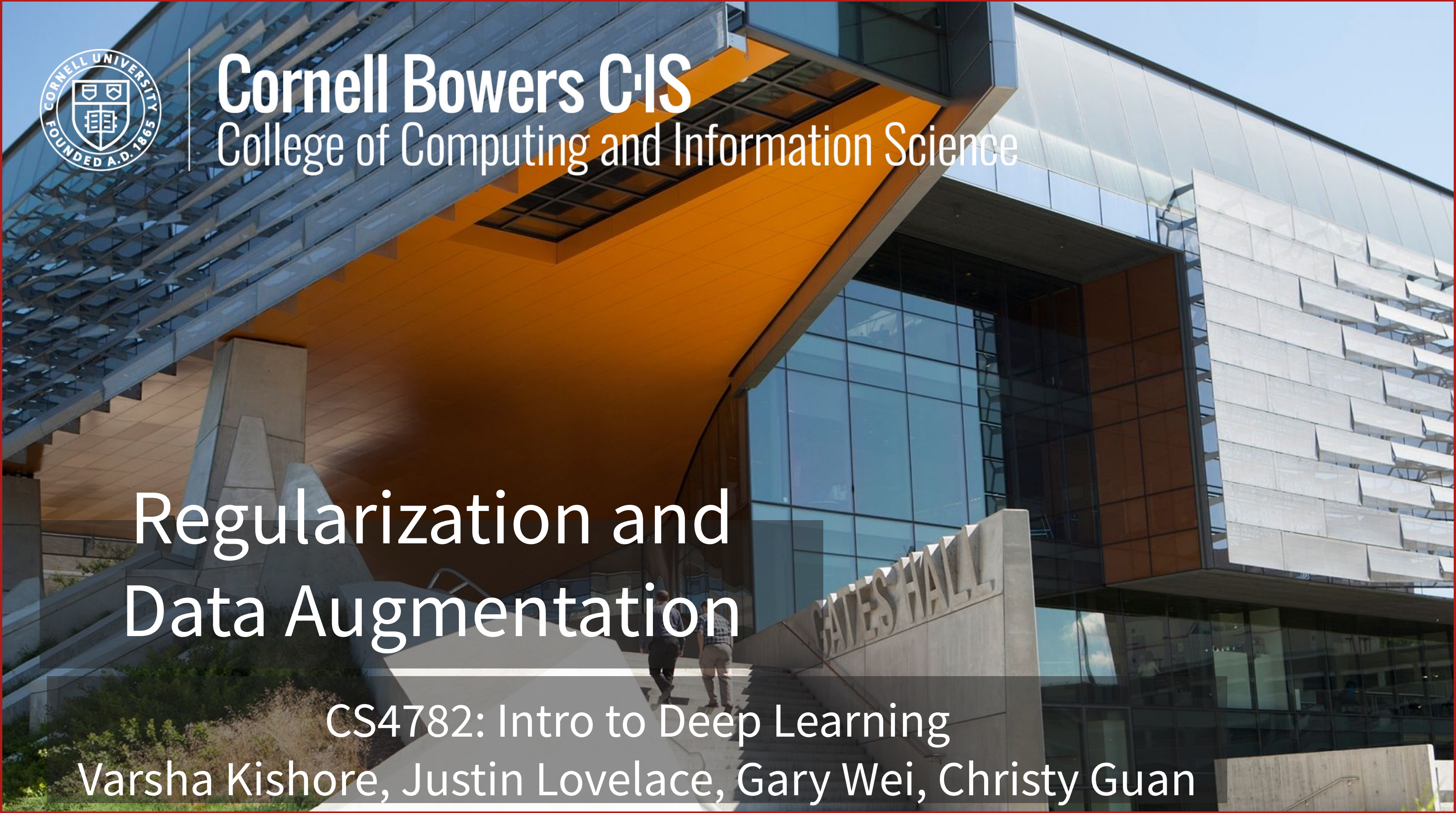


Cornell Bowers C·IS
College of Computing and Information Science

Regularization and Data Augmentation

CS4782: Intro to Deep Learning

Varsha Kishore, Justin Lovelace, Gary Wei, Christy Guan

Cornell Bowers C·IS

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**

Cornell Bowers C·IS

Agenda

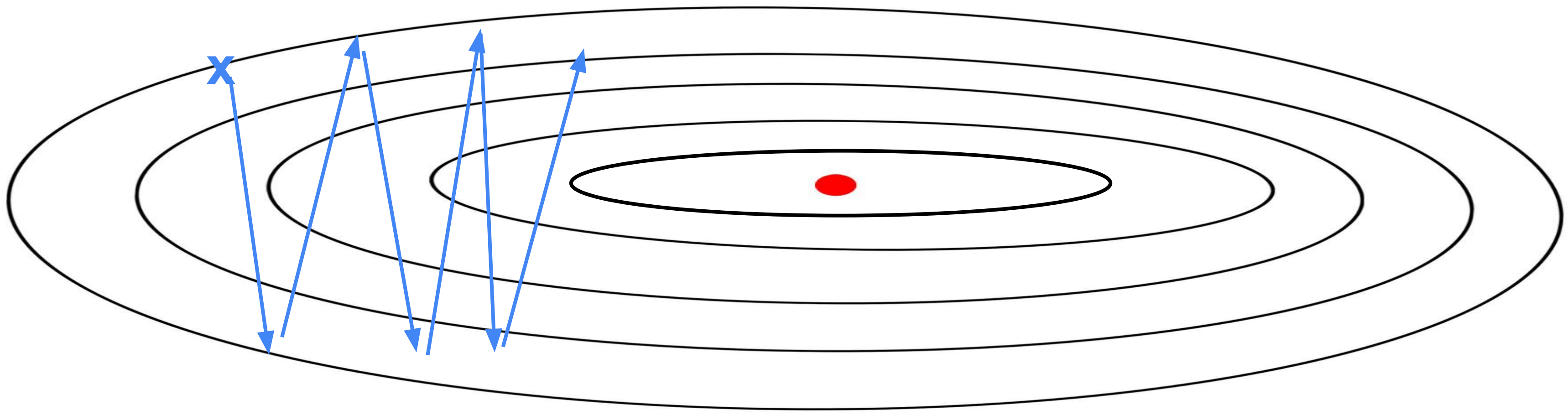
- Backpropagation
- Optimizers
  - Gradient Descent
  - Stochastic Gradient Descent
  - SGD w. Momentum
  - AdaGrad
  - RMSProp
  - Adam
- **Learning rate scheduling**

Cornell Bowers C·IS

Recall: Draw the gradients

- Smaller learning rate
- Larger learning rate

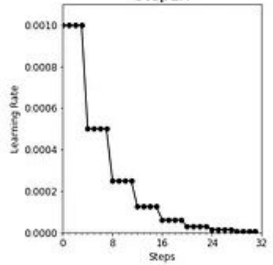
Local Minimum



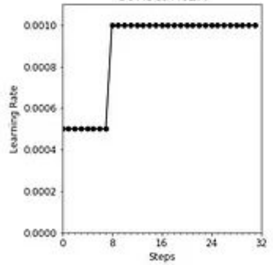
Cornell Bowers C·IS

Learning Rate Scheduling

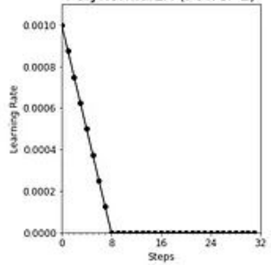
StepLR



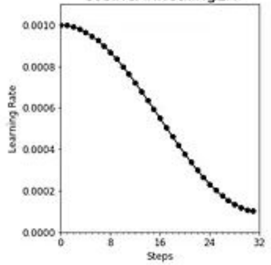
ConstantLR



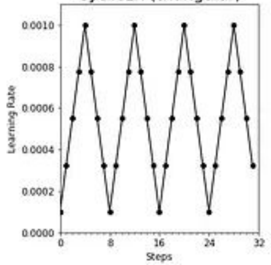
PolynomialLR (Power 1)



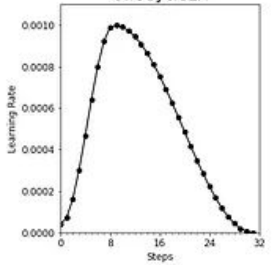
CosineAnnealingLR



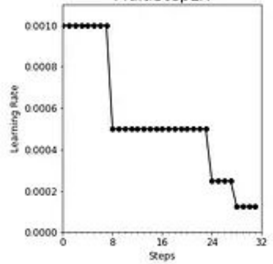
CyclicLR (triangular)



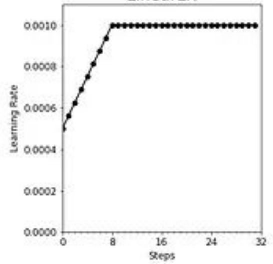
OneCycleLR



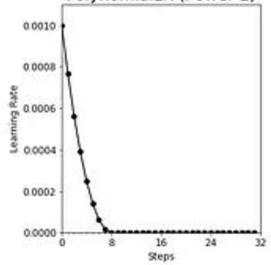
MultiStepLR



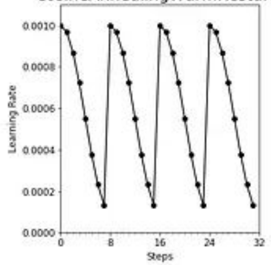
LinearLR



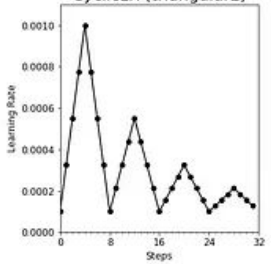
PolynomialLR (Power 2)



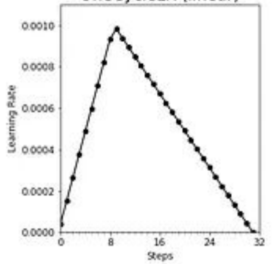
CosineAnnealingWarmRestarts



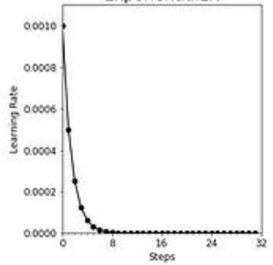
CyclicLR (triangular2)



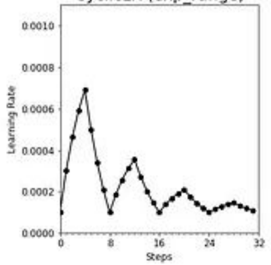
OneCycleLR (linear)



ExponentialLR



CyclicLR (exp_range)



Cornell Bowers C·IS

OPT: Open Pre-trained Transformer Language Models

OPT is an open source LLM like GPT-4 from Meta.

