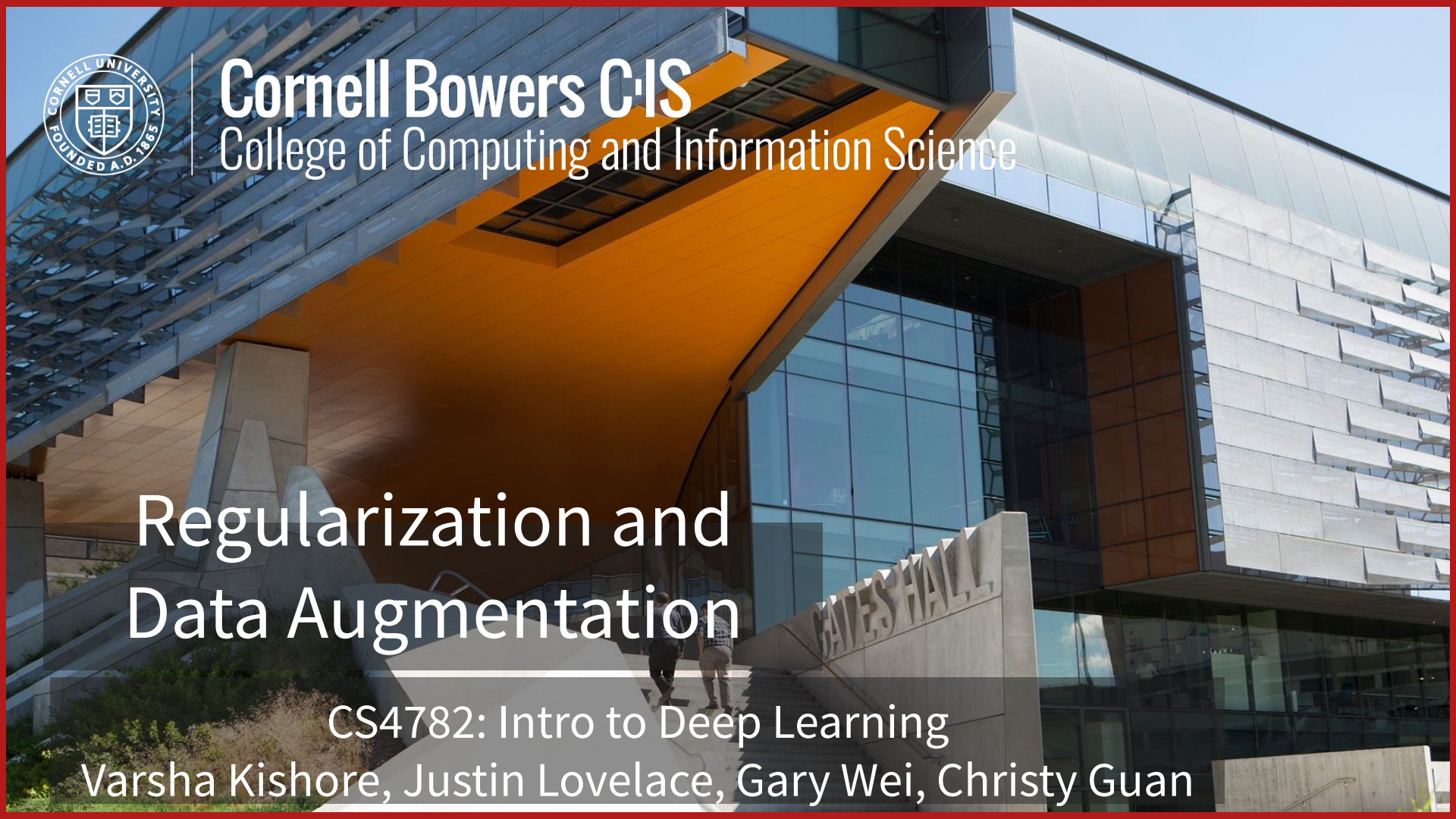


Regularization and Data Augmentation

CS4782: Intro to Deep Learning

Varsha Kishore, Justin Lovelace, Gary Wei, Christy Guan

Recap- Optimizers

- Gradient Descent
  - *Vanilla, costly, but for best convergence rate*
- Stochastic Gradient Descent
  - *Simple, lightweight*
- **Mini-batch SGD**
  - *balanced between SGD and GD*
  - ***1st choice for small, simple models***

- SGD w. Momentum
  - *Faster, capable to jump out local minimum*

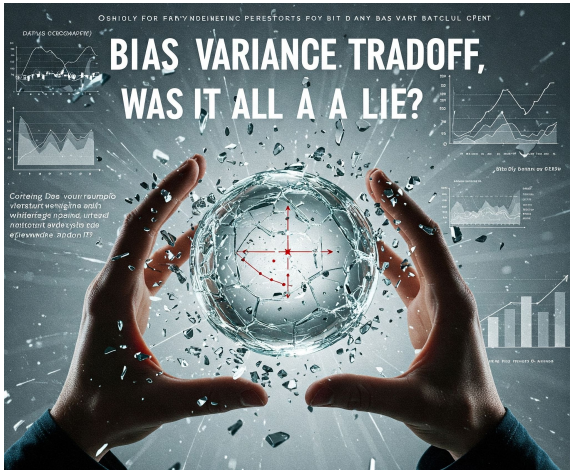
- AdaGrad
- RMSProp
- **Adam**
  - **Just use Adam if you don't know what to use in deep learning**

Agenda

- Backpropagation
- Optimizers
  - Gradient Descent
  - Stochastic Gradient Descent
  - SGD w. Momentum
  - AdaGrad
  - RMSProp
  - Adam
- **Learning rate scheduling**

Agenda

- Motivation behind regularization
- Regularization in deep learning
- Data Augmentation
- Normalization methods



Are all Optimizers equivalent somehow?

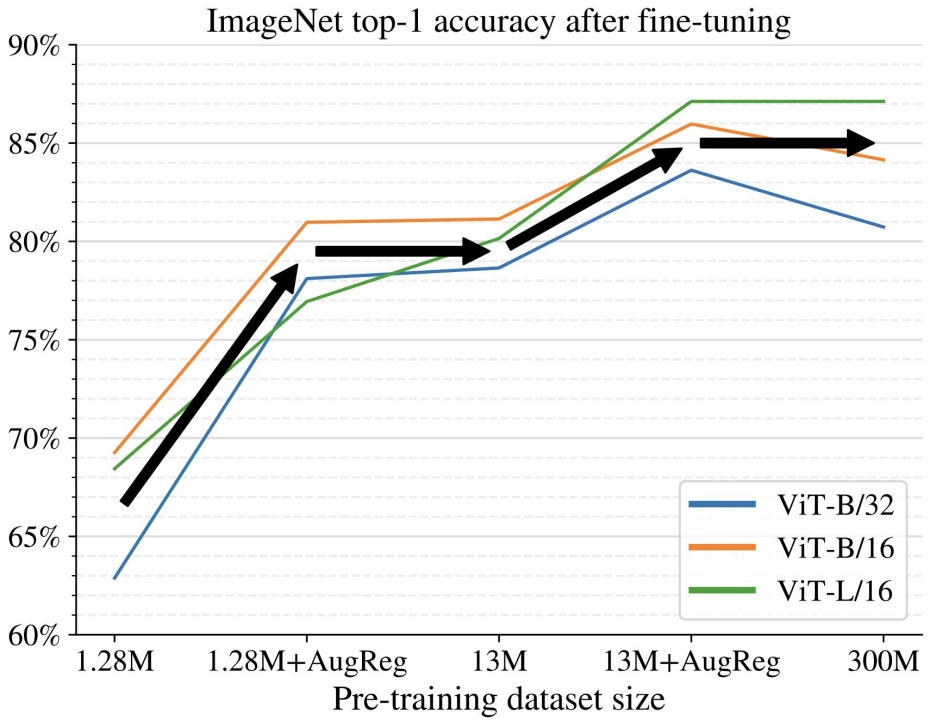
No!

