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

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

VC-Dimension

Motivation: Do we need to consider **every** classifier in $\mathscr{F}$?
Intuitively, **pattern of classifications** on the training set should suffice. (Two predictors that predict identically on the training set should generalize similarly).

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

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

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

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

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

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

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

Examples:

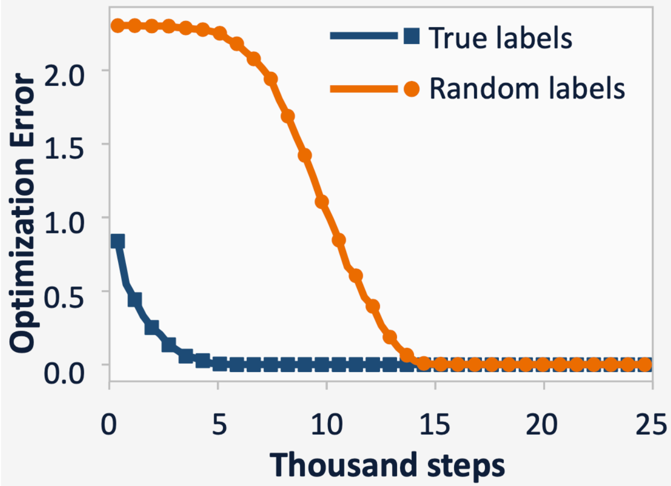
- Linear functions: VC-dim = O(dimension)
- Neural network: VC-dimension of fully-connected net with width W and H layers is $\widetilde{\Theta}(WH)$ (Bartlett et al., '17).

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 $\mathscr{F}$, 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

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

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 $\sigma_i \sim$ 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

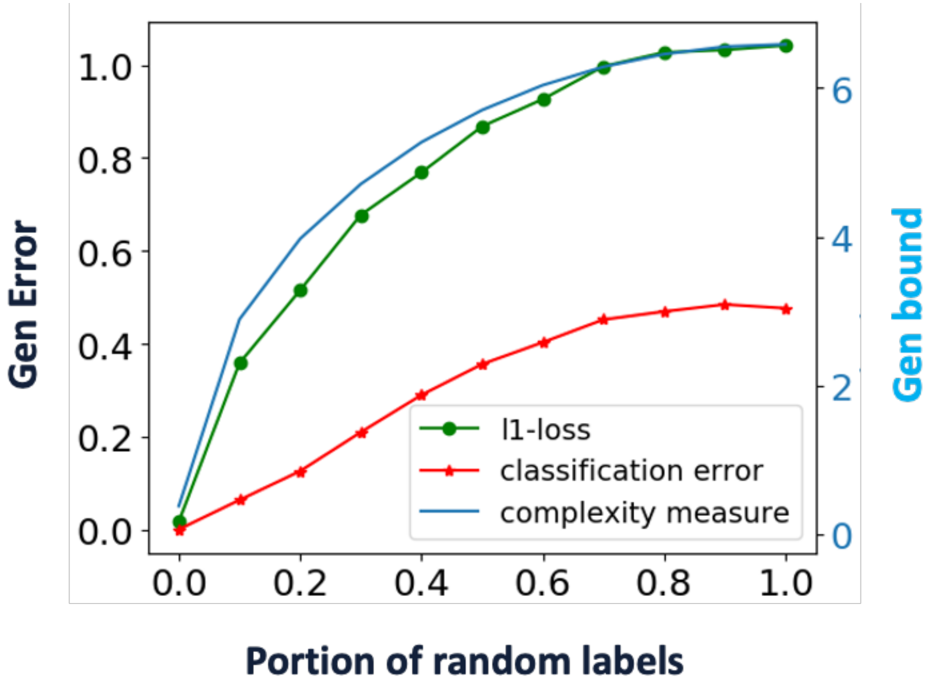
$$\sup_{f\in\mathcal{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i),y_i)-\mathbb{E}_{(x,y)\sim D}\left[\ell(f(x),y)\right]\right|=O\left(\frac{\hat{R}_n}{n}+\sqrt{\frac{\log 1/\delta}{n}}\right)$$

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

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.



