10707
Deep Learning: Spring 2020

Andrej Risteski

Machine Learning Department

Lecture 7:
Generalization in deep learning

Recap: definition of generalization

Generalization: difference in performance on train set vs all data.

The i.i.d. assumption: the samples in training set are identically, independently distributed.

Mathematically, there is an underlying data distribution $\mathcal{D}$. A new training sample is generated by drawing a sample from $\mathcal{D}$ independently from the previous ones.

"Ideal" (all data) loss is the expected loss over $\mathcal{D}$. We minimize expected loss over the training set. Hope: for (not too large training sets), training loss $\approx$ ideal loss.

Theoretical idealization, broken in many ways in practice:

Data is not "identically" distributed, e.g.:

Images (ala Imagenet) are scraped in slightly different ways.

Data has systemic bias (e.g. patients are tested based on symptoms they exhibit)

Data is result of interaction (e.g. agent interacting with environment in RL)

Recap: definition of generalization

Generalization: differ… …data.

The i.i.d. assumpt…
identically, indepe…

Mathematically, th… …training sample is generated by draw… …vious ones.

"Ideal" (all data) lo… …cted loss over the training set. Hope: …deal loss.

Theoretical idealiz…

Data is not "identi…

Images (ala Imagene…

Data has systemic bias (e.g. patients are tested based on symptoms they exhibit)

Data is result of interaction (e.g. agent interacting with environment in RL)

But!!
We **really** don't understand even this simple idealization.

Recap: definition of generalization

Mathematical formalization of generalization

For a classifier f, and loss l, the **generalization gap** is:

$$\left| \mathop{\mathbb{E}}_{(x,y):\text{random training sample}} l(f(x), y) - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} l(f(x), y) \right|$$

As shorthand, we denote this as

$$\widehat{\mathbb{E}}\, l(f(x), y) - \mathbb{E}\, l(f(x), y)$$

Good generalization: $\widehat{\mathbb{E}}\, l(f(x), y) \approx \mathbb{E}\, l(f(x), y)$

Bad generalization\overfitting: $\widehat{\mathbb{E}}\, l(f(x), y) \ll \mathbb{E}\, l(f(x), y)$

The classifier f "looks" much better on the training set

Classical view of generalization

Decoupled view of generalization and optimization: the generalization depends on the complexity of the predictor class considered.

Ignore the algorithm: to have a good generalization, make sure that the predictor class we are using is not too "complex".

Some popular strategies to do this:

Capacity control: bound the size of the network, bound the amount of connections, clip the weights during training, etc.

Regularization: add to the objective a penalization term for "complex" predictors, e.g. l_2 norm of weights of network

Classical view of generalization

Meta-"theorem" of generalization: with probability $1-\delta$ over the choice of a training set of size m, we have

$$\sup_{f\in\mathcal{F}} \left| \widehat{\mathbb{E}}\, l(f(x),y) - \mathbb{E}\, l(f(x),y) \right| \leq O\left(\sqrt{\frac{\text{complexity}(\mathcal{F})+\ln 1/\delta}{m}}\right)$$

Some measures of complexity:

(Effective) number of elements in $\mathcal{F}$

VC (Vapnik-Chervonenkis)

Rademacher complexity

Covering dimension

Basic version: Chernoff and union bound

Basic version (discrete classes): with probability $1-\delta$ over the choice of a training set of size m, for a loss l bdd. by 1, we have

$$\sup_{f\in\mathcal{F}} \left| \widehat{\mathbb{E}}\, l(f(x),y) - \mathbb{E}\, l(f(x),y) \right| \le O\left(\sqrt{\frac{\ln |\mathcal{F}| + \ln 1/\delta}{m}}\right)$$

Proof: (1): Write empirical loss as $\widehat{\mathbb{E}}\, l(f(x),y) = \frac{1}{m}\sum_i l(f(x_i),y_i)$

Consider a **fixed** classifier f. As this is an average of i.i.d random variables, each bounded by 1, the Chernoff bound applies. Recall:

Chernoff bound: Let X_i, $i \in [n]$ be i.i.d. random variables bounded by 1 with expectation μ. Then,

$$\Pr\left[\left|\frac{1}{m}\sum_i X_i - \mu\right| \ge \alpha\right] \le 2e^{-2\alpha^2 m}$$

Hence, apply **Chernoff** with $l(f(x_i),y_i)$ corresponding to X_i and **union bounding**:

$$\Pr_{\exists f}\left[\left|\widehat{\mathbb{E}}\, l(f(x),y) - \mathbb{E}\, l(f(x),y)\right| \gtrsim \sqrt{\frac{\ln |\mathcal{F}| + \ln 1/\delta}{m}}\right] \le \delta$$

Infinite classes?

