# Neural Networks and Deep Learning

## Bias/Variance Tradeoff and Regularization II

Adam Bloniarz

Department of Computer Science
adam.bloniarz@colorado.edu

March 16, 2020

University of Colorado **Boulder**

## Ridge regression

In ridge regression, we add a term to the likelihood to penalize the norm of the coefficient vector:

$$l(\boldsymbol{\beta}) = \frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|^2$$

$$\arg\min_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) = \left(\frac{1}{n}\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \left(\frac{1}{n}\mathbf{X}^T\mathbf{y}\right)$$

## Ridge regression

Consider covariates and outcome with mean-0. Assume covariates are independent. Assume outcome $\mathbf{y}$ has standard deviation of 1. Then

$$\frac{1}{n}\mathbf{X}^T\mathbf{X} = \begin{bmatrix} a_1^2 & & \\ & \ddots & \\ & & a_p^2 \end{bmatrix} \qquad \frac{1}{n}\mathbf{X}^T\mathbf{y} = \begin{bmatrix} a_1 \text{corr}(\mathbf{x}_1, \mathbf{y}) \\ \vdots \\ a_p \text{corr}(\mathbf{x}_p, \mathbf{y}) \end{bmatrix}$$

Therefore, coefficient of ridge solution is:

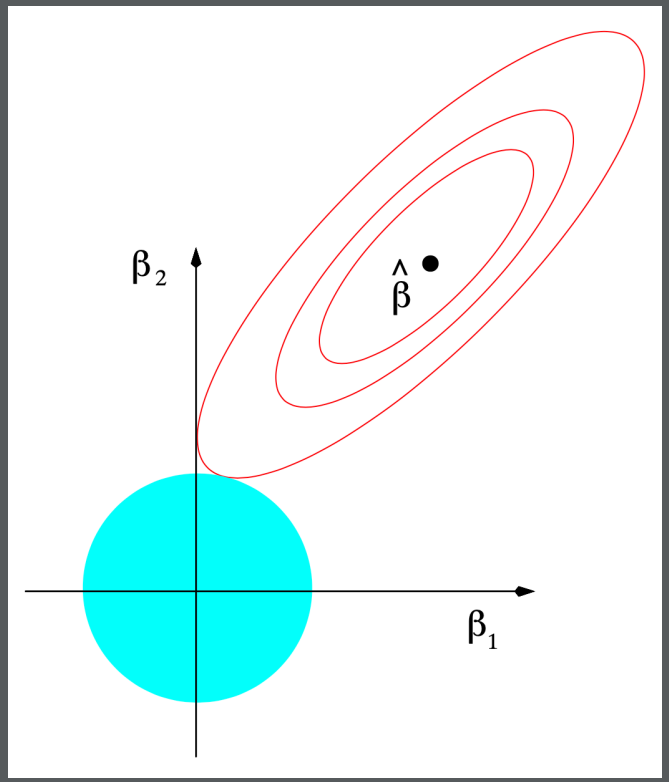
$$\hat{\beta}_j = \frac{1}{a_j + \frac{\lambda}{a_j}} \text{corr}(\mathbf{x}_j, \mathbf{y})$$

## Ridge regression bias-variance tradeoff

Assume that $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, and independent covariates as above. Then one can show that

$$\left|\mathbb{E}\hat{\beta}_j - \beta_j\right| = \frac{\lambda}{a_j^2 + \lambda}\beta_j$$

