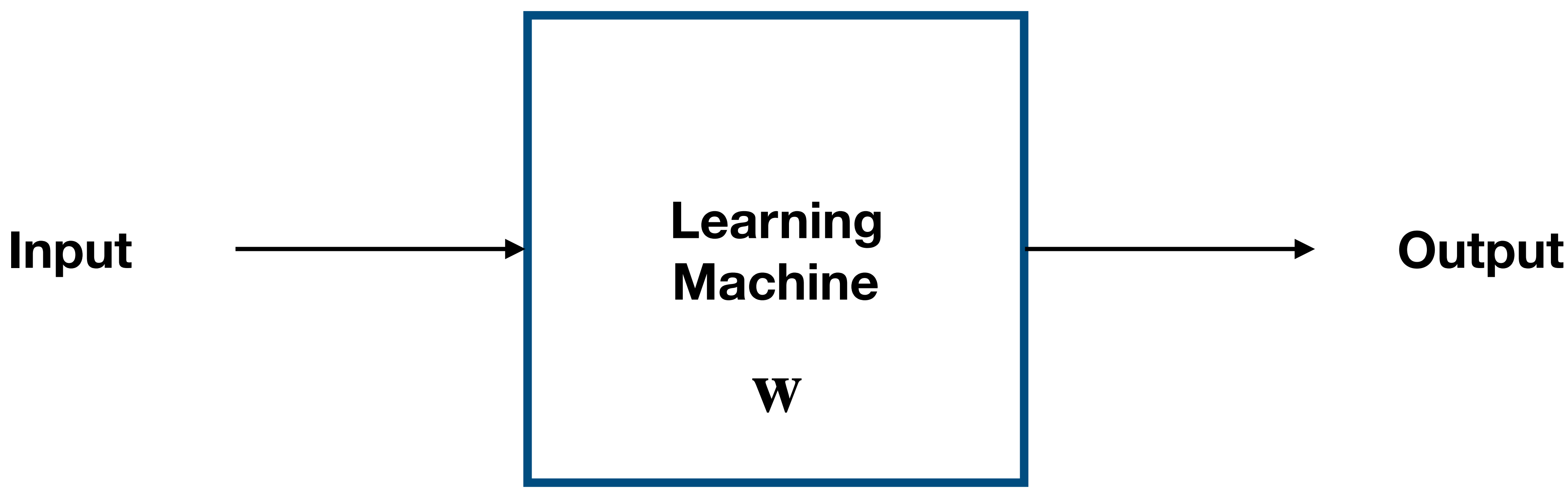
Harvard John A. Paulson
School of Engineering
and Applied Sciences

Why doesn’t the brain overfit?

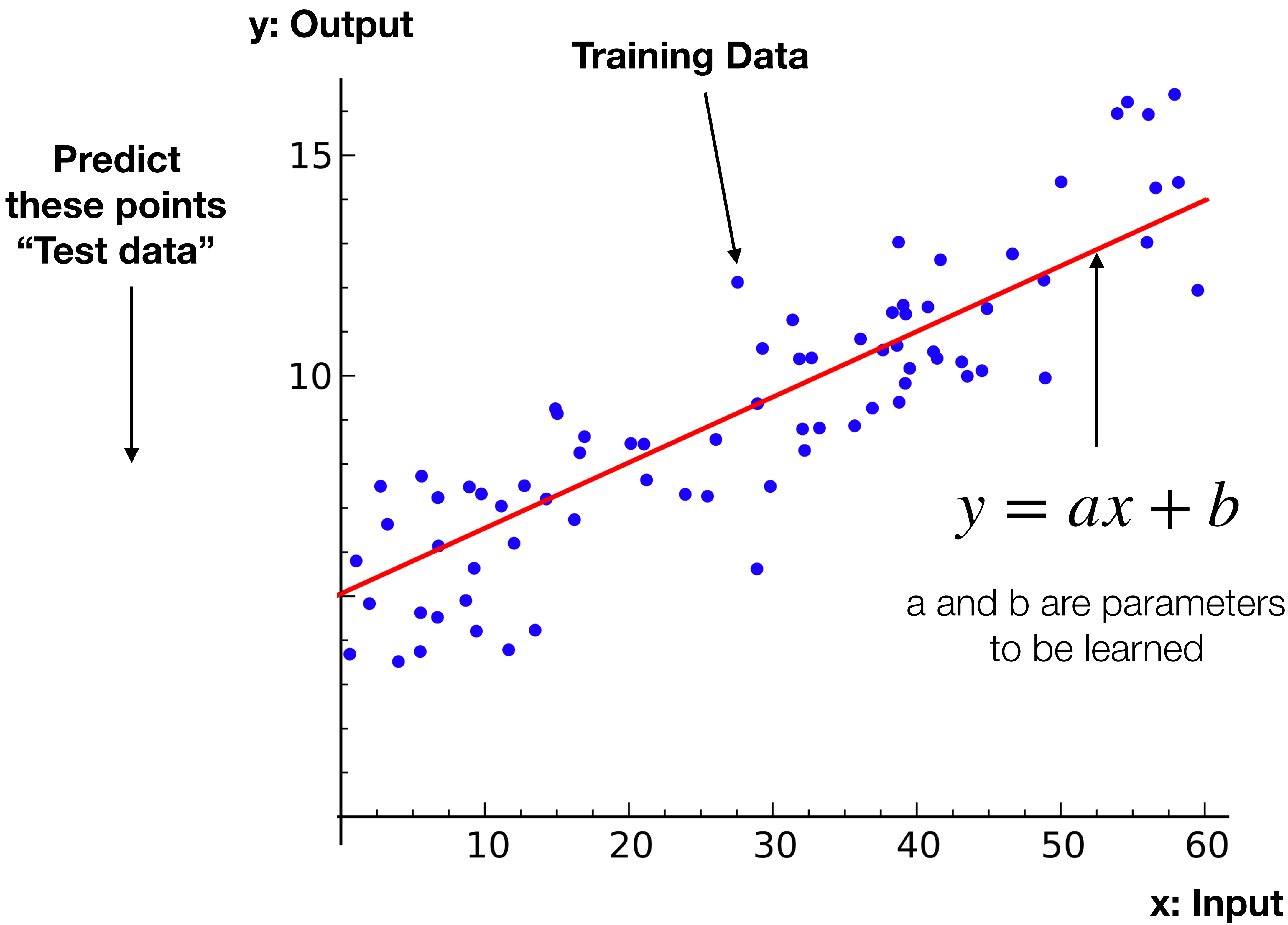
Cengiz Pehlevan
Assistant Professor of Applied Mathematics

Learning machines

Learning machines, natural or artificial, find statistical patterns in data that generalize to previously unseen samples.



$\mathbf{w}$:   **“Parameters” to be learned from data**


y: Output
Training Data
Predict
these points
“Test data”
15
10
$y = ax + b$
a and b are parameters
to be learned
10
20
30
40
50
60
x: Input

How do we find/learn the parameters of the learning machine?

How well does the machine predict?

$$y = ax + b \qquad y = ax^2 + bx + c \qquad y = ax^{15} + \ldots$$

1. More data is better
2. Too many parameters is not good

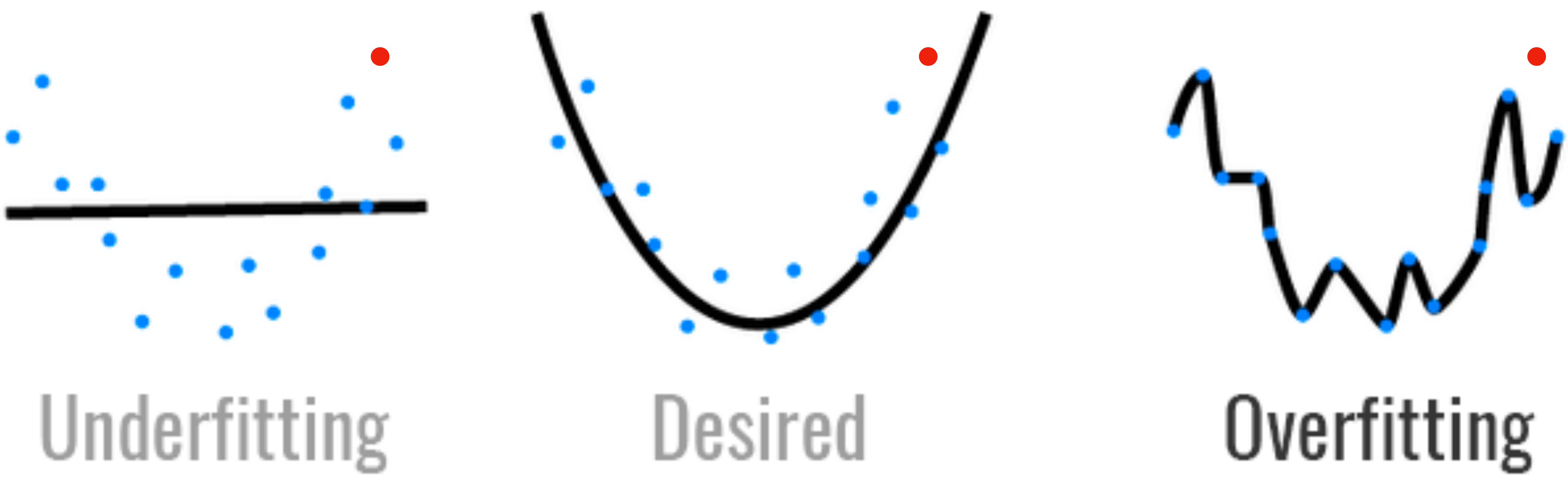
Figure Reference: https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

