Sensitivity and Generalization in Neural Networks: an Empirical Study

Roman Novak, Yasaman Bahri, Daniel A. Abolafia,
Jeffrey Pennington, Jascha Sohl-Dickstein

ICLR 2018

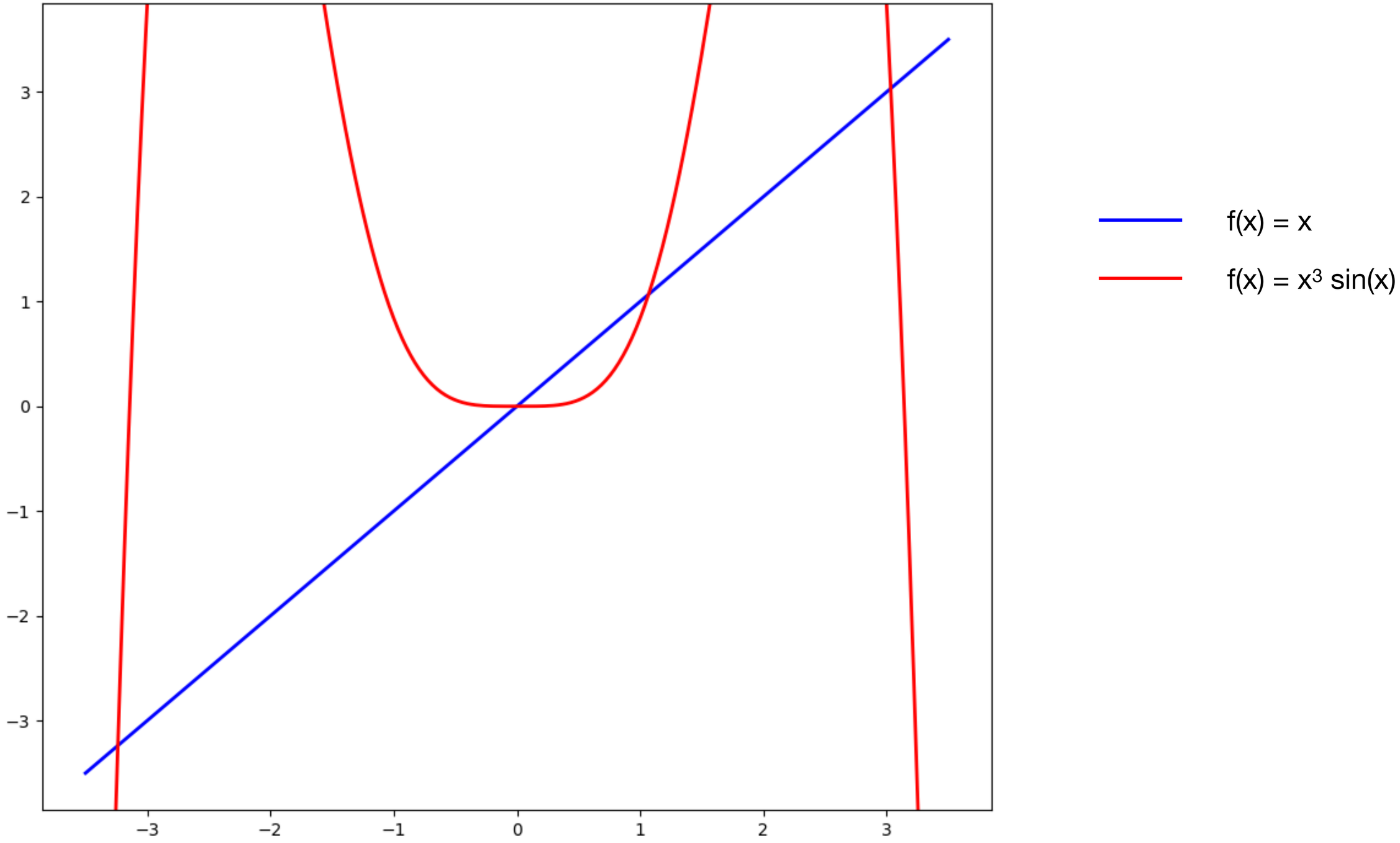
Deep Learning Classics and Trends, 31 May 2018
Paper review by Jason Yosinski

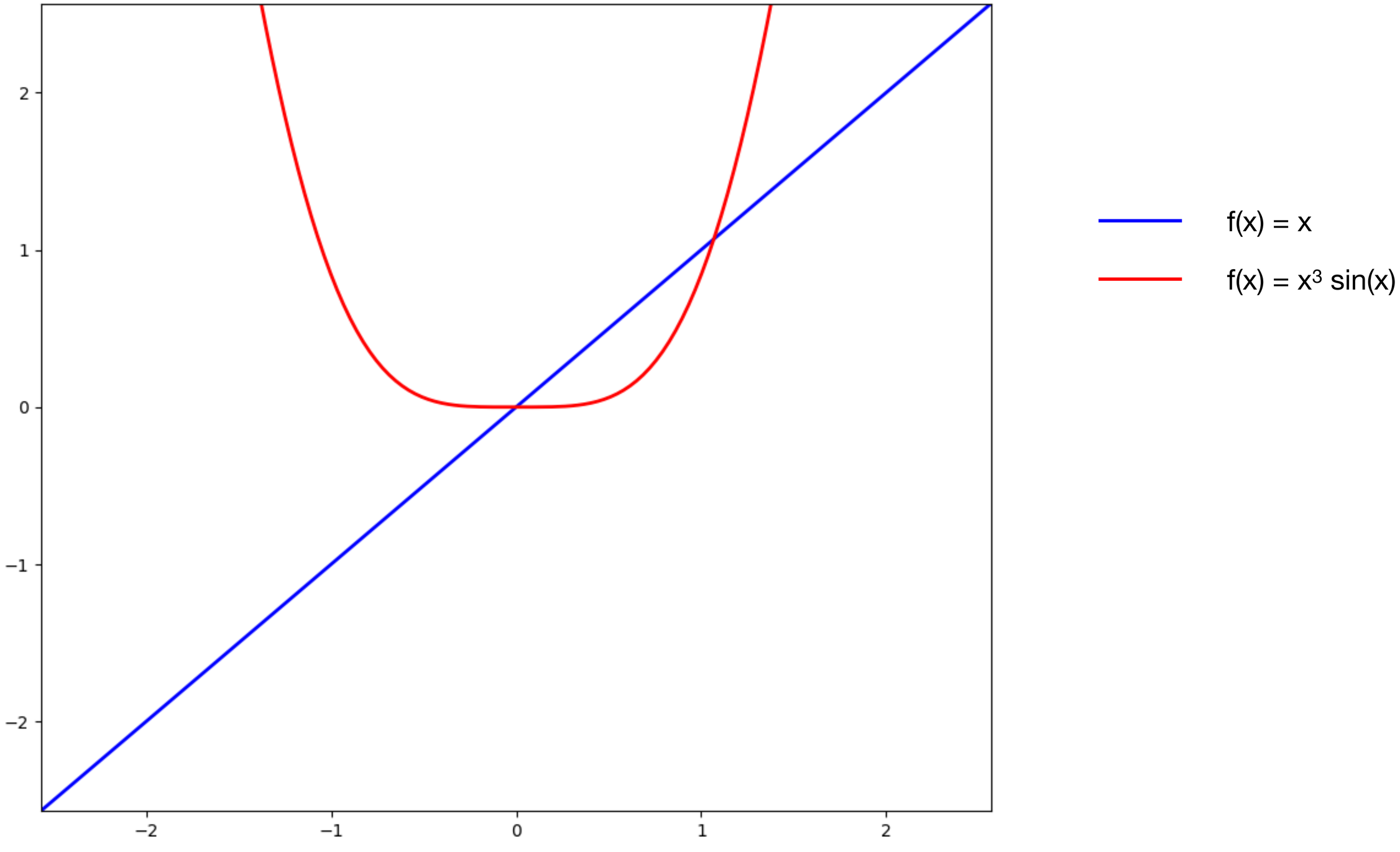
Many slides from Roman Novak

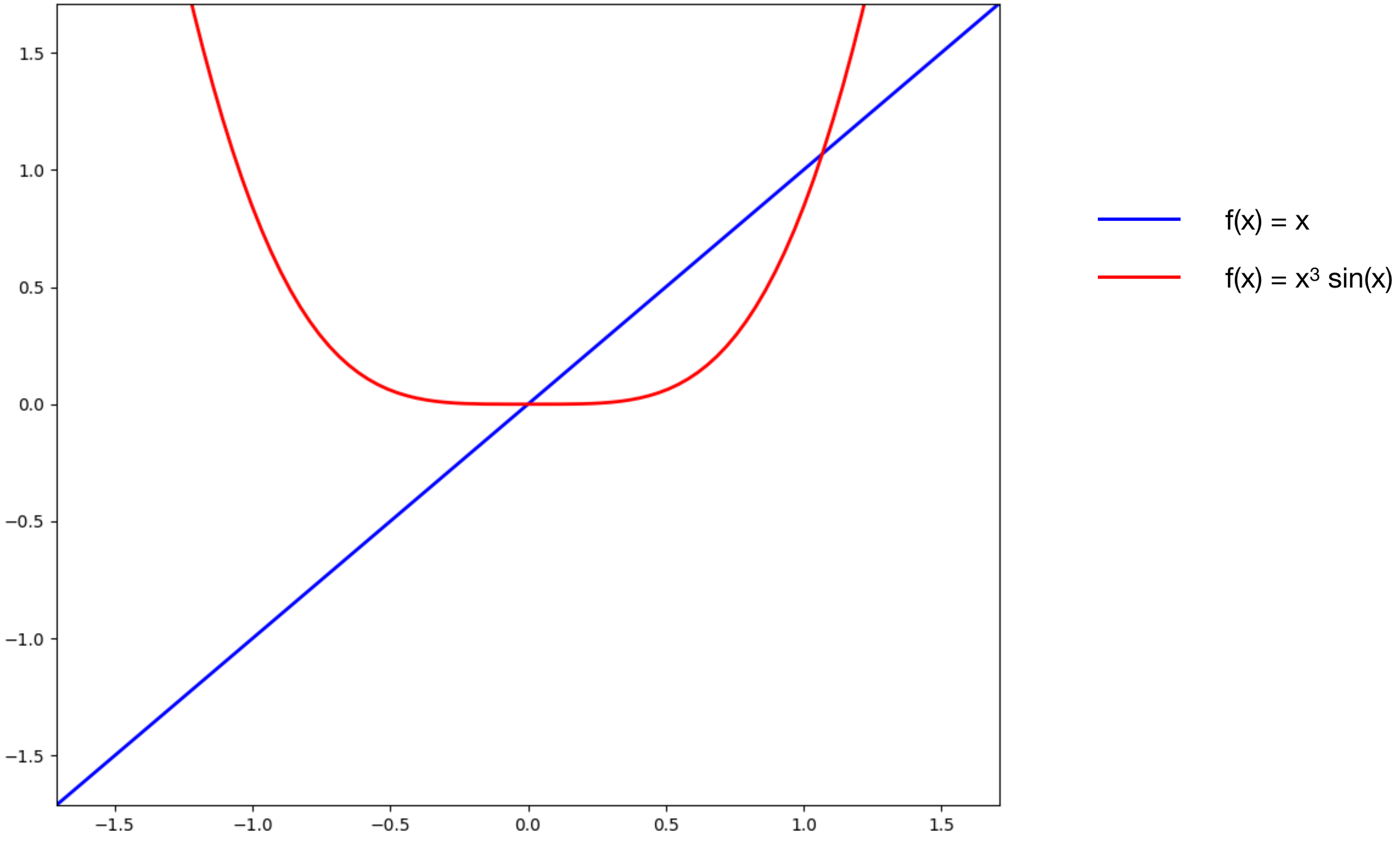
Which generalizes better:

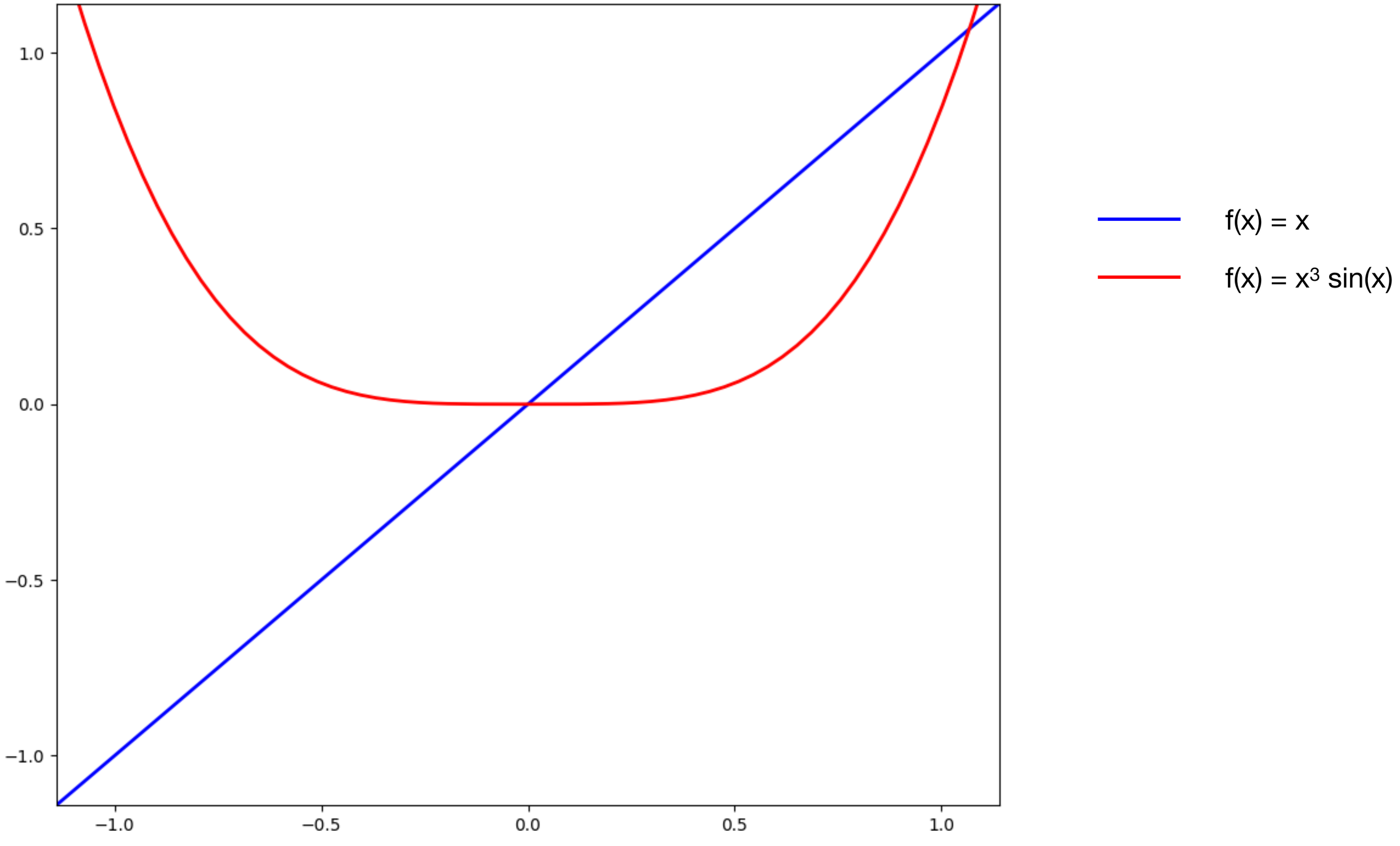
Bigger networks or smaller?

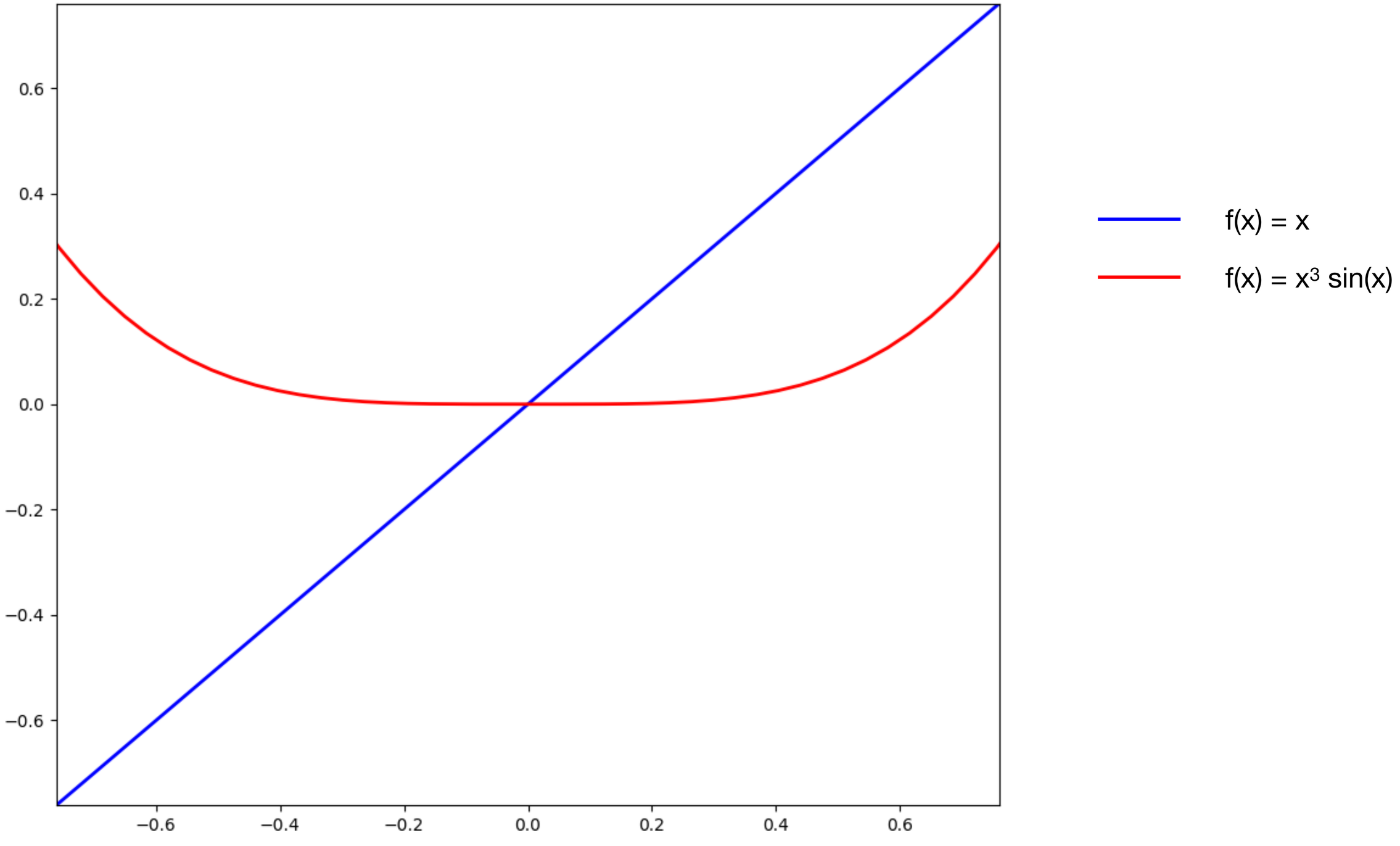
More complex models or simpler models?

