Generalization Theory for Deep Learning

Basic version: finite hypothesis class

Finite hypothesis class: with probability $1-\delta$ over the choice of a training set of size n, for a bounded loss ℓ, we have

$$\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) - \mathbb{E}_{(x,y)\sim D} \left[\ell(f(x), y) \right] \right| = O\left(\sqrt{\frac{\log |\mathcal{F}| + \log 1/\delta}{n}} \right)$$

VC-Dimension

Motivation: Do we 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 training set should generalize similarly).

Let $\mathcal{F} = \{f : \mathbb{R}^d \to \{+1, -1\}\}$ be a class of binary classifiers.

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

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

The VC dimension of $\mathcal{F}$ is defined as:

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

VC-dimension Generalization bound

Theorem (Vapnik-Chervonenkis): with probability $1-\delta$ over the choice of a training set, for a bounded loss ℓ, we have

$$\sup_{f\in\mathcal{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i),y_i)-\mathbb{E}_{(x,y)\sim D}\left[\ell(f(x),y)\right]\right| = O\left(\sqrt{\frac{\mathrm{VCdim}(\mathcal{F})\log n+\log 1/\delta}{n}}\right)$$

Examples:

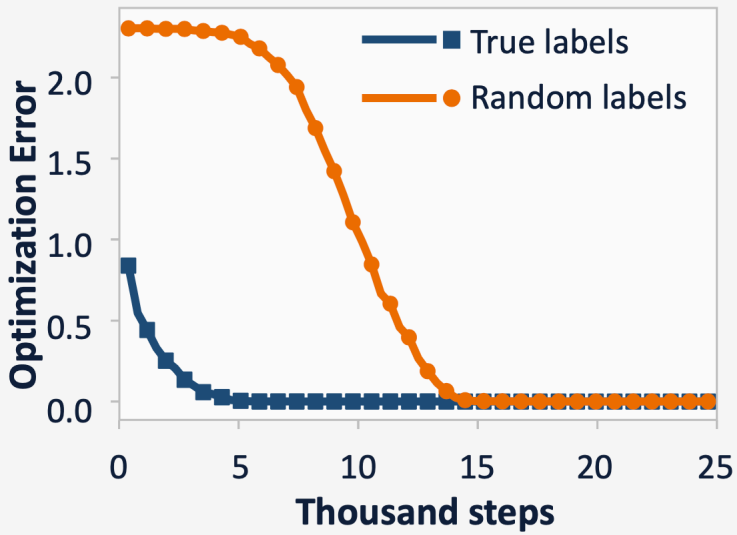
- Linear functions: VC-dim = O(dimension)
- Neural network: VC-dimension of fully-connected net with width W and H layers is $\widetilde{\Theta}(WH)$ (Bartlett et al., '17).

Problems with VC-dimension bound

1. In over-parameterized regime, bound >> 1.
2. Cannot explain the random noise phenomenon:
   - Neural networks that fit random labels and that fit true labels have the same VC-dimension.

Practice: gradient descent

$$\theta(t+1) \leftarrow \theta(t) - \eta \frac{\partial L(\theta(t))}{\partial \theta(t)}$$



Optimization error -> **0** for both ***true labels*** and ***random labels*** !

Zhang Bengio Hardt Recht Vinyals 2017
Understanding DL Requires Rethinking Generalization

PAC Bayesian Generalization Bounds

Setup: Let P be a prior over function in class $\mathcal{F}$, let Q be the posterior (after algorithm's training).

Theorem: with probability $1-\delta$ over the choice of a training set, for a bounded loss ℓ, we have

$$\sup_{f\in\mathcal{F}} \left| \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i) - \mathbb{E}_{(x,y)\sim D}\left[\ell(f(x), y)\right] \right| = O\left(\sqrt{\frac{KL(Q \,||\, P) + \log 1/\delta}{n}}\right)$$

data-dependent bound

$P(W_{ij}) \sim \mathcal{N}(0, 1)$

P does not depend on data

Rademacher Complexity

Intuition: how well can a classifier class **fit random noise?**

(Empirical) **Rademacher complexity:** For a training set $S = \{x_1, x_2, \ldots, x_n\}$, and a class $\mathcal{F}$, denote:

$$\hat{R}_n(S) = \mathbb{E}_\sigma \sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \sigma_i f(x_i) \, .$$

where $\sigma_i \sim \text{Unif}\{+1, -1\}$ (Rademacher R.V.).

(Population) **Rademacher complexity:**

$$R_n = \mathbb{E}_S \left[\hat{R}_n(s) \right].$$

Rademacher Complexity Generalization Bound

Theorem: with probability $1 - \delta$ over the choice of a training set, for a bounded loss ℓ, we have

$$\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) - \mathbb{E}_{(x,y)\sim D} \left[\ell(f(x), y) \right] \right| = O\left(\frac{\hat{R}_n}{n} + \sqrt{\frac{\log 1/\delta}{n}} \right)$$