Classical wisdom: Overparametrization = Overfitting

$y = ax + b$ $\qquad$ $y = ax^2 + bx + c$ $\qquad$ $y = ax^{15} + \ldots$

Figure Reference: https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

Classical wisdom: Overparametrization = Overfitting



$y = ax + b$

$y = ax^2 + bx + c$

$y = ax^{15} + \ldots$

High training error
High test error

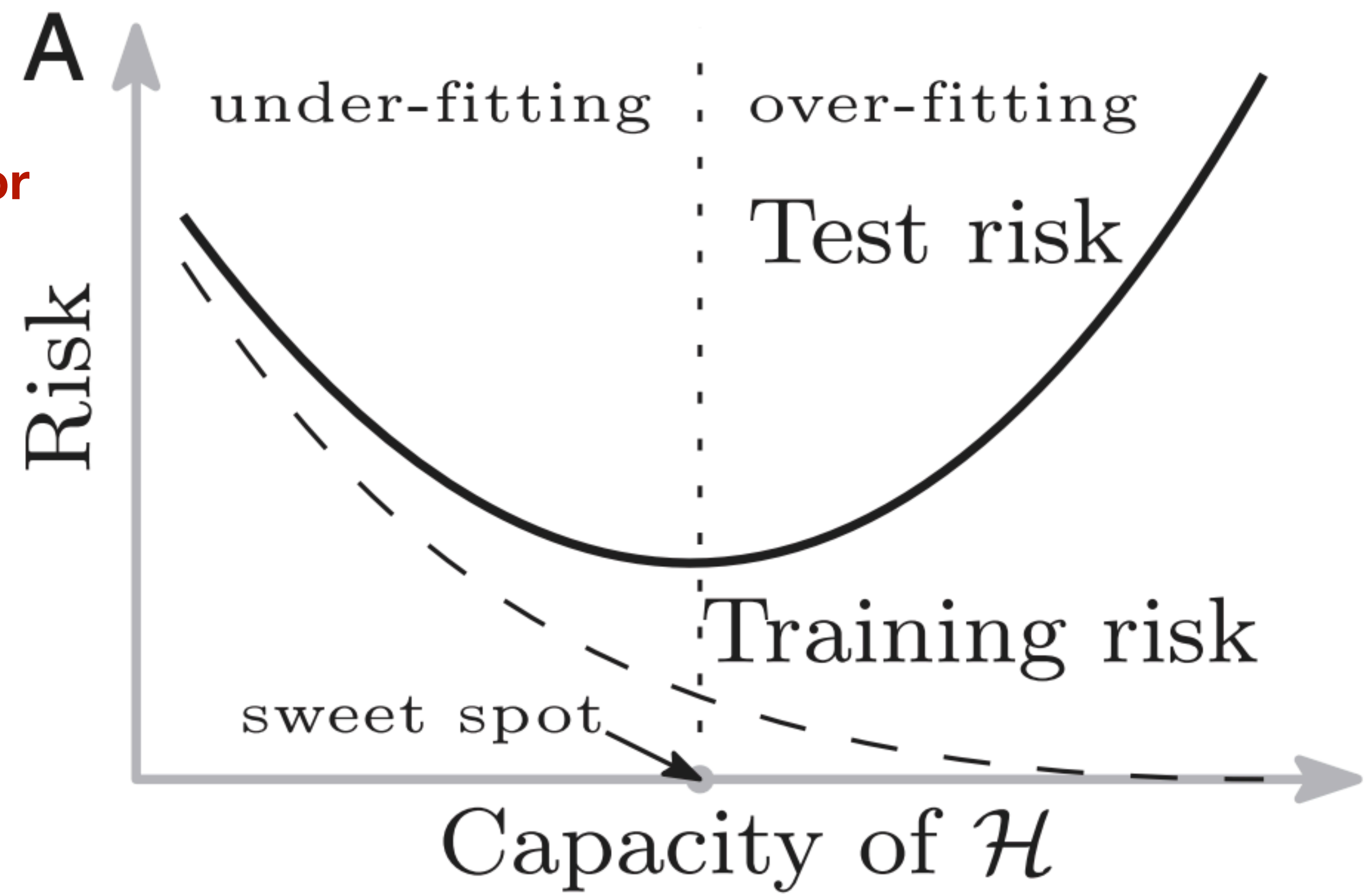
Sweet spot

Zero training error
High test error

Figure Reference: https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

Main result of classical statistical learning theory

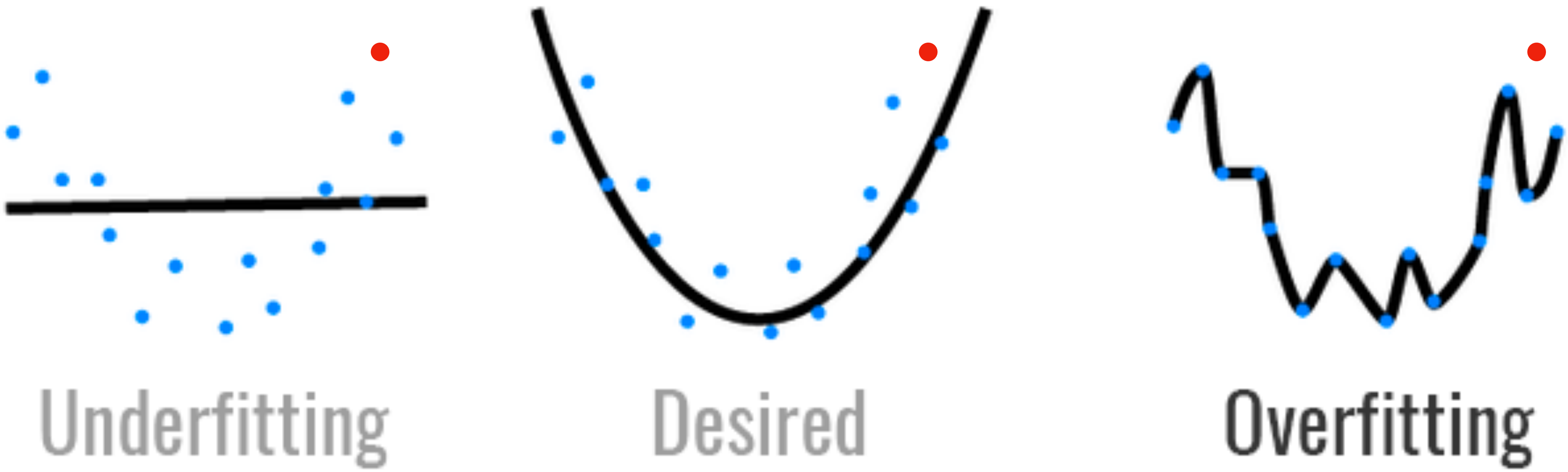
Risk = Error



# of parameters

Figure reference: Belkin et al., PNAS 2019

Classical wisdom: Overparametrization = Overfitting



$y = ax + b$

$y = ax^2 + bx + c$

$y = ax^{15} + \ldots$

High training error
High test error

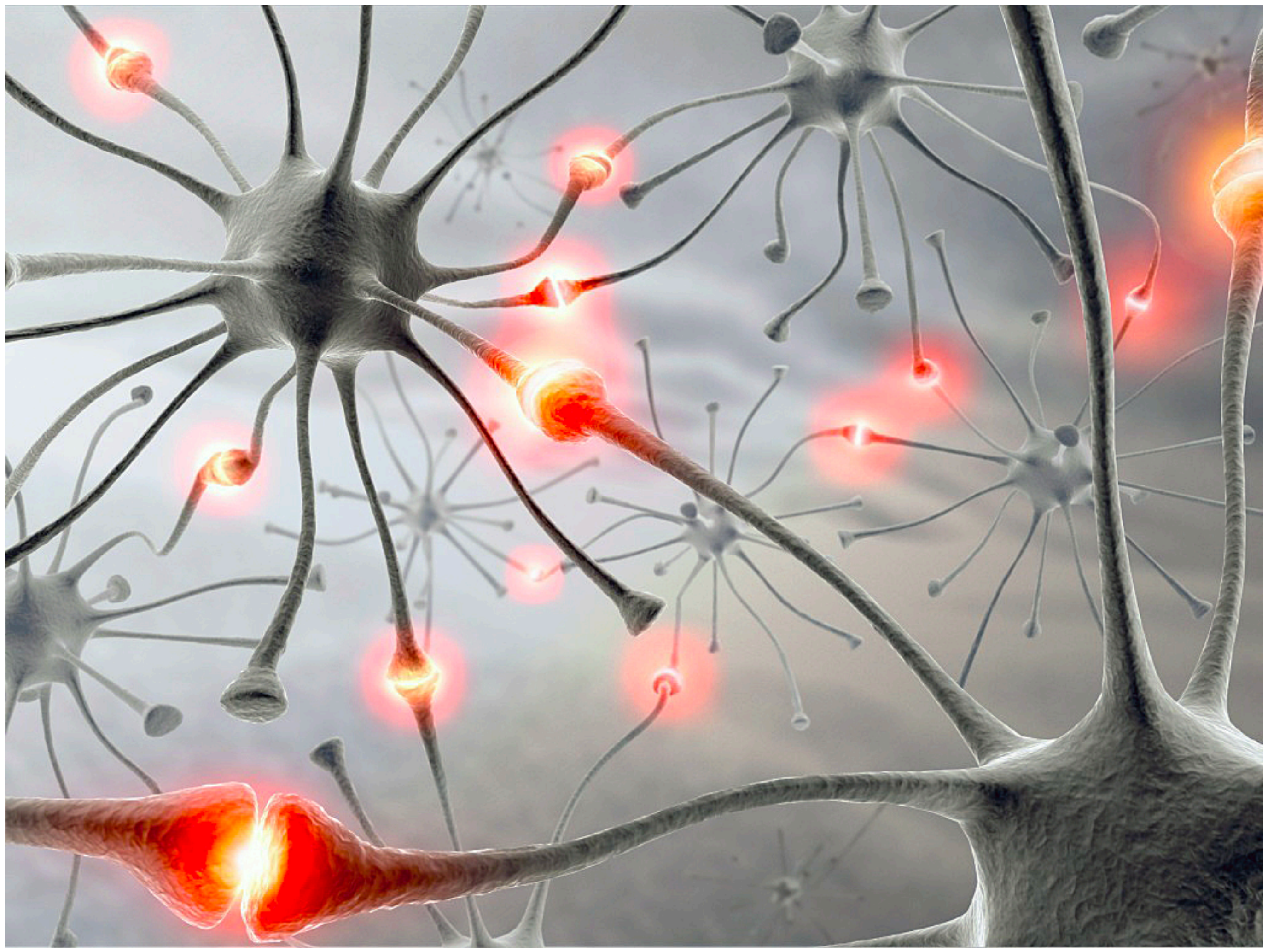
Sweet spot

Zero training error
High test error

Figure Reference: https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

Why doesn't the brain overfit?

“Parameters” of the brain



10^{11} neurons
10^{14} synapses

What changes as one learns?

Rapid formation and selective stabilization of synapses for enduring motor memories

Tonghui Xu[1]*, Xinzhu Yu[1]*, Andrew J. Perlik[1], Willie F. Tobin[1], Jonathan A. Zweig[1], Kelly Tennant[2], Theresa Jones[2] & Yi Zuo[1]

Rapid formation and selective stabilization of synapses for enduring motor memories

Tonghui Xu[1]*, Xinzhu Yu[1]*, Andrew J. Perlik[1], Willie F. Tobin[1], Jonathan A. Zweig[1], Kelly Tennant[2], Theresa Jones[2] & Yi Zuo[1]

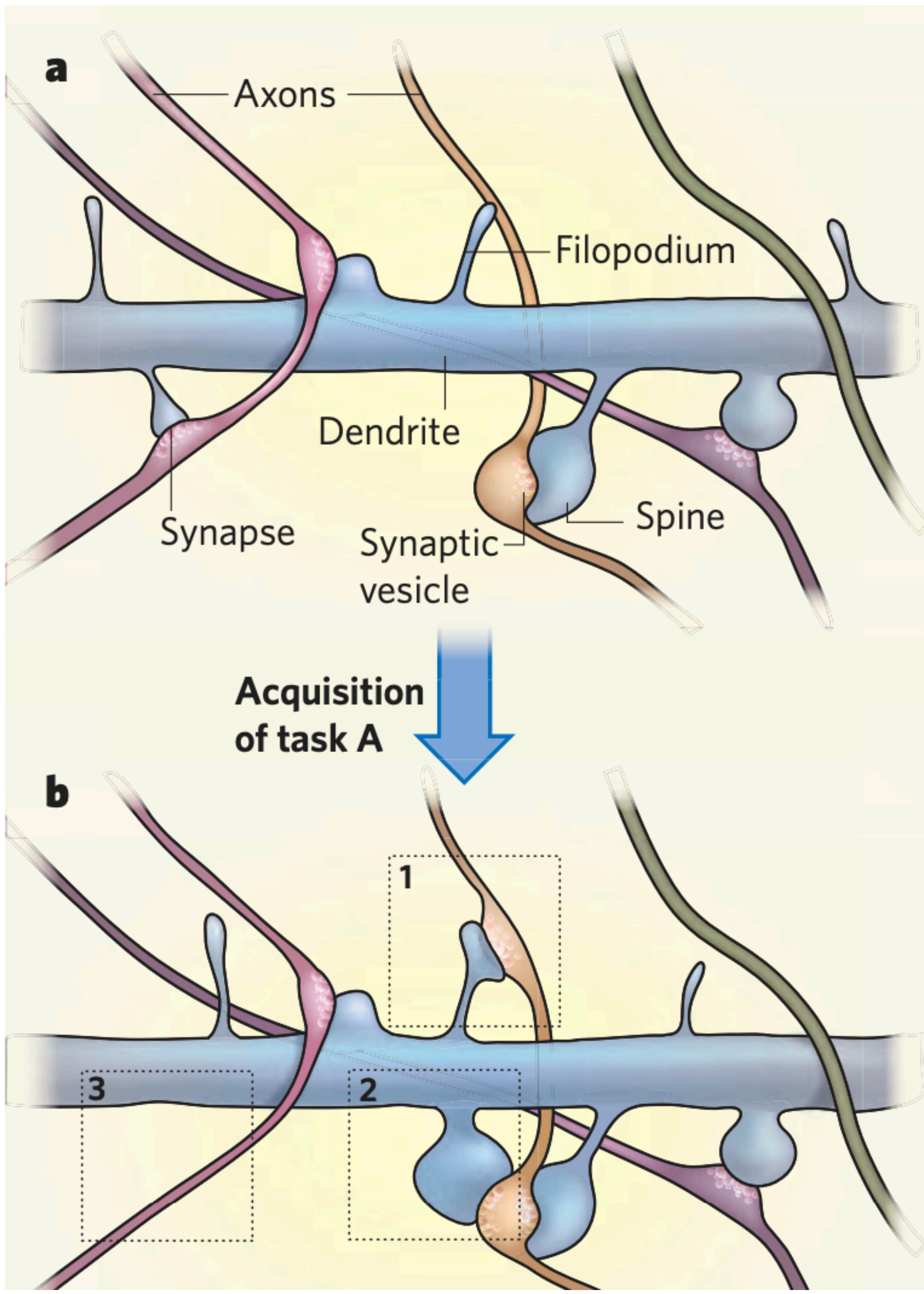
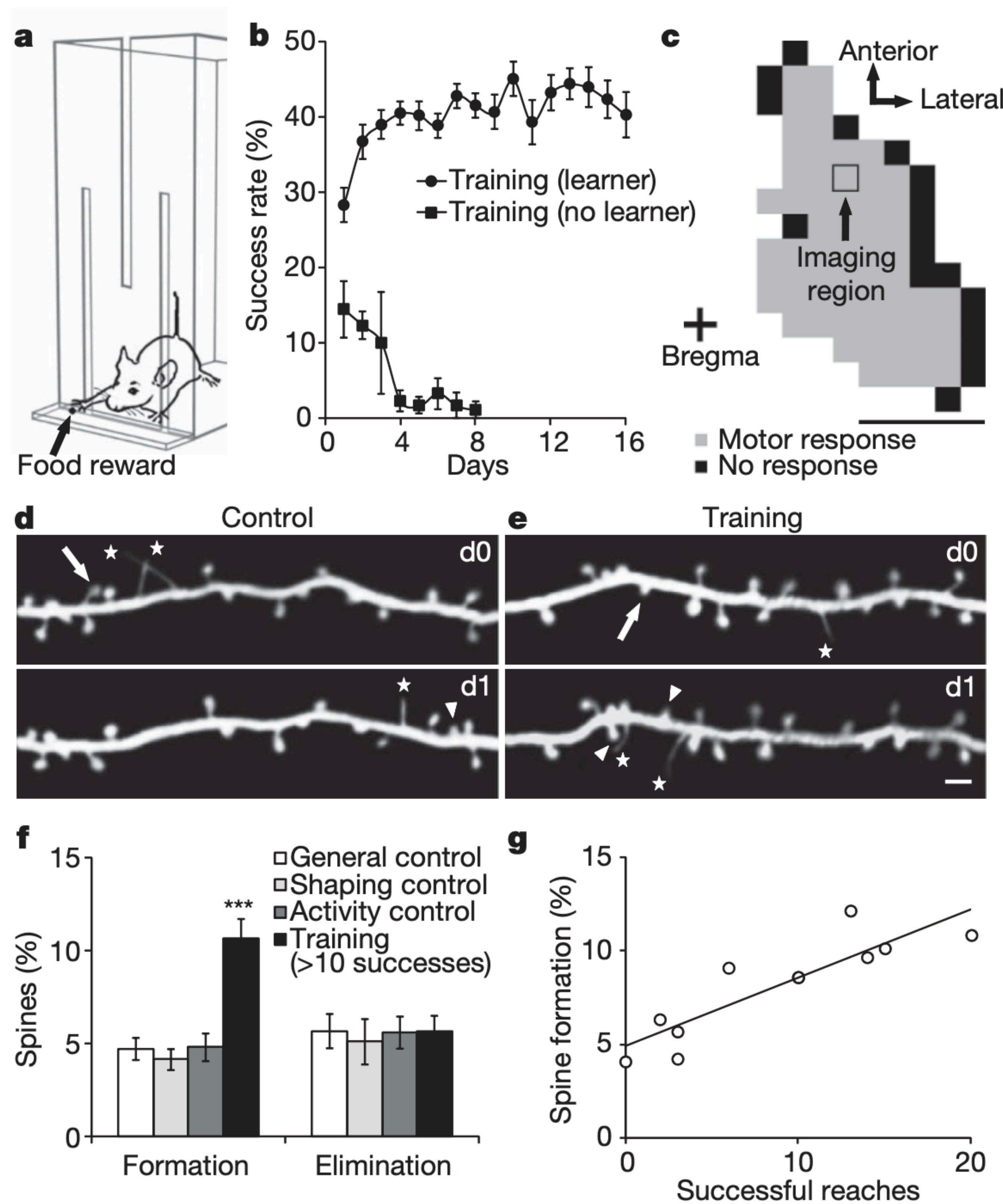
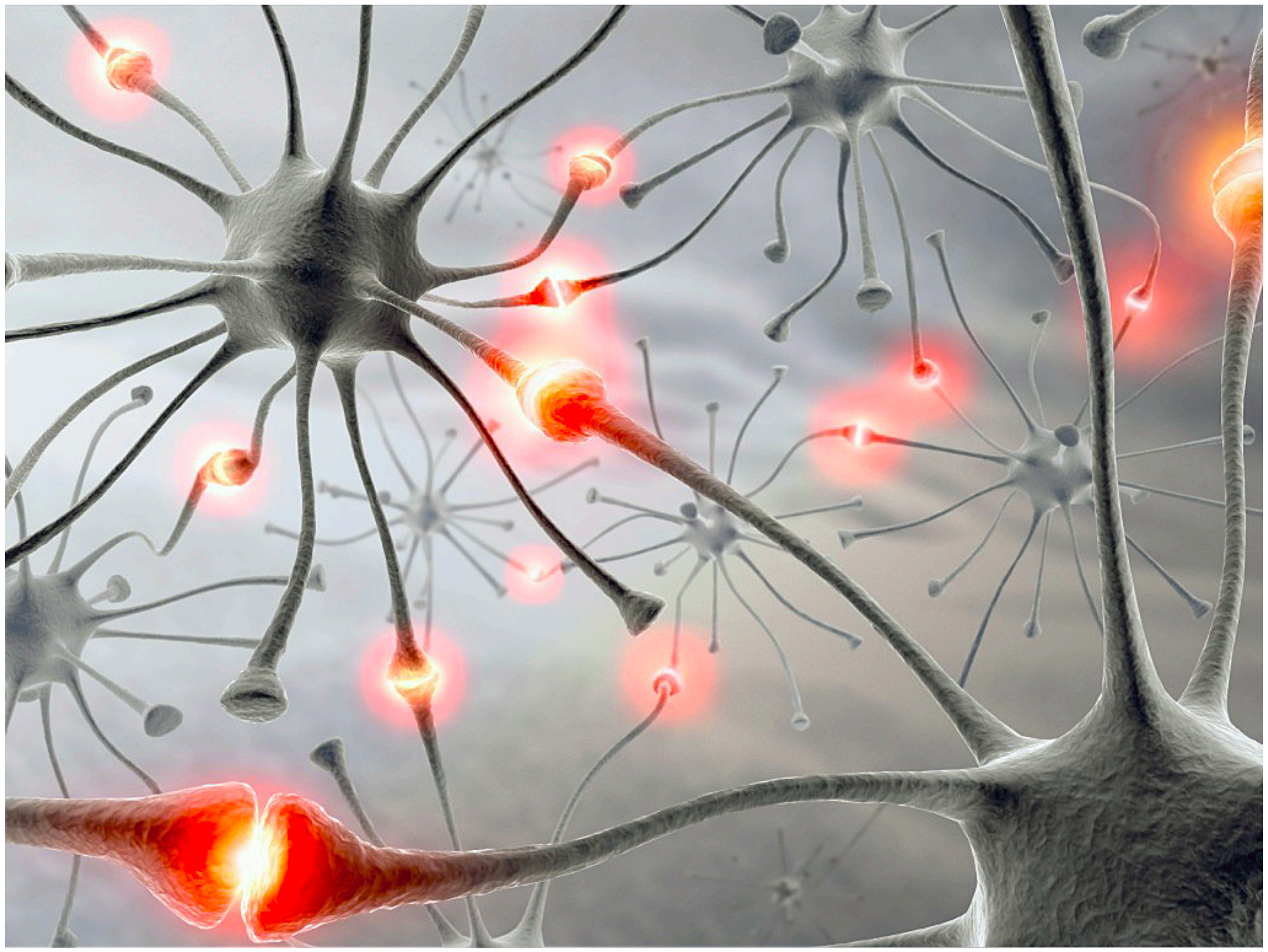