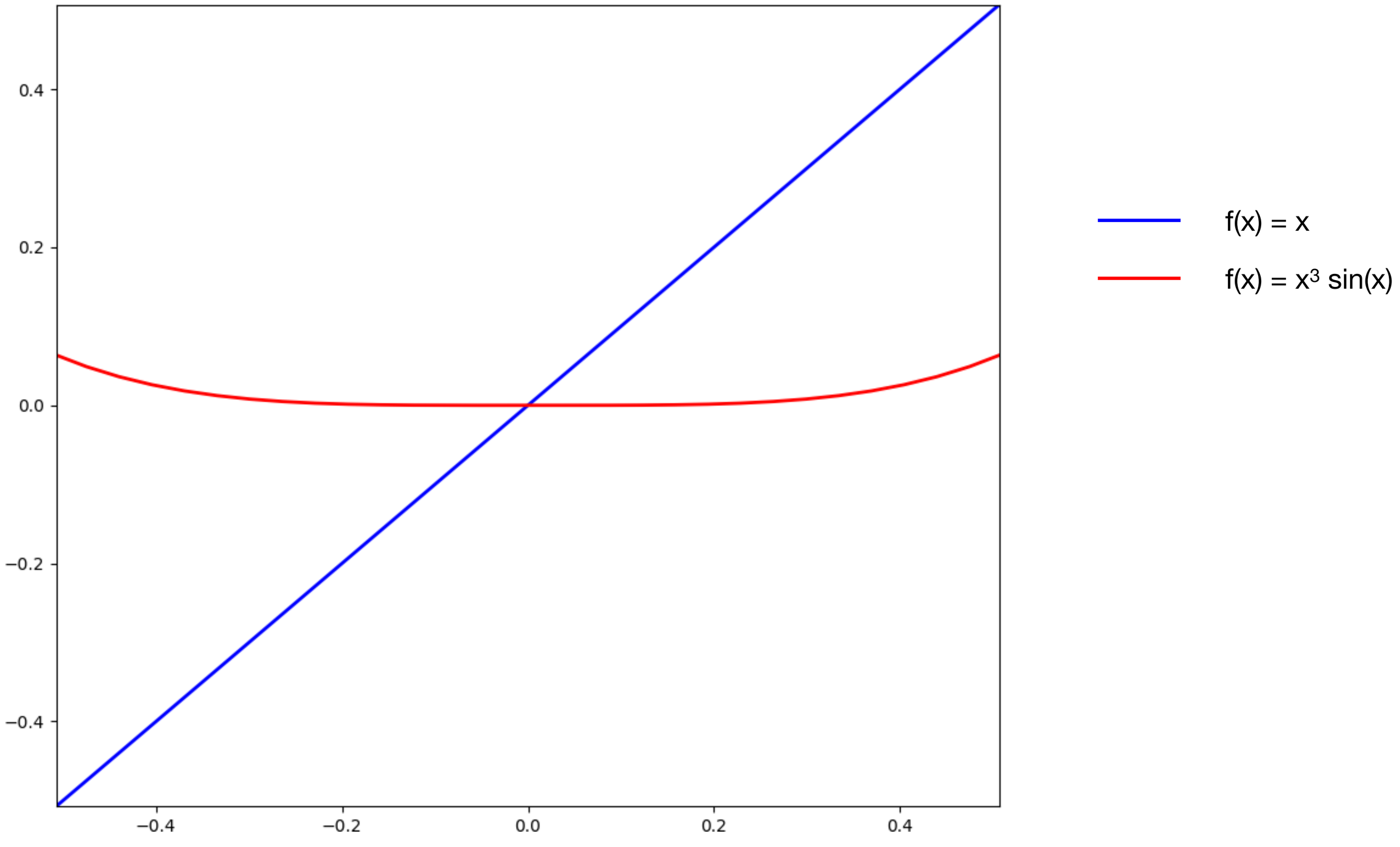

3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
f(x) = x
f(x) = x³ sin(x)

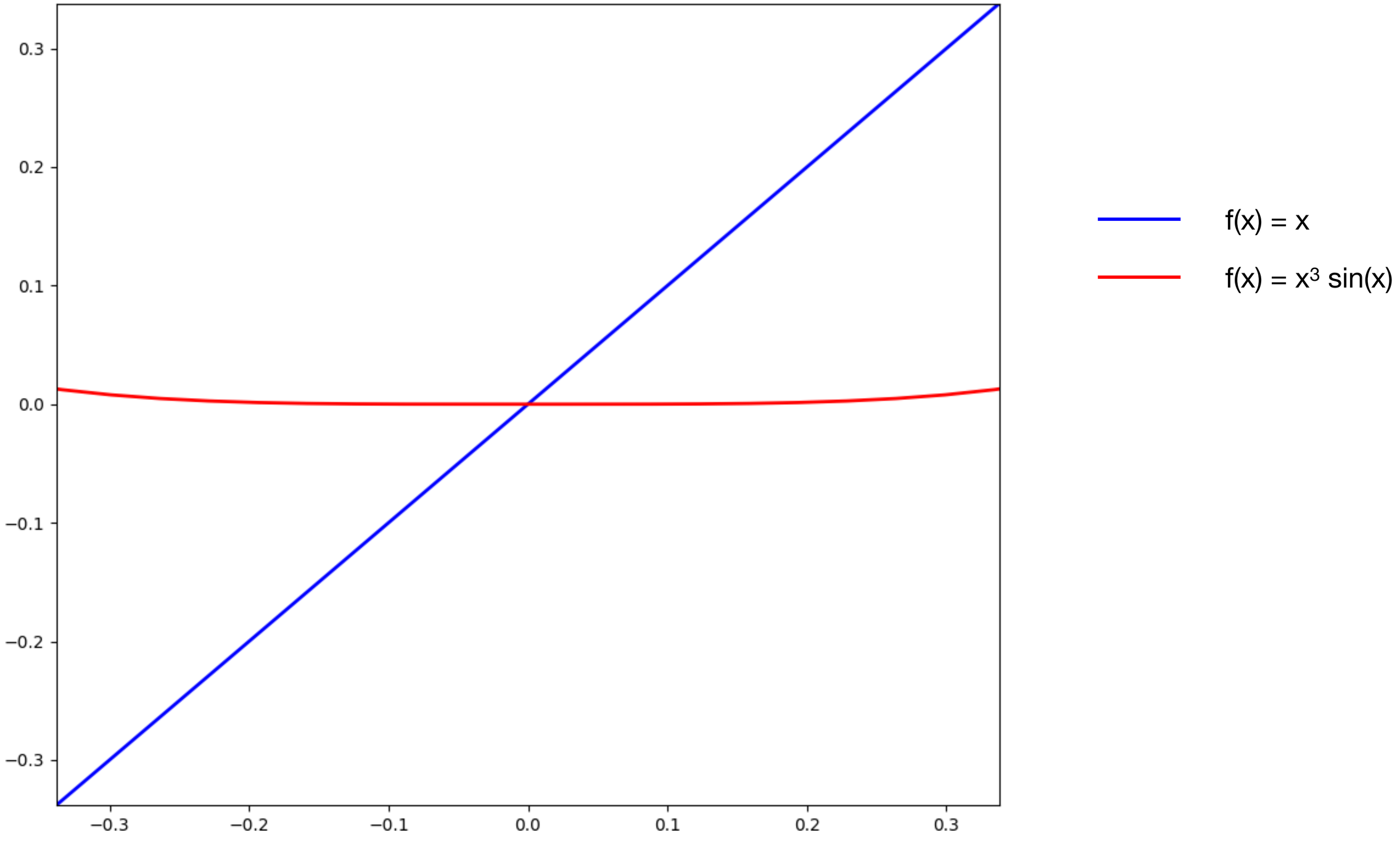

2
1
0
−1
−2
−2
−1
0
1
2
f(x) = x
f(x) = x³ sin(x)

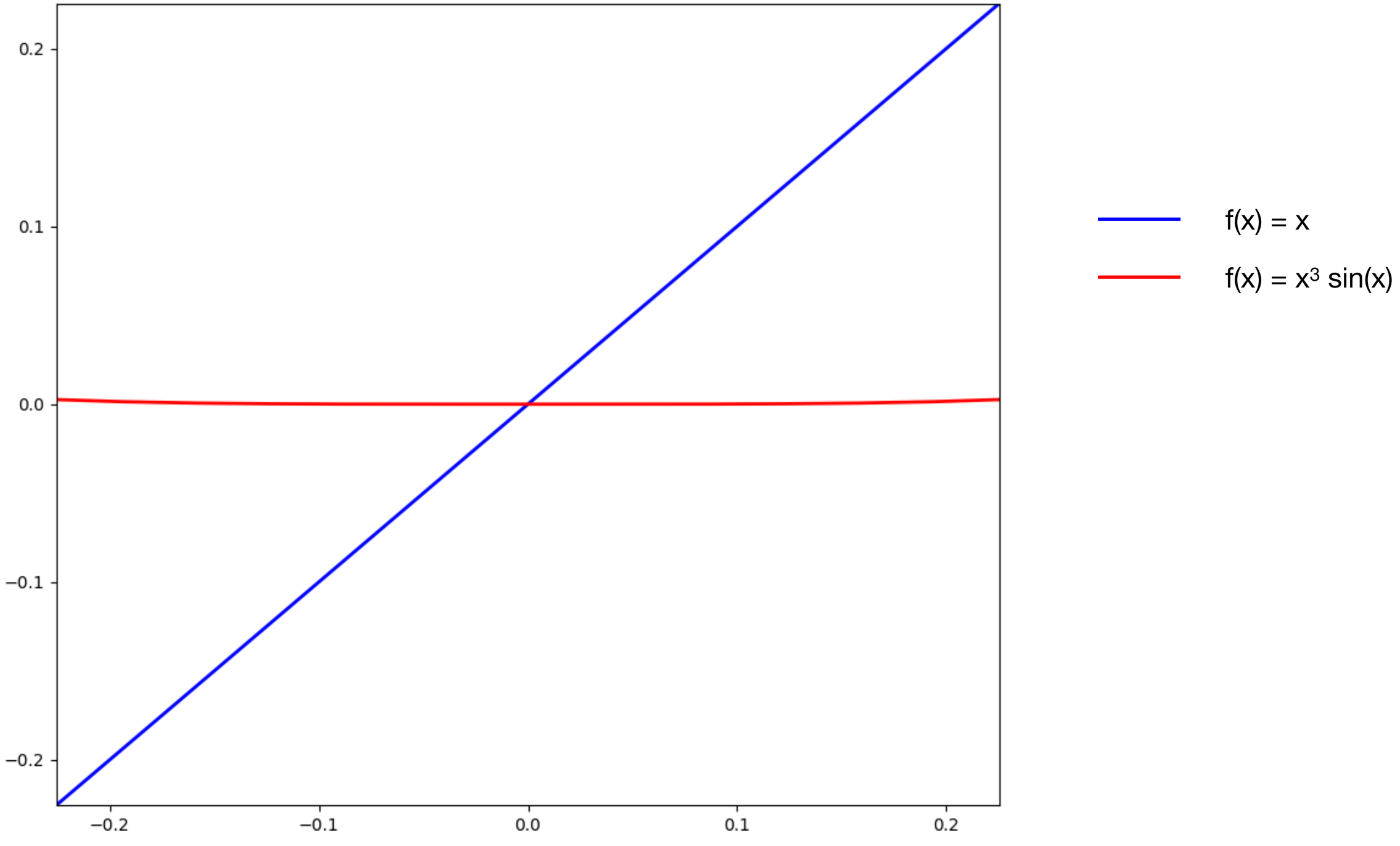

1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
f(x) = x
f(x) = x3 sin(x)


1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
f(x) = x
f(x) = x³ sin(x)


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
f(x) = x
f(x) = x3 sin(x)


0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4
f(x) = x
f(x) = x³ sin(x)


0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
f(x) = x
f(x) = x3 sin(x)


0.2
0.1
0.0
−0.1
−0.2
−0.2
−0.1
0.0
0.1
0.2
f(x) = x
f(x) = x³ sin(x)

TL;DR:

Sensitivity of a trained neural network to test inputs correlates with test error.

TL;DR:

Sensitivity of a trained neural network to test inputs correlates with test error.

(Generalization)

TL;DR:

Sensitivity of a trained neural network to test inputs correlates with test error.

(Generalization)

Plan:

1. Motivation / interpretation
2. Sensitivity metrics
3. Experimental results

Why Search for Correlates of Generalization?

- Understanding neural networks (NNs);
- Regularizers;

Why Search for Correlates of Generalization?

- Model comparison:

$$\mathbb{P}\left[\mathcal{H}|\mathcal{D}\right] \propto \mathbb{P}\left[\mathcal{D}|\mathcal{H}\right]\mathbb{P}\left[\mathcal{H}\right]$$

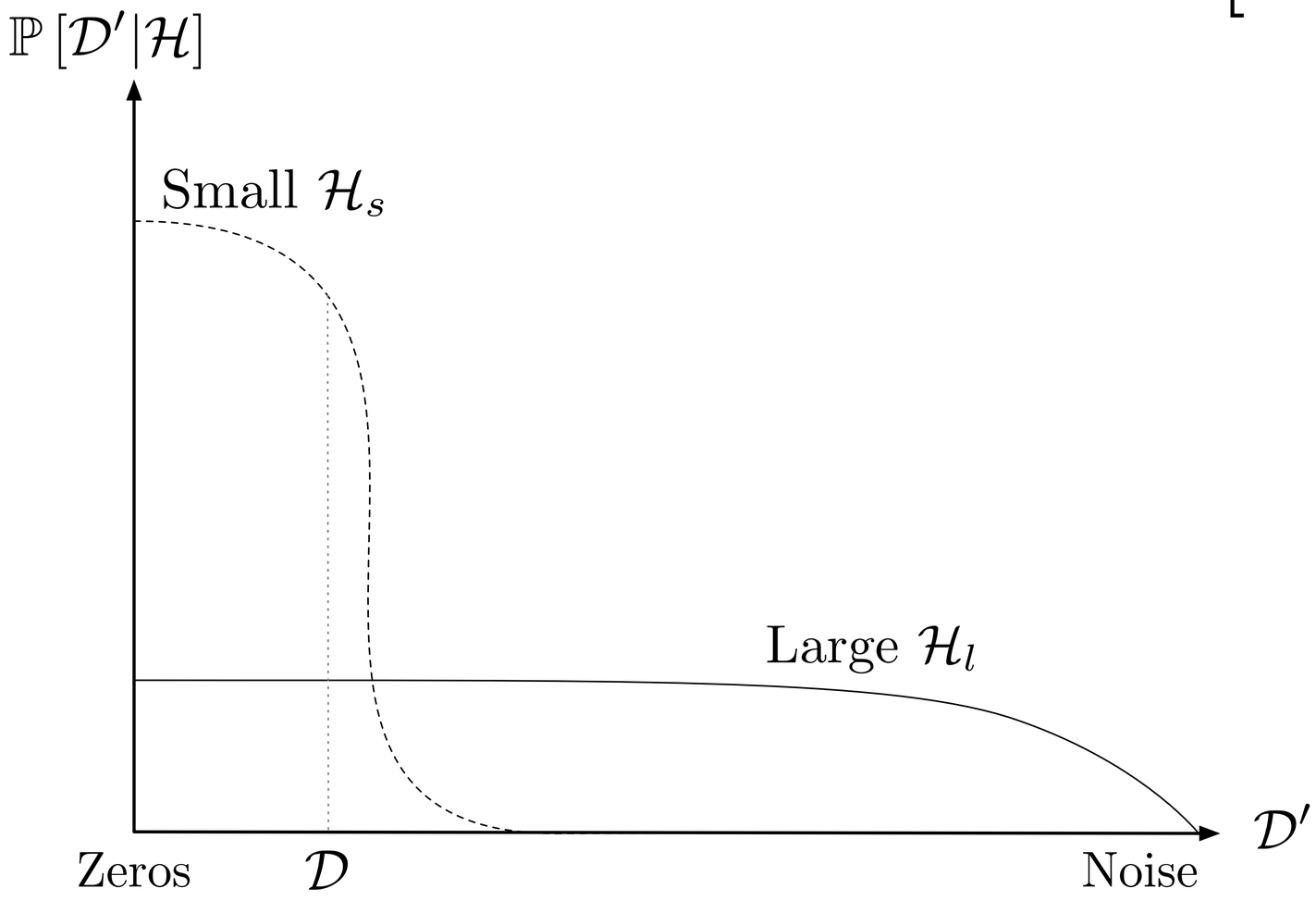
Evidence Prior

Why Search for Correlates of Generalization?

- Model comparison:

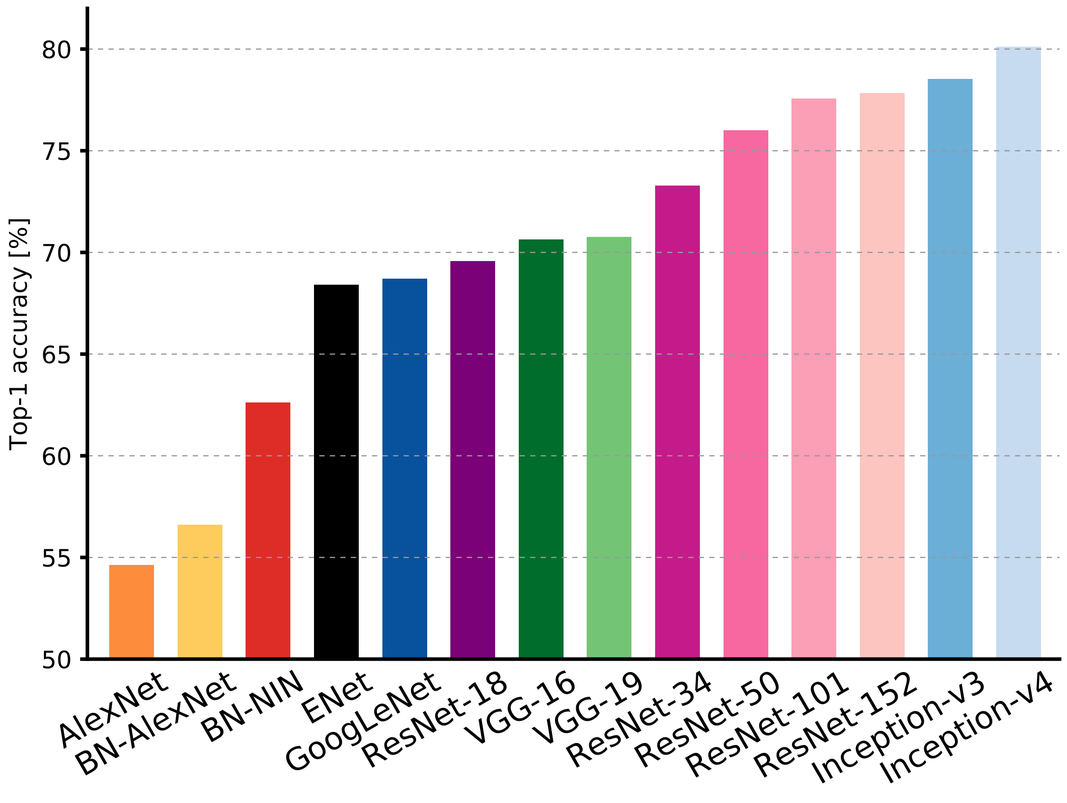
$$\mathbb{P}\left[\mathcal{H}|\mathcal{D}\right] \propto \mathbb{P}\left[\mathcal{D}|\mathcal{H}\right]\mathbb{P}\left[\mathcal{H}\right]$$

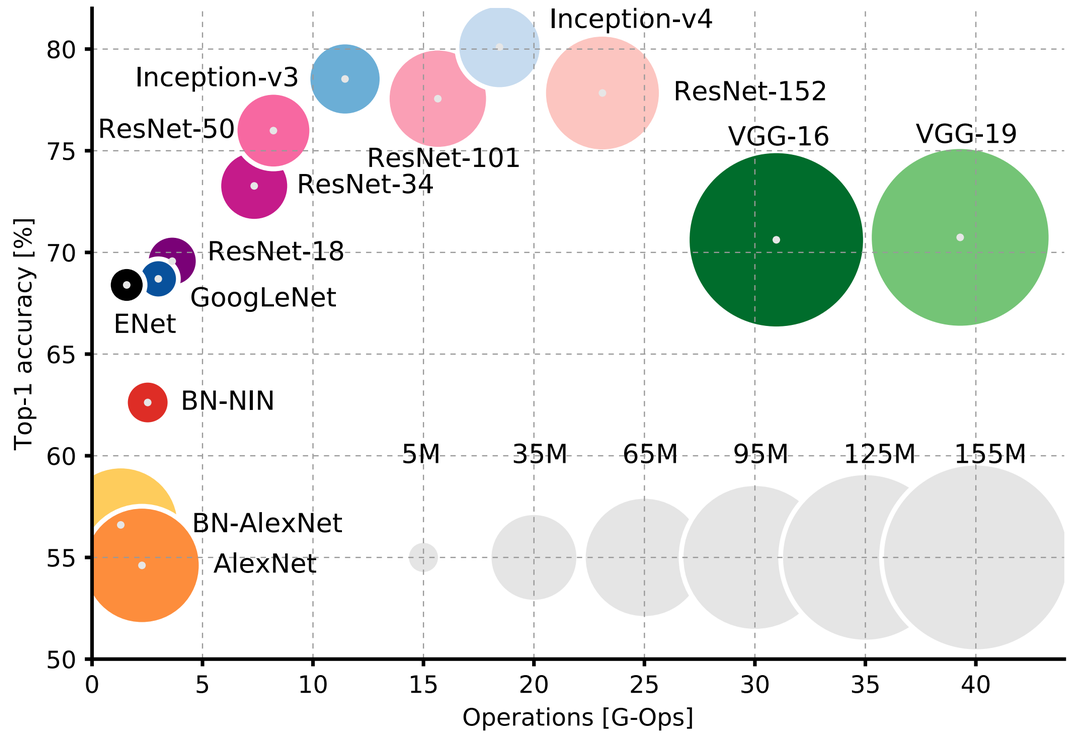
Evidence Prior



Bigger Networks Generalize *Better*

Historical trends of ImageNet competitions:





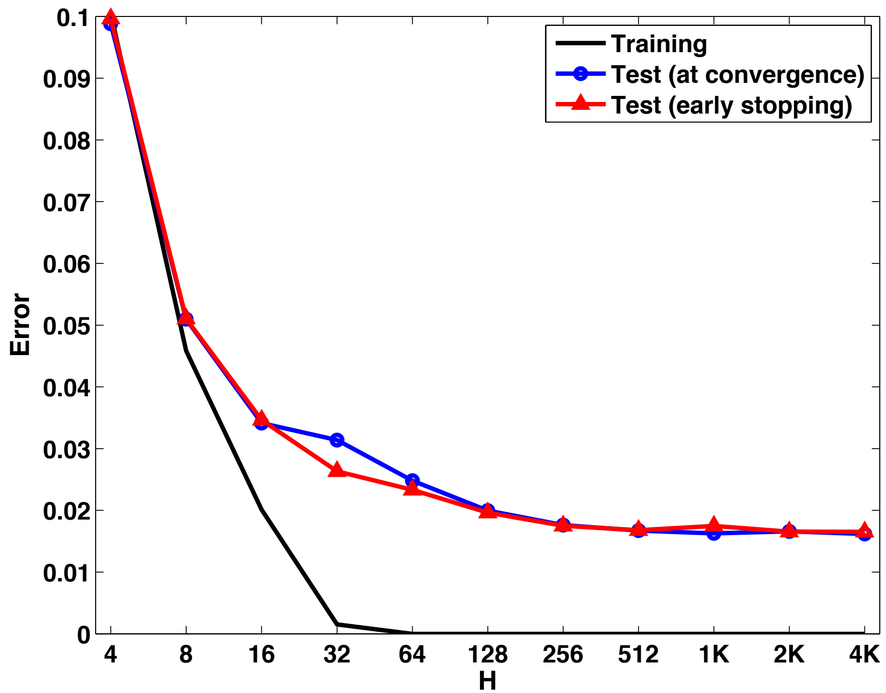
(radius ~ #parameters)

(An Analysis of Deep Neural Network Models for Practical Applications
by Alfredo Canziani, Adam Paszke, Eugenio Culurciello)

