Neural Tangent Kernel and Double Descent

Jacob Steinhardt

Stat 240 Lecture 28

Tangent Kernel

Talked last time about random feature models and kernels, e.g. $k(x, y) = \mathbb{E}_\phi[\phi(x)\phi(y)]$

Neural networks (or any parameterized family) also look locally like kernels

Tangent Kernel

Talked last time about random feature models and kernels, e.g. $k(x, y) = \mathbb{E}_\phi[\phi(x)\phi(y)]$

Neural networks (or any parameterized family) also look locally like kernels

For a parameterized function $f_\theta : \mathcal{X} \to \mathbb{R}$, define the **tangent kernel**

$$k(x, y; \theta) = \langle \nabla f_\theta(x), \nabla f_\theta(y) \rangle$$

Tangent Kernel

Talked last time about random feature models and kernels, e.g. $k(x, y) = \mathbb{E}_\phi[\phi(x)\phi(y)]$

Neural networks (or any parameterized family) also look locally like kernels

For a parameterized function $f_\theta : \mathcal{X} \to \mathbb{R}$, define the **tangent kernel**

$$k(x, y; \theta) = \langle \nabla f_\theta(x), \nabla f_\theta(y) \rangle$$

For neural nets, basically sum over all edges in network. Full rank as long as $p \gg n$.

Neural Tangent Kernel

$$k(x, y; \theta) = \langle \nabla f_\theta(x), \nabla f_\theta(y) \rangle$$

Depends on θ: varies with random initialization, changes over course of training

Neural Tangent Kernel

$$k(x, y; \theta) = \langle \nabla f_\theta(x), \nabla f_\theta(y) \rangle$$

Depends on θ: varies with random initialization, changes over course of training

Infinite-width limit: independent of initialization (concentration of measure)

Neural Tangent Kernel

$k(x, y; \theta) = \langle \nabla f_\theta(x), \nabla f_\theta(y) \rangle$

Depends on θ: varies with random initialization, changes over course of training

Infinite-width limit: independent of initialization (concentration of measure)

Small learning rate limit: changes negligibly over training

Neural Tangent Kernel

$k(x, y; \theta) = \langle \nabla f_\theta(x), \nabla f_\theta(y) \rangle$

Depends on θ: varies with random initialization, changes over course of training

Infinite-width limit: independent of initialization (concentration of measure)

Small learning rate limit: changes negligibly over training

Jacot et al. (2018) take both limits at once and characterize the resulting kernel

- This was the first use of the phrase neural tangent kernel

Realistic Regimes

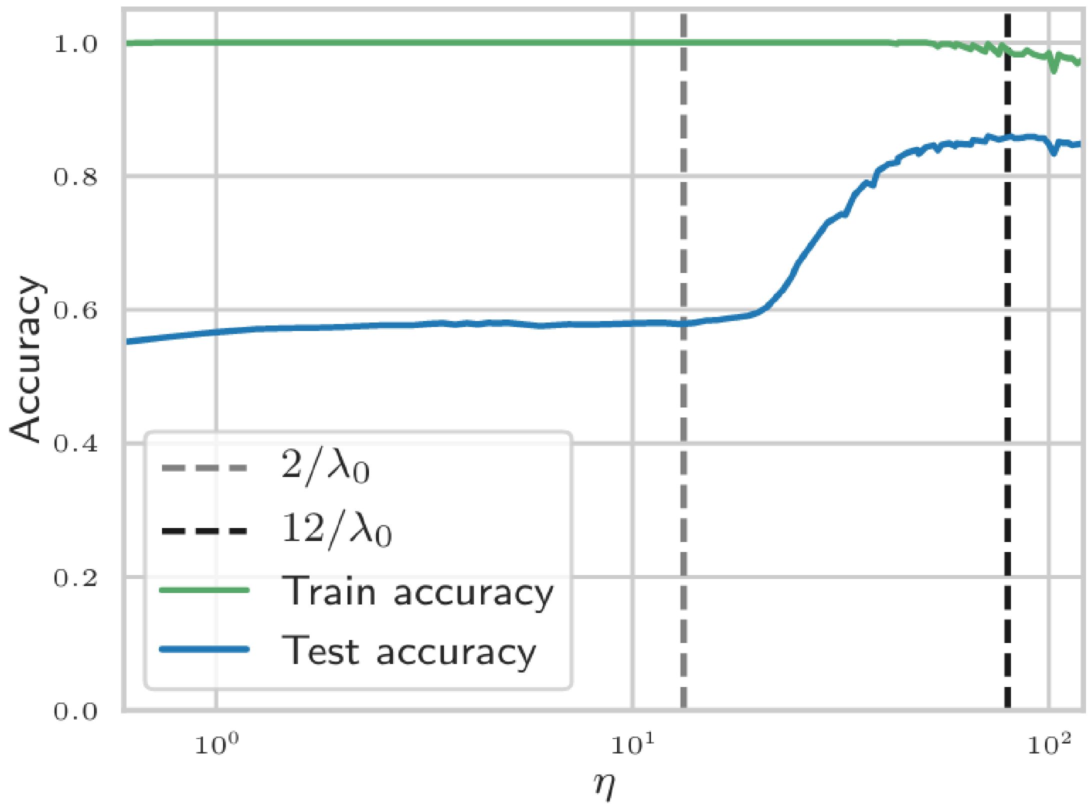
The infinite-width limit is reasonable: most networks have large width

Small learning rate is not: effectively implies that no feature learning happens (obviously false)

Lewkowycz et al. (2020) go beyond this: **catapult mechanism**

- Takes effect at intermediate learning rates (diverge at high learning rate)
- Removes high-curvature ($\approx$ high-variance) directions

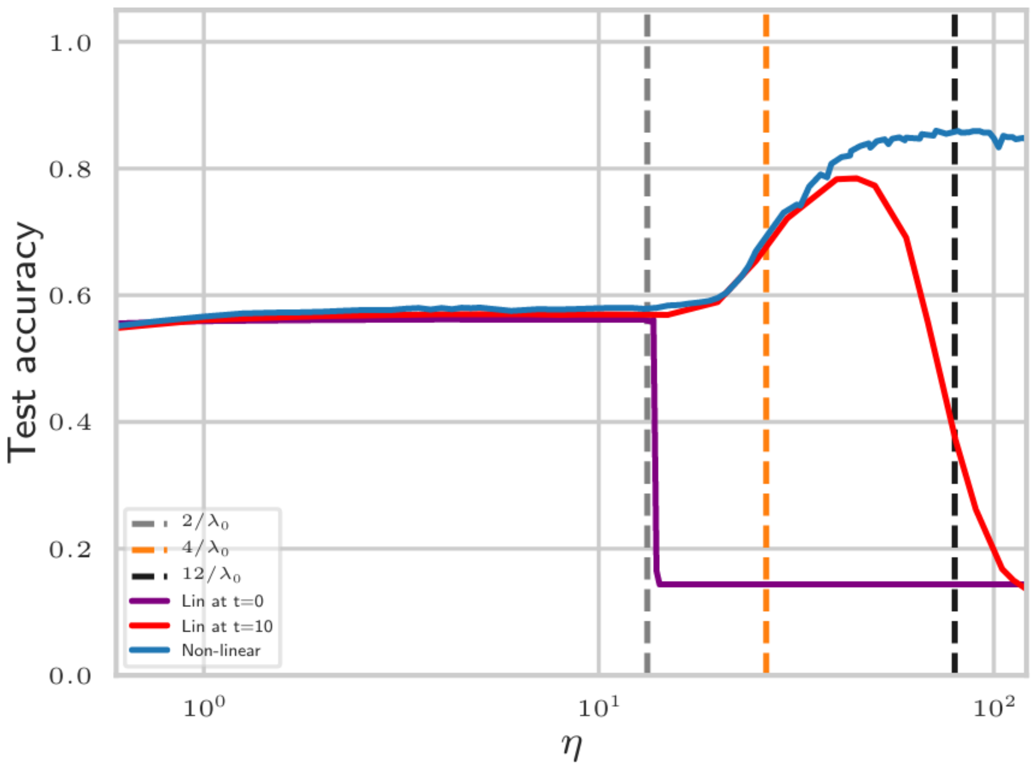
Evidence for Catapult Mechanism



Return to Linearity

Math also predicts good linear approximation after $\log(n)$ steps.