Figure reference:
Ziv and Ahissar, 2009



“Parameters” of the brain



10^{11} neurons
10^{14} synapses

Do we have enough “data” to “fit” all the synaptic weights of our brains?

The brain is overparametrized

Geoffrey Hinton:
(Turing Medalist, “Godfather of Deep Learning”)

The brain has about 10^{14} synapses and we only live for about 10^9 seconds. So we have a lot more parameters than data.

This motivates the idea that we must do a lot of unsupervised learning since the perceptual input (including proprioception) is the only place we can get 10^5 *dimensions of constraint per second.*

(Reddit forum)

The brain is overparametrized

Is the missing information in our genome?

Anthony Zador:

The human genome has about 3×10^9 nucleotides, so it can encode no more than about 1 GB of information—an hour or so of streaming video. But the human brain has about 10^{11} neurons, and more than 10^3 synapses per neuron. Since specifying a connection target requires about $\log 10^{11} = 37$ bits/synapse, it would take about 3.7×10^{15} bits to specify all 10^{14} connections.

Zador, 2019

Why doesn’t the brain overfit?

Why don’t deep networks overfit?

Deep networks as models of brain function

PERSPECTIVE FOCUS ON NEURAL COMPUTATION AND THEORY

nature neuroscience

Using goal-driven deep learning models to understand sensory cortex

Daniel L K Yamins[1,2] & James J DiCarlo[1,2]

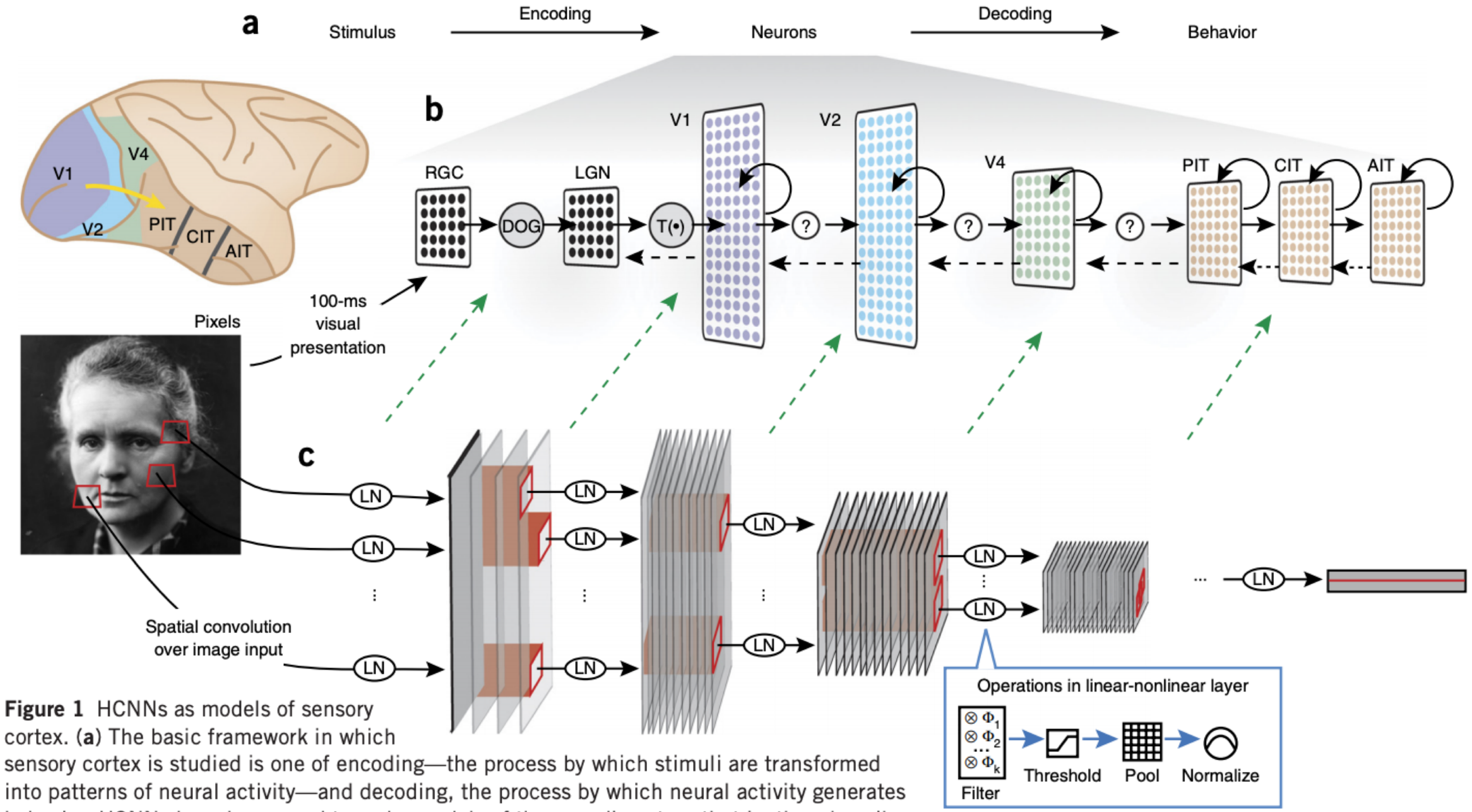
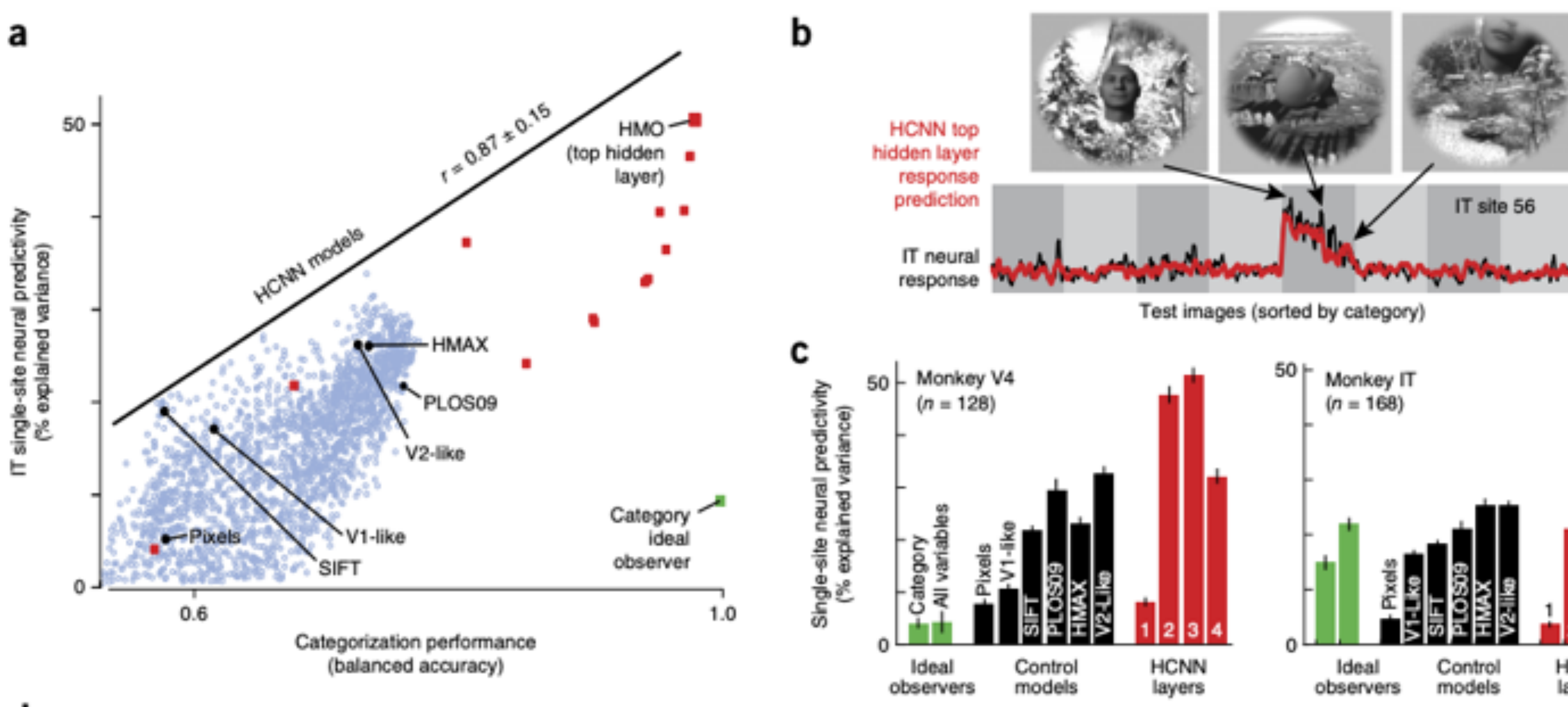


Figure 1 HCNNs as models of sensory cortex. (**a**) The basic framework in which sensory cortex is studied is one of encoding—the process by which stimuli are transformed into patterns of neural activity—and decoding, the process by which neural activity generates behavior. HCNNs have been used to make models of the encoding step; that is, they describe