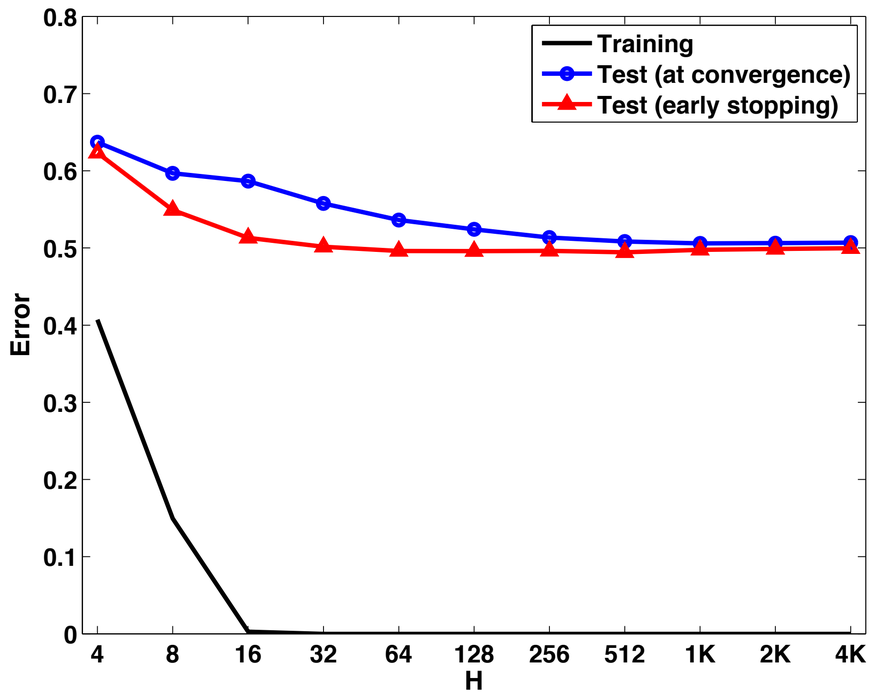
Bigger Networks Generalize *Better*

Single hidden layer networks:

MNIST



CIFAR10



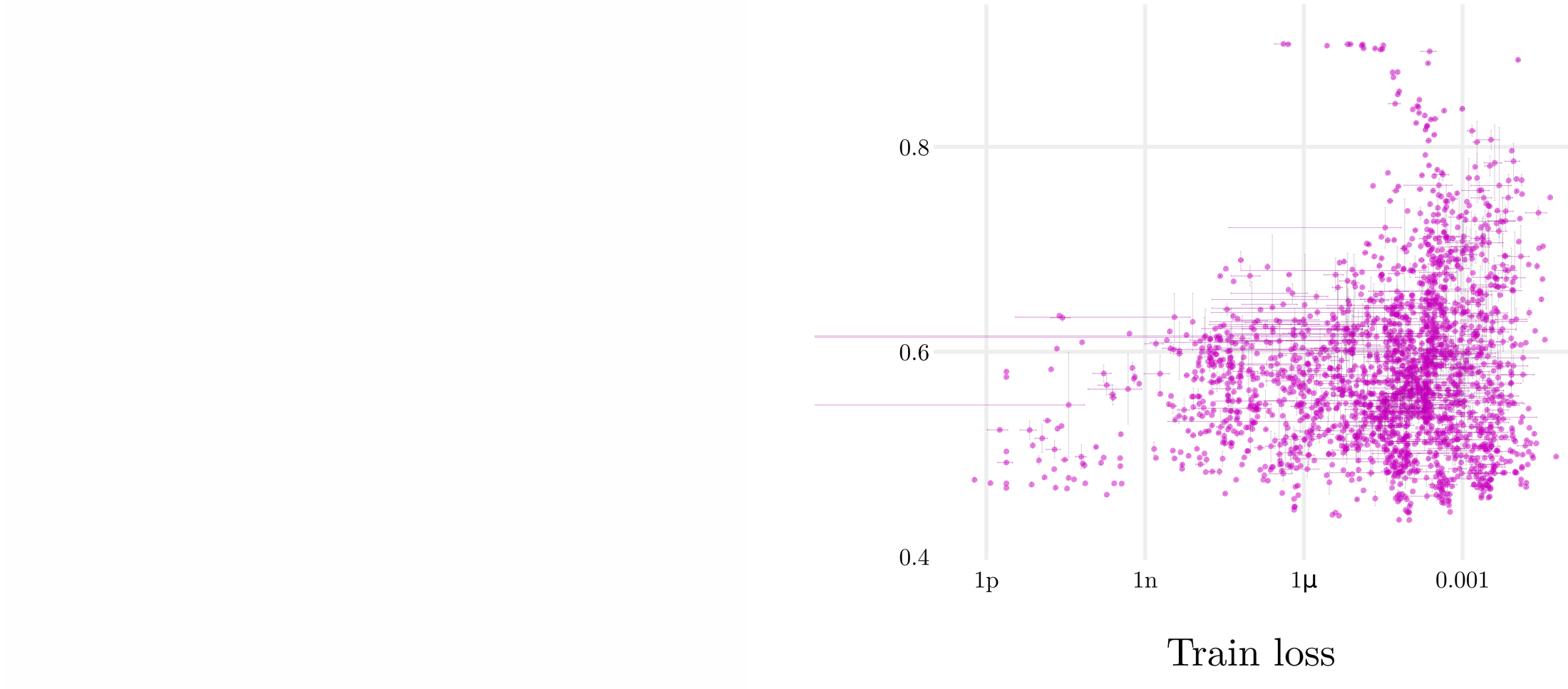
(H - number of hidden units)

([In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning](#)
by Behnam Neyshabur, Ryota Tomioka, Nathan Srebro)

Bigger Networks Generalize *Better*

Many architectures and optimization choices:

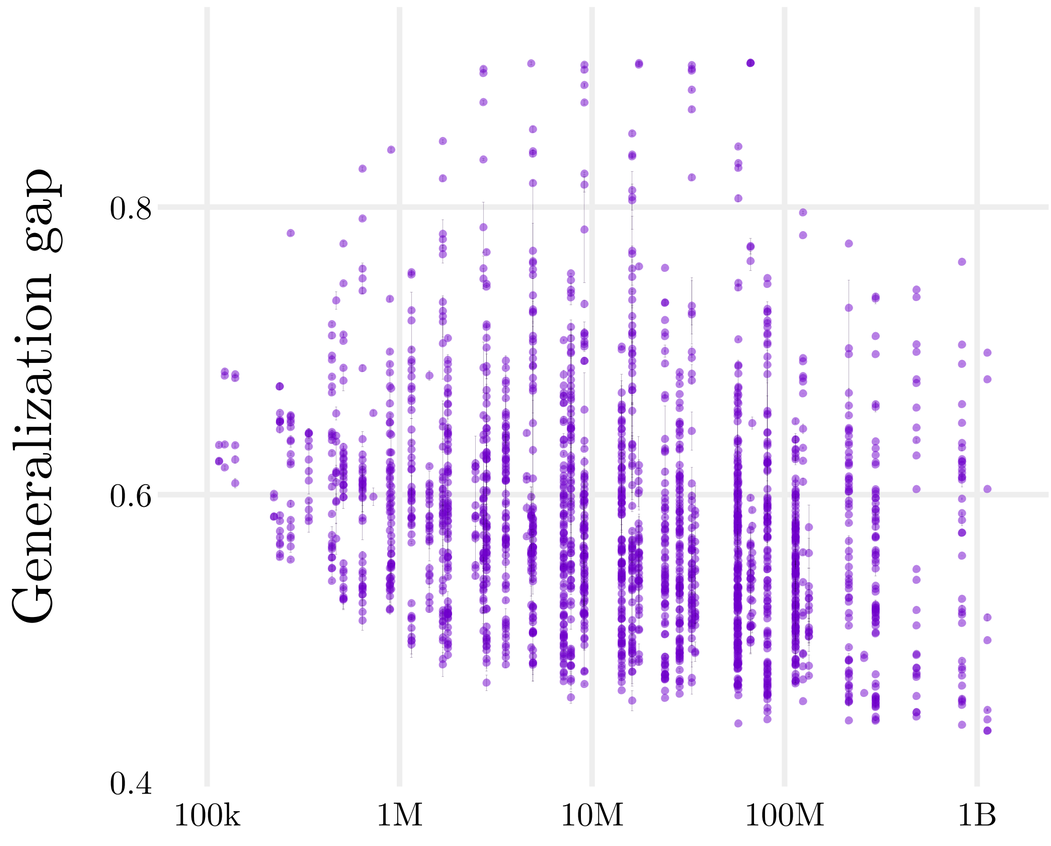
(each point is a different network, 100% accurate on the whole training CIFAR10)

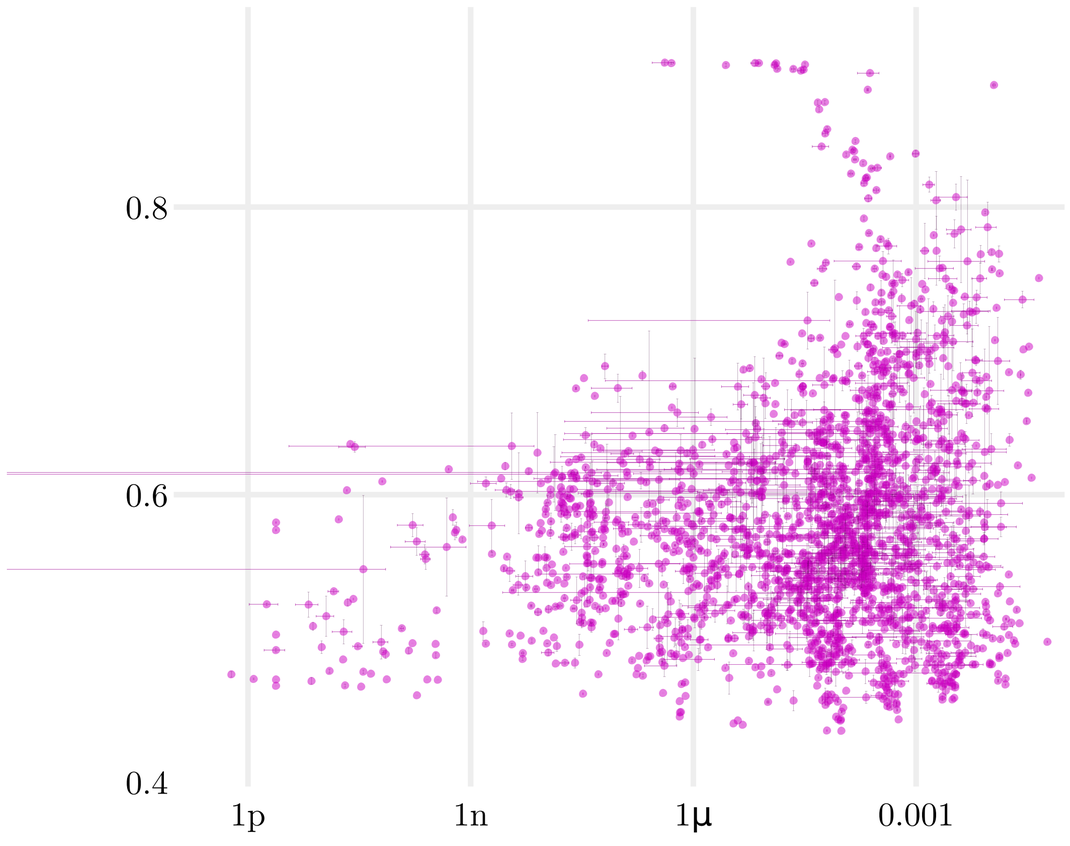


Bigger Networks Generalize *Better*

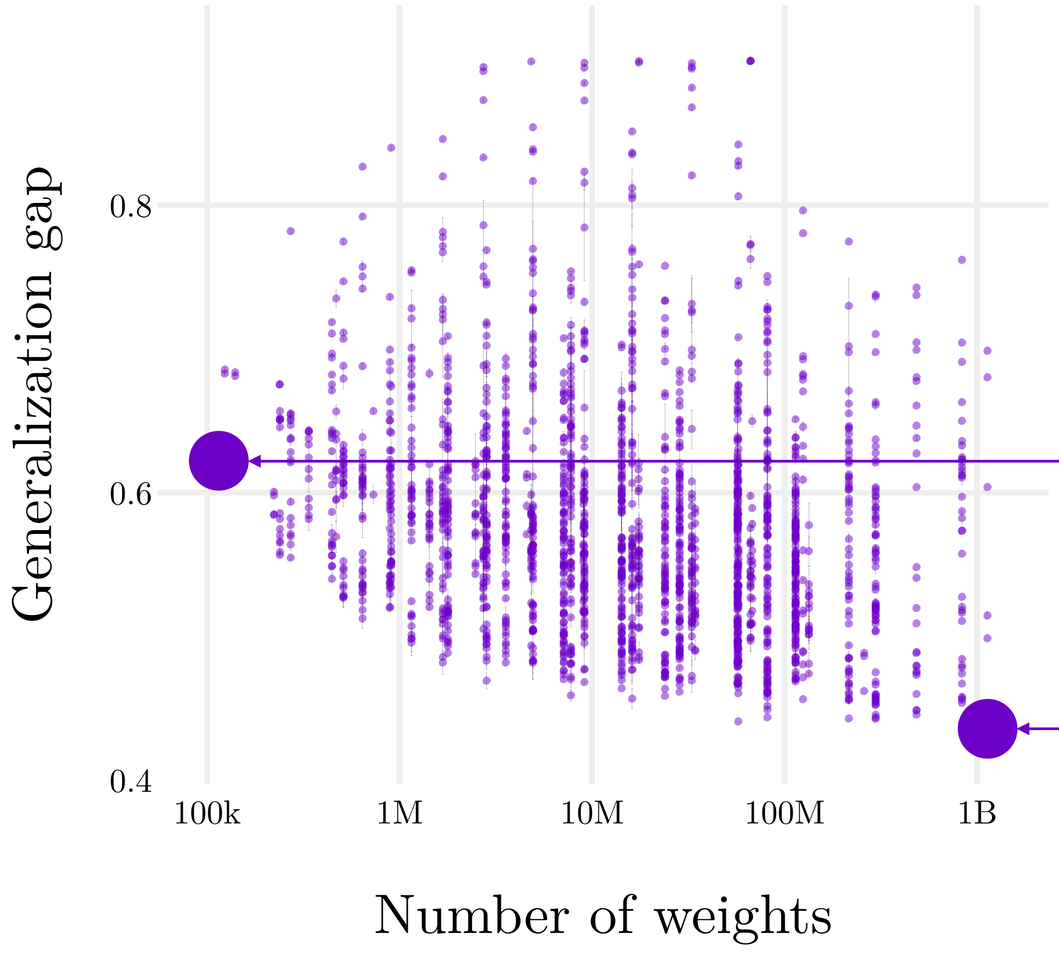
Many architectures and optimization choices:

(each point is a different network, 100% accurate on the whole training CIFAR10)



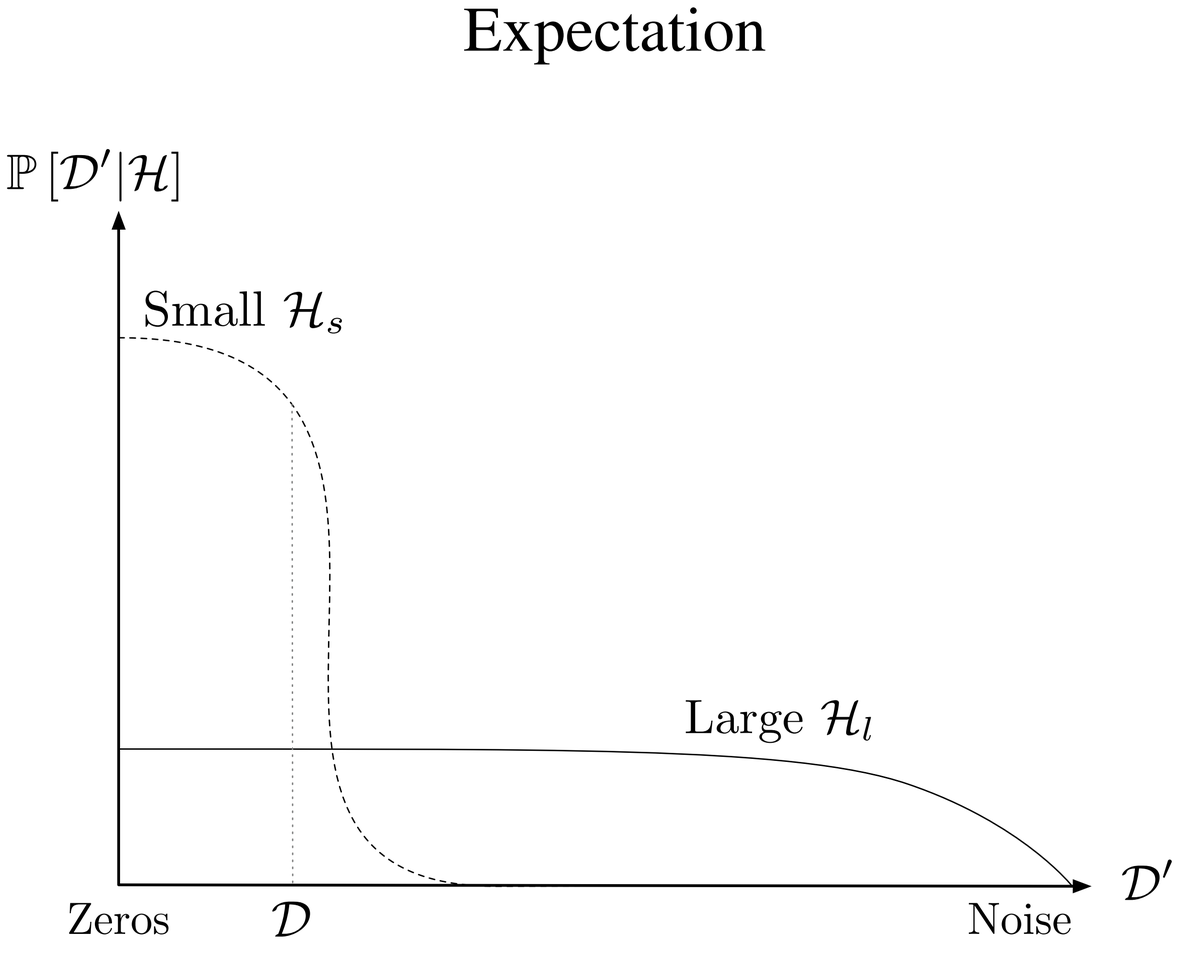


Bigger Networks Generalize *Better*



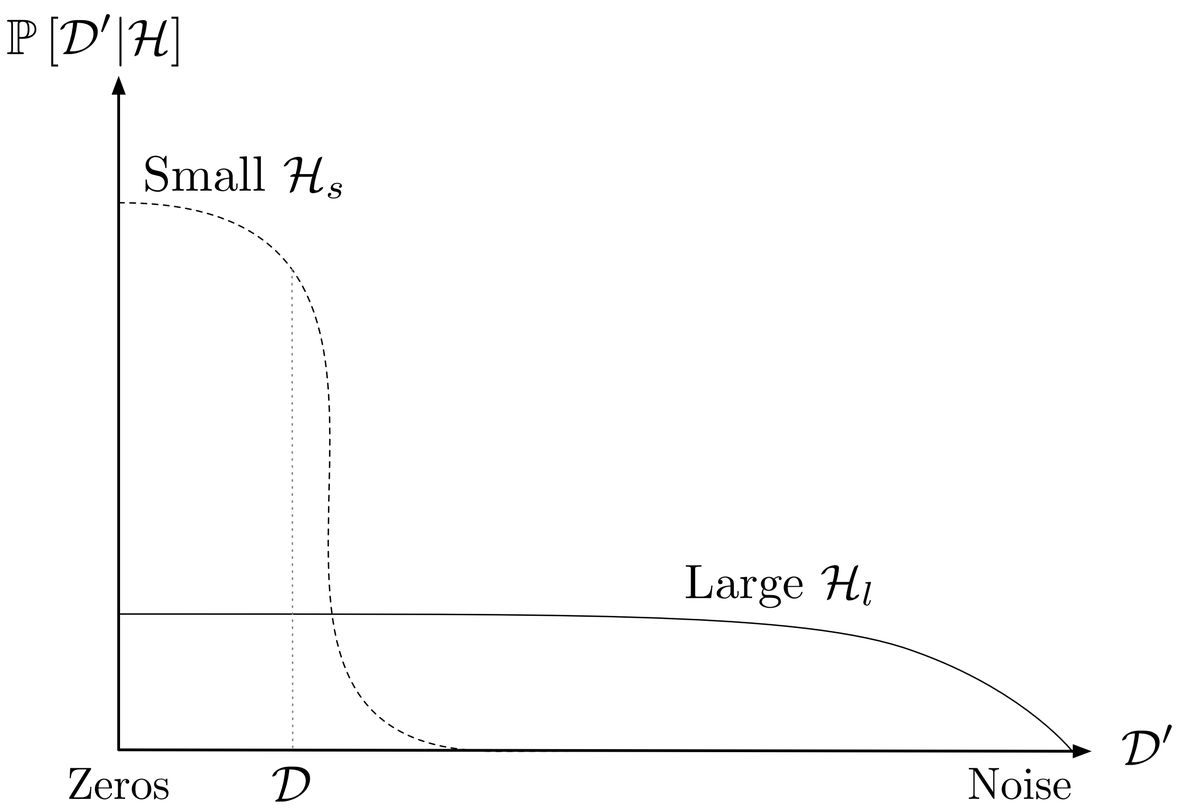
#parameters	Test error
$\approx 10^5$	≈ 0.6
$\ll$	$\gg$
$\approx 10^9$	≈ 0.4

Bigger Networks Generalize *Better*

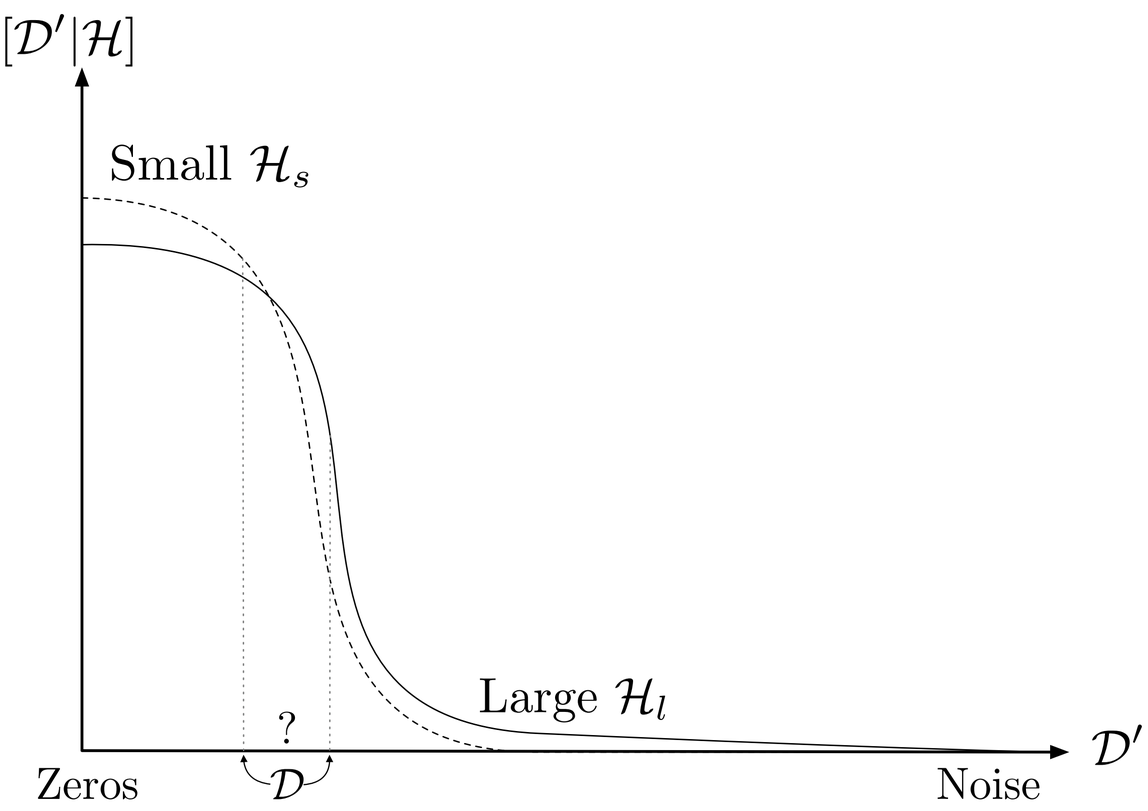


Bigger Networks Generalize *Better*

Expectation



Reality?