Norm-based Rademacher complexity bound

Theorem: If the activation function is σ is ρ-Lipschitz. Let $\mathcal{F} = \{x \mapsto W_{H+1}\sigma(W_h\sigma(\cdots\sigma(W_1 x)\cdots)), \|W_h^T\|_{1,\infty} \leq B \forall h \in [H]\}$ 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.

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.

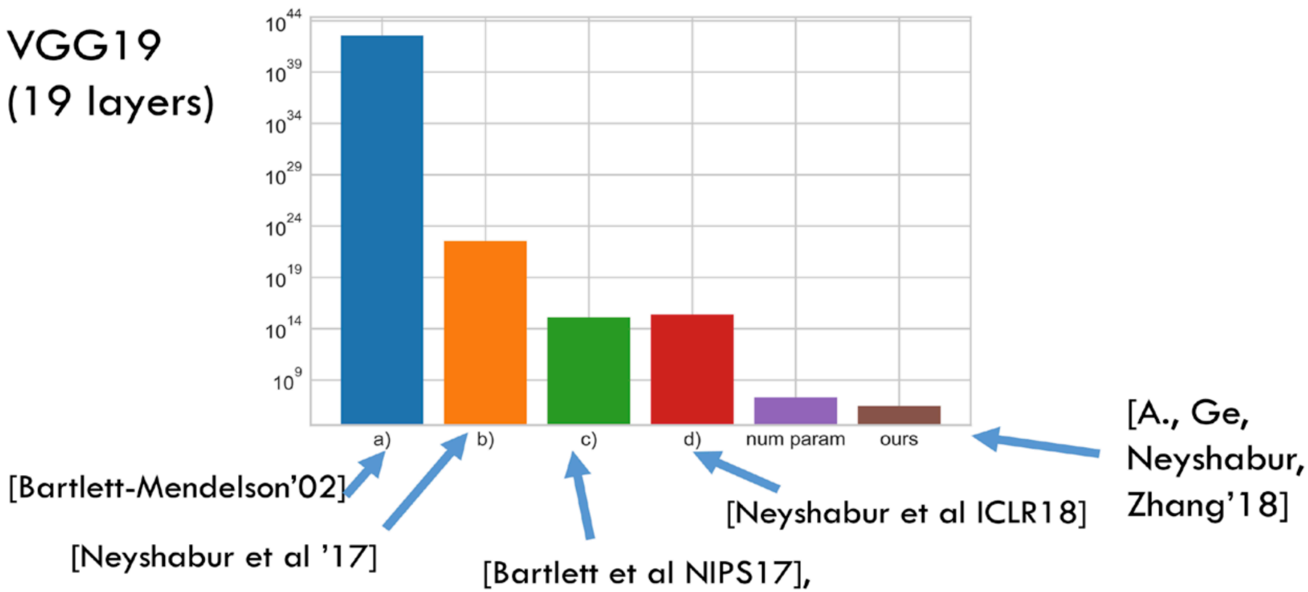


Image credits to Andrej Risteski

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 $f \in \mathscr{F}$
  - Exists example that 1) can generalize, 2) uniform convergence fails.

- 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 $\mathscr{F}$ such that $y = f(x)$ for some $f \in \mathscr{F}$:
  - no kernel is sample-efficient;
  - Exists a neural network that is sample-efficient.