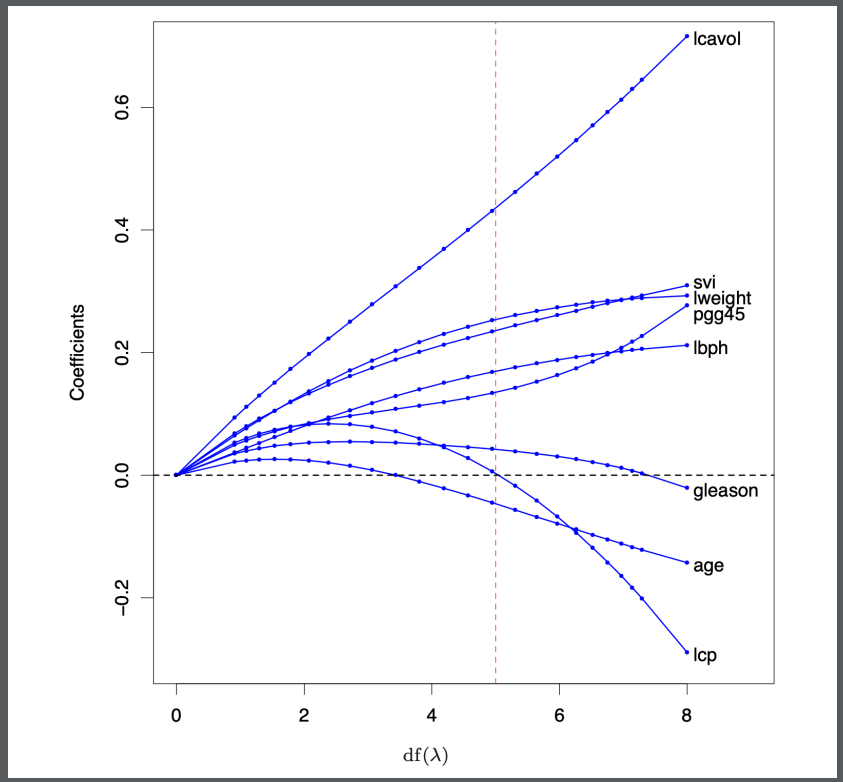
$$\mathrm{Var}\hat{\beta}_j = \frac{\sigma^2 a^2}{(a^2 + \lambda)^2}$$




Source: Hastie, Tibshirani *The Elements of Statistical Learning, figure 3.11*

Ridge regression is equivalent to solving least squares subject to an $l_2$ ball constraint. This pulls the solution toward the origin.

# Ridge regression solution path




Source: Hastie, Tibshirani *The Elements of Statistical Learning, figure 3.8*

As the penalty is increased (small df($\lambda$)), the coefficients converge to zero.

## Ridge regression

Consider a gradient descent update for ridge regression:

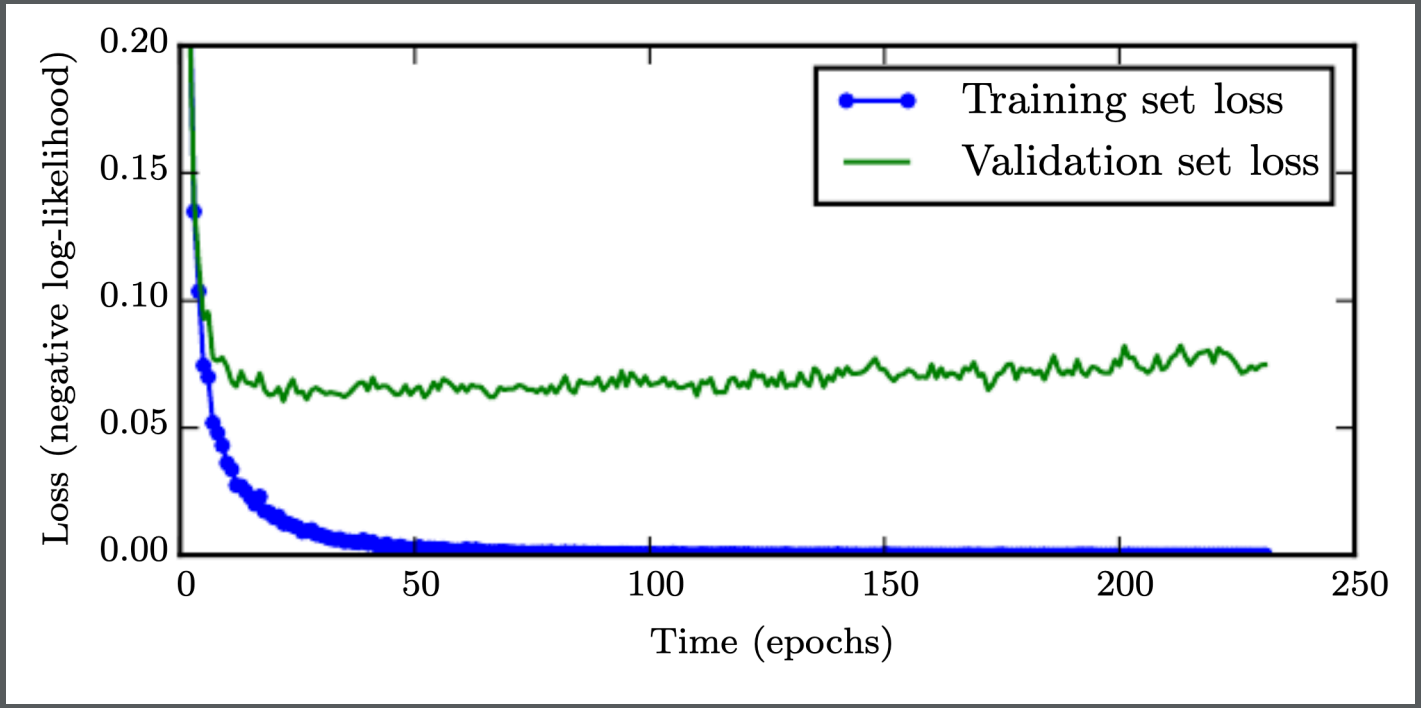
$$\begin{aligned}\boldsymbol{\beta}_{t+1} &\leftarrow \boldsymbol{\beta}_t + \eta\left[\frac{1}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_t) - \lambda\boldsymbol{\beta}_t\right] \\ &= (1 - \eta\lambda)\boldsymbol{\beta}_t + \eta\frac{1}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_t)\end{aligned}$$