Why Sensitivity?

- Can reasonably assume a prior $\mathbb{P}\left[\mathcal{H}\right]$ in favor of robustness to small perturbations in the inputs.
- Many regularization / adversarial defense strategies employ sensitivity to inputs.
- Can be computed independently of parametrization.

Experimental Setup

$$h(x) = W_k \sigma(W_{k-1} \sigma(\cdots W_1 x)$$

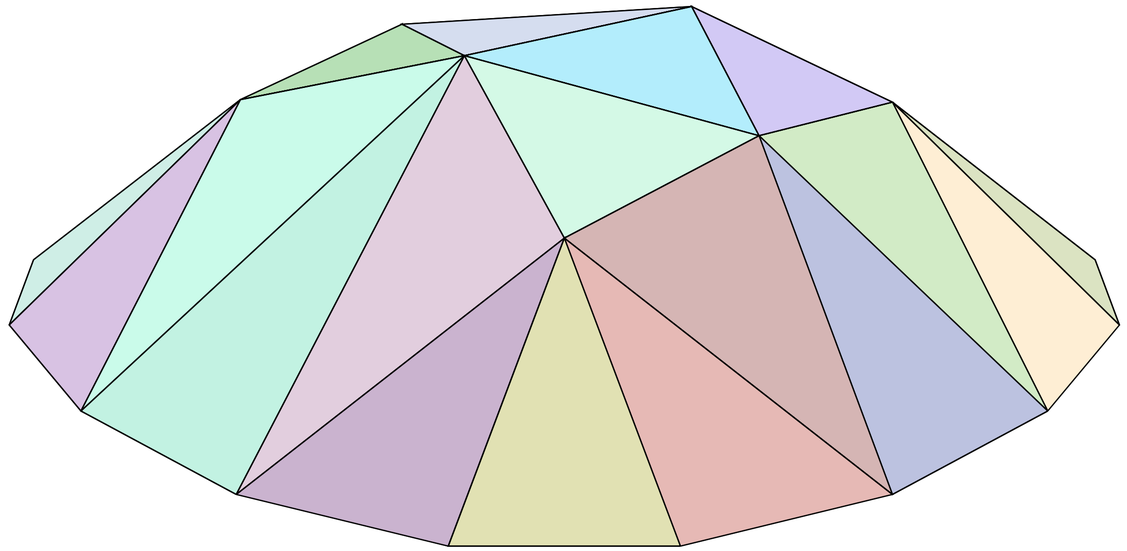
- Feedforward network;
- With a piecewise-linear activation σ;

(e.g. ReLU, Hard-Tanh, Hard-Sigmoid, …)

$\Downarrow$

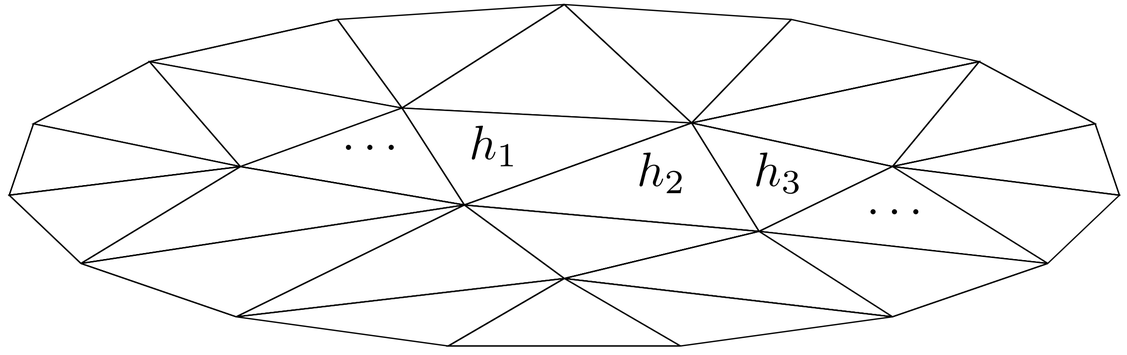
h is piecewise-linear on $\mathbb{R}^d$.

Example of a Pw-linear Function



$\mathbb{R}^n$

$h \uparrow \{h_l\}$ are linear.



$\mathbb{R}^d$

(Wikipedia)

Input Domain: Interiors

$$h(x) - h(y) = h'(x)(x - y)$$

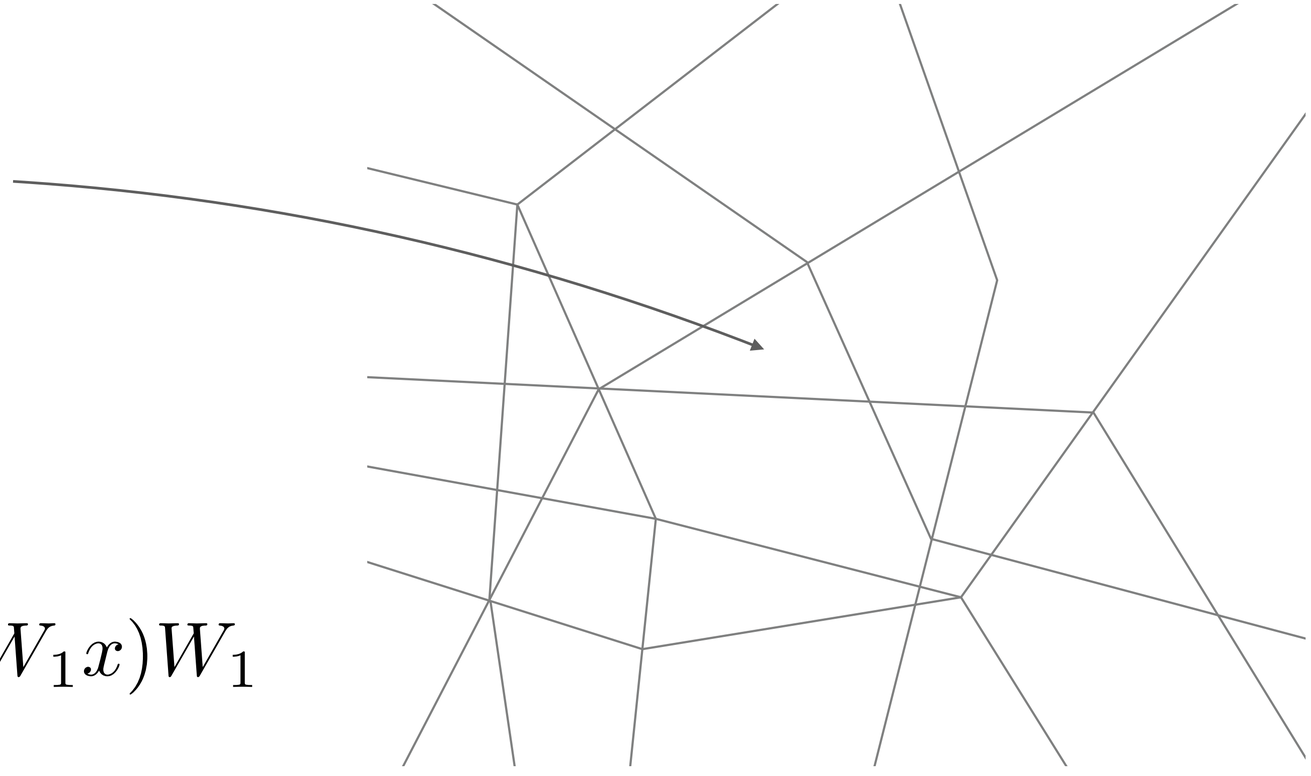
A linear function,

computable explicitly:

$$h'(x) = W_k \sigma'(\cdots) \cdots W_2 \sigma'(W_1 x) W_1$$

$$\Downarrow$$

Local measure of sensitivity $\sim \|h'(x)\|$

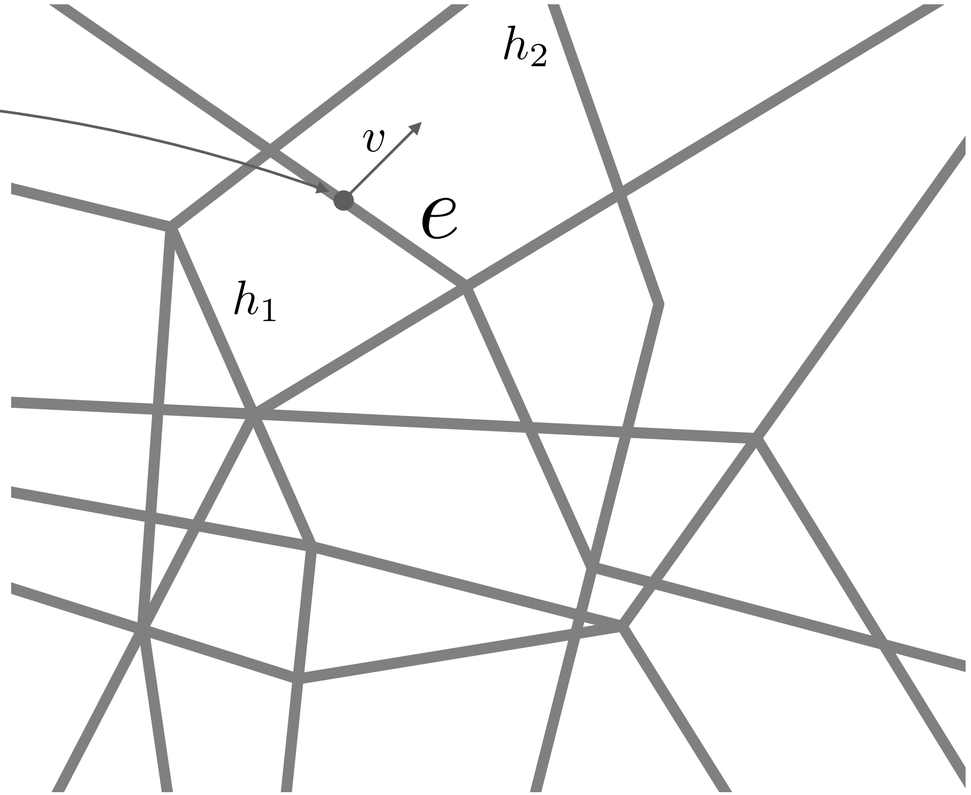


Input Domain: Faces

$$\frac{\partial h'}{\partial v}(e) = \lim_{t \to 0} \frac{h'(e+tv) - h'(e-tv)}{2t}$$

$$\sim (h'_2 - h'_1)\delta(e)$$

$$\|h'_1 - h'_2\| \sim \|h'_1\| \, \mathbf{1}_{h'_1 \neq h'_2}$$



Local measure of sensitivity: $\mathbf{1}_{h'_1 \neq h'_2}$

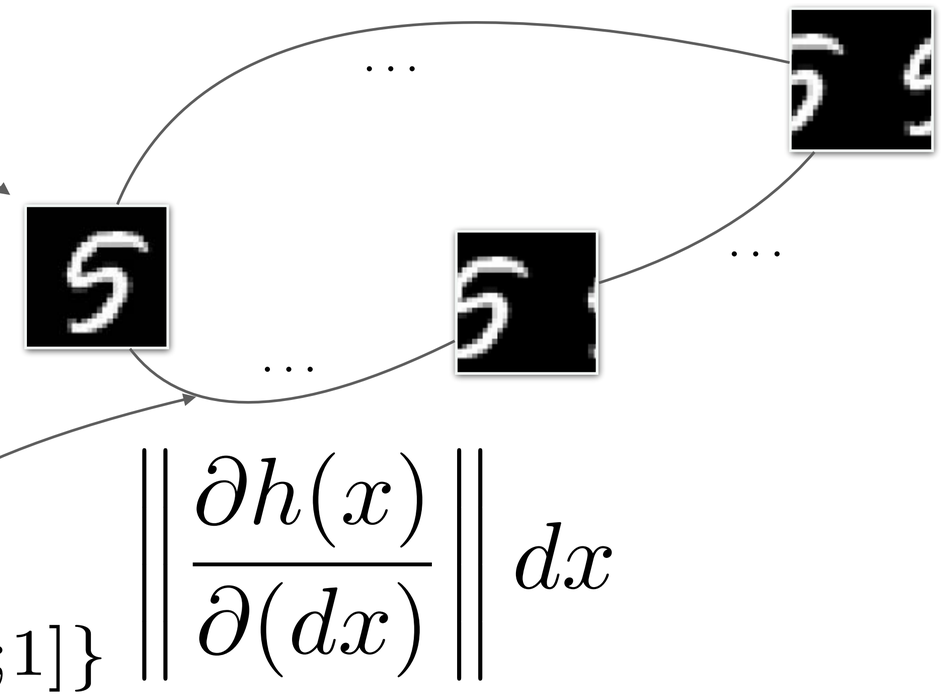
Global Measures of Sensitivity

- Interiors: $\mathbb{E}_{x \sim \mathbb{P}_{\mathcal{D}}} \left\| h'(x) \right\|$

  (Jacobian norm at a data point)

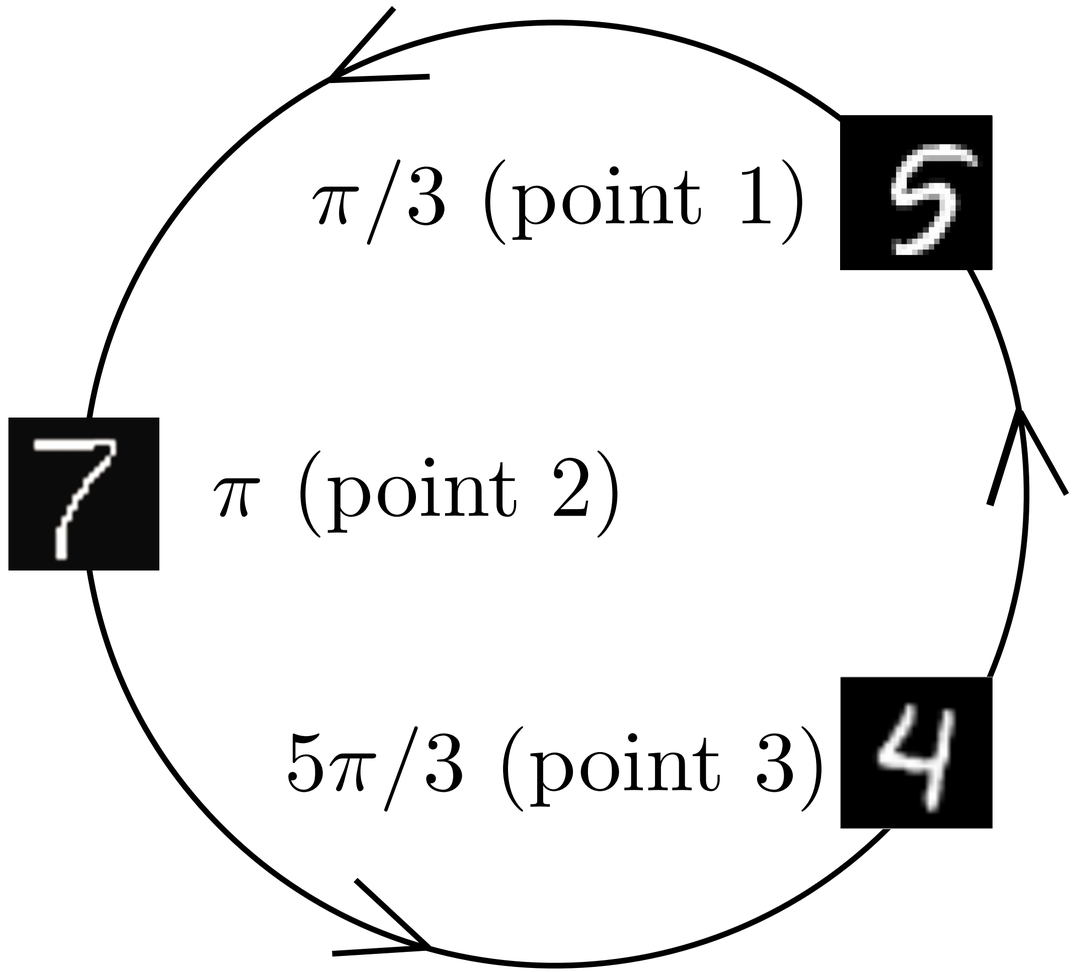
- Faces: $\mathbb{E}_{x_0 \sim \mathbb{P}_{\mathcal{D}}} \oint_{x \in \{T^t x_0 \,|\, t \in [0;1]\}} \left\| \frac{\partial h(x)}{\partial (dx)} \right\| dx$

  (estimated via transitions along a trajectory of pixel translations of a data point)



- Correspond to first and second terms in Taylor expansion.

(1) Sensitivity on/off Training Data Manifold



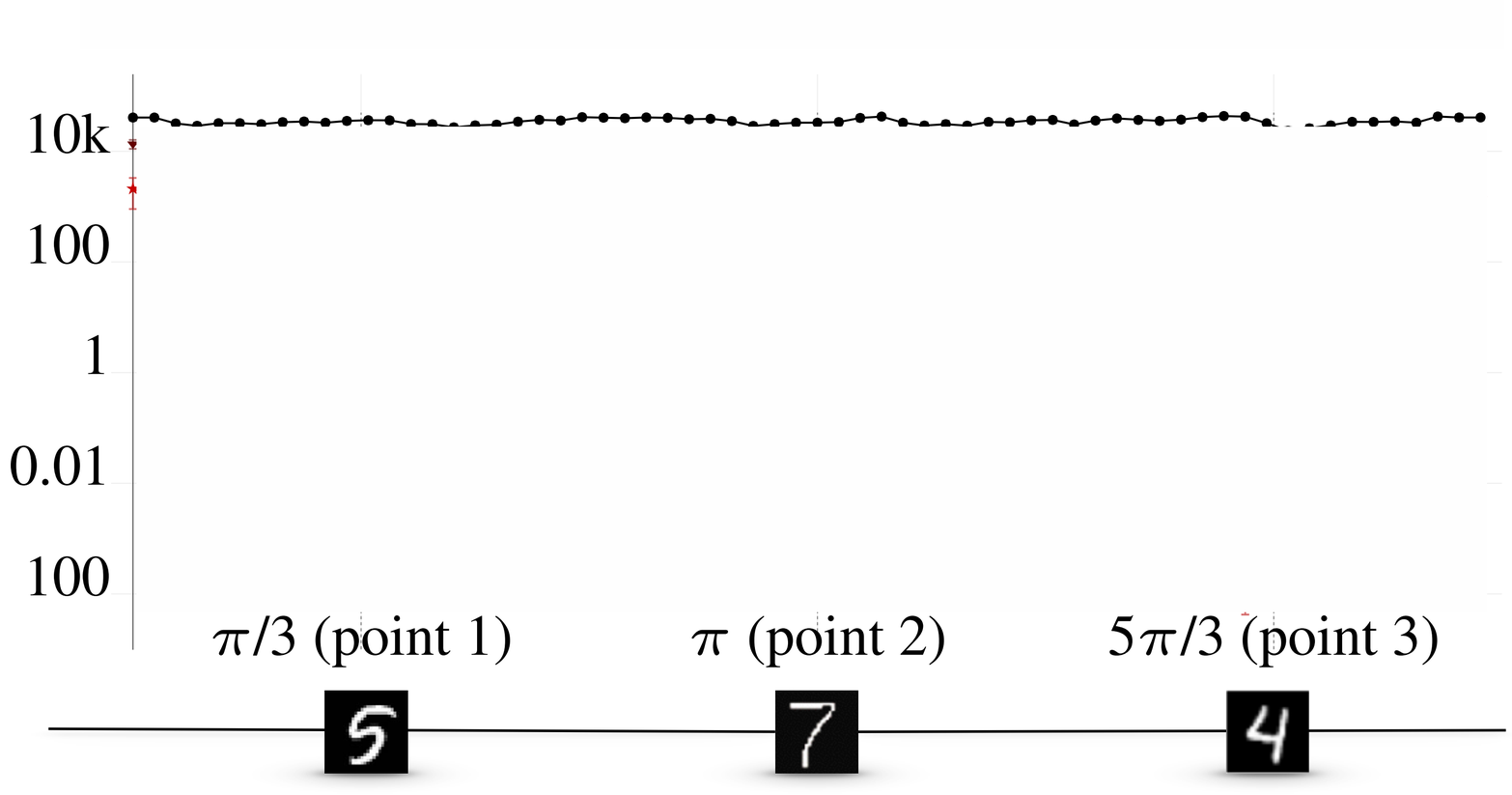
- Measure sensitivity along a circular trajectory through 3 training points.
- Traversing such a trajectory corresponds to approaching and departing from the training data manifold.