Supported empirically:



Bias-Variance Decomposition

Learned classifier $f(x)$ (depends on dataset $\mathcal{D}$), predict y

Bias-Variance Decomposition

Learned classifier $f(x)$ (depends on dataset $\mathcal{D}$), predict y

Recall **bias-variance decomposition** for mean-squared error:

$$\underbrace{\mathbb{E}_{\mathcal{D}}[(y - f(x))^2]}_{\text{MSE}} = \underbrace{(y - \mathbb{E}_{\mathcal{D}}[f(x)])^2}_{\text{Bias}^2} + \underbrace{\text{Var}_{\mathcal{D}}[f(x)]}_{\text{Variance}}$$

Bias-Variance Decomposition

Learned classifier $f(x)$ (depends on dataset $\mathcal{D}$), predict y

Recall **bias-variance decomposition** for mean-squared error:

$$\underbrace{\mathbb{E}_{\mathcal{D}}[(y - f(x))^2]}_{\text{MSE}} = \underbrace{(y - \mathbb{E}_{\mathcal{D}}[f(x)])^2}_{\text{Bias}^2} + \underbrace{\text{Var}_{\mathcal{D}}[f(x)]}_{\text{Variance}}$$

Expectation taken over randomness in training data $\mathcal{D}$ (or over random seed, etc.)

Bias-Variance Decomposition

Learned classifier $f(x)$ (depends on dataset $\mathcal{D}$), predict y

Recall **bias-variance decomposition** for mean-squared error:

$$\underbrace{\mathbb{E}_{\mathcal{D}}[(y - f(x))^2]}_{\text{MSE}} = \underbrace{(y - \mathbb{E}_{\mathcal{D}}[f(x)])^2}_{\text{Bias}^2} + \underbrace{\mathrm{Var}_{\mathcal{D}}[f(x)]}_{\text{Variance}}$$

Expectation taken over randomness in training data $\mathcal{D}$ (or over random seed, etc.)

Intuition: more complex models have lower bias but higher variance

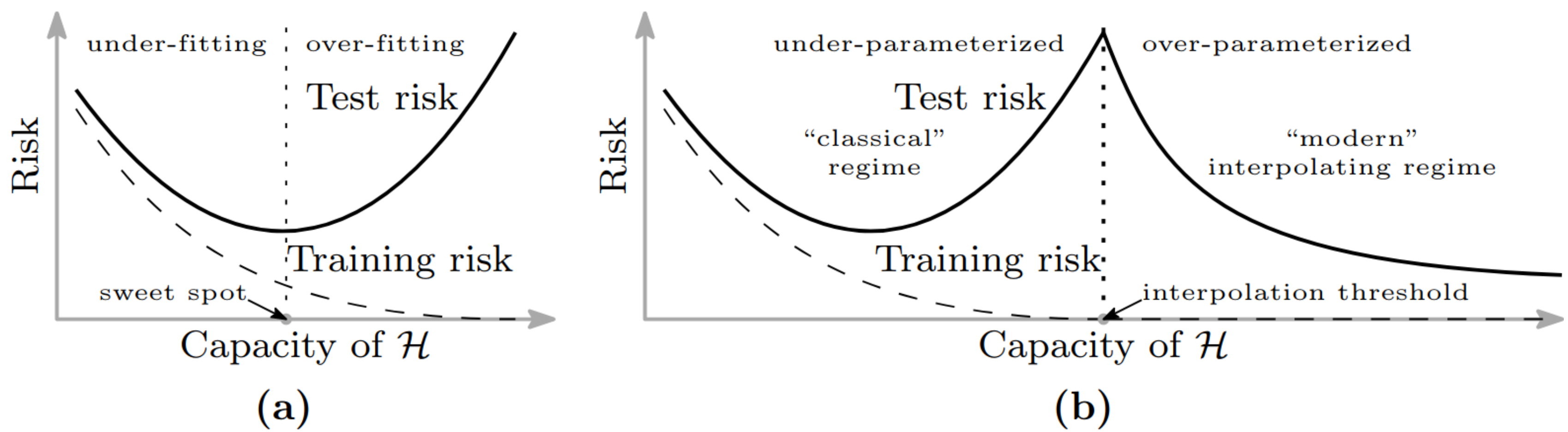
Bias-Variance for Modern Neural Nets

Classic bias-variance decomposition appears to contradict modern practice: **bigger models generalize better**, rather than overfitting.

Bias-Variance for Modern Neural Nets

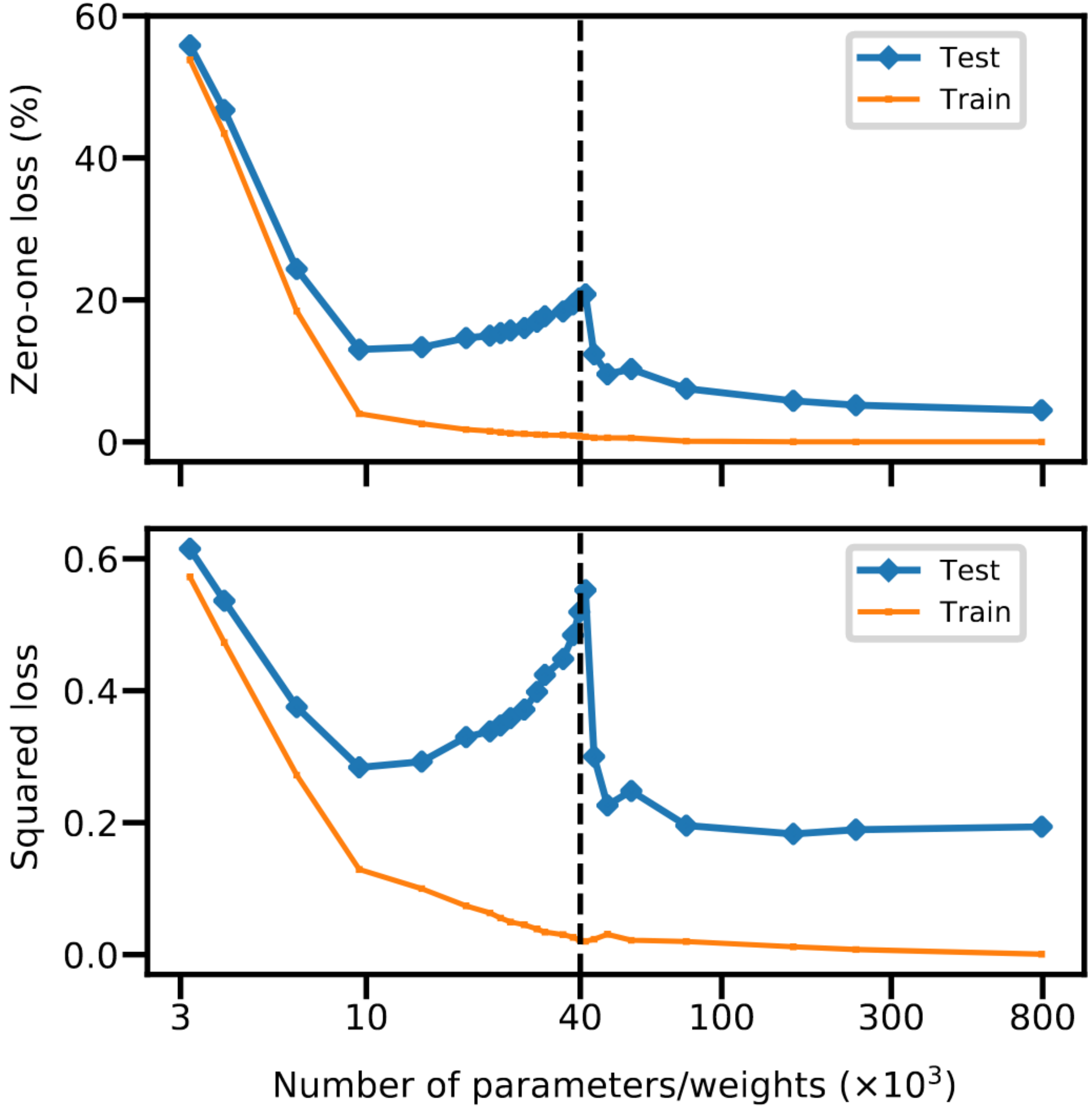
Classic bias-variance decomposition appears to contradict modern practice: **bigger models generalize better**, rather than overfitting.