We can think of this as applying a weight decay prior to applying the update.

Besides weight decay, what other regularization methods are there for deep learning?

- Early stopping
- Data augmentation
- Dropout
- Label smoothing

# Early stopping




Source: deeplearningbook.org

Training / validation loss on MNIST

## Early stopping

- Validation error may rise while training error continues to decrease.
- Rather than waiting until training error converges, stop training early.
- Can be shown to have good properties for kernel regression (Raskutti, Wainwright, Yu 2014).
  - » "Roughly speaking, it is clear that each step of an iterative algorithm will reduce bias but increase variance ..."

Test-set error is noisy. Early stopping requires a strategy to smooth over random fluctuations:

- Patience: how many iterations to allow error to increase before stopping.
- Number of steps allowed without improvement.

## Data Augmentation - Visual Data

Augmenting an existing dataset can, in effect, create more examples → better generalization. We augment using transformations that we want our classifier to ignore.

- Mirroring
- Random cropping
- Rotation
- Color shifting

## Data Augmentation - Text Data

Augmenting an existing dataset can, in effect, create more examples → better generalization.

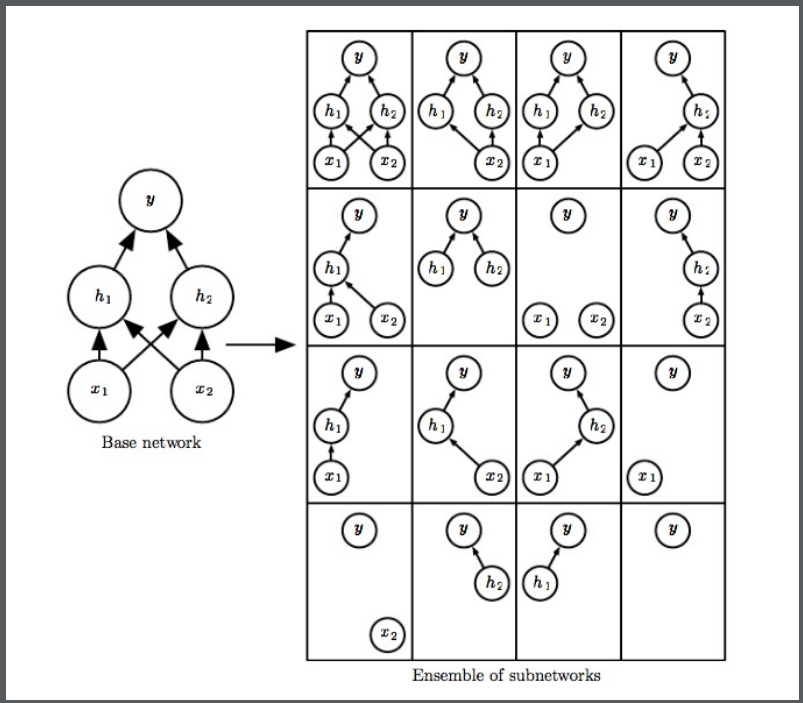
- Replacing words with synonyms
- Word dropout (e.g. Deep Unordered Composition Rivals Syntactic Methods for Text Classification)