Separation between NN and kernel

Separation between NN and kernel

Separation between NN and kernel

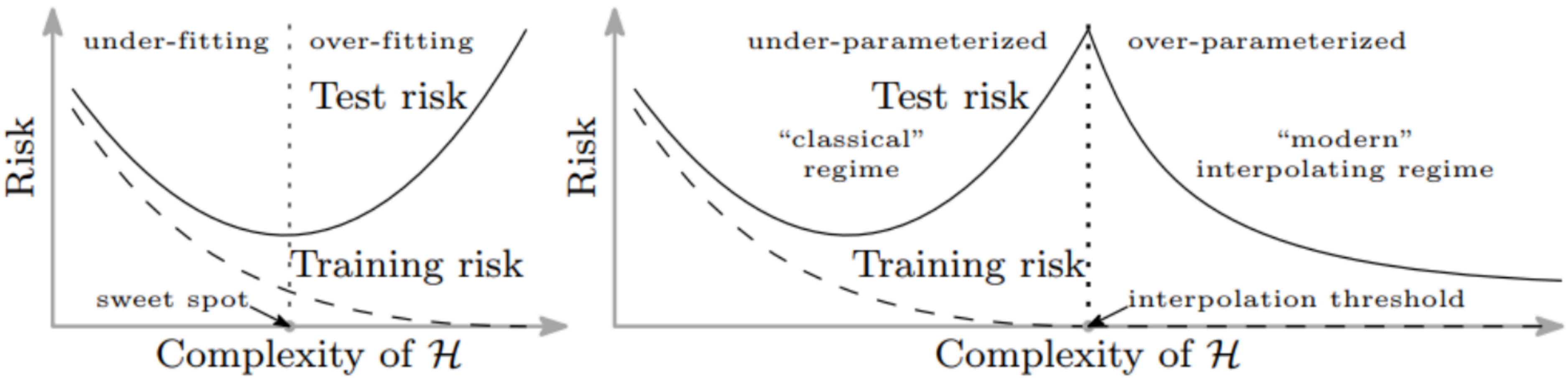
Separation between NN and kernel

Separation between NN and kernel

Separation between NN and kernel

Separation between NN and kernel

Double descent



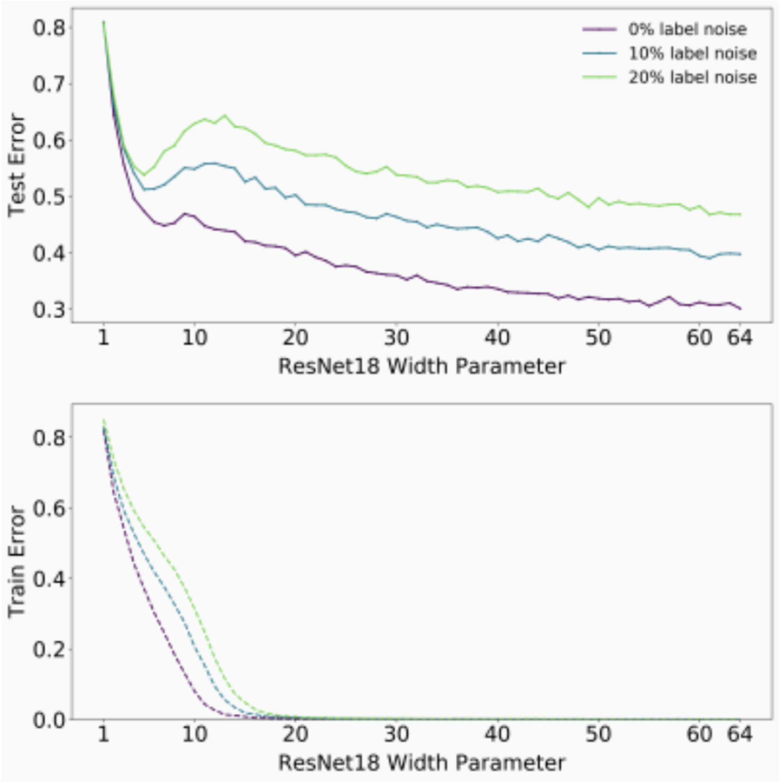
(a) U-shaped "bias-variance" risk curve (b) "double descent" risk curve

