Sensitivity and Generalization in Neural Networks: an Empirical Study

R. Novak, Y. Bahri, D. Abolafia, J. Pennington, J. Sohl-Dickstein

ICLR 2018

Matteo Gamba

21st January 2020

Understanding Deep Networks

model interpretability

Perspective of model interpretability.

Leading question:

- make sense of the generalization ability of deep networks
- explain “empirical priors” (design choices that boost generalization)
- explain empirical phenomena

Main Tracks

Main tracks (non-exhaustive):

- Loss landscape and quality of solutions (*Keskar et al. 2017. Choromanska et al. 2015*).
- Optimization paths (*Arora 2019, Arora et al. 2019*).
- Implicit bias of network architectures (and overparametrization) (*Arora et al., 2018*).
- Implicit bias of SGD (*Neyshabur et al., 2015, 2017 and 2018*).

Main Tracks

Main tracks (non-exhaustive):

- Capacity control and complexity measures (bounds on population error) (*Dziugaite et al. 2017. Neyshabur et al. 2015*).
- Expressivity (*Montufar el al. 2014, Pascanu et al. 2013. Raghu et al. 2016*).
- Effective capacity of deep networks (*Hanin and Rolnick, 2019. Zhang et al., 2016*).

Common goals

Common goals:

- Find good predictors of generalization.
- Novel regularization strategies (impose explicit constraints).
- Improved architectures / initialization schemes / optimization algorithms.
- Novel theory? (Arora: describe and study optimization paths).
- Explain empirical observations.

Focus on “understanding” as making predictions.

Empirical observations

Exploring generalization in deep learning

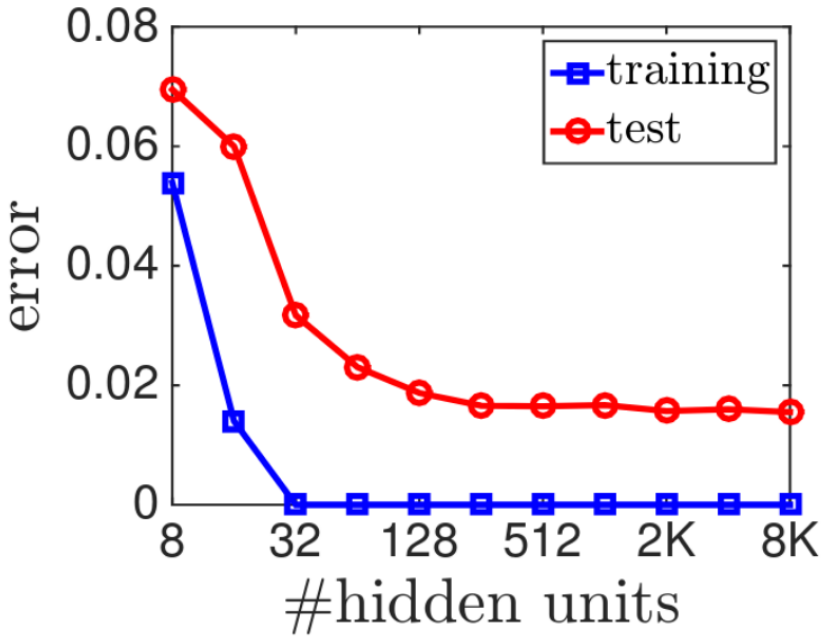


Figure: Single-layer perceptron trained on MNIST.

(source: Neyshabur et al. Exploring generalization in deep learning. 2017)

Empirical Observations

Understanding deep learning requires rethinking generalization

Understanding Deep Learning Requires Rethinking Generalization. Zhang el al. 2016.

- state-of-the-art networks can fit random labellings of the data (MNIST and CIFAR-10).
- related to empirical Rademacher complexity and empirical VC dimension.

Preliminary Experiments

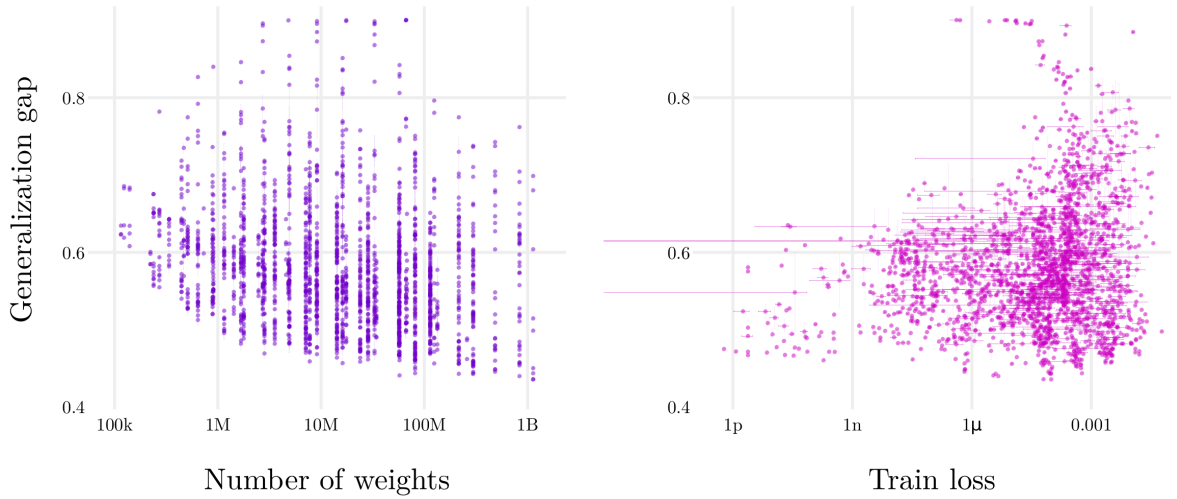


Figure: 2160 networks with various hyperparameters trained on CIFAR-10.

- Increasing the capacity of a model allows for overfitting, but the largest models are the best performing.
- Train loss does not correlate well with generalization. Best model has high loss compared to others.

Occam's Razor and Model Selection

Apparent empirical paradox: deep networks seem to defy Occam's razor for model selection.

