Synaptic Failure is a Flat Minima Optimizer

A THESIS PRESENTED
BY
DEEPAK SINGH
TO
THE FACULTY OF THE COMMITTEE ON DEGREES IN NEUROSCIENCE
AND THE DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE WITH HONORS
OF BACHELOR OF ARTS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MARCH 2023

Neuroscience Concentration
Division of Life Sciences
Harvard University

The Harvard College Honor Code

Members of the Harvard College community commit themselves to producing academic work of integrity – that is, work that adheres to the scholarly and intellectual standards of accurate attribution of sources, appropriate collection and use of data, and transparent acknowledgement of the contribution of others to their ideas, discoveries, interpretations, and conclusions. Cheating on exams or problem sets, plagiarizing or misrepresenting the ideas or language of someone else as one's own, falsifying data, or any other instance of academic dishonesty violates the standards of our community, as well as the standards of the wider world of learning and affairs.

Signature:

List of Contributions

The following authors contributed to Chapter 1: Alexander Davies.

Thesis advisor: Professor Gabriel Kreiman Deepak Singh

Synaptic Failure is a Flat Minima Optimizer

Abstract

Synaptic failure is a well-known phenomenon in which weaker synapses fail more frequently. Whether this has any purpose is still unknown. In this work, we modify Dropout to implement synaptic failure in artificial neural networks, through a novel activation function we call NormOut. NormOut sets a neuron's probability of successfully firing equal to the ratio p of its activation to the maximum activation in some set of neurons. We propose variants inspired by lateral inhibition and firing thresholds, and show that they have hugely different effects on activation dynamics. We find that NormOut improves the performance of a baseline VGG-16 on CIFAR-10, with one variant even outperforming Dropout in achieving both better test accuracy and a significantly flatter minimum. In addition to the effect on overfitting, we explore NormOut's impact on adversarial robustness against a suite of white and black-box attacks. Intriguingly, we find that some variants of NormOut produce extreme gradient masking without obfuscation. Rather than masking through flattening, we find that these variants actually induc high curvature in the loss landscape, suggesting an as yet unknown form of gradient masking. Overall, we show that simply modelling synaptic failure in two layers has a significant impact on the topology of the loss landscape, with the best implementations of synaptic failure optimizing strongly for flat minima. We claim this as evidence that synaptic failure is a feature, and not a bug, of the brain.

Contents

To my family, who taught me to value education.

Acknowledgments

I would like to thank my thesis advisor Gabriel Kreiman for his support and belief in my work from the moment I joined his lab. I am very grateful to have worked in such a welcoming and innovative environment.

I would also like to thank Alexander Davies for his contributions to NormOut, in particular his invaluable work on the codebase. Thanks you to Mengmi Zhang for her mentorship over the summer of 2022, and especially for her suggestion to use a Softmax with temperature.

Thanks also to Ryan Draft, Sam Gershman, Leslie Valiant, Stephen Casper, Trenton Bricken, Darius Lam, and Gabrielle Landry for their hugely helpful feedback.

0 Introduction

0.1 Generalization

Deep learning has undergone a revolution in the last decade. Artificial neural networks (ANNs) can now play Go at a superhuman level (Silver et al., 2017), design antibodies (Shanehsazzadeh et al., 2023), and write poetry (Brown et al., 2020), yet they remain poor at generalization to novel tasks and concepts (Barrett et al., 2018) to which they were not exposed during training. In contrast,

humans excel at extracting concepts quickly and learning from few examples. It is unknown how biological brains learn so efficiently. Nonetheless, as the best existing example of a learning machine, the brain has long been a potent source of inspiration for machine learning researchers. In particular, the field of deep learning has borrowed much from neuroscience, not least the name "neural network".

The brain is our most energy-intensive organ. Millions of years of evolution have enabled the brain to operate at a metabolic cost far lower than the analagous cost for an ANN of the same size. In general, it has been safe to assume that most aspects of the brain serve a purpose. However, there has long been debate on whether the inherent noisiness and stochasticity of the brain is intended, or merely a consequence of such a complex biophysical system. In particular, researchers have known for decades that neural transmission is unreliable. Whether this is a biological quirk, or whether it has a deeper function is yet to be discovered.

0.2 Motivation

This thesis is motivated by a simple observation. Synaptic failure is a process in the brain through which weaker synapses spontaneously fail to fire more frequently, and vice versa (Branco & Staras, 2009). Meanwhile, Dropout is a regularization technique in deep networks where neuronal activations are kept with some fixed probability p, and otherwise set to 0. Both synaptic failure and Dropout instantiate functions of the same type. Namely, both map input stimuli to an output "keep probability" which determines whether the neuron is allowed to fire, or whether it will fail

and output a 0 activation. The function of synaptic failure is hotly debated, but the function of Dropout as a technique to improve generalization is well established. This begs the question: can modelling synaptic failure in ANNs teach us anything about its purpose?

0.3 Outline

Chapter 2 begins with introductory background on deep learning, such as the basics of neural networks. It then covers two key aspects of generalization. First, in detailing overfitting, we provide an analysis of Dropout regularization. Second, we give background on the field of adversarial learning, with a focus on white-box attack methods and gradient masking. Chapter 2 then closes with details on synaptic failure and neuronal competition in the brain. Chapter 3 motivates the modification to Dropout further before introducing NormOut, which is our implementation of synaptic failure in ANNs. We detail a number of NormOut variants and the experimental regime in which we tested them. Chapter 4 contains extensive discussion of the results and a description of ongoing and future work.

1

Background

1.1 Generalization

1.1.1 Formalizing ANNs

To formalize the generalization task, we borrow notation from Hochreiter & Schmidhuber (1997).

Let $A = \{(x, y)|x \in X \subset \mathbb{R}^n, y \in Y \subset \mathbb{R}^k\}$ be the set of all possible input-output pairs.

Let $D = \{(x_i, y_i)|0 \leq i \leq n\}$ be the data, where $D \subset A$. D is divided into the training set

$D_0 = \{(x_i, y_i) | 0 \leq i \leq m\}$ and the test set $D \setminus D_0 = \{(x_i, y_i) | m < i \leq n\}$. The generalization task is to approximate some unknown function $f \subset X \times Y$ which maps a finite set of inputs $X \subset \mathbb{R}^n$ to a finite set of outputs $Y \subset \mathbb{R}^k$.

Using notation from (Baldi & Sadowski, 2013), let $\vec{x}$ be the input vector of data to a neuron, $\vec{w}$ its weights vector, $\vec{z}$ its pre-activation output, $\vec{o}$ its post-activation output, b its bias, and g the activation function. We can then write the activity of a given neuron as a function:

$$NEURON(\vec{x}) = \vec{o} = g(\vec{z}) = g(\vec{w} \cdot \vec{x} + b)$$

Now assume we have a set of neurons, which we will call a layer. If we have m examples in our dataset, each with n features, we will have a matrix $\mathbf{X}$ which is $m \times n$, where each column is an example. We can then consider the weights of all neurons in the layer as a single $n \times k$ matrix, where k is the output dimension of the layer. We can thus write the layer function as:

$$LAYER(\mathbf{X}) = \mathbf{O} = g(\mathbf{Z}) = g(\mathbf{W}\mathbf{X} + \vec{b})$$

The last layer of a neural network maps output probabilities to labels, typically using a sigmoid or softmax function. Both layers can be expressed as:

$$LAYER_L(\mathbf{X}) = \hat{\mathbf{Y}}$$

We can thus write an L-layer ANN as instantiating some overall function:

$$NET(\mathbf{I}) = LAYER_L(LAYER_{L-1}(LAYER_{L-2}(\ldots LAYER_0(\mathbf{I}))))$$

Where $\mathbf{I}$ is the input data. Note that in practice, we do not pass the entire dataset at once, but rather in mini-batches. Once the ANN has produced a set of predicted labels, it then computes a loss function which essentially scores how well the network has done, for example through mean squared error. For a network training on a task with multiple possible labels, however, we typically use the categorical cross-entropy loss function:

$$L_{CE}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\frac{1}{N}(\mathbf{Y} \odot \log(\hat{\mathbf{Y}}))$$

Where N is the number of examples in the batch. The loss is a function of the model's weights and biases, or its set of parameters. The loss function can be understood as high-dimensional surface across which the model moves when it adjusts its parameters. The minima of this surface, or the points with the lowest loss, correspond to the set of parameters which produce the highest performance on the given task. Learning, i.e. moving across the surface from the initial point to a minimum, is implemented through backpropagation (Kelley, 1960; Rumelhart et al., 1986; Linnainmaa, 1970). Backpropagation calculates the gradient of the loss with respect to the model's parameters. Once the direction of steepest descent has been found, the algorithm updates the corresponding parameters to move the model towards the minimum. This can be thought of as akin to a ball rolling down a hill until it reaches some minimum point. Backpropagation is repeatedly applied until the ANN converges on some approximation of the target function.

Through this combination of learning and hierarchical function composition, ANNs can theoretically approximate any expressible function (Hornik et al., 1989). An ANN which closely approximates the function $f \subset X \times Y$ is said to generalize well since it has learnt the function from just the training set D_0. In practice, ANNs rarely converge on the true target function, though the looseness of the approximation can be masked in several ways, as the following sections will explore.

1.2 Overfitting and Regularization

A network is said to "overfit" when its error rate on the test set $D \setminus D_0$ is higher than its error rate on the training set D_0. Such a discrepancy typically means the network has approximated some function f' which describes the training set well, but which poorly approximates the target function f and so fails to generalize to the test set. Overfitting can occur for a variety of reasons. For example, if the training set D_0 is a small or otherwise unrepresentative sample of the larger dataset D, the network may not have the right information to learn f and may instead learn an overly simple f'. Alternatively, the model could be overparameterized or trained for too long on the training set, leading to it learning an overly complex f'. In both cases, f' likely describes the training set near perfectly, but fails to capture the general abstraction captured by the true target function f. Note that recent work has shown that this does not necessarily hold for very deep models, in which a phenomenon known as "double descent" has been observed (Nakkiran et al., 2021), which is contrary to classical theories of overfitting. However, double descent only occurs in heavily overparameterized models, and so is out of the scope of this thesis.

1.2.1 Regularization

Regularization refers to a set of techniques which can be applied to result in a simpler answer to a given problem. In the machine learning context, regularization is applied to reduce overfitting (i.e. improve the approximation of f by f') through simplifying the learned function. The theoretical justification for regularization is that it imposes Occam's Razor on f', biasing the model towards approximating simpler functions. From a Bayesian viewpoint, most regularization methods impose certain priors on model parameters such that f' is a better approximation of f.

Regularization can be either explicit (i.e. by adding a term to the optimization problem) or implicit (i.e. everything else). Explicit regularization includes L1 and L2 regularization, which both add terms to the loss function to penalize large weights. Implicit regularization includes data augmentation, in which more data is added or transformations are made to existing input data, and early stopping, in which training is stopped early to prevent the network learning an overly complex function.

1.2.2 Dropout

Dropout (Hinton et al., 2012) is an implicit regularization technique in which each neuron's activation in a given layer is kept with some probability p, and otherwise with probability $1 - p$ set to 0 (i.e. "dropped"). This effectively prevents the co-adaptation of features by different neurons. Intuitively, the network can no longer afford to have each neuron learn a few bits of highly specialized information which only relate to a subset of the data. Instead, each neuron must learn "general knowledge"

which the network can rely on across the dataset. Empirically, Dropout reduces overfitting in tasks such as classification and regression (Srivastava, 2013; Simonyan & Zisserman, 2014; Szegedy et al., 2015), typically with $p = 0.5$.

Note that Dropout is turned off at inference time. A technique called inverted Dropout is applied during training to ensure the expected value of the overall activations remains the same at test time as it does during training.

