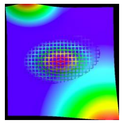


Information geometry and generalization in deep neural networks

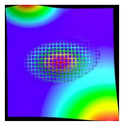
Bálint Daróczy

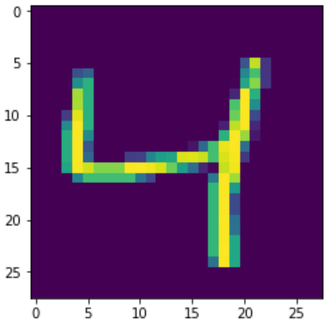
SZTAKI

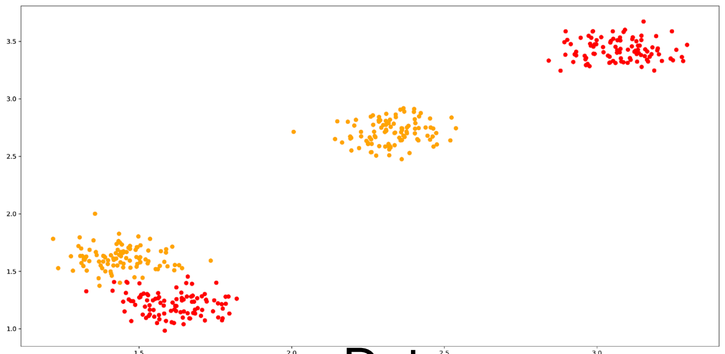
& UCLouvain

21.10.2020

Motivation: manifolds and point clouds in ML





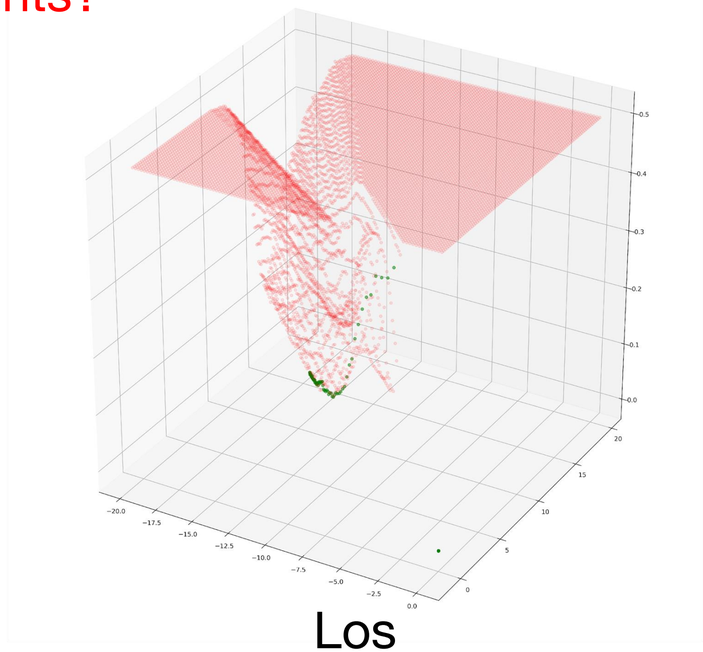


Dat
a

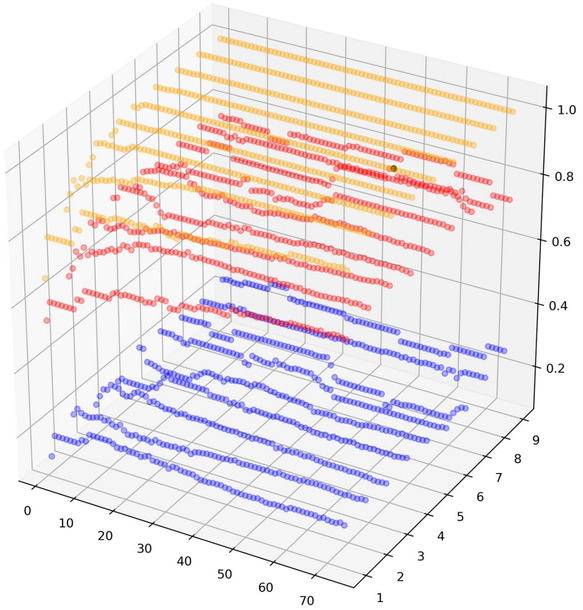
Given data $X=\{(x_1,y_1),..,(x_T,y_T)\}$,

Our goal is to find a hypothesis h(x), which approximate y accurately over X.

Manifolds or just sets of independent points?

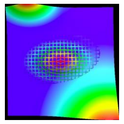


Los
s

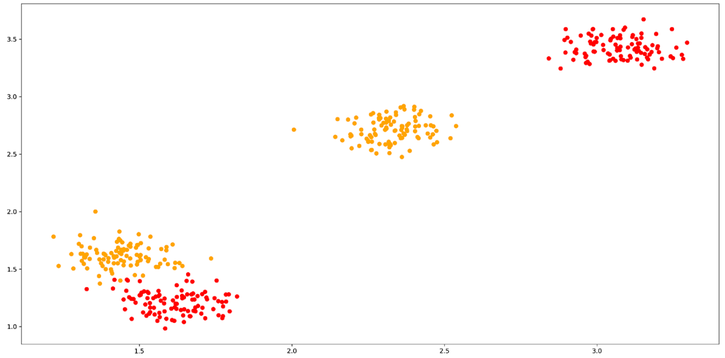


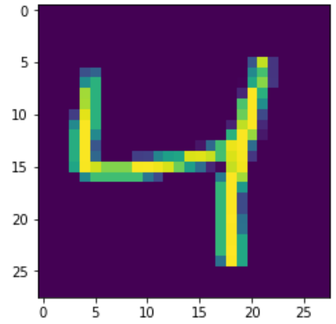
Hyperparameter surface

Manifolds or point clouds?



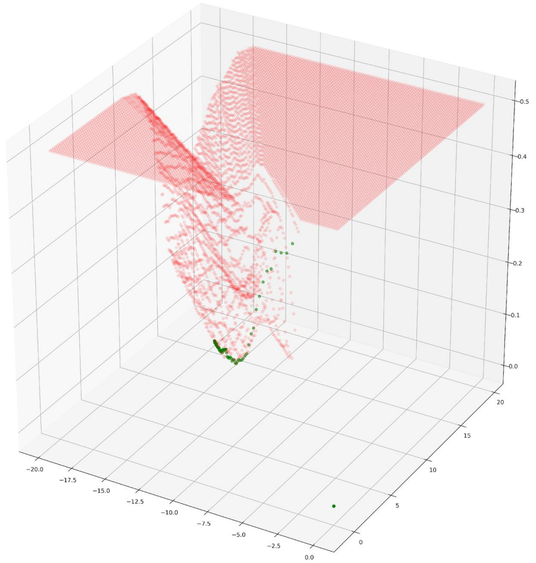
Data





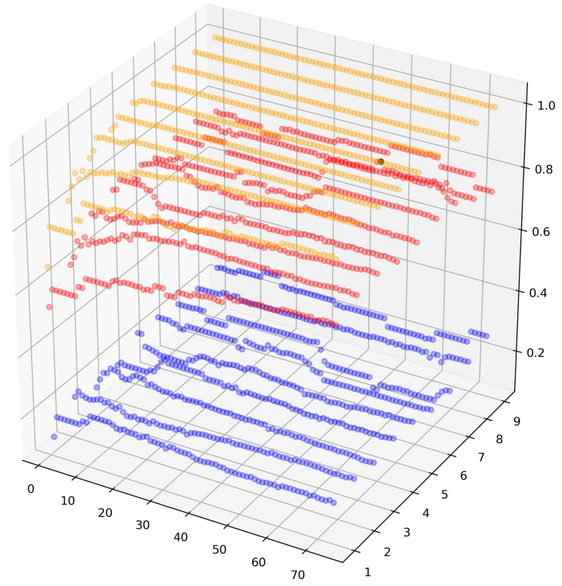
usually NOT a topological manifold

Loss



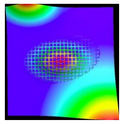
usually a differentiable manifold
[Ollivier et al, 2015,
Choromanska et al., 2015]
feed-forward NNs define
statistical manifolds
[Cencov, 1982,
Campbell, 1986, Amari, 1996]

Hyperparameter surface

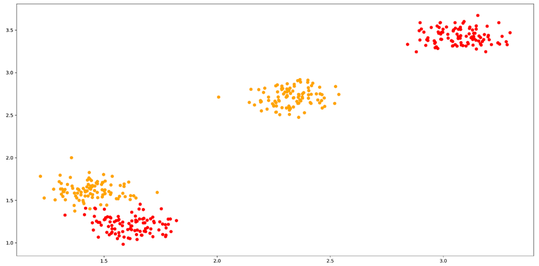


usually a set of topological manifolds but not differentiable

Manifolds or point clouds?



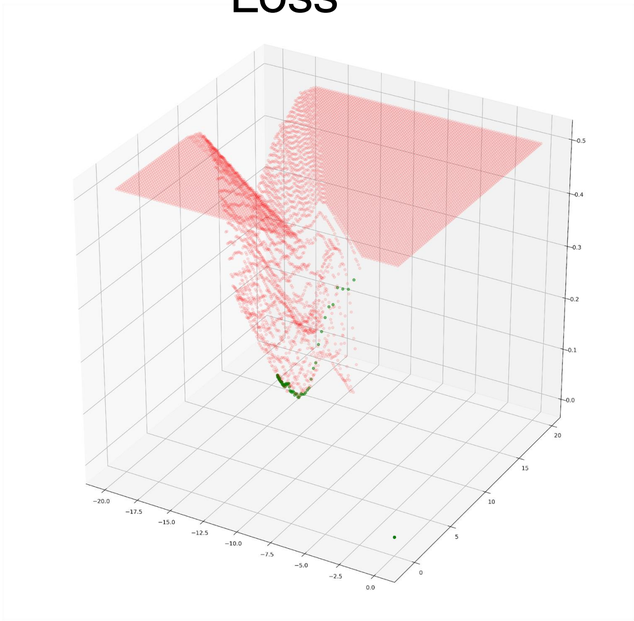
Data



usually NOT a topological manifold

Augmentation
[Khrizhevsky et al., 2012]

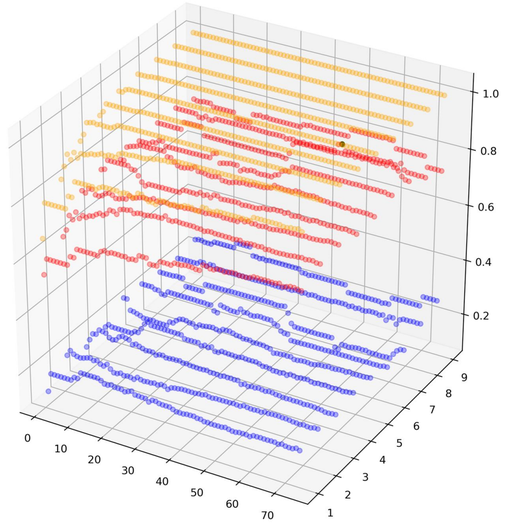
Loss



usually a differentiable manifold

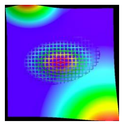
Regularization
and Dropout [Hinton et al., 2012]

Hyperparameter surface



usually a set of topological manifolds, but not differentiable

Network structure



Generalization?

Given data $X=\{(x_1,y_1),..,(x_T,y_T)\}$,

Our goal is to find a hypothesis, h(x) which approximate y over X.

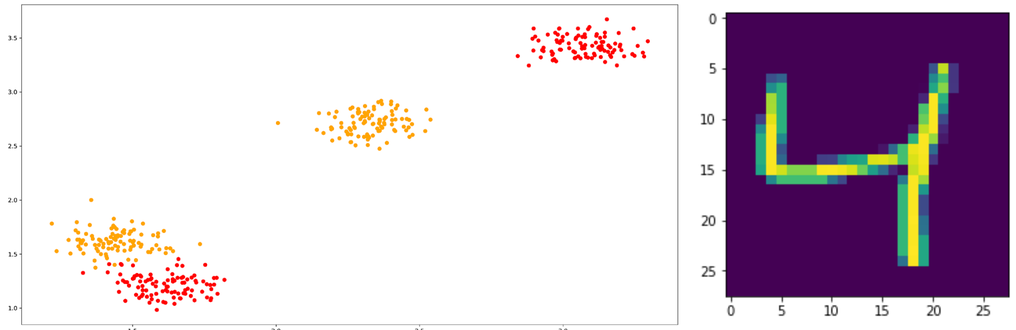
1. How to measure the performance of the approximation?

2. How to generalize? Difference between the true loss and the empirical loss?

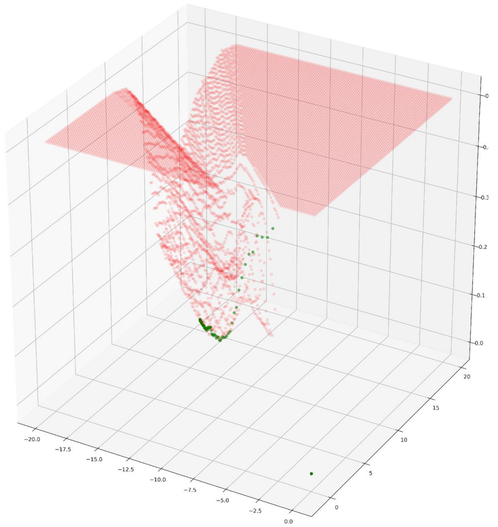
3. How to choose the function class?
e.g. linear separators, NN, etc.

4. How to find a particular element in the function class?
e.g. “random walk on Loss”

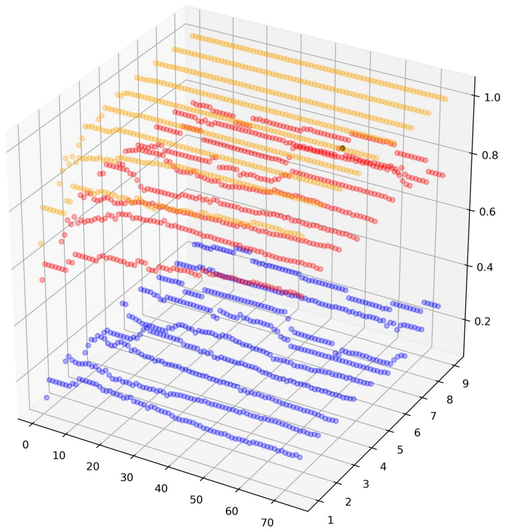
Today we will focus on 2 for NNs, with relation to manifolds.



