CS 229br Lecture 2: Dynamics & Bias

Boaz Barak

Ankur Moitra
MIT 18.408

Yamini Bansal
Official TF

Dimitris Kalimeris
Unofficial TF

Gal Kaplun
Unofficial TF

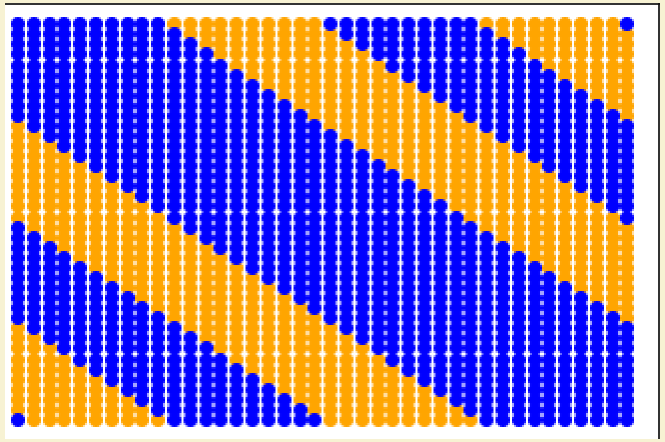
Preetum Nakkiran
Unofficial TF

- **#hw0** – Deadline Mon 2/8 11:59pm gradescope
- **#lectures** – Decordings and slides
- **#qanda** – Questions about course material
- **#sys-help** – Help with tech / training
- **#papers** – Talk about cool papers

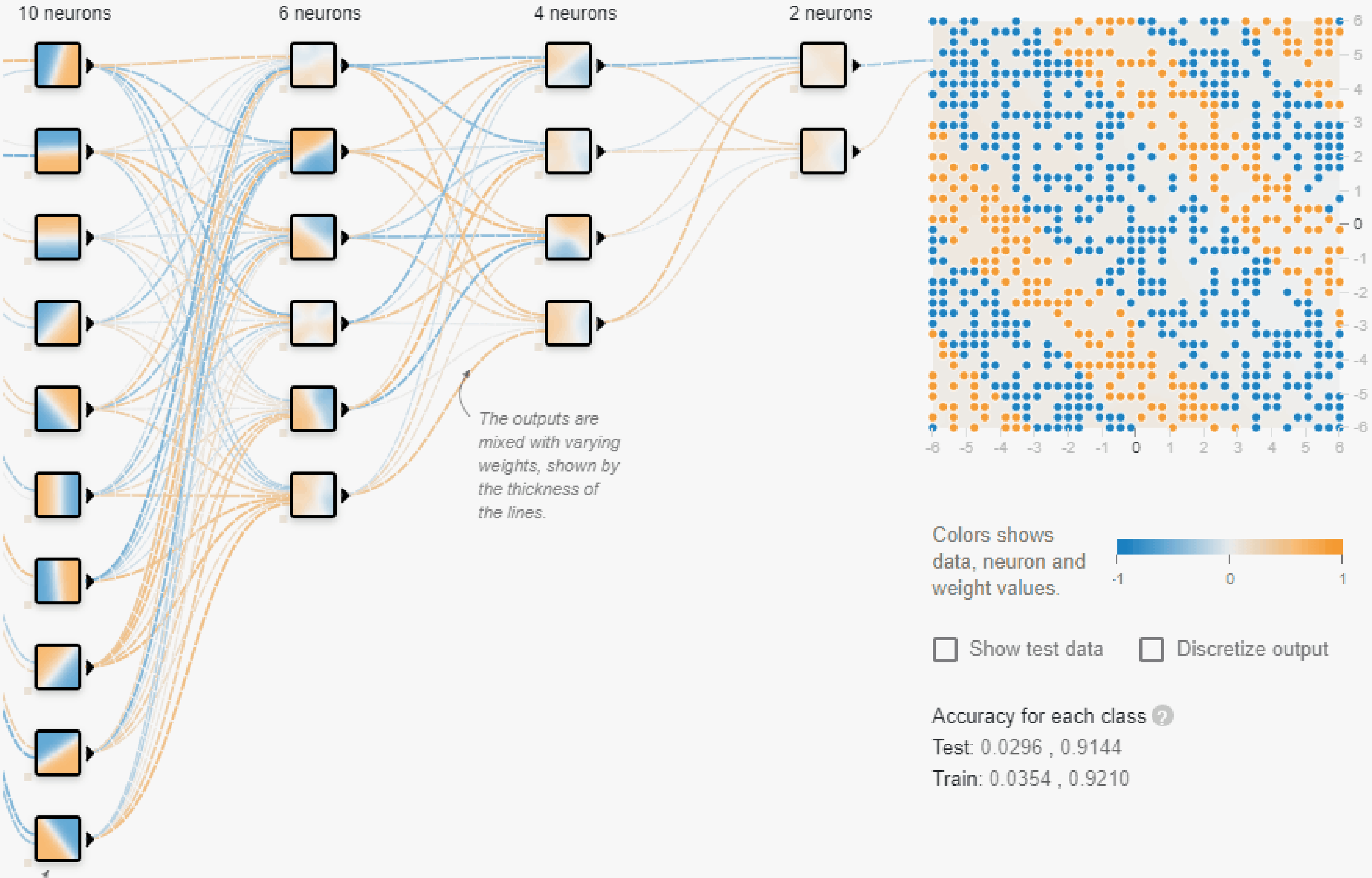
What do networks learn and when do they learn it?

- Simplicity bias
- Learning dynamics – what is learned first
- Different layers – what is learned by which layers?
- Some experimental evidence
- What can we prove?

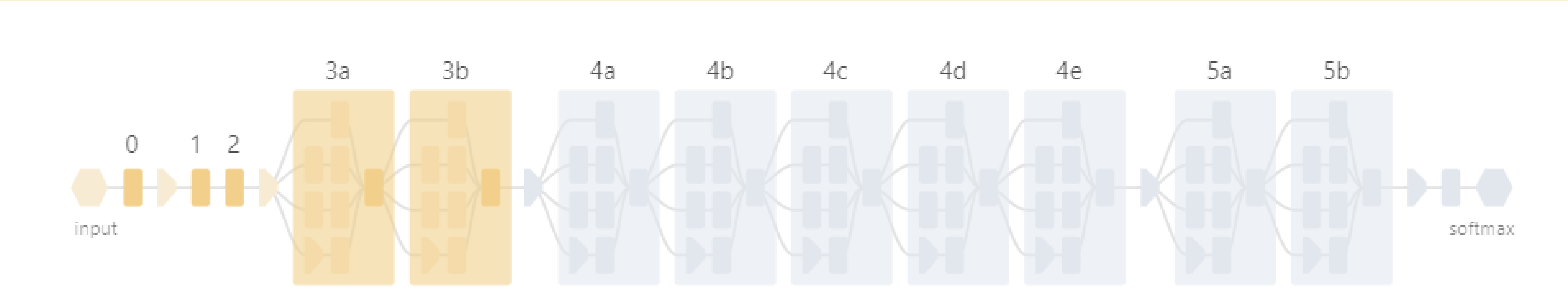
What do networks learn?



http://tfmeter.icsi.berkeley.edu/


10 neurons
6 neurons
4 neurons
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output
Accuracy for each class
Test: 0.0296 , 0.9144
Train: 0.0354 , 0.9210

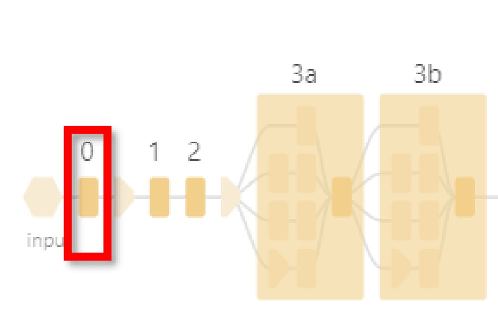
Inception V1 (Olah et al, 2020)



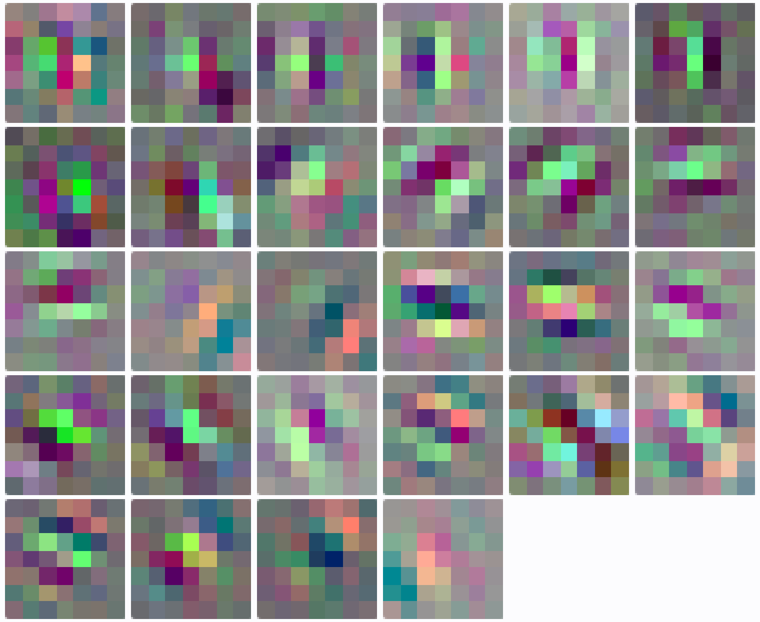
Olah, Cammarata, Schubert, Goh, Petrov, Carter 2020

Inception V1 (Olah et al, 2020)

conv2d0



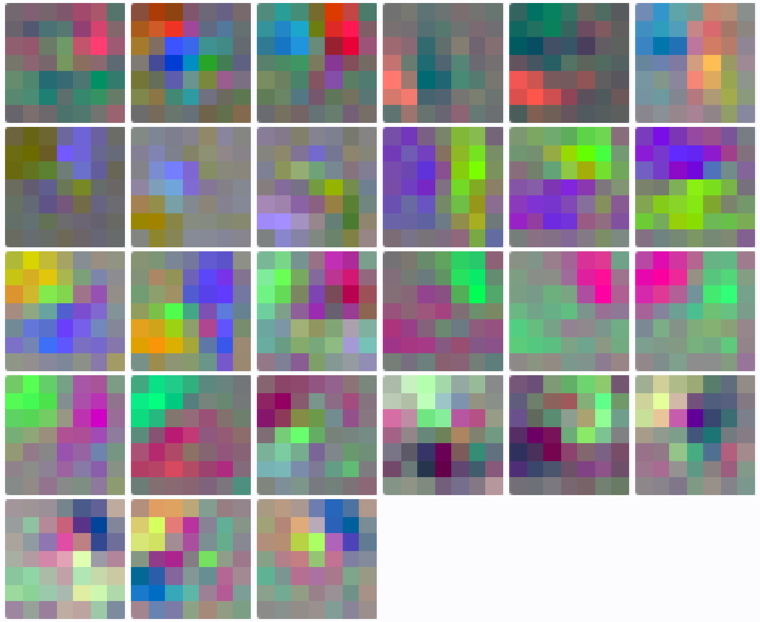
Gabor Filters 44%



Collapse neurons.

Gabor filters are a simple edge detector, highly sensitive to the alignment of the edge. They're almost universally found in the fist layer of vision models. Note that Gabor filters almost always come in pairs of negative reciprocals.

Color Contrast 42%



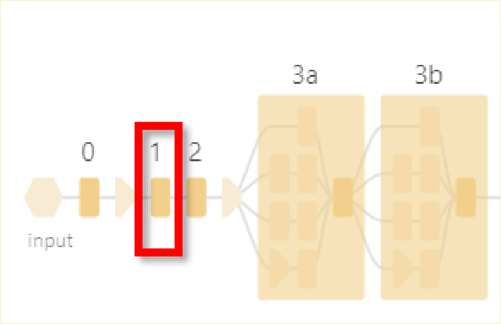
Collapse neurons.

These units detect a color one side of their receptive field, and the opposite color on the other side. Compare to later color contrast (`conv2d1`, `conv2d2`, `mixed3a`, `mixed3b`).

Other Units 14%

Units that don't fit in another category.

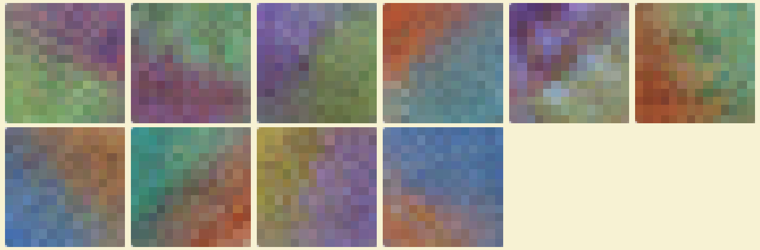
Inception V1 (Olah et al, 2020)



conv2d1

Gabor Like 17%

Color Contrast 16%

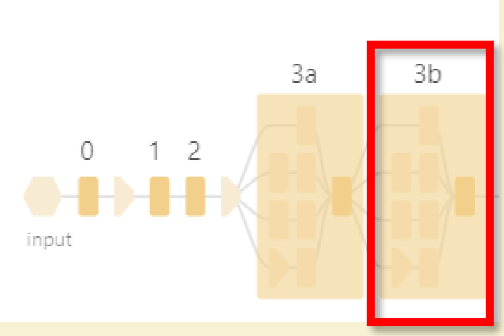


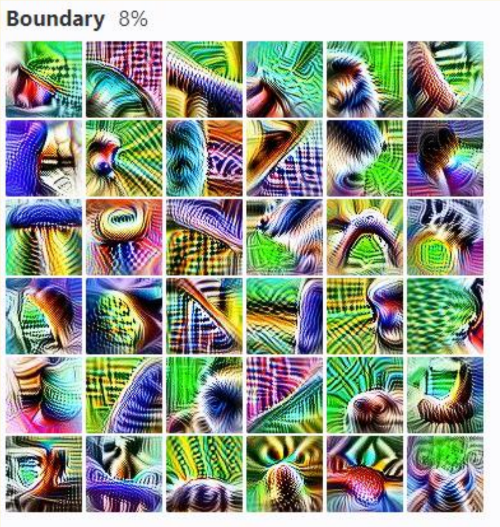
Low Frequency 27%

Complex Gabor 14%

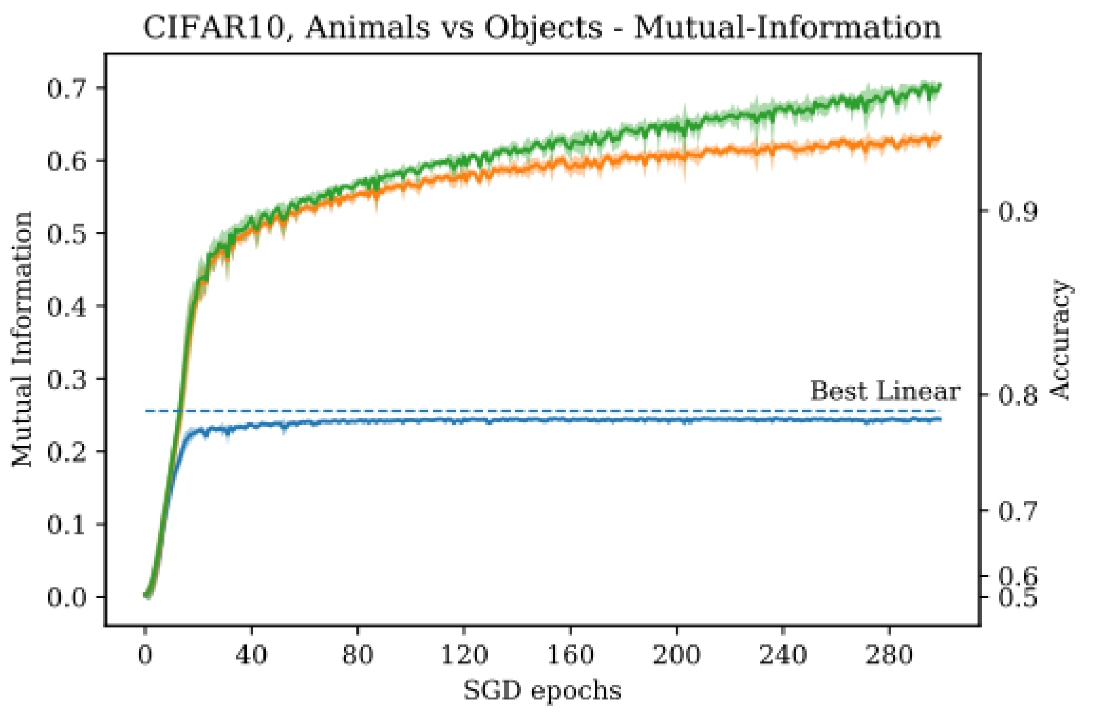
Inception V1 (Olah et al, 2020)