and

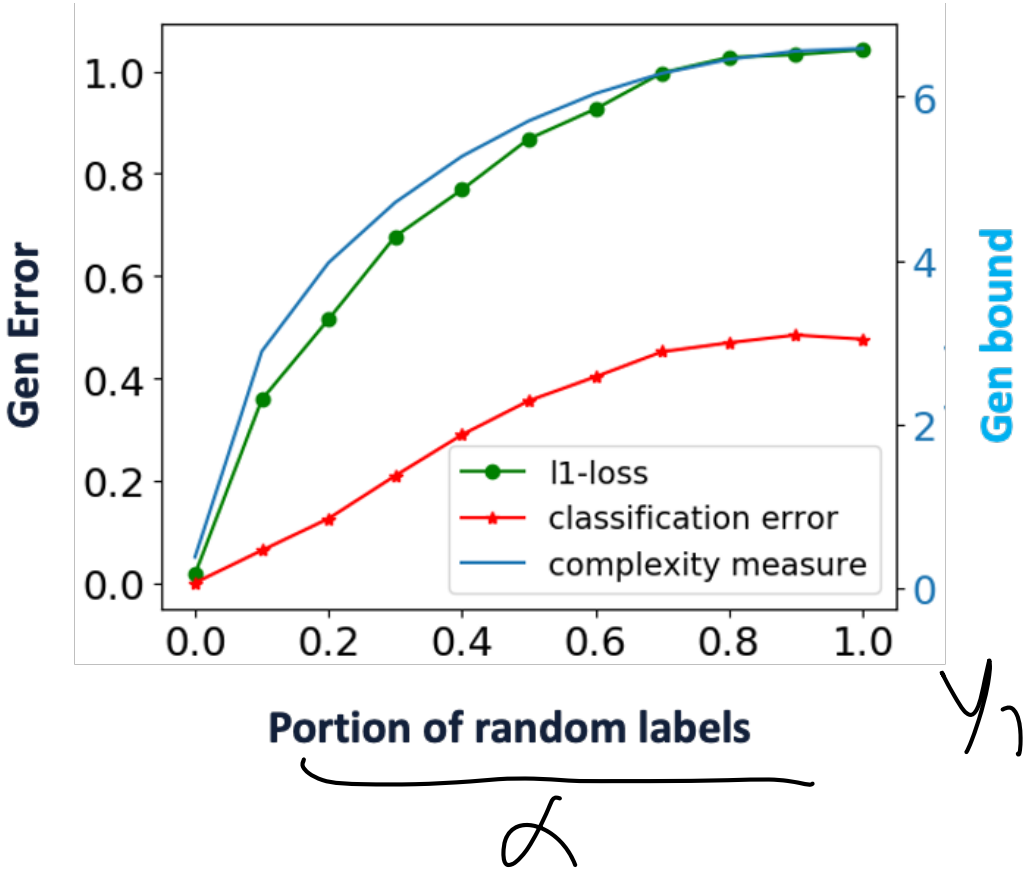
$$\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) - \mathbb{E}_{(x,y)\sim D} \left[\ell(f(x), y) \right] \right| = O\left(\frac{R_n}{n} + \sqrt{\frac{\log 1/\delta}{n}} \right)$$

Kernel generalization bound

data-dependent

H^*, y

Use Rademacher complexity theory, we can obtain a generalization bound $O(\sqrt{y^\top (H^*)^{-1} y / n})$ where $y \in \mathbb{R}^n$ are n labels, and $H^* \in \mathbb{R}^{n \times n}$ is the kernel (e.g., NTK) matrix.



α

$y_j = (1-\alpha) \cdot y_j^{true} + \alpha \cdot noise$

Norm-based Rademacher complexity bound

ReLU: $\rho = 1$

Theorem: If the activation function is σ is ρ-Lipschitz. Let

$$\mathcal{F} = \{x \mapsto W_{H+1}\sigma(W_h\sigma(\cdots\sigma(W_1x)\cdots), \|W_h^T\|_{1,\infty} \leq B \forall h \in [H]\}$$

S: training data

then $R_n(\mathcal{S}) \leq \|X^\top\|_{2,\infty}(2\rho B)^{H+1}\sqrt{2\ln d}$ where

$X = [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$ is the input data matrix.

$$\|X^T\|_{2,\infty} = \max_{i=1,\ldots,d} \|X_i'\|_2$$

$$\|W_h^T\|_{1,\infty} = \max_{\text{row}} \|W_h'^{\text{row}}\|_1$$

pf: induction layer by layer

Comments on generalization bounds

- When plugged in real values, the bounds are rarely non-trivial (i.e., smaller than 1)
- "*Fantastic Generalization Measures and Where to Find them*" by Jiang et al. '19 : large-scale investigation of the correlation of extant generalization measures with true generalization.