Deep networks work well in the overparametrized regime

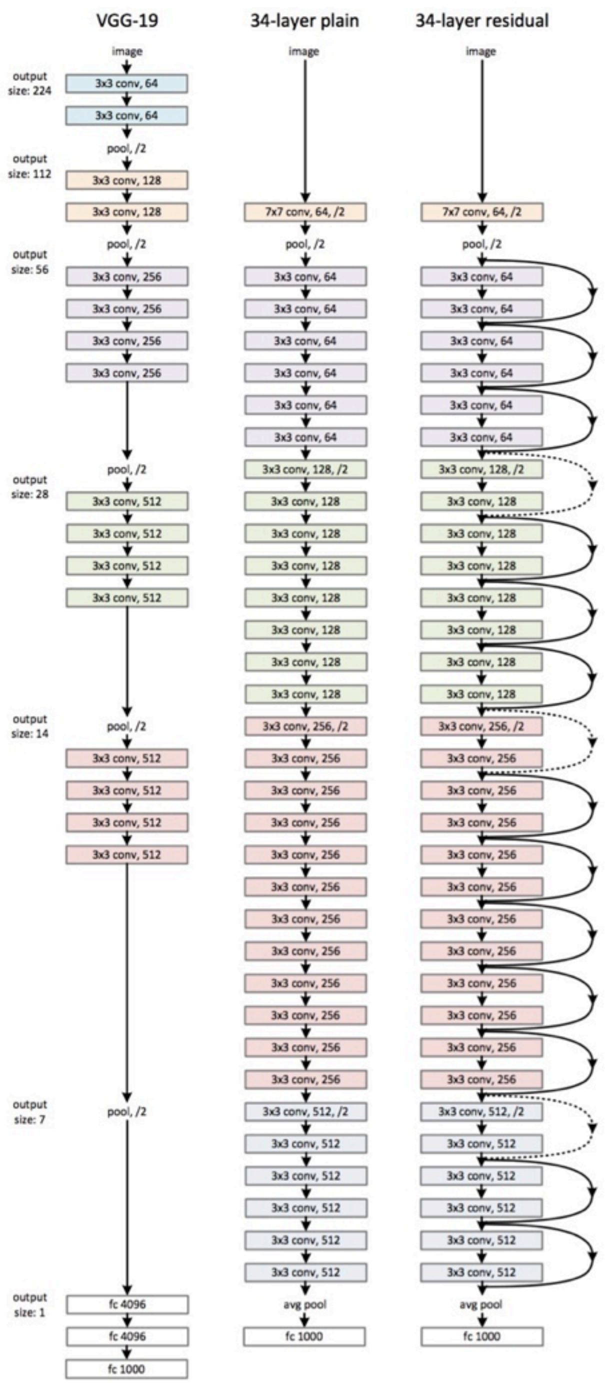


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Number of Layers	Number of Parameters
ResNet 18	11.174M
ResNet 34	21.282M
ResNet 50	23.521M
ResNet 101	42.513M
ResNet 152	58.157M

ImageNet dataset has ~1.2M images

Deep Double Descent: Where Bigger Models and More Data Hurt

Preetum Nakkiran*
Harvard University

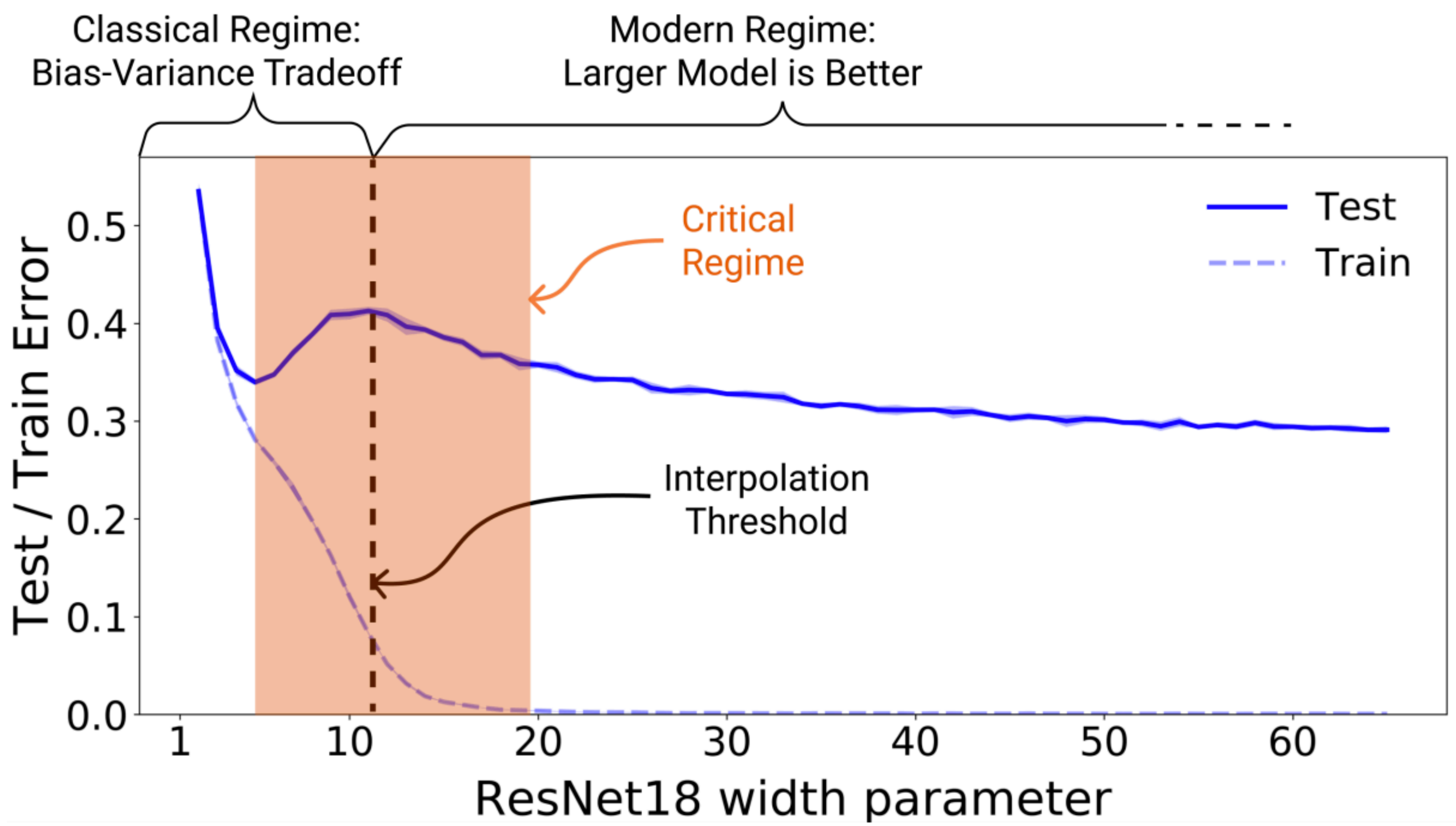
Gal Kaplun†
Harvard University

Yamini Bansal†
Harvard University

Tristan Yang
Harvard University

Boaz Barak
Harvard University

Ilya Sutskever
OpenAI



PNAS

Check for updates

Reconciling modern machine-learning practice and the classical bias–variance trade-off

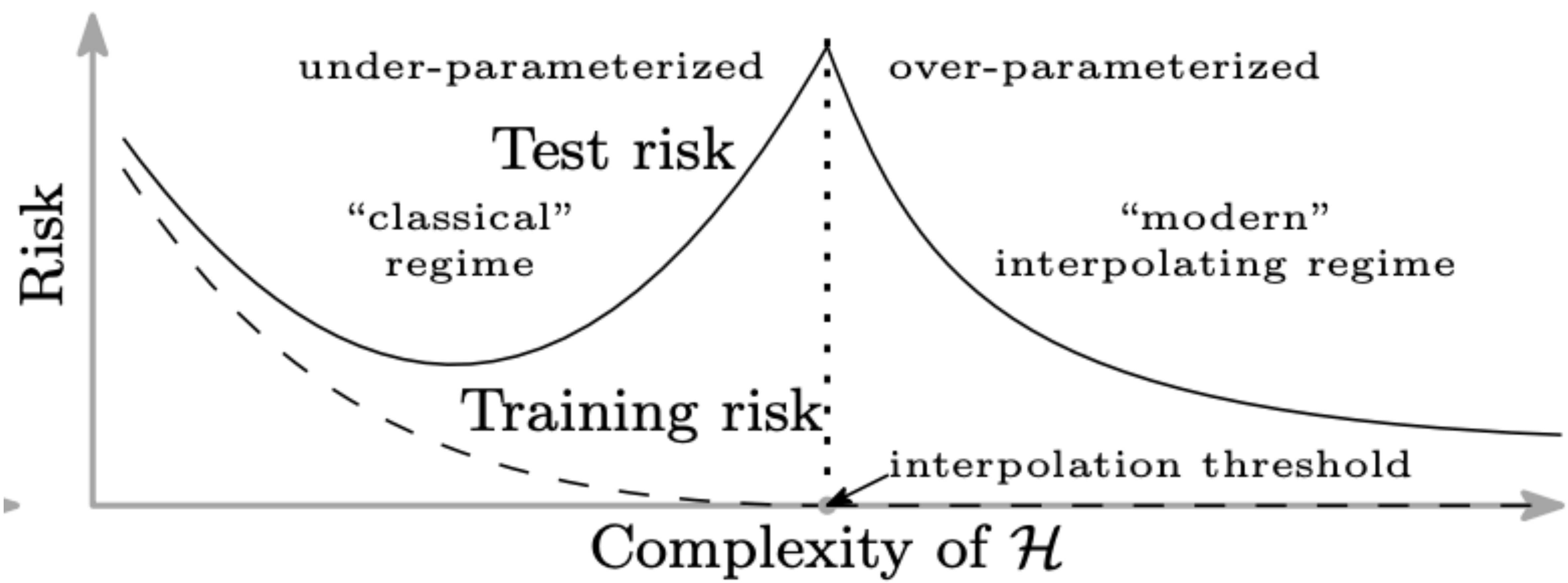
Mikhail Belkin[a,b,1], Daniel Hsu[c], Siyuan Ma[a], and Soumik Mandal[a]

[a]Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210; [b]Department of Statistics, The Ohio State University, Columbus, OH 43210; and [c]Computer Science Department and Data Science Institute, Columbia University, New York, NY 10027

Edited by Peter J. Bickel, University of California, Berkeley, CA, and approved July 2, 2019 (received for review February 21, 2019)

Breakthroughs in machine learning are rapidly changing science and society, yet our fundamental understanding of this technol-

ing data (i.e., have large empirical risk) and hence predict poorly on new data. 2) If $\mathcal{H}$ is too large, the empirical risk minimizer



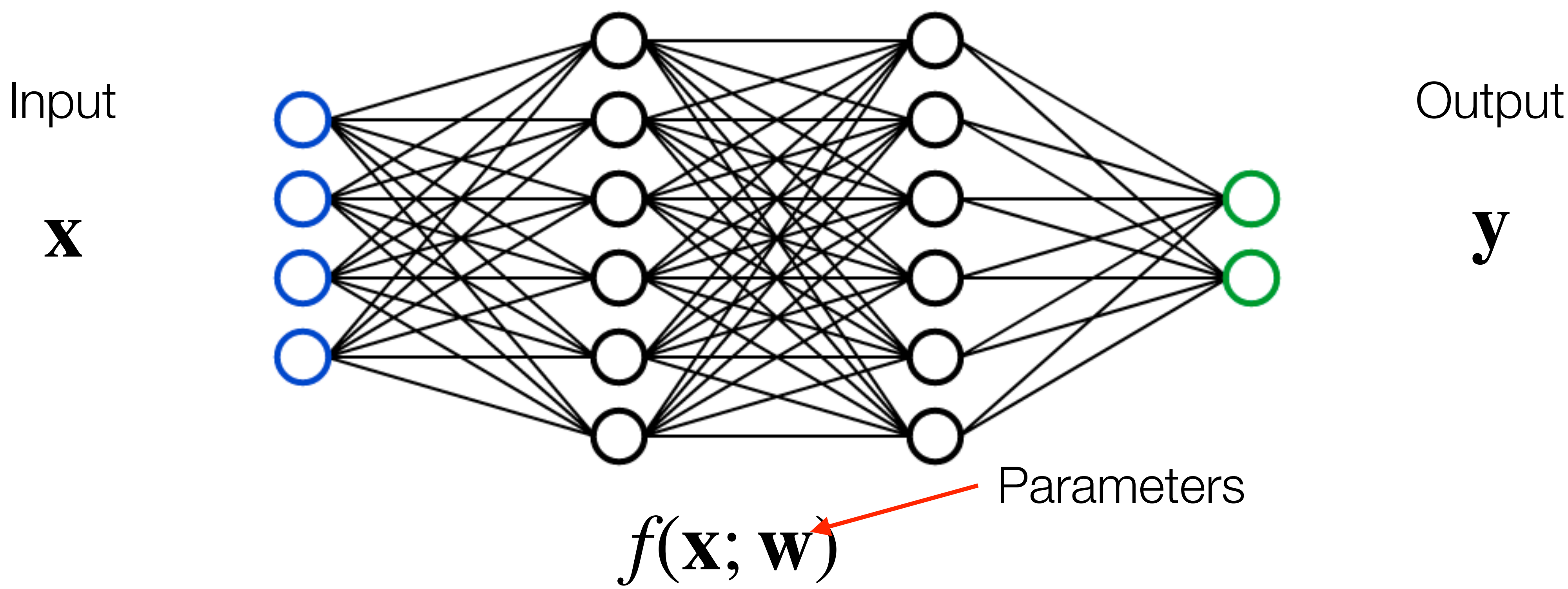
Read as # of parameters

Why?

Spectrum Dependent Learning Curves in Kernel Regression and Wide Neural Networks

Blake Bordelon [1] **Abdulkadir Canatar** [2] **Cengiz Pehlevan** [1,3]

Deep Learning and Generalization



$f_T(\mathbf{x})$: target function where (possibly noisy) training examples come from

Question: How many training examples do we need to learn a function?
Depends on network architecture, training algorithm and the nature of the target function.