For large models like OPT-175B, more engineering efforts are needed.

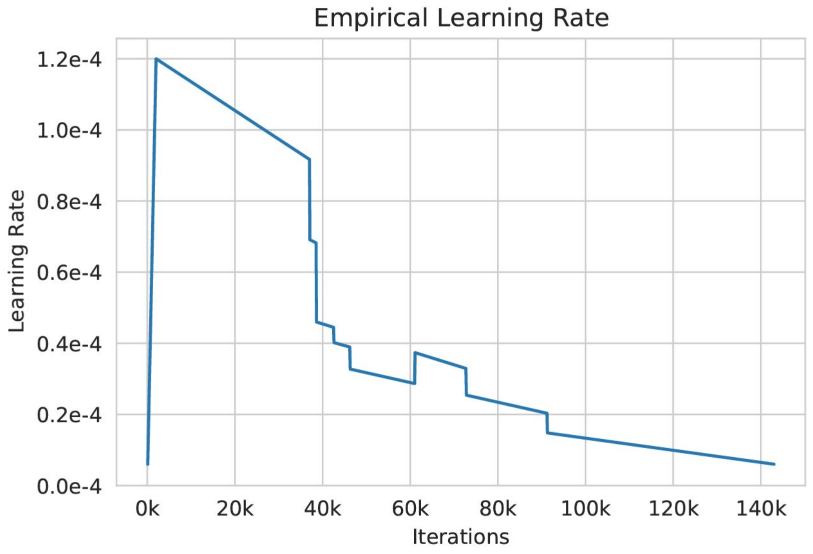


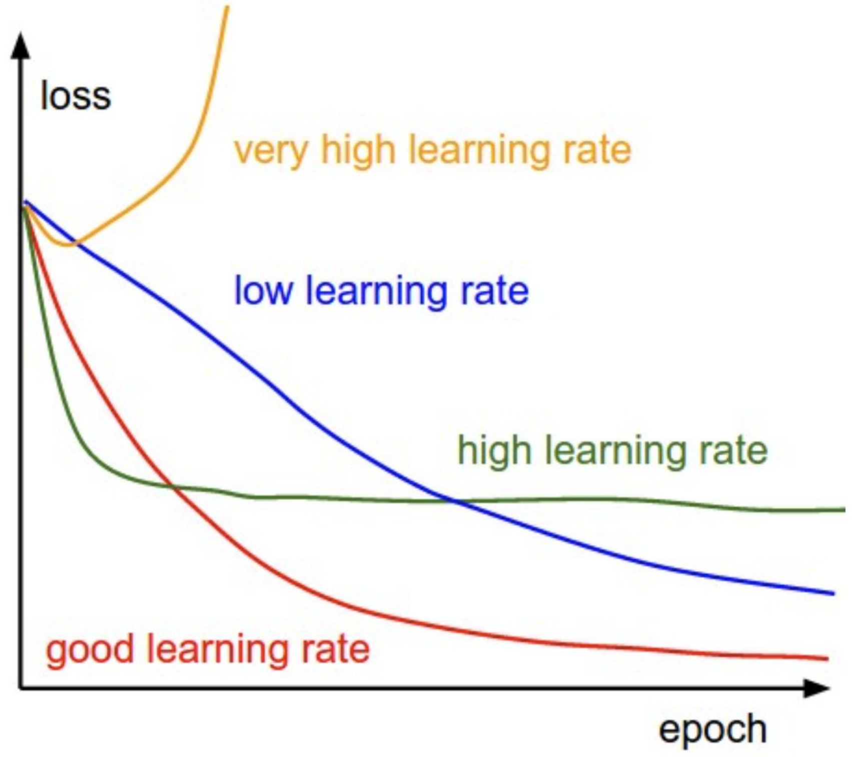
Figure 1: **Empirical LR schedule.** We found that lowering learning rate was helpful for avoiding instabilities.

Cornell Bowers C·IS

Hyperparameters

- Learning rate
- Batch size
- Beta1 & beta2 of adam
- Regularization strength

These are all hyperparameters that affect performance!



Source: https://cs231n.github.io/neural-networks-3/

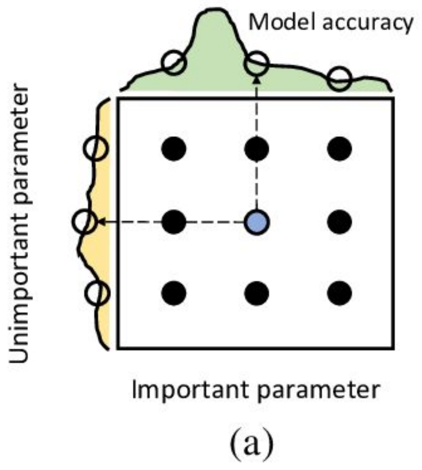
Cornell Bowers C·IS

Hyperparameter Optimization (HPO)

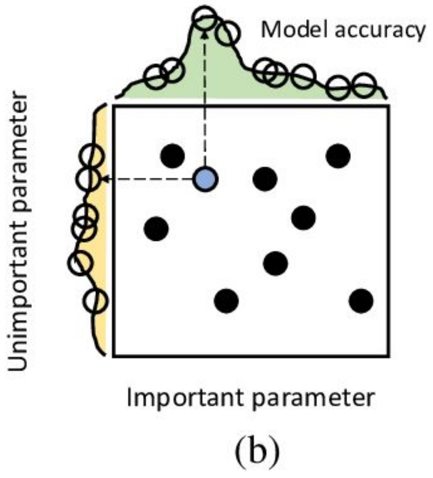
- Learning rate
- Batch size
- Beta1 & beta2 of adam
- Regularization strength

These are all hyperparameters that affects performance!

Random search HPO is the efficient and simple way to start!



Grid Search



Random Search

Agenda

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

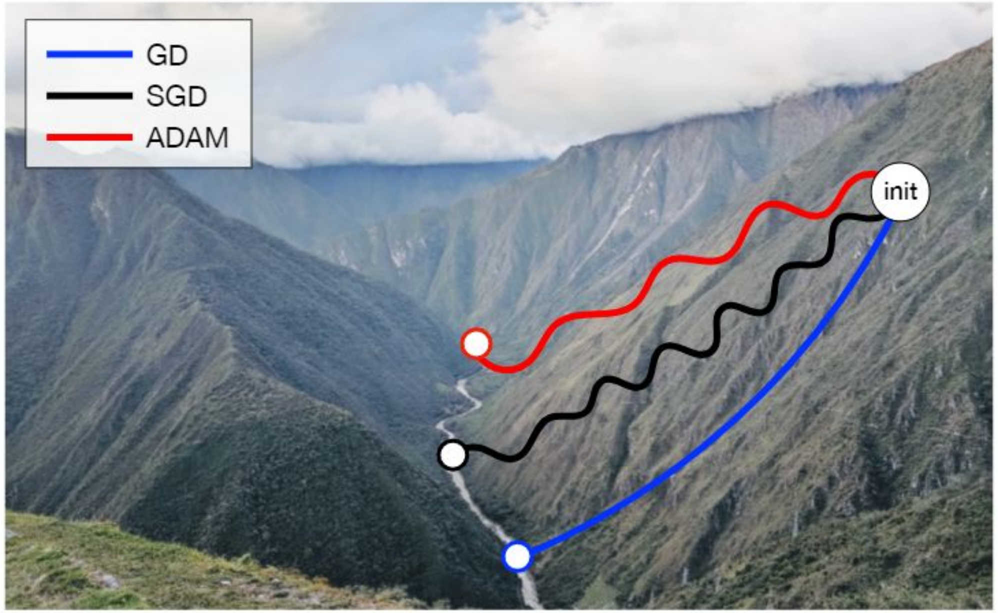
Cornell Bowers C·IS

Are all Optimizers equivalent somehow?

No!

There are *many* minimizers of the training loss

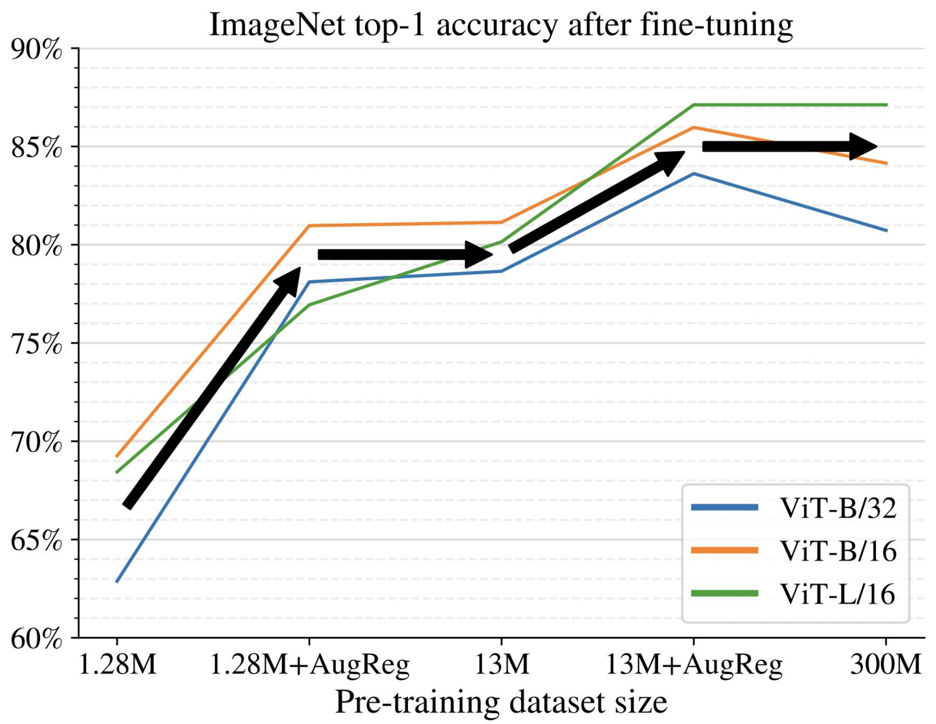
The **optimizer** determines which minimizer you converge to



Cornell Bowers C·IS

Why do we care?