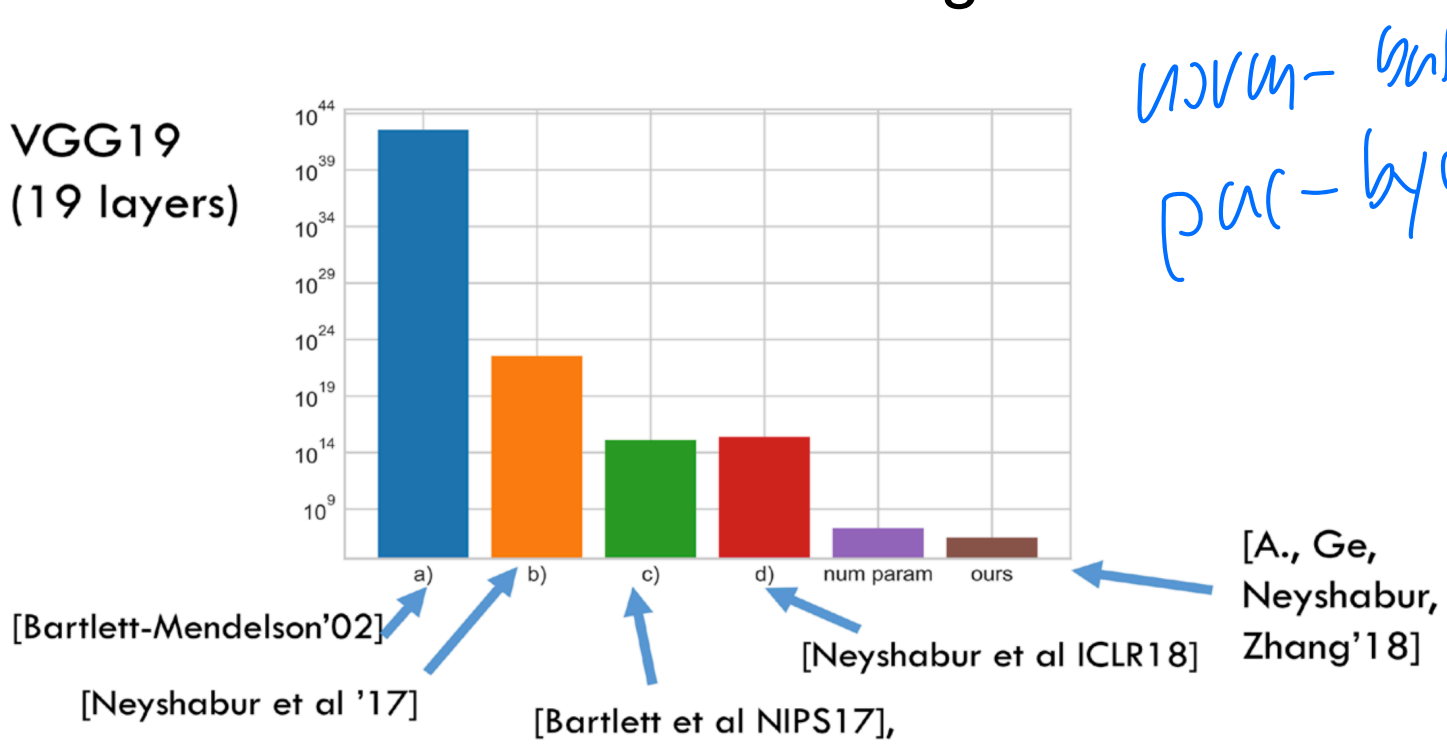


Image credits to Andrej Risteski

Comments on generalization bounds

- Uniform convergence may be unable to explain generalization of deep learning [Nagarajan and Kolter, '19]
  - Uniform convergence: a bound for all $f \in \mathcal{F}$
  - Exists example that 1) can generalize, 2) uniform convergence fails.

non-uniform: linear reg ression

bias^2 + Variance

- Rates:
  - Most bounds: $1/\sqrt{n}$.
  - Local Rademacher complexity: $1/n$.

Separation between NN and kernel

- For approximation and optimization, neural network has no advantage over kernel. Why NN gives better performance: generalization.

- [Allen-Zhu and Li '20] Construct a class of functions $\mathcal{F}$ such that $y = f(x)$ for some $f \in \mathcal{F}$:
  - no kernel is sample-efficient; need exponential # of data
  - Exists a neural network that is sample-efficient. polynomial samples

Separation between NN and kernel

Def: Kernel method is a linear method with an embedding $\phi: \mathbb{R}^d \to \mathcal{H}$, Hilbert space

$\Rightarrow$ it turns an element $f \in \mathcal{H}$ into a prediction function $y = \langle f, \phi(x) \rangle \triangleq f(x)$

The method uses samples $\{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^d$ observes $\{y_i\}_{i=1}^n$

$$f \in \text{span}\left(\phi(x_i) \right)_{i=1}^n, \quad i \in [n]$$

e.g. $$\underset{f}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n (y_i - \langle \phi(x_i), f \rangle)^2 + \lambda \|f\|^2$$

Separation between NN and kernel

Thm: $\exists$ a class of functions $C \subseteq \{c : \mathbb{R}^d \to \mathbb{R}\}$ and a distribution over $\mathbb{R}^d$ s.t.

i) $\forall$ kernel method, $\forall c \in C$, given $y_i = c(x_i)$ if $\mathbb{E}_{x \sim \mu}[(c(x) - \langle f, \phi(x) \rangle)^2] \leq \frac{1}{q}$ then $q \geq 2^{d-1}$ exponential

ii) $\exists$ simple procedure, that can output the true c as long as $n \geq d$

$\star$ the procedure can be simulated by a neural network (of)

Separation between NN and kernel

Pf: $\mathcal{U}$: uniform distribution over $\{-1,1\}^d$

2^d element

$$\mathcal{C} = \{C_S = \prod_{s \in S} x_s \;,\; S \subseteq \{1,\dots,d\}\}$$

Pf of (Part ii) choose a basis $\underbrace{\begin{pmatrix}1\\-1\\\vdots\\1\end{pmatrix}}_{e_1}, \underbrace{\begin{pmatrix}1\\1\\\vdots\\1\end{pmatrix}}_{e_2}, \dots, \underbrace{\begin{pmatrix}1\\1\\\vdots\\-1\end{pmatrix}}_{e_d}$ 2^d

$\Rightarrow y_i = C_S(e_i)$, if $i \in S$, $y_i = -1$

$i \notin S$, $y_i = 1$

$\Rightarrow$ know whether i is in S or not

$\Rightarrow$ identify $S \Rightarrow$ learn C_S □

Separation between NN and kernel

Part i) $\mathcal{C}$ is a basis for $\{f : \{-1,1\}^d \to \mathbb{R}\}$ with distribution μ.

$$\mathbb{E}_{x\sim\mu}\left[C_S(x)\cdot C_{S'}(x)\right] = \begin{cases} 0 & \text{if } S \neq S' \\ 1 & \text{if } S = S' \end{cases}$$

Goal: $\mathbb{E}_{x\sim\mu}\left[\left(C_S^*(x) - \langle f, \phi(x)\rangle\right)^2\right]$

1) since $f \in \text{span}\left(\phi(x_i)\right)_{i=1}^n$

$$f = \sum_{i=1}^n a_i \phi(x_i), \quad f(x) = \sum_{i=1}^n a_i \langle \phi(x_i), \phi(x)\rangle$$

$$x \mapsto \langle \phi(x_i), \phi(x)\rangle$$

$$= \sum_{S\subseteq[d]} \lambda_{i,S}\, C_S(x)$$

Separation between NN and kernel

$$\mathbb{E}_{x\sim\mu}\left[\left(C_{S^*}(x) - \langle f, \phi(x)\rangle\right)^2\right]$$

$$= \mathbb{E}_{x\sim\mu}\left[\left(f^*(x) - \sum_{S\subseteq[d]}\sum_{i=1}^{n} a_i \cdot \lambda_{i,S}\, C_S(x)\right)^2\right]$$

$$= \left(1 - \sum_{i=1}^{n} a_i \lambda_{i,S^*}\right)^2 + \sum_{S\neq S^*}\left(\sum_{i=1}^{n} a_i \lambda_{i,S}\right)^2$$

by assumption error $\leq \frac{1}{q}$

$$\Rightarrow \left(1 - \sum_{i=1}^{n} a_i \lambda_{i,S^*}\right)^2 \leq \frac{1}{q}$$

$$\& \sum_{S\neq S^*}\left(\sum_{i=1}^{n} a_i \lambda_{i,S}\right)^2 \leq \frac{1}{q}$$