There are *many* minimizers of the training loss

The **optimizer** determines which minimizer you converge to

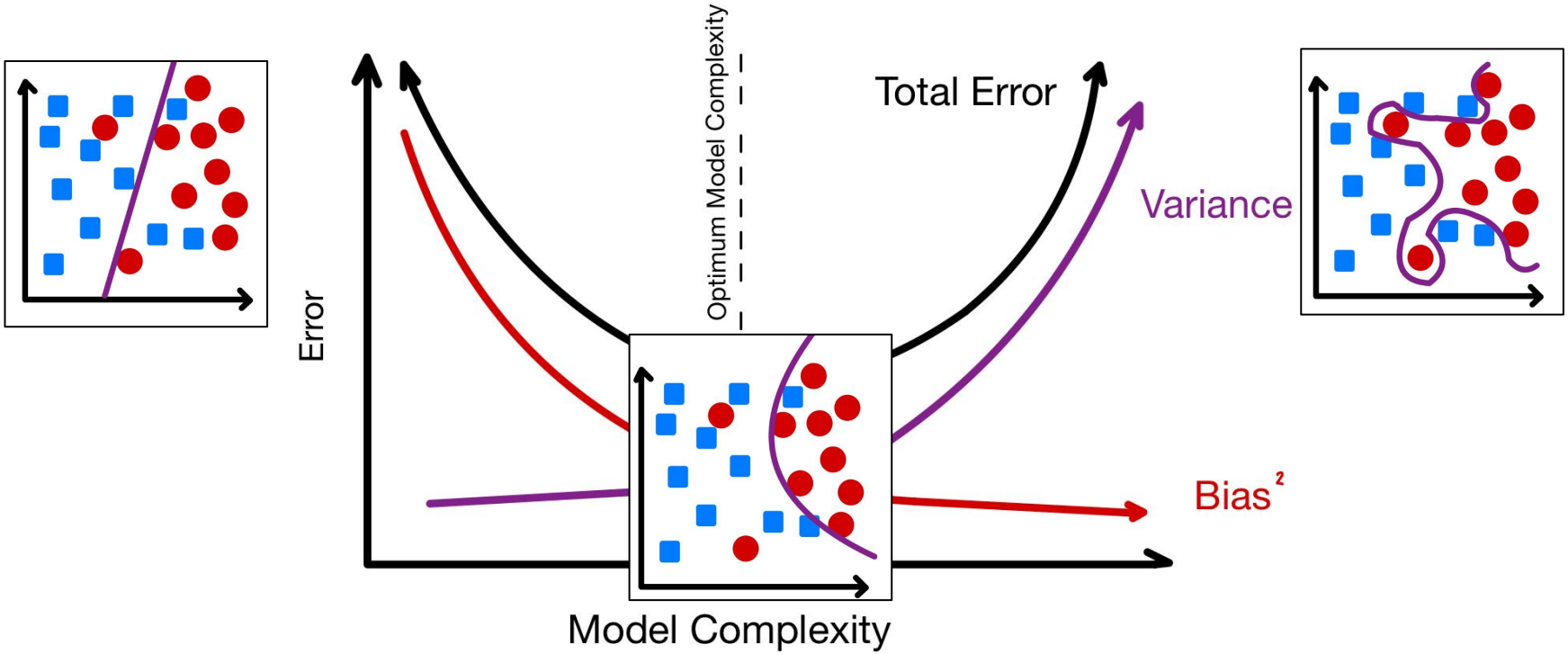
Why do we care?

- Regularization and data augmentation are really effective!
- Can be worth millions of additional training images

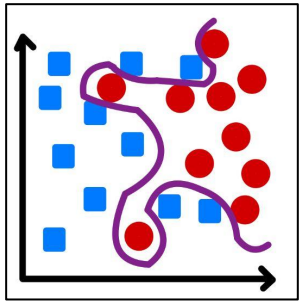


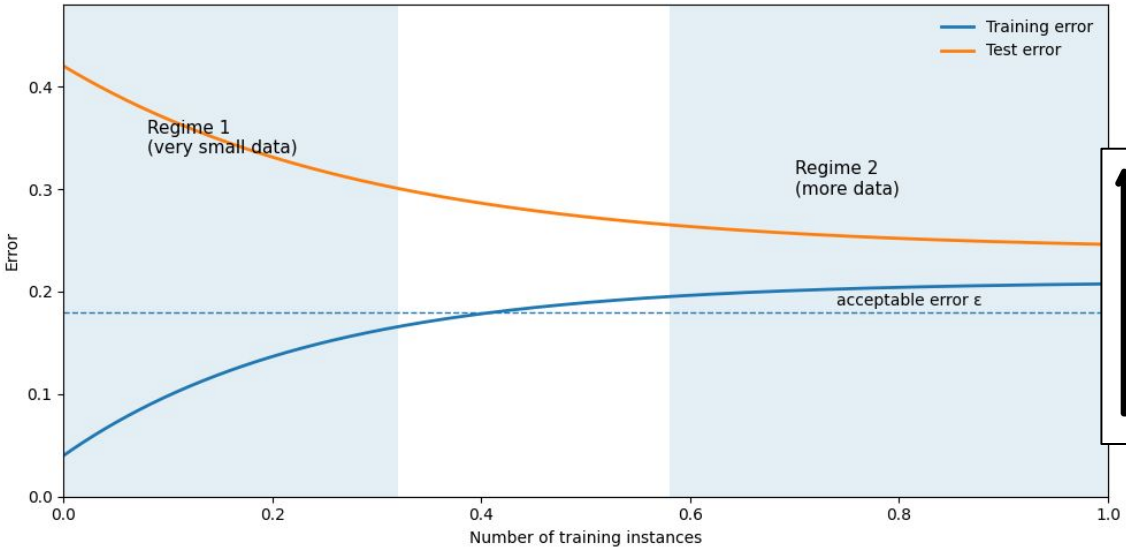
"How to train your ViT? Data, Augmentation,and Regularization in Vision Transformers", by Steiner et al. 2022

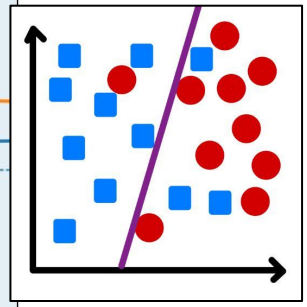
Expected Test Error = Variance + Noise + Bias^2



Cornell Bowers CIS







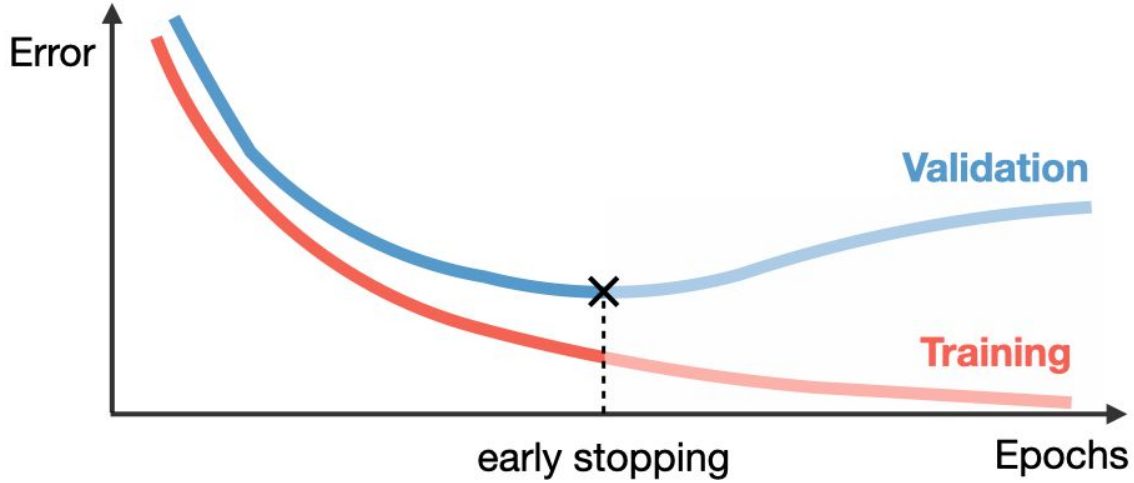
Remedies against overfitting:

- Collecting more training data
- Use a simpler/smaller model
- Early stopping
- Add regularization
  - L2/L1 regularization, weight decay
- Data augmentation

Remedies against underfitting:

- Increase model capacity
  - Add more layers / hidden nodes
  - More expressive architecture
- Optimize for longer
- More aggressive optimization
  - Larger learning rate (e.g. use Batchnorm)
- Reduce regularization

Early Stopping



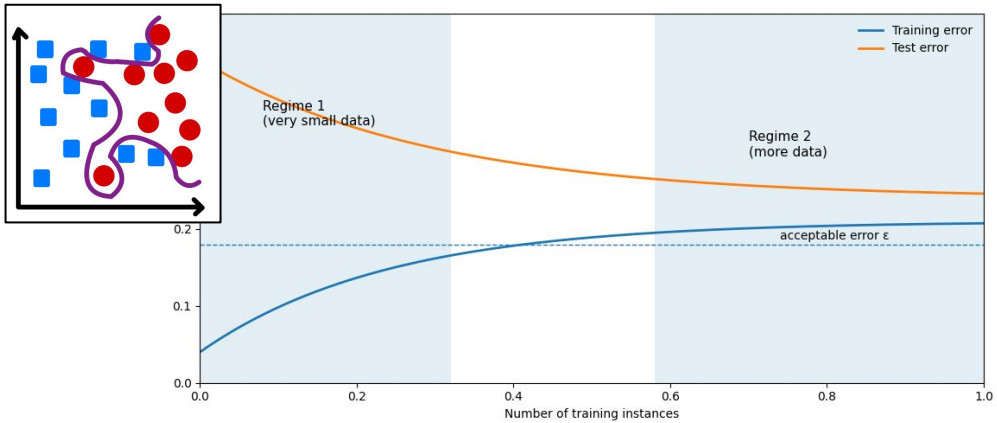
- Pick the training checkpoint with the strongest validation performance
- Easy to implement, should use by default

https://wandb.ai/ayush-thakur/huggingface/reports/Examples-of-Early-Stopping-in-HuggingFace-Transformers--VmlIdzo0MzE2MTM

What is Regularization?

Regularization refers to **techniques** used to prevent machine learning models from overfitting in order to minimize the loss function (Regime 1).

Models that overfit can have large generalization gaps.



Comparing Error and Number of Training Instances

Regularizers

Regularizers are used to quantify the complexity of a model.

