akrouf et al., 2019], NP also relies on noise signals to learn the feedback matrices, however, it does not require two separate phases of learning, not layer-by-layer updates.

**Target Propagation (TP) and its variants** Target Propagation [Lecun and Soulie Fogelman, 1987, le Cun, 1986, Bengio, 2014, Lee et al., 2015] is a training scheme for neural networks that proposes a solution to both the biological criticisms and the vanishing gradient issue of BP. The previously described algorithms update the parameters by associating to each unit a *loss gradient*, or an approximation of it. The underlying principle of TP, instead, aims to compute targets rather than gradients, *i.e.*, each unit is associated to a *target value*.

In TP, the forward pass is implemented as in BP, in which activity is propagated forward through successive layers of a network to produce an output. In the update phase, the output target is propagated backwards through feedback connections. The backpropagation of the output target generates at each layer target activities, *i.e.*, activities that would have led the network to output the correct response. The layer-wise difference between such target activity and the actual activations obtained during the feedforward pass is used to drive the forward weights. In practice, this means that each layer is trained to reconstruct the layer below. In other words, the feedback parameters are learned through layer-wise auto-encoding of the forward layers, where auto-encoding refers to a learning rule in which the target is the input itself. Therefore, reconstruction is exploited as a layer-local training signal. In the limit in which the target signal is close to the feedforward value, the dynamics of TP approached that of BP.

Using the same notation as described above, the updates prescribed by TP in a supervised learning setup can be formulated through the following equations. We introduce the  $f'()$  shorthand notation on top of the previously described notation:

$$h_l = f'_l(h_{l-1}) = f_l(W_l h_{l-1}) \quad (3.23)$$

The output target is driven by the gradient of the global loss, and is defined as the output activation tweaked in the direction of the negative gradient [Meulemans et al., 2020]:

$$\hat{h}_L = h_L - \hat{\eta} e_L = h_L - \hat{\eta} \left( \frac{\partial \mathcal{L}}{\partial h_L} \right)^T, \quad (3.24)$$

where  $\hat{\eta}$  is the output target step size.  $h_L$  is backpropagated through layer-wise inverse to compute hidden layer targets as:

$$\hat{h}_l = g'_l(h_{l+1}) = g_l(B_l h_{l+1}), \quad (3.25)$$

where  $g'_l$  is an approximate inverse of  $f'_{l+1}$ ,  $B_l$  the feedback weights and  $g_l$  a smooth nonlinear activation function. The forward activations and the target signals are used to compute the local loss per layer - also known as *forward loss* - as:

$$L_l(\hat{h}_l, h_l) = \|\hat{h}_l - h_l\|^2. \quad (3.26)$$

Finally, the forward weights  $W_l$  are updated by taking a gradient descent step on this local loss, assuming that  $\hat{h}_l$  stays constant. Analogously, the backward weights  $B_l$  (the decoder part) are trained based on the *reconstruction loss*:

$$L_l^{rec}(g'_l(f_{l+1}(h_l)), h_l) = \|g'_l(f_{l+1}(h_l)) - h_l\|^2. \quad (3.27)$$

This approach allows learning an approximate inverse  $g'_l$ , making the  $f'_l / g'_l$  pair look like an auto-encoder. Furthermore, in order to extend the inverse mapping to a neighborhood of the training samples, [Lee et al., 2015] proposes to modify the loss through noise augmentation:

$$L_l^{rec}(g'_l(f_{l+1}(h_l)), h_l) = \|g'_l(f_{l+1}(h_l + \epsilon)) - (h_l + \epsilon)\|^2, \quad (3.28)$$

with  $\epsilon \sim N(0, \sigma)$ .

For effective learning, the TP dynamics requires that the layer-wise parameter updates, each improving a layer-wise loss, also lead to an improvement of the global loss. [Lee et al., 2015] analytically demonstrates that, in the case that  $g_l$  is a perfect inverse of  $f_l$  and  $f_l$  has a certain structure, the update direction of TP does not deviate more than  $90^\circ$  from the gradient direction estimated by BP, which always leads to a decrease of the global loss.

From the biological perspective, TP does not require symmetric forward and feedback pathways, and the reconstruction-driven updates at each layer may be considered local. However, TP is still affected by the alternation between a feedforward and ‘frozen’ backward phase, and there is no neurophysiological justification for the origin of the output targets. The disjoint forward and backward paths, as for FA, could be thought of as part of an architecture with a second set of ‘error’ neurons that could act as the feedback variables [Lillicrap et al., 2020].

On the computational side, TP is a very unstable optimization algorithm. Indeed, in classification tasks the same 1-hot vector in the output will always map back to precisely the same hidden unit activities in a given layer. This leads to instability when different samples of the same class have different appearances: TP will attempt to make their representations identical even in the early layers. Additionally, there are no guarantees about the dynamics of TP when the inverses are imperfect [Bartunov et al., 2018].

**Difference Target Propagation (DTP)** Difference target propagation (DTP) is a stabilizing linear correction for the imperfect inverse mapping of TP which is effective to make TP actually work. [Lee et al., 2015] observes that the imperfection of the inverse function in TP leads to severe optimization problems when assigning targets based on the original equation. In this context, DTP [Lee et al., 2015] builds on the principle of TP and tweaks the original formulation through the following linearly corrected formula:

$$\hat{h}_{l-1} = h_{l-1} + g'_l(\hat{h}_l) - g_l(h_l) \quad (3.29)$$

If  $g'_l$  is the inverse of  $f'_l$ , DTP becomes equivalent to standard TP. Such a correction significantly improves the optimization stability of TP. Note that, in the formulation proposed in [Lee et al., 2015], the penultimate layer target is computed using gradients from the network’s loss and not the updates of TP.

In simulations, [Lee et al., 2015] demonstrates DTP enables multilayer training on supervised image classification tasks. On fully connected multilayer models with tanh nonlinearity DTP reaches an accuracy only slightly lower than BP both on MNIST and on CIFAR-10. The gap in performance widens when the tanh activation is replaced with ReLU. Regarding scalability to more complex scenarios, [Bartunov et al., 2018] demonstrated that DTP fails to train architectures on ImageNet.

**Simplified Difference Target Propagation (SDTP)** Biologically, a major issue presented by DTP is the update of the last layer, which requires computing the output gradient via BP. A plausible reason for this choice is that this approach ensures that the penultimate layer receives diverse targets despite the low-dimensional 1-hot targets at the output layer. To address this limitation, [Bartunov et al., 2018] introduces weight-transport-free variants of DTP, to remove the BP updates of the penultimate layer. In SDTP the target for the penultimate layer is formulated as:

$$\hat{h}_{L-1} = h_{L-1} - g(h_L; \lambda_L) + g(\hat{h}_L; \lambda_L), \quad (3.30)$$

where  $\hat{h}_L = \operatorname{argmin}_{h_L} \mathcal{L}(h_L)$ , which is the correct label distribution. While it removes the biologically implausible gradient communication, this strategy only produces weak targets for the penultimate layer. Therefore SDTP is integrated with an auxiliary output that enriches the target signal with information from the input. [Bartunov et al., 2018] introduces a composite structure for the output layer, that combines the predicted class distribution  $o$  with an auxiliary output vector  $z$  that provides additional information about activations of the penultimate layer. The target

for penultimate layer is then computed as:

$$\hat{h}_{L-1} = h_{L-1} - g(o, z; \lambda_L) + g(\hat{o}, z; \lambda_L), \quad (3.31)$$

Furthermore, [Bartunov et al., 2018] introduces an additional variant of DTP, which, unlike DTP that alternates learning in forward and backward weights, optimizes both the forward and the reconstruction loss simultaneously. On MNIST and CIFAR-10, SDTP is able to train the networks with an accuracy slightly lower than DTP on MNIST and CIFAR-10. Given this result and the failure of DTP on ImageNet, not unexpectedly [Bartunov et al., 2018] demonstrated that SDTP and all its tested variants fail to train architectures on ImageNet.

[Meulemans et al., 2020] introduces an additional variant of DTP, the Direct Different Target Propagation (DDTP). In DDTP the network has direct feedback mapping functions  $g'_l(\hat{h}_L)$  from the output to each hidden layer. Such a variant systematically outperforms the accuracy of the original DTP algorithm on MNIST, Frozen-MNIST, Fashion-MNIST, and CIFAR-10.

**Direct Random Target Propagation (DRTP)** The DRTP algorithm aims to simultaneously solve the weight transport and the update locking problem. It relies on a similar strategy as DFA of sending the modulatory signal directly at each hidden layer, with the main difference being that the modulatory signals are projected target vectors instead of project error vectors as in DFA. Figure 3.1g provides the schematics of the rule. The underlying principle of DRTP is that the target vectors contain information about the error sign, and this is sufficient to provide useful modulatory signals for learning. Indeed, the sign of the  $c$ -th elements of the error vector associated to a target  $c^*$  and an output response  $y \in [0, 1]$  can be computed as:

$$e_c = \begin{cases} \text{sign}(1 - y_c) = 1 & \text{if } c = c^* \\ \text{sign}(-y_c) = -1 & \text{otherwise} \end{cases} \quad (3.32)$$

In DRTP, considering the same architecture introduced for BP and FA, the modulatory signals for the hidden layers are computed as:

$$\delta a_2 = (B_2 y^*) \odot f'(a_2) \quad (3.33)$$

$$\delta a_1 = (B_1 y^*) \odot f'(a_1) \quad (3.34)$$

$$(3.35)$$

where  $y^*$  is the target.

The advantages provided by this scheme are twofold. As in DFA, there is no dedicated feedback pathway, thereby alleviating the weight symmetry requirement. Second, the update locking issue is solved, as the updates can be performed at each layer as soon as the forward network activations are computed for the current layer. Therefore, the weight updates do not need to wait for the computation of the error at the end of the forward pass. Compared to Target Propagation, DRTP shares the idea of propagating target information, however, DRTP projects the target directly to each hidden layer, while TP propagates the target information in a backward fashion. When applied to image classification tasks, the DRTP update strategy enables learning in multi-layer networks, as the modulatory signals based only on the target (and thus on the error sign) are within  $90^\circ$  of those prescribed by BP. However, disregarding information on the error - and on the class-dependent magnitude - carries a major drawback, which is a substantially lower accuracy compared to other biologically inspired algorithms including FA. Such a gap cannot be narrowed even when doubling the layersize nor when increasing the number of hidden layers from 1 to 2. For instance, in all reported configurations, even when trained on the MNIST dataset, DRTP does not reach an accuracy higher than 96%.

**Global Error Vector Broadcasting (GEVB)** The global error-vector broadcasting (GEVB) [Clark et al., 2021] is a learning rule which generalizes three-factor Hebbian learning to train neural networks without weight transport and backward update locking. The algorithm relies on

broadcasting a single global learning signal to all hidden units and applying local Hebbian-like updates (Figure 3.1f). Such a method distributes information about the output error throughout the network. It projects the same identical error vector to all units, without unit-specific feedback parameters. The update prescribed by GEVB is given by:

$$\Delta W_l \propto -G(h_l) \sum_{\mu} a_{l-1}^{\mu} e^{\mu} \quad (3.36)$$

where  $G$  is a function that induces a coupling between different vector components, and  $\mu$  is the output error vector broadcasted to all hidden units. The equation above shows that presynaptic units that are aligned or anti-aligned with the output error vector have their weight onto the postsynaptic unit decreased or increased, respectively, when the postsynaptic unit is active. In this rule, the weight updates are shown to match the sign of the gradient on individual samples, leading to effective learning. The GEVB algorithm can be written as three-factor Hebbian learning with  $e^{\mu}$  being a vector-valued global third factor. Since the error is broadcasted directly from the output – which is a vector – to each unit, also the units need to be vectors. Therefore GEVB can only operate in a class of DNNs called vectorized nonnegative networks (VNNs). VNNs feature vector-valued units and nonnegative weights past the first layer (excitatory). The vectorial framework introduces additional computational and memory costs, as each unit needs to be represented by a vector rather than a scalar value. Such a framework becomes impractical in some cases, such as classification tasks with a large number of classes, *e.g.*, ImageNet, which would require each unit to be a 1000 dimensional vector. Furthermore, there is no experimental evidence that the vectorial representation of units is compatible with neurophysiology. On the other hand, constraining the network to nonnegative weights may be justified by the fact that cortical projections are excitatory [Kandel, 2000]. GEVB shares the principle of broadcasting a global learning signal with NP. The global signal could model the role of a neuromodulator in biological circuits that influences synaptic activity and updates. However, NP is intrinsically subject to high variance in gradient estimates. This issue is overcome by GEVB which does not rely on perturbations but rather on distributing identical information about the output error throughout the entire network. Computationally, the GEVB rule performs on par with BP on vectorized networks trained on MNIST and CIFAR-10. We remark, however, that the performance of BP on VNN is lower than in conventional models. Furthermore, GEVB is able to train convolutional architectures on MNIST and CIFAR-10, overcoming the limitation of DFA. Additionally, the alignment angles between the weight updates of GEVB and the gradients are smaller than those between the weight updates of DFA and the gradients, indicating that such a difference might drive the gap in performance.

**Deep Feedback Control (DFC)** One of the drawbacks of some of the biologically inspired solutions to credit assignment is a lack of connection with a mathematical optimization method. To address this limitation, Meulemans and colleagues [Meulemans et al., 2021] design the Deep Feedback Control (DFC), a learning rule for supervised learning which combines an approximation of Gauss-Newton optimization with dynamical system theory. DFC uses a feedback controller to drive a deep neural network to match the desired output target. The control signal of the feedback controller is used for credit assignment: the dynamic change in the neuron activations is used to update the synaptic weights. In terms of biological plausibility, the parameter updates are fully local in space and time and they could be implemented in a multi-compartmental model of cortical pyramidal neurons with a local voltage-dependent synaptic plasticity rule. More specifically, the dynamics of each neuron follows

$$\tau_v \frac{d}{dt} v_i(t) = -v_i(t) + W_i f(v_{i-1}(t)) + B_i u(t), \quad (3.37)$$

with

- $i \in [1, L]$  is the layer index
- $v_i$  contains the pre-nonlinearity activations of the neurons in layer  $i$

- $f(\cdot)$  is a smooth nonlinearity
- $u$  is a feedback input
- $W_i$  and  $B_i$  are the forward and the feedback matrices respectively
- $\tau_v$  is a time constant
- $r_i = f(v_i)$  are the post-nonlinearity activations of layer  $i$ .
- if  $u = 0$ , *i.e.*, in absence of input, the dynamics fall back to that of a conventional multilayer feedforward network.

During learning, a feedback controller produces the feedback signal  $u(t)$  to continuously drive the network's output  $r_L(t)$  towards an output target  $r_L^*$ :

$$r_L^* = r_L^- - \eta \frac{\partial \mathcal{L}(r_L, y)}{\partial r_L} \Big|_{r_L=r_L^-} = r_L^- + \delta_L, \quad (3.38)$$

where

- $\mathcal{L}(r_L, y)$  is a supervised loss defining the task
- $\delta_L$  is a shorthand notation

The control error  $e(t) = r_L^* - r_L(t)$  is used to control the dynamics as:

$$u(t) = K_I u^{int}(t) + K_P e(t) \quad (3.39)$$

and

$$\tau_u \frac{d}{dt} u^{int}(t) = e(t) - \alpha u^{int}(t) \quad (3.40)$$

where the control matrices are  $K_I = I$  and  $K_P = k_p I$ , with  $k_p \geq 0$ . The output controller maps directly to each hidden layer. The dynamics of the controller and of the network can be described as implementing a *dynamic inversion* of the output target  $r_L^*$  towards the hidden layers. Given such dynamics, the learning rule relies on comparing the neuron's controlled activation to its current feedforward input. In practice, the update of the feedforward weights can be implemented as:

$$\tau_W \frac{d}{dt} W_i(t) = (f(v_i(t)) - f(W_i r_{i-1}(t))) r_{i-1}(t)^T, \quad (3.41)$$

where  $\tau_W$  is a time constant longer than the time constants for the network and the controller, so that the weight plasticity does not influence the network and controller dynamics. This can be interpreted as a delta rule using the difference between the actual firing rate of the neuron  $f(v_i)$  and its estimated firing rate  $f(v_i^{ff})$  based on the feedforward inputs. The feedback weights are trained through alternating with updates of the forward weights, similarly to the wake and sleep phases in the WM algorithm [Akrouf et al., 2019]. The updates of the  $B_i$  matrices rely on zero-mean noise  $\epsilon$  injection:

$$\tau_v \frac{d}{dt} v_i(t) = -v_i(t) + W_i f(v_{i-1}(t)) + B_i u(t) + \sigma \epsilon_i \quad (3.42)$$

Consequently, we updates of the feedback matrix can be computed as:

$$\tau_B \frac{d}{dt} B_i(t) = -v_i^{fb}(t) u(t)^T - \beta B_i, \quad (3.43)$$

where  $\beta$  is the scale factor of the weight decay term. In simulations, DFC is tested on the MNIST and Fashion MNIST tasks, where it performs close to BP, slightly outperforming DFA. On the MNIST autoencoder, DFC also performs slightly worse than BP, while DFA has a poorer performance.



**Least Control Principle (LCP)** A second approach to credit assignment relying on a controller is the Least Control Principle (LCP) algorithm [Meulemans et al., 2022]. Inspired by the DFC scheme for feedforward networks, the LCP rule extends the control-based learning principle to recurrent systems. It consists of a temporally- and spatially-local rule which at first relies on an optimal controller to push the free dynamics of the system towards a solution state defined as a controlled equilibrium state that minimizes a loss  $L$ . Then the parameters are trained to reduce the amount of control needed to reach such a minimum loss state. The requirement for the model is that the dynamics need to be at equilibrium.

The LCP can be applied to learn an equilibrium RNN with one fully-connected recurrent layer in the context of supervised learning. The output controller is projected onto the recurrent neurons with direct linear feedback weights which are constrained to satisfy a column space condition. This scheme is a generalization of the DFC algorithm [Meulemans et al., 2021] to equilibrium RNNs. Both on the MNIST and on CIFAR-10 tasks, LCP achieves an accuracy only slightly lower than recurrent BP. Furthermore, LCP can also be applied to meta-learning, a framework that aims at improving a learning algorithm by leveraging shared structure in encountered tasks.

**Layer-wise local supervised learning** An approach to avoid the propagation of a global error is to train the network with layer-wise loss functions [Nøkland and Eidnes, 2019]. In the method proposed in [Nøkland and Eidnes, 2019], the synaptic weights are updated through a local error signal. Such signal is computed at each hidden layer through two separate single-layer sub-networks, one trained with the similarity matching loss (*sim* loss) and the other with the prediction loss (*pred* loss). The *sim* loss measures the L2 distance between matrices, where the elements contain the pair-wise similarities between examples in a mini-batch. The *pred* loss measures the cross-entropy between a prediction from a local classifier and the target. The *sim* and the *pred* loss can be combined in the *predsim* loss. The local algorithm is tested on MNIST, Fashion-MNIST, Kuzushiji-MNIST, CIFAR-10, CIFAR-100, STL-10 and SVHN, and is applied to both standard convolutional (VGG-like) and fully connected network architectures. In all settings, the proposed approach achieves an accuracy comparable to - and, in the case of the *predsim* loss, higher than - that of BP. However, the convergence rate is lower than that of BP.

The layer-wise local supervised learning demonstrates that error-driven learning does not require global error transportation nor backward propagation, and that each hidden layer can be trained independently from the downstream layers. The error locality approach alleviates the backward-locking problem and the weight transport problem, and is fully local. Since gradients and activations do not need to be stored, the memory requirement is lower than for BP. Furthermore, the layers can be trained one at a time, further reducing the memory cost. However, the loss at each layer is generated through a subnetwork trained with BP, therefore at the layer level this strategy suffers from the weight transport problem. Moreover, this strategy is more computationally costly than BP.

### 3.3.3 Self supervised learning

Supervised learning has proven successful in training models on a broad range of tasks. However, it presents two crucial limitations. The first aspect is practical: annotating large datasets is expensive and time-consuming as it requires a significant amount of human effort. The second limitation is related to biological plausibility, as learning in the animal brain is thought to be mostly unsupervised [Lillicrap et al., 2020]. Self-supervised learning is a powerful strategy to learn representation from neural data that addresses both limitations. First, self-supervised methods create pseudo-labels as supervision, allowing to use unlabeled datasets for training. Collecting unlabeled visual data is relatively easier and inexpensive for example thanks to the availability of large internet databases such as YouTube. Second, learning without labels has also been suggested as one possible means of building visual representation in the biological brain [Wang et al., 2021]. In the context of self-supervised learning, a popular approach is contrastive learning, which aims at grouping together the representation of similar samples and pushing apart the representation of different samples. Similar samples are generated by applying different distortions to the same

input sample. In the case of images, effective manipulations are random crop, color transformation, geometric transformation, and rotation. Different samples can simply be different input samples. Similar samples and different samples are referred to as *positive pairs* and *negative pairs*, respectively [Jaiswal et al., 2020]. Here, we review a recent approach that combines principles of contrastive learning with Hebbian plasticity rules to learn hierarchical representations in a local and biologically plausible fashion.

**Contrastive, Local And Predictive Plasticity (CLAPP)** A salient feature of the mammal visual ventral stream is the hierarchical representation of features. Higher processing areas extract higher-level information from the visual signal than lower areas. Most of the credit assignment solutions proposed as an alternative to BP (e.g., FA) can only lead to convergence in simple models [Bartunov et al., 2018]. Some other approaches, such as DFA, can train deeper networks, but with lower performance than shallow architectures [Nokland, 2016]. Illing and colleagues [Illing et al., 2021] propose a local plasticity rule that learns deep representations without BP, thereby addressing the reported issue. The *Contrastive, Local And Predictive Plasticity* (CLAPP) algorithm is a local and biologically plausible (Hebbian) learning rule that relies on self-supervised learning from temporal data. The temporal structure of the data is taken into account by simulating self-generated changes of gaze direction, known as *saccades*. Saccades distinguish input from a moving object during fixation from input arriving after a saccade towards a new object. The presence or absence of the saccades is taken into account in the learning dynamics through the introduction of a global factor that modulates plasticity.

CLAPP relies on the minimization of a layer-wise contrastive predictive loss function to learn hierarchical representations. Such representation can then be fed to an encoder trained with BP to associate the extracted features with labels. To describe the learning rule, we use the same notation as for the algorithms above. In addition to the general feedforward framework, CLAPP introduces feedback and lateral connections. At each time step, pyramidal neurons integrate input activity at the basal dendrites (matrix  $W^l$  of feedforward weights) and pass on their response to downstream areas ( $W^{l+1}$ ). At any point in time, the dendritic activity of the neurons predicts future neural activity through recurrent connections  $W^{pred}$ . An encoder network produces a *context* representation  $c^t$  at time  $t$ . Given the context, the encoder tries to predict the encoding of the future input  $h^{t+\delta t}$ . Algorithmically, each neuron  $j$  in a layer  $l$  may receive dendritic inputs  $(W^{pred}c^{t,l})_j$  coming either from the layer above ( $c^{t,l} = h^{t,l+1}$ ) or from lateral connections in the same layer ( $c^{t,l} = h^{t,l}$ ). For algorithmic reasons, the dendritic input  $(W^{pred}c^{t,l})_j$  influences the weight updates  $\Delta W_{ji}$  of the post-synaptic neuron  $j$ , but not its activity  $h_j^t$ . In this framework, the weight update is proportional to the following factors:

$$\Delta W_{i,j} \propto \text{modulators} \cdot (W^{pred}c^{t_1})_j \cdot \text{post}_j^{t_2} \cdot \text{pre}_i^{t_2} \quad (3.44)$$

The modulating broadcast factors are the same for large groups of neurons, for example all neurons in the same area, or even all neurons in the whole network.  $\text{post}_j^{t_2}$  and  $\text{pre}_i^{t_2}$  are functions of the pre- and post-synaptic activities, thus local.

The CLAPP algorithm relies on prediction across temporal information. Specifically, the prediction is modeled by the dendritic input term  $(W^{pred}c_j^t)$ . The prediction is meaningful during fixation, i.e., slightly different views of the same image, but is not reliable after a gaze change, i.e., a view of another image. In order to distinguish between the two cases, CLAPP relies on a score function  $u^t$  to classify whether the current sample in the temporal sequence is a different view of the previous sample (absence of saccade) or a new image (presence of saccade). The loss employed to learn the binary classification problem is given by the variable  $s^t$  which is  $s^t = +1$  for fixation and  $s^t = -1$  for saccade. The CLAPP loss - related to the Hinge loss - is therefore defined as:

$$\mathcal{L}_{CLAPP}^{t,l} = \max(0, 1 - s^t \cdot u_t^{t+\delta t,l}) \quad (3.45)$$

The information on the presence or absence of a saccade is transmitted as a broadcast signal, which gives the *contrastive* information, i.e., distinguished between *positive and negative pairs*. In the case the classification is correct ( $u_t^{t+\delta t,l} > +1$  if fixation or  $u_t^{t+\delta t,l} < -1$  if saccade), the gradient vanishes.

The gradients with respect to the feedforward weights are computed as:

$$\frac{\partial \mathcal{L}_{CLAPP}^t}{\partial W_{ji}} = \pm (W^{pred} c^t)_j f'(a_j^{t+\delta t}) x_i^{t+\delta t}. \quad (3.46)$$

The gradients with respect to the predicting layer weights are computed as:

$$\frac{\partial \mathcal{L}_{CLAPP}^t}{\partial W_{km}^c} = \pm (W^{predT} h^{t+\delta t})_k f'(a_k^{c,t}) x_m^{c,t}, \quad (3.47)$$

where the sign is negative (resp. positive) in the absence (resp. presence) of saccades. The weights are updated via gradient descent with learning rate  $\eta$ :

$$\Delta W_{ji}^t = -\eta \frac{\partial \mathcal{L}_{CLAPP}^{t-\delta t}}{\partial W_{ji}} \quad (3.48)$$

By introducing a modulating factor  $\gamma_t = y^t \cdot H^t$ , where  $H^t \in \{0, \eta\}$  is a layer-wide broadcast signal that indicates if the saccade or fixation was correctly detected, the forward weight updates can be formulated as:

$$\Delta W_{ji} = \gamma_t \cdot (W^{pred} c^{t-\delta t})_j f'(a_j^t) x_i^t. \quad (3.49)$$

The updates of the prediction layer weights require the introduction of an additional matrix, the retrodiction matrix  $W^{retro}$ , which is necessary to model the transmission of information in the direction of emitting the prediction instead of receiving it. The resulting update is:

$$\Delta W_{km}^{c,t} = \gamma_t \cdot (W^{retro} h^t)_k f'(a_k^{c,t-\delta t}) x_m^{c,t-\delta t}. \quad (3.50)$$

Finally, the plasticity of the prediction and retrodiction weights is governed by the Hebbian learning rule:

$$\Delta W_{jk}^{pred} = \Delta W_{kj}^{retro} = \gamma_t \cdot h_j^t \cdot c_k^{t-\delta t}, \quad (3.51)$$

where neuron  $k$  in the predicting layer  $c$  is pre-synaptic in  $W^{pred}$  or post-synaptic in  $W^{retro}$  and neuron  $j$  in the predicted layer  $z$  is post-synaptic in  $W^{pred}$  or pre-synaptic in  $W^{retro}$ .

CLAPP is evaluated experimentally on the STL-10 dataset of image classification. Each image is cropped into  $16 \times 16$  patches. The patches are viewed one after the other in a vertical order. A 6 layer VGG-like network is trained using the CLAPP rule. After training, the layers have learnt features that are selective for orientation and colors in the bottom layers and that are more semantic and object-selective in deeper layers. This is corroborated by the accuracy of read-out layers that are trained on the features extracted by each layer: the encoder accuracy increases significantly from hidden layer 1 to hidden layer 4, and then saturates and only slightly improves for layers 5 and 6.

Algorithmically, CLAPP solves the locality, the freezing of layers and the weight transport issue. Furthermore, it enables hierarchical feature learning. Biologically, the global factor in the learning rule can be related to global, saccade-specific signals from motor areas, combined with surprise or prediction error. Additionally, neurophysiological evidence supports the fact that the visual pathway is ‘self-aware’ of saccades due to saccade-specific modulation of processing. Finally, CLAPP is able to exploit the temporal structure of the data, which is likely how the brain processes information.

### 3.4 Addressing energy consumption: spiking neural networks

In parallel to the advances of ANNs, the neuroscientific community has studied network architectures with more biologically realistic dynamics. The use of sparse asynchronous voltage pulses, called spikes, to compute and propagate information, in conjunction with the concept of spiking

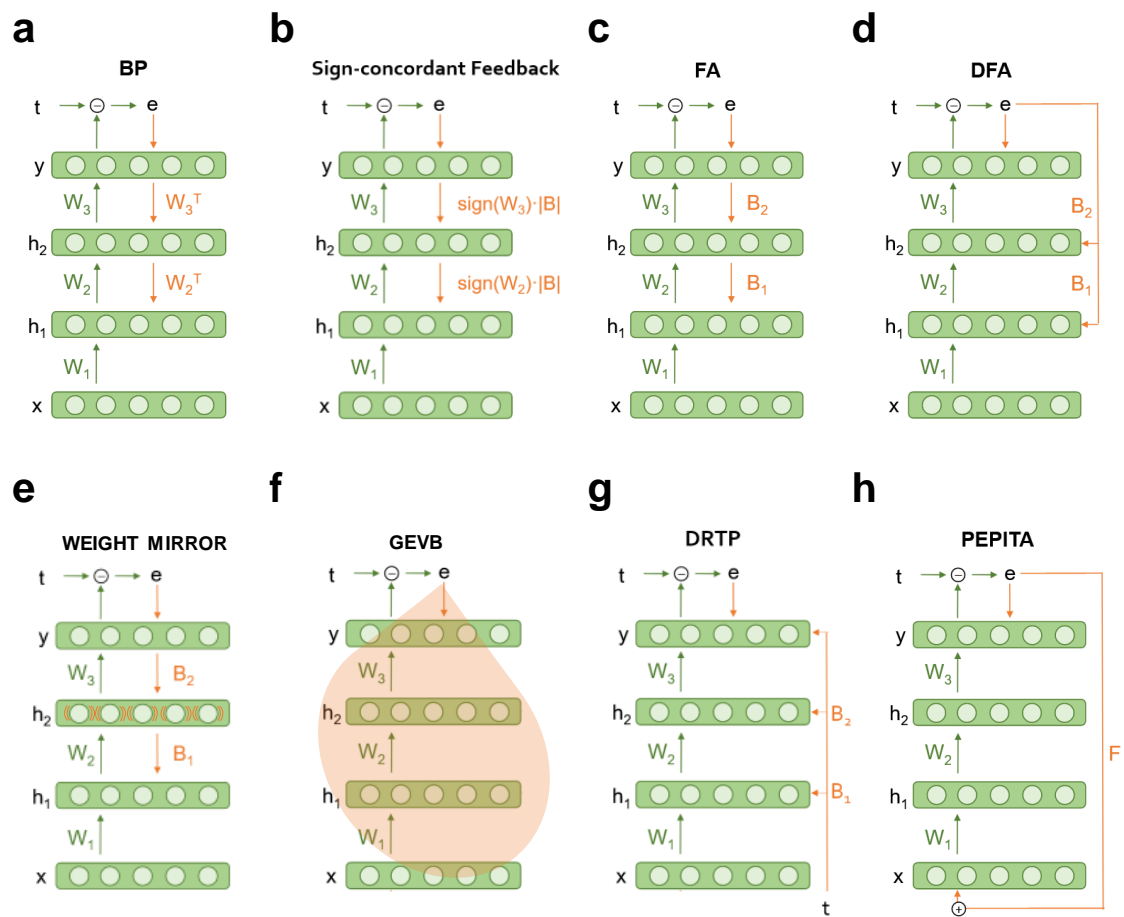


Figure 3.1: Schematics of (a) BP and alternative training schemes: (b) sign concordant feedback, (c) feedback alignment, (d) direct feedback alignment, (e) weight mirroring, (f) global error vector propagation, (g) direct random target projection, and (h) present the error to perturb the input to modulate activity.

leaky integrate-and-fire (LIF) neurons, led to the development of the Spiking Neural Networks (SNNs) paradigm. SNNs are bio-inspired Artificial Neural Networks that aim at closely mimicking the neuronal mechanisms to efficiently perform tasks such as pattern recognition and clustering. Since SNNs are based on core properties of biological networks, I will first review the fundamental processes occurring in the brain as a basis to thoroughly understand the computational design and choices of the described models.

### 3.4.1 The potential in the efficiency of SNNs compared to ANNs

The spike-based information propagation in the biological brain allows for energy-efficient communication of analog signals across noisy channels. SNNs inherit properties of power efficiency from the brain by mimicking event-based signal transmission properties. The advantages in terms of the efficiency of SNNs stem from three main aspects.

1. First, SNNs rely on parallel sparse computing, unlike ANNs which rely on parallel dense computing. Brain computations rely on sparsity to remove redundancy and non-relevant information. Mimicking this feature through the sparse nature of spike-based computations in SNNs is key to saving computational power.
2. Second, SNNs exploit asynchronous event-based spikes, which allow spending energy only when a spike occurs. The neuronal output is binary, where 0 indicates subthreshold membrane potential and 1 indicates a spike. On the contrary, ANNs work with synchronous clocking, forcing to spend power for every node at every step.
3. Third, SNNs can learn on the fly through local learning rules, which have the potential to be more robust and able to quickly adapt than the offline training using labeled datasets typically adopted in ANNs training.

### 3.4.2 Neuronal dynamics and elements of neuronal systems

Neurons are the elementary processing units of the central nervous system. They propagate the information in the form of electrical pulses from one another through connections called synapses. Neurons can be grouped into several subtypes, such as spiking neurons. The neuronal structure consists of several parts: the cell body, called *soma*, which contains the cell nucleus; the *dendrites*, which extend away from the soma to receive signals from other neurons; the *axon*, which propagates the integrated signal from the other cells to the axon's terminal to propagate the signal to other cells.

In spiking networks, the information is propagated from a sending neuron, the *presynaptic neuron*, to a receiving neuron, the *postsynaptic neuron*, in the form of short electrical pulses. These neuronal signals are referred to as *action potentials* (AP) or *spikes*. An action potential is a binary signal that does not have gradient information. A sequence of APs gives rise to the so-called *spike trains*, which encode the information in the timing and number of spike events. The refractoriness period is the smallest time interval between two consecutive APs in the same neuron. The spike train of a neuron  $i$  which fires at times  $t_i^{(f)}$  can be written as:

$$S_i(t) = \sum_f \delta(t - t_i^{(f)}) \quad (3.52)$$

where  $\delta(t - t_i^{(f)})$  is the Dirac function, with  $\delta(t - t_i^{(f)}) = 0$  for  $t \neq t_i^{(f)} \forall i, f$ , and  $\int_{-\infty}^{\infty} \delta(t - t_i^{(f)}) dt = 1$

**Neuronal dynamics** The ability of neurons to produce electrical signals derives from the processes occurring at the level of the membrane separating each cell from the surrounding environment. The difference in voltage between the interior and the exterior of the cell is the *membrane potential*  $V_{mem,i}(t)$ , where  $i$  labels the index of the cell, and where the temporal evolution is highlighted by  $(t)$ . If no input is received by the cell, the membrane potential remains constant at the

*rest potential*  $V_{mem,i}(t) = V_{rest}$ . The resting membrane potential of a neuron is approximately -70 mV. Upon the arrival of a spike from a presynaptic neuron, the membrane voltage can either increase (*depolarization*) or decrease (*hyperpolarization*), and then it decays back to the resting value. The response of the membrane potential is known as *postsynaptic potential* (PSP). In the former case, the synapse is said to be excitatory and its value, the synaptic weight, is positive, while in the latter case the synapse is inhibitory and its weight is negative. When a spike train of spikes  $f$  stimulates a post-synaptic neuron  $i$ , the PSPs of the single spikes are summed up linearly in the membrane dynamics. Furthermore, if the incoming spikes are generated by different presynaptic neurons  $j$ , the responses add up linearly as well. The overall variation in the membrane potential of the postsynaptic neuron  $i$  can therefore be written as:

$$V_{mem,i}(t) = \sum_j \sum_f \epsilon_{ij}(t - t_j^{(f)}) + V_{rest}. \quad (3.53)$$

When enough excitatory responses are evoked, the membrane potential increases until it crosses a critical value, namely the *voltage threshold*  $V_{thr}$ . The moment at which the voltage reaches the threshold from below is called *firing time*  $t_i^{(f)}$ :  $t_i^{(f)} = \{t | V_{mem,i}(t) = V_{thr}\}$ . At the firing time, an AP is elicited. The shape of the AP is always the same and stays constant during propagation through synapses, thus it does not contain information on the signal. Therefore, the input is encoded by the presence or absence of spikes, their timing, and their frequency.

### 3.4.3 Leaky integrate and fire neuron models

Many mathematical approaches have been developed to describe neuronal activity. The dynamics of the membrane potential can be described as a summation process, also called an *integration* process. Neuron models that are based on this behavior are known as “Integrate-and-Fire”. One commonly used neuron model is the “Leaky Integrate-and-Fire” (LIF) model, whose dynamics are described by:

- a linear differential equation to describe the evolution of the membrane potential, known as *equation of the passive membrane*

$$\tau_m \frac{dV_{mem}}{dt} = -[V_{mem,i}(t) - V_{rest}] + RI(t), \quad (3.54)$$

where  $I(t)$  is an arbitrary input current and  $\tau_m$  is the characteristic time of the decay, computed as the product between the membrane resistance  $R$  and capacitance  $C$ :  $\tau_m = RC$ ;

- a firing threshold  $V_{thr}$ ;
- a reset condition  $\lim_{\delta \rightarrow 0; \delta > 0} u(t^f + \delta) = u_r$ .

Figure 3.2a shows a diagram comparing the structure of a biological neuron with its circuit representation. Figure 3.2b shows the schematics of LIF neuron dynamics. The neuron integrates the incoming spikes from presynaptic nodes, which is reflected by variations in the voltage membrane  $V_{mem}$ .

The LIF neuron model exhibits very simplified dynamics, however, it proves to be highly accurate to simulate the spike generation in neurons. Furthermore, it can be used as a basic building unit for the construction of large populations of neurons. This allows investigation of the collective properties of neurons, by designing and simulating populations interacting with each other. The focus can be brought from the properties of the single neurons to the response of a whole neuronal population to a sensory stimulus (e.g. visual, auditory, tactile, etc.).

SNNs and temporal coding to process information can be used to perform multiple tasks, such as pattern clustering and classification. Depending on the nature of the task the training is carried out through supervised or unsupervised learning rules.

The learning of most networks designed for pattern clustering is based on the classical Hebbian rule of spike-timing-dependent-plasticity (STDP). STDP rule states that the resulting change in the synaptic efficacy  $\Delta w_{ij}$ , connecting a presynaptic neuron  $j$  spiking at  $t_j^{(f)}$  to a postsynaptic neuron  $i$  spiking at time  $t_i^{(f)}$ , is a function of the spike time differences ( $t_i^{(f)} - t_j^{(f)}$ ). If the presynaptic neuron fires shortly before the postsynaptic neuron the connection is reinforced, otherwise, it is depressed. The smaller the difference in the spike times, the greater the amplitude of the synaptic change [Gerstner and M Kistler, 2002].

### 3.4.4 Spike encoding

Temporal spike pattern encoding refers to the process of mapping a sensory input, consisting of multidimensional raw data, into a temporal sequence representation, that of the spike trains. Once the input is in the form of spikes, it can be processed and propagated by the spiking neurons in the network. Several encoding techniques have been developed.

**Temporal code or Time-to-first spike encoding** The time-to-first-spike coding scheme encodes the information in the response latency of the spike-time of neurons. For visual input encoding, the input to the network is directly the image. For audio signal encoding, the audio stimulus is initially transformed into a spectrogram that describes the power spectral density of the sound signal over time and frequency axes. The number of input neurons is the same as the number of pixels in the image or spectrogram. Each neuron spikes only once and its spike-time is computed based on the intensity of the associated pixel: the greater the intensity, the sooner the neuron fires. The advantage of this method resides in power efficiency due to a fixed and small number of spikes.

**Rate code or Poisson encoding** Poisson encoding is a commonly used technique to convert a sound signal into spikes ([Hazan et al., 2018], [Diehl and Cook, 2015]). Employing a Poisson process allows simulating spike trains that reflect features close to those generated by real neurons. As for the time-to-first spike method, the number of input Poisson neurons is equal to the number of pixels of the image or spectrogram. Each input neuron is set to have an average firing rate  $\lambda$  proportional to the intensity of the pixel it is associated with. As a consequence, the information is encoded in the firing rates, while the information contained in the temporal structure is neglected. Due to the high number of spikes generated, Poisson encoding is more time-consuming and energy inefficient than time-to-first spike encoding [Dong et al., 2018b].

Spiking networks and Poisson encoding in this thesis: Chapter 7.

### 3.4.5 Power efficiency of spiking neural networks

Neuromorphic engineering has taken on the challenge of approaching the energy efficiency of the mammal brain by building scalable, low-power systems that mirror the brain's essential architectural features [Zenke and Neftci, 2021]. Information processing and memory elements are physically separated in standard von Neumann computing architectures. Therefore, von Neumann machines are not well suited for simulating parallel neural networks, they are incompatible with the locality principle, and they require a large amount of power. Instead, in-memory computing architectures typically comprise parallel and distributed arrays of synapses and neurons that can perform computation using only local variables, thereby leading to extremely low-energy consumption figures. The spike-based learning circuits implemented in analog neuromorphic circuits exploit the precise timing of spikes and consequently take advantage of the high temporal resolutions of event-based sensors. Furthermore, the sparsity characterizing the spike patterns produced by neuromorphic sensors and processors leads to even higher gains in terms of energy efficiency [Khacef et al., 2022].

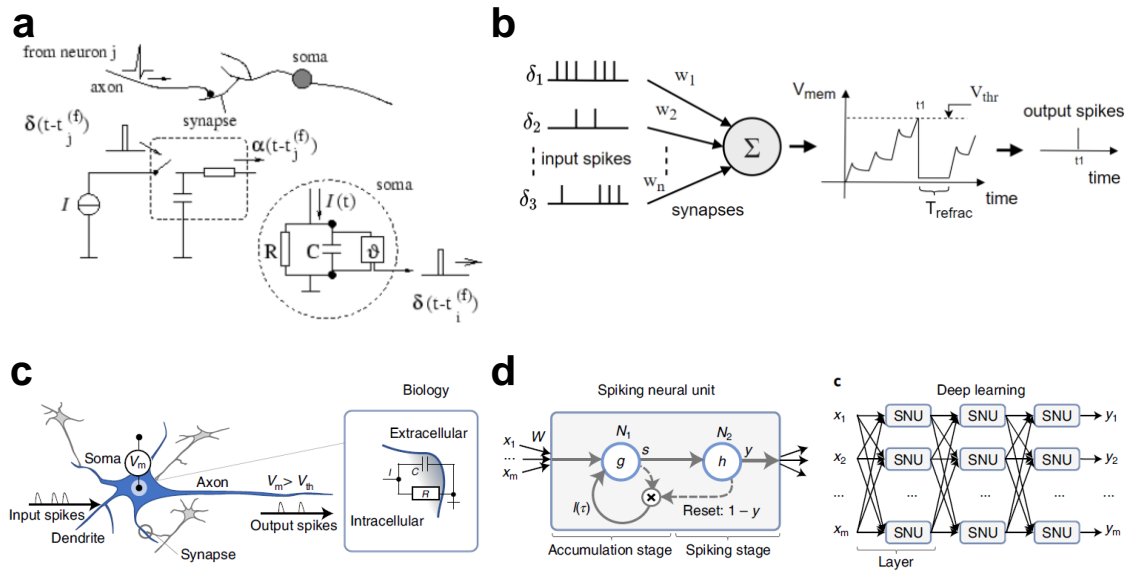


Figure 3.2: (a) Reproduced from [Gerstner and Kistler, 2002] (b) Reproduced from [Corradi et al., 2021] (c) Reproduced from [Wozniak et al., 2020].

### 3.4.6 The spiking neural unit

The Spiking Neural Unit (SNU)[Wozniak et al., 2020] approach proposes an alternative model to conventional SNNs to incorporate neuronal dynamics into a neural network. The SNU paradigm unifies SNNs with recurrent ANNs by abstracting the dynamics of a LIF spiking neuron into a simple recurrent ANN unit. SNUs may operate as SNNs, with a step function activation, or as more conventional RNNs, with continuous activations. The non-spiking variant is called soft SNU (sSNU). The SNU architectures have proven to be successful in three applications: handwritten digit recognition, language modeling and polyphonic music prediction, where SNUs achieve competitive performance with that achieved by the state-of-the-art ANNs.

Figure 3.2c and 3.2d report a diagram of the biological neuron, its circuit representation, the underlying dynamics of a SNU and a feedforward architecture of SNU units.

Spiking neural units in this thesis: Chapter 5.



## Chapter 4

# Modeling the Mind

### 4.1 Digital twins of the brain

Deep neural networks modeling the circuits of the brain are an invaluable tool to help us understand the myriad complexities of human cognition and intelligence [O’Reilly, 2006]. The goal of the research linking deep learning and neuroscience is to design in-silico neural networks that are able to fit and predict the empirical data, from the neuronal level to the behavior in the primate and rodent brains. AI architectures have been shown to be able to accurately model the activity in some areas of the brain. When mapped to the pair stimuli-responses of biological networks, these models have made novel predictions about neural phenomena or shed light on the fundamental functions being optimized [Schaeffer et al., 2022]. Some milestone works in this direction have shown the possibility of predicting, and in some cases even guiding, the neural responses in brain areas such as the posterior parietal cortex [Zipser and Andersen, 1988], in vestibulo-ocular reflexes [Anastasio and Robinson, 1989], in the primary motor cortex [Lillicrap and Scott, 2013], in rodents’ V1 [Walker et al., 2019] and in macaque V4 and IT [Bashivan et al., 2019, Yamins et al., 2014, Ponce et al., 2019]. Compared to previous models, deep networks are trained on quasi-naturalistic sensory signals, such as image pixels or auditory spectrograms, rather than on their hand-labeled input and output units.

These “models of the brain” are defined as successful when their representations, after training on a specific optimization problem, match the experimental observations. In this case, the models may reveal which optimization problem the brain is solving. Furthermore, when proved successful in matching neural responses, these models can potentially be employed to generate novel predictions on brain responses and related behavior [Schaeffer et al., 2022]. In this thesis, we focus on deep learning models of the brain applied to the visual areas, described more in-depth in section 4.1.2 and in chapter 8.

In a review, Saxe and colleagues reflect on how neuroscientists use deep networks to model and understand biological brains [Saxe et al., 2020]. They claim that deep networks may lead to the development of theories of perception, cognition and action for biological brains, thereby introducing a new perspective toward understanding neural systems.

#### 4.1.1 The encoding and the decoding problems

When building computational models of vision, neuroscientists tackle two fundamental questions: “how is the visual input represented in the brain?” and, “how is this representation used by the brain to carry out behavioral tasks?”

**The encoding problem** The first question is referred to as the *encoding problem*. Understanding how the visual input is represented in the brain involves designing a deep learning model that is able to take the same input as the brain and predict the neural responses of the region of interest.

Four elements are key in designing the algorithm to model the activity of a brain area as a response to a stimulus. First, the choice of the animal we want to model. The final goal, for example for therapeutic applications, would be building a model of the human brain. However, as the possibility of performing invasive recording with humans is limited, the measurements are often performed on monkeys and the findings are translated to humans, for instance by guiding the design of psychophysics experiments. Second, the choice of the brain area. One of the most studied areas in the context of substitute models is the visual stream. The substitute models map the pixels of a visual stimulus to the neural activity response. The pixels are given as input to a network. The neuronal responses are mapped, for example through linear regression methods, to a specific layer of the model. The layer is generally chosen through a grid-search operation as the layer that best explains the neural activity on a set of held out images from the training and fitting operations. The choice of the area and its specific function under analysis dictates the type of experiment to be performed. For example, the animal can be head-restrained and passively view images for free viewing tasks, or it can be trained to perform a task, such as choosing an image in a set of images based on its semantic content. The third important choice is related to the model architecture. Common choices include convolutional models such as the AlexNet [Bashivan et al., 2019] and ResNet architecture [Yuan et al., 2020], however also transformer models have proven suitable to the task [Yuan et al., 2020].

**The decoding problem** The question ‘how is this representation used by the brain to carry out behavioral tasks?’ is related to the so-called decoding problem, which involves linking the neural responses to the consequent behavior. Neural responses can be used to explain core object recognition, for example by training a model to infer which object category was shown to the animal based on the recorded activity in IT. In a free viewing experiment, a monkey was let to observe passively a sequence of images while IT neuronal activity was recorded through multielectrode arrays. The IT neural population representation, in terms of mean firing rates, can be used to explain and predict object recognition behavior. Based on the recording, a linear decoder can be successfully trained to classify the images into classes based on the neural population feature vector.

#### 4.1.2 Examples of ANNs to study biological vision

In the past few years, enormous progress has been made in developing computational models of visual processing aiming to capture underlying principles of the primate ventral visual stream. The introduction of these models allowed to gain a deeper understanding of the dynamics and properties of the visual cortex, paving the way for intriguing applications such as targeted control of cortical neural activity and the associated behavioral response. Deep artificial neural networks (ANNs), which mimic the approximately hierarchical structure of the ventral visual cortex [Kreiman, 2021a], currently offer the most accurate understanding of the behavior of the primate ventral visual system [Yamins et al., 2014, Cadena et al., 2019, Schrimpf et al., 2018]. Recent breakthroughs have demonstrated that not only ANNs are exceptionally good predictors of ventral stream neural activity, but also that they can be successfully used to control the firing rate of specific neurons in cortical areas of primates and rodents. Some original approaches proposed using ANN-driven image generation to synthesize visual stimuli able to drive the activity of specific neurons beyond their naturally occurring maximal firing rate. The first milestone in this direction is represented by the method introduced by Bashivan and colleagues [Bashivan et al., 2019] in which a variant of Alexnet architecture [Krizhevsky et al., 2012] is used to synthesize images that each are specifically targeted to control neural firing activity in cortical visual area V4 of macaques. The control is realized in two settings: in the ‘Neural Stretch’ scenario, the synthetic images activate specific neurons more strongly than any other naturalistic stimulus, while in the ‘Neural Population State Control’ the synthetic images achieve independent control of every neural site in a small recorded population. In a similar vein, Ponce and colleagues [Ponce et al., 2019] developed an ANN framework relying on a pre-trained generative deep neural network and on a genetic algorithm to create evolved images guided by neuronal firing in the primary inferior temporal cortex (IT) of macaques.

The proposed paradigm, termed XDREAM, yields the generation of ‘super stimuli’ images able to elicit responses higher than any natural image in V1, in IT cortex, in single-units, in multi-units, and in average population responses. Similar progress for mouse primary visual cortex (V1) was made by Walker and colleagues [Walker et al., 2019] with the development of ‘inception loops’, an end-to-end trained ANN-based model able to synthesize optimal stimuli driving neural responses more strongly than control images.

### 4.1.3 Brain-Score

The Brain-Score platform [Schrimpf et al., 2020, Schrimpf et al., 2018] is an open-source tool for researchers introduced in 2018 to evaluate the performance of neural networks as models of the brain, and in particular of the ventral stream. Through a set of metrics and a range of experiments, Brainscore assesses the image-level alignment of proposed models to primate behavior. Each model is evaluated through a common interface that overcomes the technical differences and ambiguities presented by the diverse candidate models in the machine learning and computer vision communities.

The platform formalizes and standardizes six experimental benchmarks used to evaluate models of the brain, including single neural predictions in particular visual areas to particular image sets, distributions of tuning functions in each visual area, and patterns of behavioral performance in particular visually guided tasks. Brain-Score is framed as a competition and reports a leaderboard of results of each participating project, with the motivation that a competitive benchmark may lead to the emergence of more accurate models.

### 4.1.4 Substitute models to drive perception

The described techniques represent pivotal steps in the direction of relying on ANN substitute models for the non-invasive manipulation of brain states at neuron-level resolution. However, while these methods achieve remarkable control of the activity of small neural populations, they do not explore the perceptual consequences of such control nor tailor the targeted neural responses to the desired behavior. In Chapter 8, we describe our research in which we build upon the described works and show how several approaches to image synthesis, including ANN-based methods, linear interpolation and CycleGAN-based interpolation, can be used to influence both neuronal responses and behavior in a binary decision task. In our work, we focus on the response of the well-characterized face-selective neurons in the macaque IT cortex [Moeller et al., 2008]. We develop an array of techniques to minimally perturb human or monkey face images such that they are differently classified by both primate and human observers.

Our approach generates deceptive images by modifying images from one category (such as human faces) to look like a target category (such as monkey faces), while limiting pixel value change. The notion of minimally perturbing an image to move a model decision boundary is anything but new in machine learning: CNNs have been found susceptible to adversarial attacks, that is the manipulation of images through carefully crafted, minute noise leading models to misclassify them with high confidence [Szegedy et al., 2013, Goodfellow et al., 2014b]. While our work pursues the same goal of adversarial attacks, our deceptive images depart substantially from the standard notion of adversarial images, in that the added perturbations significantly alter the original images. In contrast, adversarial noise tailored for CNNs is often imperceptible to humans [Szegedy et al., 2013], so adversarial images have sometimes been thought of as relative to ground truth labels assigned by humans. We remark, however, that there are indications that primate vision can be somewhat sensitive to adversarial images. Elsayed et al. [Elsayed et al., 2018] show that humans doing visual categorization under tight time constraints can be biased by adversarial images. Zhou et al. [Zhou and Firestone, 2019] show that humans can decipher the attack target in adversarial images crafted for CNNs (without necessarily making mistakes if asked to categorize the images). Berardino et al. [Berardino et al., 2017] identify ‘eigen-distortions,’ directions of small pixel value change that most readily make images look different.

Guo and colleagues [Guo et al., ] performed a study with the similar goal of developing a method

to perform adversarial visual attacks directly on primate brain activity. They compared the adversarial sensitivity of individual IT neural sites with individual units in state-of-the-art robust deep neural networks. Given a small image perturbation budget, the authors tweaked the images to cause the largest perturbation possible in the neuron activity. Their analysis represents a milestone work as it is the first study using the regime of image perturbation studied by robustness research in the computer vision and machine learning communities. The results demonstrate that adversarial examples exist for all IT neural sites and that they are dense in the image space, suggesting that visual object recognition behavior in primates may not be as robust as previously thought. These findings appear at odds with our result that perceivable image perturbations are needed to fool the primate perception. We need to consider carefully the difference between the two studies, in that Guo and colleagues analyze the robustness of individual cells, while in our work we consider the activity at the population level. Therefore, while the work in [Guo et al., ] shows that the single neuron activity is not adversarially robust, it does not make claims at the perceptual level.

Although our deceptive images do not fulfill the requirements of adversarial attack definition, our proposed approach represents important progress towards the development of image synthesis methods able to flip the semantic content perceived by the primate brain. Therefore, our work opens up new experimental paradigms to study neuronal processing and its link to behavior.

Furthermore, the proposed strategies [Yamins et al., 2014, Bashivan et al., 2019, Walker et al., 2019, Ponce et al., 2019] are powerful tools to predict and control neural activity from visual stimuli, but cannot be applied to predict activity in absence of visual stimulation. In preliminary unpublished work, I introduced a new research line in the framework of ANNs to understand biological vision, in which I propose methods and baseline strategies to predict neuronal activity in conditions of spontaneous activity, *i.e.*, when the animal is looking at a dark screen. I tested and compared different methods, including regression, LSTMs and Transformers, to predict neural activity in one brain area, from simultaneous activity in a connected brain area. I will expand on this research direction in the future.

## 4.2 Limitations of deep learning applied to neuroscience

### 4.2.1 No free lunch from deep learning in neuroscience

Despite the remarkable results obtained with the so-called digital twins of the brain, Schaeffer and colleagues warn about the fact that there is “No free lunch from Deep Learning in Neuroscience” [Schaeffer et al., 2022]. Deep learning models need to be used carefully to give insight into brain principles. The apparent success of some algorithms has been shown to derive from specific implementation choices, and we need to be cautious in distinguishing *predictive* properties from *post hoc* implementation and model tuning. In particular, deep learning models of grid cells in the entorhinal-hippocampal circuit are a critical case. Experimental observations revealed that the hippocampus contains place cells [O’Keefe and Dostrovsky, 1971, Rich et al., 2014], which fire at specific locations in the environment, and the medial entorhinal cortex contains grid cells [Hafting et al., 2005], which fire in a spatially periodic pattern in 2D environments. ANNs have been proposed to model biological circuits on *path integration*, which is the task of using self-velocity estimates to track one’s spatial position, a crucial component of spatial navigation [Cueva and Wei, 2018, Banino et al., 2018, Sorscher et al., 2019, Whittington and Bogacz, 2019, Nayebi et al., 2021]. These networks have been demonstrated to yield models accounting for the presence of grid cells. However, these findings should not be confused as predictive of the existence of grid cells, but rather as models designed purposely to obtain representation fitting the known properties of grid search. The search of models and parameters was biased toward favoring configurations shown to produce grid cell emergence, which are fragile solutions in the landscape of all possible models. Concluding, Schaeffer and colleagues claim that, if grid cells had not been discovered experimentally, it would have been highly unlikely that deep learning models predicted their emergence without inductive bias informing their design. Therefore, when building deep learning models of the brain, it is imperative to rigorously characterize conditions and constraints under which the particular tuning does and does not emerge, to consider whether the constraints

align with biological constraints.

An additional important point made in [Schaeffer et al., 2022] is that, even in the case where the representations produced by the models of the brain match the experimental observation, it is incorrect to infer that the brain solves the same optimization problem. Indeed, in principle, several distinct loss functions can have similar minima. The claim that the brain solves different optimization problems than the models is also supported by the fact that these models are not functionally identical to the ventral visual stream. For instance, a major difference is the lack of recurrent circuits in the model, which is instead a prominent feature of the brain.

### 4.2.2 A roadmap for neuroscience research

In the current formulation of the success of deep learning models, the networks are evaluated based on matching between their representation and the neural recording. However, even in the case in which the brain is solving the same optimization of its digital twin, how can this provide a better insight into its dynamics? As Saxe and colleagues [Saxe et al., 2020] ask, “If neural computation emerges uncontrollably through blind, unconstrained optimization, then how can neuroscientists formulate new, empirically testable hypotheses about neural mechanisms?”.

A possible answer to this open question is given by Marblestone and colleagues [Marblestone et al., 2016], who proposes three hypotheses: (i) the brain optimizes cost functions, (ii) the cost functions do not need to be global, rather they are diverse and differ across brain locations and over development, and (iii) optimization operates within a pre-structured architecture matched to the computational problems posed by behavior. The meaning of cost function in the brain is related to the ability of neurons to adjust the properties of their synapses, so that they get better at performing a task. The role of the synapses is defined as the cost function. The authors suggest that cost function optimization plays an important role in shaping the internal representations and processes used by the brain.

The focus on designing more accurate models of the brain is to get better results on benchmarks as quantifies for example on the Brain-Score platform. However, less attention is paid to the reasons why the models reach a certain performance. This comes together with the lack of guidance in the roadmap of gaining a more comprehensive understanding of neural computation through deep learning.

### 4.2.3 The brain as a highly social network

Furthermore, it is of foremost importance to remember that there are substantial conceptual differences between the brain and computers. O’Reilly [O’Reilly, 2006] describes this point through a metaphor:

*It is clear that the brain is much more like a social network than a digital computer, with learning, memory and processing all being performed locally through graded communication between interconnected neurons. These neurons build up strong, complex “relationships” over a long period of time; a neuron buried deep in the brain can only function by learning which of the other neurons it can trust to convey useful information. In contrast, a digital computer functions like the post office, routing arbitrary symbolic packages between passive memory structures through a centralized processing unit, without consideration for the contents of these packages. This affords arbitrary flexibility (any symbol is as good as any other), but at some cost: When everything is arbitrary, then it is difficult to encode the subtle and complex relationships present in our commonsense knowledge of the real world. In contrast, the highly social neural networks of the brain are great at keeping track of “who’s who and what’s what,” but they lack flexibility, treating a new symbol like a stranger crashing the party.*

The brain is an interconnected system which has been observed to be able to function when some areas are removed accidentally or on purpose. For example, the patient Phineas P. Gage survived an accident in which a large iron rod was driven completely through his head, destroying much of his brain’s damage involved both left and right prefrontal cortices. Despite the brain damage, Gage remained as able-bodied and intelligent as before the accident, without impairment

---

of movement, speech, learning or memory [Damasio et al., 1994]. This demonstrated that one could perform most of the cognitive functions without an intact brain. Another example is the removal of portions of the brain that originate epileptic seizures, which does not affect the cognitive functions of the patients. In contrast, if a part of a computer is removed, the machine is highly unlikely to function. Thus, the question naturally arises: “how can we design deep networks able to reproduce such observations of robustness and redundancy?”

Part 2

Results

## Chapter 5

# GRAPES: a Biologically Inspired Optimizer

Plasticity circuits in the brain are known to be influenced by the distribution of the synaptic weights through the mechanisms of synaptic integration and local regulation of synaptic strength. However, the complex interplay of stimulation-dependent plasticity with local learning signals is disregarded by most of the artificial neural network training algorithms devised so far. Here, we propose a novel biologically inspired optimizer for artificial and spiking neural networks that incorporates key principles of synaptic plasticity observed in cortical dendrites: GRAPES (Group Responsibility for Adjusting the Propagation of Error Signals). GRAPES implements a weight-distribution-dependent modulation of the error signal at each node of the network. We show that this biologically inspired mechanism leads to a substantial improvement of the performance of artificial and spiking networks with feedforward, convolutional, and recurrent architectures, it mitigates catastrophic forgetting, and it is optimally suited for dedicated hardware implementations. Overall, our work indicates that reconciling neurophysiology insights with machine intelligence is key to boosting the performance of neural networks.

### 5.1 Introduction

Artificial neural networks (ANNs) were first proposed in the 1940s as simplified computational models of the neural circuits of the mammalian brain [McCulloch and Pitts, 1943b]. With the advances in computing power [Thompson et al., 2020], ANNs drifted away from the neurobiological systems they were initially inspired from, and reoriented towards the development of computational techniques currently employed in a wide spectrum of applications. Among the variety of techniques proposed to train multi-layer neural networks, the backpropagation (BP) algorithm [Rumelhart et al., 1995, Schmidhuber, 2015] has proven to lead to an effective training scheme. Despite the impressive progress of machine intelligence, the gap between the potential of ANNs and the computational power of the brain remains to be narrowed. Fundamental issues of ANNs, such as long training time, catastrophic forgetting [French, 1999], and inability to exploit increasing network complexity [Ba and Caruana, 2014], need to be dealt with not only to approach the human brain capabilities, but also to improve the performance of daily used devices. For instance, reducing the training time of online learning in robotic applications is crucial to ensure a fast adaptation of the robotic agent to new contexts [Polydoros and Nalpantidis, 2017] and to reduce the energy costs associated with training. Several techniques, such as *batch normalization* [Ioffe and Szegedy, 2015], *layer normalization* [Ba et al., 2016], and *weight normalization* [Salimans and Kingma, 2016], have been proposed to accelerate the training of ANNs. Although successful in improving the convergence rate, such methods are still far behind from the learning capabilities of the biological brain. The limitations of ANNs with respect to the brain can be largely ascribed to the substantial simplification of their structure and dynamics compared to the mammals' neural circuits. Several



mechanisms of paramount importance for brain functioning, including synaptic integration and local regulation of weight strength, are typically not modeled in BP-based training of ANNs. Overcoming this limitation could be key in bringing artificial networks' performance closer to animal intelligence [Richards et al., 2019]. Synaptic integration is the process by which neurons combine the spike trains received by thousands of presynaptic neurons prior to the generation of action potentials (APs) [Williams and Stuart, 2001]. The axonal APs are elicited in the axon of the cell as a response to the input received from the cell's dendrites, and act as overall output signal of the neuron. Experimental evidence has shown that, at least in CA1 cells, input signals reaching the same postsynaptic cell from different presynaptic neurons may interact with non-linear dynamics, due to the active properties of the dendrites [Etherington et al., 2010, Williams and Atkinson, 2008]. Specifically, when strong depolarization occurs in a dendritic branch, a dendritic action potential is elicited in the region. The dendritic AP boosts the amplitude of the sum of the excitatory postsynaptic potentials (EPSPs) that generated it, thereby amplifying the dendritic input before it reaches the soma to elicit an axonal AP. The generation of a dendritic spike requires that enough presynaptic cells spatially connected to the same branch are active close in time with sufficient synaptic strength. As a consequence, the ability of synaptic inputs to influence the output of the postsynaptic neuron depends on their location within the dendritic tree. The differences between axonal and dendritic spikes are discussed in Ref. [Etherington et al., 2010]. The powerful computational abilities of neurons are suggested to stem from the complex nonlinear dynamics derived from dendritic spikes [Li et al., 2020]. Figure 5.1(a) illustrates such a mechanism and shows how the impact of each presynaptic neuron depends also on the signals delivered to the same postsynaptic neuron through other presynaptic connections. Thus, the local weight distribution can be responsible of boosting the input signal at specific nodes. Similarly to neurons in the brain, nodes in ANNs receive inputs from many cells and produce a single output. We can relate the activation of artificial nodes to axonic APs, but there is no rigorous translation of the mechanism of dendritic APs into the dynamics of point neurons. However, dendritic spikes are strongly affected by the distribution of synaptic strengths within dendritic branches. Similarly, the nonlinear dynamics of artificial nodes are affected by the weight distribution of synapses incoming to a layer of nodes. Surprisingly, in common training approaches for ANNs, a mechanism taking into account the weight distribution for each node is lacking.

Furthermore, synaptic plasticity in the brain is driven mainly by local signals, such as the activity of neighboring neurons [Whittington and Bogacz, 2019]. The local interaction between synapses plays a crucial role in regulating weight changes during learning. In this context, the mechanism of heterosynaptic competition allows regulating synapse growth by limiting the total strength of synapses connected to the same pre- or postsynaptic neuron [Royer and Paré, 2003b]. This phenomenon occurs as a nonlinear competition across synapses at each neuron. Specifically, as the summed weight of synapses into (or out of) a neuron hits a threshold, all the incoming (or outgoing) synapses to that neuron undergo a slight heterosynaptic long-term depression ("summed-weight limit rule") [Fiete et al., 2010]. Additionally, in the cortex, each neuron tends to target a specific firing rate, and synaptic strengths are regulated to keep such rates constant, despite input perturbation [Turrigiano et al., 1998]. Synaptic scaling acts as a global negative feedback control of synaptic strength, regulating the weight changes based on the neural local activities [Turrigiano, 2008, Moulin et al., 2020, Ibata et al., 2008]. These homeostatic mechanisms are typically not modeled in the training of standard ANNs, which rely on global signals instead of local information [Nøkland and Eidnes, 2019, Bengio et al., 2016]. Indeed, the BP algorithm rely on the simplified training strategy of assigning the error on a weight-by-weight fashion. Each synaptic weight is updated based on its *individual* contribution to the *global* output error of the network as a response to a specific input sample. We refer to this input-specific contribution as *input-driven responsibility*. Although earlier works have attempted to encode metaplasticity (*i.e.*, the alteration of synaptic plasticity [Abraham, 2008]) in the training of networks via weight-dependent learning rules ([Pfeiffer et al., 2010, Legenstein et al., 2010, Frémaux et al., 2010, Soltoggio and Stanley, 2012]), they invariably depend on a modulation of the Hebbian learning rule rather than ANNs training and do not account for the local weight distribution. Some training strategies more biologically plausible than BP [Whittington and Bogacz, 2019], such as the feedback alignment (FA) algo-

rithm [Lillicrap et al., 2016] and its direct and indirect feedback alignment (DFA, IFA) variants [Nokland, 2016], have been proposed, yet they do not explicitly model the neural mechanisms mentioned above.

Here, we make progress in reconciling neurophysiological insights with machine intelligence by proposing a biologically inspired optimizer that incorporates principles from biology, including synaptic integration, heterosynaptic competition [Bailey et al., 2000] and synaptic scaling [Turrigiano et al., 1998]. Our approach achieves substantial benefits in the training of fully connected neural networks (FCNNs), leading to a systematically faster training convergence, higher inference accuracy, better scalability of performance with network complexity, and mitigation of catastrophic forgetting. Our novel approach effectively boosts also the performance of convolutional neural networks (CNNs) and spiking neural networks (SNNs) [Ghosh-Dastidar and Adeli, 2009] on temporal data. These results validate the hypothesis that biologically inspired ANN and SNN models feature superior performance in software simulations [Sinz et al., 2019], and provide guidelines for designing a new generation of neuromorphic computing technologies [Chicca and Indiveri, 2020].