Proposed solution: double descent curve



Belkin et al., 2018

Double Descent on MNIST



Belkin et al., 2018

Computing Bias+Variance (Fixed Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Computing Bias+Variance (Fixed Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Fixed design: hold $X_{1:n}$ fixed, imagine $y_{1:n}$ vary

Computing Bias+Variance (Fixed Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Fixed design: hold $X_{1:n}$ fixed, imagine $y_{1:n}$ vary

Assuming each y_i has Gaussian error with variance σ^2, can compute e.g. for linear regression

(cf. previous few lectures)

Computing Bias+Variance (Fixed Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Fixed design: hold $X_{1:n}$ fixed, imagine $y_{1:n}$ vary

Assuming each y_i has Gaussian error with variance σ^2, can compute e.g. for linear regression

(cf. previous few lectures)

Requires lots of assumptions, so also consider **random design**

Computing Bias+Variance (Random Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Computing Bias+Variance (Random Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Split data into two halves $\mathcal{D}_1, \mathcal{D}_2$

Computing Bias+Variance (Random Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Split data into two halves $\mathcal{D}_1, \mathcal{D}_2$

Train classifiers f_1, f_2

Computing Bias+Variance (Random Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Split data into two halves $\mathcal{D}_1, \mathcal{D}_2$

Train classifiers f_1, f_2

Unbiased estimate of variance: $\frac{1}{2}(f_1(x) - f_2(x))^2$

Computing Bias+Variance (Random Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Split data into two halves $\mathcal{D}_1, \mathcal{D}_2$

Train classifiers f_1, f_2

Unbiased estimate of variance: $\frac{1}{2}(f_1(x) - f_2(x))^2$

Average over multiple random splits to get better estimate

Computing Bias+Variance (Random Design)

How to compute from data? (Only one dataset $\mathcal{D}$)

Split data into two halves $\mathcal{D}_1, \mathcal{D}_2$

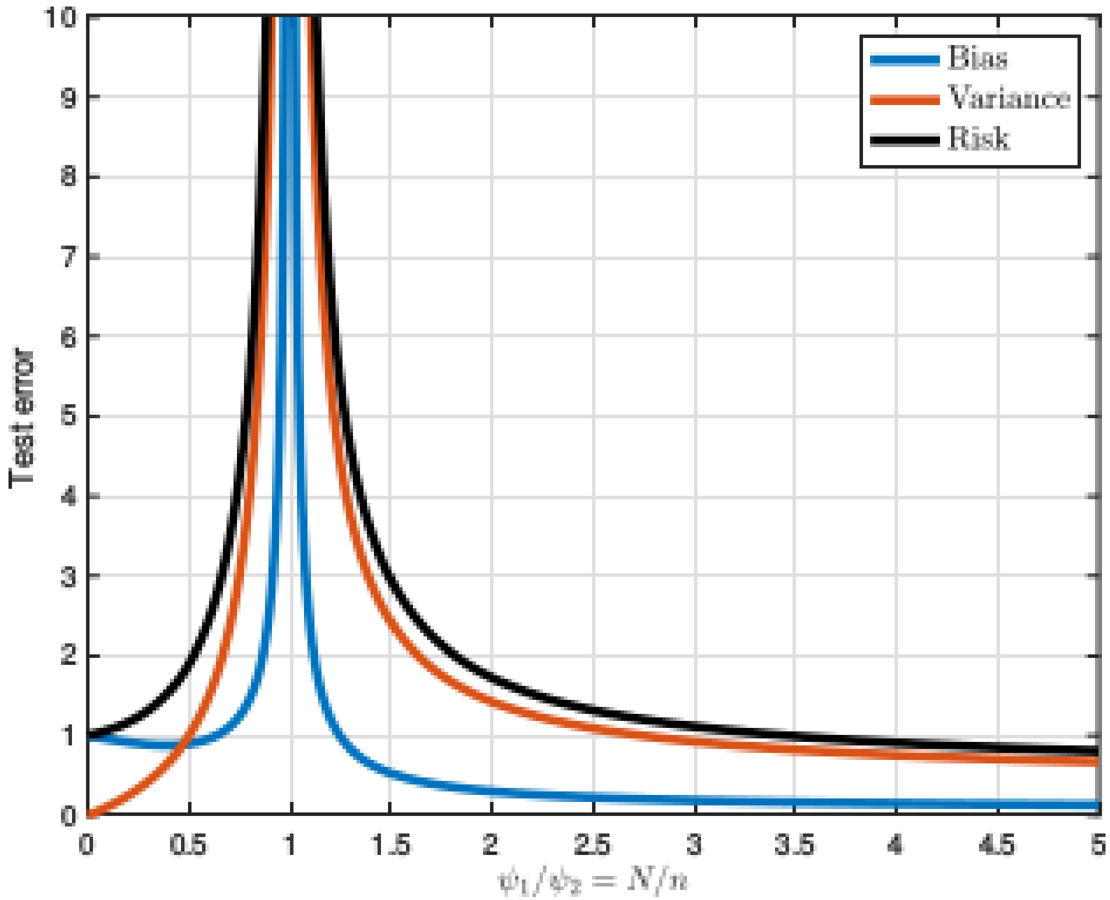
Train classifiers f_1, f_2

Unbiased estimate of variance: $\frac{1}{2}(f_1(x) - f_2(x))^2$

Average over multiple random splits to get better estimate

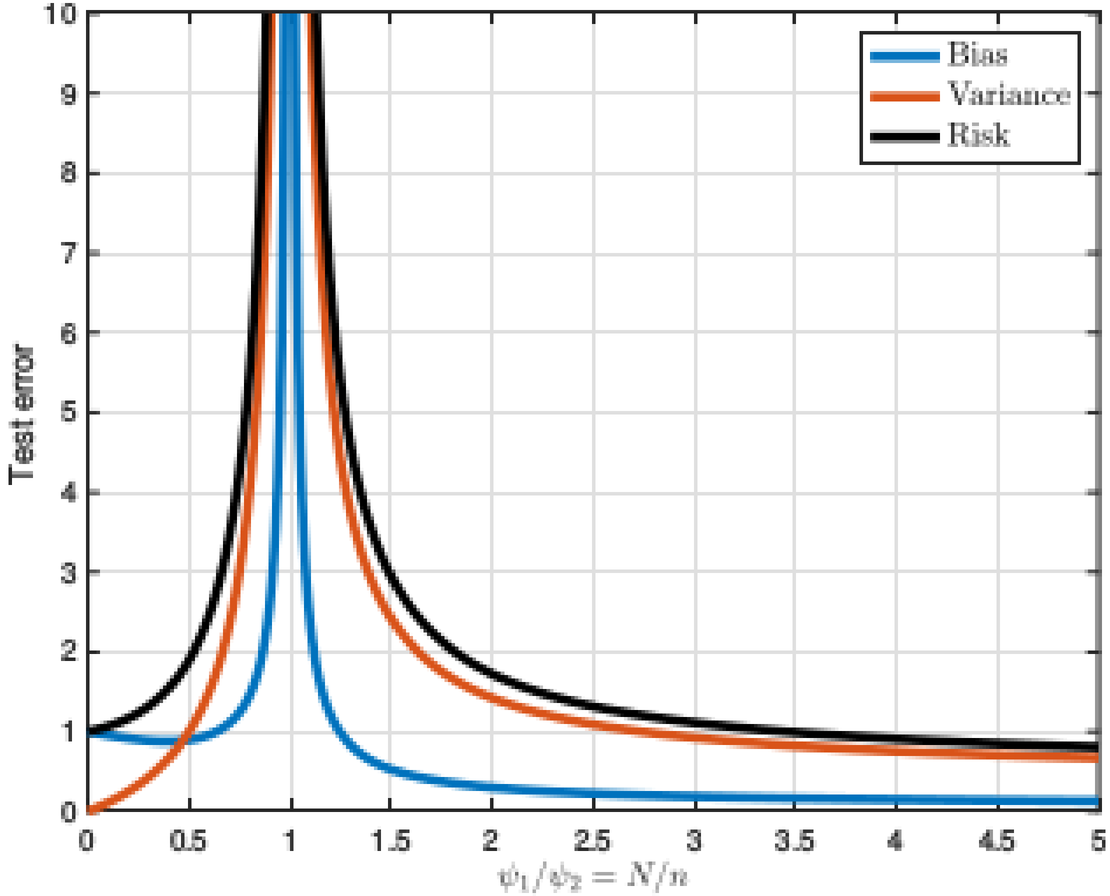
Compute bias via $\mathrm{Bias}^2 = \mathrm{MSE} - \mathrm{Variance}$

Theoretical Characterization (Fixed Design)