Empirical Risk Minimization:

$$\mathbf{w} = \operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \frac{1}{n} \sum_i \ell(\mathbf{w}, \mathbf{x}_i, y_i)$$

Regularized Empirical Risk Minimization:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + \lambda \cdot r(\mathbf{w})$$

where $r(\mathbf{w})$ is some measure of model complexity that we want to control.

Optimization problem:

$$\operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}, D)$$

+L2:

$$\operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}, D) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Gradient update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t, D)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t, D) - \alpha\lambda\mathbf{w}_t$$

Optimization problem:

Gradient update:

$$\operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}, D)$$

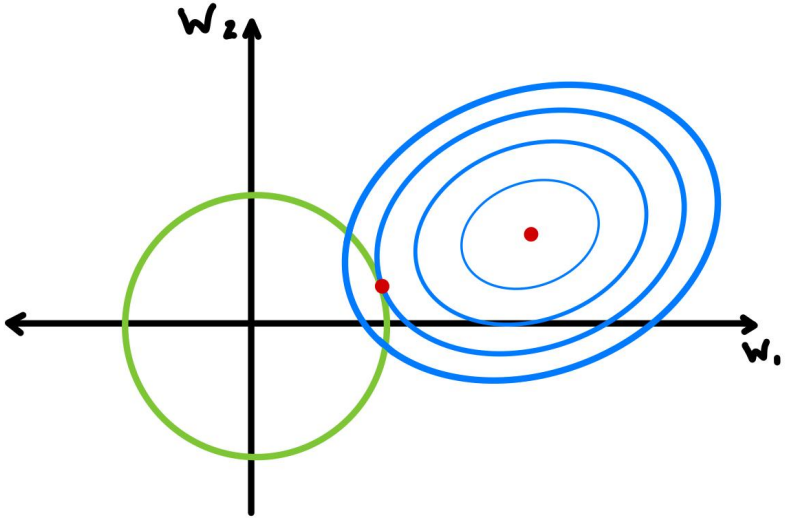
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t, D)$$

+L2: $$\operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}, D) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

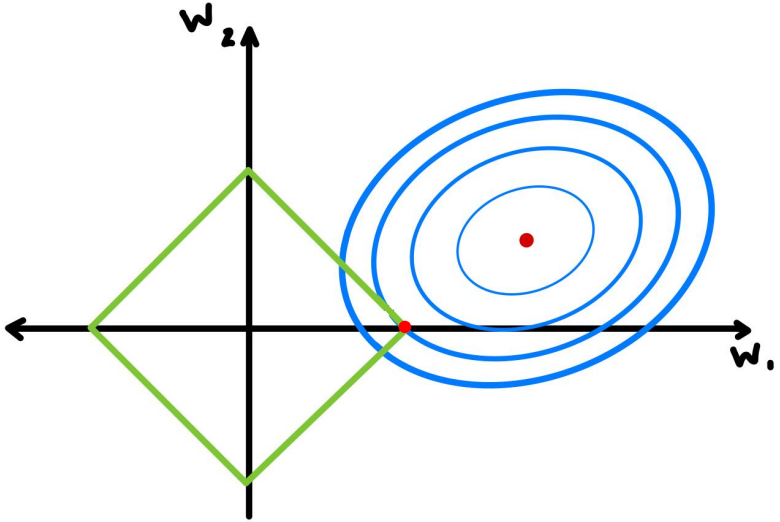
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t, D) - \alpha\lambda\mathbf{w}_t$$

+L1: $$\operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}, D) + \lambda|\mathbf{w}|$$

Geometric Interpretation



$$\underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{w}, D) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$



$$\underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{w}, D) + \lambda|\mathbf{w}|$$

Connection Between Weight Decay and L2 Regularization

Almost the same thing, but subtle differences.

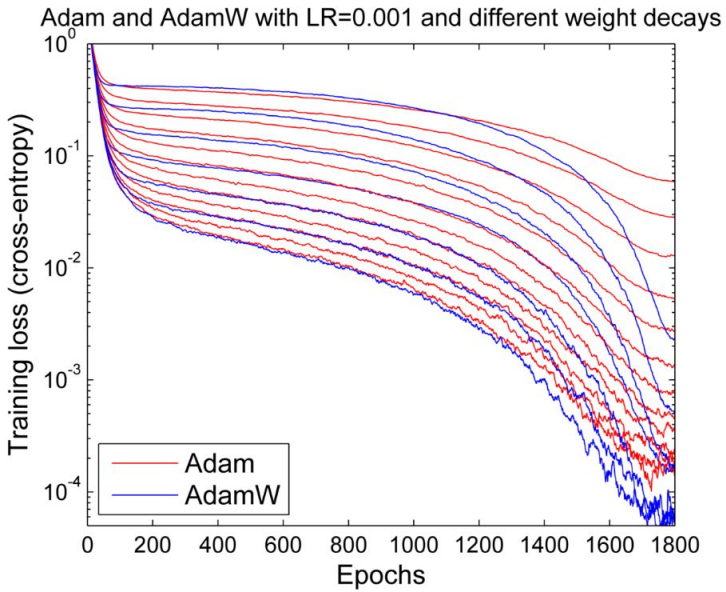
- L2 regularization: Optimizer treats regularizer just like the loss
- Weight Decay: Regularizer is **independent** of optimizer's **adaptive scaling**

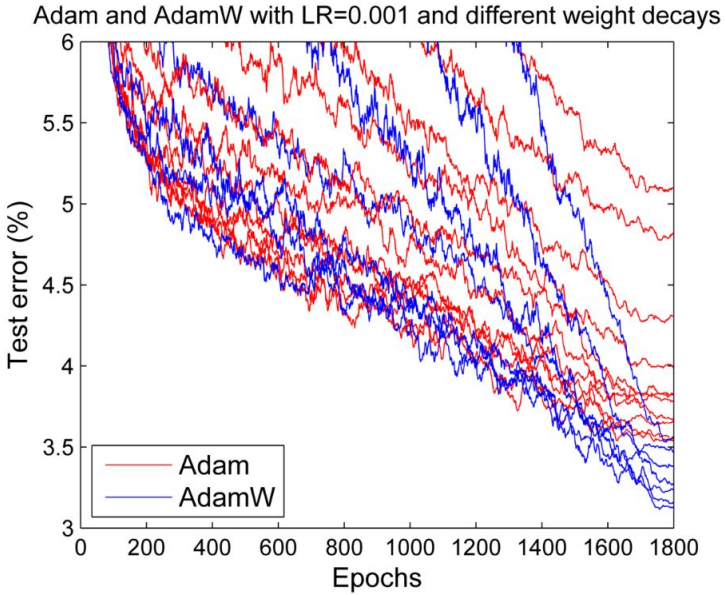
Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

Adam w/ L2 Regularization vs Adam w/ Weight Decay (AdamW)

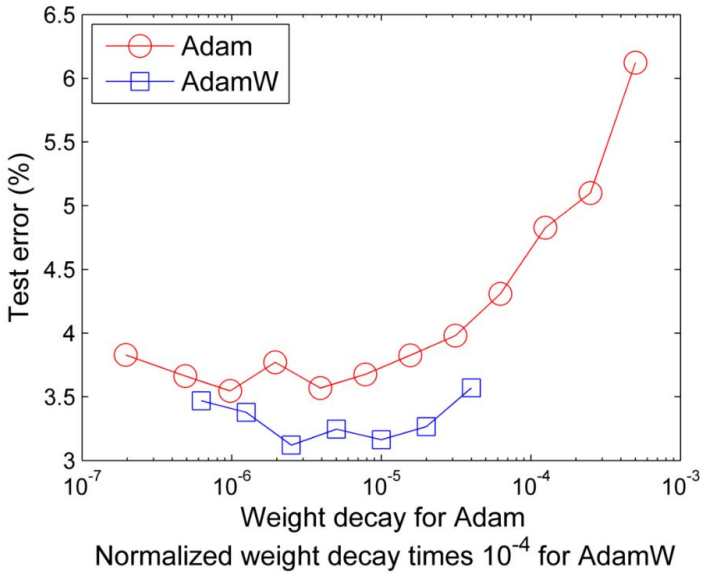
- Weight decay is more effective than L2 regularization when using Adam

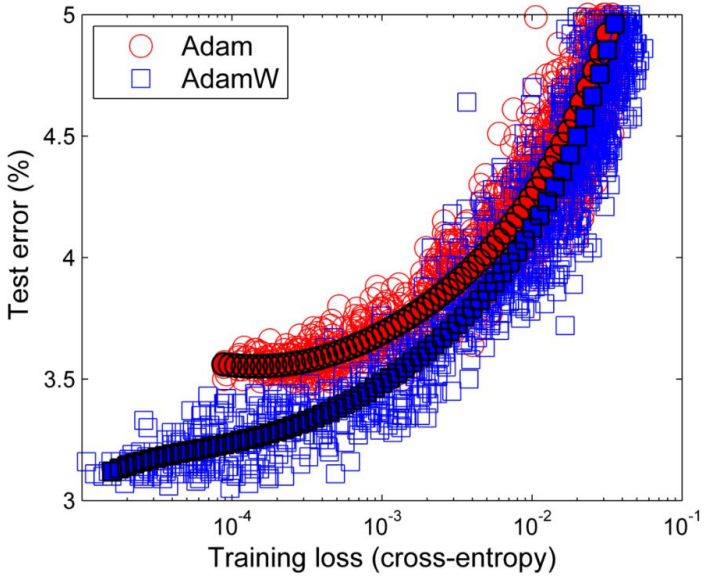




Adam w/ L2 Regularization vs Adam w/ Weight Decay (AdamW)