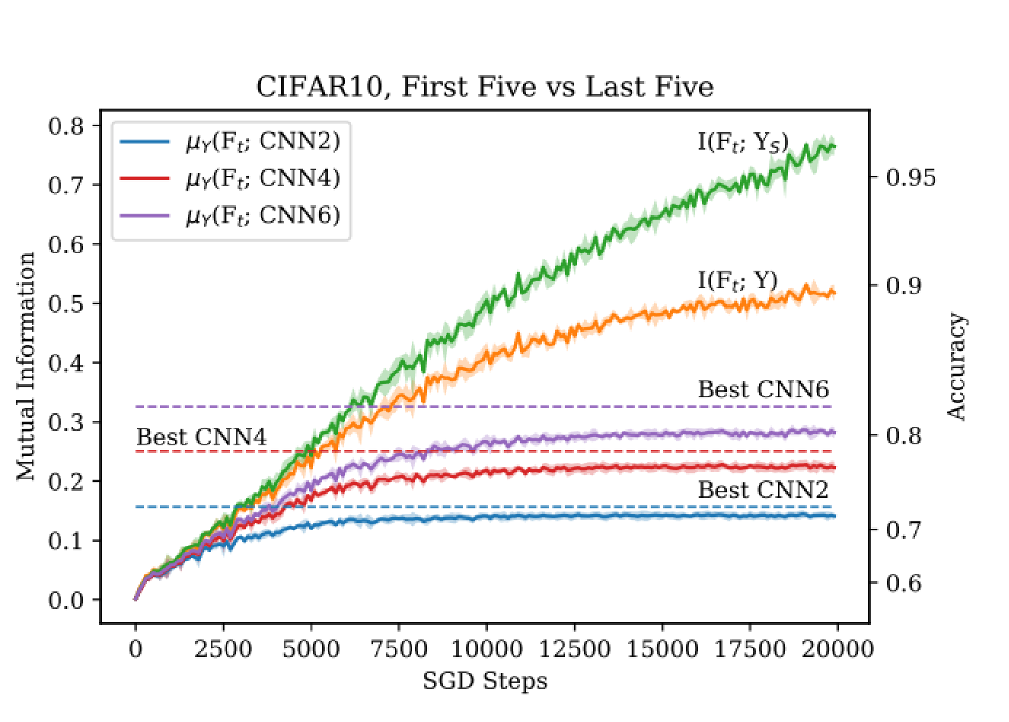
mixed3b





SGD Learns simple concepts first



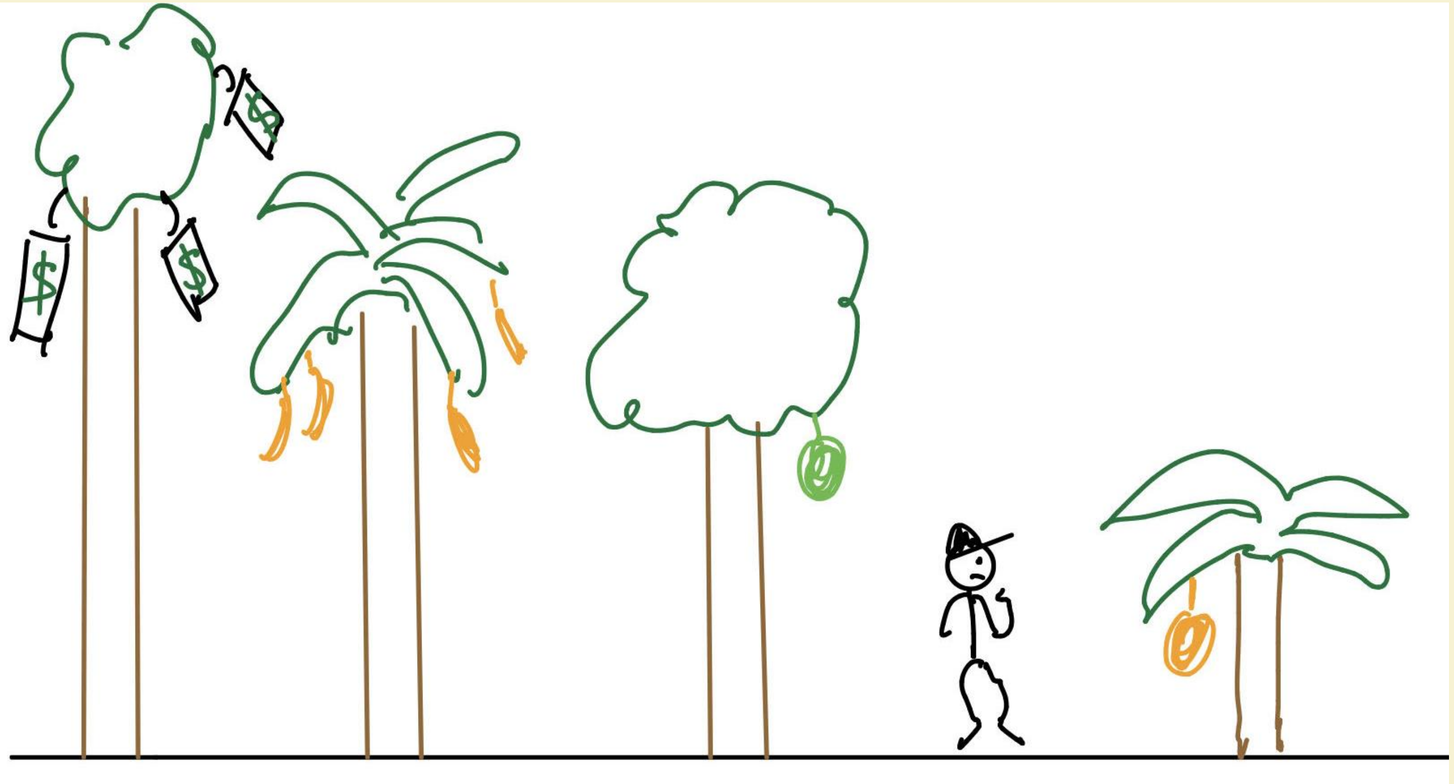


Nakkiran, Kaplun, Kalimeris, Yang, Edelman, Yang, Zhang, B. 2020

Simplicity bias is a good thing…

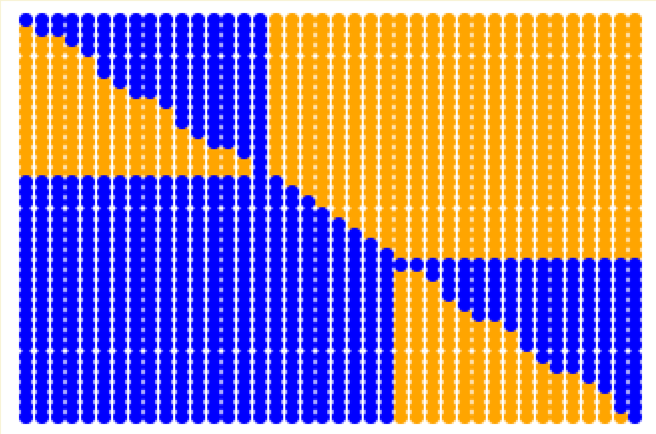
A random f fitting $(x_i, y_i)_{i=1..n}$ will never generalize.

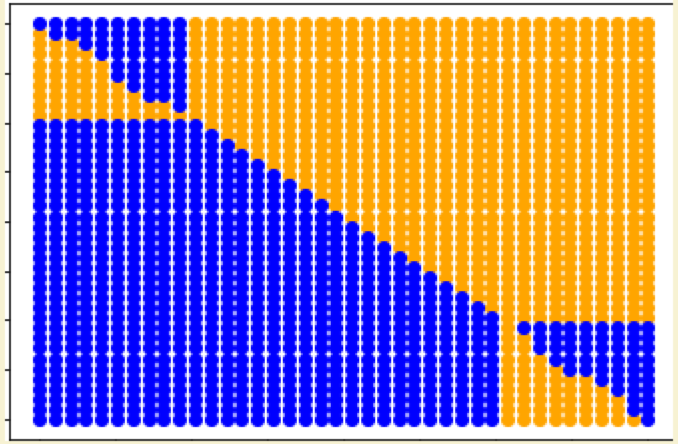
… and a bad thing

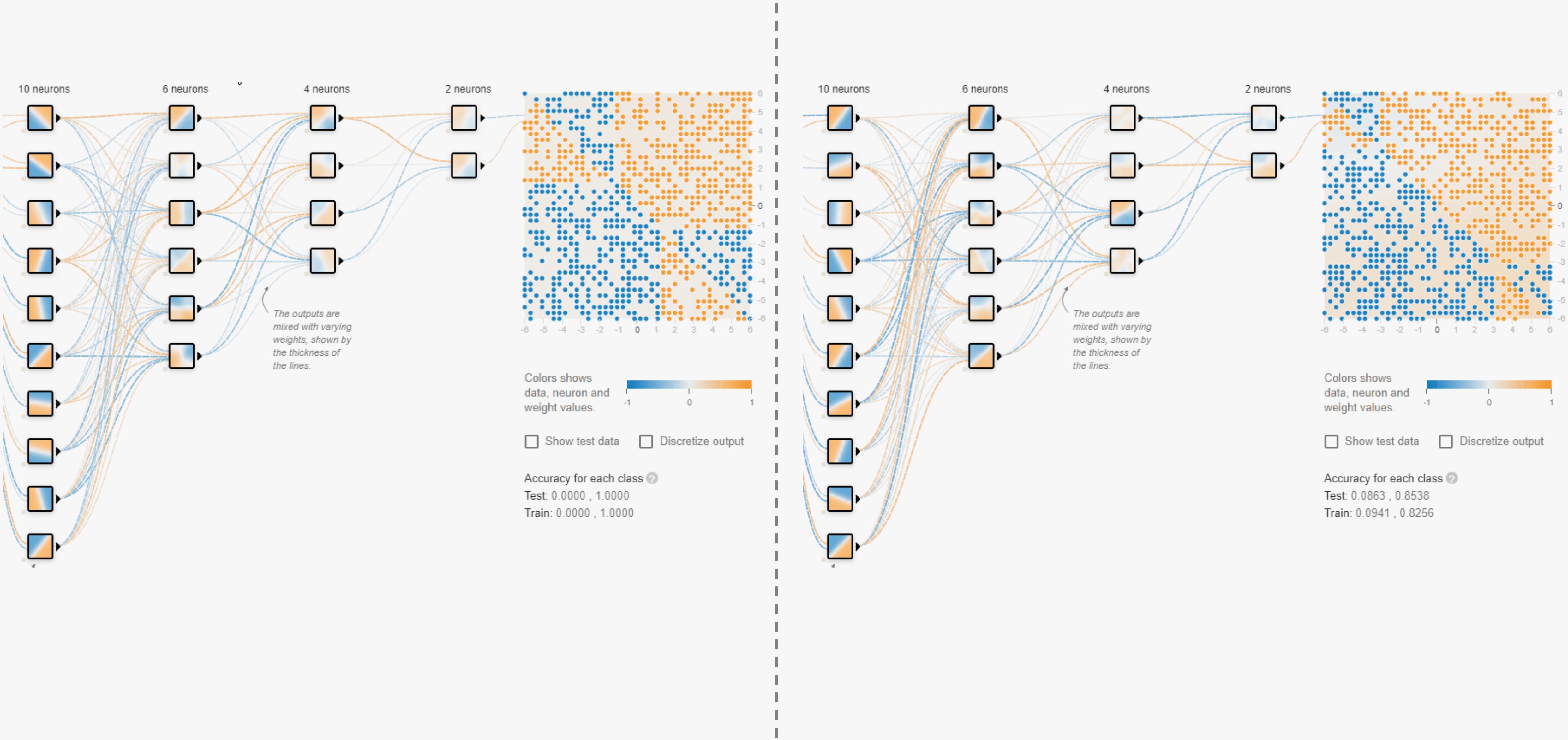


$\boldsymbol{x}$	$\boldsymbol{f(x)}$
x_1	y_1
x_2	y_2
x_3	y_3
…	…
x_n	y_n
$\boldsymbol{x}$	__

Example:






10 neurons
6 neurons
4 neurons
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output
Accuracy for each class
Test: 0.0000 , 1.0000
Train: 0.0000 , 1.0000
10 neurons
6 neurons
4 neurons
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output
Accuracy for each class
Test: 0.0863 , 0.8538
Train: 0.0941 , 0.8256