To formalize Dropout, we can return to our notation from earlier. Let $\delta \sim \text{Bern}(p)$ be i.i.d gating Bernoulli random variables, such that p represents the probability a neuron is kept. When we apply Dropout, we can write the activity of the following layer as:

$$LAYER^l(\mathbf{X}) = \mathbf{O} = \delta \odot \mathbf{X}$$

For neuron i in layer l, and for a fixed input vector, we can take the expectation of the activity of a neuron over all possible sub-networks realized by the possible combinations of gating variables as:

$$E(O_i^l) = E(g^l(z_i^l)) = E\left(g^l\left(\sum_j w_{i,j}^{l,l-1} p_j^{l-1} E(g^{l-1}(z_j^{l-1}))\right)\right)$$

As noted by (Baldi & Sadowski, 2013), we can compute the ensemble average by simply replacing the weights $w_{i,j}^{l,l-1}$ with $w_{i,j}^{l,l-1} p_j^{l-1}$ during feedforward propagation. The consistency $C(O_i^l, I)$ of neuron i in layer l can be measured using the variance $\text{Var}(O_i^l(X))$ taken over all possible sub-networks and their distribution when the input I is fixed. The larger the variance, the less consistent the neu-

ron. We can write the variance of neuron i in layer l as:

$$\mathrm{Var}(O_i^l) = g^l\Big(\sum_j (w_{i,j}^{l,l-1})^2 (O_j^{l-1})^2 p_j^{l-1}(1-p_j^{l-1})\Big)$$

We can see that $\mathrm{Var}(O_i^l)$ is reduced by: (1) small weights, (2) small activations, and (3) small (close to 0) or large (close to 1) keep probabilities p_i^l (Baldi & Sadowski, 2013). This explains Dropout's effect in enforcing small weights, as well as its asymmetric preference for small activations. We can also see that the regularization effect is maximized when $p = 0.5$, which fits with empirical observations of optimal p values. Small weights are believed to reduce overfitting because they mean models pay less attention to single aspects of the input data. For example, a very large weight could be useful during training if it amplifies some feature which is useful for classification on the training set. However, if that feature is not also useful in the test set, then the model would overfit.

1.3 Adversarial Robustness

It turns out that classical overfitting is neither sufficient nor necessary for poor generalization. It has been empirically observed that ANNs typically learn input-output mappings which are relatively discontinuous, as evidenced by the existence of adversarial examples (Szegedy et al., 2013). An adversarial example is an input which has been perturbed by some small amount of noise norm-bounded by ε, which is found by *maximizing* the network's loss function. Adversarial examples, whilst perceptually identical with the original image to humans, are generally misclassified by the network. Vulnerability to adversarial examples implies discontinuity in the learned mappings because a con-

tinuous mapping would mean that small perturbations to the input do not drastically change the output. Even models which do not appear to suffer from overfitting (i.e. they have equal train and test performance) can and do suffer from adversarial vulnerability.

The nature of these perturbations is not a random artifact of the variability in backpropagation-based learning. That is, a set of adversarial examples which are difficult for one network are generally difficult for another network, even when the networks differ in architecture, activations, and on which subset of the data they were trained (Szegedy et al., 2013).

1.3.1 Adversarial Attacks

The process of finding an adversarial example is called an adversarial attack. It can be thought of as a constrained optimization problem, in which an example must be perturbed with noise norm-bounded by ε, subject to the constraint that the loss of the model is maximized on that input. The norm used is typically the $\mathcal{L}_\infty$ norm, in which only tha largest component of the vector has any effect. The $\mathcal{L}_\infty$ norm thus represents the hardest challenge for an adversary, since its potential perturbation is most constrained. Also popular in the literature is the $\mathcal{L}_2$ norm, which is simply the Euclidean norm of the vector's components. This is a slightly more relaxed norm and therefore easier for the adversary to work with.

There are two main classes of adversarial attacks. First, white-box attacks are attacks in which the adversary has access to the model's weights and gradients. This allows the adversary to perturb an example with knowledge of how the perturbation will affect the model's loss. Second, there are black-box attacks, which do not have access to model gradients and must instead estimate them.

Attack efficacy is usually measured by the target model's error rate on an adversary's attacks, with a more effective adversary inducing a higher error rate. Since white-box attacks optimize based on the true gradient of the target model, they are typically more effective than black-box attacks.

1.3.2 White-Box Attacks

Fast Gradient Sign Method

First proposed by Goodfellow et al. (Goodfellow et al., 2014), the fast gradient sign method (FGSM) attack is a white-box attack which can rapidly generate adversarial examples capable of fooling a network into misclassifying.

Let θ be the parameters of a model, $\mathbf{x}$ the input, y the labels associated with $\mathbf{x}$ and $\mathcal{J}(\theta, x, y)$ be the cost used to train the network. We can linearize the cost function around the current value of θ, obtaining an optimal max-norm:

$$\eta = \varepsilon \text{sign}(\Delta_{\mathbf{x}} \mathcal{J}(, \mathbf{x}, \dagger)$$

The required gradient is calculated using backpropagation.

Projected Gradient Descent

Projected gradient descent (PGD) is an iterative form of FGSM which applies step-based gradient descent to the optimization problem given above. It is typically much harder to defend against than FGSM and has been cited as a good measure of general robustness (Madry et al., 2017).

1.3.3 Untargeted and Targeted Attacks

Adversarial attacks can be either untargeted or targeted. In an untargeted attack, the adversary simply seeks to make the model output any incorrect label. In a targeted attack, the adversary instead seeks to make the model output.a specific incorrect label. Intuitively, a targeted attack should be easier to defend against, since given a dataset with c labels, the model can output $c - 1$ different labels and still not succumb to the attack in the targeted case. On the other hand, in untargeted attacks, the model must ensure it outputs the exact $c_i \in c$ which is the ground truth label.

1.3.4 Black-Box Attacks

There have been a number of black-box attacks proposed in the adversarial learning literature. Bhambri et al. (2020) provide a taxonomy of black-box attacks as follows. First, a transfer attack is one in which thre adversary trains a substitute model on the same task as the target model in order to estimate the target's weights. Second, if the adversary cannot train a substitute, it may instead use queries, which are randomly and continuously crafted adversarial examples it sends to the target in order to observe the target's response. Seeing the target's outputs to a known input can help massively in gradient estimation. Third, the adversary might optimize this query feedback mechanism by attempting to locate the target model's decision boundaries, which it does by consistently adding noise to an input and observing the change in the target's confidence scores.

One such black-box attack is the Square Attack (Andriushchenko et al., 2020). Square utilizes query-efficient sampling in combination with random search to avoid using gradient information

altogether. As we will see later, this makes it particularly robust against a number of potential defenses.

1.3.5 Adversarial Defences

There have been a vast number of proposed defences against adversarial attacks. Verma & Swami (2019) suggest taxonomizing adversarial defences according to the part of the learning pipeline they protect. Input-based defences directly modify the input to mitigate the impact of adversarial perturbations. They include (1) quantization, (2) manifold projections, or (3) randomization. Model-base defences employ either (1) adversarial training, or (2) architectural features specifically designed to thwart adversarial attacks.

In practice, adversarial training has proven to be the most effective defence (Madry et al., 2017), reaching up to 66% accuracy against untargeted PGD attacks. Researchers are yet to develop defenses which can defend against strong black-box attacks like Square.

1.3.6 Gradient Masking

In a white-box setting, both the attacker and the target have access to the same gradient information. As previously mentioned, this explains why white-box attacks are more effective than black-box attacks. However, now assume a setting where the target model knows it is under white-box attack and accordingly that the attacker is using the network's gradient information to choose perturbations. The network could now try a broad range of strategies to alter its gradients such that they are misleading or otherwise no longer useful to the adversary. This strategies belong to the common

class of *gradient masking* (Papernot et al., 2017).

Gradient Obfuscation

There are many types of gradient masking. One subset of gradient masking strategies seeks to break gradient descent, for example by making gradients point the wrong way Tramèr et al. (2017). In contrast, another subset of strategies seek to design defenses in such a way that they necessarily cause gradient masking. These strategies collectively fall under the term *gradient obfuscation.* Athalye et al. (2018) distinguish between three different kinds of gradient obufscation.

First, shattered gradients refer to gradients which are either incorrect or nonexistent. These can result from non-differentiable defenses, numerical instability, or through intentional design choices which make it such that following the gradients does not maximize the loss globally. Second, stochastic gradients result from some part of the network or input being randomized at test time, which causes adversaries using only a single sample of the randomness to incorrectly estimate the gradient. For example, leaving Dropout on at inference time would make it near impossible for an adversary to correctly estimate the gradient, since some random subset of the neurons (and subsequently their gradients) would be dropped. Third, vanishing/exploding gradients are caused by highly convoluted defenses which, when unrolled, are essentially one extremely deep network. These gradients necessarily cause gradient masking since they make it harder for an adversary to estimate the impact of a perturbation on the overall loss of the target model.

Gradient Masking is Not Robust

It is important to note that whilst gradient masking often improves a model's performance against white-box adversaries, this is not the same as improving the model's adversarial robustness. Gradient masking does not alter a model's sensitivity to adversarial examples; instead, it simply hides the directions in which the model is sensitive from gradient optimizing adversaries. This does not prevent these directions being discovered by other means (Papernot et al., 2017). For example, attacks which take random steps across the loss surface often perform well against gradient masking models, since the target network has optimized its loss surface to be robust against loss-maximizing attacks, not random attacks. Since the loss landscapes of deep neural networks are very high-dimensional, these random steps can often quickly find the discontinuities in a target model's learned mapping.

Given that (1) gradient masking does not increase robustness (worse, it actually gives a false sense of security), and (2) it can be done unintentionally, it is critical that we are able to identify it. Athalye et al. (2018) describe 5 "red flags" which signal gradient masking. First, we know that in a white-box setting, multi-step (iterative) attacks are strictly stronger than single-step attacks. If a model performs better against multi-step attacks than single-step attacks, this would suggest the iterative attack is getting stuck at a local minimum during its optimization search. Second, we know that black-box attacks are a strict subset of white-box attacks, so should never outperform them. However, it has been empirically observed that black-box attacks consistently outpeform white-box attacks on gradient masking models (Athalye et al., 2018; Papernot et al., 2017). Third, we know that an attack which is allowed to add unbounded noise to an attack should eventually reach a 100%

success rate. Seeing anything less than this for an unbounded attack would suggest that a model is gradient masking since the attack is failing to perform optimally agains the target model. Fourth, as a correlate of this, increasing the distortion bound should monotonically increase success rate. If the distortion bound is significantly increased but the attack success rate is not, the target model is likely gradient masking since the adversary is not functioning optimally. Fifth and finally, as discussed earlier, random searching within some ε-ball should not find adversarial examples where gradient-based methods cannot.

1.4 Flatness

Flattened loss surfaces are a somewhat common form of gradient masking, since a flatter surface means the gradients in every direction are more similar (and accordingly, no one direction stands out as the best attack vector). However, there are two longstanding theories which connect the flatness of a network's loss landscape more deeply to its generalization ability. The first is an argument based on minimum description length (MDL) (Rissanen, 1978), which suggests that flat minima correspond to weights which can be given with low precision, and correspondingly, described with fewer bits of information. The MDL principle suggests that low network complexity corresponds with high generalization performance (Rissanen, 1978; Hochreiter & Schmidhuber, 1997). A second, Bayesian, argument suggests that "fat" maxima of the posterior weight distribution, or maxima with a lot of probability mass, are favourable for generalization. Hochreiter & Schmidhuber (1997) shows that fat maxima *are* flat minima, bringing the two theories together. Since their seminal work,

there have been a multitude of studies exploring the connection between flatness and generalization (Chaudhari et al., 2019; Foret et al., 2020; Keskar et al., 2016; Liang et al., 2019; Sun et al., 2021; Wu et al., 2020; Yao et al., 2018; Zheng et al., 2021) with a number of these producing large leaps in performance for deep learning models.

1.4.1 Fisher-Rao Norm

Whilst there are many ways to measure flatness, one which we will focus on in this thesis is the Fisher-Rao norm (Liang et al., 2019). The Fisher-Rao norm is defined as follows:

$$|\theta|_{fr} = \theta^T I_\theta \theta$$

Where I is the Fisher information matrix:

$$I(\theta) = \mathbb{E}_{x,y}\left[\Delta_\theta \mathcal{L}(f_\theta(x), y) \Delta_\theta \mathcal{L}(f_\theta(x), y)^T\right]$$

The Fisher-Rao norm essentially puts a lower bound on the generalization gap by measuring how big a network's weights are. There are two key reasons we will be using the Fisher-Rao norm in this thesis. First, the norm is an implicit measure of flatness since the Fisher information matrix approximates the Hessian at minima of the loss under certain conditions. Second, and most importantly, the Fisher-Rao norm is invariant under reparameterization. This means that if a model potentially has two parameter sets θ_1 and θ_2 which implement the same function, the Fisher-Rao norm will be

the same regardless of which parameters are used. This is crucial because the vast majority of flatness measures used in the deep learning literature today are not invariant under reparameterization, making them unreliable.

1.5 Synaptic Failure

We now move to cover the requisite neuroscience background, starting with a discussion of synaptic failure.

In most of the brain, information transfer occurs at chemical synapses, where a presynaptic electrical signal causes synaptic vesicles to fuse with the presynaptic membrane and release neurotransmitter. The strength of a synaptic connection between two neurons can be decomposed into (1) the number of neurotransmitter release sites, (2) the magnitude of the depolarization induced by the neurotransmitter released from a single synaptic vesicle (known as the 'quantal size'), and (3) the probability of neurotransmitter release at each neurotransmitter release site (known as p_r or p_{rel}) Holler et al. (2021). The quantal size (2) is thought to be invariant once given correct measurement of (1) Holler et al. (2021); we thus leave it outside the scope of this paper.