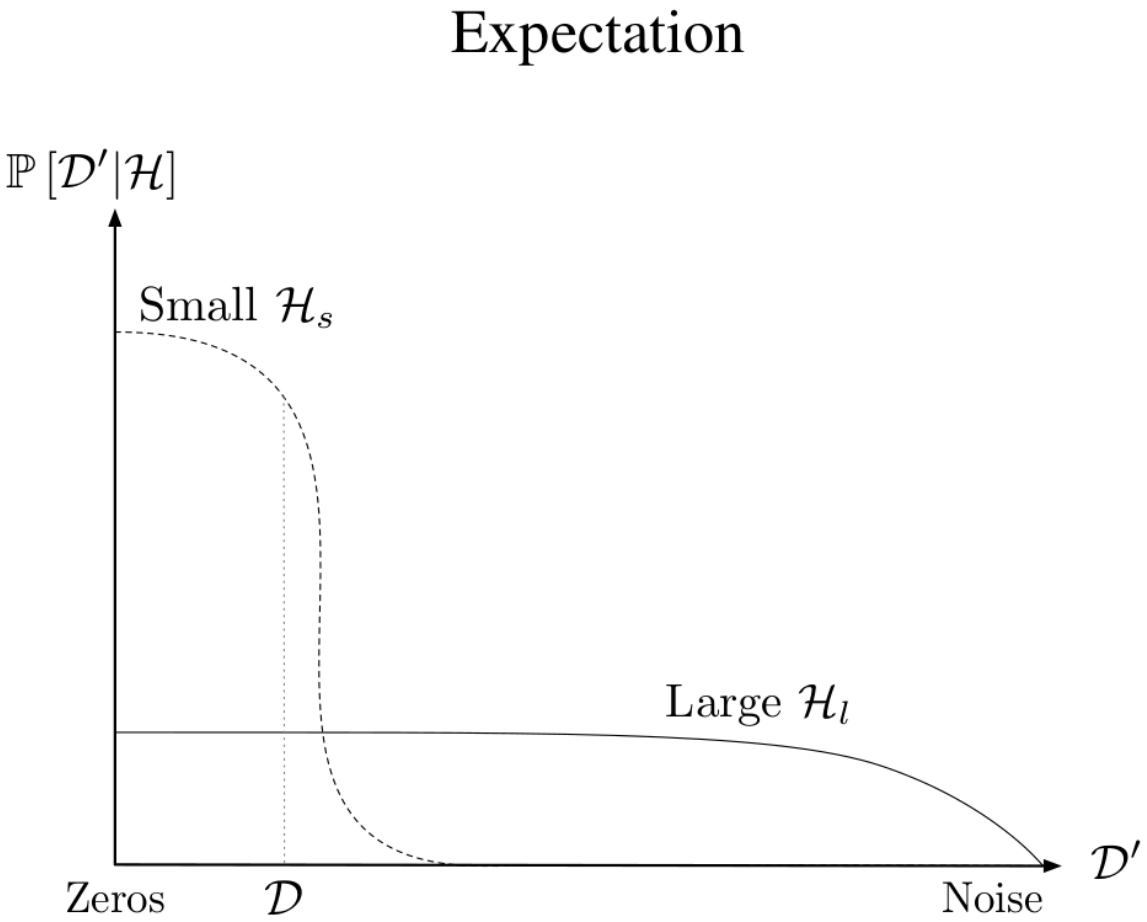
Occam's Razor and Model Selection

Apparent empirical paradox: deep networks seem to defy Occam's razor for model selection.

Intuitive argument:

- consider statistical hypothesis $H \in \mathcal{H}$.
- evidence $\mathbb{P}(\mathcal{D}|H)$ interpreted as a normalized probability distribution over "dataset space"
- a larger model can in principle fit more datasets, hence distributes the evidence more evenly across datasets
- model selection based on evidence favours smaller models

Occam's Razor and Model Selection



Occam's Razor and Model Selection

apparent paradox

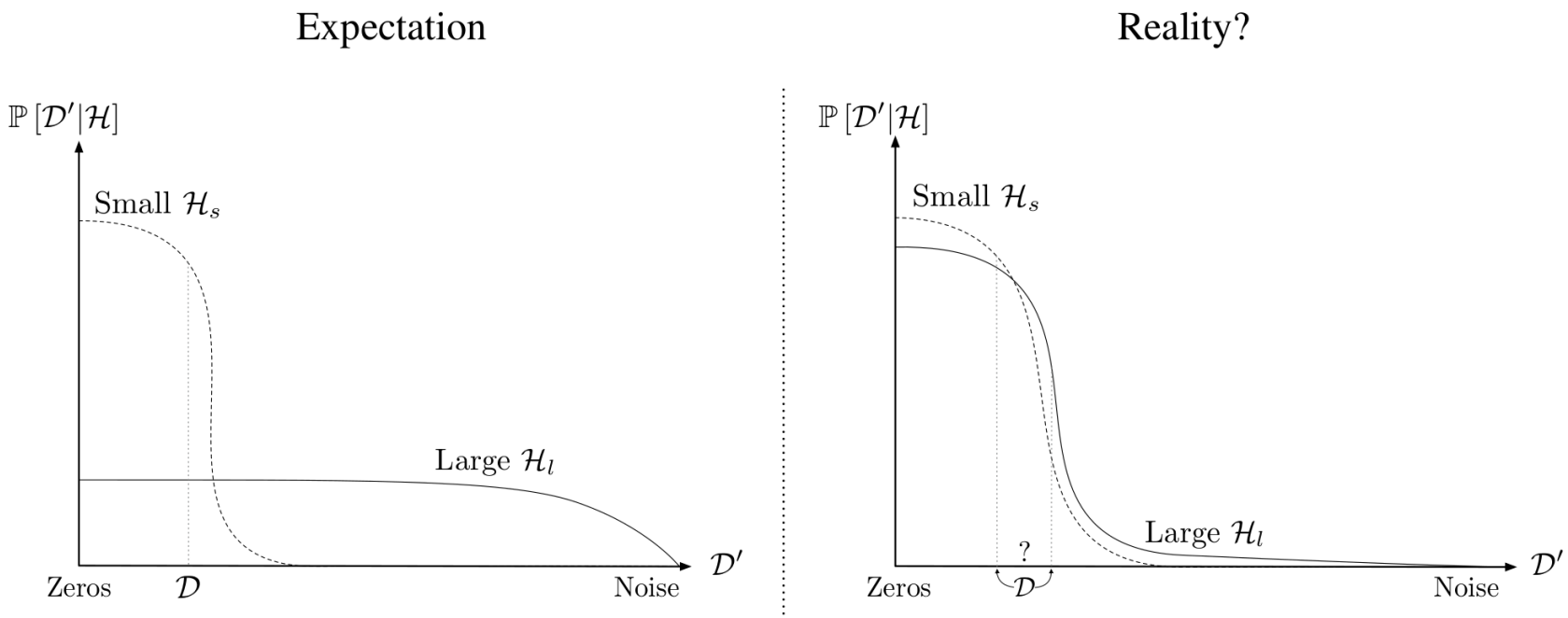
- expressivity studies show exponential gains in representational power for deeper networks
- larger networks can fit arbitrary labellings of data $\Longrightarrow$ more weights may correspond to more usable capacity
- in practice, higher capacity doesn't necessarily imply higher complexity

Occam's Razor and Model Selection

hypothesis

- Stochastic optimization and natural image data may concentrate probability mass on simple functions.
- I.e. available capacity is biased towards "simple" functions.
- If the difference in evidence between two competing models is small, keeping into account input data is crucial.
- Imposing a good prior $\mathbb{P}(H)$ (non-uniform and computable) enables model comparison.

Occam's Razor and Model Selection



Contributions

Large-scale empirical study of fully connected networks.

Main contributions:

- Study the behaviour of trained networks on and off the "data manifold".
- Study local properties (sensitivity) of the learned function, in the vicinity of training data.
- Relate the proposed measures of sensitivity to generalization.

Toy Example

Intuitively, why is input sensitivity important?

Consider, in $[-0.1, 0.1]$, the functions:

- $f(x) = x$
- $g(x) = x^3 \sin x$

In a small neighbourhood of zero, g is constant while f remains linear.

Why Linear Regions?

Goal: find good predictors of generalization.

- Intuition: nonlinearity is key to expressing complex decision functions.
- Evidence in the literature that measures of nonlinearity correlate well with generalization (Jiang et al. 2018. Collins et al. , 2018).
- Number of linear regions related to nonlinear behaviour & expressivity.

Why Linear Regions?

Goal: find good predictors of generalization.

- Intuition: nonlinearity is key to expressing complex decision functions.
- Evidence in the literature that measures of nonlinearity correlate well with generalization (Jiang et al. 2018. Collins et al. , 2018).
- Number of linear regions related to nonlinear behaviour & expressivity.
- Track: study linear regions to find proxy for model complexity.
- Open question: is the number of linear regions a good predictor of generalization ability?