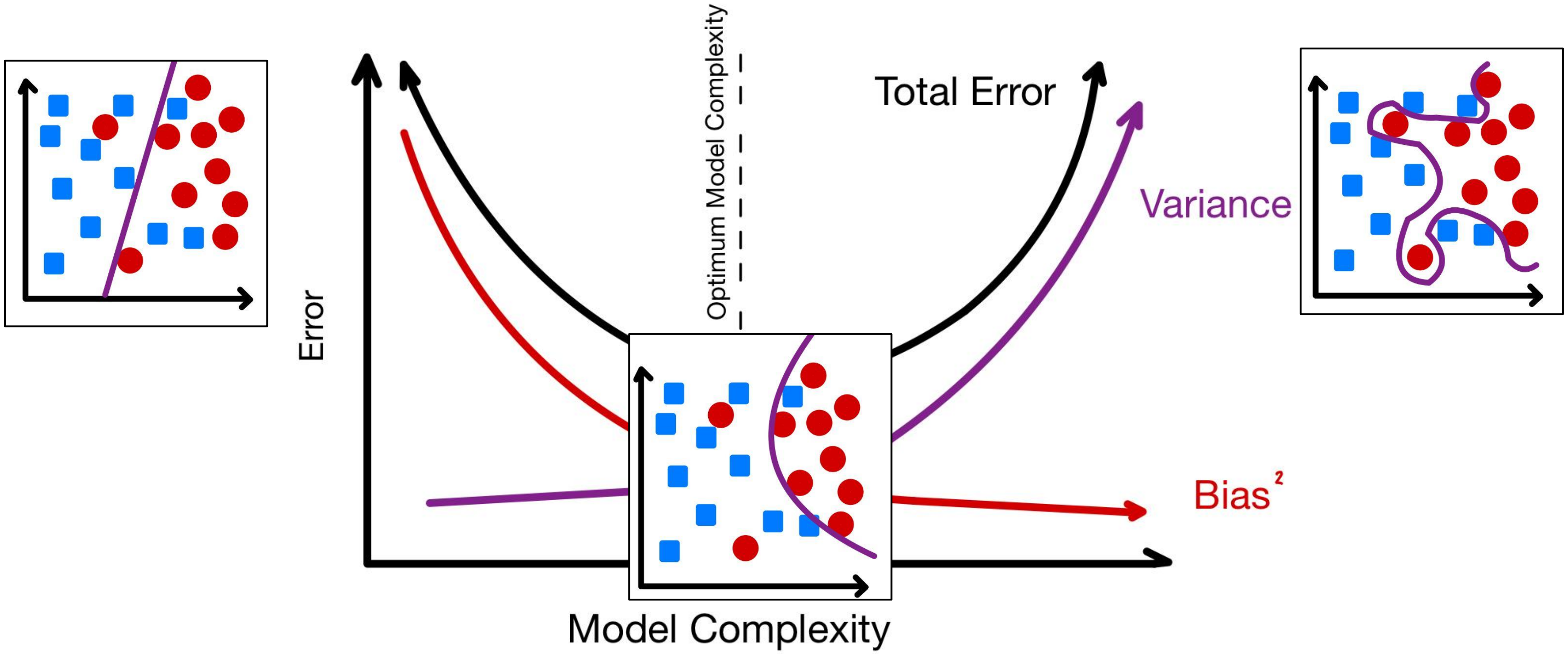
- Regularization and data augmentation are really effective!
- Can be worth millions of additional training images

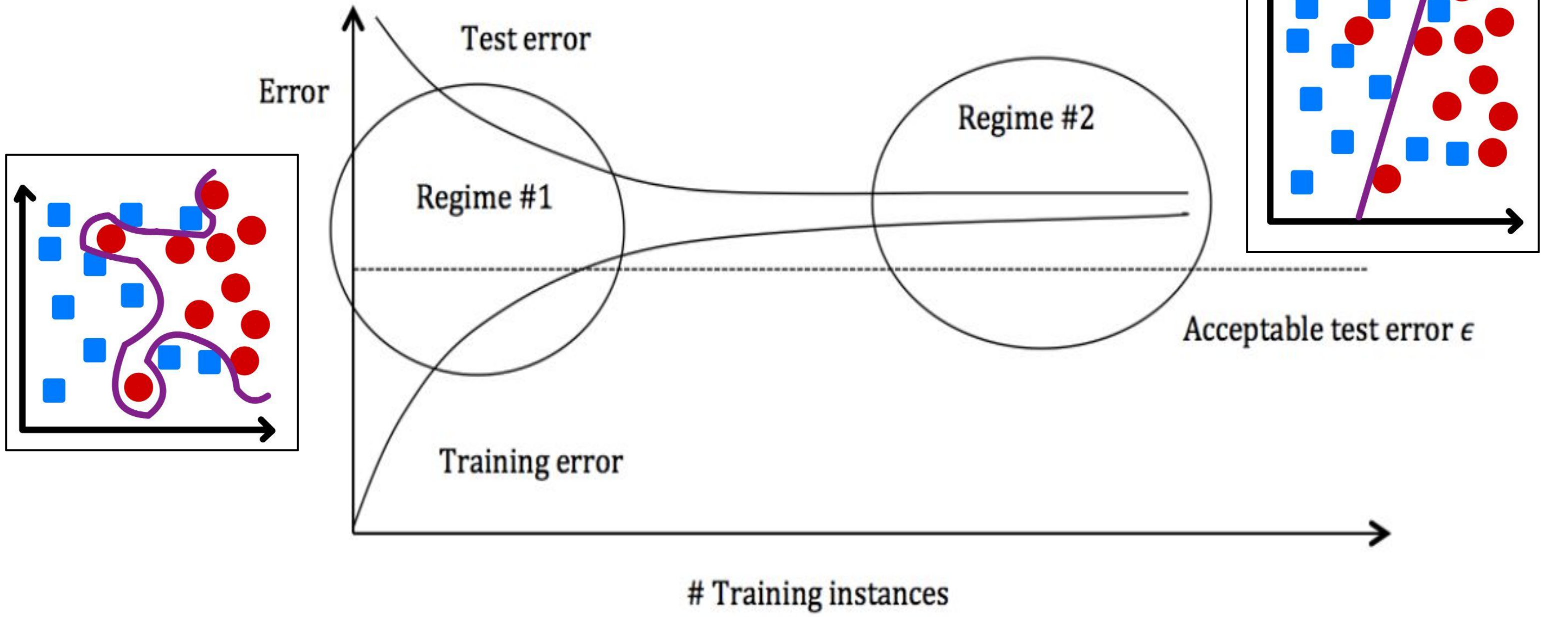


“How to train your ViT? Data, Augmentation,and Regularization in Vision Transformers”, by Steiner et al. 2022

Cornell Bowers C·IS

Expected Test Error = $\boxed{\text{Variance}}$ + Noise + $\boxed{\text{Bias}^2}$

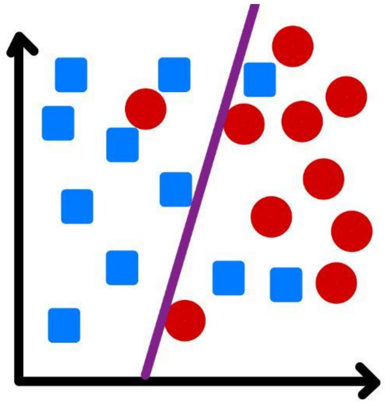


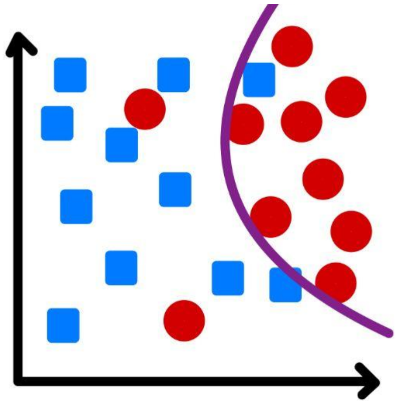

Error
Test error
Regime #1
Regime #2
Acceptable test error ϵ
Training error
# Training instances

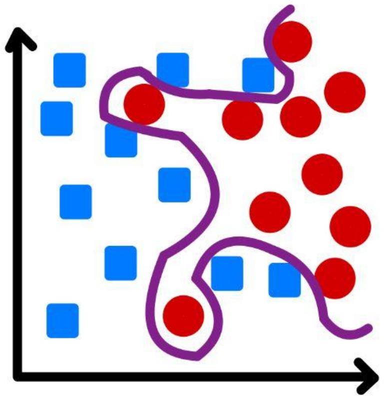
Cornell Bowers C·IS

Complex models have high variance

An overfit model performs well on training data, but does not perform well on test data.