The number of neurotransmitter release sites between connected neurons is known to vary dramatically between neurons. The number of neurotransmitter release sites was originally conflated with the number of synaptic connections between two neurons (which itself varies dramatically), with the assumption being that each synapse released a similar quantity of neurotransmitter (the one-synapse, one-quantum model) Branco & Staras (2009). Recently, Holler et al. found that

synapses actually contain a variable number of neurotransmitter release sites Holler et al. (2021). The formation and strengthening of synaptic connections is known to be a prominent form of neuronal plasticity.

As early as 1954, neurotransmitter release was known to be probabilistic, with identical electrical stimulations resulting in variable quantities of neurotransmitter release (exocytosis) Del Castillo & Katz (1954). We refer to this probabilistic process as "synaptic failure" when an axon terminal which receives a sufficient stimulus fails to release a quantal of neurotransmitter (i.e., unsuccessfull exocytosis), though a more accurate term would be "exocytosis failure" (since this occurs at each neurotransmitter release site). The stochasticity of exocytosis has been directly linked to the unreliability of neural transmission Allen & Stevens (1994), where release failures prevent post-synaptic depolarization. There have been many attempts to quantify the likelihood of neurotransmitter release given a presynaptic signal; Holler et al. examined 10 connected pairs of layer 23 pyramidal neurons, and estimated the release probability per neurotransmitter site (p_r) to be 0.6 ± 0.2 Holler et al. (2021). This estimate is likely more reliable than previous attempts Allen & Stevens (1994); Branco & Staras (2009), since they correctly quantify neurotransmitter release sites (and not just synapses).

Ribrault et al. examined the sequence of reactive and diffusive processes which underlie synaptic transmission (and thus synaptic failure) Ribrault et al. (2011). They note stochasticity in every stage of the transmission process, ranging from the stochastic opening of presynaptic voltage-gated calcium channels to the Brownian motion (intrinsically random motion) of calcium ions and neurotransmitter. These processes can be characterized deterministically for large numbers of molecules, but not for the small number of molecules involved in synaptic transmission. The specific processes

responsible for stochasticity in transmission are beyond the scope of this paper.

1.5.1 Activation Strength and Neuronal Failure

Synaptic failure is usually considered only at the synapse level. In this paper, we are interested specifically in the effect of synaptic failure on the neuron as a whole. At a fixed time, the result of an individual synaptic failure is to lower the cumulative potential of the postsynaptic cell (as a result of summation along the dendrites). Ignoring temporal dynamics, an individual synaptic failure lowers the cumulative potential which either (1) prevents the firing of an action potential (by lowering the cumulative potential below the neuronal threshold), or (2) has no effect (the neuron still fires an action potential). We refer to (1) as neuronal failure, when synapse failure prevents a neuron from firing an action potential. Again ignoring temporal dynamics, synaptic failure has a binary effect on the firing of a neuron's action potential.

Importantly, neurons with synaptic potentials summing to be far greater than the neuronal threshold (a stronger activation) are less likely to be effected by synaptic failure. Accordingly, the amount by which the summation of the synaptic potentials exceeds the neuronal threshold is negatively correlated with the probability that an individual synaptic failure prevents action potential firing. Neurons just above their firing threshold are more likely to be derailed by an individual synaptic failure, while neurons comfortably exceeding their threshold are unlikely to be prevented from firing.

Synapses from one axon onto a single dendritic branch typically have more similar p_rs than synapses onto different branches Branco & Staras (2009). Branco and Staras say that feedback ad-

justments of p_r at individual synapses may be an attempt to filter out local noise while maintaining a postsynaptic cell-wide p_r. Branco & Staras (2009).

1.6 Sparsity

Biological neural networks are characterized by (1) sparse connections, and (2) sparse activations (Barth & Poulet, 2012). This is believed to have a number of benefits, including reduced metabolic costs, efficient coding, and better robustness (Oldfield et al., 2010; Olshausen & Field, 2004). There a number of proposed mechanisms behind this sparsity.

Gating as Multiplicative Interaction

Gating mechanisms are commonly used to produce sparsity in the brain (Rao-Ruiz et al., 2019; Ehlis et al., 2009). Underlying gating (and indeed also convolutions and attention) are multiplicative interactions. Multiplicative interactions are known to have a specialized function in gain modulation, in which a neuron's tuning profile is multiplicatively modulated by contextual factors such as attention (Ferguson & Cardin, 2020). Whether multiplicative interaction functions as a canonical nonlinear transformation in sensory coding, and thus has a wider role in cortical computation, is still an open problem (Jayakumar et al., 2020).

1.6.1 Modularity and Redundancy

The brain is highly modular, both in its explicit architecture and in the sub-networks of neurons which activate for given stimuli, as determined by their selectivity. The relationship between spar-

sity and sub-network formation is still poorly understood, but may have massive ramifications for explaining the robustness of neural circuitry. This is because neural circuitry is often highly redundant, which in turn allows for high fault tolerance (Panousis et al., 2021).

1.6.2 Inhibitory Interneurons

Inhibitory interneurons form inhibitory feedback loops in parts of the brain like the cerebellum Eccles et al. (1967), where they play an essential role in maintaining a high sparsity level in some network of cells (e.g. like Golgi interneurons with granule cells). The ubiquity of inhibitory interneurons throughout the brain is testament ti the careful control of sparsity levels which biology maintains, given the extortionate metabolic costs of havving dense activations.

1.7 Competition

1.7.1 Normalization Induces k-WTA Dynamics

Divisive normalization is a canonical neural computation in the brain, and has been used to explain phenomena such as light adaptation in the retina, odour detection in *Drosophilia melanogaster*, and neuronal responses in V1 to increasing stimulation (Carandini & Heeger, 2012). Normalization can be applied both temporally (i.e. for the same neuron across time) and spatially (i.e. across a given normalization pool of nearby neurons) (Carandini & Heeger, 2012). These two methods can also be combined to produce spatiotemporal normalization (Louie et al., 2011).

One mechanism based on temporal normalization involves a postsynaptic terminal decreasing p_r

in response to previous activity via local feedback. Prolonged stimulation of the postsynaptic target has been shown to suppress neurotransmitter release, meaning that the probability of successful exocytosis is dependent on the history of its postsynaptic terminal (Branco & Staras, 2009).

However, normalization is not solely employed to regulate individual neuronal failure, but is also applied spatially to produce *k*-Winners-Take-All (*k*-WTA) neuronal dynamics, in which the *k* largest activations suppress all other neurons. The recurrent on-center, off-surround organization of neurons, observed in many cortical (Andersen et al., 1969), sub-cortical (e.g. hippocampal (Andersen et al., 1969)), and cerebellar (Eccles et al., 1967)) regions of the brain, has been argued to induce *k*-WTA dynamics (Grossberg, 1982).

1.7.2 Lateral Inhibition

Another mechanism which produces competitive dynamics is lateral inhibition. For example, in the cerebellum, inhibitory stellate interneurons modulate the activity of Purkinje cells through a negative feedback loop (Eccles, 1965; Dizon & Khodakhah, 2011; Brown et al., 2019; Andersen et al., 1964; Mittmann et al., 2005; Häusser & Clark, 1997; Oldfield et al., 2010), whilst Golgi interneurons use GABA-ergic lateral inhibition to produce a center-surround organization in granule cells (D'Angelo, 2008). Lateral inhibition is especially important in networks involved in sensory perception, since inhibition can increase contrast within a signal. For example, lateral inhibition is thought to be critical for the spatial tuning of retinal ganglion cells (Cook & McReynolds, 1998).

2

Methods and Experiments

2.1 Motivating an Improvement to Dropout

This thesis is driven by the insight that Dropout and synaptic failure can be modelled as functions in the same class; that is, they are both ways of deciding the keep probability for a neuron. Whereas Dropout is a constant function, synaptic failure ties the probability of a neuron's failure to its activation strength. We now move to modify Dropout such that we can implement synaptic failure in

ANNs.

Dropout is a Poor Approximation of Synaptic Failure

Dropout relies on a number of independence assumptions which we argue make it a poor approximation of synaptic failure. These assumptions were originally made in (Srivastava, 2013) to maximize the regularizing effect of Dropout. However, we suggest that these assumptions both (1) limit Dropout's potential, and (2) make it a worse approximation of synaptic failure. We propose that more closely modelling synaptic failure by removing these assumptions may instead produce tangible benefits.

First, the gating variables are assumed to be independent of the neuronal activations. Dropout will thus drop neurons with equal probability regardless of activation strength. This is in direct contrast to synaptic failure under which neuronal failure rate is inversely correlated with activation strength. The magnitude of a neuronal activation can be thought of as a relative approximation of information content; larger activations typically mean more information (and thus the neuron is more useful), and vice versa. As such, using the activation-dependent firing probabilities found in synaptic failure would be a way to ensure Dropout keeps high information content with higher probability. This in turn may allow Dropout to function as a filter which suppresses out low information content.

Second, we speculate that different inputs may require differently sized sub-networks to correctly classify the input. Under Dropout, for a layer of n neurons, in expectation a sub-network of $\lceil np \rceil$ neurons will activate regardless of the input. However, under Dropout, if $p = 0.5$, activation

sparsity will always (in expectation) be at 50%, regardless of the input. As such, Dropout represent a heavy-handed approach to achieving network sparsity. Deciding neuronal firing probabilities per input, as in synaptic failure, would instead allow Dropout to dynamically adapt its sub-network size to best suit the input.

Finally, each neuron's gating variables δ are assumed to be independent of each other. By consequence, neurons under Dropout are never in competition with each other, which is biologically implausible. Moreover, failing to induce competition means Dropout does not take advantage of well-initialized neurons early in training by updating them significantly for rapid learning.

We speculate that Dropout's search of the space of possible sub-networks is highly inefficient because it fails to use competition as a guide. We suggest that using synaptic failure inspired, activation-dependent firing probabilities would instead provide a guiding influence on sub-network search, through a "rich get richer" principle which strengthens the weights of neurons which activate early on and thus make them specialize more and thus form a sub-network.

Modifying Dropout for Better Synaptic Failure Modeling

In order to incorporate synaptic failure modelling into an artificial neural network, we make two key simplifying assumptions. First, we model neuronal failure resulting from synaptic failure, as opposed to individual synaptic failure. Second, we ignore the temporal dynamics of synaptic failure between time steps (where neuronal failure at a given time step makes neuronal success more likely at subsequent time steps).

Together, this lets us model synaptic failure per neuron as a function $g : \mathbf{O} \rightarrow \mathbf{p}$. In an arti-

ficial neural network, we model $\mathbf{O}$ as an $m \times n$ matrix corresponding to the absolute value of the activations of a single n-neuron layer across an m-input batch. We index our batch dimension via superscript j, and our neuron dimension by subscript i. We model $\mathbf{p}$ as an $m \times n$ matrix where p_i^j is the keep probability for the i^{th} neuron on the j^{th} input. Our decision to use magnitude instead of raw activation reflects our intuition that strong activation represents high information content regardless of its sign.

In the brain, we consider O_i^j as the summation of all dendrosomatic potentials which would be activated given no synaptic failure occurs, minus the neuron's threshold. We consider p_i^j to be the probability that this cumulative potential does in fact trigger the firing of an action potential (as opposed to synaptic failure lowering the total to below the neuron's threshold).

Note that this mapping in neither case influences the magnitude of the neuron should it fire (i.e., the magnitude of the action potential), but instead only determines whether the neuron is zeroed (obstructed from firing) or left untouched.

Types of Dropout in Modeling Synaptic Failure

We note that Dropout can be applied to either the weights (synaptic Dropout/DropConnect (Wan et al., 2013)), or to neurons (Srivastava, 2013) (neuronal Dropout). We focus on the latter for two reasons. First, neuronal Dropout is the most commonly used form of Dropout and empirically has a higher performance-to-training-cost ratio (Wan et al., 2013), since synaptic Dropout requires averaging across an ensemble of networks. Training efficiency is an important bioplausibility consideration since biological brains face metabolic constraints. Second, under synaptic Dropout, a

neuron will still fire even if a number of its weights have been dropped (albeit with a smaller activation). However, a biological neuron will only fire if the sum of its synaptic potentials is above its firing threshold. Under synaptic failure, if enough synapses fail such that the neuron is below its firing threshold, the neuron will not fire at all. Accordingly, we propose that neuronal Dropout is a better analogue to synaptic failure than synaptic Dropout.

2.2 NormOut: Synaptic Failure in ANNs

We now present NormOut, an improved version of Dropout modelled on synaptic failure. Let $\mathbf{O}$ be an $m \times n$ matrix corresponding to the activations of a single n-neuron layer across an m-input batch. We index our batch dimension via superscript j, and our neuron dimension by subscript i. We model $\mathbf{p}$ as an $m \times n$ matrix where p_i^j is the keep probability for the i^{th} neuron on the j^{th} input. NormOut calculates individual keep probabilities per neuron per input by dividing that neuron's activation by some maximum activation defined temporally (across the batch), spatially (across the layer), or spatiotemporally (across both). We now detail each of these methods in turn.