Do we really need to consider *__every__* classifier in $\mathcal{F}$?
Intuitively, pattern of classifications on the training set should suffice.

(Two predictors that predict identically on the train set should generalize "identically")

Consider binary prediction (say, labels are $\{\pm 1\}$), with l being 0-1 loss.

Theorem (Vapnik-Chervonenkis): with probability $1-\delta$ over the choice of a training set of size m, we have

$$\sup_{f\in\mathcal{F}} \left| \widehat{\mathbb{E}}\, l(f(x),y) - \mathbb{E}\, l(f(x),y) \right| \leq O\left(\sqrt{\frac{\mathrm{VCdim}(\mathcal{F})\log m + \ln 1/\delta}{m}}\right)$$

VC dimension and related measures of complexity

Let $\mathcal{F} = \{f: \mathcal{X} \to \{\pm 1\}\}$ be a class of predictors.

Max # of possible label sequences

The **growth function** $\Pi_{\mathcal{F}}: \mathbb{N} \to \mathbb{N}$ of $\mathcal{F}$ is defined as

$$\Pi_{\mathcal{F}}(m) = \max_{(x_1, x_2, \dots x_m)} \left|\{\left(f(x_1), f(x_2), \dots, f(x_m)\right) \middle| f \in \mathcal{F}\}\right|$$

The **VC (Vapnis-Chervonenkis) dimension** of $\mathcal{F}$ is defined as

$$\mathrm{VCdim}(\mathcal{F}) = \max\{m: \Pi_{\mathcal{F}}(m) = 2^m\}$$

Equivalently: the largest m, s.t. $\mathcal{F}$ can **shatter** some set of size m:

that is, for **some** $(x_1, x_2, \dots x_m)$ and **any** labeling $(b_1, b_2, \dots b_m)$, $b_i \in \{\pm 1\}$,

some $f \in \mathcal{F}$ can produce that labeling, that is

$$(f(x_1), f(x_2), \dots f(x_m)) = (b_1, b_2, \dots b_m)$$

VC dimension and related measures of complexity

Let $\mathcal{F} = \{f: \mathcal{X} \to \{\pm 1\}\}$ be a class of predictors.

Max # of possible label sequences

The **growth function** $\Pi_{\mathcal{F}}: \mathbb{N} \to \mathbb{N}$ of $\mathcal{F}$ is defined as

$$\Pi_{\mathcal{F}}(m) = \max_{(x_1, x_2, \ldots x_m)} \left|\{\left(f(x_1), f(x_2), \ldots, f(x_m)\right) \middle| f \in \mathcal{F}\}\right|$$

The **VC (Vapnis-Chervonenkis) dimension** of $\mathcal{F}$ is defined as

$$\mathrm{VCdim}(\mathcal{F}) = \max\{m: \Pi_{\mathcal{F}}(m) = 2^m\}$$

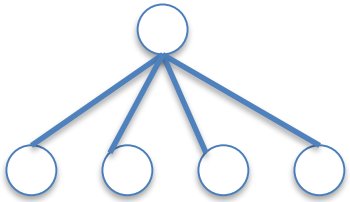
The two are closely related (**Sauer's lemma**):

$$\Pi_{\mathcal{F}}(m) = O(m^{\mathrm{VCdim}(\mathcal{F})})$$

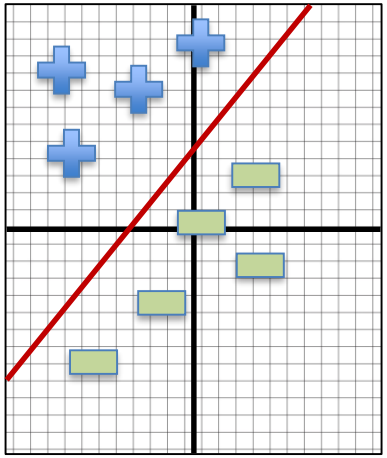
Bounding the VC dimension of a neural network

We'll prove a very simple bound on the VC dimension of a neural network with *__binary__* activation function. (Makes the proof the simplest).