Mei and Montanari, 2019

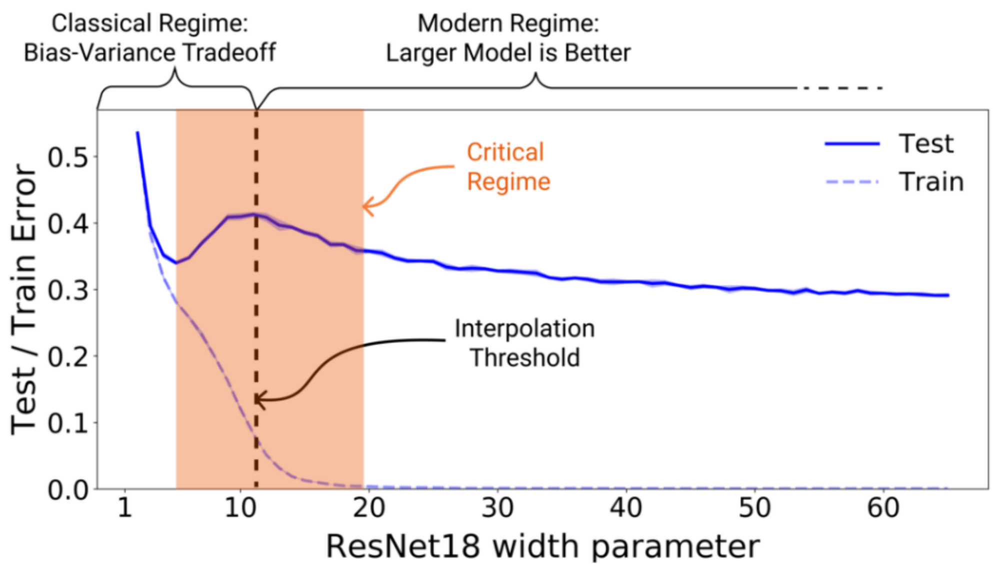
Theoretical Characterization (Fixed Design)



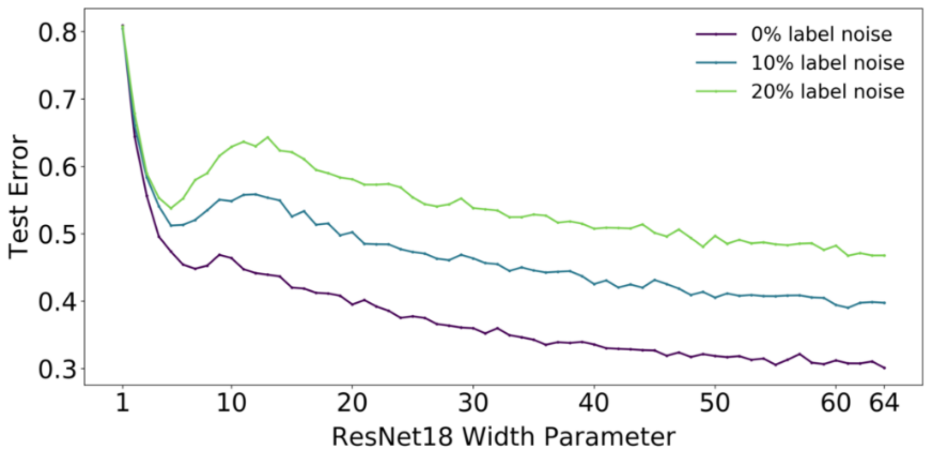
Mei and Montanari, 2019

Fixed-design: attributes some **variance** to **bias**.

Double Descent on CIFAR



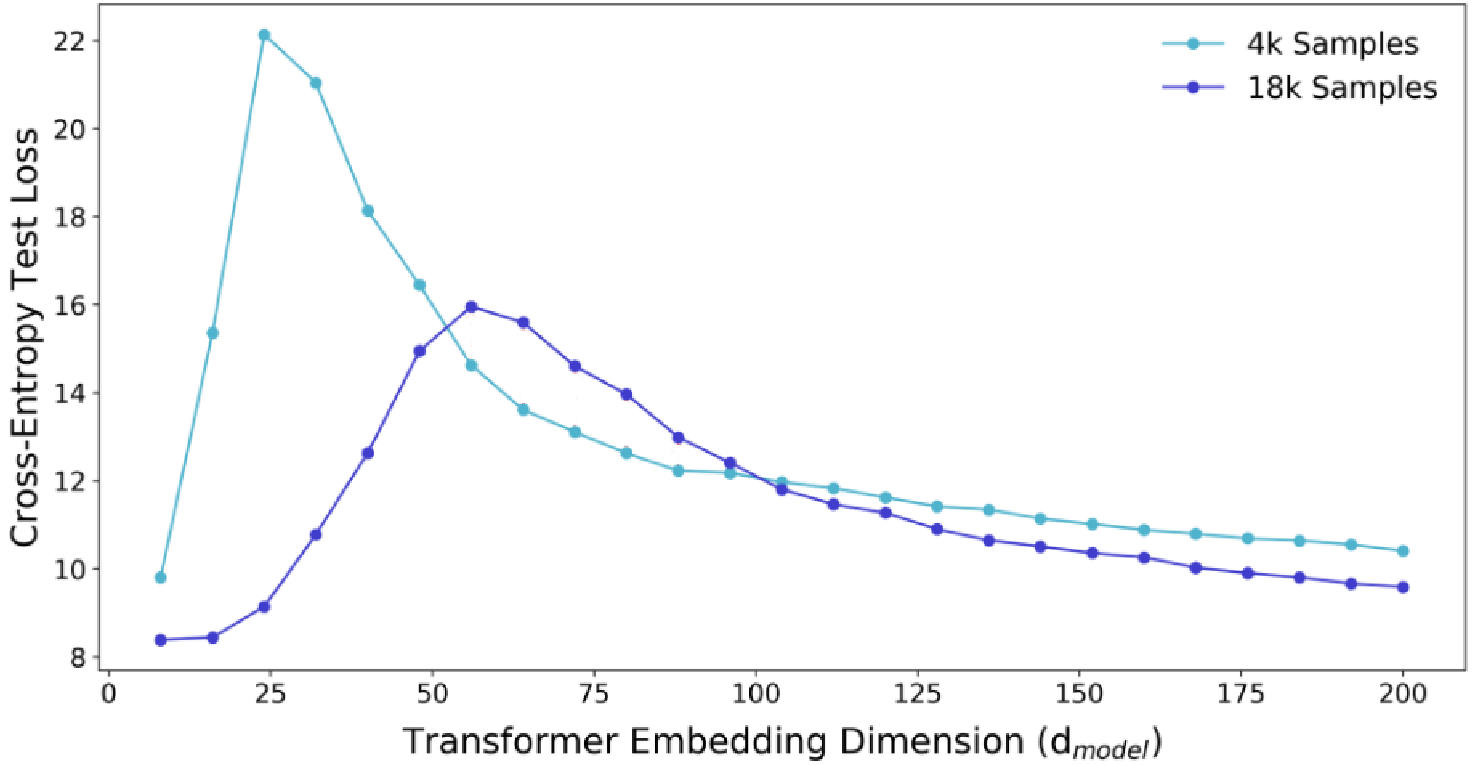
CIFAR-10.



CIFAR-100.