Recap: Linear Regions

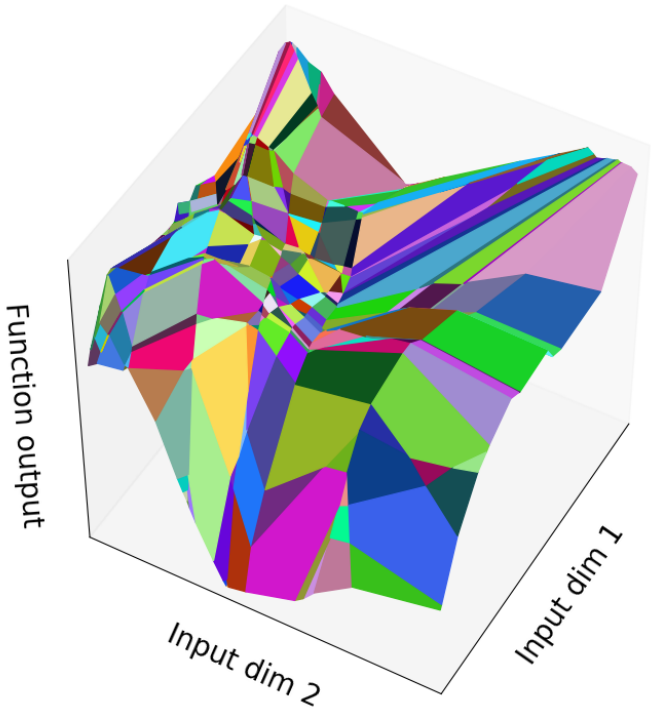
- ReLU nets compute continuous piecewise linear functions
- affine transformations in the input space
- transition between linear “pieces” corresponds to discontinuity in the gradient $\nabla_{\mathbf{x}}\mathbf{f}$
- linear regions: connected components of input space $\setminus$ transition boundaries

Recap: Linear Regions

- ReLU nets compute continuous piecewise linear functions
- affine transformations in the input space
- transition between linear “pieces” corresponds to discontinuity in the gradient $\nabla_{\mathbf{x}}\mathbf{f}$
- linear regions: connected components of input space $\setminus$ transition boundaries
- on each region, the network computes a single affine function
- in general, a linear region is the union of polytopes in the input space
- on each polytope, the activation pattern is constant

Recap: Linear Regions

visualization



(source: Hanin and Rolnick. “Deep ReLU Networks Have Surprisingly Few Activation Patterns”. 2019)

Sensitivity Metrics

Leading questions

Sensitivity measures w.r.t. input for FC networks with no biases + piece-wise linear activation functions.

Leading questions:

1. How does the output change when input perturbed within a linear region?
2. If input perturbed, how likely is it to change linear region?

Sensitivity Metrics

Proposed measures

Let $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^n$ function learned by the network. Let $\mathbf{f}_\sigma = \sigma \circ \mathbf{f}$ network + softmax.

1. Input-output Jacobian norm: $||\mathbf{J}(\mathbf{x})||_F$.
2. Transition count $t(\mathbf{x})$.

Jacobian Norm

definition

For an input $\mathbf{x} \in \mathbb{R}^d$:

- $\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}_\sigma(\mathbf{x})}{\partial \mathbf{x}^T}$
- $\mathbf{J}(\mathbf{x})_{ij} = \frac{\partial \mathbf{f}_\sigma(\mathbf{x})_i}{\partial x_j}$
- recall $||\mathbf{J}||_F := \sqrt{\sum_{ij} J_{ij}^2}$.

Jacobian Norm

definition

For an input $\mathbf{x} \in \mathbb{R}^d$:

- $\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}_\sigma(\mathbf{x})}{\partial \mathbf{x}^T}$
- $\mathbf{J}(\mathbf{x})_{ij} = \frac{\partial \mathbf{f}_\sigma(\mathbf{x})_i}{\partial x_j}$
- recall $||\mathbf{J}||_F := \sqrt{\sum_{ij} J_{ij}^2}$.

Strategy:

- Given data points $\mathbf{x}_{\text{test}}$:
- compute $\mathbb{E}_{\mathbf{x}_{\text{test}}}[||\mathbf{J}(\mathbf{x}_{\text{test}})||_F]$.

Jacobian Norm

interpretation

The Jacobian norm measures the avg sensitivity of $\mathbf{f}_\sigma(\mathbf{x})$ around $\mathbf{x}$.

Let $\Delta\mathbf{x} \sim \mathcal{N}(0, \epsilon\mathbf{I})$.

$\mathbb{E}_{\Delta\mathbf{x}}[||\mathbf{f}_\sigma(\mathbf{x}) - \mathbf{f}_\sigma(\mathbf{x} + \Delta\mathbf{x})||_2^2]$ (finite difference approx.)

$\approx \mathbb{E}_{\Delta\mathbf{x}}[||\mathbf{J}(\mathbf{x})\Delta\mathbf{x}||_2^2] = \mathbb{E}_{\Delta\mathbf{x}}[\sum_i (\sum_j J_{ij} x_j)^2]$

$= \sum_{ijj'} J_{ij} J_{ij'} \mathbb{E}_{\Delta\mathbf{x}}[x_j x_{j'}]$

$= \sum_{ij} J_{ij}^2 \mathbb{E}_{\Delta\mathbf{x}}[x_j^2]$

$= \epsilon||\mathbf{J}(\mathbf{x})||_F^2.$

Transition Count

Goal: starting from a sample $\mathbf{x}$, move along a trajectory in the input space and count the number of linear regions crossed by the trajectory.

- How to identify linear regions?

Transition Count

Goal: starting from a sample $\mathbf{x}$, move along a trajectory in the input space and count the number of linear regions crossed by the trajectory.

- How to identify linear regions?
- Strategy: use activation patterns
- Generally, activation regions fall in different linear regions*
- (*corner cases exist)

Transition Count

Linear regions

Recall:

- each data point $\mathbf{x}$ lies in one linear region
- on such region the network computes a single linear function
- the linear region is defined as the preimage of the activation $\mathbf{f}(\mathbf{x})$ of the network, for each neuron in the network

Transition Count

activity patterns

Idea:

- Identify each linear region by the corresponding activity pattern.
- If activity pattern is used, different codes identify different regions* (by def. of preimage)
- *corner cases are not considered in the paper

Transition Count

code

Code $\mathbf{x} \mapsto \mathbf{c}(\mathbf{x})$:

- concatenation of the values of activity patterns for all neurons in the network
- e.g. for ReLU, $\mathbf{c}(\mathbf{x}) \in \{0,1\}^N$, $N = \#$ neurons in the network

Transitions computed by detecting changes in the code $\mathbf{c}(\mathbf{x})$.

Transition Count

sampling

Consider a one-dimensional trajectory $\mathcal{T}(\mathbf{x})$ in the input space.

- sample k equidistant points $\mathbf{z}_0, \ldots, \mathbf{z}_{k-1}$ along $\mathcal{T}(\mathbf{x})$.
- count the transitions along the trajectory:

$$t(\mathbf{x}) := \sum_{i=0}^{k-1} ||\mathbf{c}(\mathbf{z}_i) - \mathbf{c}(\mathbf{z}_{(i+1)\%k})||_1$$

- $\approx \oint_{\mathbf{z} \in \mathcal{T}(\mathbf{x})} \left|\left| \frac{\partial \mathbf{c}(\mathbf{z})}{\partial (d\mathbf{z})} \right|\right|_1 d\mathbf{z}$

Estimate $\mathbb{E}_{\mathbf{x}_{\text{test}}}[t(\mathbf{x}_{\text{test}})]$.

Transition Count

interpretation

Given a function $\mathbf{f}$, its curvature along a path $\mathcal{T}(\mathbf{x})$ is:

- $C(\mathbf{f}, \mathcal{T}(\mathbf{x})) = \oint_{\mathbf{z} \in \mathcal{T}(\mathbf{x})} \left|\left| \frac{\partial \mathbf{f}'(\mathbf{z})}{\partial(d\mathbf{z})} d\mathbf{z} \right|\right|_F$.
- piecewise linear function has constant first derivative everywhere apart from transition boundaries
- for $k \gg 1$, $\quad C(\mathbf{f}, \mathcal{T}(\mathbf{x})) = \frac{1}{2} \sum_{i=0}^{k-1} ||\mathbf{f}'(\mathbf{z}_i) - \mathbf{f}'(\mathbf{z}_{(i+1)\%k})||_F$.
- $\mathbf{z}_0, \ldots, \mathbf{z}_{k-1}$ equidistant samples on $\mathcal{T}(\mathbf{x})$

Observations

Qualitatively, if we consider Taylor expansion of **f** centered at **x**:

- Jacobian norm encodes first-order information of **f**
- Transition count encodes second-order information **f**

Experiments

- Analyze a large number of FC networks with different hyper-parameters and optimization algorithms.
- 8 seeds per configuration, each trained until 100% training accuracy or discarded.