Underfitting


Appropriate Fitting


Overfitting

Common Remedies for Overfitting

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

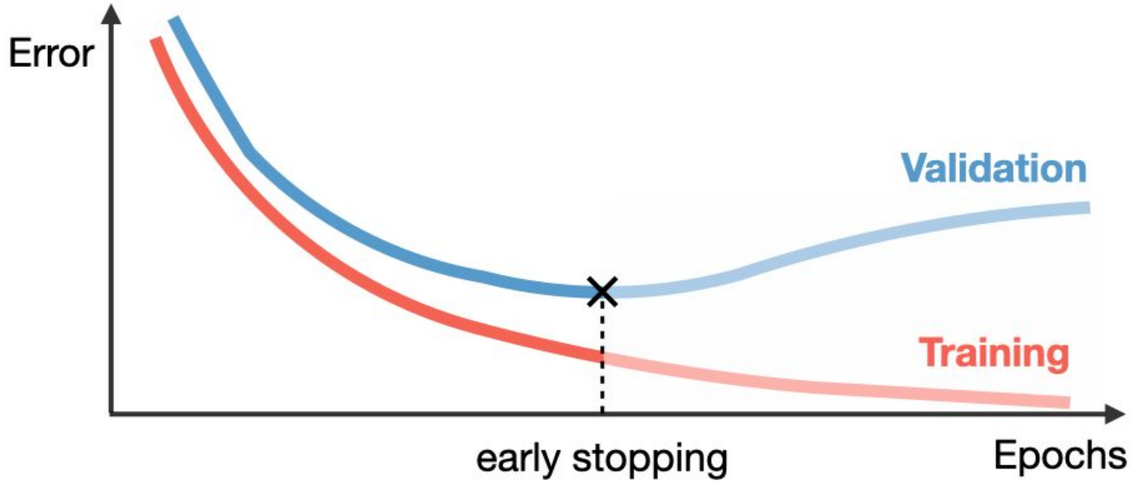
Cornell Bowers C·IS

Demo: Overfitting

[Tensorflow Playground]

Cornell Bowers C·IS

Early Stopping



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

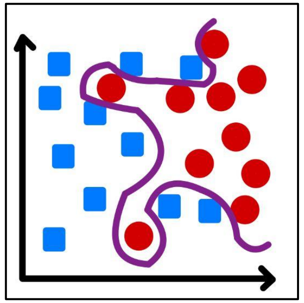
https://wandb.ai/ayush-thakur/huggingface/reports/Examples-of-Early-Stopping-in-HuggingFace-Transformers--Vmlldzo0MzE2MTM

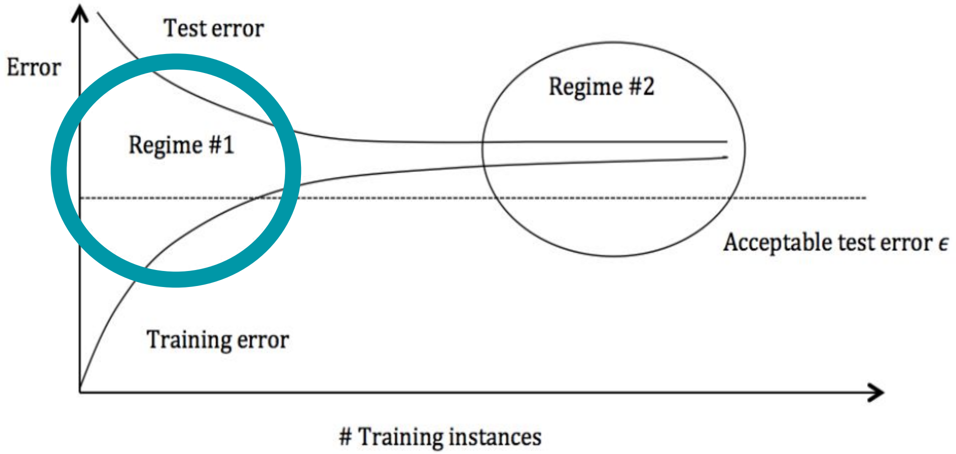
Cornell Bowers C·IS

What is Regularization?

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

Models that overfit can have large generalization gaps.





Comparing Error and Number of Training Instances

Cornell Bowers C·IS

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:

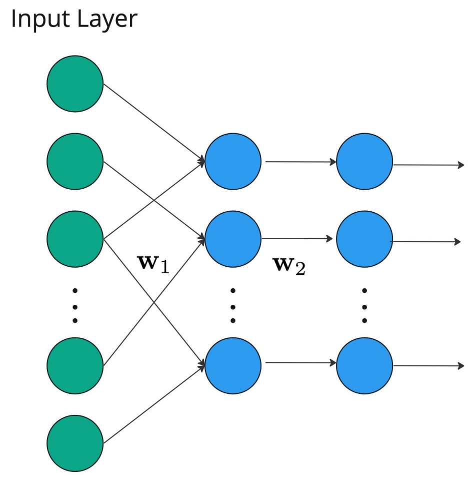
$$\mathbf{w} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + \lambda \cdot r(\mathbf{w})$$

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

Cornell Bowers C·IS

Regularizers

Regularizers are used to quantify the complexity of a model.



Deep net

Cornell Bowers C·IS

Optimization problem:

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

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

Gradient update:

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

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

Cornell Bowers C·IS

Optimization problem:

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

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

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

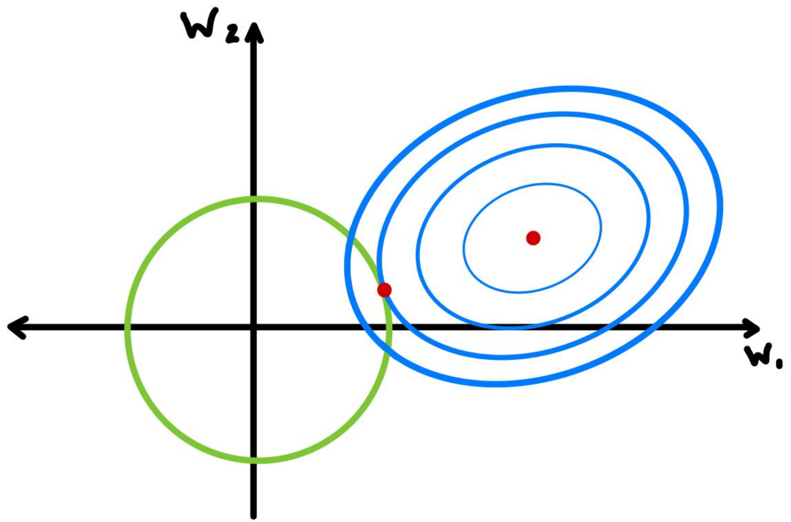
Gradient update:

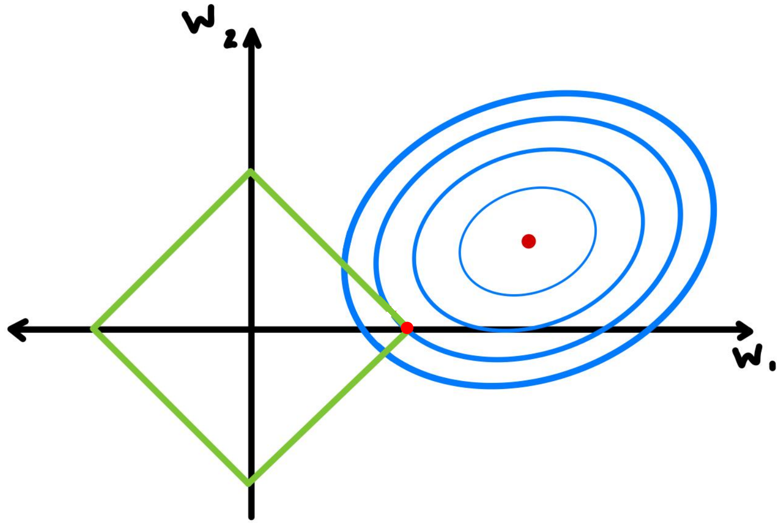
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\nabla\mathcal{L}(\mathbf{w}_t, D)$$

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

Cornell Bowers C·IS

Geometric Interpretation





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

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

Demo: L1/L2 Regularization

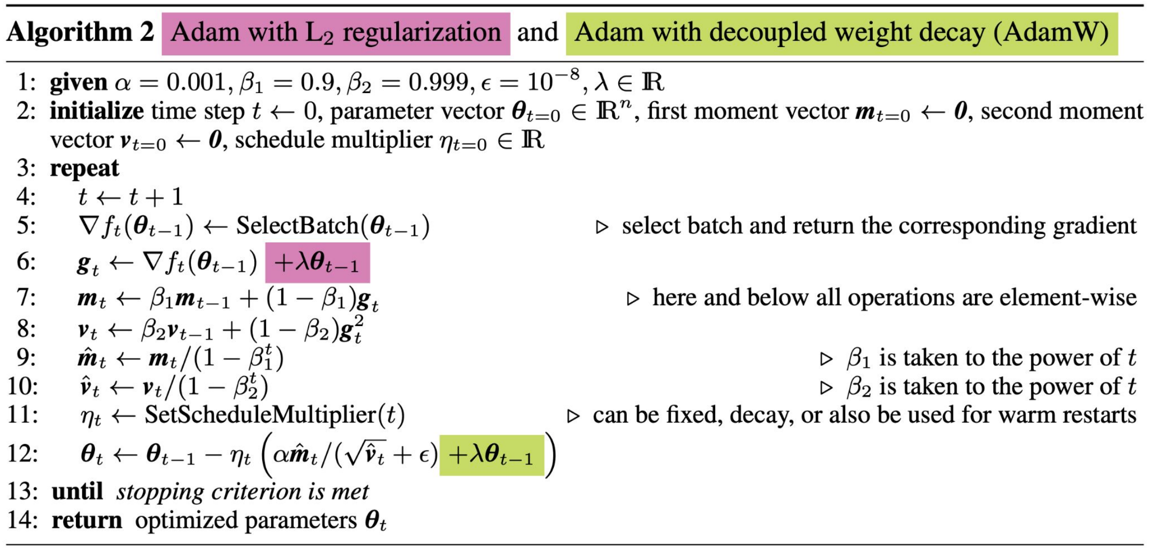
Tensorflow Playground

Cornell Bowers C·IS

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)

```
1:  given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
2:  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 $\eta_{t=0} \in \mathbb{R}$
3:  repeat
4:      $t \leftarrow t + 1$
5:      $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$          ▷ select batch and return the corresponding gradient
6:      $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) \ +\lambda\boldsymbol{\theta}_{t-1}$
7:      $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$          ▷ here and below all operations are element-wise
8:      $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$
9:      $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1-\beta_1^t)$          ▷ $\beta_1$ is taken to the power of $t$
10:     $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1-\beta_2^t)$          ▷ $\beta_2$ is taken to the power of $t$
11:     $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$          ▷ can be fixed, decay, or also be used for warm restarts
12:     $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left( \alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) \ +\lambda\boldsymbol{\theta}_{t-1} \right)$
13: until stopping criterion is met
14: return optimized parameters $\boldsymbol{\theta}_t$
```

Cornell Bowers C·IS

Connection Between Weight Decay and L2 Regularization

Are weight decay and L2 regularization equivalent in general?