Nakkiran et al., 2019

Unimodal Risk in in NLP



Nakkiran et al., 2019

Mysteries

Sometimes need label noise to produce

Mysteries

Sometimes need label noise to produce

More often get monotonic or unimodal behavior in practice

Mysteries

Sometimes need label noise to produce

More often get monotonic or unimodal behavior in practice

Model sizes small relative to practice

Mysteries

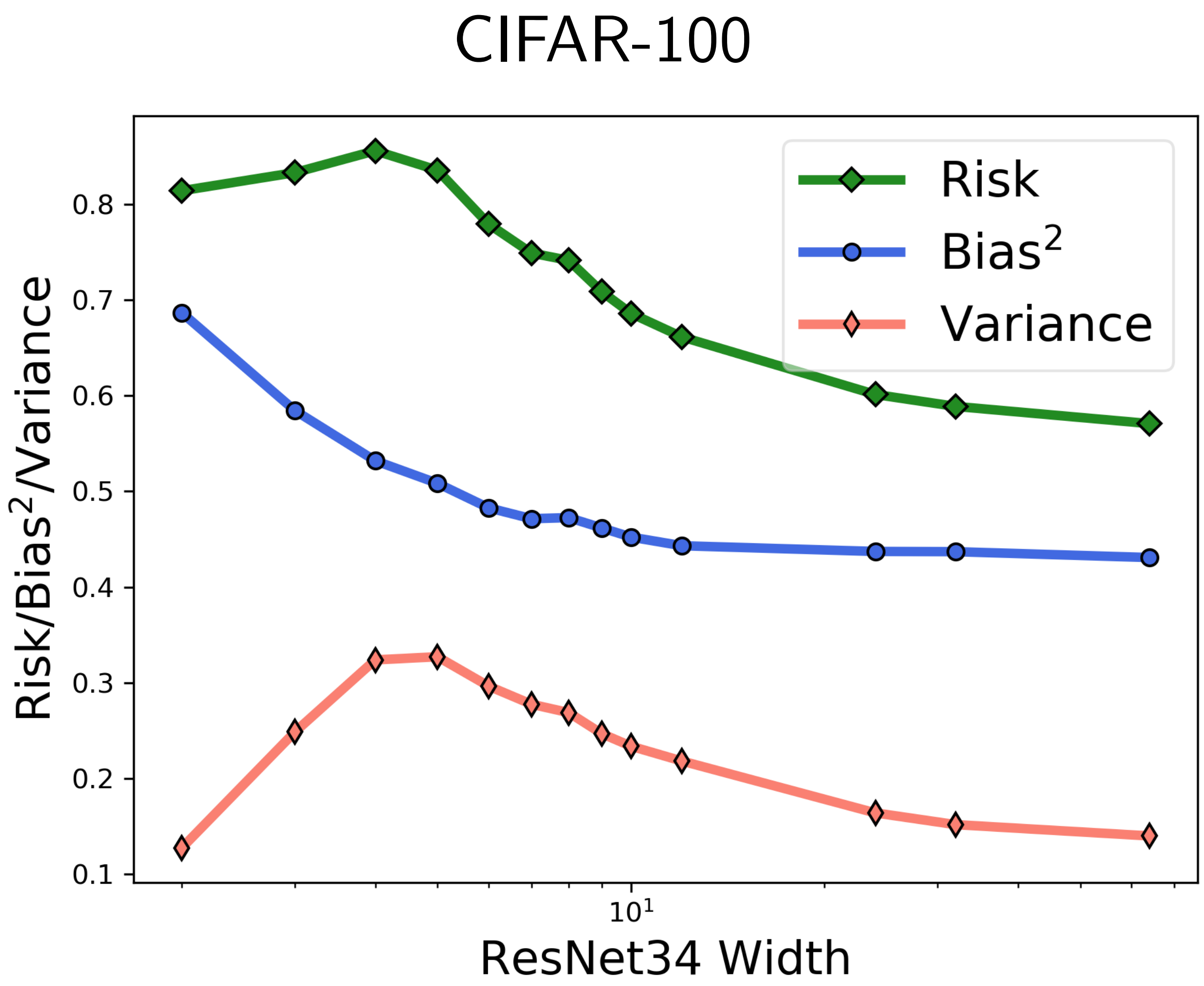
Sometimes need label noise to produce

More often get monotonic or unimodal behavior in practice

Model sizes small relative to practice

Is there a simpler underlying phenomenon?

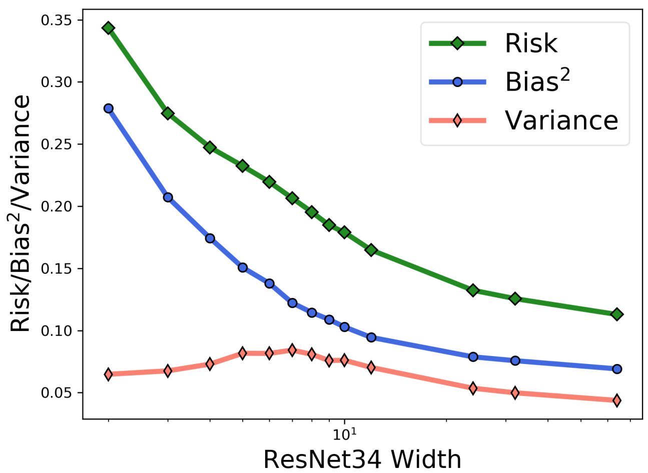
Explanation: Revisiting Bias-Variance



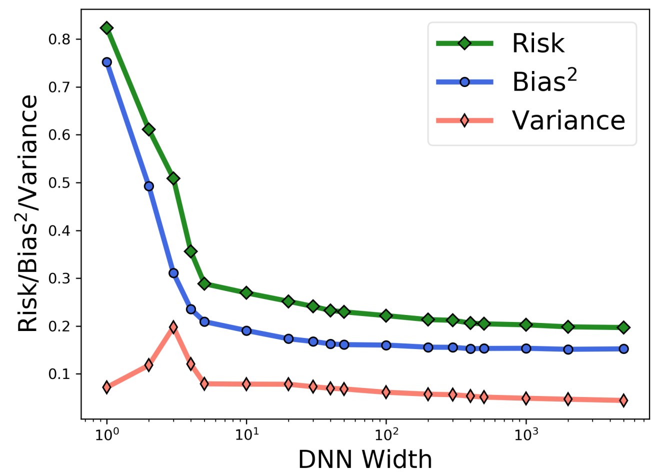
Phenomenon: monotonic bias + **unimodal** variance