$\text{sgn}(\langle w, x\rangle + b)$



(Each unit calculates a "hyperplane" $\text{sgn}(\langle w, x\rangle + b)$)



For starters, let's bound the VC dimension of a single unit, namely consider

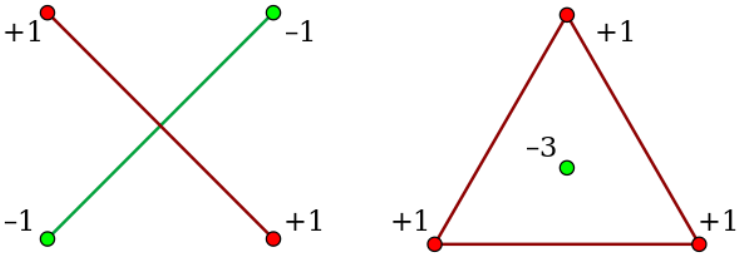
$$\mathcal{F} = \{\text{sgn}(\langle w, x\rangle + b) \,\big|\, w \in \mathbb{R}^d, b \in \mathbb{R}\}.$$

We'll show that $\text{VCdim}(\mathcal{F}) \leq d + 1$.

Bounding the VC dimension of a neural network

VCdim$(\mathcal{F}) \leq d + 1$: We want to that for every $(x_1, \ldots, x_{d+2})$, there exists a set of labels $(b_1, \ldots, b_{d+2})$ that cannot be linearly separated.

(*Radon's theorem*): For any $(x_1, \ldots, x_{d+2})$, there exists a partition of the points into sets S_1, S_2, s.t. convex hulls of S_1 and S_2 intersect.



How to use Radon's theorem: label pts in S_1, S_2 with +'s and –'s respectively.

Claim: no linear hyperplane perfectly separates S_1, S_2.

Pf: All pts in S_1 lie on one side of hyperplane, hence their convex hull does too.

All pts in S_2 lie on one side of hyperplane, hence their convex hull does too.

But, intersection is non-empty!

Bounding the VC dimension of a neural network

We will recursively use this to bound VC dimension of neural nets with binary activation function.

We will show: VC-dimension of **fully connected net** with **N** edges in total is O(**N log N**)

Main idea: growth function behaves nicely wrt to compositions and concatenations.

Bounding the VC dimension of a neural network

Claim 1: If $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2$ (**Cartesian product, i.e. concatenation**), we have $\Pi_{\mathcal{F}}(m) \leq \Pi_{\mathcal{F}_1}(m)\Pi_{\mathcal{F}_2}(m)$

Pf: Follows since any $\big(f(x_1), f(x_2), \dots, f(x_m)\big)$ can be written as

$$\Big(\big(f_1(x_1), f_2(x_1)\big), \big(f_1(x_2), f_2(x_2)\big), \dots, \big(f_1(x_m), f_2(x_m)\big)\Big).$$

Claim 2: If $\mathcal{F} = \mathcal{F}_1 \circ \mathcal{F}_2$ (**i.e. compositions**), we have $\Pi_{\mathcal{F}}(m) \leq \Pi_{\mathcal{F}_1}(m)\Pi_{\mathcal{F}_2}(m)$

Pf: $\big|\big\{\big(f(x_1), f(x_2), \dots, f(x_m)\big)\big|\, f \in \mathcal{F}\big\}\big|$ can be written as

$$\Big|\bigcup_{(y_1,\dots,y_m)\in\{(f_1(x_1),f_1(x_2),\dots,f_1(x_m))|\, f_1\in\mathcal{F}} \{(f_2(y_1), f_2(y_2), \dots, f_2(y_m))|f_2 \in \mathcal{F}_2\}\Big|$$

$$\leq \sum_{(y_1,\dots,y_m)\in\{(f_1(x_1),f_1(x_2),\dots,f_1(x_m))|\, f_1\in\mathcal{F}} |\{(f_2(y_1), f_2(y_2), \dots, f_2(y_m))|f_2 \in \mathcal{F}_2\}|$$

$$\leq \Pi_{\mathcal{F}_1}(m)\Pi_{\mathcal{F}_2}(m)$$

Simple bounds on VC dimension of neural networks

We write a neural net as a sequence of concatenations and compositions.

Let $\mathcal{F}_{ij}$ be the set of functions (as we vary the input weights) calculated at the jth node on the ith layer.