Belkin, Hsu, Ma, Mandal '18

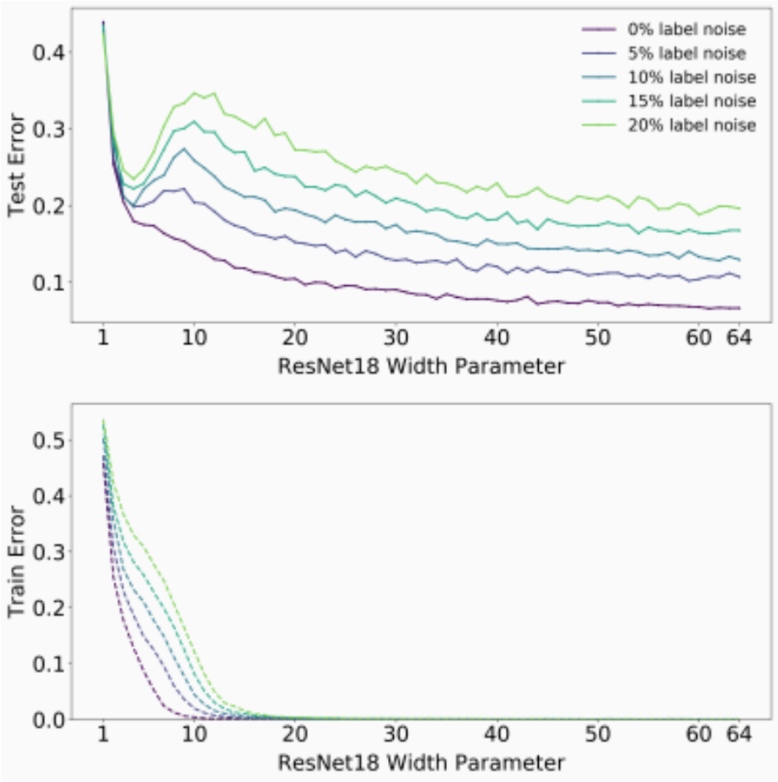
- There are cases where the model gets bigger, yet the (test!) loss goes down, sometimes even lower than in the classical "under-parameterized" regime.
- Complexity: number of parameters.

Double descent

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



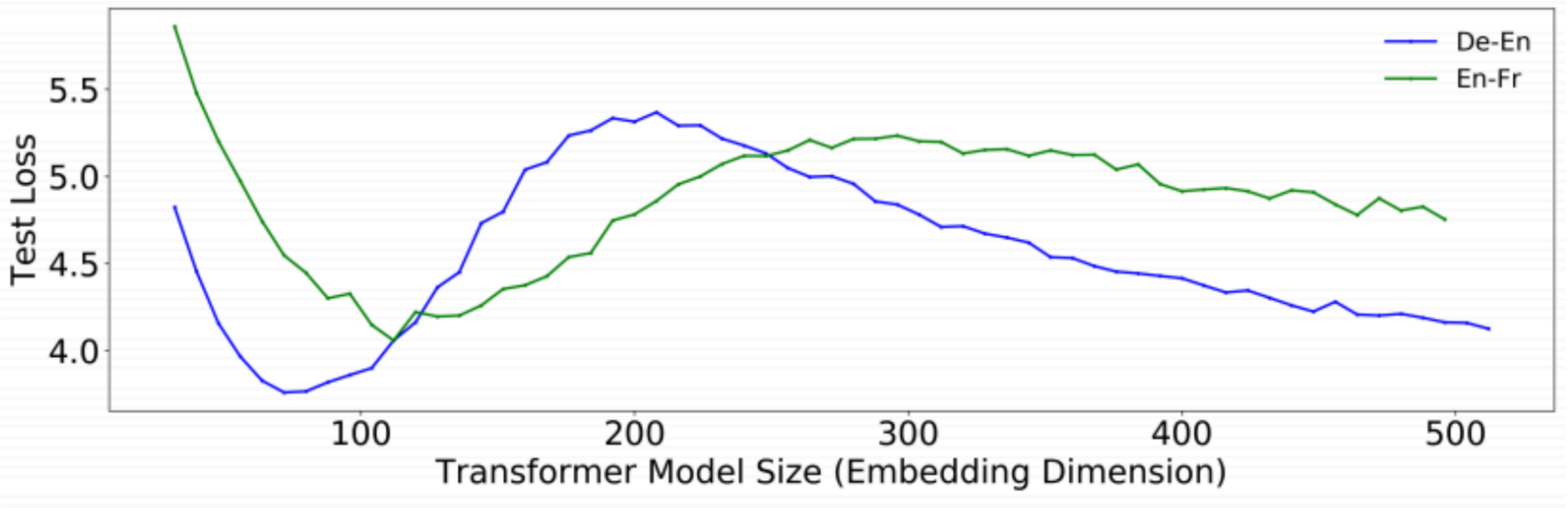
(a) **CIFAR-100.** There is a peak in test error even with no label noise.