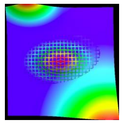
Data



Loss

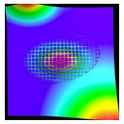


Hyperparameter surface



Outline

1. Generalization: Vapnik-Chervonenkis theorem and deep neural networks
2. Information geometry of neural networks: Fisher-Rao norm and K-FAC approximation of Fisher information
3. Complexity of ReLU networks: evaluation of linear regions and tangent space sensitivity



Generalization

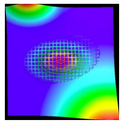
Three related approaches:

1. Capacity: worst/best case scenarios of a function class, e.g. VC-dim [V&C,1971, Bartlett, 2003, Maas, 1993 etc.]

> Recently realized NN issue [Nagarajan et al., 2019]: uniform convergence may be unable to explain generalization in deep NN

1. Stability: robustness of learning algorithms, e.g. algorithmic and uniform argument stability, loss stability [Bartlett et al., 2003, Liu et al., 2017]

1. Local sensitivity: robustness of the already visited solutions, e.g. flatness [Hochreiter, 1997, Neyshabur et al. 2018, Dinh et al., 2018, Novak et al. 2018]



Capacity: background, the VC dimension

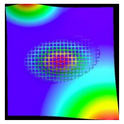
Vapnik-Chervonenkis theorem: connection between generalization, training set selection, and model selection

Empirical risk:

$$R_{emp}(f) = \frac{1}{T}\sum_{t=1}^{T} l(f(x_i), y_i))$$

where X = $\{(x_i,y_i)\}$ has cardinality T.

Theorem (informal): if we optimize for a binary loss function (0 if $f(x_i) = y_i$ and 1 if not) over a set of independent samples from a fixed distribution D with known labels (the training set), then the true risk $R_{true}(f)$ (the expected value of the loss function over D) is upper bounded by the empirical risk plus an additional value depending on function class capabilities.



Capacity

The VC-theorem [Vapnik and Chervonenkis, 1971]: the worst case scenario

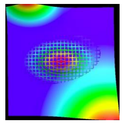
For binary classification with a binary loss function and function class F, the generalization (the difference between the true and the empirical risk) is bounded as follows (here we assume uniform convergence of relative frequencies)

$$P(\sup_{f \in \mathcal{F}} \mid R_{emp}(f) - R_{true}(f) \mid > \epsilon) \leq 8\mathcal{S}(\mathcal{F}, T)\mathrm{e}^{-\frac{T\epsilon^2}{32}}$$

and

$$\mathbf{E}[\sup_{f \in \mathcal{F}} \mid R_{emp}(f) - R_{true}(f) \mid] \leq 2\sqrt{\frac{\log \mathcal{S}(\mathcal{F}, T) + \log 2}{T}}$$

Optimization for low true risk is a balance between low empirical risk and low VC-dimension.



VC dimension and feed-forward Neural Networks

VC-dimension (VCdim) of linear separator is d+1
VC-dimension of polynomial separator ... is infinite (with sufficiently high degree, poly kernel SVM)

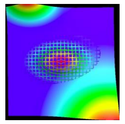
Arbitrary feed-forward neural network [Cover, 1968, Baum & Haussler, 1989, Maas, 1993, Sakurai, 1993] with linear threshold, piecewise linear or sigmoidal activation functions and parameters w:

- with fixed depth VCdim = O(w log w)
- if the depth is unbounded the VCdim is $O(w^2)$

There exists a feed-forward network with infinite VCdim: a special activation function and the network has only a single hidden layer [Sontag, 1992].

Uniform convergence bounds are a bad choice for complex classifiers because these hypotheses classes have infinite VC-dimension or the bound is meaningless.

Idea [Nagarajan & Kolter, 2019]: what if we select a meaningful subset of the hypothesis class?



Uniform convergence in case of NN [Nagarajan&Kolter, 2019]

V. Nagarajan & J.Z. Kolter: Uniform convergence may be unable to explain generalization in deep learning, NeurIPS 2019

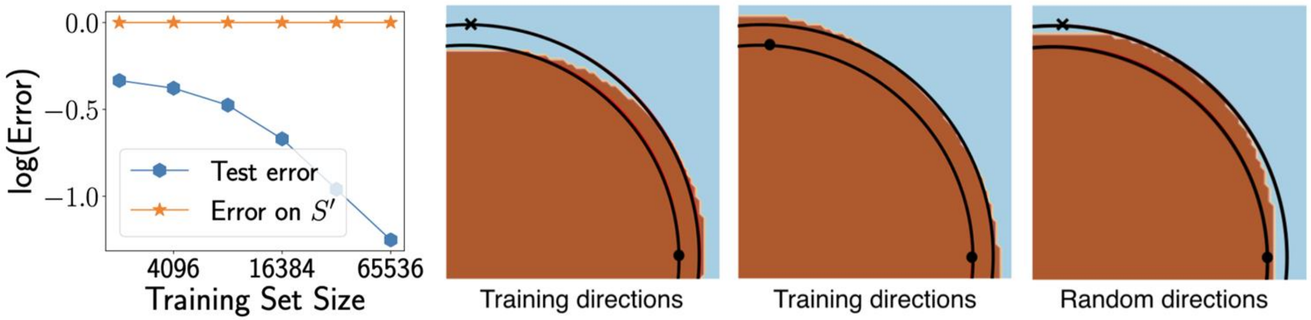
Even these hypothesis sets (in case of overparametrized networks and GD) result useless bounds…

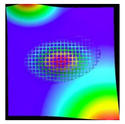
They show:

1. Generalization gap increases if the training set is getting larger
2. Hypothesis: learned boundary of overparametrized networks is too complex

Idea: They construct a “bad” data set (S’) which is completely missclassified and similarly sized as the training set -> low test error and low training error does not indicate low generalization gap -> uniform convergence fails (Q: polynomial separation?)

In the example on the right they pick S′ by simply projecting every training datapoint on the inner hypersphere onto the outer and vice versa, and then flipping the labels (to the correct one).





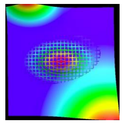
Generalization

Three not so independent approaches:

1. Capacity: worst/best case scenarios of a function class e.g. VC-dim [V&C,1971, Bartlett, 2003, Maas, 1993 etc.]

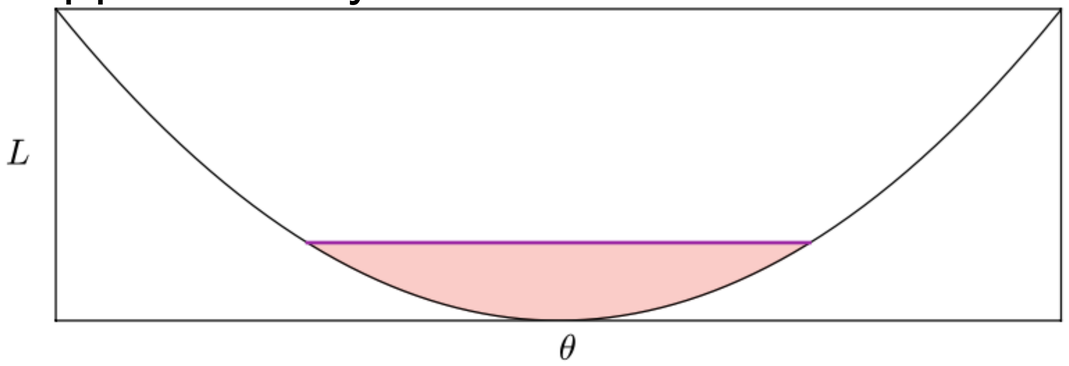
1. Stability: robustness of learning algorithms e.g. algorithmic and uniform argument stability, loss stability [Bartlett et al., 2003, Liu et al., 2017]
   Note: we do not have time for this today, but they are wonderful results!

1. **Local sensitivity: robustness of the already visited solutions e.g. flatness [Hochreiter et al., 1997, Neyshabur et al. 2018, Dinh et al., 2018, Novak et al. 2018]**



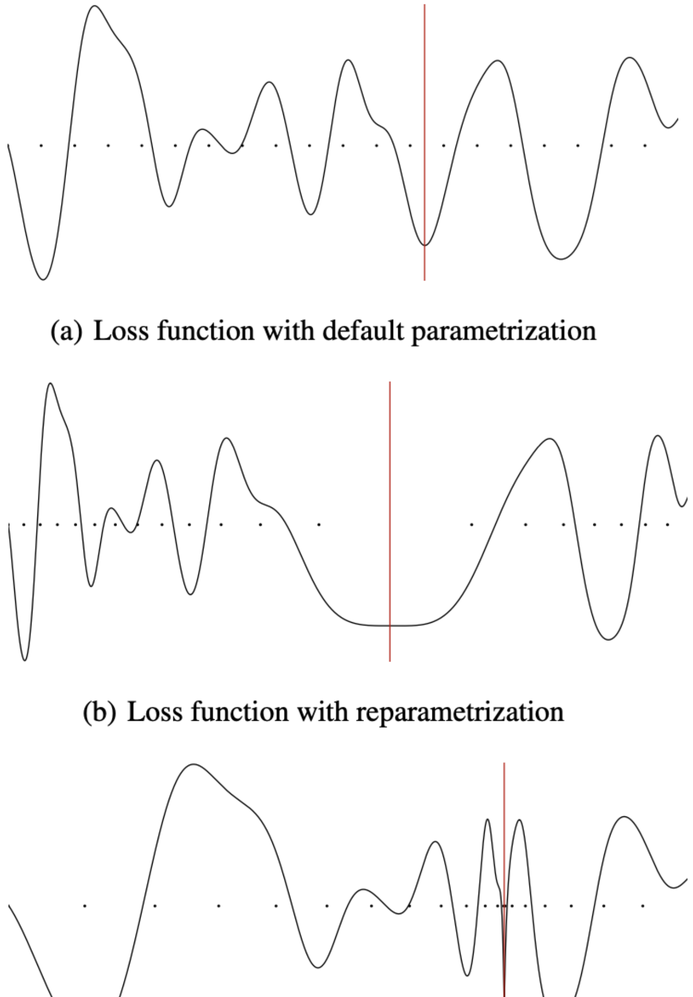
Flatness hypothesis?

[Hochreiter & Schmidhuber, 1997] flat minimum is "a large connected region in weight space where the error remains approximately constant". Loss manifold!



But ReLU is 1-homogeneous ...

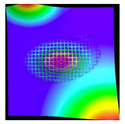
[Dinh et al., 2018]



(a) Loss function with default parametrization

(b) Loss function with reparametrization

(c) Loss function with another reparametrization

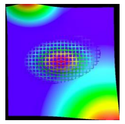


Flatness hypothesis

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, Nathan Srebro Exploring Generalization in Deep Learning, 2017

Properties which should hold for a proper complexity measure:

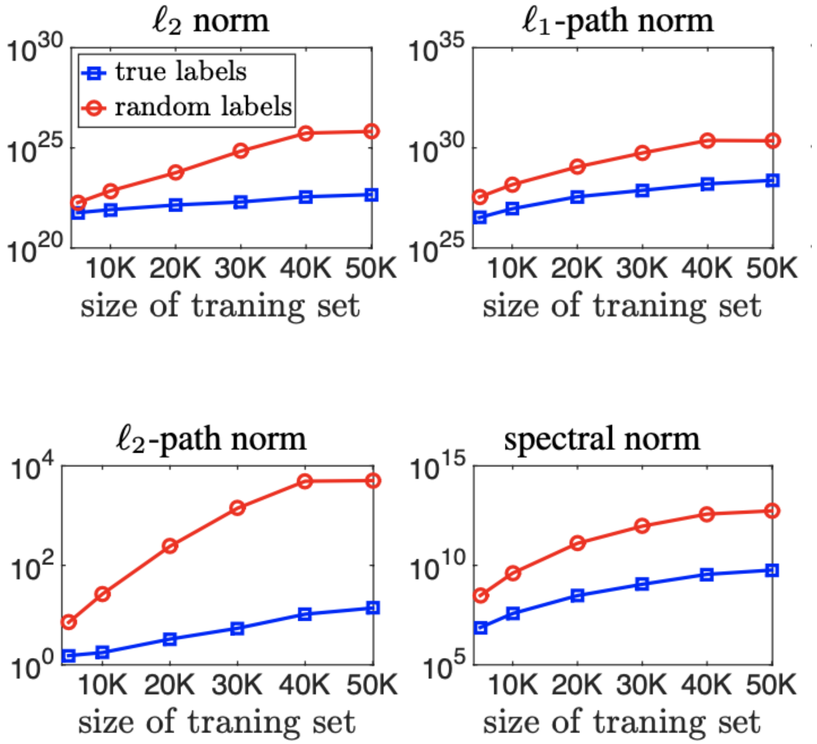
- “networks learned using real labels (and which generalizes well) have much lower complexity than networks learned using random labels (and which obviously do not generalize well).”
- “complexity measure decrease as we increase the number of hidden units.“
- “We expect a correlation between the complexity measure and generalization ability among zero-training error models. “



Flatness hypothesis [Neyshabur et al., 2017]

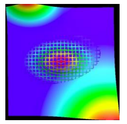
They investigate four norm-based measures ($f_w(x)$ function with parameter w):

- ℓ_2 norm with capacity proportional to $\frac{1}{\gamma^2_{\text{margin}}} \prod_{i=1}^{d} 4 \|W_i\|_F^2$
- ℓ_1-path norm with capacity proportional to $\frac{1}{\gamma^2_{\text{margin}}} \left(\sum_{j \in \prod_{k=0}^{d}[h_k]} \left| \prod_{i=1}^{d} 2W_i[j_i, j_{i-1}] \right| \right)^2$ 18].
- ℓ_2-path norm with capacity proportional to $\frac{1}{\gamma^2_{\text{margin}}} \sum_{j \in \prod_{k=0}^{d}[h_k]} \prod_{i=1}^{d} 4h_i W_i^2[j_i, j_{i-1}]$.
- spectral norm with capacity proportional to $\frac{1}{\gamma^2_{\text{margin}}} \prod_{i=1}^{d} h_i \|W_i\|_2^2$.



where the margin is defined as (max margin as in SVM :)):

$$f_{\mathbf{w}}(\mathbf{x})[y_{\text{true}}] - \max_{y \neq y_{\text{true}}} f_{\mathbf{w}}(\mathbf{x})[y]$$