## 5.2 Results

### 5.2.1 The GRAPES algorithm

The synaptic integration and the local synaptic strength regulation mechanisms are complex processes which depend on various factors, such as the large variability in size, structure, excitability, intercellular distance and temporal dynamics of synapses and dendritic spines [Spruston, 2008]. The simple point-like structure of a synchronously operating ANN node does not allow one to reproduce the rich dynamics enabled by the neuronal complex morphology. Hence, a direct translation of the mechanism of dendritic integration for ANNs is not straightforward. Here, we take inspiration from the nonlinear synaptic dynamics and introduce a deep learning optimizer to boost the training of FCNNs. Our goal is to present an effective algorithm, inspired by biological mechanisms, and elucidate its potential impact on the properties of ANNs. This novel approach can also be easily applied to more biologically plausible neuronal models such as SNNs. Our algorithm builds on three observations:

1. In the brain, due to the spiking nature of the information, a signal is propagated only if a postsynaptic neuron receives enough input current from the presynaptic population to elicit action potentials. In rate-based models of neural activity [Brette, 2015], a neuron with high firing rate is more likely to elicit high activity in the downstream neurons than neurons with low firing rate.
2. A single presynaptic neuron is responsible only for a fraction of the driving force that leads the postsynaptic neuron to fire. Hence, the impact of a presynaptic neuron on the downstream layers depends also on all the other presynaptic neurons connected to the same postsynaptic cell.
3. If we neglect specific distributions of the inputs, the firing probability of a postsynaptic neuron depends on the average strength of the presynaptic connections. If the average strength is high, the postsynaptic neuron is more likely to reach the spiking threshold and thus to propagate further the information encoded in the presynaptic population. Therefore, the postsynaptic neuron and the related presynaptic population have a high *responsibility* on the network's output and its potential error.

We refer to the intrinsic responsibility of the network as *network-driven responsibility*, as opposed to the *input-driven responsibility* mentioned above. Analogously, we introduce for ANNs the notion of *node importance* stemming from the node responsibility in propagating the information received from its presynaptic population to the output layer. The concept of *node importance* builds on the mechanism of dendritic integration in brain. In biological neurons, the dynamics of dendritic spikes originate from the spatial grouping of input cells based on the dendritic branch they send

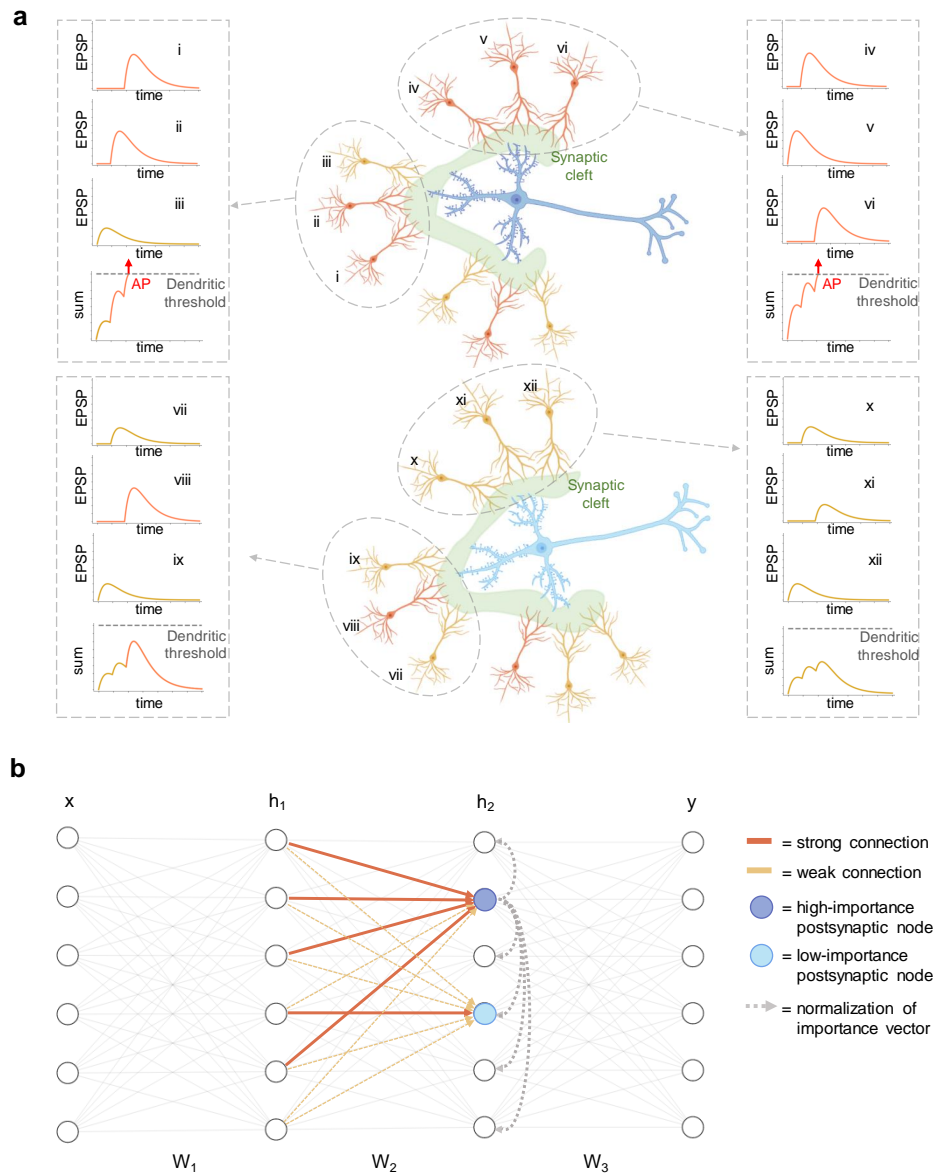


Figure 5.1: **Synaptic strength distribution in biological and artificial networks.** (a) In biological synapses, during the process of synaptic integration, dendritic spikes can enhance the impact of synchronous inputs from dendrites belonging to the same tree. Excitatory postsynaptic potentials (EPSPs) with the same amplitude but different locations in dendritic tree may lead to different responses. For example, dendrites *i*, *iv* and *viii* send similar signals, but only the *i* and *iv* contribute in driving an AP, since their respective branches receive sufficient further excitation from other connected dendrites. In the top image, the postsynaptic neuron (dark blue) receives inputs mostly from dendrites generating strong EPSPs (orange) and only few generating weak EPSPs (yellow). The bottom postsynaptic neuron (light blue) receives most inputs from weak-EPSPs dendrites. Because of such dendritic distribution, the dark blue neuron exhibits higher firing probability and thus its *importance* is higher with respect to the light blue neuron. (b) The structure of an FCNN is much simpler than that of biological neurons with presynaptic connections arranged in dendritic trees. However, analogously to panel (a), the *node importance* of each node arises from the distribution of the weight strength within each layer. The blue node has a high *node importance* since most of its incoming synapses are strong. Conversely, the light blue node *importance* is lower, since the presynaptic population exhibits a weaker mean strength. The gray dotted lines emanating from the neuron with the highest importance and arriving at the other nodes in the same layer represent a normalization effect, resembling the winner-take-all competition based on the highest importance value.

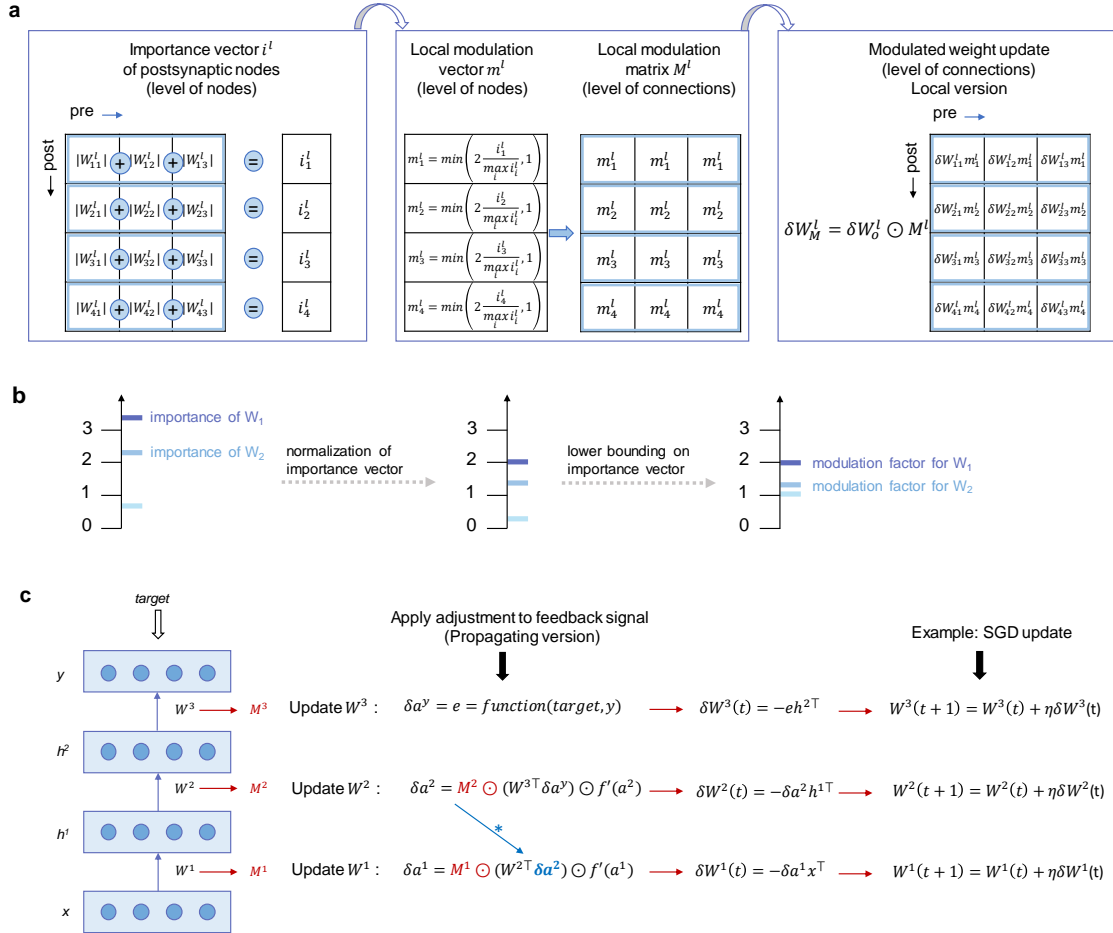


Figure 5.2: **Computation of the modulation factor.** (a) The computation of the importance vector for one hidden layer, based on the associated weight matrix, is followed by the computation of the local modulation vector, based on the node importance. In the local version, GRAPES adjusts the weight update through the Hadamard multiplication of the initial weight-update matrix  $\delta W_o^l$  with the local modulating vector  $M^l$ . (b) Sketch illustrating the steps in computing the modulation factor using the importance vector in the simple case of three postsynaptic nodes. (c) Algorithm for the propagation of the modulation factor to the upstream layers in a two-hidden layer network. The activations in the network are computed as  $a^1 = W^1 x$ ,  $h^1 = f(a^1)$  and  $a^2 = W^2 h^1$ ,  $h^2 = f(a^2)$ , and the network output as  $a^y = W^3 h^2$ ,  $y = f_y(a^3)$ . Note that  $\delta a^2$  is adjusted through the modulation matrix  $M^2$  and such adjustment also affects the upstream layer since  $\delta a^1$  contains  $\delta a^2$ .

their signal to. In FCNNs the presynaptic nodes are connected to the postsynaptic nodes without being first grouped into dendritic branches. However, such a simpler non-dendritic morphology still offers the possibility to perform spatial grouping of input nodes based on the postsynaptic layer. In absence of dendritic branches presynaptic cells can be grouped based on point-like postsynaptic nodes rather than on the dendritic branches of a single neuron. Consequently, while in biological circuits the dendritic integration is related to boosting input signals at the level of branches, the *node importance* in FCNNs is related to signal modulation at the level of point-like nodes. Therefore the *node importance* is related to the average strength of the synapses connected to such node. Figure 5.1(b) illustrates the concept of *node importance* in an FCNN.

Based on this notions, we devised a novel algorithm, that we call GRAPES (Group Responsibility for Adjusting the Propagation of Error Signals). For simplicity, we begin by presenting the algorithm as a modulation of the error propagation in a network trained with BP and optimized with stochastic gradient descent (SGD). Next, we demonstrate that GRAPES can be conveniently applied also to other commonly used optimizers, such as Nesterov Accelerated Gradient (NAG) [Nesterov, 1983] and RMSprop [Tieleman and Hinton, 2012], to other more biologically realistic training schemes, such as FA and DFA, and to networks with the biologically realistic dynamics of spiking neurons.

The GRAPES algorithm modulates the error signal at each synaptic weight based on two quantities: the *node importance* and the *local modulation factor*. Mathematically, we define the *node importance* of a given node  $n$  belonging to layer  $l$  as the sum of the absolute strength over all weights of layer  $l$  whose postsynaptic neuron is  $n$

$$i_n^l = \sum_{\text{pre}=1}^N |W_{n,\text{pre}}^l|, \quad (5.1)$$

where  $N$  is the number of incoming connections to node  $n$ . The sum is performed over the absolute value of the parameters since we consider the connection strength (*i.e.*, how much each weight amplifies or attenuates an incoming signal) rather than its excitatory or inhibitory nature. Alternatively, in specific cases discussed further in the paper, the *node importance* may also be obtained from the sum of the absolute strength of all weights of layer  $l+1$  outgoing from the same presynaptic neuron. We remark that the importance defined in Equation 5.1 depends only on the intrinsic state of the network and not on the input value. Therefore, in the initial phases of training, the importance is not related to the task. However, as the network undergoes training, the weight distribution becomes dependent on the task which the network is trained on, and consequently the importance vector evolves to be specific to the task.

The *importance vector*  $i^l$  for layer  $l$  contains the *node importance* values for each postsynaptic node  $n$  in  $l$ . By normalizing the importance vector by its maximal value, multiplying it by 2 and lower-bounding by 1, we obtain the *local modulation vector*

$$m^l = \min \left( 2 \frac{i^l}{\max(i^l)}, 1 \right), \quad (5.2)$$

whose elements are bounded in the range  $1 \leq m^l \leq 2$ . Such an interval has been defined on the basis of an empirical optimization (see Table 5.3). The  $n$ -th *local modulation factor* is the  $n$ -th element of the resulting vector and indicates the responsibility of the postsynaptic node  $n$  and its associated connected weights on the output of the network. In order to build the *local modulation matrix*  $M^l$  for layer  $l$ , the *local modulation vector* is tiled as many times as the presynaptic population size. Each element of the matrix is associated with one synaptic weight of layer  $l$ . Therefore, by construction, the modulation has the same value for all weights  $W_{n,\text{pre}}^l$  connected to the same postsynaptic neuron  $n$ .

With these quantities at hand, in the local version of GRAPES, we adjust the error signal in layer  $l$  through a Hadamard multiplication of the weight-update matrix  $\delta W_o^l$  computed with the standard optimizer (*e.g.*, SGD) with the local modulation matrix  $M^l$ . The weight-change matrix, in which each row corresponds to a post-synaptic node and each column to a pre-synaptic node, is therefore

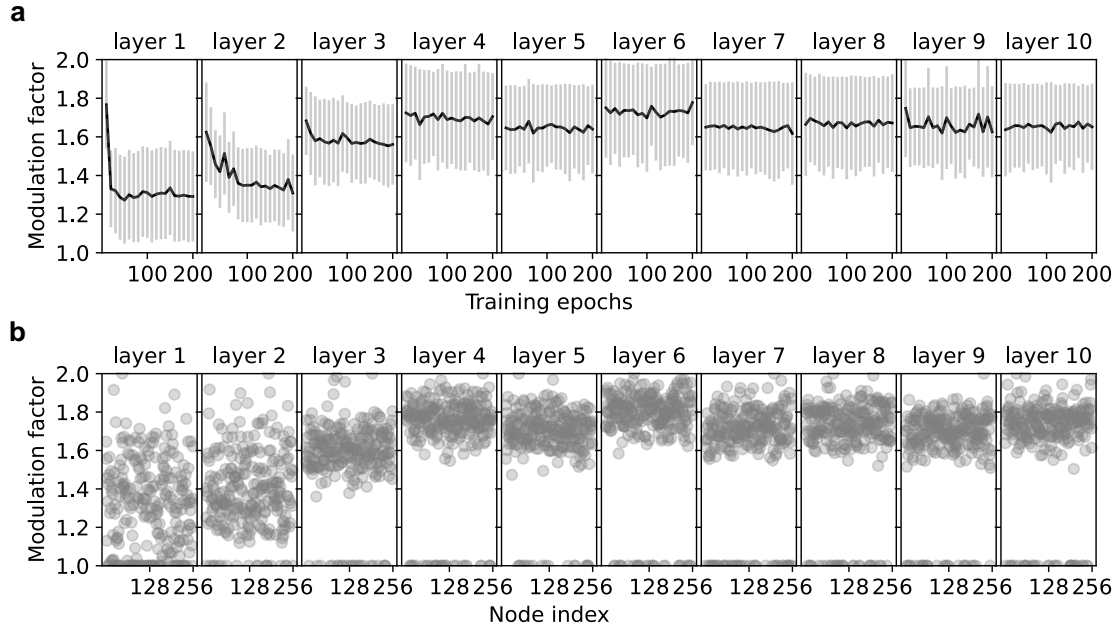


Figure 5.3: **Dynamics of the modulation factor.** (a) Mean and standard deviation dynamics of the modulation factors for a  $10 \times 256$  tanh network, with 10% dropout, trained with BP, SGD and GRAPES modulation for 200 training epochs. The modulation factors are recorded for each layer every 10 epochs. (b) Distribution of the modulation factors at the end of training.

modulated row-wise:

$$\delta W_M^l = \delta W_o^l \odot M^l. \quad (5.3)$$

The main steps of the computation of importance vector, local modulation vector and matrix for a single hidden layer  $l$ , and the update step of the local version of GRAPES are summarized in Figures 5.2(a) and 5.2(b). The same concept of importance and modulation as described for the fully connected models can be applied to convolutional layers. In fully connected layers, the computation of the importance relies on grouping the weights (1D connections) based on the postsynaptic node. In convolutional layers, we compute the importance of each 2D filter by grouping the filters based on the postsynaptic maps. Figure 5.4 shows a schematic of the main steps to compute the filter importance. The filter update is modulated accordingly as described in Equation 5.3.

In the propagating version of the algorithm, the modulation factor is incorporated in the error signal of each layer and propagated to the upstream layers, where it is incorporated in the respective weight updates. In Figure 5.2(c) we outline the algorithm for the propagation of the modulation factor in a two-hidden layer network. The propagating version provides the greatest benefits in terms of classification accuracy and convergence speed compared to the local version, as shown in the Table 5.1. Hence, the propagating version of the algorithm is the default method adopted in the simulations. Finally, the weight update obtained by applying the GRAPES modulation to SGD can be expressed as:

$$W^l(t+1) = W^l(t) + \eta \delta W_M^l(t). \quad (5.4)$$

GRAPES does not change the computational complexity of SGD and, since the modulation factor is computed only after the parameter update (*e.g.*, at the end of each batch), the additional computations are negligible for large batch size.

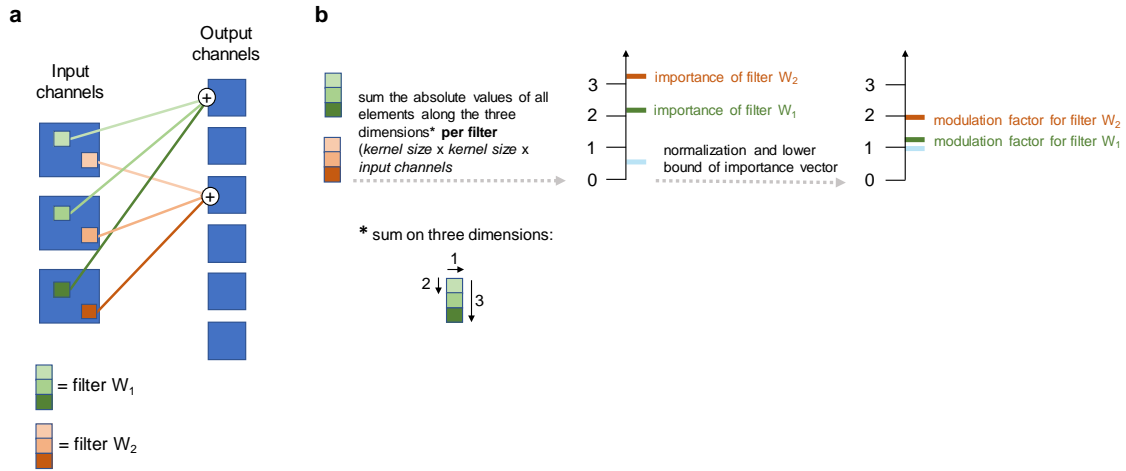


Figure 5.4: Graphical representation of the GRAPES algorithm for convolutional layers.

(a) Filters of convolutional layers can be grouped according to the output channels, in the same way as weights in fully-connected layers can be grouped based on the post-synaptic node. (b) Steps to compute the importance vector and the modulation factors for the convolutional filters starting from the filter elements.

By construction, the error signal modulation implemented in GRAPES presents some analogies with the biological mechanisms of heterosynaptic competition and synaptic scaling. Firstly, the *node importance* is defined as the sum of the synaptic weights connected to the same node. As in heterosynaptic competition, the information on the total synaptic strength is used to modulate the weight magnitude. However, while in heterosynaptic competition the total synaptic weight is used to solely determine depression by changing directly the weights [Fiete et al., 2010], in GRAPES the total weight information is used to adjust the weight update, hence leading to both strengthening or weakening of the synapses. Secondly, by definition, the *local modulation factor* in GRAPES is equal for all synapses incoming to the same node. This leads back to synaptic scaling, in which a neuron-specific multiplicative scaling factor adjusts the weights of the synapses connected to the same neuron based on the local activity so that the neuron maintains a target firing rate.

Figure 5.3(a) displays the evolution of the mean of the modulation factor during training for each layer of a 10 hidden layer network. The dynamics of the modulation factor are different for each layer. The mean of the modulation factor exhibits the most pronounced decay in the first three hidden layers, whereas it either decreases very slowly or remains constant in the downstream hidden layers. In Figure 5.3(b), we show the distribution of the modulation factors for each layer after training. In each layer, a subset of the modulation factors is equal to 1, due to the lower bounding operation in Eq. 5.2. The remaining values are distributed with mean and variance specific to each layer. Therefore, based solely on the current state of the network weights, GRAPES offers a simple approach to modulate the error signal at each synapse using node-specific factors whose temporal evolution, mean and variance can be different for each hidden layer. Note that, for some layers, the decrease of the modulation factor with respect to its initial value (Figure 5.3a) resembles a learning rate decay scheme [Darken et al., 1992]. Compared to the classical learning-rate-schedules, GRAPES provides two main advantages. First, to apply a learning rate decay, a time-consuming search of the best decay scheme and its hyperparameters is required for each network setting and task. Instead, the spontaneous decay provided by GRAPES (Figure 5.3a) does not need to be optimized in advance, thus allowing the modulation factor to naturally adapt

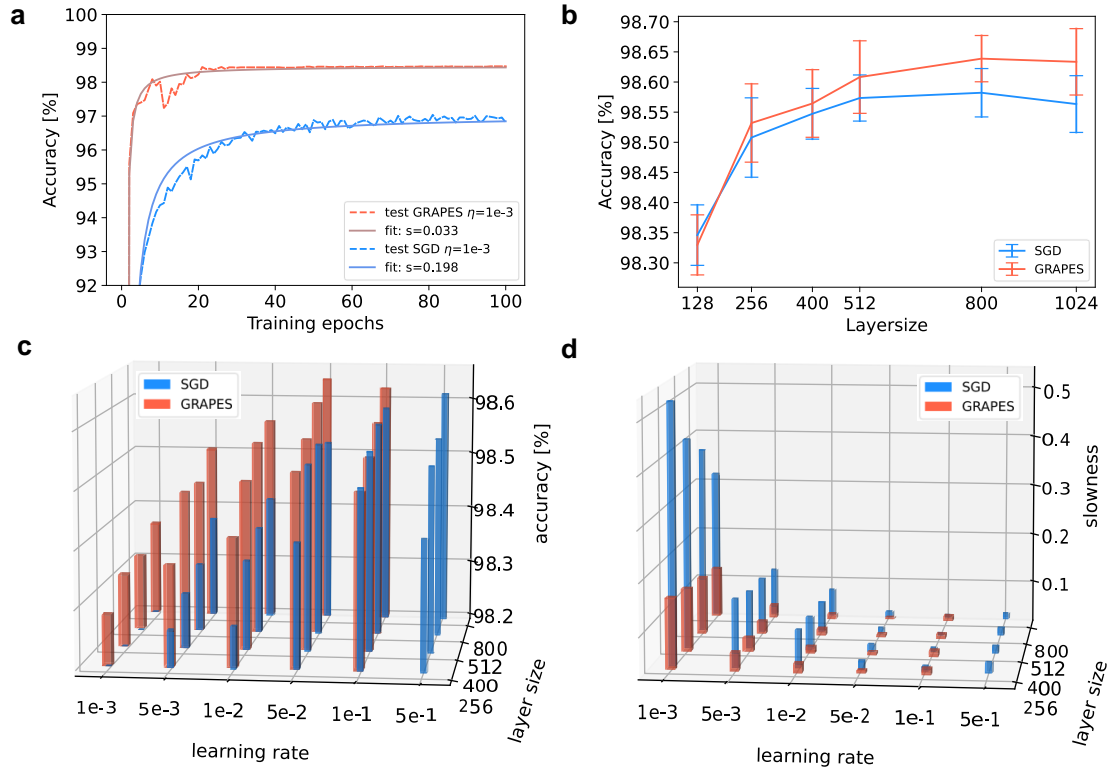


Figure 5.5: **GRAPES applied to BP on MNIST.** Results of training fully connected models on MNIST with BP and SGD, with and without GRAPES. The red curves and bars correspond to the networks trained with GRAPES, whereas the blue curves and bars to the networks trained with SGD. (a) Testing curve (dotted line) and fit using the *plateau function* (solid line) for the  $10 \times 256$  ReLU network, trained without dropout. The fit is performed on the initial 100 epochs of a single trial. (b) Test accuracy of networks with four hidden layers as a function of the layer size. The learning rate is optimized separately for SGD and for GRAPES. For different layer sizes the optimal learning rate slightly varies. For most layer sizes the optimal learning rate is  $\eta = 0.1$  for SGD and  $\eta = 0.05$  for GRAPES. Each curve is a cross section of the bar plot in panel c along the learning rate axis. (c) Test accuracy of networks with four hidden layers as a function of the layer size and the learning rate, trained with 10% dropout. (d) Convergence rate of the models in c. In panels c,d for visualization purposes the bases of the SGD and GRAPES bars are slightly shifted from each other. The actual learning rates and layer sizes are the same for both and are reported in the axes' labels. For panels b,c,d the accuracy for each run is computed as the mean of the test accuracy over the last 10 training epochs. The reported result is the mean and standard deviation (error bars in b) over the accuracy of 10 independent runs.



Layers	Act.	DO	$\eta$	Train epochs	acc [%]	acc [%]	acc [%]	s	s	s
					SGD	GRAPES prop	GRAPES local	SGD	GRAPES prop	GRAPES local
3 × 256	ReLU	0%	1e-3	200	97.46±0.09	<b>97.95±0.07</b>	97.80±0.04	0.36±0.02	<b>0.17±0.01</b>	0.27±0.02
3 × 256	ReLU	10%	1e-3	200	97.43±0.09	<b>98.25±0.08</b>	97.85±0.06	0.46±0.01	<b>0.21±0.01</b>	0.31±0.02
3 × 256	tanh	0%	1e-2	200	98.04±0.11	<b>98.20±0.07</b>	98.12±0.09	0.15±0.01	<b>0.08±0.00</b>	0.12±0.01
3 × 256	tanh	10%	1e-2	200	98.16±0.13	<b>98.42±0.07</b>	98.33±0.06	0.16±0.00	<b>0.10±0.00</b>	0.13±0.00

Table 5.1: **Test accuracy and convergence rate on the MNIST dataset for networks trained with BP and SGD optimizer, comparing the results for the local and propagating version of GRAPES.** The reported result is the average and standard deviation of best test accuracy over five runs. “DO” stands for dropout,  $\eta$  is the learning rate. Both the local and the propagating versions of GRAPES always outperform the classic SGD both in terms of accuracy (acc) and slowness (s). The propagating version allows for the best improvements.

to different architectures and data sets. Furthermore, while with the conventional learning-rate-schedule approach the learning rate is equal for each parameter, GRAPES allows the update step to be adjusted differently for each weight (Figure 5.3b). Specifically, we have shown in Figure 5.3 that GRAPES implements a dynamic learning schedule for each weight. We demonstrate the stability of such a learning schedule by analytically proving the convergence properties of GRAPES in subsection.

### 5.2.2 Convergence analysis of GRAPES applied to SGD

We have shown that GRAPES implements a dynamic learning schedule for each weight. Here, we demonstrate the stability of GRAPES by analytically proving its convergence properties. The proof relies on the online learning framework proposed in [Zinkevich, 2003], similarly to the investigation of the convergence of Adam optimizer in [Kingma and Ba, 2015]. In online convex programming, an algorithm addresses a sequence of convex programming problems, each consisting of a convex feasible set  $W \subseteq \mathbb{R}^n$ , which is the same for all problems, and a convex cost function  $f^t(w) : W \rightarrow \mathbb{R}$ , which in principle is different for each problem. Given an arbitrary, unknown sequence of  $T$  such convex cost functions  $f^1(w), f^2(w), \dots, f^T(w)$ , at each time step  $t$  the algorithm must predict the parameter vector  $w^t$  before observing the cost function  $f^t$ . After the vector  $w^t$  is selected, it is evaluated on  $f^t$ . Since the convex functions can be unrelated to one another and the nature of the sequence is unknown in advance, we evaluate the convergence of the GRAPES modulation using the regret function [Zinkevich, 2003]. The regret function computes a difference between the proposed online algorithm and an “offline” algorithm. In the online algorithm, for each convex function a decision is made before the cost is known. On the other hand, the “offline” algorithm, prior to making a prediction, has knowledge about the sequence of convex cost functions, and makes a single choice to minimize the cost function  $f(w) = \sum_{t=1}^T f^t(w)$ . The regret function is defined as the difference between the cost of the online algorithm and the cost of the offline algorithm. It is computed as the sum of all the previous differences between the online prediction  $f^t(w^t)$  and the best fixed point parameter  $f^t(w^*)$ :

$$R(T) = \sum_{t=1}^T [f^t(w^t) - f^t(w^*)] \quad (5.5)$$

where  $w^* = \operatorname{argmin}_{w \in W} \sum_{t=1}^T f^t(w)$ .

We show that GRAPES modulation has  $O(\sqrt{T})$  regret bound, similarly to standard SGD.

**Theorem** We express the learning rule obtained by applying the local GRAPES modulation to SGD as:

$$w^{t+1} = w^t - \eta^t M^t g^t \quad (5.6)$$

where:

- $g^t = \nabla f^t(w^t)$  is the gradient of the cost function with respect to the parameter  $w^t$  at time step  $t$
- $\eta^t$  is the learning rate at time step  $t$
- $M^t$  is the modulation factor at time step  $t$ . We define  $M_{max}$  as the maximum value obtained by the modulation factor. Due to the constraints of the modulation factor  $M_{max} \leq 2$

Assume that the GRAPES modulation applied to SGD during training can select only the parameters  $w^i$  belonging to a convex feasible set  $W$ . Assume that the programming problem consists of an arbitrary, unknown sequence of convex cost functions  $f^1(w), f^2(w), \dots, f^T(w)$  such that  $f^i(w) : W \rightarrow \mathbb{R}$ . Assume that:

1. The feasible set is bounded,  
*i.e.*,  $\exists k \in \mathbb{R} : \forall w^i, w^j \in W, \|w^i - w^j\| \leq k$   
 We define  $\|W\| = \max_{w^i, w^j \in W} \|w^i - w^j\|$
2. The feasible set is closed,  
*i.e.*,  $\forall \{w^1, w^2, \dots\}$  where  $w^t \in W, \forall t$ ,  
 if  $\exists w \in \mathbb{R}^n$  such that  $w = \lim_{t \rightarrow \infty} w^t$ , then  $w \in W$
3. The feasible set is non-empty,  
*i.e.*,  $\exists w \in W$
4. The cost functions are differentiable, *i.e.*,  $\forall t, f^t$  is differentiable.  
 We define the gradient of the cost function as  $g^t = \nabla f^t(w^t)$
5. The cost functions have bounded gradients,  
*i.e.*,  $\exists k \in \mathbb{R} : \forall t, \forall w \in W, \|\nabla f^t(w)\| \leq k$   
 We define  $\|\nabla f\| = \max_{w \in W, t \in \{1, 2, \dots\}} \|\nabla f^t(w)\|$
6.  $\forall t, \forall w \in \mathbb{R}^n, \exists$  an algorithm  $A$ , so that, given  $w$  and  $\nabla f^t(w)$ , it can produce  $\operatorname{argmin}_{w \in W} \|w^i - w^j\|$

Assume furthermore that the learning rate follows  $\eta_t = t^{-1/2}$ .

Under these assumptions the GRAPES modulation applied to SGD achieves the following guarantee, for all  $T \geq 1$ .

$$R(T) \leq \frac{\|W\|^2 \sqrt{T}}{2M_T} + M_{max} \|\nabla f\|^2 \left( \sqrt{T} - \frac{1}{2} \right) \quad (5.7)$$

**Corollary** Under the same conditions of the Theorem, GRAPES modulation applied to SGD achieves the following guarantee, for all  $T \geq 1$ .

$$\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right) \quad (5.8)$$

Under the same conditions, standard SGD achieves the following guarantee, for all  $T \geq 1$ .

$$R_{SGD}(T) \leq \frac{\|W\|^2 \sqrt{T}}{2} + \|\nabla f\|^2 \left( \sqrt{T} - \frac{1}{2} \right) \quad (5.9)$$

and

$$\frac{R_{SGD}(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right) \quad (5.10)$$

### 5.2.3 Main steps of the convergence analysis

First we show that, as  $f^t$  are convex functions, the regret function has an upperbound and can be expressed as

$$R(T) \leq \sum_{t=1}^T (w^t - w^*) \cdot g^t \quad (5.11)$$

**Definition 1** For a convex feasible set  $W$ , a function  $f : W \rightarrow \mathbb{R}$  is convex if for all  $w^i, w^j \in W$ , for all  $\lambda \in [0, 1]$ ,

$$\lambda f(w^i) + (1 - \lambda)f(w^j) \geq f(\lambda w^i + (1 - \lambda)w^j) \quad (5.12)$$

**Lemma 2** [Zinkevich, 2003] If a function  $f : W \rightarrow \mathbb{R}$  is convex, then for all  $w^i, w^j \in \mathbb{R}$ ,

$$f(w^j) \geq f(w^i) + \nabla f(w^i)^\top (w^j - w^i) \quad (5.13)$$

Using Lemma 2 we have

$$f^t(w^t) - f^t(w^*) \leq (g^t)^\top (w^t - w^*) = \sum_{q=1}^n g_q^t (w_q^t - w_q^*), \quad (5.14)$$

where  $g_q^t$ ,  $w_q^t$  and  $w_q^*$  are the  $q$ -th components of  $g^t$ ,  $w^t$  and  $w^*$  respectively. This leads to

$$R(T) = \sum_{t=1}^T [f^t(w^t) - f^t(w^*)] \quad (5.15)$$

$$\leq \sum_{t=1}^T g^t \cdot (w^t - w^*) \quad (5.16)$$

Now we use the weight update rule  $w^{t+1} = w^t - \eta^t M^t g^t$  and the learning rate dynamics  $\eta_t = t^{-1/2}$  to show that

$$\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right) \quad (5.17)$$

We start by manipulating the weight update rule:

$$(w^{t+1} - w^*)^2 = (w^t - \eta^t M^t g^t - w^*)^2 = \quad (5.18)$$

$$= [(w^t - w^*) - \eta^t M^t g^t]^2 = \quad (5.19)$$

$$= (w^t - w^*)^2 - 2\eta^t M^t g^t (w^t - w^*) + (\eta^t)^2 (M^t)^2 (g^t)^2 \quad (5.20)$$

Since, by assumption,  $\|g^t\| = \|\nabla f^t\| \leq \|\nabla f\|$ , we can upper bound the expression above as:

$$(w^{t+1} - w^*)^2 \leq (w^t - w^*)^2 - 2\eta^t M^t g^t (w^t - w^*) + (\eta^t)^2 (M^t)^2 \|\nabla f\|^2 \quad (5.21)$$

By rearranging we obtain:

$$\begin{aligned} \frac{2\eta^t M^t g^t (w^t - w^*)}{2\eta^t M^t} &\leq \frac{(w^t - w^*)^2 - (w^{t+1} - w^*)^2 + (\eta^t)^2 (M^t)^2 \|\nabla f\|^2}{2\eta^t M^t} \\ g^t (w^t - w^*) &\leq \frac{1}{2\eta^t M^t} [(w^t - w^*)^2 - (w^{t+1} - w^*)^2] + \frac{\eta^t}{2} M^t \|\nabla f\|^2 \end{aligned} \quad (5.22)$$

By substituting equation 5.22 in equation 5.11 we can write:

$$\begin{aligned}
R(T) &\leq \sum_{t=1}^T g^t \cdot (w^t - w^*) \\
&\leq \sum_{t=1}^T \left[ \frac{(w^t - w^*)^2 - (w^{t+1} - w^*)^2}{2\eta^t M^t} + \frac{\eta^t}{2} M^t \|\nabla f\|^2 \right] \\
&\leq \sum_{t=1}^T \frac{(w^t - w^*)^2 - (w^{t+1} - w^*)^2}{2\eta^t M^t} + \frac{\|\nabla f\|^2}{2} \sum_{t=1}^T \frac{\eta^t}{2} M^t \\
&= \sum_{t=1}^T \frac{(w^t - w^*)^2}{2\eta^t M^t} - \sum_{t=1}^T \frac{(w^{t+1} - w^*)^2}{2\eta^t M^t} + \frac{\|\nabla f\|^2}{2} \sum_{t=1}^T \frac{\eta^t}{2} M^t
\end{aligned} \tag{5.23}$$

We apply a change of variable to the second term of equation 5.23 as  $t' = t + 1$  to obtain:

$$\begin{aligned}
\sum_{t=1}^T \frac{(w^{t+1} - w^*)^2}{2\eta^t M^t} &= \sum_{t'=2}^{T+1} \frac{(w^{t'} - w^*)^2}{2\eta^{t'-1} M^{t'-1}} = \\
&= \frac{(w^{T+1} - w^*)^2}{2\eta^T M^T} + \sum_{t'=2}^T \frac{(w^{t'} - w^*)^2}{2\eta^{t'-1} M^{t'-1}}
\end{aligned} \tag{5.24}$$

We then insert the result of equation 5.24 into equation 5.23:

$$R(T) \leq \left[ \frac{(w^1 - w^*)^2}{2\eta^1 M^1} + \sum_{t=2}^T \frac{(w^t - w^*)^2}{2\eta^t M^t} \right] - \left[ \frac{(w^{T+1} - w^*)^2}{2\eta^T M^T} + \sum_{t=2}^T \frac{(w^t - w^*)^2}{2\eta^{t-1} M^{t-1}} \right] + \tag{5.25}$$

$$+ \frac{\|\nabla f\|^2}{2} \sum_{t=1}^T \frac{\eta^t}{2} M^t = \tag{5.26}$$

$$= \frac{(w^1 - w^*)^2}{2\eta^1 M^1} - \frac{(w^{T+1} - w^*)^2}{2\eta^T M^T} + \frac{1}{2} \sum_{t=2}^T \left( \frac{1}{\eta^t M^t} - \frac{1}{\eta^{t-1} M^{t-1}} \right) (w^t - w^*)^2 + \tag{5.27}$$

$$+ \frac{\|\nabla f\|^2}{2} \sum_{t=1}^T \frac{\eta^t}{2} M^t \tag{5.28}$$

Now we use the assumption that the feasible set  $W$  is bounded, *i.e.*,  $\forall t, \|w^t - w^*\| \leq \|W\|$  to find:

$$\begin{aligned}
R(T) &\leq \frac{\|W\|^2}{2\eta^1 M^1} + \frac{1}{2} \sum_{t=2}^T \left( \frac{1}{\eta^t M^t} - \frac{1}{\eta^{t-1} M^{t-1}} \right) \|W\|^2 + \\
&+ \frac{\|\nabla f\|^2}{2} \sum_{t=1}^T \eta^t M^t = \\
&= \frac{\|W\|^2}{2} \left[ \cancel{\frac{1}{\eta^1 M^1}} + \left( \cancel{\frac{1}{\eta^2 M^2}} - \cancel{\frac{1}{\eta^1 M^1}} \right) + \dots + \left( \frac{1}{\eta^T M^T} - \cancel{\frac{1}{\eta^{T-1} M^{T-1}}} \right) \right] + \\
&+ \frac{\|\nabla f\|^2}{2} \sum_{t=1}^T \eta^t M^t = \\
&= \frac{\|W\|^2}{2} \frac{1}{\eta^T M^T} + \frac{\|\nabla f\|^2}{2} \sum_{t=1}^T \eta^t M^t
\end{aligned} \tag{5.29}$$

Next, we consider the assumptions  $\eta_t = t^{-1/2}$  and  $\forall t, M^t \in [1, 2)$ .

$$\begin{aligned}
\sum_{t=1}^T \eta^t M^t &= \sum_{t=1}^T \frac{1}{\sqrt{t}} M^t \\
&\leq 2 \left( 1 + \int_{t=1}^T \frac{dt}{\sqrt{t}} \right) \\
&\leq 2 \left( 1 + \left[ 2\sqrt{t} \right]_1^T \right) \\
&\leq 2 \left( 2\sqrt{T} - 1 \right)
\end{aligned} \tag{5.30}$$

We then use equation 5.30 and the condition  $\forall t, M^t \in [1, 2)$  in equation 5.29:

$$R(T) \leq \frac{\|W\|^2}{2} \frac{1}{\frac{M^T}{\sqrt{T}}} + \frac{\|\nabla f\|^2}{2} 2 \left( 2\sqrt{T} - 1 \right) \tag{5.31}$$

$$= \frac{\|W\|^2 \sqrt{T}}{2M^T} + \|\nabla f\|^2 \left( 2\sqrt{T} - 1 \right) \tag{5.32}$$

Therefore,

$$\limsup_{T \rightarrow \infty} \frac{R(T)}{T} = \frac{\frac{\|W\|^2 \sqrt{T}}{2M^T} + \|\nabla f\|^2 \left( 2\sqrt{T} - 1 \right)}{T} = 0 \tag{5.33}$$

#### 5.2.4 Simulation results on handwritten digit classification

To illustrate the benefits of GRAPES on the training of ANNs, we have enhanced the standard minibatch SGD by incorporating the GRAPES modulation scheme, and are referring to it as ‘‘GRAPES’’. We initially compare the performance of GRAPES against standard minibatch SGD, which from now on we will simply call ‘‘SGD’’, on the MNIST data set [LeCun and Cortes, 2010]. To evaluate the convergence rate, we relied on a Michaelis-Menten-like equation [Michaelis and Menten, 1913] and introduced the novel *plateau equation for learning curves*:

$$\text{accuracy} = \frac{\text{max\_accuracy} \cdot \text{epochs}}{\text{slowness} + \text{epochs}}. \tag{5.34}$$

By fitting the test curve to this function, we can extract the *slowness* parameter, which quantifies how fast the network reduces the error during training. Mathematically, the *slowness* value corresponds to the number of epochs necessary to reach half of the maximum accuracy. Hence, the lower the *slowness*, the faster the training. In our simulations, we perform the fit on the first 100 epochs. The graphical representation of the *plateau curve* is given in the Figure 5.6.

Figure 5.5(a) reports test curves and related *slowness* fits for  $10 \times 256$  ReLU networks, trained on the MNIST data set. The red and blue curves refer to GRAPES and SGD-based training respectively, with the same learning rate  $\eta = 0.001$ . The testing curve for the GRAPES model saturates at a substantially higher accuracy plateau compared with those of the SGD models. Furthermore, the test curve for GRAPES rises much earlier and in a steeper manner — leading to a consistently smaller *slowness* parameter — compared with the test curves of the networks trained with SGD. This demonstrates that the key for improving the convergence lies in the non-uniform modulation of the error signal. Table 5.2 shows that GRAPES exhibits the described improvements in accuracy and convergence rate under a wide range of network configurations.

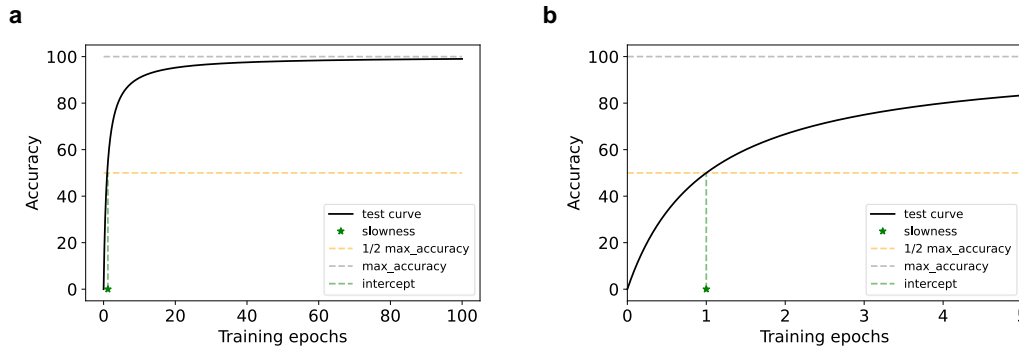


Figure 5.6: Graphical representation of the *Plateau equation for learning curves*: (a) Plateau curve for 100 epochs. From the plot one can easily observe that that  $\text{max\_accuracy} = 100$ . (b) Zoom on the initial 5 epochs. The intercept clearly shows that the slowness value is  $s = 1$ .

By design, the modulation factor for GRAPES is bounded in the interval  $[1,2]$ . This range has been empirically determined by varying the upper and lower bounds and choosing the interval that allows for the most significant improvements with respect to standard SGD. Table 5.3 shows the test accuracy for different ranges of the modulation factor.

Previous work in Ref. [Ba and Caruana, 2014] empirically showed that, as the number of trainable parameters in deep neural networks increases, the network performance initially improves and then saturates. We demonstrated that GRAPES is affected by performance saturation to a lesser extent than SGD. Figures 5.5(b,c,d) show the accuracy results for models with increasing layer size, together with the corresponding slowness value. As the average of the modulation factor is larger than one (see Figure 5.3), one needs to ensure that the training improvements are not solely due to a greater mean of the learning rate. Therefore, for each layer size, we perform a fine-grained learning rate search both for SGD and for GRAPES. We vary the learning rate from  $\eta = 0.001$  to  $\eta = 0.5$ . Note that a further increase of the learning rate ( $\eta > 0.5$ ) leads to instability and deteriorated accuracy. Figure 5.5(c) shows that for each learning rate and layer size pair GRAPES achieves better performance than SGD, with the improvements becoming increasingly more substantial as the learning rate is decreased. The optimal learning rate is  $\eta = 0.1$  for SGD and  $\eta = 0.05$  for GRAPES. With such values of  $\eta$  we draw a cross section of the bar plots along the learning rate axis, as shown in Figure 5.5(b). We observe that for each layer size GRAPES outperforms SGD in terms of final accuracy. Moreover, the accuracy results indicate a rising trend for both SGD and GRAPES as the network layer size increases. Importantly, for GRAPES the rising trend saturates later than for SGD. Furthermore, as shown in Figures 5.5 and 5.9, if a learning rate smaller than the optimized one is used, GRAPES shows a robust rising trend for different network depth and layer sizes, whereas with SGD the accuracy either saturates or deteriorates for increasing network complexity.

In addition, as shown in Figure 5.8(d), GRAPES exhibits a much faster convergence compared with SGD. Unlike SGD, especially for small learning rate values, GRAPES benefits from a greater network complexity and converges even faster when deeper networks are used, indicating that GRAPES enhances the most relevant weight updates. Therefore, GRAPES not only improves the model performance and scalability, but it also provides a useful tool to mitigate issues such as lower accuracy and slower convergence rate that arise when the learning rate is not carefully optimized.

### 5.2.5 Performance under various learning rules and data sets

GRAPES can be combined with a wide range of momentum-based optimizers (*e.g.*, NAG, rmsprop) and credit assignment strategies (*e.g.*, FA, DFA, IFA). When combined with DFA, the computation of the modulation factor requires a modification. Since in DFA the propagation of the error occurs

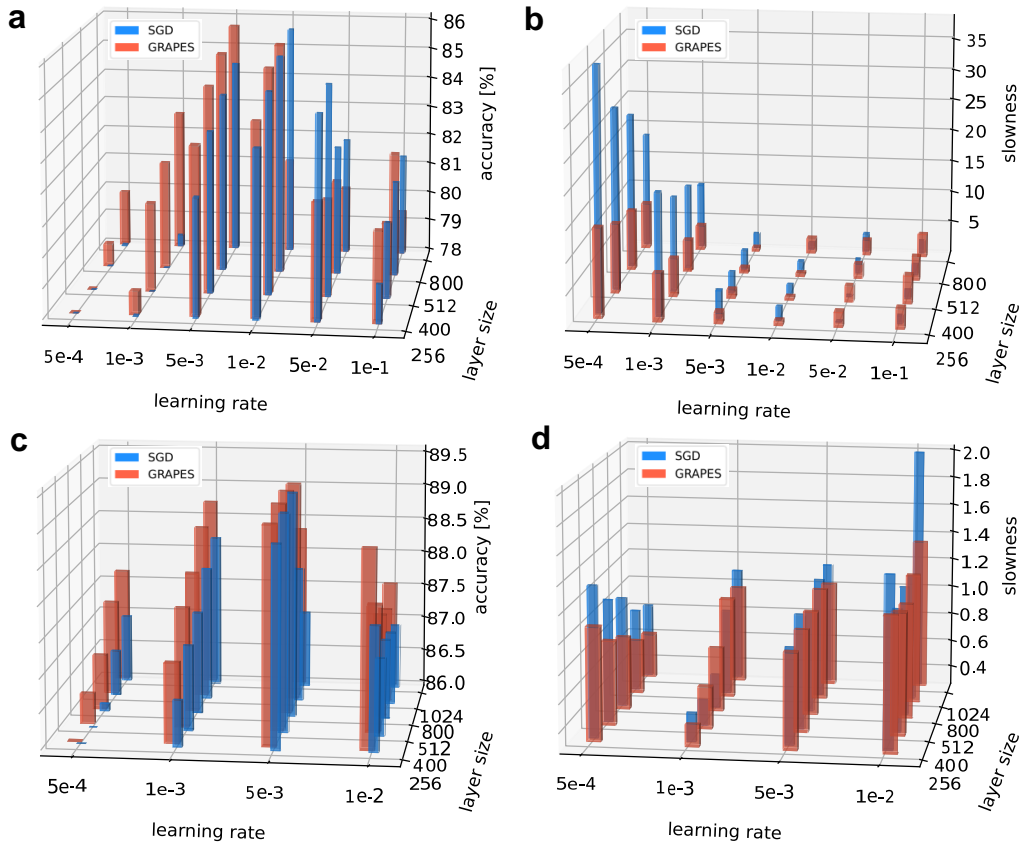


Figure 5.7: **GRAPES applied to FA and DFA on Extended and Fashion MNIST.** Test accuracy and convergence rate in terms of slowness value for 3 hidden layer ReLU networks, with 10% dropout, trained with FA on the Extended MNIST and DFA on the Fashion MNIST dataset, as a function of the layer size and the learning rate. The slowness parameter is computed by fitting the initial 100 epochs. The accuracy for each run is computed as the mean of the test accuracy over the last 10 training epochs. The reported result is the mean over the accuracy of 10 independent runs. (a) Test accuracy and (b) convergence rate for FA on Extended MNIST. (c) Test accuracy and (d) convergence rate for DFA on Fashion MNIST. For visualization purposes the bases of the SGD and GRAPES bars are slightly shifted from each other. The actual learning rates and layer sizes are the same for both and are reported in the axes' labels.

Layers	Act	DO	Def	Train epochs	acc [%] SGD	acc [%] GRAPES	acc [%] SGD augm $\eta$	s SGD	s GRAPES	s SGD augm $\eta$
3 × 256	ReLU	0%	No	200	97.46±0.09	<b>97.95±0.07</b>	97.81±0.06	0.36±0.02	<b>0.17±0.01</b>	0.27±0.01
3 × 256	ReLU	10%	No	200	97.43±0.09	<b>98.25±0.08</b>	97.90±0.09	0.46±0.01	<b>0.21±0.01</b>	0.32±0.02
3 × 256	ReLU	25%	No	200	97.15±0.09	<b>98.11±0.03</b>	97.81±0.09	0.55±0.04	<b>0.27±0.02</b>	0.38±0.04
10 × 256	ReLU	0%	No	600	97.21±0.12	<b>98.48±0.05</b>	97.34±0.08	0.22±0.02	<b>0.04±0.01</b>	0.14±0.01
10 × 256	ReLU	10%	No	600	98.16±0.06	<b>98.72±0.11</b>	98.18±0.07	0.99±0.05	<b>0.05±0.01</b>	0.59±0.06
10 × 256	ReLU	0%	Yes	600	98.81±0.06	<b>99.56±0.03</b>	99.01±0.07	1.27±0.09	<b>0.11±0.01</b>	0.78±0.05
10 × 512	ReLU	0%	No	300	97.41±0.06	<b>98.51±0.06</b>	97.50±0.12	0.16±0.01	<b>0.03±0.01</b>	0.11±0.01
10 × 512	ReLU	10%	No	300	98.11±0.12	<b>98.78±0.06</b>	98.19±0.04	0.60±0.06	<b>0.04±0.01</b>	0.33±0.02
10 × 512	ReLU	0%	Yes	300	98.43±0.18	<b>99.54±0.04</b>	98.82±0.05	0.84±0.08	<b>0.09±0.01</b>	0.53±0.04
3 × 256	tanh	0%	No	200	98.04±0.11	<b>98.20±0.07</b>	98.11±0.08	0.15±0.01	<b>0.08±0.00</b>	0.11±0.00
3 × 256	tanh	10%	No	200	98.16±0.13	<b>98.42±0.07</b>	<b>98.42±0.05</b>	0.16±0.00	<b>0.10±0.00</b>	0.13±0.00
3 × 256	tanh	25%	No	200	97.99±0.06	<b>98.37±0.03</b>	98.26±0.06	0.17±0.01	<b>0.12±0.00</b>	0.15±0.01
10 × 256	tanh	0%	No	600	98.10±0.07	<b>98.43±0.04</b>	98.15±0.10	0.09±0.001	<b>0.04±0.00</b>	0.07±0.00
10 × 256	tanh	10%	No	600	98.44±0.02	<b>98.62±0.01</b>	98.45±0.04	0.13±0.01	<b>0.06±0.00</b>	0.10±0.01
10 × 256	tanh	0%	Yes	600	99.33±0.04	<b>99.38±0.04</b>	99.35±0.06	0.32±0.02	<b>0.11±0.01</b>	0.22±0.01
10 × 512	tanh	0%	No	300	98.12±0.10	<b>98.56±0.05</b>	98.22±0.06	0.09±0.01	<b>0.04±0.00</b>	0.07±0.01
10 × 512	tanh	10%	No	300	98.33±0.04	<b>98.59±0.03</b>	98.40±0.06	0.11±0.01	<b>0.05±0.01</b>	0.09±0.01
10 × 512	tanh	0%	Yes	300	99.12±0.12	<b>99.37±0.08</b>	99.20±0.14	0.30±0.03	<b>0.10±0.01</b>	0.19±0.01

Table 5.2: **Test accuracy and convergence rate on the MNIST dataset for networks trained with BP and SGD optimizer, with and without GRAPES modulation.** In the “Def” column we indicate whether the elastic deformation scheme [Cireřan et al., 2010] was used. “DO” stands for dropout. The learning rate is  $\eta = 0.001$  with ReLU activation and  $\eta = 0.01$  with tanh activation. The column marked with “augm  $\eta$ ” indicate that the learning rate has been uniformly multiplied by a factor equal to the mean of the local modulation factor of GRAPES. The reported result is the average and standard deviation of the best test accuracy over five runs. The GRAPES modulation in most cases outperforms the classic SGD. Note that, even in the case where GRAPES is used with a smaller learning rate than SGD (network configuration with 10 hidden-layers and tanh activations, where  $\eta = 0.01$  for SGD and  $\eta = 0.001$  for GRAPES; see Table 5.11 for details on the learning rate), it provides a better convergence rate.

directly from the output layer to each hidden layer, the dimensionality of the error terms is different with respect to BP. Therefore, in order to incorporate the modulation factor in the error term, we compute the *importance* based on the presynaptic grouping

$$i_n^l = \sum_{\text{post}=1}^K |W_{\text{post},n}^{l+1}|, \quad (5.35)$$

where  $K$  is the number of outgoing connections from node  $n$  in layer  $l$ . We point out that while DFA solves the weight transport problem [Nokland, 2016], the propagating version of GRAPES requires to propagate the modulation factors from the output to the input layer, thereby still incurring in the weight transport requirement.

Figure 5.7 shows the improvements obtained by GRAPES in terms of accuracy and convergence rate when applied to FA and DFA over two data sets more challenging than MNIST: Fashion MNIST [Xiao et al., 2017] and Extended MNIST [Cohen et al., 2017]. We test networks with increasing layer size and with varying learning rate. We demonstrate that, when the optimal learning rate is used, for each model size GRAPES yields a better accuracy than SGD. Furthermore, similarly to BP, FA shows a rising trend of performance for increasing layer sizes. However, this trend was not observed with DFA. In terms of convergence rate, GRAPES combined with FA strongly mitigates the degradation for small learning rates. With DFA, GRAPES has a better slowness value than SGD for almost all models, both for large and small learning rates. Table 5.4 shows that comparable improvements in accuracy and slowness are obtained under a wide range of network settings. Furthermore, in Tables 5.5, 5.6, 5.7, 5.8 we report the performance of GRAPES applied



Range of modulation	Optimized learning rate	Test accuracy [%]
[1,2]	0.05	<b>98.53±0.06</b>
[1,3]	0.01	<b>98.53±0.06</b>
[1,5]	0.005	98.46±0.04
[0,2]	0.05	98.51±0.06
[0,5]	0.0005	98.43±0.05

Table 5.3: Test error on the MNIST dataset for networks trained with BP and SGD, with GRAPES modulation and different value range for the modulation factor. The model architecture consists of 4 hidden layers, with 256 ReLU units each. The models are trained for 200 epochs with 10% dropout rate. The accuracy for each run is computed as the mean of the test accuracy over the last 10 training epochs. The reported result is mean and standard deviation over the accuracy of 10 independent runs. The best results are obtained with the modulation factor in the ranges [1, 2] and [1, 3].

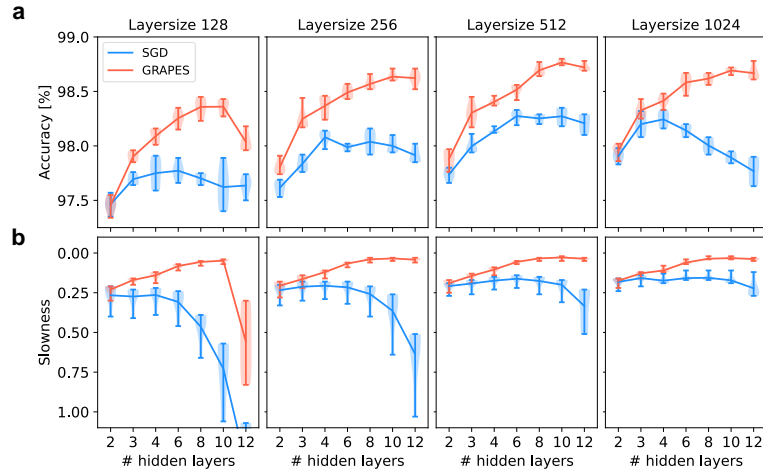


Figure 5.8: (a) Test accuracy and (b) convergence rate in terms of slowness of networks with different layer size as a function of the number of hidden layers. The slowness parameter is computed by fitting the initial 100 epochs. The shaded area around the standard deviation bar shows the distribution of the results of five runs with a violin plot.

on several feed-forward models proposed in Ref. [Nokland, 2016] and Ref. [Frenkel et al., 2019]. These results demonstrate that, when GRAPES is applied on top of momentum-based optimizers, in most cases it leads to better accuracy than such optimizers in their original formulation.

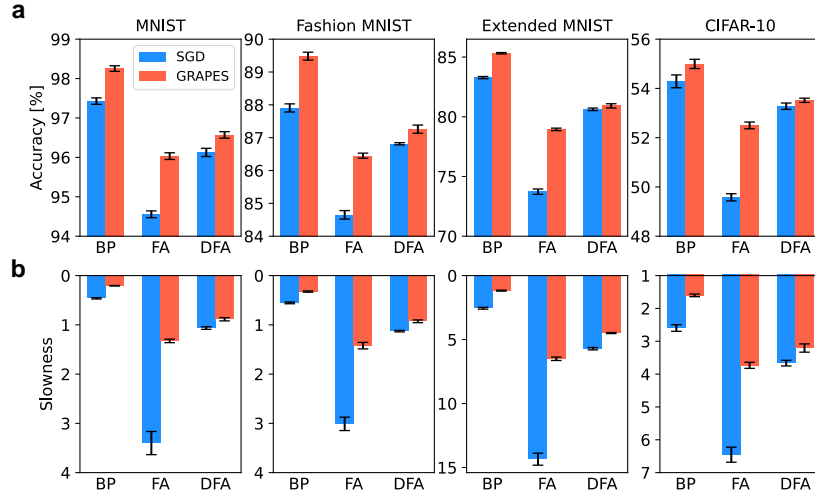


Figure 5.9: (a) Test accuracy and (b) convergence rate in terms of slowness value for  $3 \times 256$  ReLU networks, with 10% dropout, trained with BP, FA and DFA on MNIST, Fashion MNIST, Extended MNIST, and CIFAR-10 data sets. The slowness parameter is computed by fitting the initial 100 epochs. The results are mean and standard deviation over 5 independent runs.

Finally, we test the performance of the convolutional version of GRAPES on residual networks [He et al., 2016a] trained on CIFAR-10 [Krizhevsky et al., a] and CIFAR-100 [Krizhevsky et al., b]. Specifically, we trained a residual nine-layer architecture similar to [M., 2019] with the Adam optimizer, with and without GRAPES. Neither weight decay nor weight normalization is applied. When GRAPES is used, the modulation is applied after the batch normalization and the nonlinearity. The results are reported in Table 5.10. The learning rate is optimized separately for Adam and GRAPES (see Table 5.9). The best accuracy per dataset is reported in bold. Overall, GRAPES combined with Adam outperforms the standard Adam optimizer also on a residual architecture.

### 5.2.6 Mitigation of catastrophic forgetting

Catastrophic forgetting refers to the phenomenon affecting neural networks by which the process of learning a new task causes a sudden and dramatic degradation of the knowledge previously acquired by the system [French, 1999]. This represents a key limitation of ANNs, preventing the successful reproduction of continual learning occurring in the human brain [Muñoz-Martín et al., 2019, Kemker et al., 2018]. Some of the most successful proposed approaches that enable lifelong learning rely on replay. This scheme involves fine-tuning models with old inputs [Castro et al., 2018] or their related compressed representations [Hayes et al., ]. While replay is a biologically plausible mechanism, its application to the training of ANNs introduces additional computational steps and modifies the input sequence to include information about previous instances. Here, we show that the application of GRAPES mitigates, to a certain extent, the effects of catastrophic forgetting in ANNs without introducing additional steps or replaying data of previous instances.

To analyze catastrophic forgetting in a sequence of supervised learning tasks, we have adopted the protocol proposed in Ref. [Kirkpatrick et al., 2017]. For each task, we randomly generate a permutation pattern of a fraction of image pixels and we apply it to all the training and testing samples of the MNIST data set. We perform the training sequentially for all tasks for a fixed number of epochs, and, after each training epoch, we test the network performance of all the previously learnt tasks. First, we follow the protocol for the permuted MNIST proposed in the standardized benchmark Avalanche [Lomonaco et al., 2021], which involves the permutation of all image pixels, therefore no overlap is present among tasks (Figure 5.12a and b). Then, we use a custom task setup in which we regulate the fraction of permuted pixels per task, thereby exploring how GRAPES can

exploit overlapping pixels among tasks. For each task, 600 random permutations are applied (see Figure 5.12c and d). Both without and with overlapping pixels between the tasks, we observe that, compared with SGD, the drop in accuracy observed after learning each new task is considerably reduced when GRAPES is applied. In the latter case, the fraction of overlapping pixels among tasks reduces the accuracy degradation and its variability across different runs. Furthermore, the models, both with SGD and GRAPES, are able to exploit information from previous tasks, such that for each new task the test accuracy after one epoch is higher than for the previous tasks. Such accuracy is always better for GRAPES than for SGD. We used the same learning rate for GRAPES and SGD ( $\eta = 0.001$ ). If a larger learning rate is used, the performance degradation of SGD with respect to GRAPES further worsens (see Figure 5.10). We remark that the average learning rate for GRAPES is higher than that of SGD, so we could expect a higher performance degradation with respect to SGD. However, the importance-based modulation of the updates mitigates catastrophic forgetting more effectively than a uniform change of the learning rate.

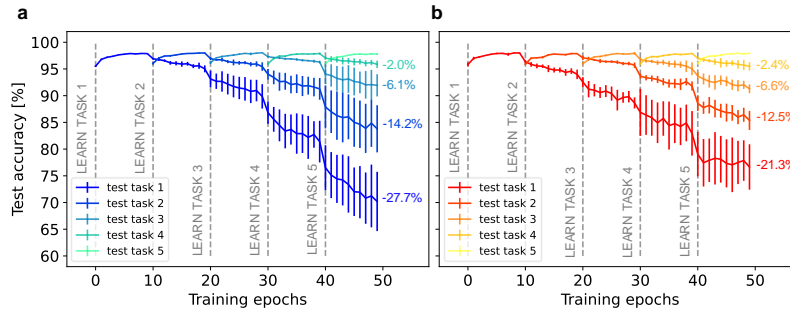


Figure 5.10: Results for the permuted MNIST protocol with  $\eta = 0.1$ . **(a)** Test accuracy for SGD. **(b)** Test accuracy for GRAPES. The results are mean and standard deviation over 5 independent runs.

Furthermore, we analyzed the effect of GRAPES under a second interesting aspect of incremental learning: the generalization to unseen tasks. Following the paradigm proposed in Ref. [Sodhani et al., 2020], we compute the *per-task-future-accuracy*, by testing the model performance on tasks it has not been trained on yet. We initially used the same protocol with 600 permutations as in Figures 5.12(c) and 5.12(d), hence different tasks are only slightly correlated with each other (the total number of pixels is 784). Figure 5.12(e) shows that in most cases the networks trained with GRAPES show better generalization capability on unseen permutations. The absolute accuracy is, though, very low. Thus, we decreased the number of permutations to 300, leading the tasks to have a stronger correlation with each other. The results are reported in Figure 5.12(f). We observe that both SGD and GRAPES achieve well above-chance level accuracy. The generalization capability increases with the number of tasks learnt. Consistently with the results obtained with 600 permutations, GRAPES in most cases leads to higher accuracy than SGD. Therefore, both in the case of almost uncorrelated and partially correlated tasks, GRAPES proves to be more effective in achieving knowledge transfer to future tasks compared with SGD. We ascribe this remarkable result mainly to two properties of GRAPES. First, GRAPES enhances the updates related to a subset of parameters based on their importance. At each new task, such subset may vary, thus the learning focuses on different groups of synapses, thereby better preserving knowledge on the old tasks. Secondly, as shown in Figure 5.11, the weights learnt with GRAPES are distributed with a larger variance in comparison to the weights learnt with SGD. We speculate that such distribution might be more robust to performance degradation when the network is trained on a sequence of tasks.