(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.

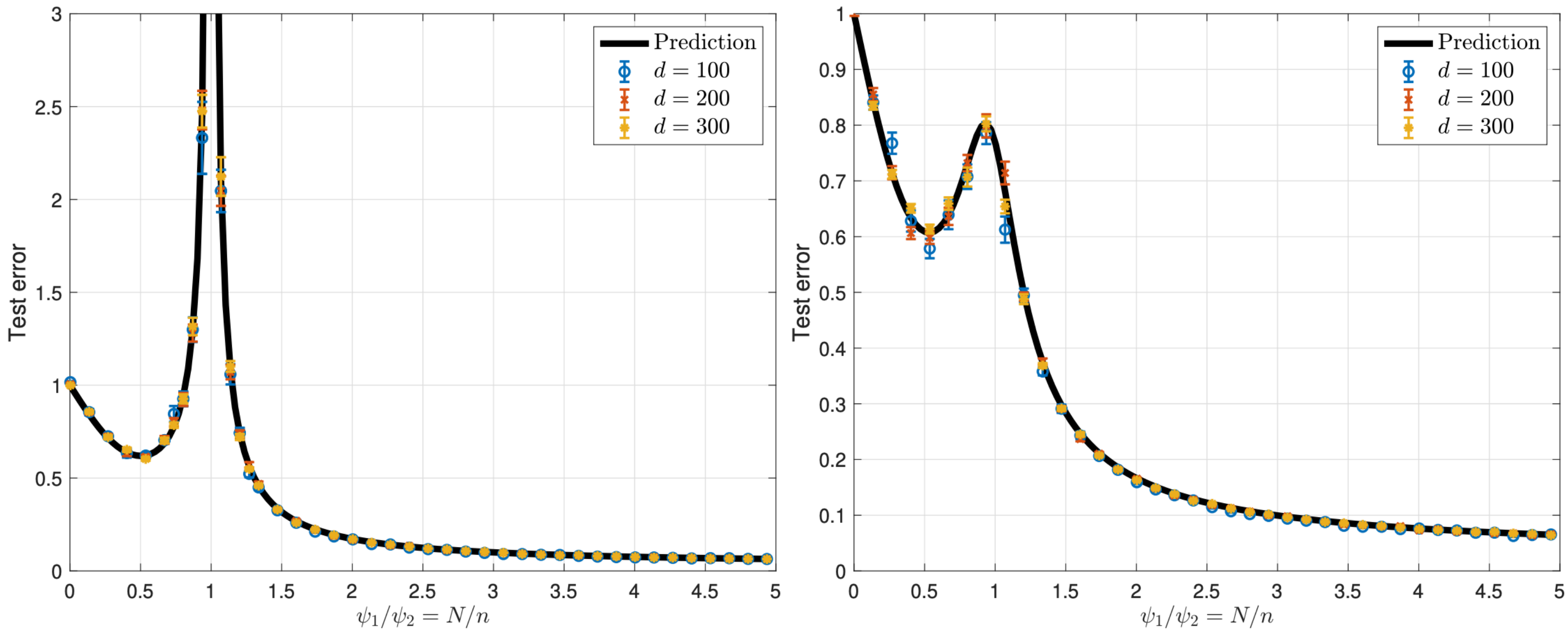
Double descent

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):