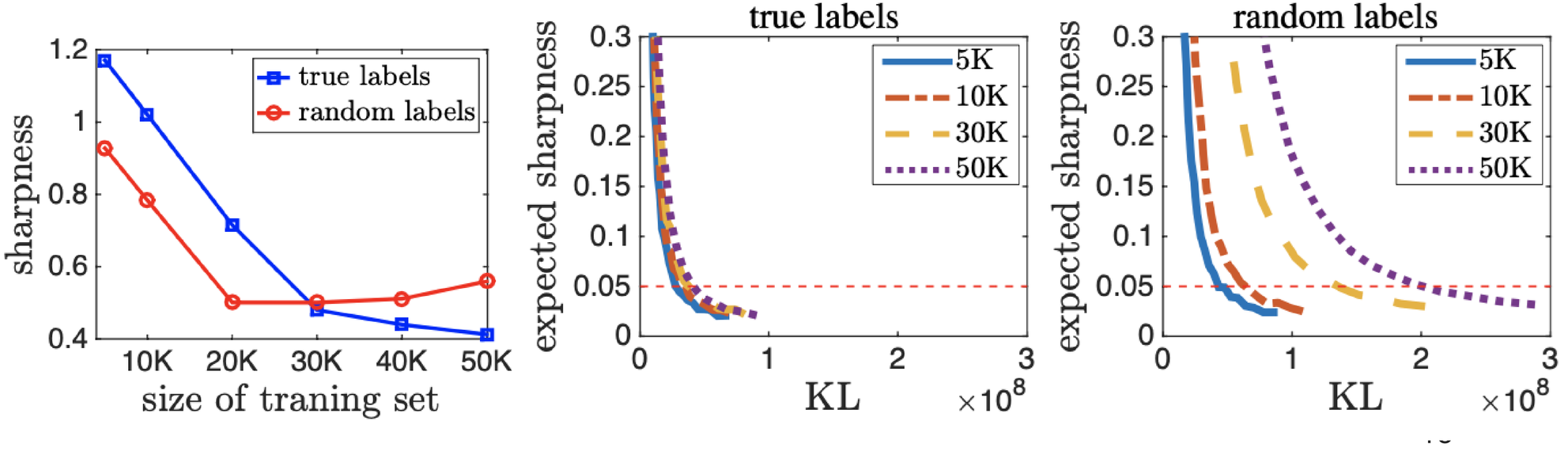
Flatness hypothesis [Neyshabur et al., 2017]

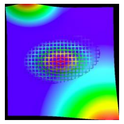
Since the above measures are still not good enough they suggest the expected sharpness:

Sharpness [Keskar et al., 2016]: $\zeta_\alpha(\mathbf{w}) = \frac{\max_{|\boldsymbol{\nu}_i| \le \alpha(|\mathbf{w}_i|+1)} \widehat{L}(f_{\mathbf{w}+\boldsymbol{\nu}}) - \widehat{L}(f_{\mathbf{w}})}{1+\widehat{L}(f_{\mathbf{w}})} \simeq \max_{|\boldsymbol{\nu}_i| \le \alpha(|\mathbf{w}_i|+1)} \widehat{L}(f_{\mathbf{w}+\boldsymbol{\nu}}) - \widehat{L}(f_{\mathbf{w}})$

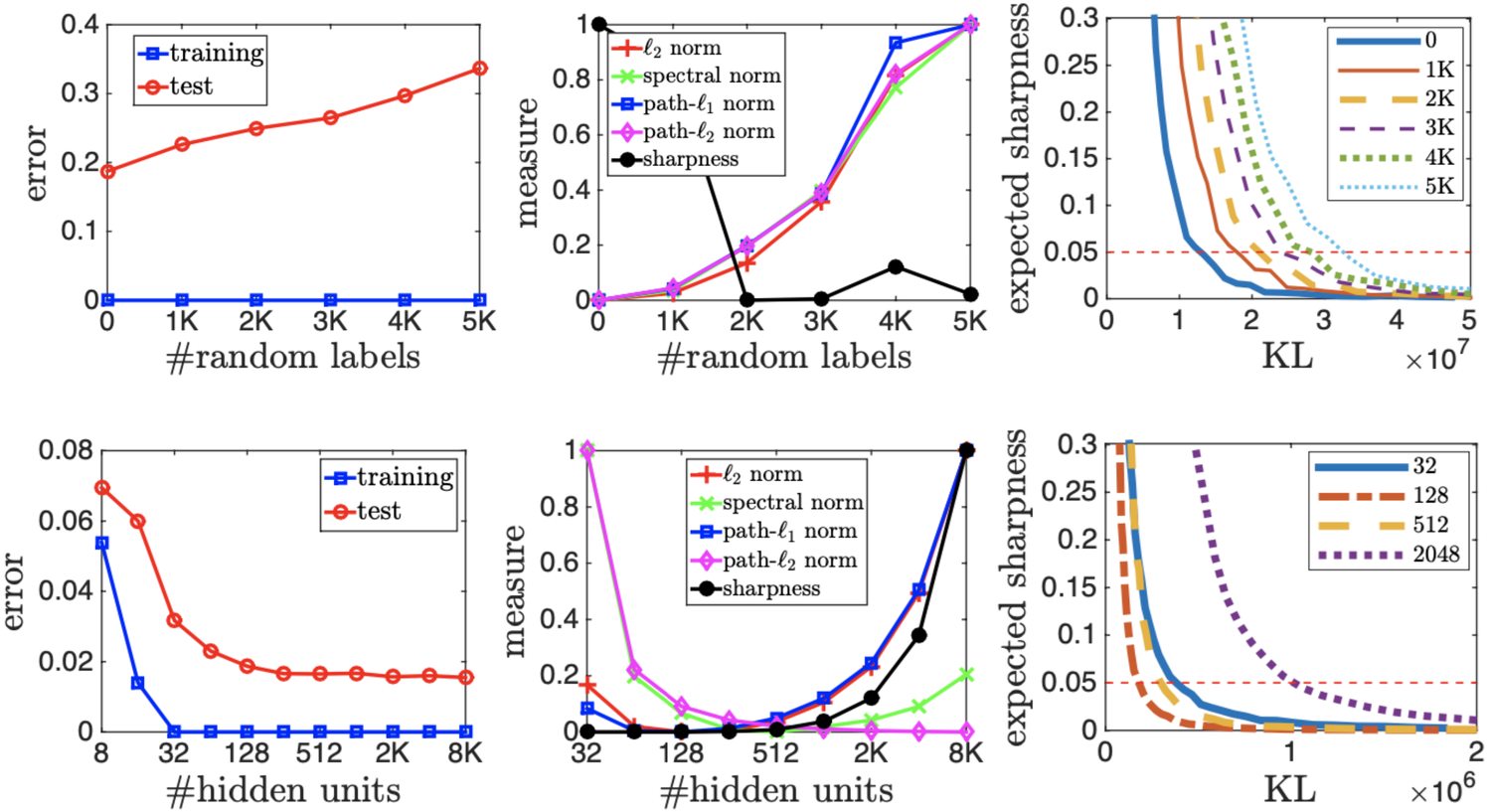
Expected sharpness: $\mathbb{E}_{\boldsymbol{\nu}}[\widehat{L}(f_{\mathbf{w}+\boldsymbol{\nu}})] - \widehat{L}(f_{\mathbf{w}})$

(it is more complicated, please read the paper)



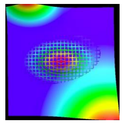


Flatness hypothesis [Neyshaur et al., 2017]



Unfortunately these bounds are not connected to the properties of the optimization

The starting hypothesis was that sharpness, flatness are realistic measures for generalization… or?



Flatness hypothesis [Dinh et al., 2018]

Laurent Dinh, Razvan Pascanu, Samy Bengio, Yoshua Bengio: Sharp Minima Can Generalize For Deep Nets, 2017

Definition 1. *Given $\epsilon > 0$, a minimum θ, and a loss L, we define $C(L, \theta, \epsilon)$ as the largest (using inclusion as the partial order over the subsets of Θ) connected set containing θ such that $\forall \theta' \in C(L, \theta, \epsilon), L(\theta') < L(\theta) + \epsilon$. The ϵ-flatness will be defined as the volume of $C(L, \theta, \epsilon)$. We will call this measure the volume ϵ-flatness.*

Theorem 2. *For a one-hidden layer rectified neural network of the form*

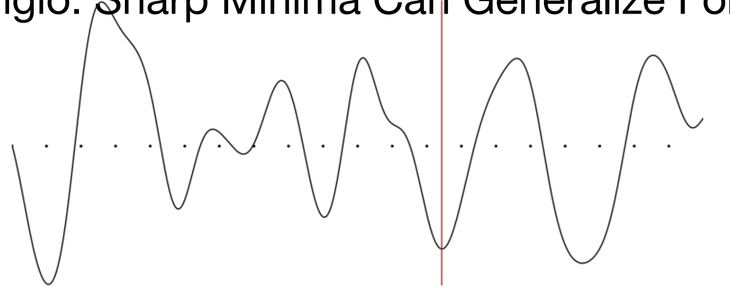
$$y = \phi_{rect}(x \cdot \theta_1) \cdot \theta_2,$$

and a minimum $\theta = (\theta_1, \theta_2)$, such that $\theta_1 \neq 0$ and $\theta_2 \neq 0$, $\forall \epsilon > 0$ $C(L, \theta, \epsilon)$ has an infinite volume.

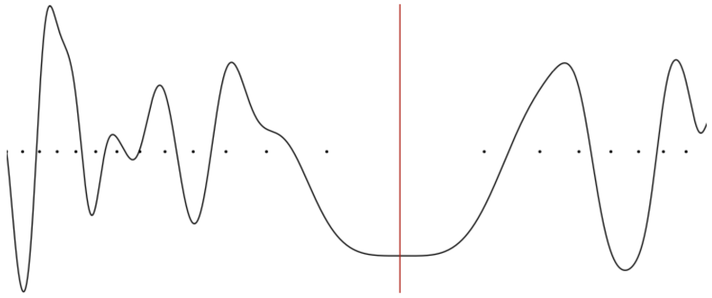
Definition 5. *For a single hidden layer rectifier feedforward network we define the family of transformations*

$$T_\alpha : (\theta_1, \theta_2) \mapsto (\alpha\theta_1, \alpha^{-1}\theta_2)$$

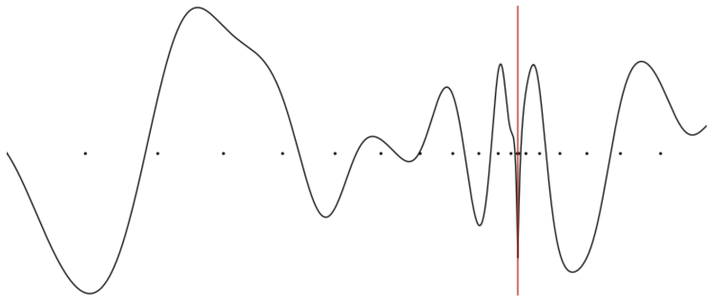
which we refer to as a α-scale transformation.



(a) Loss function with default parametrization



(b) Loss function with reparametrization



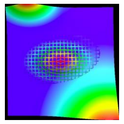
(c) Loss function with another reparametrization

Transformations which do not change the output but allow rescaling of parameters… (OK, reg.?)

Flatness is arbitrary and symmetry of ReLU deep networks allow us to make, delete or shift flat valleys… :(

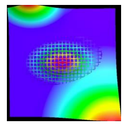
Great, what's next?

Geometry!



Outline

1. Generalization: Vapnik-Chervonenkis theorem and deep neural networks
2. Information geometry of neural networks: Fisher-Rao norm and K-FAC approximation of Fisher information
3. Complexity of ReLU networks: evaluation of linear regions and tangent space sensitivity



Fisher-Rao metric and complexity of neural networks [Liang et al., 2017]

Liang, Poggio, Rakhlin & Stokes, 2017: Fisher-Rao Metric, Geometry, and Complexity of Neural Networks

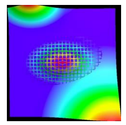
They propose the following questions:

1. What are the complexity notions that control the generalization aspects of neural networks?
2. Why does stochastic gradient descent, or other variants, find parameters with small complexity?

Motivation:

1. There are many continuous operations on the parameters of ReLU nets that will result in exactly the same prediction -> generalization can only depend on the equivalence class obtained by identifying parameters under these transformations

2. Flatness of the loss function: not too robust [Dinh et al., 2018], although

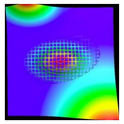


Fisher-Rao metric and complexity of neural networks [Liang et al., 2017]

Flatness of the loss function: not too robust [Dinh et al., 2018], although…

- Geometric characterization of generalization that is invariant under some transformations which causes flat minima measures to fail
- in case of information geometry … Fisher information :)
- they assume bias-less ReLU networks...

For linear functions: $\langle \partial f / \partial \theta, \theta \rangle = f_\theta(x)$ the tangent space of a ReLU nets.



sidenote: Fisher information

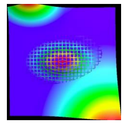
Let us consider a parametric class of probability models $P(X|\theta)$ where $\theta \in \Theta \subseteq R^d$.

Provided that the dependence on θ is sufficiently smooth, the collection of models with parameters from Θ can then be viewed as a (statistical) manifold M_Θ. M_Θ can be turned into a Riemannian manifold by giving a scalar product at the tangent space of each point $P(X|\theta) \in M_\Theta$ via a positive semi-definite matrix F(θ), which varies smoothly with the base point θ.

Classical gradient based learning -> walking on a manifold via a continuously differentiable function and the dimension is … (DNN?)

Special metrics on NN [Ollivier et al., 2015] for better optimization. Fisher information (F(θ) or I_θ)is a special case of Hessian metrics [Shima et al., 1995, Amari, 2000]:

$$F(\theta) := \mathbf{E}(\nabla_\theta \log P(X|\theta) \nabla_\theta \log P(X|\theta)^T)$$



sidenote: Fisher information

In particular, if P(X|θ) is a probability density function, then the ij-th entry of F(θ) is

$$f_{ij} = \int_X P(X|\theta)(\frac{\partial}{\partial\theta_i}\log P(X|\theta))(\frac{\partial}{\partial\theta_j}\log P(X|\theta))dX$$

If we refer the vector $G_X = \nabla_\theta \log P(X|\theta)$ as the Fisher score of the example X, we can define a mapping X -> $G_x F^{-1/2}$ and a kernel

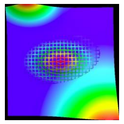
$$K_\theta(x,y) := G_x^T F^{-1} G_x$$

An intuitive interpretation is that G_X gives the direction where the parameter vector θ should be changed to fit best the data X.

Why F^{-1}? Steepest (the gradient has fixed length) natural gradient by Shun-ichi Amari [Amari,1996]!

It can be proved that the Fisher metric under congruent embeddings by Markov morphisms is distance preserving (isometry) [Cencov,1982]

Moreover the Fisher metric is essentially the unique Riemannian metric with this property [Campbell,1985,Campbell,1986, Lebanon, 2004, Petz et al.,1999, Jaakkola and Haussler, 1998].