Robustness of the Phenomenon

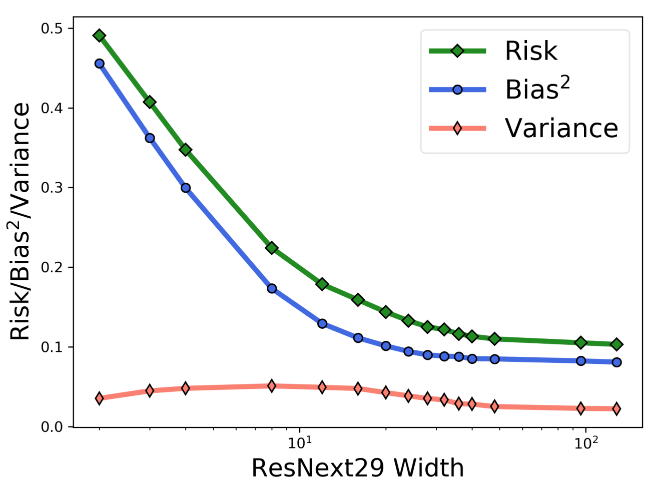
CIFAR-10



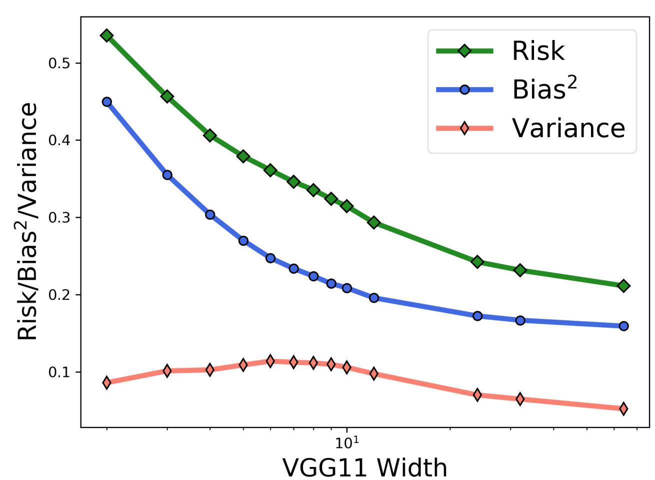
Fashion-MNIST



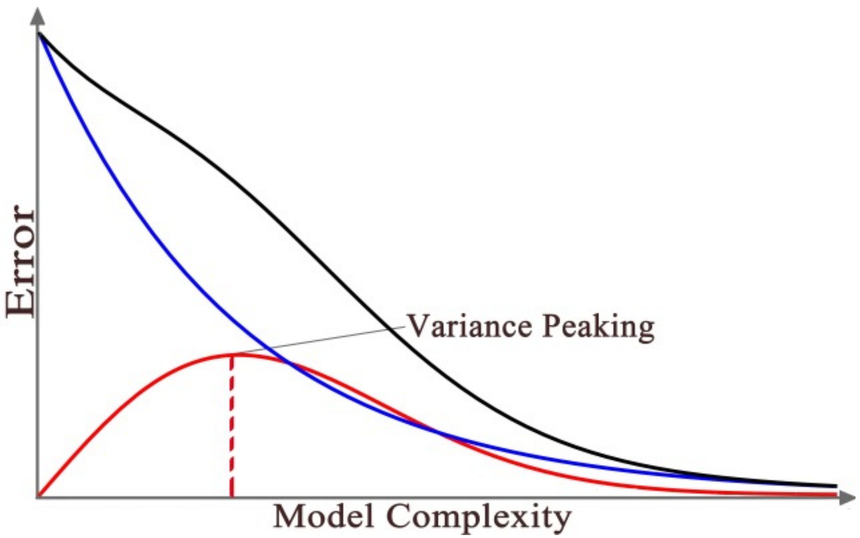
ResNext29

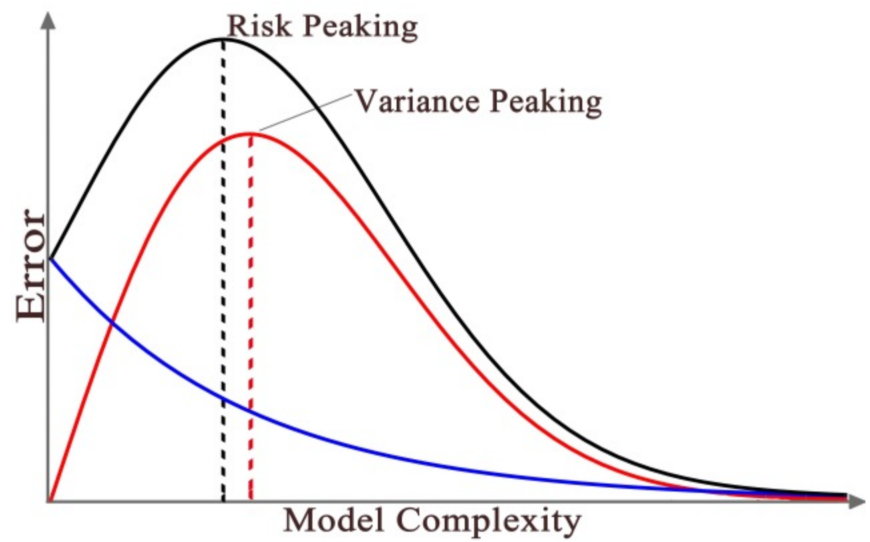


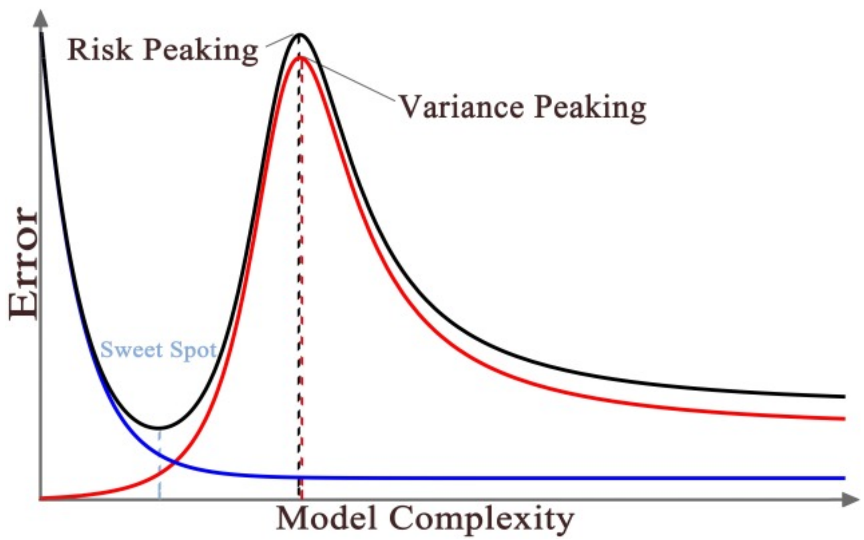
VGG11



Three Possible Behaviors







Bias-Variance for Cross-Entropy

Most networks trained with cross-entropy loss, not MSE

Generalized bias-variance decomposition for Bregman divergence

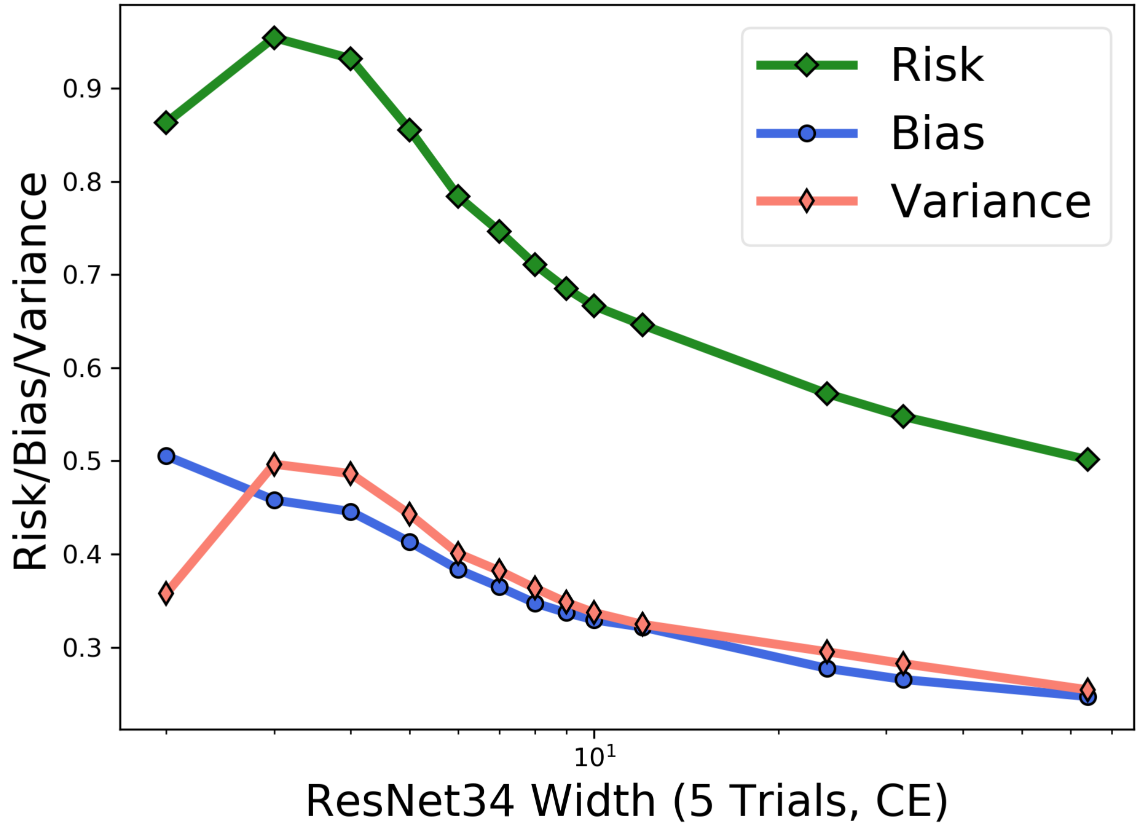
Pfau, 2013

Bias-Variance for Cross-Entropy

Most networks trained with cross-entropy loss, not MSE

Generalized bias-variance decomposition for Bregman divergence

Pfau, 2013



Characterizing Sources of Error

MSE: Get unbiased estimate, but how much finite-sample variability?

Characterizing Sources of Error

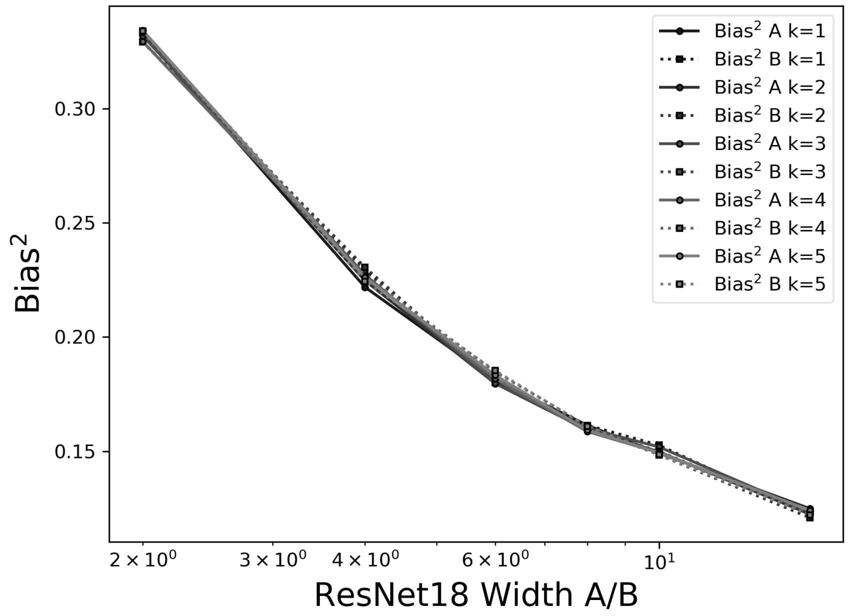
MSE: Get unbiased estimate, but how much finite-sample variability?