Separation between NN and kernel

Notations

$\Lambda : 2^d \times n$

$\Lambda_{S,i} = \lambda_{i,S}$

$A = n \times 2^d$

$A_{i,S^*} = a_{i,S^*}$

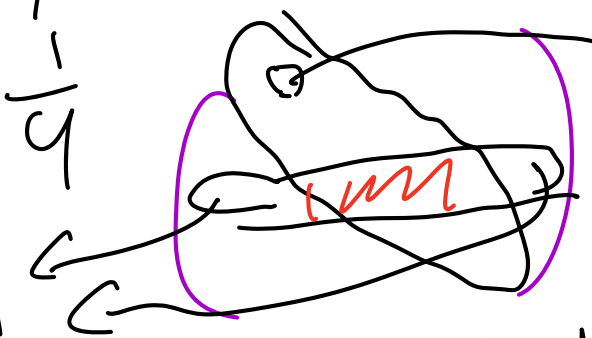
$\Omega_{S^*,S^*} = \frac{n}{2} \lambda_{i,S} \cdot a_{i,S^*}$

$\Omega = \Lambda A : 2^d \times 2^d$ of rank-n

$(1 - \Omega_{S^*,S^*})^2 \le \frac{1}{q} \rightarrow \Omega_{S^*,S^*} \ge \frac{4}{q}$

$\sum_{S \neq S^*} \Omega^2_{S,S^*} \le \frac{1}{q}$

$\le \frac{1}{q}$



$\Rightarrow$ rank at least $\frac{3}{4} \cdot 2^d$

Separation between NN and kernel

$\Omega = \mathrm{diag}(\Omega) + \Omega'$, Ω' : off-diagonal

$\|\Omega'\|_F^2 = \frac{2^d}{4} = \sum \mathrm{eigen}^2(\Omega)$

$\Rightarrow$ Ω' has at most $\frac{2^d}{4}$ eigenvalues $\geq \frac{2}{3}$

$\Rightarrow$ consider subspace with eigenvalues $\geq \frac{2}{3}$ which has dimension at least $\frac{3}{4} \cdot 2^d$

$\forall x \in$ subspace

$\|\Omega x\|_2 = \|\mathrm{diag}(\Omega) x + \Omega' x\|_2$

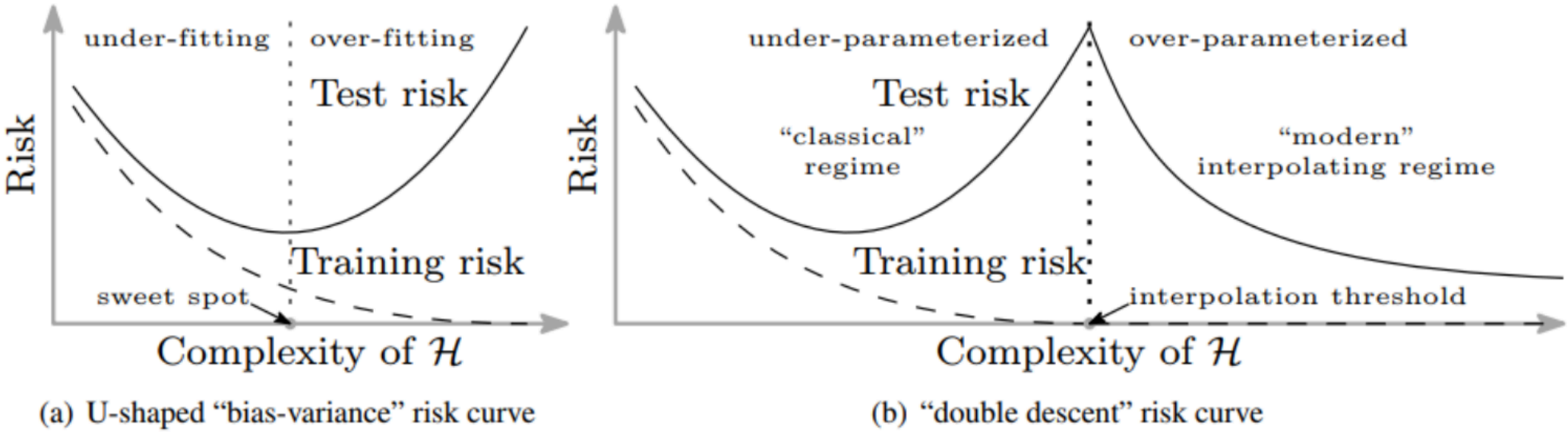
$\geq \|\mathrm{diag}(\Omega) x\|_2 - \|\Omega' x\|_2$