Fisher-Rao metric and complexity of neural networks [Liang et al., 2017]

1. Let data generating process belong to a parametric fam $\mathcal{P} \in \{\mathcal{P}_\theta \,|\, \theta \in \Theta_L\}$

2. Fisher-Rao metric on a parametric family is defined via an inner product for each configuration

3. For each pair of parameter (drawn from the fam.) define a pair of tangent vectors:

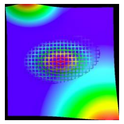
$$\bar{\alpha} := \mathrm{d}p_{\theta+t\alpha}/\mathrm{d}t|_{t=0},\ \bar{\beta} := \mathrm{d}p_{\theta+t\beta}/\mathrm{d}t|_{t=0}$$

and a local inner product (posdef -> Fisher-Rao metric):

$$\langle \bar{\alpha}, \bar{\beta} \rangle_{p_\theta} := \int \frac{\bar{\alpha}}{p_\theta} \frac{\bar{\beta}}{p_\theta} p_\theta$$

Seems like something connected to Fisher information, and no surprise it is:

$$\langle \bar{\alpha}, \bar{\beta} \rangle_{p_\theta} = \langle \alpha, I_\theta \beta \rangle$$



Fisher-Rao metric and complexity of neural networks [Liang et al., 2017]

Definition 2. *The Fisher-Rao norm for a parameter* θ *is defined as the quadratic form* $\|\theta\|_{\text{fr}}^2 := \langle \theta, \mathbf{I}(\theta)\theta \rangle$ *where* $\mathbf{I}(\theta) = \mathbb{E}[\nabla_\theta \ell(f_\theta(X), Y) \otimes \nabla_\theta \ell(f_\theta(X), Y)]$.

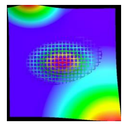
Theorem 3.1 (Fisher-Rao norm). *Assume the loss function* $\ell(\cdot, \cdot)$ *is smooth in the first argument. The following identity holds for a feedforward neural network (Definition 1) with* L *hidden layers and activations satisfying* $\sigma(z) = \sigma'(z)z$*:*

$$\|\theta\|_{\text{fr}}^2 = (L+1)^2 \, \mathbb{E} \left\langle \frac{\partial \ell(f_\theta(X), Y)}{\partial f_\theta(X)}, f_\theta(X) \right\rangle^2 . \quad (3.1)$$

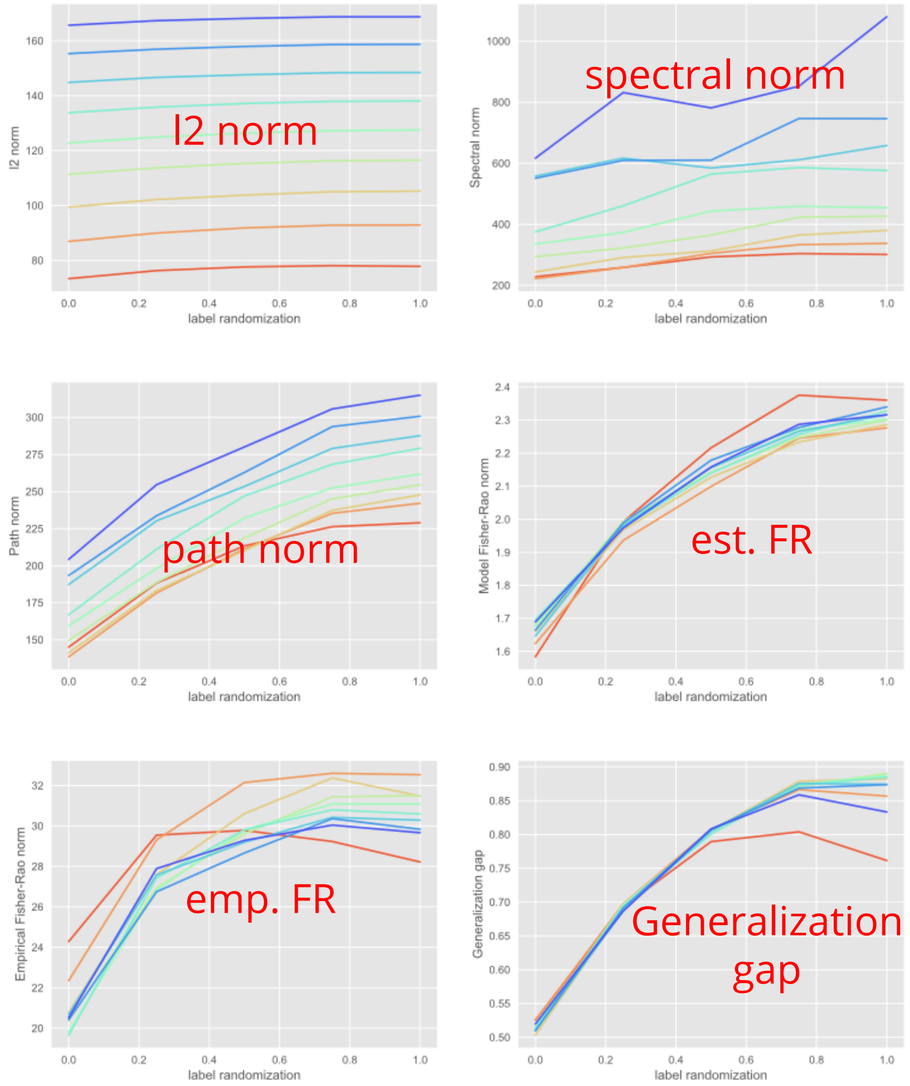
Equivalence classes are determined by FR.
FR vs. α-scaling?

Corollary 3.1 (Invariance). *If there are two parameters* $\theta_1, \theta_2 \in \Theta_L$ *such that they are equivalent, in the sense that* $f_{\theta_1} = f_{\theta_2}$*, then their Fisher-Rao norms are equal, i.e.,* $\|\theta_1\|_{\text{fr}} = \|\theta_2\|_{\text{fr}}$.

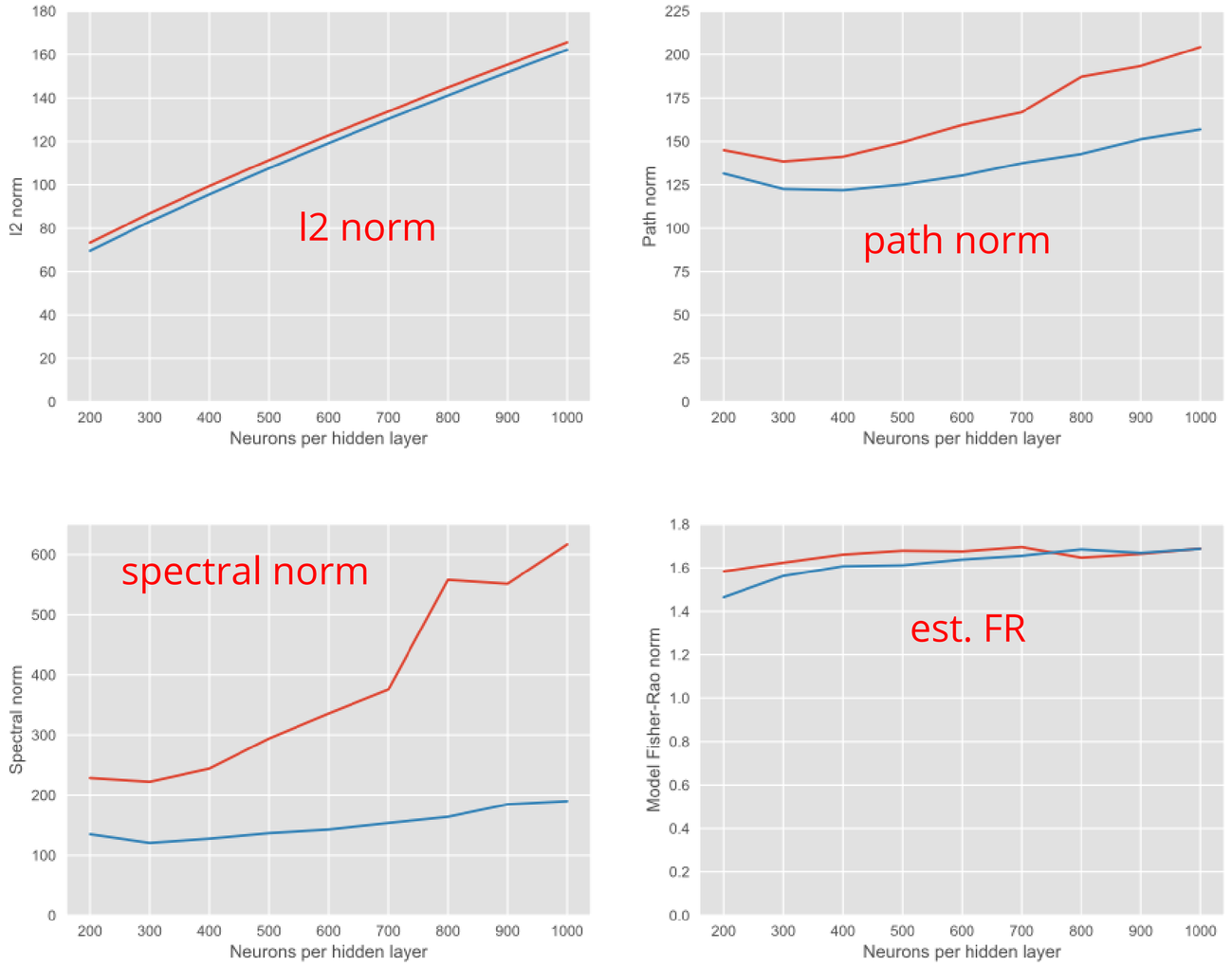
Is there any relation to generalization?



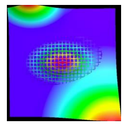
Fisher-Rao metric and complexity of neural networks [Liang et al., 2017]



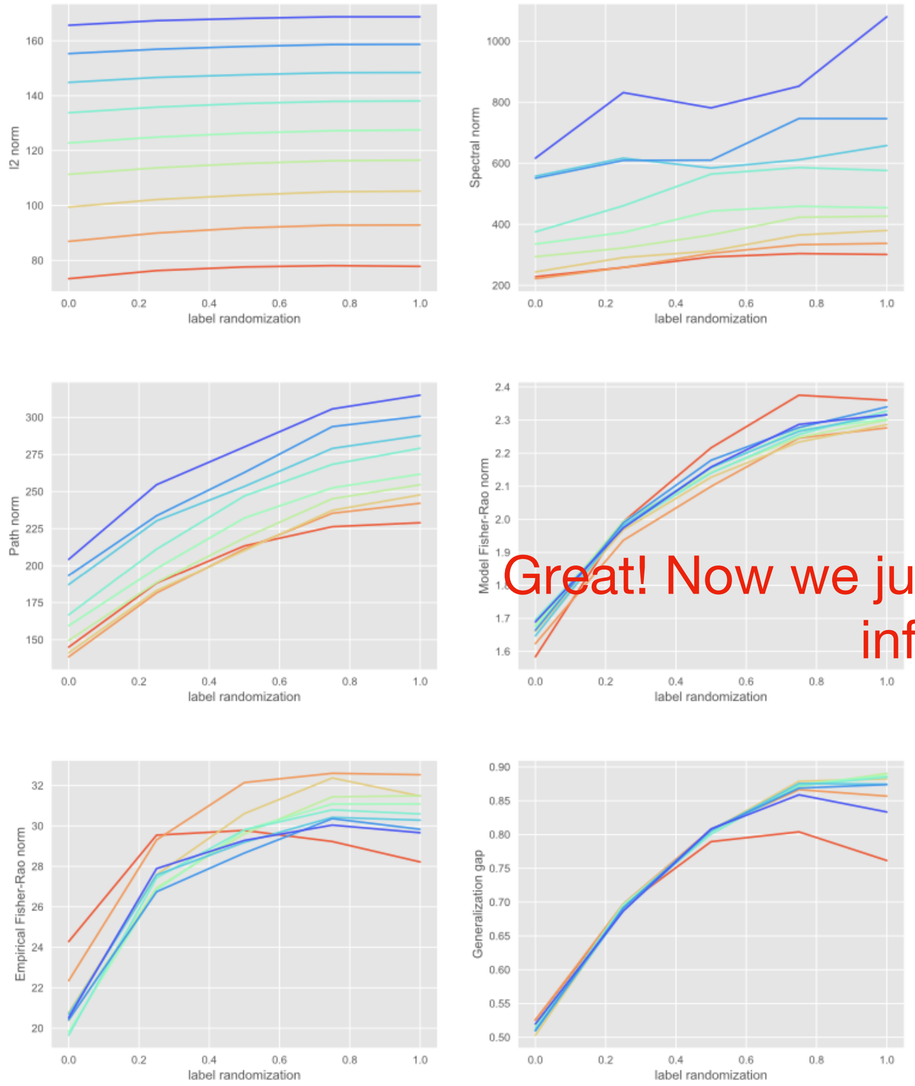
Capacity measures on label randomization after opt. with GD.



Neurons per hidden layer

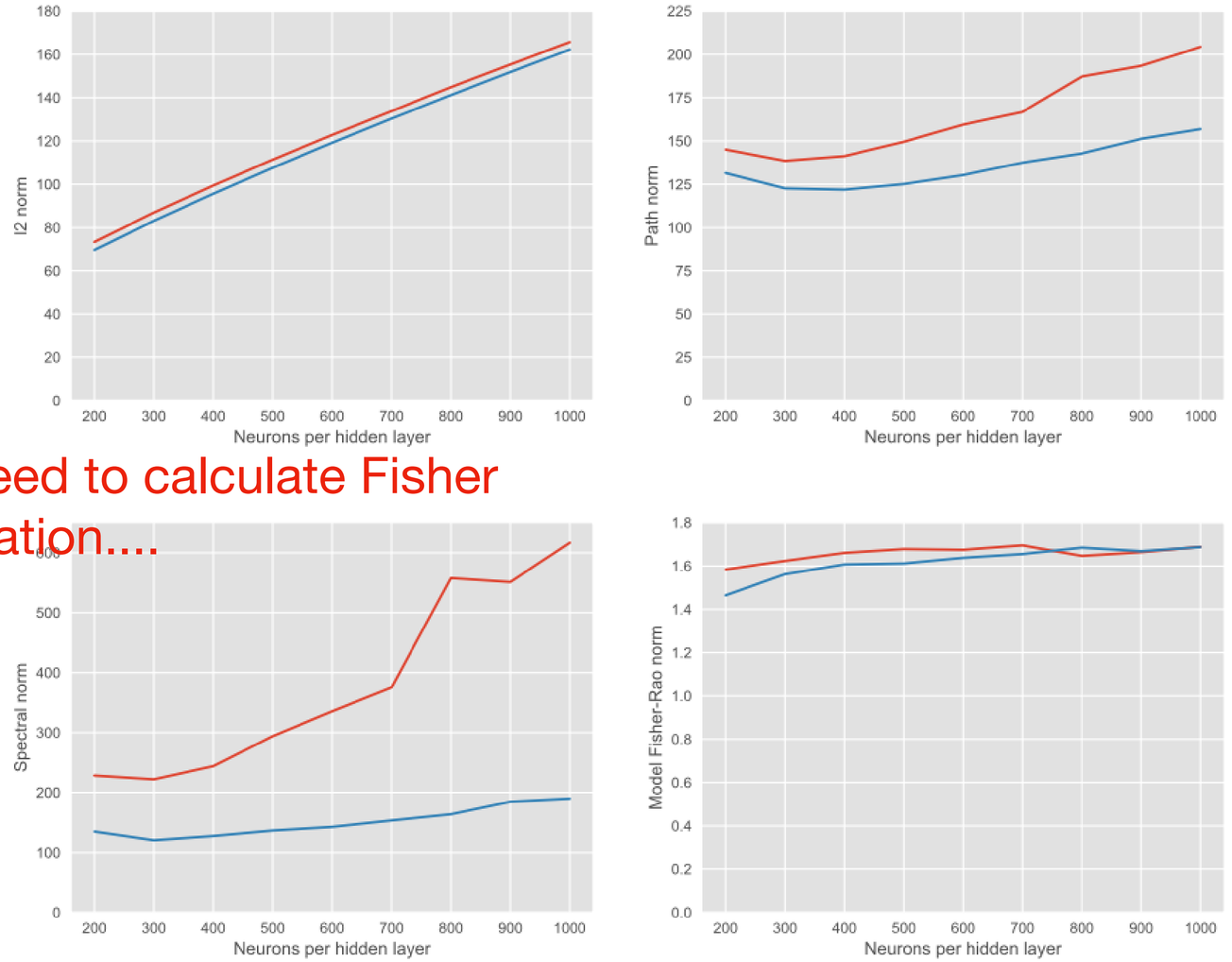


Fisher-Rao metric and complexity of neural networks [Liang et al., 2017]

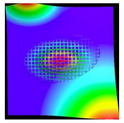


Capacity measures on label randomization after opt. with GD.