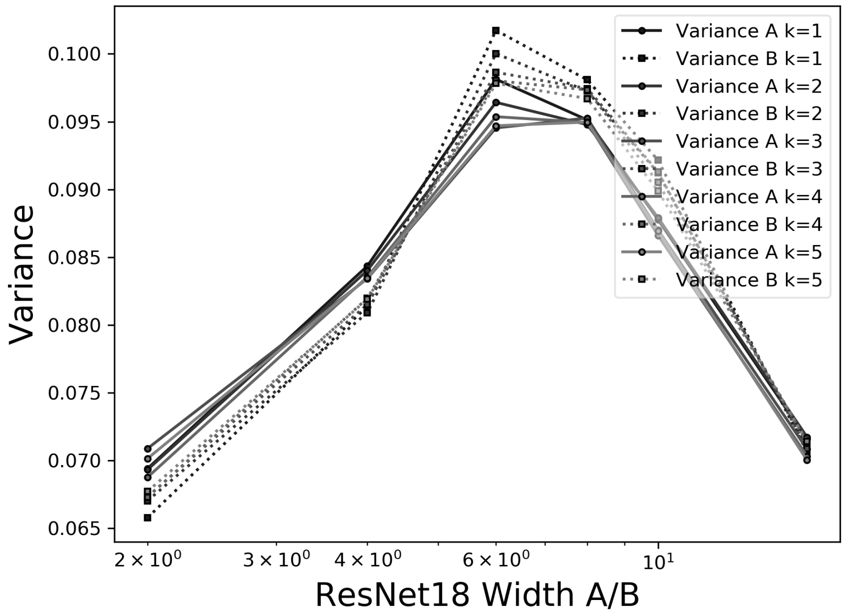
Idea: replicate entire experiment on two halves of training data

Characterizing Sources of Error

MSE: Get unbiased estimate, but how much finite-sample variability?

Idea: replicate entire experiment on two halves of training data

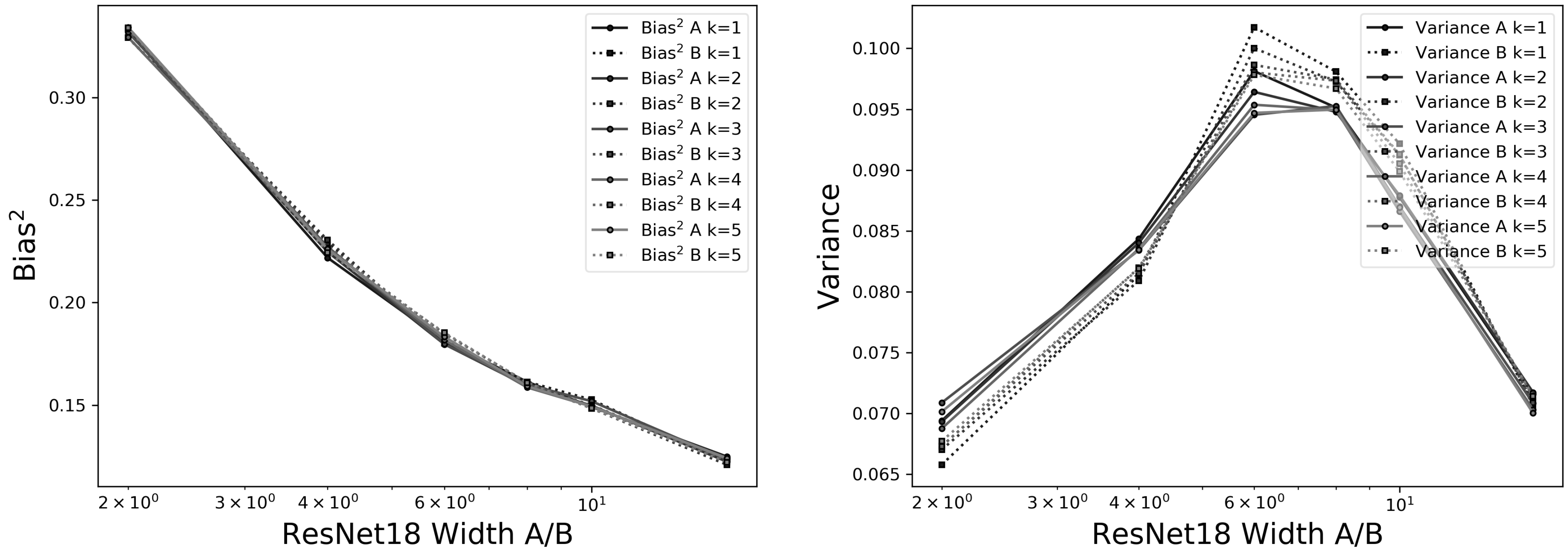




Characterizing Sources of Error

MSE: Get unbiased estimate, but how much finite-sample variability?

Idea: replicate entire experiment on two halves of training data

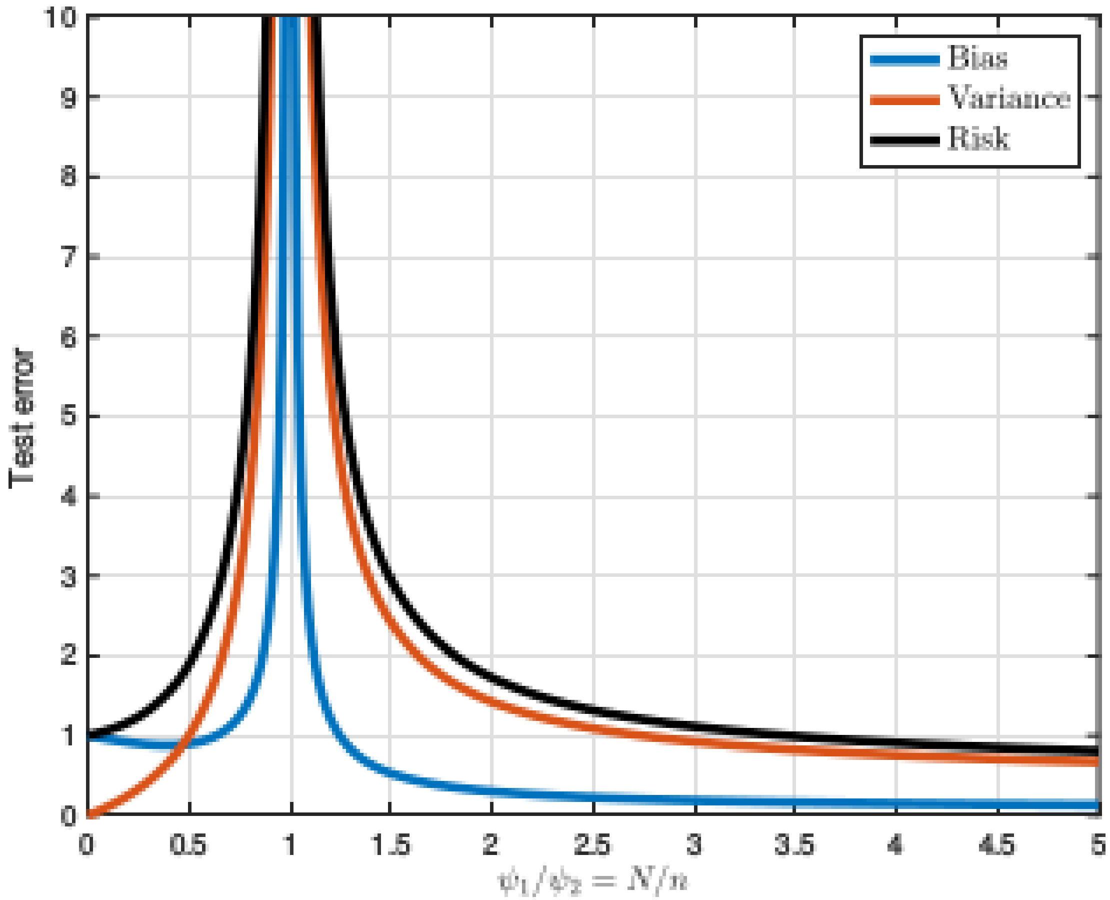


Cross-entropy: harder (no unbiased estimate)