$\geq \frac{2}{3}\|x\| - \frac{2}{3}\|x\| = 0$

$\Rightarrow x \in \mathrm{span}(\Omega)$

$\Rightarrow \mathrm{span}(\Omega)$ dim $\geq \frac{3}{4} \cdot 2^d$

Double descent



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

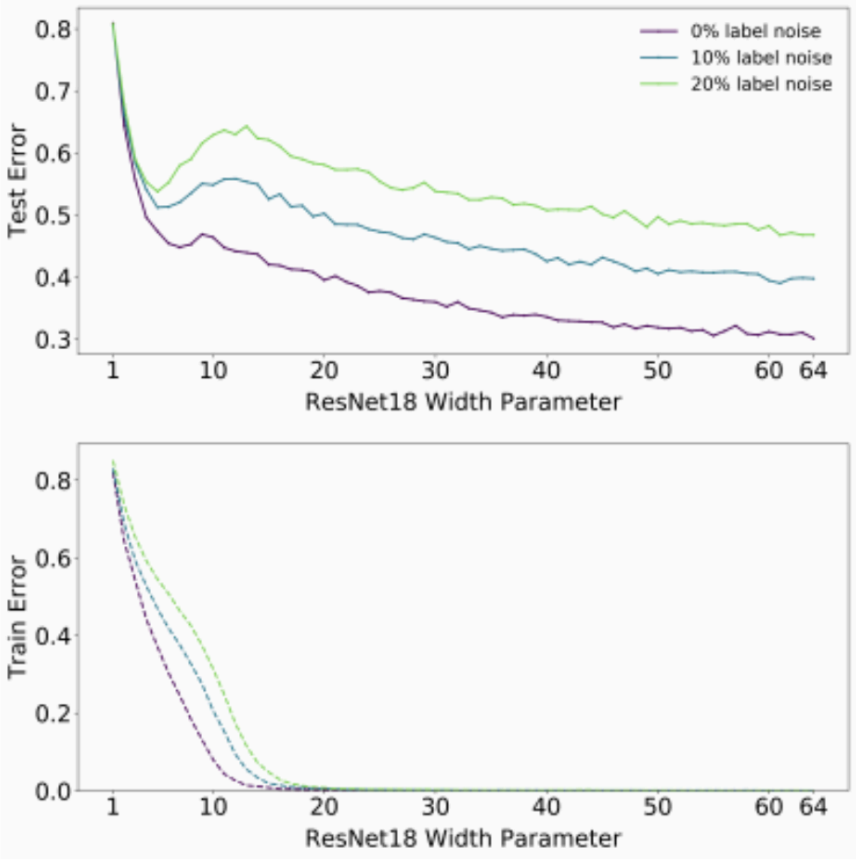
(b) "double descent" risk curve

Belkin, Hsu, Ma, Mandal '18

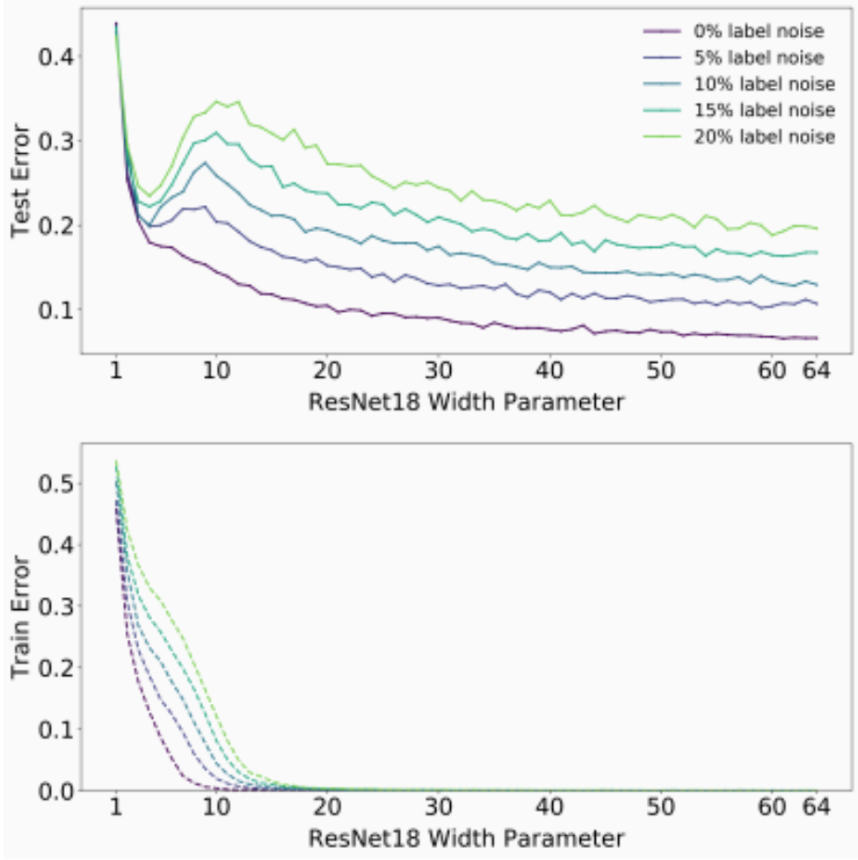
- There are cases where the model gets bigger, yet the (test!) loss goes down, sometimes even lower than in the classical "under-parameterized" regime.
- Complexity: number of parameters.

Double descent

Widespread phenomenon, across architectures (Nakkiran et al. ’19):



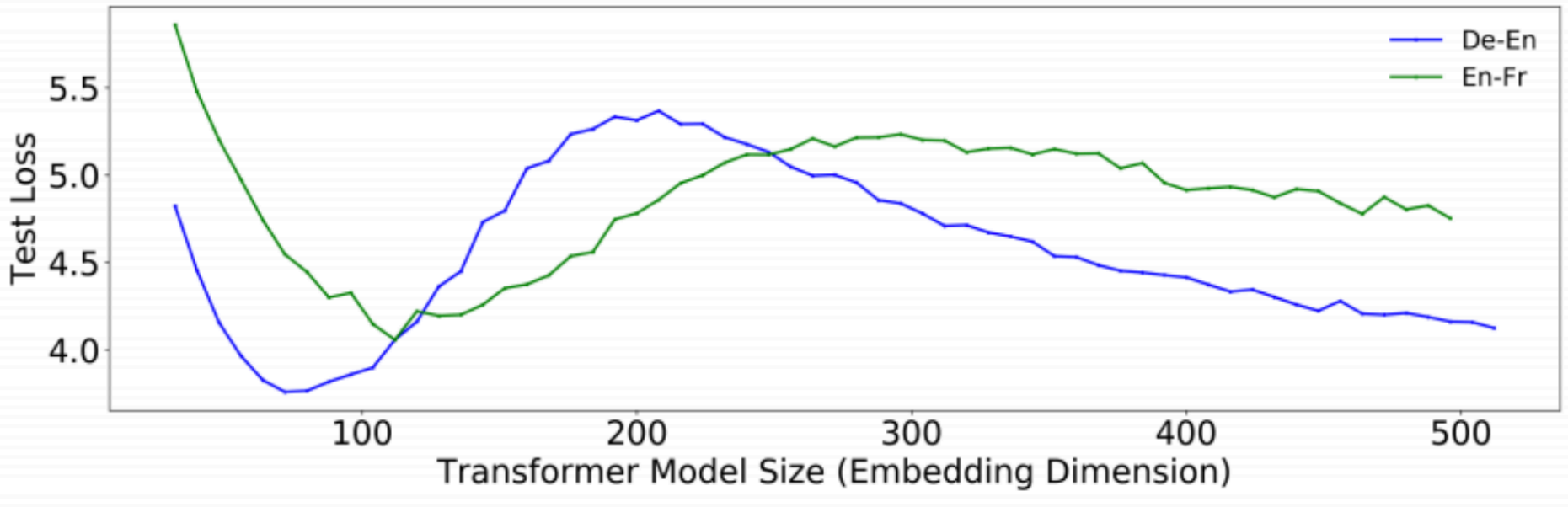
(a) **CIFAR-100.** There is a peak in test error even with no label noise.