# Dropout

1. Dropout
   - On each training trial, randomly remove a portion of units (hidden and possibly input)
     - leads to robustness of network to lesions
     - leads to less specificity of hidden unit responses, or more sharing of responsibility
     - leads to better generalization
   - On each test trial, leave all units in, but reweight connections with factor of $\alpha$ (expectation of contribution)

# Dropout

## Dropout as an ensemble method




Source: deeplearningbook.org

## Dropout

With H hidden units, each of which can be dropped, we have $2^H$ possible models

Each of the $2^{H-1}$ models which include hidden unit h must share the same weights for the units

- serves as a form of regularization
- makes the models cooperate

Including all hidden units at test with $\alpha$-scaling is equivalent to computing the geometric mean of all $2^H$ models

- Exact equivalence with one hidden layer
- "Pretty good approximation" according to Geoff with multiple hidden layers

# Dropout

Advantages

- Computationally cheap
- Seems to work better than weight penalties

Disadvantages

- Sometimes doesn't make a difference, sometimes makes test error worse
- Adds noise to gradient descent, which makes it hard to control learning rates and know when training has reached asymptote

## Label smoothing

Recall the standard loss function for multi-class classification:

$$l = -\sum_k \mathbf{1}_{\{y=k\}} \log \hat{p}_k$$

This is also called cross-entropy loss, because we can think of it as an entropy calculation.

Let $p$ be the distribution over classes that puts all weight on the correct class, i.e. $p(k) = \mathbf{1}_{\{y=k\}}$. Let $\hat{p}$ be the distribution inferred by the model (i.e. the output of the softmax layer).

$$l = -H(p, \hat{p}) = \mathbb{E}_p \log \hat{p}$$

## Label smoothing

With label smoothing, we smooth out the 'true' probability distribution:

$$p_k^{LS} = p_k(1-\alpha) + \frac{\alpha}{K}$$

For example, for $\alpha = 0.1$:

$$\text{if } p = (1, 0, 0, 0)$$
$$\text{then } p^{LS} = (0.925, 0.025, 0.025, 0.025)$$

Intuition: prevents overconfident predictions.

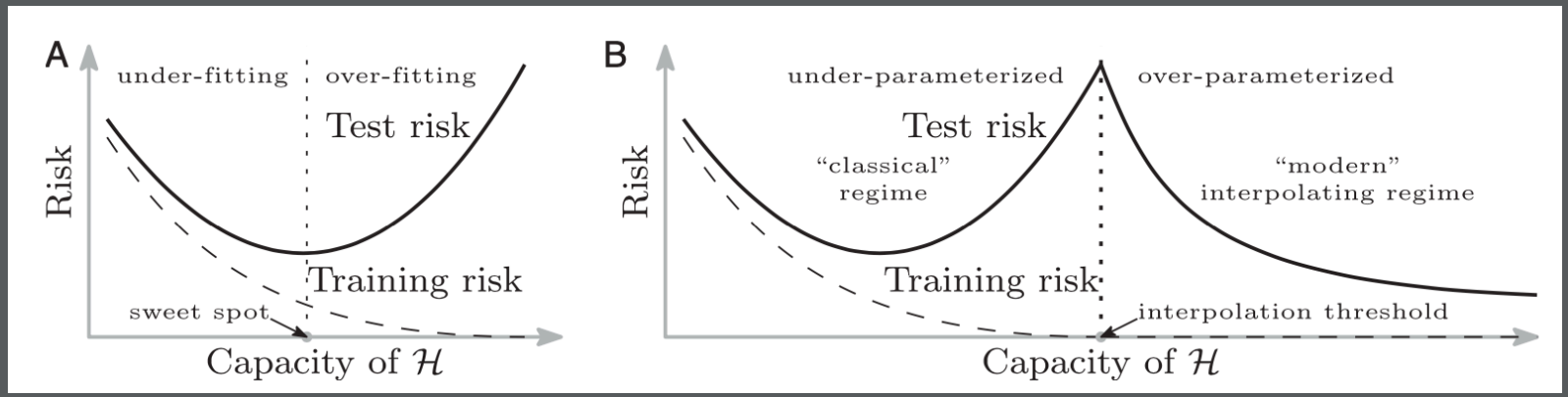
See Szegedy Vanhoucke, Ioffe, 2016: Rethinking the inception architecture for computer vision