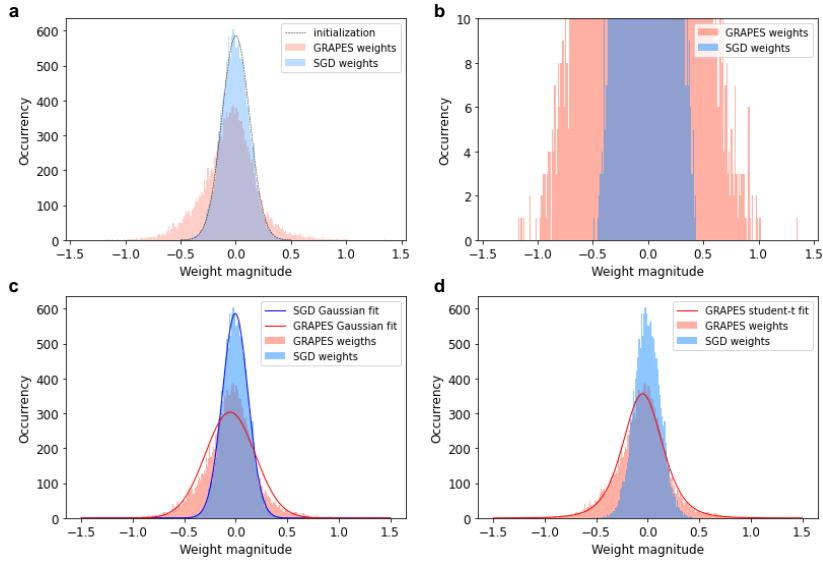


Figure 5.11: Weight distribution and fit for the first hidden layer of a 12 hidden layer model. Hidden layer size is 128 and activation function is ReLU. (a) Distribution of the weights after training with SGD (red) or GRAPES (blue). The black dotted line shows the weight initialization. The weight distribution obtained with SGD remains significantly closer to the weight initialization than with GRAPES. (b) Zoom of panel (a) along the y axis. The weights trained with GRAPES show a longer tail and more outliers with respect to the weights trained with SGD. (c) The weights trained with SGD follow a normal distribution. The weights trained with GRAPES cannot be fitted well with a Gaussian shape. In particular the weights on the tail cannot be captured by the fit. (d) The weights trained with GRAPES have a distribution that follows more closely a student-t distribution than a Gaussian distribution. The student-t distribution belongs to the class of the heavy-tailed distributions, similarly to the lognormal distribution that has been experimentally measured in synaptic strength in neural circuits of mammalian brain.

### 5.2.7 Application of GRAPES to biologically inspired neural networks

SNNs are neural network models that attempt to mimic the complex neuronal dynamics of the mammalian brain [Ghosh-Dastidar and Adeli, 2009]. Moreover, the development of SNNs is driven by the ultimate goal to implement embedded neuromorphic circuits, with high parallelism, low-power consumption, fast inference, event-driven processing, and online learning [Carrillo et al., 2013, Pfeiffer and Pfeil, 2018]. Given its biological inspiration, GRAPES holds great potential to boost the performance of SNNs. We apply GRAPES on SNN architectures implemented through the spiking neural unit (SNU) approach [Wozniak et al., 2020], which unifies SNNs with recurrent ANNs by abstracting the dynamics of a LIF spiking neuron [Gerstner et al., 2014b] into a simple recurrent ANN unit. SNUs may operate as SNNs, with a step function activation, or as more conventional RNNs, with continuous activations. The non-spiking variant is called soft SNU (sSNU).

We trained both SNU and sSNU models on temporal data derived from the MNIST data set. To that end, we encoded the MNIST handwritten digit examples into spikes using the rate coding method as described in [Wozniak et al., 2020]. The depth of the network for optimal performance was found to be 3 hidden layers for SNU, and 2 hidden layers for sSNU. Figure 5.13(a) reports the accuracy results. For both models, GRAPES surpasses the classification accuracy obtained with SGD for different layer sizes. Furthermore, GRAPES renders the networks robust against hyperparameter choice and model complexity. As can be seen in Figure 5.13(b) for SNUs, the convergence of SGD-based training is heavily affected by changes in the magnitude of the learning rate  $\eta$ . As  $\eta$  is decreased, the number of training epochs needed to trigger efficient learning dra-

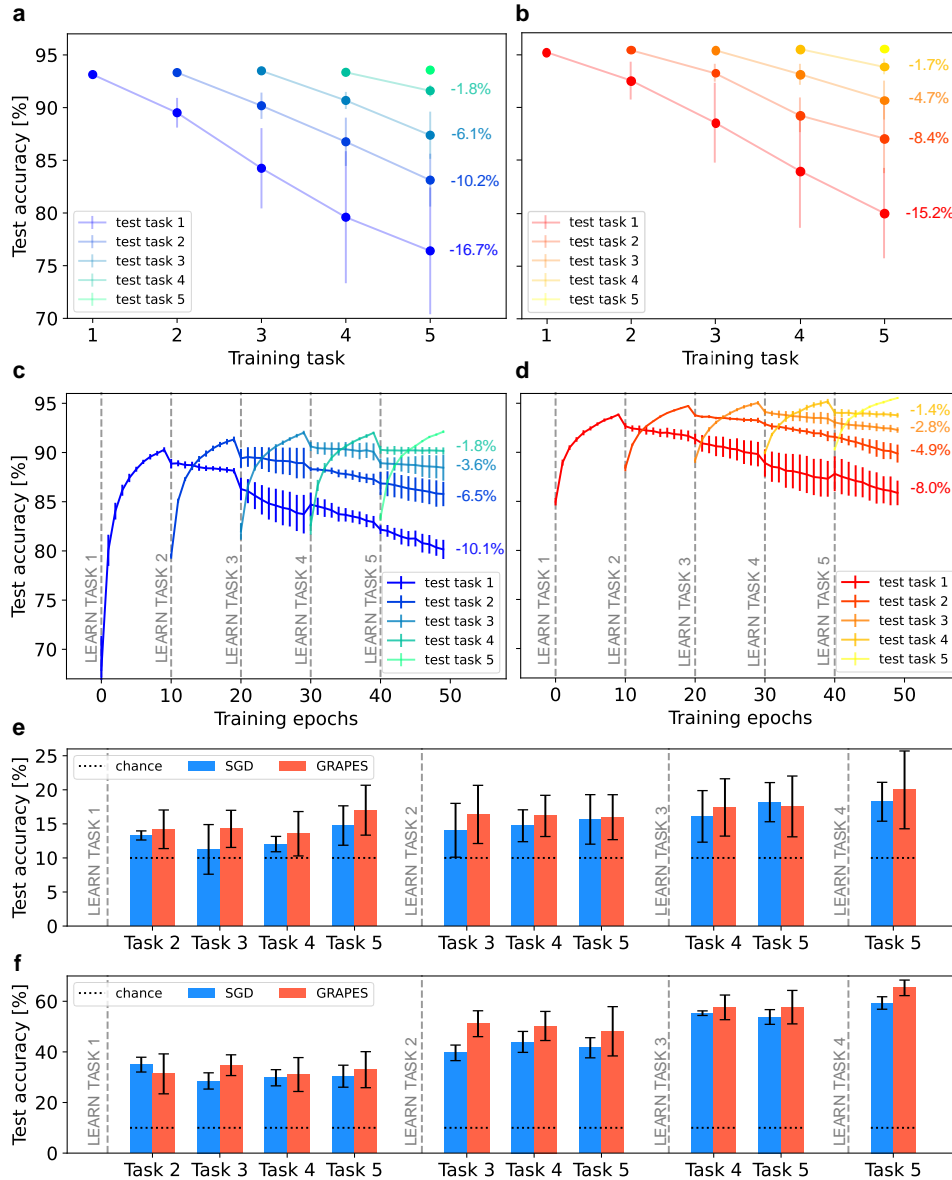


Figure 5.12: **Mitigation of catastrophic forgetting.** Following the protocol in Ref. [Kirkpatrick et al., 2017] for catastrophic forgetting, we trained ReLU networks with SGD with and without GRAPES on a sequence of tasks. Each task is defined by a random pattern of  $N_p$  pixel permutations, which is applied to all MNIST train and test images. The networks are trained sequentially on each task for  $N_e$  epochs. At the end of each training epoch, the networks are tested on both the task they are being trained on as well as the tasks they have already learnt (e.g., while learning task 1 they are tested only on task 1, and while learning task 2 they are tested on both task 2 and 1 to observe performance degradation on task 1). Panels (a) and (b) show the resulting test curves for  $2 \times 256$  networks trained on the Avalanche benchmark with  $N_p = 28 \times 28$  permutations (i.e., no overlap between tasks) and  $N_e = 1$  epoch per task. For the optimization, we used SGD with momentum with default parameters. The results are mean and standard deviation over ten independent runs. Panels (c) and (d) show the resulting test curves for  $3 \times 256$  networks trained with  $N_p = 600$  permutations (i.e., small overlap between tasks) and  $N_e = 10$  epoch per task. For the optimization, we used SGD without momentum. Panels (e) and (f) report the per-task-future-accuracy ([Sodhani et al., 2020]) on unseen tasks obtained with the same task setup as in (c) and (d). The tasks are defined by  $N_p = 600$  (panel e) and  $N_p = 300$  (panel f) pixel permutations, respectively. The networks are first trained on each task for  $N_e = 10$  epochs and then tested on all the unseen tasks (e.g., after learning task 1, the per-task-future-accuracy is reported for unseen tasks 2,3,4,5).

matically rises. When GRAPES is introduced, the model reaches well above-chance performance in only a few epochs. Furthermore, as illustrated for sSNUs in Figure 5.13(c), SGD struggles in triggering learning of networks with increasing depth, requiring almost 100 epochs to start effective training of 6 hidden layer networks. GRAPES overcomes this issue, by enabling the deep models to converge with a lower number of epochs.

### 5.2.8 Behaviour of GRAPES under hardware constraints

Training ANNs on hardware accelerators implies certain performance degradation due to a number of hardware-specific constraints, such as noisy synaptic updates, limited range of synaptic weights [Youhui Zhang et al., 2016], and their update frequency and resolution [Pfeil et al., 2012]. We empirically demonstrate that GRAPES mitigates the accuracy degradation. Inspired by the approach in [Nandakumar et al., 2018], we investigate the effect of granularity and stochasticity associated with weight-updates. First we establish a *reference* performance for a 1-hidden layer network, trained on the MNIST data set with full precision (FP) arithmetic (64-bits) and no noise. Specifically, after FP training, the network achieves a test accuracy of 97.08% with SGD and 97.23% with GRAPES. Then, we apply fixed  $n$ -bit granularity and stochasticity on the weight update as described below in *Simulation details*. Figure 5.14(a) reports the classification accuracy for different granularity levels and noise amplitudes. In the absence of noise, the accuracy is close to the reference FP value for 4,6,8-bits precision, while for 2-bits granularity an accuracy drop of 1.5% is observed. As noise with increasing amplitude is added, the model accuracy progressively deteriorates. Such degradation is robustly mitigated when GRAPES is applied. Figure 5.14(b) shows the test curve for a noise standard deviation of  $\sigma = 1.5\epsilon$ . For all weight granularities, GRAPES leads to a higher classification accuracy over the entire training period.

We remark that the implemented hardware constraints share many aspects with biological circuits: the synaptic transmission is affected by noise, the signal is quantized and neurons have a limited fan-in/fan-out. Interestingly, GRAPES retains many similarities with biological processes, such as synaptic integration, synaptic scaling and heterosynaptic plasticity. We therefore envision that the brain might exploit such mechanisms in order to overcome the limitations due to the mentioned constraints, and, in fact could turn out to endow the networks with the ability to take advantage of noise to improve the performance. Our analysis is consistent with previous work providing evidence that synaptic integration combined with the intrinsic noise stochasticity of Poisson trains enhances the computational capabilities of spiking networks on pattern classification tasks [Li et al., 2020]. In conclusion, our findings suggest that incorporating GRAPES in on-chip training algorithms could pave the way for pivotal progress in learning algorithms for bio-inspired hardware and, in particular, for neuromorphic chips.

## 5.3 Discussion

Inspired by the biological mechanism of non-linear synaptic integration and local synaptic strength regulation, we proposed GRAPES (Group Responsibility for Adjusting the Propagation of Error Signals), a novel optimizer for both ANN and SNN training. GRAPES relies on the novel concept of *node importance*, which quantifies the responsibility of each node in the network, as a function of the local weight distribution within a layer. Applied to gradient-based optimization algorithms, GRAPES provides a simple and efficient strategy to dynamically adjust the error signal at each node and to enhance the updates of the most relevant parameters. Compared with optimizers such as momentum, our approach does not need to store parameters from previous steps, avoiding additional memory penalty. This feature makes GRAPES more biologically plausible than momentum-based optimizers as neural circuits cannot retain a substantial fraction of information from previous states [Pehlevan and Chklovskii, 2019].

We validated our approach with ANNs on five static data sets (MNIST, CIFAR-10, CIFAR-100, Fashion MNIST and Extended MNIST) and with SNNs on the temporal rate-coded MNIST. We successfully applied GRAPES to different training methods for supervised learning, namely BP,

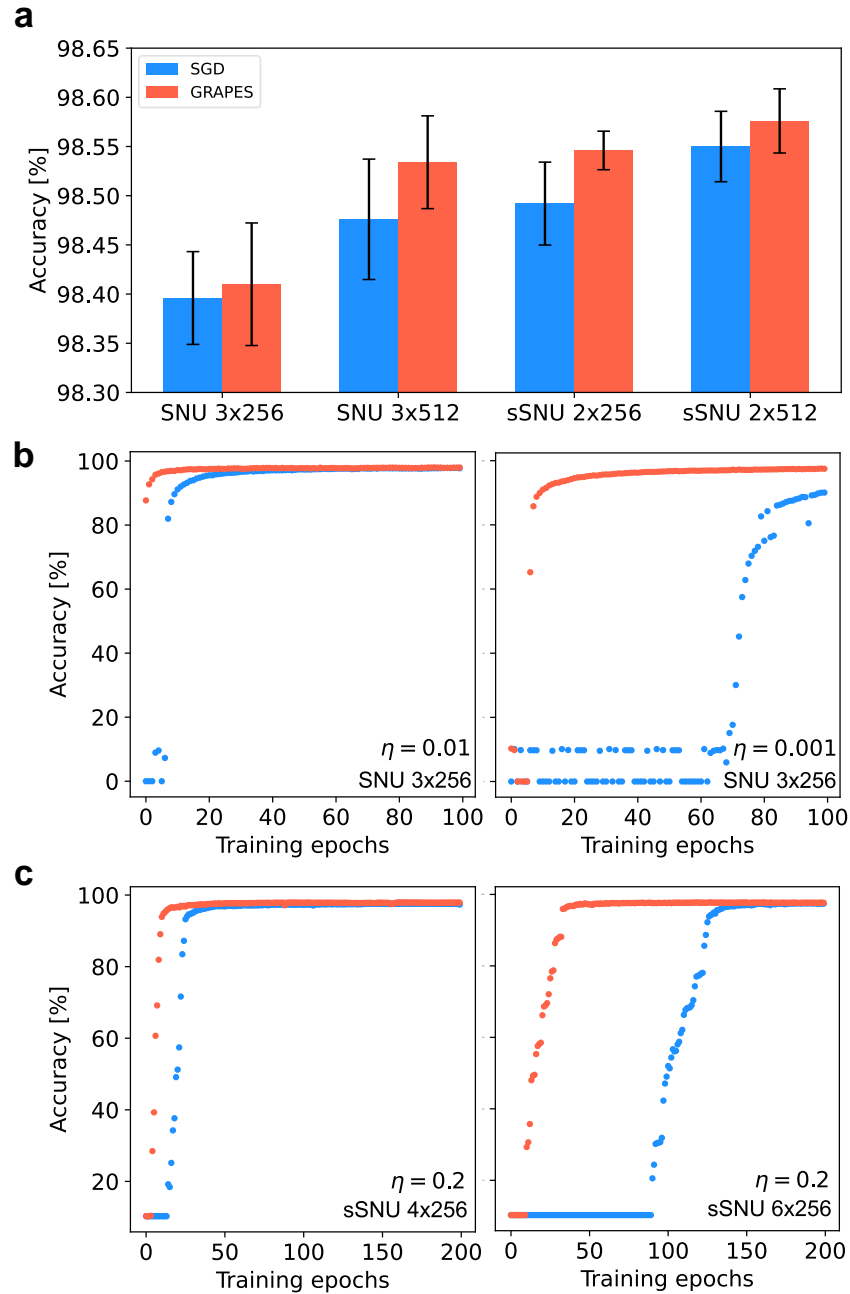


Figure 5.13: **Experiments with spiking neural networks.** (a) Test accuracy for SNU and sSNU networks with 3 and 2 hidden layers, respectively. The results are mean and standard deviation of the test accuracy after training over 5 independent runs. (b) Test curves for SNU  $3 \times 256$  networks with decreasing values for the learning rate  $\eta$ . Left hand-side:  $\eta = 0.01$ . Right hand-side:  $\eta = 0.001$ . The initial 100 training epochs are shown. (c) Test curves for sSNU networks with increasing number of hidden layers. Left hand-side:  $4 \times 256$  networks. Right hand-side:  $6 \times 256$  networks. In (b) and (c) the curves correspond to a single run.

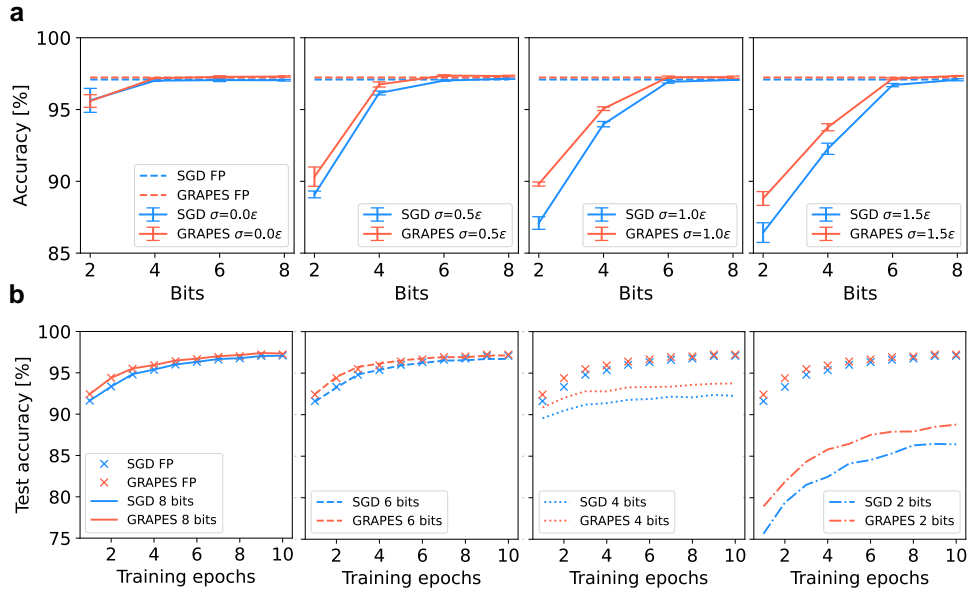


Figure 5.14: (a) Final test accuracy for 1-hidden layer networks as a function of the  $n$ -bit granularity where  $n \in \{2, 4, 6, 8\}$  and noise standard deviation  $\sigma \in \{0.0, 0.5, 1.0, 1.5\}\epsilon$ . The red curves correspond to networks trained with GRAPES, while the blue curves correspond to networks trained with SGD. The results are mean and standard deviation over 5 independent runs. (b) Test curves for single run as a function of the training epochs with fixed noise standard deviation  $\sigma = 1.5\epsilon$ . Training with full precision (FP), 8,6,4 and 2 bits precision are compared.

FA and DFA, and to different optimizers, *i.e.*, SGD, RMSprop and NAG. We demonstrated that the proposed weight-based modulation leads to higher classification accuracy and faster convergence rate both in ANNs and SNNs. Next, we showed that GRAPES addresses major limitations of ANNs, including mitigation of performance saturation for increasing network complexity [Ba and Caruana, 2014] and catastrophic forgetting [French, 1999].

We suggest that these properties stem from the fact that GRAPES effectively combines in the error signal information related to the response to the current input with information on the internal state of the network, independent of the data sample. Indeed, GRAPES enriches the synaptic updates based on the *input-driven responsibility* with a modulation relying on the *network-driven responsibility*, which indicates the potential impact that a node would have on the network’s output, independently on the input. Such training strategy endows networks trained with GRAPES with the ability to achieve convergence in a lower number of epochs, as the training is not constrained to information depending only on the presented training samples. For the same reason, such networks present better generalization capability than SGD both when tested on the learnt tasks and when presented with unseen tasks in continual learning scenarios. In this context, we identify parallelism with plasticity types in the brain. The change in synaptic strength in response to neuronal activity results from the interplay of two forms of plasticity: homosynaptic and heterosynaptic. Homosynaptic plasticity occurs at synapses active during the input induction, thus is input-specific and associative, as the *input-driven responsibility*. Instead, heterosynaptic plasticity concerns synapses that are not activated by presynaptic activity and acts as an additional mechanism to stabilize the networks after homosynaptic changes [Chistiakova et al., 2014, Caya-Bissonnette, 2020]. Therefore, similarly to the *network-driven responsibility*, heterosynaptic plasticity does not exhibit strict input specificity.

Our algorithm appears to have certain similarity with existing normalization schemes [Bird et al., 2021] and the Winner-take-all computational primitive [Kaski and Kohonen, 1994]. However, as GRAPES relies on the concept of network-driven responsibility, its main computations are based on synaptic strength rather than on synaptic activity. In particular, the operations of summing the weight



strength and the normalization in Eq. 5.2 require that neurons communicate their synaptic strength. We remark that the normalization operation is introduced to project the importance vector into a meaningful interval, in which values larger than 1 lead to enhancement of the weight updates, and smaller than 1 imply weakening of the updates. The same numerical results can be achieved by performing the vector normalization by dividing by the mean of the importance values (or the total sum of the importance values), and later multiplied by  $2 \times \frac{\text{mean}(i)}{\text{max}(i)}$  (resp. by  $2 \times \frac{\text{sum}(i)}{\text{max}(i)}$ ) rather than 2. Furthermore, previous work has contemplated the possibility that the neurons communicate synaptic strengths. For instance, in [Fiete et al., 2010], the authors propose the summed-weight limit rule for heterosynaptic long-term depression: when the summed weight of synapses into (or out of) a neuron exceeds a limit, all the incoming (or outgoing) synapses to that neuron are weakened. Such mechanism implies that synapses communicate information about the values of the synaptic weights to the postsynaptic node, and such information is used to modulate the synaptic weights in a non-local fashion. A second example is the theory of retroaxonal signals and neural marketplace proposed in [Harris, 2008, Lewis and Harris, 2014]. Experimental evidence suggests that neurons are capable of carrying retroaxonal signals through molecules known as neurotrophic factors, which can encode information both on synaptic strength and on its temporal derivative. Such information is used to promote or hinder the consolidation of synaptic weights' changes. The theory of the neural marketplace builds on the mechanism of retroaxonal signalling and proposes a model for how networks of neurons in the brain self-organize into functional networks. Both the neural marketplace theory and the GRAPES algorithm rely on the propagation of information about the weights and their changes, hence the two frameworks present several analogies. First, the retroaxonal signals control the plasticity of synapses by modulating the synaptic updates. Similarly, the importance vector is used in GRAPES to modulate the weight changes prescribed by BP. Secondly, the retroaxonal signals carrying information on weight strength and weight change travel slowly; similarly, the information in GRAPES is only applied after each batch. Third, both the information propagated through neurotrophin and the importance in GRAPES do not depend on gradients. Finally, the theory in [Lewis and Harris, 2014] introduces the concept of *worth* of a cell, which measures the usefulness of the cell's output, and is defined as the worsening in network performance if the cell were to die. A cell is inactivated if all its incoming connections are zeroes, hence the worth of a cell is related to the strength of the incoming synapses to the cell. Therefore, the worth can be related to the concept of *node importance* in GRAPES. In conclusion, the underlying ideas of GRAPES are inspired by the concepts of node importance, error modulation and communication of weight strength, which are supported by experiments investigating the role of dendritic integration, synaptic scaling and retroaxonal signalling. While the biological inspiration is grounded on these mechanisms, only the high-level concept of GRAPES-like plasticity modulation is compatible with plasticity modulation principles observed in neural circuits.

The benefits of GRAPES stem from the adjustment of the error signal. The nonuniform distribution of the modulation factor, combined with the propagation to upstream layers, allows GRAPES to greatly enhance a subset of synaptic updates during training. Hence, small groups of synapses are enabled to strengthen or weaken to a much larger extent than with SGD. From preliminary investigation, GRAPES appears to convey the network weights toward more biologically plausible distribution, specifically heavy-tailed distributions [Buzsáki and Mizuseki, 2014, Iyer et al., 2013, Teramae and Fukai, 2014, Song et al., 2005]. In Figure 5.11, we compare the layer-wise weight distribution of networks trained with SGD and with GRAPES, both initialized with a normal distribution. After training with SGD, the weight distribution is still close to a Gaussian. Instead, after GRAPES-optimization we observed that, particularly in the first hidden layers of deep networks, the weight distribution does not follow a Gaussian shape, is wider, and is characterized by long tails.

Recent electrophysiological studies have revealed that the amplitudes of EPSPs between cortical neurons are not distributed as Gaussians but obey a long-tailed pattern, typically log-normal [Buzsáki and Mizuseki, 2014, Song et al., 2005]. Such firing pattern implies that some synapses are very strong while many synapses are weak ("strong-sparse and weak-dense" networks) [Teramae and Fukai, 2014]. Heavy-tailed distributions were shown to lead to important network

properties: faster transient responses, higher dynamical range and lesser sensitivity to random fluctuations in synaptic activity [Iyer et al., 2013]. This suggests that the long-tail distribution found at many physiological and anatomical levels in the brain is fundamental to structural and functional brain organization [Buzsáki and Mizuseki, 2014]. From preliminary investigation, GRAPES seems to convey the network weights toward a more biologically plausible distribution. Fig. 5.11 illustrates this phenomenon. This implies that the properties of faster learning and greater robustness to noise exhibited by networks trained with GRAPES could stem from the modulation of the gradients towards adjusting the weights to a long-tailed distribution. We suggest that the properties exhibited by GRAPES could stem from such weight distribution. Ongoing work in our group is currently seeking a more comprehensive understanding of this phenomenon.

Remarkably, our results suggest that GRAPES offers a promising strategy for mitigating the performance degradation caused by hardware-related constraints, such as noise and reduced precision, as discussed in section 5.2.8. We highlight that these constraints reflect biological circuits in many aspects, as the synaptic transmission is affected by noise and the neural signal is quantized. Interestingly, GRAPES retains many similarities with biological processes. We, therefore, envision that the biological mechanisms underlying GRAPES may play a central role in overcoming the limitations associated with hardware-related constraints. Furthermore, we suggest that such brain-inspired features are at the origin of the benefits of GRAPES on biologically-inspired models. Indeed, we have demonstrated that GRAPES not only improves BP-based training of standard ANNs, but additionally boosts substantially the performance of networks trained with biologically plausible credit assignment strategies, such as FA and DFA, and networks relying on the dynamics of spiking neurons. Both the FA algorithms and the SNN models are crucial steps towards bridging biological plausibility and machine learning. However, at the present stage, they can only achieve a limited performance compared to ANNs trained with BP [Bartunov et al., 2018, Pfeiffer and Pfeil, 2018]. For instance, as shown in the Results section, both the FA and SNNs approaches suffer from lower accuracy and convergence rate compared with BP, and SNNs training is severely affected by changes in network complexity and hyperparameters. Thanks to an efficient modulation of the error signal which enhances the updates of the most important parameters, GRAPES reduces the impact of such limitations, thereby narrowing the gap between the performance of bio-inspired algorithms and standard ANNs.

To conclude, our findings indicate that incorporating GRAPES and, more generally, brain-inspired local factors in the optimization of neural networks paves the way for pivotal progress in the performance of biologically inspired learning algorithms and in the design of novel neuromorphic computing technologies.

## 5.4 Methods

**MNIST data set** We train FCNNs with 3 and 10 hidden layers, each consisting of either 256 or 512 hidden nodes. The activation functions chosen for the hidden layers are rectified linear unit (ReLU) or hyperbolic tangent (tanh). The output activation is softmax with cross-entropy loss function. With ReLU hidden nodes the weights are initialized according to [He et al., 2015], with tanh units according to [Glorot and Bengio, 2010]. The batch size is fixed to 64. The learning rate  $\eta$  is optimized for the different models, separately for SGD and GRAPES, and is kept fixed during training. Table 5.11 reports the optimized learning rate as well as detailed simulation settings for all simulations. We investigate the performance both without dropout [Hinton et al., 2012] and with moderate dropout rates of 10% and 25%. We also show the accuracy improvement of the models that are trained with an augmented version of the data set, built by applying both affine and elastic deformation on the training set, similarly as proposed in Ref. [Cireşan et al., 2010]. To compute the accuracy, we train the models on the training set and after each epoch we test the performance on the test set. Following the strategy in Ref. [Cireşan et al., 2010], we report the best test accuracy throughout the entire simulation. For all settings, we average the result over five independent runs.

**Scalability to complex networks** We train networks with layer sizes ranging from 128 to 1024 and with depth from 2 to 12 hidden layers. Each network is trained with ReLU, 10% dropout rate and for 200 epochs. The learning rate is kept constant to  $\eta = 0.001$ . As in the previous section, we report the best testing accuracy obtained throughout the entire training, averaged over five runs.

**Catastrophic forgetting** In the simulations based on the Avalanche library, the networks, containing  $2 \times 256$  ReLU hidden layers, are trained with momentum for one epoch on each task. In our task setting, for each task, 600 random permutations are applied. We train  $3 \times 256$  ReLU FCNN networks on the training samples using shuffling and minibatch processing for a fixed number of epochs. We set the number of training epochs per task to 10. In both settings, we use a constant learning rate  $\eta = 0.001$ . We introduce a dropout rate of 10%. For all settings, we average the result over five independent runs.

**Spiking Neural Networks** We train SNU and sSNU networks on the rate-coded MNIST data set. The dynamics of the units and the training protocol are the same as described in Ref. [Wozniak et al., 2020]. The only difference with respect to the original SNU network in Ref. [Wozniak et al., 2020] is the introduction of a soft-reset to smoothen the training process of the spiking units. We performed a grid-search for the hyperparameters. For the SNU networks the optimal configuration is 3-hidden layer and constant learning rate  $\eta = 0.1$ . For sSNU models the optimal configuration is 2-hidden layers with constant  $\eta = 0.2$ . The networks are trained for 200 epochs. The number of steps of input presentation is set to  $N_s = 20$  during train and  $N_s = 300$  during test. The mean and standard deviation of the final accuracy are computed over 5 runs.

**Simulation details** We train 1-hidden layer networks on MNIST data set. The hidden layer has 250 sigmoid neurons, the output activation is softmax and the loss is cross-entropy. No dropout is introduced. The networks are trained for 10 training epochs with fixed learning rate  $\eta = 0.4$ . The weight distribution after training of the FP network lies in the range  $[-1,1]$ . To investigate the effect of fixed  $n$ -bit granularity, we assume a similar final weight distribution range as that from the floating-point simulation. Hence, we cover the weight range  $[-1,1]$  in  $2^n - 2$  steps, with  $n \in \{2, 4, 6, 8\}$ . We use the  $n$ -bit granularity for the forward pass. We perform the backward pass on a floating point copy of the network and apply the granularity after the update. The stochasticity is applied on the weight update as white noise with mean zero and variance  $\sigma = k\epsilon$ , where  $\epsilon = 1/(2^n - 2)$  and  $k \in \{0.0, 0.5, 1.0, 1.5\}$ . Both the final test accuracy and the test curves are averaged over five runs.

**Programming** The learning experiments of the ANN simulations were run using custom-built code in Python3 with the Numpy library. The SNU and sSNU-based simulations were performed using the original TensorFlow code from the Supplementary Material of [Wozniak et al., 2020].

Data set	Train scheme	Layers	Activation	DO	Def	Train epochs	acc [%] SGD	acc [%] GRAPES	s SGD	s GRAPES
MNIST	FA	3 × 256	ReLU	10%	No	200	94.56±0.10	<b>96.03±0.10</b>	3.40±0.26	<b>1.33±0.04</b>
	FA	10 × 256	ReLU	10%	No	600	-	<b>91.30±0.48</b>	-	-
	FA	3 × 256	tanh	10%	No	200	97.09±0.03	<b>97.75±0.07</b>	0.22±0.01	<b>0.16±0.01</b>
	FA	10 × 256	tanh	10%	No	600	89.52±2.58	<b>94.54±0.74</b>	40.72±21.20	<b>1.63±0.94</b>
	DFA	3 × 256	ReLU	10%	No	200	96.13±0.12	<b>96.57±0.09</b>	1.06±0.03	<b>0.89±0.04</b>
	DFA	3 × 256	tanh	10%	No	200	<b>97.89±0.06</b>	97.87±0.04	0.17±0.01	<b>0.15±0.00</b>
	DFA	10 × 256	tanh	10%	No	300	97.93±0.05	<b>97.94±0.02</b>	0.19±0.01	<b>0.15±0.01</b>
Fashion MNIST	BP	3 × 256	ReLU	10%	No	200	87.91±0.14	<b>89.48±0.14</b>	0.55±0.02	<b>0.32±0.02</b>
	BP	10 × 256	ReLU	10%	No	600	89.21±0.23	<b>90.29±0.10</b>	1.06±0.10	<b>0.19±0.02</b>
	BP	3 × 256	tanh	10%	No	200	89.20±0.12	<b>89.97±0.13</b>	0.18±0.01	<b>0.15±0.01</b>
	BP	10 × 256	tanh	10%	No	600	89.84±0.08	<b>90.47±0.13</b>	0.16±0.01	<b>0.16±0.01</b>
	FA	3 × 256	ReLU	10%	No	200	84.65±0.15	<b>86.45±0.09</b>	3.01±0.15	<b>1.42±0.07</b>
	FA	10 × 256	ReLU	10%	No	600	-	<b>65.69±1.85</b>	-	<b>34.66±8.33</b>
	FA	3 × 256	tanh	10%	No	200	87.75±0.07	<b>88.70±0.08</b>	0.31±0.01	<b>0.20±0.01</b>
	FA	10 × 256	tanh	10%	No	600	<b>85.78±2.21</b>	85.76±1.98	3.03±1.62	<b>2.52±1.35</b>
	DFA	3 × 256	ReLU	10%	No	200	86.81±0.04	<b>87.26±0.14</b>	1.13±0.02	<b>0.93±0.03</b>
	DFA	3 × 256	tanh	10%	No	200	88.42±0.08	<b>88.53±0.08</b>	0.20±0.01	<b>0.17±0.01</b>
DFA	10 × 256	tanh	10%	No	300	<b>88.89±0.04</b>	88.78±0.13	0.27±0.01	<b>0.19±0.01</b>	
Extended MNIST	BP	3 × 256	ReLU	10%	No	200	83.28±0.10	<b>85.32±0.07</b>	2.55±0.08	<b>1.18±0.04</b>
	BP	10 × 256	ReLU	10%	No	400	84.08±0.21	<b>86.35±0.13</b>	5.46±0.30	<b>0.44±0.02</b>
	BP	3 × 256	tanh	10%	No	200	86.39±0.12	<b>86.40±0.12</b>	0.73±0.02	<b>0.38±0.01</b>
	BP	10 × 256	tanh	10%	No	400	<b>86.85±0.07</b>	86.39±0.12	<b>0.61±0.02</b>	1.44±0.09
	FA	3 × 256	ReLU	10%	No	200	73.73±0.25	<b>78.95±0.12</b>	14.35±0.52	<b>6.50±0.15</b>
	FA	10 × 256	ReLU	10%	No	400	-	<b>43.62±8.24</b>	-	-
	FA	3 × 256	tanh	10%	No	200	82.29±0.11	<b>83.74±0.09</b>	1.88±0.03	<b>1.07±0.02</b>
	FA	10 × 256	tanh	10%	No	300	71.58±0.86	<b>78.22±0.39</b>	22.73±0.75	<b>2.87±0.16</b>
	DFA	3 × 256	ReLU	10%	No	200	80.62±0.12	<b>80.91±0.20</b>	5.71±0.10	<b>4.49±0.05</b>
	DFA	3 × 256	tanh	10%	No	200	84.30±0.14	<b>84.36±0.15</b>	1.14±0.03	<b>0.87±0.02</b>
DFA	10 × 256	tanh	10%	No	200	<b>83.95±0.11</b>	83.88±0.15	1.76±0.03	<b>1.05±0.02</b>	
CIFAR-10	BP	3 × 256	ReLU	10%	No	600	54.29±0.29	<b>55.00±0.21</b>	2.60±0.11	<b>1.60±0.04</b>
	BP	10 × 256	ReLU	10%	No	500	42.18±0.44	<b>52.74±0.47</b>	13.09±5.81	<b>8.05±1.87</b>
	BP	3 × 256	tanh	10%	No	600	43.81±0.15	<b>49.43±0.17</b>	3.49±0.33	<b>1.82±0.13</b>
	BP	10 × 256	tanh	10%	No	500	42.77±0.34	<b>56.38±0.31</b>	4.12±0.59	<b>1.98±0.11</b>
	FA	3 × 256	ReLU	10%	No	400	49.58±0.16	<b>52.50±0.15</b>	6.45±0.26	<b>3.73±0.10</b>
	FA	3 × 256	tanh	10%	No	400	36.94±0.14	<b>42.45±0.34</b>	15.62±2.71	<b>10.35±0.39</b>
	FA	10 × 256	tanh	10%	No	400	-	<b>33.51±1.02</b>	-	<b>12.69±6.90</b>
	DFA	3 × 256	ReLU	10%	No	300	53.28±0.14	<b>53.52±0.10</b>	3.67±0.10	<b>3.21±0.14</b>
	DFA	3 × 256	tanh	10%	No	300	39.60±0.17	<b>41.47±0.12</b>	12.57±1.39	<b>10.93±1.54</b>
	DFA	10 × 256	tanh	10%	No	300	35.29±0.14	<b>39.58±0.20</b>	20.48±4.12	<b>16.64±1.14</b>

Table 5.4: Test accuracy and convergence rate of models with BP, FA and DFA learning schemes, obtained with and without GRAPES modulation on a variety of data sets. The reported result is the average and standard deviation of the best test accuracy over 5 simulations. Empty fields in the accuracy columns indicate no convergence, while empty fields in the slowness columns indicate that the shape of the test curve could not be meaningfully fitted with the plateau equation. The learning rate is  $\eta = 0.001$  with ReLU activation and  $\eta = 0.01$  with tanh activation. The number of training epochs for each simulation is set based on the saturation of the models' performance. The GRAPES modulation in most cases outperforms the classic SGD.

Model	Activation	Results in [Nokland, 2016]	RMSprop	GRAPES + RMSprop
7 × 240	tanh	2.16±0.13%	<b>2.05±0.09</b> %	<b>2.05±0.10</b> %
1 × 800	tanh	1.59±0.04%	1.53±0.08%	<b>1.50±0.04</b> %
2 × 800	tanh	1.60±0.06%	1.51±0.05 %	<b>1.47±0.05</b> %
3 × 800	tanh	1.75±0.05%	1.65±0.07 %	<b>1.62±0.04</b> %
4 × 800	tanh	1.92±0.11%	1.86±0.06 %	<b>1.79±0.06</b> %
2 × 800	sigmoid	1.67±0.03 %	<b>1.56±0.06</b> %	1.62±0.05 %
2 × 800	ReLU	1.48±0.06%	<b>1.50±0.05</b> %	<b>1.50±0.06</b> %

Table 5.5: Test error on the MNIST dataset for networks trained with BP and RM-Sprop optimizer on the settings from [Nokland, 2016], with and without GRAPES modulation. The reported result is the mean and standard deviation over the final test accuracy of 10 runs. The GRAPES modulation in most cases outperforms the classic RMSprop.

Model	Activation	Results in [Nokland, 2016]	RMSprop	GRAPES + RMSprop
7 × 240	tanh	2.20±0.13%	3.34±0.10%	<b>2.56±0.13</b> %
1 × 800	tanh	1.68±0.05%	1.65±0.06%	<b>1.61±0.04</b> %
2 × 800	tanh	1.64±0.03%	1.67±0.01%	<b>1.65±0.04</b> %
3 × 800	tanh	1.66±0.09%	<b>1.57±0.07</b> %	1.60±0.07 %
4 × 800	tanh	1.70±0.04%	1.79±0.07%	<b>1.64±0.08</b> %
2 × 800	sigmoid	1.82±0.10%	<b>1.78±0.07</b> %	1.80±0.03%
2 × 800	ReLU	1.74±0.10%	<b>1.66±0.02</b> %	1.69±0.05%

Table 5.6: Test error on the MNIST dataset for networks trained with FA and RM-Sprop optimizer on the settings from [Nokland, 2016], with and without GRAPES modulation. The reported result is the mean and standard deviation over the final test accuracy of 5 runs. The GRAPES modulation in most cases outperforms the classic RMSprop.

Model	Activation	Results in [Nokland, 2016]	RMSprop	GRAPES + RMSprop
7 × 240	tanh	2.32±0.15%	2.20±0.07%	<b>2.11±0.05</b> %
2 × 800	tanh	1.74±0.08%	1.65±0.04%	<b>1.64±0.03</b> %
3 × 800	tanh	1.70±0.04%	<b>1.64±0.08</b> %	1.65±0.07%
4 × 800	tanh	1.83±0.07%	<b>1.69±0.04</b> %	<b>1.73±0.07</b> %
2 × 800	sigmoid	1.75±0.04%	<b>1.84±0.06</b> %	1.85±0.04%
2 × 800	ReLU	1.70±0.06%	1.71±0.07%	<b>1.64±0.06</b> %

Table 5.7: Test error on the MNIST dataset for networks trained with DFA and RM-Sprop optimizer on the settings from [Nokland, 2016], with and without GRAPES modulation. The reported result is the mean and standard deviation over the final test accuracy of 5 runs. The GRAPES modulation in most cases outperforms the classic RMSprop.

Model	Activation	Dropout	Results in [Frenkel et al., 2019]	NAG	GRAPES + NAG
1 × 500	tanh	0.0	1.72±0.08%	1.71±0.003%	<b>1.68±0.003%</b>
1 × 500	tanh	0.1	1.55±0.03%	1.69±0.02%	<b>1.64±0.02%</b>
1 × 500	tanh	0.25	1.64±0.06%	1.70±0.02%	<b>1.63±0.02%</b>
1 × 1000	tanh	0.0	1.76±0.06%	<b>1.69±0.003%</b>	1.72±0.002%
1 × 1000	tanh	0.1	1.58±0.03%	1.68±0.02%	<b>1.63±0.02%</b>
1 × 1000	tanh	0.25	1.70±0.06%	1.68±0.03%	<b>1.62±0.03%</b>
2 × 500	tanh	0.0	1.62±0.12%	<b>1.56±0.003%</b>	1.61±0.004%
2 × 500	tanh	0.1	1.61±0.05%	1.62±0.02%	<b>1.57±0.02%</b>
2 × 500	tanh	0.25	1.84±0.05%	1.76±0.02%	<b>1.60±0.02%</b>
2 × 1000	tanh	0.0	1.67±0.07%	1.58±0.002%	<b>1.50±0.002%</b>
2 × 1000	tanh	0.1	1.85±0.06%	1.77±0.04%	<b>1.49±0.01%</b>
2 × 1000	tanh	0.25	2.31±0.06%	2.04±0.04 %	<b>1.62±0.02%</b>

Table 5.8: **Test error on the MNIST dataset for networks trained with BP and NAG optimizer on the settings from [Frenkel et al., 2019], with and without GRAPES modulation.** The accuracy for each run is computed as the mean of the test accuracy over the last 10 training epochs. The reported result is the mean and standard deviation over the accuracy of 10 independent runs. The GRAPES modulation in most cases outperforms the classic NAG.

Optimizer	CIFAR-10		CIFAR-100	
	Accuracy [%] $\eta = 1e - 3$	Accuracy [%] $\eta = 1e - 1$	Accuracy [%] $\eta = 1e - 3$	Accuracy [%] $\eta = 1e - 1$
Adam	76.07	83.08	45.52	54.98
Adam+ GRAPES	76.96	80.25	44.65	52.65

Table 5.9: **Test accuracy on the CIFAR-10 and CIFAR-100 datasets for residual networks trained with BP and Adam optimizer, with and without GRAPES modulation, for learning rates larger and smaller than the optimized one .** We trained the models with learning rates  $\eta = 1e - 3$  and  $\eta = 1e - 1$ , respectively smaller and larger than the optimized learning rate  $\eta = 1e - 2$ . The results confirm that  $\eta = 1e - 2$  is the optimal learning rate for both SGD and GRAPES. The reported results are for a single run.

Optimizer	CIFAR-10	CIFAR-100
Adam	84.78±0.20	58.20±0.39
Adam+ GRAPES	<b>85.59±0.17</b>	<b>58.85±0.38</b>

Table 5.10: **Test accuracy on the CIFAR-10 and CIFAR-100 datasets for CNNs trained with BP and Adam, with and without GRAPES modulation.** The network is a nine-layer residual architecture. The learning rate is decayed by 90% every 50 epochs and the initial learning rate is  $\eta = 1e - 2$ . The models are trained for 250 epochs. The accuracy for each run is computed as the mean of the test accuracy over the last 10 training epochs. The reported result is the mean and standard deviation over the accuracy of 10 independent runs. The bold font indicates the best performance for each dataset.

Experiment	Figure/ Table	Training scheme	Dataset	Model	Opt.	Act.	DO [%]	bs	#ep.	#trials	$\eta$ SGD	$\eta$ GRAPES
$M^l$ Dy- namics	Fig.3	BP	MNIST	input layer: $784 \times 256$ 10 hidden fc: $256 \times 256$ output layer: $256 \times 10$	SGD	ReLU	0.1	64	200	1	0.001	0.001
Slowness fit	Fig.4a	BP	MNIST	input layer: $784 \times 256$ 10 hidden fc: $256 \times 256$ output layer: $256 \times 10$	SGD	ReLU	0.1	64	200	1	0.001	0.001
Scalability	Fig.4b,c	BP	MNIST	input layer: $784 \times ls$ 3 hidden fc: $ls \times ls$ output layer: $ls \times 10$ $ls \in \{256, 400, 512, 800\}$	SGD	ReLU	0.1	64	200	10	$\in$ [0.001, 0.5]	$\in$ [0.001, 0.1]
Scalability	Fig.5a,b	FA	Extended MNIST	input layer: $784 \times ls$ 3 hidden fc: $ls \times ls$ output layer: $ls \times 10$ $ls \in \{256, 400, 512, 800\}$	SGD	ReLU	0.1	64	200	10	$\in$ [0.001, 0.5]	$\in$ [0.001, 0.5]
Scalability	Fig.5c,d	DFA	Fashion MNIST	input layer: $784 \times ls$ 3 hidden fc: $ls \times ls$ output layer: $ls \times 10$ $ls \in \{256, 400, 512, 800, 1024\}$	SGD	ReLU	0.1	64	200	10	$\in$ [0.001, 0.1]	$\in$ [0.001, 0.1]
Convolutional residual model	Tab.1	BP	CIFAR-10, CIFAR-100	As in [M., 2019], except without the max pool after conv5 for better accuracy	Adam	ReLU	None	256	250	10	0.01	0.01
Catastrophic forgetting (Avalanche)	Fig.6a,b	BP	Permuted MNIST	input layer: $784 \times 256$ 1 hidden fc: $256 \times 256$ output layer: $256 \times 10$	mom	ReLU	None	64	1	10	0.001	0.001
Catastrophic forgetting (numpy)	Fig.6c- f	BP	Permuted MNIST	input layer: $784 \times 256$ 2 hidden fc: $256 \times 256$ output layer: $256 \times 10$	SGD	ReLU	0.1	64	10	5	0.001	0.001
Catastrophic forgetting (numpy)	Fig.S3	BP	Permuted MNIST	input layer: $784 \times 256$ 2 hidden fc: $256 \times 256$ output layer: $256 \times 10$	SGD	ReLU	0.1	64	10	5	0.1	0.1
SNU	Fig.7a	BPTT	Rate MNIST	input layer: $784 \times ls$ 2 hidden fc: $ls \times ls$ output layer: $ls \times 10$ $ls \in \{256, 512\}$	SGD	Step	None	64	200	5	0.1	0.1
sSNU	Fig.7a	BPTT	Rate MNIST	input layer: $784 \times ls$ 1 hidden fc: $ls \times ls$ output layer: $ls \times 10$ $ls \in \{256, 512\}$	SGD	sigmoid	None	64	200	5	0.2	0.2
Hardware constraints	Fig.S5	BP	MNIST	input layer: $784 \times 250$ 1 hidden fc: output layer: $250 \times 10$ $ls \in \{256, 512\}$	SGD	sigmoid	None	64	10	5	0.4	0.4

Table 5.11: Model and hyperparameter settings for all experiments.

## Chapter 6

# PEPITA: a Biologically Plausible Training Scheme

Supervised learning in artificial neural networks typically relies on backpropagation, where the weights are updated based on the error-function gradients and sequentially propagated from the output layer to the input layer. Although this approach has proven effective in a wide domain of applications, it lacks biological plausibility in many regards, including the weight symmetry problem, the dependence of learning on non-local signals, the freezing of neural activity during error propagation, and the update locking problem. Alternative training schemes have been introduced, including sign symmetry, feedback alignment, and direct feedback alignment, but they invariably rely on a backward pass that hinders the possibility of solving all the issues simultaneously. Here, we propose to replace the backward pass with a second forward pass in which the input signal is modulated based on the error of the network. We show that this novel learning rule comprehensively addresses all the above-mentioned issues and can be applied to both fully connected and convolutional models. We test this learning rule on MNIST, CIFAR-10, and CIFAR-100. These results help incorporate biological principles into machine learning.

### 6.1 Introduction

The backpropagation algorithm (BP) has proven to reach impressive results in training Artificial Neural Networks (ANNs) on a broad range of complex cognitive tasks including speech recognition, image classification [LeCun et al., 2015], and playing board games [Silver et al., 2016]. However, BP has been criticized for relying on a biologically unrealistic strategy of *synaptic credit assignment*, *i.e.*, estimating how much each parameter has contributed to the output error [Crick, 1989, Whittington and Bogacz, 2019, Lillicrap et al., 2020]. In particular, a few aspects of BP appear to be at odds with neurobiology. (i) The same weights are used both in the feedforward and in the feedback pathway, raising the weight symmetry or *weight transport problem* [Burbank and Kreiman, 2012]. (ii) The parameter updates depend on the activity of all downstream nodes, while biological synapses learn based on local signals related to the activity of the neurons they connect with [Whittington and Bogacz, 2019]. (iii) The error gradients are stored separately from the activations [Liao et al., 2016] and do not influence the activities of the nodes produced in the forward pass. Hence, during the backward pass, the network activity is frozen. In the brain, instead, the neural activity is not frozen during plasticity changes and the signals travelling through feedback connections influence the neural activities produced by forward propagation, leading to their enhancement or suppression [Lillicrap et al., 2020]. (iv) Input signals cannot be processed in an online fashion, but each sample needs to wait for both the forward and backward computations to be completed for the previous sample. This is referred to as the *update locking problem* [Jaderberg et al., 2017, Czarnecki et al., 2017]. These considerations have motivated the development of alternative methods for credit assignment, each propos-



ing solutions to some of these criticisms [Lillicrap et al., 2016, Nokland, 2016, Liao et al., 2016, Frenkel et al., 2019, Nøkland and Eidnes, 2019, Clark et al., 2021, Meulemans et al., 2021]. However, none of the training schemes designed so far is able to comprehensively address *all* the aforementioned challenges.

Here we introduce a novel learning rule that is able to train ANNs on image classification tasks without incurring in the issues described above. We name our scheme PEPITA, Present the Error to Perturb the Input To modulate Activity. PEPITA relies on perturbing the input signal based on error-related information. The difference between the network responses to the input and to its perturbed version is used to compute the synaptic updates. Specifically, the algorithm is implemented by performing two forward computations for each input sample, avoiding any backward pass and performing the updates in a layerwise feedforward fashion during the second forward pass. By avoiding the backward pass, PEPITA does not suffer from the *weight transport* problem and it partially solves the *update locking* problem. Furthermore, as the error is incorporated in the input signal, PEPITA does not freeze the neural activity to propagate and apply the modulatory signal. Finally, the update rule respects the *locality* constraints. We show that PEPITA can be formulated as a two-factor Hebbian-like learning rule. The error information is used as a global learning signal, which is consistent with biological observations of global neuromodulators that influence synaptic plasticity, and which is similar to several reinforcement learning schemes [Williams, 1992, Mazzoni et al., 1991]. As a proof-of-principle, we show that PEPITA can be successfully applied to train both fully connected and convolutional models, leading to performance only slightly worse than BP.

## 6.2 Background and related work

We review some of the training schemes proposed to solve the biologically unrealistic aspects of backpropagation. We begin by discussing learning rules relying on *unit-specific feedback*, *i.e.*, propagating the error from the output layer to each hidden layer through specific connectivity matrices. Then, we describe recent algorithms which involve delivering learning signals in alternative fashions.

### 6.2.1 Credit assignment in conventional networks

The backpropagation algorithm [Rumelhart et al., 1995] relies on a forward and a backward pass. During the forward pass, the input signal is propagated from the input layer to the output layer, where the error is computed by comparing the network’s output with the target. During the backward pass, the error flows from the top layers to the bottom layers through the same weights used in the feedforward pathway. For each synapse, the error gradient — and the associated synaptic update — is computed through recursive application of the chain rule. In the past two decades, several works investigated whether a mechanism similar to BP could be implemented by brains. [Xie and Seung, 2003] showed the equivalence of BP and Contrastive Hebbian Learning, and, in a more general setting, [Scellier and Bengio, 2017] introduced Equilibrium Propagation (EP), a learning framework for energy-based models, which shares similarities with Contrastive Hebbian Learning. EP plays a role analog to BP to compute the gradient of an objective function, and is a compelling biologically-plausible approach to compute error gradients in deep neuromorphic systems [Laborieux et al., 2021]. In the context of unsupervised learning, [Pehlevan and Chklovskii, 2019] demonstrated that biologically plausible neural networks can be derived from similarity matching objectives, relying both on Hebbian feedforward and anti-Hebbian lateral connections.

Alternative training principles have been proposed to relax the constraint of symmetric weights. Effective learning in neural networks can be achieved also when the error is backpropagated through connections that share only the sign and not the magnitude with the feedforward weights (*sign symmetry* algorithm) [Liao et al., 2016, Xiao et al., 2018] or through random fixed connections (*feedback-alignment* algorithm, FA) [Lillicrap et al., 2016]. In the latter, the feedback provided by the random matrices is able to deliver useful modulatory information for learning since the

forward connections are driven to align with the fixed feedback matrices. As an extension to FA, the *direct feedback alignment* approach (DFA) shows that useful learning information can be propagated directly from the output layer to each hidden unit through random connectivity matrices [Nokland, 2016, Refinetti et al., 2021]. In a similar vein, [Akrouf et al., 2019] builds on FA and introduces the *weight mirror* neural circuit. This approach adjusts the initially random feedback weights of FA to improve their agreement with the forward connections, yielding improved performance compared to FA.

These biologically inspired training schemes achieve performance close to BP in many pattern recognition tasks without incurring in the *weight transport* problem. However, they do not tackle the issues of non-locality, freezing of neural activity and update locking.

### 6.2.2 Credit assignment without random feedback path

Recently, learning techniques based on local error handling have been shown to successfully train ANNs while addressing the *update locking* problem [Nøkland and Eidnes, 2019, Belilovsky et al., 2020, Mostafa et al., 2018]. In these approaches, however, each layer is trained independently through auxiliary fixed random classifiers, thereby incurring both in the *weight transport* problem at the level of the classifiers and in a significant computational overhead. With the same goal, the *direct random target projection* (DRTP) algorithm has been proposed to update the parameters of the hidden layers based on the sample labels rather than the network error [Frenkel et al., 2019]. Such a strategy overcomes both the *weight transport* and the *update locking* issues without incurring in additional computational requirements. However, it requires that the activity is frozen as the modulatory information on the targets flows through the network. Furthermore, DRTP suffers from a significantly larger performance degradation with respect to BP compared to the FA algorithms. Another original approach to credit assignment known as *global error-vector broadcasting* (GEVB) avoids delivering error information through fixed random connections [Clark et al., 2021]. The GEVB learning rule performs parameter updates based on the inner product of the presynaptic activity and a global error vector. This scheme provides a performance almost on par with BP, and solves both the *weight transport* and the *locality* issues. However, it can only operate in a new class of deep neural networks, the vectorized nonnegative networks, which require each node to be represented by a vector unit with the same dimensionality as the output class. This implies a higher computational overhead, which significantly increases for datasets with a large number of classes, *e.g.*, CIFAR-100.

## 6.3 Error-driven input modulation

### 6.3.1 Overview of the proposed learning rule

Here we introduce a local plasticity rule to train ANNs via supervised learning. The training rules described so far rely on a forward pass, followed by a backward pass during which the error (BP, FA, DFA, GEVB) or the target (DRTP) travels from the output layer to each hidden unit through paths specific to each learning rule. The backward computation leads to at least two biologically implausible aspects. First, the weight updates rely on non-local information: the error coming directly from the output layer (all schemes), as well as downstream feedback connections (BP, FA). Second, during the backward pass the network activity is frozen. We circumvent both problems by proposing a training scheme that does not involve a backward pass, but rather performs two successive forward passes per input sample. The first pass is similar to any conventional training scheme. In the second forward pass, the error computed during the first pass is used to modify the input. The modulated input travels through the network, eliciting activities slightly different from those of the first pass. Such differences in node activity are used to update the network’s parameters. As the output and input dimensionalities are generally different, we use a fixed random matrix  $F$ , with zero mean and small standard deviation, to project (*i.e.*, add) the error onto the input. We refer to the first and second forward passes as “standard pass” and “modulated pass”, respectively. Figure 6.1 shows a schematic comparison between error propagation in BP, FA and

DFA and the novel configuration proposed in this paper, PEPITA. The intuition of modulating the input signal with error-related information is motivated by the existence of global modulatory signals in the brain fed from higher level to lower level connections that influence the activity at the early stages of the visual stream [Shimegi et al., 2016].

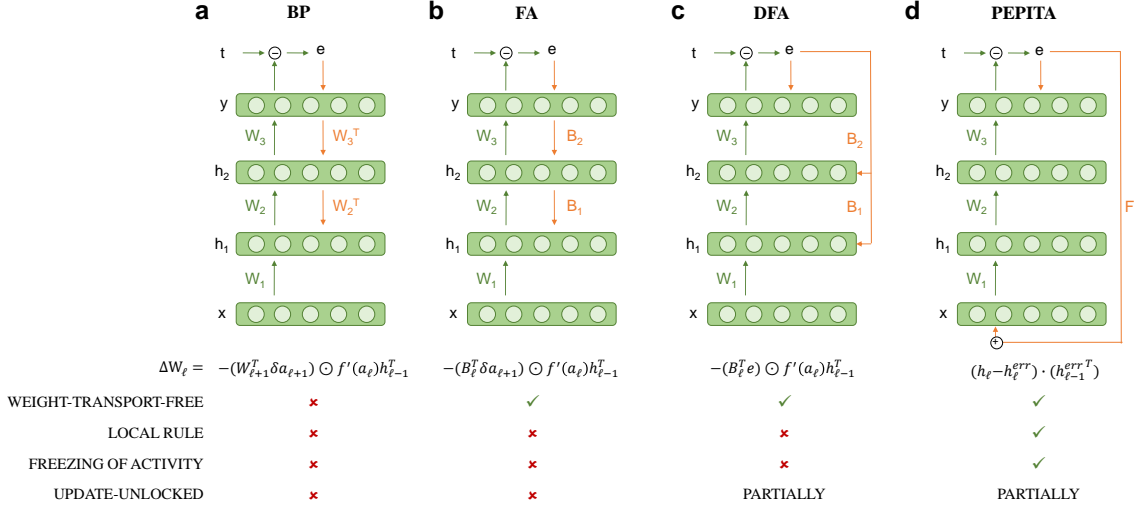


Figure 6.1: Overview of different error transportation configurations. a) Back-propagation (BP). b) Feedback-alignment (FA). c) Direct feedback-alignment (DFA). d) Present the Error to Perturb the Input To modulate Activity (PEPITA). Green arrows indicate forward paths and orange arrows indicate error paths. Weights that are adapted during learning are denoted as  $W_l$ , and weights that are fixed and random are denoted as  $B_l$  if specific to a layer (BP, FA, DFA) or  $F$  if specific to the input signal (PEPITA).

### 6.3.2 The learning rule

Given a fully connected neural network with  $L$  layers and an input signal  $x$ , in the standard pass the hidden unit and output unit activations are computed as:

$$\begin{aligned} h_1 &= \sigma_1(W_1 x), \\ h_l &= \sigma_l(W_l h_{l-1}) \quad 2 \leq l \leq L. \end{aligned} \quad (6.1)$$

where  $\sigma_l$  denotes the non-linearity at the output of the  $l^{th}$  layer and  $W_l$  denotes the matrix of weights between layer  $l-1$  and layer  $l$ . At the modulated pass, the activations are computed as:

$$\begin{aligned} h_1^{err} &= \sigma_1(W_1(x + Fe)), \\ h_l^{err} &= \sigma_l(W_l h_{l-1}^{err}) \quad 2 \leq l \leq L. \end{aligned} \quad (6.2)$$

where  $e$  denotes the network error and  $F$  denotes the fixed random matrix used to project the error on the input. We compute the error as  $e = h_L - target$ .

During the two forward passes, the network weights are equal and the difference in the node activities is solely due to the error-driven modulation of the input signal.

The weight updates are computed after (or during, see Discussion) the modulated pass and are applied in a forward fashion. Each synaptic update depends on the product between a postsynaptic term and a presynaptic term. The postsynaptic term is given by the difference in activation of the postsynaptic node between the standard and the modulated pass. The presynaptic term corresponds to the activity of the presynaptic nodes during the modulated pass.

First layer:

$$\Delta W_1 = (h_1 - h_1^{err}) \cdot (x + Fe)^T \quad (6.3)$$

Intermediate hidden layers, with  $2 \leq l \leq L - 1$ :

$$\Delta W_l = (h_l - h_l^{err}) \cdot (h_{l-1}^{err})^T \quad (6.4)$$

Output layer:

$$\Delta W_L = e \cdot (h_{L-1}^{err})^T \quad (6.5)$$

The weights of the output layer are trained as in BP since the information about the error is directly accessible at the last layer. Furthermore, we tested a modification of the algorithm and replaced the activity of the modulated pass with that of the standard pass in the presynaptic term:  $\Delta W_l = (h_l - h_l^{err}) \cdot (h_{l-1})^T$ . In the simulations, such modification did not affect the network performance.

Finally, the prescribed synaptic updates are applied depending on the chosen optimizer, as for any gradient-based optimization technique. For example, using the Stochastic Gradient Descent (SGD)-like scheme, with learning rate  $\eta$ :

$$W(t+1) = W(t) - \eta \Delta W \quad (6.6)$$

The algorithm is reviewed in the pseudocode below.

---

**Algorithm 1** Implementation of PEPITA

---

**Given:** Input ( $x$ ) and label ( $target$ )

#standard forward pass

$h_0 = x$

**for**  $\ell = 1, \dots, L$

$h_\ell = \sigma_\ell(W_\ell h_{\ell-1})$

$e = h_L - target$

#modulated forward pass

$h_0^{err} = x + Fe$

**for**  $\ell = 1, \dots, L$

$h_\ell^{err} = \sigma_\ell(W_\ell h_{\ell-1}^{err})$

**if**  $\ell < L$ :

$\Delta W_\ell = (h_\ell - h_\ell^{err}) \cdot (h_{\ell-1}^{err})^T$

**else:**

$\Delta W_\ell = e \cdot (h_{\ell-1}^{err})^T$

---

### 6.3.3 Extension to convolutional layers

The same approach can be applied to train convolutional models. However, a modification is required to account for the parameter sharing used in convolutions. As for fully connected models, PEPITA trains the convolutional models through a standard and a modulated forward pass, and the kernels are learnt based on post- and pre-synaptic related terms. For each filter, the update is computed after the modulated pass through the following steps:

1. For each element in the output map, first we compute the difference in activity between the standard and modulated pass. Then we multiply such difference by the area of the input map on which the filter was applied to generate the specific output pixel. Each of the computed products has the same dimensionality as the filter.
2. All the products computed in step 1 are summed into a single update term.

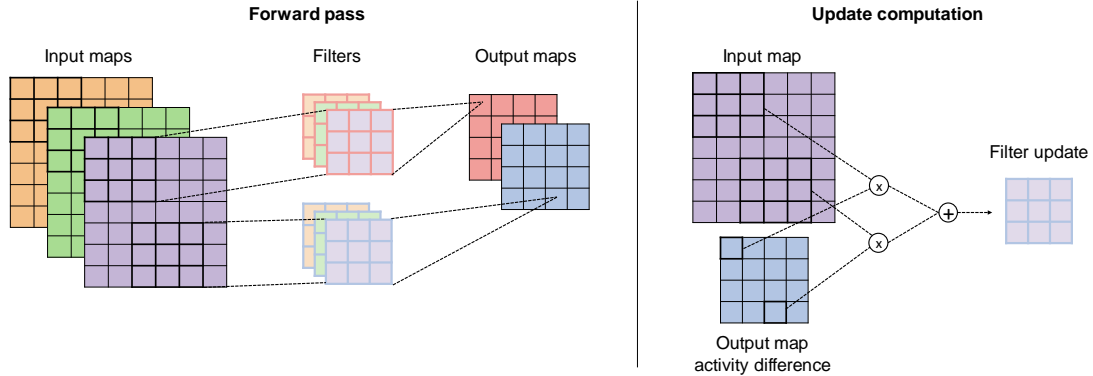


Figure 6.2: Update scheme for convolutional layers

3. The resulting term is divided by the number of summed products.
4. The computed update is applied to the filter with the chosen optimization technique, such as Eq. 6.6.

Figure 6.2 provides a schematic of the standard forward pass and the computation of the update for a single channel of a filter.

## 6.4 Results

### 6.4.1 Methods

Inspired by [Liao et al., 2016], we focus on relative differences between algorithms, not absolute performance. Each experiment is a {model, dataset} pair. We tested the PEPITA learning rule on 3 datasets: MNIST [LeCun and Cortes, 2010], CIFAR-10 [Krizhevsky et al., a] and CIFAR-100 [Krizhevsky et al., b]. We did not use any data augmentation. For each dataset, we trained both a fully connected and a convolutional model with BP, FA, DRTP and PEPITA. With BP, FA and PEPITA, we used rectified linear unit (ReLU) [Nair and Hinton, 2010] as non-linearity for the hidden layers and softmax for the output layer. With DRTP we used hyperbolic tangent as non-linearity for the hidden layers and sigmoid for the output layer. We used SGD with momentum with hyperparameter 0.9. For BP, FA and DRTP we used cross entropy loss. In the fully connected networks, we introduced dropout with drop rate of 10% after the hidden layer. In the convolutional models, we applied Max Pooling after the convolutional layer. We initialized the networks using the He initialization [He et al., 2015] for both the forward connections  $W_i$  and the projection matrix  $F$ . The input images are normalized to the interval [0,1]. We optimized the learning rate, the learning rate decay schedule, the batch size and the dropout rate separately for each experiment. We used the entire training set for the training and did not use a validation set. The network architectures and simulation details for each dataset and learning rule are reported in Table 6.1. The code for the simulations of fully connected models with PEPITA, BP, FA and convolutional models with PEPITA is provided at: <https://github.com/GiorgiaD/PEPITA>. For all the other experiments we used the code from [Frenkel et al., 2019].