(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.

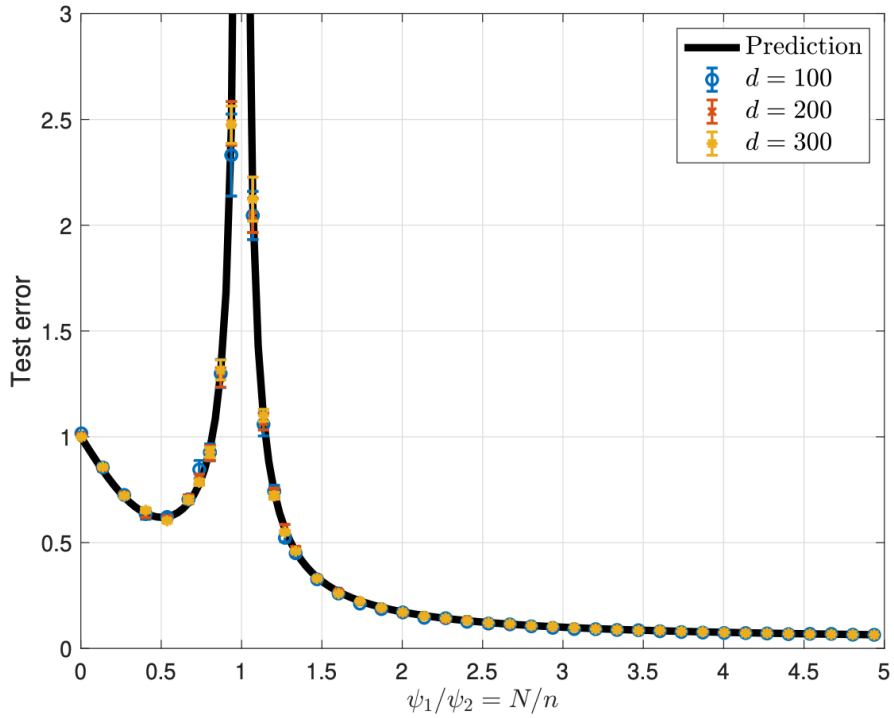
Double descent

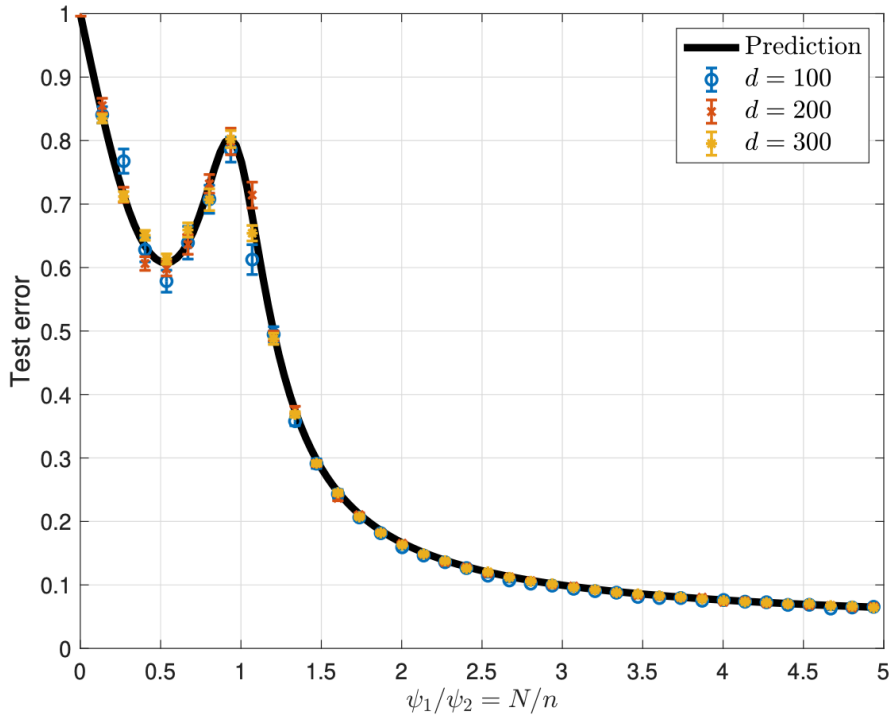
Widespread phenomenon, across architectures (Nakkiran et al. ’19):



Double descent

Widespread phenomenon, also in kernels (can be formally proved in some concrete settings [Mei and Montanari ’20]), random forests, etc.





Double descent

Also in other quantities such as train time, dataset, etc (Nakkiran et al. ’19):