If we view the **ith layer** as a function, it can be written as the concatenation of the outputs of all the nodes, namely $\mathcal{F}_i = \mathcal{F}_{i1} \times \mathcal{F}_{i2} \times \cdots \times \mathcal{F}_{n_i}$

Furthermore, the **entire network** can be written as $\mathcal{F}_l \circ \mathcal{F}_{l-1} \circ \cdots \circ \mathcal{F}_1$

Using the lemmas on the previous slide, we have $\Pi_{\mathcal{F}}(m) \leq \Pi_{ij}\Pi_{\mathcal{F}_{ij}}(m)$

Note that each $\mathcal{F}_{ij}$ is a **hyperplane**, so by Sauer's lemma and the VC dimension bound we proved, we have $\Pi_{\mathcal{F}_{ij}}(m) \leq m^{d_{i-1}}$

Putting together, we have $\Pi_{\mathcal{F}}(m) \leq m^N$.(N is the # of params in net.)

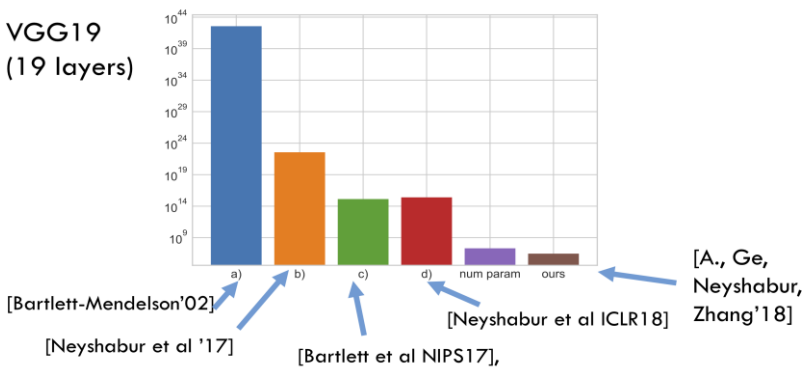
If a size m set is shattered: we have $2^m \leq m^N \Rightarrow$ m = O(N log N)

By definition of VC dimension, the proof follows.

Modern improvements

Previous bounds don't even depend on the **weights**! (Just on the # of parameters.) Clearly, misses the point when we overparametrize.

Recent works [*Bartlett-Foster-Telgarsky '18, Golowich-Rakhlin-Shamir '19, Barron-Klusowski '19*] prove "complexity" bounds depending on weights.



Typical quantities that appear are **products** of some **norms** of the layer matrices (e.g. spectral norms, as they capture the Lipschitz constant of the matrix).

The estimates on the generalization gap they give are abysmal.

In practice, the most typical regularizer is the square of the l_2 norm of weights (easy to evaluate and backprop through.)

Problems with the standard view I

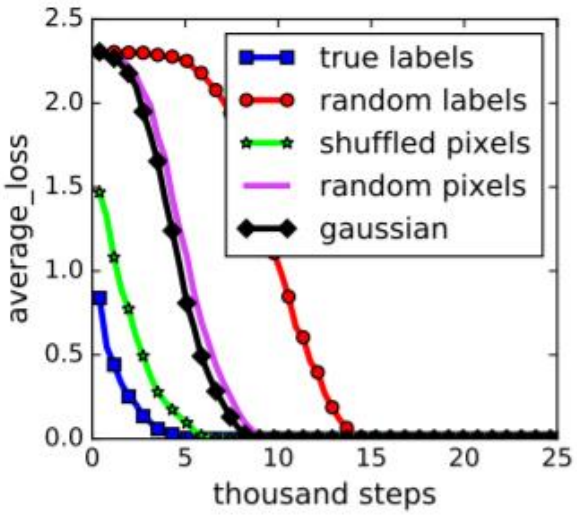
Modern models are **very** complex, by any measure of complexity.
(*Not uncommon*: # of params in neural net >> # training samples)

Observations in seminal paper of *Zhang-Bengio-Hardt-Recht-Vinyals '17:*

Modern architectures can fit **random noise!**

Can achieve 0 training error even if we:

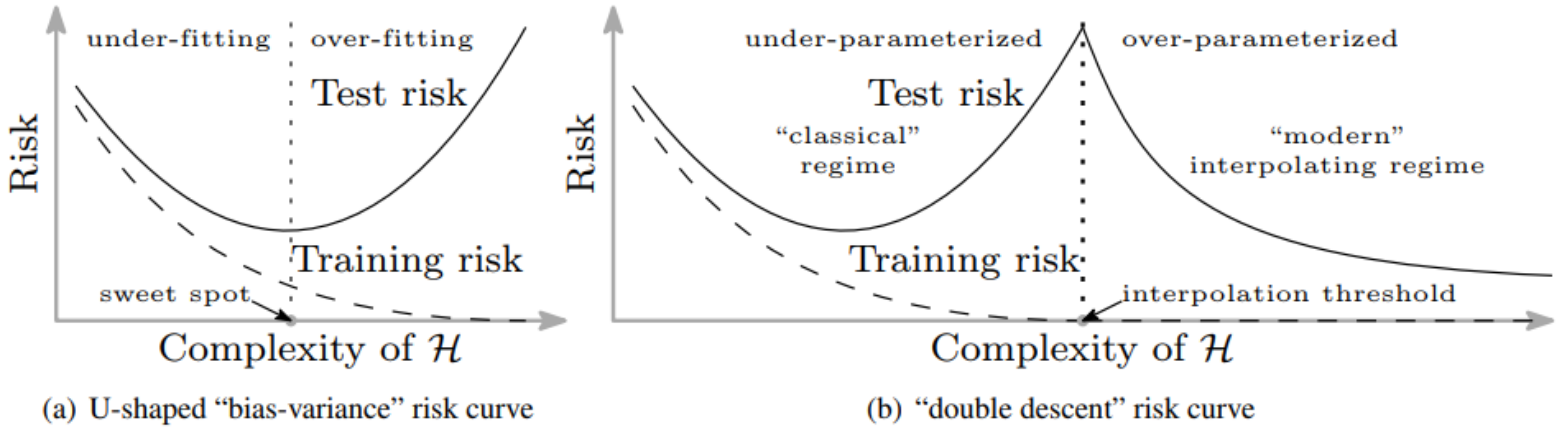
- Randomly permute pixels by a fixed permutation.
- Randomly permute pixels using a different permutation for each image.
- Replace the true labels by random labels;



(a) learning curves

Problems with the standard view II

Classical view of the "model complexity"/"over-fitting" curve is broken, as observed in a seminal paper by *Belkin-Hsu-Ma-Mandal '18*