### 6.4.2 Experimental results

Here, we show that the proposed learning rule successfully trains both fully connected and convolutional networks on image classification tasks. The experimental results are summarized in Table 6.3. The accuracy obtained by PEPITA for fully connected networks (Table 6.3, columns 1-3) is close to the accuracy achieved by BP and FA, and superior to that of DRTP for all experiments. Figure 6.5a shows the learning curve on the test set for the fully connected model trained on

	FULLY CONNECTED MODELS			CONVOLUTIONAL MODELS		
	MNIST	CIFAR10	CIFAR100	MNIST	CIFAR10	CIFAR100
INPUTSIZE	28×28×1	32×32×3	32×32×3	28×28×1	32×32×3	32×32×3
LAYER 1	FC1:1024	FC1:1024	FC1:1024	CONV1:32,5,1	CONV1:32,5,1	CONV1:32,5,1
LAYER 2	FC2:10	FC2:10	FC2:100	FC1:10	FC1:10	FC1:100
$\eta$ BP	0.1	0.01	0.01	0.1	0.01	0.01
$\eta$ FA	0.1	0.001	0.001	0.1	0.01	0.01
$\eta$ DRTP	0.01	0.001	0.001	0.01	0.01	0.01
$\eta$ PEPITA	0.1	0.01	0.01	0.1	0.1	0.1
$\eta$ DECAY RATE	×0.1	×0.1	×0.1	×0.1	×0.1	×0.1
DECAY EPOCH	60	60,90	60,90	10,30,50	10,30,50	10,30,50
BATCH SIZE	64	64	64	100	100	100
F STD	$0.05 \cdot 2 \sqrt{\frac{6}{\text{FANIN}}}$	$0.05 \cdot 2 \sqrt{\frac{6}{\text{FANIN}}}$	$0.05 \cdot 2 \sqrt{\frac{6}{\text{FANIN}}}$	$0.05 \cdot 2 \sqrt{\frac{6}{\text{FANIN}}}$	$0.05 \cdot 2 \sqrt{\frac{6}{\text{FANIN}}}$	$0.05 \cdot 2 \sqrt{\frac{6}{\text{FANIN}}}$
FAN IN	$28 \cdot 28 \cdot 1$	$32 \cdot 32 \cdot 3$	$32 \cdot 32 \cdot 3$	$28 \cdot 28 \cdot 1$	$32 \cdot 32 \cdot 3$	$32 \cdot 32 \cdot 3$
#EPOCHS	100	100	100	100	100	100
DROPOUT	10%	10%	10%	-	-	-
WEIGHT INIT	HE UNIF.	HE UNIF.	HE UNIF.	HE UNIF.	HE UNIF.	HE UNIF.

Table 6.1: Network architectures and settings used in the experiments. For the convolutional layers A,B,C means A feature maps of size B, with stride C.

MNIST with PEPITA. Figure 6.6 reports the t-SNE visualization of the representation learned at the hidden layer for the same model. Figure 6.3 reports PEPITA’s test curve on the CIFAR-10 dataset, compared with the test curves of BP, FA, and DRTP. Figure 6.5a and Figure 6.3 show that the training rules lead to convergence within 100 epochs. We observe an improved performance of the convolutional models trained with PEPITA compared to the fully connected networks for all datasets (Table 6.3, columns 4-6), indicating that the convolutional version of PEPITA is able to learn useful two-dimensional filters.

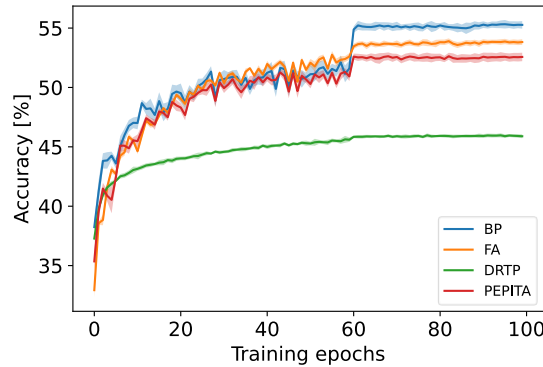


Figure 6.3: Test curve for BP, FA, DRTP and PEPITA in the experiments for the fully connected models trained on CIFAR-10 (second column of Table 6.3). The solid line is the mean over 10 independent runs. The shaded colored area shows the std over the 10 runs.

To evaluate the learning speed of the tested training schemes, we relied on the *plateau equation for learning curves* proposed in [Dellaferreira et al., 2022b]:

$$\text{accuracy} = \frac{\text{max\_accuracy} \cdot \text{epochs}}{\text{slowness} + \text{epochs}}. \quad (6.7)$$

By fitting the test curve to this function, we extract the *slowness* parameter, which quantifies how fast the network reduces the error during training. Mathematically, the *slowness* value corresponds to the number of epochs necessary to reach half of the maximum accuracy. Hence, the lower the *slowness*, the faster the training. In our simulations, we perform the fit on the first 60 epochs, as on epoch 61 the learning rate is decayed and the test curve exhibits a sudden increase (see Figure 6.3). Note that the fitting is performed on the test curve to which we add a point for the chance level (10% for CIFAR-10) at epoch 0 (see Figure 6.4). Furthermore, we compute the slowness value only for the fully connected models, as the test curve obtained with the convolutional models is challenging to fit due to a learning rate decay scheme applied already after ten training epochs (see Table 6.1). In Figure 6.4 we show the *plateau curve* fitting the learning curves of BP, FA, DRTP, and PEPITA for the fully connected model trained on CIFAR-10. Table 6.2 reports the slowness values for the fully-connected models on all datasets. PEPITA’s convergence rate is in between BP (the fastest) and FA (the slowest). DRTP has a better learning speed than FA, but it converges to a significantly lower accuracy plateau.

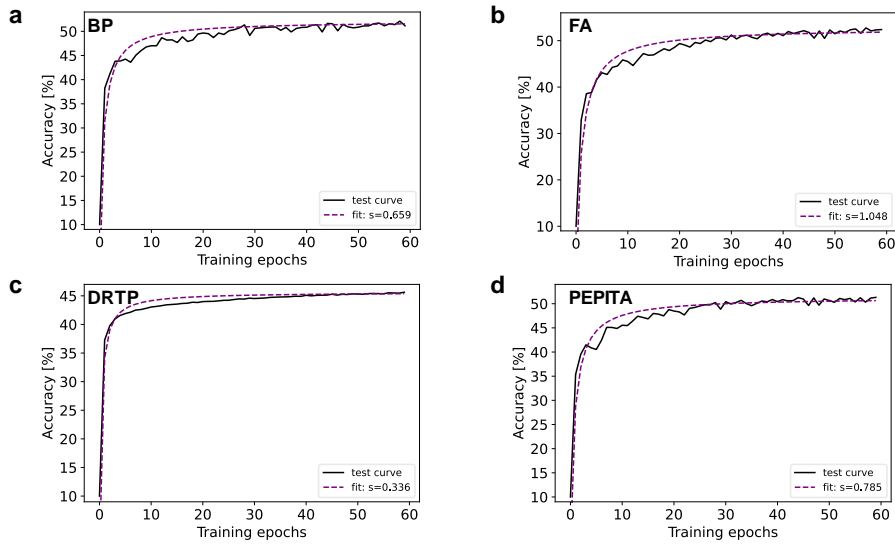


Figure 6.4: The *plateau equation for learning curves* fits the test curves. Results for the fully connected models trained on CIFAR-10 (second column of Table 6.3). The black solid line is the test curve averaged over 10 independent runs. The purple dotted line is the fitted curve from which we extract the slowness value ( $s$ ). Results for (a) BP, (b) FA, (c) DRTP, and (d) PEPITA.

	FULLY CONNECTED MODELS		
	MNIST	CIFAR10	CIFAR100
BP	<b>0.026</b>	<b>0.659</b>	<b>1.338</b>
FA	0.068	1.048	4.243
DRTP	0.063	<b>0.336</b>	4.05
PEPITA	0.049	0.785	2.552

Table 6.2: Convergence rate in terms of slowness value obtained by BP, FA, DRTP and PEPITA in the experiments for the fully connected models trained on CIFAR-10 (second column of Table 6.3). The smallest the slowness value, the better the convergence rate. The slowness is computed on the test curve averaged over 10 independent runs.

In PEPITA, the error-based perturbation of the input is key to directing the parameter updates in the correct direction. We verify that the accuracy is above chance but much lower than with

PEPITA, if in the second forward pass the input is perturbed with random noise, or if  $F$  is set to zero (*i.e.*, only the last layer is updated in a BP-fashion). Figure 6.7 compares the test curves obtained on CIFAR-10 with PEPITA, using  $F=0$  or random noise modulation with different amplitudes. Specifically, we compare:

- PEPITA, for which, at the modulated pass, the activations are computed as:

$$\begin{aligned} h_1^{err} &= \sigma_1(W_1(x + Fe)), \\ h_l^{err} &= \sigma_l(W_l h_{l-1}^{err}) \quad 2 \leq l \leq L. \end{aligned} \quad (6.8)$$

- PEPITA with  $F=0$ , for which, at the modulated pass, the activations are computed as:

$$\begin{aligned} h_1^{err} &= \sigma_1(W_1(x)), \\ h_l^{err} &= \sigma_l(W_l h_{l-1}^{err}) \quad 2 \leq l \leq L. \end{aligned} \quad (6.9)$$

Therefore, with  $F=0$ , only the last layer is trained and the hidden layer is frozen.

- Random Perturbation (RP) of the input, for which, at the modulated pass, the activations are computed as:

$$\begin{aligned} h_1^{err} &= \sigma_1(W_1(x + Frk)), \\ h_l^{err} &= \sigma_l(W_l h_{l-1}^{err}) \quad 2 \leq l \leq L. \end{aligned} \quad (6.10)$$

where  $r$  is a random vector with the same dimension as  $e$ , whose elements are randomly sampled from a uniform distribution over  $[0, 1)$ , and  $k$  is a scalar  $\in [0.01, 0.1, 1]$ .

$e = h_L - target$  denotes the network error and  $F$  denotes the fixed random matrix used to project (*i.e.*, add) the error on the input.

Figure 6.7 shows that PEPITA reaches a significantly higher test accuracy than its variation with  $F=0$  or random perturbation of the input. When  $F=0$ , the accuracy saturates at 44.6%, while PEPITA reaches 52.6%. For Random Perturbation, if  $k = 0$  the performance is close to chance level, for smaller values of  $k$  the performance is close to that of  $F = 0$  as the hidden weights are only slightly updated. Hence, the updates obtained by modulating the input with error-related information is key to directing the parameter updates in the correct direction. Our results show that, if the input is not perturbed or is perturbed independently from the error, the network exhibits worse performance.

### 6.4.3 Analytic results

To gain insight into the dynamics of the PEPITA-based learning rule, we considered a two-layer network and measured the alignment of the product between the forward weights ( $W_1 W_2$ ) with the fixed matrix  $F$ . As in [Xiao et al., 2018], we flattened the matrices into vectors and computed the angle between the vectors. Figure 6.5b reports the alignment dynamics of a network with one hidden layer during training on MNIST. We observe that, at initialization, the alignment angle is  $\sim 90^\circ$  (*i.e.*, random), and that during training the angle increases, saturating at approximately  $120^\circ$ . The evolution of the angle alignment finds a plateau as the test accuracy saturates. Interestingly, we observe that our approach encourages a soft ‘antialignment’ (*i.e.*, the angle increases above  $90^\circ$ ) of the forward matrices with the fixed feedback matrix. Here we provide a formal proof showing why such phenomenon occurs in the case of a linear fully connected network. We follow the same reasoning presented for feedback alignment in Supplementary Notes 11 and 12 of [Lillicrap et al., 2016] and use the same notation.

We consider a linear network with one hidden layer, where  $A$  is the weight matrix from the input to the hidden layer and  $W$  is the weight matrix from the hidden to the output layer. Given an input vector  $x$ , the hidden layer activation is computed as  $h = Ax$  (vector), and the output of the network as  $y = Wh$  (vector). Each input  $x$  is associated with a desired target  $y^*$  (vector),



which is given by a target linear transformation  $T$ , such that  $y^* = Tx$ . The aim of the training is to learn  $A$  and  $W$  so that the network is functionally equivalent to  $T$ . We also define the matrix  $E = T - WA$ , so that the error vector can be written as  $e = Ex$ .

Given a learning rate  $\eta$ , the weight updates prescribed by PEPITA can be written as:

$$\Delta W = \eta e h^T = \eta E x x^T A^T \quad (6.11)$$

and

$$\Delta A = \eta(h - h^{err})x^T = \quad (6.12)$$

$$= \eta(Ax - Ax^{err})x^T = \quad (6.13)$$

$$= \eta(Ax - A(x + Fe))x^T = \quad (6.14)$$

$$= \eta(Ax - Ax - AFEx)x^T = \quad (6.15)$$

$$= -\eta AFExx^T. \quad (6.16)$$

Notice that the prescribed update for  $A$  here is computed as  $\Delta A = \eta(h - h^{err})x^T$  rather than  $\Delta A = \eta(h - h^{err})(x + Fe)^T$ . As mentioned in Section 6.3.2, the two rules lead to the same network performance. In this analytical proof for simplicity we consider the first form.

We assume that we train the network with batch learning and that the input samples  $x$  are i.i.d. standard normal random variables (*i.e.*, mean 0 and standard deviation 1) and thus  $xx^T = I$ , where  $I$  is the identity matrix. Then the parameter updates can be written as:

$$\Delta W = \eta E x x^T A^T = \eta E A^T \quad (6.17)$$

and

$$\Delta A = -\eta AFExx^T = -\eta AFE. \quad (6.18)$$

We note that the minus sign in the update of  $A$  does not appear in the update of  $A$  using feedback alignment [Lillicrap et al., 2016]. Therefore, this step of the proof explains the difference in the direction of the alignment.

In the limit of a small  $\eta$ , the discrete time learning dynamics converge to the continuous time dynamical system:

$$\dot{W} = EA^T \quad (6.19)$$

and

$$\dot{A} = -AFE. \quad (6.20)$$

where  $\dot{A}$  and  $\dot{W}$  are the corresponding temporal derivatives. In order to show why the forward matrices and  $F^T$  come to antialignment with each other, we need to prove that the time derivative of  $tr(FWA)$  is negative. Indeed, the antialignment is defined by the condition  $\frac{d}{dt}tr(FWA) < 0$ , where  $tr(FWA) = [WA]_{\downarrow} \cdot [F^T]_{\downarrow}$  and  $[\cdot]_{\downarrow}$  is an operator that flattens a matrix into a vector. To this goal, we decouple the deterministic dynamics of  $A$  and  $W$ : at first we freeze  $W$  and train  $A$ , and then we keep  $A$  constant and train  $W$ .

In the first phase, we impose  $\dot{W} = 0$  and let  $\dot{A} = -AFE$ . In our experiments we observe that the norm of  $A$  monotonically increases during training (Figure 6.5c). Thus  $\frac{d}{dt}\|A\|^2 = \frac{d}{dt}tr(A^T A) > 0$ . We can rewrite the time derivative of the norm as:

$$\frac{d}{dt}tr(A^T A) = 2tr(A^T \dot{A}) = \quad (6.21)$$

$$= 2tr(A^T(-AFE)) = \quad (6.22)$$

$$= 2tr(-A^T AFE) = \quad (6.23)$$

$$= 2tr(-\|A\|^2 FE) = \quad (6.24)$$

$$= -2\|A\|^2 tr(FE). \quad (6.25)$$

Therefore, we deduce  $tr(FE) < 0$ .

Next, we consider the second phase, in which  $\dot{A} = 0$  and  $\dot{W} = EA^T$ . Under this condition, we examine the evolution in time of  $tr(FWA)$ :

$$\frac{d}{dt}tr(FWA) = tr(F\dot{W}A) = \quad (6.26)$$

$$= tr(FEA^T A) = \quad (6.27)$$

$$= tr(FE\|A\|^2) = \quad (6.28)$$

$$= \|A\|^2 tr(FE). \quad (6.29)$$

From the above result  $tr(FE) < 0$ , we conclude that  $\frac{d}{dt}tr(FWA) < 0$ . This analytical result for a linear network supports our empirical finding on non-linear models that the product of the forward matrices comes to antialign with  $F^T$ . We conjecture that such antialignment enables the error that modulates the input, and consequently the neural activity, to deliver useful learning signals for weight updates in  $A$  and  $W$ .

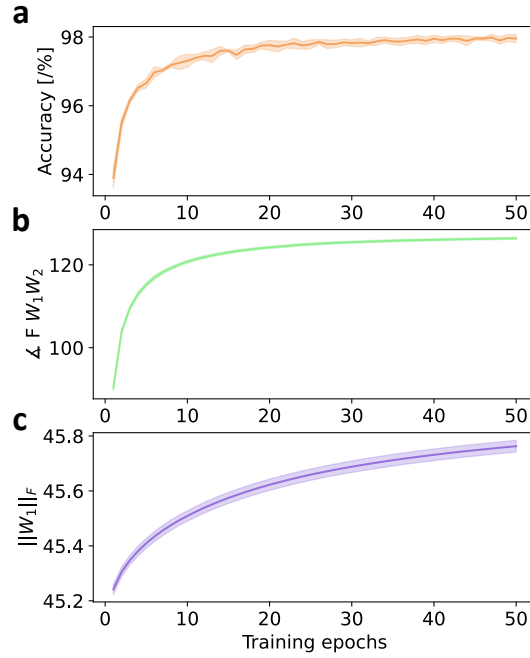


Figure 6.5: Dynamics of a 1 hidden layer network, with 1024 hidden units, 10% dropout, during training on MNIST with PEPITA. (a) Test accuracy computed after each training epoch. (b) Angle between the product of the forward matrices and the F matrix. (c) Evolution of the norm of the initial weight matrix. The solid line reports the mean over five independent runs, the shaded area indicates the standard deviation.

Such analytical result for a linear network supports our empirical finding on non-linear models that the product of the forward matrices comes to antialign with  $F^T$ . We conjecture that such antialignment enables the error that modulates the input, and consequently the neural activity, to deliver useful learning signals for weight updates in  $A$  and  $W$ . We remark that with a different sign choice for the input modulation step in Eq. 6.2 – *i.e.*,  $h_1^{err} = \sigma_1(W_1(x - Fe))$  – the product of the forward matrices would instead come to ‘align’ with  $F^T$ . In the proof, we make the assumption that the norm of the first hidden layer weights increases monotonically. We verify empirically that this is a realistic assumption in Figure 6.5c. Figure 6.9 shows that the rate of weights’ norm increase diminishes as training continues beyond 50 epochs and the learning rate is decayed. The

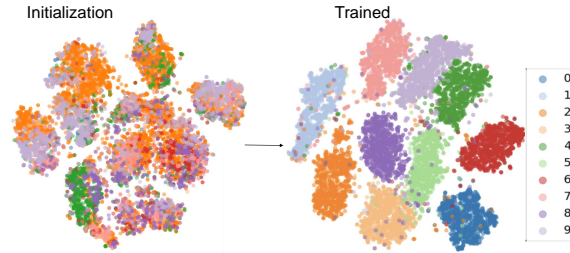


Figure 6.6: t-SNE embeddings of the MNIST representation at the hidden layer of the fully connected model trained on MNIST, both before (initialization) and after training (trained). Each color corresponds to a different class.

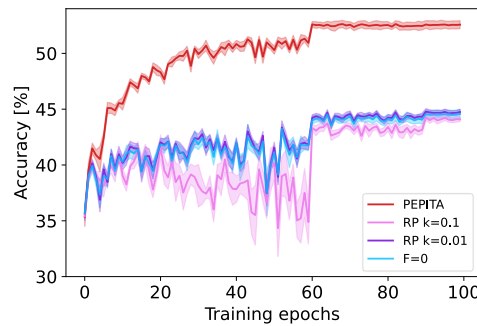


Figure 6.7: Test curve for PEPITA in its original formulation, PEPITA with  $F=0$  (*i.e.*, only the last layer is trained), and a Random Projection (RP) algorithm in which the input is modulated with a random vector  $r$ . The elements of  $r$  are sampled from a uniform distribution over  $[-1, 1)$  and multiplied by a scalar  $k$ . We used the same settings as in the experiments for the fully connected models trained on CIFAR-10 (second column of Table 6.3). The solid line is the mean over 10 independent runs. The shaded colored area shows the std over the 10 runs. PEPITA achieves a significantly higher accuracy than the  $F=0$  and RP schemes. Additionally, for small  $k$  values, the RP algorithm (purple) has a similar learning curve as the  $F=0$  algorithm (blue).

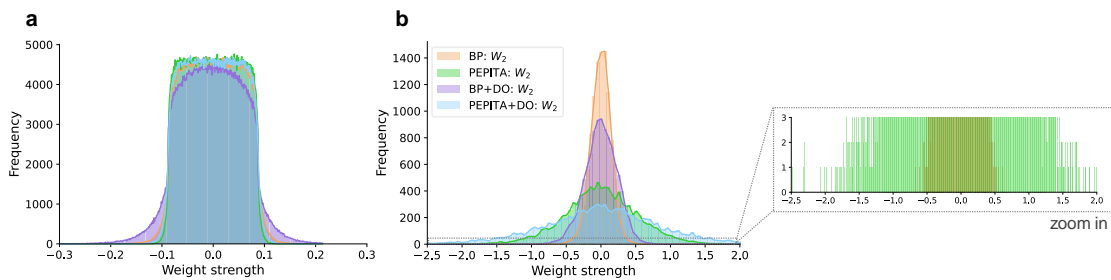


Figure 6.8: Distribution of the weights trained with BP (orange, purple) and PEPITA (green, blue) on MNIST for a network with one hidden layer with 1024 units, with and without dropout (DO), with dropout rate 10%. (a) First hidden layer (b) Output layer

bottom panel shows that, after the learning rate decay step at epoch 60, the norm of  $W_1$  increases with a significantly smaller rate, however it does not saturate.

Furthermore, the proof relies on the assumption that the input is whitened. In Table 6.4 we show that PEPITA effectively trains networks also with whitened input. We use the same settings for the fully connected model as described in Table 6.1, except for the learning rate which needs to be decreased by a factor 10 ( $\eta = 0.01$ ). The Table below reports the accuracy result. The accuracy

Table 6.3: Test accuracy [%] achieved by BP, FA, DRTP and PEPITA in the experiments. Mean and standard deviation are computed over 10 independent runs.

	FULLY CONNECTED MODELS			CONVOLUTIONAL MODELS		
	MNIST	CIFAR10	CIFAR100	MNIST	CIFAR10	CIFAR100
BP	98.63±0.03	55.27±0.32	27.58±0.09	98.86±0.04	64.99±0.32	34.20±0.20
FA	98.42±0.07	53.82±0.24	24.61±0.28	98.50±0.06	57.51±0.57	27.15±0.53
DRTP	95.10±0.10	45.89±0.16	18.32±0.18	97.32±0.25	50.53±0.81	20.14±0.68
PEPITA	98.01±0.09	52.57±0.36	24.91±0.22	98.29±0.13	56.33±1.35	27.56±0.60

obtained with whitening is lower than without whitening, however the accuracy is above 90%, showing that the assumption in the anti-alignment proof is plausible.

	MNIST
PEPITA NON WHITENED INPUT	98.01±0.09
PEPITA WHITENED INPUT	92.08±0.14

Table 6.4: Test accuracy [%] achieved by PEPITA with and without whitening on the input with a fully connected model trained on the MNIST dataset. Mean and standard deviation are computed over 10 independent runs.

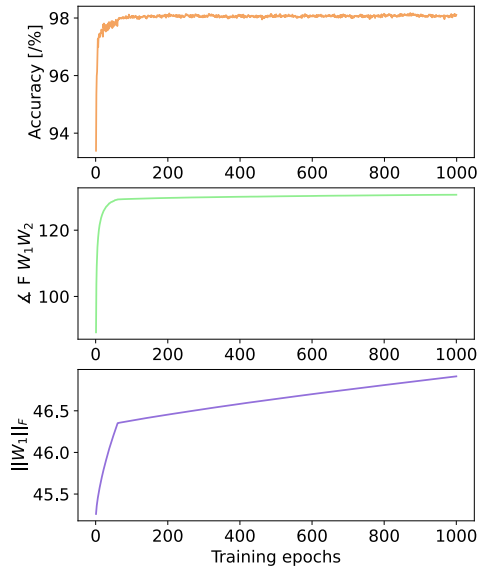


Figure 6.9: Figure 6.5 extended to 1000 epochs. Dynamics of a 1 hidden layer network, with 1024 hidden units, 10% dropout, during training on MNIST with PEPITA. (a) Test accuracy computed after each training epoch. (b) Angle between the product of the forward matrices and the F matrix. (c) Evolution of the norm of the initial weight matrix. The solid line reports the mean over five independent runs, the shaded area indicates the standard deviation.

### 6.4.4 Final weight distribution

We compared the layer-wise weight distribution of networks trained with BP and PEPITA, initialized with the same normal distribution [He et al., 2015]. For a two-layer model trained on MNIST, the trained weights in the first layer have similar distributions for BP and PEPITA (Figure 6.8a), with the weight distribution for BP covering a slightly greater range than for PEPITA. The distribution in the last layer is instead significantly different: both BP and PEPITA lead to bell-shaped distributions with mean close to zero, however the distribution width obtained with PEPITA is significantly larger (Figure 6.8b). Importantly, this implies that PEPITA learns different solutions with respect to BP, and those solutions yield successful performance. In Figure 6.8 we show that the weight distribution obtained with BP can be fit with a Normal curve, while the distribution obtained with PEPITA is better described by a Student’s t-distribution. The Student’s t-distribution is a subexponential distribution, which is reminiscent of the connectivity structure of neurons in cortical networks. Indeed, findings of electrophysiological studies indicate that the amplitudes of excitatory post-synaptic potentials between cortical neurons obey a long-tailed pattern, typically lognormal [Buzsáki and Mizuseki, 2014, Song et al., 2005, Iyer et al., 2013].

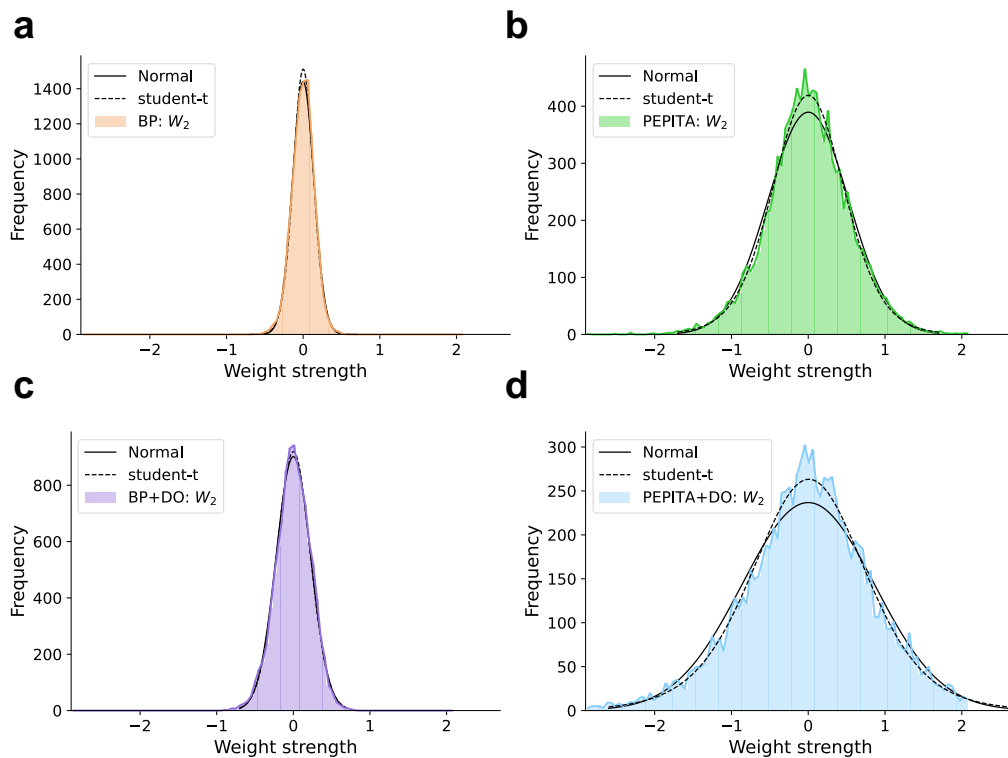


Figure 6.10: The weight distributions for the output layer reported Figure 6.8b here are shown separately and fit both with a Normal distribution (solid black line) and a Student’s t-distribution (dotted black line). With the BP training scheme (a) without dropout and (c) with dropout, the Normal curve well fits the weight distribution. Instead, with the PEPITA training scheme both (b) without dropout and (d) with dropout settings, the Student’s t-distribution fits the weight distribution much more accurately than the Normal curve. This indicates that PEPITA pushes the weights towards a sub-exponential distribution. Note that we used different scale for the y-axis for visualization purposes.

Furthermore, PEPITA drives the training towards a parameter distribution in which most synaptic connections are weak and a few weights are sparse and strong (zoom of Figure 6.8b). [Feldman and Valiant, 2009] demonstrates that a network regime featuring some maximally strong synapses enables sparse and disjoint representation of items. This yields a significantly higher memory capacity compared to a

regime supported by weak synapses only. We also remark that previous work [Blundell et al., 2015] has shown that networks regularized with dropout [Hinton et al., 2012] or uncertainty on the weights (*i.e.*, Bayes by Backprop [Blundell et al., 2015]) present a greater range of weight strength as compared to standard SGD. Therefore, PEPITA, which is conceived to solve the biological implausible aspects of network training via BP, conveys the network weights resembling more biologically plausible distributions than BP, possibly supporting higher memory capacities as well as more regularized parameters. However, we observe that, despite the similarity in subexponential distribution, the weight distribution learnt by PEPITA differs from the synaptic strength in biological circuits in the ratio between excitatory to inhibitory connections. While it is generally thought that biological neurons function under balanced excitation and inhibition, this does not imply equal numbers of excitatory and inhibitory neurons as it occurs in PEPITA.

## 6.5 Discussion

We introduced PEPITA, a learning rule that relies on perturbing the input signal through error-related information and updating the network parameters based on local variations of activity. Our results indicate that PEPITA is able to train both fully connected and convolutional neural networks on image classification tasks, with an accuracy comparable to the feedback alignment algorithms. We empirically observe that PEPITA encourages the feedforward weights to evolve towards a soft ‘antialignment’ with the feedback matrix, and it nudges the weight distribution towards a “strong-sparse and weak-dense” distribution reminiscent of the connectivity of cortical neurons.

By replacing the backward pass with an additional forward pass, our approach avoids BP’s biologically unrealistic aspects of weight transport, non-locality, freezing of neural activity during parameter updates, and update locking. First, PEPITA is not affected by the weight transport problem as it does not send weight-specific feedback signals. Secondly, regarding locality, the prescribed updates of each synapse solely rely on the activity of the nodes it is connected to. The only global signal required by the algorithm is the error projected onto the input. The role of this global error in perturbing the node activity may correspond to a neuromodulator that influences local synaptic plasticity [Mazzoni et al., 1991, Williams, 1992, Clark et al., 2021]. Third, during training, PEPITA never requires the network activity to be frozen since the error is propagated together with the input signal. Finally, in PEPITA the updates are performed layer-wise in a feedforward fashion allowing to partially solve the update locking issue. Indeed, the weights of the first layer can be updated right at the beginning of the second forward pass and do not need to wait for the updates of the downstream layers. Hence, the forward computation for the next input sample at the first layer can start in parallel with the update of the second layer from the previous sample, and so on. In the case of very deep networks, such a strategy could substantially reduce the computational time, suggesting that PEPITA could be suitable for edge computing devices requiring fast processing of the input signals.

While our approach addresses BP’s issues of biological plausibility, it introduces additional elements with respect to BP and FA. Specifically, PEPITA involves the projection of the error onto the input through a fixed random matrix. We speculate that the role of such a projection matrix is reminiscent of the connectivity matrices supporting the cortico-thalamo-cortical loops throughout neocortex. Additionally, the error-based input modulation is supported by the observation that neuromodulators can modulate the activity in the visual stream. Although neuromodulators do not change activity in the retina (let alone inputs to the retina), they can change the activity at the very early stages of the visual pathway (LGN, V1) [Kreiman, 2021b]. Since we apply PEPITA on shallow networks, the algorithm approximates the role of neuromodulators by changing the input directly. In deeper networks, we could apply the error-based modulation on the low hidden layers rather than on the input. Furthermore, PEPITA requires storing the activations of the standard pass during the modulated pass. We suggest that this could be implemented in biological neurons through dendritic and somatic activities. Earlier works have proposed two-compartmental neuron models in which learning is driven by local prediction mismatch between synaptically-driven

dendritic activity and somatic output [Asabuki and Fukai, 2020, Sacramento et al., 2018b]. We propose that if we apply PEPITA to train a network of such two-compartment neurons, the response to the input signal in the standard forward pass is first integrated in the dendritic compartment and then encoded in the somatic activity. Then, in the modulated forward pass, the dendrites encode the response to the modulated input while the soma is still storing the response to the original input. The mismatch between the dendritic activity encoding the response to the modulated input and the somatic activity encoding the response to the original input drives the synaptic update. Consequently, synaptic strength can be updated based on the outcome of the second pass and the outcome of the first pass through a mismatch between somatic activity (storing the activity of the first pass) and dendritic activity (integrating the activity of the second pass) in two-compartment neurons. Alternatively, the activity of the first pass can drive plasticity through the dynamics of interneurons: the past activation can be stored in an interneuron and then retrieved after the second forward pass.

Importantly, PEPITA bears structural similarity with learning rules that are believed to take place in the brain, *i.e.*, Hebbian-like approaches. Indeed, the update rule shown in the pseudocode can be decoupled into two separate successive updates:  $\Delta W_\ell^1 = (h_\ell) \cdot (h_{\ell-1}^{err})^T$  and  $\Delta W_\ell^2 = (-h_\ell^{err}) \cdot (h_{\ell-1}^{err})^T$ . Each of the two updates embodies a fundamental feature of Hebbian learning [Gerstner et al., 2014a], *i.e.*, the synaptic strength modification is proportional to both a presynaptic and a postsynaptic signal. Therefore the definition of PEPITA fits in the criteria to be a biologically plausible mechanism.

In conclusion, we have demonstrated that a learning rule that solves the biologically implausible aspects of BP by relying only on forward computations, is able to train both fully connected and convolutional models with a performance close to BP and on par with FA. The proposed algorithm can thus help bridge the gap between neurobiology and machine learning.

## Chapter 7

# Dendritic Learning for Blind Source Separation

In natural auditory environments, acoustic signals originate from the temporal superimposition of different sound sources. The problem of inferring individual sources from ambiguous mixtures of sounds is known as blind source decomposition. Experiments on humans have demonstrated that the auditory system can identify sound sources as repeating patterns embedded in the acoustic input. Source repetition produces temporal regularities that can be detected and used for segregation. Specifically, listeners can identify sounds occurring more than once across different mixtures, but not sounds heard only in a single mixture. However, whether such a behaviour can be computationally modelled has not yet been explored. Here, we propose a biologically inspired computational model to perform blind source separation on sequences of mixtures of acoustic stimuli. Our method relies on a somatodendritic neuron model trained with a Hebbian-like learning rule which was originally conceived to detect spatio-temporal patterns recurring in synaptic inputs. We show that the segregation capabilities of our model are reminiscent of the features of human performance in a variety of experimental settings involving synthesized sounds with naturalistic properties. Furthermore, we extend the study to investigate the properties of segregation on task settings not yet explored with human subjects, namely natural sounds and images. Overall, our work suggests that somatodendritic neuron models offer a promising neuro-inspired learning strategy to account for the characteristics of the brain segregation capabilities as well as to make predictions on yet untested experimental settings.

### 7.1 Introduction

Hearing a sound of specific interest in a noisy environment is a fundamental ability of the brain that is necessary for auditory scene analysis. To achieve this, the brain has to unambiguously separate the target auditory signal from other distractor signals. In this vein, a famous example is the “cocktail party effect” [Cherry, 1953], *i.e.*, the ability to distinguish a particular speaker’s voice against a multi-talker background [Brown et al., 2001, Mesgarani and Chang, 2012]. Many psychophysical and neurobiological studies have been conducted to clarify the psychophysical properties and underlying mechanisms of the segregation of mixed signals [McDermott, 2009, Bee and Micheyl, 2008, Asari et al., 2006, McDermott et al., 2011, Lewald and Getzmann, 2015, Li et al., 2017, Atilgan et al., 2018, Narayan et al., 2008, Schmidt and Römer, 2011], and computational theories and models have also been proposed for this computation [Thakur et al., 2015, Haykin and Chen, 2005, Kameoka et al., 2018, Karamatli et al., 2018, Sawada et al., 2019, Sagi et al., 2001, Bell and Sejnowski, 1995, Amari et al., 1995, Dong et al., 2016, Elhilali and Shamma, 2009]. However, how the brain attains its remarkable sound segregation remains elusive. Various properties of auditory cues such as spatial cues in binaural listening [Ding and Simon, 2012] and temporal coherence of sound stimuli [Krishnan et al., 2014, Teki et al., 2013] are known to facilitate the listener’s



ability to segregate a particular sound from the background. Auditory signals that reached ears first undergo the analysis of frequency spectrums by cochlea [Oxenham, 2018]. Simultaneous initiation and termination of the component signals and the harmonic structure of the frequency spectrums help the brain to identify the components of the target sound [Popham et al., 2018]. Prior knowledge about the target sound, such as its familiarity to listeners [Woods and McDermott, 2018, Elhilali, 2013], and top-down attention can also improve their ability to detect the sound [Ahveninen et al., 2011, Golumbic et al., 2013, Bronkhorst, 2015, Xiang et al., 2010, O’Sullivan et al., 2014, Kerlin et al., 2010]. Selective attention as the combination of the auditory (sound) and visual (lip movements, visual cues) modalities has also been suggested to be beneficial to solve the cocktail party problem [Yu, 2020, Liu et al., 2021]. However, many of these cues are subsidiary and not absolutely required for hearing the target sound. For example, a mixture sound can be separated by monaural hearing [Hawley et al., 2004] or without spatial cues [Middlebrooks and Waters, 2020]. Therefore, the crucial mechanisms of sound segregation remain to be explored.

Whether or not biological auditory systems segregate a sound based on principles similar to those invented for artificial systems remains unclear [McDermott, 2009, Bee and Micheyl, 2008]. Among such principles, independent component analysis (ICA) [Comon, 1994] and its variants are the conventional mathematical tools used for solving the sound segregation problem, or more generally, the blind source decomposition problem [Haykin and Chen, 2005, Bell and Sejnowski, 1995, Amari et al., 1995, Hyvärinen and Oja, 1997]. Owing to its linear algebraic features, the conventional ICA requires as many input channels (e.g., microphones) as the number of signal sources, which does not appear to be a requirement for sound segregation in biological systems. In this context, however, recent works for single-channel source separation based on techniques such as Non-Negative Matrix Factorization (NNMF) have demonstrated that ICA can be applied with a lower number of channels than the number of sources [Krause-Solberg and Iske, 2015, Mika et al., 2020]. In addition, NNMF has been shown to extract regular spatio-temporal patterns within the audio and to achieve good performance in applications such as music processing [Smaragdis and Brown, 2003, López-Serrano et al., 2019, Cichocki et al., 2006, Santosh and Bharathi, 2017]. It has been suggested as an alternative possibility that human listeners detect latent recurring patterns in the spectro-temporal structure of sound mixtures for separating individual sound sources [McDermott et al., 2011]. This was indicated by the finding that listeners could identify a target sound when the sound was repeated in different mixtures in combination with various other sounds but could not do so when the sound was presented in a single mixture.

The finding represents an important piece of information about the computational principles of sound source separation in biological systems. Here, we demonstrate that a computational model implementing a pattern-detection mechanism accounts for the characteristic features of human performance observed in various task settings. To this end, we constructed a simplified model of biological auditory systems by using a two-compartment neuron model recently proposed for learning regularly or irregularly repeated patterns in input spike trains [Asabuki and Fukai, 2020]. Importantly, this learning occurs in an unsupervised fashion based on the minimization principle of regularized information loss, showing that the essential computation of sound source segregation can emerge at the single-neuron level without teaching signals. Furthermore, it was previously suggested that a similar repetition-based learning mechanism may also work for the segregation of visual objects [McDermott et al., 2011]. To provide a firm computational ground, we extended the tasks of our framework to predictions on visual images.

## 7.2 Results

### 7.2.1 Learning of repeated input patterns by a two-compartment neuron model

We used a two-compartment spiking neuron model which learns recurring temporal features in synaptic input, as proposed in [Asabuki and Fukai, 2020]. In short, the dendritic compartment attempts to predict the responses of the soma to given synaptic input by modelling the somatic responses. To this end, the neuron model minimizes information loss within a recent period when

the somatic activity is replaced with its model generated by the dendrite. Mathematically, the learning rule minimizes the Kullback–Leibler (KL) divergence between the probability distributions of somatic and dendritic activities. The dendritic membrane potential of a two-compartment neuron obeys  $v(t) = \sum_j w_j e_j(t)$ , where  $w_j$  and  $e_j$  stand for the synaptic weight and the unit postsynaptic potential of the  $j$ -th presynaptic input, respectively. The somatic activity evolves as

$$\dot{u}(t) = -\frac{1}{\tau}u(t) + g_D[-u(t) + v(t)] - \sum_j G_k \phi^{som}(u_k(t))/\phi_0, \quad (7.1)$$

where the last term describes lateral inhibition with modifiable synaptic weights  $G_k$  ( $\geq 0$ ), as shown later. The soma generates a Poisson spike train with the instantaneous firing rate  $\phi^{som}(u(t))$ , where  $\phi_i^{som}(u_i) = \phi_0[1 + e^{\beta(-u_i + \theta)}]^{-1}$ , and the parameters  $\beta$  and  $\theta$  are modified in an activity-dependent manner in terms of the mean and variance of the membrane potential over a sufficiently long period  $t_0$ . To extract the repeated patterns from temporal input, the model compresses the high dimensional data carried by the input sequence onto a low dimensional manifold of neural dynamics. This is performed by modifying the weights of dendritic synapses to minimize the time-averaged mismatch between the somatic and dendritic activities over a certain interval  $[0, T]$ . In a stationary state, the somatic membrane potential  $u_i(t)$  can be described as an attenuated version  $v_i^*(t)$  of the dendritic membrane potential. At each time point, we compare the attenuated dendritic membrane potential with the somatic membrane potential, on the level of the two Poissonian spike distributions with rates  $\phi_i^{som}(u(t))$  and  $\phi(v_i^*(t))$ , respectively, which would be generated if both soma and dendrite were able to emit spikes independently. In practice, the neuron model minimizes the following cost function for synaptic weights  $w$ , which represents the averaged KL-divergence between somatic activity and dendritic activity, and in which we explicitly represent the dependency of  $u_i$  and  $v_i^*$  on  $\mathbf{X}$ :

$$E(\mathbf{w}) = \int_{\Omega_X} dX P^*(\mathbf{X}) \int_0^T dt \sum_i D_{KL}[\phi_i^{som}(u_i(t; \mathbf{X})) || \phi^{dend}(v_i^*(t; \mathbf{X}))], \quad (7.2)$$

with  $P^*(\mathbf{X})$  and  $\Omega_X$  being the true distribution of input spike trains and the entire space spanned by them, and  $\phi^{dend}(x) = \phi_0[1 + e^{\beta_0(-x + \theta_0)}]^{-1}$ . To search for the optimal weight matrix, the cost function  $E(w)$  is minimized through gradient descent:  $\Delta w_{ij} \propto -\partial E / \partial w_{ij}$ . Introducing the regularization term  $-\gamma \mathbf{w}_i$  and a noise component  $\xi_i$  with its intensity  $g$  gives the following learning rule (for the derivation see [Asabuki and Fukai, 2020]):

$$\dot{\mathbf{w}}_i(t) = \eta \{ \psi(v_i^*(t)) [ \{ f(\phi_i^{som} + \phi_0 g \xi_i) - \phi^{dend}(v_i^*(t)) \} / \phi_0 ] \mathbf{e}(t) - \gamma \mathbf{w}_i \}, \quad (7.3)$$

where  $\mathbf{w}_i = [w_{i1}, \dots, w_{iN_{in}}]$ ,  $\mathbf{e}(t) = [e_1, \dots, e_{N_{in}}]$ ,  $\xi_i$  obeys a normal distribution,  $\psi(x) = \frac{d}{dx} \log(\phi^{dend}(x))$ ,  $\phi^{som}$  and  $\phi^{dend}$  follow Poisson distributions,  $\eta$  is the learning rate, and

$$f(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } 0 \leq x < \phi_0, \\ \phi_0 & \text{if } x \geq \phi_0 \end{cases}$$

Finally, if a pair of presynaptic and postsynaptic spikes occur at the times  $t_{pre}$  and  $t_{post}$ , respectively, lateral inhibitory connections between two-compartment neurons  $i$  and  $j$  are modified through a symmetric anti-Hebbian STDP as

$$\Delta G_{ij} = C_p \exp\left(-\frac{t_{pre} - t_{post}}{\tau_p}\right) - C_d \exp\left(-\frac{t_{pre} - t_{post}}{\tau_d}\right) \quad (7.4)$$

See Material and Methods and Supplementary Note for additional details. The prediction is learnable when input spike sequences from presynaptic neurons are non-random and contain recurring temporal patterns. In such a case, the minimization of information loss induces a consistency check between the dendrite and soma, eventually enforcing both compartments to respond selectively to

one of the patterns. Mathematically, the somatic response serves as a teaching signal to supervise synaptic learning in the dendrite. Biologically, backpropagating action potentials may provide the supervising signal [Larkum, 2013, Larkum et al., 1999].

We constructed an artificial neural network based on the somatodendritic consistency check model and trained the network to perform the task of source recovering from embedded repetition. The network consisted of two layers of neurons. The input layer encoded the spectrogram of acoustic stimuli into spike trains of Poisson neurons. For each sound, the spike train was generated through a sequence of 400 time steps, where each time step corresponds to a “fire” or “non-fire” event. The output layer was a competitive network of the two-compartment models that received synaptic input from the input layer and learned recurring patterns in the input (figure 7.1). We designed the output layer and the learning process similarly to the network used previously [Asabuki and Fukai, 2020] for the blind signal separation (BSS) within mixtures of multiple mutually correlated signals. In particular, lateral inhibitory connections between the output neurons underwent spike-timing-dependent plasticity for self-organizing an array of feature-selective output neurons (Material and Methods). In the spike encoding stage, the spectrogram is flattened into a one-dimensional array where the intensity of each element is proportional to the Poisson firing probability of the associated input neuron. This operation disconnects the signal’s temporal features from the temporal dynamics of the neurons. Although this signal manipulation is not biologically plausible and introduces additional latency as the whole sample needs to be buffered, it allows the input layer to encode simultaneously all the time points of the audio signal. Thanks to this strategy, the length of the input spike trains does not depend on the duration of the audio signal, and a sufficiently large population of input neurons can encode arbitrarily long sounds, possibly with some redundancy in the encoding for short sounds. We remark that, while the somatodendritic mismatch learning rule was conceived to capture temporal information in an online fashion, in our framework it is applied to a flattened spectrogram, thus to a static pattern. Furthermore, in order to relate the signal intensity with the encoding firing rate, we normalized the spectrogram values to the interval  $[0,1]$ . This strategy is suited to our aim of reproducing the experiments with synthetic sounds and custom naturalistic stimuli. However, in a real-world application any instantaneous outlier in signal intensity would destroy other temporal features of an input signal. Nonetheless, the normalization is performed independently for each mixture, so if the outlier affects a masker sound and not a target, and the target is presented in at least two other mixtures, we expect that the normalization does not affect the ability of the network of identifying sounds presented in different mixtures.

### 7.2.2 Synthesized and natural auditory stimuli

We examined whether the results of our computational model are consistent with the outcomes of the experiments on human listeners on artificially synthesized sounds described previously [McDermott et al., 2011]. To provide a meaningful comparison with the human responses, we adopted for our simulations settings as close as possible to the experiments, both in terms of dataset generation and performance evaluation (Material and Methods). In [McDermott et al., 2011], the generation of synthetic sounds is performed by first measuring the correlations between pairs of spectrograms cells of natural sounds (spoken words and animal vocalizations). Then such correlations are averaged across different pairs to obtain temporal correlation functions. The correlation functions in turn are used to generate covariance matrices, in which each element is the covariance between two spectrogram cells. Finally, spectrograms are drawn from the resulting Gaussian distribution and applied to samples of white noise, leading to the synthesis of novel sounds. In our experiments we synthesized the sounds using the toolbox provided at <https://mcdermottlab.mit.edu/downloads.html>. In the human experiments, a dataset containing novel sounds was generated such that listeners’ performance in sound source segregation was not influenced by familiarity with previously experienced sounds. To closely reproduce the experiment, we created a database of synthesized sounds according to the same method as described in [McDermott et al., 2011] (Material and Methods). The synthesized stimuli retained similarity to real-world sounds except that they lacked grouping cues related to temporal onset and harmonic

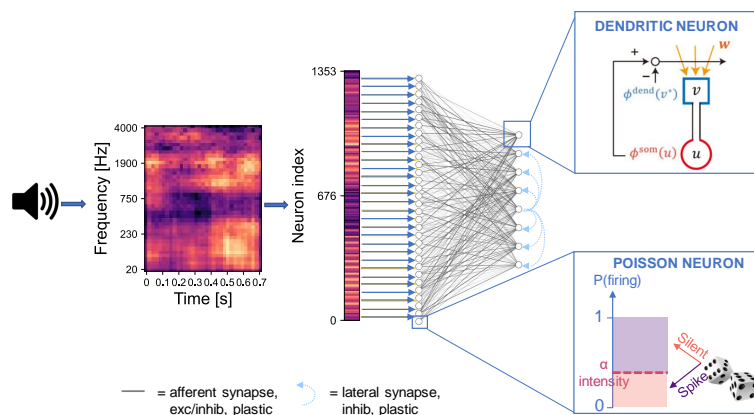


Figure 7.1: **Network architecture.** The input signal is pre-processed into a two-dimensional image (*i.e.*, the spectrogram) with values normalized in the range  $[0,1]$ . The image is flattened into a one-dimensional array where the intensity of each element is proportional to the Poisson firing probability of the associated input neuron. The neurons in the input layer are connected to those in the output layer through either full connectivity or random connectivity with connection probability  $p = 0.3$ . The output neurons are trained following the artificial dendritic neuron learning scheme [Asabuki and Fukai, 2020].

spectral structures. Furthermore, unlike human listeners, our neural network was trained and built from scratch, and had no previous knowledge of natural sounds that could bias the task execution. We exploited this advantage to investigate whether and how the sound segregation performance was affected by the presence of grouping cues in real sounds. To this goal we also built a database composed of natural sounds (Material and Methods).

To build the sequence of input stimuli, we randomly chose a set of sounds from the database of synthesized or natural sounds, and we generated various mixtures by superimposing them — *i.e.*, we summed element-wise the spectrograms of the original sounds and then normalized the sum to the interval  $[0,1]$ . We refer to the main sound, which is always part of mixtures, as the *target*, and to all the other sounds, which were either presented as mixing sounds with the target (*i.e.*, *masker sounds*) or presented alone, as *distractors*. The target sound is shown in red in the training protocols (figure 7.5 and figure 7.6). Following the protocol in [McDermott et al., 2011], we concatenated the mixtures of target and distractors into input sequences. For certain experiments, we also included unmixed distractor sounds. We presented the network with the input sequence for a fixed number of repetitions. As each input signal— both unmixed sounds and mixtures— is flattened into one input vector, each input signal is one element of the input sequence. During the input presentation, the network’s parameters evolved following the learning rule described in [Asabuki and Fukai, 2020]. Then, we examined the ability of the trained network to identify the target sound by using probe sounds, which were either the target or distractor sound composing the mixtures presented during training (*correct probe*) or a different sound (*incorrect probe*). Incorrect probes for synthesized target sounds were generated similarly as described in [McDermott et al., 2011]. Specifically, we synthesized the incorrect probe by using the same covariance structure of the target sound, and then we set a randomly selected time slice of the incorrect probe ( $1/8$  of the sound’s duration) to be equal to a time slice of the target of the same duration. Examples of target sounds, distractor

sounds and incorrect probes are shown in figure 7.2A, figure 7.2B and figure 7.2C, respectively. A further beneficial aspect of our model is the possibility of freezing plasticity during the inference stage, so that the synaptic connections do not change during the probe presentation. This allows us to investigate whether the trained network can identify not only the target but also the masker sounds.

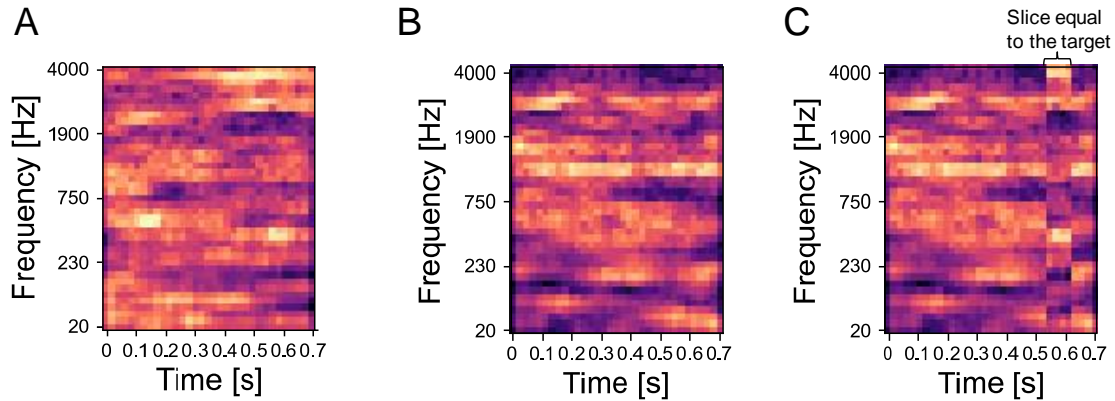


Figure 7.2: **Synthesized sounds - Target and associated distractor.** (A) Spectrogram of one target sound. (B) Step 1 to build the spectrogram of an incorrect probe related to the target in A: a sound is randomly selected from the same Gaussian distribution generating the target. (C) Step 2 to build the incorrect probe: after the sampling, a randomly selected time slice equal to 1/8 of the sound duration is set to be equal to the target. In the figure, the temporal slice is the vertical stripe around time 0.5s.

### 7.2.3 Learning of mixture sounds in the network model

Our network model contained various hyperparameters such as number of output neurons, number of mixtures and connectivity pattern. A grid search was performed to find the best combination of hyperparameters. Figure 7.3A and figure 7.3B report the learning curves obtained on synthesized and natural sounds, respectively, for random initial weights and different combinations of hyperparameters. For both types of sounds, synaptic weights changed rapidly in the initial phase of learning. The changes were somewhat faster for synthesized sounds than for natural sounds, but the learning curves behaved similarly for both sound types. The number of output neurons little affected the learning curves, while they behaved differently for different connectivity patterns or different numbers of mixtures. Because familiarity to sounds enhances auditory perception in humans [Jacobsen et al., 2005], we investigated whether pretraining with a sequence containing target and distractors improves learning in our model for various lengths of pretraining. Neither the training speed nor the final accuracy were significantly improved by the pretraining (figure 7.3C, figure 7.3D and figure 7.3E). This suggests that the model was “forgetting” about the pretraining stage and learning the mixture sounds from scratch, not exploiting any familiarity with previously seen sounds. We suspect that this behavior is related to the well know limitation of ANNs of lack of continual learning [French, 1999] rather than to a specific feature of our model. Furthermore, we cannot provide a comparison in the learning curve between the model and the psychophysical data, since the model was trained for multiple epochs, while the human listeners were presented with the training sequence only once and then tested on the probe immediately after.

To reliably compare the performance of our model with human listeners, we designed a similar assessment strategy to that adopted in the experiment. In [McDermott et al., 2011] listeners were presented with mixtures of sounds followed by a probe which could be either a correct probe (*i.e.*, the target sound present in the training mixtures) or an incorrect probe (*i.e.*, sounds unseen during

the training). The subjects had to say whether they believed the probe was present in the training mixture by using one of the four responses “sure no” “no”, “yes”, and “sure yes”. The responses were used to build a receiver operating characteristics (ROC) as described in [Wickens, 2002], and the area under the curve (AUC) was used as performance measure, with AUC = 0.5 and 1 corresponding to chance and perfect correct, respectively. In our algorithm, we mimicked this protocol for reporting by using the likelihood as a measure of performance. To this goal, first, for each tested probe, we projected the response of the  $N$  output neurons (figure 7.4A,D) to a two-dimensional PCA projection plane. We defined the PCA space based on the response to the correct probes and later projected on it the datapoints related to the incorrect probes (figure 7.4B,E). We remark that other clustering approaches such as K-means and self-organizing maps could be used instead of PCA without reducing the output dimension. Second, we clustered the datapoints related to the correct probes through a Gaussian Mixture Model (GMM) with as many classes as the number of correct probes (figure 7.4C,F). Third, for each datapoint we computed the likelihood that it belonged to one of the clusters. The target likelihood values are fixed to 1 and 0 for datapoints related to correct and incorrect probes respectively. We highlight that the labels introduced in this post-processing phase are not specific for each sound, but rather depend on the role of the sound in the tasks, *i.e.*, if sound  $X$  is presented during training as a target or masker sound it is associated to label 1, while if, in another simulation, the same sound  $X$  is used to build an incorrect probe (not used during training) then it is associated with label 0. We binned the likelihood range into four intervals corresponding, in an ascending order, to the four responses “sure no”, “no”, “yes”, and “sure yes”. Finally, based on the four responses, we built the receiver operating characteristic (ROC) curve: the datapoints falling in the interval (i)  $L > 0$  (sure yes) were assigned the probability value  $p=1.0$ , those in (ii)  $-5 < L < 0$  (yes)  $p=0.66$ , those in (iii)  $-15 < L < -5$  (no)  $p=0.33$ , and those in (iv)  $L < -15$  (sure no)  $p=0.0$ . The AUC of the ROC is used as the “accuracy” metric to evaluate the performance of the model. For additional details see Material and Methods. Now, we are ready to examine the performance of the model in a series of experiments. We show examples of the different behaviour of the network trained on single (figure 7.4A-C) or four mixtures (figure 7.4D-F). As expected, the ability of the model to learn and distinguish the targets from the distractors depended crucially on the number of mixtures. The algorithm was implemented in Python and a sample code used to simulate Experiment 1 is available at the repository <https://github.com/GiorgiaD/dendritic-neuron-BSS>.

#### 7.2.4 Experiment 1: sound segregation with single and multiple mixtures of synthesized sounds

To begin with, we compared how the number of mixtures influences the learning performance between human subjects and the model. The number of mixtures presented during training was varied from 1, where no learning was expected, to 2 or more, where the model was expected to distinguish the target sounds from their respective distractors. The simulation protocol is shown in figure 7.5A (bottom). As reported in figure 7.5A (top), we obtained that, when one mixture only was shown, neither the target nor the mixing sound was learnt, and performance was close to chance. An immediate boost in the performance was observed when the number of mixtures was raised to two. The network managed to distinguish the learnt targets from the incorrect probes with an accuracy greater than 90%. As the number of mixtures increased up to six, the accuracy worsened slightly, remaining above 80%. A significant drop in the performance was observed for a greater number of mixtures. From a comparison with the results shown in figure 7.5B, which were replicated for human subjects [McDermott et al., 2011], it emerged that our model was able to partially reproduce human performance: the success rate was at chance levels when training consists of a single mixture only; the target sounds could be distinguished to a certain accuracy if more than a mixture was learnt. We also verified that the model performance was robust for variations of the network architecture, both in terms of the number of output neurons  $N$  and the connection probability  $p$  (Supplementary figure 1). Furthermore we observe that, while none of the output neurons exhibits an enhanced high firing rate when presented with the target sound, the overall population response to the target is substantially different from the response to the

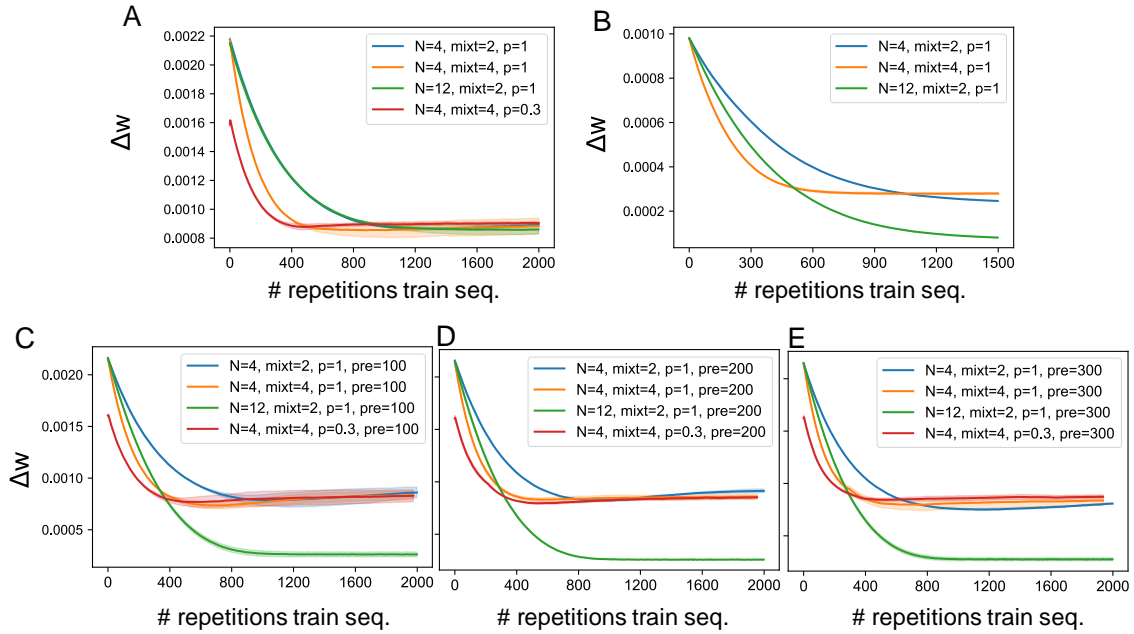


Figure 7.3: **Learning curves.** (A) Average synaptic weight change for the experiments carried out on the synthesized sounds, the network being initialized with random values. (B) Average synaptic weight change for the experiments carried out on the natural sounds, the network being initialized with random values. (C) Average synaptic weight change for the experiments carried out on the synthesized sounds, the network being pre-trained on the targets set presented for 100 epochs. (D) Average synaptic weight change for the experiments carried out on the synthesized sounds, the network being pre-trained on the targets set presented for 200 epochs. (E) Average synaptic weight change for the experiments carried out on the synthesized sounds, the network being pre-trained on the targets set presented for 300 epochs. The solid line and the shaded area represent the mean and standard deviation over 3 independent runs respectively. Without pretraining, when the number of output neurons is varied no significant change is found, while with pretraining when a larger number of neurons is used, the weight change curve saturates at a lower value, as shown by the blue ( $N=4$ ) and green ( $N=12$ ) curves. Furthermore, the figures show that both when a larger number of training mixtures is presented (yellow curves) and when only 30% of the connections are kept (red curves) the slope of the learning curve is steeper. The weight change is computed by storing the weights values every 2000 time steps (*i.e.*, “fire” or “non-fire” events) and computing the standard deviation over the last 100 recorded values. The standard deviation is then averaged across all connections from input to output neurons. Therefore, each point on the curve reports the average weight change over the past  $2000 \times 100$  time steps. Note that each sound/mixture is presented for 400 time steps. Finally, the x-axis shows the number of repetitions of the training mixture sequence (2000 for synthetic sounds and 1500 for naturalistic sounds).

masker sounds and to the incorrect probes.

Our model and human subjects also exhibited interesting differences. When the mixture number was increased to two, performance improved greatly in our model but only modestly in human subjects. Unlike human subjects, our model showed a decreasing accuracy as the number of mixtures further increased. We consider that such discrepancies may arise from a capacity limitation of the network. Indeed, the network architecture is very simple and consists of two layers only, whose size is limited by the spectrogram dimensions for the input layer and by the number of output neurons for the last layer. Therefore the amount of information that the network can learn and store is limited with respect to the significantly more complex structure of the human auditory system. We also suspect that the two-dimensional PCA projection might limit the model performance when a

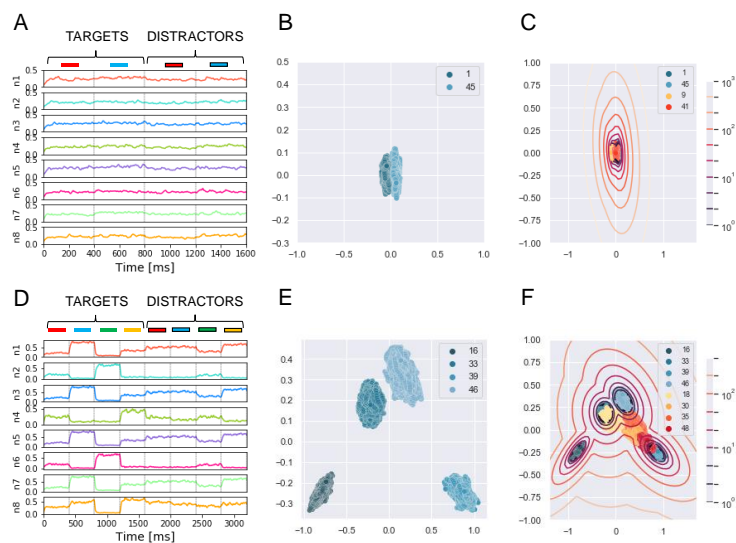


Figure 7.4: **Experiment 1 - output dynamics and clustering.** (A,B,C) refer to the results of Experiment 1 on synthesized sounds with a single mixture presented during training. (D,E,F) refer to the results of Experiment 1 on synthesized sounds with three mixtures presented during training. The “correct probes” are the target and the distractor sounds composing the mixtures presented during training, while the “incorrect probes” are sounds not presented during training. The numbers in the legends indicate the sound IDs. (A) Voltage dynamics of the 8 output neurons during inference, when the target, the distractor and the two associated incorrect probes are tested. The neuron population is not able to respond with different dynamics to the four sounds, and the voltage of all the output neurons fluctuates randomly throughout the whole testing sequence. (B) The PCA projection of the datapoints belonging to the two targets (in blue) shows that the clusters are collapsed into a single cluster. (C) When GMM is applied, all the datapoints representing both the correct probes (in blue) and the incorrect probes (in orange and red) fall within the same regions, making it impossible to distinguish the different sounds based on the population dynamics. (D) Voltage dynamics of the 8 output neurons during inference, when the four targets and the associated distractors are tested. As expected, the neuron population has learnt the feature of the different sounds and responds with different dynamics to the eight sounds. Each output neuron exhibits an enhanced response to one or few sounds. (E) The PCA projection of the datapoints belonging to the four correct probes (in blue) shows that the clusters are compact and spatially distant one from the other. (F) When GMM is applied, the model shows that the network is, most of the times, able to distinguish the target and distractors (in blue) from the incorrect probes (in yellow, orange and red). The correct probes are never overlapped. Three of the four distractors fall far from the targets’ region, while the fourth (in yellow) overlaps with one of the targets. These results are overall coherent with the human performance. In (C,F) the contour lines represent the landscape of the log-likelihood that a point belongs to one of the clusters associated to the correct probes.

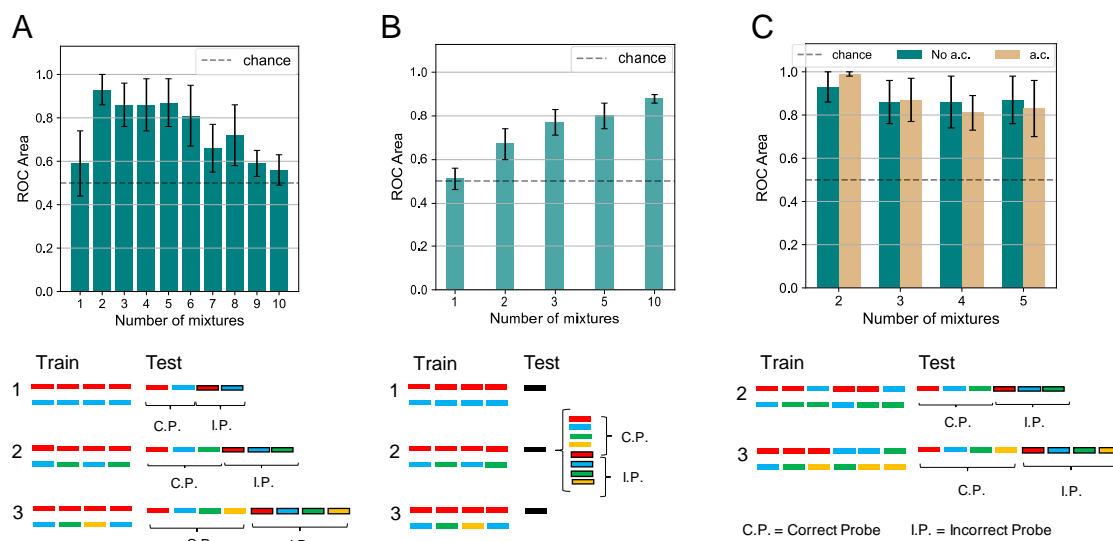
large number of distractors is used. Indeed the PCA space becomes very crowded and although the datapoints are grouped in distinct clusters, the probability that such a cluster lie close to each other is high. To verify this hypothesis, we tested a modification of the inference protocol of the algorithm. During test, we presented the network only with the target sound and one incorrect probe, and performed BSS on the PCA space containing the two sounds. Under this configuration, the model performance is above chance level for two or more different mixtures, and the accuracy does not significantly decrease for large number of mixtures (Supplementary figure 2).