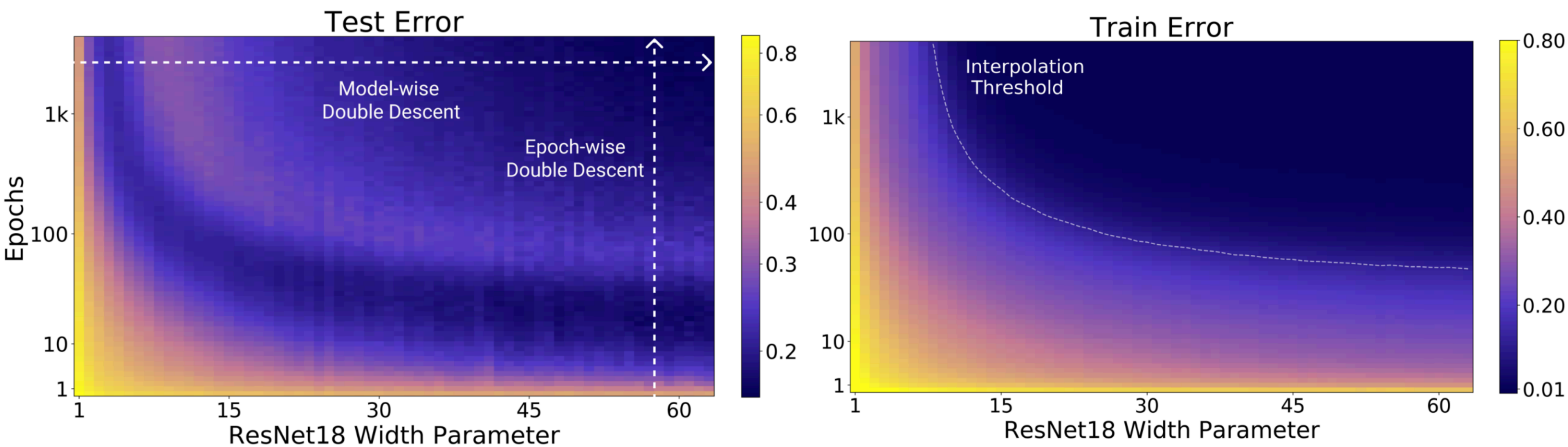