Experiments

- Analyze a large number of FC networks with different hyper-parameters and optimization algorithms.
- 8 seeds per configuration, each trained until 100% training accuracy or discarded.

Empirical study:

1. Study sensitivity metrics on and off the training data manifold.
2. Sensitivity metrics and model selection (factors of generalization).
3. Sensitivity metrics as predictors of the generalization gap.
4. Sensitivity metrics and per-point generalization.

1. Sensitivity on and off the Data Manifold

strategy

Strategy: measure the sensitivity metrics along circular trajectories intersecting the data “manifold” at certain points.

1. Sensitivity on and off the Data Manifold

strategy

Strategy: measure the sensitivity metrics along circular trajectories intersecting the data “manifold” at certain points.

1. Random ellipse.
   - Unlikely to pass from training data.
   - Sensitivity at points not seen during training.

1. Sensitivity on and off the Data Manifold

strategy

Strategy: measure the sensitivity metrics along circular trajectories intersecting the data “manifold” at certain points.

1. Random ellipse.
   - Unlikely to pass from training data.
   - Sensitivity at points not seen during training.
2. Ellipse through training points of different classes.
   - Include linear combination of points of different classes.
   - Study sensitivity on and off the data manifold.

1. Sensitivity on and off the Data Manifold

strategy

Strategy: measure the sensitivity metrics along circular trajectories intersecting the data “manifold” at certain points.

1. Random ellipse.
   - Unlikely to pass from training data.
   - Sensitivity at points not seen during training.
2. Ellipse through training points of different classes.
   - Include linear combination of points of different classes.
   - Study sensitivity on and off the data manifold.
3. Ellipse through points of the same class.
   - Include linear combinations of points of the same class.
   - Overall closer to data manifold.

Trajectories and Data “Manifold”

Trajectories close to the “data manifold” are generated by:

- Augmenting training data with horizontal and vertical translations.
- For a test sample $\mathbf{x}_{\text{test}}$, $\mathcal{T}(\mathbf{x}_{\text{test}})$ is constructed through horizontal translations of $\mathbf{x}_{\text{test}}$ in pixel space.

Trajectories and Data “Manifold”

example

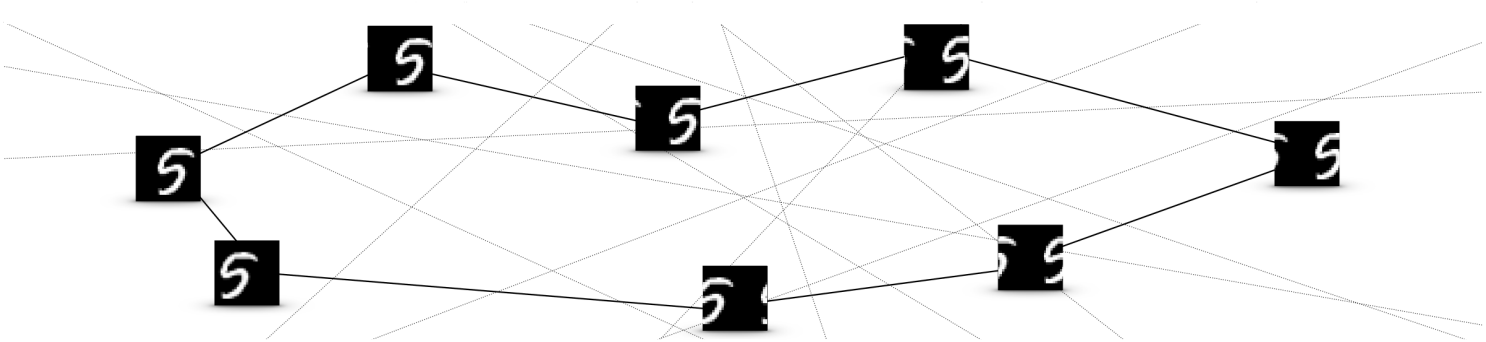
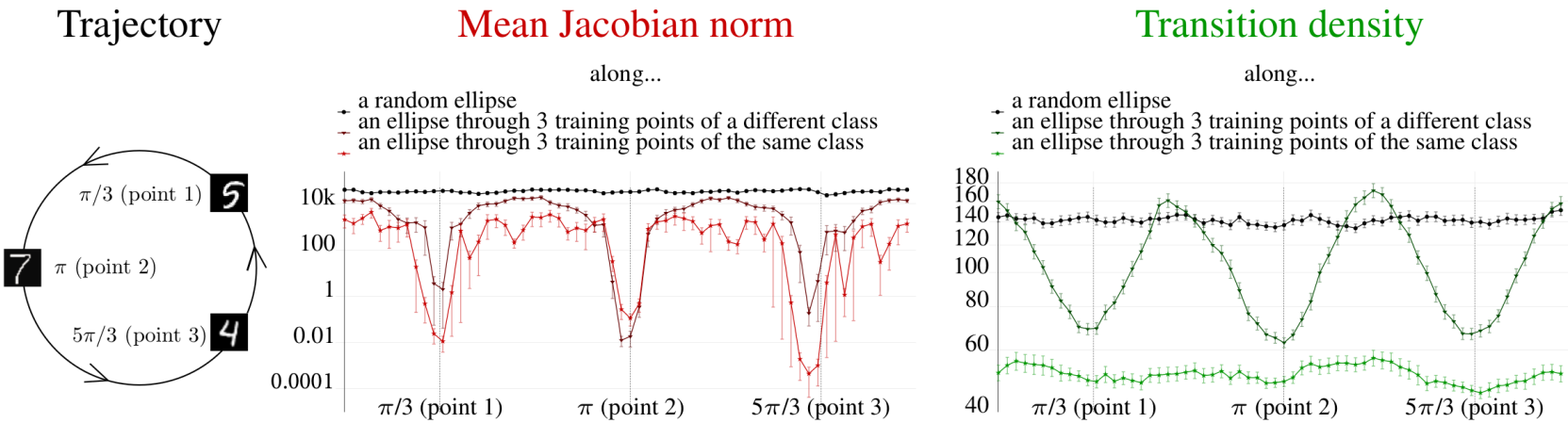


Figure: Interpolation between 28 horizontal translations of $\mathbf{x}_{\text{test}}$. All points lie close to the translation-augmented data. $k = 2^{20}$ points sampled per trajectory.

1. Sensitivity on and off the Data Manifold

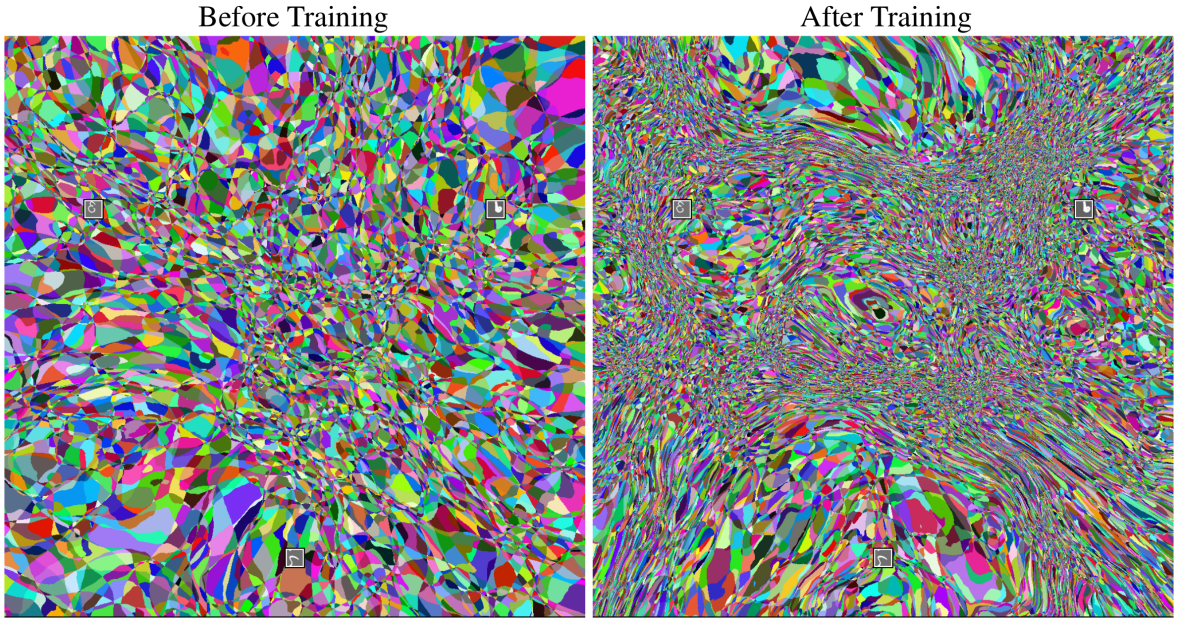
results



- Both metrics show increased robustness near training points.
- Measures of generalization should consider input data as well.