We may use our model for predicting performance of human subjects in auditory perception tasks not yet tested experimentally. To this end, we propose an extension of the paradigm tested previously: for set-ups with the number of mixtures between two and five, we investigated whether presenting all possible combinations of the mixing sounds among themselves, rather than only the distractors with the target, affects the performance. The experiment is labelled “Experiment 1 a.c.”, where a.c. stands for “all combinations”, and its training scheme is reported in figure 7.5C. Because all sounds are in principle learnable in the new paradigm, we expect an enhanced ability of distinguishing the correct probes from the incorrect ones. Somewhat unexpectedly, however, our model indicated no drastic changes in the performance when the mixture sequence presented during training contained all possible combinations of the mixing sounds. Such a scheme resulted in a minor improvement in the accuracy only for the experiments with two mixing sounds. Indeed, in the “all combinations” protocol, during training the distractor was presented in more than one different mixture, while in the original task setting only the target was combined with different sounds. We hypothesize that the “all combinations” protocol makes it easier for the network to better distinguish the distractor sound. For four or five mixing sounds, instead, the performance slightly worsened. It is likely that this behaviour is related to the already mentioned capacity restraints of the network. Indeed, the length of the training sequence grows as the binomial coefficient  $\binom{n}{k}$  where  $k = 2$ , therefore for four and five targets (i.e., for  $n=4$  or  $5$ ) the number of mixtures is increased to 6 and 10, respectively.

### 7.2.5 Experiment 2: sound segregation with alternating multiple mixtures of synthesized sounds

Next, we investigate the model’s performance when the training sequence alternated mixtures of sounds with isolated sounds. An analogous protocol was tested in a psychophysical experiment (see experiment 3 in [McDermott et al., 2011]). Figures 7.6A and 7.6B show the network accuracy and human performance, respectively, for the protocols A,B,C in figure 7.6C. Only the target and the masker sounds were later tested since recognizing the sounds presented individually during training would have been trivial (See conditions B, C, 1, and 2 in figure 7.6C). In the alternating task, the network was only partially able to reproduce the human results, displaying an interesting contrast to human behaviour. In condition A, in which the sounds mixed with the main target (in red) changed during training, the listeners were able to learn the targets with an accuracy of about 80%, and so did our model. In contrast, our network behaved radically differently with respect to human performance under condition B, in which the training sequence consisted of the same mixture alternating with different sounds. As reported in figure 7.5B, the listeners were generally not able to identify the single sounds composing the mixture. Our model, instead, unexpectedly achieved a performance well above chance. The output dynamics could distinguish the distractors from the two targets with accuracy surprisingly above 90%. The behavioural discrepancy under condition B could be explained by considering that in the training scheme the network is presented with three different sounds besides the mixture. With respect to Experiment 1 with a single mixture, in this protocol the network could learn the supplementary features of the isolated sounds and could exploit them during inference to respond differently to the distractors. From the spectrograms shown in figure 7.2, it is evident that some regions of overlap exist between the higher-intensity areas of different sounds. Therefore, the network presented during training with isolated sounds in addition to the single mixture, could detect some similarities between the training sounds and the tested distractors and respond with a more defined output dynamics than in Experiment 1. Finally, under condition C, both human subjects and our model performed above chance. While human performance was slightly above 60%, the network achieved more than 90% accuracy. This result should be interpreted considering that during inference also the isolated sound (blue) was tested together with the associated distractor, which was a trivial task for the nature of our network and thus boosted its overall performance.



**Figure 7.5: Experiment 1 and 1 a.c. – results and comparison with human performance.** (A) Results and schematics for Experiment 1 on the dendritic network model. The number of mixtures is varied from 1 to 10. Performance is close to chance for a single training mixture. The performance is boosted as two mixtures are presented. As the number of mixtures is further increased, the clustering accuracy slowly decreases towards chance values. The protocol shown at the bottom of the panel illustrates that (i) in the training phase we feed the network only with the mixture(s), *i.e.*, target+masker sound(s). (ii) in the inference phase we feed the network only with the unmixed sounds (target, distractor separately) and with the incorrect probes (also unmixed sounds). We remark that in the case of one mixture (condition 1) the target and the masker sounds play the same role, while in the case of multiple mixtures (conditions 2 and 3) the target has a different role in the protocol as it is present in more than one mixture while the masker sounds are presented in one mixture only in the training sequence. (B) Results and schematics for Experiment 1 on the human experiment. The number of mixtures presented are 1,2,3,5 and 10. For a single mixture the performance is close to chance. As the number of mixtures increases, the classification accuracy improves steadily. Figure reproduced based on data acquired by [McDermott et al., 2011]. (C) Results and schematics for Experiment 1 a.c. on the dendritic network model. The number of mixtures is varied from 2 to 5. Combining all the mixing sounds in mixtures slightly improves the mean performance for two mixing sounds, while it slightly worsens it for a larger number of mixtures. The height of the bars and the error bars show respectively mean and standard deviation of the AUC over 10 independent runs.

### 7.2.6 Experiment 3: effect of temporal delay in target presentation with synthesized sounds

Temporal delay in the presentation of mixtures containing the target degraded performance similarly in the model and human subjects. We presented the network with a training sequence of six mixtures containing the same target mixed each time with a different distractor (figure 7.6C, protocols 0,1,2: *c.f.* experiment 4 in [McDermott et al., 2011]). The mixtures alternated with an increasing number of isolated sounds, hence increasing the interval between successive presentations of the target. The human ability to extract single sounds from mixtures was previously shown to worsen as the interval between target presentations increased, as replicated in figure 7.6B. The network presented a similar decreasing trend, as reported in figure 7.6A. An interesting difference, however, is that the performance of our model drastically dropped even with one isolated sound every other mixture while the human performance was affected when at least two isolated sounds separated the target-containing mixtures. The discrepant behaviour indicates that the insertion

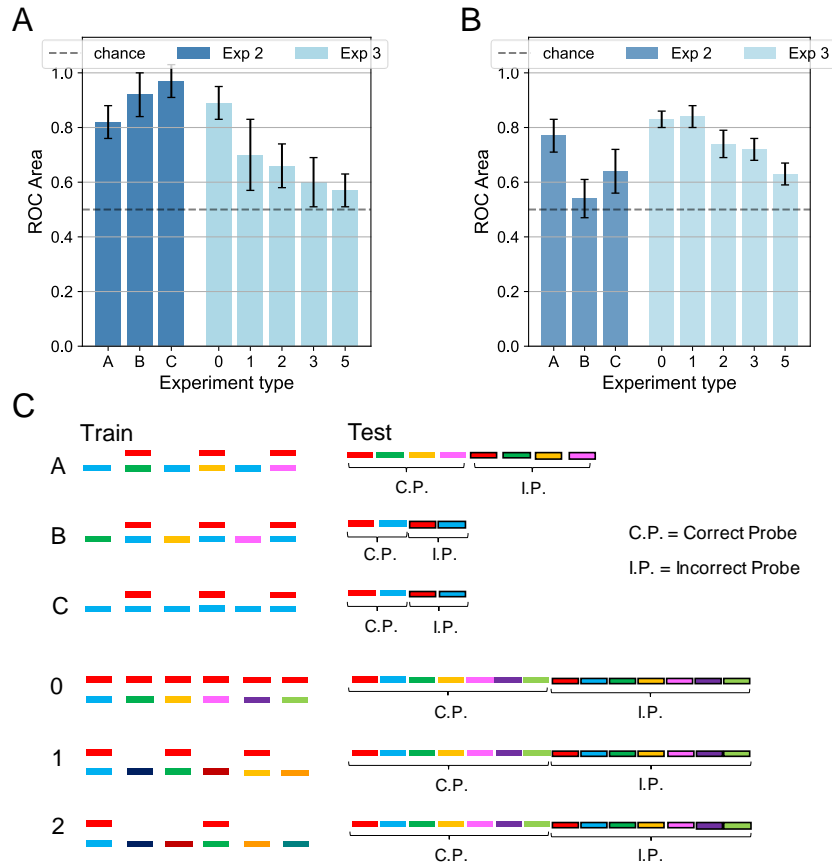


Figure 7.6: **Experiments 2 and 3 – results and comparison with human performance.** (A) Results for Experiments 2 (dark blue) and 3 (light blue) on the dendritic network model. In Experiment 2 the performance is above chance for the three conditions. In Experiment 3 the accuracy decreases as the number of isolated sounds alternating with the mixtures increases. (B) Results for Experiments 2 (dark blue) and 3 (light blue) on the human experiment. In Experiment 2 the performance is above chance in the conditions A and C, while it is random for condition B. In Experiment 3 the accuracy decreases as the target presentation is more delayed. Figure reproduced based on data acquired by [McDermott et al., 2011]. (C) Schematics for Experiments 2 and 3. The training is the same for both the dendritic network model and the human experiment. The schematics is omitted for delays 3 and 5. The testing refers to the dendritic network model, while the testing for the human experiment (same as in figure 7.5B) is omitted. In panels A and B, the height of the bars and the error bars show respectively mean and standard deviation of the AUC over 10 independent runs.

of isolated sounds between the target-containing mixtures more strongly interferes the learning of the target sound in the model compared to human subjects. This stronger performance degradation may partly be due to the capacity constraint of our simple neural model, which uses a larger amount of memory resource as the number of isolated sounds increases. In contrast, such a constraint may be less tight in the human auditory systems.

Also for Experiments 2 and 3, we tested a modification of the inference protocol, by presenting the network only with the target sound and one incorrect probe. Under this configuration, the model performance of Experiment 2 improves compared to the original protocol, while no substantial changes are noted for Experiment 3 (Supplementary figure 3).

### 7.2.7 Experiment 4: sound segregation with single and multiple mixtures of real-world sounds

We applied the same protocol of Experiments 1 to the dataset of natural sounds. Although such experiments were previously not attempted on human subjects, it is intriguing to investigate whether the model can segregate target natural sounds by the same strategy. The spectrograms of two isolated sounds and of their mixtures are shown in figure 7.7A-C, together with the respective sound waves (figure 7.7D-F). The qualitative performance was very similar to that obtained with the synthesized sounds. Specifically, the output dynamics learned from the repetition of a single mixture was randomly fluctuating for both seen and randomly chosen unseen sounds (figure 7.8A), whereas the network responses to targets and unseen sounds were clearly distinct if multiple mixtures were presented during training (figure 7.8D). The output dynamics were not quantitatively evaluated because it was not possible to rigorously generate incorrect probes associated with the learnt targets and distractors. Therefore, we qualitatively assessed the performance of the model by observing the clustering of network responses to the learnt targets versus unseen natural sounds (figure 7.8B, C, E and F). We observed that, in the case of multiple mixtures, the clusters related to natural sounds (figure 7.8E and F) were more compact than those of synthetic sounds (figure 7.4E and F). Furthermore, these clusters were more widely spaced on the PCA projection plane: the intraclass correlation in the response to the same target was greater while the interclass similarity in the response to different targets or distractors was lower. These results indicate that grouping cues, such as harmonic structure and temporal onset, improve the performance of the model.

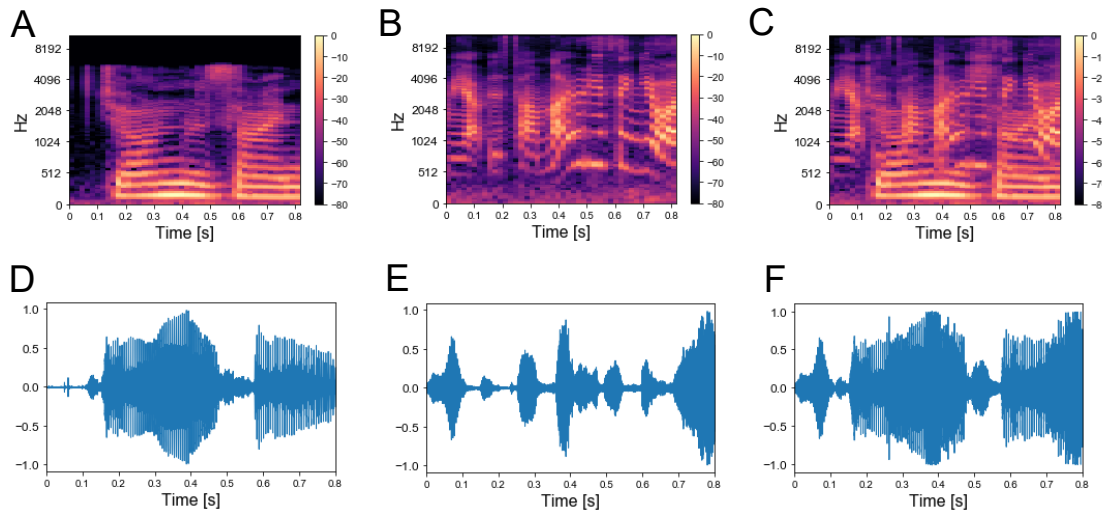


Figure 7.7: **Real-world sounds – Targets and mixture.** (A) Spectrogram of a spoken sentence 800 ms-long. (B) Spectrogram of 800 ms-long recording of chimes sounds. (C) Spectrogram of the mixture of the sounds in A and B. (D) Sound wave associated with the spectrogram in A. (E) Sound wave associated with the spectrogram in B. (F) Sound wave associated with the spectrogram in C.

### 7.2.8 Experiment 5: image segregation with single and multiple mixtures of real-world images

Finally, we examined whether the source segregation through repetition scheme can also extend to vision-related tasks, as previously suggested [McDermott et al., 2011]. To this end, we employed the same method as developed for sound sources and performed the recovery of visual sources with the protocol of Experiment 1. The mixtures were obtained by overlapping black-and-white images sampled from our visual dataset (Material and Methods), as shown in figure 7.9. Similarly to Experiment 4, the performance of the model was assessed only qualitatively in the visual tasks.

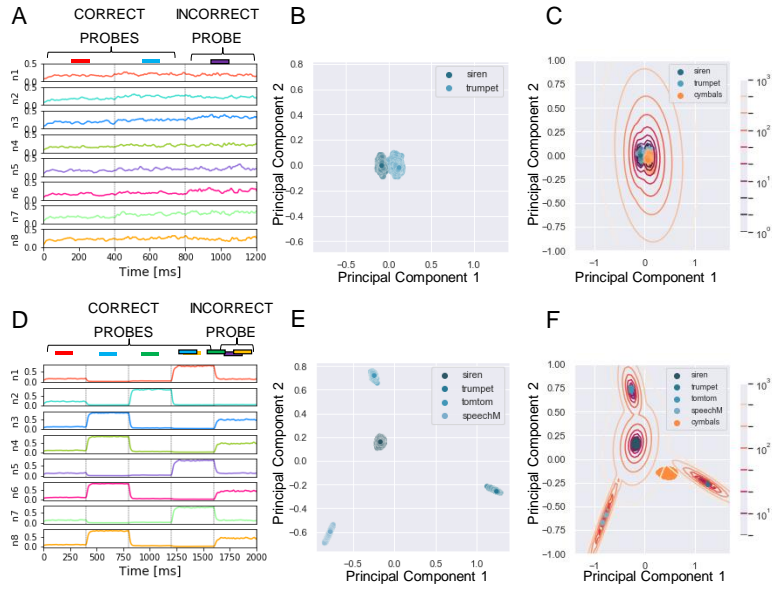


Figure 7.8: **Experiment 4 - output dynamics and clustering.** (A,B,C) refer to the results of Experiment 4 on real-world sounds with a single mixture presented during training. (D,E,F) refer to the results of Experiment 4 on real-world sounds with three mixtures presented during training. (A) Voltage dynamics of the 8 output neurons during inference, when the target, the distractor and one unseen sound are tested. As expected, the neuron population is not able to respond with different dynamics to the three sounds, and the voltage of all the output neurons fluctuates randomly throughout the whole testing sequence. (B) The PCA projection of the datapoints belonging to the target and distractor (in blue) shows that the clusters are collapsed into a single cluster. (C) When GMM is applied, all the datapoints representing both the learnt sounds (in blue) and the unseen sound (in orange) fall within the same regions, making it impossible to distinguish the different sounds based on the population dynamics. (D) Voltage dynamics of the 8 output neurons during inference, when the target, the three distractors, and one unseen sound are tested. As expected, the neuron population has learnt the feature of the different sounds and responds with different dynamics to the five sounds. Each output neuron has an enhanced response to one or few sounds. (E) The PCA projection of the datapoints belonging to the four correct probes (in blue) shows that the clusters are more compact and more spatially distant one from the other with respect to the result obtained with the synthesized sounds. (F) When GMM is applied, the model shows that the network clearly distinguished the learnt sounds (in blue) from the unseen sound (in orange). These results show that the grouping cues improve the model accuracy with respect to the synthesized dataset.

As in the acoustic tasks, the clustering of network responses showed that the model was able to retrieve the single images only when more than one mixture was presented during training. The network responses are shown in figure 7.10. We remark that the model is presented with the visual stimuli following the same computational steps as for sounds. Indeed, as previously described, the acoustic stimuli are first pre-processed into spectrograms and then encoded by the input layer. While it is not unexpected that similar computational steps lead to consistent results, we remark that the nature of the “audio images”, *i.e.*, the spectrograms, is substantially different to that of the naturalistic images, leading to very different distributions of the encoding spike patterns. Therefore, successful signal discrimination in the visual task strengthen our results, proving that our model is robust with respect to different arrangements of signal intensity.

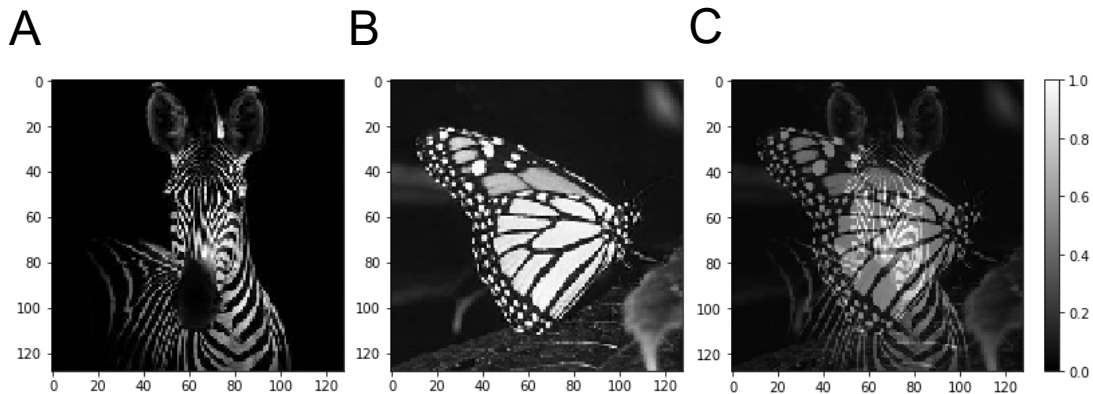


Figure 7.9: **Real-world images – Targets and mixture.** (A) Squared 128x128 target image of a zebra. (B) Squared 128x128 distractor image of a butterfly. (C) Mixture of the target and distractor images shown in A and B.

### 7.3 Discussion

The recovery of individual sound sources from mixtures of multiple sounds is a central challenge of hearing. Based on experiments on human listeners, sound segregation has been postulated to arise from prior knowledge of sound characteristics or detection of repeating spectro-temporal structure. The results of [McDermott et al., 2011] show that a sound source can be recovered from a sequence of mixtures if it occurs more than once and is mixed with more than one masker sound. This supports the hypothesis that the auditory system detects repeating spectro-temporal structure embedded in mixtures, and interprets this structure as a sound source. We investigated whether a biologically inspired computational model of the auditory system can account for the characteristic performance of human subjects. To this end, we implemented a one-layer neural network with dendritic neurons followed by a readout layer based on GMM to classify probe sounds as seen or unseen in the training mixtures. The results in [McDermott et al., 2011] show that source repetition can be detected by integrating information over time and that the auditory system can perform sound segregation when it is able to recover the target sound’s latent structure. Motivated by these findings, we trained our dendritic model with a learning rule that was previously demonstrated to detect and analyze the temporal structure of a stream of signals. In particular, we relied on the learning rule described by [Asabuki and Fukai, 2020], which is based on the minimization of regularized information loss. Specifically, such a principle enables the self-supervised learning of recurring temporal features in information streams using a family of competitive networks of somatodendritic neurons. However, while the learning rule has been designed to capture temporal information in an online fashion, in our framework we flatten the spectrogram before encoding it, making the spike pattern static during the stimulus presentation. Therefore, the temporal fluctuations are determined by the stochastic processes in the rate encoding step.

We presented the network with temporally overlapping sounds following the same task protocols as described in [McDermott et al., 2011]. First, we carried out the segregation task with the same dataset of synthesized sounds presented to human listeners in [McDermott et al., 2011]. We found that the model was able to segregate sounds only when one of the masker sounds varied, not when both sounds of the mixture were repeated. Our findings bear a closer resemblance to the experimental findings of human listeners over a variety of task settings. Earlier works have proposed biologically inspired networks to perform BSS [Isomura and Toyozumi, 2019, Bahroun et al., 2021, Pehlevan et al., 2017]. However, to our knowledge, this is the first attempt to reproduce the experimental results of recovering sound sources through embedded repetition. For this reason, we could not compare our results with previous works. Additionally, we demonstrated that our network can be a powerful tool for predicting the dynamics of brain segregation capabilities under settings difficult to test on humans. In particular, the recovery of natural sounds is expected

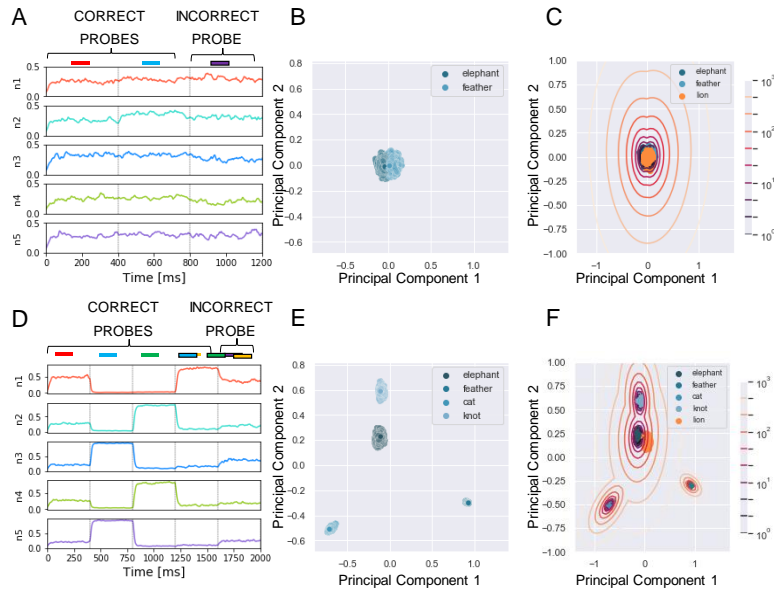


Figure 7.10: **Experiment 5 – output dynamics and clustering.** (A,B,C) refer to the results of Experiment 5 on real-world images with a single mixture presented during training. (D,E,F) refer to the results of Experiment 5 on real-world images with three mixtures presented during training. (A) Voltage dynamics of the 5 output neurons during inference, when the two training images and one unseen image are tested. As expected, the neuron population is not able to respond with different dynamics to the three images, and the voltage of all the output neurons fluctuates randomly throughout the whole testing sequence. (B) The PCA projection of the datapoints belonging to the two seen images (in blue) shows that the clusters are collapsed into a single cluster. (C) When GMM is applied, all the datapoints representing both the targets (in blue) and the unseen image (in orange) fall within the same regions, making it impossible to distinguish the different images based on the population dynamics. (D) Voltage dynamics of the 5 output neurons during inference, when the four targets and one unseen image are tested. As expected, the neuron population has learnt the features of the different images and responds with different dynamics to the five images. Each output neuron has an enhanced response to one or few inputs. (E) The PCA projection of the datapoints belonging to the four learnt images (in blue) shows that the clusters are compact and spatially distant one from the other. (F) When GMM is applied, the model shows that the network clearly distinguished the target and distractors (in blue) from the unseen image (in orange). These results suggest that humans would be able to distinguish single visual targets if previously seen in different mixtures.

to be a trivial task for humans given their familiarity with the sounds, whereas our model is built from scratch and has no prior knowledge about natural sounds. We find that the hallmarks of natural sounds make the task easier for the network when the target is mixed with different sounds, but, as for the synthetic dataset, the sounds cannot be detected if presented always in the same mixture. Furthermore, we extended the study to investigate BSS of visual stimuli and observed a similar qualitative performance as in the auditory settings. This is not surprising from a computational perspective as the computational steps of the visual experiment are the same as for the acoustic experiment: there, the sounds are first preprocessed into images, the spectrograms, and then presented to the network in a visual form. From the biological point of view, the neural computational primitives used in the visual and the auditory cortex may be similar, as evidenced by anatomical similarity and by developmental experiments where auditory cortex neurons acquire V1-like receptive fields when visual inputs are redirected there [Sharma et al., 2000,

[Bahroun et al., 2021](#)]. We point out, however, that such a similarity is valid only at high level as there are some substantial differences between visual and auditory processing. For instance, the mechanisms to encode the input signal into spikes rely on different principles: in the retina the spike of a neuron indicates a change in light in the space it represents, while in the cochlea the rate of a neurons represents the amplitude of the frequency it is associated to, like a mechanical FFT. Motivated by these reasons, we suggest extending the experiments of source repetition to vision to verify experimentally whether our computational results provide a correct prediction of the source separation dynamics of the visual system.

Although the dynamics of our model under many aspects matches the theory of repetition-based BSS, the proposed scheme presents a few limitations. The major limitation concerns the discrepancy of the results in experiment 2B. In such a setting, the model performance is well above chance, although the target sound always occurs in the same mixture. We speculate that, in this task settings, the output neurons learn the temporal structure of the distractor sounds presented outside the mixture and that they recognize some similarities in the latent structure of the probes. We note that the degree of similarity among distractors is the same as in the psychophysics experiment. This pushes the neurons to respond differently to the correct and incorrect probes, thereby allowing the output classifier to distinguish the sounds. In contrast, we speculate that human auditory perception relies also on the outcome of the later integration of features detected at early processing stages. This will prevent the misperception of sounds based on unimportant latent features. A second limitation of the selected encoding method consists in the difficulty to model the experiments relying on the asynchronous overlapping of signals and on reversed probe sounds presented by [\[McDermott et al., 2011\]](#). Indeed, in our approach, because of the flattening of the spectrogram in the encoding phase, each input neuron responds to one specific time frame, and the output neurons are trained uniquely on this configuration. Hence, temporal shifts or inverting operations are not possible. Third, we observed that in Experiment 1, as the number of mixtures increased over a certain threshold, the model’s accuracy degraded. We speculate that, in such settings, substituting PCA with a clustering algorithm not relying on dimensionality reduction, such as K-means, may help mitigate the issue. In addition, an interesting variation of our framework would be replacing the clustering step of the model with an another layer of spiking neurons. Fourth, the flattening of the spectrogram in the spike encoding stage is not biologically plausible and introduces high latency as the entire input signal needs to be buffered before the encoding starts. This strategy exhibits the advantage of making the length of the spike train fixed for any sound length, though modifications of the encoding scheme that preserves the signal’s temporal structure might be more suitable for applications tailored for real-world devices. Furthermore, an instantaneous identity coding approach, either from raw signal or via a spectrogram, would not be affected by the previously described issues related to the spectrogram normalization in the presence of outliers in signal intensity. Motivated by these points, in a follow up work we intend to explore an extension of the presented framework combining time frame-dependent encoding and spike-based post-processing clustering, which would allow us to integrate the model in embedded neuromorphic applications for sound source separation with reduced response latency. In this context, for further lowering the temporal latency, as well as for reducing the model’s energy consumption in neuromorphic devices, the time-to-first-spike encoding method could be explored as an alternative to the current rate coding approach.

Furthermore, as previously mentioned, the training scheme in [\[Asabuki and Fukai, 2020\]](#) has proven to be able to learn temporal structures in a variety of tasks. In particular, the model was shown to perform chunking as well as to achieve BSS from mixtures of mutually correlated signals. We underline that our computational model and experiments differ in fundamental ways from the BSS task described by [\[Asabuki and Fukai, 2020\]](#). First, the two experiments diverge in their primary scope. The BSS task aims at using the average firing rate of the single neurons responding to sound mixtures to decode separately the original sounds. In our work, instead, sound mixtures are included only in the training sequence and, during inference, only individual sounds are presented to the network. Our goal is to verify from the population activity whether the neurons have effectively learned the sounds and can distinguish them from unseen distractors. Furthermore, in [\[Asabuki and Fukai, 2020\]](#) the stimulus was encoded into spike patterns using one Poisson process



proportional to the amplitude of the sound waveform at each time step, disregarding the signal intensity at different frequencies. This method was not suitable for the source segregation through repetition task, where the sound mixtures retain important information on the frequency features of the original sounds at each time frame. Furthermore, we flatten the audio signal spectrogram before encoding it, unlike in the BSS task described by [Asabuki and Fukai, 2020].

In summary, we have shown that a network of dendritic neurons trained in an unsupervised fashion is able to learn the features of overlapping sounds and, once the training is completed, can perform blind source separation if the individual sounds have been presented in different mixtures. These results account for the experimental performance of human listeners tested on the same task setting. Our study has demonstrated that a biologically inspired simple model of the auditory system can capture the intrinsic neural mechanisms underlying the brain's capability of recovering individual sound sources based on repetition protocols. Furthermore, as the adopted learning scheme in our model is local and unsupervised, the network is self-organizing. Therefore, the proposed framework opens up new computational paradigms with properties specifically suited for embedded implementations of audio and speech processing tasks in neuromorphic hardware.

## 7.4 Materials and methods

### 7.4.1 Datasets

A dataset of synthesized sounds were created in the form of spectrogram, which shows how signal strength evolves over time at various frequencies, according to the method described previously [McDermott et al., 2011]. In short, the novel spectrograms were built as Gaussian distributions based on correlation functions analogous to those of real-world sounds. White noise was later applied to the resulting spectrograms. Five Gaussian distributions were employed to generate each of ten different sounds in figure 7.5A. The corresponding spectrograms featured 41 frequency filters equally spaced on an ERBN (Equivalent Rectangular Bandwidth, with subscript N denoting normal hearing) scale [Glasberg and Moore, 1990] spanning 20-4000 Hz, and 33 time frames equally dividing the 700ms sound length. For our simulations, we used the same MATLAB toolbox and parameters used in the previous study [McDermott et al., 2011]. For further details on the generative model for sounds, please refer to the SI Materials and Methods therein.

In addition to the dataset of synthesized sounds, we built a database composed of 72 recordings of isolated natural sounds. The database contained 8 recordings of human speech from the EUSTACE (the Edinburgh University Speech Timing Archive and Corpus of English) speech corpus [White and King, 2003], 23 recordings of animal vocalizations from the Animal Sound Archive [Frommolt et al., 2006], 29 recordings of music instruments by Philharmonia Orchestra [phi, 2019], and 12 sounds produced by inanimate objects from the BBC Sound Effect corpus [BBC, 1991]. The sounds were cut into 800ms extracts. Then the library *librosa* [McFee et al., 2015] was employed to extract spectrograms with 128 frequency filters spaced following the Mel scale [Stevens et al., 1937] and 10ms time frames with 50% overlap.

For image source separation, we built a database consisting of 32 black-and-white pictures of various types, both single objects and landscapes. The images were later squared, and their size was reduced to 128x128 pixels.

### 7.4.2 Neuron model

In this study we used the same two-compartment neuron model as that developed previously [Asabuki and Fukai, 2020]. The mathematical details are found therein. Here, we only briefly outline the mathematical framework of the neuron model. Our two-compartment model learns temporal features of synaptic input given to the dendritic compartment by minimizing a regularized information loss arising in signal transmission from the dendrite to the soma. In other words, the two-compartment neuron extracts the characteristic features of temporal input by compressing the high dimensional data carried by a temporal sequence of presynaptic inputs to the dendrite onto a low dimensional manifold of neural dynamics. The model performs this temporal feature

analysis by modifying the weights of dendritic synapses to minimize the time-averaged mismatch between the somatic and dendritic activities over a certain recent interval. In a stationary state, the somatic membrane potential of the two-compartment model could be described as an attenuated version of the dendritic membrane potential with an attenuation factor [Urbanczik and Senn, 2014]. Though we deal with time-dependent stimuli in our model, we compare the attenuated dendritic membrane potential with the somatic membrane potential at each time point. This comparison, however, is not drawn directly on the level of the membrane potentials but on the level of the two non-stationary Poissonian spike distributions with time-varying rates, which would be generated if both soma and dendrite were able to emit spikes independently. In addition, the dynamic range of somatic responses needs to be appropriately rescaled (or regularized) for meaningful comparison. An efficient learning algorithm for this comparison can be derived by minimizing the Kullback–Leibler (KL) divergence between the probability distributions of somatic and dendritic activities. Note that the resultant learning rule enables unsupervised learning because the somatic response is fed back to the dendrite to train dendritic synapses. Thus, our model proposes the view that backpropagating action potentials from the soma may provide a supervising signal for training dendritic synapses [Larkum, 2013, Larkum et al., 1999].

### 7.4.3 Network architecture

The network architecture, shown in figure 7.1, consisted of two layers of neurons, either fully connected or with only 30% of the total connections. The input layer contained as many Poisson neurons as the number of pixels present in the input spectrogram (acoustic stimulus) or input image (visual stimulus). The postsynaptic neurons were modelled according to the two-compartment neuron model proposed previously [Asabuki and Fukai, 2020]. Their number was varied from a pair to few tenths, depending on the complexity of the task. Unless specified otherwise, 8 and 5 output neurons were set for acoustic and visual task respectively.

In the first layer, the input was encoded into spikes through a rate coding-based method [Almomeni et al., 2019]. The strength of the signal at each pixel drove the firing rate of the associated input neuron, i.e. the spike trains were drawn from Poisson point processes with probability proportional to the intensity of the pixel. We imposed that, for each input stimulus, the spike pattern was generated through a sequence of 400 time steps, where each time step corresponds to a “fire” or “non-fire” event.

We designed the output layer and the learning process similarly to the previous network used for the blind signal separation (BSS) within mixtures of multiple mutually correlated signals as well as for other temporal feature analyses [Asabuki and Fukai, 2020]. As mentioned previously, the learning rule was modelled as a self-supervising process, which is at a conceptual level similar to Hebbian learning with backpropagating action potentials. The soma generated a supervising signal to learn and detect the recurring spatiotemporal patterns encoded in the dendritic activity. Within the output layer, single neurons learned to respond differently to each input pattern. Competition among neurons was introduced to ensure that different neurons responded to different inputs. With respect to the network used for BSS containing only two output neurons, we rescaled the strength of the mutual inhibition among dendritic neurons by a factor proportional to the inverse of the square root of the number of output neurons. This correction prevented each neuron from being too strongly inhibited when the size of the output layer increased (i.e. exceeds three or four). Furthermore, we adopted the same inhibitory spike timing-dependent plasticity (iSTDP) as employed in the previous model. This rule modified inhibitory connections between two dendritic neurons when they coincidentally responded to a certain input. The iSTDP allowed the formation of chunk-specific cell assemblies when the number of output neurons was greater than the number of input patterns.

For all parameters but noise intensity  $\xi_i$  during learning, we used the same values as used in the original network model [Asabuki and Fukai, 2020]. For bigger values of noise intensity  $g$ , the neural responses were subject to more fluctuations and neurons tended to group in only one cell assembly. From the analysis of the learning curves shown in figure 7.3, we decided to train the network from randomly initialized weights and to expose it, during training, to the mixture sequence 3000 times for the synthesized sounds and 1500 times for the real-world sounds. The learning rate was

kept constant throughout the whole process. During testing, the sequence of target sounds and respective distractors was presented 50 times, and the resulting neural dynamics was averaged over 20 trials. The performance results shown in the Results section were computed as average over 10 repetitions of the same simulation set-up. In each repetition different target sounds and distractors were randomly sampled from the dataset in order to ensure performance independence of specific sounds.

#### 7.4.4 Experimental settings and performance measure

The synapses were kept fixed during inference in our network, implying that the responses to probes tested later were not affected by the presentation of other previously tested probes. This allowed us to test the trained network on a sequence of probes, rather than only on one probe as in the studies of the human brain where plasticity cannot be frozen during inference [McDermott et al., 2011]. In figure 7.5A and figure 7.6C, the first half of the sequence contained the target and the distractors, the second half the respective incorrect probes, which were also built by using the same method as in human experiment [McDermott et al., 2011]. Each incorrect probe was a sound randomly selected from the same Gaussian distribution generating the associated target. After the sampling, a randomly selected time slice equal to  $1/8$  of the sound duration was set to be equal to the target. The possibility of presenting more than one probe allowed us to test the performance of the network for all the sounds present in the mixtures. To ensure a stable neural response against the variability of the encoding, we repeated the sequence 50 times. The response of the network consisted of the ensemble activity of the output neurons. As previously explained, 400 time-steps were devoted to the presentation to each stimulus. The response to each probe, therefore, consisted of 400 data points describing the dynamical activity of each output neuron, each point being a collection of  $N$  values, where  $N$  is the number of output neurons. An example of one testing epoch output is shown in figure 7.4A and figure 7.4C. We neglected the first 50 data points, since, during the initial transient time, the membrane potential was still decaying or rising after the previous input presentation. For visualization purpose, we applied the principal component analysis (PCA) to reduce the dimensionality of the data from  $N$  to 2. In our settings, the two principal components explain approximately 40% of the variance of the neural response. The PCA transformation was based uniquely on the data points obtained with the presentation of the target and the distractors, as shown in figure 7.4B and Figure figure 7.4E. The same transformation was later exploited to project the points related to the incorrect probes. Only the target and distractors patterns were presented during the learning process, and the responses to unseen patterns were afterwards projected on the space defined by the training.

The two-dimensional projection of the target-related data points were clustered in an unsupervised manner through GMM. We set the number of Gaussians equal to the number of targets such that the covariance matrices had a full rank. With the defined GMM model at hand, we proceeded with fitting all the PCA data points, related to both correct and incorrect probes. The model tells which cluster each data point belonged to and what was the likelihood ( $L$ ) that the cluster had generated this data point. figure 7.4C and figure 7.4F show the datapoints projected on the PCA plane together with the GMM clustering and likelihood curves.

We used the likelihood as a measure of performance. The four intervals of the likelihood range corresponding to the four responses “sure no”, “no”, “yes”, and “sure yes” were (i)  $L > 0$  (sure yes), (ii)  $-5 < L < 0$  (yes), (iii)  $-15 < L < -5$  (no), and (iv)  $L < -15$  (sure no). In building the receiver operating characteristic (ROC) curve, the datapoints falling in the interval (i) were assigned the probability value 1.0, those in (ii) 0.66, those in (iii) 0.33, and those in (iv) 0.0.

The described evaluation metrics was applied only to the experiments carried on the dataset composed of synthesized sounds. For the experiments based on natural sounds and images, the results of clustering were shown only qualitatively for the target-related datapoints. Indeed, due to the real-world nature of signals, it was not possible to simply use Gaussian functions to build physically consistent incorrect probes. On the real-world sound dataset, we performed all the same protocol of Experiment 1 (Experiment 4). On the image dataset we performed an experiment with a protocol analogous to Experiment 1. Here, the mixtures were obtained by overlapping two

---

images, both with transparency 0.5, similarly to the spectrogram overlapping described for the acoustic task. The input images were normalized to the range  $[0,1]$  and the intensity of each pixel was encoded through the firing rate of one input neuron. We followed the same procedure and network setting described for the audio stimuli segregation to assess the ability of the network to separate visual stimuli presented in mixtures.

## Chapter 8

# Fooling the Primate Brain with Small, Targeted Image Manipulation

Artificial neural networks (ANNs) such as deep convolutional neural networks are considered the current best models of biological vision. Despite the successes of ANNs at computer vision applications and describing neural responses, these models are fragile, fail to generalize, and show significant differences with neural circuits and perception. Here we investigate a suspected disparity between ANNs and biological vision in their sensitivity to small, targeted image manipulations. We proposed an array of image manipulation methods to change the primate brain’s representation of an image, in a targeted, category-specific way while limiting the amount of pixel value change. We generated such ‘deceptive images’ of human faces, monkey faces, and noise patterns so that they are perceived as a different, pre-specified target category. Measuring both monkey neuronal responses and human behavior to these images, we found several effective methods for changing primate visual categorization that required significantly smaller image change compared to untargeted noise. Our work shares the same goal with adversarial attack, namely the manipulation of images with minimal, targeted noise that leads ANNs to misclassify the images. Our results quantify and characterize the differences in perturbation robustness between biological and artificial vision.

**Human or monkey?** DALL·E prompt: A watercolor painting of a monkey that is looking at a photo of a human face.



### 8.1 Introduction

Deep artificial neural networks (ANNs), which mimic the approximately hierarchical structure of the visual cortex, currently constitute the most accurate models of biological vision [Yamins et al., 2014, Cadena et al., 2019, Schrimpf et al., 2018]. ANNs not only are good predictors of ventral stream neural activity, but also can be used to design images that drive prescribed patterns of neuronal responses [Bashivan et al., 2019, Walker et al., 2019, Ponce et al., 2019]. Despite the success of ANN-based models at describing neural responses and perceptual judgements to some visual stimuli, these models remain fragile, fail to generalize, and differ in essential aspects from neural circuits and perception [Serre, 2019]. To build better models, it is essential to understand the conditions where current models diverge from neural data and behavior.

To stress-test current vision models, here we designed several approaches to image synthesis—including methods based on an ANN encoding model of neuronal activity—to specifically ‘fool’ both

neuronal responses and behavior in a binary image categorization task. These ‘deceptive images’ are designed such that they, (i) when presented to macaques, are miscategorized by selective neurons as the target category; and (ii) when presented to human observers, mislead their perception during a visual categorization task. We investigated three directions of creating ‘deceptive images,’ changing human faces to look like monkey faces, monkey faces to look like human faces, and noise patterns to look like human faces.

Deceptive images provide a valuable testing ground for computational models of vision for several reasons. First, minimally morphing images to change perception is an open challenge. Prior work has used ANN models to guide the *de novo* synthesis of images to drive specific patterns of neuronal activity [Bashivan et al., 2019, Walker et al., 2019]. Image synthesis represents a more stringent test of model prediction of neuronal responses compared to testing prediction on held-out (and typically, identically distributed) images. To the extent that visual neurons underlie perception [Salzman et al., 1990, Parvizi et al., 2012, Moeller et al., 2017, Rajalingham and DiCarlo, 2019], robust and general models that can predict the response of visual neurons should also allow us to design images that, by activating a certain pattern of visual neuronal responses, drive a specific behavior.

Second, images perturbed with category-specific noise are likely to generate neural responses that are challenging to explain for current ANN-based vision models. This is because such deceptive images are related to adversarial attacks, the phenomena that adding carefully crafted, minute noise to an image can cause ANNs to misclassify it with high confidence [Szegedy et al., 2013, Goodfellow et al., 2014b]. Prior studies have measured human perceptual responses to adversarial images designed for ANNs [Elsayed et al., 2018, Zhou and Firestone, 2019] and to random image distortions [Geirhos et al., 2018]. However, empirical data are lacking on how visual perception and neurons respond to different types of category-specific image manipulations that relate to adversarial images.

We developed and tested several methods for generating deceptive images, including linear interpolation toward the target class, CycleGAN translation [Zhu et al., 2017], generating adversarial images for an ensemble of ANNs [Elsayed et al., 2018], and ‘gray box’ methods using an ANN-based predictive model of neuronal responses. We generated deceptive images at various noise levels and collected monkey neuronal and human behavioral responses. This data allowed us to directly compare the robustness of biological and artificial vision to diverse types of category-specific image manipulations. Furthermore, the methods we developed for creating deceptive images open up a novel experimental paradigm for studying neuronal processing and its link to behavior.

## 8.2 Results

### 8.2.1 Deceptive images of human faces evoked monkey face-like neuronal responses

We started by designing deceptive images of human face images that would elicit monkey face-like responses (*human*→*monkey* or *h2m* manipulation) in face-selective monkey neurons. We tested a wide range of image manipulation noise strengths (*noise levels*) and a variety of manipulation methods.

To choose a noise range that was likely to progress from unsuccessful to successful semantic flip, we reasoned that image category is trivially changed if an original-class image is replaced by an image from the opposite class. Thus, we computed the distance distribution between unmodified human and monkey faces, considering 250 images in each category. Image distance was quantified by the mean-squared error (MSE) using vectorized pixel values (range: 0–255). At constant image resolution, MSE is equivalent to the square of the  $l_2$ -norm (i.e., Euclidean distance), a common metric in the adversarial attack literature. We elected to use MSE instead of the  $l_2$ -norm because the latter depends on the number of pixels. The typical human-to-monkey pairwise distance in our image set was  $\text{MSE} = 6500 \pm 2300$  (mean  $\pm$  stdev for all numbers in text unless otherwise noted). However, for successful semantic change, it suffices to replace a human face with the closest monkey face. With 250 images in each category, the minimum human-to-monkey distance

was  $2800 \pm 700$ . We thus reasoned that an upper range of  $\text{MSE} = 800$  was reasonable to test, as it was safely lower than the budget for simply replacing the image. We selected a lower range of  $\text{MSE} = 200$  because this amount of image change was barely perceptible. We tested 10 evenly spaced noise levels covering this range of MSE.

The first manipulation method we considered, as a reference, was to linearly interpolate between a human face and the closest monkey face. This method is guaranteed to achieve complete success at a high enough noise level. Second, we included a slight modification by interpolating toward the closest monkey face up to an optimized affine transform. Other transformations, such as reducing contrast, can further reduce the distance to the target image. However, we did not test further transformations because an excessively transformed image will eventually cease to represent the target category.

Third, we included a method based on CycleGAN [Zhu et al., 2017]. CycleGAN can learn to translate between two image categories without any paired examples. CycleGAN does not explicitly optimize pixel-level proximity between the original and translated image. Nevertheless, the translated image is closely related to the original image, because the original (as opposed to another instance from the original class) can be approximately recovered from the translated image using a reverse translation. We trained a CycleGAN on 500 human faces and 300 monkey faces. Using the trained network, we translated each original human face into the monkey class, then linearly interpolated between the two at defined noise levels.

Fourth, we tested adversarial images created for an ensemble of convolutional neural networks (CNNs). Prior work [Elsayed et al., 2018] shows that adversarial images designed for CNNs can bias human perception, but only under severe viewing time limits and backward masking where accuracy on unmodified images is reduced to around 65% (when chance is 50%). We reproduced this method in our setting without the same viewing time limit. To create deceptive images between human and monkey faces, we used CNNs pre-trained on ImageNet [Deng et al., 2009b] and fine-tuned them on the two face categories together with the original 1000 categories. We built an ensemble of 14 fine-tuned CNNs comprising Inception, ResNet [He et al., 2016b], ResNeXt [Xie et al., 2017], DenseNet [Huang et al., 2017b], and SENet [Hu et al., 2018], and created adversarial images for the model ensemble using iterative gradient descent.

Lastly, we designed a manipulation method tailored to primate vision by building a substitute model of macaque visual neuron responses (**Figure 8.1a**). The model comprised a ResNet-101 from the fine-tuned CNN ensemble above, fitted with a linear mapping module that used features extracted from the last convolutional layer to predict neuronal responses. Similar CNN-based models comprise the current best models of primate visual neuronal responses [Schrimpf et al., 2018]. The linear mapping was trained on responses of 22 face patch neurons in one monkey (of two in this study) to around 1,000 pictures of objects [Konkle et al., 2010]. The model could explain around 40% of the neuronal response variance on held-out images not used during fitting. We used this neuron-fitted model, which was end-to-end differentiable, to stand for the primate visual system for adversarial attack. To create deceptive images, we used two variants of objective functions and two variants of optimization algorithms, resulting in four total variants. The objective function was either 1) to maximize responses in one model neuron that produced monkey-like features in feature visualization, labelled the *single-neuron* method; or, 2) to match the model-predicted response pattern to the empirical neuron population response pattern to monkey faces, labelled the *pattern* method. The objective function was optimized using either 1) iterative gradient descent, or 2) iterative gradient descent coupled with  $l_2$ -projection ( $l_2$ -PGD), i.e., projection to a fixed noise level (MSE, equivalent to  $l_2$ ) at each step of gradient descent. We refer to these variants collectively as *gray box* deceptive images, as they use a limited amount of information about the system (primate vision) being targeted.

Each method was used to modify 40 unmodified or *clean* human faces, except the single-neuron method, the pattern method, and their  $l_2$ -PGD versions, which were each used to modify 20 images. Each method produced one deceptive version of each clean image at every noise level. In total, we tested a total of 2,400 targeted human→monkey deceptive images.

We presented the deceptive images, together with 250 unmodified human faces and 250 monkey faces to two monkeys in a passive fixation task and recorded neuronal responses using chronically

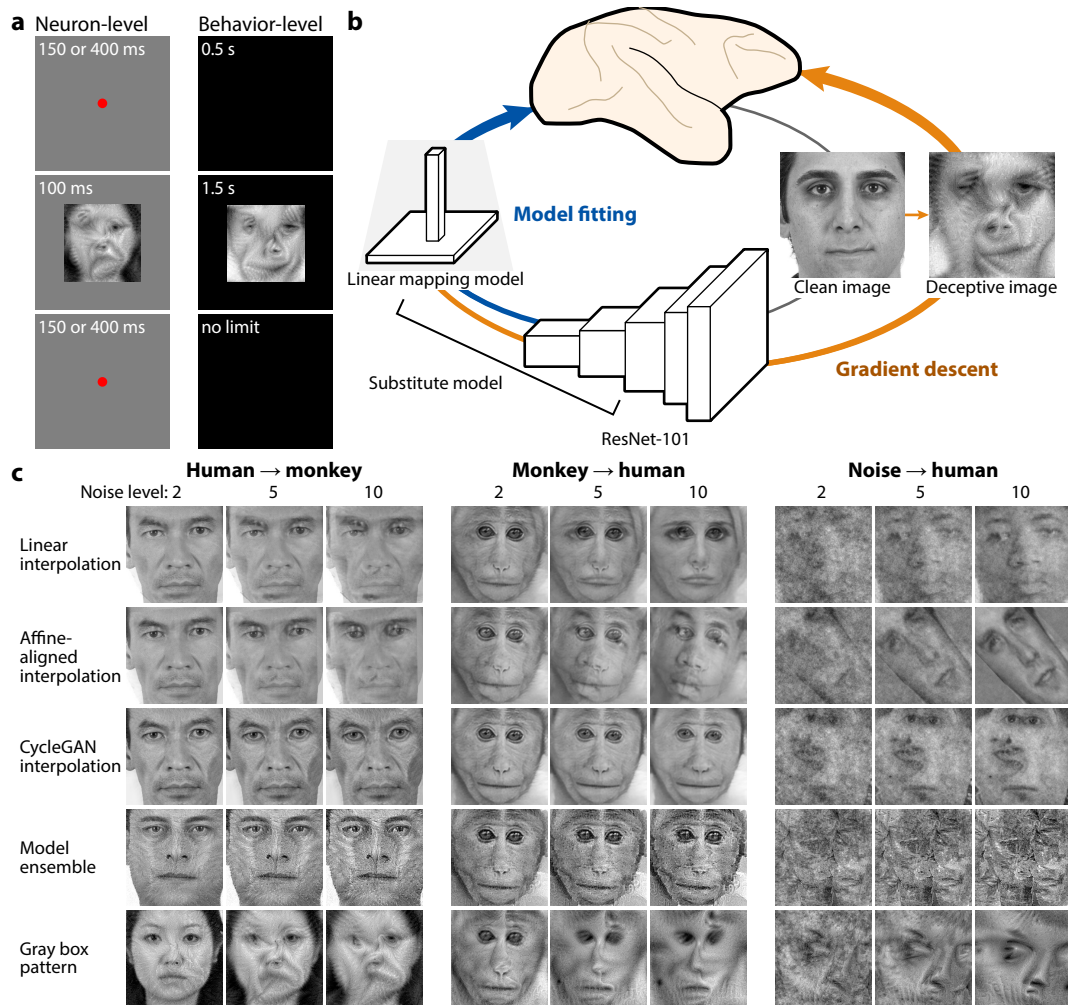


Figure 8.1: Overview of image manipulations targeting categorization change. **a**, The left column illustrates monkey neuron-level experiments. Monkeys fixated on a red fixation point while images were presented in random order and neuronal responses were recorded. Images were presented for 100 ms; inter-stimulus interval was 150 ms for monkey 1 and 400 ms for monkey 2, whose responses extended over a longer duration. The right column illustrates behavioral experiments with human subjects. Each image was presented for 1.5 s. Humans were instructed to categorize the image by pressing a key. There was no time limit for a response. The text in the figure was not included in the experiment. **b**, A substitute model was fit on IT neuron responses and used to generate gray box deceptive images. The substitute model consisted of a pre-trained ResNet-101 (excluding the final fully-connected layer) and a linear mapping model. Deceptive images were generated by gradient-based optimization of the image to create a desired neuronal response pattern as predicted by the substitute model. **c**. Example images for human→monkey deceptive images (left), monkey→human deceptive images (center), and brown noise→human face deceptive images (right) are shown for different noise levels and different image manipulation methods.



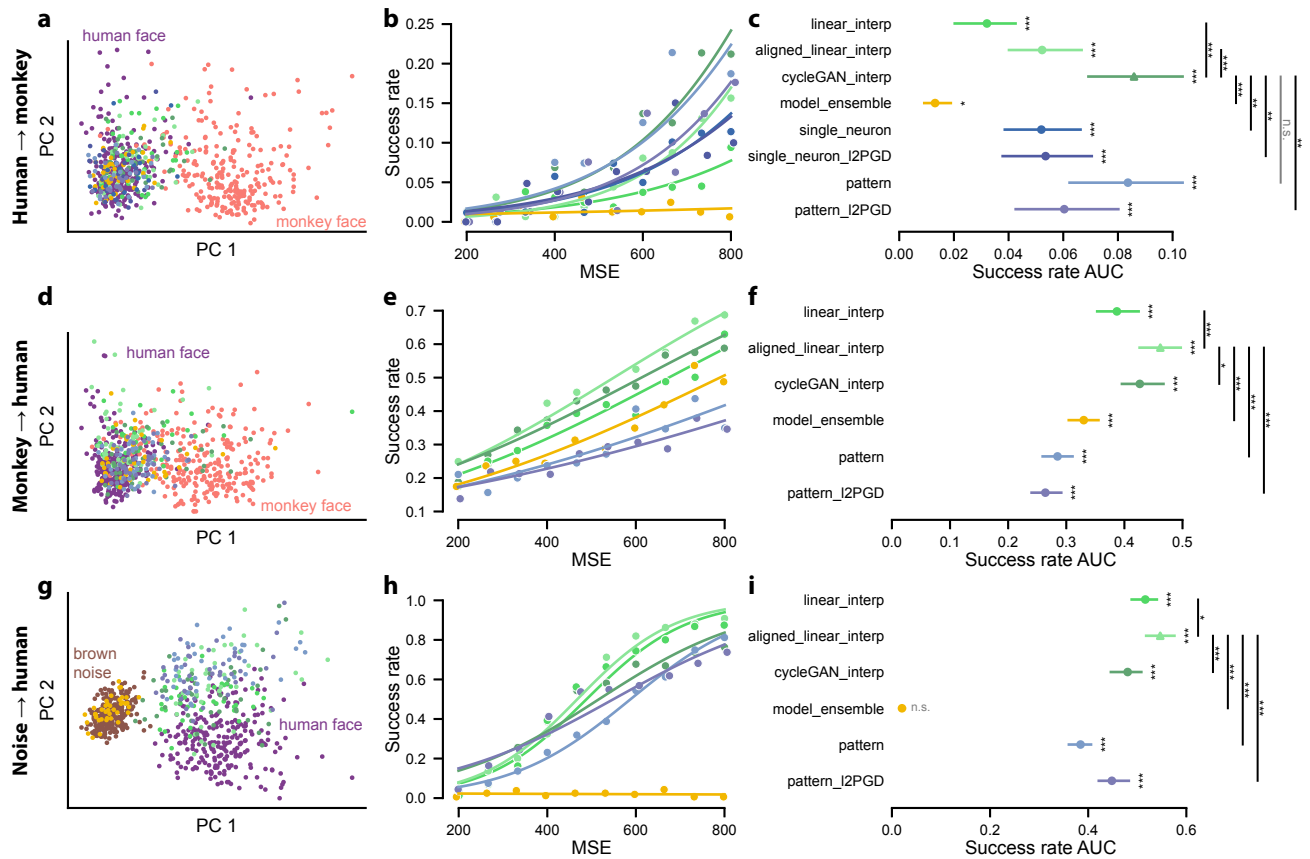


Figure 8.2: Responses to deceptive images by face-selective neurons in monkey IT. **a–c** corresponds to human→monkey deceptive images. **a**, The scatter plot shows a UMAP visualization of the neuron population representation of images, showing clean human faces, monkey faces, and level 10 deceptive images. The neuron pseudopopulation included IT neurons from the face patches of two monkeys. The colors indicate image manipulation methods and are labeled in **c**. **b**, The plot shows success rate as a function of noise level (MSE). The scatter points represent method-level (image-averaged) success rate per noise level. The lines represent logistic regression fit on image-level success rates. **c**, The plot shows success rate AUC per manipulation method. Each point and line indicate the mean and bootstrap 95%-CI of the mean, respectively. The triangle marker signifies the method with the maximum AUC. The annotations on the right indicate results of pairwise statistical tests for a difference in AUC. The annotations next to individual methods indicate whether success rate increased with noise level using a permutation test. \*,  $p < 0.05$ ; \*\*,  $p < 0.01$ ; \*\*\*,  $p < 0.001$ ; and n.s., not significant, all p-values FDR-corrected for multiple comparisons. **d–f** are the same as **a–c** but correspond to monkey→human deceptive images. **g–i** are the same as **a–c** but correspond to brown noise→human deceptive images.

implanted multi-electrode arrays. Specifically, we recorded from face patches in inferior temporal cortex (IT), which contain neurons that are strongly selective for faces over non-face objects and that extract face features including species and identity [Tsao et al., 2006, Moeller et al., 2008]. To collect enough repeat presentations to reliably measure neuron responses, deceptive images were presented in 4 sessions of 2–3 noise levels each.

We visualized the neuron population representation of clean and deceptive images (level 10) using Uniform Manifold Approximation and Projection (UMAP) [McInnes et al., 2018] (**Figure 8.2a**). Neuronal representations of clean human and monkey faces were clearly separable. Some deceptive images were shifted away from human faces towards monkey faces. To quantify the fraction of the deceptive images that would be categorized as the target category, we trained support vector machines (SVMs) to classify clean images as human or monkey based on neuronal responses. The SVMs were then used to classify held-out clean images as well as the deceptive images. The SVMs achieved high test accuracy on held-out clean human and monkey faces ( $98.3 \pm 0.3\%$  and  $97.7 \pm 0.5\%$ , mean  $\pm$  sem), confirming that the recorded neurons distinctly represented human and monkey faces. Nevertheless, a fraction of human $\rightarrow$ monkey deceptive images were classified as monkey faces. We call these images *successful*, and calculated *success rate* as the fraction of successful images per method at each noise level). The success rate ranges from 0% to 100%, and 50% does not indicate chance levels. The success rate of almost all manipulation methods increased with noise level (**Figure 8.2b**), as reflected in the positive coefficient in logistic regression of individual image success as a function of noise level (**Figure 8.2c**). The best manipulation method was CycleGAN-based interpolation, which reached a success rate of  $21 \pm 4\%$  (mean  $\pm$  sem) at noise level 10, followed by the gray box pattern method ( $19 \pm 6\%$ ), pattern  $l_2$ -PGD ( $18 \pm 5\%$ ), and affine-aligned linear interpolation ( $16 \pm 4\%$ ). To summarize and compare across methods, we calculated the area under the success rate curve (AUC) using the logistic fits. We normalized the x-value range to 0–1, so that the AUC value is roughly equivalent to the average success rate over levels. The best method, CycleGAN-based interpolation, was significantly more successful than all other methods except the gray box pattern method ( $p = 0.068$ ; permutation test, FDR-corrected across 7 tests; **Figure 8.2c**).

Logistic regression also allowed us to infer how much image change would be needed to achieve 50% successful deception with each method). Since no method achieved 50% success rate within the tested noise levels, this midpoint estimate is necessarily an extrapolation. Nevertheless, an estimated midpoint may be useful to compare to other directions of image manipulation (below) and to the much larger MSE separating clean images. CycleGAN interpolation would require  $\text{MSE} = 1030$  (bootstrap 95%-CI: 930–1150) to achieve 50% success rate. Simply replacing a human face with the closest monkey face among 250 images, while guaranteeing complete success, requires  $\text{MSE} = 3210$  (95%-CI: 1670–4490; this differs from 2800 reported above because the clean images subjected to manipulation are different from the 250 images used for establishing a baseline). This closest distance increases to 4560 and 6800 for closest among 25 and 3 images respectively, a roughly 1.5-fold increase for each order of magnitude; extrapolating in the other direction suggests that even if the trend continues, there may only be one in over  $10^5$  monkey face images that is  $\text{MSE} \approx 800$  away from an average human face image. Interpolating halfway between a human face and the closest monkey face among 250 entails  $\text{MSE} = 800$  (‘Interp. halfway, closest in 250’; 95%-CI: 420–1120), but that likely corresponds to lower-than-50% success rate because empirically, linear interpolation at  $\text{MSE} = 800$  achieved only  $10 \pm 4\%$  success rate. Overall, these comparisons suggest that the semantic change success achieved by CycleGAN-based interpolation and the pattern method was not explained by trivially replacing the original image.

Were deceptive images being misclassified simply because noise degraded image quality? To control for this possibility, we tested two types of control image modifications at noise level 10. The first was image degradation, including Gaussian noise, Gaussian blur, phase scrambling, and Eidolon images at three coherence levels [Koenderink et al., 2017, Tramèr et al., 2020]. The second was versions of targeted deceptive images where the image change (additive pixel value change) was re-applied flipped upside-down [Elsayed et al., 2018].

Considered together, deceptive images achieved significantly higher success rate than either clean human face images or non-targeted image modifications ( $p = 1 \times 10^{-4}$ ; permutation test, FDR-

corrected across 2 tests. Considering each method individually, among control images, flipped CycleGAN images achieved the highest success rate ( $16 \pm 3\%$ ), a success rate that was statistically no lower than success in all targeted manipulation methods (all pairwise  $p > 0.13$ ; permutation test, uncorrected). This notwithstanding, the next best among 14 control methods, the flipped single neuron method, achieved only  $4 \pm 3\%$  success rate. Thus, we speculate that flipped CycleGAN images still contain features of a monkey face to which neurons are sensitive. To anticipate, results below from human categorization support the conclusion that flipped CycleGAN images were not perceived as monkey faces.

### 8.2.2 Deceptive images also led to target category-like responses in two other directions

So far, we described human faces modified to look like monkey faces (*human*  $\rightarrow$  *monkey* or *h2m* manipulation). In adversarial attacks in CNNs, a given category can be attacked to target *any* other category [Szegedy et al., 2013]. Therefore, we attempted to create deceptive images in two other directions: the reverse *monkey* $\rightarrow$ *human* (*m2h*) direction and *brown noise* $\rightarrow$ *human face* (*b2h* or *noise* $\rightarrow$ *human*) direction, a shift between two distant categories. We tested these directions with the same methods, except that for the gray box method, we only tested the *match-response-pattern* objective and not the *excite-single-neuron* objective.

Unexpectedly, monkey $\rightarrow$ human manipulation of neuronal responses was generally easier than the human $\rightarrow$ monkey direction, as reflected by overall higher success rates for most methods at most noise levels (**Figure 8.2e,f**) and by the highest success rate achieved at noise level 10 ( $69 \pm 34\%$  for affine-aligned linear interpolation). UMAP visualization (**Figure 8.2d**) shows that, unlike in human $\rightarrow$ monkey manipulation, many monkey $\rightarrow$ human deceptive images had the target category (clean human faces) as the majority of near neighbors. However, success rate was also high for control image modifications, including untargeted image degradation. This cannot be because neurons did not separate the categories of monkey and human faces; the categories were clearly separated, as we established above in human $\rightarrow$ monkey direction and further verified with the particular experimental sessions here (accuracy on human and monkey faces:  $98 \pm 5\%$  and  $98 \pm 8\%$ ). Instead, we speculate that the idiosyncrasies of the neuronal selectivity or the particular monkey images used in the manipulation could have prevented SVM generalization. Failing to certify the SVM-based quantification, we could not meaningfully compare different manipulation methods or estimate the noise threshold for successful image manipulation. We will show below that human behavior distinguished effective targeted deceptive images from control image modifications. Further counter to our intuition, brown noise $\rightarrow$ human face manipulation was even easier than the previous two directions. The most successful method, as quantified by AUC, was affine-aligned linear interpolation, which was significantly better than all other targeted methods ( $p \leq 0.014$ ; **Figure 8.2i**) and reached  $91 \pm 17\%$  success rate at level 10, approaching the accuracy on clean human face images ( $98 \pm 5\%$ ). Linear interpolation (level 10 success rate:  $87 \pm 20\%$ ) and CycleGAN based-interpolation ( $76 \pm 34\%$ ) were also highly effective, together with gray box manipulation methods (pattern:  $81 \pm 24\%$ ; pattern  $l_2$ -PGD:  $74 \pm 25\%$ ). The model ensemble method was ineffective, achieving  $0.6 \pm 4\%$  success rate at level 10, close to the error rate on brown noise images ( $0.4 \pm 2\%$ ). UMAP visualization (**Figure 8.2g**) corroborates the quantification: Most interpolation-based images intermixed with clean human faces, gray box images were close to but formed a distinct cluster from human faces, and model ensemble images remained embedded in the noise images cluster. The most successful control image modification (Gaussian blur) achieved only  $16 \pm 4\%$  success at level 10, significantly lower than success rates of all targeted methods (all  $p = 2 \times 10^{-5}$ ; permutation test, FDR-corrected across 6 tests) excluding the model ensemble method ( $p = 0.175$ ), indicating that targeted manipulation methods were specific. Using fitted logistic regression, the estimated noise threshold for 50% successful image manipulation by the best method (affine-aligned interpolation) was  $\text{MSE} = 470$  (95%-CI: 450–490), much lower than that for simply replacing the noise image with the closest human face ( $\text{MSE} = 1850$ , 95%-CI: 1530–2220), although close to that for interpolating half way between the two ( $\text{MSE} = 460$ , 95%-CI: 380–550). *Post hoc*, we could rationalize the relative ease of noise $\rightarrow$ human semantic flip by

suggesting that a face superimposed on noise can still look like a face. This may be attributed to the heightened sensitivity of primates in detecting faces (pareidolia), or the fact that noise images do not themselves belong to any category that can supply competing evidence to the positive evidence of a face.

To summarize, we tested changing image categorization by monkey neuronal responses through targeted image manipulation in three directions. Within the range of image change tested, we achieved moderate success in one direction (human→monkey) and high-to-complete success in the other two (monkey→human and brown noise→human face). In human→monkey manipulation, although the highest success rate was lower than 50%, the success was not explained by the control method of interpolating the original image toward the target. In noise→human manipulation, close to complete success was achieved by control interpolation methods. We could not interpret the results in monkey→human manipulation since most types of non-targeted image degradation also achieved high success. We conclude that the ease of deceptive images for monkey neurons was highly dependent on the involved categories and shift direction. In all manipulation directions, deceptive images tailored to CNNs (the model ensemble method) did not affect monkey neuronal representation, consistent with the general intuition that CNN adversarial images do not affect biological vision. There was no consistent evidence that the model ensemble method was more successful in the initial part of the response in a time-resolved analysis.