The Pitfalls of Simplicity Bias in Neural Networks

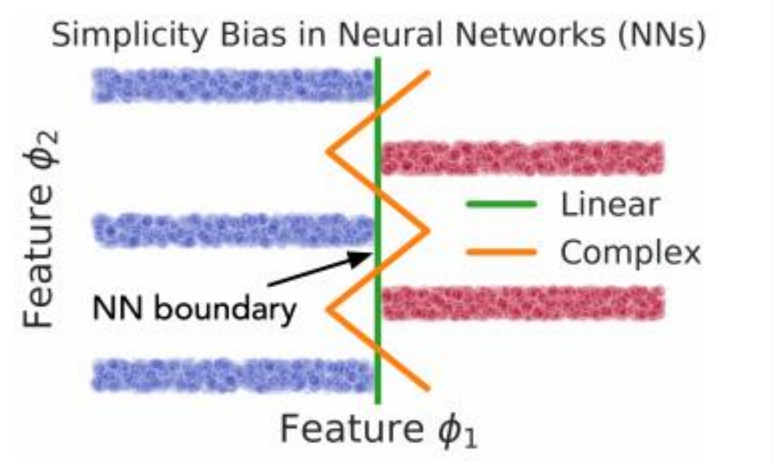
Harshay Shah
Microsoft Research
harshay.rshah@gmail.com

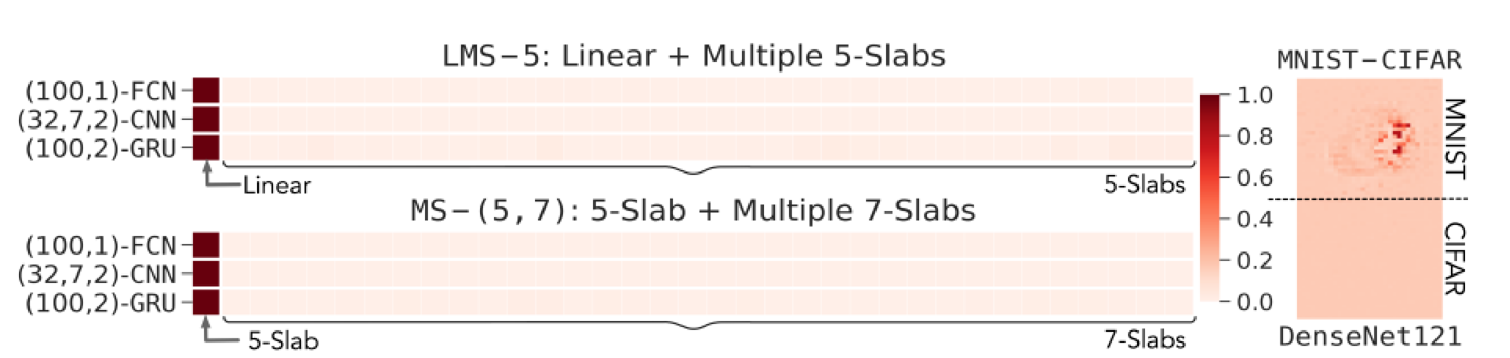
Kaustav Tamuly
Microsoft Research
ktamuly2@gmail.com

Aditi Raghunathan
Stanford University
aditir@stanford.edu

Prateek Jain
Microsoft Research
prajain@microsoft.com

Praneeth Netrapalli
Microsoft Research
praneeth@microsoft.com





What can we prove?

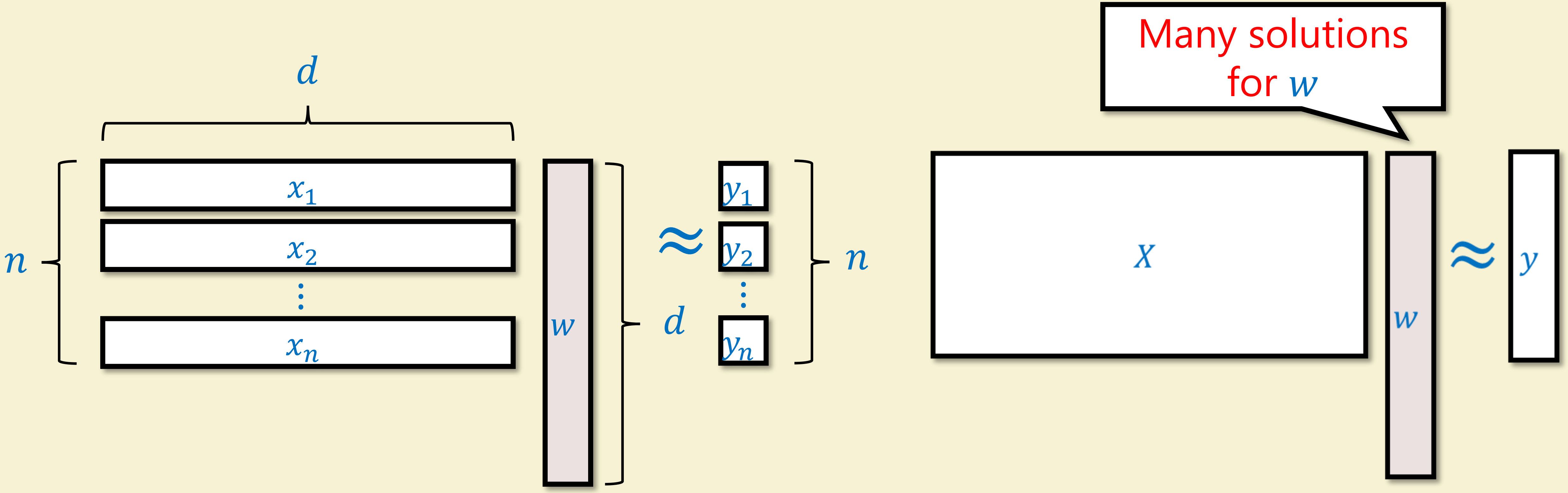
(Over-parameterized) Linear Regression

* Ignoring bias / assuming $x_i = (1, \dots)$

Input: $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^{d+1}$, $d \gg n$

Goal: Find $w \in \mathbb{R}^d$ s.t. $\langle w, x_i \rangle \approx y_i$

and (more importantly) $\langle w, x \rangle \approx y$ for fresh (x, y)

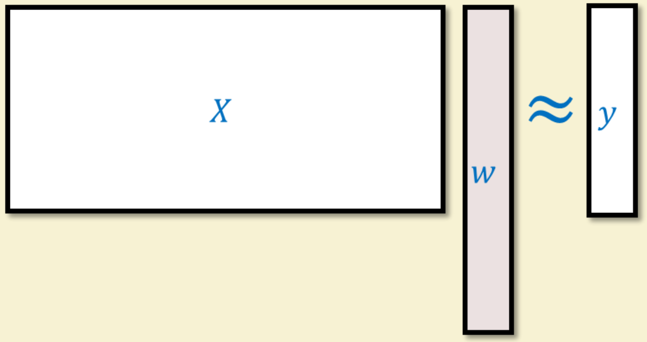


(Over-parameterized) Linear Regression

Input: $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^{d+1}$, $d \gg n$

Goal: Find $w \in \mathbb{R}^d$ s.t. $\langle w, x_i \rangle \approx y_i$

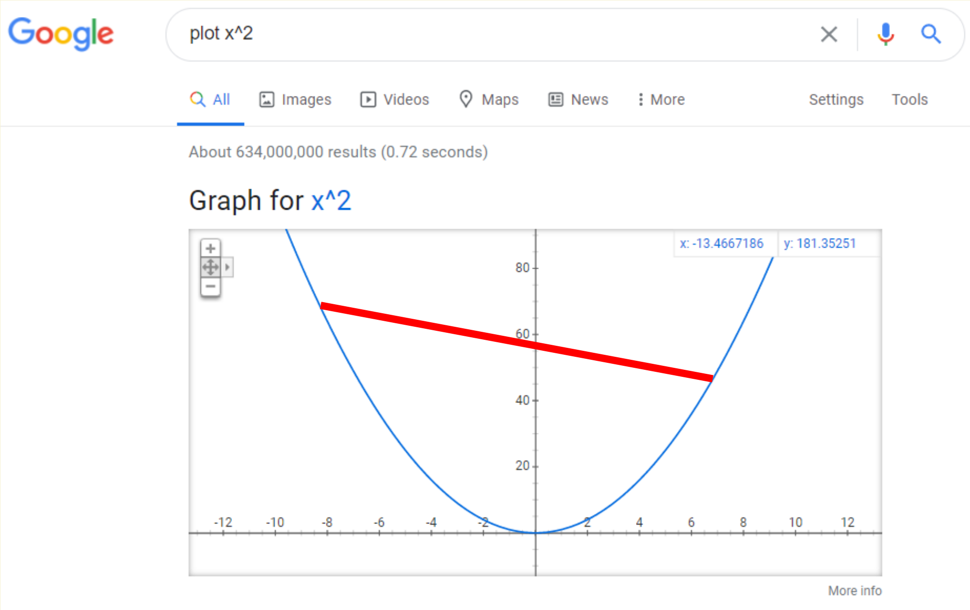
and (more importantly) $\langle w, x \rangle \approx y$ for fresh (x, y)



THM: GD / SGD on $\mathcal{L}(w) = \|Xw - y\|^2$ converges to $\arg\min_{w: Xw = y} \|w\|^2$

$$= \lim_{\lambda \to 0} \arg\min_{w} \|Xw - y\|^2 + \lambda \|w\|^2$$

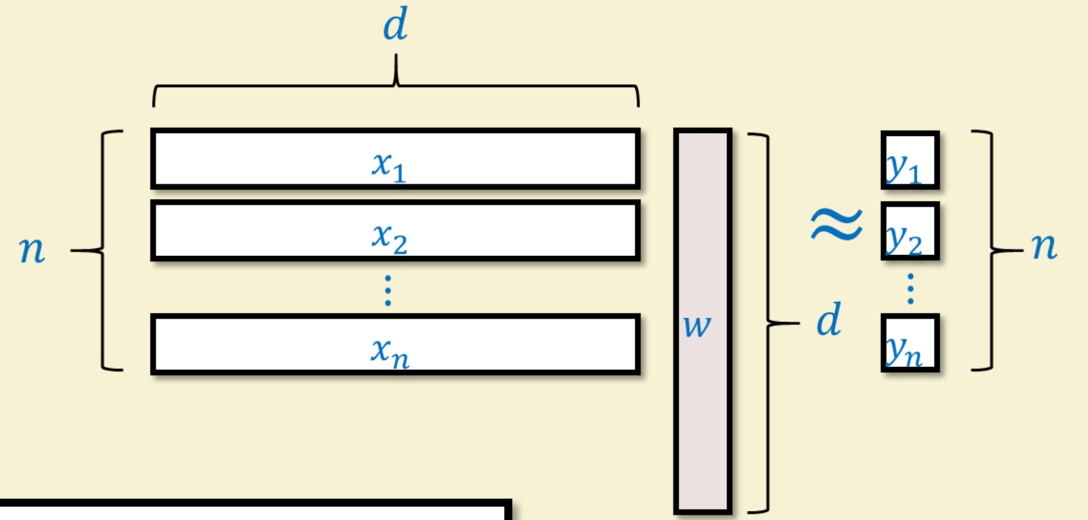
Convexity reminders



- $f(x) = x^2$ is convex
- If $f: \mathbb{R}^k \to \mathbb{R}$ convex and $g: \mathbb{R}^d \to \mathbb{R}^k$ linear
  $f \circ g$ (i.e. $x \mapsto f(g(x))$) is convex
- If $f_1, \ldots, f_m$ convex and $\alpha_1, \ldots, \alpha_m \geq 0$
  $\sum \alpha_i f_i$ convex
- If f convex and $\lambda > 0$
  $g(x) = f(x) + \lambda ||x||^2$ strongly convex.

Linear regression SGD

$$\arg\min \|Xw - y\|^2$$



1. Let $w_0 \leftarrow 0^d$
2. For $t = 0,1, \ldots$:
   - Pick $i \sim [n]$
   - Let $w_{t+1} = w_t - \eta$ ~~$\nabla_w (\langle x_i, w\rangle - y_i)^2$~~ $x_i^\top(\langle x_i, w\rangle - y_i)$

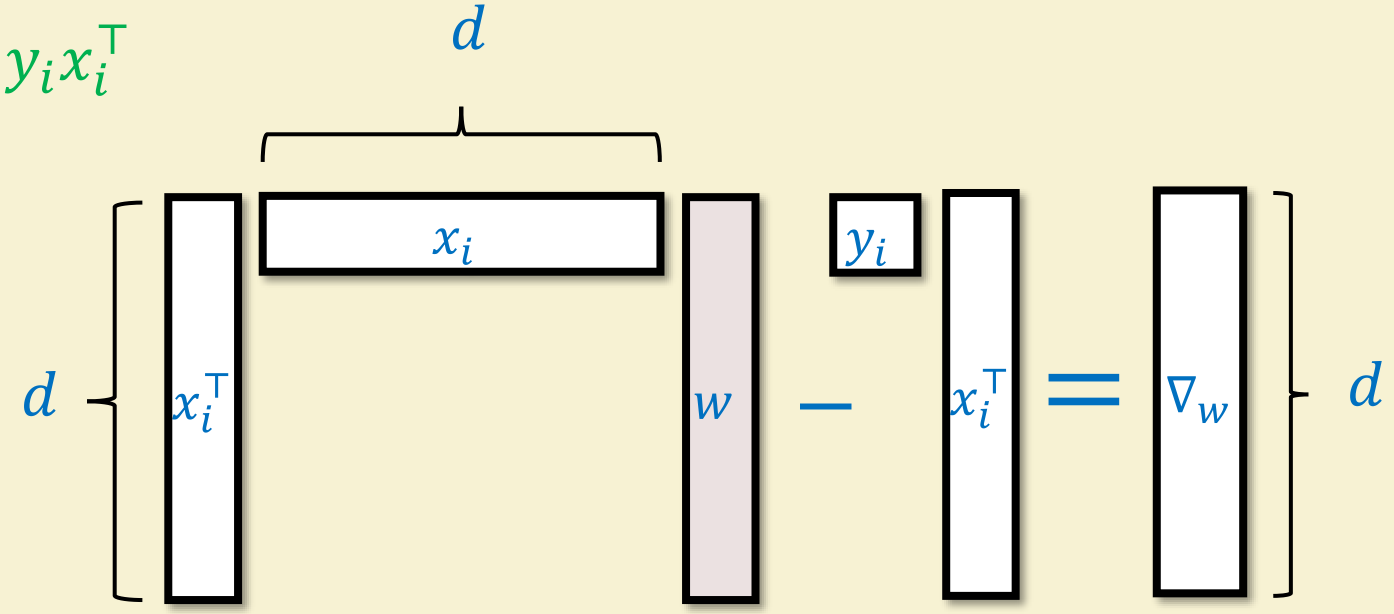
Can ignore factor of 2

CLAIM: $\nabla(\langle x_i, w\rangle - y_i)^2 = 2\, x_i^\top x_i w - 2 y_i x_i^\top$