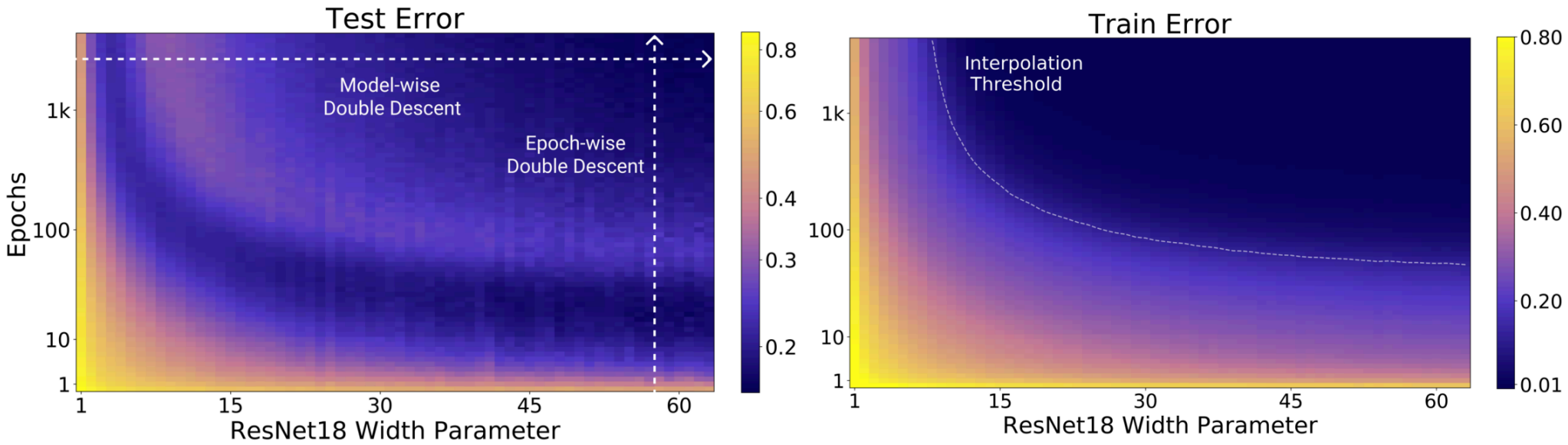
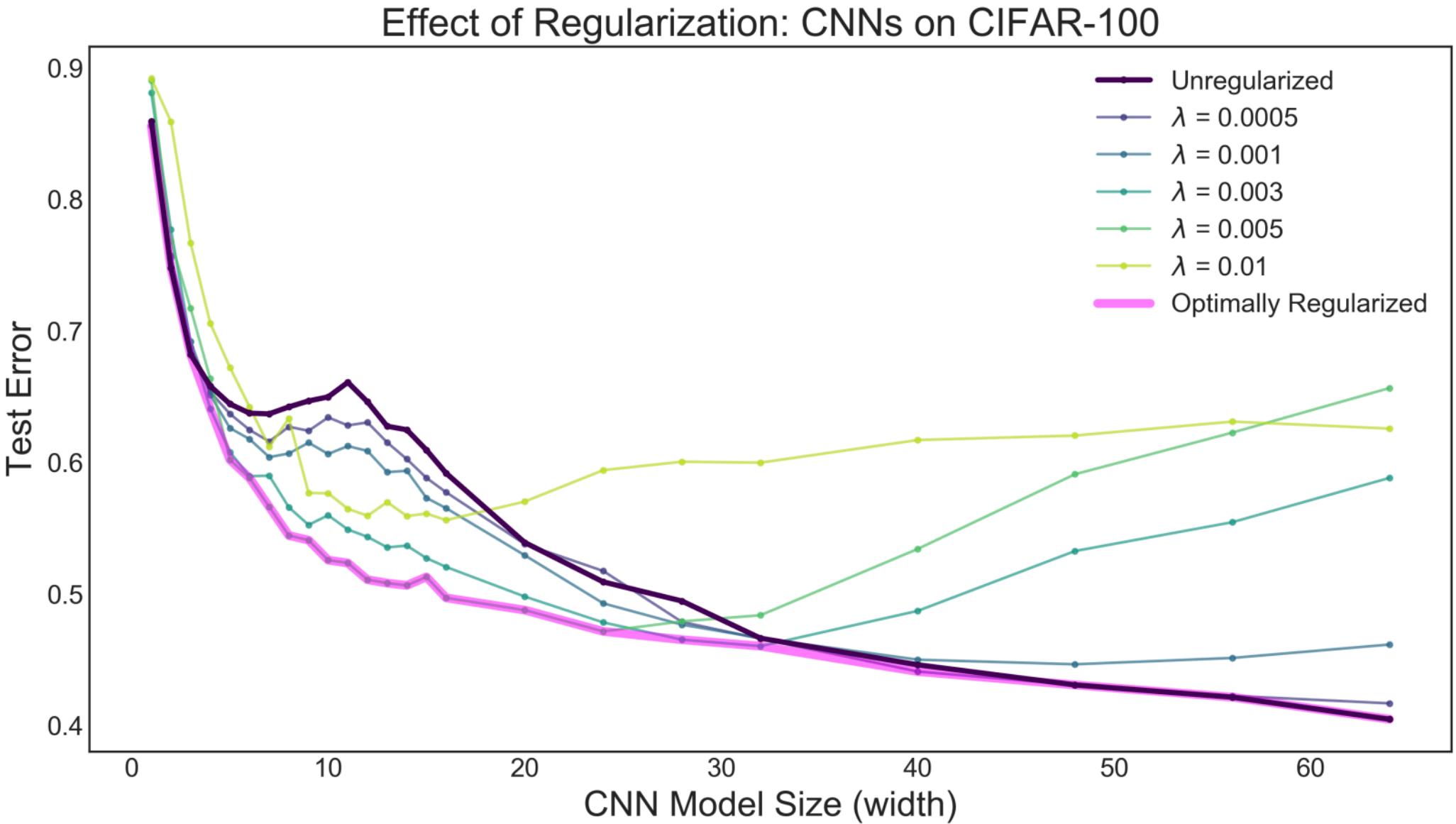


Figure 2: **Left:** Test error as a function of model size and train epochs. The horizontal line corresponds to model-wise double descent–varying model size while training for as long as possible. The vertical line corresponds to epoch-wise double descent, with test error undergoing double-descent as train time increases. **Right** Train error of the corresponding models. All models are Resnet18s trained on CIFAR-10 with 15% label noise, data-augmentation, and Adam for up to 4K epochs.