- Weight decay is more effective than L2 regularization when using Adam





Optimizers Recap

- Gradient Descent
  - *Vanilla, costly, but for best convergence rate*
- Stochastic Gradient Descent
  - *Simple, lightweight*
- **Mini-batch SGD**
  - *balanced between SGD and GD*
  - ***1st choice for small, simple models***

- SGD w. Momentum
  - *Faster, capable to jump out local minimum*

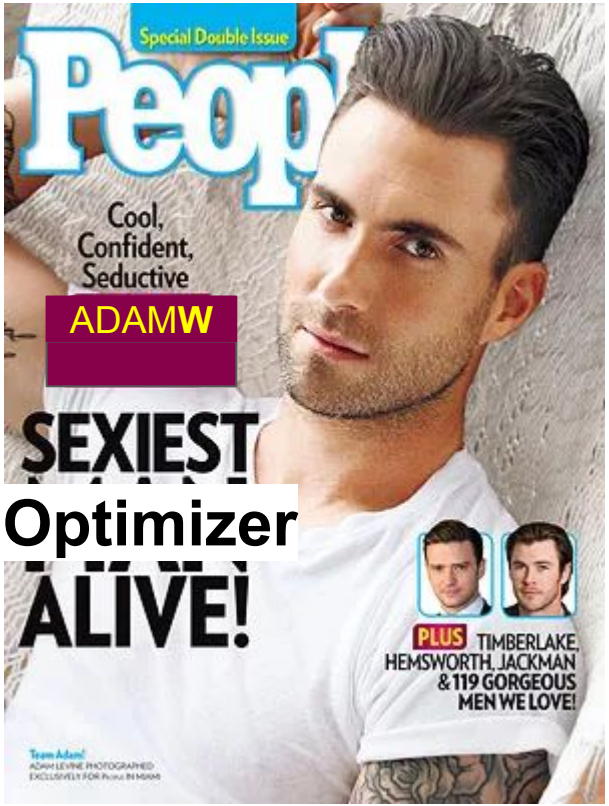
- AdaGrad
- RMSProp
- **Adam**
  - **Just use Adam if you don't know what to use in deep learning**

(Updated) Optimizers Recap

- Gradient Descent
  - *Vanilla, costly, but for best convergence rate*
- Stochastic Gradient Descent
  - *Simple, lightweight*
- **Mini-batch SGD**
  - *balanced between SGD and GD*
  - ***1st choice for small, simple models***

- SGD w. Momentum
  - *Faster, capable to jump out local minimum*

- AdaGrad
- RMSProp
- Adam
- **AdamW**
  - **Just use AdamW if you don't know what to use in deep learning**

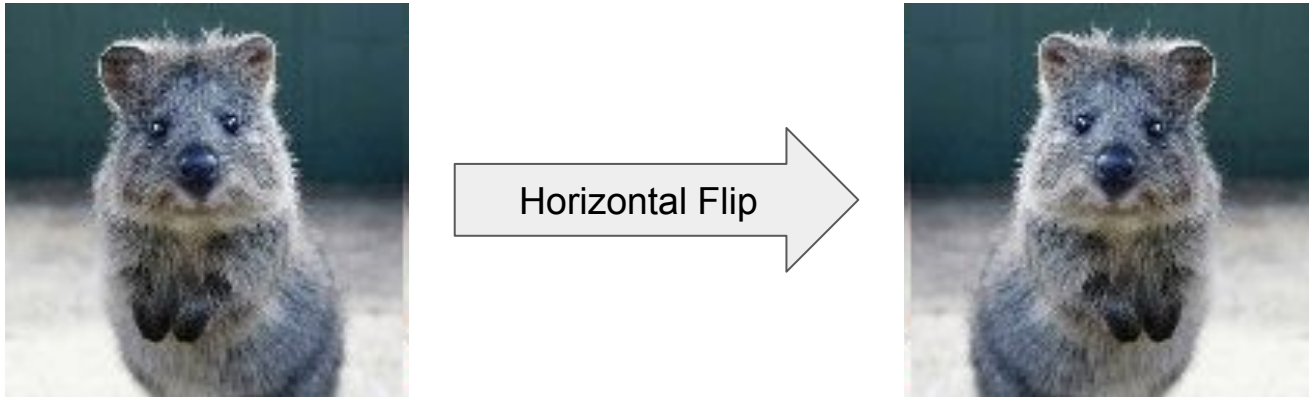


Data Augmentation

Discuss: Image Classification

How can we make a model for image classification more robust?

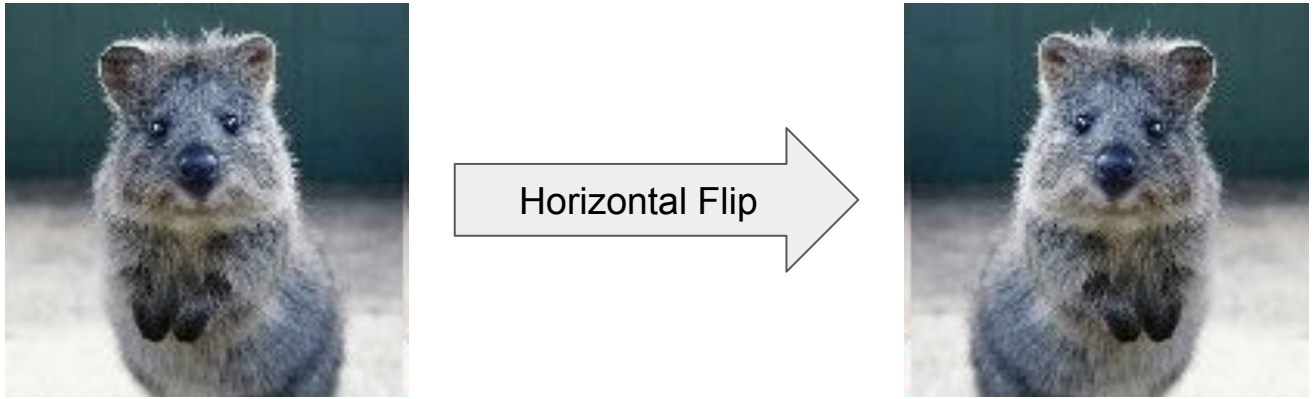
Can we augment the training data without annotating more images?



https://imgaug.readthedocs.io/en/latest/index.html

Data Augmentation!

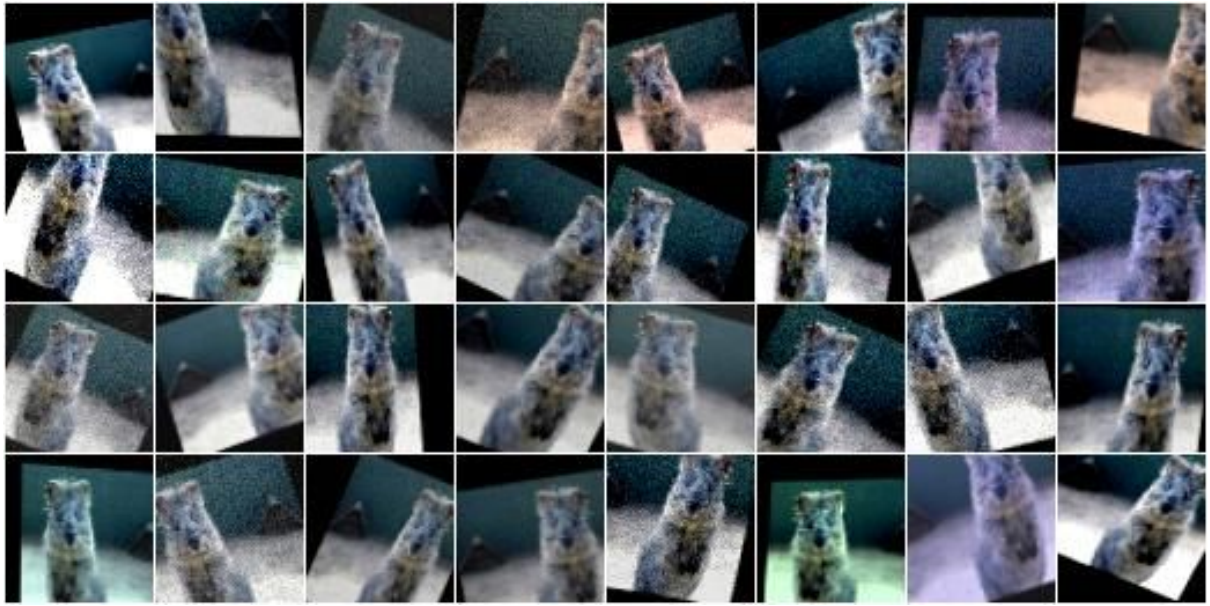
- Use our domain knowledge to transform the image in ways that preserve the class label



https://imgaug.readthedocs.io/en/latest/index.html

Data Augmentations

- Horizontal flips
- Rotate image
- Zoom/crop image
- Brighten/darken image
- Shift colors



https://imgaug.readthedocs.io/en/latest/index.html

Discuss: Text Classification

How can we make a model for sentiment classification more robust?