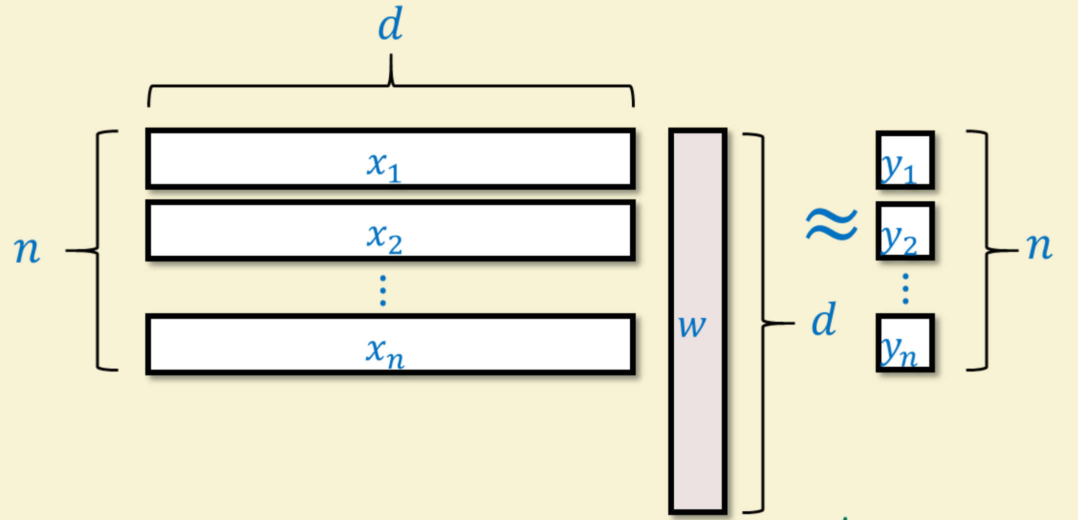
"PF":

i) In one dim $\frac{d(xw-y)^2}{dw} = 2\mathrm{x}^2\mathrm{w} - 2\mathrm{yx}$

ii) Dimensions match



Linear regression SGD



$$\arg\min \|Xw - y\|^2$$

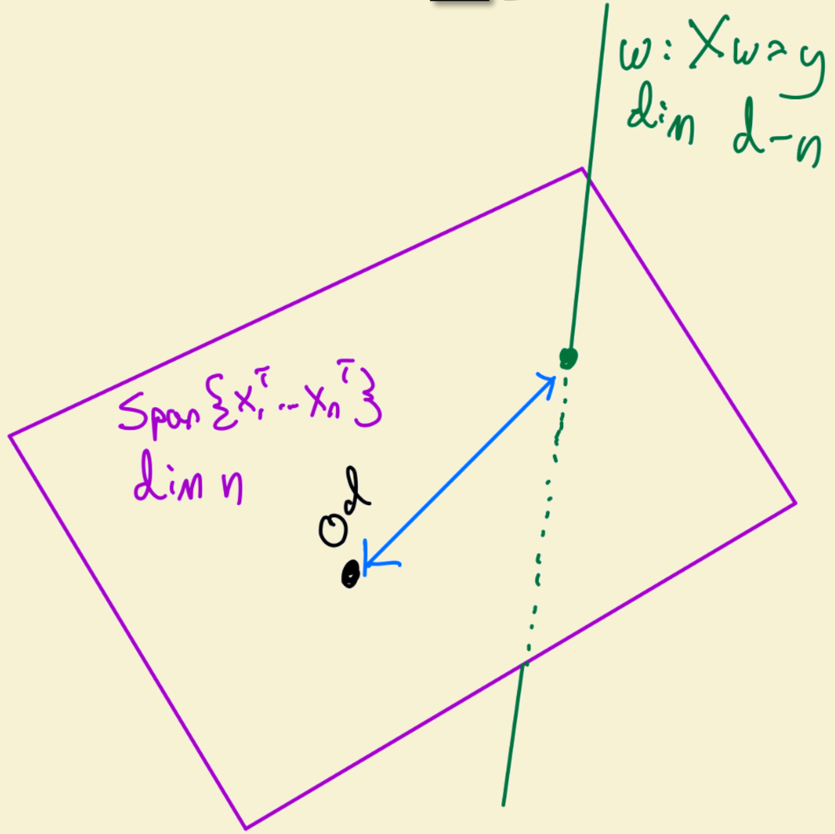
1. Let $w_0 \leftarrow 0^d$
2. For $t = 0,1,\ldots$:
   - Pick $i \sim [n]$
   - Let $w_{t+1} = w_t - \eta\, x_i^\top (\langle x_i, w\rangle - y_i)$

COR 1: If $w_t \in Span\,\{x_1^\top .. x_n^\top\}$ then $w_{t+1} \in Span\,\{x_1^\top .. x_n^\top\}$

COR 2: If $rank(X) = n$ then
$Xw_\infty = y$ & $w_\infty \in Span\,\{x_1^\top .. x_n^\top\}$

COR 3: $w_\infty = \arg\min_{w:Xw=y} \|w\|^2$

COR 4: $w_\infty = \lim_{\lambda\to 0} \arg\min_{w} \|Xw - y\|^2 + \lambda\|w\|^2$



GD / SGD dynamics

$$w_{t+1} = w_t - \eta\, \nabla\|Xw_t - y\|^2 \quad = w_t - \eta\nabla(w_t^\top X^\top X w_t - w_t^\top X^\top y)(w_t)$$

$$= w_t - \eta(X^\top X w_t - X^\top y)$$

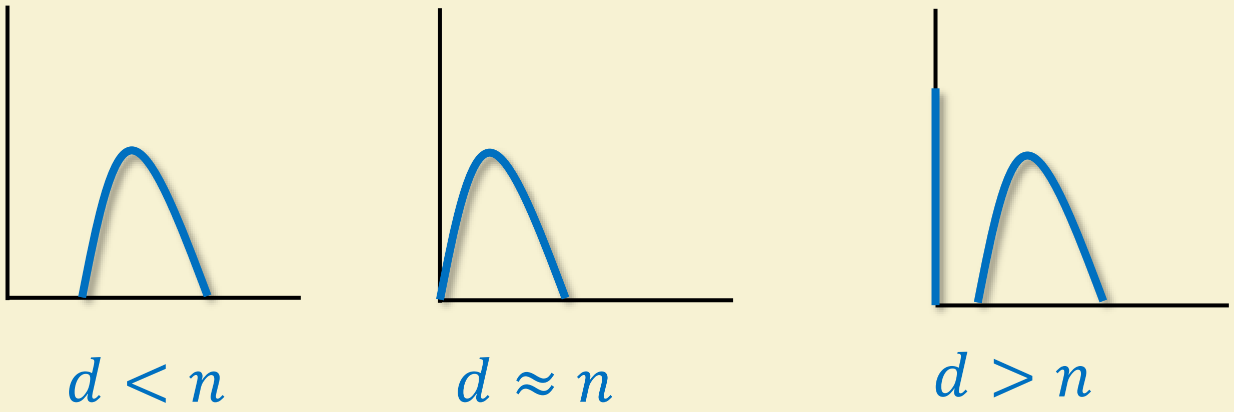
Let w_∞ s.t. $Xw_\infty = y$. Then $w_{t+1} = w_t - \eta(X^\top X w_t - X^\top X w_\infty)$

$$w_{t+1} - w_\infty = (I - \eta X^\top X)(w_t - w_\infty)$$

Make progress as long as $0 \prec I - \eta X^\top X \prec 1$: $\eta < \frac{1}{\lambda_1}$, progress $\approx \frac{\lambda_d}{\lambda_1} = \frac{1}{\kappa}$

$$X^\top X = \begin{pmatrix} \lambda_1 & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \lambda_d \end{pmatrix}$$

Random X:

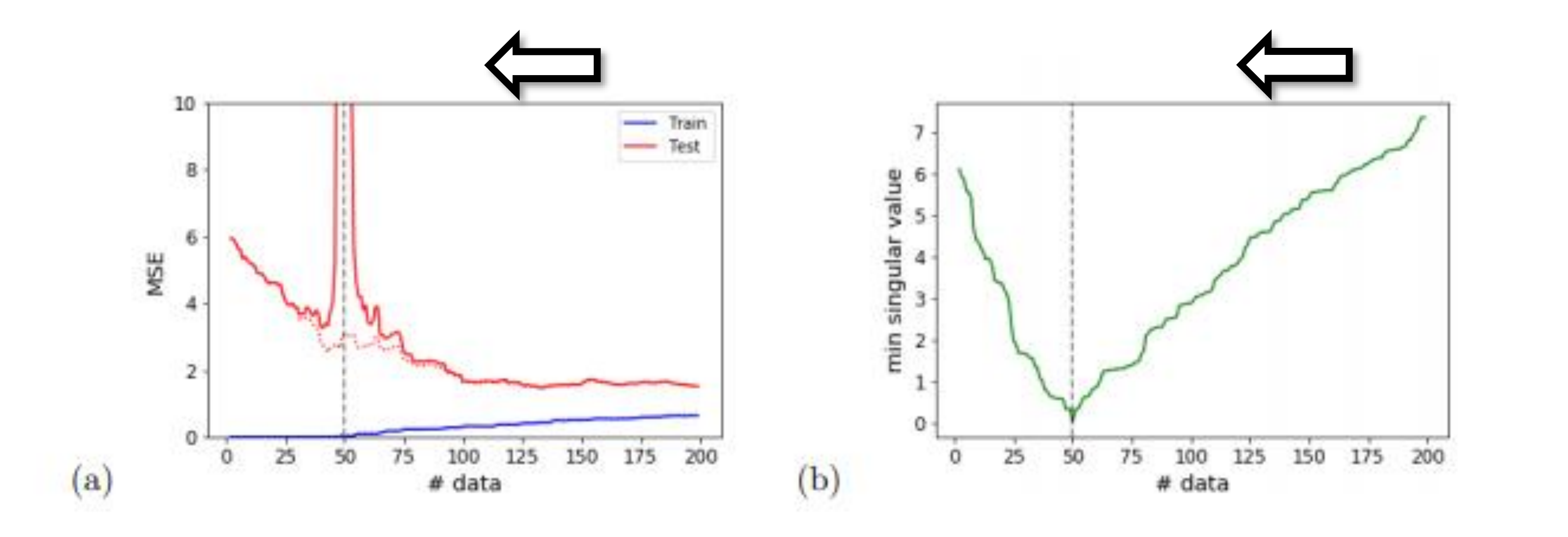


* dropping 2's throughout

Actual GD / SGD

$$w_{t+1} = w_t - \eta \nabla \|$$

Let w_∞ s.t. $Xw_\infty = y$

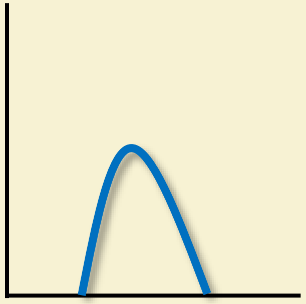


$$w_{t+1} - w_\infty = (I - \eta X^\top X)(w_t - w_\infty)$$

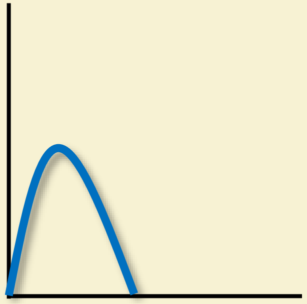
Make progress as long as $0 \prec I - \eta X^\top X \prec 1$: $\eta < \frac{1}{\lambda_1}$, progress $\approx \frac{\lambda_d}{\lambda_1} = \frac{1}{\kappa}$

$$X^\top X = \begin{pmatrix} \lambda_1 & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \lambda_d \end{pmatrix}$$

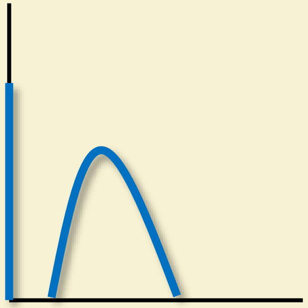
Random X:



$d < n$



$d \approx n$



$d > n$

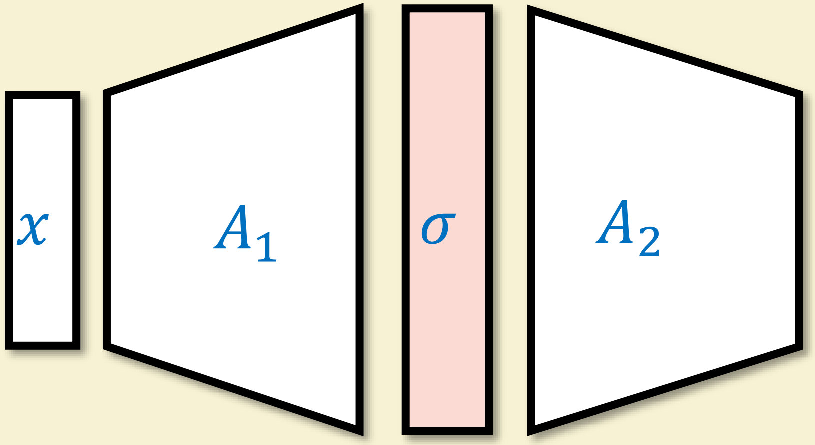
Grosse lecture notes

Hastie-Montanari-Rosset-Tibshirani, 20

Beyond linear regression

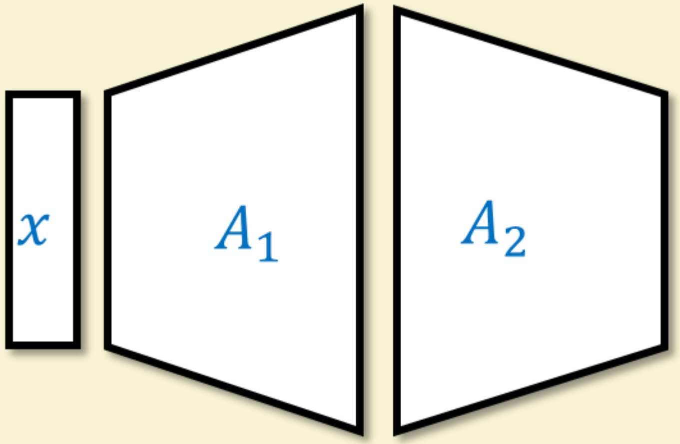
Implicit regularization in deep networks