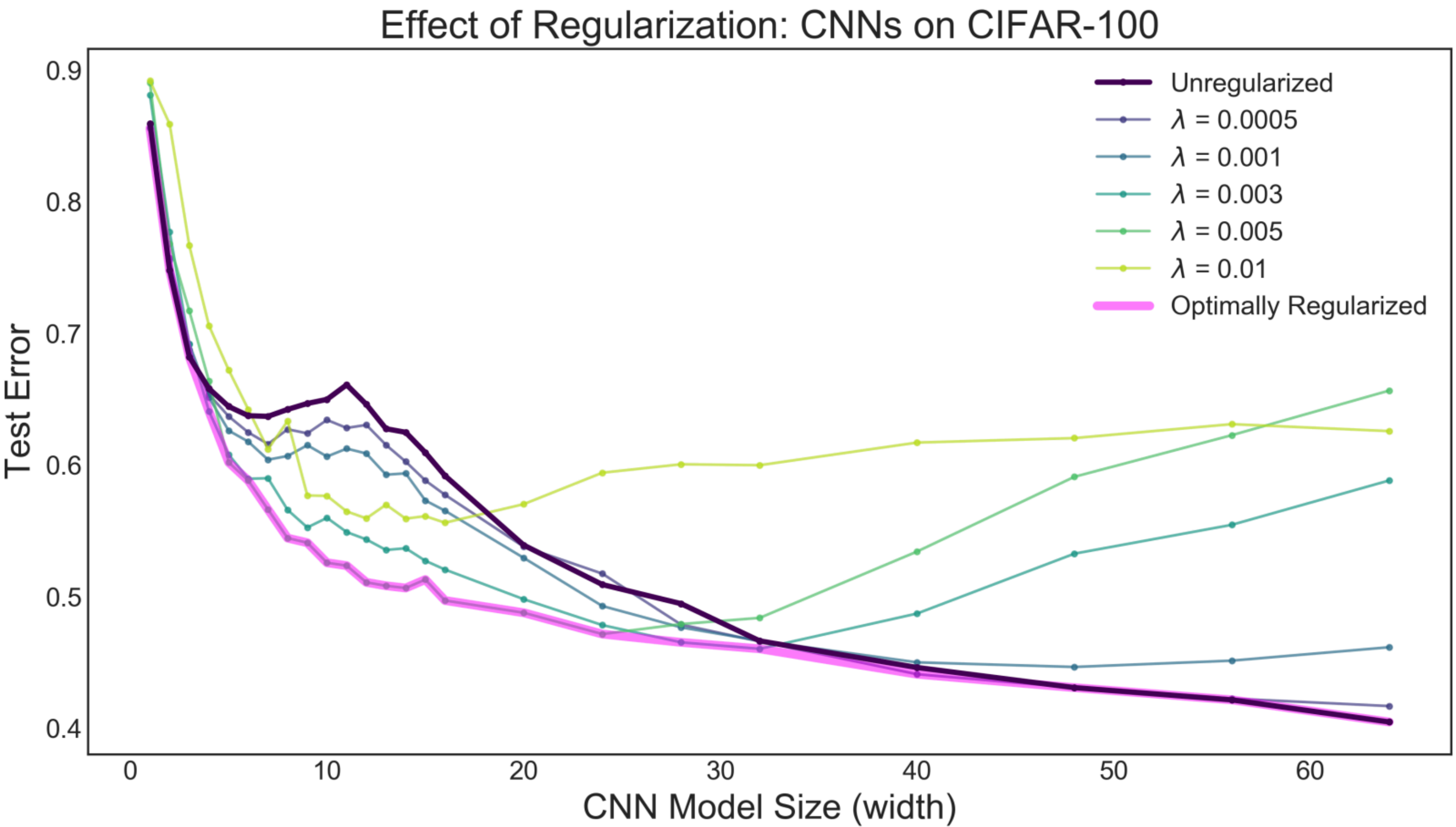