Transition Boundaries

visualization



- 2D slice in MNIST pixel space.
- 15-layer network of width 300, trained for 2^{18} epochs.
- SGD with momentum and data augmentation (flips).

2. Sensitivity and Generalization Factors

Study relationship between sensitivity and generalization factors, for networks trained to 100% training accuracy.

Factors considered:

- Random labels.
- Data augmentation.
- Activation functions (ReLU vs HardSigmoid)
- minibatch SGD+momentum vs full-batch training.

Dataset used: CIFAR-10.

2. Sensitivity and Generalization Factors

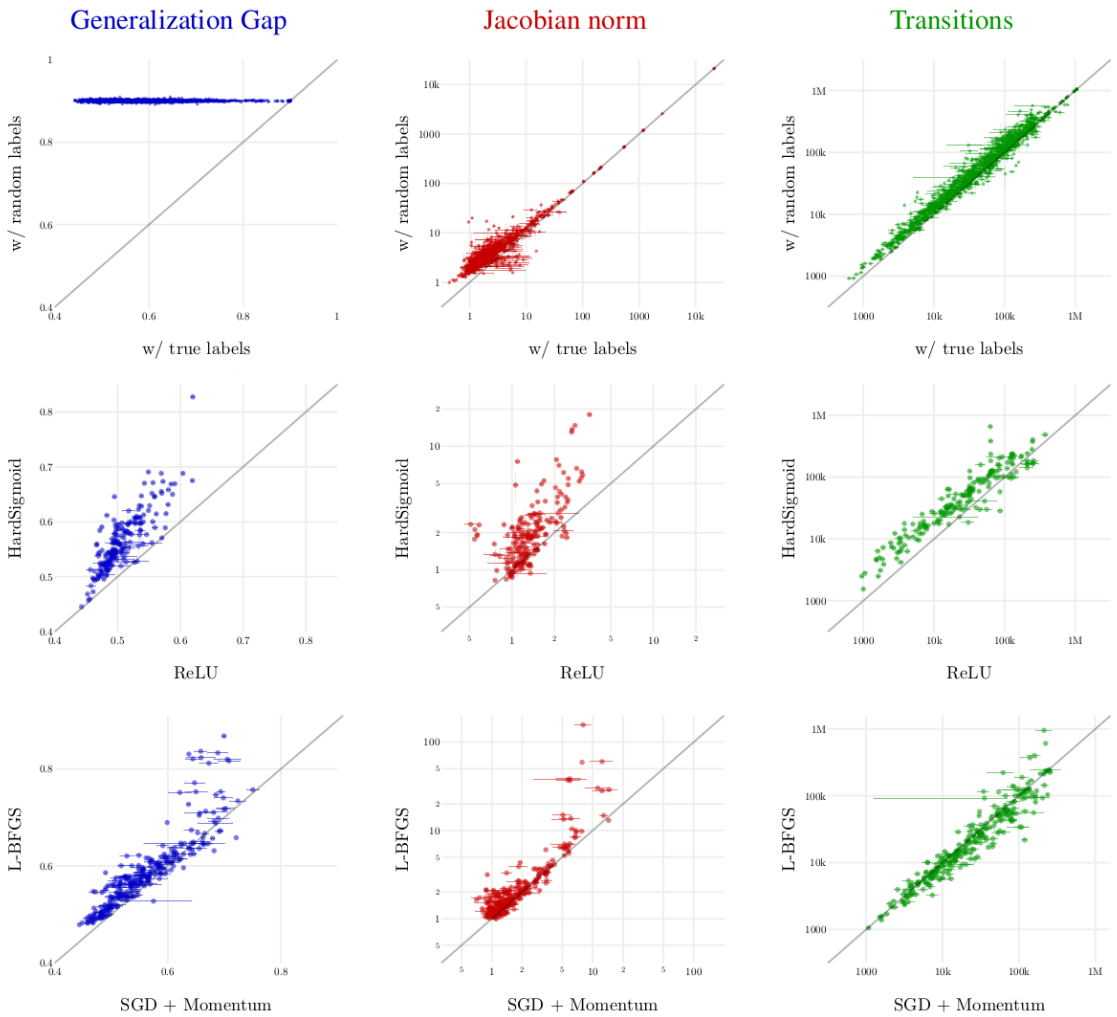
experimental details

335671 networks trained for 2^{19} steps with random hyper-parameters, including:

- SGD, Momentum, Adam, RMSProp, LBFGS.
- Learning rates (0.01, 0.005, 0.0005) & batch size (128, 512).
- Activation functions (ReLU, ReLU6, Tanh, HardSigmoid, HardTanh).
- Widths $(1, 2, 4, \ldots 2^{16})$.
- Depths $(2, 3, 5, \ldots, 2^6 + 1)$.
- True and random labels.

2. Sensitivity and Generalization Factors

results



Generalization gap := training accuracy - test accuracy

2. Sensitivity and Generalization Factors

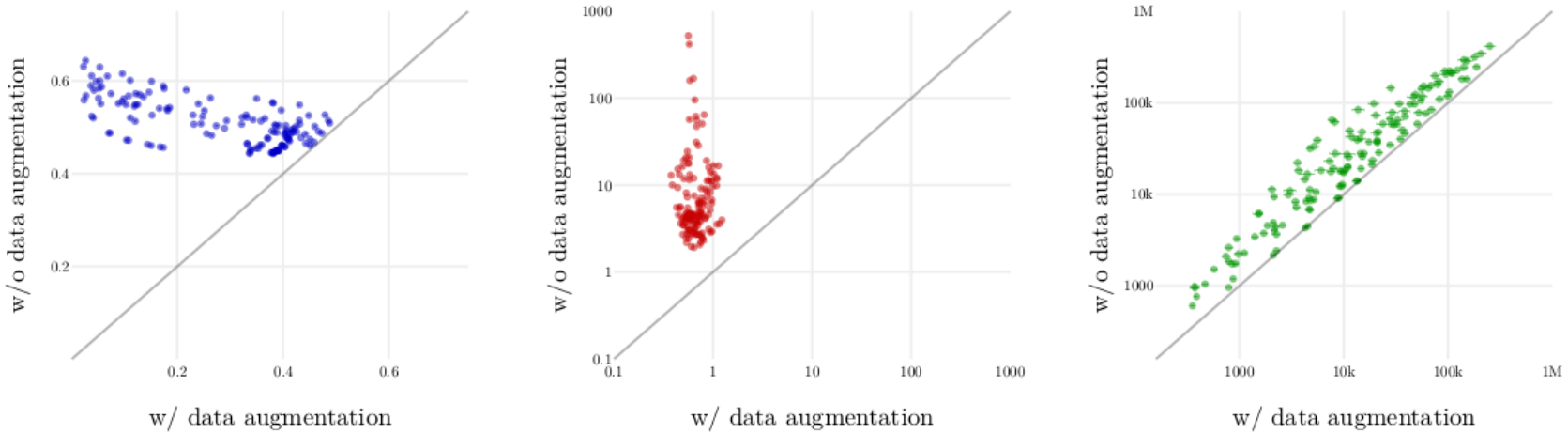
data augmentation

Experimental details:

- CIFAR-10
- SGD + momentum for 2^{18} steps, learning rate 0.005
- Width of 100, 200, 500, 1000, 2000, 3000
- depth of 2, 3, 5, 10, 15, 20
- ReLU, ReLU6, HardTanh, HardSigmoid
- Random translation of input by 4 px; flipping.
- Keep models for which data augmentation resulted in higher test accuracy than the same model without.