Depth 2 network



Parameter space: $\mathbb{R}^{d\times h + h\times m}$

Depth 2 linear network

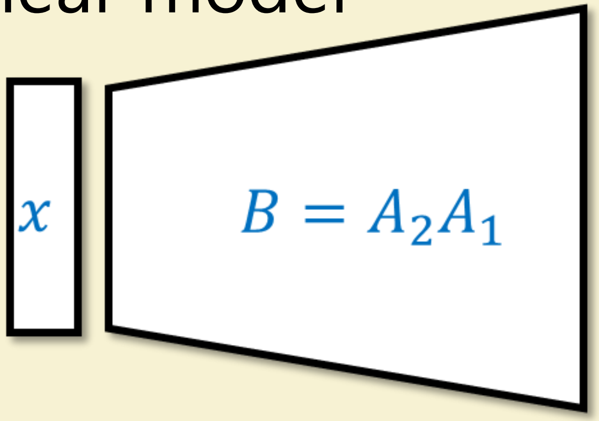


Parameter space: $\mathbb{R}^{d\times h + h\times m}$

$Bx = A_2 A_1 x$:
Same expressiveness / functional space

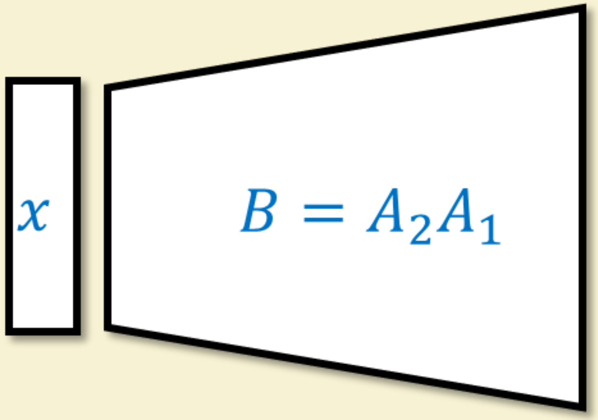
Different parameter space

Linear model



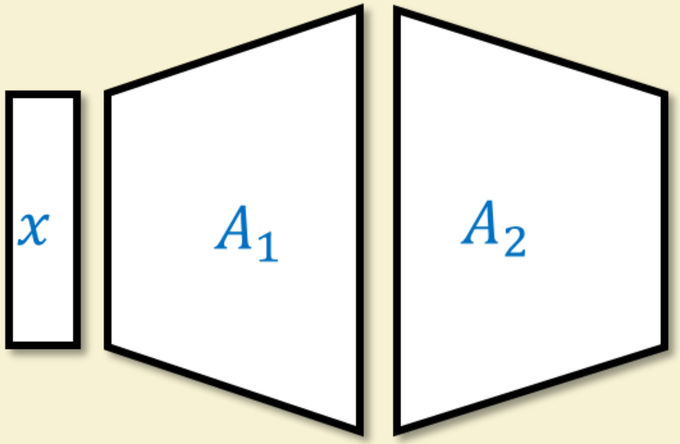
Parameter space: $\mathbb{R}^{d\times m}$

Linear model



Parameter space: $\mathbb{R}^{d\times m}$

Depth 2 linear network



Parameter space: $\mathbb{R}^{d\times h + h\times m}$

For every loss function $\mathcal{L}$:

$$\min\ \mathcal{L}(B) \quad = \quad \min\ \mathcal{L}(A_1, A_2)$$

BUT

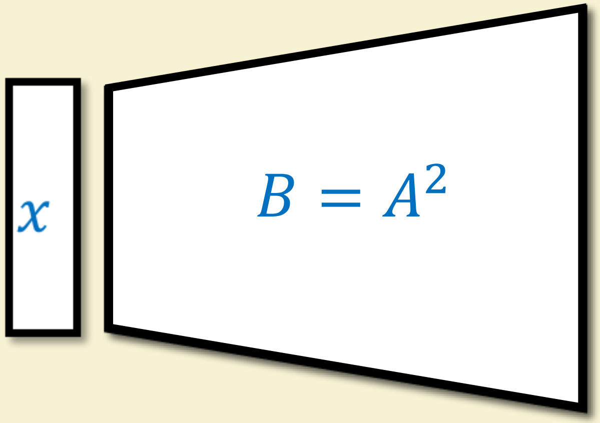
SGD/GD on ↑ $\neq$ SGD/GD on ↑

Potentially convex function in $B \in \mathbb{R}^{d\times m}$

Non-convex function in $(A_1, A_2) \in \mathbb{R}^{d\times h + h\times m}$

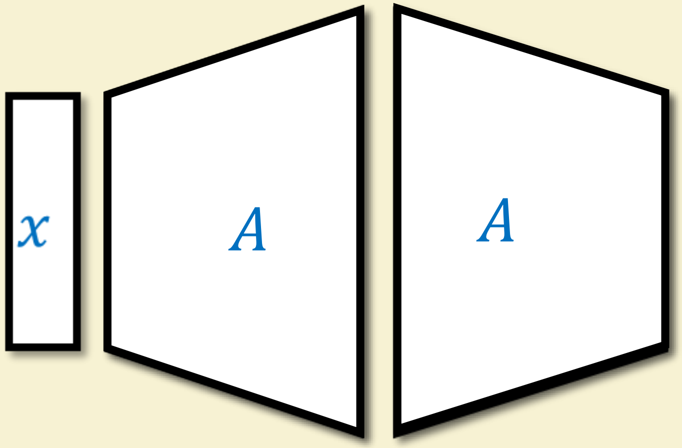
Gradient flow on deep linear nets

Linear model



Parameter space: $\mathbb{R}^{d\times m}$

Depth 2 linear network



Parameter space: $\mathbb{R}^{d\times h + h\times m}$

Simplifying assumptions: $A_1 = A_2$ symmetric

$$\Rightarrow \; B = A^2, A = \sqrt{B}$$

Analyze GD with $\eta \to 0$ on $\min \tilde{\mathcal{L}}(A)$ where $\tilde{\mathcal{L}}(A) = \mathcal{L}(A^2)$

Gradient flow on deep linear nets

$$\tilde{\mathcal{L}}(A) = \mathcal{L}(A^2)$$
$$B = A^2$$

$$\frac{dA(t)}{dt} = -\nabla\tilde{\mathcal{L}}(\mathrm{A(t)})$$

GF on linear model:

$$\frac{dB(t)}{dt} = -\nabla\mathcal{L}(\mathrm{B(t)})$$

$\tilde{\nabla}= A\nabla$

By chain rule $\underbrace{\nabla\tilde{\mathcal{L}}(A)}_{\tilde{\nabla}} = \underbrace{\nabla\mathcal{L}(A^2)}_{\nabla}A = \mathrm{A}\nabla\mathcal{L}(A^2)$

Hence $\frac{dA^2(t)}{dt} = \frac{dA(t)}{dt}\cdot A \;=\; -\tilde{\nabla}\cdot A = -A\cdot\nabla\cdot A$

GF on deep linear net $B = A^2$:

$$\frac{dB(t)}{dt} = -A\,\nabla\mathcal{L}\big(B(t)\big)A = -\sqrt{B}\,\nabla\mathcal{L}\big(B(t)\big)\sqrt{B}$$

"The big get bigger"

* dropping 2's throughout

GF on deep linear net $B = A^2$:

$$\frac{dB(t)}{dt} = -A\,\nabla\mathcal{L}\big(B(t)\big)A = -\sqrt{B}\,\nabla\mathcal{L}\big(B(t)\big)\sqrt{B}$$

Generally GF on deep linear net B evolves* by

$$\frac{dB(t)}{dt} = -\psi_{B(t)}\big(\nabla\mathcal{L}\big(B(t)\big)\big)$$

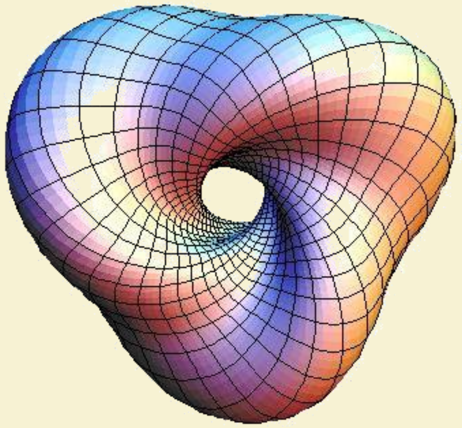
$$\psi_B(\nabla) =^{*} \sum B^{\alpha}\,\nabla B^{1-\alpha}$$

Gradient flow on a Riemannian Manifold

* not equivalent to $\min \mathcal{L}(B) + \lambda\, R(B)$

Saxe, McClelland, Ganguli 2013
Arora, Cohen, Hazan, 2018
Bah, Rauhut, Terstiege, Westdickenberg, 2019

Riemannian Manifolds



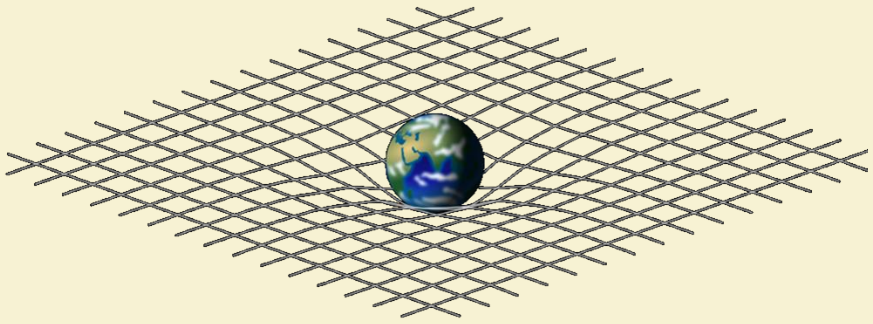
External description: A smooth subset $\mathcal{M} \subseteq \mathbb{R}^N$

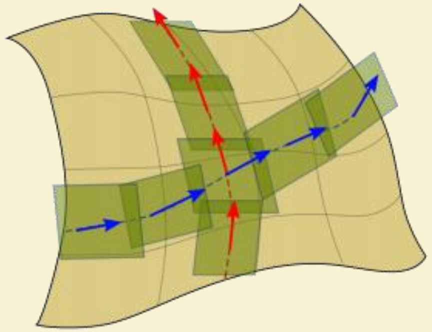
Intrinsic description: Set $\mathcal{M}$ with "local geometry" at each $x \in \mathcal{M}$

For every $x \in \mathcal{M}$, tangent space T_x - set of directions we can move in

(Gradient of $f(x)$: shortest direction from x to increase f)

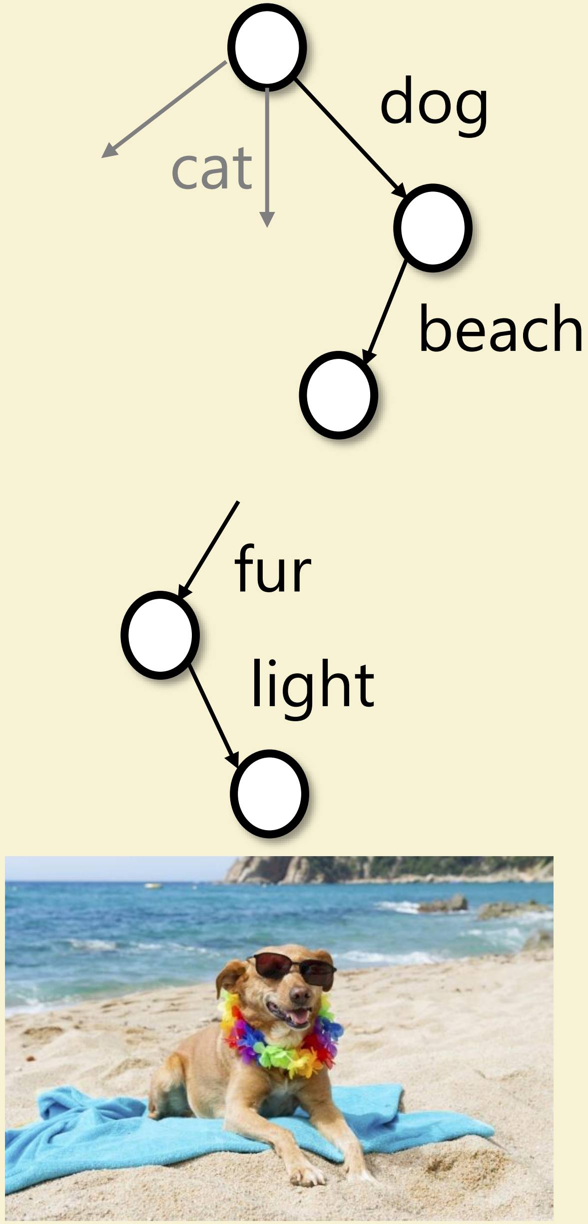
local inner product on T_x - defined via PSD matrix M_x on T_x

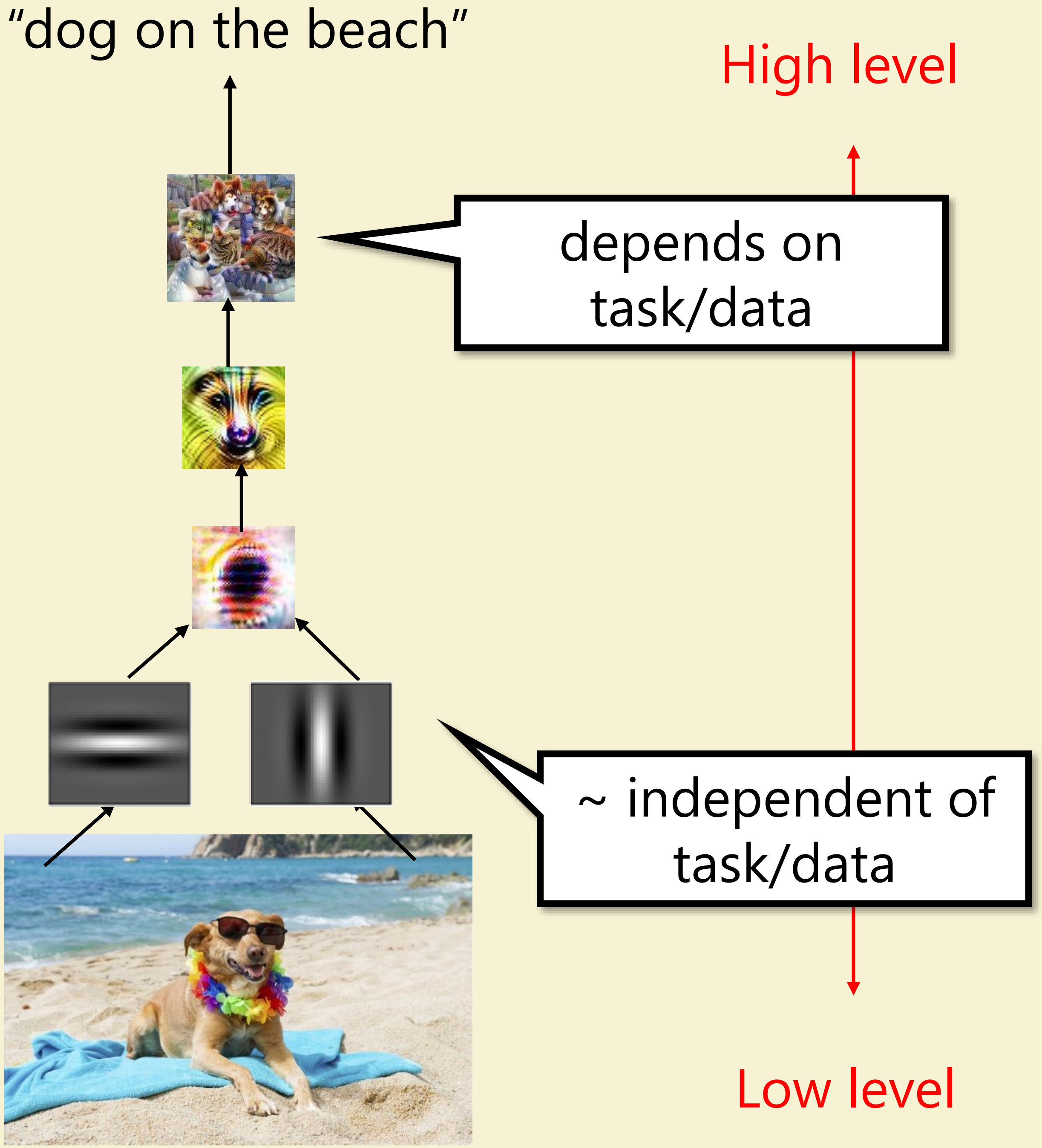




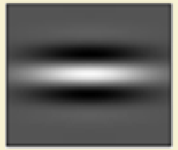
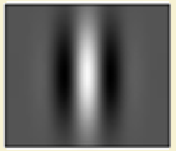
Learning in different layers