2.2.1 Temporal NormOut

We can simply divide each activation by the maximum activation experienced by the neuron over the batch. To do so, we evaluate the keep probabilities for the i^{th} neuron at the j^{th} input in the batch as:

$$p_i^j = \frac{O_i^j}{\max(\vec{O_i})}$$

We refer to this as "TemporalMax."

There is a sense in which TemporalMax approximates a neuronal firing threshold. Across a given batch of data, the neuron will activate in every input. However, in expectation, only activations which are at least half of the maximum activation will be kept. As such, the neuron's effective firing threshold becomes half its maximal activation in the batch.

2.2.2 Spatial NormOut

Rather than dividing each activation by the maximum activation experienced by the neuron over the batch, we can instead divide each activation by the maximum activation across the layer for a single input. To do so, we evaluate the keep probability for the i^{th} neuron at the j^{th} input as:

$$p_i^j = \frac{O_i^j}{\max(\vec{O}^j)}$$

We call this "SpatialMax."

There is a sense in which SpatialMax models lateral inhibition. If one neuron in the layer activates strongly, but the other neurons only have mild activations, we would see the strongly activated neuron suppress the other neuron's activations by driving their keep probabilities down to 0.

2.2.3 Spatiotemporal NormOut

Spatiotemporal normalization dynamics are known to exist in parts of the brain (Louie et al., 2011). Accordingly, we also considered combining the spatial and temporal models. We divide by the max-

imum activation magnitude over both the batch and spatial dimension. This time, we model the value of the i^{th} neuron for the j^{th} input as:

$$p_i^j = \frac{O_i^j}{\max(\mathbf{O})}$$

We refer to this as "SpatiotemporalMax."

2.2.4 Softmax NormOut

$$p_i^j = \frac{\exp(O_i^j/T)}{\exp(\max(\mu)/T)}$$

Where $\mu \in \{\vec{O_i}, \vec{O^j}, \mathbf{O}\}$, depending on the underlying NormOut variant chosen.

Introducing a Softmax allows for the introduction of a temperature hyperparameter. This in turn allows for precise control of the keep probability and thus resulting sparsity. For example, assume a layer containing 2 neurons, and an input batch of size 5. Let the activations for neuron 1 be the set $A_1 = \{2, 3, 4, 5\}$ and the activations for neuron 2 be the set $A_2 = \{4, 6, 8, 10\}$. Now assume we pass a single input to the layer and it produces the activations $\{1, 2\}$. Under standard TemporalMax, the corresponding keep probabilities would be $[0.2, 0.2]$ as 1 and 2 are both 20% of the maximal activation for each respective neuron. However, this fails to take into account the fact that the absolute difference between the current and maximal activations for neuron 1 is only 4, compared to 8 for neuron 2.We ought to penalize neuron 2 more for being so much further away from its maximal activation. Under Temporal Softmax (with no temperature), our keep probabilities would instead be $[0.02, 0.0003]$, which penalizes neuron 2 much more heavily. The importance

of the temperature hyperparameter also becomes clear from this example, since the current keep probabilities are too low for learning. At a temperature of 2.5, the keep probabilities are instead [0.2, 0.02], which is more in line with what is desired. Note that 2.5 is a high temperature for this toy dataset and network.

2.3 Experiments

2.3.1 VGG-16 on CIFAR10

We trained a Baseline VGG-16 model (see Figure **??**) with Batch Normalization on the CIFAR-10 dataset for 100 epochs, with a batch size of 256, the SGDM optimizer with $\beta = 0.9$, a learning rate of 0.01, and weight decay with $\lambda = 0.0001$. We used a set of standard data augmentations including random flips, crops, rotations, and translations.

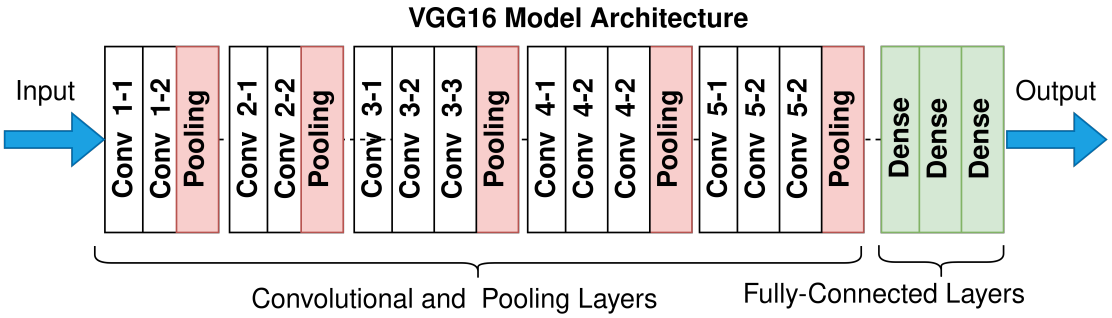

Figure 2.1: Architecture of VGG-16.

In order to test our normalization methods, we constructed different custom layers, each of which implemented one of the six possible normalization methods (i.e. TemporalMax, SpatialMax, SpatiotemporalMax, Temporal Softmax, Spatial Softmax, and Spatiotemporal Softmax). Our base-

line VGG-16 model does not use Dropout. We also trained a baseline model without data augmentation to see its effects, but all other models used data augmentation throughout.

The classifier of our Dropout model was as follows (where the number in parentheses is the layer index):

```
(46): Linear(in-features = 25088, out-features = 4096, bias = True)
(47): ReLU(inplace = True)
(48): Dropout(p=0.5, inplace=False)
(49): Linear(in features=4096, out$features = 4096, bias = True$)
(50): ReLU(inplace=True)
(51): Dropout(p=0.5, inplace=False)
(52): Linear(in-features=4096, out-features=10, bias=True)
```

To test each normalizatio method, we tested first the standard methods (i.e. without Softmax) by swapping out the Dropout at layer 48, the Dropout and layer 51, and both. We then tried all Softmax variants swapping out both layers at indices 48 and 51. We recorded our train and test accuracies on a standard 90-10% split. We also recorded the average activation sparsity per batch for each custom layer. Note that to correctly log sparsity, our Baseline VGG-16 model is slightly different to the standard VGG-16, in that we switched the order of the Dropout and ReLU layers in the classifier. However, as the two operations commute, this does not change the network in any meaningful way. We always remove the probabilistic layers during inference (i.e., change them to identity functions), so as to avoid gradient hacking (Athalye et al., 2018). We also recorded the mean, maximum, and standard deviation of the activations in each custom layer.

2.3.2 Robustness Testing

We also used CleverHans (Papernot et al., 2018) to implement both Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) and Projected Gradient Descent (PGD) adversarial attacks. Both FGSM and PGD are white-box attacks, meaning they have access to the model's weights and gradients. We used $\varepsilon = \frac{8}{255}$ for both attacks, and for PGD used a step size of $\frac{3}{255}$ with 10 attack iterations. Our PGD attacks were divided into untargeted PGD-CE and targeted PGD-T attacks. In PGD-CE attacks, the adversary seeks to simply make the model misclassify by outputting any incorrect label, whilst in PGD-T attacks, the adversary seeks to make the model output a *specific* incorrect label. We tested both the $\mathcal{L}_\infty$ and $\mathcal{L}_2$ regimes.

2.3.3 Flatness

We calculated the Fisher-Rao norm for a selection of our best performing models.

3

Results

3.1 Benign Accuracy

We first present the train and test accuracies of all tested models in Table 3.1. Model names are of the format "Layer-Type indices temperature". Note that overfitting here is calculated as simply the absolute difference between train and test accuracy.

In Table 3.1, we can see that nearly all forms of NormOut performed as well or better (as mea-

Table 3.1: Benign train and test accuracies of all models. Overfitting recorded as the absolute difference between train and test accuracy.

Model	Train Accuracy (%)	Test Accuracy (%)	Overfitting
Baseline-no-data-aug	100	86.7	13.3
Baseline	96.9	90.1	6.8
Dropout 48	96.6	91.2	5.4
Dropout 51	96.8	91.3	5.5
Dropout 48 51	96.6	91.3	5.3
SpatialMax 48	96.2	90.6	5.6
SpatialMax 51	96	90.2	5.8
SpatialMax 48 51	94.2	90.1	4.1
TemporalMax 48	96	90.9	5.1
TemporalMax 51	96.1	90.1	6
TemporalMax 48 51	95.6	90.2	5.4
SpatiotemporalMax 48	95.4	90.6	4.8
SpatiotemporalMax 51	95.8	91.2	4.6
SpatiotemporalMax 48 51	93.5	91.1	**2.4**
Spatiotemporal Softmax 48 51 T=5	10.5	17.2	6.7
Spatiotemporal Softmax 48 51 T=10	10.8	18.9	8.1
Spatiotemporal Softmax 48 51 T=50	96.6	91.3	5.3
Spatiotemporal Softmax 48 51 T=80	96.6	91.1	5.5
Spatiotemporal Softmax 48 51 T=100	96.8	91.1	5.7
Spatiotemporal Softmax 1 4 T=80	98	91.1	6.9
Temporal Softmax 48 51 (T=10)	96.67	90.95	5.7
Temporal Softmax 48 51 (T=25)	96.49	91.19	5.3
Temporal Softmax 48 51 (T=50)	96.94	90.98	6.0
Temporal Softmax 48 51 (T=100)	97.13	**92.58**	4.6
Temporal Softmax 48 51 (T=150)	96.65	91.01	5.6

sured by test accuracy) than the baseline VGG-16 without Dropout, suggesting that NormOut has a net positive effect on our model's learning ability. The two models which failed to learn at all were the Spatiotemporal Softmax 48 51 models with temperatures of 5 and 10. Validation accuracy did not reach above 20% for either model. We included them in Table 3.1 nonetheless to contrast with the same Spatiotemporal Softmax 48 51 model at higher temperatures, which all achieved performance comparable with Dropout. As expected, models performed better when using two NormOut layers rather thanone. This makes intuitive sense given that NormOut is merely a modification of Dropout, so we would expect its positive effects to stack in the same way that we observe for Dropout.

Intriguingly, whilst no models which actually learnt performed worse than the baseline, only one actually outperformed Dropout. The Temporal Softmax 48 51 model with temperature of 100 reached 92.6% test accuracy, which was a a 1.3% improvement on Dropout's best performance. Whilst this may seem small, this is actually a not-insignificant increase. For context, a VGG-16 model pretrained on ImageNet and then modified specifically for performance on CIFAR-10 reached 93.4% test accuracy (Giuste & Vizcarra, 2020), which shows the difficulty in improving VGG-16's accuracy on CIFAR-10 once above 91%.

As expected, all tested models overfit to some degree. We find that data augmentation provides the largest decrease in overfitting, which is why we use it as a prerequisite with all NormOut models.

3.2 Regularization Effects

We now present the regularization effects of our various models. For ease of comparison, we compare only the best performing models from Table 3.1. In Tables **??** and **??**, we present the maximum, mean, standard deviation, variance, and average sparsity of the activations in layers 48 and 51 across the best performing models as measured at the end of training. Note we present the exponentially weighted moving average of each metric to smooth out the effect of input variability.

Table 3.2: Activation statistics for the best performing models across layer 48.

Model	Activations				Mean Sparsity (%)
	Mean	Max	Standard Deviation	Variance	
Baseline	0.242	33.7	0.791	0.626	88.2
Dropout 48 51	0.106	31.5	0.558	0.311	94.5
SpatialMax 48 51	0.122	143.1	1.11	1.23	97.8
TemporalMax 48 51	0.192	15	0.696	0.484	93.7
SpatiotemporalMax 48 51	0.23	36.2	1.32	1.74	97.1
Spatiotemporal Softmax 48 51 T=80	0.25	28.2	0.841	0.707	89.8
Temporal Softmax 48 51 (T=100)	0.245	29.1	0.814	0.663	88.6

Table 3.3: Activation statistics for the best performing models across layer 51.

Model	Activations				Mean Sparsity (%)
	Mean	Max	Standard Deviation	Variance	
Baseline	0.145	6.39	0.186	0.0346	70.7
Dropout 48 51	0.134	7.42	0.148	0.0219	70.8
SpatialMax 48 51	0.069	39.6	0.232	0.0538	94
TemporalMax 48 51	0.11	5.67	0.155	0.024	87.7
SpatiotemporalMax 48 51	0.0816	12.2	0.263	0.0692	96
Spatiotemporal Softmax 48 51 (T=80)	0.145	8.84	0.187	0.035	72.7
Temporal Softmax 48 51 (T=100)	0.15	8.48	0.194	0.0376	71.1

In Figures 3.1 and 3.2, we plot the mean activations with standard deviation as the error bar. Mean activations have been sorted in ascending order. Note that these change considerably over time. To account for this, we present our recordings of these quantities over the duration of train-

ing in Figures 3.6 through ??. Note that in each of the graphs presented below, the data has been smoothed considerably for better readability.