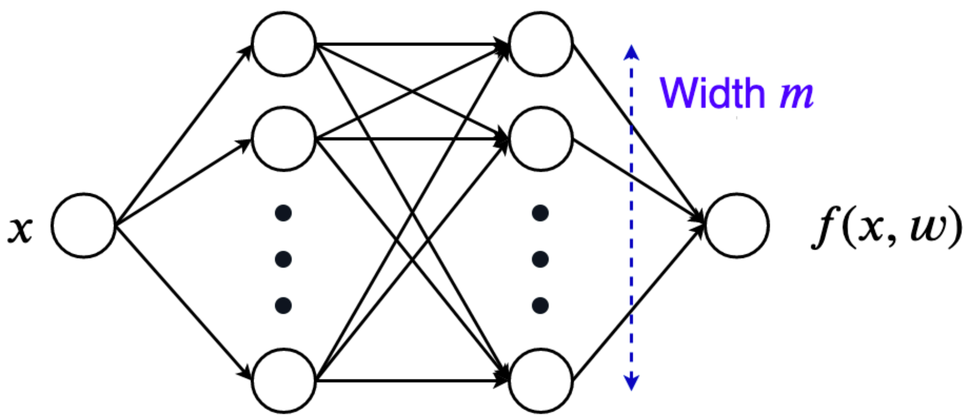
Rajat's Blog

A blog about machine learning and math.

Work of Rajat Vadiraj Dwaraknath

Blog Projects About

Understanding the Neural Tangent Kernel



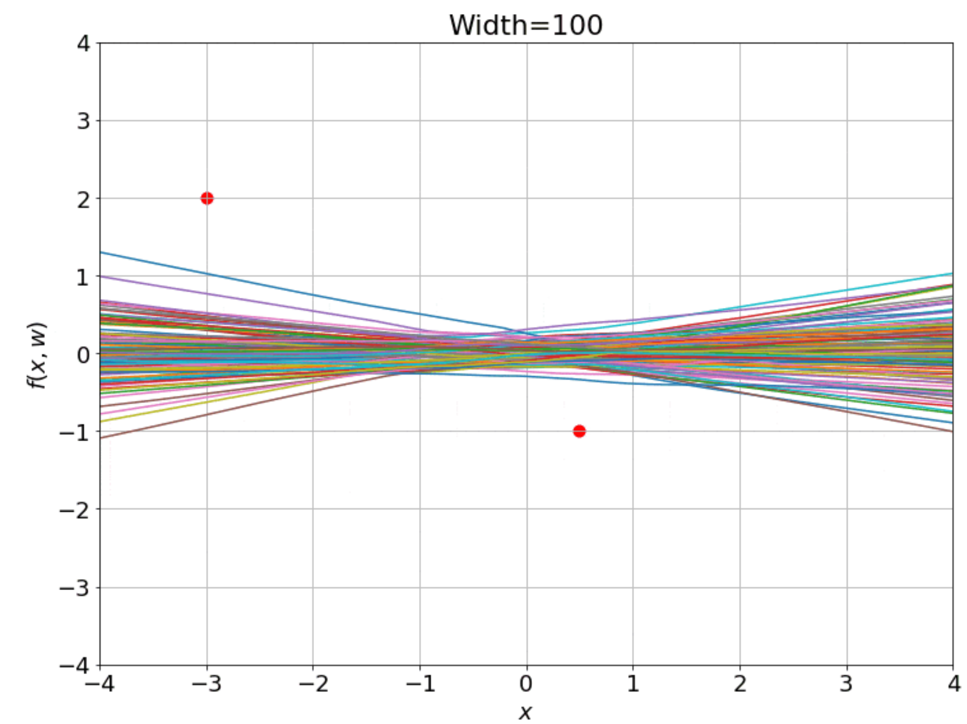
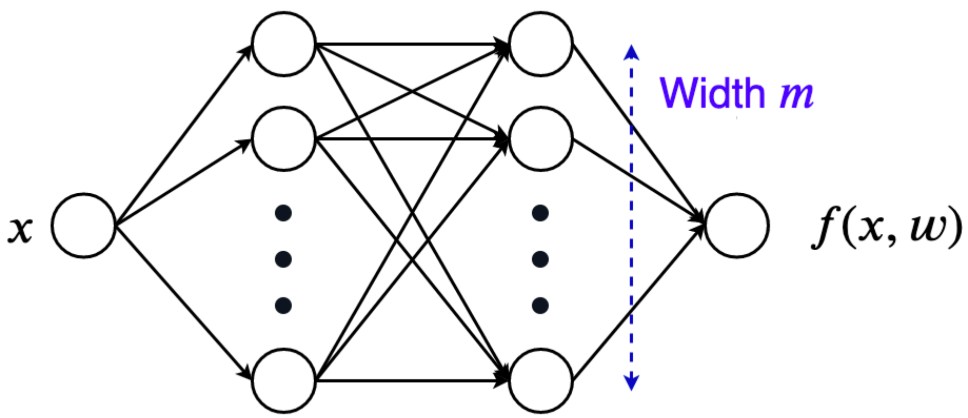
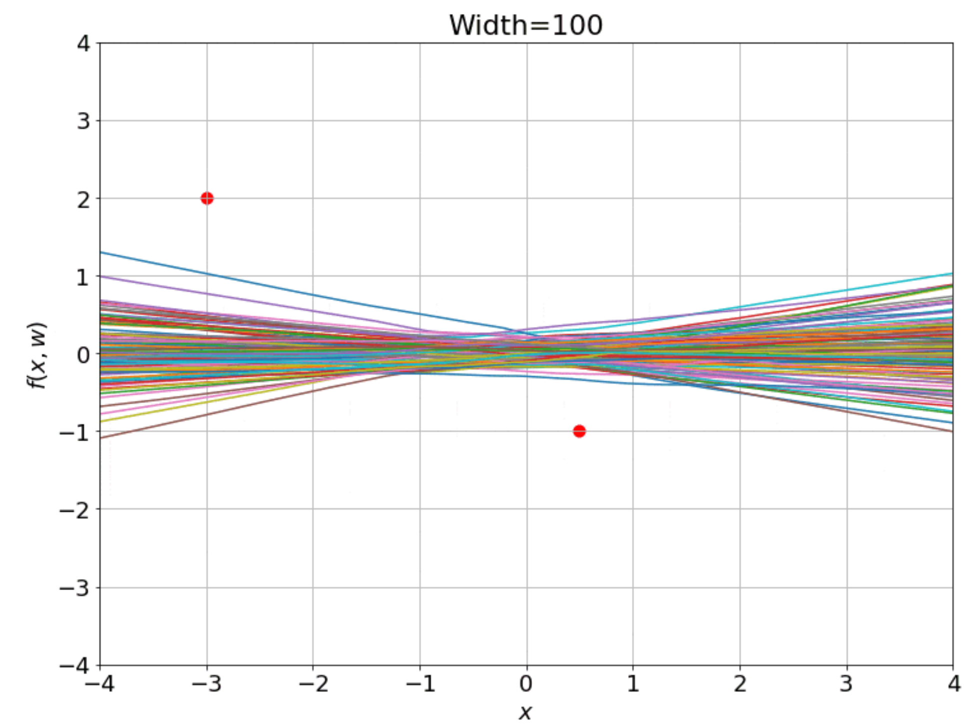