Take-away

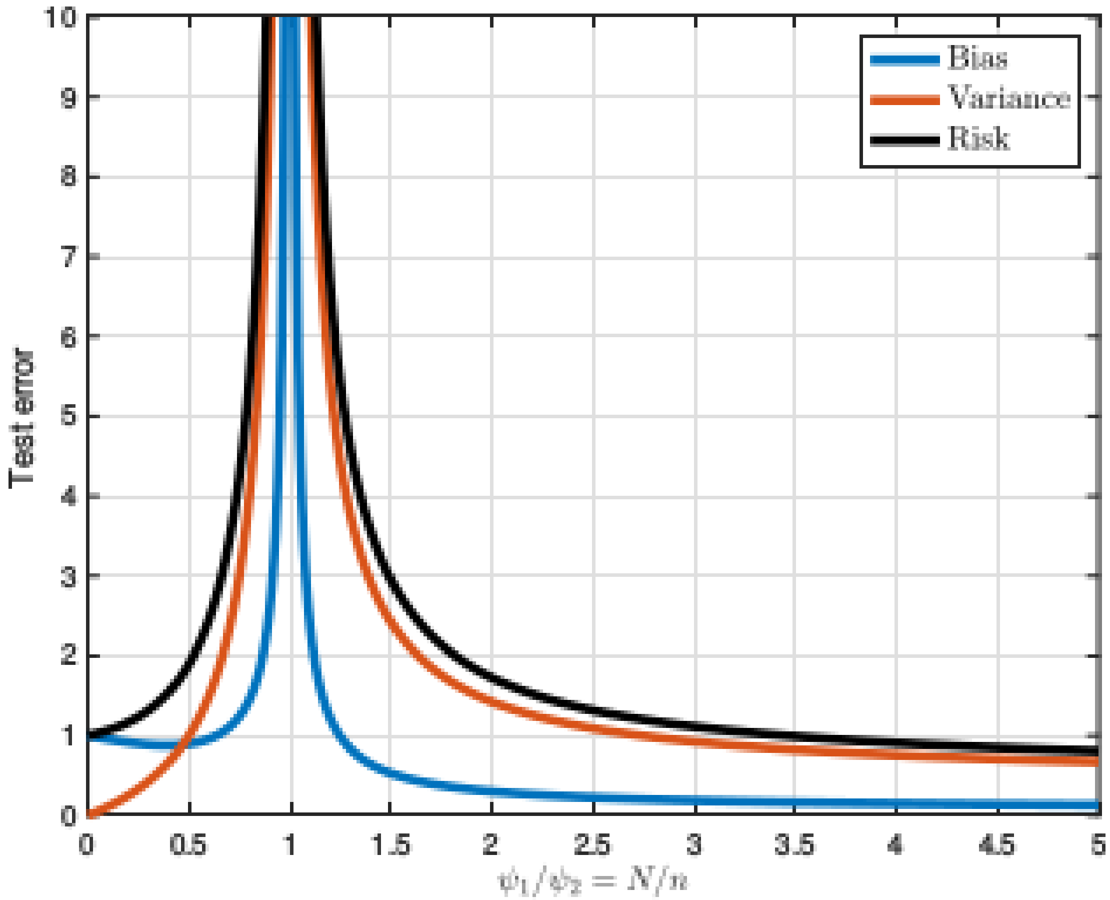
Use computer simulation to assess **all** sources of error

Revisiting Fixed-Design Case



Mei and Montanari, 2019

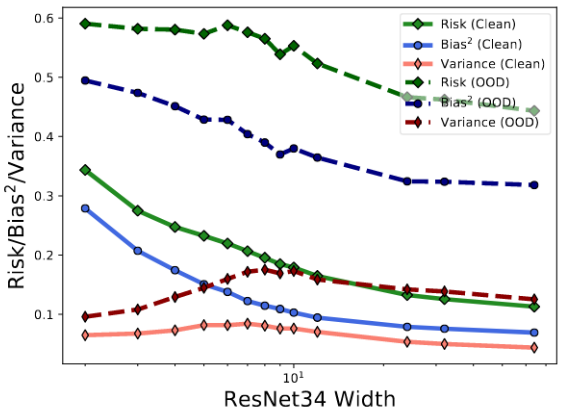
Revisiting Fixed-Design Case



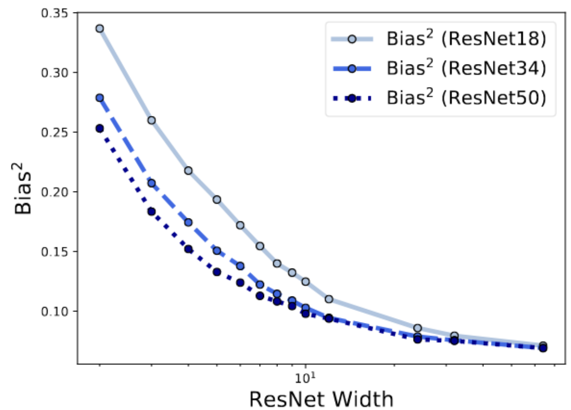
Mei and Montanari, 2019

Fixed-design: attributes some **variance** to **bias**.

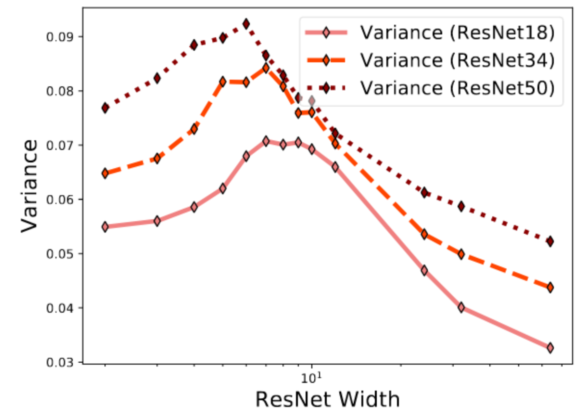
Effect of Depth



(a) OOD Example

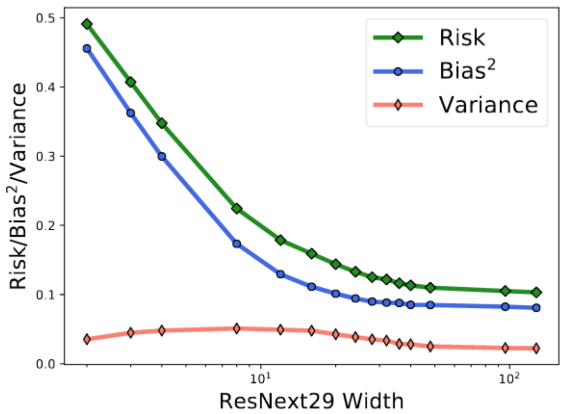


(b) Bias of model with different depth

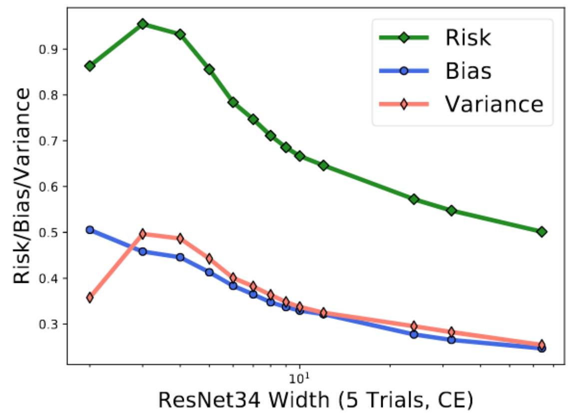


(c) Variance of model with different depth

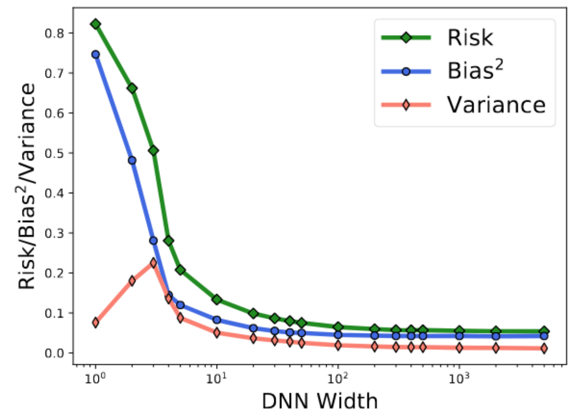
More Robustness Checks



(a) ResNext29, MSE loss, CIFAR10



(b) ResNet34, CE loss, CIFAR10



(c) DNN, MSE loss, MNIST

Ongoing Work

Extensions to classification (e.g. Montanari, Ruan, Sohn, Yan 2020)

Bias-variance for other settings (e.g. Yu, Yang, Dobriban, Steinhardt, Ma 2021)

Characterizing when more data hurts (e.g. Raghunathan, Xie, Yang, Duchi, Liang 2020)

Using random features models to explain scaling laws (e.g. Bahri, Dyer, Kaplan, Lee, Sharma 2021)