Statistical learning theory provides bounds of the sort:

$$\text{Generalization error} \leq \text{Training error} + f(\text{model complexity}, n) \quad \text{w.h.p.}$$

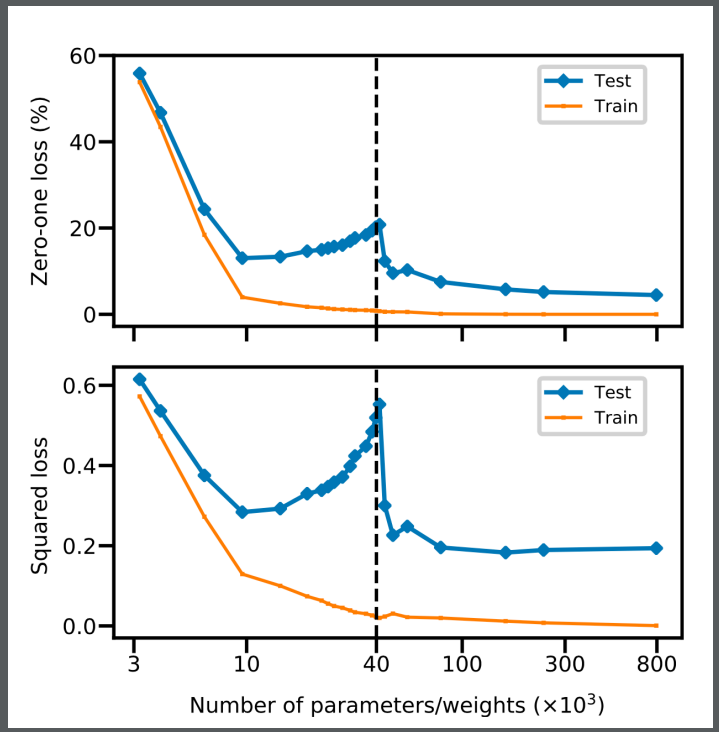
Where model complexity is measured with VC-dimension / Rademacher complexity.

- Observation: modern neural nets are overparametrized and can easily get to training error of 0.
- **They can achieve 0 training error even with fully randomized labels!**
- Statistical learning theory does not explain this behavior (it gives an uninformative bound).
- See Understanding deep learning requires rethinking generalization.
- Unlike prior ML methods (regression, kernel methods), regularization is a tuning parameter, rather than something necessary to prevent trivial solutions.
- Deep learning often only does marginally worse without regularization!