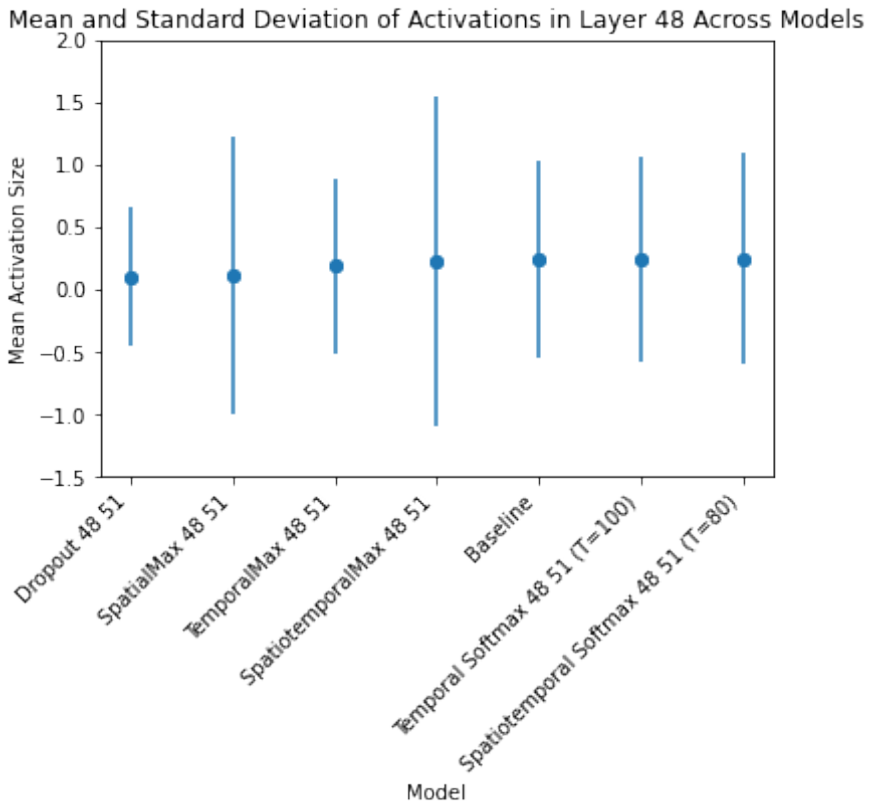

Figure 3.1: Mean and standard deviation of activations in layer 48 across models.

3.2.1 Dropout

As discussed in Section 1.2.2, Dropout theoretically incentives smaller activations and keep probabilities which are close to 0 or 1. Empirically, we observe this occurring, with Dropout producing very small mean activations with a low standard deviation across layers 48 and 51 as can be seen in Figures 3.1 and 3.2 respectively. In Tables 3.2 and 3.3, as well as Figures 3.1 and 3.2, we can see that

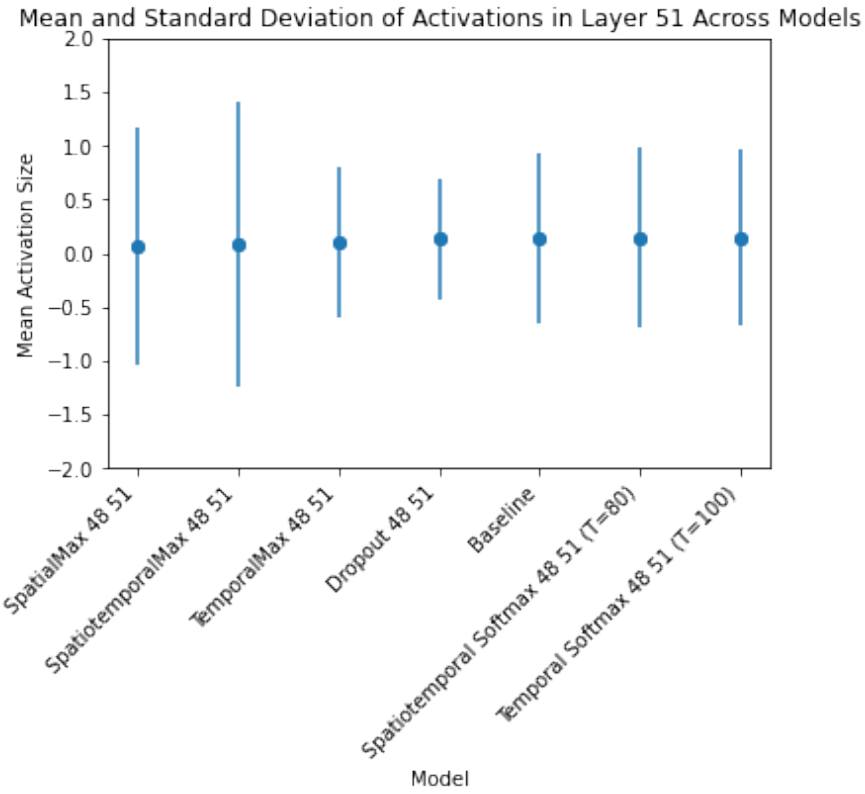

Figure 3.2: Mean and standard deviation of activations in layer 51 across models.

Dropout drastically reduces variance, again aligning with theoretical intuition. We can also that Dropout increases sparsity in layer 48, with the average sparsity increasing by around 5% to 93.3%. A reader might be confused that Dropout with keep probabiltiy set to 0.5 is only increasing sparsity by 5%. However, note that we report absolute average sparsity (i.e. the number of elements in the final tensor which are 0), and not sparsity relative to the number of non-zero elements in the input. That is, the baseline model, which reports 88.2% sparsity in layer 48, already only has 11.2% of its neurons active; Dropout reduces this to 6.7%, which is close to half the baseline activity. This is in line with our keep probability of 0.5, since roughly half the active neurons are being dropped.

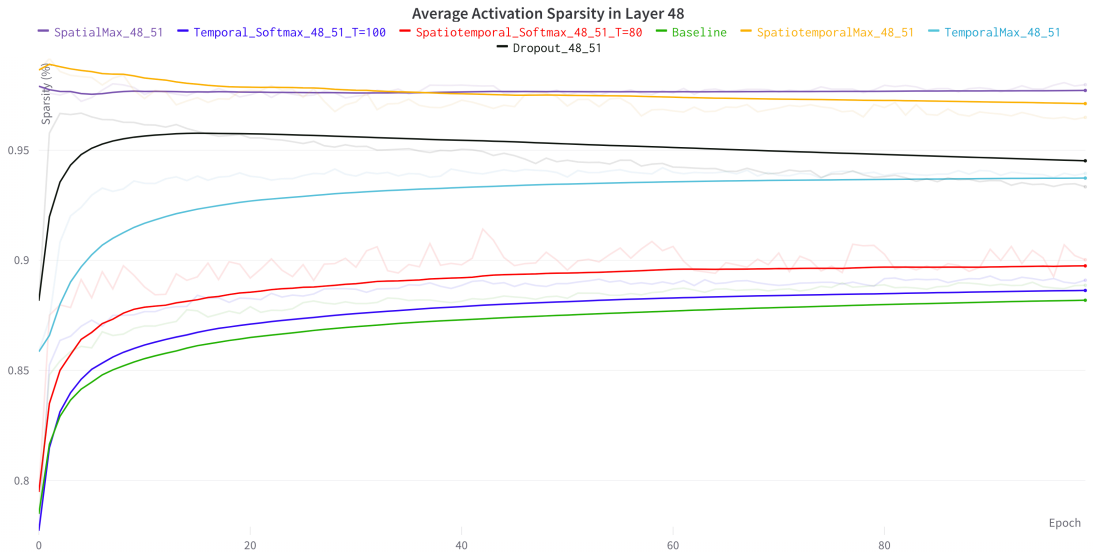

Figure 3.3: Average activation sparsity in layer 48 across models.

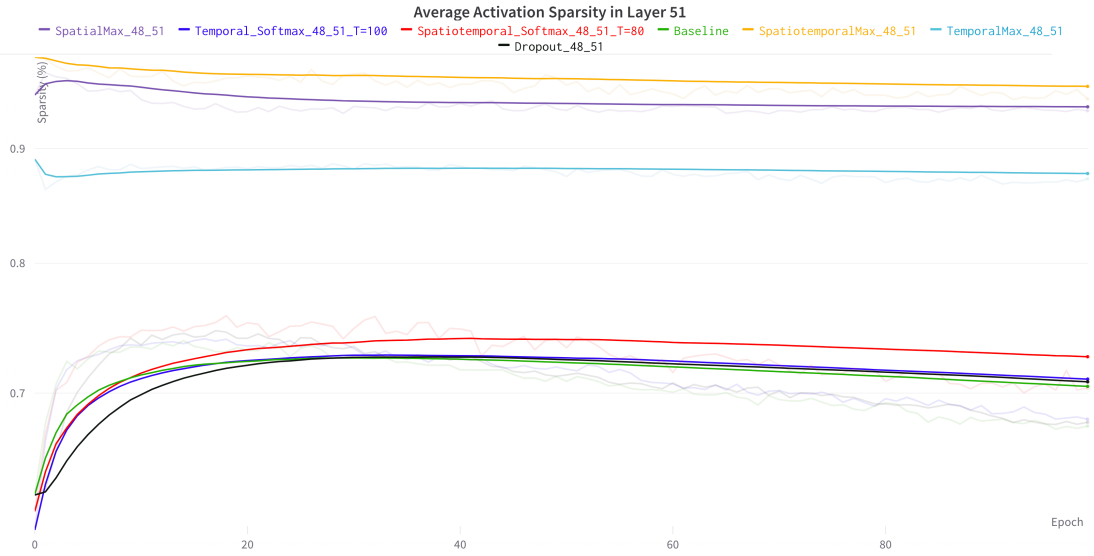

Figure 3.4: Average activation sparsity in layer 51 across models.

3.2.2 SpatialMax

Different versions of NormOut had very different effects on activations in layers 48 and 51. The statistics of SpatialMax activations are largely explained by their incredibly high sparsity (97.8% and

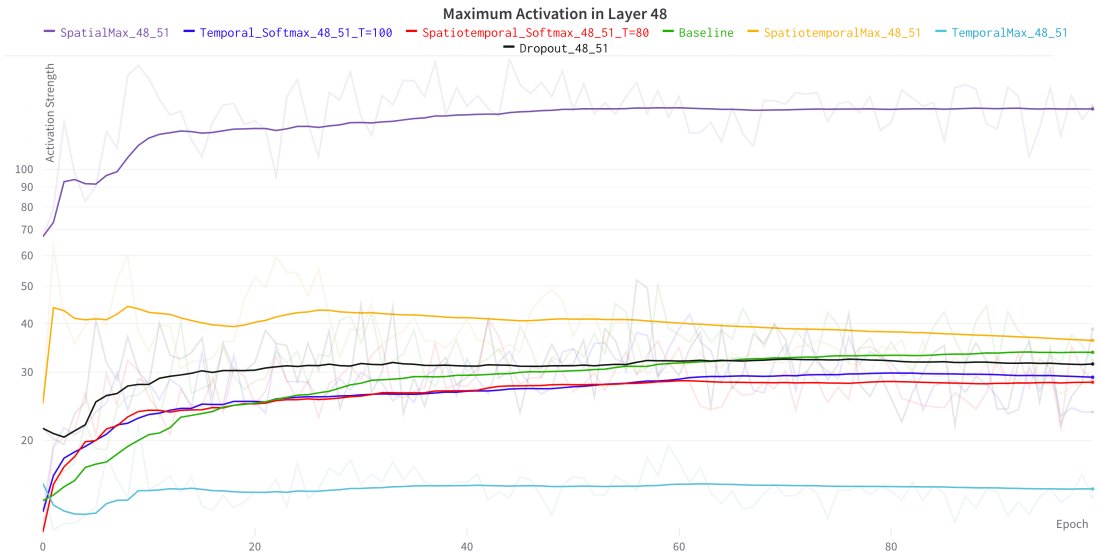

Figure 3.5: Maximum activation size in layer 48 across models.