Cartoon

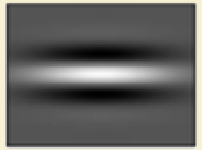
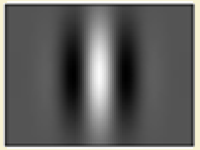
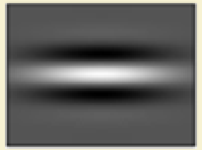
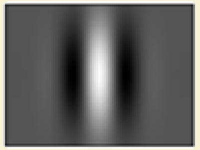

dog
cat
beach
fur
light

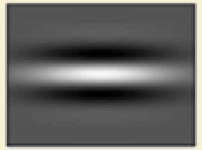
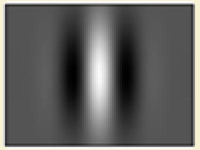
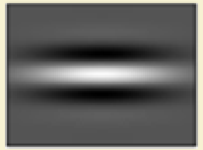
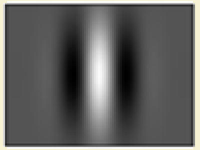

"dog on the beach"
High level
depends on task/data
~ independent of task/data
Low level

Non-convexity & symmetry breaking

$\frac{1}{2}$  $+ \ \frac{1}{2}$  $= \ JUNK$

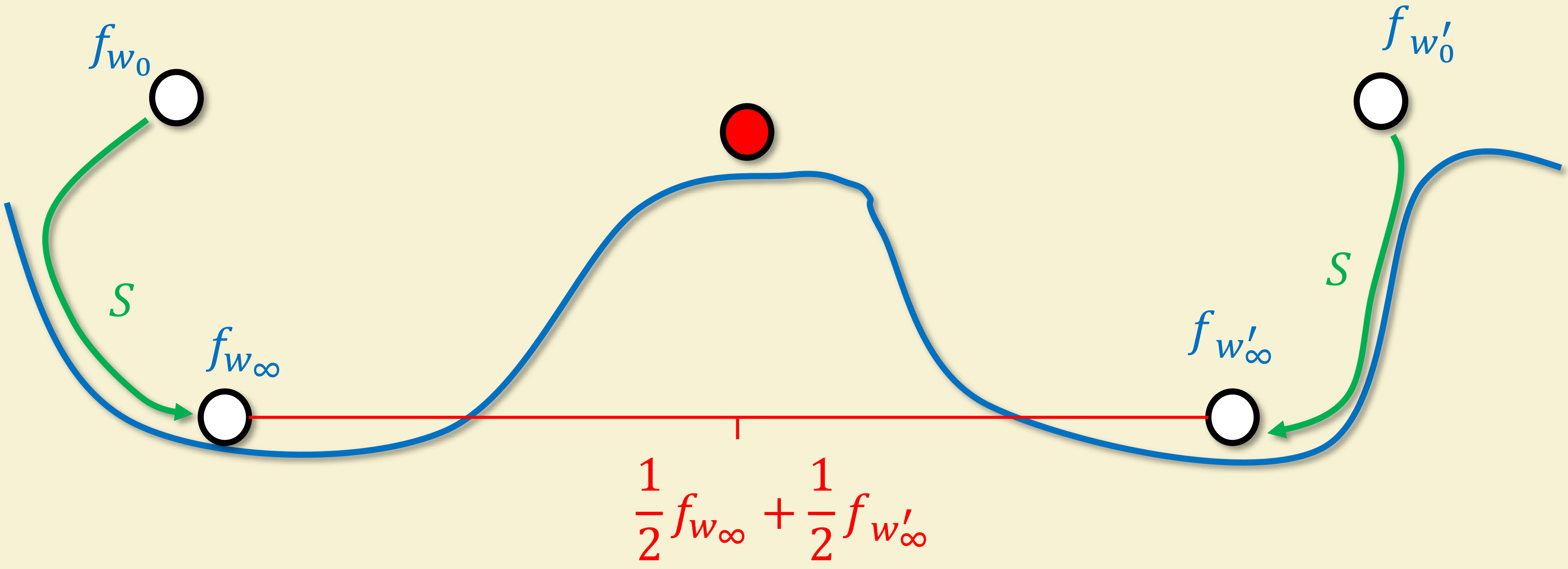
Intuition:

Initial weights: 0.49  $+ \ 0.51$  0.51  $+ \ 0.49$ 

Final weights: 0.01  $+ \ 0.99$  0.99  $+ \ 0.01$ 

(Too) strong hypothesis: All nets are similar up to transformations, depending on initialization and data

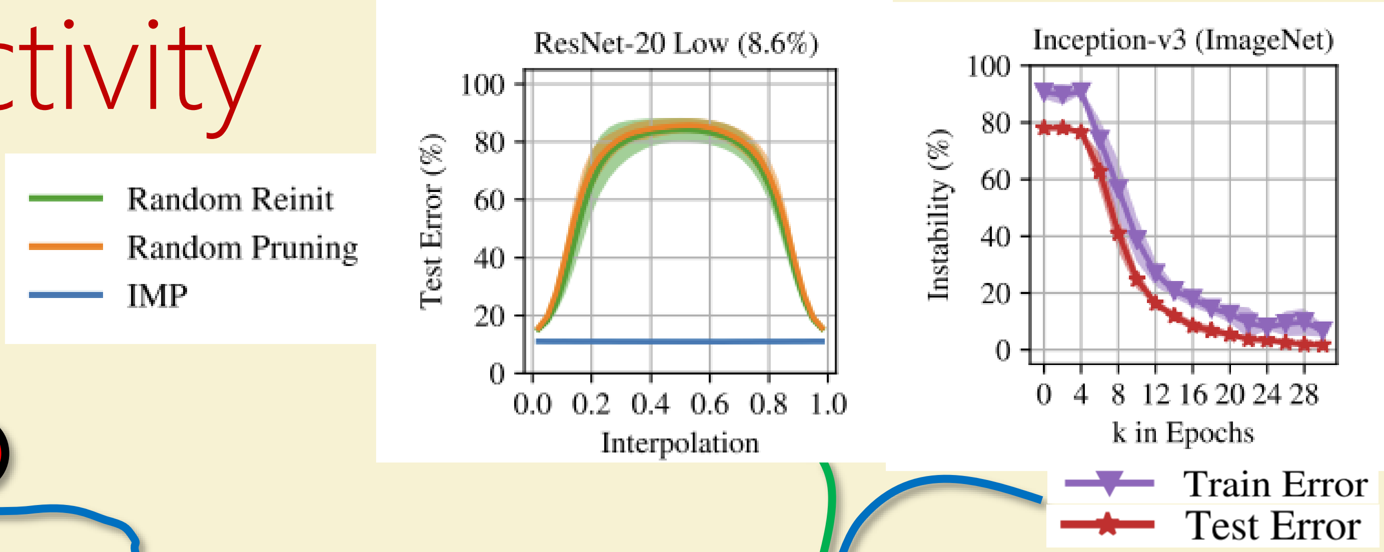
Linear mode connectivity

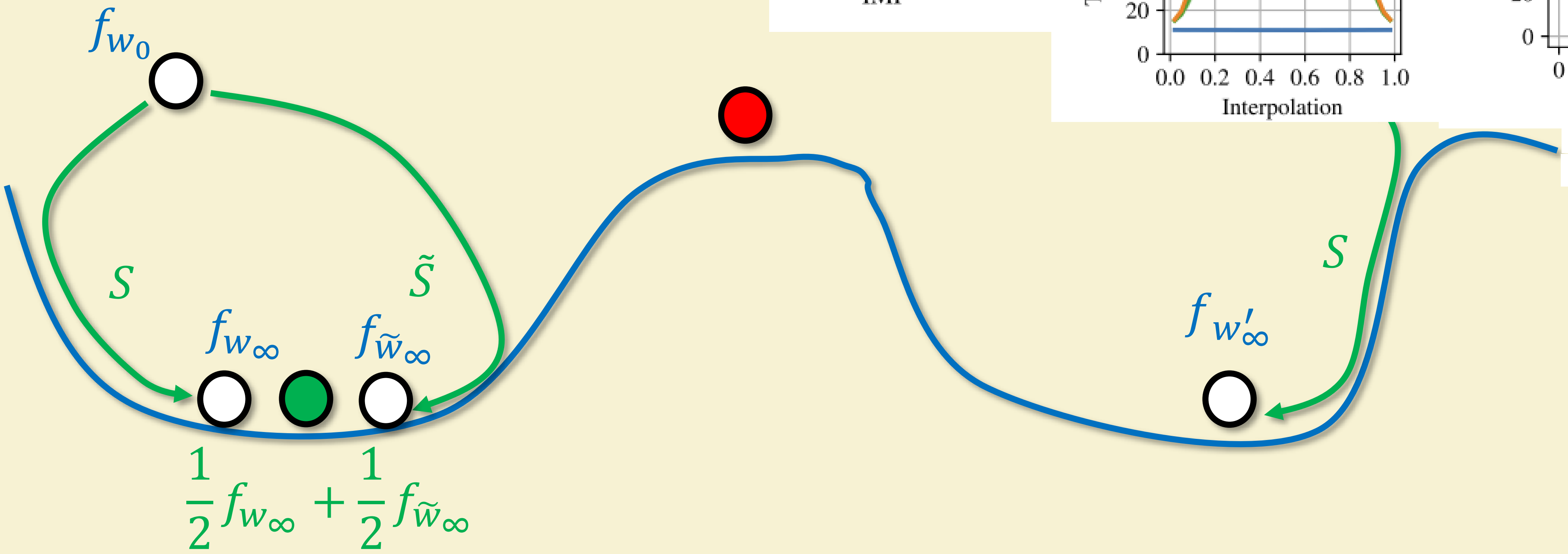


$$\mathcal{L}\left(\frac{1}{2} f_{w_\infty} + \frac{1}{2} f_{w'_\infty}\right) \gg \frac{1}{2}\mathcal{L}(f_{w_\infty}) + \frac{1}{2}\mathcal{L}(f_{w'_\infty})$$

Frankle, Dziugaite, Roy, Carbin, 2019

Linear mode connectivity





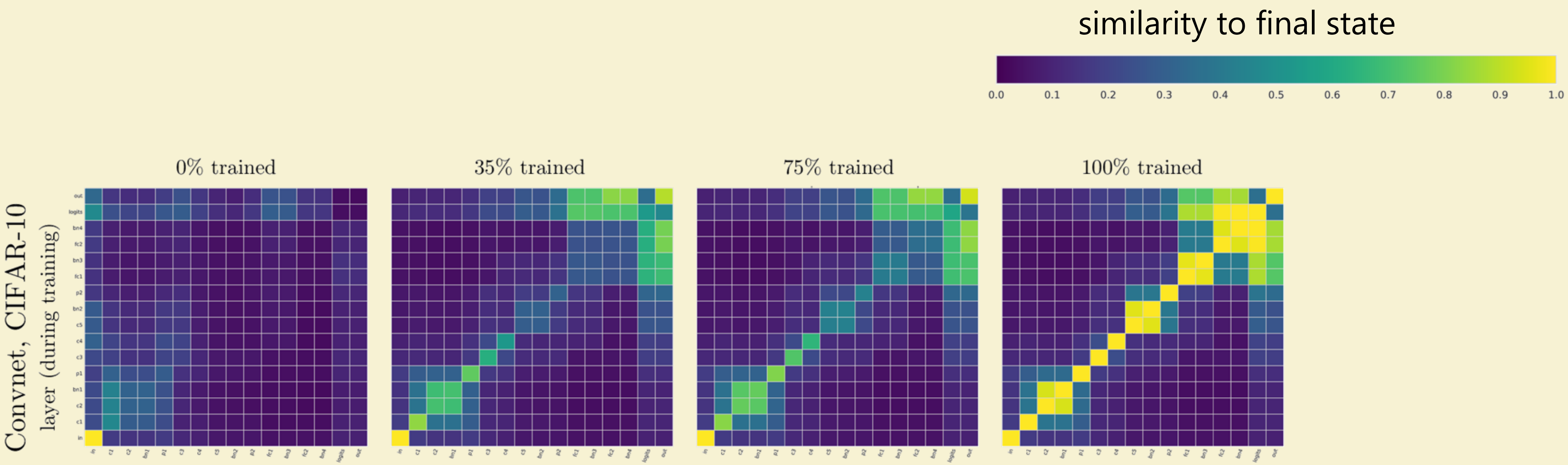
$$\mathcal{L}\left(\frac{1}{2} f_{w_\infty} + \frac{1}{2} f_{w'_\infty}\right) \gg \frac{1}{2}\mathcal{L}(f_{w_\infty}) + \frac{1}{2}\mathcal{L}(f_{w'_\infty})$$