Figure Reference: https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

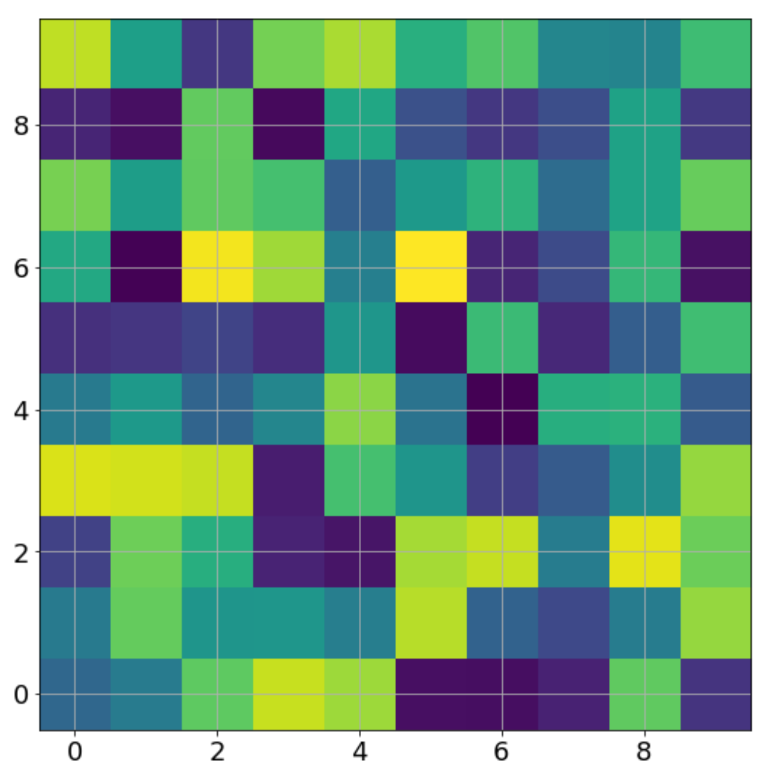
Work of Rajat Vadiraj Dwaraknath

Understanding the Neural Tangent Kernel

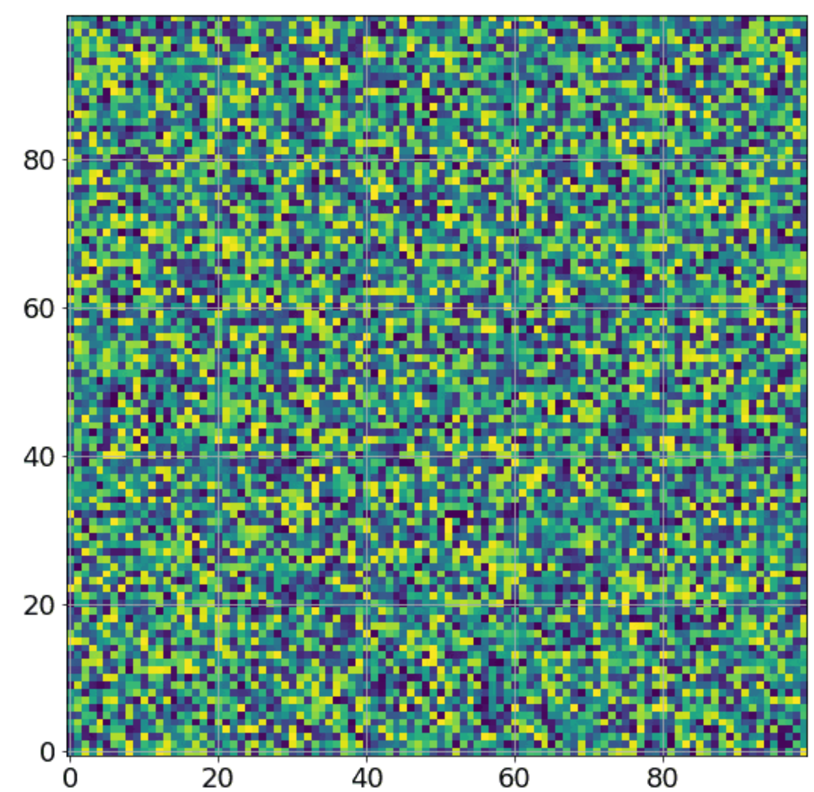




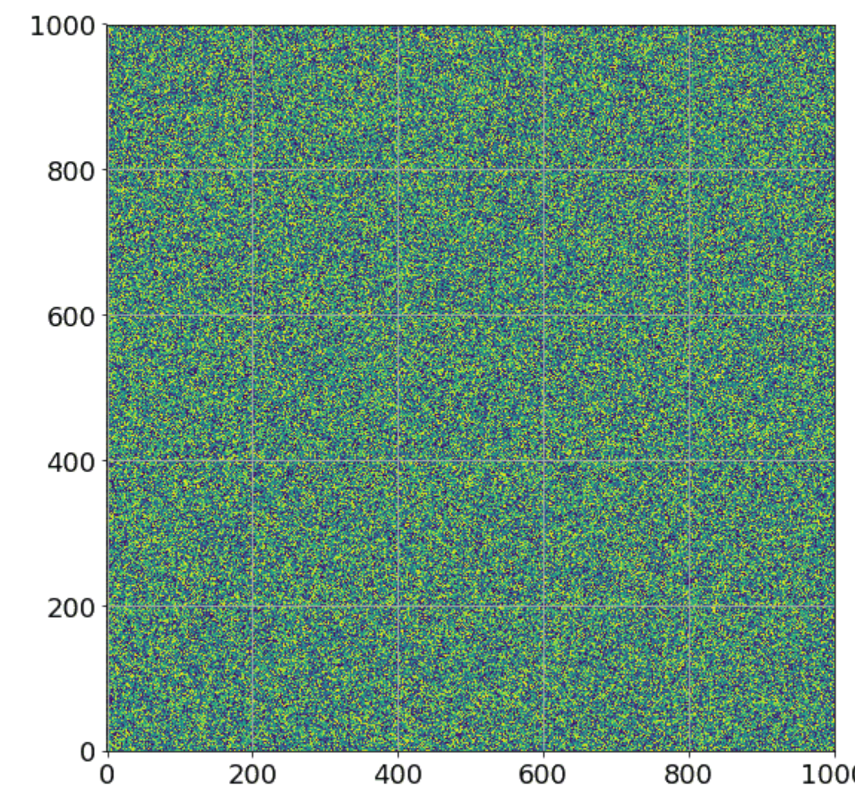
m = 10



m = 100



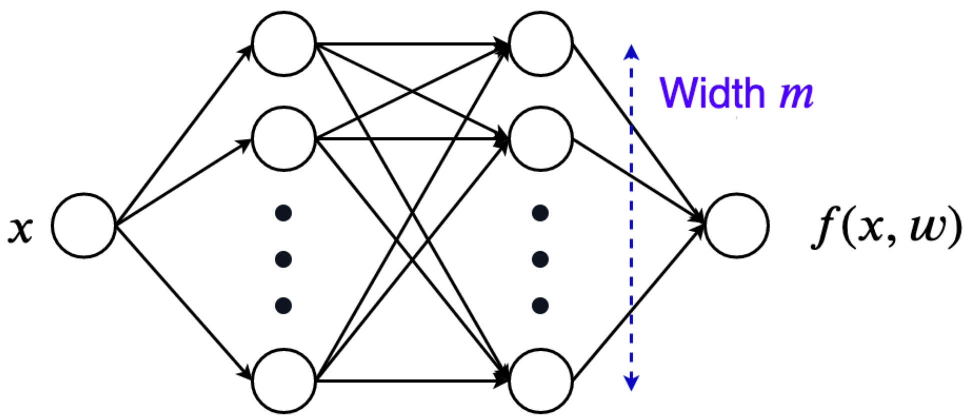
m = 1000

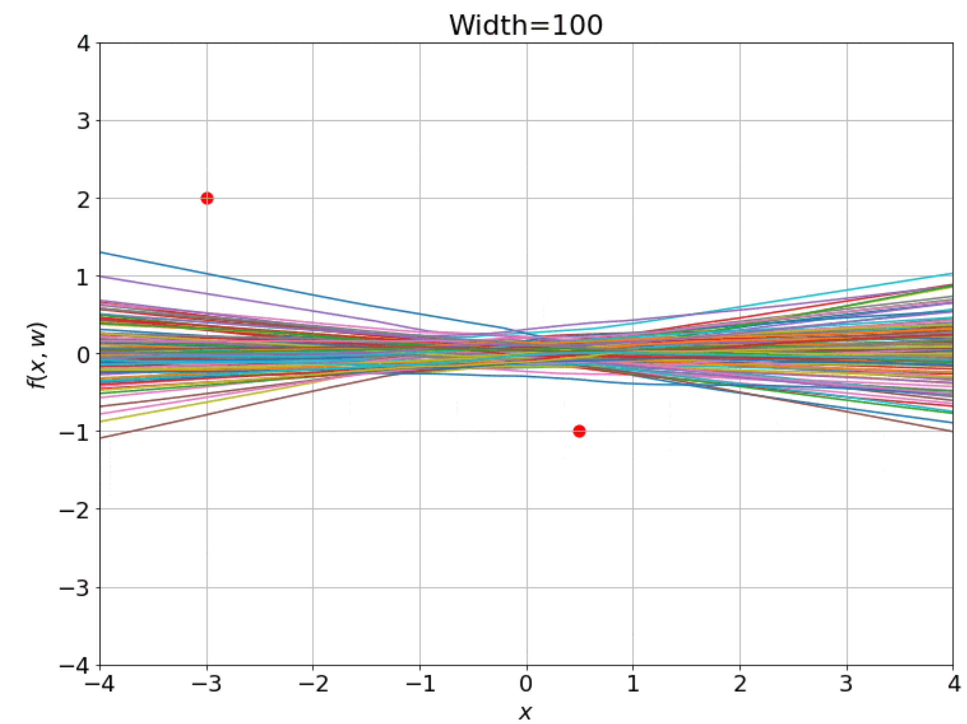


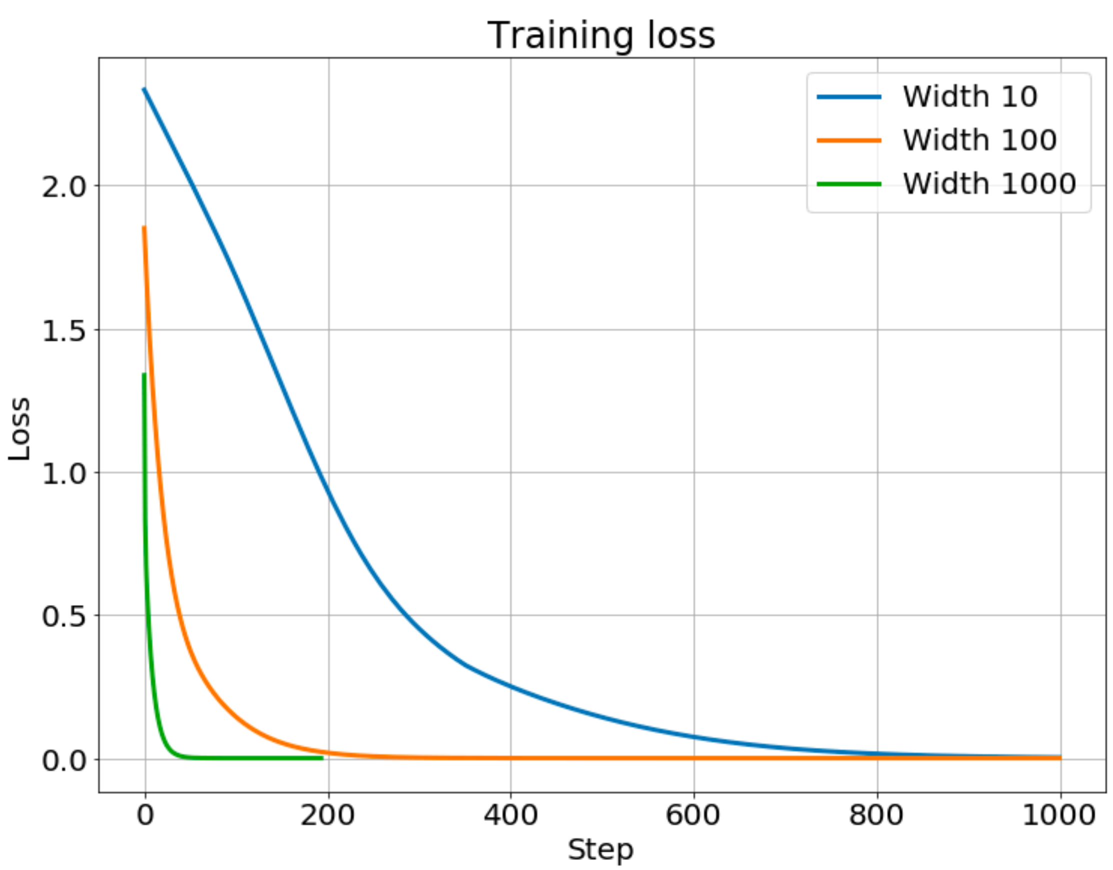
Rajat's Blog

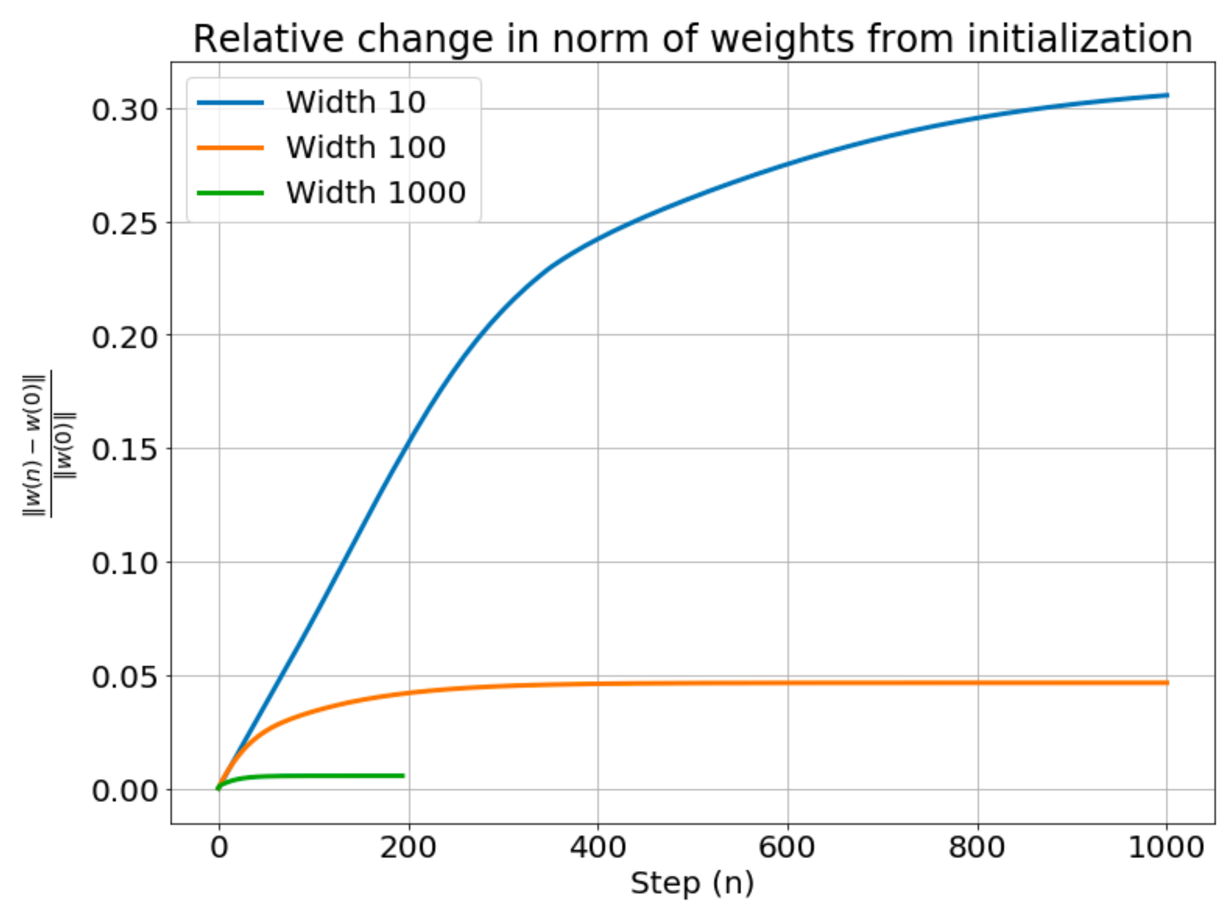
A blog about machine learning and math.

Understanding the Neural Tangent Kernel

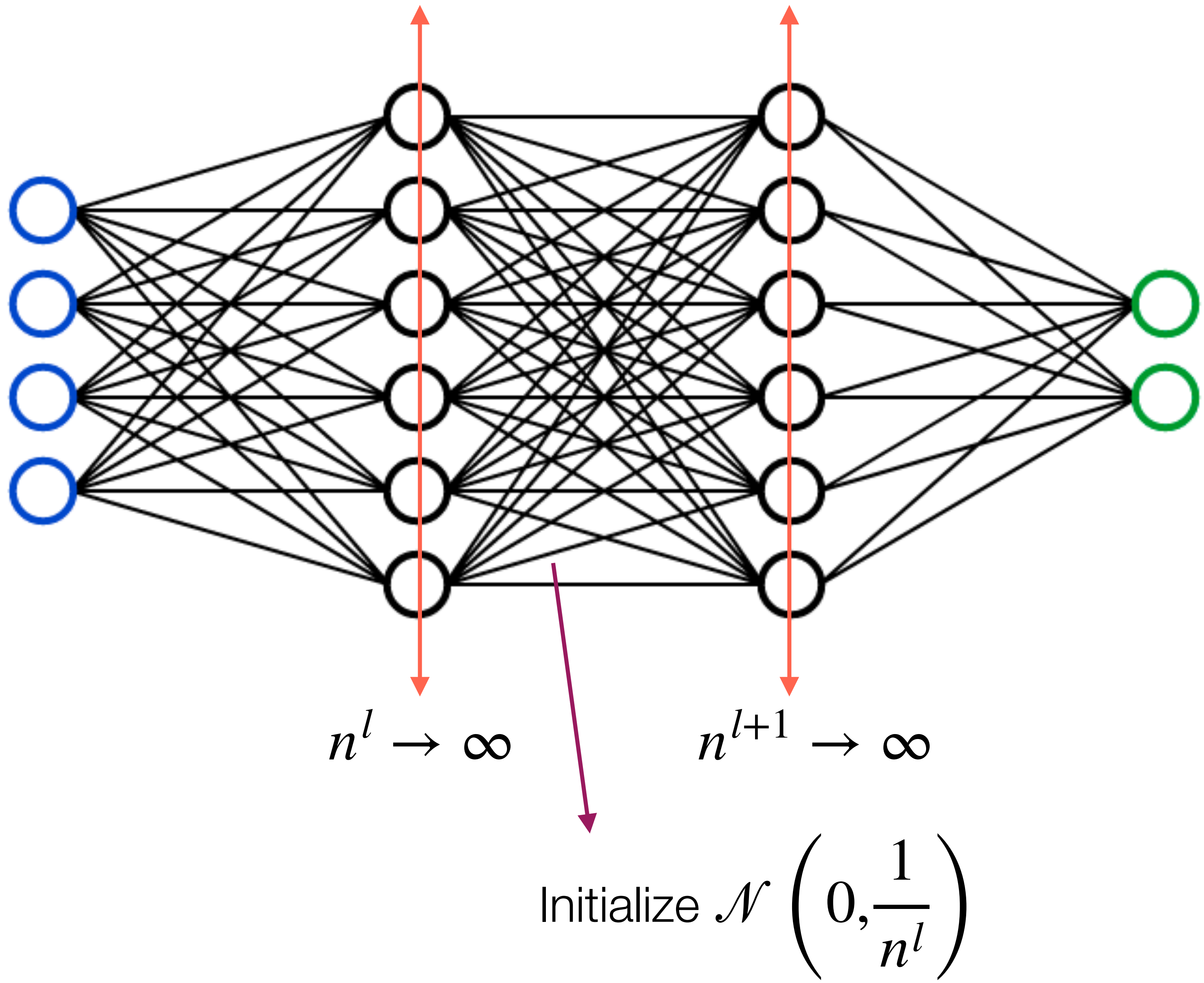








Infinite-Width Limit of Neural Networks



In this limit, *gradient descent training* of a fully connected neural network is equivalent to kernel regression with a kernel called the Neural Tangent Kernel.