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}^{\text{reg}}(\mathbf{w}_t) = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t) - \alpha \lambda_0 \mathbf{w}_t$$

$$\mathbf{w}_{t+1} = (1 - \lambda_1)\mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t) = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t) - \lambda_1 \mathbf{w}_t$$

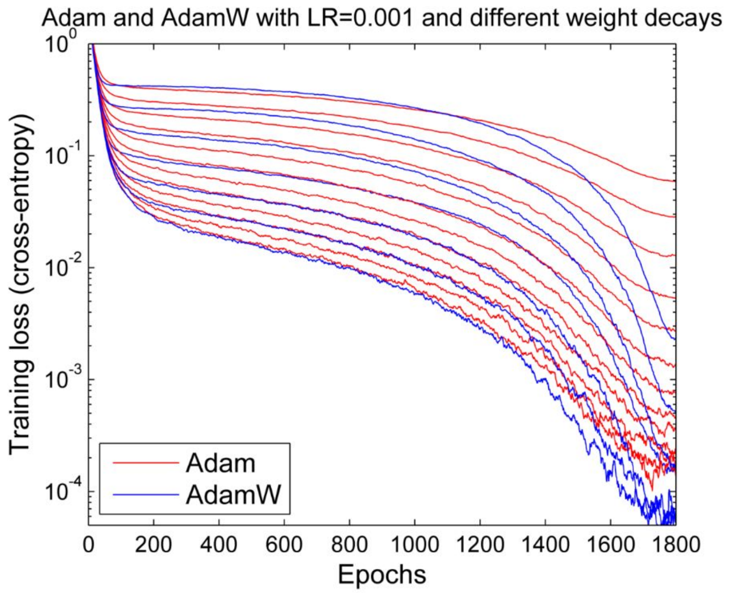
Cornell Bowers C·IS

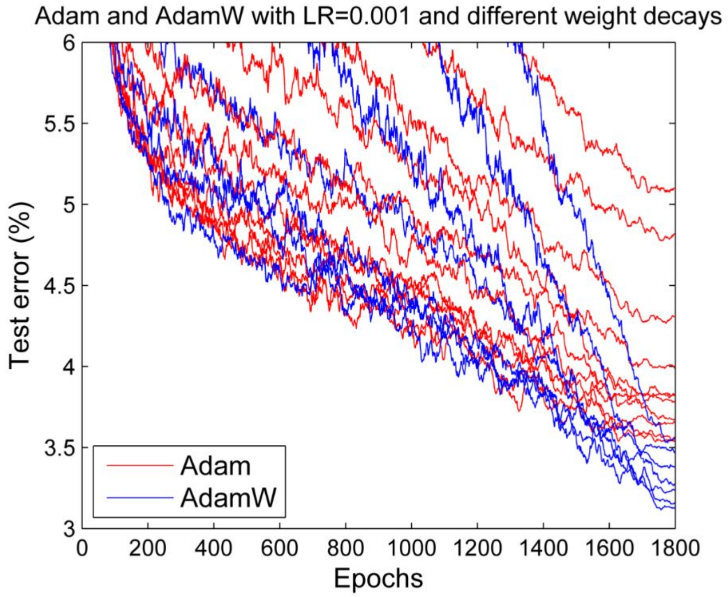
AdamW

Cornell Bowers C·IS

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

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

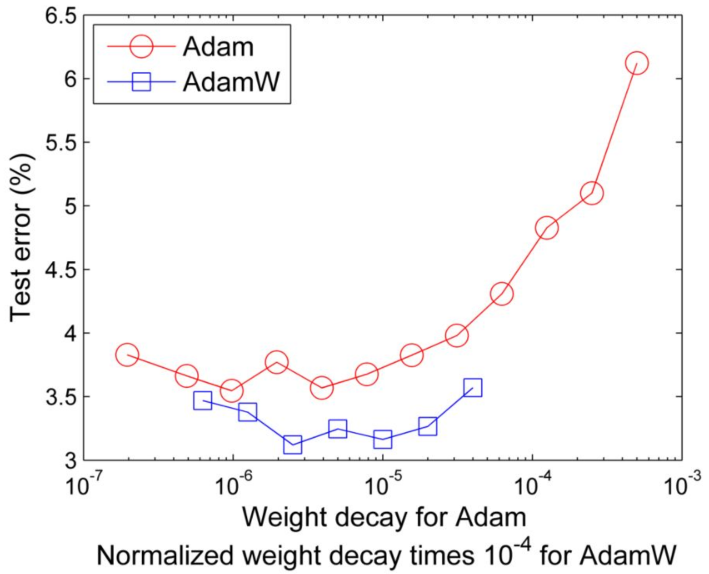


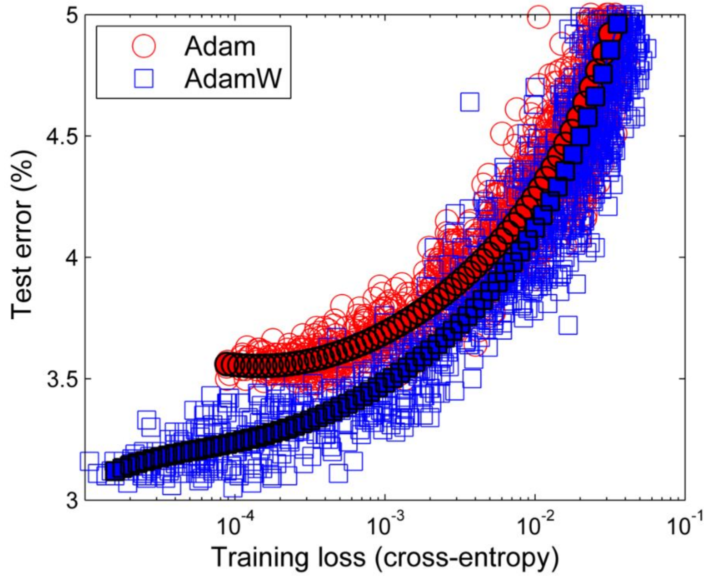


Cornell Bowers C·IS

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

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





Cornell Bowers C·IS

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**

Cornell Bowers C·IS

(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**

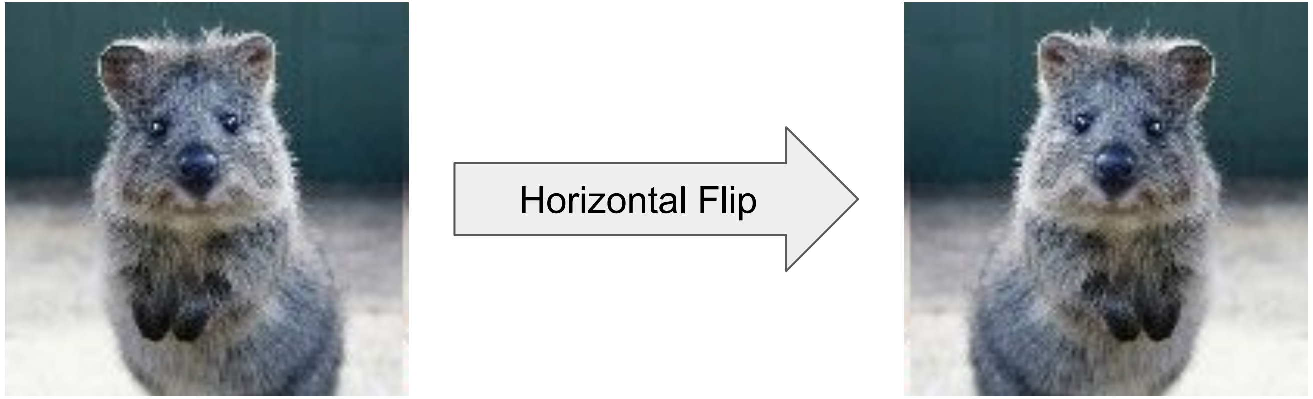
Cornell Bowers C·IS

Data Augmentation

Discuss: Image Classification

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

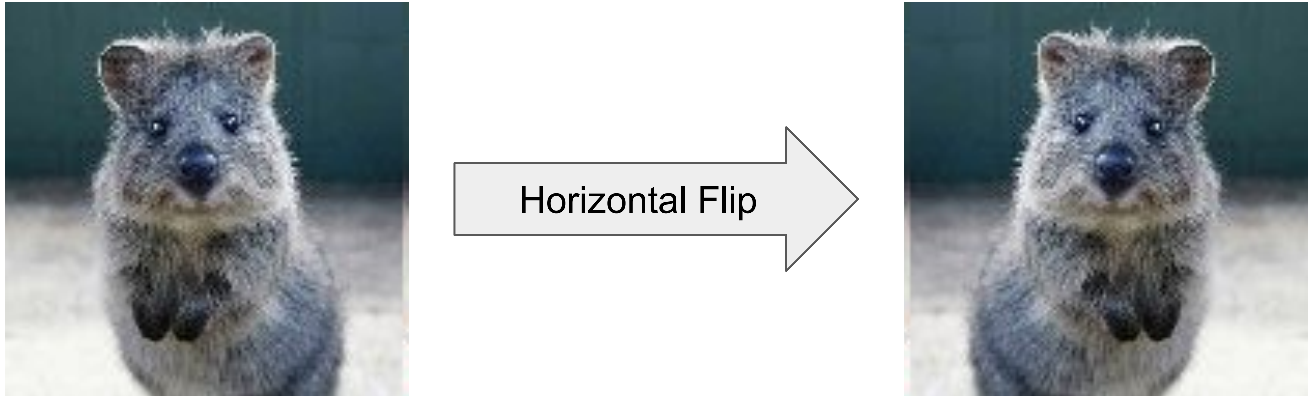
Can we augment the training data without annotating more images?