Figure 2: **Left:** Test error as a function of model size and train epochs. The horizontal line corresponds to model-wise double descent–varying model size while training for as long as possible. The vertical line corresponds to epoch-wise double descent, with test error undergoing double-descent as 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.

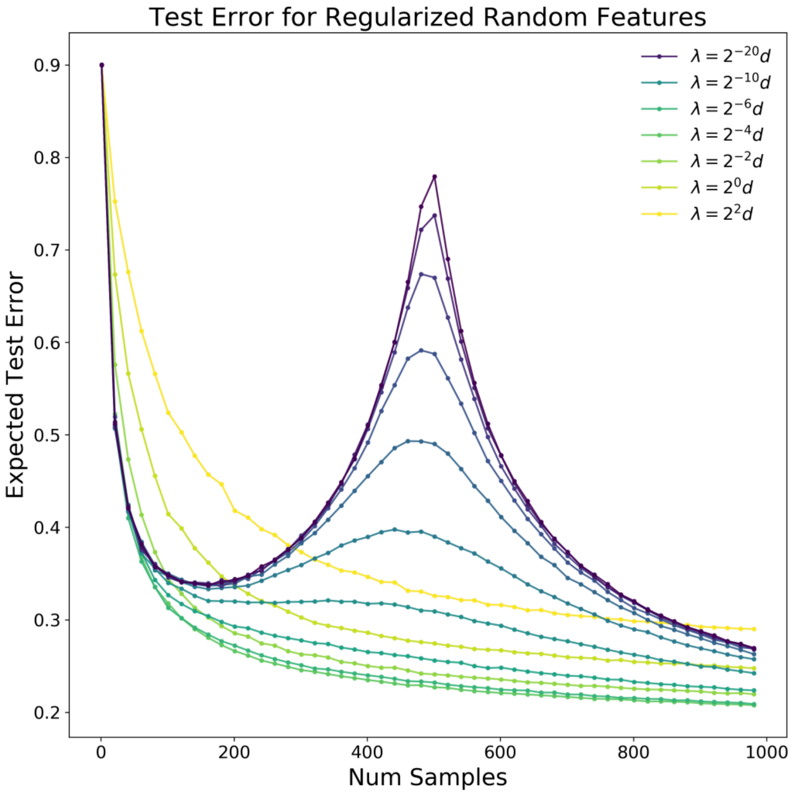
Double descent

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

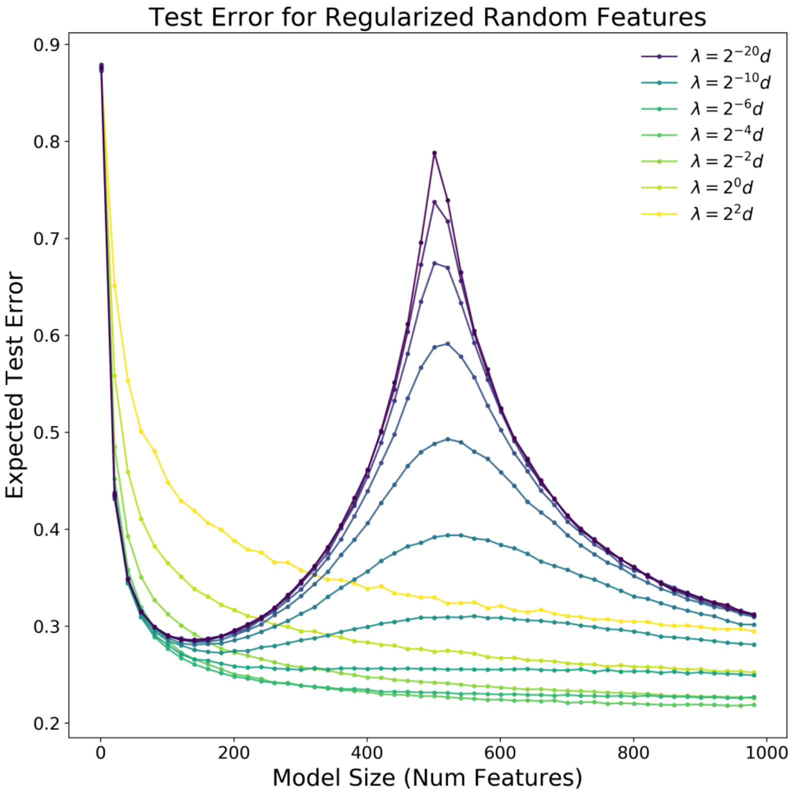


Double descent

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



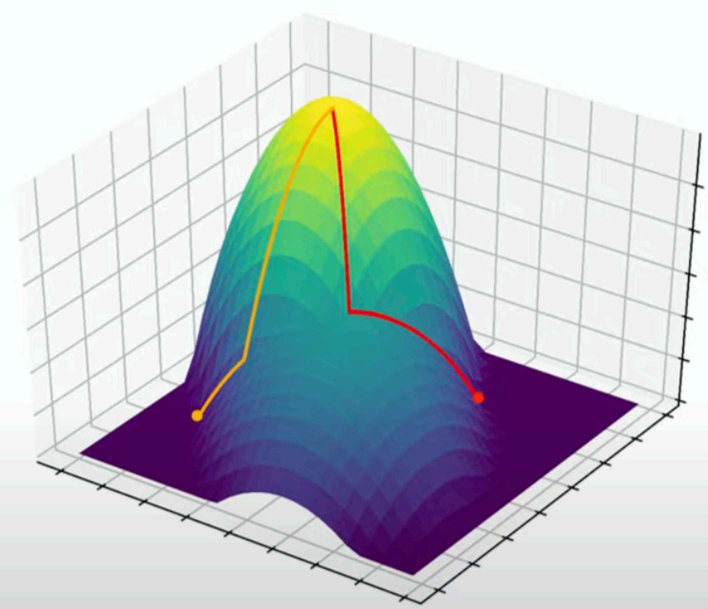
a) Test Classification Error vs. Number of Training Samples.



(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.