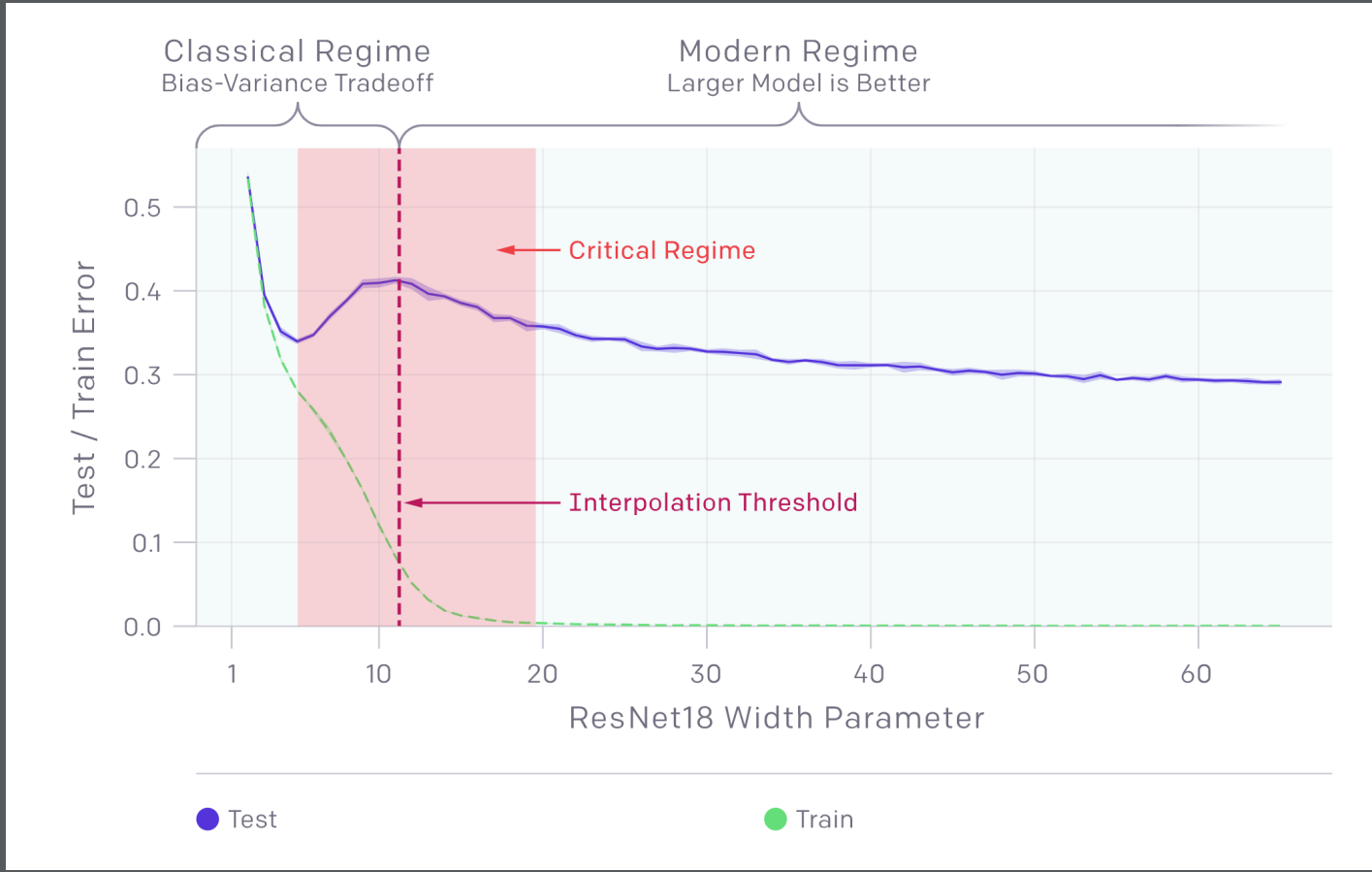

Source: Reconciling modern machine-learning practice and the classical biasvariance trade-off

- Many model types actually exhibit this behavior (not just deep learning).
- Highest error is right at the interpolation threshold.
- More parameters allows training procedure to find a smaller-norm submodel in a larger space.