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

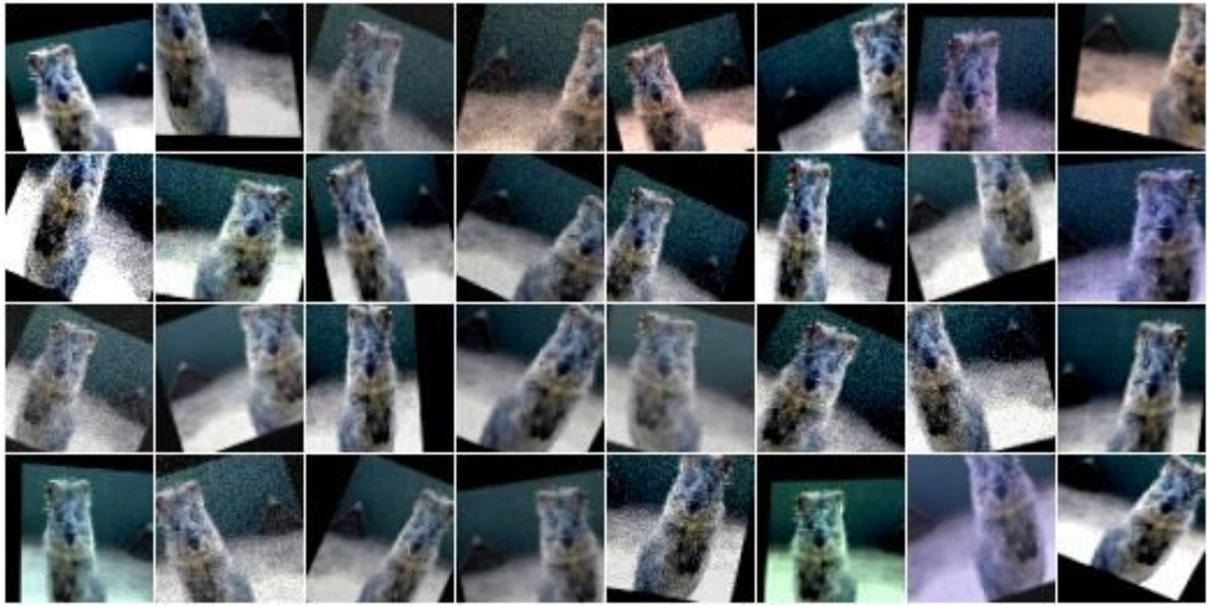


https://imgaug.readthedocs.io/en/latest/index.html

Cornell Bowers C·IS

Data Augmentations

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



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

Cornell Bowers C·IS

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?

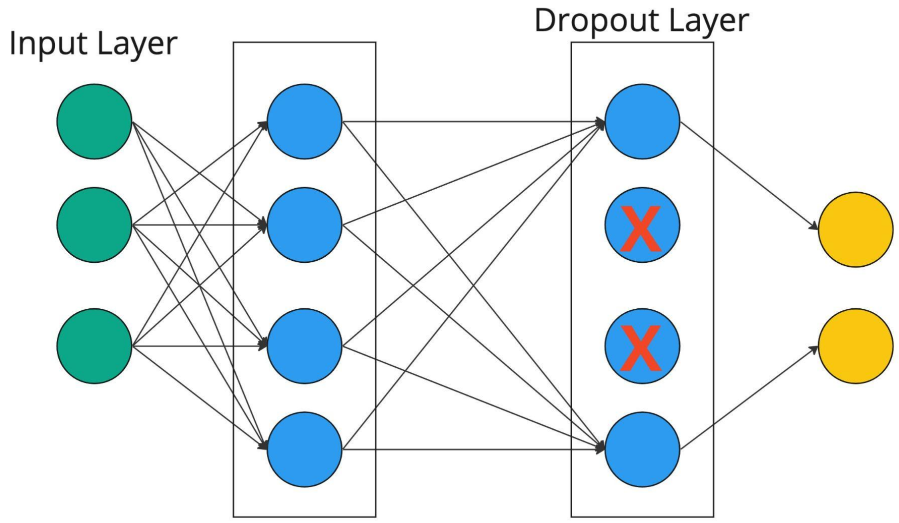
DropOut

Cornell Bowers C·IS

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.

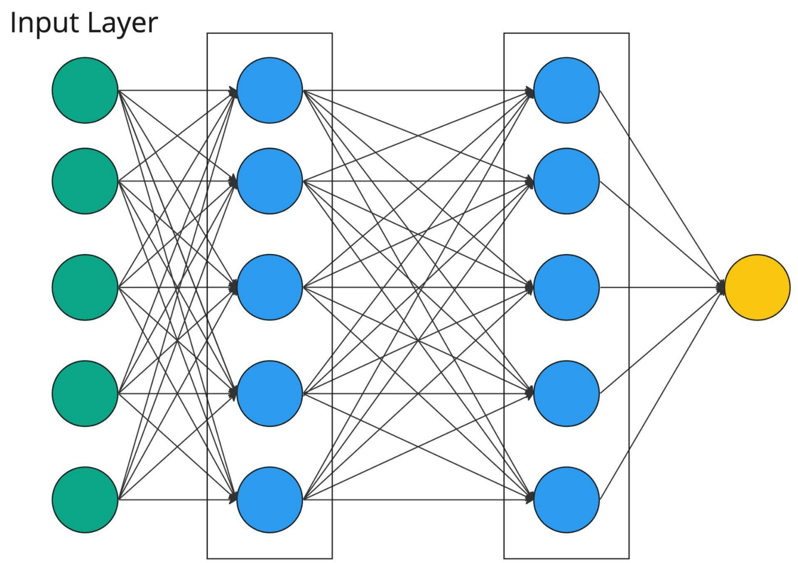


Deep Net with Dropout Layer

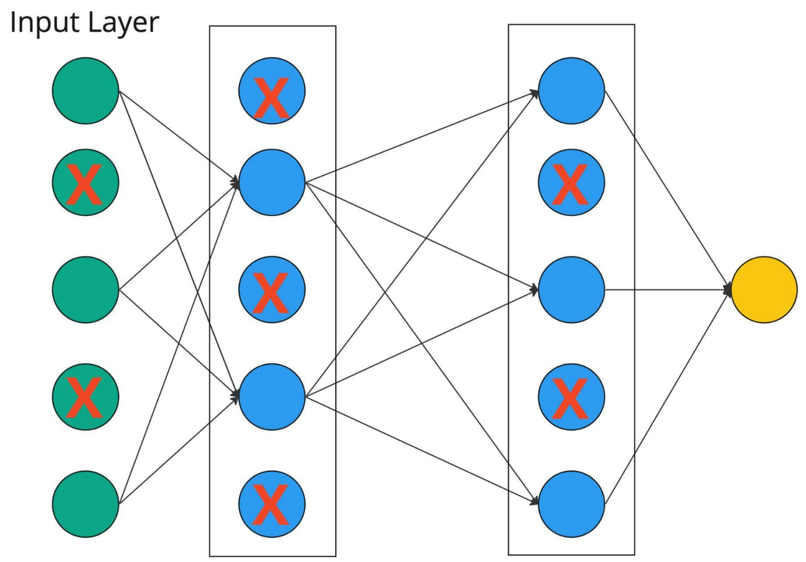
[Srivastava et al. 2014]

Cornell Bowers C·IS

Implementing Dropout



Standard deep net with two hidden layers



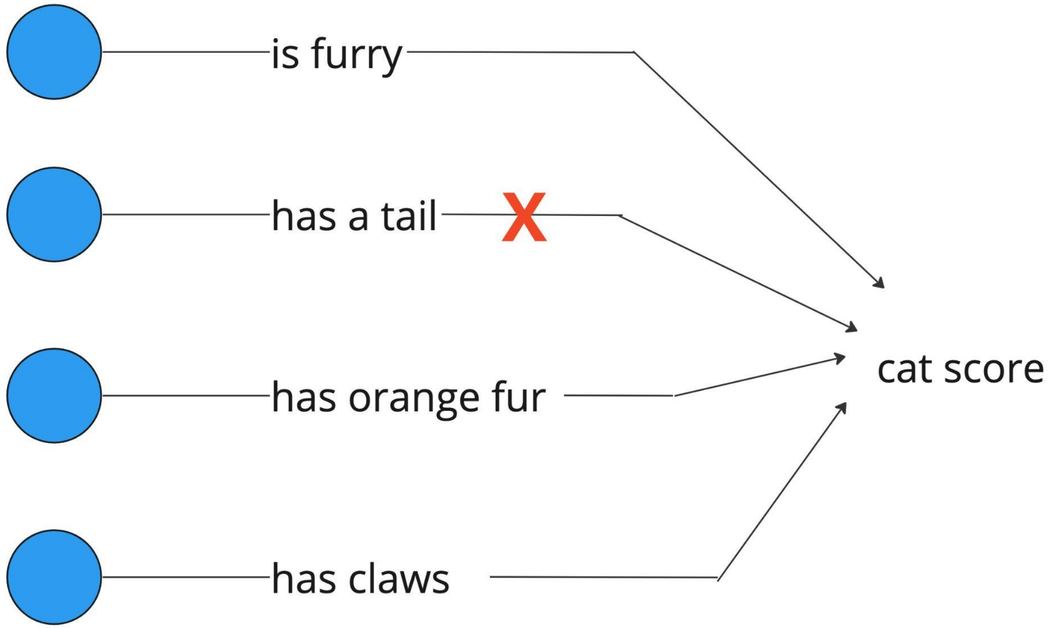
Deep net produced by applying dropout.
Crossed units have been dropped

[Srivastava et al. 2014]

Cornell Bowers C·IS

Why is Dropout a good idea?