Can we augment the training data without annotating more examples?

Positive Movie Review:
Still, this flick is fun, and host to some truly excellent sequences.

Negative Movie Review:
begins with promise , but runs aground after being snared in its own tangled plot .

Data Augmentation for Text

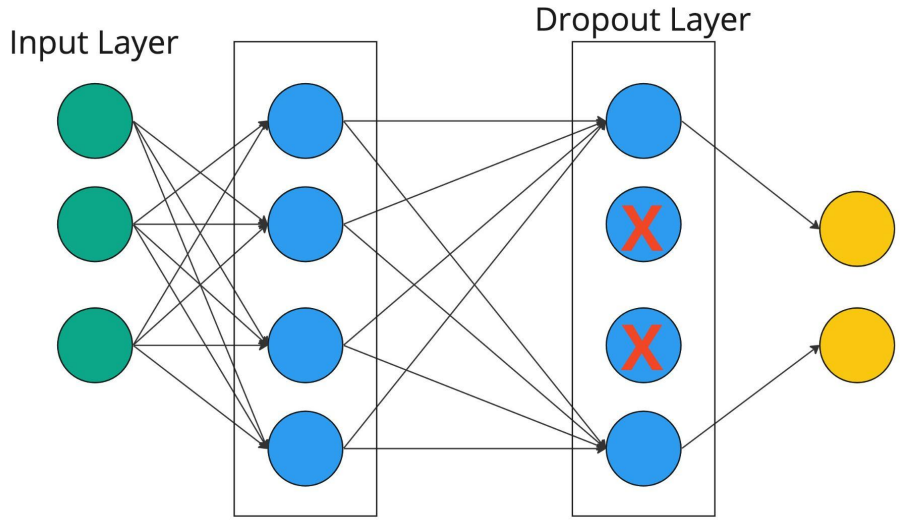
- Much harder for text!
  - How to change the text without breaking the meaning?

DropOut

Dropout

In each forward pass, randomly set some neurons to zero.

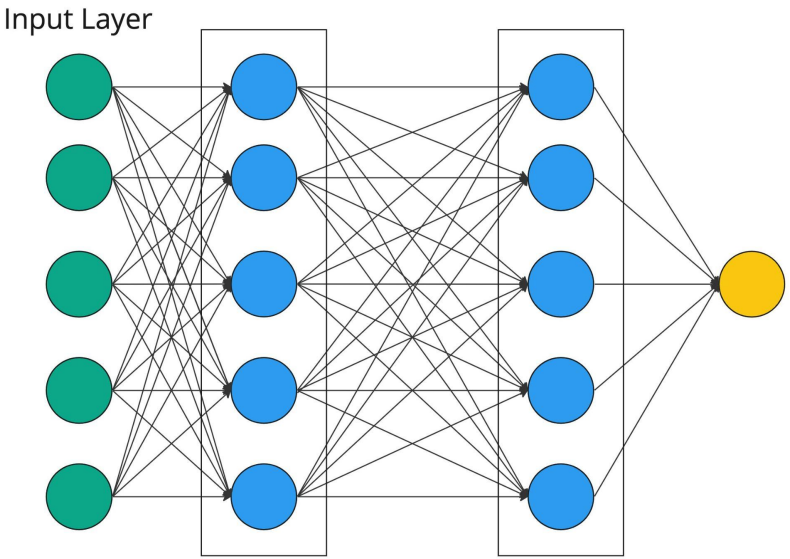
The probability of keeping a neuron is a hyperparameter; p=0.5 is common.



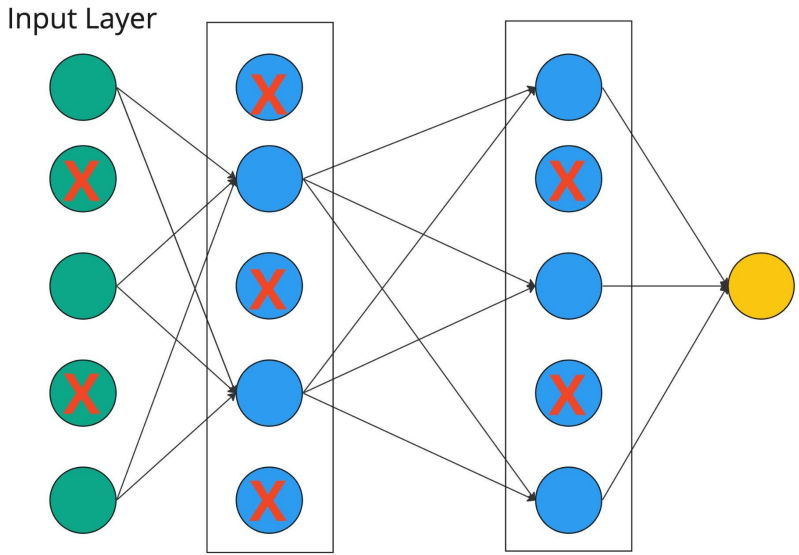
Deep Net with Dropout Layer

[Srivastava et al. 2014]

Implementing Dropout



Standard deep net with two hidden layers

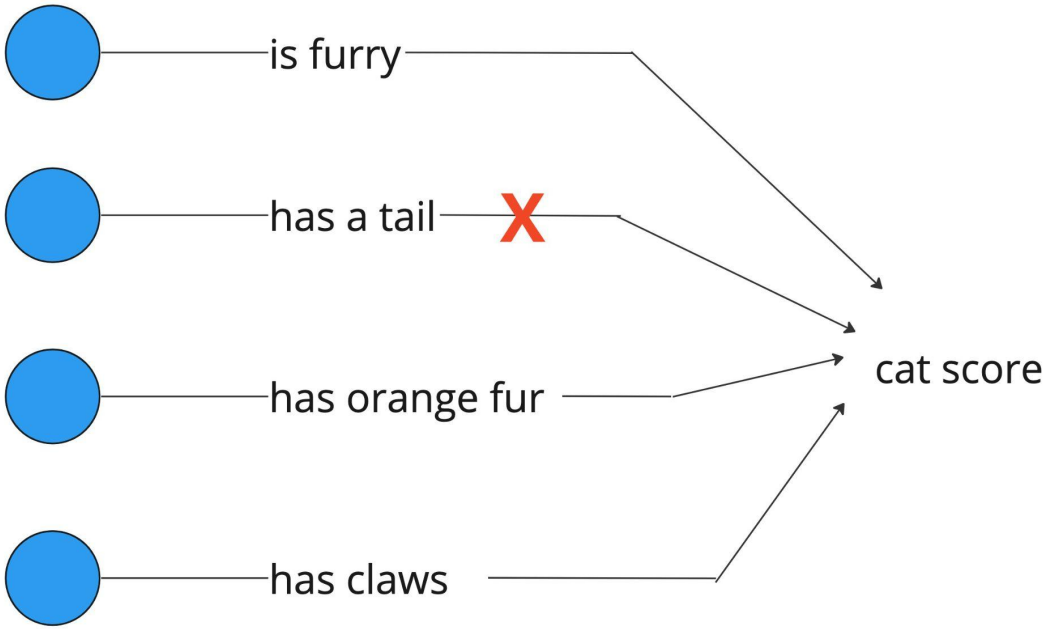


Deep net produced by applying dropout.
Crossed units have been dropped

[Srivastava et al. 2014]

Why is Dropout a good idea?

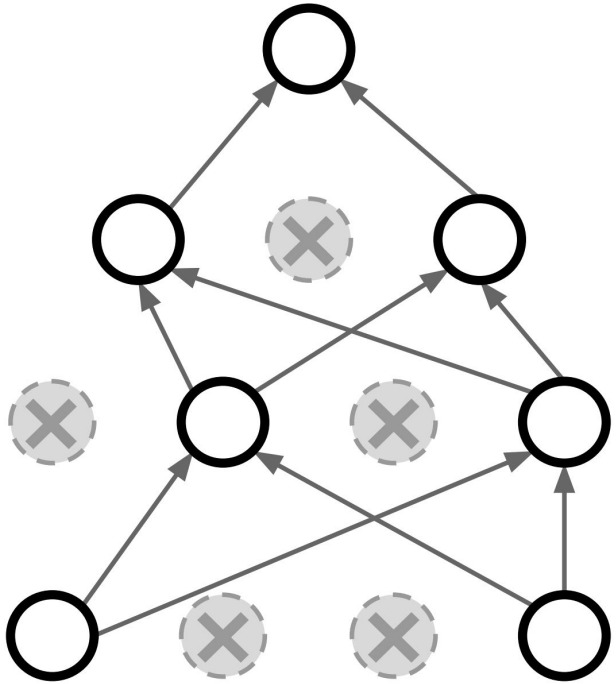
Dropout forces the network to have a redundant representation, which prevents co-adaptation of features.



[Srivastava et al. 2014]

Why is Dropout a good idea?