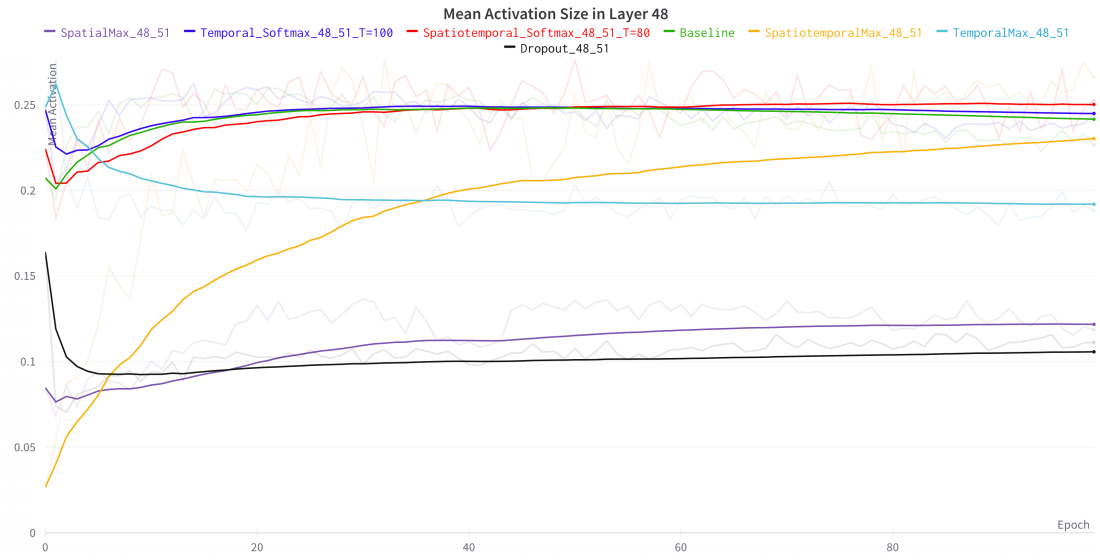

Figure 3.6: Mean activation size in layer 48 across models.

94.0% in layers 48 and 51 respectively) as well as the extremely large maximal activations, which were many multiples larger than any other tested model. These two facts mean there were very few

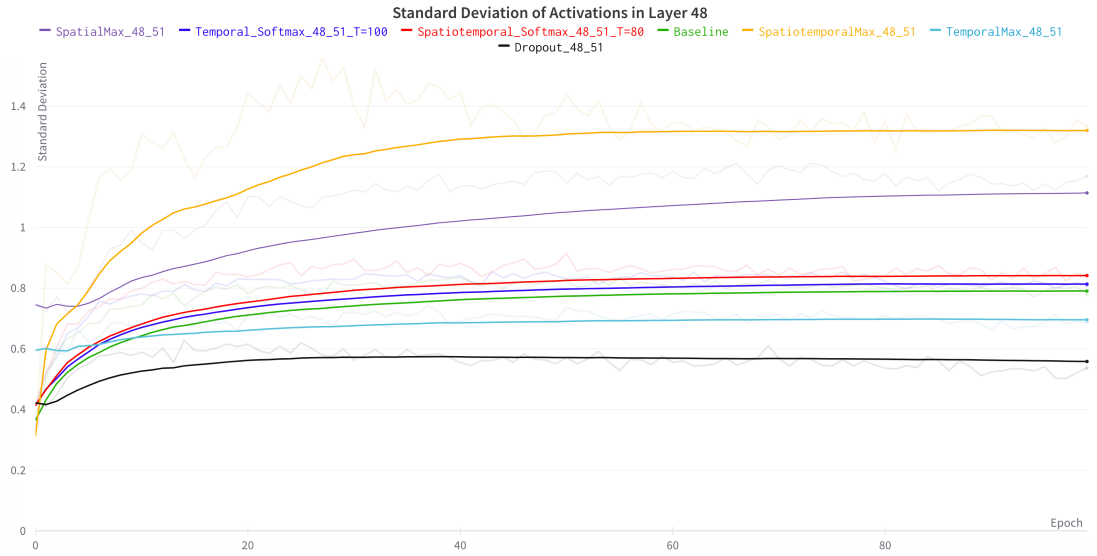

Figure 3.7: Standard deviation of activations in layer 48 across models.

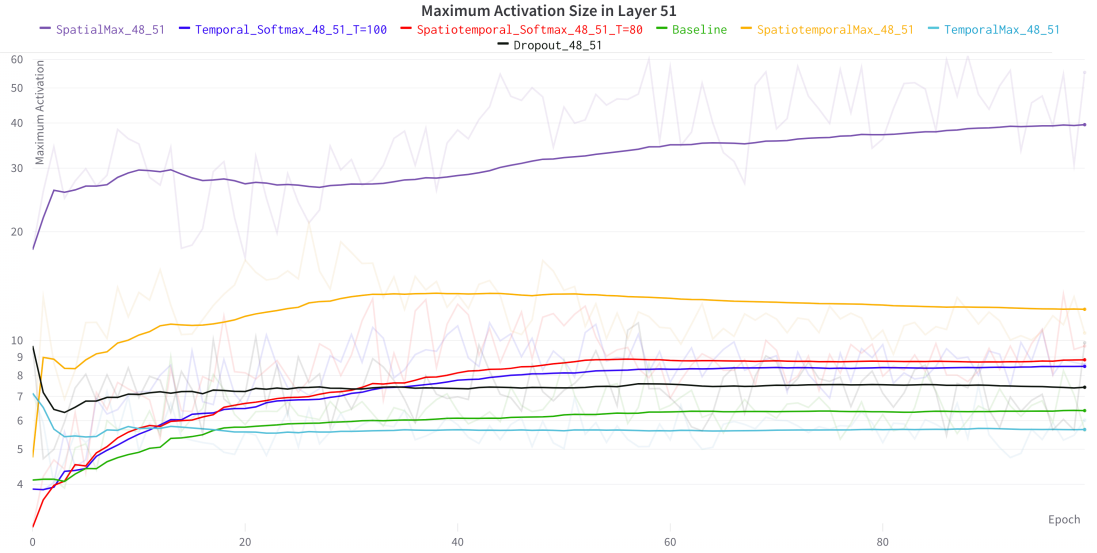

Figure 3.8: Maximum activation size in layer 51 across models.

neurons active, save for a few with very large activations. This makes the small mean activation of SpatialMax misleading; this is an artifact of the tail heavy distribution.

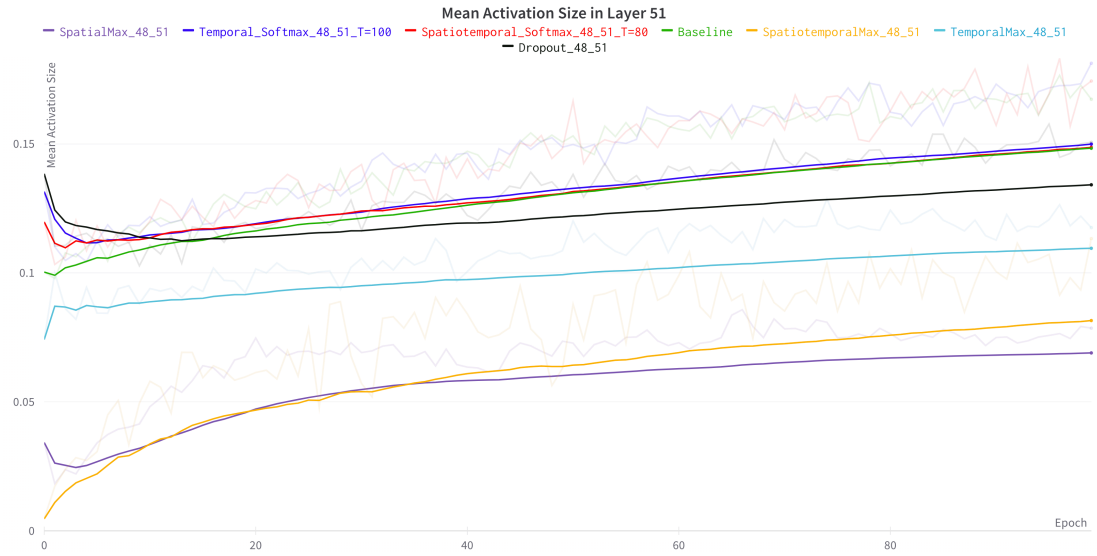

Figure 3.9: Mean activation size in layer 51 across models.

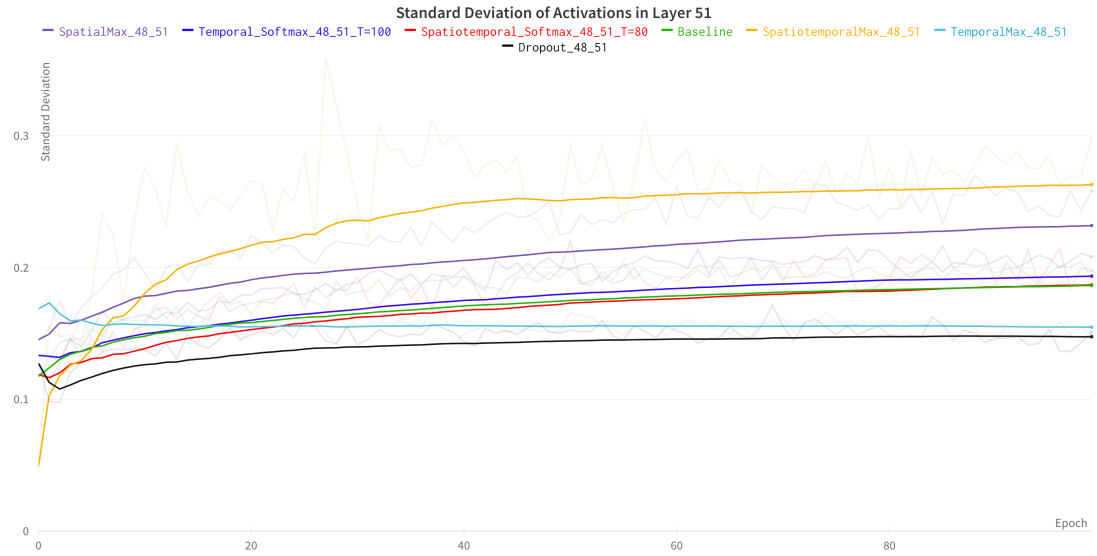

Figure 3.10: Standard deviation of activations in layer 51 across models.

3.2.3 TemporalMax

TemporalMax produced activations which were relatively similar to Dropout. Although activations were, on average, larger and more variable in layer 48 for TemporalMax as compared to Dropout,

in layer 51 TemporalMax produced smaller activations with near identical variance to Dropout. Sparsity in Layer 51 was considerably higher (more than 17% more) for the TemporalMax model as compared to Dropout.

3.2.4 SpatiotemporalMax

SpatiotemporalMax models produced results which look like a dampened version of the SpatialMax model. Sparsity and variance were extremely high in layers 48 and 51, comparable with and even exceeding SpatialMax, whilst the effect of mean suppression towards 0 was slightly lessened by the lower maximal activation.

3.2.5 Softmax

The Softmax models both had very similar activation statistics despite one using SpatiotemporalMax with temperature 80 and the other using TemporalMax with temperature 100. Both models produced relatively large, high variance activations with baseline sparsity levels in layer 48. In Layer 51, nearly all statistics were identical to the baseline model. This is particularly surprising given that the Temporal Softmax (T=100) model outperformed the baseline model by 2.57% in terms of test accuracy.

3.3 Adversarial Robustness

We now show the performance of our models against the various adversarial attack regimes described in the previous chapter. In Table 3.4, we show the accuracy of the models agaisnt $\mathcal{L}_\infty$ at-

tacks, and in Table 3.5, we show the accuracy against $\mathcal{L}_2$ attacks.

Table 3.4: Robust accuracies against various adversarial attacks in the $\mathcal{L}_\infty$ regime. We used $\varepsilon = \frac{8}{255}$ for all attacks, and for PGD used a step size of $\frac{3}{255}$ with 10 attack iterations.

Model	FGSM	PGD-CE	PGD-T	SQUARE
Baseline	13.2	4.8	11.7	0
Dropout 48 51	8.98	3.5	11	0
SpatialMax 48 51	56.3	36.3	14.2	0
TemporalMax 48 51	30.9	7.4	12.3	0
SpatiotemporalMax 48 51	89.1	84.8	16.2	0
Spatiotemporal Softmax 48 51 (T=50)	21.1	5.43	10.8	0
Spatiotemporal Softmax 48 51 (T=80)	17.4	5.67	10.3	0
Temporal Softmax 48 51 (T=10)	33.9	16.4	11.2	0
Temporal Softmax 48 51 (T=25)	22.2	5.56	10.4	0
Temporal Softmax 48 51 (T=50)	22.2	5.56	10.4	0
Temporal Softmax 48 51 (T=100)	22.2	5.56	10.4	0

Table 3.5: Robust accuracies in the $\mathcal{L}_2$ regime.