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

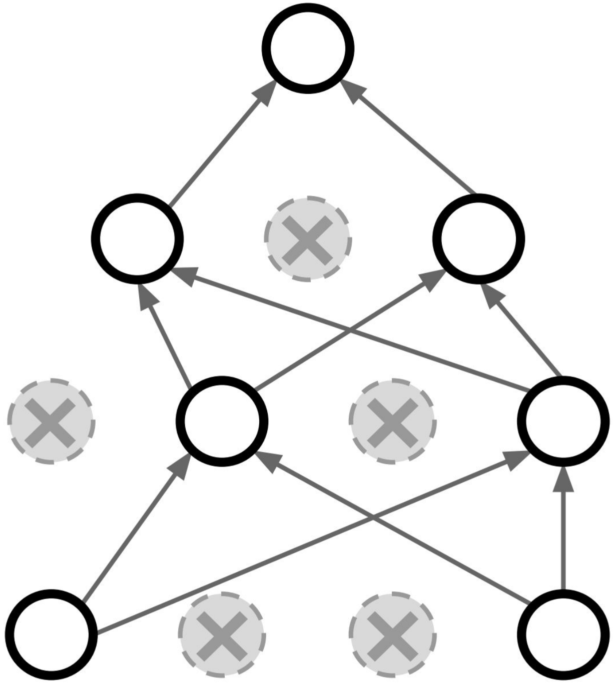


[Srivastava et al. 2014]

Cornell Bowers C·IS

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

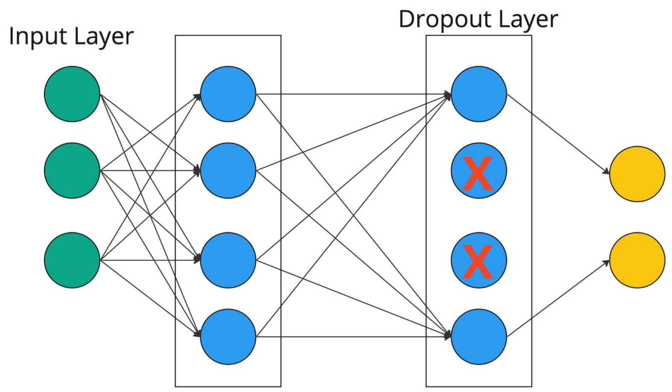


Cornell Bowers C·IS

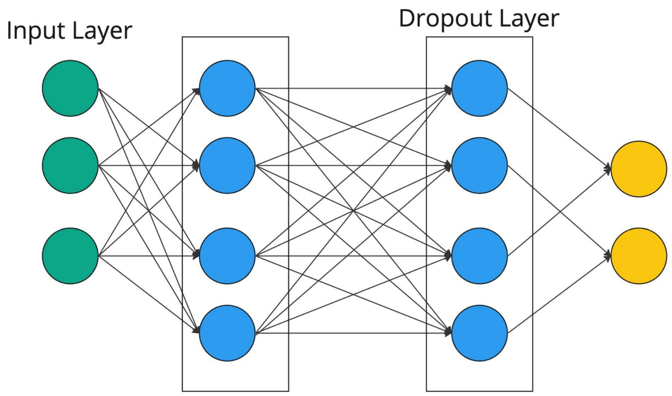
Dropout During Test Time

Use all of the neurons in the network

Does this introduce any problems?



Training Time

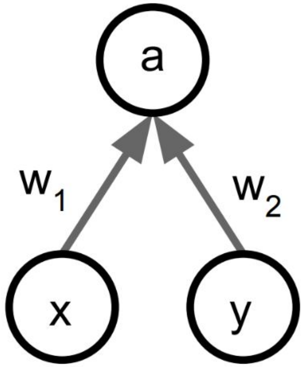


Test Time

Cornell Bowers C·IS

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: $E[a] = w_1x + w_2y$

During training we have:
$$\begin{aligned} E[a] &= \frac{1}{4}(w_1x + w_2y) + \frac{1}{4}(w_1x + 0y) \\ &\quad + \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2y) \\ &= \frac{1}{2}(w_1x + w_2y) \end{aligned}$$

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

Cornell Bowers C·IS

Effectiveness of Dropout

- Improves generalization of neural nets when training with limited data

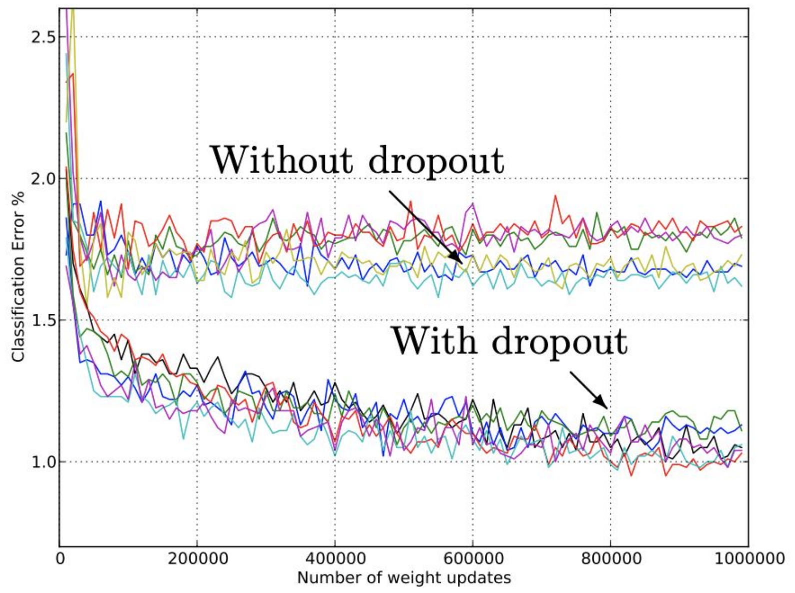


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

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

Cornell Bowers C·IS

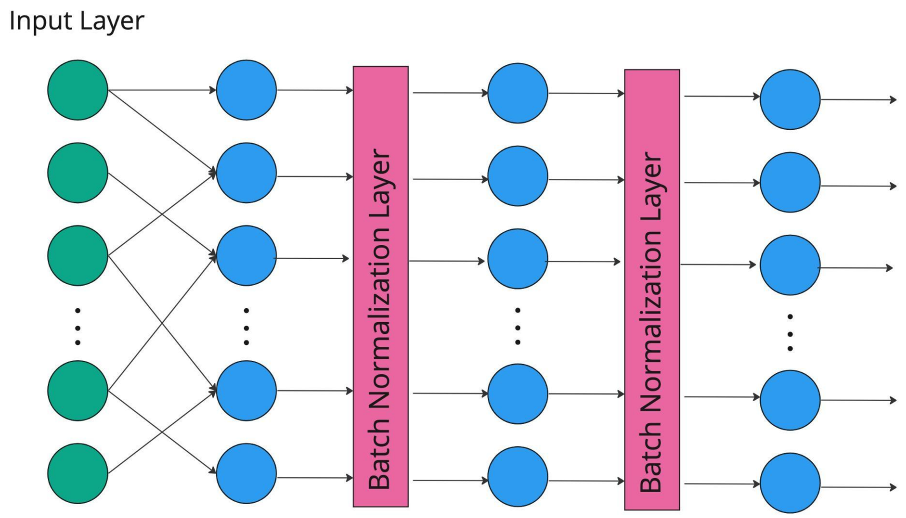
Batch Normalization

Cornell Bowers C·IS

Batch Normalization

Batch Normalization normalizes the intermediate features in neural networks.

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



Cornell Bowers C·IS

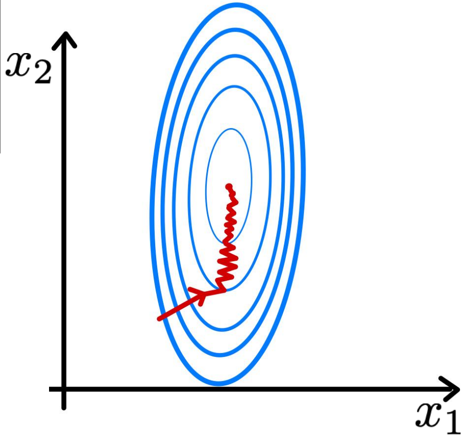
Why should we standardize data?

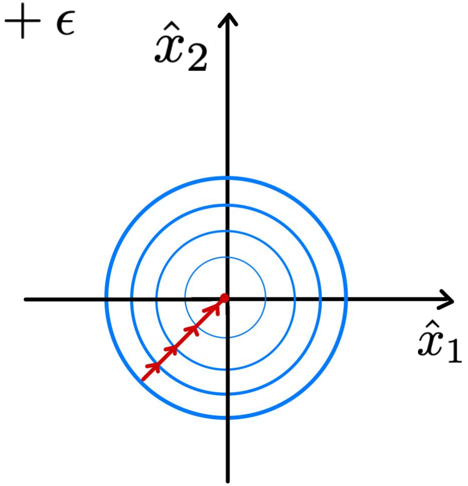
- Standardization ensures all features have a similar scale
  - Beneficial for optimization
- We do not know a priori which features will be relevant and we do not want to penalize or upweight features
- Example: Predict house prices