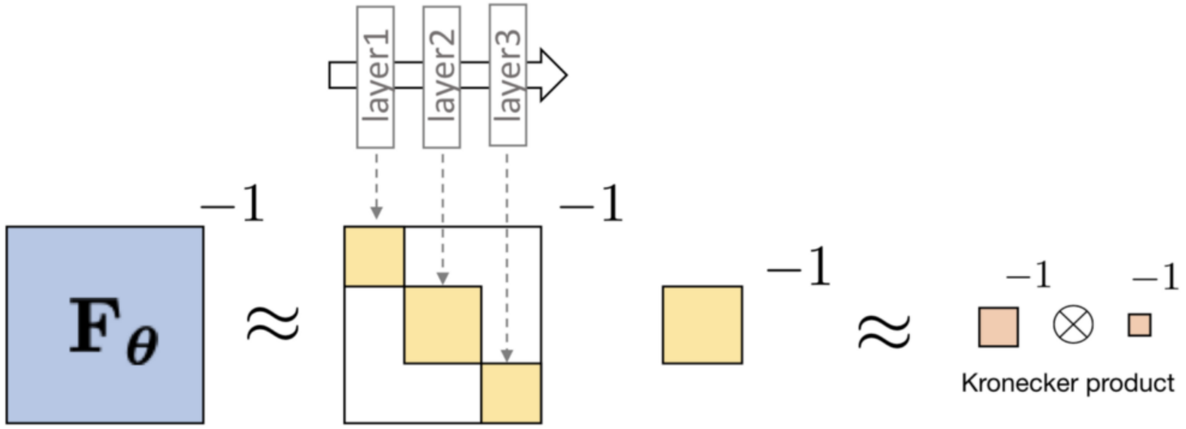
Great! Now we just need to calculate Fisher information....



Neurons per hidden layer



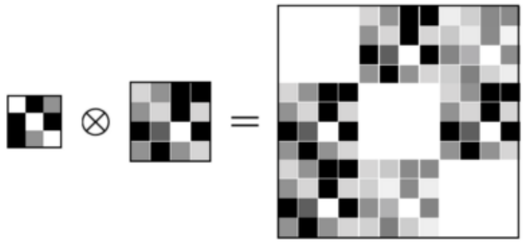
Martens, James, and Roger Grosse. "Optimizing neural networks with kronecker-factored approximate curvature." In *International conference on machine learning*, pp. 2408-2417. 2015. Fun fact: [Heskes, 2000]



Kronecker product (inverse!!!!!):

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1}\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1}\mathbf{B} & \cdots & [\mathbf{A}]_{m,n}\mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$



$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

Fisher information:

$$\mathbf{F}_i = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})} \left[\nabla_i E(\boldsymbol{\theta}) \nabla_i E(\boldsymbol{\theta})^{\mathrm{T}} \right]$$

Fisher score:

$$\nabla_i E(\boldsymbol{\theta}) = \mathbf{g}_i \otimes \mathbf{a}_{i-1} \in \mathbb{R}^{d_{i-1} \cdot d_i}$$

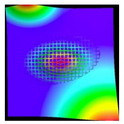
$\mathbf{a}_{i-1} \in \mathbb{R}^{d_{i-1}}$: the input to i-th layer (activation of (i-1)-th layer)

$\mathbf{g}_i = \dfrac{\partial E(\boldsymbol{\theta})}{\partial \mathbf{s}_i} \in \mathbb{R}^{d_i}$: the gradient for the output of i-th layer

thus

$$\begin{aligned} \mathbf{F}_i &\approx \mathbb{E}\left[\mathbf{g}_i \mathbf{g}_i^{\mathrm{T}}\right] \otimes \mathbb{E}\left[\mathbf{a}_{i-1} \mathbf{a}_{i-1}^{\mathrm{T}}\right] \\ &= \mathbf{G}_i \otimes \mathbf{A}_{i-1} \end{aligned}$$

$$\begin{aligned} \mathbf{F}_i &\in \mathbb{R}^{d_{i-1} \cdot d_i \times d_{i-1} \cdot d_i} \\ \mathbf{A}_{i-1} &\in \mathbb{R}^{d_{i-1} \times d_{i-1}} \\ \mathbf{G}_i &\in \mathbb{R}^{d_i \times d_i} \end{aligned}$$



K-FAC [Martens & Grosse, 2015]

All layers in AlexNet
60,000,000 parameters

Fisher information matrix

$$\mathbf{F}_{\boldsymbol{\theta}} \in \mathbb{R}^{60{,}000{,}000 \times 60{,}000{,}000}$$

$^{-1}$

Final layer of AlexNet
Input dimension: 4,096
Output dimension: 1,000
4,096,000 parameters

Fisher block

$$\mathbf{F}_i \in \mathbb{R}^{4{,}096{,}000 \times 4{,}096{,}000}$$

$^{-1}$

Kronecker factors

$$\mathbf{A}_{i-1} \in \mathbb{R}^{4{,}096 \times 4{,}096}$$

$$\mathbf{G}_i \in \mathbb{R}^{1{,}000 \times 1{,}000}$$

$^{-1}$ $\otimes$ $^{-1}$

Great! Wait a minute, why are we even doing this?

Fisher information:

$$\mathbf{F}_i = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})} \left[\nabla_i E(\boldsymbol{\theta}) \nabla_i E(\boldsymbol{\theta})^{\mathrm{T}} \right]$$

Fisher score:

$$\nabla_i E(\boldsymbol{\theta}) = \mathbf{g}_i \otimes \mathbf{a}_{i-1} \in \mathbb{R}^{d_{i-1} \cdot d_i}$$

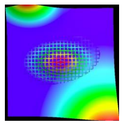
$\mathbf{a}_{i-1} \in \mathbb{R}^{d_{i-1}}$: the input to i-th layer (activation of (i-1)-th layer)

$\mathbf{g}_i = \dfrac{\partial E(\boldsymbol{\theta})}{\partial \mathbf{s}_i} \in \mathbb{R}^{d_i}$: the gradient for the output of i-th layer

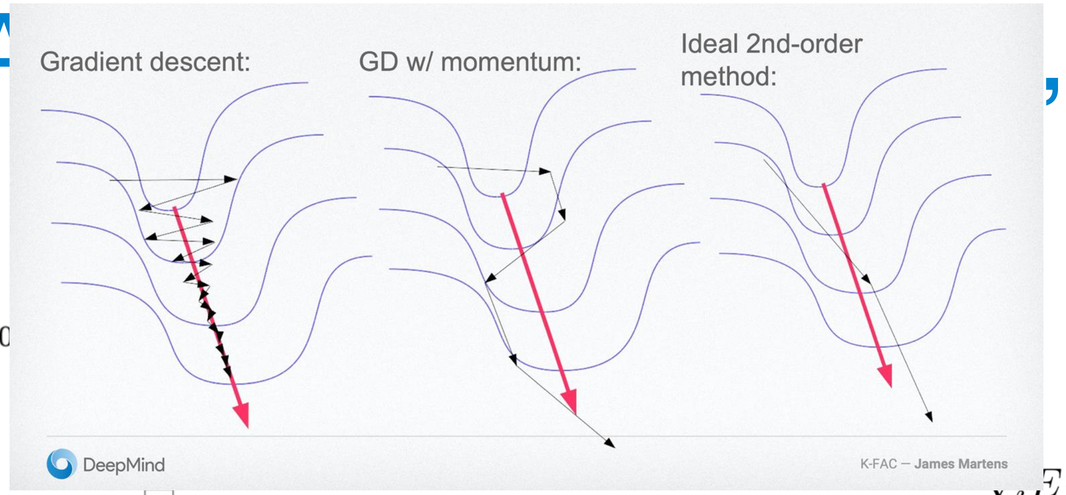
thus

$$\begin{aligned} \mathbf{F}_i &\approx \mathbb{E}\left[\mathbf{g}_i \mathbf{g}_i^{\mathrm{T}}\right] \otimes \mathbb{E}\left[\mathbf{a}_{i-1} \mathbf{a}_{i-1}^{\mathrm{T}}\right] \\ &= \mathbf{G}_i \otimes \mathbf{A}_{i-1} \end{aligned}$$

$$\begin{aligned} \mathbf{F}_i &\in \mathbb{R}^{d_{i-1} \cdot d_i \times d_{i-1} \cdot d_i} \\ \mathbf{A}_{i-1} &\in \mathbb{R}^{d_{i-1} \times d_{i-1}} \\ \mathbf{G}_i &\in \mathbb{R}^{d_i \times d_i} \end{aligned}$$



K-FA



, 2015]

All layers in AlexNet
60,000,000 parameters

Fisher information matrix

$$\mathbf{F}_{\boldsymbol{\theta}} \in \mathbb{R}^{60{,}000{,}000 \times 60{,}000}$$

Final layer of AlexNet
Input dimension: 4,096
Output dimension: 1,000
4,096,000 parameters

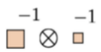
Fisher block

$$\mathbf{F}_i \in \mathbb{R}^{4{,}096{,}000 \times 4{,}096{,}000}$$

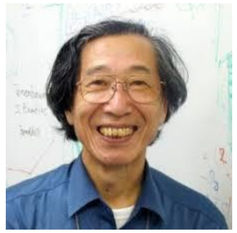
Kronecker factors

$$\mathbf{A}_{i-1} \in \mathbb{R}^{4{,}096 \times 4{,}096}$$

$$\mathbf{G}_i \in \mathbb{R}^{1{,}000 \times 1{,}000}$$



Natural gradient
[Amari, 1998] is finally
possible on large-scale NN!



Fisher information:

$$\mathbf{F}_i = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})} \left[\nabla_i E(\boldsymbol{\theta}) \nabla_i E(\boldsymbol{\theta})^{\mathrm{T}} \right]$$

Fisher score:

$$\nabla_i E(\boldsymbol{\theta}) = \mathbf{g}_i \otimes \mathbf{a}_{i-1} \in \mathbb{R}^{d_{i-1} \cdot d_i}$$

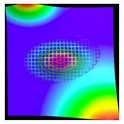
$\mathbf{a}_{i-1} \in \mathbb{R}^{d_{i-1}}$: the input to i-th layer (activation of (i-1)-th layer)

$\mathbf{g}_i = \dfrac{\partial E(\boldsymbol{\theta})}{\partial \mathbf{s}_i} \in \mathbb{R}^{d_i}$: the gradient for the output of i-th layer

thus

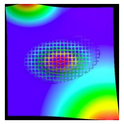
$$\begin{aligned} \mathbf{F}_i &\approx \mathbb{E}\left[\mathbf{g}_i \mathbf{g}_i^{\mathrm{T}}\right] \otimes \mathbb{E}\left[\mathbf{a}_{i-1} \mathbf{a}_{i-1}^{\mathrm{T}}\right] \\ &= \mathbf{G}_i \otimes \mathbf{A}_{i-1} \end{aligned}$$

$$\begin{aligned} \mathbf{F}_i &\in \mathbb{R}^{d_{i-1} \cdot d_i \times d_{i-1} \cdot d_i} \\ \mathbf{A}_{i-1} &\in \mathbb{R}^{d_{i-1} \times d_{i-1}} \\ \mathbf{G}_i &\in \mathbb{R}^{d_i \times d_i} \end{aligned}$$



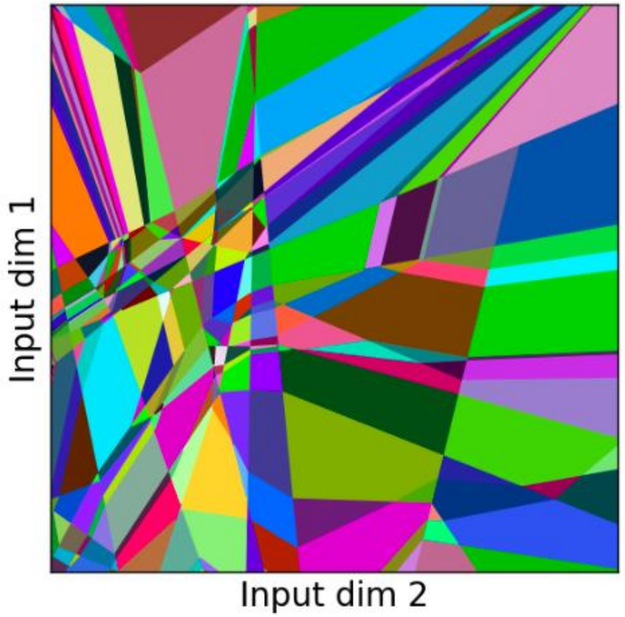
Outline

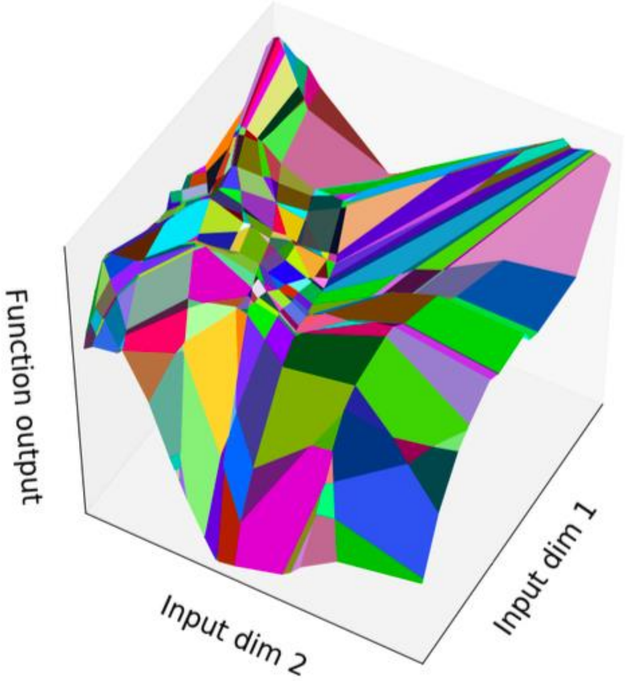
1. Generalization: Vapnik-Chervonenkis theorem and deep neural networks
2. Information geometry of neural networks: Fisher-Rao norm and KFAC approximation of Fisher information
3. Complexity of ReLU networks: evaluation of linear regions and tangent space sensitivity