2. Sensitivity and Generalization Factors

results



Generalization gap := training accuracy - test accuracy

3. Sensitivity and Generalization Gap

Establish direct relationship between sensitivity and generalization.

- Consider all architectural choices and HPs simultaneously.
- CIFAR-10, CIFAR-100, MINIST, Fashion MNIST.

3. Sensitivity and Generalization Gap

experimental details

335671 networks trained for 2^{19} steps with random hyper-parameters, including:

- SGD, Momentum, Adam, RMSProp, LBFGS.
- Learning rates (0.01, 0.005, 0.0005) & batch size (128, 512).
- Activation functions (ReLU, ReLU6, Tanh, HardSigmoid, HardTanh).
- Widths $(1, 2, 4, \ldots 2^{16})$.
- Depths $(2, 3, 5, \ldots, 2^6 + 1)$.
- True and random labels.

3. Sensitivity and Generalization Gap

results

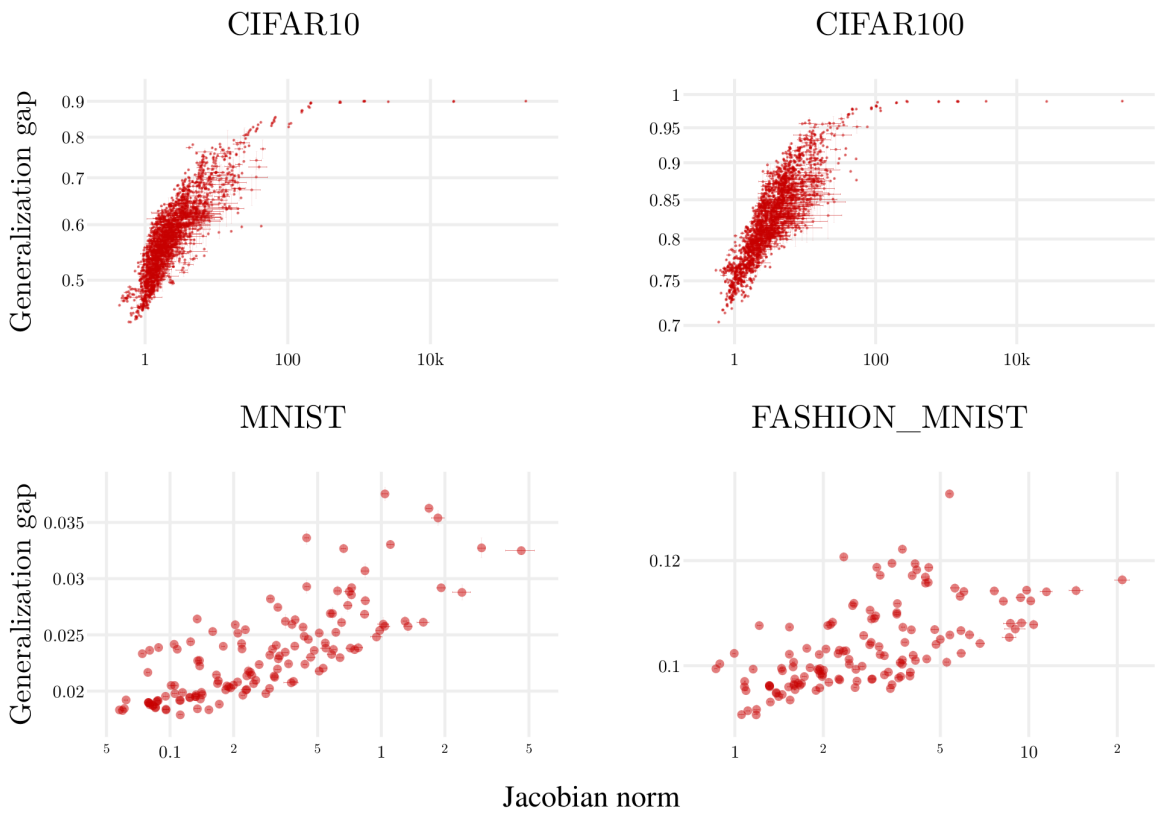


Figure: Jacobian norm correlates with generalization gap.

3. Sensitivity and Generalization Gap

results

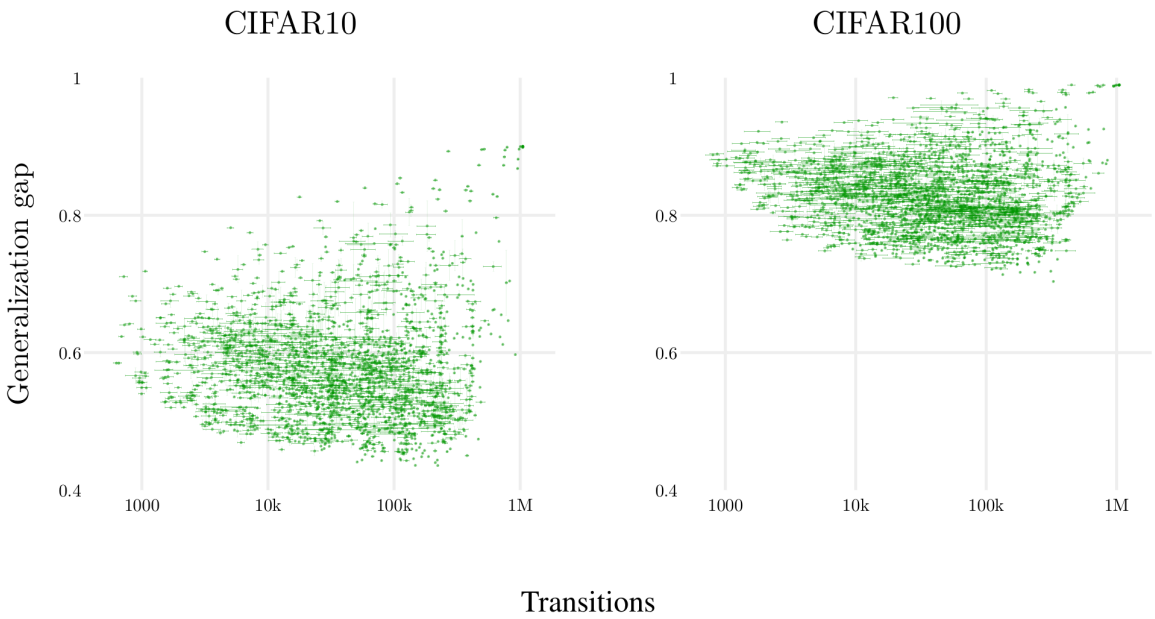


Figure: Transition count does not correlate with generalization gap.

4. Sensitivity and Per-Point Generalization

Is the Jacobian norm predictive of generalization at individual test points?

- Study relationship between Jacobian norm and test loss for 1000 test points.
- Consider 5 networks trained to 100% train accuracy.