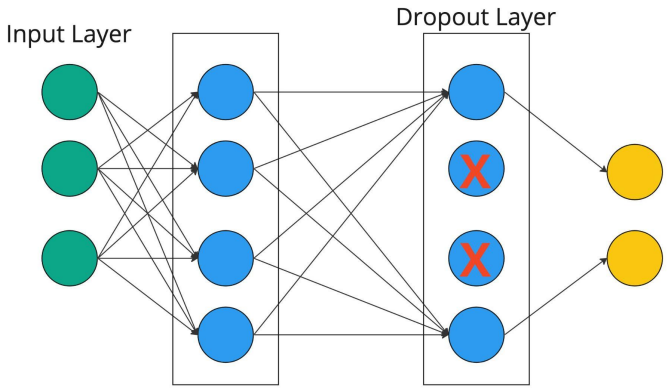
- Another interpretation: Dropout trains a large ensemble of models with shared weights
- Each dropout mask corresponds to a different “model” within the ensemble.
- A fully connected layer with 4096 units has $2^{4096} \sim 10^{1233}$ possible masks!
  - Only ~ 10^{82} atoms in the universe



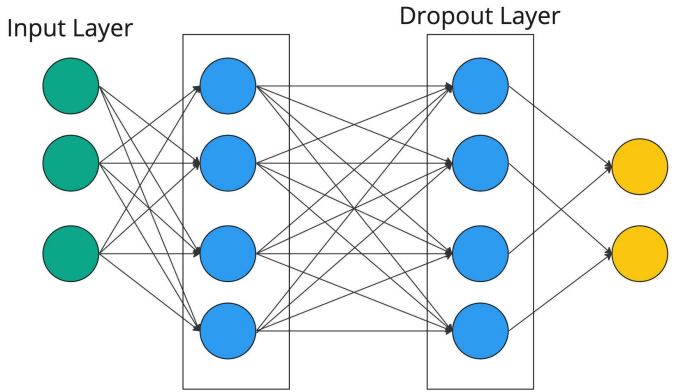
Dropout During Test Time

Use all of the neurons in the network

Does this introduce any problems?



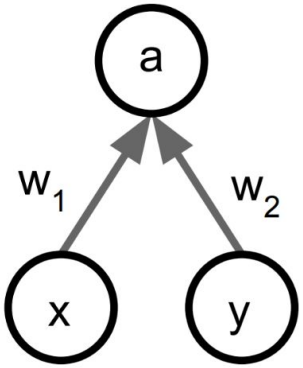
Training Time



Test Time

Dropout During Test Time

Need to re-scale activations so they are the same (in expectation) during training and testing



Consider a single neuron.

At test time we have: $E[a] = w_1 x + w_2 y$

During training we have:

$$E[a] = \frac{1}{4}(w_1 x + w_2 y) + \frac{1}{4}(w_1 x + 0y) + \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2 y) = \frac{1}{2}(w_1 x + w_2 y)$$

At test time, **multiply** by dropout probability

Effectiveness of Dropout

- Improves generalization of neural nets when training with limited data

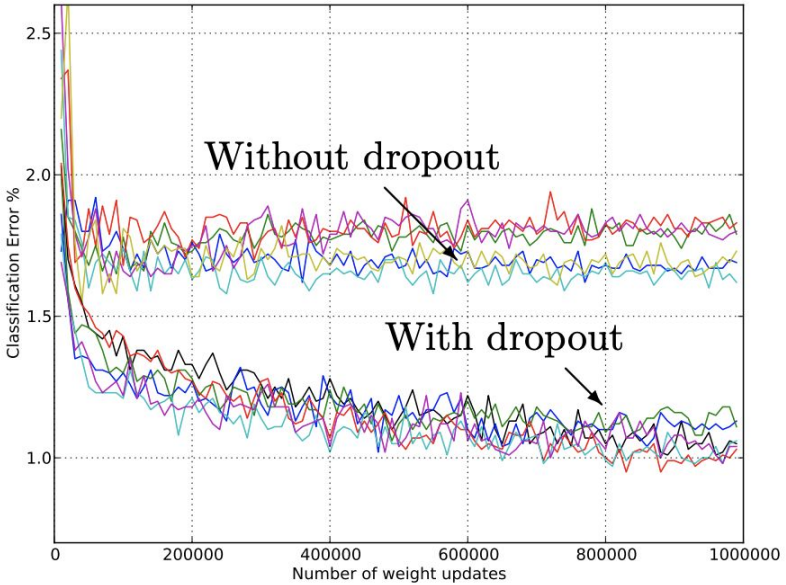


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

"Dropout: A Simple Way to Prevent Neural Networks from Overfitting" by Srivastava et al., 2014

Batch Normalization

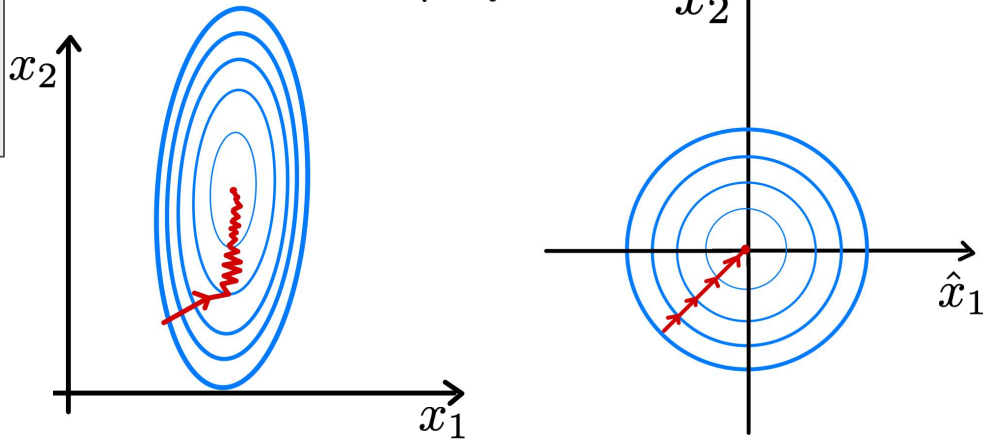
Why should we standardize data?

- Standardization ensures all features have a similar scale
  - Beneficial for optimization
- We do not know a priori which features will be relevant and we do not want to penalize or upweight features
- Example: Predict house prices

x_1 **Bedrooms: 1 to 5**

x_2 **Square footage: 0 to 2000 sq feet**

$$\hat{x}_i \leftarrow \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$



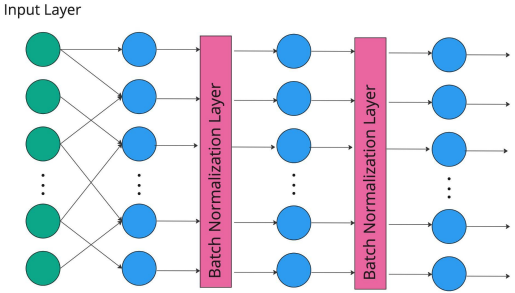
Efficient Backprop [LeCun et al. 1998]

Cornell Bowers CIS

Batch Normalization

[Ioffe and Szegedy 2015]
Batch Normalization normalizes the intermediate features in neural networks.

We standardize the inputs to each layer by normalizing the output of the prior layer



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

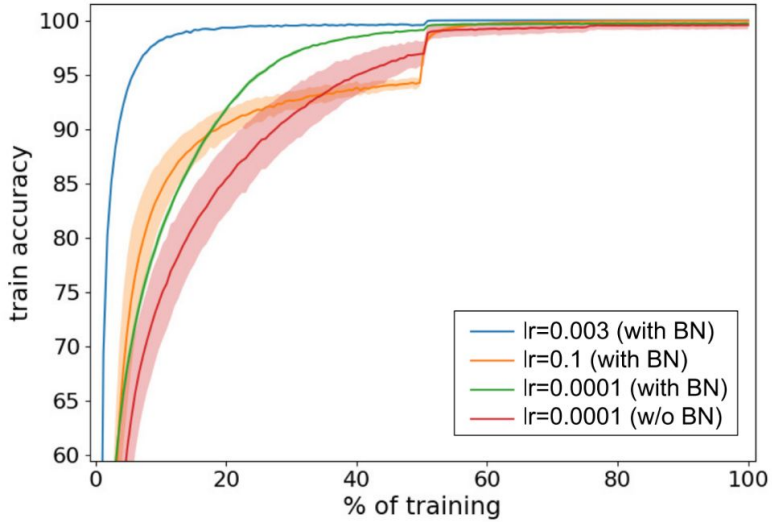
BatchNorm: Inference Behavior

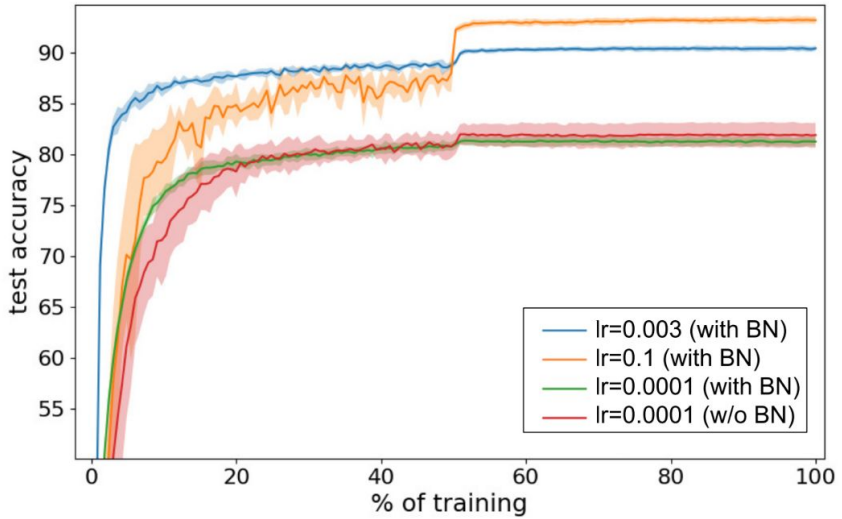
- Model inference should be deterministic
  - Normalization depends on the elements in the batch
- Solution: Use running average statistics calculated during training as:

$$\mu_{\text{inf}} = \lambda \mu_{\text{inf}} + (1 - \lambda)\mu_{\mathcal{B}}$$
$$\sigma^2_{\text{inf}} = \lambda \sigma^2_{\text{inf}} + (1 - \lambda)\sigma^2_{\mathcal{B}}$$