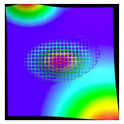
Complexity of linear regions

Hanin & Rolnick, Deep ReLU Networks Have Surprisingly Few Activation Patterns, 2019
Hanin & Rolnick, Complexity of Linear Regions in Deep Networks, 2019

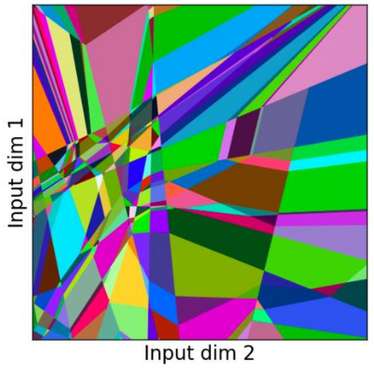


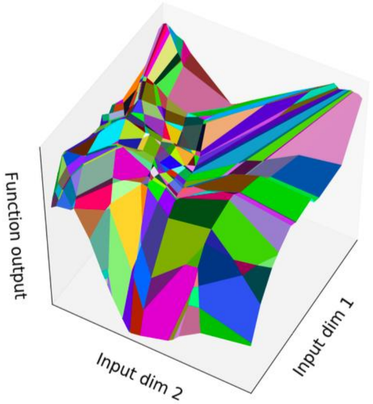


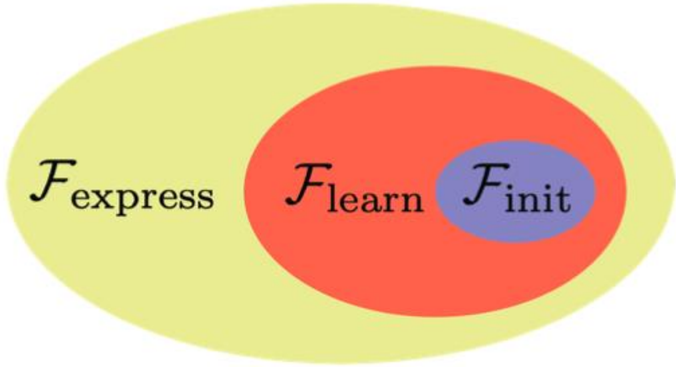
Given a vector θ of its trainable parameters, N computes a continuous and piecewise linear function x ⍰ → N (x; θ). Thus each θ is associated with a partition of input space R^{input} into activation regions, polytopes on which N (x; θ) computes a single linear function corresponding to a fixed activation pattern in the neurons of N .



Complexity of linear regions [Hanin & Rolnick, 2019](2x)

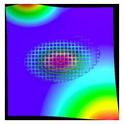




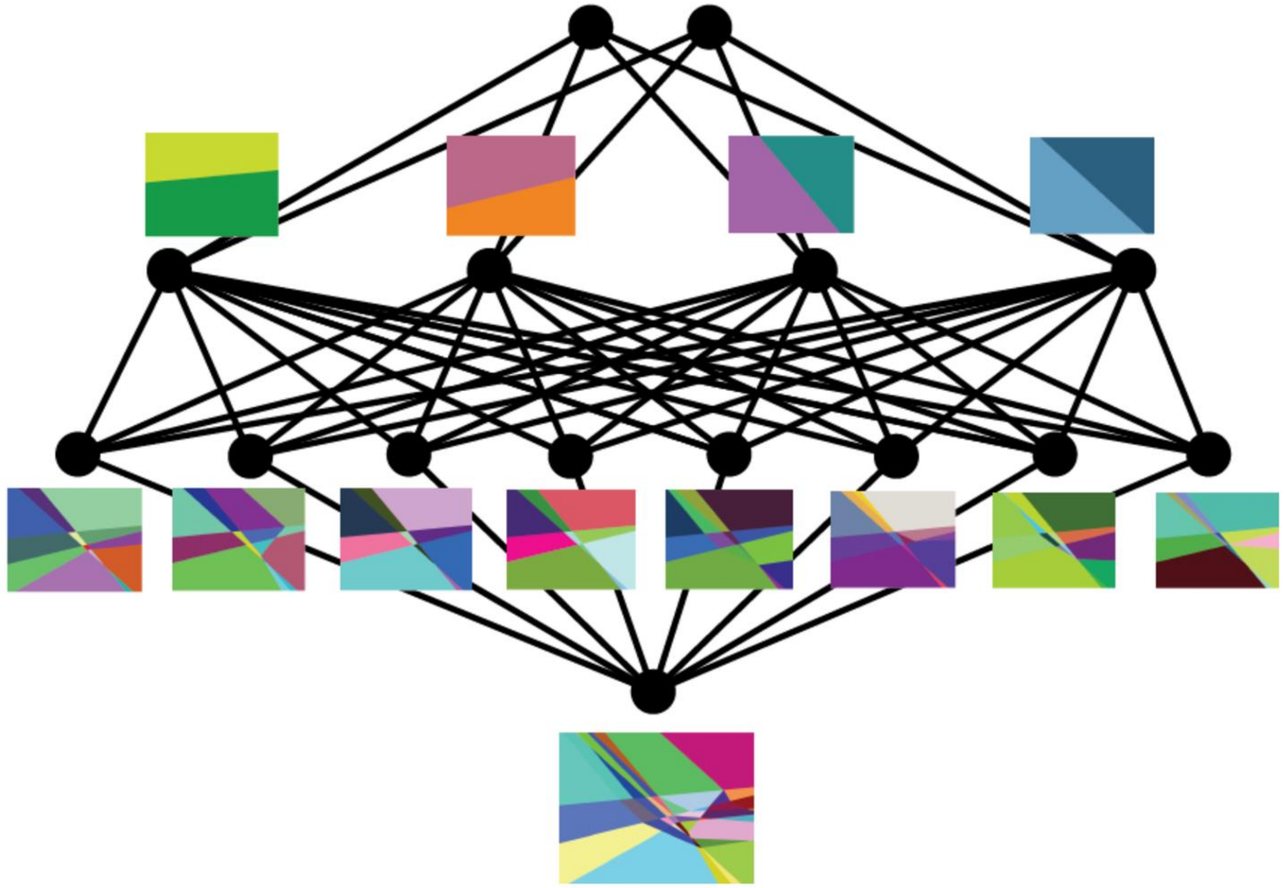


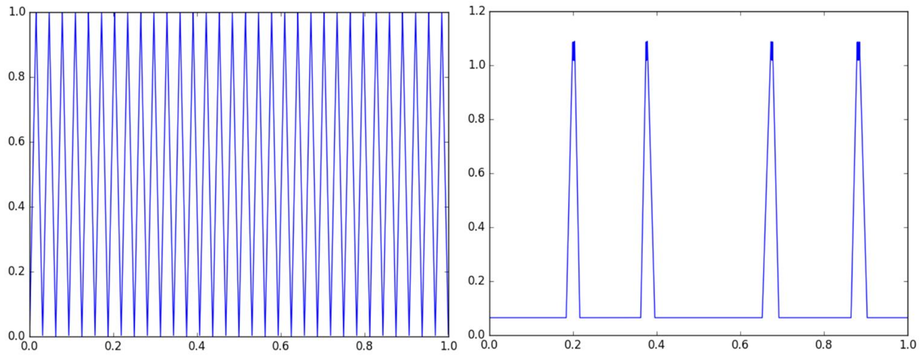
1. surprisingly small gap between F_{init} and F_{learn}
2. the number of activation regions at start ($(\#\text{neurons})^2/2$) is not increasing exponentially during training

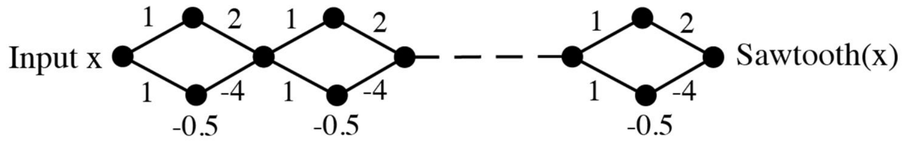
What is the difference between the maximum complexity of deep networks (exponential [Montufar et al., 2014]) and the complexity of functions that are actually learned in case of ReLU networks?



Complexity of linear regions [Hanin & Rolnick, 2019](2x)



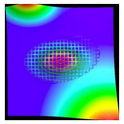




Extreme states (exp. number of regions) are not stable!

Even with small perturbations the complexity may fall…

Activation regions (input points with the same neuron activation pattern) are convex -> Are linear regions (connected components where the function is linear) convex?



Complexity of linear regions [Hanin & Rolnick, 2019](2x)

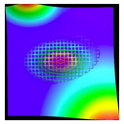
Activation regions (for any piecewise linear activation, ot just ReLU) are convex -> Are linear regions convex?

No, they are not!

Imagine a two convex regions joined together with a smooth (diff.!) border… or just a C-shaped smooth linear region with three convex activation regions.

The number of activation regions is always at least as large as the number of linear regions.

E.g. an entire layer of the network is zeroed out by ReLUs, leading many distinct activation regions to coalesce into a single linear region.



Complexity of linear regions [Hanin & Rolnick, 2019](2x)

Main result: upper bounds on the average number of activation regions per unit volume of input space for a feed-forward ReLU net with random weights/biases.

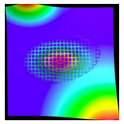
Theorem 5 (Counting Activation Regions). *Let $\mathcal{N}$ be a feed-forward ReLU network with no tied weights, input dimension n_{in}, output dimension 1, and random weights/biases satisfying:*

1. *The distribution of all weights has a density with respect to Lebesgue measure on $\mathbb{R}^{\#\text{weights}}$.*
2. *Every collection of biases has a density with respect to Lebesgue measure conditional on the values of all weights and other biases (for identically zero biases, see Appendix D).*
3. *There exists $C_{\text{grad}} > 0$ so that for every neuron z and each $m \geq 1$, we have*
$$\sup_{x \in \mathbb{R}^{n_{\text{in}}}} \mathbb{E}\left[\|\nabla z(x)\|^m\right] \leq C_{\text{grad}}^m.$$
4. *There exists $C_{\text{bias}} > 0$ so that for any neurons $z_1, \ldots, z_k$, the conditional distribution of the biases $\rho_{b_{z_1},\ldots,b_{z_k}}$ of these neurons given all the other weights and biases in $\mathcal{N}$ satisfies*
$$\sup_{b_1,\ldots,b_k \in \mathbb{R}} \rho_{b_{z_1},\ldots,b_{z_k}}(b_1, \ldots, b_k) \leq C_{\text{bias}}^k.$$

Then, there exists $\delta_0, T > 0$ depending on $C_{\text{grad}}, C_{\text{bias}}$ with the following property. Suppose that $\delta > \delta_0$. Then, for all cubes $\mathcal{C}$ with side length δ, we have

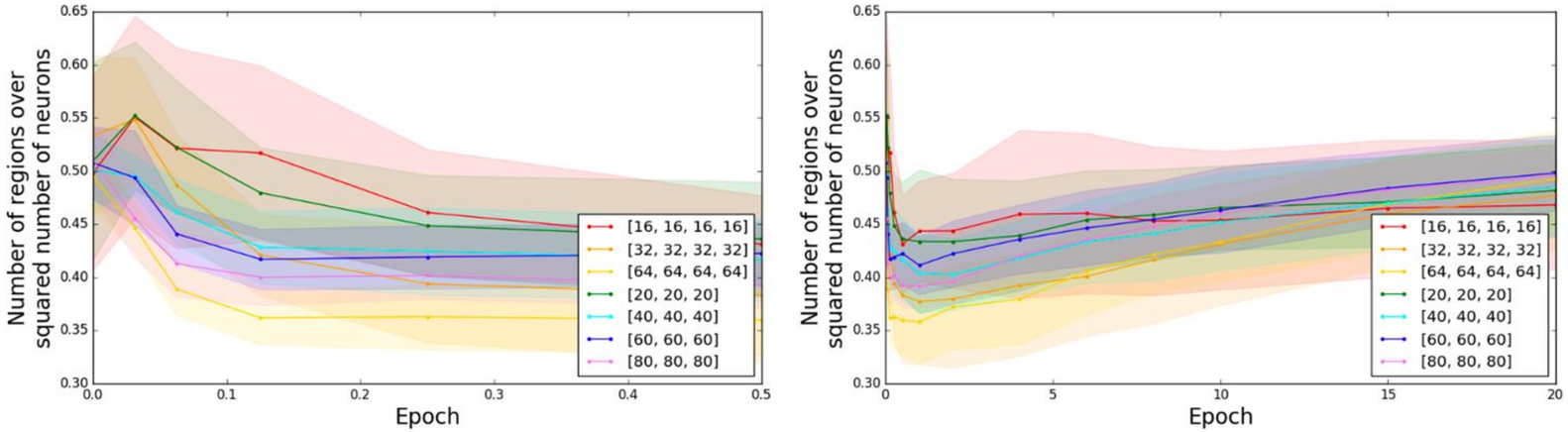
$$\frac{\mathbb{E}\left[\#\text{non-empty activation regions of } \mathcal{N} \text{ in } \mathcal{C}\right]}{\text{vol}(\mathcal{C})} \leq \begin{cases} (T\#\text{neurons})^{n_{\text{in}}}/n_{\text{in}}! & \#\text{neurons} \geq n_{\text{in}} \\ 2^{\#\text{neurons}} & \#\text{neurons} \leq n_{\text{in}} \end{cases}. \tag{5}$$

Here, the average is with respect to the distribution of weights and biases in $\mathcal{N}$.