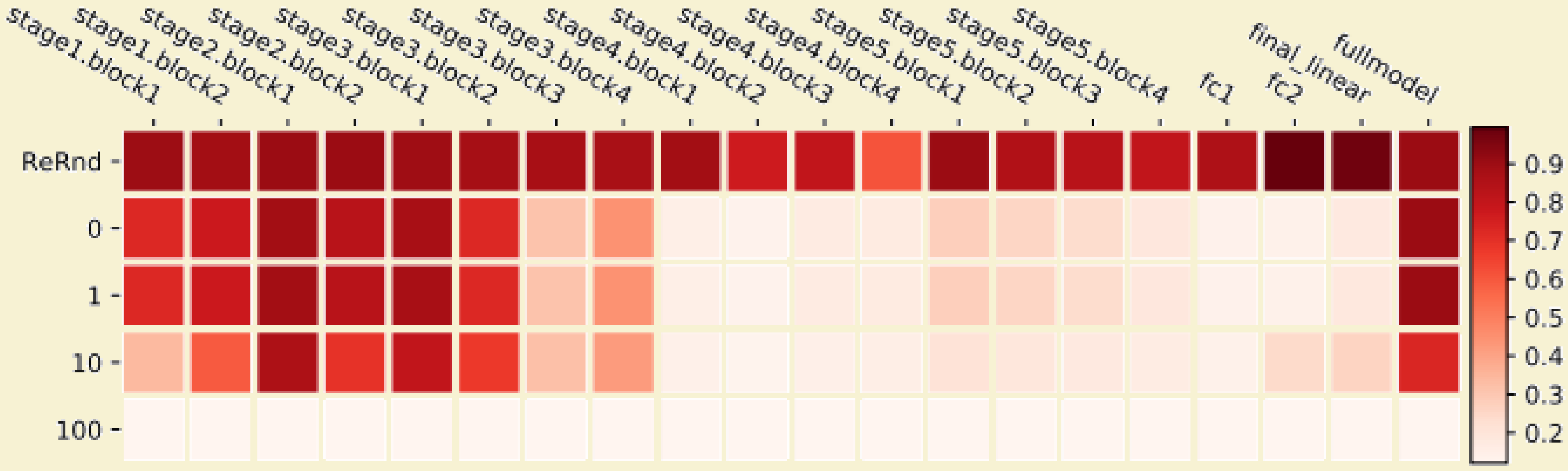
$$\mathcal{L}\left(\frac{1}{2} f_{w_\infty} + \frac{1}{2} f_{\tilde{w}_\infty}\right) \approx \frac{1}{2}\mathcal{L}(f_{w_\infty}) + \frac{1}{2}\mathcal{L}(f_{\tilde{w}_\infty})$$

Frankle, Dziugaite, Roy, Carbin, 2019

* After pruning / from w_k for $k > 0$

Layers





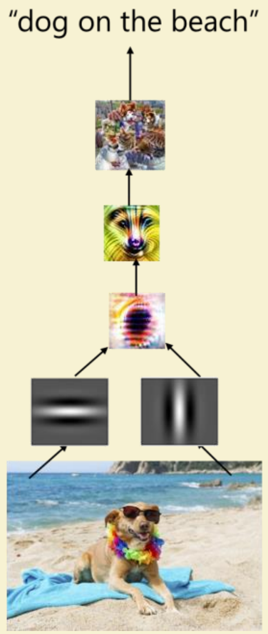
randomness just for symmetry breaking!

Raghu, Gilmer, Yosinski, Sohl-Dickstein, 2017

Zhang, Bengio, Singer 2019

Theoretical insights

Intuition: If data doesn't contain "local correlations" then "can't get off the ground" – learning will not succeed.



Failures of Gradient-Based Deep Learning

Shai Shalev-Shwartz[1], Ohad Shamir[2], and Shaked Shammah[1]

Is Deeper Better only when Shallow is Good?

Eran Malach and Shai Shalev-Shwartz

Poly-time universality and limitations of deep learning

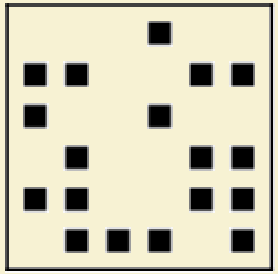
Emmanuel Abbe, EPFL — Colin Sandon, MIT

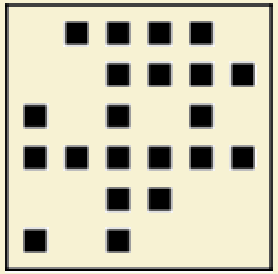
Memory, Communication, and Statistical Queries

Jacob Steinhardt* JSTEINHARDT@CS.STANFORD.EDU
Gregory Valiant† VALIANT@STANFORD.EDU
Stefan Wager‡ SWAGER@STANFORD.EDU
Departments of Computer Science,† and Statistics‡, Stanford University, Stanford, CA-94305, USA*

CONJ/THM: If X, Y k-wise independent for moderate k then can't learn

Canonical "hard" example: parities





For $\mathrm{I} \subseteq [d]$, D_I defined as: $x \sim \{\pm 1\}^d, y = \prod_{i \in I} x_i$

$$= \begin{cases} -1, & \mathrm{num}_{-1}(x_I) \text{ odd} \\ +1, & \mathrm{num}_{-1}(x_I) \text{ even} \end{cases}$$

Example: $d = 7$, $I = \{1,3,6,7\}$

1	2	**3**	4	5	**6**	**7**	
$+1$	$+1$	-1	$+1$	$+1$	$+1$	-1	$+1$
-1	-1	-1	$+1$	-1	-1	$+1$	-1
-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1

CLAIM: Given $2d$ samples $(x_i, y_i)_{i=1..2d} \sim D_I$ can recover I

Canonical “hard” example: par

$$= \begin{cases} -1, & \text{num}_{-1}(x_I) \text{ odd} \\ +1, & \text{num}_{-1}(x_I) \text{ even} \end{cases}$$

For $\mathrm{I} \subseteq [d]$, D_I defined as: $x \sim \{\pm 1\}^d, y = \prod_{i \in I} x_i$

CLAIM: Given $2d$ samples $(x_i, y_i)_{i=1..2d} \sim D_I$ can recover I

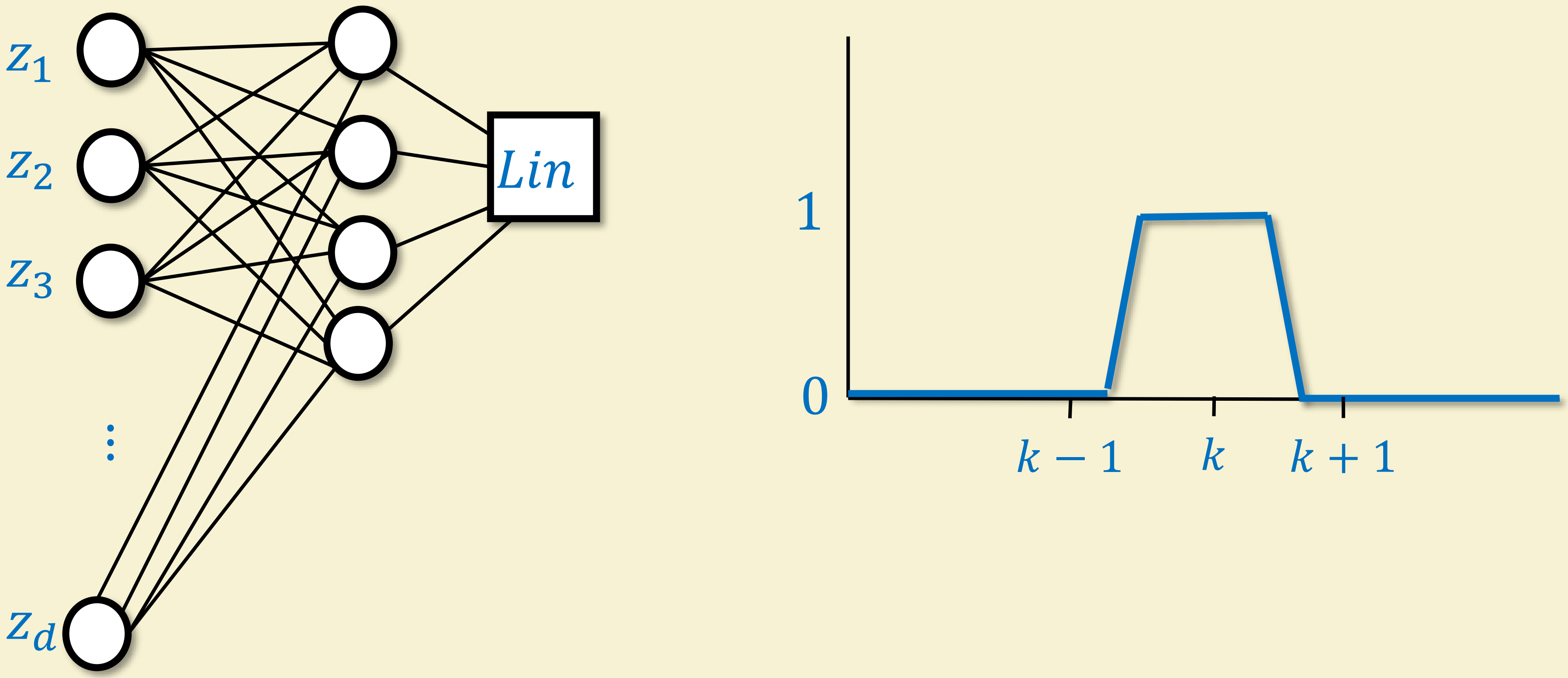
PROOF: Let $Z_{i,j} = (1 - x_{i,j})/2$ and $b_i = (1 - y_i)/2$

Let $s_i = 1$ if $i \in I$ and $s_i = 0$ otherwise

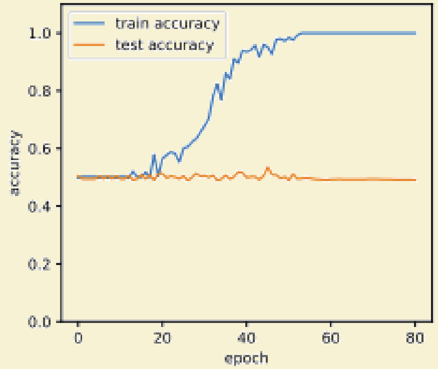
Then for every i, $\sum_j Z_{i,j} s_j = b_i \pmod 2$

$2d$ linear equations modulo 2 in d variables $s_1, \ldots, s_d$! $\blacksquare$

Parities can be expressed by few ReLUs



..but are hard to learn



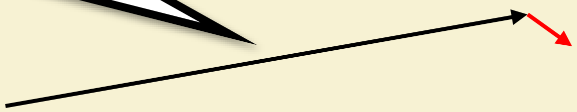
THM: For every* NN architecture $f_w(x)$, SGD on $\min \|f_w(x) - \prod_{i\in I} x_i\|^2$ will require $\exp(\Omega(d))$ steps.

Key fact: For fixed w define r.v. $\mathrm{D_w} = \nabla\|f_w(x) - \prod_{i\in I} x_i\|^2(w)$ over the choice of $\mathrm{x} \sim \{\pm 1\}^d$, $I \subseteq [d]$. Then

$$Var(D_w) \leq \frac{poly(d)}{2^d}$$

Possibly large but independent of I

Exponentially tiny



Key fact: For fixed w define r.v. $\mathrm{D_w} = \nabla\|f_w(x) - \prod_{i\in I} x_i\|^2(w)$ over the choice of $\mathrm{x} \sim \{\pm 1\}^d$, $I \subseteq [d]$. Then

$$Var(D_w) \leq \frac{poly(d)}{2^d}$$

PF: Fix w & coordinate i, and let $D_x = \frac{d}{di}\|f_w(x) - \prod_{i\in I} x_i\|^2(w)$

Then $D_x = \underbrace{2 f_w(x) \frac{d}{di} f_w(x)}_{\text{Independent of } I} - 2 \underbrace{\frac{d}{di} f_w(x) \prod_{i\in I} x_i}_{\text{Depends on } I}$

Independent of I — Depends on I

LEMMA: For every $g: \mathbb{R}^d \to \mathbb{R}$, $\left(\mathbb{E}_{x\sim\{\pm 1\}^d} \mathbb{E}_{I\subseteq[d]} g(x) \prod_{i\in I} x_i\right)^2 \leq \frac{\mathbb{E}_x g(x)^2}{2^d}$

LEMMA ⇒ FACT ⇒ THM

LEMMA: For every $g\colon \mathbb{R}^d \to \mathbb{R}$, $\left(\mathbb{E}_{x\sim\{\pm1\}^d}\mathbb{E}_{I\subseteq[d]}\, g(x)\prod_{i\in I} x_i\right)^2 \le \frac{\mathbb{E}_x g(x)^2}{2^d}$

PF: $\left(\mathbb{E}_{x\sim\{\pm1\}^d}\mathbb{E}_{I\subseteq[d]}\, g(x)\prod_{i\in I} x_i\right)^2 \le \left(\mathbb{E}_x g(x)^2\right)\cdot\left(\mathbb{E}_x \left(\mathbb{E}_I \prod_{i\in I} x_i\right)^2\right)$

$$\mathbb{E}_x \left(\mathbb{E}_I \prod_{i\in I} x_i\right)^2 = \mathbb{E}_x\left(\mathbb{E}_I \prod_{i\in I} x_i\right)\left(\mathbb{E}_J \prod_{j\in J} x_j\right)$$

$$= \mathbb{E}_I \mathbb{E}_J\, \mathbb{E}_{x\sim\{\pm1\}^d} \prod_{i\in I} x_i \prod_{j\in J} x_j$$

$$\mathbb{E}_{x\sim\{\pm1\}^d} \prod_{i\in I} x_i \prod_{j\in J} x_j = \prod_{i=1}^{d} \mathbb{E}_{\sigma\in\{\pm1\}} \sigma^{n_i\in\{0,1,2\}} = \begin{cases} 1, & I = J \\ 0, & I \neq J \end{cases}$$