Sensitivity on/off Training Data Manifold

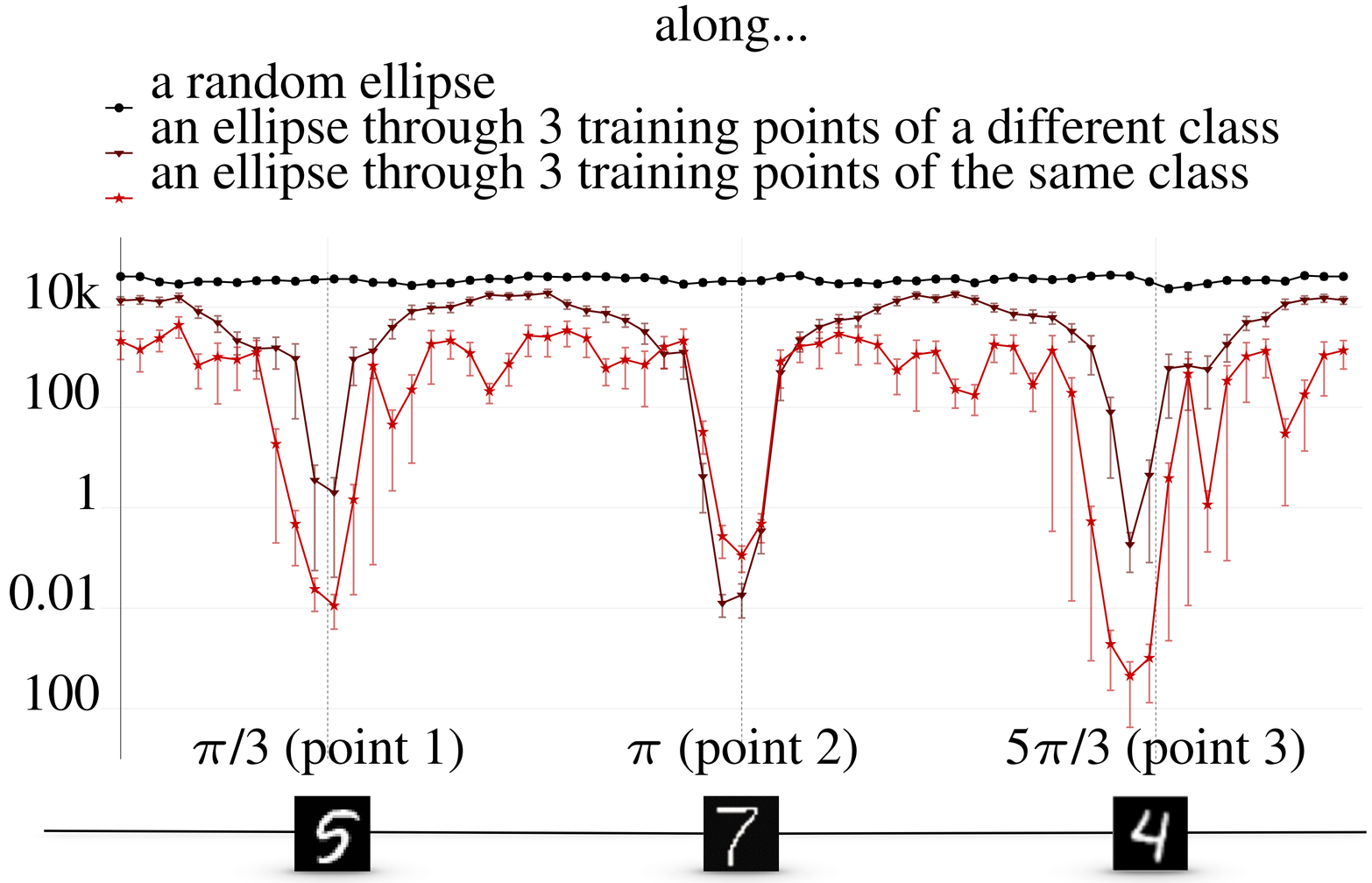
Mean Jacobian norm

along...

a random ellipse



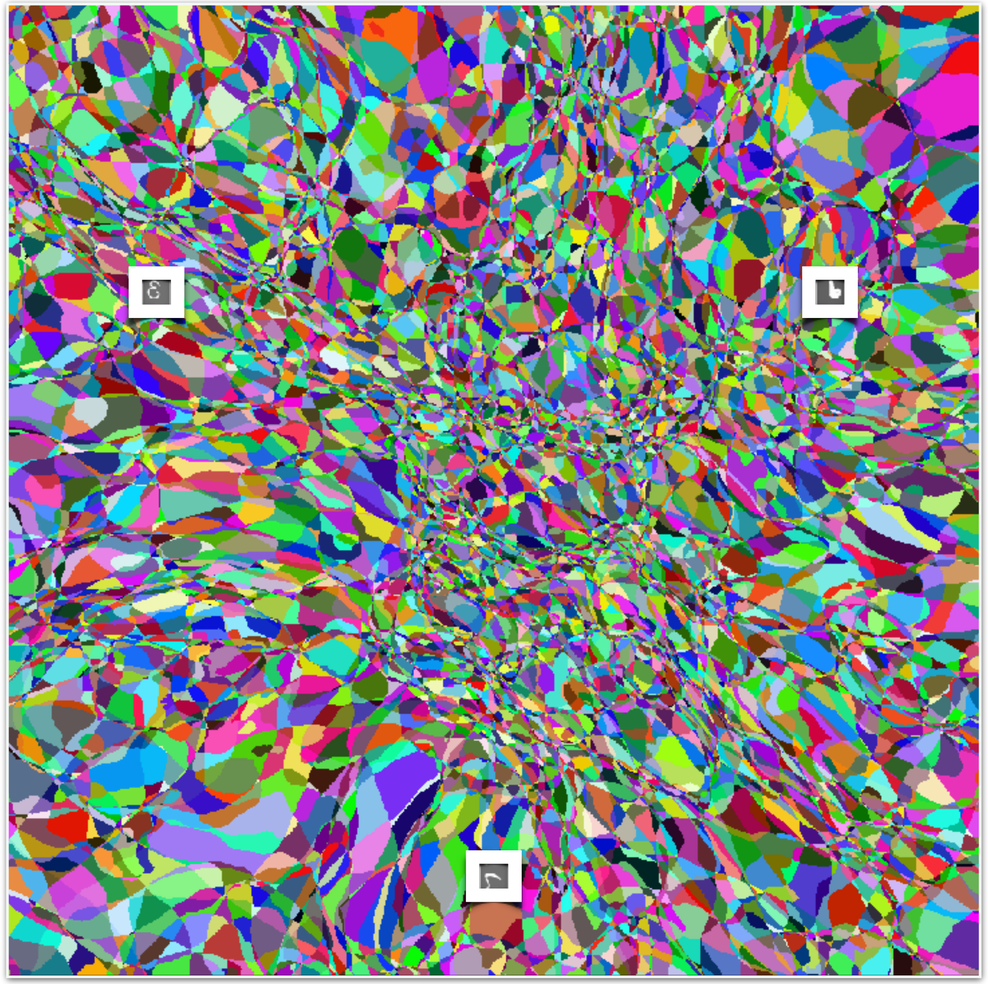
Sensitivity on/off Training Data Manifold



- Jacobian norm dips dramatically around training data.
- Interpolating different digits results in higher norm.

Sensitivity on/off Training Data Manifold

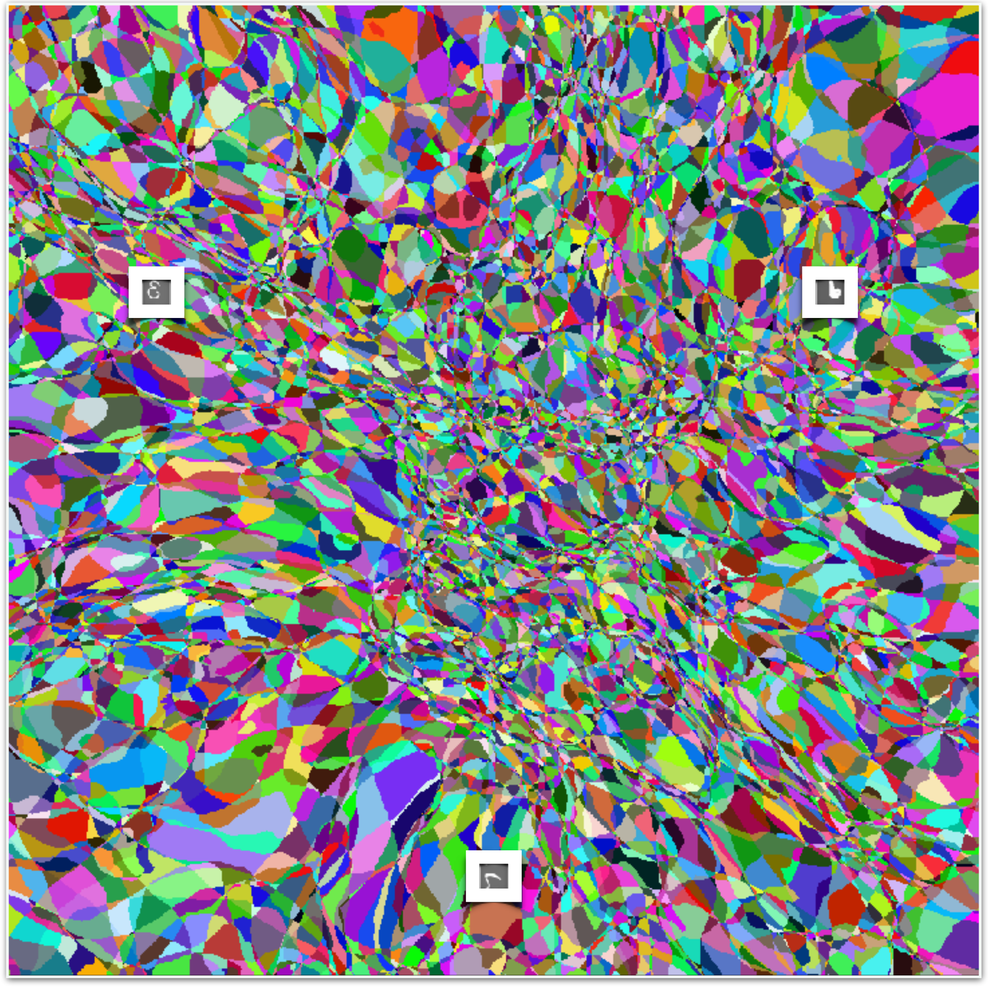
Before training



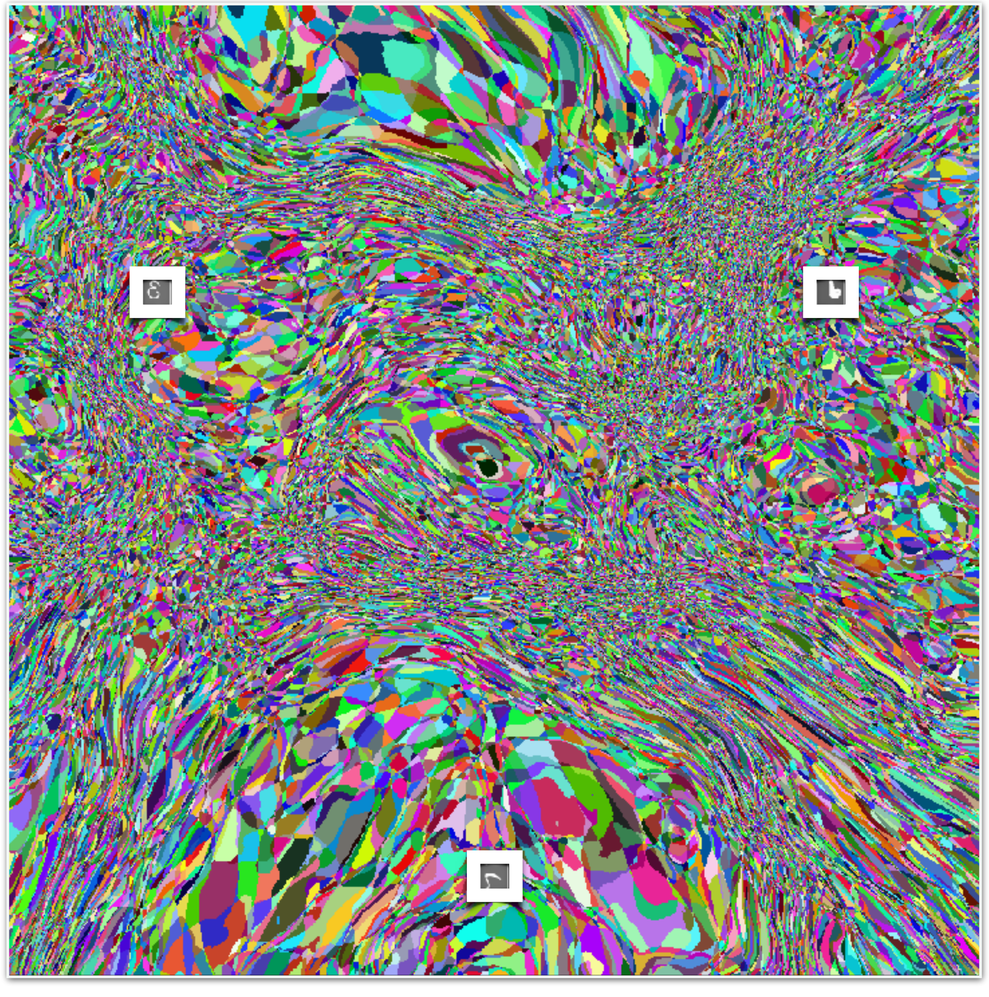
(Linear region boundaries of the last (pre-logit) over a 2D slice in the input space through three training points)

Sensitivity on/off Training Data Manifold

Before training



After training



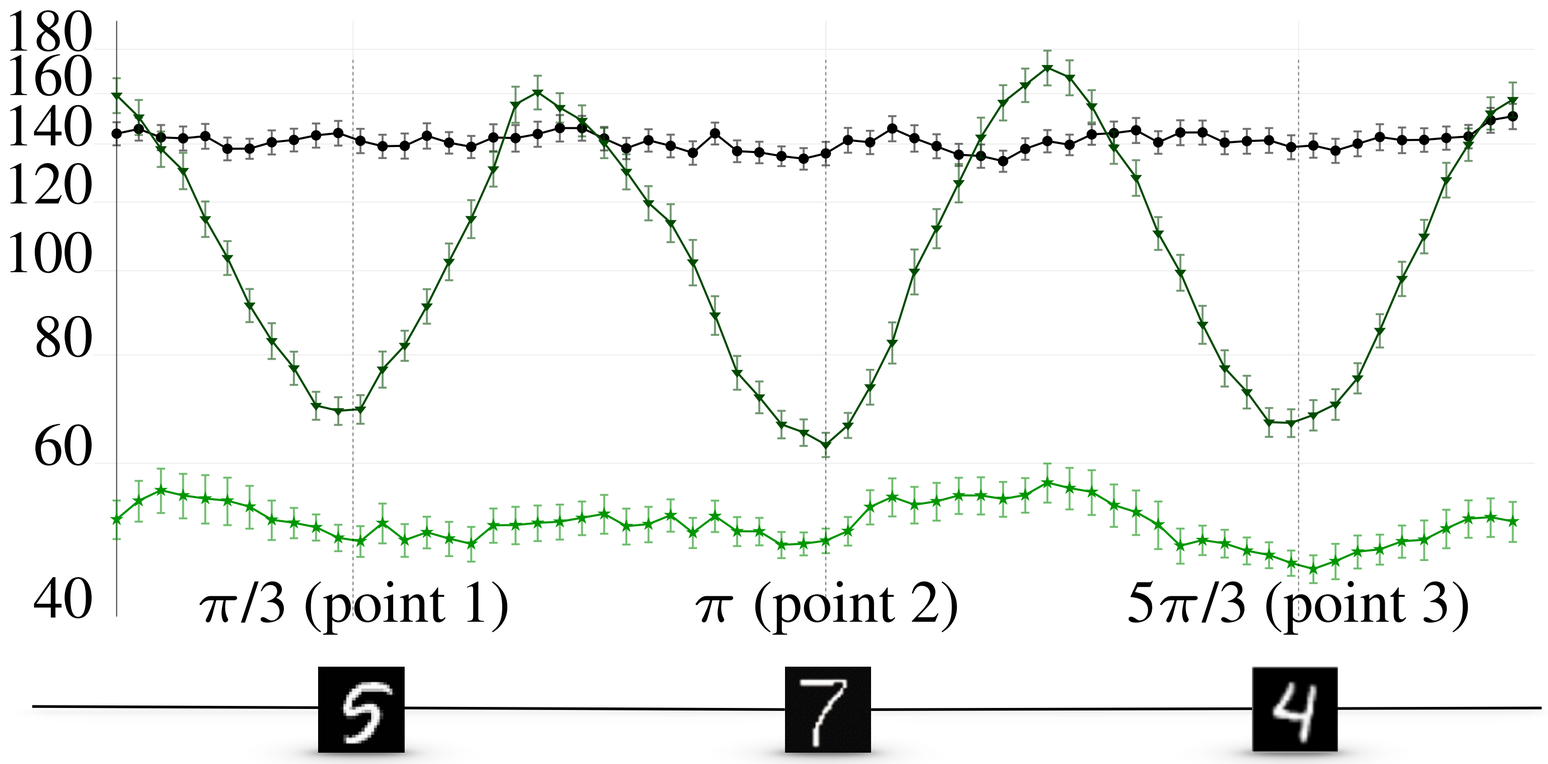
(Linear region boundaries of the last (pre-logit) over a 2D slice in the input space through three training points)

Sensitivity on/off Training Data Manifold

Transition density

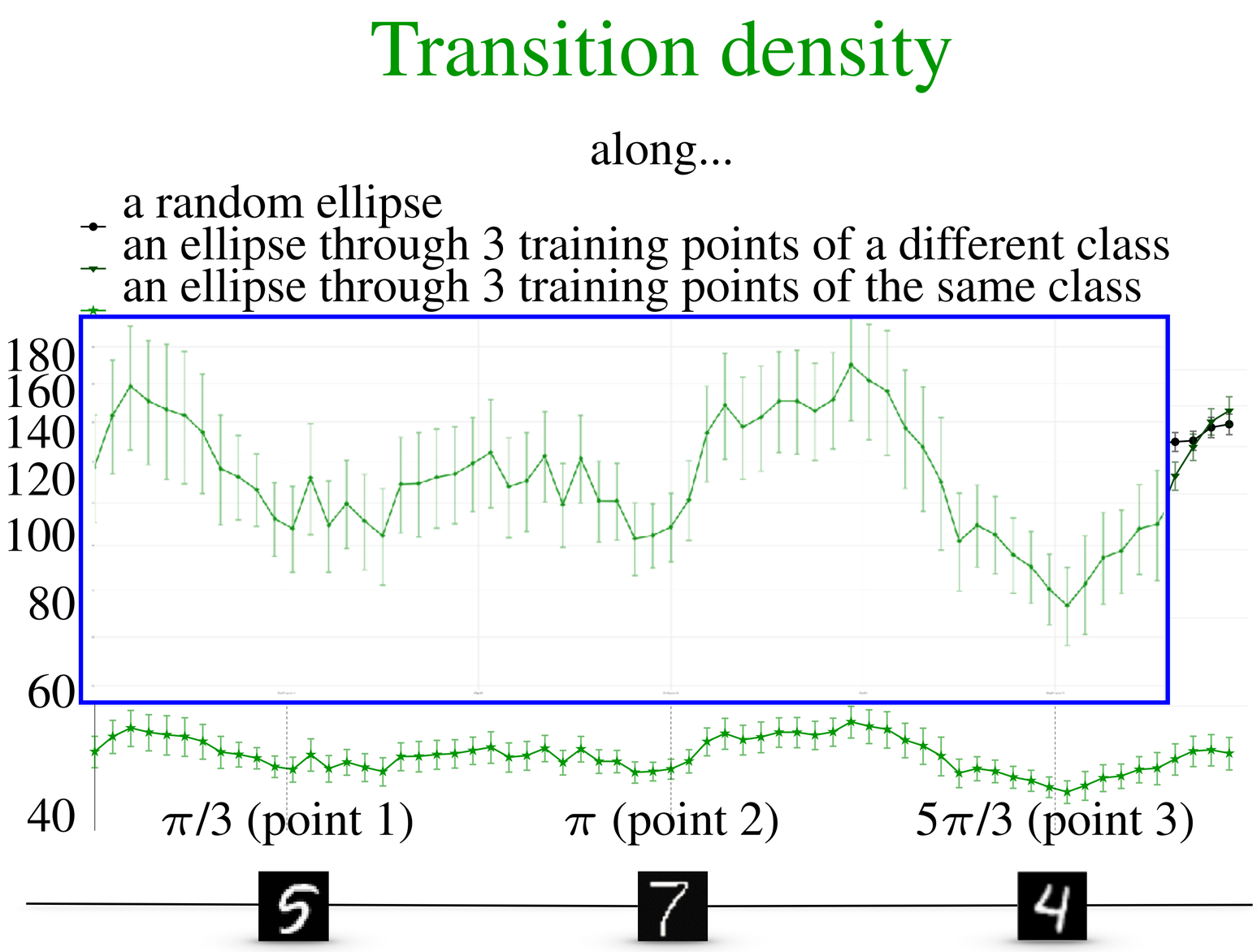
along...