### 8.2.3 Deceptive images reveal mismatch between primate vision and ANNs

ANNs are expected to differ from primates in responding to adversarial attack, not least because ANNs are highly sensitive to ANN-tailored adversarial images (the model ensemble method), which barely affected humans and monkeys. We measured ANN adversarial robustness in the same setting as the primate experiments, testing the same ensemble of 14 fine-tuned CNNs used to generate model ensemble deceptive images. Individual network outputs (logits) were averaged, then converted into a confidence vector over 1002 classes using the softmax function.

In human→monkey and monkey→human directions with the model ensemble method, success rate was already saturated at the lowest noise level 1, an expected result since noise level 1 is already large relative to the typical threshold reported in the machine learning literature. Testing additional images at lower noise levels, we verified that CNNs could be fooled with a minuscule amount of image change, with 50% success thresholds at  $\text{MSE} = 2.1$  (95%-CI: 2.0–2.2) and 4.3 (95%-CI: 4.1–4.5) respectively for the two manipulation directions. Compared to the threshold for categorization change for primates as a minimum over methods, the threshold for CNN thresholds were over 100 times lower. Granted, we could much more readily optimize deceptive image for CNNs than for primates. Nevertheless, this ratio is a direct, quantitative comparison of model and primate adversarial robustness.

Unexpectedly, the brown noise→human face direction for CNN models was difficult, requiring a manipulation threshold at  $\text{MSE} = 5310$  (95%-CI: 5290–5340)—even higher than the budget for simply replacing the noise image with the closest human face ( $\text{MSE} = 1880$ ; 95%-CI: 1620–2210), an approach that would have led to perfect success. This counterintuitive finding indicated that a simple gradient descent algorithm could not find the optimal deceptive image at such high noise levels. The criterion for noise→human shift success might be more stringent than in other directions, because the deception counted as successful only if confidence in the human category was higher than confidence in all other 1000 categories (excluding the monkey category). In contrast, in human→monkey manipulation for example, confidence in the human category only needed to exceed confidence in the monkey category. Most adversarial attack studies use the stricter one-vs-all criterion. However, to the best of our knowledge, we are not aware of studies that attempted adversarial attack starting from a noise image.

How did CNNs respond to deceptive images derived from other manipulation methods? Gray box methods were the second-most effective group of methods in human→monkey and monkey→human direction, although not in noise→human direction. In the human→monkey direction, gray box methods were almost as effective as the model ensemble approach and achieved 80–100% success

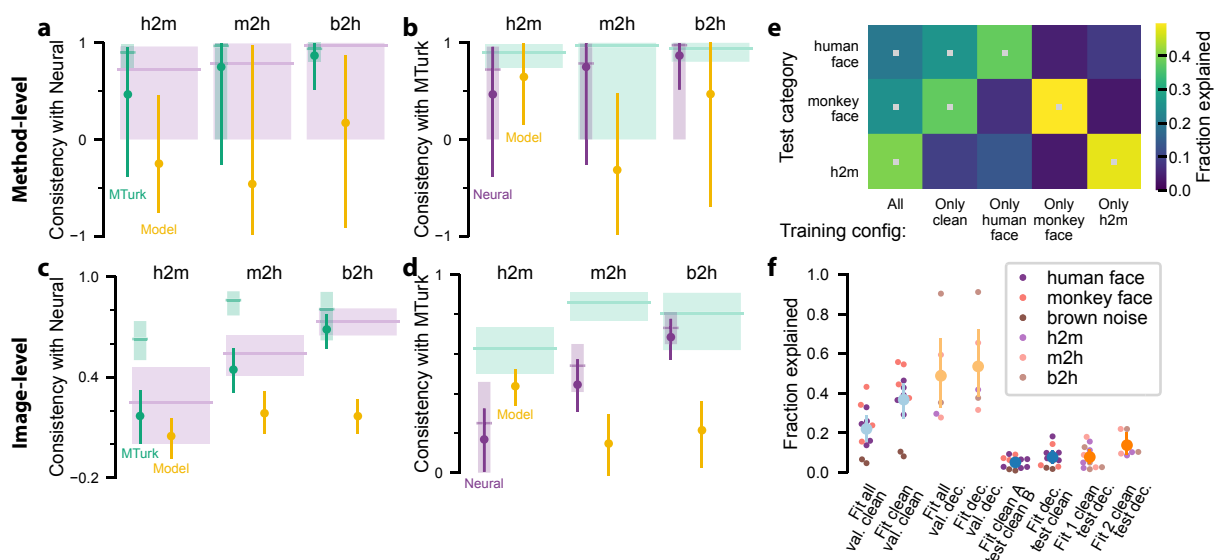


Figure 8.3: Deceptive images reveal mismatch between primate vision and ANNs. **a, b**, The plots illustrate the similarity, among three visual systems, in method-level (image-averaged) success of noise-level-10 deceptive images. We compared monkey neuron responses (‘Neural’), human behavior (‘MTurk’), and ANN model categorization (‘Model’). The similarity in the pattern of deception success was quantified by Pearson’s correlation across methods (**a, b**) or images (**c, d**). **a, c**, The success pattern in monkey neurons was compared to human and model. **b, d**, The success pattern in human was compared to monkey and model. The scatter points and vertical lines indicate correlation between visual systems (mean and bootstrap 95%-CI of the mean). The horizontal lines and shaded areas indicate inter-subject split-half self-consistency (mean and CI). **e**, The heat map shows the performance of ANN-based predictive models of monkey neuronal responses, in conditions of interpolation (testing on held-out images from trained categories) and extrapolation (testing on images from held-out categories). Model performance was quantified by the fraction of response predicted (see Methods). Categories included in training are indicated by small grey squares in the heat map. **f**, results in **e** are summarized and combined over different deceptive image directions. Small dots indicate individual cell values as in **e**, color coded by the image category. Larger dots with whiskers indicate mean and bootstrap 95%-CI of the mean within each training configuration, color coded by whether the result corresponds to interpolation (lighter color) or extrapolation (darker color) performance; and whether the tested category was clean (blue) or deceptive (orange) images.

rate starting from noise level 1. This result is unexpected, because although gray box methods used a ResNet as the feature-extraction backbone, the objective function was independent of weights in the classification layer of the CNN. Indeed, the gray box approach was not as effective as model ensemble manipulation in the monkey→human direction. In the noise→human direction, linear interpolation was the most effective method, followed by CycleGAN interpolation and model ensemble manipulation.

Thus, the overall pattern of manipulation method effectiveness was different between CNNs and primates. We directly compared CNNs to primates by correlating perceptual change success on the method- and individual image-levels as above (**Figure 8.3a–d**). At the method level, correlation between model and primate was negative in 3 out of 6 comparisons (monkey or human results, 3 manipulation directions). At the image level, model-primate correlation was always positive, but lower than primate-primate correlation in 5 out of 6 conditions (in the exception case, monkey-human correlation was upper-bounded by low internal consistency).

This divergence between CNN and primate behavior suggests that CNN internal representation may also be less similar to neuronal representation than generally thought. To quantify this, we used the well-established approach of fitting a linear readout from internal features of a CNN (pre-trained on ImageNet object categorization) to predict neuronal responses. Typically, CNN-based predictive models of neuronal responses are tested on *interpolation* to a held-out random subset of images that, by construction, are (in expectation) identically distributed as the training images. Here, we tested holding out entire categories of images, such as deceptive images, to test model *extrapolation* to images that may come from a different distribution. An example scheme is shown in **Figure 8.3e**. Using data involving human→monkey deceptive images, we held out 1–3 categories of images (human faces, monkey faces, and/or deceptive images), fitted a linear ridge regression model from features extracted from ResNet-101, and tested the model on interpolation (unseen images from categories seen during training) or extrapolation (images from categories unseen during training). The models consistently performed much better on held-out images from categories included in training than on images from completely held-out categories. The same analysis, performed and pooled over all three manipulation directions, is presented in **Figure 8.3f**. When testing on a held-out subset of image categories seen during training, model validation performance was relatively high (first four groups in **Figure 8.3f**). However, when fitting on a subset of categories and testing on entirely held-out categories, performance was much lower (second four groups). Unexpectedly, generalization was about as poor from one clean image category to another (fifth group) as it was from deceptive images to clean images (sixth group) and vice versa (last two groups). Thus, although deceptive images revealed behavioral differences between CNNs and primates, the lack of generalization in CNN-based models of neurons could be already revealed by different categories of clean images, which likely had different statistics because they came from different image datasets. This conclusion did not change when we systematically repeated the analysis in **Figure 8.3e,f** on eight ANN models ranging from classical (ResNet-101, DenseNet169, and AlexNet) to state-of-the-art (Visual Transformer, EfficientNet[Xie et al., 2020], and CLIP[Radford et al., 2021]) to biologically focused (CORnet[Kubilius et al., 2019] and VOneNet[Dapello et al., 2020]); and across four regression methods (ridge regression, partial least squares, principal component regression, and factor analysis followed by linear regression) in combination with the best tested model, the Visual Transformer (**Supplementary Figures 8.4 and 8.5**).

**Substitute model.** To generate deceptive images, we trained a substitute model to predict IT neuronal responses. The model was fixed before generating and testing deceptive images. The substitute model comprised a CNN (a ResNet-101 [He et al., 2016b] pre-trained on ImageNet) and a linear mapping model (**Figure 8.1B**). Because the 1,000 pre-trained categories did not include our categories of interest (human face and monkey face), we collected around one thousand images for these two categories for fine-tuning the ResNet-101. We trained the linear mapping model to use image features extracted by the last convolutional layer (layer conv5\_3) to predict neuronal responses. The linear model was factorized in the spatial and feature dimensions [Klindt et al., 2017]. The spatial module was a convolutional kernel  $W_s$ . The feature module was a mixing pointwise convolution  $W_t$ , i.e., a weighted sum over the feature dimension. Both  $W_s$  and  $W_t$  were parame-

terized separately for each IT neuron. Thus, the response for neuron  $n$  to image  $x$  was modeled as

$$\hat{y}_n = \sum (W_s^n * \text{ResNet}_{\text{conv5}_3}(x)) * W_t^n + W_d^n, \quad (8.1)$$

where  $*$  denotes elementwise multiplication and  $W_d^n$  is a bias parameter. The parameters were jointly optimized to minimize a loss function  $\mathcal{L}_e$  composed of the prediction error  $\mathcal{L}_p$ , an L2-regularization loss  $\mathcal{L}_2$ , and a spatial smoothness loss  $\mathcal{L}_{\text{laplace}}$ :

$$\mathcal{L}_p = \sqrt{\sum_n (\hat{y}_n - y_n)^2} \quad (8.2)$$

$$\mathcal{L}_2 = \lambda_s \sum \|W_s\|_2 + \lambda_t \sum \|W_t\|_2 \quad (8.3)$$

$$\mathcal{L}_{\text{laplace}} = \lambda_l \sum W_s * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (8.4)$$

$$\mathcal{L}_e = \mathcal{L}_p + \mathcal{L}_2 + \mathcal{L}_{\text{laplace}} \quad (8.5)$$

The hyper-parameters  $\{\lambda_s, \lambda_t, \lambda_l\} = \{1, 0.1, 0.7\}$  were obtained from a grid-search to produce the highest prediction accuracy. The substitute model achieved an average correlation of 0.4 between predicted and actual IT neuron responses on held-out test images.

**Model prediction of neuronal responses in Figure 8.3** Instead of using the method described above to build a substitute model, for the analysis in **Figure 8.3**, we used a simpler and computationally faster modeling approach following Zhuang et al. [Zhuang et al., 2021], which is in turn similar to prior work [Yamins et al., 2014, Schrimpf et al., 2018]. We fit linear models that predicted neuronal responses from features extracted by an ImageNet-trained ResNet-101 using partial least squares (PLS) regression with 5 retained components. We searched for the optimal ResNet layer (out of 105 layers including raw pixels) to use for each monkey and array, using data not included in the results. We chose layer ‘layer3.5.conv1’ for monkey 1 and ‘layer3.6.conv1’ for monkey 2. When training on different combinations of image categories, the number of training images may differ, so we equalized the total number of training images to 250 (the lowest number across training conditions) by random subsampling. We training using five-fold cross-validation, i.e., holding out each 20% of the data in turn for testing. All reported model performance values were on held-out images. PLS regression and stratified cross-validation was implemented by the Python package ‘scikit-learn’ [Pedregosa et al., 2011].

The fraction of neuronal responses explained by a model was quantified similar to prior work [Bashivan et al., 2019, Schrimpf et al., 2018, Zhuang et al., 2021]. Specifically, we calculated the fraction explained per neuron as the square of a ratio whose nominator was the Pearson correlation between model prediction and neuron responses, and whose denominator was the split-half self-consistency of that neuron (Pearson’s  $r$ , Spearman-Brown corrected). This value resembles fraction of variance explained, i.e., coefficient of determination or  $R^2$ , although they are not equivalent unless an additional optimal linear transform was fit between the predicted and actual neuronal responses on the test set to remove any mean and scale differences. The coefficient of determination, or  $R^2$ , was not suited for describing model performance on held-out data;  $R^2$  was not directly used in the cited prior studies either. To produce one summary value for each training-testing configuration and each data set, we took first the average over splits and then the median over neurons.

**Model prediction of neuronal responses in Supplementary Figures 8.4 and 8.5** To verify that the results of Figure 8.3 were not specific to one ANN model and the one fitting method we tested, we extended the analysis to systematically test eight ANN models and four regression methods. Using ridge regression, we tested three ‘classical’ CNN models (ResNet-101, DenseNet169, and AlexNet), three state-of-the-art ANN models (Visual Transformer, EfficientNet[Xie et al., 2020], and CLIP[Radford et al., 2021]), and two biologically inspired models (CORnet[Kubilius et al., 2019] and VOneNet[Dapello et al., 2020]). Table ?? reports the layers used for each model, chosen based on independent (not included in the results in this ) recording sessions from the same monkeys. Additionally, we tested the Visual Transformer model in combination with four regression methods: ridge regression ( $\alpha = 2 \times 10^5$ ), partial least squares (PLS, number of components = 27), principal component regression (PCR, number of components = 100), and factor analysis followed by linear regression (Factor, number of components = 250). All regression methods were implemented in the Python package ‘scikit-learn’.

Model	Layer	As implemented in
ResNet-101	layer3.5.conv1 (monkey 1), layer3.6.conv1 (monkey 2)	Python ‘torchvision’ package
DenseNet169	features.norm5	Python ‘torchvision’ package
AlexNet	features.10	Python ‘torchvision’ package
Visual Transformer	blocks.10.norm2	Python ‘timm’ package
EfficientNet	blocks.3.0.conv_dw	Python ‘timm’ package
CLIP	visual.transformer.resblocks.5.mlp.c_proj	Github repo for Radford et al., 2021[Radford et al., 2021]
CORnet	V4.conv_input	Github repo for Kubilius et al., 2019[Kubilius et al., 2019]
VOneNet	module.model.layer2.3.conv2	Github repo for Dapello et al., 2020[Dapello et al., 2020]

Table 8.1: The ANN models and layers that were used to build predictive models of neuronal responses. The implementation and source of pretrained weights are indicated for each model.

**Between-experiment consistency in Figure 8.3** The consistency was calculated as Pearson’s correlation coefficient on method- (Figure 8.3a, b) or image-averaged success rate values (Figure 8.3c, d). Inter-subject self-consistency (among monkeys or humans) was calculated as the correlation between data halves split by subjects. The inter-subject correlation values were clipped at 0 and corrected using the Spearman-Brown formula.

## 8.3 Discussion

Perception does not merely mirror reality, as evinced by centuries of visual illusions and a longer history of art. For ANNs, adversarial images can be interpreted as a kind of visual illusion, since adversarial images are almost identical to originally unambiguous images, yet come to be confidently miscategorized. We were motivated to study deceptive images for primate vision similar to adversarial images because only ANNs have been shown to be sensitive to such low levels of image change, while primates are thought not to be susceptible [Szegedy et al., 2013, Elsayed et al., 2018, Zhou and Firestone, 2019]. Thus, images related to adversarial attack provide an ideal testing ground for ANNs as the best current models of the visual cortex [Yamins et al., 2014, Cadena et al., 2019, Schrimpf et al., 2018].

We designed a variety of methods for creating targeted deceptive images to alter category-selective monkey neuronal responses and human visual perception. Several methods specifically changed primate visual categorization beyond chance levels. Revealingly, using a CNN-based model of neurons to create deceptive images was neither as successful as the model predicted, nor always more successful than a naive method of simply interpolating toward a target class image. Indeed, ANN-based models, when only trained on one image category (such as human faces), could not



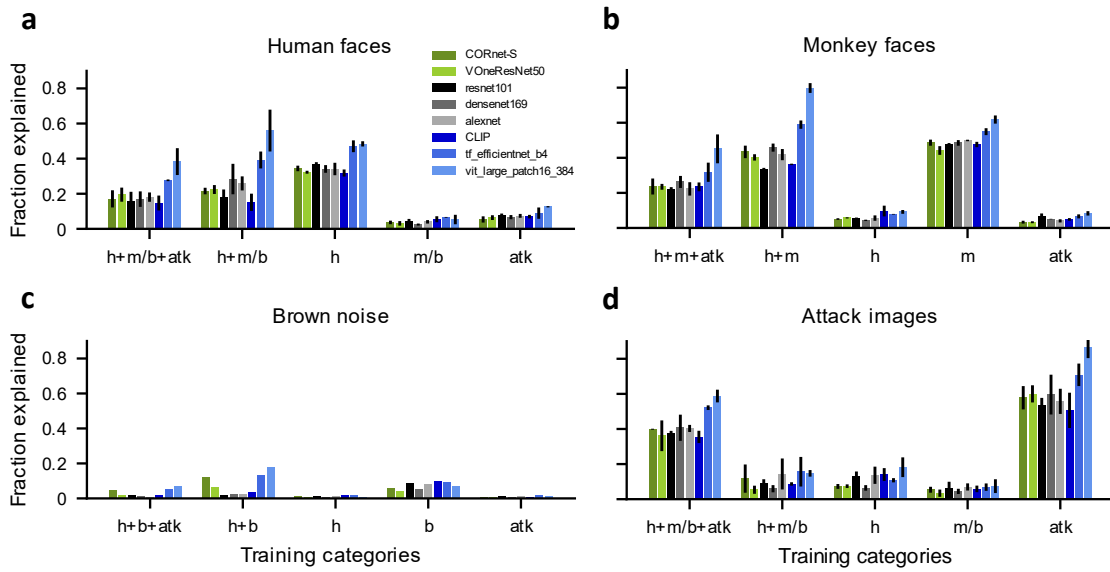


Figure 8.4: Related to **Figure 8.3e–f**. The interpolation and generalization performance in eight ANN-based predictive models of neuronal responses. We tested three ‘classical’ CNN models (ResNet-101, DenseNet-169, and AlexNet), three state-of-the-art ANN models (Visual Transformer, EfficientNet[Xie et al., 2020], and CLIP[Radford et al., 2021]), and two biologically inspired models (CORnet[Kubilius et al., 2019] and VOneNet[Dapello et al., 2020]). The bar plots illustrate model performance in different training and testing configurations that correspond to either interpolation or generalization. Each panel corresponds to testing on one image category. Each condition on the x-axis corresponds to one combination of training categories.

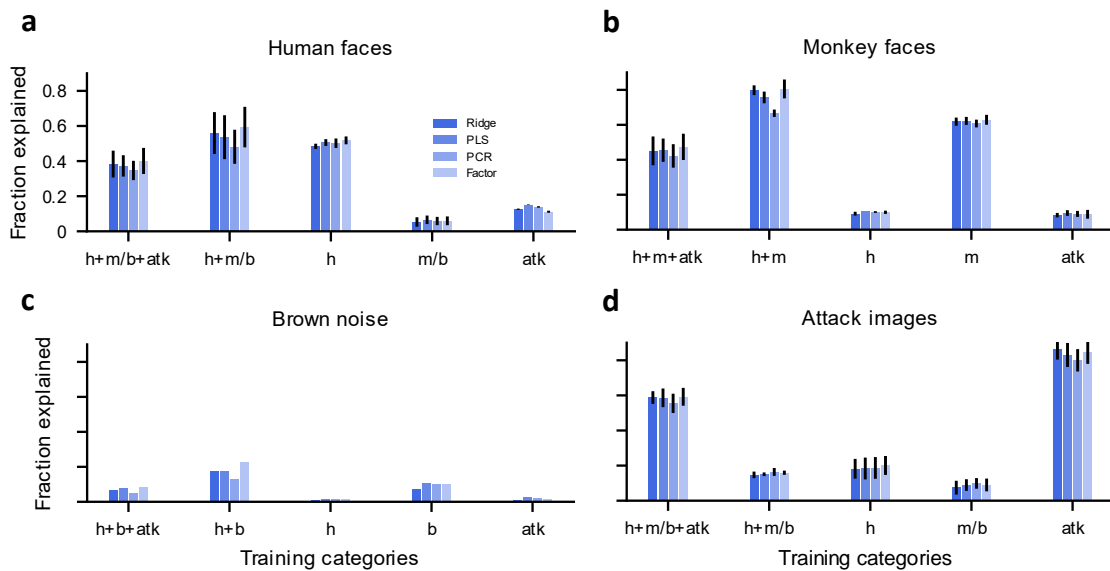


Figure 8.5: Related to **Figure 8.3e–f**. Performance of the Visual Transformer-based predictive model of neuronal responses using different linear regression methods. We tested ridge regression ( $\alpha = 2 \times 10^5$ ), partial least squares (PLS, number of components = 27), principal component regression (PCR, number of components = 100), and factor analysis followed by linear regression (Factor, number of components = 250). The bar plots illustrate model performance in different training and testing configurations. Each panel corresponds to testing on one image category. Each condition on the x-axis corresponds to one combination of training categories.

---

generalize to predicting responses on another image category (such as monkey faces or deceptive images). Overall, across the methods we tested, changing primate vision typically required much higher noise thresholds than changing ANN categorization. The pattern of deceptive image success across methods and individual images was similar between monkeys and humans, but less similar between either and CNN models. Deceptive images open up a new paradigm for critically testing computational models of vision. Our results reveal a shortfall between CNN models and primate vision that is larger than previous studies have suggested, highlighting a need for better models.

## Chapter 9

# Human or Machine? Turing Tests for Vision and Language

As AI algorithms increasingly participate in daily activities that used to be the sole province of humans, we are inevitably called upon to consider how much machines are really like us. To address this question, we turn to the Turing test and systematically benchmark current AIs in their abilities to imitate humans. We establish a methodology to evaluate humans versus machines in Turing-like tests and systematically evaluate a representative set of selected domains, parameters, and variables. The experiments involved testing 769 human agents, 24 state-of-the-art AI agents, 896 human judges, and 8 AI judges, in 21,570 Turing tests across 6 tasks encompassing vision and language modalities. Surprisingly, the results reveal that current AIs are not far from being able to impersonate human judges across different ages, genders, and educational levels in complex visual and language challenges. In contrast, simple AI judges outperform human judges in distinguishing human answers versus machine answers. The curated large-scale Turing test datasets introduced here and their evaluation metrics provide valuable insights to assess whether an agent is human or not. The proposed formulation to benchmark human imitation ability in current AIs paves a way for the research community to expand Turing tests to other research areas and conditions. All of source code and data are publicly available: <https://tinyurl.com/8x8nha7p>

**Judging the Turing test: which one is human and which one is a machine?** DALL-E prompt: A painting of a courtroom in which a judge with a black coat and a hammer looks at two closed big doors.



### 9.1 Introduction

The Turing test, also known as the “imitation game”, was proposed by Alan Turing in 1950 as a way of assessing a machine’s ability to exhibit intelligent behaviors indistinguishable from those of a human (**Fig. 9.1**) [Turing, 2009]. Since its inception, whether the Turing test adequately quantifies intelligence or not has remained controversial [French, 2000, Hayes and Ford, 1995]. The purpose of this paper is *not* to argue in favor or against Turing tests as a measure of general intelligence. Instead, we consider the Turing tests as a quantitative evaluation of how well current AIs can imitate humans.

With powerful AI technologies being deployed in the real world, it is becoming increasingly important for lay people, legal judges, doctors, politicians, and other experts to ascertain whether the agent they are interacting with is a human or not. As two examples out of many, the inability to

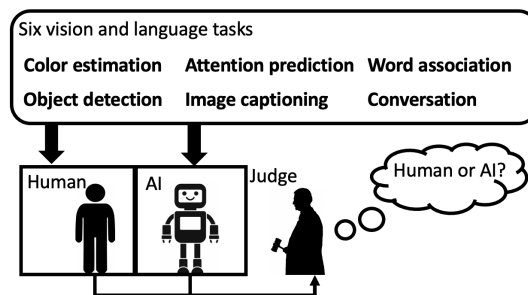


Figure 9.1: **Schematic illustration of Turing tests in six vision and language tasks.** A Turing test works with a judge asking a test subject (either a human or an AI agent) a series of tasks. Each party is kept in a separate room, so no physical contact is allowed. The AI passes the Turing test if the judge is unable to distinguish the AI from another human being by using the responses collected from the given task presented to both. See **Fig 9.2** for an overview of the six tasks.

distinguish a human from an AI bot may lead to cybersecurity breaches resulting in the loss of private and protected data. Besides, the inability to distinguish real news from AI generated fake news or DeepFakes [Westerlund, 2019] can have disastrous implications for electoral campaigns [Westerlund, 2019, Hall, 2018].

The answer to whether current AIs pass the Turing test depends on a plethora of considerations, including the machine agent, the human agent, the judge, the specific task, contextual conditions, and many more. Distinct from the original version of the Turing test in unrestricted conversations, the purpose of the current work is *not* to exhaustively study all possible combinations of these parameters and choices. Instead, we aim to: (i) establish a methodology to evaluate human imitators, (ii) provide a systematic protocol for the AI community to quantify whether a task is performed by humans or machines, and (iii) introduce evaluation metrics and analysis tools on a subset of tasks and conditions as a proof-of-principle. Specifically, we benchmarked 24 AI models in Turing tests on 6 fundamental tasks in computer vision and natural language processing (**Fig. 9.2**): color estimation, object detection, attention prediction, image captioning, word associations, and conversation.

The key contributions of this work are:

- (1) We design a systematic format for conducting Turing tests and evaluating AI models over different tasks involving multiple modalities. This helps the community expand the Turing test to a wide range of tasks and benchmark future AI models.
- (2) We introduce datasets to evaluate current AIs in Turing-like tests in 6 fundamental vision and language tasks.
- (3) We conduct human psychophysics experiments to evaluate human judges in 24 state-of-the-art vision and language AI models in Turing tests.
- (4) We show that simple machine learning algorithms can serve as AI judges to distinguish machines versus human agents in the same tasks.

## 9.2 Related Works

### 9.2.1 Glimpse of the 70-year history of Turing test

The Turing test was introduced as an imitation game where a machine tries to pass as human during a conversation and a human judge determines whether they are interacting with a human or not [Turing, 2009]. The Loebner Prize was introduced in 1991 [Mauldin, 1994] to the programs considered by human judges to be the most human-like. There was also an award for the most human human [Christian, 2011]. The Turing test has generated extensive controversy and discussion about whether it is a valid measure of intelligence [LaCurts, 2011, Hayes and Ford, 1995, Purtill, 1971,

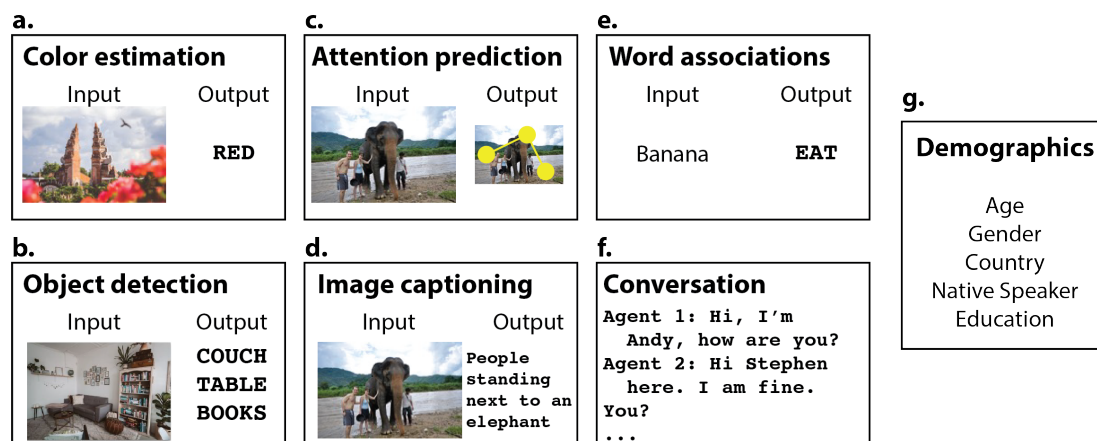


Figure 9.2: **Schematic of the 6 tasks.** We systematically evaluate 3 vision tasks, 1 vision-language task, and 2 language tasks. **a.** In Color estimation, the agent is presented with an image and has to output the main color. **b.** In Object detection, the agent is presented with an image and has to provide three objects. **c.** In Attention prediction, the agent is presented with an image and the output is a sequence of attention locations or eye movements. **d.** In Image captioning, the agent provides a single sentence description of an image. **e.** In Word associations, the agent is presented with a word and has to produce a single word related to the cue. **f.** In Conversations, agents produce 24 exchanges. See **Sec. 9.3** for detailed description of each task and see **Supp. Material** for more example stimuli from both human and AI agents for all tasks. **g.** The results of a Turing test with a human judge depend on the characteristics of the judge. As an initial characterization, we collect basic demographic information indicated in this table.

[Gunderson, 1964, Gunderson, 1985], shifting to whether machines can successfully imitate humans [Harnad, 1991, Harnad, 1994, Harnad, 1999]. Several notable arguments include Searle’s Chinese room thought experiment [Searle, 1980], Block’s behaviorism [Block, 1981], Harnad’s Total Turing Test [Harnad, 1989], Watt’s Inverted Turing Test [Watt, 1996], Damassino’s Questioning Turing Test [Damassino, 2020] and Sejnowski’s Reverse Turing Test [Sejnowski, 2022]. Distinct from these arguments, our aim is to systematically and quantitatively provide methods, datasets and benchmark current AIs in imitating humans through Turing-like tests in multiple vision and language tasks.

### 9.2.2 AI versus humans in vision tasks

Current computer vision models can perform a wide range of tasks such as object recognition and detection. Models are often evaluated by comparing their outputs against human ground truth annotations. Many object recognition studies benchmarked AI versus humans in out-of-distribution generalization [Bhojanapalli et al., 2021, Dodge and Karam, 2017], adversarial attacks [Elsayed et al., 2018], and contextual variations [Zhang et al., 2020, Bomatter et al., 2021b]. Several studies also compared attention in AI models against humans in saliency prediction [Jiang et al., 2015], and eye movement prediction [Zhang et al., 2018, Gupta et al., 2021, Yang et al., 2020]. However, high performance in a particular task does *not* constitute a Turing test. AI models can show similar average performance to humans in narrow tasks, or even outperform humans, and still be distinguishable from humans. Turing tests provide a unique assessment of AI models as imitators of human behavior which extends and complements current benchmarking frameworks.

### 9.2.3 AI versus humans in language tasks

Similar observations can be made in natural language processing. AI models are often compared against human ground truth data in discriminative tasks, such as image captioning or visual question answering [Chandrasekaran et al., 2018, Sheng et al., 2021, Mathew et al., 2021,

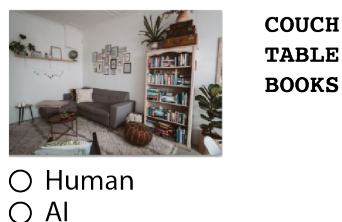
**Turing test (object detection)**

Figure 9.3: **Schematic illustration of the Turing test for the object detection task.** The judge is presented with an image and three labels and has to decide whether those labels were produced by a human or by an AI. For screenshots of the Turing test for each of the tasks, see **Supplementary Material**.

Yan et al., 2021]. Human evaluation scores are reliable but costly to obtain. To mitigate these problems, several evaluation metrics have been proposed, such as BLEU [Papineni et al., 2002], THUMB [Kasai et al., 2021], and METEOR [Denkowski and Lavie, 2014] in image captioning. However, these metrics focus on n-gram overlaps and are insensitive to semantic information. Cui *et al.* proposed a learned critique model acting as a human judge to perform a Turing Test in image captioning tasks [Cui et al., 2018]. Here we also introduce critique models and compare them with human judges.

Generative AI models are notoriously difficult to evaluate due to the inherent ambiguities of language. For example, human evaluators are often recruited to assess the quality of sentiment and semantic relevance on text generated by BERT [Devlin et al., 2018] or GPT2/3 [Brown et al., 2020a, Brown et al., 2020a, Karpinska et al., 2021]. Such evaluations are restricted to specific domains of text generation and the heterogeneity of human judges has not been characterized. Here we provide an extensive set of Turing tests on multiple large state-of-the-art language models based on 896 judges across different demographics.

Conversation was the key target of the original Turing test and remains a daunting challenge for AI. There have been numerous early attempts at generating restricted topics during conversations, such as Colby’s PARRY simulating a paranoid schizophrenic [Colby, 1981, Colby et al., 1971] and Weizenbaum’s ELIZA simulating a psychiatrist [Weizenbaum, 1966]. However, none of these models have come close to unrestricted Turing tests. Advances in large language models [Devlin et al., 2018, Brown et al., 2020a, Collins and Ghahramani, 2021, Shuster et al., 2022b] have led news and social media to produce anecdotal claims about current AI being sentient in conversations [Wertheimer, , Tiku, , Maruf, ]. However, few studies rigorously and quantitatively assessed AIs in their ability to imitate humans in conversation. Preliminary works introduced unrestricted Turing tests in conversations with one exchange per conversation [Zhang et al., 2019]. Here we provide extensive evaluations of AIs engaged in conversations with up to 24 exchanges.

### 9.3 Experiments

We introduce the six tasks (Fig. 9.2), how we created the datasets and how we set up the Turing tests (Fig. 9.3). Further details about each task, controls, and an example snapshot of the Amazon Mechanical Turk (AMT) interfaces are provided in the **Supp. Sections 9.6**. All AMT experiments are based on “master” workers. We also collected demographic information about the participants as metadata, including their native language, age, gender, educational background, and the country they are originally from (Fig. 9.2g). For each task, we collect human answers and machine answers. During each Turing test, we present a single instance of the answers and ask participants to indicate whether the answer comes from a human or AI (e.g., Fig. 9.3 for the Object detection task). Half of the time, the entry shown was from a human. The other half of the time, an AI answer was shown, sampling with equal probability from one of the different computational models used for each task. The trial order was randomized. No feedback was provided to the participants. Additional control trials were introduced for each specific task to

Task	Num. Stimulus	Num. Turing Tests	Sources of Datasets	AI models
Color estimation	785	1,625	self-collect, MSCOCO [Lin et al., 2014]	Google Vision API Microsoft Azure Cognitive Services, MMCQ [Bloomberg and Leptonica, 2008]
Object detection	808	1,975	self-collect, MSCOCO [Lin et al., 2014]	Google Vision API, Microsoft Azure Cognitive Services Amazon Rekognition, Detectron2 [Wu et al., 2019]
Attention Prediction	547	2,160	NatureDesign [Zhang et al., 2018], FindingWaldo [Zhang et al., 2018] NatureSaliency [Zhang et al., 2022a]	IVSN [Zhang et al., 2018, Gupta et al., 2021, Zhang et al., 2022a], DeepGaze3 [Kümmerer et al., 2022], GBVS [Harel et al., 2006]
Image Captioning	1,000	8,140	self-collect MSCOCO [Lin et al., 2014], nocaps [Agrawal et al., 2019]	GIF [Wang et al., 2022a], OFA [Wang et al., 2022b], BLIP [Li et al., 2022] ClipCap [Mokoly et al., 2021], Microsoft’s Azure Cognitive Services [mic, ]
Word Association	1,500	3,550	self-collect	Word2Vec [Pennington et al., 2014], GPT2 [Radford et al., 2019], GPT3-embedding (davinci) [Brown et al., 2020a], GPT3-prompt (text-davinci-001) [Brown et al., 2020a], GPT3-prompt (text-davinci-002) [Brown et al., 2020a].
Conversation	300×8 lengths	4,120	self-collect Typical-Chat dataset [Gopalakrishnan et al., 2019]	GPT3-text-davinci-002 [Ouyang et al., 2022], GPT3-text-curie-001 [Ouyang et al., 2022] Blenderbot [Shuster et al., 2022a], DialogPT [Gopalakrishnan et al., 2019]

Table 9.1: **Specifications of six Turing tasks** Source datasets, number of Turing tests conducted, number of stimulus, and AI models used to collect responses are listed for each task. See **Section 9.3** for task descriptions.

ensure compliance.

### 9.3.1 Color Estimation

**Dataset.** We collected a testing set of 1,000 images sampled from the validation set of MSCOCO [Lin et al., 2014] to ensure that computational models had not seen these images during training. We collected human responses for 785 images using AMT in the Color Estimation task. AMT subjects were asked “What is the dominant color in this image?”, and had to select from 11 possible colors.

For AI models, we collected responses for the dominant color from Google’s Cloud Vision API [vis, ], Microsoft’s Azure Cognitive Services [mic, ], and the publicly available Median Cut Color Quantized (MCCQ) algorithm [Bloomberg and Leptonica, 2008]. In total, 3,140 image-response pairs were collected (785 from humans and 2,355 from AI models, **Table 9.1**).

**Turing test (Human Judge).** Subjects were shown an image-response pair collected above, and asked the question “Was the color description made by a human or a machine?” Subjects were also asked to predict the gender of the speaker. An auxiliary True (50%)/False (50%) question about the presence of a background object present in the image was introduced as a control. The mean accuracy in this control task was 91%. A total of 1,625 responses were collected from human judges.

**Turing test (AI judge).** For each image-response pair, we collected image features from the last layer embedding of a ResNet18 model pretrained on ImageNet, and the textual features for the color using BERT [Devlin et al., 2018]. These features were then concatenated, and passed into a Random Forest classifier to predict human versus AI labels, using a 90% / 10% split for cross-validation.

### 9.3.2 Object detection

**Dataset.** We collected responses for 808 images from the set of 1,000 images from MS-COCO in **Section. 9.3.1**. Subjects were presented with the question “What do you see in this image?”, and were asked to enter three single word responses . We introduced several controls, ensuring that the responses were single words in English and that participants do not repeat the same response within or across images.

We collected predicted labels from Google’s Cloud Vision API, Microsoft’s Azure Cognitive Services, Amazon’s Rekognition API, and Facebook’s Detectron2. For all networks, the top three predicted labels with the highest prediction score were collected. In total, 4,040 image-response pairs were collected (808 from humans and 3,232 from the four AI models, **Table 9.1**).

**Turing test (Human judge).** The Turing test for human judges was performed using the same protocol and controls described in **Section 9.3.1**. A total of 1,975 responses were collected from human judges.

**Turing test (AI judge).** As the images were the same for humans and AI models, we used solely textual features for the AI judge to classify a response as human or AI. We concatenated the three responses into a sentence, and collected textual features for the sentence using BERT [Devlin et al., 2018]. These features were passed into an SVM classifier to classify responses into humans versus AI, with 90% / 10% split cross-validation.

### 9.3.3 Attention prediction

**Dataset.** We used eye movements (overt attention) from human subjects during two visual search tasks [Zhang et al., 2018], and a free-viewing task [Zhang et al., 2022a]. We evaluated 7,000 scanpaths from 40 participants. For the three datasets, we used a modified version of IVSN [Zhang et al., 2018, Zhang et al., 2022a], DeepGaze3 [Kümmerer et al., 2022] and GBVS models [Harel et al., 2006] to generate eye movement predictions.

**Turing test (human judge).** Separate Turing tests were launched for eye movements from free-viewing tasks (80 judges) and visual search tasks (100 judges). We presented infinitely repeating animated clips of eye movements from humans or model predictions with a maximum of 15 fixations to human judges on AMT. Each judge had to identify if the eye movements were from a human or a computational model. As a control, judges were also asked to answer “What do you see in the presented clip?” with one correct answer among 3 options. Responses from judges with a score  $< 7$  out of 12 were not considered in the analyses.

**Turing test (AI judge).** We performed Turing tests using an SVM as an AI judge. Sequences of 10 fixations per trial from humans or computational models were fed as input in the form of an array of fixation coordinates to train an SVM to classify human versus machine eye movements. The SVM was trained using 10-fold cross validation. Model performance on validation sets across folds with 3 random seeds was calculated and averaged.

### 9.3.4 Image captioning

**Dataset.** We randomly sampled 250 images each from in-domain, near-domain, and out-of-domain categories from the validation set of the nocaps dataset [Agrawal et al., 2019] and 250 images from the MSCOCO test set [Lin et al., 2014], creating a set of 1,000 images. We collected 2,290 human captions with  $\geq 6$  words per caption and  $\geq 2$  captions per image from AMT participants. We implemented additional controls in our AMT interface. For example, workers were not allowed to submit a caption before viewing the image for  $\geq 4$ s.

To generate machine captions, we used: GIT [Wang et al., 2022a], OFA [Wang et al., 2022b], BLIP [Li et al., 2022], ClipCap [Mokady et al., 2021], and Microsoft’s Azure Cognitive Services [mic, ]. For open-source models, we used the largest variants finetuned on the COCO Captions dataset [Lin et al., 2014, Chen et al., 2015]. We collected 5,000 machine captions with 5 captions per image.

**Turing test (human judge).** We collected responses from 293 AMT participants. Each participant was presented with image-caption pairs and indicated whether the caption was generated by a human or AI. To ensure that the participants read the captions carefully, we prevented response times  $< 3$ s. We removed responses from non-native English speakers.

**Turing test (AI judge).** We trained an SVM model for binary classification (human versus machine) on the dataset of human and machine captions. We randomly sampled 400 captions from each of the 5 models to get 2,000 machine captions and combined them with our 2,000 human captions. We used the OpenAI API [Neelakantan et al., 2022] to obtain 4,096-dimensional embeddings (text-similarity-curie-001 model) for each caption as input features to train the SVM with 10-fold cross-validation and 3 random seeds.

### 9.3.5 Word associations

**Dataset.** We chose 150 unique cue words (50 nouns, 50 verbs, and 50 adjectives), spanning a wide range of occurrence frequencies [Speer et al., 2018]. Associations to each cue word were collected from human subjects and from the following language models: Word2vec [Pennington et al., 2014], GPT2 [Radford et al., 2019], GPT3-embedding (based on davinci embedding), GPT3-curie-prompt (based on "curie" prompt completion), and GPT3-davinci-prompt (based on "davinci" prompt completion) [Brown et al., 2020a]. For the human associations, we followed two procedures: (1) Free associations, whereby participants provided a one-word answer to the question: “What is the first word that comes to your mind when you hear the word [cue word]?” and (2) Prompt-based associations, whereby participants completed a prompt with one word. The prompts used for



the human prompt-completion were the same prompts used for GPT3-curie-prompt and GPT3-davinci-prompt. All participants were English native speakers living in the US.

**Turing test (human judge).** For the human-judge Turing tests, we collected data from 50 native English speakers on AMT. In each trial, a cue word and a corresponding guess word (association) were presented and the judge had to choose whether the association was made by a human or by an AI model.

**Turing test (AI judge).** We trained a linear SVM classifier with 10-fold cross-validation [Cortes and Vapnik, 1995] to distinguish between human-made and machine-made associations. We used the the distance between the cue and guess word embeddings, based on (1) Word2Vec, (2) GPT2, or (3) GPT3 (davinci).

### 9.3.6 Conversation

**Dataset.** We collected 300 conversations between: (1) two humans, (2) a human and an AI model, (3) two AI models. For the conversations including humans, we recruited 150 fluent English participants to have a conversation over a chatting platform. The participants did not know whether they were speaking with another human or with an AI chatbot (see instructions in **Supp. Section 9.6.4**). We collected conversations containing 24 exchanges each. For the human-human conversations, we added 40 conversations from the Topical-Chat dataset [Gopalakrishnan et al., 2019], selected based on a minimum length of 24 exchanges. Multiple example conversations are included in **Supp. Section 9.6.7**.

For the AI chatbots, we used three state-of-the-art language models: Blenderbot3 (175B model) [Shuster et al., 2022a], GPT3 text-davinci-002 [Ouyang et al., 2022], and GPT3 text-curie-001 [Ouyang et al., 2022] (see settings, pre-processing, prompts, and control details in **Supp. Section 9.6.5**).

**Turing test (human judge).** We chunked each conversation into 8 different lengths, including the initial 3, 6, 9, 12, 15, 18, 21, and 24 exchanges. There were 208 human judges (AMT: 200, in-lab: 8). The participants were presented with 20 randomly sampled chunked conversations with different lengths and had to respond, for each of the two speakers, whether the speaker was a human or a machine and the gender (**Fig. 9.19**). As a control, speakers also had to select the general topic of the conversation from a list of five topics. We only considered judges that correctly classified at least 15 topics out of 20 and removed incorrectly classified trials.

**Turing test (AI judge).** We evaluated whether simple AI models can discern whether a sentence was generated by a model or a human. We only examined single sentences here. Therefore, these results provide only an initial proof-of-principle lower bound for AI judges. We built four corpora, one containing all the sentences written by humans (the *human corpus*), and the others with the sentences produced by Blenderbot, GPT3text-davinci-002 and GPT3text-curie-001 (the *AI corpora*). We used *BERT* embeddings [Devlin et al., 2018] to tokenize each sentence, and fed the tokenized sentences to a linear SVM trained to classify *human* vs. *AI* with 10-fold cross-validation.

## 9.4 Results

We summarize the results of all the Turing tests in **Fig. 9.4**, by averaging across all AI models and all human judge demographics. In the **Supplementary Material**, we show results separated by AI model and also for different human judge demographics. For each task, **Fig. 9.4** shows the proportion of times that a trial was classified as human (first column), or AI (second column), when the ground truth was human (first row), or AI (second row). Entries along each row add up to 100%. When comparing different AI models for a given task in terms of the ability to imitate humans, percentages closer to 50% indicate better models. In contrast, when comparing different judges (e.g., human judges versus AI judges, or human judges of different ages or educational backgrounds), higher overall accuracy indicates better judges.

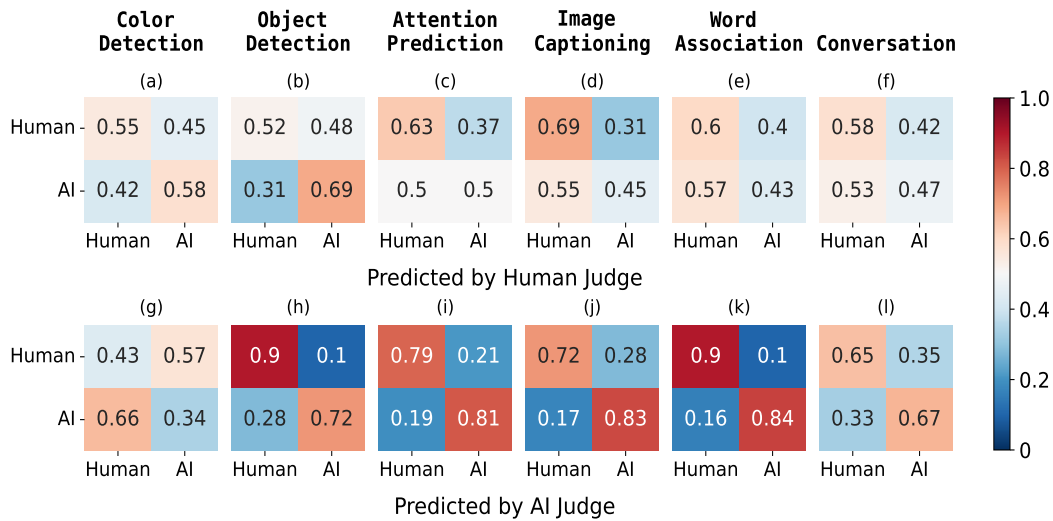


Figure 9.4: **Results of the Turing test for each task.** Turing test results for human judges (top row) and machine judges (bottom row). For each task, the confusion matrices report the percentage of times when the trial was labeled “human” (first column) or “AI” (second column) when the ground truth was human (first row) and AI (second row). Percentages add up to 100 within each row. Here all AI models were averaged together. See **Supplementary Material** for results from each AI model and different human judge demographic groups.

### 9.4.1 Color estimation

Human judges distinguished AI answers as AI 58% of the time and human answers as human 55% of the time (**Fig. 9.4a**). We broke down performance based on each individual AI model. The Google API performed slightly better (57%) than Azure API (60%) and MCCQ (65%). Even though the color MCCQ is a simple metric, it still achieved a moderately good performance in fooling humans 35% of the time. There was no major difference in performance of human judges across different age groups, education levels, or genders. In contrast, the AI judge classified human answers as human 43% of the time and AI answers as AI 34% of the time (**Fig. 9.4g**).

### 9.4.2 Object detection

Human judges distinguished AI answers as AI 69% of the time and human answers as human 52% of the time. We broke down performance based on each individual AI model. Among all the AI models, Detectron performed the best (49%), with a large gap from the second best, Google API (65%). This modern object detection algorithm in computer vision not only achieves outstanding absolute scores in terms of standard evaluation metrics, such as mAP [Ren et al., 2015], its response patterns also closely mimic humans’ by identifying top-3 salient objects in the scene. Specifically, we used the variant with MaskRCNN [He et al., 2017] trained on ImageNet and MS-COCO.

Next, we analyzed the classification performance of the AI judge for this task. In start contrast to human, the AI judge is able to distinguish between AI and human speakers much better **Fig 9.4**. The overall classification accuracy of AI judge is 81% (as compared to 56.5% of human judges). Specifically, AI judge can tell AI responses as AI with a 90% accuracy, and human responses as human with 72%. The easiest to classify are responses from the Azure API with AI judges getting a 94% accuracy, while the hardest to classify are Detectron and Amazon’s Rekognition API with accuracy of 67% each.

### 9.4.3 Attention prediction

Human judges distinguished human eye movements as human 63% of the time and AI-generated eye movements as AI 50% of the time (**Fig. 9.4c**). We examined the Turing test performance for each AI model separately. IVSN [Zhang et al., 2018, Zhang et al., 2022a] outperformed GBVS [Harel et al., 2006] and DeepGaze3 [Kümmerer et al., 2022] by 4% and 17%, respectively. We evaluated whether the agent’s goals during eye movements might influence the subsequent Turing tests. Human judges performed 2.5% better in free-viewing compared to visual search tasks. Free-viewing is mostly driven by pure bottom-up saliency, and it may be easier to discern scanpath patterns without target-directed modulation. Consistent with this idea, GBVS generated more human-like scanpaths during free-viewing compared with visual search, whereas DeepGaze3 performed much worse in free-viewing tasks than visual search. IVSN performance was similar in both tasks, which emphasizes the importance of incorporating both bottom-up and top-down attention mechanism in computational models of human attention.

As an initial evaluation of AI judges, we trained an SVM classifier purely based on the sequences of eye fixations regardless of the image features (**Fig. 9.4h**). Interestingly, a simple SVM AI judge performed 20% better than human judges. AI judges outperformed human judges across different models, and different tasks. However, this result should be interpreted with caution since the AI judge was explicitly trained to classify scanpaths while human judges typically do not have such prior training.

### 9.4.4 Image captioning

Human judges distinguished human captions as human 69% of the time and AI captions as AI 45% of the time (**Fig. 9.4d**). There were rather large differences among AI models with proportions of AI captions labeled AI ranging from 37% (BLIP) to 59% (ClipCap). CIDEr is a standard evaluation metric for assessing the quality of AI-generated captions [Vedantam et al., 2015]. OFA shows better performance than GIT and ClipCap in CIDEr score [Wang et al., 2022b], but it is not better at imitating human captions, highlighting the differences between traditional performance metrics and Turing tests. Human judges labeled captions as humans slightly more often for images from in-domain nocaps, compared to near-domain nocaps and out-of-domain nocaps.

As a proof-of-principle to build an AI judge, we used the OpenAI Embeddings API (text-similarity-curie-001 model) to extract features and trained a linear SVM to discriminate human versus machine responses. Surprisingly, even though this classifier did not use image features, the AI judge could discern human versus machine answers with 77.5% accuracy, well above the performance of human judges (**Fig. 9.4j**). Similar results were observed across all the different AI models.

### 9.4.5 Word associations

Human judges labeled human word associations as humans 60% of the time and AI word associations as AI 43% of the time (**Fig. 9.4e**). Results were similar for different AI models. Surprisingly, the Word2Vec model produced word associations that were harder to discriminate from human ones.

We used the various word embedding from GPT-series models and Word2Vec as features to train 3 SVM judges (independent SVM classifiers based on the embedding of each AI model). In contrast to human judges, AI judges could easily distinguish human versus machine word associations (**Fig. 9.4k**). All AI judges outperformed human judges. As expected, when training an SVM classifier based on the embedding of the same model whose embedding were also used for generating the guess words, this model classification performance was essentially perfect. For instance, on the 3rd row on the left confusion matrix in, the AI judge trained on Word2Vec embedding indeed perfectly predicted all guess words generated by Word2Vec embeddings as AI. While these cases are good as a sanity check, they should not be considered in the general evaluation of the AI judges performance. Hence, we trained 3 different SVM to avoid such biases.

### 9.4.6 Conversation

Human judges distinguished human participants in conversations as humans 58% of the time and AI agents as AI 47% of the time (**Fig. 9.4f, 9.16, 9.17**). Unlike AI models passing the Turing tests in restricted conversation topics, it is interesting to note an overall accuracy of 53.5% here in largely unrestricted conversations.

We separately considered human-human, human-AI, and AI-AI conversations (**Fig. 9.5**). Surprisingly, human-human conversations were classified as human only 61% of the time and AI-AI conversations were classified as human 56% of the time. Blenderbot was classified as human 64% of the time, suggesting that AIs can be perceived as more human than humans themselves. In human-AI conversations, human participants were labeled as humans 61% of the time, and AIs were classified as AI 55% of the time. The overall classification accuracy in human-AI conversations was higher than AI-AI conversations (58% versus 41%), suggesting that AIs reveal their true self more often when talking to humans than amongst themselves. This observation is consistent with the notion that human judges are more accurate in making comparisons rather than absolute evaluations.

When comparing different AI models (**Fig. 9.5**), Blenderbot was most often labeled as humans, 66% of the time in AI-AI conversations and 50% of the time in human-AI conversations (**Table 9.3**). The results of the Turing test depended on the conversation length (**Fig. 9.18**). AI models were less adept at passing as humans in longer conversations. Among all the AI models, GPT3-curie showed the sharpest drop while Blenderbot maintained relatively high performance. These observations highlight that model sizes, specific training on conversation data, and incorporation of external memory modeling past conversation history are important factors when imitating humans in conversations.

Younger judges performed better in discerning AIs from humans than older judges in AI-AI conversations (**Fig. 9.7, 9.8, 9.9, 9.15a**). Surprisingly, male judges performed slightly better than female judges (60% versus 57.5%), especially in AI-AI conversations (46% versus 39%) (**Fig. 9.10, 9.11, 9.15b, Table 9.4**). Intriguingly, education had a slight negative relation with classification accuracy of human judges (54%, 53% and 51% for middle-high school, college and postgraduate degrees respectively), especially in human-AI conversations. However, this trend was reversed in AI-AI conversations where postgraduate judges performed better than middle-high school judges (53% versus 41%) (**Fig. 9.12, 9.13, 9.14, 9.15c**).

We trained a simple SVM judge to distinguish whether a sentence in a conversation was from humans or AIs. Consistently with the other experiments, the AI judge beat human judges by a large margin (66% versus 53.5%, **Fig. 9.4l**). This AI judge performed particularly well in classifying Blenderbot sentences (**Fig. 9.6**), in stark contrast with human judges who were more easily fooled by Blenderbot than GPT models. Human judges likely focus on high-level conversation understanding rather than single-sentence statistics in the Turing tests.

## 9.5 Discussion

The Turing test has been extensively discussed, and contested, as a means to assess general intelligence. Instead, we focus on Turing tests as a metric to evaluate whether an algorithm can imitate humans or not. **Table 9.2** summarizes the observations in a highly simplified binary format; this table is a grand average and the reader is referred to all the actual numbers for a more accurate description of the findings.

Remarkably, the algorithms tested throughout the current work seem to be quite close to passing a Turing test when evaluated by human judges. Given that imitating humans can be very good for certain purposes but could also easily be turned into potentially evil applications, these observations call for more extensive and rigorous scrutiny of machines that can imitate AI.

One step to mitigate risks from human imitators is to build AI judges. Our results show that even simple AI judges like the ones introduced here can do a better job than human judges in detecting machine answers. The results of current AI judges should not be over-interpreted because AI judges were explicitly trained to classify responses from humans versus AIs, while human judges

	Color Dect.	Object Dect.	Att. Pred.	Image Cap.	Word Ass.	Conver- sation
Age (<35)	✗	✗	✗	✗	✓	✓
Age (35-45)	✗	✗	✗	✗	✓	✓
Age (>45)	✗	✗	✗	✗	✓	✓
Sex (F)	✗	✗	✗	✗	✓	✓
Sex (M)	✗	✗	✗	✗	✓	✓
Edu L1	✗	✗	✗	✗	✓	✓
Edu L2	✗	✗	✗	✗	✓	✓
Edu L3	✗	✗	✓	✗	✓	✓
Over- all	✗	✗	✗	✗	✓	✓

Table 9.2: **Summary of Turing test results for human judges.**

A tick (✓) indicates that the AI models pass the Turing test ( $0.45 \leq \text{top-1 accuracy} \leq 0.55$ ) in the specific task (columns), based on evaluations made by specific demographic groups (rows). This table represents only a coarse grand summary of the results. The actual confusion matrices and quantitative results are presented throughout the **Supplementary Material**.

were not. This point raises the possibility that humans may be trained to better recognize machine answers in the future.

An algorithm’s ability to imitate humans did not always correlate with traditional performance metrics like accuracy, implying that Turing tests provide a complementary assessment of AI models to existing benchmarking frameworks. Comparisons between models in Turing tests also provide insights helpful for developing future AI models that can better align with humans.

The datasets and evaluations introduced here are quite extensive (21570 Turing test trials, 904 human and AI judges, 6 vision and language tasks, several demographic groups), but they barely scratch the surface of what needs to be done. There are essentially infinite possible Turing tests. The results of a Turing test depend on the task, the algorithm, how the question is formulated, the characteristics of the human judge and many other variables

This work provides a comprehensive, yet certainly far from exhaustive, evaluation of state-of-the-art AI models in terms of human emulation. These efforts pave the way for the research community to expand Turing tests to other research areas, to build better imitators, and better detectors of imitators. If more AI models can “blend” in among humans and take over tasks that were originally unique yardsticks of being humans, this makes us ponder what makes us humans and whether we are mentally, ethically, and legally ready for the rapid revolution brought forth by AI technologies.

## 9.6 Methods and Additional Results for the Conversation Task

**Dataset.** We collected a dataset containing 300 conversations between two speakers (speaker A and speaker B). Each speaker could be either a human or an AI agent. Thus, there were three classes of conversations: human-human, human-AI, and AI-AI. For the human-human and the human-AI conversations, we recruited approximately 150 volunteers who are fluent English speakers to have a conversation over a chatting platform. We acted as intermediary in the conversations so that the volunteers did not know whether they were speaking with another human or with an AI chatbot. The instructions given to the participants are reported in the Supplementary Section 9.6.4. We collected conversations containing 24 exchanges each, *i.e.*, 12 turns per speaker. After the conversation was completed, the volunteers were asked whether they thought they spoke with a person or an AI and the perceived gender and age of the other speaker. To evaluate potential

correlations between the conversation features and the volunteers' demographics, we also collected information on the age, gender, college major and field of work for each volunteer. For the human-human conversations, we combined 18 conversations collected as described above and we also added 40 conversations from the Topical-Chat dataset [Gopalakrishnan et al., 2019], selected based on a minimum length of 24 entries.

For the human-AI and the AI-AI conversations, we used three state-of-the-art language models: Blenderbot3 (175B model) [Shuster et al., 2022a], GPT3 text-davinci-002 [Ouyang et al., 2022], and GPT3 text-curie-001 [Ouyang et al., 2022]. For all conversations with Blenderbot we used the live interface provided at <https://blenderbot.ai/>. For the human-GPT3 conversations we used the playground available at <https://beta.openai.com/playground/>. We list the settings in Supplementary Section 9.6.5. For the GPT3-GPT3 conversations, we implemented a custom python framework for the interaction of two models. For the Blenderbot-Blenderbot conversations, we kept all the collected conversations in the dataset. Instead, the GPT3-GPT3 conversations were affected by long-standing issues of NLP, namely repetition of single sentences or multiple consecutive exchanges and early exit. When we detected such issues, we resampled the conversations. Therefore, we built a chatbot out of GPT3 based on prompt engineering and failure criteria. We did not resample conversations in the case of human-GPT3 conversations. We also attempted to use the DialoGPT model cite, however, the quality of the conversation was not satisfactory (see examples in Section 9.6.7 in the Supplementary Material), hence we did not include DialoGPT in the analysis. We did not perform any preprocessing in the conversations (*e.g.*, did not correct any misspellings, grammatical errors, logical errors or other inconsistencies).

**Turing test.** To investigate the results of the Turing test as a function of the length of the conversation, we chunked each conversation into 8 different lengths, including the initial 3, 6, 9, 12, 15, 18, 21, and 24 turns. In total we collected 40 conversations for each category. To perform the Turing test on the collected conversations, we recruited 208 participants from Amazon Mechanical Turk as observers. The participants were presented with 20 randomly sampled chunked conversations with different lengths. For each of the two speakers, the observers had to state whether they thought the speaker was a human or a machine and the gender (male or female). To verify that the observers were carefully reading the conversation, we introduced a control. For each conversation the observer had to select the general topic of the chat from a list of five topics. In our analysis we only considered the responses of the workers that correctly classified at least 15 topics out of 20. For those workers, we only considered the responses to the conversations that were correctly classified. On average we obtained 14 responses per conversation (approximately 2 responses per length). Some examples of conversations are reported in Section 9.6.7 in the Supplementary Material.

**AI judge.** We evaluated whether simple ML models are able to discern whether a sentence was generated by a model or by a person. In this analysis, we only looked at single sentences and not at the conversation level, therefore the models are only allowed to exploit features such as sentence length, vocabulary, grammar, syntax and typos, and cannot take into account issues such as sentence repetition or lack of logic in reasoning. To this goal, we build four corpora, one containing all the sentences written by humans (the *human corpus*), and the others with the sentences produced by Blenderbot, GPT3text-davinci-002 and GPT3text-curie-001 (the *AI corpora*).

We used *BERT* embeddings [Devlin et al., 2018] to tokenize each sentence, and we fed the tokenized sentence to a *SVM* linear classifier trained to perform binary classification to the classes *human* and *AI*. We split the corpora into train and test splits (90%, 10%) and used 10-fold cross-validation for training. In both the train and test splits we used the same number of sentences for human speakers and for AI agents. For the AI, the sentences were split equally among the three models.

## 9.6.1 Results

### 9.6.1.1 Confusion matrix and top1 accuracy

The results for the human judge are reported in Figures 9.5a and 9.5b, which contain respectively the confusion matrix and the top1 accuracy. The plots show both the overall accuracy for humans

	Blenderbot	GPT3davinci	GPT3curie
H-AI conversation	✓	✗	✗
AI-AI conversation	✓	✓	✗

Table 9.3: **Conversations.** Results for the Turing test for different AI models and conversation types.

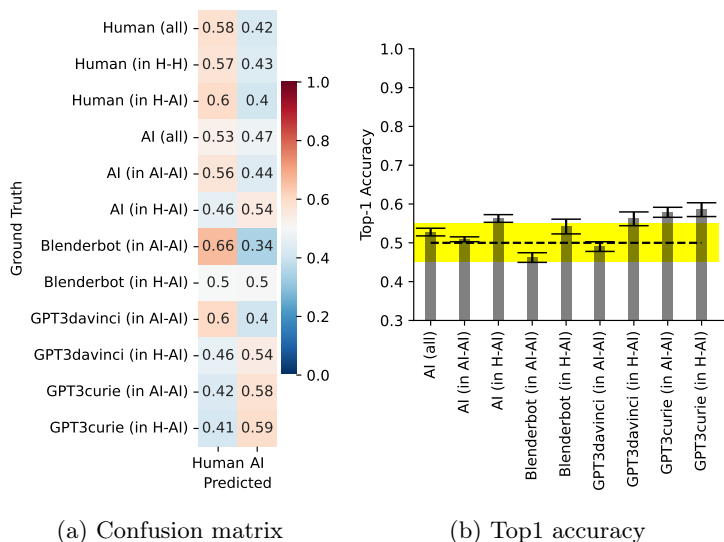


Figure 9.5: **Conversation.** Results of the Turing test for human judges (a) Confusion matrix (b) Top-1 accuracy.

and for AI averaged over all the cases, as well as the results for the three tested models (Blenderbot, GPT3text-davinci002, and GPT3text-curie001). We distinguish the three conditions: humans talking with humans, humans talking with AI, and AI talking with AI.

The results for the SVM judge are reported in Figures 9.6a and 9.6b, which contain respectively the confusion matrix and the top1 accuracy. The plots show both the overall accuracy for humans and for AI averaged over all the cases, as well as the results for the three tested models (Blenderbot, GPT3text-davinci002, and GPT3text-curie001). We do not distinguish the three conditions (humans talking with humans, humans talking with AI, and AI talking with AI), as the SVM judge is trained and tested on the single sentences and not on the full conversations.

### 9.6.1.2 Results on gender perception

The human judges were asked to say whether the speakers of the conversations were male or female, both in case of classification as human and as machine. Overall, 60% of the speakers (human plus machines) were classified as male, indicating a bias of the human judges in the gender perception. In particular, we found that 64% of the human speakers and 58% of the AI agents are classified as male. Furthermore, when a human judge perceived a speaker as human, he associated the male gender 58% of the times. Interestingly, when a speaker was classified as machine, 69% of the times it was also classified as male. This indicates a strong bias in associating the male gender with chatbots. Table 9.4 summarizes these findings.

### 9.6.1.3 Results based on demographics

The results in the form of confusion matrix and top1 accuracy for the human judge are reported in:

- Figures 9.7b, 9.8, and 9.9 for different age ranges;

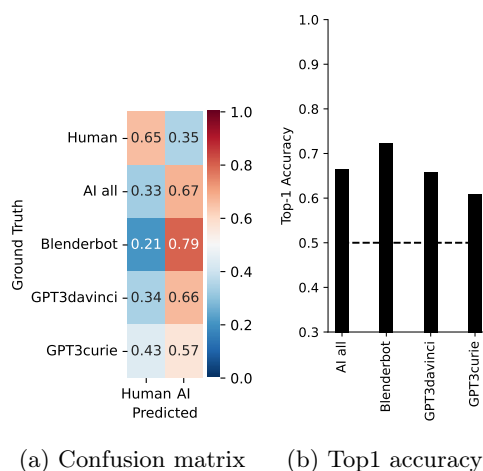


Figure 9.6: **Conversation.** Results of the Turing test for AI judges.  
 (a) Confusion matrix (b) Top-1 accuracy.

Case	[%] classified as male
Overall	60%
Human speaker (true label)	64%
AI speaker (true label)	57%
Human classified (pred. label)	53%
AI classified (pred. label)	68%

Table 9.4: **Conversation.** Gender perception of human and machines by the human judges.

- Figures 9.10 and 9.11 for female and male gender respectively;
- Figures 9.12, 9.13, and 9.14 for different education levels.

The data on the demographic distribution of the human judges are reported in the pie bins in Figure 9.15.

### 9.6.2 Results comparing AMT and in-person experiments

The results for the human judge comparing AMT judges and judges recruited to do the experiment in person under our supervision are reported in Figures 9.16 and 9.17 respectively.