Jacot et al., 2018; Arora et al., 2019; Lee et al., 2019

Goal of Network Training

Cost function for training: $$\frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^{\mu};\mathbf{w}) - f_T(\mathbf{x}^{\mu})\right)^2$$

$\mathbf{x}^{\mu}$: training datum

$\mathbf{w}$: training datum

$f(\mathbf{x}^{\mu}, \mathbf{w})$: Network's output

$f^T(\mathbf{x}^{\mu}, \mathbf{w})$: Desired output, target output, teacher, supervision …

P: number of training samples

Training Goal: Find best $\mathbf{w}$

Solution "Manifold"

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{\mu=1}^{P} \left(f(\mathbf{x}^\mu; \mathbf{w}) - f_T(\mathbf{x}^\mu) \right)^2$$

We found the minimum if $f(\mathbf{x}^\mu; \mathbf{w}) = f_T(\mathbf{x}^\mu)$ for all $\mu = 1, \ldots, P$

Suppose we have N parameters/weights: $N \gg P$ if overparametrized (infinite width)

Many more equations than unknowns!

(Possibly) many optimal $\mathbf{w}$ on an $N - P$ dimensional manifold? Which one to chose?

(Let this slide represent the N dimensional $\mathbf{w}$-space)

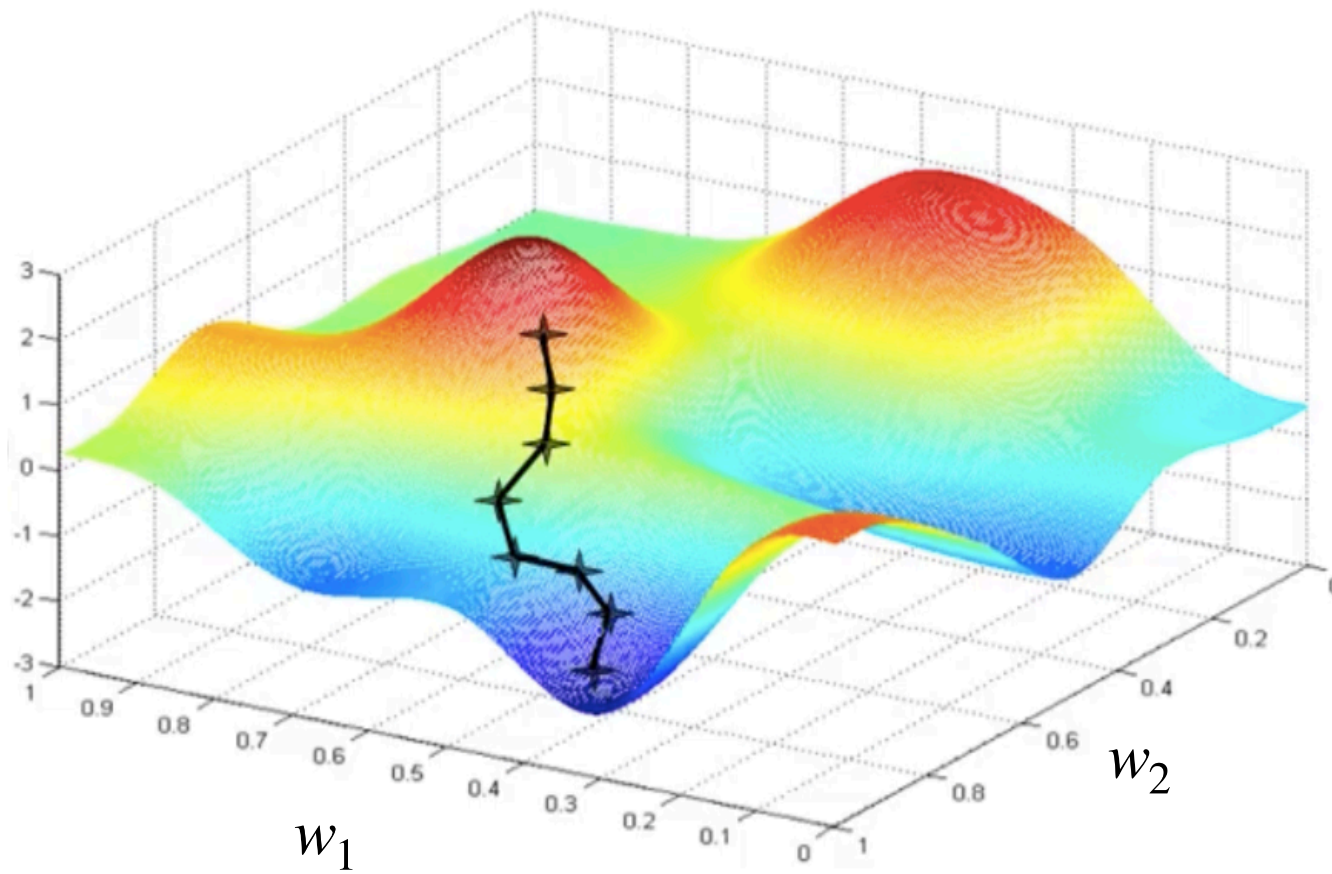
$N - P$ dimensional solution space

Learning/Training method selects the solution

Popular Method of Network Training: Stochastic Gradient Descent

Cost

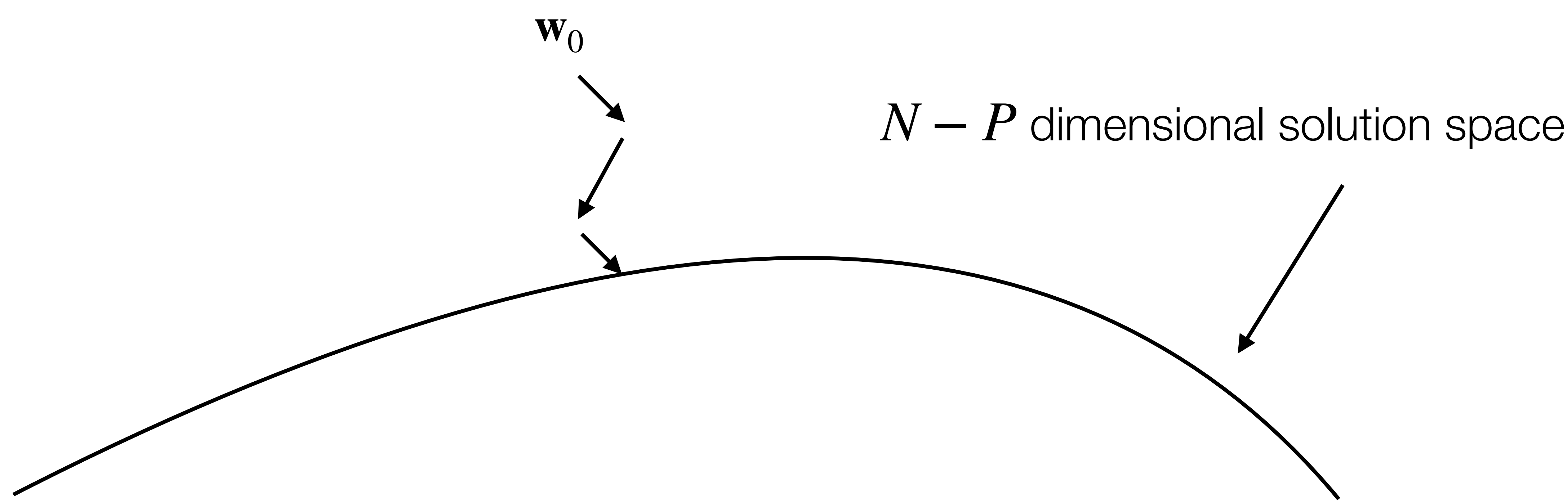
$$\frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^\mu;\mathbf{w}) - f_T(\mathbf{x}^\mu)\right)^2$$



Gradient Descent: $$\mathbf{w} \longleftarrow \mathbf{w} - \nabla_{\mathbf{w}}\frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^\mu;\mathbf{w}) - f_T(\mathbf{x}^\mu)\right)^2$$

Stochastic Gradient Descent: $$\mathbf{w} \longleftarrow \mathbf{w} - \nabla_{\mathbf{w}}\frac{1}{2}\left(f(\mathbf{x}^\mu;\mathbf{w}) - f_T(\mathbf{x}^\mu)\right)^2$$

(Let this slide represent the N dimensional $\mathbf{w}$-space)



What kind of solutions does stochastic gradient descent choose?
(Inductive bias of SGD)

Reduce Deep Network Complexity by Taylor Expansion!

$$\frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^{\mu};\mathbf{w})-f_T(\mathbf{x}^{\mu})\right)^2$$

Taylor expansion: $$f(\mathbf{x};\mathbf{w}) \approx f(\mathbf{x};\mathbf{w}_0) + (\mathbf{w}-\mathbf{w}_0)\cdot\nabla_{\mathbf{w}} f(\mathbf{x};\mathbf{w}_0)$$

$$\frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^{\mu};\mathbf{w})-f_T(\mathbf{x}^{\mu})\right)^2 \approx \frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^{\mu};\mathbf{w}_0)+(\mathbf{w}-\mathbf{w}_0)\cdot\nabla_{\mathbf{w}} f(\mathbf{x}^{\mu};\mathbf{w}_0)-f_T(\mathbf{x}^{\mu})\right)^2$$

Linearized Networks and Neural Tangent Kernel (NTK)

$$f(\mathbf{x};\mathbf{w}) \approx f(\mathbf{x};\mathbf{w}_0) + (\mathbf{w}-\mathbf{w}_0)\cdot\nabla_{\mathbf{w}} f(\mathbf{x};\mathbf{w}_0)$$

$$\frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^\mu;\mathbf{w}) - f_T(\mathbf{x}^\mu)\right)^2 \approx \frac{1}{2}\sum_{\mu=1}^{P}\left(f(\mathbf{x}^\mu;\mathbf{w}_0) + (\mathbf{w}-\mathbf{w}_0)\cdot\nabla_{\mathbf{w}} f(\mathbf{x}^\mu;\mathbf{w}_0) - f_T(\mathbf{x}^\mu)\right)^2$$

Gradient flow to zero error: $f(\mathbf{x}) = \mathbf{y}^\top \mathbf{K}_{NTK}^{-1}\mathbf{k}(\mathbf{x})$

$$K_{NTK}(\mathbf{x},\mathbf{x}') = \nabla_{\mathbf{w}} f(\mathbf{x};\mathbf{w}_0)\cdot\nabla_{\mathbf{w}} f(\mathbf{x}';\mathbf{w}_0) \quad \text{NTK}$$

$\mathbf{K}_{NTK}$ $\quad P\times P$ kernel gram matrix $\mathbf{K}_{NTK,\mu\nu} = K_{NTK}(\mathbf{x}^\mu,\mathbf{x}^\nu)$,

$k(\mathbf{x})_\mu = K_{NTK}(\mathbf{x},\mathbf{x}^\mu)$

$y_\mu = f_T(\mathbf{x}^\mu)$

Approximation is exact in the infinite-width limit
Jacot et al., 2018; Arora et al., 2019; Lee et al., 2019

Neural Tangent Kernel (NTK) and Its Spectrum

$$K_{NTK}(\mathbf{x}, \mathbf{x}') = \nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}_0) \cdot \nabla_{\mathbf{w}} f(\mathbf{x}'; \mathbf{w}_0)$$ NTK

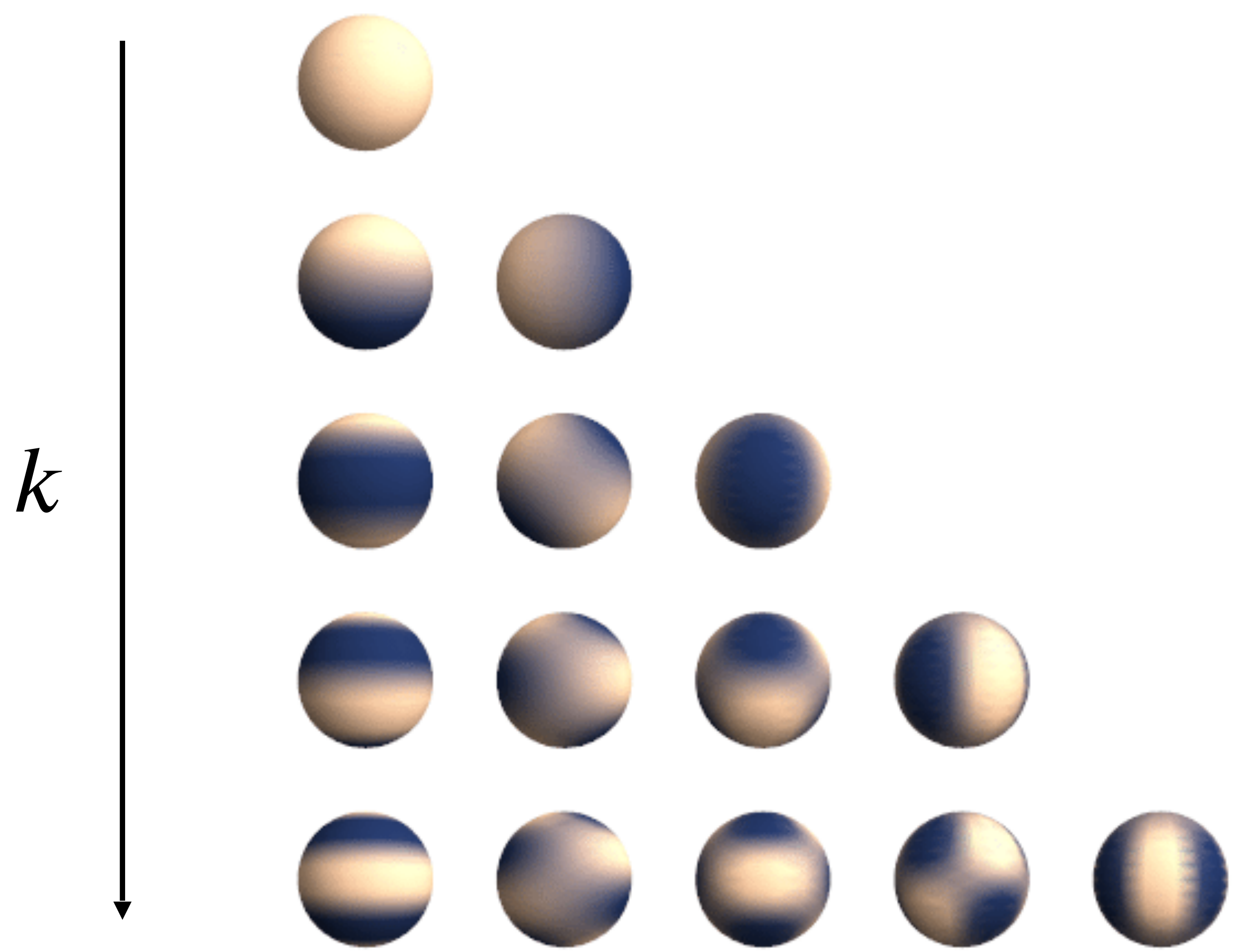
$\mathbf{K}_{NTK}$ $P \times P$ matrix $\mathbf{K}_{NTK,\mu\nu} = K_{NTK}(\mathbf{x}^\mu, \mathbf{x}^\nu)$,