Model	PGD-CE	PGD-T
Baseline	5.14	5.33
Dropout 48 51	5.23	5.15
SpatialMax 48 51	79.9	5.14
TemporalMax 48 51	5.62	10.5
SpatiotemporalMax 48 51	84.8	16.2
Spatiotemporal Softmax 48 51 (T=50)	33.2	11.2
Spatiotemporal Softmax 48 51 (T=80)	15.3	11.3
Temporal Softmax 48 51 (T=100)	5.67	10.2

In Table 3.4 the baseline and Dropout models show classic adversarial vulnerability, performing at chance (i.e. around 10%) on FGSM and PGD-T, and worse than chance on PGD-CE and SQUARE. It makes sense that the models perform worse than random on these latter two attacks as they are specifically designed to drive the model's accuracy down to 0. TemporalMax models perform significantly better on FGSM attacks, reaching over 30% accuracy but perform just as badly as baseline on all forms of PGD attack.

Models which utilize spatial normalization produce very results against these adversarial attacks. The SpatialMax model reached 56.3% accuracy against FGSM and 36.3% against PGD-CE. How-

ever, it performed at chance against PGD-T attacks, in a telltale sign of gradient masking which we will discuss in Chapter 4. SpatiotemporalMax nearly completely thwarted FGSM and PGD-CE attacks, reaching 89.1% and 84.8% accuracy on each respective attack. However, like SpatialMax, it performed close to chance on PGD-T, again signalling strong gradient masking.

Adding a Softmax and increasing the temperature up to 50 and 80 brought SpatiotemporalMax's performance against the adversaries much closer in line with the baseline and Dropout models. Similarly, all Temporal Softmax models performed very close to baseline, and significantly worse on FGSM attacks than the standard TemporalMax model.

The same trends held true for the $\mathcal{L}_2$ attacks described in Table 3.5, with the SpatialMax and SpatiotemporalMax models displaying very high accuracy against PGD-CE attacks but random accuracy against PGD-T.

3.4 Flatness and Generalization Gap

We now present an approximate measure of the flatness of each model's loss surface and its generalization gap as given by the Fisher-Rao norm. Note that since the Fisher-Rao norm represents a lower bound on the generalization gap, a smaller Fisher-Rao norm should correlate with flatter minima and better generalization (and vice versa) (Liang et al., 2019).

In Table 3.6, we can see that Dropout massively reduces the Fisher-Rao norm from the baseline, whereas SpatialMax, TemporalMax, and SpatiotemporalMax models all greatly increased the Fisher-Rao norm. Intriguingly, Spatiotemporal Softmax models brought the norm down to Dropout

Table 3.6: Fisher-Rao norm for the best performing models.

Model	Fisher-Rao Norm
Baseline	31.8
Dropout 48 51	7.68
SpatialMax 48 51	117.8
TemporalMax 48 51	90.56
SpatiotemporalMax 48 51	175.4
Spatiotemporal Softmax 48 51 T=50	13.6
Spatiotemporal Softmax 48 51 T=80	10.92
Temporal Softmax 48 51 (T=10)	10.68
Temporal Softmax 48 51 (T=50)	9.4
Temporal Softmax 48 51 (T=100)	4.2
Temporal Softmax 48 51 (T=150)	7.14

levels, whilst Temporal Softmax models cut the norm down to nearly half that of Dropout in the case of the Temporal Softmax (T=100) model. Note that this is the same model which achieved the highest test accuracy.

Lastly, we also show the test loss for the best performing models. This result is important for our analysis of the generalization gap. In Figure 3.11, we show test loss on a logarithmic scale and with data smoothing)i.e. exponentially weighted moving averages) for better readability. We can see from Figure 3.11 that the SpatiotemporalMax model failed to reduce the test loss, instead increasing it from the original starting point. SpatialMax and TemporalMax models also failed to properly minimize the test loss. In contrast, all models using a Softmax reduced the loss to near Dropout levels. Note, however, that even amongst these models, Spatiotemporal Softmax reduced the loss the least.

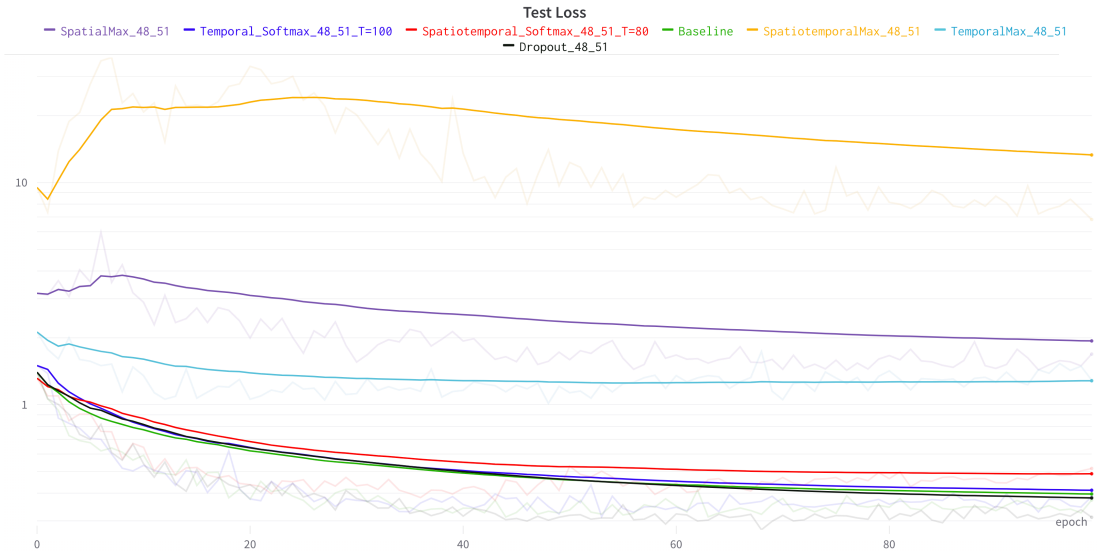

Figure 3.11: Test loss across models.

4

Discussion

4.1 Synaptic Failure Improves Dropout

In this work we modeled synaptic failure as a version of Dropout. Instead of deciding keep probabilities using a constant function, we designed a variety of possible functions which belong to a broad class we termed NormOut. Our results show that NormOut is consistently a net improvement on a baseline VGG-16 model applied to CIFAR-10. This alone is not particularly noteworthy, since it

has long been known that Dropout (of which NormOut is a variant) produces net improvements in VGG-16. However, we found that the Temporal Softmax (T=100) model actually outperforms Dropout to a somewhat significant degree, producing an improvement comparable to that induced by pre-training on ImageNet. To understand why this is, we must first look at why the other NormOut models performed as they did.

4.1.1 The Degree of Competition is Crucial for Specialization

The TemporalMax model was essentially a worse version of Dropout and failed to produce any meaningfully different behaviour. There was initially hope that TemporalMax would incentivize neuronal specialization by dropping a neuron's activation unless it was close to its own maximal activation (i.e. drop it unless it is responding to the input it is best specialized for). However, it seems that basic divisive normalization across time does not produce enough competition for neurons to specialize.

In comparison, the SpatialMax model produced highly specialized neurons. In layer 48, for example, around 80 of the 4096 neurons actually fired, and even amongst those 80 there was a mean activation comparable with Dropout. However, the maximal activation was over 1000x greater than the mean, indicating that at least some neurons ahd become highly specialized and were responding strongly to some inputs. We can infer that this specialization was somewhat widespread, as Figure 3.5 shows that the maximum activation for the SpatialMax model was consistently high. This indicates that there are a number of specialized neurons, since the inputs are all different yet there are neurons responding with large activations. It would be next to impossible for this to be the same

neuron responding every time, since test accuracy was around 90, and one (or a handful) of neurons cannot achieve that accuracy on CIFAR-10. In SpatialMax models, the keep probabilities were decided through comparison to the maximal activation in the given layer. The very large and very rare maximal activations suggest that spatial divisive normalization produces high competition, driving many neurons down to 0, and concentrates gradient updates to only a few neurons which subsequently become specialized.

The SpatiotemporalMax model produced results which were more similar to SpatialMax than to TemporalMax, again suggesting that competition is higher between different neurons than it is for the same neuron competing with itself. This is perhaps explained by the different weight initializations that neurons in an ANN receive. The i^{th} neuron will receive some initial set of weights $W_{(i)}$. Upon seeing the i^{th} input, the neuron calculates the function $W_{(i)} \cdot X_{(i)} + b_{(i)}$. Similarly, the j^{th} neuron calculates $W_{(j)} \cdot X_{(i)} + b_{(j)}$. It would appear that, on average, the difference between $X_{(i)}$ and $X_{(j)}$ is smaller than the difference between $W_{(i)}$ and $W_{(j)}$, since the latter produces larger maximal activations. That is, randomly initialized weights should have higher variance than inputs from a dataset. However, we should remember that these comparisons are happening dynamically during training. Even after the first epoch, some weights will have received large updates, such that $|W_{(i)} - W_{(j)}| >> |X_{(i)} - X_{(j)}|$. This explains why we see greater competition when performing spatial normalization.

4.1.2 SoftMax at High Temperatures Beneficially Alters Competition

When we introduce the Softmax into our models, we find that accuracy generally increases. Softmax models produce activation statistics which look like dampened versions of their underlying normalization method. For example, in layer 48, Spatiotemporal Softmax produces relatively large activations with slightly lower maximal activations, variance, and sparsity than the Spatiotemporal-Max model. Most interesting of all is that the Softmax models produce sparsity levels which match the baseline model, contradicting our assumptions about higher sparsity being good for model performance.

Temporal Softmax (T=100) appears to uncover the promise of temporal normalization by reaching the highest test accuracy. This is because using a Softmax helps to distinguish between activations in terms of their absolute size, rather than just their relative size to one another. As a result, we see Temporal Softmax (T=100) performing extremely well despite activation statistics similar to the baseline. The model functions as a direct improvement to Dropout simply by being careful not to drop useful activations, which are assigned a high keep probability.

4.2 NormOut Induces Gradient Masking

Whilst our results on test accuracy are a pleasing validation of the idea that modeling synaptic failure can improve Dropout, our most interesting findings concern the relationship between synaptic failure and the loss landscape. This relationship first becomes apparent in the results of the adversarial robustness testing.

4.2.1 Gradient Masking

From the results, it is evidently clear that NormOut is performing gradient masking. We focus in this discussion on the SpatiotemporalMax model. This model achieved over 80% accuracy against FGSM and PGD-CE attacks, which would be well above state-of-the-art if a valid results. However, the random performance against PGD-T and the 0% accuracy against the SQUARE attack are telltale signs of gradient masking, as described in Chapter 1.

Intuition might say that NormOut is clearly obfuscating model gradients in some way. However, our results are exciting because NormOut is *provably not* performing gradient obfuscation. We can say this because we have formulated NormOut as simply one class of functions in a broader set of functions which determine keep probabilities. Dropout is one such function (i.e. the constant function which sets all keep probabilities to whatever input value p was passed). Dropout avoids gradient obfuscation because it is turned off at test time, when adversarial robustness testing occurs. This means that, at test time, the Dropout layers might as well not exist, meaning they cannot interfere with model gradients by shattering them or hiding them through stochasticity. NormOut is exactly the same – we turn it off at test time, so the NormOut layer practically have no effect during adversarial robustness testing. The only remaining possibility is that NormOut is contributing to an exploding gradients problem, which one might try to support by looking at the large maximal activations of the SpatialMax model. However, note that the SpatiotemporalMax model, which achieves the highest gradient masking performance, has maximal and mean activations comparable to the baseline model, which does not gradient mask at all.

4.2.2 NormOut is a Novel Gradient Mask

Gradient obfuscation is just one subset of strategies within gradient masking. As discussed in Chapter 1, gradient masking can also arise from a model explicitly defending itself by changing the loss landscape in order to fool the adversary into weak attack vectors. However, NormOut is not an explicit defense – in fact, it is not an adversarial defense at all, but rather an improved regularizer. Why, and how, do we see such strong gradient masking without obfuscation?

The only form of gradient masking which we have not ruled out is the flattening of the loss landscape. This works well against gradinet-based optimization attacks like FGSM and PGD-CE, but not against black-box atatcks like Square (which aligns with our results). In Table 3.6, we can see that the SpatiotemporalMax model actually has the largest Fisher-Rao norm of any tested model. This raises a number of questions. First, the Fisher-Rao norm is supposedly a measure of flatness, which correlates well with generalization. We have good evidence to believe that this holds, namely that the Dropout model has a much smaller norm than baseline, and the Temporal Softmax (T=100) model, has the smallest norm of all. However, the fact that the SpatiotemporalMax's Fisher-Rao norm is many multiples of the baseline's is curious given that the former outperforms the latter in terms of test accuracy. Strikingly, the very large norm of the SpatiotemporalMax model contradicts the notion that it is flattening minima in order to mask gradient information.