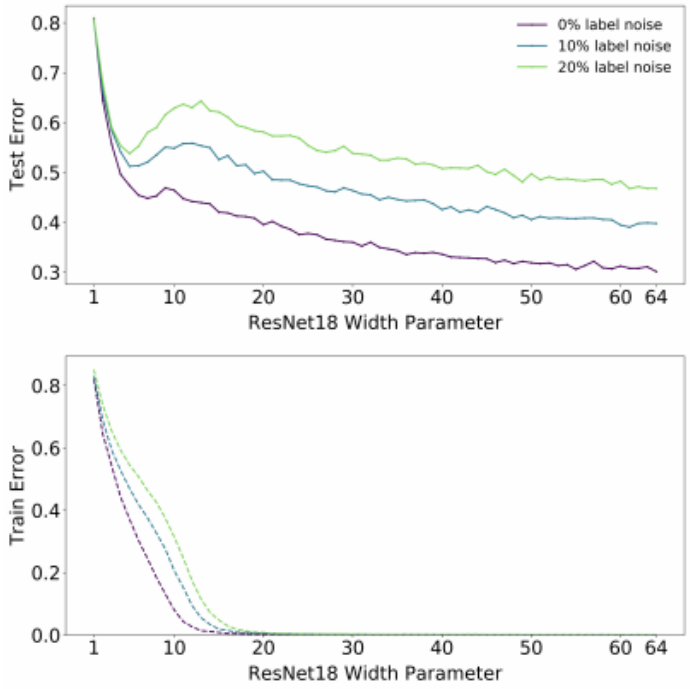
(a) U-shaped "bias-variance" risk curve

(b) "double descent" risk curve

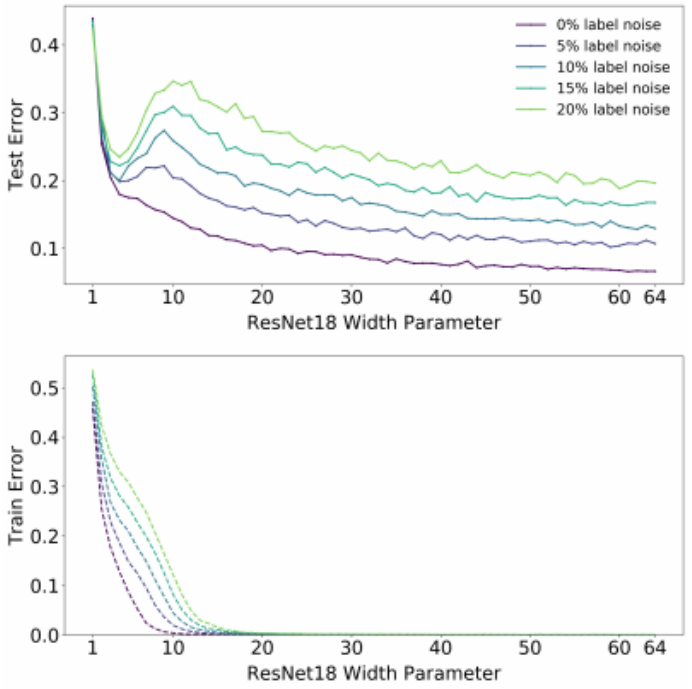
There are cases where the model gets bigger, yet the (test!) loss goes down, sometimes even lower than in the "underparametrized" regime.

Problems with the standard view II

Widespread phenomenon, across architectures, and **even other** quantities like train time, dataset size (Nakkiran et al '19):



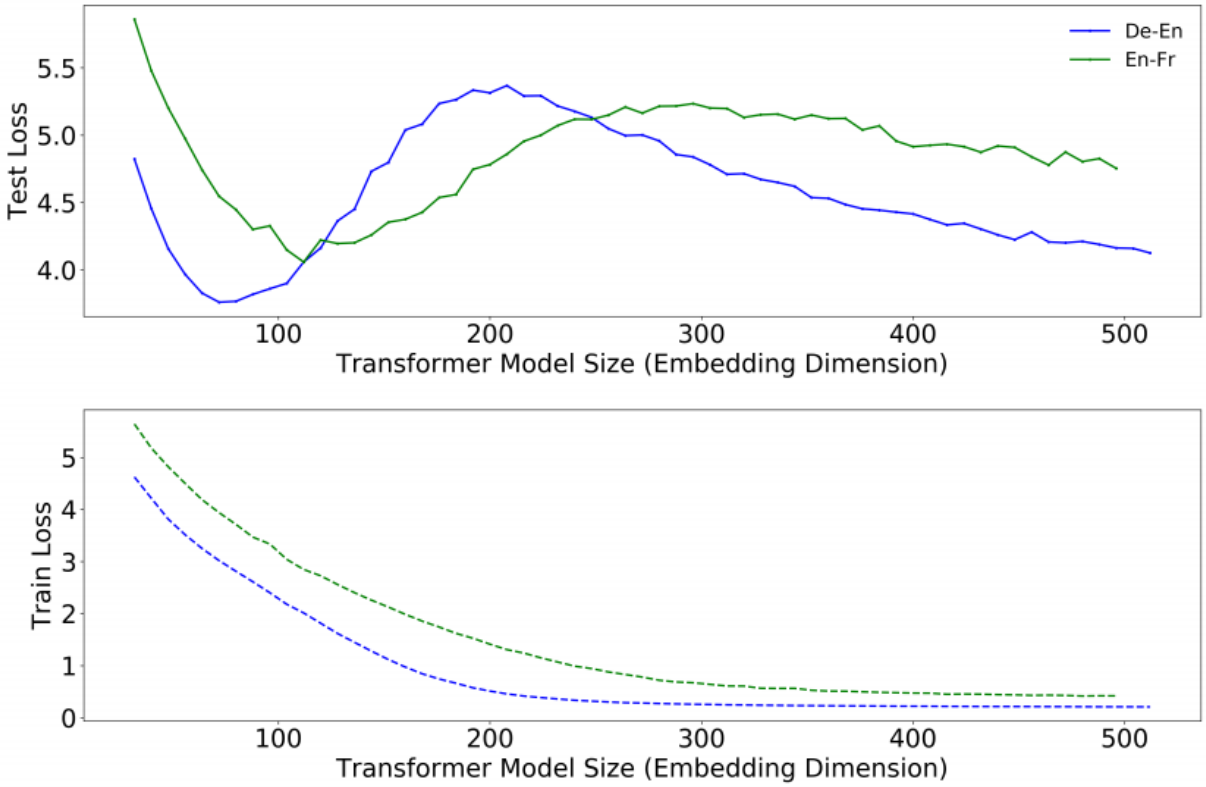
(a) **CIFAR-100.** There is a peak in test error even with no label noise.