- a random ellipse
- an ellipse through 3 training points of a different class
- an ellipse through 3 training points of the same class



- Transition density dips around training data.
- Interpolating different digits results in higher number of transitions.

Sensitivity on/off Training Data Manifold



From Roman:

> Is the plot correct?

I believe the plot is correct, but admittedly the trend to have dips around points (see attached plot for zoomed-in version) is much noisier than with other metrics.

> Or is this possibly related to the fact that over the entire circular trajectory through three points from the same class, network output never really changes?

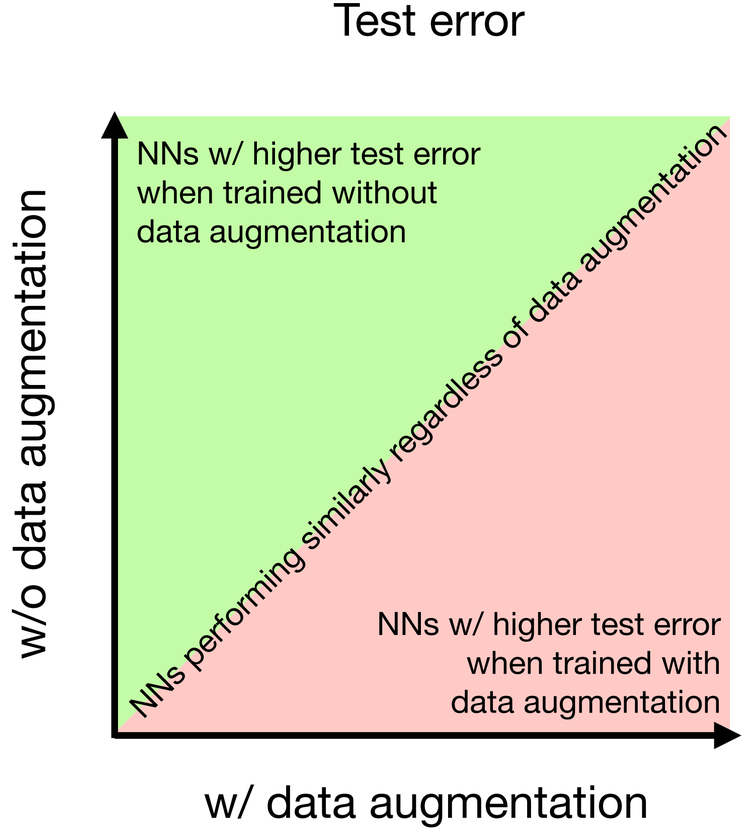
I believe it is. I think it makes sense for the change to be small relative to that along the ellipse through three different points since same-class points might be closer + their interpolations might lie closer to the data manifold.

(2) Sensitivity and Generalization Factors

- Train (to 100% training accuracy) two neural networks sharing the same architecture and optimization procedure but differing in a single binary hyper-parameter.

- Compare the resulting test error and sensitivity metrics.

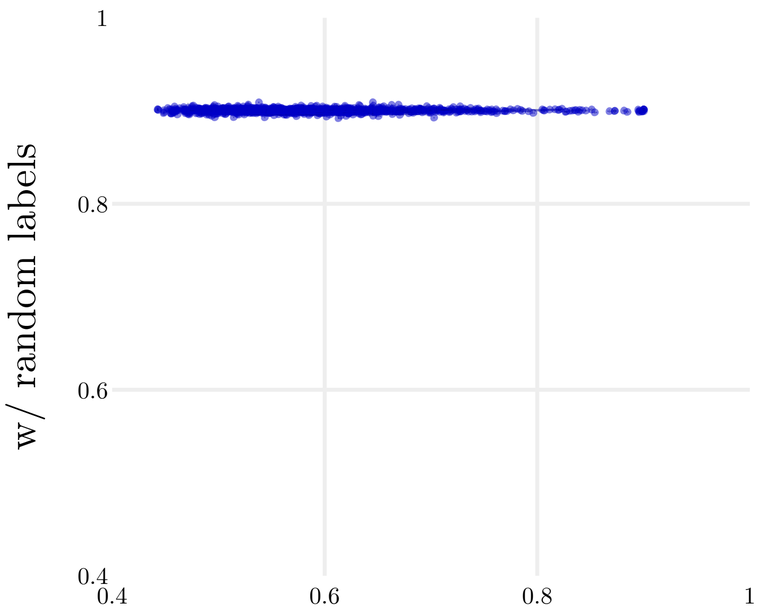
Example:



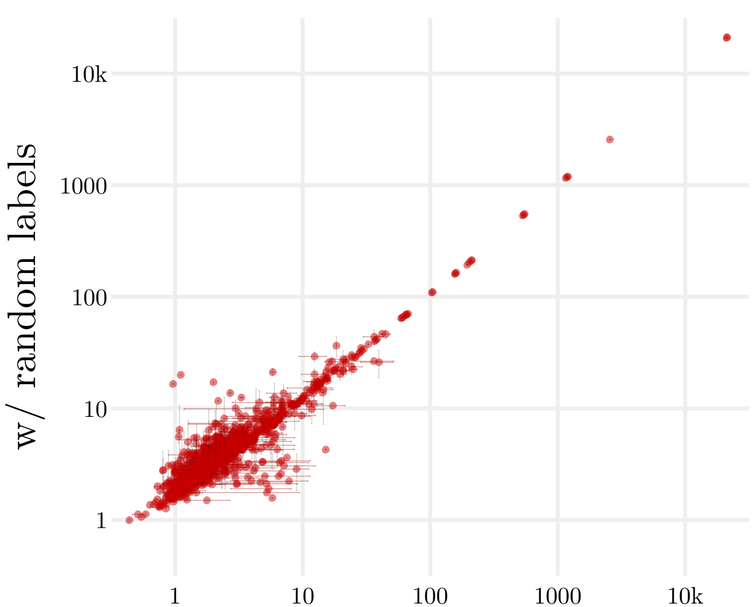
Sensitivity and Generalization Factors

Random Labels

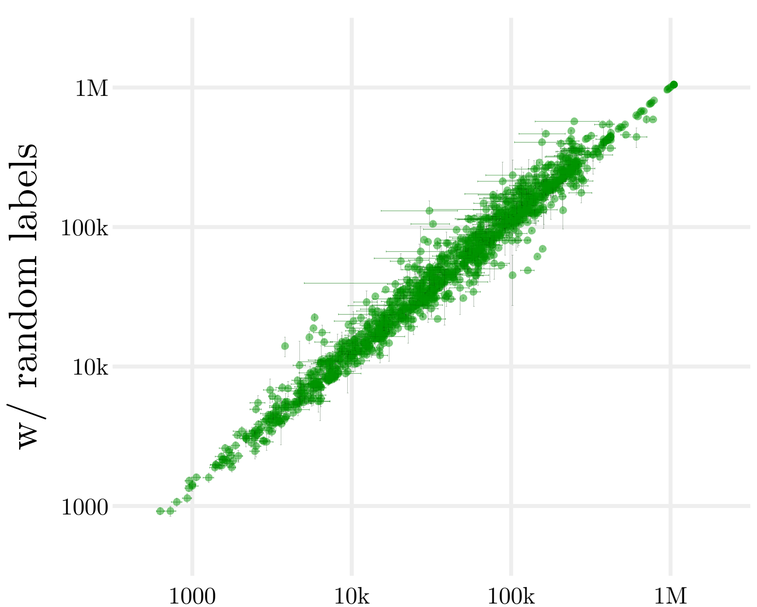
Generalization Gap



Jacobian norm



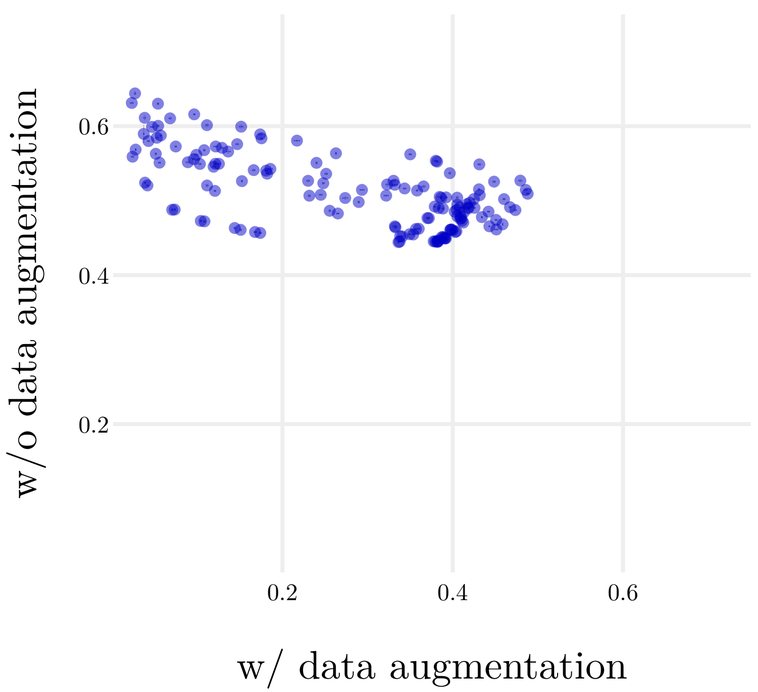
Transitions



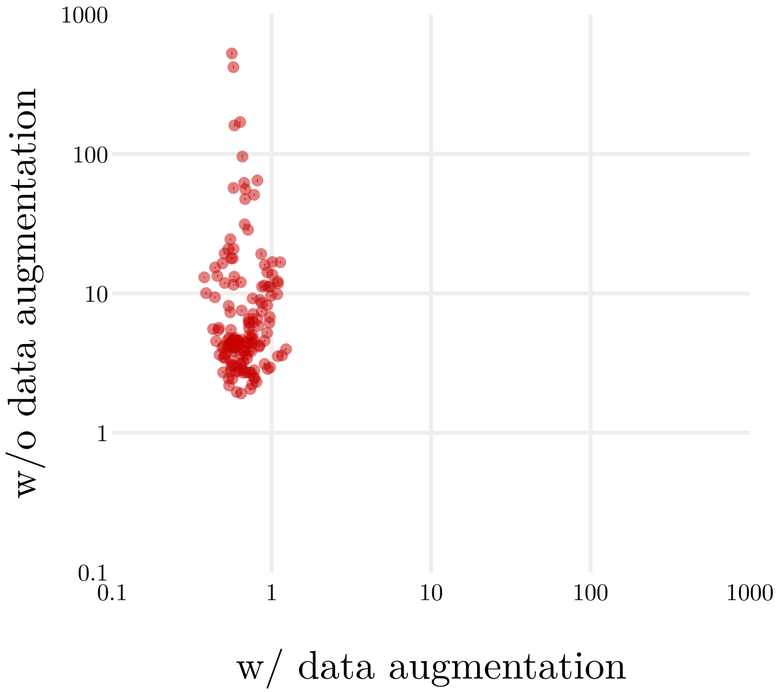
Sensitivity and Generalization Factors

Data Augmentation

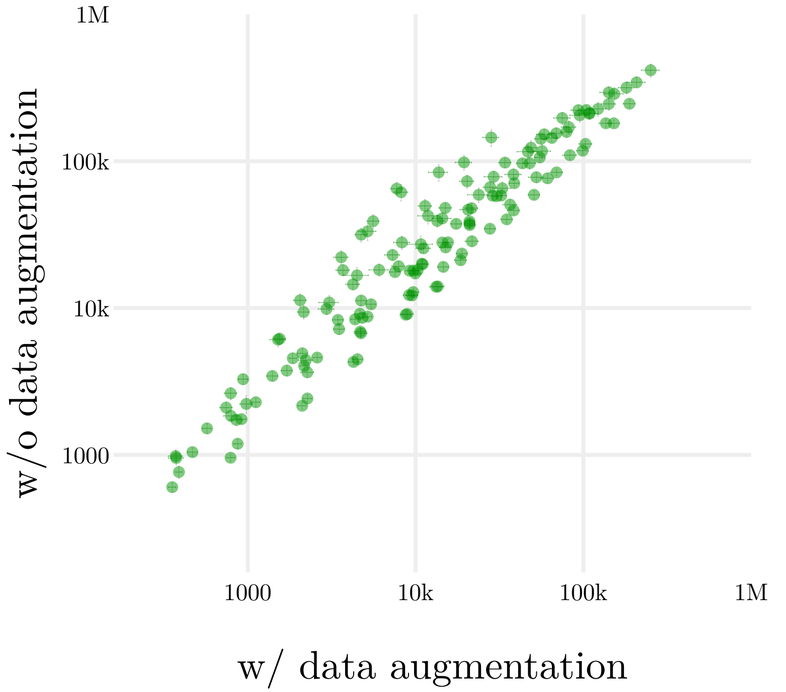
Generalization Gap



Jacobian norm



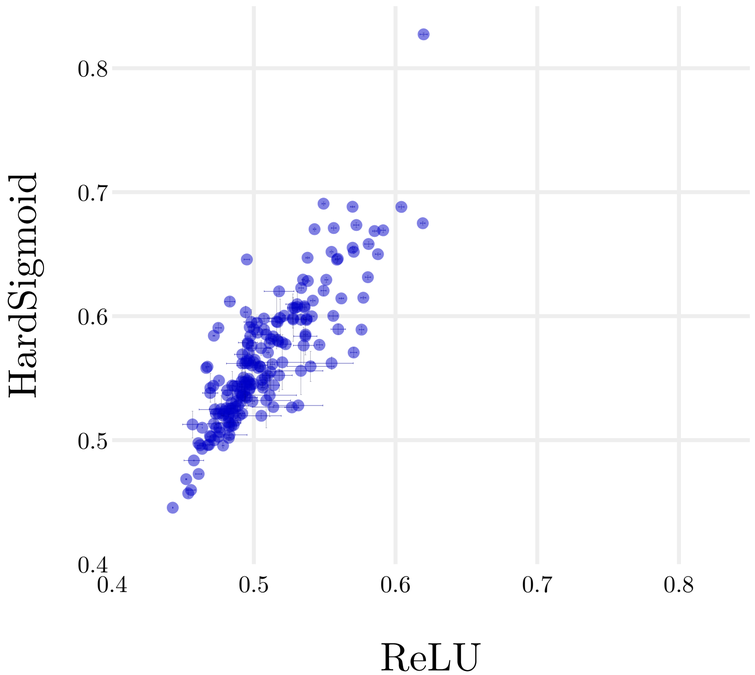
Transitions



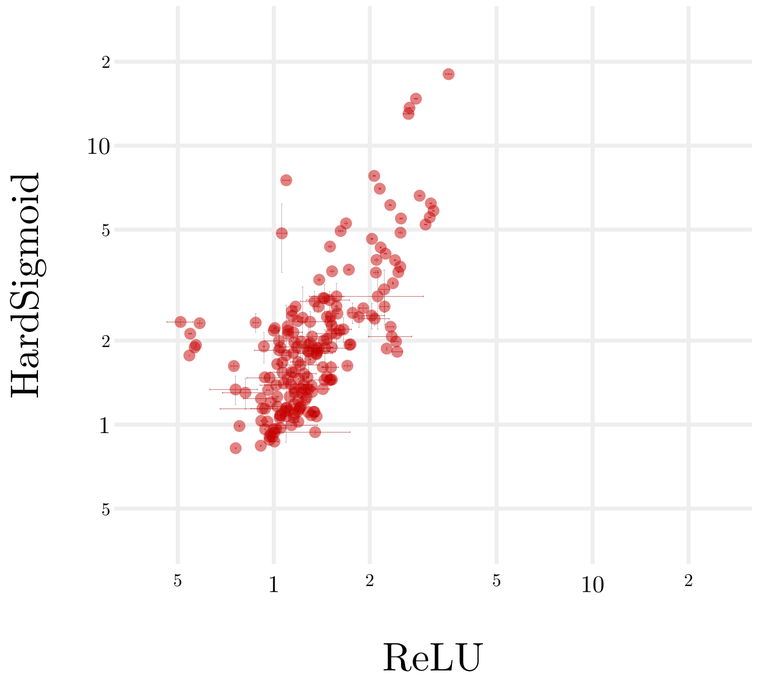
Sensitivity and Generalization Factors

ReLUs

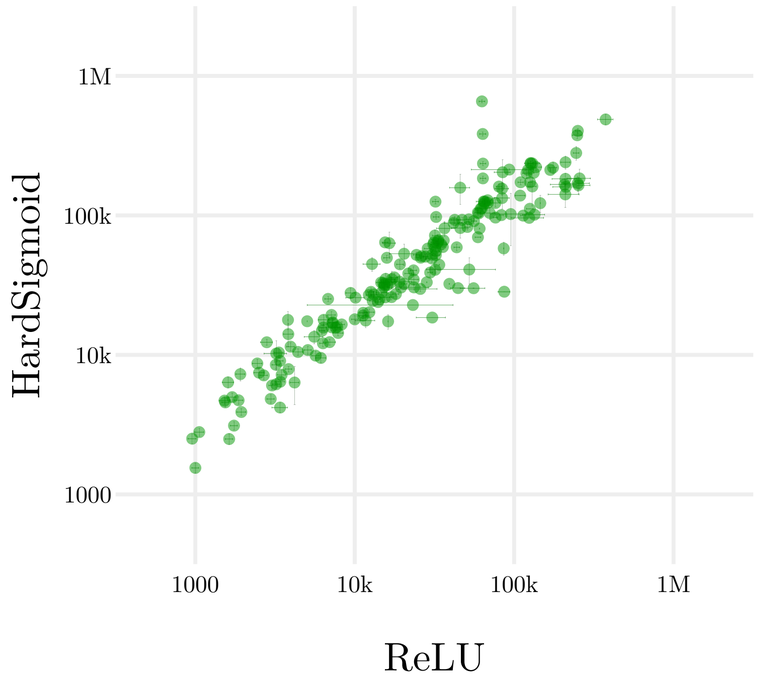
Generalization Gap



Jacobian norm



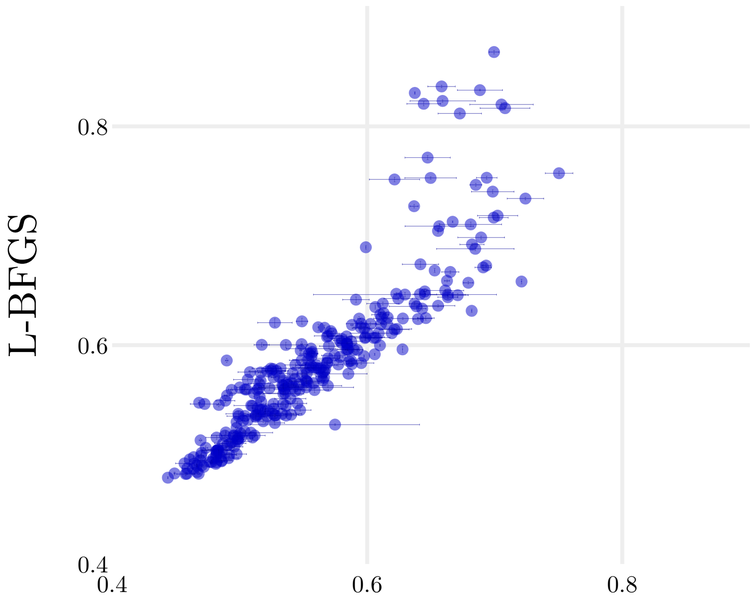
Transitions



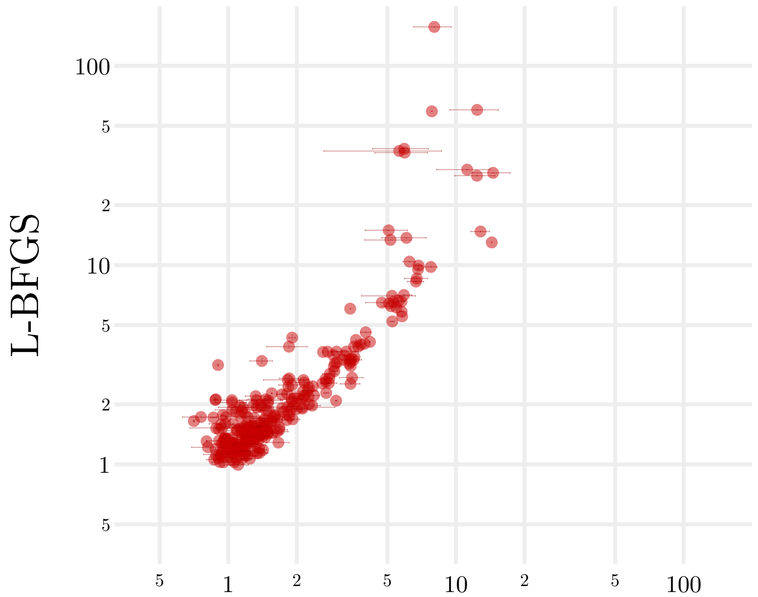
Sensitivity and Generalization Factors

Full-batch optimization

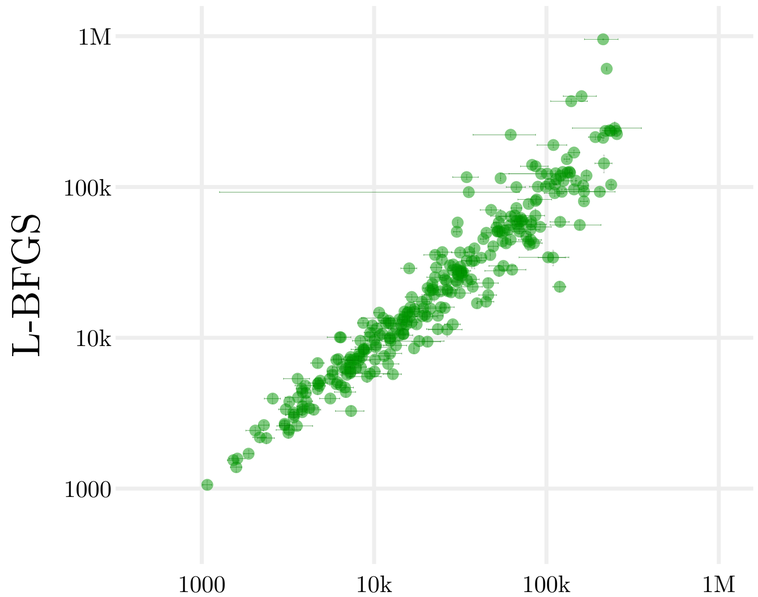
Generalization Gap



Jacobian norm



Transitions

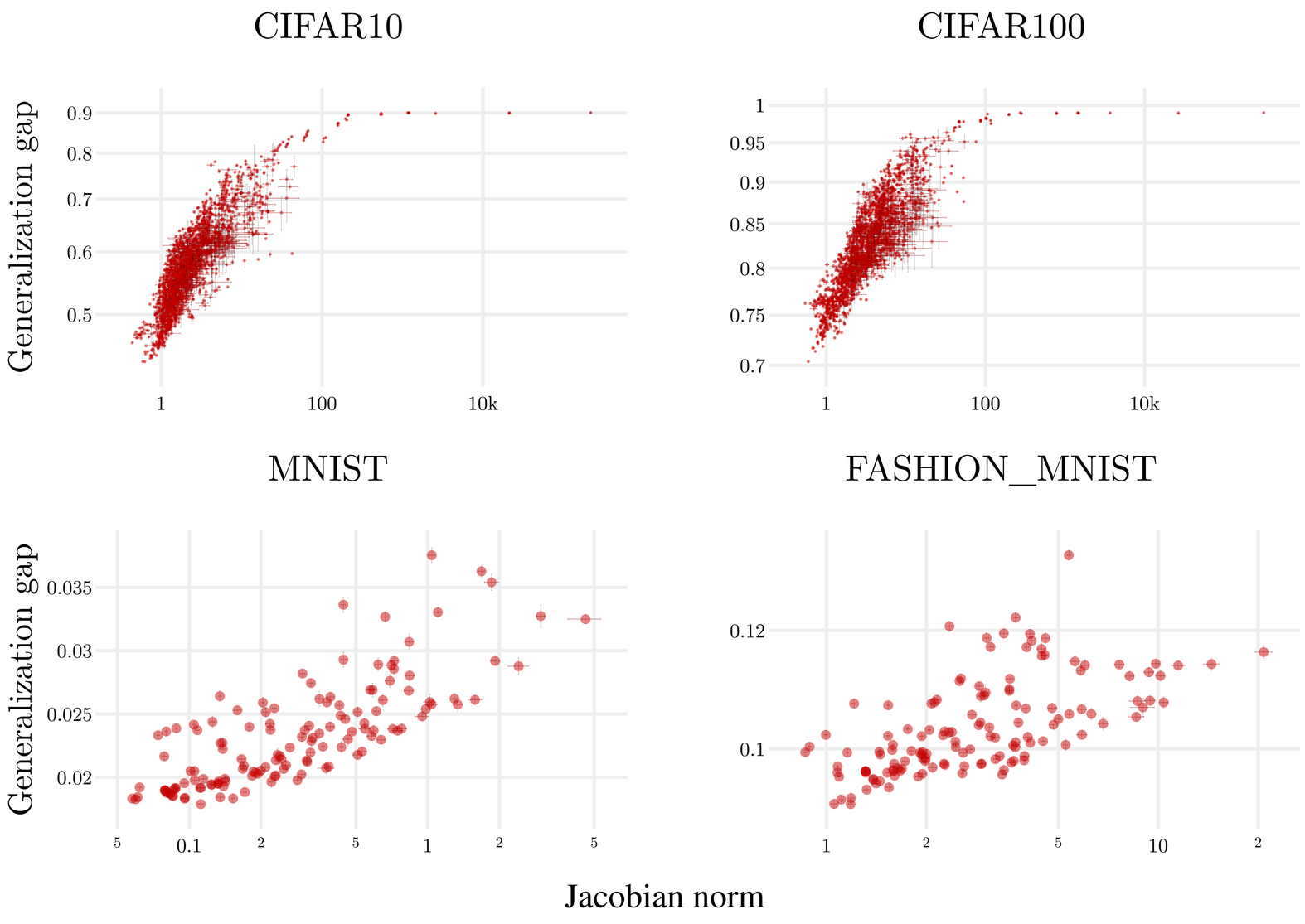


(3) Sensitivity and Generalization Gap

335671 networks were trained for 2^{19} steps with random hyper-parameters; if training did not complete, a checkpoint at step 2^{18} was used instead, if available. When using L-BFGS, the maximum number of iterations was set to 2684. The space of available hyper-parameters included[5]:

1. CIFAR10 and CIFAR100 datasets cropped to a 24×24 center region;
2. all 5 non-linearities from §A.4;
3. SGD, Momentum, ADAM (Kingma & Ba, 2014), RMSProp (Hinton et al., 2012) and L-BFGS optimizers;
4. learning rates from $\{0.01, 0.005, 0.0005\}$, when applicable. Secondary coefficients were fixed at default values of Tensorflow implementations of respective optimizers;
5. batch sizes of $\{128, 512\}$ (unless using L-BFGS with the full batch of 50000);
6. standard deviations of initial weights from $\{0.5, 1, 4, 8\}$ multiplied by the default value described in §A.5;
7. widths from $\left\{1, 2, 4, \cdots, 2^{16}\right\}$;
8. depths from $\left\{2, 3, 5, \cdots, 2^6 + 1\right\}$;
9. true and random training labels;
10. random seeds from 1 to 8.

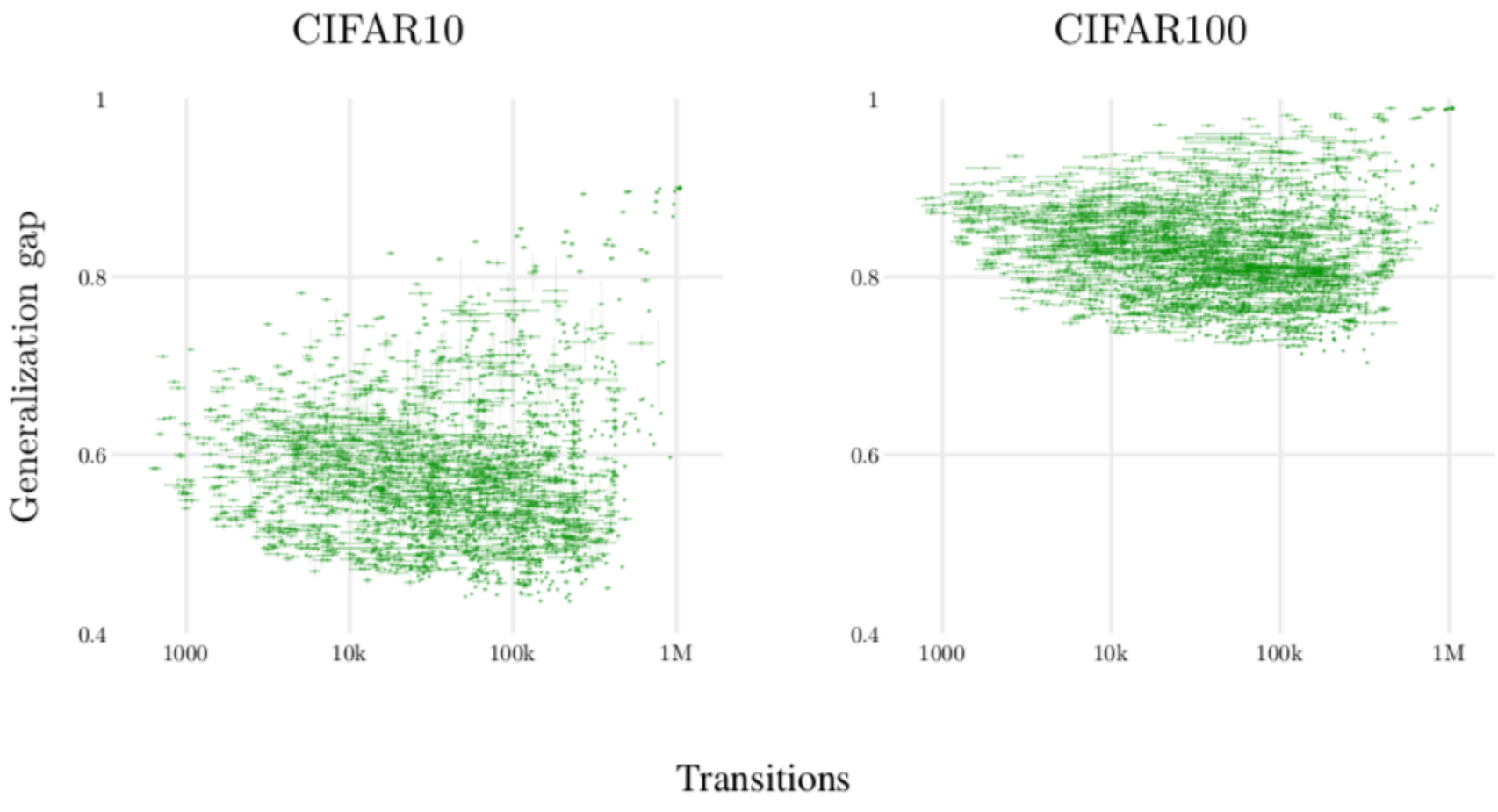
Sensitivity and Generalization Gap



All networks trained to 100% train accuracy (99.9% for CIFAR-100)

- Jacobian norm correlates with test error on 4 different image classification datasets when sweeping over many different architectural and optimization choices.
- Transition density does not.

Sensitivity and Generalization Gap

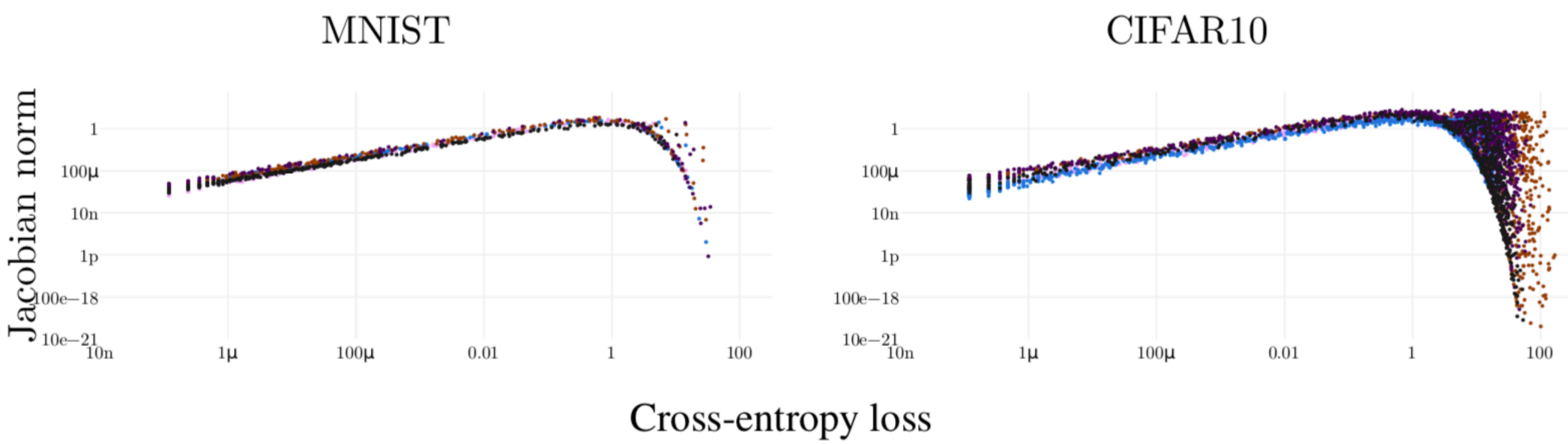


In contrast, Figure App.7 demonstrates that the number of transitions is not alone sufficient to compare networks of different sizes, as the number of neurons in the networks has a strong influence on transition count.

All networks trained to 100% train accuracy (99.9% for CIFAR-100)

Sensitivity and Per-point Generalization

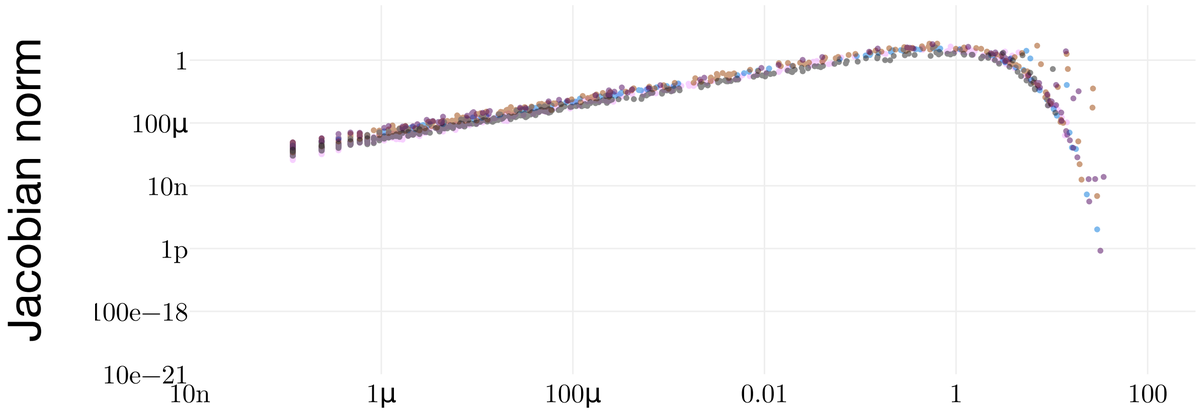
(Each point corresponds to a trained neural network evaluated at an individual test point)

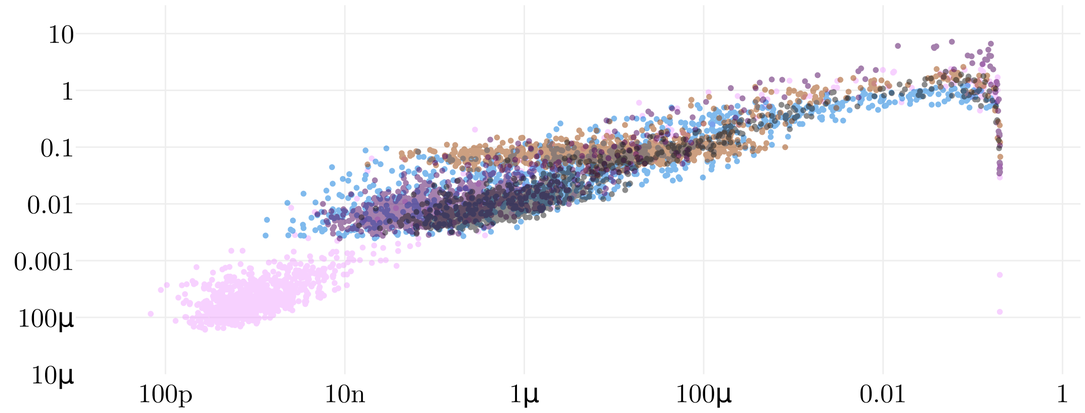


Jacobian norm is predictive of the test loss for individual points (see paper for detailed analysis).

Sensitivity and Per-point Generalization

(Each point corresponds to a trained on MNIST neural network evaluated at an individual test point)





Jacobian norm is predictive of the test loss for individual points
(see paper for detailed analysis).

A.3 BOUNDING THE JACOBIAN NORM

Here we analyze the relationship between the Jacobian norm and the cross-entropy loss at individual test points as studied in §4.4.

Target class Jacobian. We begin by relating the derivative of the target class probability $\mathbf{J}_{y(\mathbf{x})}$ to per-point cross-entropy loss $l(\mathbf{x}) = -\log\left[\mathbf{f}_\sigma(\mathbf{x})\right]_{y(\mathbf{x})}$ (where $y(\mathbf{x})$ is the correct integer class).

We will denote $\mathbf{f}_\sigma(\mathbf{x})$ by $\boldsymbol{\sigma}$ and drop the $\mathbf{x}$ argument to de-clutter notation (i.e. write $\mathbf{f}$ instead of $\mathbf{f}(\mathbf{x})$). Then the Jacobian can be expressed as

$$\mathbf{J} = \left[\left(\boldsymbol{\sigma}\mathbf{1}^T\right) \odot \left(\mathbf{I} - \boldsymbol{\sigma}\mathbf{1}^T\right)^T\right]\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}^T}\right),$$

where $\odot$ is the Hadamard element-wise product. Then indexing both sides of the equation at the correct class y yields

$$\mathbf{J}_y = \sigma_y\left((\mathbf{e}_y - \boldsymbol{\sigma})^T\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}^T}\right)\right),$$

where $\mathbf{e}_y$ is a vector of zeros everywhere except for $e_y = 1$. Taking the norm of both sides results in

$$\|\mathbf{J}_y\|_2^2 = \sigma_y^2 \sum_{k=1}^{d}\left[(1-\sigma_y)^2\left(\frac{\partial f_y}{\partial x_k}\right)^2 + \sum_{j\neq y}^{n}\left(\sigma_j \frac{\partial f_j}{\partial x_k}\right)^2\right] \tag{2}$$

$$= \sigma_y^2\left[(1-\sigma_y)^2\sum_{k=1}^{d}\left(\frac{\partial f_y}{\partial x_k}\right)^2 + \sum_{j\neq y}^{n}\sigma_j^2\sum_{k=1}^{d}\left(\frac{\partial f_j}{\partial x_k}\right)^2\right] \tag{3}$$

$$= \sigma_y^2\left[(1-\sigma_y)^2\left\|\frac{\partial f_y}{\partial \mathbf{x}^T}\right\|_2^2 + \sum_{j\neq y}^{n}\sigma_j^2\left\|\frac{\partial f_j}{\partial \mathbf{x}^T}\right\|_2^2\right] \tag{4}$$

We now assume that magnitudes of the individual logit derivatives vary little in between logits and over the input space [3]:

$$\left\|\frac{\partial f_i}{\partial \mathbf{x}^T}\right\|_2^2 \approx \frac{1}{n}\mathbb{E}_{\mathbf{x}_{\text{test}}}\left\|\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{\text{test}}^T}\right\|_F^2,$$

which simplifies Equation 4 to

$$\|\mathbf{J}_y\|_2^2 \approx M\sigma_y^2\left[(1-\sigma_y)^2 + \sum_{j\neq y}^{n}\sigma_j^2\right],$$

where $M = \mathbb{E}_{\mathbf{x}_{\text{test}}}\left\|\partial \mathbf{f}/\partial \mathbf{x}_{\text{test}}^T\right\|_F^2/n$. Since σ lies on the $(n-1)$-simplex Δ^{n-1}, under these assumptions we can bound:

$$\frac{(1-\sigma_y)^2}{n-1} \leqslant \sum_{j\neq y}^{n}\sigma_j^2 \leqslant (1-\sigma_y)^2,$$

and finally

$$\frac{n}{n-1}M\sigma_y^2(1-\sigma_y)^2 \lessapprox \|\mathbf{J}_y\|_2^2 \lessapprox 2M\sigma_y^2(1-\sigma_y)^2,$$

or, in terms of the cross-entropy loss $l = -\log\sigma_y$:

$$\sqrt{\frac{nM}{n-1}}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right) \lessapprox \|\mathbf{J}_y\|_2 \lessapprox \sqrt{2M}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right). \tag{5}$$

We validate these approximate bounds in Figure App.11 (top).

Full Jacobian. Equation 5 establishes a close relationship between $\mathbf{J}_y$ and loss $l = -\log\sigma_y$, but of course, at test time we do not know the target class y. This allows us to only bound the full Jacobian norm from below:

$$\sqrt{\frac{nM}{n-1}}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right) \lessapprox \|\mathbf{J}_y\|_2 \leqslant \|\mathbf{J}\|_F. \tag{6}$$

For the upper bound, we assume the maximum-entropy case of σ_y: $\sigma_i \approx (1-\sigma_y)/(n-1)$, for $i \neq y$. The Jacobian norm is

$$\|\mathbf{J}\|_F^2 = \sum_{i=1}^{n}\|\mathbf{J}_i\|_2^2 = \|\mathbf{J}_y\|_2^2 + \sum_{i\neq y}^{n}\|\mathbf{J}_i\|_2^2,$$

[3] In the limit of infinite width, and fully Bayesian training, deep network logits are distributed exactly according to a Gaussian process (Neal, 1994; Lee et al., 2018). Similarly, each entry in the logit Jacobian also corresponds to an independent draw from a Gaussian process (Solak et al., 2003). It is therefore plausible that the Jacobian norm, consisting of a sum over the square of independent Gaussian samples in the correct limits, will tend towards a constant.

where the first summand becomes:

$$\|\mathbf{J}_y\|_2^2 \approx M\sigma_y^2\left[(1-\sigma_y)^2 + (n-1)\left(\frac{1-\sigma_y}{n-1}\right)^2\right] = \frac{Mn}{n-1}\sigma_y^2(1-\sigma_y)^2,$$

and each of the others

$$\|\mathbf{J}_i\|_2^2 \approx M\left(\frac{1-\sigma_y}{n-1}\right)^2\left[\left(1-\frac{1-\sigma_y}{n-1}\right)^2 + \left(\sigma_y^2 + (n-2)\left(\frac{1-\sigma_y}{n-1}\right)^2\right)\right]$$

$$= \frac{M}{(n-1)^3}(1-\sigma_y)^2\left(n\sigma_y^2 + n - 2\right)^2.$$

Adding $n-1$ of such summands to $\|\mathbf{J}_y\|_2^2$ results in

$$\|\mathbf{J}\|_F \approx \frac{\sqrt{M}}{(n-1)}(1-\sigma_y)\sqrt{n^2\sigma_y^2 + n - 2} = \frac{\sqrt{M}}{(n-1)}\left(1-\mathrm{e}^{-l}\right)\sqrt{n^2\mathrm{e}^{-2l} + n - 2}, \tag{7}$$

compared against the lower bound (Equation 6) and experimental data in Figure App.11.