This leads us to conclude that the SpatiotemporalMax model is performing an as yet unidentified form of gradient masking. We speculate that this might happen by divorcing the loss from the model's accuracy. We can see in Figure 3.11 that the SpatiotemporalMax model barely reduces the

test loss at all. We think this is because CIFAR-10 is a relatively simple dataset and VGG-16 is a deep, complex network. The model is able to find a "solution" to the classification problem which is very far away from the best possible solution, which we would expect to lie in a wide, deep valley. Consequently, a gradient-based optimization attack will fail because every direction will look equally "bad" to the adversary, insofar as the test loss is already very high, and moving in any direction will not change this much. Alternatively, it could be that the extreme curvature of the loss landscape acts like a maze to the adversary; as soon as it takes one step in the direction of steepest initial gradient, it finds itself presented with many more very sharp gradients, possibly in different directions to the step it has just taken. The adversary's own optimization scheme takes it further and further along this maze, but ultimately, going from a very high loss to a very very high loss has little impact on the model's classification performance.

However, none of this explain why the SpatiotemporalMax model ends up at this set of minima, nor why its loss landscape has so much curvature. These are open questions which remain for future work.

4.3 Future Work

There are an enormous number of avenues for follow-up work to this thesis. Some of this work is already being carried out, though the results were too early to include.

4.3.1 What Kind of Gradient Mask is NormOut?

Perhaps the biggest outstanding question from this work is exactly what the SpatiotemporalMax model is doing to introduce such strong gradient masking. We are currently testing the combination of adversarial training (which is known to improve gradient masking) (Madry et al., 2017) with NormOut to see if some of the gradient maskign effects can be dampened whilst retaining good performance against FGSM and PGD-CE.

Measures of Flatness

It will be very important to understand the loss landscape in more detail. We only used the Fisher-Rao norm here to measure flatness. In future work, we hope to visualize the landscape, as well as use other metrics like the trace of the Hessian.

The adversarial robustness literature has mostly discarded the FGSM attack due to how easily it can be fooled by gradient masking, as we have shown in this thesis. However, it would be interesting to see how well the attack could be repurposed as a flatness measure, given that its failure rate directly correlates with the flatness of the learned minima.

4.3.2 NormOut and Other Forms of Regularization

There are a number of other forms of regularization with which we would like to combine NormOut in future work. First, we would like to explore the addition of L1 and L2 regularization. L1 regularization penalizes the sum of the absolute value of the weights by adding a fixed term to the

loss function. In practice, this leads to high sparsity, as well as smaller weights. Perhaps L1 regularization could increase the relatively low sparsity of the Temporal Softmax (T=100) model, thus improving it even further. L2 regularization, on the other hand, penalizes the sum of the squares of the weights. In practice, it forces the model to constrain the sizes of its weights, which could potentially be beneficial to forms of NormOut which rely on spatial normalization. It would be informative to see how a SpatialMax model changes its maximal activation if large weights are heavily penalized.

Recent work has shown that either introducing or removing Dropout after some number of epochs can improve underfitting in Transformer models (Liu et al., 2023). Initial testing suggests that introducing SpatialMax for the first 5 epochs and then removing it can be beneficial for test accuracy whilst avoiding gradient masking. This could be because SpatialMax introduces a strong specialization prior which is used early in training but then removed to prevent over-specialization.

4.3.3 Other Architectures and Datasets

The Transformer models from (Liu et al., 2023) are the prime candidates in which to test NormOut next, since (1) we can again simply swap out the Dropout layers for the most direct comparison possible, (2) the Transformer is the most widely studied and best performing architecture today, and (3) the models are trained on ImageNet-1K, not CIFAR-10. ImageNet-1K is a subset of ImageNet (Russakovsky et al., 2015) which is much more complex than CIFAR-10 since it contains images from 1000 classes rather than just 10. This could be a particularly important test for the SpatiotemporalMax model, since it will be difficult to maintain both a high test loss and test accuracy on a more complex dataset.

4.4 Conclusion

Overall, our results suggest a clear relationship between NormOut and the loss landscape. We have already discussed why the Temporal Softmax (T=100) model was the most successful from a deep learning perspective. However, returning to neuroscience, we can look at this model as perhaps the purest instantiation of synaptic failure. This is because whilst we do not know exactly why synaptic failure occurs, we do know that its defining characteristic is that weaker synapses fail more frequently. Additionally, a synapse is less likely to fail the greater the summed potentials are above the given neuron's activation threshold. Under TemporalMax, we directly tie the failure rate of a neuron to its own activity over time. Whilst there is no biological analogy for the Softmax, this is simply a necessary mathematical addition which makes our keep probability function more expressive and thus makes up for all the simplifications we are making in using an ANN. In nearly all the figures presented in Chapter 3, we see that NormOut changes the model's behaviour early in training as compared to the baseline and Dropout. The fact that simply tying keep probabilities to activation strength can have such profound effects, for example like beating Dropout's test accuracy, or producing massive gradient masking, is evidence from deep learning that synaptic failure might exist for a reason. In particular, the fact that our best version of NormOut found an extremely flat minima, with a Fisher-Rao norm nearly half that of Dropout, suggests that synaptic failure is deeply tied to learning and generalization. In addition to the known biological arguments presented in Chapter 1, we think this is reason to believe synaptic failure may be a feature, and not a bug, of the brain.

References

Allen, C. & Stevens, C. F. (1994). An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Sciences*, 91(22), 10380–10383.

Andersen, P., Eccles, J., & Voorhoeve, P. (1964). Postsynaptic inhibition of cerebellar purkinje cells. *Journal of neurophysiology*, 27(6), 1138–1153.

Andersen, P., Gross, G. N., Lomo, T., & Sveen, O. (1969). Participation of inhibitory and excitatory interneurones in the control of hippocampal cortical output. In *UCLA forum in medical sciences*, volume 11 (pp. 415–465).

Andriushchenko, M., Croce, F., Flammarion, N., & Hein, M. (2020). Square attack: a query-efficient black-box adversarial attack via random search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII* (pp. 484–501).: Springer.

Athalye, A., Carlini, N., & Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning* (pp. 274–283).: PMLR.

Baldi, P. & Sadowski, P. J. (2013). Understanding dropout. *Advances in neural information processing systems*, 26.

Barrett, D., Hill, F., Santoro, A., Morcos, A., & Lillicrap, T. (2018). Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning* (pp. 511–520).: PMLR.

Barth, A. L. & Poulet, J. F. (2012). Experimental evidence for sparse firing in the neocortex. *Trends in neurosciences*, 35(6), 345–355.

Branco, T. & Staras, K. (2009). The probability of neurotransmitter release: variability and feedback control at single synapses. *Nature Reviews Neuroscience*, 10(5), 373–383.

Brown, A. M., Arancillo, M., Lin, T., Catt, D. R., Zhou, J., Lackey, E. P., Stay, T. L., Zuo, Z., White, J. J., & Sillitoe, R. V. (2019). Molecular layer interneurons shape the spike activity of cerebellar purkinje cells. *Scientific reports*, 9(1), 1–19.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877–1901.

Carandini, M. & Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1), 51–62.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., & Zecchina, R. (2019). Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12), 124018.

Cook, P. B. & McReynolds, J. S. (1998). Lateral inhibition in the inner retina is important for spatial tuning of ganglion cells. *Nature neuroscience*, 1(8), 714–719.

Del Castillo, J. & Katz, B. (1954). Quantal components of the end-plate potential. *The Journal of physiology*, 124(3), 560.

Dizon, M. J. & Khodakhah, K. (2011). The role of interneurons in shaping purkinje cell responses in the cerebellar cortex. *Journal of Neuroscience*, 31(29), 10463–10473.

D'Angelo, E. (2008). The critical role of golgi cells in regulating spatio-temporal integration and plasticity at the cerebellum input stage. *Frontiers in neuroscience*, 2, 8.

Eccles, J. (1965). Functional meaning of the patterns of synaptic connections in the cerebellum. *Perspectives in biology and medicine*, 8(3), 289–310.

Eccles, J., Ito, M., & Szentagothai, J. (1967). The cerebellum as a neuronal machine springer-verlag. *New York*, (pp. 335).

Ehlis, A.-C., Ringel, T., Plichta, M., Richter, M., Herrmann, M., & Fallgatter, A. (2009). Cortical correlates of auditory sensory gating: a simultaneous near-infrared spectroscopy event-related potential study. *Neuroscience*, 159(3), 1032–1043.

Ferguson, K. A. & Cardin, J. A. (2020). Mechanisms underlying gain modulation in the cortex. *Nature Reviews Neuroscience*, 21(2), 80–92.

Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Giuste, F. O. & Vizcarra, J. C. (2020). CIFAR-10 image classification using feature ensembles. *CoRR*, abs/2002.03846.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Grossberg, S. (1982). Contour enhancement, short term memory, and constancies in reverberating neural networks. In *Studies of mind and brain* (pp. 332–378). Springer.

Häusser, M. & Clark, B. A. (1997). Tonic synaptic inhibition modulates neuronal output pattern and spatiotemporal synaptic integration. *Neuron*, 19(3), 665–678.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S. & Schmidhuber, J. (1997). Flat minima. *Neural computation*, 9(1), 1–42.

Holler, S., Köstinger, G., Martin, K. A., Schuhknecht, G. F., & Stratford, K. J. (2021). Structure and function of a neocortical synapse. *Nature*, 591(7848), 111–116.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359–366.

Jayakumar, S. M., Czarnecki, W. M., Menick, J., Schwarz, J., Rae, J., Osindero, S., Teh, Y. W., Harley, T., & Pascanu, R. (2020). Multiplicative interactions and where to find them.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *Ars Journal*, 30(10), 947–954.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Liang, T., Poggio, T., Rakhlin, A., & Stokes, J. (2019). Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd international conference on artificial intelligence and statistics* (pp. 888–896).: PMLR.

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. PhD thesis, Master's Thesis (in Finnish), Univ. Helsinki.

Liu, Z., Xu, Z., Jin, J., Shen, Z., & Darrell, T. (2023). Dropout reduces underfitting.

Louie, K., Grattan, L. E., & Glimcher, P. W. (2011). Reward value-based gain control: Divisive normalization in parietal cortex. *Journal of Neuroscience*, 31(29), 10627–10639.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mittmann, W., Koch, U., & Häusser, M. (2005). Feed-forward inhibition shapes the spike output of cerebellar purkinje cells. *The Journal of physiology*, 563(2), 369–378.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12), 124003.

Oldfield, C. S., Marty, A., & Stell, B. M. (2010). Interneurons of the cerebellar cortex toggle purkinje cells between up and down states. *Proceedings of the National Academy of Sciences*, 107(29), 13153–13158.

Olshausen, B. A. & Field, D. J. (2004). Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4), 481–487.

Panousis, K., Chatzis, S., Alexos, A., & Theodoridis, S. (2021). Local competition and stochasticity for adversarial robustness in deep learning. In *International Conference on Artificial Intelligence and Statistics* (pp. 3862–3870).: PMLR.

Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y.-L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., Gierke, W., Dong, Y., Berthelot, D., Hendricks, P., Rauber, J., & Long, R. (2018). Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security* (pp. 506–519).

Rao-Ruiz, P., Yu, J., Kushner, S. A., & Josselyn, S. A. (2019). Neuronal competition: microcircuit mechanisms define the sparsity of the engram. *Current opinion in neurobiology*, 54, 163–170.

Ribrault, C., Sekimoto, K., & Triller, A. (2011). From the stochasticity of molecular processes to the variability of synaptic transmission. *Nature Reviews Neuroscience*, 12(7), 375–387.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5), 465–471.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115, 211–252.

Shanehsazzadeh, A., Bachas, S., Kasun, G., Sutton, J. M., Steiger, A. K., Shuai, R., Kohnert, C., Morehead, A., Brown, A., Chung, C., et al. (2023). Unlocking de novo antibody design with generative artificial intelligence. *bioRxiv*, (pp. 2023–01).

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676), 354–359.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N. (2013). Improving neural networks with dropout. *University of Toronto*, 182(566), 7.

Sun, X., Zhang, Z., Ren, X., Luo, R., & Li, L. (2021). Exploring the vulnerability of deep neural networks: A study of parameter corruption. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35 (pp. 11648–11656).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Verma, G. & Swami, A. (2019). Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32, 8646–8656.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International conference on machine learning* (pp. 1058–1066).: PMLR.

Wu, D., Xia, S.-T., & Wang, Y. (2020). Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33, 2958–2969.

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., & Mahoney, M. W. (2018). Hessian-based analysis of large batch training and robustness to adversaries. *Advances in Neural Information Processing Systems*, 31.

Zheng, Y., Zhang, R., & Mao, Y. (2021). Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8156–8165).