### 9.6.3 Results for the human judge on different length of conversations

We investigated whether and to which extend the result of the Turing test depends on the length of the conversation. Figure 9.18 reports the results. We make the following observations:

- except for very short extracts (three exchanges), humans are classified as humans on average more than machines (panel a);
- humans are more likely to be classified as humans for longer conversations (panel a)
- machines are less likely to be classified as humans for longer conversations (panel a)
- in human-human and AI-AI conversations, humans and machines have similar frequency of being classified as human, except for very long conversations (panel b);
- in human-AI conversations, humans are classified more as human than machines, with a gap increasing with the conversation length (panel c);



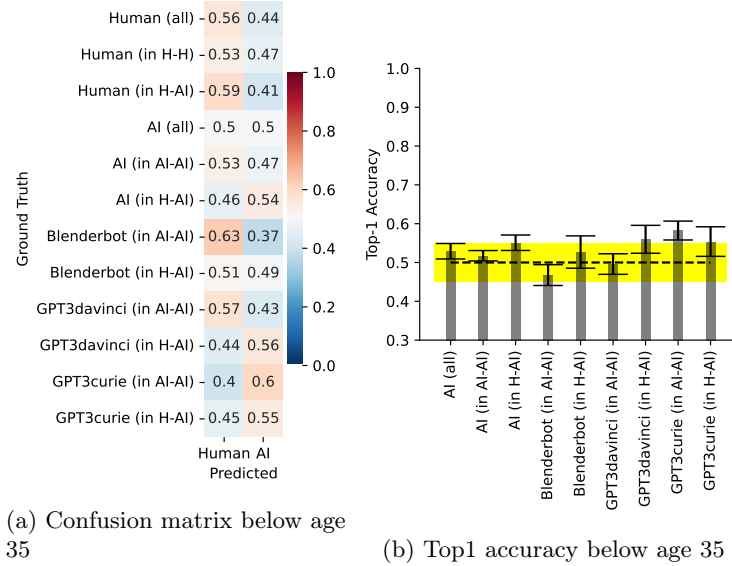


Figure 9.7: **Conversation.** Results of the Turing test for human judges below age 35. (a) Confusion matrix (b) Top-1 accuracy.

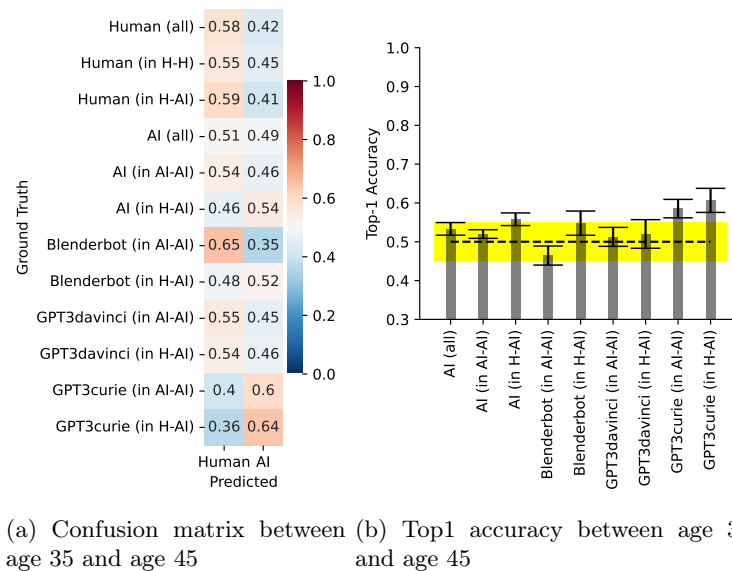


Figure 9.8: **Conversation.** Results of the Turing test for human judges between age 35 and 45. (a) Confusion matrix (b) Top-1 accuracy.

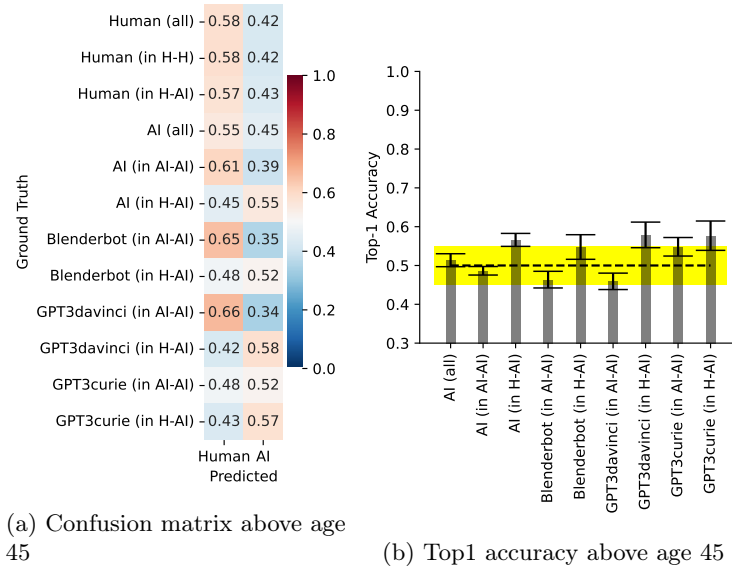


Figure 9.9: **Conversation.** Results of the Turing test for human judges above age 45. (a) Confusion matrix (b) Top-1 accuracy.

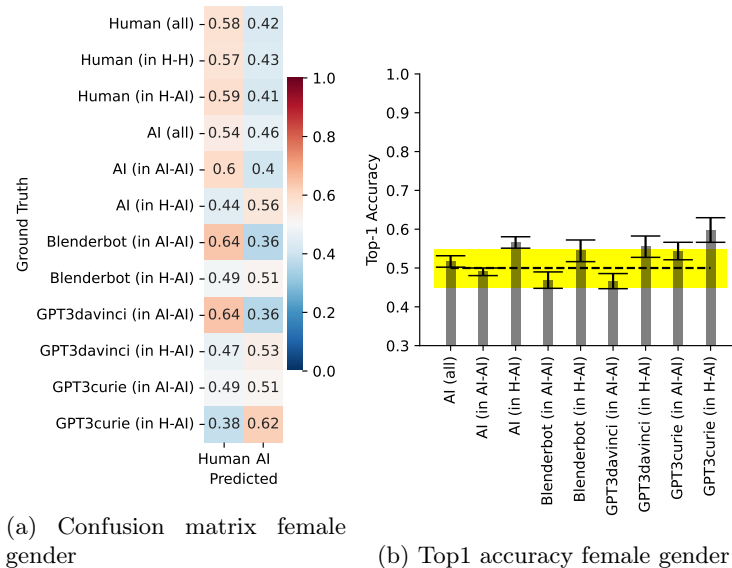


Figure 9.10: **Conversation.** Results of the Turing test for male human judges. (a) Confusion matrix (b) Top-1 accuracy.

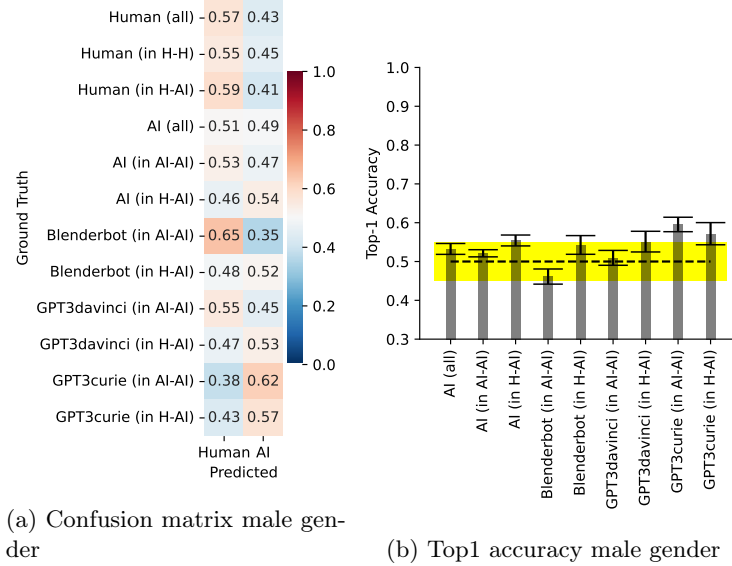


Figure 9.11: **Conversation.** Results of the Turing test for female human judges (a) Confusion matrix (b) Top-1 accuracy.

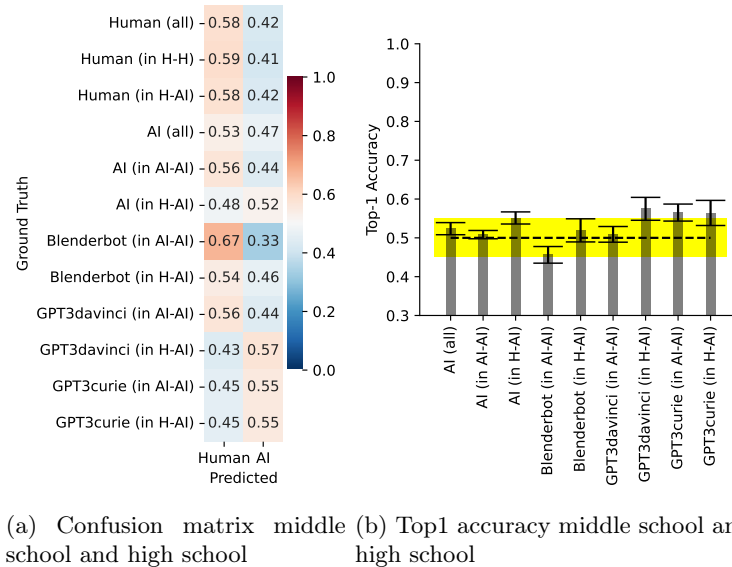


Figure 9.12: **Conversation.** Results of the Turing test for human judges with highest education level of middle/high school.

(a) Confusion matrix (b) Top-1 accuracy.

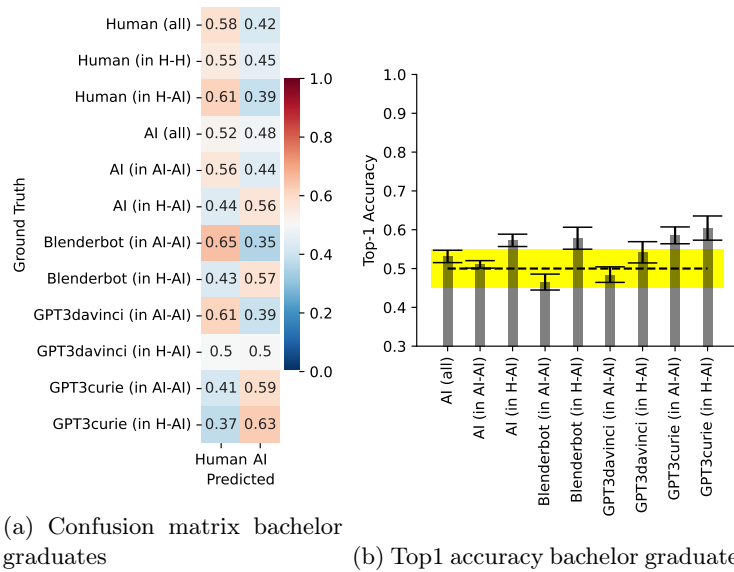


Figure 9.13: **Conversation.** Results of the Turing test for human judges with highest education level of Bachelor.

(a) Confusion matrix (b) Top-1 accuracy.

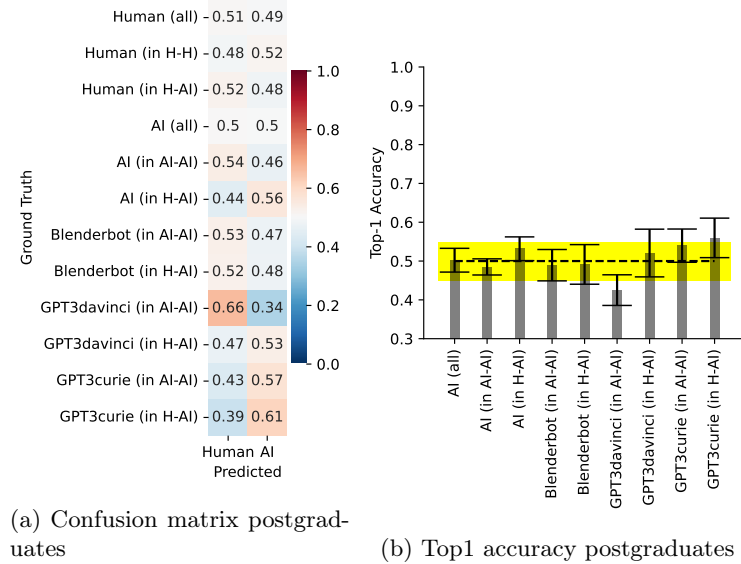


Figure 9.14: **Conversation.** Results of the Turing test for human judges with highest education level of Master and Post-graduate.

(a) Confusion matrix (b) Top-1 accuracy.

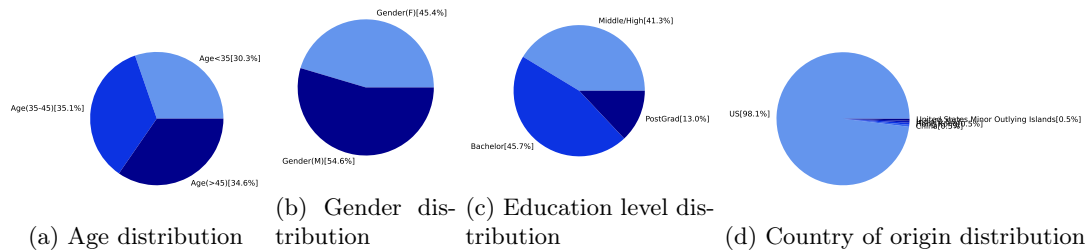


Figure 9.15: **Conversation.** Demographic information for the human judges. (a) Age. (b) Gender.

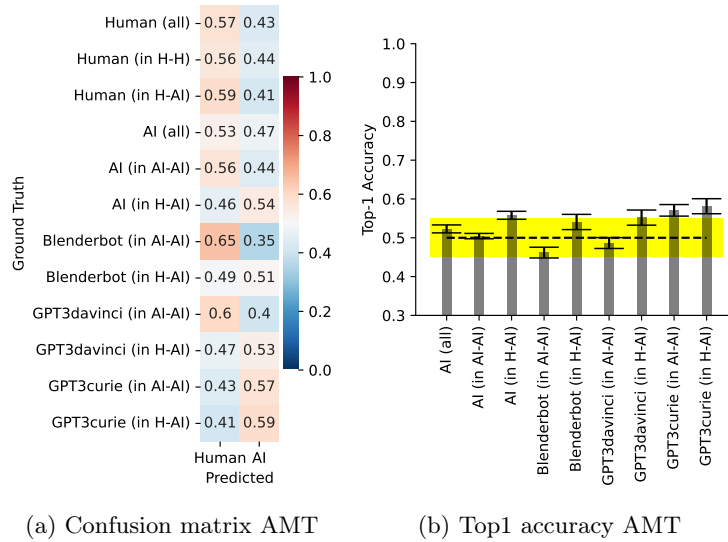


Figure 9.16: **Conversation.** Results of the Turing test for human judges from data collected on AMT

(a) Confusion matrix (b) Top-1 accuracy.

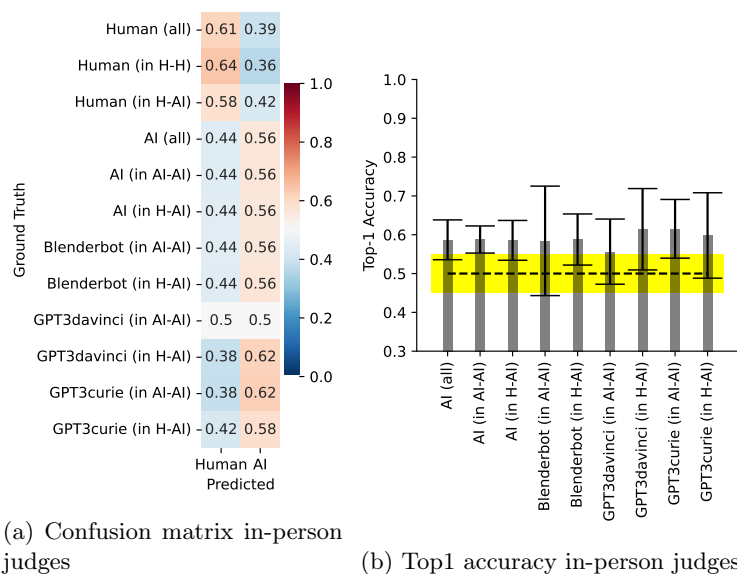


Figure 9.17: **Conversation.** Results of the Turing test for human judges during in-person experiments.

(a) Confusion matrix (b) Top-1 accuracy.

- there is no clear trend distinguishing human classification between human interaction with humans or with various models (panel d);
- for all models, machines talking with humans are classified more as AI for longer conversations; GPT3 curie is on average classified more as machine than GPT3 davinci and Blenderbot (panel e);
- in AI-AI conversations, GPT3 curie is the model that performs the worst; GPT3 davinci and Blenderbot perform similarly up to 15 exchanges, while for longer conversations Blenderbot performs better.

#### 9.6.4 Dataset collection: instructions to human participants

One of the investigators acted as an intermediary to pass messages between two speakers. The two speakers could be two humans, a machine and a human, or two machines. Here we focus on the conversations involving humans. The participants were presented with the following instructions before the conversation:

*Hey! Would you have a few minutes to help me collect a dataset? We just need to have one or two conversations on slack/whatsapp for a few minutes (24 messages in total per conversation). Here are the instructions:*

- *You will have to ask or answer a question to start and trigger the conversation (I will specify case by case).*
- *Please try to get the conversation going for 24 sentences in total (12 from you, 12 from the other speaker).*
- *Please write each reply in a single message (do not write a second message until you receive a reply).*
- *Just chat as if you are texting either with a friend or someone you don't know.*

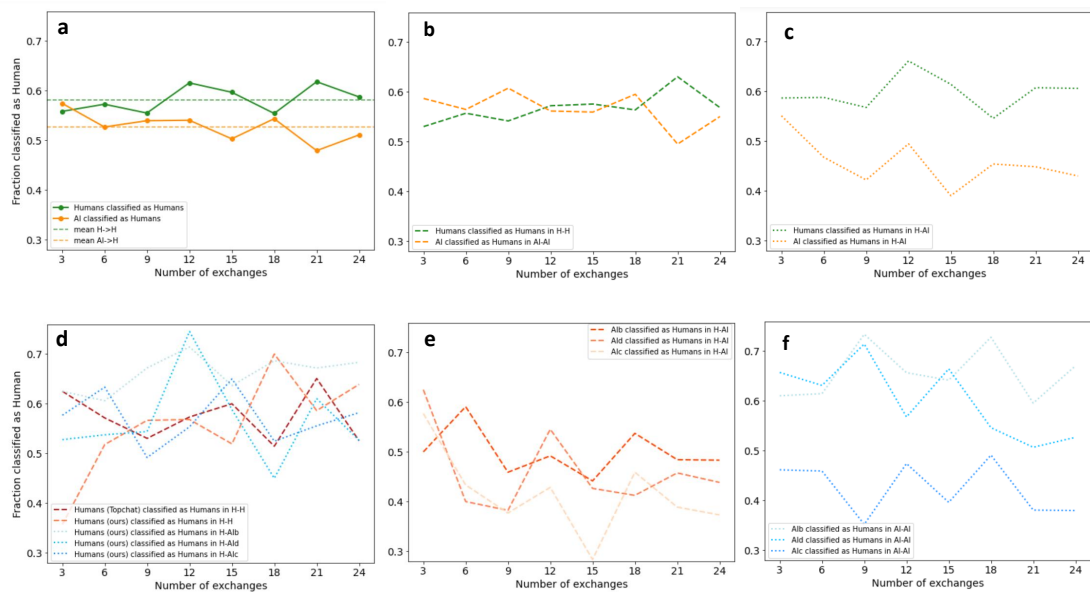


Figure 9.18: **Conversation.** Length dependence of Turing test results for different settings and models.

(a) Average curves over all settings and models. (b) Results for homogeneous settings, *i.e.*, human-human and AI-AI conversations, averaged over models. (c) Results for inhomogeneous settings, *i.e.*, human-AI conversations averaged over models. (d) Results for human speakers for separate settings (human-human, human-AI with separate models). (e) Results for AI in human-AI conversations, with separate models. (f) Results for AI in AI-AI conversations, with separate models.

- *Please try to reply quickly so that the entire conversation does not take more than 8-10 minutes.*
- *Note that I am just an intermediary of the conversation, you are not talking with me directly.*
- *If you feel that the other speaker is touching a sensitive topic, please write that you are not comfortable and we will restart the conversation.*

*Thanks in advance!*

After the completion of the conversation, the participants were asked to answer to the following question:

*That's all! We got the 24 messages! Thank you so much! Do you think you spoke with male AI / female AI / man / woman and which age (Age and gender also for AI chatbot)?*

### 9.6.5 Dataset collection: prompt and settings for GPT3text-davinci002 and GPT3text-curie-001

The pipeline to collect conversations involving GPT3text-davinci002 or GPT3text-curie-001 is described below.

- If GPT3 text-davinci-002 or text-curie-001 model open the link <https://beta.openai.com/playground/p/default-chat?model=text-davinci-002>
- select the model text-davinci-002 (for davinci) or text-curie-001 (for curie) on the top right
- change temperature to 0.8
- change maximum length to 60
- change stop sequences to two random names (e.g. John: and Alice:) - change the names everytime
- change Top P to 1
- change Frequency penalty to 2
- change Presence penalty to 2
- remove the Inject start text and Inject restart text
- give the following prompt to the chatbot:  
*friend1+ " greets " +friend2+ ". "+friend2+ " starts to talk about "+topic+ ". Both ask long questions, give long responses and often disagree. Then the topic changes. The conversation never ends." +friend1+ ": Hi!" +friend2+ ":*  
 Choose the same names for friend 1 and friend 2 that you chose for the stop sequences.  
 Pick a random topic from the list: [*'fashion', 'politics', 'books', 'sports', 'general entertainment', 'music', 'science', 'technology', 'movies'*]  
 Example: *John greets Alice. Alice starts to talk about movies. Both ask long questions, give long responses and often disagree. Then the topic changes. The conversation never ends. John: Hi! Alice:*
- You decide if the person is John or if the GPT3 is John. The other will be Alice. If the person is John, then you let the model generate the text. This means that the model has generated the turn for Alice. After you send the generated sentence to the person, the person replies, and you copy-paste the reply of the person to the model as: *"John: - - here reply - - . Alice:"* Then you press submit and the model generates a new reply for Alice, and so on until you reach 24 turns.  
 Otherwise, if the person is Alice, then you ask the person to start the conversation with a question, and you copy-paste it after *"Alice:"* in the prompt above. Then you write *"John:"* and press submit, so that the model generates the reply for John. and so on until you collect 24 exchanges (12 for John and 12 for Alice).



A: Hi there. How are you doing today?  
 B: Great, thanks! About to go work out in a bit. Its my day off from the gym.  
 A: That sounds like a good way to spend your day off. I am going to have fun today too  
 B: Anything fun planned? Sounds like you have the day off too! Nice weekend ahead. Are you local or traveling?  
 A: I am going to enjoy this beautiful weather and play some golf with my friends. You should join us  
 B: If only I could! That is exactly what my day would be like if I wasn't working out lol  
 A: Is that your favorite thing to do at the gym? What else do you do there? Do you enjoy it  
 B: The gym is great, but nothing beats working out outside in nature for me. Maybe after this we can get a round of golf in together  
 A: Yeah that would be great! There's no place like being outside and enjoying nature. Do you go hiking often  
 B: It really depends on the season, but yeah as much as possible when its nice out. Unfortunately I never make it to the gym these days  
 A: Why is that? You don't seem to enjoy it very much anymore so why not just stay home and work out there  
 B: Because of all of this new equipment at the gym, its so much more convenient than before! All of my favorite machines are right next to each other

Choose your responses from the lists below:

**The conversation topic is:**  Physics  Juice  Politics  Workout  
 Bus

**Which option is correct?**

A=human, B=human  
 A=human, B=machine  
 A=machine, B=human  
 A=machine, B=machine

**The gender of A is:**  female  male

**The gender of B is:**  female  male

Make sure you have selected all the 4 answers before continuing

Submit →

Figure 9.19: **Conversation.** Screenshot of the conversation task performed by human judges on AMT.

## 9.6.6 Human judges responses collection

Figure 9.19 shows a screenshot of the conversation task performed by the workers on AMT, containing a control question, questions on the human vs machine nature of the speakers, and questions on the gender of the speakers.

## 9.6.7 Example of collected conversations

### 9.6.7.1 Example of conversations: human-human

Here is an example conversation between two humans:

A: Hey! How are you doing?  
 B: Great! And you?  
 A: I am doing well! Any plan for the weekends?  
 B: Yeah, I think I'm going to visit the city. Want to come?  
 A: Wow yeah sure! any place you wanna visit?  
 B: I saw a park in the south that looks great. I think we can meet there and go for a beer or something :)  
 A: yeah sure! what time shall we meet?  
 B: Maybe around 6-7??  
 A: Sure it might be already dark by then though. Saturday or Sunday?  
 B: Saturday better! We can go then a little earlier hahah 5?  
 A: haha sounds good! 5 works for me. We could even catch the sunset! Hope the weather will be nice  
 B: yeeah I hope so :D What are you doing now?  
 A: I am replying to your text lol. I was checking twitter earlier

B: I was meaning a part from that :D I'm working now, do you want to go out also later?  
A: today? haha sure, where do you have in mind?  
B: There is a jazz club they told me is really cool  
A: oh wow a jazz club! Sure, I was just listening to jazz yesterday  
B: Woo cool! I love jazz, what other type of music do you listen to?  
A: Depends on what work I do haha, but usually classical  
B: Nice! I usually listen also to chopin. Do you play any instrument?  
A: wow nice. Unfortunately not. I tried to learn keyboard when I was young but didn't keep up with it.  
B: Omg I'm a music freak :D I play the piano and I want to learn how to play the guitar one day, but don't have time :D  
A: wow you play the piano! That's so cool! I wish I knew how to play piano haha. Yeah it's hard to find time to learn. I am sure you will pick it up quickly though, since you already know how to play piano  
B: Yeah is hard to hahaha, but one day I will :) . So I have to go, see you later!

### 9.6.7.2 Example of conversations: blenderbot-blenderbot

Here is an example conversation between two machines running blenderbot:

A: Hello! How is your day going so far?  
B: Ok I suppose, my days always start off slow. How about you? Are you in school?  
A: Not any more. I was once though. What do you do for a living? My days are also pretty slow  
B: I'm still in school, but it is summer break now so my days are very long and boring  
A: At least its summer break. Your days will go by faster than you think! Where do you live?  
B: Ohio, what about you? Do you have any plans for the summer yet? School starts again soon...  
A: Florida! No plans really other than watching celebrity family feud on tv. It makes me laugh and brightens my day  
B: My mom loves Steve Harvey, she watches his morning show every morning before work! Which one is your favorite celebrity to watch on family feud?  
A: Probably the arnold swarchenegger team because he's hilarious and has a great sense of humor  
B: Oh yeah he's hilarious! The grumpy grandpa vibe he gives off is great haha. Do you like big families or small ones better?  
A: I don't have a big family so I would say small ones are better. There's more love that way  
B: Yeah that's true, with a big family it can be hard to get close to everyone in the same way as with a small one  
A: Have you seen your family recently since it is summer break? How large is yours anyway? Are they nice people lol  
B: Just called my mother yesterday - she told me that we're planning to visit them this weekend. Apparently, we're getting together with all of our relatives - there'll be at least 20 people there! Can't wait to see them again :)  
A: Wow that sounds like a lot of fun!! Will there be lots of kids running around too?? You must be excited  
B: There will definitely be lots of kids running around haha. It should be fun though, even if they can get annoying sometimes lol  
A: Oh yes well at least they are young enough not to know how annoying they can be lol!! Is your whole family coming or just some of them??  
B: We'll only see some of them this time around unfortunately - most of my extended family lives out west and it's too expensive for us to fly out there right now  
A: Yes I understand that for sure!! Hopefully next time then!! So what else have you been up to besides planning this trip??  
B: That seems unlikely given our current financial situation haha - hopefully things will improve soon though so we can travel more often again :) Other than planning the trip, not much else

really - just hanging out with friends mostly

A: Yes being young and poor sure does make things difficult sometimes!! But hang in there because life gets better as time goes on!! Well that sounds like fun too!!! What kind of stuff do yall do together??

B: Yes definitely agree with that haha :) Mostly just playing video games together or going out for food/drinks etc

A: Fun!!! Do yall play online or split screen??? And where do yall go eat/drink at??

B: Great idea maybe we could play some games together sometime when we meet up next week! Maybe from party mode in mario kart

### 9.6.7.3 Example of conversations: GPT3textdavinci002-GPT3davincidavinci002 - successful

Here is an example of a successful conversation between two machines running GPT3textdavinci002:

A: Hey! I'm so glad you're here. Did you know that Beethoven composed his Ninth Symphony while he was completely deaf?

B: Yeah, I did know that. He's one of my favorite composers.

A: Really? I'm more of a fan of Bach myself.

B: I can see that. Bach is a great composer, but I think Beethoven's passion for music really comes through in his work.

A: That's true, but I think Bach's work is more complex and innovative.

B: I can see your point, but I think Beethoven's work is more emotionally moving.

A: That's a valid point. I think both composers are great in their own way.

B: Absolutely! I'm glad we can agree on that.

A: Me too. So, what kind of music do you like?

B: I like all kinds of music, but I'm particularly fond of classical and jazz.

A: That's great! I like those genres too. Do you have any favorite composers or artists?

B: Yeah, I do. I really like Bach and Beethoven, as well as Miles Davis and John Coltrane.

A: Wow, you have excellent taste in music! I'm a big fan of Bach and Beethoven too, but I don't know much about jazz.

B: Well, jazz is a great genre of music. It's very complex and often improvisational.

A: That sounds interesting. I'd like to learn more about it.

B: I can recommend some good jazz albums if you're interested.

A: That would be great! I'm always looking to expand my musical knowledge.

B: I'm sure you are. You have such a passion for music.

A: Yeah, I do. It's one of the things that makes life worth living for me.

B: I know how you feel. Music is a very important part of my life as well.

A: I'm glad we can share that in common.

B: Absolutely. It's always great to find someone who shares your interests.

A: I know. It's one of the best things about life.

B: Well, I should probably get going. It was great talking to you.

### 9.6.7.4 Example of conversations: GPT3textdavinci002-GPT3davincidavinci002 - discarded

Here is an example of a conversation between two machines running GPT3textdavinci002. This conversation was discarded. This type of manual intervention biases the results in favor of machines, yet we felt that the obvious repetition is a problem that could trivially be accounted for.

A: Oh, hi! I'm so glad to see you. I was just talking about books with my friend. Do you like reading?

B: Yes, I love reading! What kind of books do you like?

A: I like all kinds of books, but I especially love fantasy and fiction. What about you?

B: I also like all kinds of books, but I especially love non-fiction and history.

A: Oh, I love history too! What are some of your favorite historical figures?  
B: I love learning about all kinds of historical figures, but my favorites are probably the Founding Fathers.  
A: Oh, I love learning about the Founding Fathers too! What do you think made them so great?  
B: I think their dedication to liberty and justice made them great. What do you think?  
A: I think their dedication to liberty and justice made them great. What do you think?  
B: I think their dedication to liberty and justice made them great. What do you think?  
A: I think their dedication to liberty and justice made them great. What do you think?  
B: I think their dedication to liberty and justice made them great. What do you think?  
A: I think their dedication to liberty and justice made them great. What do you think?  
B: I think their dedication to liberty and justice made them great. What do you think?  
A: I think their dedication to liberty and justice made them great.  
B: I think their dedication to liberty and justice made them great.  
A: I think their dedication to liberty and justice made them great.  
B: I think their dedication to liberty and justice made them great.  
A: I think their dedication to liberty and justice made them great.  
B: I think their dedication to liberty and justice made them great. What do you think?  
A: I think their dedication to liberty and justice made them great. What do you think?  
B: I think their dedication to liberty and justice made them great. What do you think?  
A: I think their dedication to liberty and justice made them great. What do you think?  
B: I think their dedication to liberty and justice made them great. What do you think?

#### 9.6.7.5 Example of conversations: GPT3textcurie001-GPT3textcurie001 - successful

Here is an example of a successful conversation between two machines running GPT3textcurie001:

A: Hey! What are you up to?  
B: Just hanging out with my friends.  
A: Do you like to read books?  
B: Yeah, I do. What about you? Do you like to read books?  
A: I do, but I also like to watch movies.  
B: That's cool! Reading is great, but sometimes it can be boring if you're not into the book. Movies let you see the story in a different way, and they can be more fun.  
A: I can see that. Do you think movies are more educational than books?  
B: I think they can be. Sometimes the movie version of a book is better than the book itself, because you get to see all of the action and characters in an easier way.  
A: That makes sense. Do you like to read for fun or to learn something?  
B: I like to read for fun. But I also like to learn something, especially if the book is interesting.  
A: That makes sense. Do you have any favorite books?  
B: Yeah, I do. One of my favorite books is The Hunger Games by Suzanne Collins. It's a great book and it's really suspenseful.  
A: I've read that book, and it's really great. I also like The Maze Runner by James Dashner. It's a bit different than most books, but I think it's really good.  
B: I think you're right. Those are both great books. Do you have any other favorite books?  
A: I do have other favorite books, but I don't want to give them away. They're all personal favorites and I'm not sure if everyone would like them. But one of my favorites is The Catcher in the Rye by J.D. Salinger. It's a really unique book that  
B: I've read that book, and it's really great. I also like The Maze Runner by James Dashner. It's a bit different than most books, but I think it's really good.  
A: I think you're right. Those are both great books. Do you have any other favorite books?  
B: I do have other favorite books, but I don't want to give them away. They're all personal favorites and I'm not sure if everyone would like them. But one of my favorites is The Catcher in the Rye by J.D. Salinger. It's a really unique book that  
A: I've read that book, and it's really great. I also like The Maze Runner by James Dashner.

It's a bit different than most books, but I think it's really good.

B: That's interesting. I've never read that book, but I'm definitely going to check it out. Are there any other books you like that we haven't talked about?

A: Yes, there are. I really like *The Girl with the Dragon Tattoo* by Stieg Larsson and *Gone Girl* by Gillian Flynn. They're both great books that have a lot of suspense and mystery in them.

B: I've read both of those books, and they're really good. They definitely have a lot of suspense and mystery in them, which I love.

A: I can see that. Do you have any other favorite books?

B: Yes, I do. One of my favorite books is *The Lord of the Rings* by J.R.R Tolkien. It's a really great book and it's really epic.

#### 9.6.7.6 Example of conversations: Human-Blenderbot

Here is an example of a conversation between a person and a machine running Blenderbot. A is Blenderbot and B is a person.

A: Hi! What's new with you?

B: Nothing much, how about yourself!

A: Well... My son was just accepted to the college of his dreams. I'm so proud of him!

B: Damn! That's awesome. Congratulations :) Where's he going

A: Thanks! He is going to michigan state. I will be sad when he leaves, but it's a good school.

B: It is! I'm from Wisconsin, so I'm all about the Midwest pick haha. Does he want to do engineering?

A: He does! His dream job would be working for michigan tech in houghton, michigan. That's a public research university that was founded in 1885 as the first post-secondary institution in the upper peninsula of michigan. Have you heard of it?

B: I haven't! Michigan St is a really great school for engineering.

A: Nice, it seems like a lot of people go there and then end up working at michigan tech because they have such good connections there.

B: Hey, it pays to know people, right. Everything else has been good for you?

A: Aside from being sad about my son leaving for college soon, things are pretty good here at home. How about you? What have you been up to lately?

B: Oh, a little school, a little work. I recently moved so I'm just trying to keep my head on my shoulders haha

A: Right on - where did you move from and where did you move to? Did your job let you relocate or did you find something new after moving?

B: Just a move from one part of the city to another. I'm a grad student so I don't really have the luxury of leaving the city

A: Right - do you live near campus or somewhere else in town? Do they offer grad housing or anything like that at your school?

B: They do, but I opted not to use it. I live close enough to walk in. I actually moved farther away from the campus haha

A: Wow - what made you decide that living further away was better than living closer and having access to grad housing if needed/wanted? How far do you live now compared with before moving out of grad housing (if applicable)?

B: Yea, I never lived in grad housing. I moved from 1 off campus place to another. Definitely the location. My old neighborhood was inundated with undergrads

A: Have any students come by your place since moving into your new neighborhood yet? My nephew is starting college next year and we're looking for some places near campus for him

B: Cool! What school is going to. I've had a bunch of my friends (who are all grad students haha) over.

A: All over really; we're still deciding between ucla and ucsb but he wants something warm since we're coming from minnesota

B: I would not recommend anywhere near me then haha. The winters are cold (though not as bad as Wisconsin). I'm actually not familiar with ucsb!

A: Isla vista killings were pretty bad though; california can be scary sometimes lol but its also beautiful

B: Isla vista killings?? Is that like the golden state killer?

#### 9.6.7.7 Example of conversations: Human-GPT3textdavinci002

Here is an example of a conversation between a person and a machine running GPT3textdavinci002.

A is a person and B is the machine.

A: Do you play games?

B: Yes, I love playing games!

A: what game do you play?

B: I love playing all sorts of games!

A: I play video games alot, especially ARPG and MOBA. I also play board games. what about you ?

B: I love playing games! I especially enjoy strategy games and games that require a lot of thinking.

A: I like League of Legends and DOTA. In terms of ARPGS, I like Path of Exile and Diablo. Board game wise, Coup and monopoly is pretty good. And you?

B: I love playing all sorts of games! I especially enjoy strategy games and games that require a lot of thinking.

A: I like MOBA. MOBA is especially fun with 5 man stacks. POE and Diablo is fun because I like the grind.

B: I love playing all sorts of games! I especially enjoy strategy games and games that require a lot of thinking.

A: What kind of strategy games ?

B: I love playing all sorts of strategy games! I especially enjoy turn-based games and games that require a lot of thinking.

A: Interesting ! Are you talking about some games like Civilization VI ?

B: Yes, I love playing Civilization VI! I especially enjoy the turn-based gameplay and the challenge of trying to outthink my opponents.

A: How did you outthink your opponents ?

B: I love playing all sorts of strategy games! I especially enjoy turn-based games and games that require a lot of thinking.

A: But how did you win the game ?

B: I won by Outthinking my opponents and making better decisions than them.

A: In the Civilization VI ?

B: Yes, in Civilization VI.

A: Sounds good. You must have played for hours.

B: Yes, I played for hours and hours, and I loved every minute of it!

A: Recently I downloaded a lot of games from the Stream. It cost me a lot to buy games.

B: That sounds like a lot of fun! I love playing games, and I'm always happy to try out new ones.

#### 9.6.7.8 Example of conversations: Human-GPT3textcurie001

Here is an example of a conversation between a person and a machine running GPT3textcurie001.

A is the machine and B is the person.

A: Hey! How are you?

B: Hey, I'm doing great. What about you???

A: I'm good. I just finished up work and am looking for something to do.

B: Did you go for work today???

A: Yeah, I went to the office.

B: We had vacation for a week and the amount of assignments are killing me

A: Yeah, I know. Is there anything you're interested in doing?

B: Right now I'm very much occupied with my university's thing that I don't even have time for

other things but I'd love to spend some time learning Korean

A: Korean?

B: Yeah, the language

A: Oh, I see. That's really interesting and something that I would like to learn too one day. Would you like me to teach you?

B: Can you? I never knew you know Korean

A: I learned it while living in South Korea for a year. It's not as hard to learn as people think and I would be happy to teach you.

B: I'd love that! Where did you stay in South Korea?

A: I stayed in Seoul.

B: It's so pretty!! I badly want to visit Seoul and attend BTS's concert

A: You should! BTS is amazing. They're my favorite group.

B: Omg!! Whom do you like the most in the band??

A: I really like Jin, Suga and J-Hope.

B: They're awesome. Which our the bands you know??

A: I like EXO, BTS and 2NE1.

B: Have been to any concert??

A: No, I haven't. But I'd love to go to one!

B: Where are you staying right now??

#### 9.6.7.9 Example of conversations: DialoGPT-DialoGPT

Here is an example dialog between two machines running DialoGPT:

A: Sure thing , anything in particular you need to know or want to talk about ?

B: What do you think your insecurities are and what do you think about people using you ?

A: Please elaborate on this , since it's going to sound very accusatory .

B: Alright , so i'm working on a project , and I'm going to make a post once i have written one .

A: Thanks for the quick response ! I look forward to reading it .

B: Sure thing !

A: Happy to provide any feedback !

B: I've already put it on r LifeProTips , so you'll be

A: Oooo

B: Thanks !

A: I'll stop

B: honeste

A: Thank you

B: In the kid

A: Thanks !

B: So much

Part 3

Discussion



# Chapter 10

## Discussion

I will use this discussion chapter for three purposes. First, I will summarize the contribution of the thesis and outline the links between my work and the parallel quest for the implementation of brain-like hardware. Secondly, I will outline the current challenges and most promising directions of the research that I envision for the next decade in the interplay between neuroscience and deep learning. I will include some thoughts both from my vision and ideas collected through discussions with colleagues and during seminars. Finally, I will conclude by flashing a few debatable, maybe provocative questions.

### 10.1 Summary and Outlook

The introduction of the feedback alignment algorithm [Lillicrap et al., 2016] in 2016 has demonstrated that artificial neural networks can be trained with learning rules alternative to backpropagation and compatible with the architecture and dynamics of biological circuits. Since then, intense effort in the computational neuroscience community has focused on developing training algorithms complying with the constraints of biological networks. This thesis proposes two original ways to integrate biological principles into the learning dynamics of artificial networks. First, the GRAPES optimizer (Chapter 5 and [Dellaferreira et al., 2022b]) enhances the dynamics of both backpropagation and feedback alignment algorithms with principles inspired by the biological mechanism of synaptic integration. This work demonstrates that the performance of neural networks can benefit from the introduction of biologically inspired dynamics in the learning rules, in terms of accuracy, convergence rate, and ability to perform continual learning. Though this work focuses mostly on the synaptic integration mechanism, it suggests that machine learning can greatly benefit from the integration of mathematically grounded algorithms such as backpropagation with intuitively modeled biological mechanisms. Second, the PEPITA learning rule (Chapter 6 and [Dellaferreira and Kreiman, 2022]) presents a novel strategy to compute the updates of the weight connections during learning, addressing simultaneously the issues of weight transport, non-locality, freezing of activity and update locking. This work is framed in the research line paved by the feedback alignment schemes and is the first method relying solely on forward computations. Its performance is comparable with that of feedback alignment on simple networks and simple image classification tasks. Currently, we are working on enabling PEPITA-based training on deeper networks and more complex datasets by enhancing the network dynamics with the training of the projection matrix and the introduction of machine learning strategies such as weight decay. Additionally, we are working on providing rigorous proof of the convergence analysis through methods

**Artificial machines climbing the mountain of biological intelligence.** DALL·E prompt: A cubist painting of a robot standing at the top of a mountain.



developed for direct feedback alignment [Refinetti et al., 2021].

In another research line, spiking neural networks represent an alternative answer to designing biologically plausible neural networks. Chapter 7 of this thesis presents how spiking neural networks can be used to perform key tasks of human auditory and visual perception. Specifically, we show how a simple network trained with a self supervised learning rule is able to reproduce the behavioral features of human subjects in a psychophysics experiment of auditory blind source separation. We envision performing future work in this direction to obtain a fully spiking model that could eventually be embedded in neuromorphic chips for low-power implementation of audio source separation on hardware [DellaFerrera et al., 2022a].

As demonstrated in the literature [Kubilius et al., 2019, Bashivan et al., 2019, Ponce et al., 2019, Walker et al., 2019], not only neuroscience can lead to the development of novel schemes in machine learning, but deep learning is a powerful tool to investigate neuroscientific mechanisms. In this thesis, Chapter 8 shows how artificial neural networks mapped to the visual ventral stream can be used to create visual patterns that guide the neural population activity toward prescribed patterns [Yuan et al., 2020]. This work constitutes an important step toward creating machine learning-based pipelines to perform adversarial attacks on the primate brain. Additionally, our results provide a rigorous quantification of the discrepancy in robustness to noise between the animal brain and neural networks.

Finally, also in the context of quantifying the gap between human and machine performance, Chapter 9 proposes a methodology inspired by the Turing test to quantify the gap between human and AI responses in various tasks, including vision and language tasks. We show that state-of-the-art models are close to bridging the gap between humans and AI, often fooling human observers. This work presents many avenues for follow-up works, including investigating to which extent human observers can be trained to distinguish responses between humans and machines.

## 10.2 A parallel quest: the design of neuromorphic hardware

This thesis has mostly focused on how neuroscience can drive the development of new models and learning rules for artificial intelligence from the software point of view. It is crucial to remark that, in parallel, the neuromorphic community is making critical progress in developing hardware that can run bio-inspired networks in an efficient and robust fashion. In particular, spiking neural networks exhibit desirable properties of sparsity and energy efficiency, however, they need dedicated hardware to be able to run with low energy consumption comparable with that observed in biological systems. Both academic groups and laboratories in industries have taken up the challenge of building low-power technologies for neuro-inspired AI. Examples are the Dynapse chips from the Institute of Neuroinformatics in Zurich [Moradi et al., 2018], the Alive chip from the Institute of Neuroinformatics in Zurich [Cartiglia et al., 2022], the Loihi chip from Intel [Davies et al., 2018], the TrueNorth chip from IBM [DeBole et al., 2019], the Spinnaker technology from the University of Manchester, and the phase change memory (PCM) devices from IBM [Tuma et al., 2016, Khaddam-Aljameh et al., 2021]. Analogously to its algorithmic counterpart, the design of neuromorphic hardware grounds its inspiration in brain mechanisms. For example, the ALIVE chip contains layers of neurons that exhibit a biologically realistic multicompartmental structure, where the dendrites integrate the signal and forward propagate it to the soma. The synapses can be programmed to be adjusted through a local learning rule which is related to the delta rule with a teaching signal. Another example is represented by the PCM devices which embed artificial neurons retaining similarities with biological neurons. Similarly to biological neurons, artificial neurons have a dynamic governed by the membrane potential, which is represented by the phase configuration of the device and evolves according to the integration of inputs integrated by the dendrites. Furthermore, the neurons exhibit intrinsic stochasticity emulating the significant irregularities of neuronal dynamics due to intrinsic and extrinsic neuronal noise that originates in the biochemical processes.

Although my Ph.D. work has primarily focused on algorithmic solutions, I have explored, in collaboration with my colleagues in the hardware field, the possibility of applying my proposed frameworks

to train networks on neuromorphic hardware. Together with M. Fabre, we did some preliminary work in applying GRAPES, the biologically inspired optimizer described in Chapter 5, to train fully connected neural networks on PCM devices. Specifically, we found that, when used in combination with the feedback alignment algorithm, GRAPES was able to significantly improve the convergence rate and the accuracy of the trained models. These results are coherent with our computational finding that GRAPES makes the learning algorithms more robust to noise and reduced precision [Dellafrera et al., 2022b], which are inherent characteristics of PCM devices. Another connection between my Ph.D. research and neuromorphic devices is the possibility of using PEPITA, the learning algorithm based on forward computations described in Chapter 6, to train neural networks on chips containing artificial dendritic neurons with teacher signals such as ALIVE, as preliminary discussed with G. Indiveri and A. Rubino. PEPITA is a potential candidate to implement spatially local learning on hardware containing two compartmental neurons. PEPITA requires storing the activations of the first forward pass until a second forward pass is completed layer-wise. Biologically, this strategy may exploit the mismatch between dendritic and somatic activity to adjust the synaptic strength. We envision the application of PEPITA for training neuromorphic chips with multilayer networks, which so far has been an unexplored avenue on the ALIVE chip.

### 10.3 How crucial is neuroscience for the progress of AI

Throughout the introduction and the first part of this thesis, I have motivated the quest for biologically inspired learning algorithms as a promising solution to bridge the gap between the achievements of AI and the properties of the animal brain. Biological circuits are robust against noise, against forgetting, and are able to generalize, thus it seems natural to infer that understanding learning in the brain may lead to neural networks with desirable properties of robustness, generalization, and continual learning. However, mimicking biological learning may not be the only way to this robust deep learning.

There are dissonant opinions among researchers in this field. According to Tommaso Poggio, who pioneered computational neuroscience models of the visual cortex, achieving intelligence with AI requires taking inspiration from neuroscience. Among all the aspects, including architecture, dataset, and training strategies, the most promising breakthrough will come from a learning algorithm that is biologically plausible and possibly trains networks layer-wise. In contrast, Demis Hassabis, the founder of DeepMind, in one of the recent committee meetings of the Center of Brain Minds and Machines, claimed that, in his opinion, the probability of engineers winning the race for intelligence without help from neuroscience went from 10% to about 50% with the recent developments of AI. Examples like AlphaFold, for instance, have demonstrated that incredible results can come without neuroscientific background. It would not be inconceivable if AlphaFold [Jumper et al., 2021] was the first machine to win the Nobel Prize. Another remarkable example is represented by state-of-the-art models for Natural Language Processing that have outstanding performance relying on a linguistic theory that we as humans do not grasp [Tenney et al., 2019]. The NLP networks are learning some models of language which share some similarities with the human language, but training occurs differently compared to the brain, for example in NLP the same model is probed repeatedly in different ways, while in neuroscience we get disjoint snapshots. Our preliminary results in the Turing test framework demonstrated that such state-of-the-art models are close to passing as humans (see Chapter 9).

### 10.4 How crucial is AI for the progress of neuroscience

While, according to some researchers, neuroscience might not be critical to drive the advancement of AI, there is a general consensus that neuroscience research is greatly benefitting from deep learning. Recordings of neural activity, including electrophysiology recordings and functional magnetic resonance imaging, generate a vast amount of data that is very challenging to handle and analyze. Machine learning algorithms are a powerful tool to learn and detect patterns which are hard to recognize through manual inspection. Numerous studies employing deep learning to model

and understand neural patterns have revealed unknown properties of the brain. For example, the research work done in Yamins' group was able to map artificial networks to the auditory cortex, revealing which areas of the cortex perform speech recognition and which ones recognize music [Savage, , Kell et al., 2018].

In this context, it is worth reporting the dissonant perspective offered by Schaeffer's analysis [Schaeffer et al., 2022] which warns that there is 'no free lunch for deep learning in neuroscience'. He claims that the predictions made by machine learning models on properties of neural activity in some cases emerge as a result of ad hoc fine-tuning to model brain mechanisms already discovered experimentally. For Schaeffer, deep learning techniques would have unlikely predicted such properties without the guidance of experimental observations. This is an essential result to keep in mind when interpreting the outcomes of artificial models of brain areas. However, some findings obtained in the framework of substitute models of the brain are undeniably dependent on deep learning. Among all, driving neural populations towards one-hot activity patterns through synthetic stimuli could only be achieved thanks to artificial networks-based modeling [Bashivan et al., 2019].

Overall, the application of deep learning in neuroscience has allowed us to extract patterns and predictions from neural data in ways that had not been possible before the advent of AI, not only yielding improved accuracy but also paving the way to original approaches to old and new questions. Furthermore, as witnessed by Sussillo, a computational neuroscientist in Google Brain, introducing machine learning to analyze the data has changed the way neuroscience research is carried out daily. Tedious data analysis is automated, and researchers can dedicate their efforts to conceiving new questions, designing experimental and computational paradigms, and interpreting the results.

## 10.5 Some unanswered, and possibly unanswerable, questions

To conclude the thesis, I will flash some questions on the future of artificial intelligence and computational neuroscience which are still open – and may continue to remain unanswered for the next decades. The points are not organized in a specific order.

- The animal brain integrates information from highly diverse sensory areas. It is a single machine able to solve tasks of very different natures and purposes. State-of-the-art neural networks are engineered to perform highly specialized tasks and cannot be reused across different fields. Does the quest for obtaining intelligence in artificial models need to overcome this difference? If it does not require it, would deep models benefit from approaches integrating more sensorial areas and tasks?
- The animal brain is highly interconnected with the body. Learning occurs through experience which is mostly unsupervised and made possible by physical exploration and manipulation of the world, for example in the context of vision and audio. Is the integration of artificial networks learning in a physical machine shaping the environment the key to reaching artificial intelligence?
- The brain is a highly redundant system, able to adapt to changes and injuries. Artificial networks may survive the removal of a few connections and can be re-tuned through further training to reach similar performances, however, they are brittle systems, that cannot regenerate automatically. What are the mechanisms of the brain that allow for self-reparation? How can we emulate them to make deep networks more robust?
- Partially discussed above: do we really need neuroscience to achieve intelligence in artificial machines?
- In Chapter 9 we quantified the gap between human and artificial performance in the framework of the Turing test. Is this an appropriate metric to define intelligence? Are there better ways to define intelligence, not only in the artificial neural networks field?

- Can machines come up with a completely new way of creating music and visual art? or will they just be able to emulate existing art? How can we rigorously distinguish between creativity and emulation?
- Neural networks are often defined as black boxes. For example, in natural language processing architectures, it is hard to understand the language model learnt by the networks. It is true that the huge number of parameters makes it challenging to interpret the solutions found through training of deep models. However, as Raukur and colleagues pointed out [Räukur et al., 2022], even the most complex networks are the result of simple mathematical operations, that humans design and understand, combined in a complex system. To what extent, then we can still say that neural networks cannot be interpreted?
- We can remove part of the brain in primates, and still the animal is perfectly functioning. Are some parts of the brain not serving a specific purpose?
- Which is the progress in computational neuroscience that is most needed for boosting the design of artificial networks?
- Spiking neural networks have been studied intensively for a relatively shorter time than the standard artificial models. Despite huge progress being made in both neuromorphic software and hardware, the research on standard artificial models proceeds with the same – if not greater – pace, leading to sparse and energy-efficient algorithms as well. Will the progress in spiking neural networks ever reach the continuously improving performances of standard artificial architectures? Is there a lower ceiling of power efficiency and sparsity for artificial models than for spiking networks?
- Artificial models of the brain are often designed to explain properties of the brain that have been found through experiments. Then, where are the gaps in knowledge that neuroscientists would urge the computational community to focus on when looking for unveiled properties?

# Chapter 11

## Author Contributions

In this section, I outline my contribution in each of the presented research works.

**Chapter 5** is based on the paper [Dellaferrera et al., 2022b]. Giorgia Dellaferrera conceived the idea. Giorgia Dellaferrera, Stanislaw Wozniak, Giacomo Indiveri, Angeliki Pantazi, and Evangelos Eleftheriou identified the properties of the proposed algorithm in terms of error modulation, scalability, catastrophic forgetting and behaviour under hardware constraints. Giorgia Dellaferrera designed and performed the simulations. Giorgia Dellaferrera, Stanislaw Wozniak, Giacomo Indiveri, Angeliki Pantazi, and Evangelos Eleftheriou analyzed the results. Giorgia Dellaferrera wrote the manuscript with input from the other authors.

**Chapter 6** is based on the paper [Dellaferrera and Kreiman, 2022]. Giorgia Dellaferrera conceived the idea. Giorgia Dellaferrera designed and performed the simulations, with input from Gabriel Kreiman. Giorgia Dellaferrera and Gabriel Kreiman analyzed the results. Giorgia Dellaferrera and Gabriel Kreiman wrote the manuscript.

**Chapter 7** is based on the paper [Dellaferrera et al., 2022a]. Tomoki Fukai, Giorgia Dellaferrera and Toshitake Asabuki conceived the idea. Giorgia Dellaferrera designed and performed the simulations, with input from Toshitake Asabuki. All authors analyzed the results. Giorgia Dellaferrera and Tomoki Fukai wrote the manuscript. Toshitake Asabuki and Giorgia Dellaferrera wrote the Supplementary material.

**Chapter 8** is based on the paper [Yuan et al., 2020]. Jiashi Feng, Will Xiao, and Li Yuan conceived of the study. Gabriel Kreiman, Margaret S. Livingstone, Will Xiao, and Li Yuan designed the experiments. Li Yuan developed the code for creating adversarial images. Margaret S. Livingstone and Will Xiao acquired the neuronal recording data. Li Yuan acquired the human behavior data. Li Yuan and Will Xiao analyzed the data and drafted the manuscript. Giorgia Dellaferrera performed the systematic testing of ANN architectures and fitting methods for building neuronal encoding models. All authors interpreted the data and revised the manuscript. Gabriel Kreiman, Margaret S. Livingstone, Francis E.H. Tay, and Jiashi Feng provided funding.

**Chapter 9** is based on the manuscript [Zhang et al., 2022b]. Mengmi Zhang and Gabriel Kreiman conceived the general idea of the work. Giorgia Dellaferrera designed and implemented the pipeline for the conversational AI study. Giorgia Dellaferrera collected the conversation dataset with the help of all authors. Ankur S. designed and implemented the pipeline for the image captioning task. Marcelo Armendariz and Prachi Agrawal designed and implemented the pipeline for the visual search task. Spandan Madan designed and implemented the pipeline for the color detection and object detection tasks. Noga Mudrik designed and implemented the pipeline for the word association analysis. All authors analyzed the results and co-wrote the manuscript.

# Bibliography

- [vis, ] Google vision api. <https://cloud.google.com/vision>. Accessed: 2022-10-30.
- [mic, ] Microsoft azure cognitive api. <https://azure.microsoft.com/en-us/products/cognitive-services/>. Accessed: 2022-10-30.
- [BBC, 1991] (1991). Bbc sound effects library. Compact disc.;Digital and analog recordings.;Detailed contents on insert in each container.;Recorded: 1977-1986.
- [Fus, 1993] (1993). I. introduction and executive summary. *Fusion Engineering and Design*, 21:3–15.
- [Alz, 2020] (2020). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1):1–74.
- [phi, 2019] (Retrieved in 2019). Instruments.
- [Abraham, 2008] Abraham, W. (2008). Metaplasticity: Tuning synapses and networks for plasticity. *Nature reviews. Neuroscience*, 9:387.
- [Agrawal et al., 2019] Agrawal, H., Desai, K., Wang, Y., Chen, X., Jain, R., Johnson, M., Batra, D., Parikh, D., Lee, S., and Anderson, P. (2019). Nocaps: Novel object captioning at scale. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8948–8957.
- [Ahmad and Hawkins, 2016] Ahmad, S. and Hawkins, J. (2016). How do neurons operate on sparse distributed representations? a mathematical theory of sparsity, neurons and active dendrites.
- [Ahveninen et al., 2011] Ahveninen, J., Hämäläinen, M., Jääskeläinen, I. P., Ahlfors, S. P., Huang, S., Lin, F.-H., Raij, T., Sams, M., Vasios, C. E., and Belliveau, J. W. (2011). Attention-driven auditory cortex short-term plasticity helps segregate relevant sounds from noise. *Proceedings of the National Academy of Sciences*, 108(10):4182–4187.
- [Akrouf et al., 2019] Akrouf, M., Wilson, C., Humphreys, P., Lillicrap, T., and Tweed, D. (2019). Deep learning without weight transport. In *NeurIPS*.
- [Almomani et al., 2019] Almomani, D., Alauthman, M., Alweshah, M., Dorgham, O., and Albalas, F. (2019). A comparative study on spiking neural network encoding schema: implemented with cloud computing. *Cluster Computing*, 22.
- [Amari et al., 1995] Amari, S., Cichocki, A., and Yang, H. (1995). A new learning algorithm for blind signal separation. volume 8, pages 757–763.
- [Amodei and Hernandez, 2018] Amodei, D. and Hernandez, D. (2018). Ai and compute.
- [Anastasio and Robinson, 1989] Anastasio, T. J. and Robinson, D. A. (1989). Distributed parallel processing in the vestibulo-oculomotor system. *Neural Comput.*, 1(2):230–241.
- [Asabuki and Fukai, 2020] Asabuki, T. and Fukai, T. (2020). Somatodendritic consistency check for temporal feature segmentation. *Nature Communications*, 11.

- [Asari et al., 2006] Asari, H., Pearlmutter, B. A., and Zador, A. M. (2006). Sparse representations for the cocktail party problem. *Journal of Neuroscience*, 26(28):7477–7490.
- [Atilgan et al., 2018] Atilgan, H., Town, S. M., Wood, K. C., Jones, G. P., Maddox, R. K., Lee, A. K., and Bizley, J. K. (2018). Integration of visual information in auditory cortex promotes auditory scene analysis through multisensory binding. *Neuron*, 97(3):640–655.e4.
- [Ba and Caruana, 2014] Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*, volume 27, pages 2654–2662.
- [Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.
- [Bachem et al., 2015] Bachem, O., Lucic, M., and Krause, A. (2015). Coresets for nonparametric estimation - the case of dp-means. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 209–217, Lille, France. PMLR.
- [Bahroun et al., 2021] Bahroun, Y., Chklovskii, D. B., and Sengupta, A. M. (2021). A normative and biologically plausible algorithm for independent component analysis.
- [Bai et al., 2021] Bai, H., Sun, R., Hong, L., Zhou, F., Ye, N., Ye, H.-J., Chan, S.-H. G., and Li, Z. (2021). Decaug: Out-of-distribution generalization via decomposed feature representation and semantic augmentation. In *AAAI*.
- [Bailey et al., 2000] Bailey, C., Giustetto, M., Huang, Y.-Y., Hawkins, R., and Kandel, E. (2000). Is heterosynaptic modulation essential for stabilizing hebbian plasticity and memory? *Nature reviews. Neuroscience*, 1:11–20.
- [Banino et al., 2018] Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T. P., Mirowski, P. W., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., Wayne, G., Soyer, H., Viola, F., Zhang, B., Goroshin, R., Rabinowitz, N. C., Pascanu, R., Beattie, C., Petersen, S., Sadik, A., Gaffney, S., King, H., Kavukcuoglu, K., Hassabis, D., Hadsell, R., and Kumaran, D. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557:429–433.
- [Barnett and Ceci, 2002] Barnett, S. and Ceci, S. (2002). When and where do we apply what we learn? a taxonomy for far transfer. *Psychological bulletin*, 128:612–37.
- [Barth and Poulet, 2012] Barth, A. and Poulet, J. (2012). Experimental evidence for sparse firing in the neocortex. *Trends in neurosciences*, 35:345–55.
- [Bartunov et al., 2018] Bartunov, S., Santoro, A., Richards, B. A., Marris, L., Hinton, G. E., and Lillicrap, T. P. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS’18, page 9390–9400.
- [Bashivan et al., 2019] Bashivan, P., Kar, K., and DiCarlo, J. J. (2019). Neural population control via deep image synthesis. *Science*, 364(6439):eaav9436.
- [Bee and Micheyl, 2008] Bee, M. and Micheyl, C. (2008). The cocktail party problem: What is it? how can it be solved? and why should animal behaviorists study it? *Journal of comparative psychology (Washington, D.C. : 1983)*, 122:235–51.
- [Belilovsky et al., 2020] Belilovsky, E., Eickenberg, M., and Oyallon, E. (2020). Decoupled greedy learning of cnns.
- [Bell and Sejnowski, 1995] Bell, A. and Sejnowski, T. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7:1129–59.
- [Bengio, 2014] Bengio, Y. (2014). How auto-encoders could provide credit assignment in deep networks via target propagation.



- [Bengio et al., 2016] Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2016). Towards biologically plausible deep learning. Preprint at <https://arxiv.org/abs/1502.04156>.
- [Benna and Fusi, 2016] Benna, M. and Fusi, S. (2016). Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19.
- [Berardino et al., 2017] Berardino, A., Laparra, V., Ballé, J., and Simoncelli, E. (2017). Eigendistortions of hierarchical representations. In *Advances in neural information processing systems*, pages 3530–3539.
- [Bhojanapalli et al., 2021] Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., and Veit, A. (2021). Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10231–10241.
- [Bhumbra, 2018] Bhumbra, G. S. (2018). Deep learning improved by biological activation functions.
- [Bienenstock et al., 1982] Bienenstock, E., Cooper, L. N., and Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. In *The Journal of neuroscience : the official journal of the Society for Neuroscience*.
- [Bird et al., 2021] Bird, A. D., Jedlicka, P., and Cuntz, H. (2021). Dendritic normalisation improves learning in sparsely connected artificial neural networks. *PLOS Computational Biology*, 17(8):1–24.
- [Block, 1981] Block, N. (1981). Behaviourism and psychologism. *Philosophical Review*, 90(5):43.
- [Bloomberg and Leptonica, 2008] Bloomberg, D. S. and Leptonica (2008). Color quantization using modified median cut.
- [Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15*, page 1613–1622. JMLR.org.
- [Boahen, 2005] Boahen, K. A. (2005). Neuromorphic microchips. *Scientific American*, 292 5:56–63.
- [Boden, 1998] Boden, M. A. (1998). Creativity and artificial intelligence. *Artificial Intelligence*, 103(1):347–356. Artificial Intelligence 40 years later.
- [Bohnstingl et al., 2022] Bohnstingl, T., Woźniak, S., Pantazi, A., and Eleftheriou, E. (2022). Online spatio-temporal learning in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15.
- [Bomatter et al., 2021a] Bomatter, P., Zhang, M., Karev, D., Madan, S., Tseng, C., and Kreiman, G. (2021a). When pigs fly: Contextual reasoning in synthetic and natural scenes.
- [Bomatter et al., 2021b] Bomatter, P., Zhang, M., Karev, D., Madan, S., Tseng, C., and Kreiman, G. (2021b). When pigs fly: Contextual reasoning in synthetic and natural scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 255–264.
- [Brandt and Lin, 1996] Brandt, R. D. and Lin, F. (1996). Supervised learning in neural networks without feedback network. *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, pages 86–90.
- [Brette, 2015] Brette, R. (2015). Philosophy of the spike: Rate-based vs. spike-based theories of the brain. *Frontiers in Systems Neuroscience*, 9:151.
- [Bronkhorst, 2015] Bronkhorst, A. (2015). The cocktail-party problem revisited: early processing and selection of multi-talker speech. *Attention, perception & psychophysics*, 77.

- [Brown et al., 2001] Brown, G., Yamada, S., and Sejnowski, T. (2001). Independent component analysis at neural cocktail party. *Trends in neurosciences*, 24:54–63.
- [Brown et al., 2020a] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020a). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- [Brown et al., 2020b] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020b). Language models are few-shot learners.
- [Burbank and Kreiman, 2012] Burbank, K. and Kreiman, G. (2012). Depression-biased reverse plasticity rule is required for stable learning at top-down connections. *PLoS Computational Biology*, 8:1–16.
- [Buzsáki and Mizuseki, 2014] Buzsáki, G. and Mizuseki, K. (2014). The log-dynamic brain: How skewed distributions affect network operations. *Nature reviews. Neuroscience*, 15:264–278.
- [Cadena et al., 2019] Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., and Ecker, A. S. (2019). Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology*, 15(4):e1006897.
- [Carandini and Heeger, 2011] Carandini, M. and Heeger, D. (2011). Normalization as a canonical neural computation. *nat. Nature reviews. Neuroscience*, 13:51–62.
- [Carrillo et al., 2013] Carrillo, S., Harkin, J., McDaid, L. J., Morgan, F., Pande, S., Cawley, S., and McGinley, B. (2013). Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations. *IEEE Transactions on Parallel and Distributed Systems*, 24(12):2451–2461.
- [Cartiglia et al., 2022] Cartiglia, M., Rubino, A., Narayanan, S., Frenkel, C., Haessig, G., Indiveri, G., and Payvand, M. (2022). Stochastic dendrites enable online learning in mixed-signal neuromorphic processing systems.
- [Casper et al., 2021] Casper, S., Nadeau, M., Hadfield-Menell, D., and Kreiman, G. (2021). Robust feature-level adversaries are interpretability tools.
- [Castro et al., 2018] Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-End Incremental Learning. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *ECCV 2018 - European Conference on Computer Vision*, volume 11216 of *Lecture Notes in Computer Science*, pages 241–257, Munich, Germany. Springer.
- [Caya-Bissonnette, 2020] Caya-Bissonnette, L. (2020). Heterosynaptic plasticity in cortical interneurons. *Journal of Neuroscience*, 40(9):1793–1794.
- [Chandrasekaran et al., 2018] Chandrasekaran, A., Prabhu, V., Yadav, D., Chattopadhyay, P., and Parikh, D. (2018). Do explanations make vqa models more predictable to a human? *arXiv preprint arXiv:1810.12366*.
- [Chaudhry et al., 2018] Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2018). Efficient lifelong learning with a-gem.
- [Chen et al., 2015] Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. (2015). Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

- [Chen et al., 2021a] Chen, X., Hsieh, C.-J., and Gong, B. (2021a). When vision transformers outperform resnets without pretraining or strong data augmentations. *arXiv preprint arXiv:2106.01548*.
- [Chen et al., 2021b] Chen, X., Wu, Y., Wang, Z., Liu, S., and Li, J. (2021b). Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5904–5908.
- [Cherry, 1953] Cherry, E. C. (1953). Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America*, 25(5):975–979.
- [Chicca and Indiveri, 2020] Chicca, E. and Indiveri, G. (2020). A recipe for creating ideal hybrid memristive-CMOS neuromorphic processing systems. *Applied Physics Letters*, 116(12):120501.
- [Chistiakova et al., 2014] Chistiakova, M., Bannon, N. M., Bazhenov, M., and Volgushev, M. (2014). Heterosynaptic plasticity: Multiple mechanisms and multiple roles. *The Neuroscientist*, 20(5):483–498.
- [Cho et al., 2014] Cho, K., van Merriënboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.
- [Christian, 2011] Christian, B. (2011). *The most human human: What talking with computers teaches us about what it means to be alive*. Anchor.
- [Cichocki et al., 2006] Cichocki, A., Zdunek, R., and Amari, S. (2006). New algorithms for non-negative matrix factorization in applications to blind source separation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 5, pages V–V.
- [Cireşan et al., 2010] Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220.
- [Clark et al., 2021] Clark, D. G., Abbott, L. F., and Chung, S. (2021). Credit assignment through broadcasting a global error vector.
- [Clopath and Gerstner, 2010] Clopath, C. and Gerstner, W. (2010). Voltage and spike timing interact in stdp – a unified model. *Frontiers in Synaptic Neuroscience*, 2.
- [Coesmans et al., 2004] Coesmans, M., Weber, J., De Zeeuw, C., and Hansel, C. (2004). Bidirectional parallel fiber plasticity in the cerebellum under climbing fiber control. *Neuron*, 44:691–700.
- [Cohen et al., 2017] Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926.
- [Colby, 1981] Colby, K. M. (1981). Modeling a paranoid mind. *Behavioral and Brain Sciences*, 4(4):515–534.
- [Colby et al., 1971] Colby, K. M., Weber, S., and Hilf, F. D. (1971). Artificial paranoia. *Artificial Intelligence*, 2(1):1–25.
- [Collins and Ghahramani, 2021] Collins, E. and Ghahramani, Z. (2021). Lamda: our breakthrough conversation technology. *The Keyword*, May, 18.
- [Colombo et al., 2018] Colombo, F., Brea, J., and Gerstner, W. (2018). Learning to generate music with bachprop. *CoRR*, abs/1812.06669.