4. Sensitivity and Per-Point Generalization

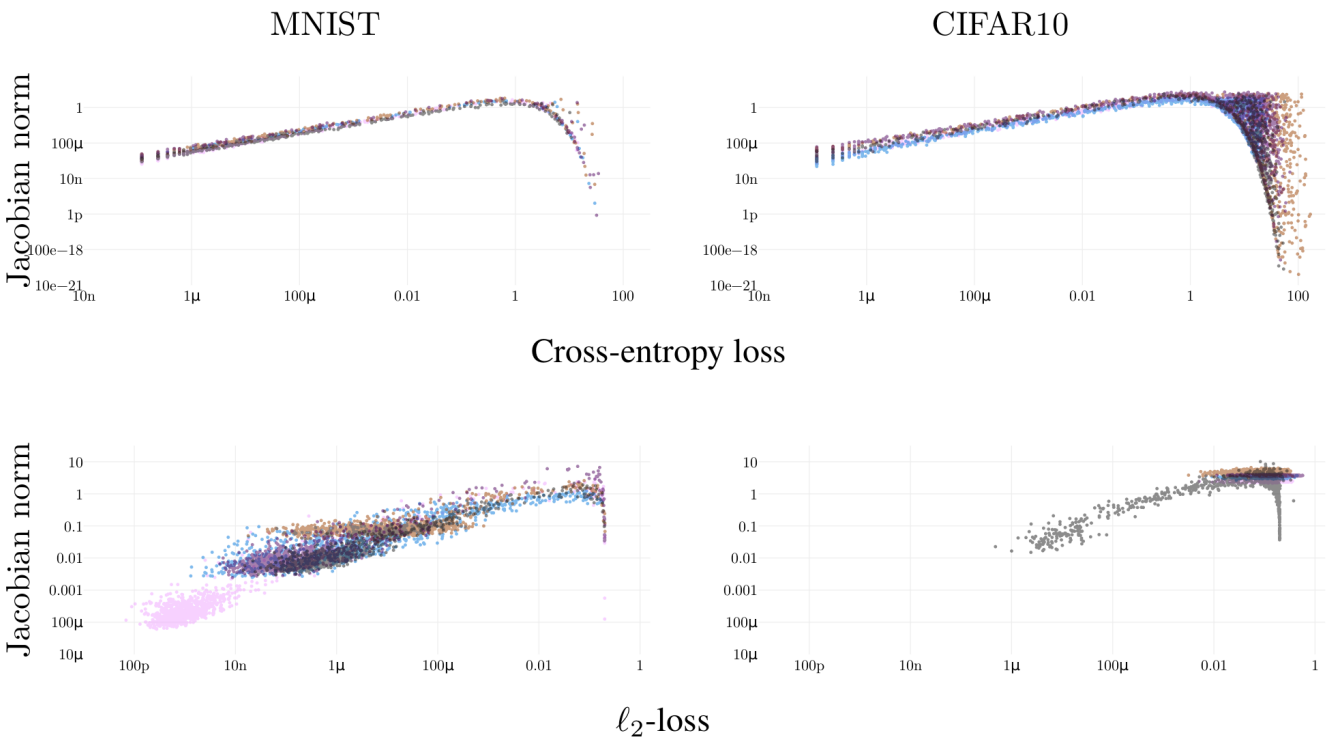
experimental setup

5 random networks trained with 100% training accuracy.

- Trained for 2^{19} steps.
- Evaluated on 1000 test images.
- ReLU, ReLU6, Tanh, HardSigmoid, HardTanh.
- SGD, Momentum, ADAM, RMSProp.
- Widths 50, 100, 200, 500, 1000.
- Depths 2, 5, 10, 20, 30.
- Learning rate 0.0001, 0.001, 0.01.
- One seed.

4. Sensitivity and Per-Point Generalization

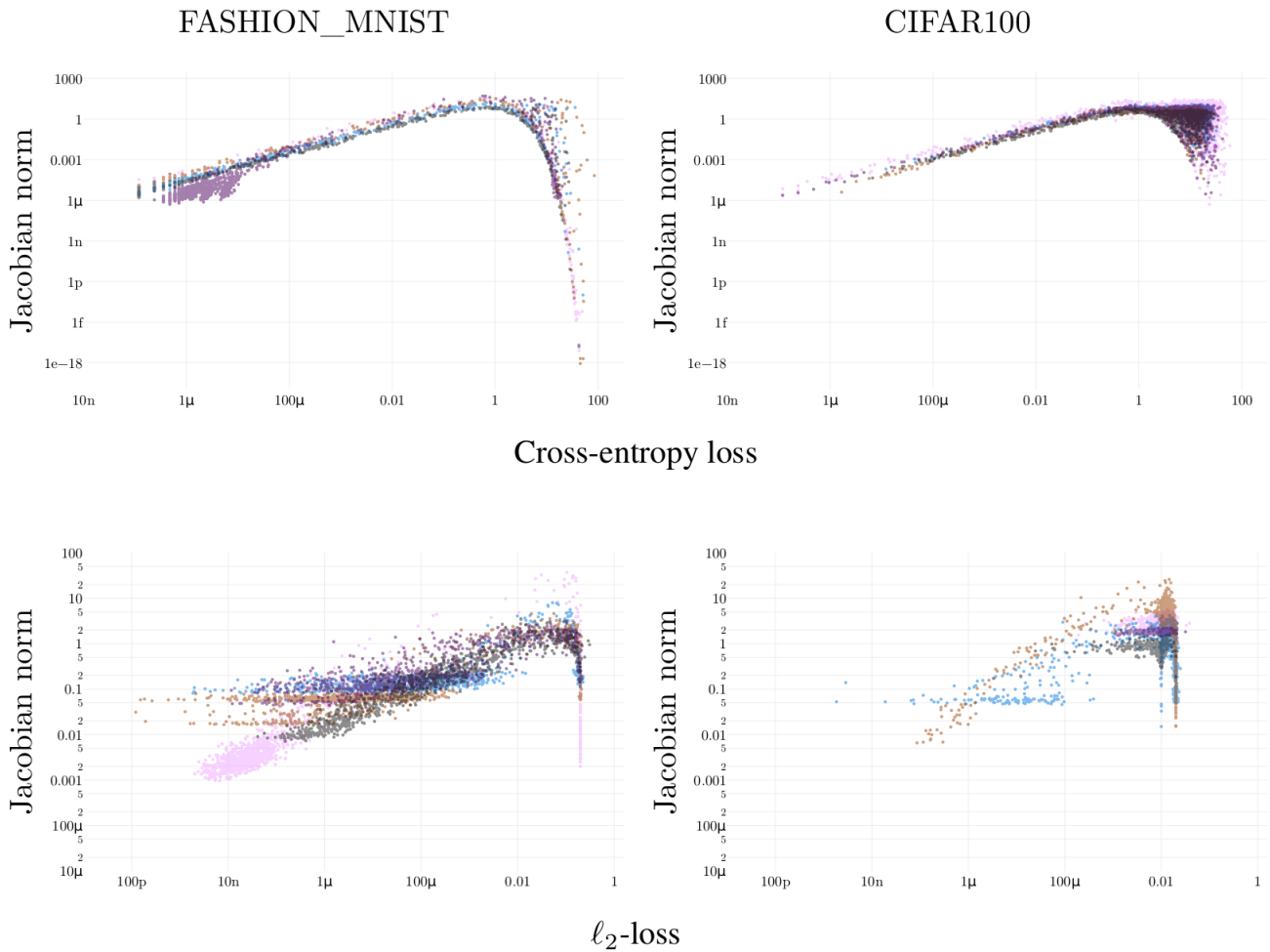
results



- Points with high Jacobian norm are mostly misclassified.
- Some misclassified points have small Jacobian norm.

4. Sensitivity and Per-Point Generalization

results



Cross-Entropy Loss and Jacobian Norm

- Jacobian norm considered for $\mathbf{f}_\sigma = \sigma \circ \mathbf{f}$.
- Jacobian norm at the logits level did not perform well (not reported in the paper).
- Authors derive analytic bounds for the Jacobian norm in terms of the cross-entropy loss.

Cross-Entropy Loss and Jacobian Norm

analytic bounds

Given a labeled test sample $(\mathbf{x}, y(\mathbf{x}))$, express the relationship between $\mathbf{J}_{y(\mathbf{x})}$ and the cross-entropy loss $\ell = -\log\left[\mathbf{f}_\sigma(\mathbf{x})\right]_{y(\mathbf{x})}$ as

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

with:

- $M = \mathbb{E}_{\mathbf{x}_{\text{test}}} || \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{\text{test}}^T} ||_F^2$.
- $n =$ # classes.