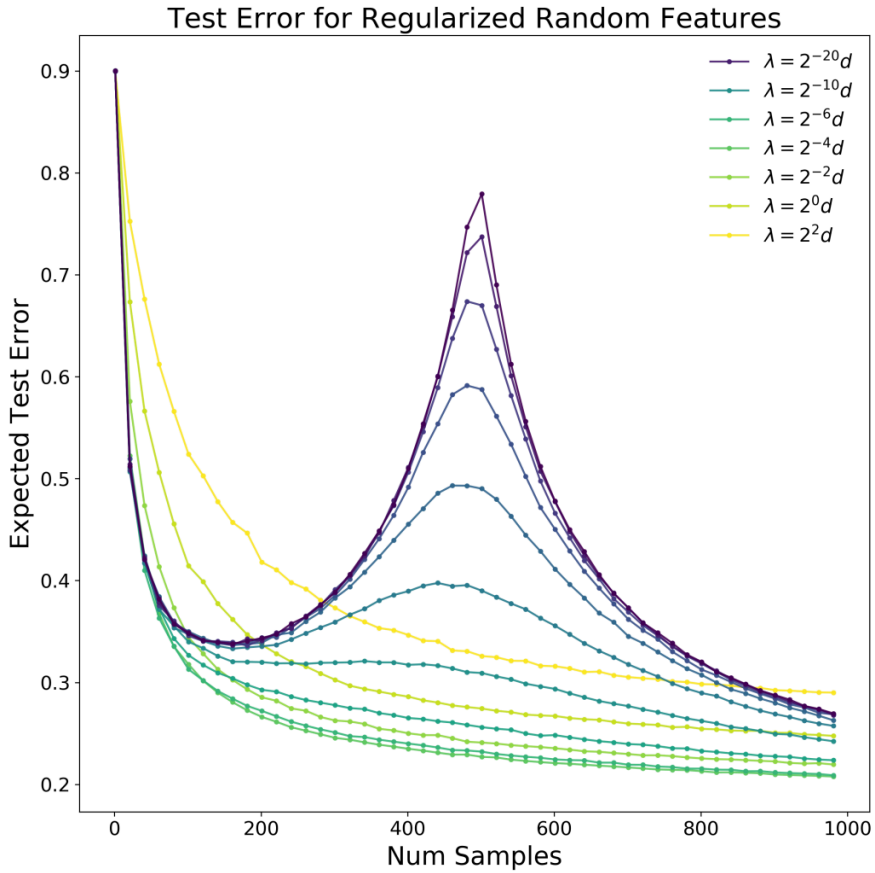
Double descent

Optimal regularization can mitigate double descent [Nakkiran et al. ’21]:

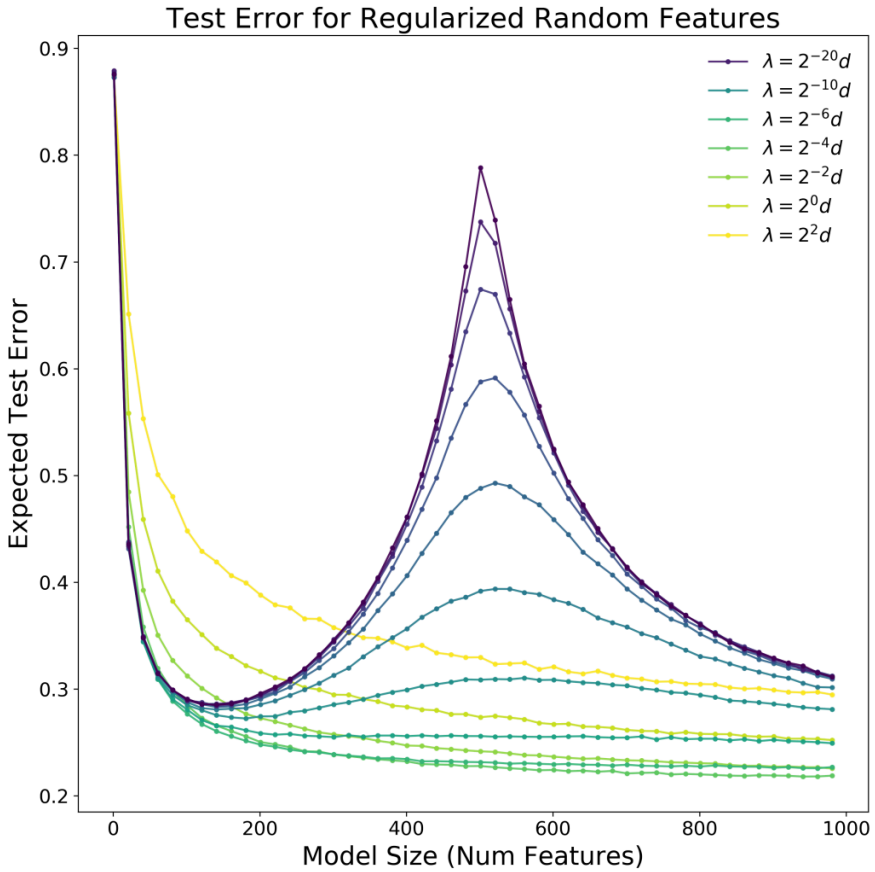


Double descent

Optimal regularization can mitigate double descent [Nakkiran et al. ’21]:



a) Test Classification Error vs. Number of Training Samples.



(b) Test Classification Error vs. Model Size (Number of Random Features).

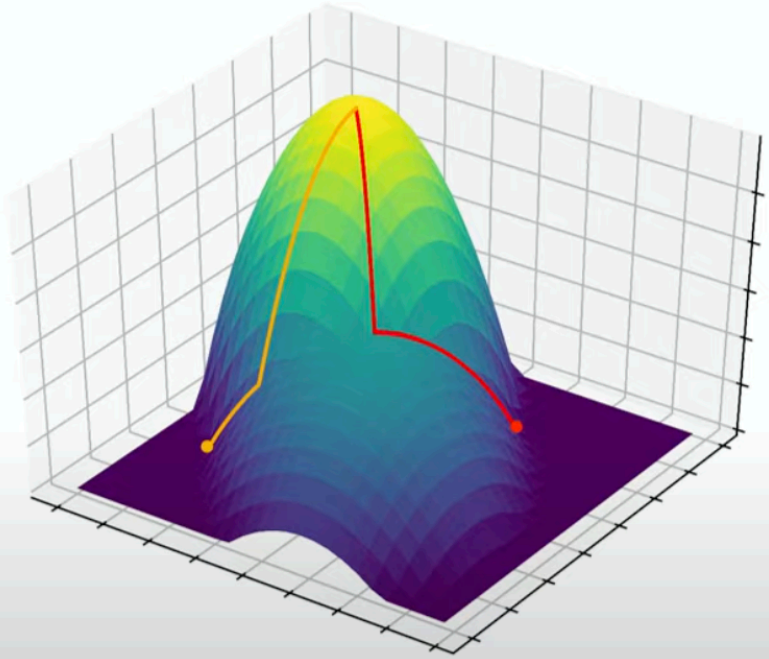
Implicit Regularization

Different optimization algorithm

➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties



Implicit Bias

Margin:

- Linear predictors:
  - Gradient descent, mirror descent, natural gradient descent, steepest descent, etc maximize margins with respect to different norms.
- Non-linear:
  - Gradient descent maximizes margin for homogeneous neural networks.
  - Low-rank matrix sensing: gradient descent finds a low-rank solution.