Benefits of batch normalization

- Improves conditioning of the network and enables using a larger learning rate
  - Benefit of batch norm disappears at small learning rates!
  - Large learning rate improves generalization

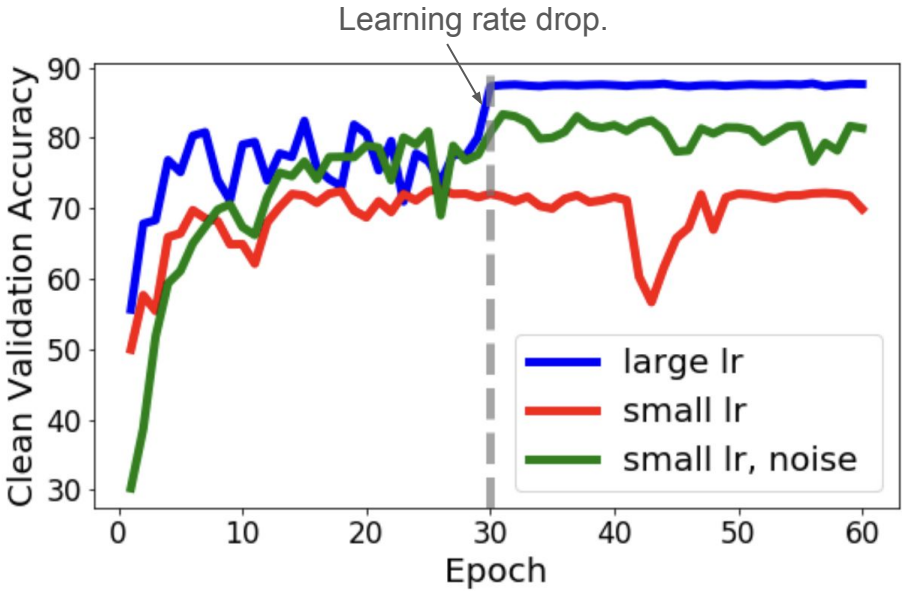




"Understanding Batch Normalization" by Bjorck et al. 2018

Why does a large learning rate help?

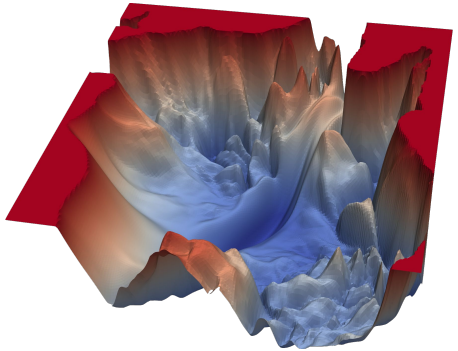
- Noise of the gradient estimate scales with the learning rate (Bjorck et al. 2018)
- Add Gaussian noise to the activations of neural net during training
  - Improves performance when using low learni rates (Li et al., 2019)



"Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks" by Li et al., 2019

Convex vs. Non-Convex Optimization

- Convex optimization: Only one global minima
  - Gradient descent is guaranteed to find it
  - Optimization is all about getting there quickly
- Non-Convex optimization: Many different minima (and saddle points)
  - No theoretical guarantees!
  - Different optimization algorithms will find different minima
  - **Flatter minima** lead to better generalization



"Visualizing the Loss Landscape of Neural Nets" by Li et al., 2017

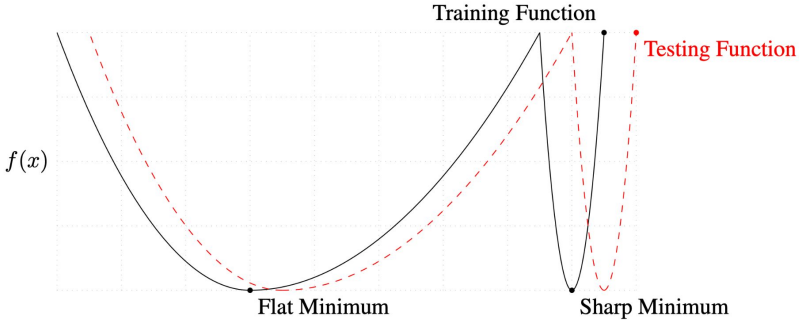


Figure 1: A Conceptual Sketch of Flat and Sharp Minima. The Y-axis indicates value of the loss function and the X-axis the variables (parameters)

Algorithmic Regularization

- Traditional regularization adds explicit penalties (e.g., L1/L2 norm) to the loss
- Algorithmic regularization results from the optimization process itself
  - Very different from convex optimization!

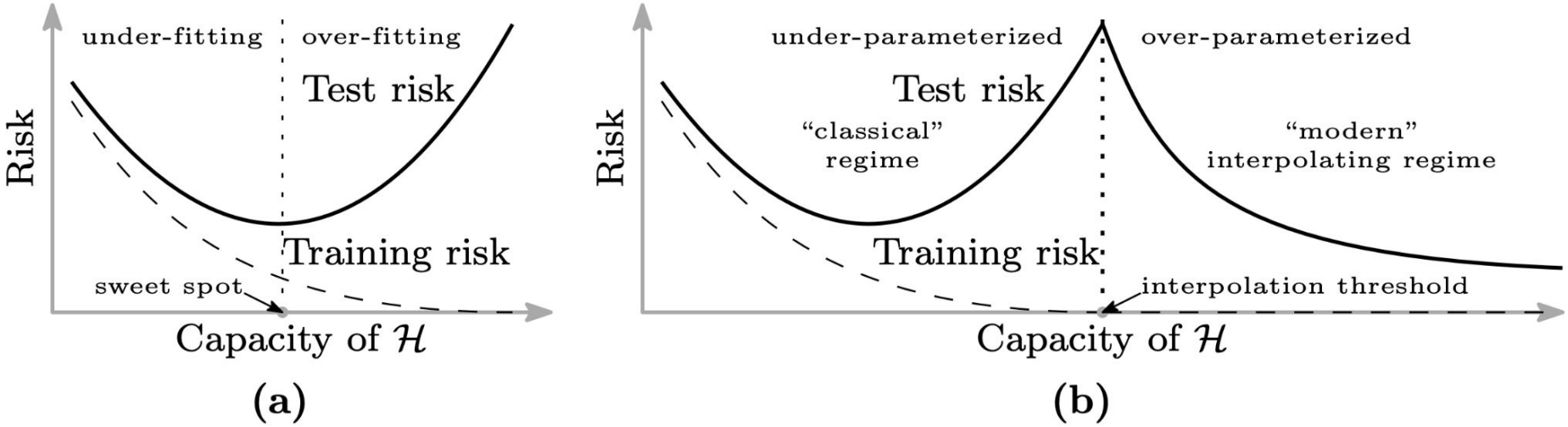
Algorithmic Regularization:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + \lambda \cdot r_{\mathcal{A}}(\mathbf{w})$$

where $r_{\mathcal{A}}(\mathbf{w})$ is some measure of model complexity implicitly controlled by the learning algorithm, $\mathcal{A}$