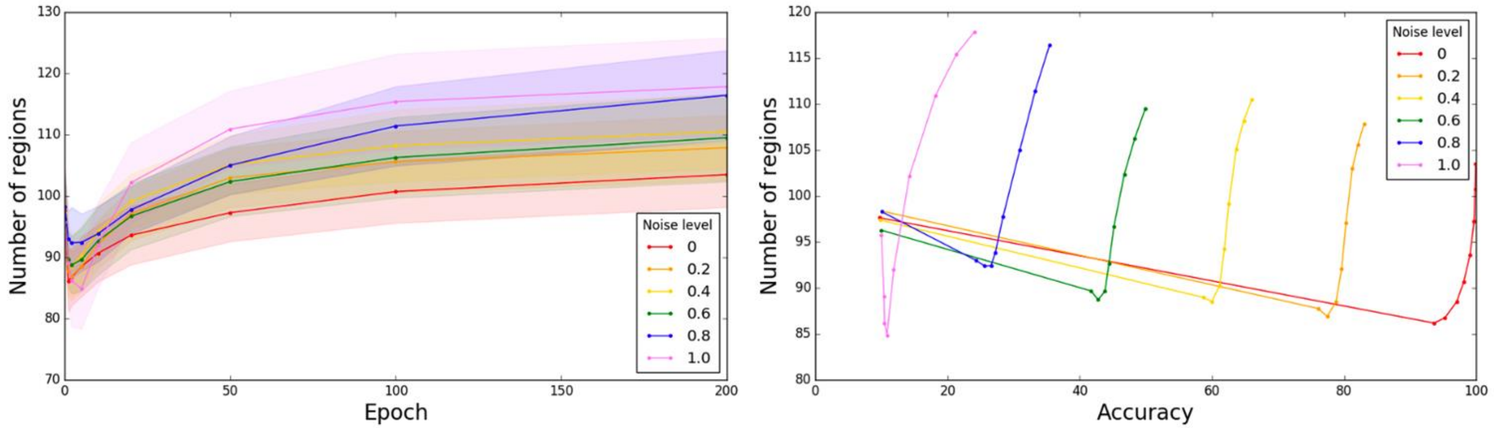
Complexity of linear regions [Hanin & Rolnick, 2019]

E.g. Average number of activation patterns in N over all of R^n is at most $(\#\text{neurons})^n/n!$, its value for depth 1 networks

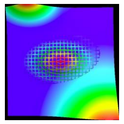


The average number of activation regions in a 2D cross-section of input space, for fully connected networks of various architectures training on MNIST. (so $(\#\text{neurons})^2/2$ in this case as n is 2)

Wait a minute: what happened with the number of activation regions throughout training?

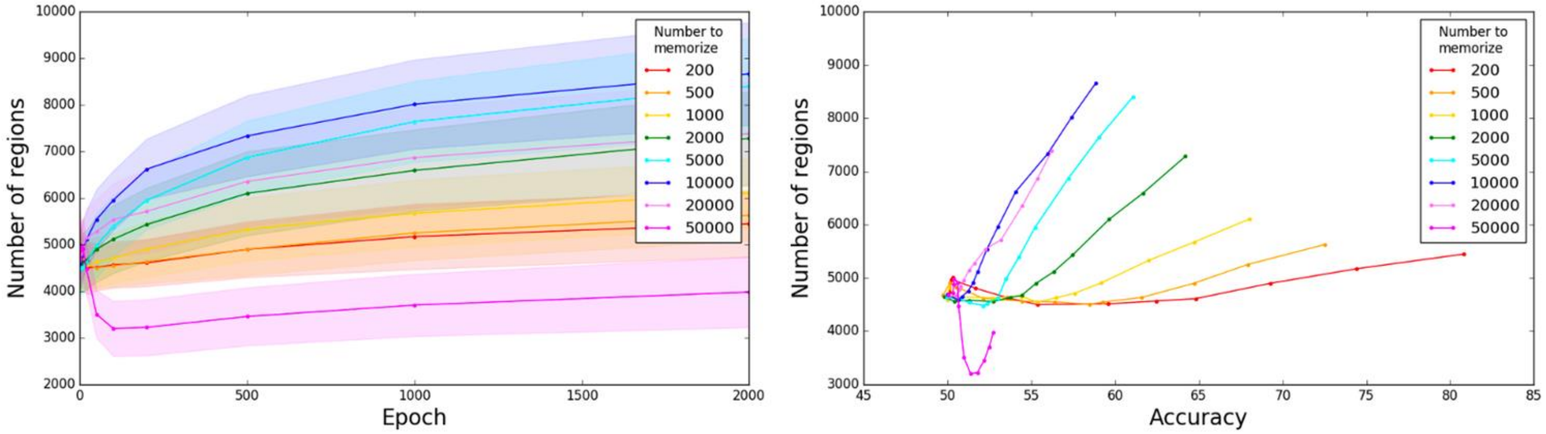


Effect of different noise levels on a [32,32,32] network, still MNIST. There are slightly, but not exponentially, more regions when memorizing more



Complexity of linear regions [Hanin & Rolnick, 2019]

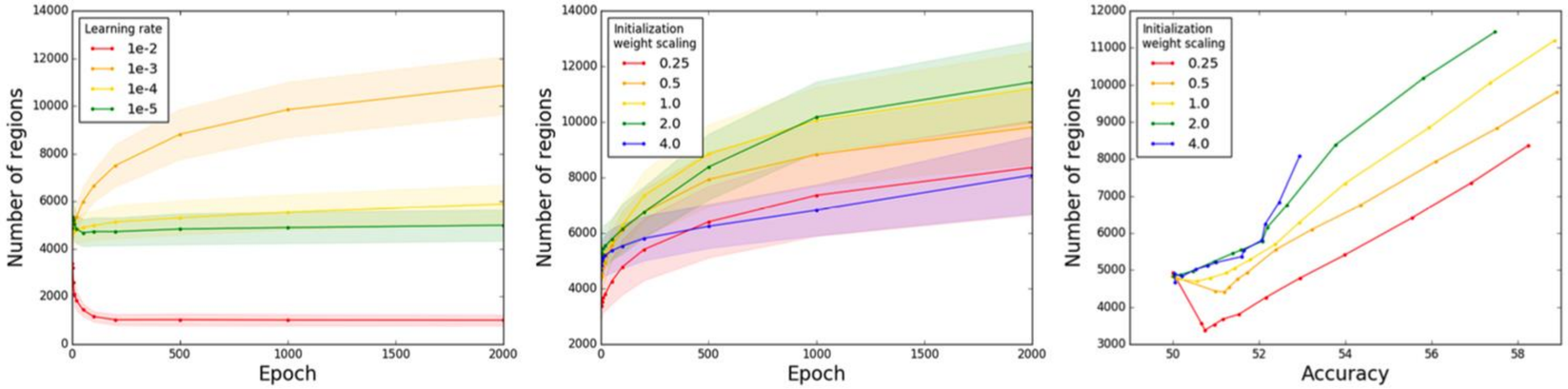
2D points with random binary labels -> memorization (predicted #act.regions: 4608).



The number of regions increased during training, and increased more for greater amounts of memorization.

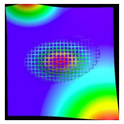
The exception for the maximum amount of memorization, where the network essentially failed to learn -> insufficient capacity?

5000 2D points with random binary labels (approx. 4608 regions), different weight scales and learning rates.



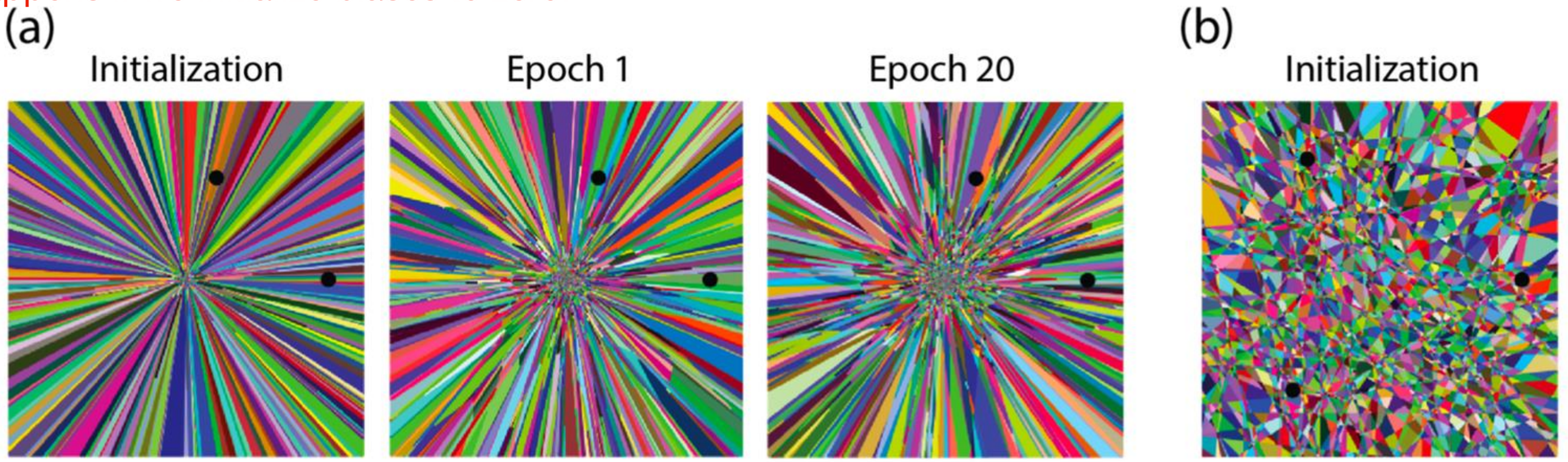
The number of activation regions after training increases slightly with increasing memorization, until the task becomes too hard for the network and training fails altogether.

Varying the learning rate yields slight increase -> hypothesis: the small increase in activation regions is probably not a result of hyperparameter choice.



Complexity of linear regions [Hanin & Rolnick, 2019]

What happens if we initialize biases to zero:

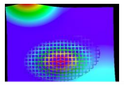


Conclusions:

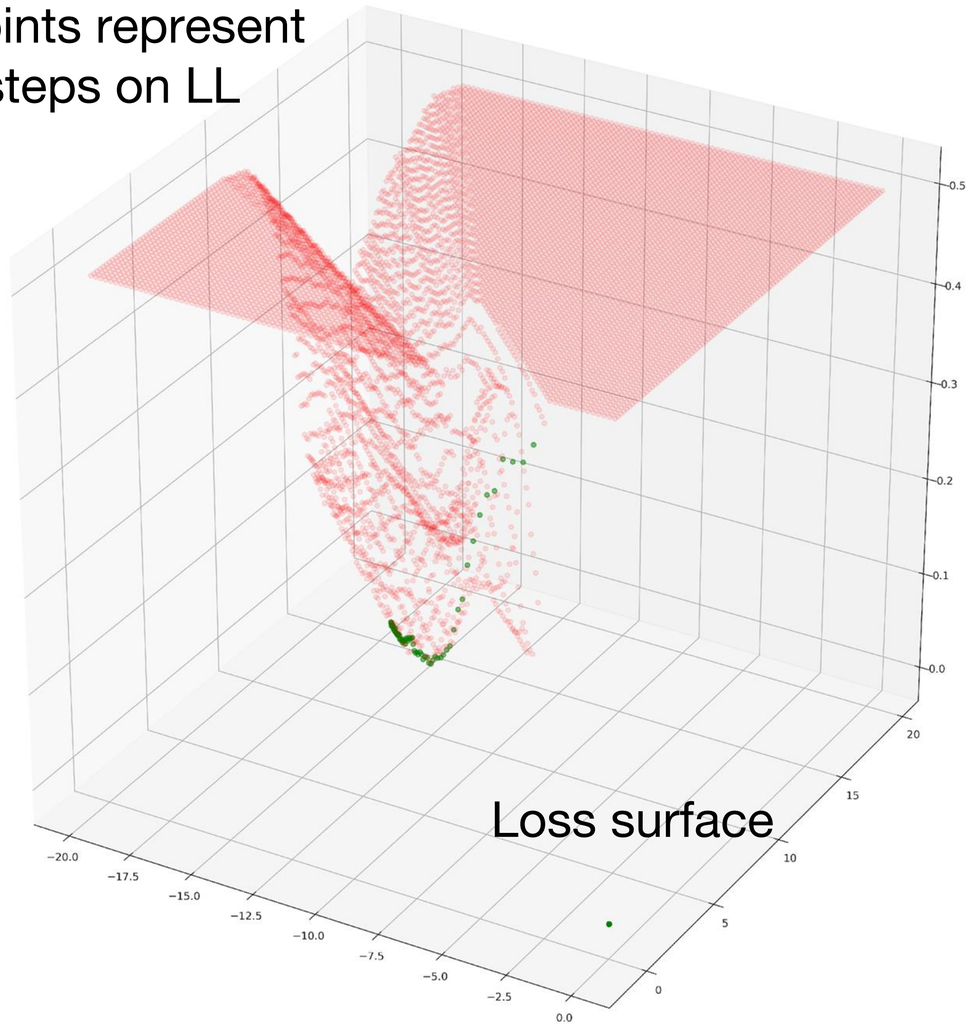
1. The number of activation regions learned in practice by a ReLU network is far from the maximum possible and depends mainly on the number of neurons in the network, rather than its depth.
2. If network gradients and biases are well-behaved (~ bounded biases and gradients), the partition of input space learned by a deep ReLU network is not significantly more complex than that of a shallow network with the same number of neurons.

Holds only for simple ReLU nets with no ties and biases, they believe that their results are true for residual and convolutional networks.

Tangent space sensitivity and the distribution of activation regions [D., 2020]



Green points represent SGD steps on LL



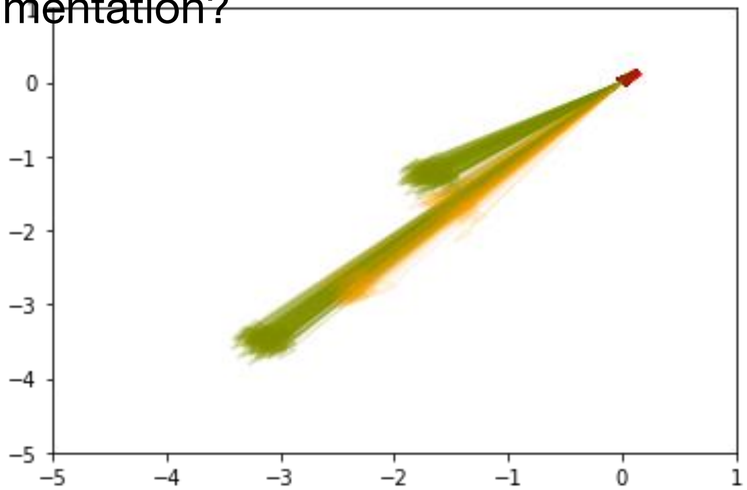
Loss surface

for simplicity here we fixed the bias (b) to a particular and good value

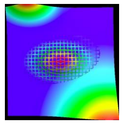
Hypothesis: stability of GD methods depends on the stability of tangent mapping

Tangent map and mini-batch learning-> a step inside a “random walk” is a result of a consensus of the tangent mapping of a small set. Can we trust it?