A.3 Bounding the Jacobian Norm

Here we analyze the relationship between the Jacobian norm and the cross-entropy loss at individual test points as studied in §4.4.

Target class Jacobian. We begin by relating the derivative of the target class probability $\mathbf{J}_{y(\mathbf{x})}$ to per-point cross-entropy loss $l(\mathbf{x}) = -\log\left[\mathbf{f}_\sigma(\mathbf{x})\right]_{y(\mathbf{x})}$ (where $y(\mathbf{x})$ is the correct integer class).

We will denote $\mathbf{f}_\sigma(\mathbf{x})$ by $\boldsymbol{\sigma}$ and drop the $\mathbf{x}$ argument to de-clutter notation (i.e. write $\mathbf{f}$ instead of $\mathbf{f}(\mathbf{x})$). Then the Jacobian can be expressed as

$$\mathbf{J} = \left[\left(\boldsymbol{\sigma}\mathbf{1}^T\right) \odot \left(\mathbf{I} - \boldsymbol{\sigma}\mathbf{1}^T\right)^T\right]\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}^T}\right),$$

where $\odot$ is the Hadamard element-wise product. Then indexing both sides of the equation at the correct class y yields

$$\mathbf{J}_y = \boldsymbol{\sigma}_y\left((\mathbf{e}_y - \boldsymbol{\sigma})^T\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}^T}\right)\right),$$

where $\mathbf{e}_y$ is a vector of zeros everywhere except for $e_y = 1$. Taking the norm of both sides results in

$$\|\mathbf{J}_y\|_2^2 = \sigma_y^2 \sum_{k=1}^{d}\left[(1-\sigma_y)^2\left(\frac{\partial f_y}{\partial x_k}\right)^2 + \sum_{j\neq y}^{n}\left(\sigma_j \frac{\partial f_j}{\partial x_k}\right)^2\right] \tag{2}$$

$$= \sigma_y^2\left[(1-\sigma_y)^2 \sum_{k=1}^{d}\left(\frac{\partial f_y}{\partial x_k}\right)^2 + \sum_{j\neq y}^{n}\sigma_j^2 \sum_{k=1}^{d}\left(\frac{\partial f_j}{\partial x_k}\right)^2\right] \tag{3}$$

$$= \sigma_y^2\left[(1-\sigma_y)^2\left\|\frac{\partial f_y}{\partial \mathbf{x}^T}\right\|_2^2 + \sum_{j\neq y}^{n}\sigma_j^2\left\|\frac{\partial f_j}{\partial \mathbf{x}^T}\right\|_2^2\right] \tag{4}$$

We now assume that magnitudes of the individual logit derivatives vary little in between logits and over the input space [3]:

$$\left\|\frac{\partial f_i}{\partial \mathbf{x}^T}\right\|_2^2 \approx \frac{1}{n}\mathbb{E}_{\mathbf{x}_{\text{test}}}\left\|\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{\text{test}}^T}\right\|_F^2,$$

which simplifies Equation 4 to

$$\|\mathbf{J}_y\|_2^2 \approx M\sigma_y^2\left[(1-\sigma_y)^2 + \sum_{j\neq y}^{n}\sigma_j^2\right],$$

where $M = \mathbb{E}_{\mathbf{x}_{\text{test}}}\left\|\partial \mathbf{f}/\partial \mathbf{x}_{\text{test}}^T\right\|_F^2/n$. Since σ lies on the $(n-1)$-simplex Δ^{n-1}, under these assumptions we can bound:

$$\frac{(1-\sigma_y)^2}{n-1} \leqslant \sum_{j\neq y}^{n}\sigma_j^2 \leqslant (1-\sigma_y)^2,$$

and finally

$$\frac{n}{n-1}M\sigma_y^2(1-\sigma_y)^2 \lessapprox \|\mathbf{J}_y\|_2^2 \lessapprox 2M\sigma_y^2(1-\sigma_y)^2,$$

or, in terms of the cross-entropy loss $l = -\log\sigma_y$:

$$\sqrt{\frac{nM}{n-1}}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right) \lessapprox \|\mathbf{J}_y\|_2 \lessapprox \sqrt{2M}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right). \tag{5}$$

We validate these approximate bounds in Figure App.11 (top).

Full Jacobian. Equation 5 establishes a close relationship between $\mathbf{J}_y$ and loss $l = -\log\sigma_y$, but of course, at test time we do not know the target class y. This allows us to only bound the full Jacobian norm from below:

$$\sqrt{\frac{nM}{n-1}}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right) \lessapprox \|\mathbf{J}_y\|_2 \leqslant \|\mathbf{J}\|_F. \tag{6}$$

For the upper bound, we assume the maximum-entropy case of σ_y: $\sigma_i \approx (1-\sigma_y)/(n-1)$, for $i \neq y$. The Jacobian norm is

$$\|\mathbf{J}\|_F^2 = \sum_{i=1}^{n}\|\mathbf{J}_i\|_2^2 = \|\mathbf{J}_y\|_2^2 + \sum_{i\neq y}^{n}\|\mathbf{J}_i\|_2^2,$$

[3] In the limit of infinite width, and fully Bayesian training, deep network logits are distributed exactly according to a Gaussian process (Neal, 1994; Lee et al., 2018). Similarly, each entry in the logit Jacobian also corresponds to an independent draw from a Gaussian process (Solak et al., 2003). It is therefore plausible that the Jacobian norm, consisting of a sum over the square of independent Gaussian samples in the correct limits, will tend towards a constant.

where the first summand becomes:

$$\|\mathbf{J}_y\|_2^2 \approx M\sigma_y^2\left[(1-\sigma_y)^2 + (n-1)\left(\frac{1-\sigma_y}{n-1}\right)^2\right] = \frac{Mn}{n-1}\sigma_y^2(1-\sigma_y)^2,$$

and each of the others

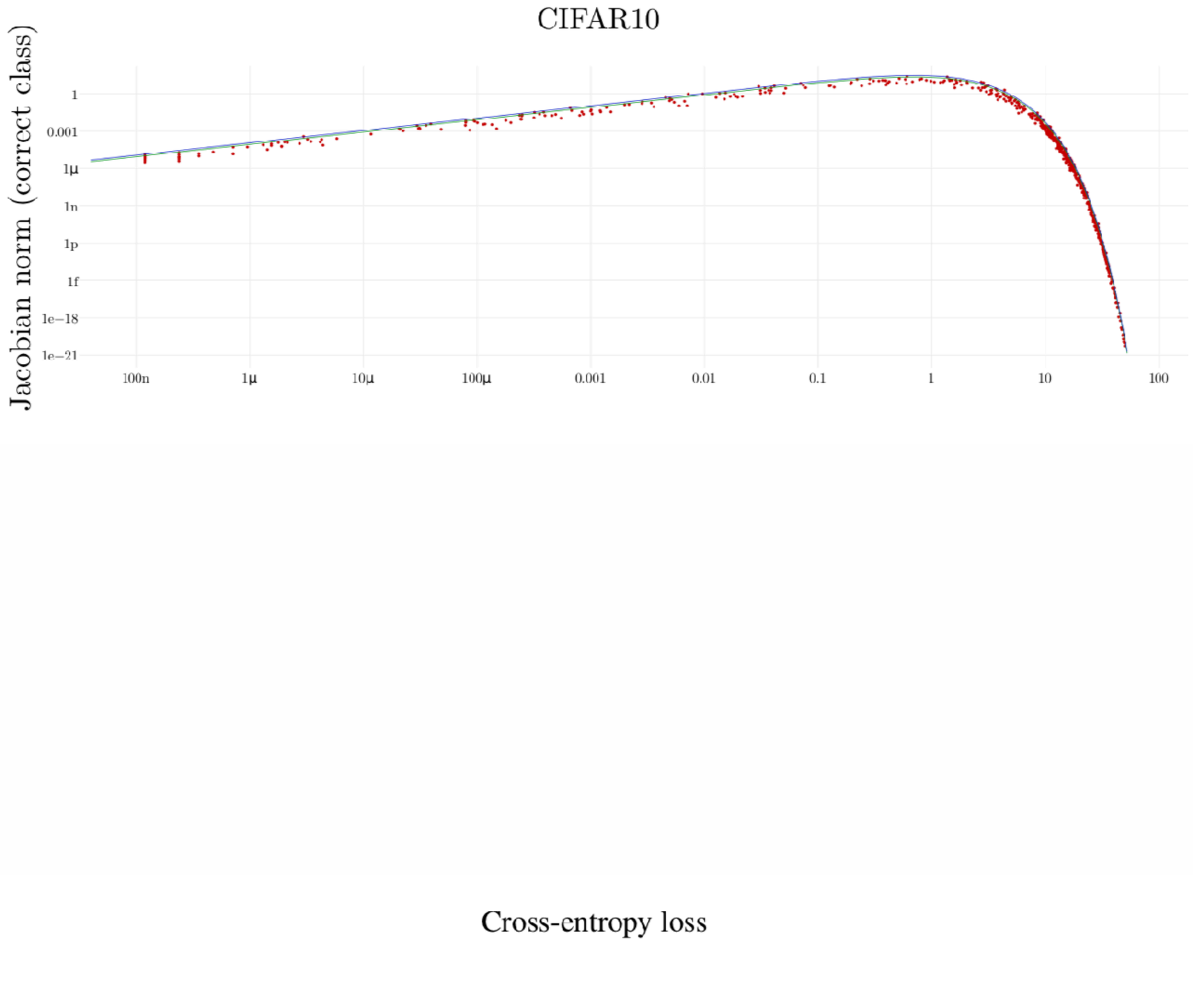
$$\|\mathbf{J}_i\|_2^2 \approx M\left(\frac{1-\sigma_y}{n-1}\right)^2\left[\left(1-\frac{1-\sigma_y}{n-1}\right)^2 + \left(\sigma_y^2 + (n-2)\left(\frac{1-\sigma_y}{n-1}\right)^2\right)\right]$$

$$= \frac{M}{(n-1)^3}(1-\sigma_y)^2\left(n\sigma_y^2 + n - 2\right)^2.$$

Adding $n-1$ of such summands to $\|\mathbf{J}_y\|_2^2$ results in

$$\|\mathbf{J}\|_F \approx \frac{\sqrt{M}}{(n-1)}(1-\sigma_y)\sqrt{n^2\sigma_y^2 + n - 2} = \frac{\sqrt{M}}{(n-1)}\left(1-\mathrm{e}^{-l}\right)\sqrt{n^2\mathrm{e}^{-2l} + n - 2}, \tag{7}$$

compared against the lower bound (Equation 6) and experimental data in Figure App.11.



A.3 Bounding the Jacobian Norm

Here we analyze the relationship between the Jacobian norm and the cross-entropy loss at individual test points as studied in §4.4.

Target class Jacobian. We begin by relating the derivative of the target class probability $\mathbf{J}_{y(\mathbf{x})}$ to per-point cross-entropy loss $l(\mathbf{x}) = -\log\left[\mathbf{f}_\sigma(\mathbf{x})\right]_{y(\mathbf{x})}$ (where $y(\mathbf{x})$ is the correct integer class).

We will denote $\mathbf{f}_\sigma(\mathbf{x})$ by $\boldsymbol{\sigma}$ and drop the $\mathbf{x}$ argument to de-clutter notation (i.e. write $\mathbf{f}$ instead of $\mathbf{f}(\mathbf{x})$). Then the Jacobian can be expressed as

$$\mathbf{J} = \left[\left(\boldsymbol{\sigma}\mathbf{1}^T\right) \odot \left(\mathbf{I} - \boldsymbol{\sigma}\mathbf{1}^T\right)^T\right]\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}^T}\right),$$

where $\odot$ is the Hadamard element-wise product. Then indexing both sides of the equation at the correct class y yields

$$\mathbf{J}_y = \sigma_y\left((\mathbf{e}_y - \boldsymbol{\sigma})^T\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}^T}\right)\right),$$

where $\mathbf{e}_y$ is a vector of zeros everywhere except for $e_y = 1$. Taking the norm of both sides results in

$$\|\mathbf{J}_y\|_2^2 = \sigma_y^2 \sum_{k=1}^{d}\left[(1-\sigma_y)^2\left(\frac{\partial f_y}{\partial x_k}\right)^2 + \sum_{j\neq y}^{n}\left(\sigma_j \frac{\partial f_j}{\partial x_k}\right)^2\right] \tag{2}$$

$$= \sigma_y^2\left[(1-\sigma_y)^2\sum_{k=1}^{d}\left(\frac{\partial f_y}{\partial x_k}\right)^2 + \sum_{j\neq y}^{n}\sigma_j^2\sum_{k=1}^{d}\left(\frac{\partial f_j}{\partial x_k}\right)^2\right] \tag{3}$$

$$= \sigma_y^2\left[(1-\sigma_y)^2\left\|\frac{\partial f_y}{\partial \mathbf{x}^T}\right\|_2^2 + \sum_{j\neq y}^{n}\sigma_j^2\left\|\frac{\partial f_j}{\partial \mathbf{x}^T}\right\|_2^2\right] \tag{4}$$

We now assume that magnitudes of the individual logit derivatives vary little in between logits and over the input space [3]:

$$\left\|\frac{\partial f_i}{\partial \mathbf{x}^T}\right\|_2^2 \approx \frac{1}{n}\mathbb{E}_{\mathbf{x}_{\text{test}}}\left\|\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{\text{test}}^T}\right\|_F^2,$$

which simplifies Equation 4 to

$$\|\mathbf{J}_y\|_2^2 \approx M\sigma_y^2\left[(1-\sigma_y)^2 + \sum_{j\neq y}^{n}\sigma_j^2\right],$$

where $M = \mathbb{E}_{\mathbf{x}_{\text{test}}}\left\|\partial \mathbf{f}/\partial \mathbf{x}_{\text{test}}^T\right\|_F^2 / n$. Since σ lies on the $(n-1)$-simplex Δ^{n-1}, under these assumptions we can bound:

$$\frac{(1-\sigma_y)^2}{n-1} \leqslant \sum_{j\neq y}^{n}\sigma_j^2 \leqslant (1-\sigma_y)^2,$$

and finally

$$\frac{n}{n-1}M\sigma_y^2(1-\sigma_y)^2 \lessapprox \|\mathbf{J}_y\|_2^2 \lessapprox 2M\sigma_y^2(1-\sigma_y)^2,$$

or, in terms of the cross-entropy loss $l = -\log\sigma_y$:

$$\sqrt{\frac{nM}{n-1}}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right) \lessapprox \|\mathbf{J}_y\|_2 \lessapprox \sqrt{2M}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right). \tag{5}$$

We validate these approximate bounds in Figure App.11 (top).

Full Jacobian. Equation 5 establishes a close relationship between $\mathbf{J}_y$ and loss $l = -\log\sigma_y$, but of course, at test time we do not know the target class y. This allows us to only bound the full Jacobian norm from below:

$$\sqrt{\frac{nM}{n-1}}\mathrm{e}^{-l}\left(1-\mathrm{e}^{-l}\right) \lessapprox \|\mathbf{J}_y\|_2 \leqslant \|\mathbf{J}\|_F. \tag{6}$$

For the upper bound, we assume the maximum-entropy case of σ_y: $\sigma_i \approx (1-\sigma_y)/(n-1)$, for $i \neq y$. The Jacobian norm is

$$\|\mathbf{J}\|_F^2 = \sum_{i=1}^{n}\|\mathbf{J}_i\|_2^2 = \|\mathbf{J}_y\|_2^2 + \sum_{i\neq y}^{n}\|\mathbf{J}_i\|_2^2,$$

[3] In the limit of infinite width, and fully Bayesian training, deep network logits are distributed exactly according to a Gaussian process (Neal, 1994; Lee et al., 2018). Similarly, each entry in the logit Jacobian also corresponds to an independent draw from a Gaussian process (Solak et al., 2003). It is therefore plausible that the Jacobian norm, consisting of a sum over the square of independent Gaussian samples in the correct limits, will tend towards a constant.

where the first summand becomes:

$$\|\mathbf{J}_y\|_2^2 \approx M\sigma_y^2\left[(1-\sigma_y)^2 + (n-1)\left(\frac{1-\sigma_y}{n-1}\right)^2\right] = \frac{Mn}{n-1}\sigma_y^2(1-\sigma_y)^2,$$

and each of the others

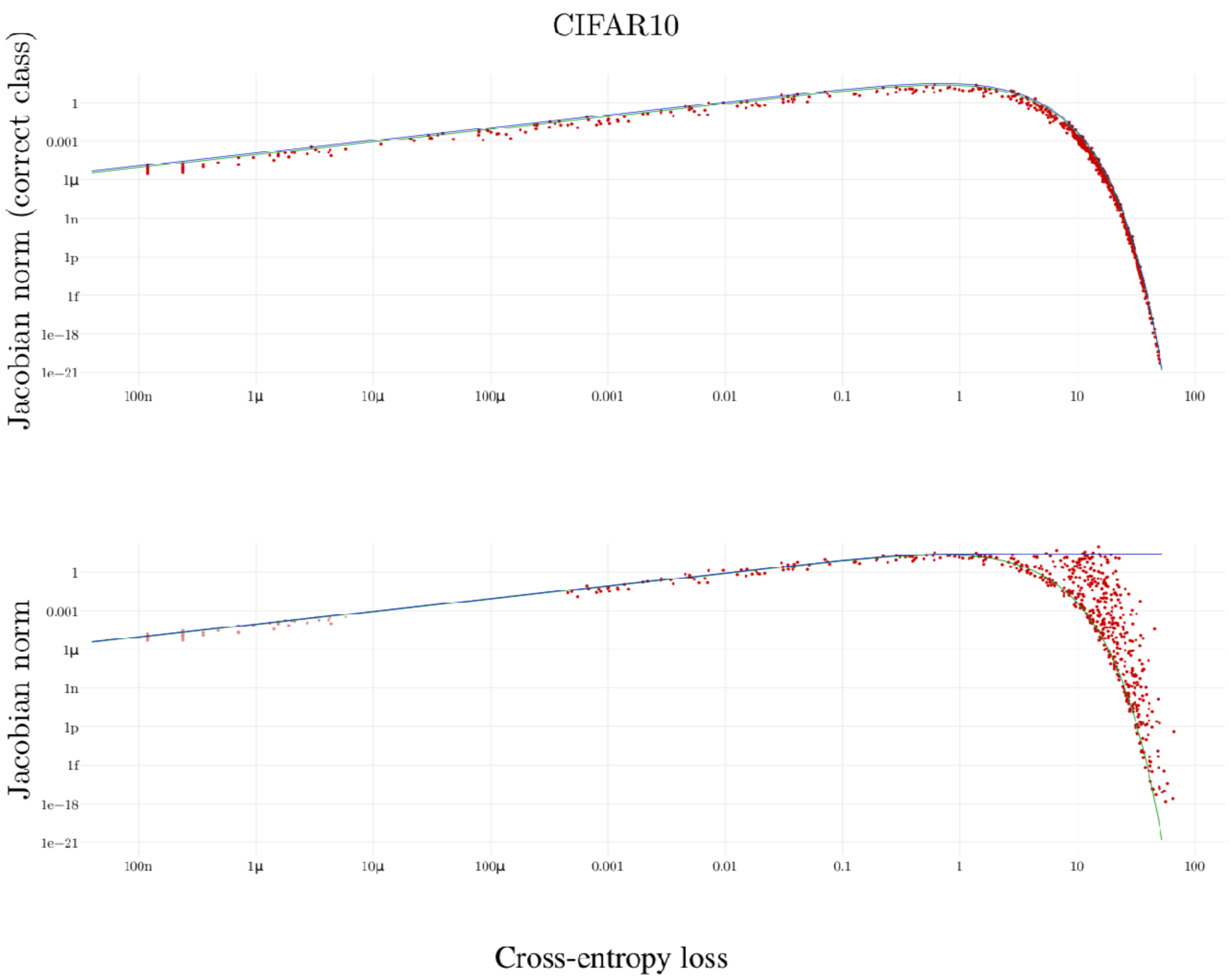
$$\|\mathbf{J}_i\|_2^2 \approx M\left(\frac{1-\sigma_y}{n-1}\right)^2\left[\left(1-\frac{1-\sigma_y}{n-1}\right)^2 + \left(\sigma_y^2 + (n-2)\left(\frac{1-\sigma_y}{n-1}\right)^2\right)\right]$$

$$= \frac{M}{(n-1)^3}(1-\sigma_y)^2\left(n\sigma_y^2 + n - 2\right)^2.$$

Adding $n-1$ of such summands to $\|\mathbf{J}_y\|_2^2$ results in

$$\|\mathbf{J}\|_F \approx \frac{\sqrt{M}}{(n-1)}(1-\sigma_y)\sqrt{n^2\sigma_y^2 + n - 2} = \frac{\sqrt{M}}{(n-1)}\left(1-\mathrm{e}^{-l}\right)\sqrt{n^2\mathrm{e}^{-2l} + n - 2}, \tag{7}$$

compared against the lower bound (Equation 6) and experimental data in Figure App.11.



Conclusion

- Trained NNs implement functions that are significantly more stable around the training data manifold than away from it.

- Jacobian norm of a trained NN evaluated at test points is predictive of test error.

Ideas for future work?

- ?