Deep Double Descent

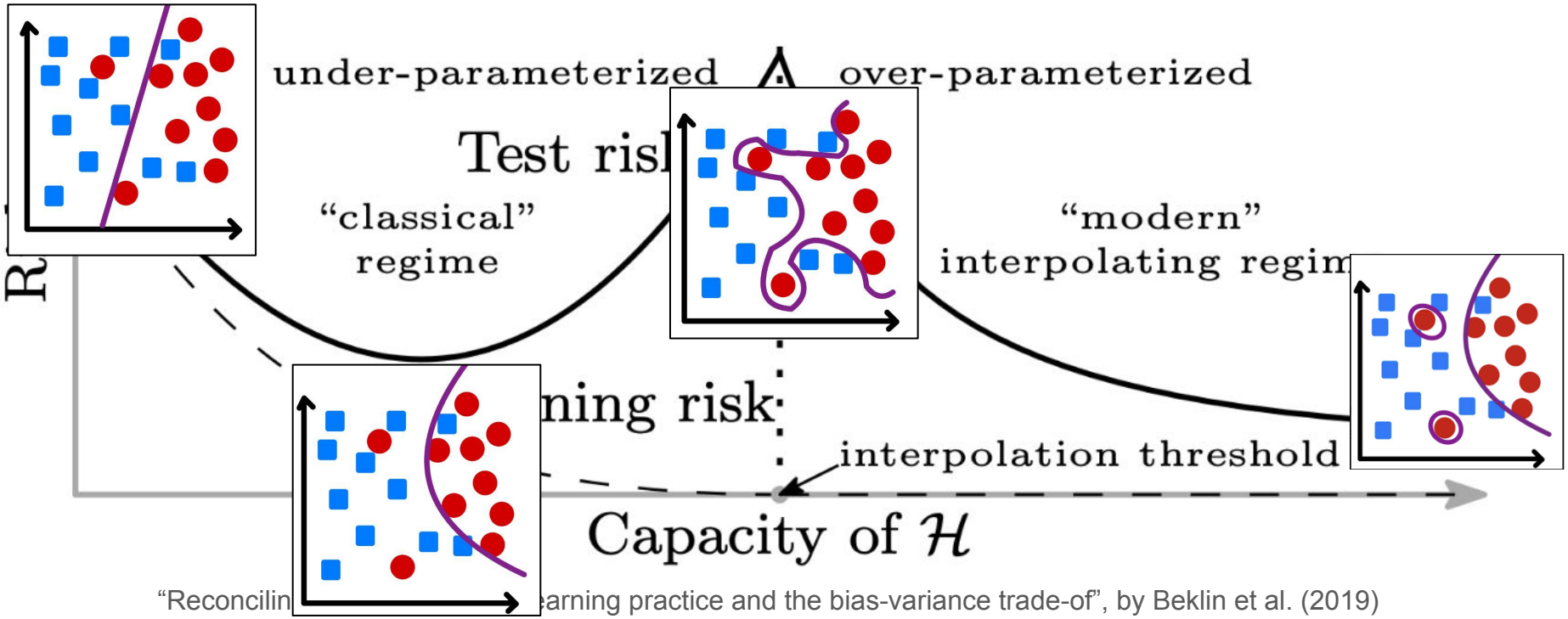
- Neural networks can exhibit a double descent curve in practice



"Reconciling modern machine learning practice and the bias-variance trade-of", by Beklin et al. (2019)

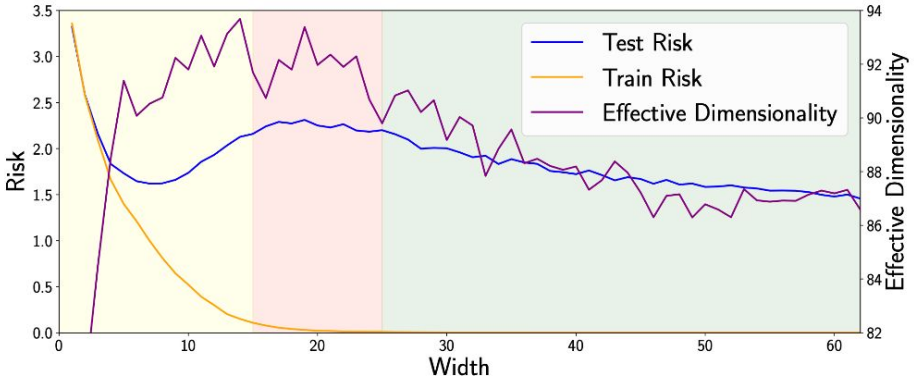
Deep Double Descent

- Neural networks can exhibit a double descent curve in practice

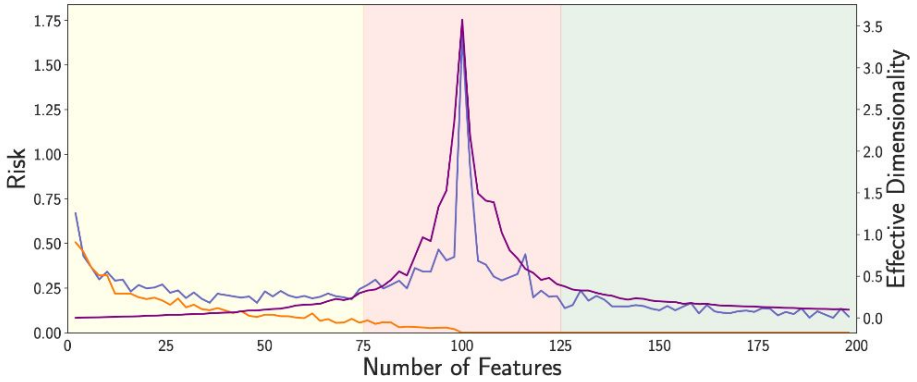


"Reconcilin earning practice and the bias-variance trade-of", by Beklin et al. (2019)

Double descent is not specific to deep learning



(f) ResNet-18

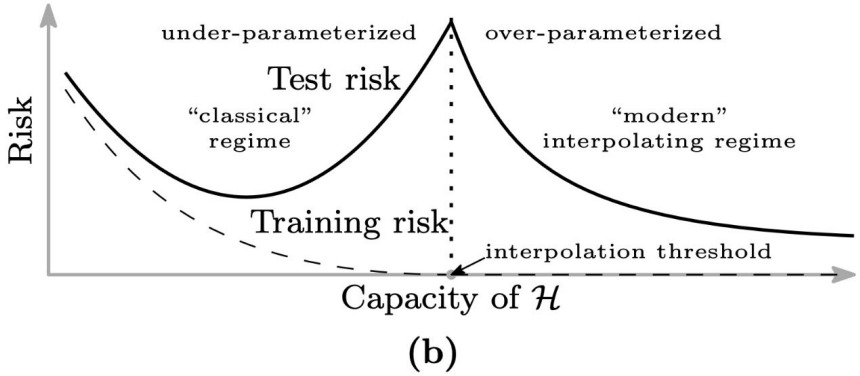


(g) Linear Random Features

[Andrew G. Wilson 2025]

Regularization in the Interpolation Regime

- Many solutions that perfectly fit the data
- Increasing the capacity of the hypothesis class means we can find a “simpler” solution



(b)

Regularization in the interpolation regime ($\mathcal{L}(h) \approx 0$):

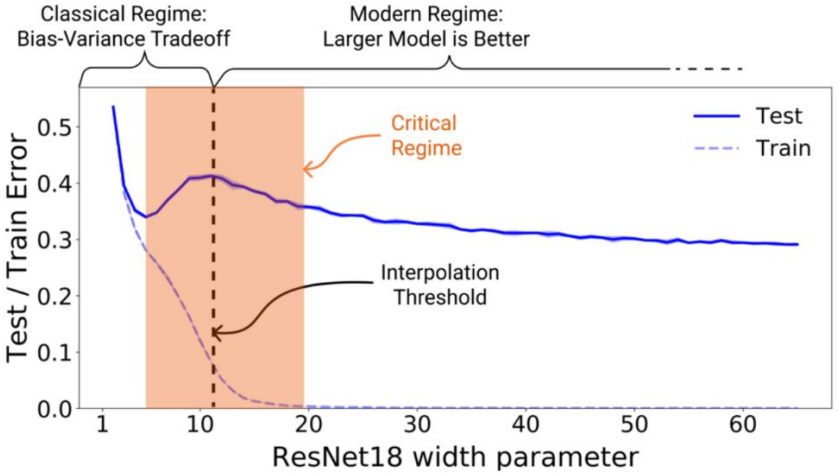
$$h = \underbrace{\arg\min_{h \in \mathcal{H}} \mathcal{L}(h) + \lambda \cdot r(h)}_{\text{Empirical risk minimization}} \approx \underbrace{\arg\min_{h \in \{h : \mathcal{L}(h) \approx 0\}} r(h)}_{\text{Interpolation Regime}}$$

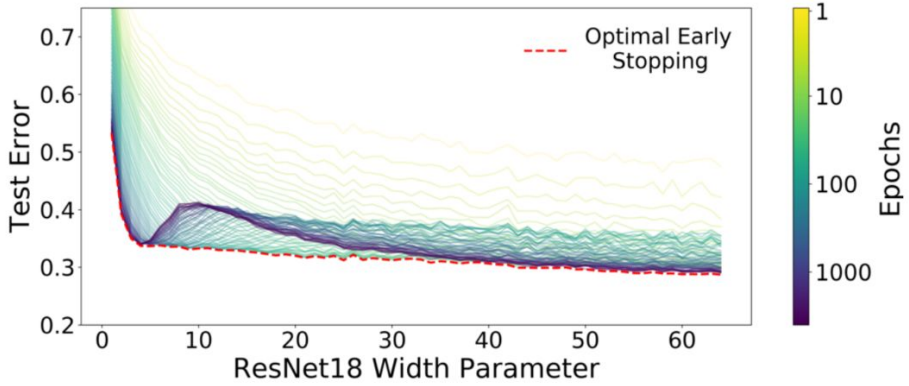
where $r(h)$ is some measure of complexity

“Reconciling modern machine learning practice and the bias-variance trade-of”, by Beklin et al. (2019)

Deep Double Descent

- In-depth empirical study observed double descent with modern architectures (ResNet, Transformers) and tasks (image classification, machine translation)





"Deep Double Descent: Where Bigger Models and More Data Hurt", by Nakkiran et al., 2019

First Homework!

- We are releasing the first homework assignment
  - Covers optimization (this week) and CNNs (next week)
- Two components:
  - Written problems - **Released today**!
  - Coding project - Released next week.
    - Use Google Colab
- **Due**: Both due at the same time two weeks from now.
- Work on it in groups of two
- Start early!
  - Can do most of the written assignment
- Ask questions on Ed
- Office hours posted on the website
- Will be submitted on Gradescope! [Regrade request optional]