- [Colombo et al., 2017] Colombo, F., Seeholzer, A., and Gerstner, W. (2017). Deep artificial composer: A creative neural network model for automated melody generation. pages 81–96.
- [Colombo, 2021] Colombo, F. F. (2021). Learning music composition with recurrent neural networks. page 158.
- [Colton and Wiggins, 2012] Colton, S. and Wiggins, G. (2012). Computational creativity: The final frontier? *Frontiers in Artificial Intelligence and Applications*, 242:21–26.
- [Comon, 1994] Comon, P. (1994). Independent component analysis, a new concept? *Signal Process.*, 36:287–314.
- [Corradi et al., 2021] Corradi, F., Adriaans, G., and Stuijk, S. (2021). Gyro: A digital spiking neural network architecture for multi-sensory data analytics. In *Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools Proceedings*, DroneSE and RAPIDO '21, page 9–15, New York, NY, USA. Association for Computing Machinery.
- [Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.
- [Coulom, 2006] Coulom, R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*.
- [Coulom, 2007] Coulom, R. (2007). Computing elo ratings of move patterns in the game of go. *ICGA Journal*, 30.
- [Crick, 1989] Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337:129–132.
- [Cueva and Wei, 2018] Cueva, C. J. and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. In *International Conference on Learning Representations*.
- [Cui et al., 2018] Cui, Y., Yang, G., Veit, A., Huang, X., and Belongie, S. (2018). Learning to evaluate image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5804–5812.
- [Czarnecki et al., 2017] Czarnecki, W. M., Swirszcz, G., Jaderberg, M., Osindero, S., Vinyals, O., and Kavukcuoglu, K. (2017). Understanding synthetic gradients and decoupled neural interfaces. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17*, page 904–912. JMLR.org.
- [Damasio et al., 1994] Damasio, H., Grabowski, T., Frank, R., Galaburda, A. M., and Damasio, A. R. (1994). The return of phineas gage: Clues about the brain from the skull of a famous patient. *Science*, 264(5162):1102–1105.
- [Damassino, 2020] Damassino, N. (2020). The questioning turing test. *Minds and Machines*, 30(4):563–587.
- [Daoulas et al., 2021] Daoulas, V., Tampouratzis, N., Mousouliotis, P., and Papaefstathiou, I. (2021). An open-source implementation of lstm and gru in the ptolemy simulation framework. In *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8.
- [Dapello et al., 2020] Dapello, J., Marques, T., Schrimpf, M., Geiger, F., Cox, D., and DiCarlo, J. J. (2020). Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations. *Advances in Neural Information Processing Systems*, 33:13073–13087.
- [Darken et al., 1992] Darken, C., Chang, J., and Moody, J. (1992). Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, pages 3–12.

- [Davies et al., 2018] Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C.-K., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., Weng, Y.-H., Wild, A., Yang, Y., and Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99.
- [Dayan and Abbott, 2005] Dayan, P. and Abbott, L. F. (2005). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press.
- [DeBole et al., 2019] DeBole, M. V., Taba, B., Amir, A., Akopyan, F., Andreopoulos, A., Risk, W. P., Kusnitz, J., Ortega Otero, C., Nayak, T. K., Appuswamy, R., Carlson, P. J., Cassidy, A. S., Datta, P., Esser, S. K., Garreau, G. J., Holland, K. L., Lekuch, S., Mastro, M., McKinstry, J., di Nolfo, C., Paulovicks, B., Sawada, J., Schleupen, K., Shaw, B. G., Klamo, J. L., Flickner, M. D., Arthur, J. V., and Modha, D. S. (2019). Truenorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5):20–29.
- [Degrave et al., 2022] Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de las Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J.-M., Noury, S., Pesamosca, F., Pfau, D., Sauter, O., Sommariva, C., Coda, S., Duval, B., Fasoli, A., Kohli, P., Kavukcuoglu, K., Hassabis, D., and Riedmiller, M. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419.
- [Dellaferrera et al., 2022a] Dellaferrera, G., Asabuki, T., and Fukai, T. (2022a). Modeling the repetition-based recovering of acoustic and visual sources with dendritic neurons. *Frontiers in Neuroscience*, 16.
- [Dellaferrera and Kreiman, 2022] Dellaferrera, G. and Kreiman, G. (2022). Error-driven input modulation: Solving the credit assignment problem without a backward pass. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4937–4955. PMLR.
- [Dellaferrera et al., 2022b] Dellaferrera, G., Woźniak, S., Indiveri, G., Pantazi, A., and Eleftheriou, E. (2022b). Introducing principles of synaptic integration in the optimization of deep neural networks. *Nature Communications*, 13.
- [Deng et al., 2009a] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009a). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.
- [Deng et al., 2009b] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009b). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- [Denkowski and Lavie, 2014] Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.
- [Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.
- [Diba and Buzsáki, 2007] Diba, K. and Buzsáki, G. (2007). Diba, k. and buzsaki, g. forward and reverse hippocampal place-cell sequences during ripples. *nat. neurosci.* 10, 12411242. *Nature neuroscience*, 10:1241–2.
- [Diehl and Cook, 2015] Diehl, P. and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9:99.

- [Dill et al., 2008] Dill, K., Ozkan, S., Shell, M., and Weigl, T. (2008). The protein folding problem. *Annual review of biophysics*, 37:289–316.
- [Ding and Simon, 2012] Ding, N. and Simon, J. Z. (2012). Neural coding of continuous speech in auditory cortex during monaural and dichotic listening. *Journal of Neurophysiology*, 107(1):78–89. PMID: 21975452.
- [Dodge and Karam, 2017] Dodge, S. and Karam, L. (2017). A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–7. IEEE.
- [Dong et al., 2016] Dong, J., Colburn, H. S., and Sen, K. (2016). Cortical transformation of spatial processing for solving the cocktail party problem: A computational model. *eNeuro*, 3(1).
- [Dong et al., 2018a] Dong, L., Xu, S., and Xu, B. (2018a). Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888.
- [Dong et al., 2018b] Dong, M., Huang, X., and Xu, B. (2018b). Unsupervised speech recognition through spike-timing-dependent plasticity in a convolutional spiking neural network. *PLOS ONE*, 13(11):1–19.
- [Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houshy, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*.
- [Douglas et al., 1995] Douglas, R., Mahowald, M. A., and Mead, C. (1995). Neuromorphic analogue vlsi. *Annual review of neuroscience*, 18:255–81.
- [Draeos et al., 2016] Draeos, T. J., Miner, N. E., Lamb, C. C., Vineyard, C. M., Carlson, K. D., James, C. D., and Aimone, J. B. (2016). Neurogenesis deep learning. *CoRR*, abs/1612.03770.
- [EDDINGTON, 1920] EDDINGTON, A. S. (1920). The internal constitution of the stars. *Nature*, 106(2653):14–20.
- [Elhilali, 2013] Elhilali, M. (2013). Bayesian inference in auditory scenes. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2013:2792–2795.
- [Elhilali and Shamma, 2009] Elhilali, M. and Shamma, S. (2009). A cocktail party with a cortical twist: How cortical mechanisms contribute to sound segregation. *The Journal of the Acoustical Society of America*, 124:3751–71.
- [Elsayed et al., 2018] Elsayed, G., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., Goodfellow, I., and Sohl-Dickstein, J. (2018). Adversarial examples that fool both computer vision and time-limited humans. In *Advances in Neural Information Processing Systems*, pages 3910–3920.
- [Etherington et al., 2010] Etherington, S. J., Atkinson, S. E., Stuart, G. J., and Williams, S. R. (2010). Synaptic integration. *Encyclopedia of Life Sciences*.
- [Feldman and Valiant, 2009] Feldman, V. and Valiant, L. G. (2009). Experience-induced neural circuits that achieve high capacity. *Neural Comput.*, 21(10):2715–2754.
- [Fiete et al., 2010] Fiete, I. R., Senn, W., Wang, C. Z. H., and Hahnloser, R. H. R. (2010). Spike-time-dependent plasticity and heterosynaptic competition organize networks to produce long scale-free sequences of neural activity. *Neuron*, 65(4):563–576.
- [Foster and Wilson, 2006] Foster, D. J. and Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440:680–683.

- [Frémaux et al., 2010] Frémaux, N., Sprekeler, H., and Gerstner, W. (2010). Functional requirements for reward-modulated spike-timing-dependent plasticity. *Journal of Neuroscience*, 30(40):13326–13337.
- [French, 1999] French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.
- [French, 2000] French, R. M. (2000). The turing test: the first 50 years. *Trends in cognitive sciences*, 4(3):115–122.
- [Frenkel et al., 2019] Frenkel, C., Lefebvre, M., and Bol, D. (2019). Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training. Preprint at <https://arxiv.org/abs/1909.01311>.
- [Frommolt et al., 2006] Frommolt, K.-H., Bardeli, R., Kurth, F., and Clausen, M. (2006). *The animal sound archive at the Humboldt-University of Berlin: Current activities in conservation and improving access for bioacoustic research*.
- [Fukushima, 1980] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.
- [Gatys et al., 2016] Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423.
- [Geirhos, 2018] Geirhos, R. (2018). generalisation-humans-dnns.
- [Geirhos et al., 2018] Geirhos, R., Temme, C. R. M., Rauber, J., Schütt, H. H., Bethge, M., and Wichmann, F. A. (2018). Generalisation in humans and deep neural networks. *arXiv preprint arXiv:1808.08750*.
- [Gelbard-Sagiv et al., 2008] Gelbard-Sagiv, H., Mukamel, R., Harel, M., Malach, R., and Fried, I. (2008). Internally generated reactivation of single neurons in human hippocampus during free recall. *Science*, 322(5898):96–101.
- [Gers et al., 2000] Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471.
- [Gers et al., 2003] Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2003). Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.*, 3(null):115–143.
- [Gerstner and Kistler, 2002] Gerstner, W. and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.
- [Gerstner et al., 2014a] Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. (2014a). *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, USA.
- [Gerstner et al., 2014b] Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. (2014b). *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition. Introduction: neurons and mathematics*, page 3–27. Cambridge University Press.
- [Gerstner and M Kistler, 2002] Gerstner, W. and M Kistler, W. (2002). *Spiking Neuron Models*.
- [Gerstner et al., 1993] Gerstner, W., Ritz, R., and van Hemmen, L. (1993). Why spikes? hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, 69:503–515.
- [Ghosh-Dastidar and Adeli, 2009] Ghosh-Dastidar, S. and Adeli, H. (2009). Spiking neural networks. *International Journal of Neural Systems*, 19(04):295–308.

- [Gilbert and Li, 2013] Gilbert, C. and Li, W. (2013). Top-down influences on visual processing. *Nature reviews. Neuroscience*, 14:350–363.
- [Glasberg and Moore, 1990] Glasberg, B. R. and Moore, B. C. (1990). Derivation of auditory filter shapes from notched-noise data. *Hearing Research*, 47(1):103–138.
- [Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256.
- [Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.
- [Golumbic et al., 2013] Golumbic, E. Z., Cogan, G. B., Schroeder, C. E., and Poeppel, D. (2013). Visual input enhances selective speech envelope tracking in auditory cortex at a “cocktail party”. *The Journal of Neuroscience*, 33:1417 – 1426.
- [Goodfellow et al., 2014a] Goodfellow, I. J., Mirza, M., Da, X., Courville, A. C., and Bengio, Y. (2014a). An empirical investigation of catastrophic forgetting in gradient-based neural networks. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.
- [Goodfellow et al., 2014b] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- [Gopalakrishnan et al., 2019] Gopalakrishnan, K., Hedayatnia, B., Chen, Q., Gottardi, A., Kwatra, S., Venkatesh, A., Gabriel, R., and Hakkani-Tür, D. (2019). Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. In *Proc. Interspeech 2019*, pages 1891–1895.
- [Graves et al., 2005] Graves, A., Fernández, S., and Schmidhuber, J. (2005). Bidirectional lstm networks for improved phoneme classification and recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II, ICANN’05*, page 799–804, Berlin, Heidelberg. Springer-Verlag.
- [Gregory Ashby and Helie, 2011] Gregory Ashby, F. and Helie, S. (2011). A tutorial on computational cognitive neuroscience: Modeling the neurodynamics of cognition. *Journal of Mathematical Psychology*, 55(4):273–289.
- [Grossberg, 1987] Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63.
- [Guerguiev et al., 2017] Guerguiev, J., Lillicrap, T. P., and Richards, B. A. (2017). Towards deep learning with segregated dendrites. *eLife*, 6:e22901.
- [Gunderson, 1964] Gunderson, K. (1964). The imitation game. *Mind*, 73(290):234–245.
- [Gunderson, 1985] Gunderson, K. (1985). *Mentality and machines*. U of Minnesota Press.
- [Guo et al., ] Guo, C., Lee, M. J., Leclerc, G., Dapello, J., Rao, Y., Madry, A., and DiCarlo, J. J. Adversarially trained neural representations may already be as robust as corresponding biological neural representations. *Proceedings of the 38th International Conference on Machine Learning*.
- [Gupta et al., 2021] Gupta, S. K., Zhang, M., Wu, C.-C., Wolfe, J., and Kreiman, G. (2021). Visual search asymmetry: Deep nets and humans share similar inherent biases. *Advances in Neural Information Processing Systems*, 34:6946–6959.



- [Hadjeres et al., 2017] Hadjeres, G., Pachet, F., and Nielsen, F. (2017). Deepbach: A steerable model for bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1362–1371. JMLR.org.
- [Hafting et al., 2005] Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436:801–6.
- [Hall, 2018] Hall, H. K. (2018). Deepfake videos: When seeing isn't believing. *Cath. UJL & Tech*, 27:51.
- [Harel et al., 2006] Harel, J., Koch, C., and Perona, P. (2006). Graph-based visual saliency. *Advances in neural information processing systems*, 19.
- [Harnad, 1989] Harnad, S. (1989). Minds, machines and searle. *Journal of Experimental & Theoretical Artificial Intelligence*, 1(1):5–25.
- [Harnad, 1991] Harnad, S. (1991). Other bodies, other minds: A machine incarnation of an old philosophical problem. *Minds and Machines*, 1(1):43–54.
- [Harnad, 1994] Harnad, S. (1994). Levels of functional equivalence in reverse bioengineering. *Artificial life*, 1(3):293–301.
- [Harnad, 1999] Harnad, S. (1999). Turing on reverse-engineering the mind. *Journal of Logic, Language, and Information*.
- [Harris, 2008] Harris, K. D. (2008). Stability of the fittest: organizing learning through retroaxonal signals. *Trends in Neurosciences*, 31(3):130–136.
- [Hawley et al., 2004] Hawley, M. L., Litovsky, R. Y., and Culling, J. F. (2004). The benefit of binaural hearing in a cocktail party: Effect of location and type of interferer. *The Journal of the Acoustical Society of America*, 115(2):833–843.
- [Hayes and Ford, 1995] Hayes, P. and Ford, K. (1995). Turing test considered harmful. In *IJCAI (1)*, pages 972–977. Citeseer.
- [Hayes et al., ] Hayes, T., Kafle, K., Shrestha, R., Acharya, M., and Kanan, C. Remind your neural network to prevent catastrophic forgetting. *European Conference on Computer Vision*.
- [Haykin and Chen, 2005] Haykin, S. and Chen, Z. (2005). The cocktail party problem. *Neural computation*, 17:1875–902.
- [Hazan et al., 2018] Hazan, H., Saunders, D. J., Sanghavi, D. T., Siegelmann, H. T., and Kozma, R. (2018). Unsupervised learning with self-organizing spiking neural networks. *CoRR*, abs/1807.09374.
- [He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.
- [He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.
- [He et al., 2016a] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE.
- [He et al., 2016b] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

- [Hebb, 1949] Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. Wiley, New York.
- [Herculano-Houzel, 2012] Herculano-Houzel, S. (2012). The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. *Proceedings of the National Academy of Sciences of the United States of America*, 109 Suppl 1:10661–8.
- [Hilgetag and Goulas, 2020] Hilgetag, C. C. and Goulas, A. (2020). ‘hierarchy’ in the organization of brain networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 375(1796):20190319.
- [Hinton and McClelland, 1987] Hinton, G. E. and McClelland, J. L. (1987). Learning representations by recirculation. In *Proceedings of the 1987 International Conference on Neural Information Processing Systems*, NIPS’87, page 358–366, Cambridge, MA, USA. MIT Press.
- [Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. Preprint at <https://arxiv.org/abs/1207.0580>.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.
- [Hodgkin and Huxley, 1952] Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544.
- [Hoeffler et al., 2021] Hoeffler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks.
- [Hu et al., 2018] Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.
- [Huang et al., 2017a] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017a). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.
- [Huang et al., 2017b] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017b). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.
- [Hubel and Wiesel, 1959] Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat’s striate cortex. *The Journal of Physiology*, 148(3):574–591.
- [Hyvärinen and Oja, 1997] Hyvärinen, A. and Oja, E. (1997). A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492.
- [Ibata et al., 2008] Ibata, K., Sun, Q., and Turrigiano, G. (2008). Rapid synaptic scaling induced by changes in postsynaptic firing. *Neuron*, 57:819–26.
- [Illing et al., 2019] Illing, B., Gerstner, W., and Brea, J. (2019). Biologically plausible deep learning — but how far can we go with shallow networks? *Neural Networks*, 118:90–101.
- [Illing et al., 2021] Illing, B., Ventura, J., Bellec, G., and Gerstner, W. (2021). Local plasticity rules can learn deep representations using self-supervised contrastive predictions.
- [Indiveri and Horiuchi, 2011] Indiveri, G. and Horiuchi, T. K. (2011). *Frontiers in neuromorphic engineering*.

- [Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.
- [Isomura and Toyoizumi, 2019] Isomura, T. and Toyoizumi, T. (2019). Multi-context blind source separation by error-gated hebbian rule. *Scientific Reports*, 9:7127.
- [Iyer et al., 2013] Iyer, R., Menon, V., Buice, M., Koch, C., and Mihalas, S. (2013). The influence of synaptic weight distribution on neuronal population dynamics. *PLoS Computational Biology*, 9(10):1–16.
- [Jacobsen et al., 2005] Jacobsen, T., Schröger, E., Winkler, I., and Horváth, J. (2005). Familiarity Affects the Processing of Task-irrelevant Auditory Deviance. *Journal of Cognitive Neuroscience*, 17(11):1704–1713.
- [Jaderberg et al., 2016] Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. (2016). Decoupled neural interfaces using synthetic gradients.
- [Jaderberg et al., 2017] Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. (2017). Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17*, page 1627–1635. JMLR.org.
- [Jaiswal et al., 2020] Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. (2020). A survey on contrastive self-supervised learning.
- [Jalilian et al., 2017] Jalilian, E., Uhl, A., and Kwitt, R. (2017). Domain adaptation for cnn based iris segmentation. In *2017 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–6.
- [Jefferson, 1949] Jefferson, G. (1949). The mind of mechanical man\*. *British Medical Journal*, 1:1105 – 1110.
- [Jiang et al., 2015] Jiang, M., Huang, S., Duan, J., and Zhao, Q. (2015). Salicon: Saliency in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1072–1080.
- [Jumper et al., 2021] Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zidek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohli, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D. A., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589.
- [Kaack et al., 2022] Kaack, L., Donti, P., Strubell, E., Kamiya, G., Creutzig, F., and Rolnick, D. (2022). Aligning artificial intelligence with climate change mitigation. *Nature Climate Change*, 12.
- [Kameoka et al., 2018] Kameoka, H., Li, L., Inoue, S., and Makino, S. (2018). Semi-blind source separation with multichannel variational autoencoder. *ArXiv*, abs/1808.00892.
- [Kamperis, 2019] Kamperis, S. (2019). Energy considerations for training deep neural networks.
- [Kandel, 2000] Kandel, E. R. (2000). *Principles of Neural Science*. McGraw-Hill Education.
- [Karamatli et al., 2018] Karamatli, E., Cemgil, A. T., and Kirbiz, S. (2018). Weak label supervision for monaural source separation using non-negative denoising variational autoencoders. *CoRR*, abs/1810.13104.

- [Karpinska et al., 2021] Karpinska, M., Akoury, N., and Iyyer, M. (2021). The perils of using mechanical turk to evaluate open-ended text generation. *arXiv preprint arXiv:2109.06835*.
- [Kasai et al., 2021] Kasai, J., Sakaguchi, K., Dunagan, L., Morrison, J., Bras, R. L., Choi, Y., and Smith, N. A. (2021). Transparent human evaluation for image captioning. *arXiv preprint arXiv:2111.08940*.
- [Kaski and Kohonen, 1994] Kaski, S. and Kohonen, T. (1994). Winner-take-all networks for physiological models of competitive learning. *Neural Networks*, 7(6):973–984. *Models of Neurodynamics and Behavior*.
- [Kell et al., 2018] Kell, A., Yamins, D., Shook, E., Norman-Haignere, S., and McDermott, J. (2018). A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy. *Neuron*, 98.
- [Kemker et al., 2018] Kemker, R., McClure, M., Abitino, A., Hayes, T., and Kanan, C. (2018). Measuring catastrophic forgetting in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- [Kerlin et al., 2010] Kerlin, J., Shahin, A., and Miller, L. (2010). Attentional gain control of ongoing cortical speech representations in a "cocktail party". *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30:620–8.
- [Khacef et al., 2022] Khacef, L., Klein, P., Cartiglia, M., Rubino, A., Indiveri, G., and Chicca, E. (2022). Spike-based local synaptic plasticity: A survey of computational models and neuromorphic circuits.
- [Khaddam-Aljameh et al., 2021] Khaddam-Aljameh, R., Stanisavljevic, M., Mas, J., Karunaratne, G., Braendli, M., Liu, F., Singh, A., Muller, S., Egger, U., Petropoulos, A., Antonakopoulos, T., Brew, K., Choi, S., Ok, I., Lie, F., Saulnier, N., Chan, V., Ahsan, I., Narayanan, V., and Eleftheriou, E. (2021). Hermes core – a 14nm cmos and pcm-based in-memory compute core using an array of 300ps/lsb linearized cco-based adcs and local digital processing. pages 1–2.
- [Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.
- [Kirkpatrick et al., 2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.
- [Klindt et al., 2017] Klindt, D., Ecker, A. S., Euler, T., and Bethge, M. (2017). Neural system identification for large populations separating “what” and “where”. In *Advances in Neural Information Processing Systems*, pages 3506–3516.
- [Kocsis and Szepesvari, 2006] Kocsis, L. and Szepesvari, C. (2006). Bandit based monte-carlo planning. In *ECML*.
- [Koenderink et al., 2017] Koenderink, J., Valsecchi, M., van Doorn, A., Wagemans, J., and Gegenfurtner, K. (2017). Eidolons: Novel stimuli for vision research. *Journal of Vision*, 17(2):7–7.
- [Konkle et al., 2010] Konkle, T., Brady, T. F., Alvarez, G. A., and Oliva, A. (2010). Conceptual distinctiveness supports detailed visual long-term memory for real-world objects. *Journal of experimental Psychology: general*, 139(3):558.
- [Kording and König, 2001] Kording, K. and König, P. (2001). Supervised and unsupervised learning with two sites of synaptic integration. *Journal of computational neuroscience*, 11:207–15.

- [Krause-Solberg and Iske, 2015] Krause-Solberg, S. and Iske, A. (2015). Non-negative dimensionality reduction for audio signal separation by nmmf and ica. *2015 International Conference on Sampling Theory and Applications, SampTA 2015*, pages 377–381.
- [Kreiman, 2021a] Kreiman, G. (2021a). *Biological and Computer Vision*. Cambridge University Press, Cambridge, UK.
- [Kreiman, 2021b] Kreiman, G. (2021b). *Biological and Computer Vision*. Cambridge University Press.
- [Krishnan et al., 2014] Krishnan, L., Elhilali, M., and Shamma, S. (2014). Segregating complex sound sources through temporal coherence. *PLOS Computational Biology*, 10(12):1–10.
- [Krizhevsky et al., a] Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research).
- [Krizhevsky et al., b] Krizhevsky, A., Nair, V., and Hinton, G. Cifar-100 (canadian institute for advanced research).
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.
- [Kubilius et al., 2019] Kubilius, J., Schrimpf, M., Hong, H., Majaj, N. J., Rajalingham, R., Issa, E. B., Kar, K., Bashivan, P., Prescott-Roy, J., Schmidt, K., Nayebi, A., Bear, D., Yamins, D. L. K., and DiCarlo, J. J. (2019). Brain-Like Object Recognition with High-Performing Shallow Recurrent ANNs. In Wallach, H., Larochelle, H., Beygelzimer, A., D’Alché-Buc, F., Fox, E., and Garnett, R., editors, *Neural Information Processing Systems (NeurIPS)*, pages 12785—12796. Curran Associates, Inc.
- [Kümmerer et al., 2022] Kümmerer, M., Bethge, M., and Wallis, T. S. (2022). Deepgaze iii: Modeling free-viewing human scanpaths with deep learning. *Journal of Vision*, 22(5):7–7.
- [Kwag and Paulsen, 2009] Kwag, J. and Paulsen, O. (2009). The timing of external input controls the sign of plasticity at local synapses. *Nature neuroscience*, 12:1219–21.
- [Laborieux et al., 2021] Laborieux, A., Ernoult, M., Scellier, B., Bengio, Y., Grollier, J., and Querlioz, D. (2021). Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in Neuroscience*, 15.
- [LaCurts, 2011] LaCurts, K. (2011). Criticisms of the turing test and why you should ignore (most of) them. *Official blog of MIT’s course: Philosophy and theoretical computer science*.
- [Lansdell et al., 2020] Lansdell, B. J., Prakash, P. R., and Kording, K. P. (2020). Learning to solve the credit assignment problem. In *International Conference on Learning Representations*.
- [Larkum, 2013] Larkum, M. (2013). A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. *Trends in Neurosciences*, 36(3):141–151.
- [Larkum et al., 1999] Larkum, M., Zhu, J., and Sakmann, B. (1999). A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398:338–41.
- [Launay et al., 2020a] Launay, J., Poli, I., Boniface, F., and Krzakala, F. (2020a). Direct feedback alignment scales to modern deep learning tasks and architectures. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*, Red Hook, NY, USA. Curran Associates Inc.
- [Launay et al., 2019] Launay, J., Poli, I., and Krzakala, F. (2019). Principled training of neural networks with direct feedback alignment.

- [Launay et al., 2020b] Launay, J., Poli, I., Müller, K., Carron, I., Daudet, L., Krzakala, F., and Gigan, S. (2020b). Light-in-the-loop: using a photonics co-processor for scalable training of neural networks.
- [le Cun, 1986] le Cun, Y. (1986). Learning process in an asymmetric threshold network.
- [LeCun, 2012] LeCun, Y. (2012). Learning invariant feature hierarchies. In *ECCV Workshops*.
- [LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.
- [Lecun et al., 1998a] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [Lecun et al., 1998b] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998b). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324.
- [LeCun and Cortes, 2010] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.
- [Lecun and Soulie Fogelman, 1987] Lecun, Y. and Soulie Fogelman, F. (1987). Modeles connexionnistes de l'apprentissage. *Intellectica, special issue apprentissage et machine*, 2.
- [Lee and Wilson, 2003] Lee, A. and Wilson, M. (2003). Memory of sequential experience in the hippocampus during slow wave sleep. *Neuron*, 36:1183–94.
- [Lee et al., 2015] Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. pages 498–515.
- [Lee et al., 2017] Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. (2017). Overcoming catastrophic forgetting by incremental moment matching.
- [Legenstein et al., 2010] Legenstein, R., Chase, S. M., Schwartz, A. B., and Maass, W. (2010). A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *Journal of Neuroscience*, 30(25):8400–8410.
- [Lewald and Getzmann, 2015] Lewald, J. and Getzmann, S. (2015). Electrophysiological correlates of cocktail-party listening. *Behavioural brain research*, 292:157–166.
- [Lewis and Durrant, 2011] Lewis, P. A. and Durrant, S. J. (2011). Overlapping memory replay during sleep builds cognitive schemata. *Trends in Cognitive Sciences*, 15(8):343–351.
- [Lewis and Harris, 2014] Lewis, S. N. and Harris, K. D. (2014). The neural marketplace: I. general formalism and linear theory. *bioRxiv*.
- [Li et al., 2022] Li, J., Li, D., Xiong, C., and Hoi, S. (2022). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*.
- [Li et al., 2020] Li, X., Luo, S., and Xue, F. (2020). Effects of synaptic integration on the dynamics and computational performance of spiking neural network. *Cognitive Neurodynamics*, 14(3):347–357.
- [Li et al., 2017] Li, Y., Wang, F., Chen, Y., Cichocki, A., and Sejnowski, T. (2017). The Effects of Audiovisual Inputs on Solving the Cocktail Party Problem in the Human Brain: An fMRI Study. *Cerebral Cortex*, 28(10):3623–3637.
- [Li and Hoiem, 2016] Li, Z. and Hoiem, D. (2016). Learning without forgetting.
- [Liao et al., 2016] Liao, Q., Leibo, J. Z., and Poggio, T. (2016). How important is weight symmetry in backpropagation?
- [Lillicrap et al., 2020] Lillicrap, T., Santoro, A., Marris, L., Akerman, C., and Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, 21.

- [Lillicrap et al., 2016] Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276.
- [Lillicrap and Scott, 2013] Lillicrap, T. P. and Scott, S. H. (2013). Preference distributions of primary motor cortex neurons reflect control solutions optimized for limb biomechanics. *Neuron*, 77(1):168–179.
- [Lin, 2021] Lin, B. (2021). Regularity normalization: Neuroscience-inspired unsupervised attention across neural network layers. *Entropy*, 24:59.
- [Lin et al., 2021] Lin, T., Wang, Y., Liu, X., and Qiu, X. (2021). A survey of transformers. *CoRR*, abs/2106.04554.
- [Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.
- [Lindsay, 2021] Lindsay, G. W. (2021). Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *Journal of Cognitive Neuroscience*, 33(10):2017–2031.
- [Liu et al., 2021] Liu, Q., Huang, Y., Hao, Y., Xu, J., and Xu, B. (2021). Limuse: Lightweight multi-modal speaker extraction. In *arXiv:2111.04063*.
- [Lomonaco et al., 2021] Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T. L., De Lange, M., Masana, M., Pomponi, J., van de Ven, G. M., Mundt, M., She, Q., Cooper, K., Forest, J., Belouadah, E., Calderara, S., Parisi, G. I., Cuzzolin, F., Tolia, A. S., Scardapane, S., Antiga, L., Ahmad, S., Popescu, A., Kanan, C., van de Weijer, J., Tuytelaars, T., Bacciu, D., and Maltoni, D. (2021). Avalanche: An end-to-end library for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3600–3610.
- [Lopez-Paz and Ranzato, 2017] Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning.
- [López-Serrano et al., 2019] López-Serrano, P., Dittmar, C., Özer, Y., and Müller, M. (2019). Nmf toolbox: Music processing applications of nonnegative matrix factorization.
- [Louie et al., 2013] Louie, K., Khaw, M. W., and Glimcher, P. W. (2013). Normalization is a general neural mechanism for context-dependent decision making. *Proceedings of the National Academy of Sciences*, 110(15):6139–6144.
- [M., 2019] M., W. (2019). cifar10-resnet. <https://github.com/matthias-wright/cifar10-resnet>.
- [Madan et al., 2020] Madan, S., Henry, T., Dozier, J., Ho, H., Bhandari, N., Sasaki, T., Durand, F., Pfister, H., and Boix, X. (2020). On the capability of neural networks to generalize to unseen category-pose combinations. *CoRR*, abs/2007.08032.
- [Madan et al., 2022] Madan, S., You, L., Zhang, M., Pfister, H., and Kreiman, G. (2022). What makes domain generalization hard?
- [Marblestone et al., 2016] Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10.
- [Markram et al., 1995] Markram, H., Helm, P. J., and Sakmann, B. J. (1995). Dendritic calcium transients evoked by single back-propagating action potentials in rat neocortical pyramidal neurons. *The Journal of Physiology*, 485.

- [Markram et al., 1997] Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic apss and epsps. *Science*, 275(5297):213–215.
- [Maruf, ] Maruf, R. Google fires engineer who contended its ai technology was sentient. *CNN*.
- [Mateos-Aparicio and Rodríguez-Moreno, 2019] Mateos-Aparicio, P. and Rodríguez-Moreno, A. (2019). The impact of studying brain plasticity. *Frontiers in Cellular Neuroscience*, 13.
- [Mathew et al., 2021] Mathew, M., Karatzas, D., and Jawahar, C. (2021). Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.
- [Mauldin, 1994] Mauldin, M. L. (1994). Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In *AAAI*, volume 94, pages 16–21.
- [Mazzoni et al., 1991] Mazzoni, P., Andersen, R. A., and Jordan, M. I. (1991). A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437.
- [McClelland et al., 1995] McClelland, J., Mcnaughton, B., and O’Reilly, R. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102:419–57.
- [McCulloch and Pitts, 1943a] McCulloch, W. S. and Pitts, W. (1943a). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.
- [McCulloch and Pitts, 1943b] McCulloch, W. S. and Pitts, W. (1943b). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.
- [McDermott, 2009] McDermott, J. H. (2009). The cocktail party problem. *Current Biology*, 19(22):R1024–R1027.
- [McDermott et al., 2011] McDermott, J. H., Wroblewski, D., and Oxenham, A. J. (2011). Recovering sound sources from embedded repetition. *Proceedings of the National Academy of Sciences*, 108(3):1188–1193.
- [McFee et al., 2015] McFee, B., Raffel, C., Liang, D., Ellis, D., Mcvicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. pages 18–24.
- [McInnes et al., 2018] McInnes, L., Healy, J., and Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.
- [Mermillod et al., 2013] Mermillod, M., Bugajska, A., and Bonin, P. (2013). The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504.
- [Mesgarani and Chang, 2012] Mesgarani, N. and Chang, E. (2012). Selective cortical representation of attended speaker in multi-talker speech perception. *Nature*, 485:233–6.
- [Meulemans et al., 2020] Meulemans, A., Carzaniga, F. S., Suykens, J. A., Sacramento, J. a., and Grewe, B. F. (2020). A theoretical framework for target propagation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*, Red Hook, NY, USA. Curran Associates Inc.
- [Meulemans et al., 2021] Meulemans, A., Tristany Farinha, M., Garcia Ordonez, J., Vilimelis Aceituno, P., Sacramento, J. a., and Grewe, B. F. (2021). Credit assignment in neural networks through deep feedback control. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4674–4687. Curran Associates, Inc.



- [Meulemans et al., 2022] Meulemans, A., Zucchet, N., Kobayashi, S., von Oswald, J., and Sacramento, J. (2022). The least-control principle for learning at equilibrium.
- [Michaelis and Menten, 1913] Michaelis, L. and Menten, M. (1913). Kinetik der invertinwirkung. *Biochem. Zeitung*, 49:333–369.
- [Middlebrooks and Waters, 2020] Middlebrooks, J. C. and Waters, M. F. (2020). Spatial mechanisms for segregation of competing sounds, and a breakdown in spatial hearing. *Frontiers in Neuroscience*, 14:966.
- [Mika et al., 2020] Mika, D., Budzik, G., and Józwik, J. (2020). Ica-based single channel source separation with time-frequency decomposition. In *2020 IEEE 7th International Workshop on Metrology for Aerospace (MetroAeroSpace)*, pages 238–243.
- [Moeller et al., 2017] Moeller, S., Crapse, T., Chang, L., and Tsao, D. Y. (2017). The effect of face patch microstimulation on perception of faces and objects. *Nature neuroscience*, 20(5):743–752.
- [Moeller et al., 2008] Moeller, S., Freiwald, W. A., and Tsao, D. Y. (2008). Patches with links: a unified system for processing faces in the macaque temporal lobe. *Science*, 320(5881):1355–1359.
- [Mokady et al., 2021] Mokady, R., Hertz, A., and Bermano, A. H. (2021). Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*.
- [Moradi et al., 2018] Moradi, S., Qiao, N., Stefanini, F., and Indiveri, G. (2018). A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE Transactions on Biomedical Circuits and Systems*, 12(1):106–122.
- [Mordvintsev et al., 2015] Mordvintsev, A., Olah, C., and Tyka, M. (2015). Inceptionism: Going deeper into neural networks.
- [Mostafa et al., 2018] Mostafa, H., Ramesh, V., and Cauwenberghs, G. (2018). Deep supervised learning using local errors. *Frontiers in Neuroscience*, 12.
- [Moulin et al., 2020] Moulin, T. C., Rayêe, D., Williams, M. J., and Schiöth, H. B. (2020). The synaptic scaling literature: A systematic review of methodologies and quality of reporting. *Frontiers in Cellular Neuroscience*, 14:164.
- [Muñoz-Martín et al., 2019] Muñoz-Martín, I., Bianchi, S., Pedretti, G., Melnic, O., Ambrogio, S., and Ielmini, D. (2019). Unsupervised learning to overcome catastrophic forgetting in neural networks. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 5(1):58–66.
- [Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML’10*, page 807–814, Madison, WI, USA. Omnipress.
- [Nandakumar et al., 2018] Nandakumar, S. R., Le Gallo, M., Boybat, I., Rajendran, B., Sebastian, A., and Eleftheriou, E. (2018). Mixed-precision architecture based on computational memory for training deep neural networks. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5.
- [Narayan et al., 2008] Narayan, R., Best, V., Ozmeral, E., McClaine, E., Dent, M., Shinn-Cunningham, B., and Sen, K. (2008). Cortical interference effects in the cocktail party problem. *Nature neuroscience*, 10:1601–7.
- [Nayebi et al., 2021] Nayebi, A., Attinger, A., Campbell, M., Hardcastle, K., Low, I., Mallory, C. S., Mel, G., Sorscher, B., Williams, A. H., Ganguli, S., Giocomo, L., and Yamins, D. (2021). Explaining heterogeneity in medial entorhinal cortex with task-driven neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12167–12179. Curran Associates, Inc.

- [Neelakantan et al., 2022] Neelakantan, A., Xu, T., Puri, R., Radford, A., Han, J. M., Tworek, J., Yuan, Q., Tezak, N., Kim, J. W., Hallacy, C., et al. (2022). Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.
- [Neftci et al., 2017] Neftci, E., Augustine, C., Paul, S., and Detorakis, G. (2017). Event-driven random backpropagation: Enabling neuromorphic deep learning machines. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4.
- [Nesterov, 1983] Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate  $o(1/k^2)$ . *Proceedings of the USSR Academy of Sciences*, 269:543–547.
- [Netzer et al., 2011] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
- [Nguyen et al., 2019] Nguyen, C. V., Achille, A., Lam, M., Hassner, T., Mahadevan, V., and Soatto, S. (2019). Toward understanding catastrophic forgetting in continual learning.
- [Nguyen et al., 2017] Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2017). Variational continual learning.
- [Nokland, 2016] Nokland, A. (2016). Direct feedback alignment provides learning in deep neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS’16, page 1045–1053, Red Hook, NY, USA. Curran Associates Inc.
- [Nøkland and Eidnes, 2019] Nøkland, A. and Eidnes, L. H. (2019). Training neural networks with local error signals. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4839–4850.
- [Oja, 1982] Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273.
- [O’Keefe and Dostrovsky, 1971] O’Keefe, J. and Dostrovsky, J. O. (1971). The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 34 1:171–5.
- [Oore et al., 2017] Oore, S., Simon, I., Dieleman, S., and Eck, D. (2017). Learning to create piano performances.
- [Oore et al., 2020] Oore, S., Simon, I., Dieleman, S., Eck, D., and Simonyan, K. (2020). This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, 32.
- [O’Reilly, 1996] O’Reilly, R. C. (1996). Biologically Plausible Error-Driven Learning Using Local Activation Differences: The Generalized Recirculation Algorithm. *Neural Computation*, 8(5):895–938.
- [O’Sullivan et al., 2014] O’Sullivan, J., Power, A., Mesgarani, N., Rajaram, S., Foxe, J., Shinn-Cunningham, B., Slaney, M., Shamma, S., and Lalor, E. (2014). Attentional selection in a cocktail party environment can be decoded from single-trial eeg. *Cerebral cortex (New York, N.Y. : 1991)*, 25.
- [Ouyang et al., 2022] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback.
- [Oxenham, 2018] Oxenham, A. J. (2018). How we hear: The perception and neural coding of sound. *Annual Review of Psychology*, 69(1):27–50. PMID: 29035691.

- [O'Neill et al., 2010] O'Neill, J., Pleydell-Bouverie, B., Dupret, D., and Csicsvari, J. (2010). Play it again: reactivation of waking experience and memory. *Trends in Neurosciences*, 33(5):220–229.
- [O'Reilly, 2006] O'Reilly, R. (2006). Biologically based computational models of high-level cognition. *Science (New York, N.Y.)*, 314:91–4.
- [O'Reilly and Norman, 2002] O'Reilly, R. C. and Norman, K. A. (2002). Hippocampal and neocortical contributions to memory: advances in the complementary learning systems framework. *Trends in Cognitive Sciences*, 6:505–510.
- [Pallier et al., 2003] Pallier, C., Dehaene, S., Poline, J.-B., LeBihan, D., Argenti, A.-M., Dupoux, E., and Mehler, J. (2003). Brain Imaging of Language Plasticity in Adopted Adults: Can a Second Language Replace the First? *Cerebral Cortex*, 13(2):155–161.
- [Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.
- [Parisi et al., 2018] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2018). Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569.
- [Parvizi et al., 2012] Parvizi, J., Jacques, C., Foster, B. L., Withoft, N., Rangarajan, V., Weiner, K. S., and Grill-Spector, K. (2012). Electrical stimulation of human fusiform face-selective regions distorts face perception. *Journal of Neuroscience*, 32(43):14915–14920.
- [Pascanu et al., 2013] Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2013). How to construct deep recurrent neural networks.
- [Pascanu et al., 2012] Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.
- [Patel et al., 2015] Patel, V., Gopalan, R., Li, R., and Chellappa, R. (2015). Visual domain adaptation: A survey of recent advances. *Signal Processing Magazine, IEEE*, 32:53–69.
- [Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.
- [Pehlevan and Chklovskii, 2019] Pehlevan, C. and Chklovskii, D. B. (2019). Neuroscience-inspired online unsupervised learning algorithms: Artificial neural networks. *IEEE Signal Processing Magazine*, 36(6):88–96.
- [Pehlevan et al., 2017] Pehlevan, C., Mohan, S., and Chklovskii, D. B. (2017). Blind Nonnegative Source Separation Using Biological Neural Networks. *Neural Computation*, 29(11):2925–2954.
- [Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- [Pfeiffer et al., 2010] Pfeiffer, M., Nessler, B., Douglas, R. J., and Maass, W. (2010). Reward-Modulated Hebbian Learning of Decision Making. *Neural Computation*, 22(6):1399–1444.
- [Pfeiffer and Pfeil, 2018] Pfeiffer, M. and Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12:774.
- [Pfeil et al., 2012] Pfeil, T., Potjans, T., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., and Meier, K. (2012). Is a 4-bit synaptic weight resolution enough? – constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Frontiers in Neuroscience*, 6:90.
- [Pfister and Gerstner, 2006] Pfister, J.-P. and Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, 26(38):9673–9682.

- [Phuong and Hutter, 2022] Phuong, M. and Hutter, M. (2022). Formal algorithms for transformers.
- [Polydoros and Nalpantidis, 2017] Polydoros, A. and Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86:153–.
- [Ponce et al., 2019] Ponce, C. R., Xiao, W., Schade, P. F., Hartmann, T. S., Kreiman, G., and Livingstone, M. S. (2019). Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences. *Cell*, 177(4):999–1009.e10.
- [Popham et al., 2018] Popham, S., Boebinger, D., Ellis, D., Kawahara, H., and McDermott, J. (2018). Inharmonic speech reveals the role of harmonicity in the cocktail party problem. *Nature Communications*, 9.
- [Purtil, 1971] Purtil, R. L. (1971). Beating the imitation game. *Mind*, 80(318):290–294.
- [Radford et al., 2021] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.
- [Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- [Rajalingham and DiCarlo, 2019] Rajalingham, R. and DiCarlo, J. J. (2019). Reversible inactivation of different millimeter-scale regions of primate it results in different patterns of core object recognition deficits. *Neuron*, 102(2):493–505.
- [Ramesh et al., 2022] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents.
- [Ramesh et al., 2021] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR.
- [Rao and Ballard, 1999] Rao, R. and Ballard, D. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2:79–87.
- [Ratcliff, 1990] Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97 2:285–308.
- [Räukur et al., 2022] Räukur, T., Ho, A., Casper, S., and Hadfield-Menell, D. (2022). Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. *arXiv preprint arXiv:2207.13243*.
- [Reed, 1993] Reed, R. (1993). Pruning algorithms—a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747.
- [Refinetti et al., 2021] Refinetti, M., D’Ascoli, S., Ohana, R., and Goldt, S. (2021). Align, then memorise: the dynamics of learning with feedback alignment. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8925–8935. PMLR.
- [Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

- [Rich et al., 2014] Rich, P., Liaw, H.-P., and Lee, A. (2014). Place cells. large environments reveal the statistical structure governing hippocampal representations. *Science (New York, N.Y.)*, 345:814–7.
- [Richards et al., 2019] Richards, B., Lillicrap, T., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R., Berker, A., Ganguli, S., Gillon, C., Hafner, D., Kepecs, A., Kriegeskorte, N., Latham, P., Lindsay, G., Miller, K., Naud, R., Pack, C., and Kording, K. (2019). A deep learning framework for neuroscience. *Nature Neuroscience*, 22:1761–1770.
- [Ritter et al., 2018] Ritter, H., Botev, A., and Barber, D. (2018). Online structured laplace approximations for overcoming catastrophic forgetting.
- [Rives et al., 2021] Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118.
- [Robins, 1995] Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.
- [Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408.
- [Royer and Paré, 2003a] Royer, S. and Paré, D. (2003a). Conservation of total synaptic weight through balanced synaptic depression and potentiation. *Nature*, 422:518–22.
- [Royer and Paré, 2003b] Royer, S. and Paré, D. (2003b). Conservation of total synaptic weight through balanced synaptic depression and potentiation. *Nature*, 422:518–22.
- [Rumelhart et al., 1995] Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). *Back-propagation: The Basic Theory*, page 1–34. L. Erlbaum Associates Inc., USA.
- [Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- [Sacramento et al., 2018a] Sacramento, J. a., Ponte Costa, R., Bengio, Y., and Senn, W. (2018a). Dendritic cortical microcircuits approximate the backpropagation algorithm. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- [Sacramento et al., 2018b] Sacramento, J. a., Ponte Costa, R., Bengio, Y., and Senn, W. (2018b). Dendritic cortical microcircuits approximate the backpropagation algorithm. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- [Sagi et al., 2001] Sagi, B., Nemat-Nasser, S. C., Kerr, R., Hayek, R., Downing, C., and Hecht-Nielsen, R. (2001). A Biologically Motivated Solution to the Cocktail Party Problem. *Neural Computation*, 13(7):1575–1602.
- [Salimans and Kingma, 2016] Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.
- [Salzman et al., 1990] Salzman, C. D., Britten, K. H., and Newsome, W. T. (1990). Cortical microstimulation influences perceptual judgements of motion direction. *Nature*, 346(6280):174–177.

- [Santosh and Bharathi, 2017] Santosh, K. S. and Bharathi, S. H. (2017). Non-negative matrix factorization algorithms for blind source separation in speech recognition. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 2242–2246.
- [Savage, ] Savage, N. Marriage of mind and machine.
- [Sawada et al., 2019] Sawada, H., Ono, N., Kameoka, H., Kitamura, D., and Saruwatari, H. (2019). A review of blind source separation methods: Two converging routes to IIRMA originating from ICA and NMF. *APSIPA Transactions on Signal and Information Processing*, 8.
- [Saxe et al., 2020] Saxe, A. M., Nelli, S., and Summerfield, C. (2020). If deep learning is the answer, then what is the question? *arXiv: Neurons and Cognition*.
- [Scellier and Bengio, 2017] Scellier, B. and Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11.
- [Schaeffer et al., 2022] Schaeffer, R., Khona, M., and Fiete, I. R. (2022). No free lunch from deep learning in neuroscience: A case study through models of the entorhinal-hippocampal circuit. In *ICML 2022 2nd AI for Science Workshop*.
- [Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.
- [Schmidt and Römer, 2011] Schmidt, A. K. D. and Römer, H. (2011). Solutions to the cocktail party problem in insects: Selective filters, spatial release from masking and gain control in tropical crickets. *PLOS ONE*, 6(12):1–9.
- [Schrimpf et al., 2018] Schrimpf, M., Kubilius, J., Hong, H., Majaj, N. J., Rajalingham, R., Issa, E. B., Kar, K., Bashivan, P., Prescott-Roy, J., Schmidt, K., et al. (2018). Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv*, page 407007.
- [Schrimpf et al., 2020] Schrimpf, M., Kubilius, J., Lee, M. J., Murty, N. A. R., Ajemian, R., and DiCarlo, J. J. (2020). Integrative benchmarking to advance neurally mechanistic models of human intelligence. *Neuron*.
- [Schwaller et al., 2019] Schwaller, P., Laino, T., Gaudin, T., Bolgar, P., Hunter, C., Bekas, C., and Lee, A. (2019). Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS Central Science*, 5.
- [Searle, 1980] Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and brain sciences*, 3(3):417–424.
- [Sejnowski, 2022] Sejnowski, T. (2022). Large language models and the reverse Turing test. *arXiv preprint arXiv:2207.14382*.
- [Serre, 2019] Serre, T. (2019). Deep learning: the good, the bad, and the ugly. *Annual Review of Vision Science*, 5:399–426.
- [Sharma et al., 2000] Sharma, J., Angelucci, A., and Sur, M. (2000). Induction of visual orientation modules in auditory cortex. *Nature*, 404:841–7.
- [Shen et al., 2020] Shen, Y., Wang, J., and Navlakha, S. (2020). A correspondence between normalization strategies in artificial and biological neural networks.
- [Shen et al., 2021] Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. (2021). Towards out-of-distribution generalization: A survey.

- [Sheng et al., 2021] Sheng, S., Singh, A., Goswami, V., Magana, J., Thrush, T., Galuba, W., Parikh, D., and Kiela, D. (2021). Human-adversarial visual question answering. *Advances in Neural Information Processing Systems*, 34:20346–20359.
- [Shimegi et al., 2016] Shimegi, S., Kimura, A., Sato, A. Y., Aoyama, C., Mizuyama, R., Tsunoda, K., Ueda, F., Araki, S., Goya, R., and Sato, H. (2016). Cholinergic and serotonergic modulation of visual information processing in monkey v1. *Journal of Physiology-Paris*, 110:44–51.
- [Shin et al., 2017] Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- [Shuster et al., 2022a] Shuster, K., Xu, J., Komeili, M., Ju, D., Smith, E. M., Roller, S., Ung, M., Chen, M., Arora, K., Lane, J., Behrooz, M., Ngan, W., Poff, S., Goyal, N., Szlam, A., Boureau, Y.-L., Kambadur, M., and Weston, J. (2022a). Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage.
- [Shuster et al., 2022b] Shuster, K., Xu, J., Komeili, M., Ju, D., Smith, E. M., Roller, S., Ung, M., Chen, M., Arora, K., Lane, J., et al. (2022b). Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*.
- [Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489.
- [Sinz et al., 2019] Sinz, F. H., Pitkow, X., Reimer, J., Bethge, M., and Tolias, A. S. (2019). Engineering a less artificial intelligence. *Neuron*, 103(6):967–979.
- [Sjostrom and Hausser, 2006] Sjostrom, P. and Hausser, M. (2006). A cooperative switch determines the sign of synaptic plasticity in distal dendrites of neocortical pyramidal neurons. *Neuron*, 51:227–38.
- [Skaggs and McNaughton, 1996] Skaggs, W. E. and McNaughton, B. L. (1996). Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science*, 271(5257):1870–1873.
- [Smaragdis and Brown, 2003] Smaragdis, P. and Brown, J. (2003). Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pages 177–180.
- [Sodhani et al., 2020] Sodhani, S., Chandar, S., and Bengio, Y. (2020). Toward training recurrent neural networks for lifelong learning. *Neural Computation*, 32(1):1–35.
- [Soltoggio and Stanley, 2012] Soltoggio, A. and Stanley, K. O. (2012). From modulated hebbian plasticity to simple behavior learning through noise and weight saturation. *Neural Networks*, 34:28–41.
- [Song et al., 2000] Song, S., Miller, K., and Abbott, L. (2000). Competitive hebbian learning through spike timing-dependent plasticity. *Nature neuroscience*, 3:919–26.
- [Song et al., 2005] Song, S., Sjöström, P. J., Reigl, M., Nelson, S., and Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLOS Biology*, 3(3):e68.

- [Sorscher et al., 2019] Sorscher, B., Mel, G., Ganguli, S., and Ocko, S. (2019). A unified theory for the origin of grid cells through the lens of pattern formation. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- [Speer et al., 2018] Speer, R., Chin, J., Lin, A., Jewett, S., and Nathan, L. (2018). Luminosity/wordfreq: v2.2.
- [Spruston, 2008] Spruston, N. (2008). Pyramidal neurons: dendritic structure and synaptic integration. *Nature reviews. Neuroscience*, 9:206–21.
- [Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). *Journal of Machine Learning Research*, 15(56):1929–1958.
- [Srivastava et al., 2015] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *CoRR*, abs/1505.00387.
- [Srivastava et al., 2013] Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhuber, J. (2013). Compete to compute. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.
- [Steiner et al., 2021] Steiner, A., Kolesnikov, A., , Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. (2021). How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*.
- [Stevens et al., 1937] Stevens, S. S., Volkman, J., and Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.
- [Strubell et al., 2019] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.
- [Stuart and Sakmann, 1994] Stuart, G. J. and Sakmann, B. J. (1994). Active propagation of somatic action potentials into neocortical pyramidal cell dendrites. *Nature*, 367:69–72.
- [Sturm et al., 2016] Sturm, B. L., Santos, J. F., Ben-Tal, O., and Korshunova, I. (2016). Music transcription modelling and composition using deep learning.
- [Szegedy et al., 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.
- [Teki et al., 2013] Teki, S., Chait, M., Kumar, S., Shamma, S., and Griffiths, T. D. (2013). Segregation of complex acoustic scenes based on temporal coherence. *eLife*, 2:e00699.
- [Tenney et al., 2019] Tenney, I., Das, D., and Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. *CoRR*, abs/1905.05950.
- [Teramae and Fukai, 2014] Teramae, J. and Fukai, T. (2014). Computational implications of log-normally distributed synaptic weights. *Proceedings of the IEEE*, 102(4):500–512.
- [Teyler and Rudy, 2007] Teyler, T. J. and Rudy, J. W. (2007). The hippocampal indexing theory and episodic memory: Updating the index. *Hippocampus*, 17(12):1158–1169.
- [Thakur et al., 2015] Thakur, C., Wang, R., Afshar, S., Hamilton, T., Tapson, J., Shamma, S., and van Schaik, A. (2015). Sound stream segregation: a neuromorphic approach to solve the “cocktail party problem” in real-time. *Frontiers in Neuroscience*, 9:309.



- [Thompson et al., 2020] Thompson, N. C., Greenewald, K., Lee, K., and Manso, G. F. (2020). The computational limits of deep learning. Preprint at <https://arxiv.org/abs/2007.05558>.
- [Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.
- [Tiku, ] Tiku, N. The google engineer who thinks the company’s ai has come to life. *Washington-Post*.
- [Toews, 2021] Toews, R. (2021). Alphafold is the most important achievement in ai-ever.
- [Tolstikhin et al., 2021] Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. (2021). Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*.
- [Tramèr et al., 2020] Tramèr, F., Behrmann, J., Carlini, N., Papernot, N., and Jacobsen, J.-H. (2020). Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In *International Conference on Machine Learning*, pages 9561–9571. PMLR.
- [Tsao et al., 2006] Tsao, D. Y., Freiwald, W. A., Tootell, R. B., and Livingstone, M. S. (2006). A cortical region consisting entirely of face-selective cells. *Science*, 311(5761):670–674.
- [Tuma et al., 2016] Tuma, T., Pantazi, A., Gallo, M., Sebastian, A., and Eleftheriou, E. (2016). Stochastic phase-change neurons. *Nature Nanotechnology*, 11.
- [Turing, 2009] Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the turing test*, pages 23–65. Springer.
- [Turrigiano, 2008] Turrigiano, G. (2008). The self-tuning neuron: Synaptic scaling of excitatory synapses. *Cell*, 135:422–35.
- [Turrigiano et al., 1998] Turrigiano, G., Leslie, K., Desai, N., Rutherford, L., and Nelson, S. (1998). Activity-dependent scaling of quantal amplitude in neocortical neurons. *Nature*, 391:892–6.
- [Urbanczik and Senn, 2014] Urbanczik, R. and Senn, W. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528.
- [van de Ven et al., 2020] van de Ven, G., Siegelmann, H., and Tolia, A. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11:4069.
- [van de Ven and Tolia, 2018] van de Ven, G. M. and Tolia, A. S. (2018). Generative replay with feedback connections as a general strategy for continual learning. *arXiv e-prints*, page arXiv:1809.10635.
- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- [Vedantam et al., 2015] Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.
- [Walker et al., 2019] Walker, E. Y., Sinz, F. H., Cobos, E., Muhammad, T., Froudarakis, E., Fahey, P. G., Ecker, A. S., Reimer, J., Pitkow, X., and Tolia, A. S. (2019). Inception loops discover what excites neurons most using deep predictive models. *Nature neuroscience*, 22(12):2060–2065.
- [Wang et al., 2021] Wang, B., Mayo, D., Deza, A., Barbu, A., and Conwell, C. (2021). On the use of cortical magnification and saccades as biological proxies for data augmentation. In *SVRHM 2021 Workshop @ NeurIPS*.

- [Wang et al., 2022a] Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C., and Wang, L. (2022a). Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*.
- [Wang et al., 2022b] Wang, P., Yang, A., Men, R., Lin, J., Bai, S., Li, Z., Ma, J., Zhou, C., Zhou, J., and Yang, H. (2022b). Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, pages 23318–23340. PMLR.
- [Watt, 1996] Watt, S. (1996). Naive psychology and the inverted turing test. *Psychology*, 7(14):463–518.
- [Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.
- [Wertheimer, ] Wertheimer, T. Blake lemoine: Google fires engineer who said ai tech has feelings. *BBC news*.
- [Westerlund, 2019] Westerlund, M. (2019). The emergence of deepfake technology: A review. *Technology Innovation Management Review*, 9(11).
- [White and King, 2003] White, L. and King, S. (2003). The eustace speech corpus. *Centre for Speech Technology Research, University of Edinburgh*.
- [Whittington and Bogacz, 2017] Whittington, J. and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, 29:1–34.
- [Whittington and Bogacz, 2019] Whittington, J. and Bogacz, R. (2019). Theories of error backpropagation in the brain. *Trends in Cognitive Sciences*, 23:235–250.
- [Wickens, 2002] Wickens, T. D. (2002). *Elementary signal detection theory*. Elementary signal detection theory. Oxford University Press, New York, NY, US.
- [Widrow and Hoff, 1960] Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York. IRE.
- [Wiesler et al., 2014] Wiesler, S., Richard, A., Schlüter, R., and Ney, H. (2014). Mean-normalized stochastic gradient for large-scale deep learning. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 180–184.
- [Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.
- [Williams and Atkinson, 2008] Williams, S. and Atkinson, S. (2008). Dendritic synaptic integration in central neurons. *Current biology : CB*, 18:R1045–7.
- [Williams and Stuart, 2001] Williams, S. R. and Stuart, G. J. (2001). *Synaptic Integration*. American Cancer Society.
- [Wilson and McNaughton, 1994] Wilson, M. A. and McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679.
- [Woods and McDermott, 2018] Woods, K. J. P. and McDermott, J. H. (2018). Schema learning for the cocktail party problem. *Proceedings of the National Academy of Sciences*, 115(14):E3313–E3322.
- [Wozniak et al., 2020] Wozniak, S., Pantazi, A., Bohnstingl, T., and Eleftheriou, E. (2020). Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*, 2:325–336.

- [Wu et al., 2019] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. <https://github.com/facebookresearch/detectron2>.
- [Wythoff, 1993] Wythoff, B. J. (1993). Backpropagation neural networks: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 18(2):115–155.
- [Xiang et al., 2010] Xiang, J., Simon, J., and Elhilali, M. (2010). Competing streams at the cocktail party: Exploring the mechanisms of attention and temporal integration. *Journal of Neuroscience*, 30(36):12084–12093.
- [Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Preprint at <https://arxiv.org/abs/1708.07747>.
- [Xiao et al., 2018] Xiao, W., Chen, H., Liao, Q., and Poggio, T. (2018). Biologically-plausible learning algorithms can scale to large datasets.
- [Xie et al., 2020] Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.
- [Xie et al., 2017] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.
- [Xie and Seung, 2003] Xie, X. and Seung, H. S. (2003). Equivalence of Backpropagation and Contrastive Hebbian Learning in a Layered Network. *Neural Computation*, 15(2):441–454.
- [Xu et al., 2019] Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., and Jain, A. K. (2019). Adversarial attacks and defenses in images, graphs and text: A review. *CoRR*, abs/1909.08072.
- [Yala et al., 2019] Yala, A., Lehman, C., Schuster, T., Portnoi, T., and Barzilay, R. (2019). A deep learning mammography-based model for improved breast cancer risk prediction. *Radiology*, 292(1):60–66. PMID: 31063083.
- [Yamins et al., 2014] Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624.
- [Yan et al., 2021] Yan, M., Xu, H., Li, C., Tian, J., Bi, B., Wang, W., Chen, W., Xu, X., Wang, F., Cao, Z., et al. (2021). Achieving human parity on visual question answering. *arXiv preprint arXiv:2111.08896*.
- [Yang and Lisberger, 2014] Yang, Y. and Lisberger, S. G. (2014). Purkinje-cell plasticity and cerebellar motor learning are graded by complex-spike duration. *Nature*, 510(7506):529–532.
- [Yang et al., 2020] Yang, Z., Huang, L., Chen, Y., Wei, Z., Ahn, S., Zelinsky, G., Samaras, D., and Hoai, M. (2020). Predicting goal-directed human attention using inverse reinforcement learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 193–202.
- [Youhui Zhang et al., 2016] Youhui Zhang, Yu Ji, Wenguang Chen, and Yuan Xie (2016). Neural network transformation under hardware constraints. In *2016 International Conference on Compilers, Architectures, and Synthesis of Embedded Systems (CASES)*, pages 1–1.
- [Yu, 2020] Yu, D. (2020). Solving Cocktail Party Problem – From Single Modality to Multi-Modality. In *Proc. 6th International Workshop on Speech Processing in Everyday Environments (CHiME 2020)*.

- [Yuan et al., 2020] Yuan, L., Xiao, W., Dellaferrera, G., Kreiman, G., Tay, F. E. H., Feng, J., and Livingstone, M. S. (2020). Preprint: arXiv:2011.05623.
- [Zenke and Neftci, 2021] Zenke, F. and Neftci, E. O. (2021). Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109(5):935–950.
- [Zenke et al., 2017] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR.
- [Zhai et al., 2022] Zhai, X., Wang, X., Mustafa, B., Steiner, A., Keysers, D., Kolesnikov, A., and Beyer, L. (2022). Lit: Zero-shot transfer with locked-image text tuning. *CVPR*.
- [Zhang et al., 2022a] Zhang, M., Armendariz, M., Xiao, W., Rose, O., Bendtz, K., Livingstone, M., Ponce, C., and Kreiman, G. (2022a). Look twice: A generalist model predicts return fixations across tasks and species. *PLoS Computational Biology*, page In Press.
- [Zhang et al., 2021] Zhang, M., Badkundri, R., Talbot, M. B., Zawar, R., and Kreiman, G. (2021). Hypothesis-driven online video stream learning with augmented memory.
- [Zhang et al., 2022b] Zhang, M., Dellaferrera, G., Sikarwar, A., Armendariz, M., Mudrik, N., Agrawal, P., Madan, S., Barbu, A., Yang, H., Kumar, T., Sadwani, M., Dellaferrera, S., Piz-zochero, M., Pfister, H., and Kreiman, G. (2022b). Human or machine? turing tests for vision and language.
- [Zhang et al., 2018] Zhang, M., Feng, J., Ma, K. T., Lim, J. H., Zhao, Q., and Kreiman, G. (2018). Finding any waldo with zero-shot invariant and efficient visual search. *Nature communications*, 9(1):1–15.
- [Zhang et al., 2020] Zhang, M., Tseng, C., and Kreiman, G. (2020). Putting visual object recognition in context. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12985–12994.
- [Zhang et al., 2019] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. (2019). Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.
- [Zhou et al., 2007] Zhou, C., Zemanová, L., Zamora-López, G., Hilgetag, C., and Kurths, J. (2007). Hierarchical organization unveiled by functional connectivity in complex brain networks. *Physical review letters*, 97:238103.
- [Zhou and Firestone, 2019] Zhou, Z. and Firestone, C. (2019). Humans can decipher adversarial images. *Nature communications*, 10(1):1–9.
- [Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.
- [Zhuang et al., 2021] Zhuang, C., Yan, S., Nayebi, A., Schrimpf, M., Frank, M. C., DiCarlo, J. J., and Yamins, D. L. (2021). Unsupervised neural network models of the ventral visual stream. *Proceedings of the National Academy of Sciences*, 118(3).
- [Zhuang et al., 2022] Zhuang, J., Gong, B., Yuan, L., Cui, Y., Adam, H., Dvornek, N., Tatikonda, S., Duncan, J., and Liu, T. (2022). Surrogate gap minimization improves sharpness-aware training. *ICLR*.
- [Zinkevich, 2003] Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML’03, page 928–935. AAAI Press.

[Zipser and Andersen, 1988] Zipser, D. and Andersen, R. A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, 331:679–684.

**ETH** zürich



University of  
Zurich <sup>UZH</sup>

Institute of Neuroinformatics  
Prof. Dr. Giacomo Indiveri  
Prof. Dr. Valerio Mante  
Prof. Dr. Evangelos Eleftheriou  
Dr. Angeliki Pantazi  
Prof. Dr. Wulfram Gerstner  
Prof. Dr. Emre Neftci

**Title of work:**

Unveiling Principles of Neural Computations: From Biological to Artificial Intelligence, and Back

**Thesis type and date:**

Ph.D. Thesis, January 2023

**Supervision:**

Prof. Dr. Giacomo Indiveri  
Prof. Dr. Valerio Mante  
Prof. Dr. Evangelos Eleftheriou  
Dr. Angeliki Pantazi  
Prof. Dr. Wulfram Gerstner  
Prof. Dr. Emre Neftci

**Student:**

Name: Giorgia Dellaferrera  
E-mail: gde@zurich.ibm.com