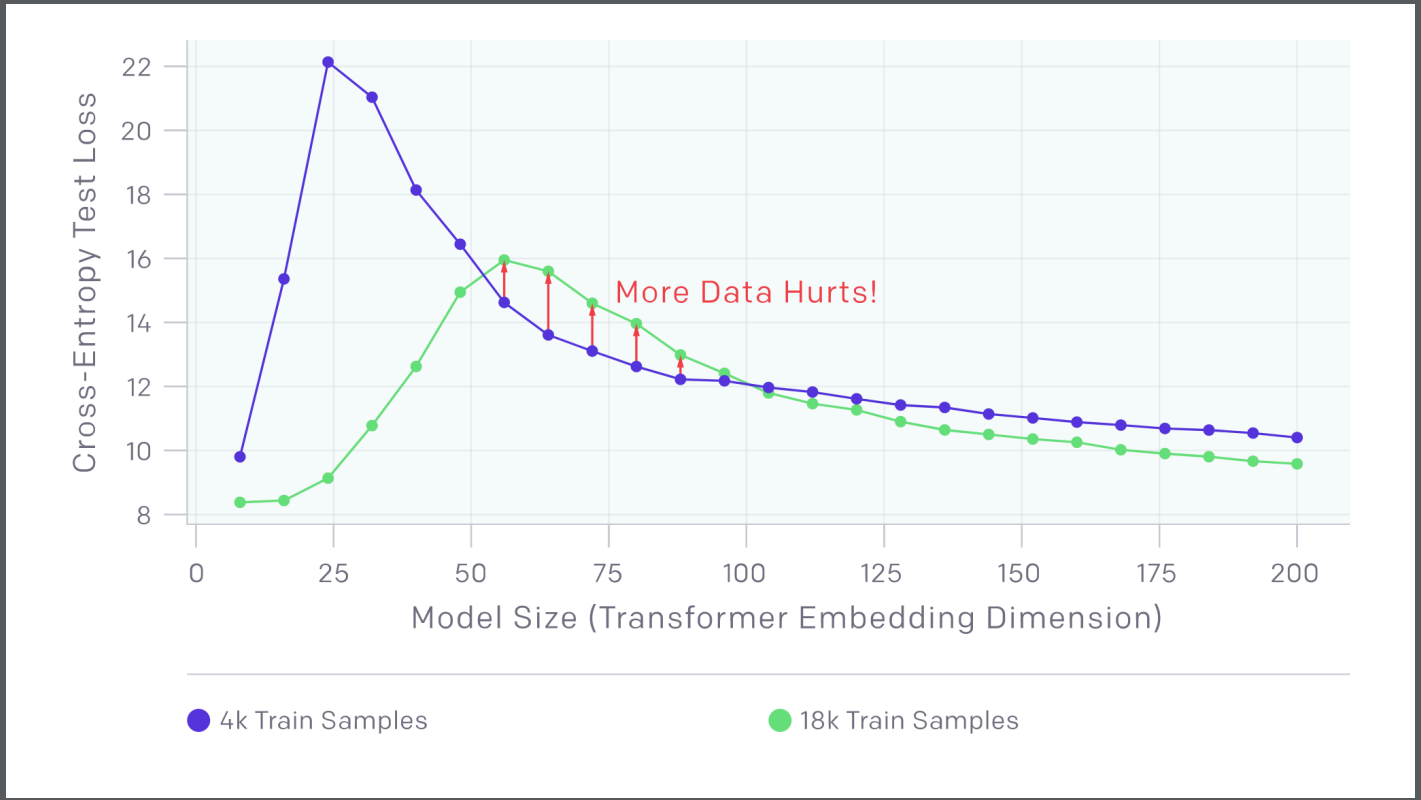



Source: Reconciling modern machine-learning practice and the classical biasvariance trade-off

- Conjecture that NN training algorithms have a "small-norm" inductive bias.

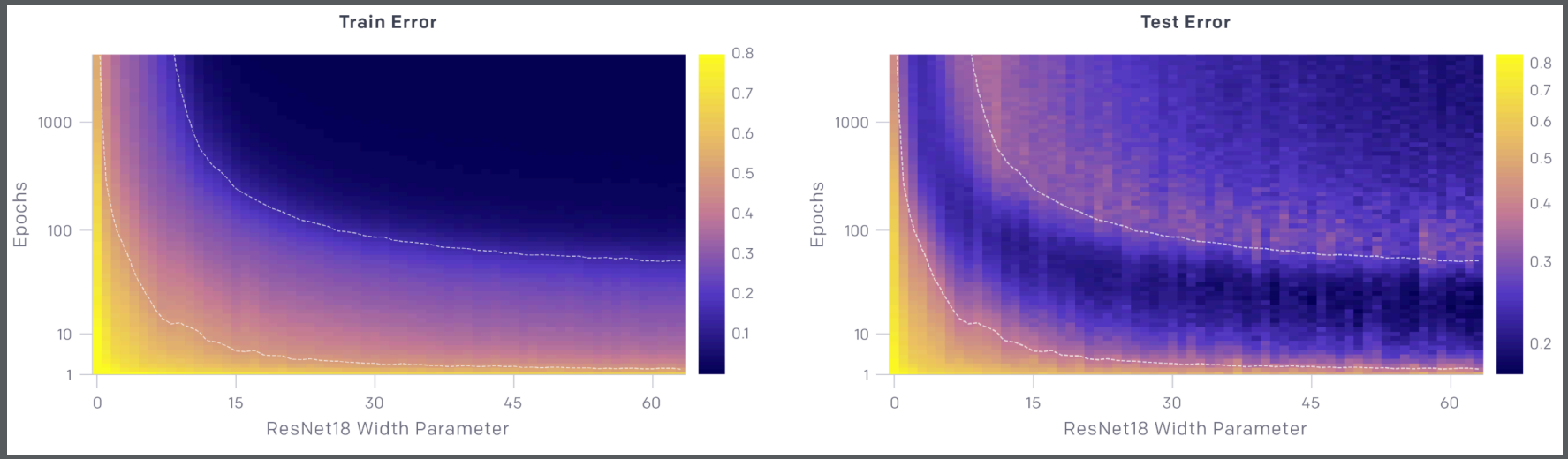



Source: Deep double descent




Source: Deep double descent

More data with same model: performance is worse!




Source: Deep double descent

Double descent occurs both w.r.t number of training epochs and model size.