(b) **CIFAR-10.** There is a "plateau" in test error around the interpolation point with no label noise, which develops into a peak for added label noise.

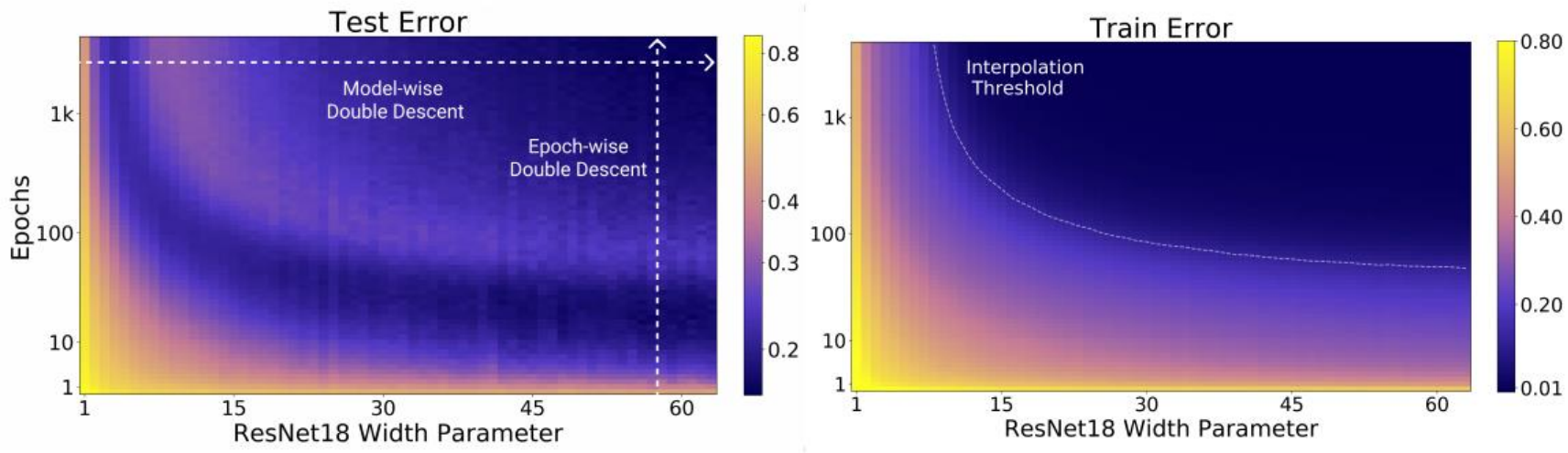
Problems with the standard view II

Widespread phenomenon, across architectures, and **even other** quantities like train time, dataset size (Nakkiran et al '19):



Problems with the standard view II

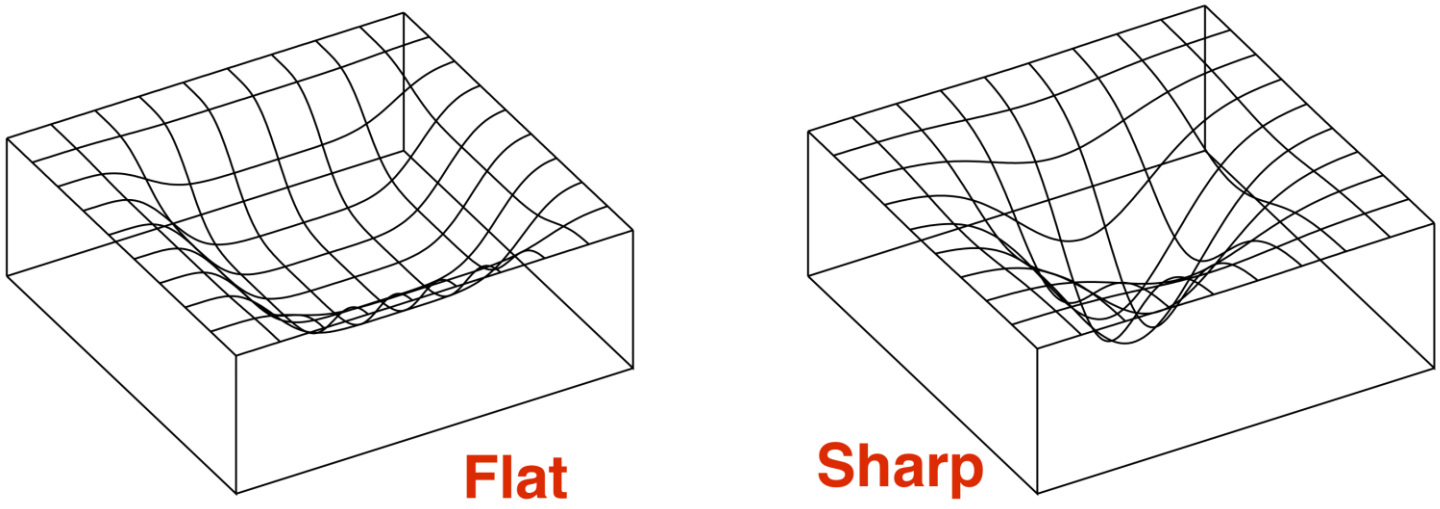
Widespread phenomenon, across architectures, and **even other** quantities like train time, dataset size (Nakkiran et al '19):