Let us assume Gaussian perturbations (smooth)
Adversarial case? Non-smooth augmentation?



Gradient vectors at step zero



Tangent space sensitivity and the distribution of activation regions

Effect of small perturbations on tangent mapping?

Gaussian perturbed example:

$$\|x - \phi(x)\|_p \leq \rho$$

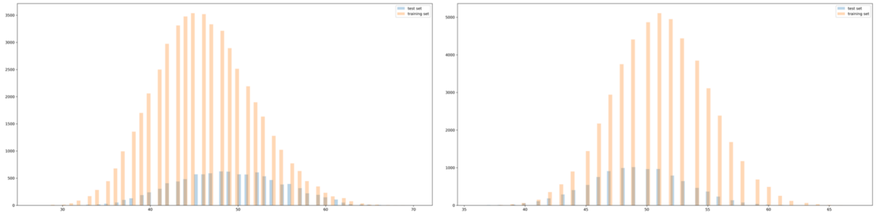
$$\delta(x) = x - \phi(x) \sim \mathcal{N}(0, c\mathbf{I})$$

Change in the tangent space:

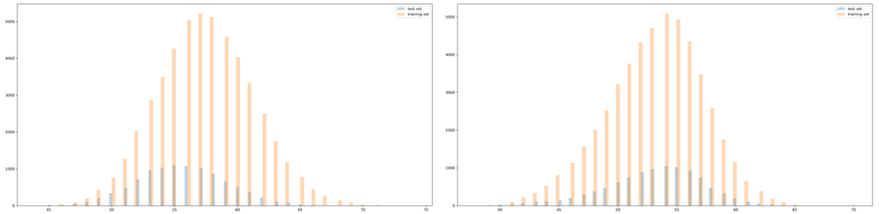
$$\begin{aligned}
&\mathbf{E}_{\delta(x)}[\|\nabla_\theta f(x;\theta) - \nabla_\theta f(\phi(x);\theta)\|_2^2] \\
&\sim \mathbf{E}_{\delta(x)}\left[\left\|\frac{\partial \nabla_\theta f(x;\theta)}{\partial x}\delta(x)\right\|_2^2\right] \\
&\leq c\left\|\frac{\partial \nabla_\theta f(x;\theta)}{\partial x}\right\|_2^2
\end{aligned}$$

Definition 3.1. *Tangent sample sensitivity* of a parametric, smooth feed-forward network f with output in $\mathbb{R}^{d_{out}}$ at input $x \in \mathbb{R}^{d_{in}}$ is a $N_\theta \times d_{in}$ dimensional matrix, $Sens_{tan}(x;\theta) := \frac{\nabla_\theta f(x;\theta)|_\theta}{\partial x}\Big|_x = \frac{\partial^2 f(x;\theta)}{\partial\theta\partial x}\Big|_{\theta,x}$. We define *tangent sensitivity* as the expectation of *tangent sample sensitivity*: $Sens_{tan}(\theta) = \mathbf{E}_{x\sim D}[Sens_{tan}(x;\theta)]$.

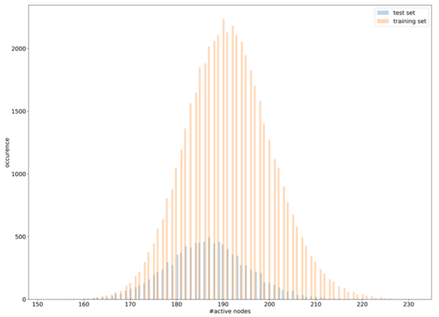
#active neurons per layer



(a) Layer-1 (b) Layer-2

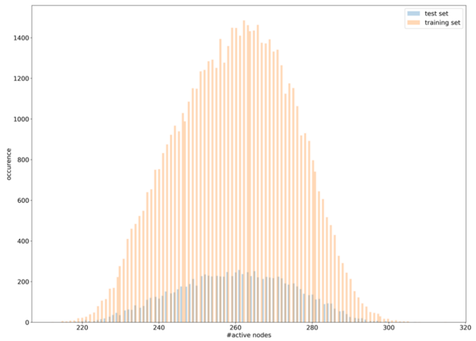


(c) Layer-3 (d) Layer-4

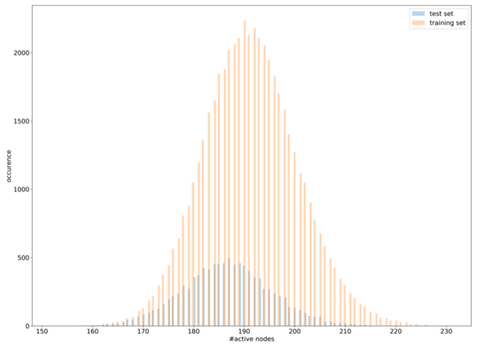


(e) All neurons

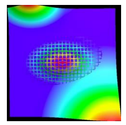
#active neurons



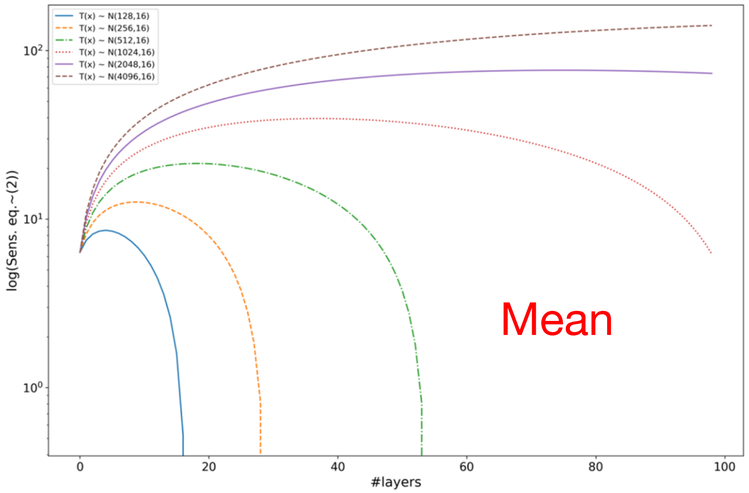
(a) MNIST [LeCun et al., 1998]



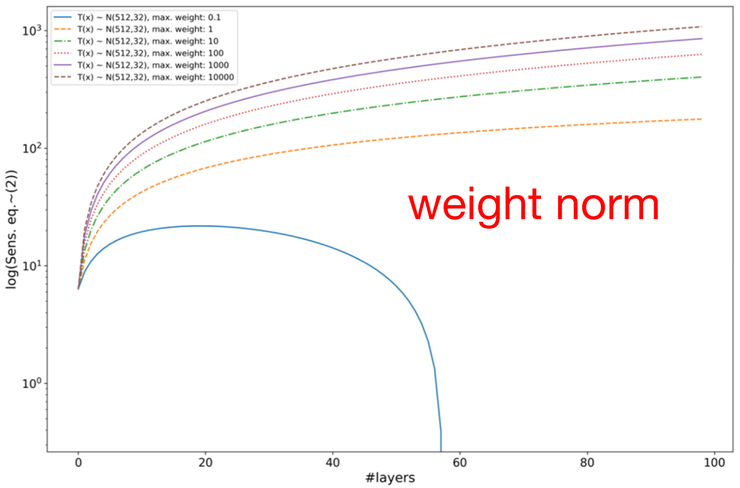
(b) CIFAR-10 [Krizhevsky et al., 2009]



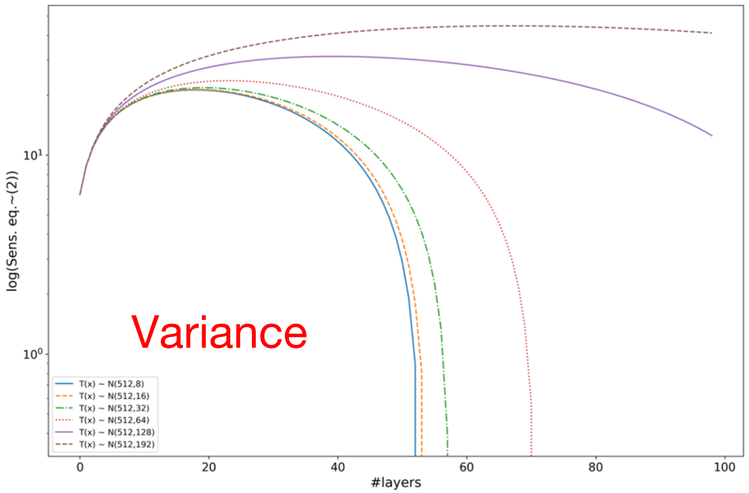
Tangent space sensitivity and the distribution of activation regions [D., 2020]



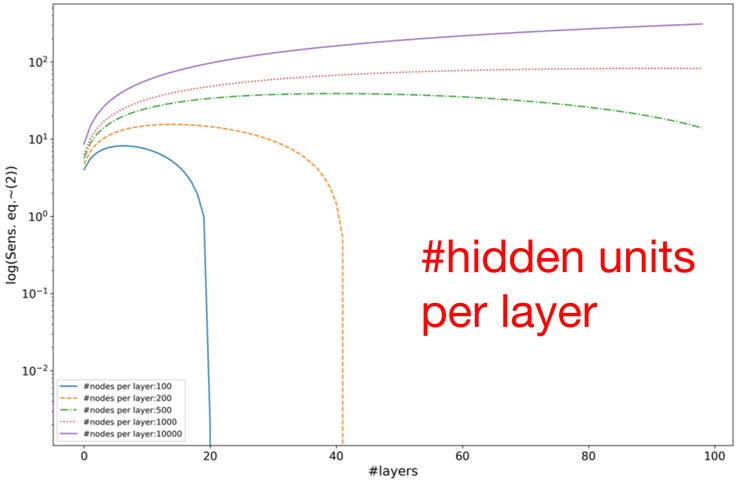
(a) $N_{layer} = 1k, w_{max} = 0.1$, log-scale



(c) $N_{layer} = 1k$, log-scale



(b) $N_{layer} = 1k, w_{max} = 0.1$, log-scale



(d) $\mu = 0.48N_\theta$, $\sigma = 0.04N_\theta$ (1b), log-scale

Tangent sensitivity and the effect of various network parameters and #layers

Proposition 3.2. For $x \sim D$ and a biasless feed-forward ReLU network with $w_{max_i} = \max_{w \in \theta_i} |w|$, with the number of active nodes $T(x)$ following a normal distribution $\mathcal{N}(\mu, \sigma)$, the Forbenius norm of *tangent sensitivity* is upper bound by

$$N_\theta d_{in} \sigma^{2(k-1)} \frac{2^{k-1}}{k^{2k}} \frac{(\Gamma(k/2))^2}{\Pi} \Psi^2 (\Pi_{i=1}^k w_{max_i})^2 \quad (1)$$

where $\Psi = \Psi(-(k-1)/2, 1/2, -\mu^2/(2\sigma^2))$ is Krummer's confluent hypergeometric function.

Lemma 3.1. For each element in an activation region $R(A;\theta)$ *tangent sample sensitivity* is identical.

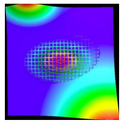
thus

$$\mathbf{E}_{x \sim D}[\|Sens_{tan}(x;\theta)\|_F^2] = \mathbf{E}_{A \sim p(A;x,\theta)}[\|Sens_{tan}(A;\theta)\|_F^2]$$

Distribution of activation regions (compact input space)

Q: Volume of convex polytopes? :(

see [Lovász & Simonovits, 1993] [Lovász & Vempala, 2006]



Conclusions and future work

We know much more than five or ten years ago about deep neural networks.

Evolution of linear regions could be the key.

Some ideas:

1. Tangent representations: rank of tangent mapping vs. representation learned by the network

$$Pr_{x,y\sim D}(\mathbb{E}_{x^*,y^*\sim D}[\nabla_\omega f(x)^T G_\omega \nabla_\omega f(x^*)|y=y^*] - \mathbb{E}_{x^*,y^*\sim D}[\nabla_\omega f(x)^T G_\omega \nabla_\omega f(x^*)|y\neq y^*] > \epsilon)$$

1. Graph limits: take the gradient graph, identify “representation sets” (regularity lemma)

   and check randomness

1. Pushforward, local diffeomorphisms

   Find lower or higher dimensional, but a sparser tangent space:

   i. dot product representation of the "gradient graph" (not planar)

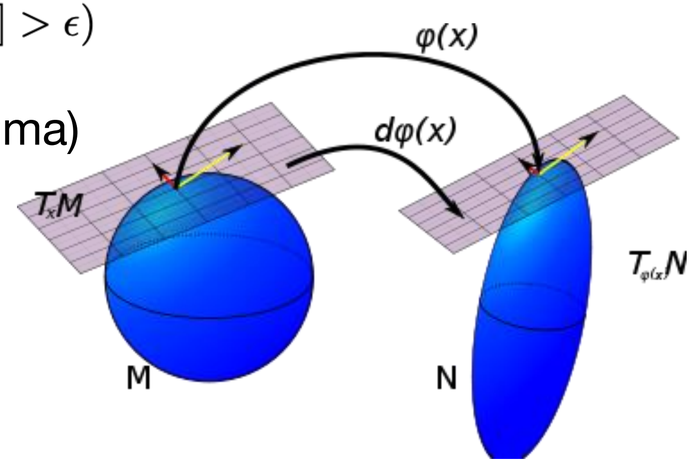
   ii. Johnson-Lindenstrauss theorem -> random orthogonal projection, independent of the dimension of the manifold

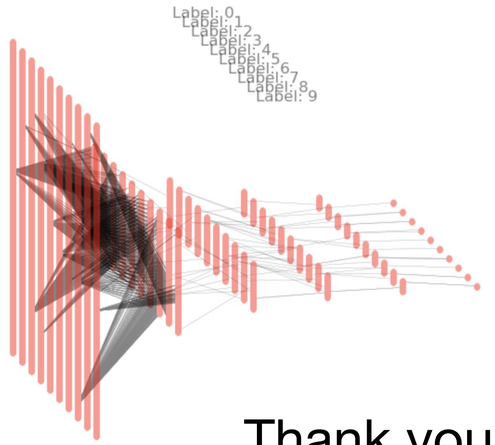
   non trivial network structures? Ensemble of structures.

1. Lie groups -> left/right/bi-invariant transformations

   [Myers & Strodeent, 1938, Rao and Ruderman, 1999, Hyland and R¨atsch, 2016, Wisdom et al., 2016, Ox et al., 2017]

   Some interesting results: [Pascanu & Bengio, 2014,Sra et al, 2018]





Thank you!