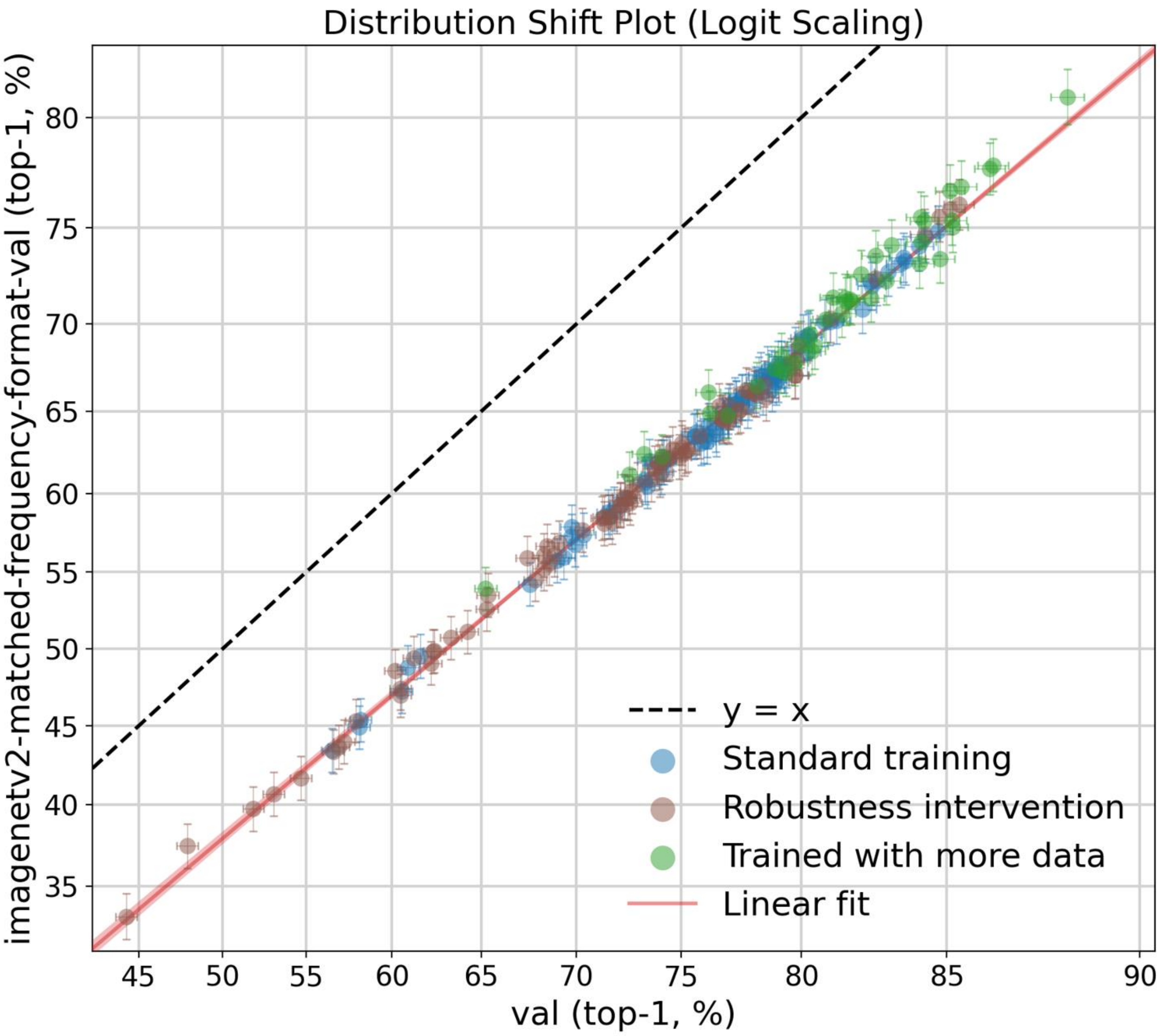
$\blacksquare$

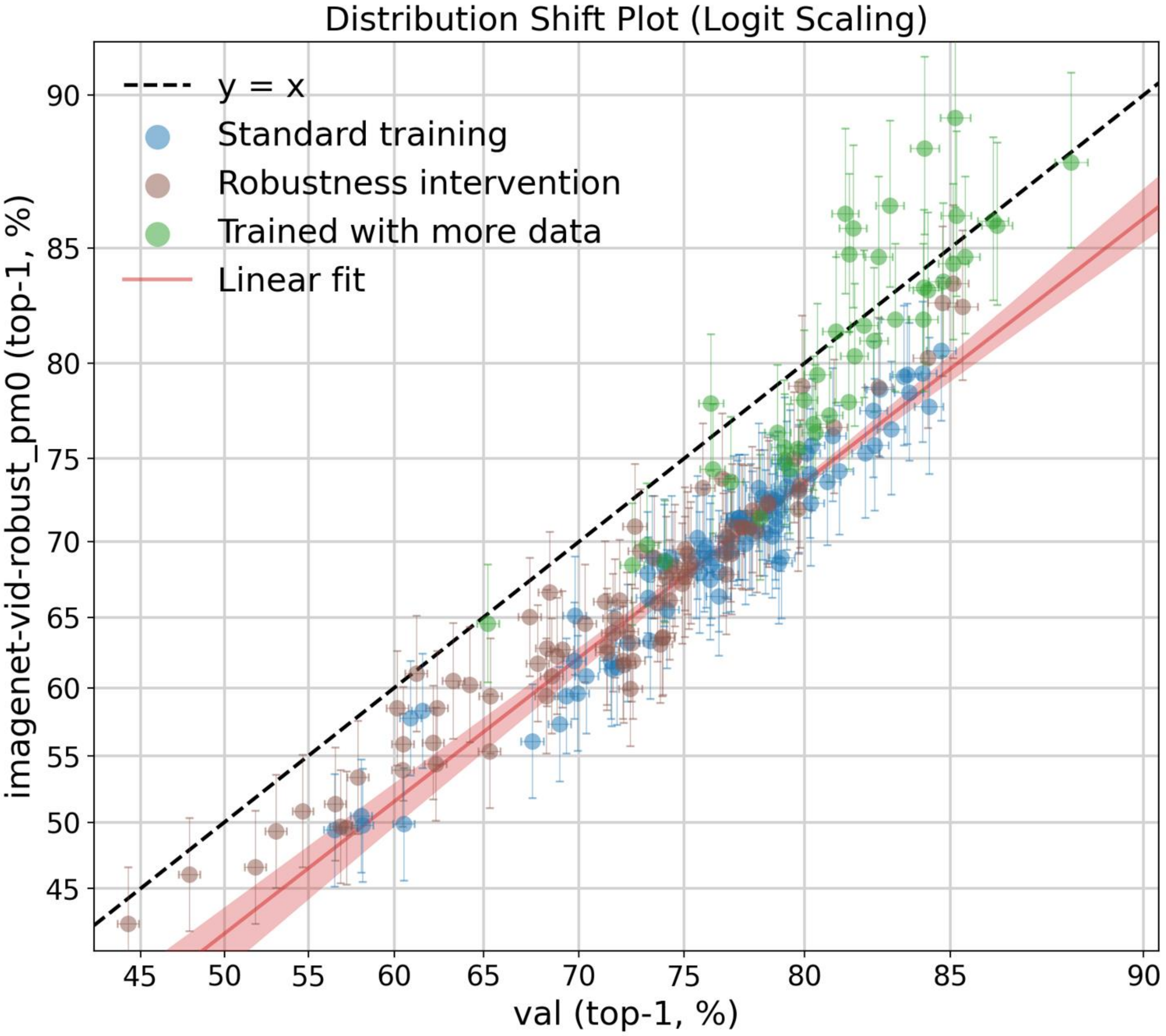
Bonus

Measuring Robustness to Natural Distribution Shifts in Image Classification

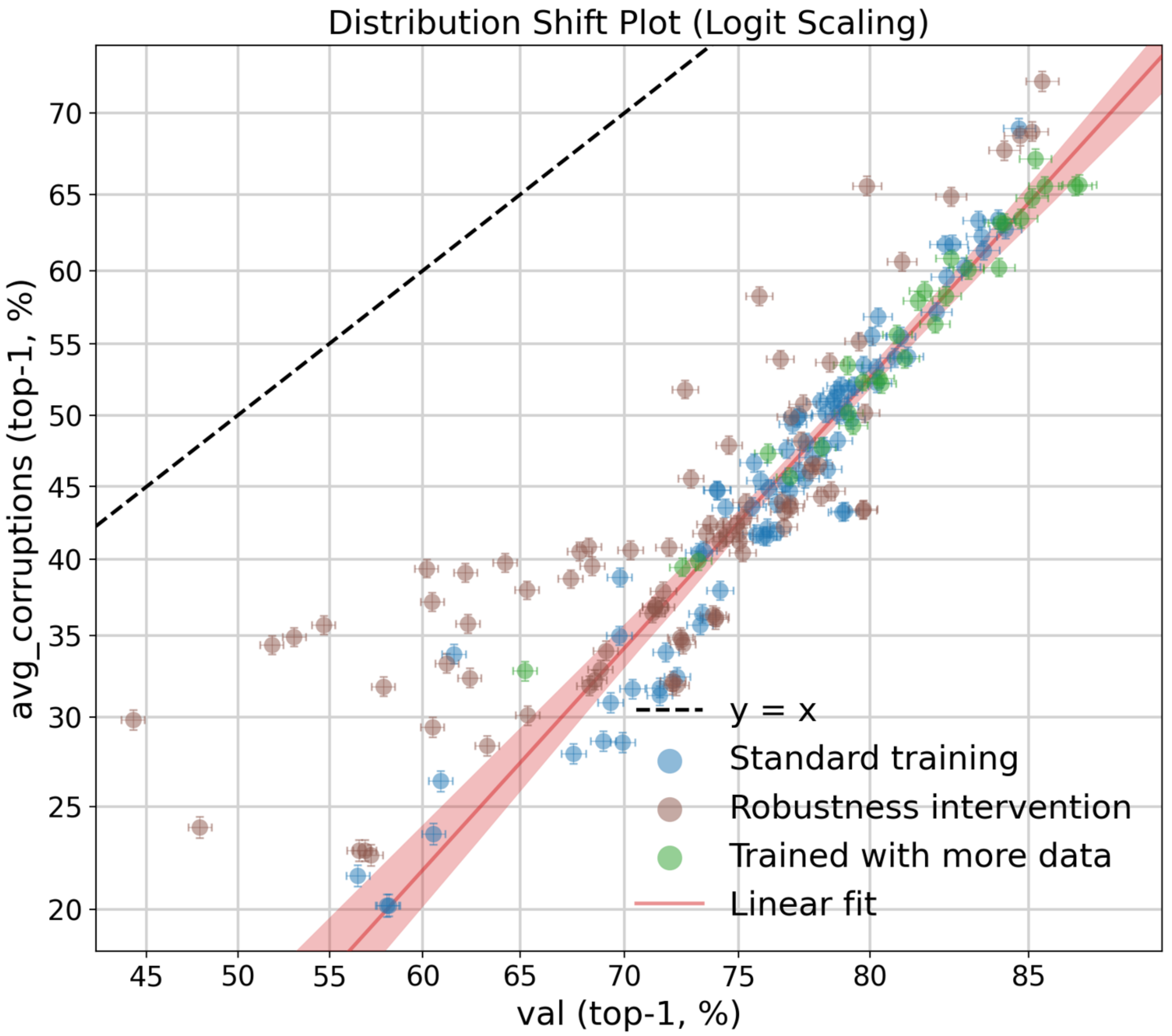
Taori, Dave, Shankar, Carlini, Recht, Schmidt

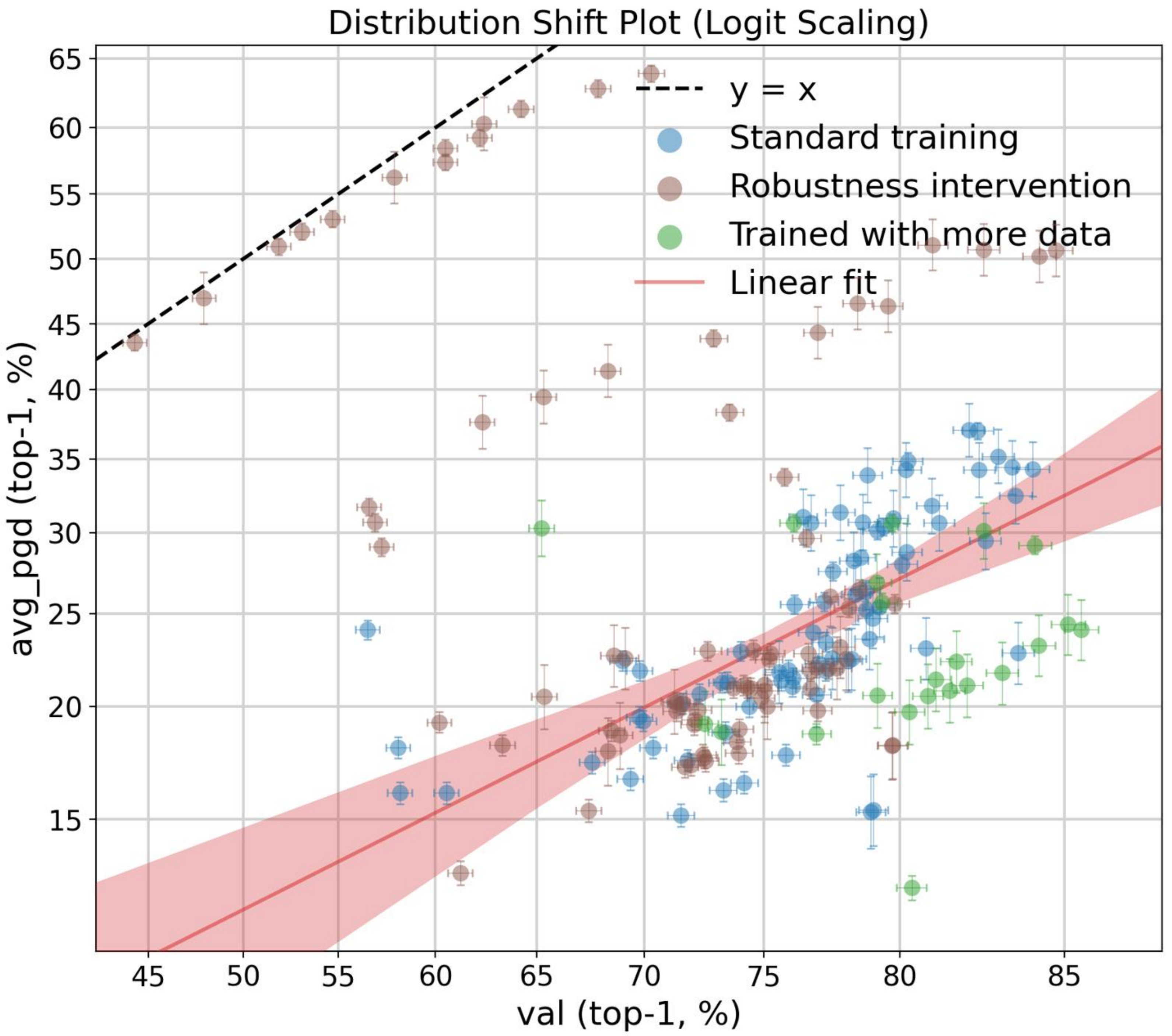
https://modestyachts.github.io/imagenet-testbed/


Distribution Shift Plot (Logit Scaling)
imagenetv2-matched-frequency-format-val (top-1, %)
val (top-1, %)
35
40
45
50
55
60
65
70
75
80
45
50
55
60
65
70
75
80
85
90
y = x
Standard training
Robustness intervention
Trained with more data
Linear fit



Distribution Shift Plot (Logit Scaling)
y = x
Standard training
Robustness intervention
Trained with more data
Linear fit
imagenet-vid-robust_pm0 (top-1, %)
val (top-1, %)
45
50
55
60
65
70
75
80
85
90


Distribution Shift Plot (Logit Scaling)
avg_corruptions (top-1, %)
val (top-1, %)
70
65
60
55
50
45
40
35
30
25
20
45
50
55
60
65
70
75
80
85
y = x
Standard training
Robustness intervention
Trained with more data
Linear fit


Distribution Shift Plot (Logit Scaling)
y = x
Standard training
Robustness intervention
Trained with more data
Linear fit
avg_pgd (top-1, %)
val (top-1, %)
65
60
55
50
45
40
35
30
25
20
15
45
50
55
60
65
70
75
80
85