Train time manifests, but only for large enough models!

Flat vs sharp minima

The practical observation: “flat” minima generalize better.
(Hochreiter and Schmidhuber 1995, Keskar et al 2016, Hinton and Camp 1993)

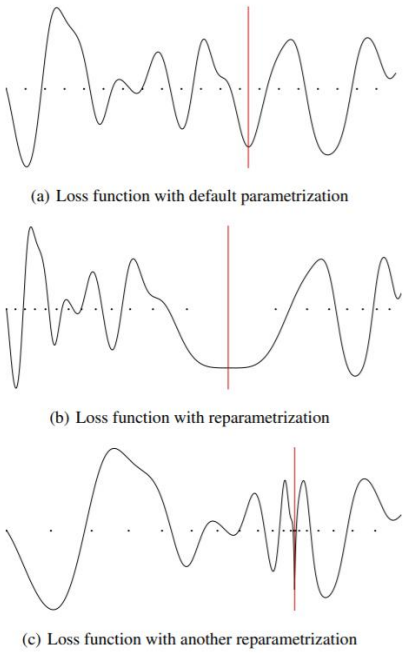


Original intuition (Hinton and Camp 1993): if the minimum occupies “volume” V, the number of distinct “minima regions” is at most total_volume/V.
Hence, the “cardinality” of the neural nets corresponding to flat minima is on the order of total_volume/V.

Flat vs sharp minima

How to measure flatness?

- **Volume** of nearby points where loss is approx. the same? (How to check??)
- **Curvature** of loss (i.e. magnitude of eigenvalues of Hessian)?
- **Stability** of loss wrt to random perturbations?



(a) Loss function with default parametrization

(b) Loss function with reparametrization

(c) Loss function with another reparametrization

Some positive results: the last interpretation has given some positive results (stability of loss wrt to Gaussian perturbations can guarantee some generalization guarantee).

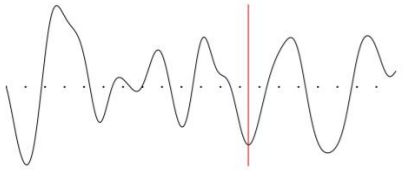
The formalization is the so-called PAC-Bayes framework.

Also some results when the networks are stable wrt to "dropping out" nodes.

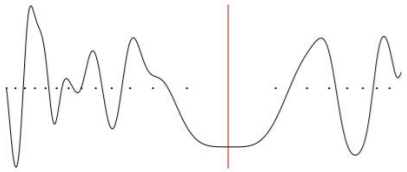
Flat vs sharp minima

How to measure flatness?

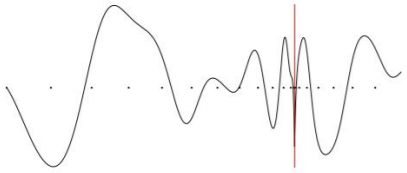
- **Volume** of nearby points where loss is approx. the same? (How to check??)
- **Curvature** of loss (i.e. magnitude of eigenvalues of Hessian)?
- **Stability** of loss wrt to random perturbations?



(a) Loss function with default parametrization



(b) Loss function with reparametrization



(c) Loss function with another reparametrization

Potential issues: multiple parameter settings might represent the same function. (Recall, neural nets have lots of invariances.)

For all these metrics, easy to choose different parameters, s.t. function is the same, but the "flatness" changes immensely. [Homework problem!]

There've been some attempts to come up with "reparametrization-invariant" measures, but gradient descent *is* sensitive to normal Euclidean metric.