If $\mathscr{X} = \mathbb{S}^{d-1}$, and kernel is rotation invariant, as is K_{NTK}, then

$$K(\mathbf{x}, \mathbf{x}') = \sum_{k=0}^{\infty} \lambda_k \sum_{m=1}^{N(d,k)} Y_{km}(\mathbf{x}) Y_{km}(\mathbf{x}')$$

Spherical
Harmonics
(Mercer Decomposition)
Smola et. al, 2001; Bietti & Mairal, 2019

Spherical Harmonics



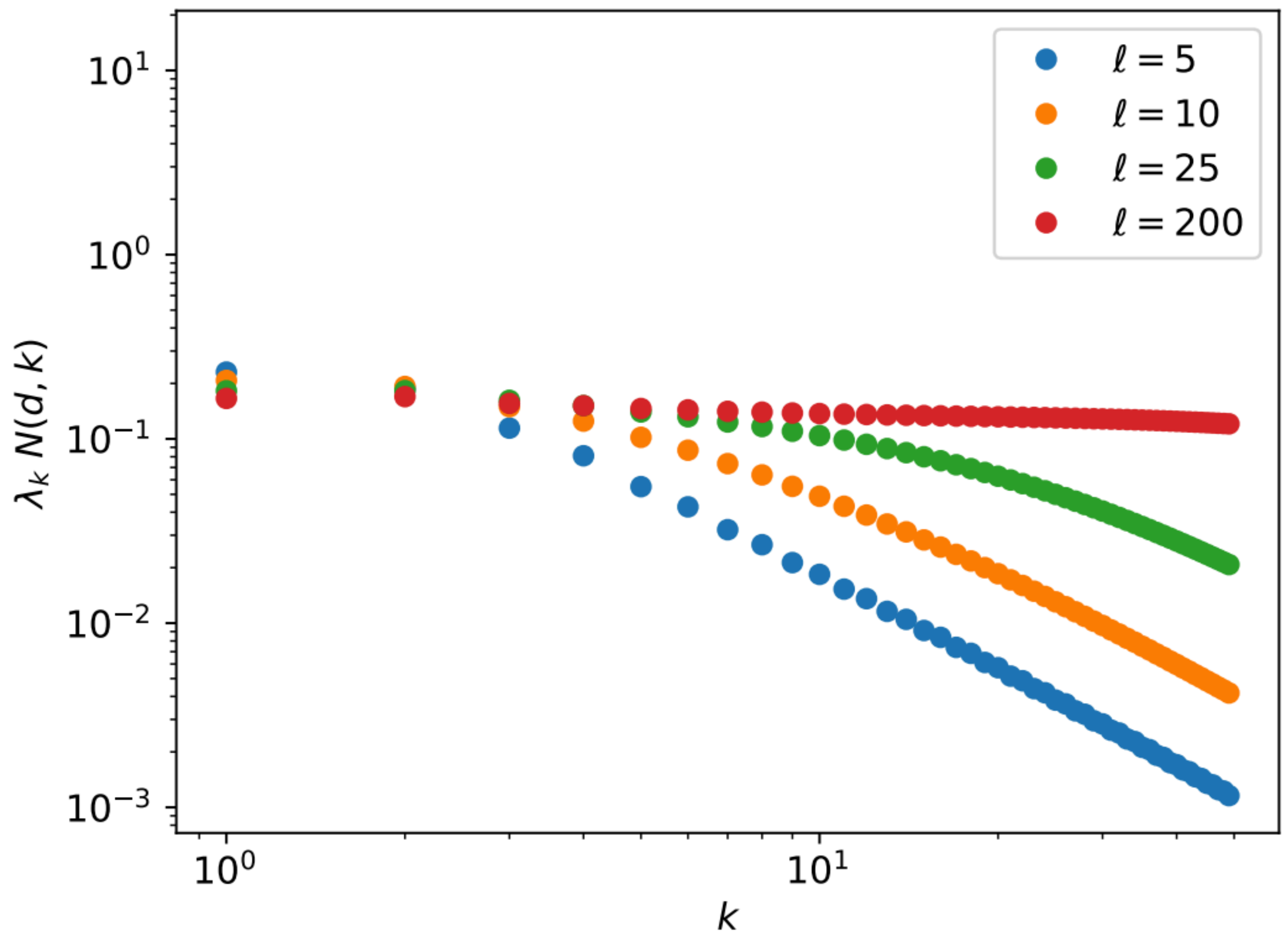


Figure SI.1. Spectrum of fully connected ReLU NTK without bias for varying depth ℓ. As the depth increases, the spectrum whitens, causing derivatives of lower order to have infinite variance. As $\ell \to \infty$, $\lambda_k N(d, k) \sim 1$ implying that the kernel becomes a Delta function possibly added to a scalar $K(\mathbf{x}, \mathbf{x}') \sim a\delta(\mathbf{x} - \mathbf{x}') + b$ for some constants a and b.

Expression for Generalization Error

$$E_g = \left\langle \left(f(\mathbf{x}) - f_T(\mathbf{x}) \right)^2 \right\rangle =: \sum_{\rho} E_\rho$$

$$E_\rho = \frac{\overline{w}_\rho^2}{\lambda_\rho} \left[\frac{(\lambda + t)^2}{(\lambda + t)^2 - P\gamma} \right] \frac{1}{\left(\frac{1}{\lambda_\rho} + \frac{P}{\lambda + t} \right)^2}$$

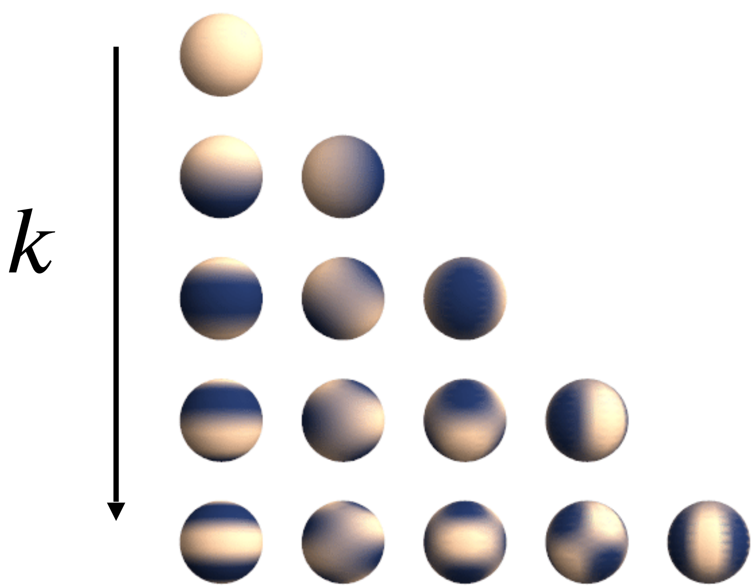
$$t = \sum_{\rho} \frac{1}{\frac{1}{\lambda_\rho} + \frac{P}{\lambda + t}}$$

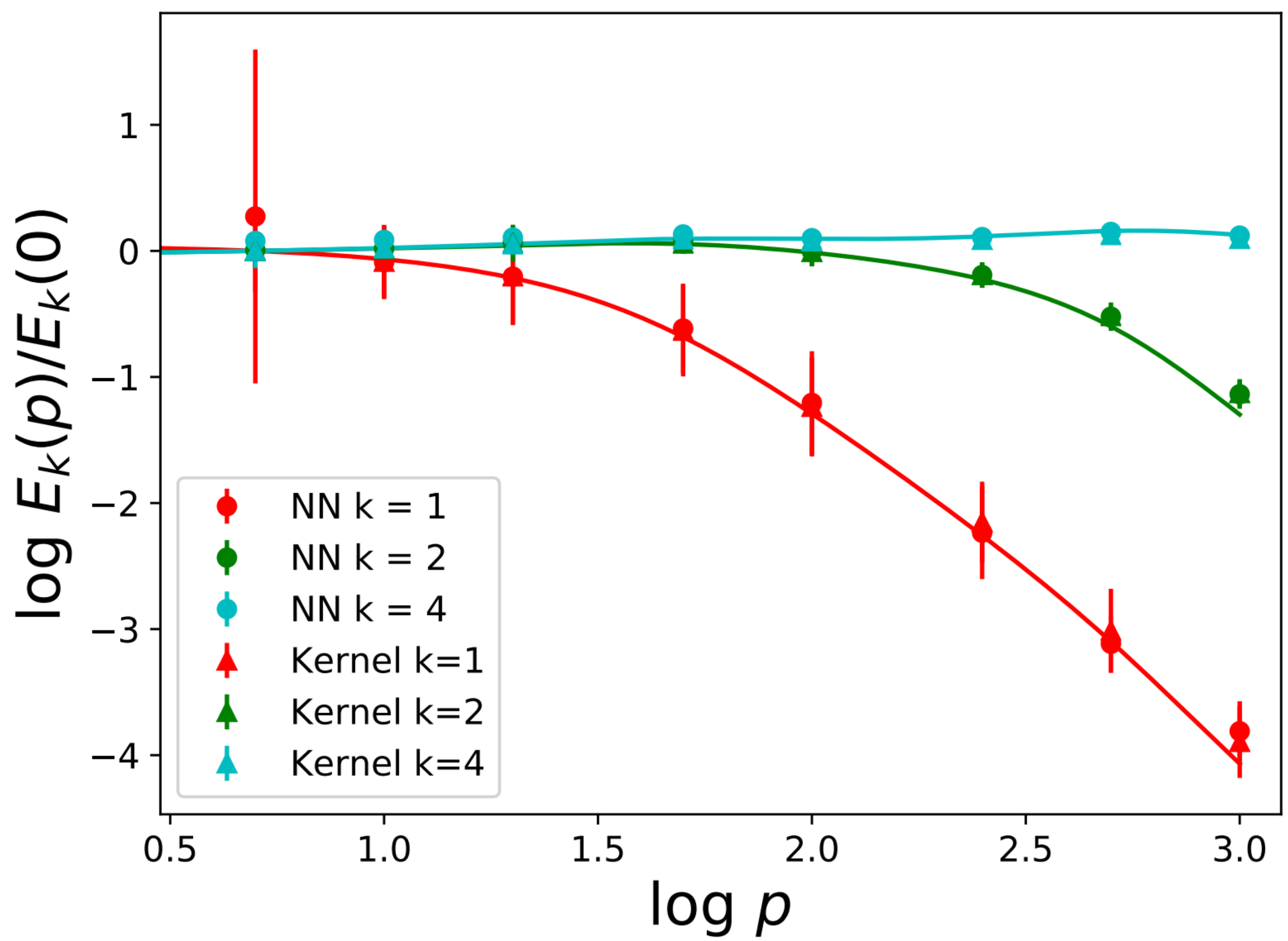
$$\gamma = \sum_{\rho} \frac{1}{\left(\frac{1}{\lambda_\rho} + \frac{P}{\lambda + t} \right)^2}$$

Why don’t deep networks overfit?

Learning algorithms are biased toward simple functions

Spherical Harmonics





(a) 2 layer NN $N = 10^4$

Large-d limit

To gain insight, we take $d \to \infty$ limit. In this limit $\lambda_\rho \sim O(d^{-\rho})$. We assume $P = \alpha d^l$,

We can analyze the asymptotes of the learning curves:

for $k < l$, $\dfrac{E_{km}(P)}{E_{km}(0)} = 0$ $\longrightarrow$ Perfect generalization

for $k = l$, $\dfrac{E_{km}(P)}{E_{km}(0)} = f(\alpha)$ $\longrightarrow$ Learning

for $k > l$, $\dfrac{E_{km}(P)}{E_{km}(0)} = 1$ $\longrightarrow$ Not learned

Sharpening the question
(one of the many ways the study of mathematical models is useful)

The brain is not overfitting because it has an "inductive bias".
Theory suggests that we need to understand (at least)

1. The learning algorithms of the brain

2. The (tangent) kernel of the brain

Acknowledgments

<u>Group Members:</u>

Ricardo Alves

Blake Bordelon

Abdulkadir Canatar

Ugne Klibaite

Rong Liu

Shanshan Qin

Jacob Zavatone-Veth

Siyan Zhou

intel

Google

HDSI

NIH

Harvard John A. Paulson School of Engineering and Applied Sciences