Let $\ell = -\log \sigma_y$, then:

$$\sqrt{\frac{nM}{n-1}} \mathrm{e}^{-\ell}(1-\mathrm{e}^{-\ell}) < ||\mathbf{J}_y||_2 < \sqrt{2M}\mathrm{e}^{-\ell}(1-\mathrm{e}^{-\ell})$$

Cross-Entropy Loss and Jacobian Norm

analytic bounds

When the target class y is unknown:

- the lower bound still holds.
- Assuming maximum entropy case for $\sigma_y : \sigma_i \approx (1 - \sigma_y)/(n-1)$, for $i \neq y$.
- Lower bound:

$$\sqrt{\frac{nM}{n-1}} e^{-\ell}(1 - e^{-\ell}) < ||\mathbf{J}_j||_2 \leq ||\mathbf{J}||_F$$

- Norm approximation:

$$||\mathbf{J}||_F \approx \frac{\sqrt{M}}{(n-1)}(1 - e^{-\ell})\sqrt{n^2 e^{-2\ell} + n - 2}$$

Cross-Entropy Loss and Jacobian Norm

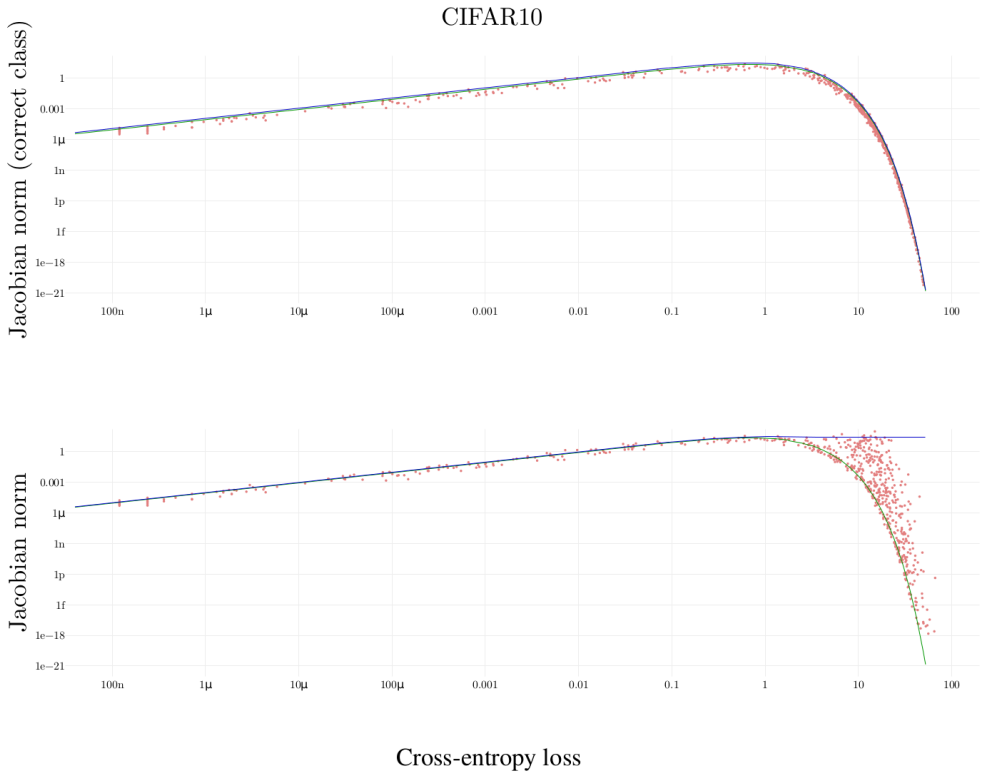
experimental details

Single random trained network with 100% accuracy on CIFAR-10, sampled from networks trained.

- for 2^{18} steps.
- evaluated on 1000 test points.
- ReLU, ReLU6, Tanh, HardSigmoid, HardTanh.
- Widths 50, 100, 200.
- Depths 2, 5, 10, 20.
- One seed.

Cross-Entropy Loss and Jacobian Norm

results



- Lines (top) show analytic bounds.
- Lines (bottom) lower bound and norm approximation.

Conclusions

Empirical study of two sensitivity measures in the vicinity of input data.

- FC networks seem biased towards functions that are robust in proximity of training points.
- For FC networks, the local geometry around input data is predictive of generalization.
- The behaviour of the learned function drastically changes away from training data.

Conclusions

Empirical study of two sensitivity measures in the vicinity of input data.

- FC networks seem biased towards functions that are robust in proximity of training points.
- For FC networks, the local geometry around input data is predictive of generalization.
- The behaviour of the learned function drastically changes away from training data.
- Number of linear regions does not clearly correlate with generalization.

Conclusions

Empirical study of two sensitivity measures in the vicinity of input data.

- FC networks seem biased towards functions that are robust in proximity of training points.
- For FC networks, the local geometry around input data is predictive of generalization.
- The behaviour of the learned function drastically changes away from training data.
- Number of linear regions does not clearly correlate with generalization.
- Left unexplained: why do large networks converge to more robust functions?
- What is the role of optimization (implicit bias of SGD)?

Further Work

Complexity of Linear Regions in Deep Networks

Hanin and Rolnick, ICML19.

- for a FC network with output dimension 1
- weights at initialization
- expected density of transition boundaries between linear regions is upper bounded by $\binom{\#\text{ neurons}}{k} C^k$
- C constant, $k \in \{1, \ldots, d\}$.
- irrespective of depth

Further Work

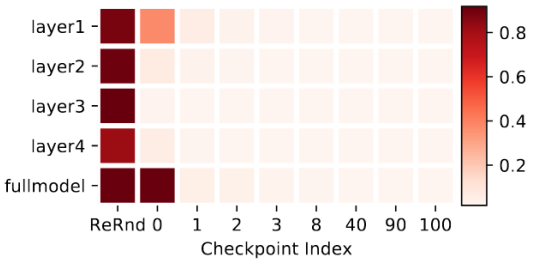
Deep ReLU Networks Have Surprisingly Few Activation Patterns

Hanin and Rolnick, NeurIPS19.

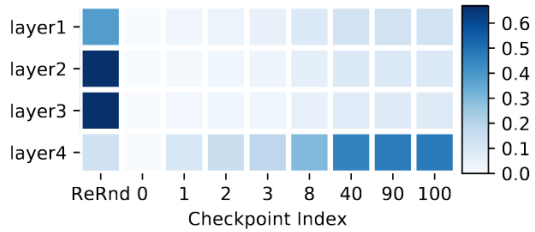
- for a FC network at initialization with no tied weights
- further hypothesis in distribution of gradients and weights
- expected local density of activation regions upper bounded by $(T \,\#\, \text{neurons})^d / d!$
- irrespective of depth

Further Work

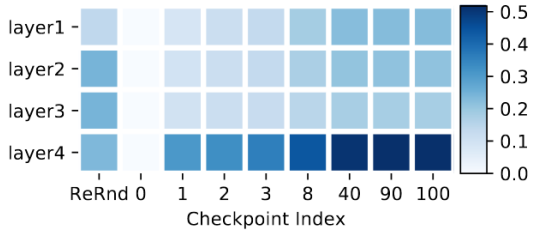
Are All Layers Created Equal?



(a) Test error



(b) $\|\theta_d^\tau - \theta_d^0\|$



(c) $\|\theta_d^\tau - \theta_d^0\|_\infty$

- FC network trained on MNIST
- At each checkpoint (x-axis), the corresponding layer (y-axis) is reinitialized to its value before training.
- How are the sensitivity measures affected?

Are All Layers Created Equal? Zhang, Bengio and Singer. ICLM19, Workshops Track.

Distinguishing Linear Regions

Activation regions typically fall in different linear regions.

Edge cases[1]:

- Formally, # linear regions $\leq$ # activity regions
- In fact, multiple neighboring activity regions can collapse to a single linear region if they are all zeroed-out by a layer
- the set of weights for which this happens is a zero-measure set
- activity regions typically fall on different linear regions
- on average # linear regions $\approx$ # activity regions

[1] (from Rolnick and Hanin, NeurIPS 19)