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

x_1 **Bedrooms: 1 to 5**

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

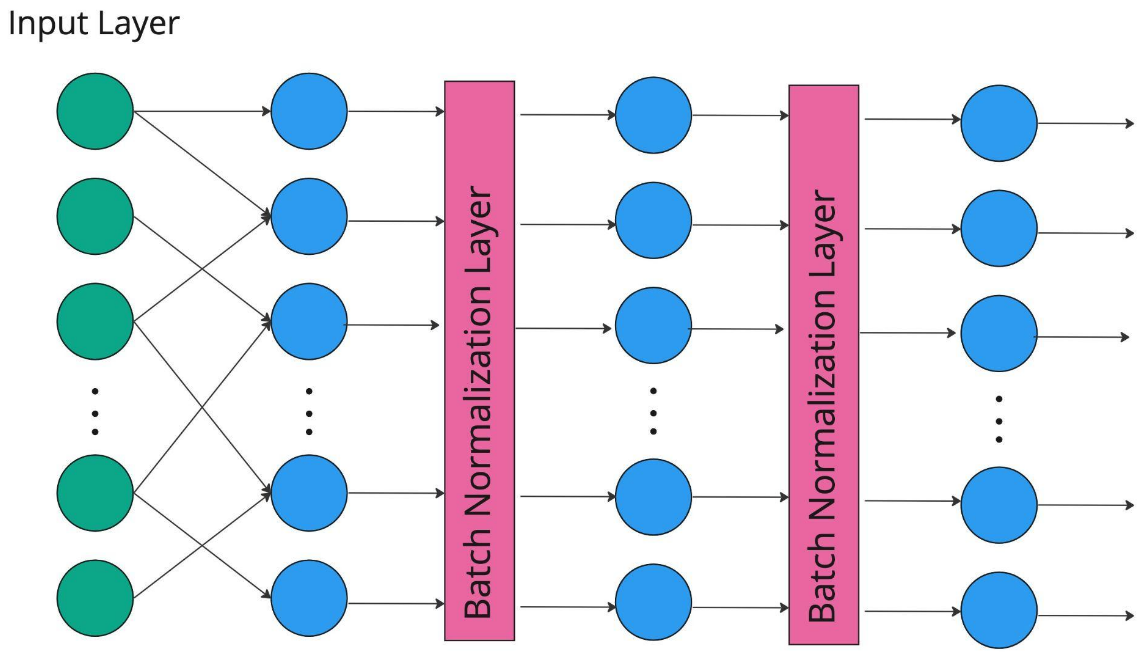




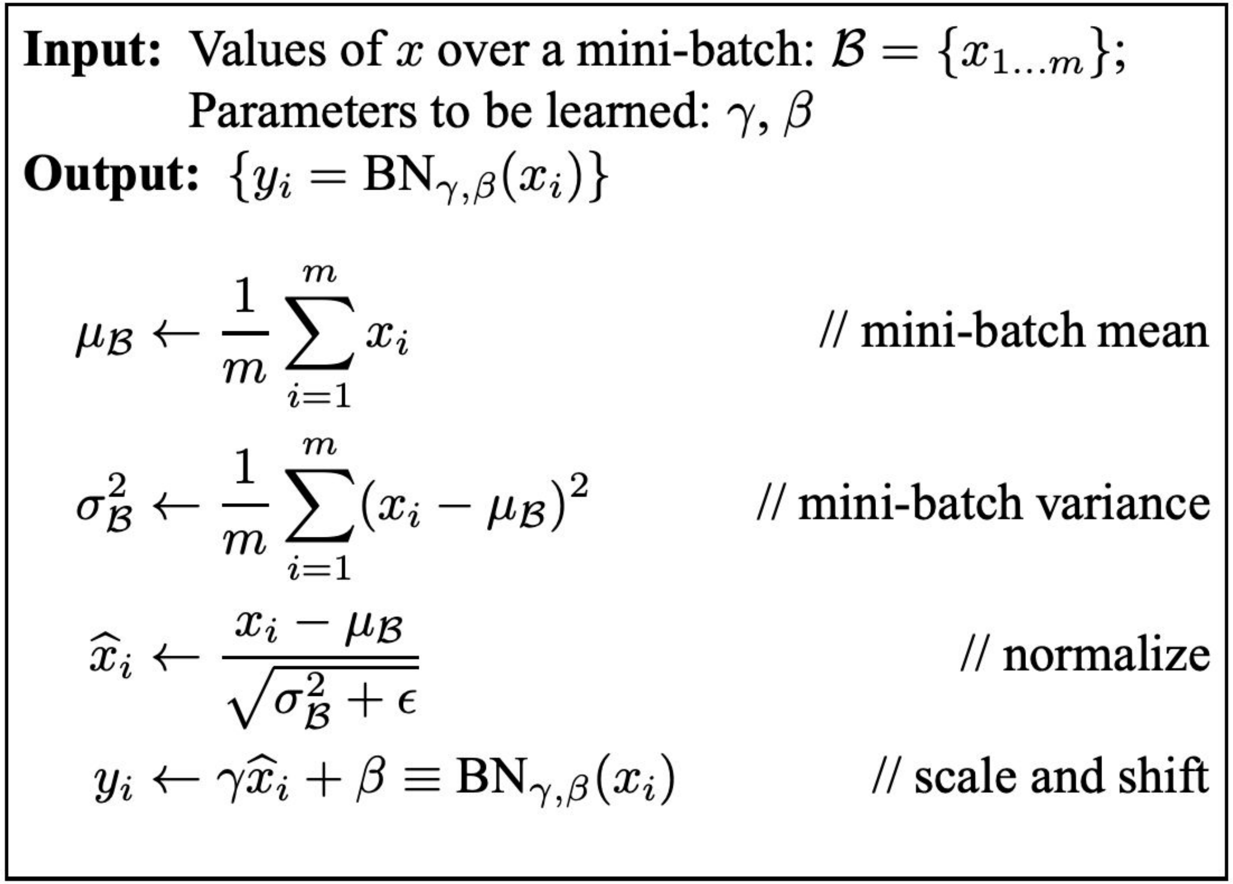
Efficient Backprop [LeCun et al. 1998]

Cornell Bowers C·IS

Batch Normalization



The Batch Normalization Algorithm



```
Input: Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
       Parameters to be learned: $\gamma$, $\beta$
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

    $\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                          // mini-batch mean
    $\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2$          // mini-batch variance
    $\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}$                      // normalize
    $y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$              // scale and shift
```

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

Cornell Bowers C·IS

BatchNorm: Inference Behavior

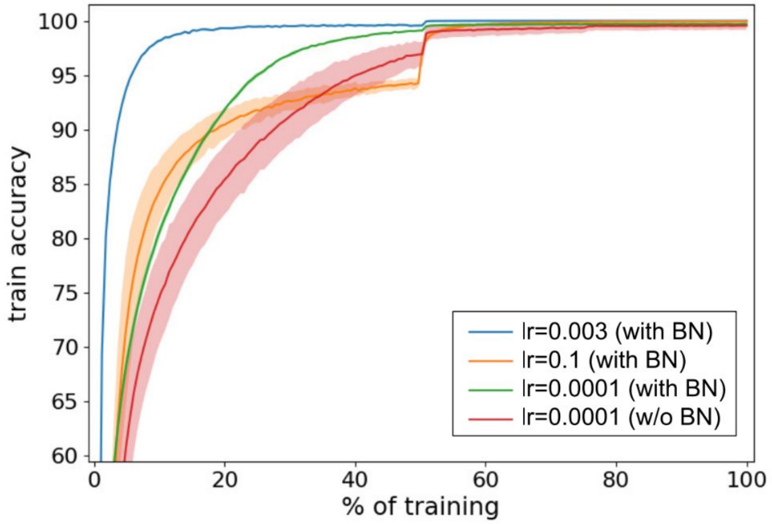
- Model inference should be deterministic
  - Normalization depends on the elements in the batch
- Solution: Use running average statistics calculated during training as:

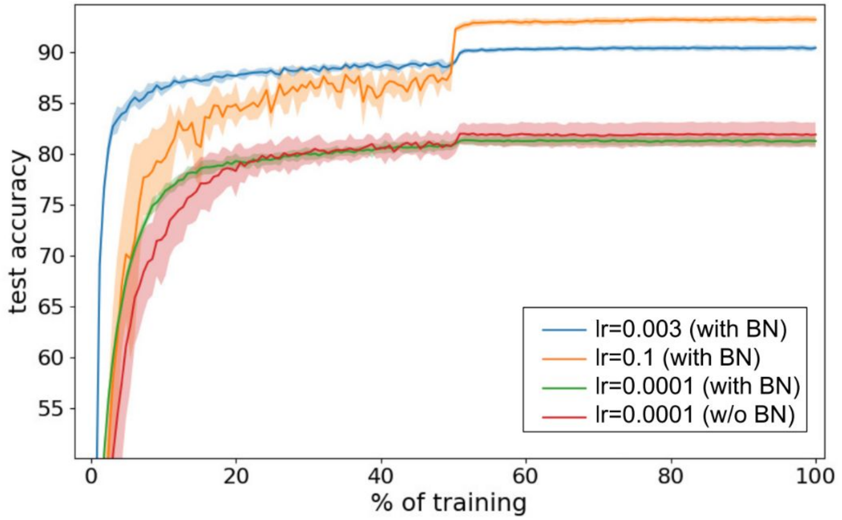
$$\mu_{\text{inf}} = \lambda \mu_{\text{inf}} + (1-\lambda)\mu_{\mathcal{B}}$$
$$\sigma^2_{\text{inf}} = \lambda \sigma^2_{\text{inf}} + (1-\lambda)\sigma^2_{\mathcal{B}}$$

Cornell Bowers C·IS

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

Cornell Bowers C·IS

Why does a large learning rate help?

- Noise of the gradient estimate scales with the learning rate (Bjorck et al. 2018)

$$\alpha\nabla_{SGD}(x) = \underbrace{\alpha\nabla\ell(x)}_{\text{gradient}} + \underbrace{\frac{\alpha}{|B|}\sum_{i\in B}\big(\nabla\ell_i(x) - \nabla\ell(x)\big)}_{\text{error term}}$$

$$\mathbb{E}\Big[\tfrac{\alpha}{|B|}\textstyle\sum_{i\in B}\big(\nabla\ell_i(x) - \nabla\ell(x)\big)\Big] = 0 \qquad C = \mathbb{E}\big[\|\nabla\ell_i(x) - \nabla\ell(x)\|^2\big]$$

$$\mathbb{E}\big[\|\alpha\nabla\ell(x) - \alpha\nabla_{SGD}(x)\|^2\big] \le \frac{\alpha^2}{|B|}C$$

Cornell Bowers C·IS

Why does a large learning rate help?

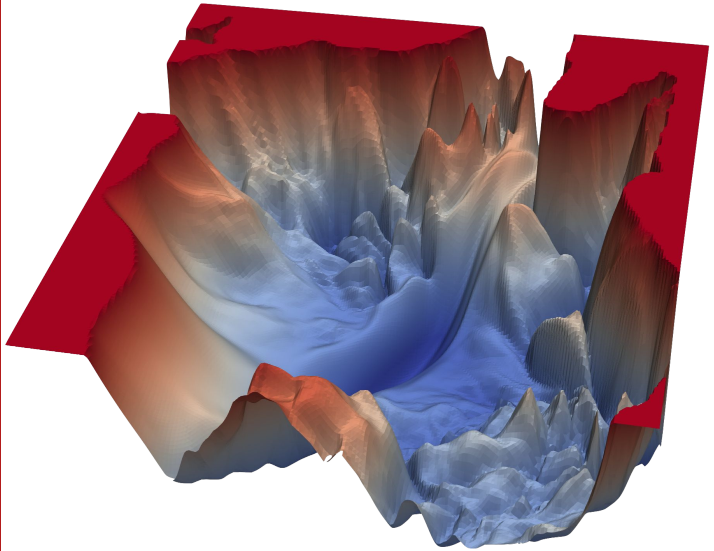
- Noise of the gradient estimate scales with the learning rate (Bjorck et al. 2018)
- Large learning rates have noisier updates
  - Actually improves generalization
- Large learning rate acts like a regularizer

$$\mathbb{E}\left[\|\alpha\nabla\ell(x) - \alpha\nabla_{SGD}(x)\|^2\right] \leq \frac{\alpha^2}{|B|}C$$

Cornell Bowers C·IS

Conceptual Sketch

- Noisy updates are good at escaping sharp minima
- Flatter minima generalize better



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

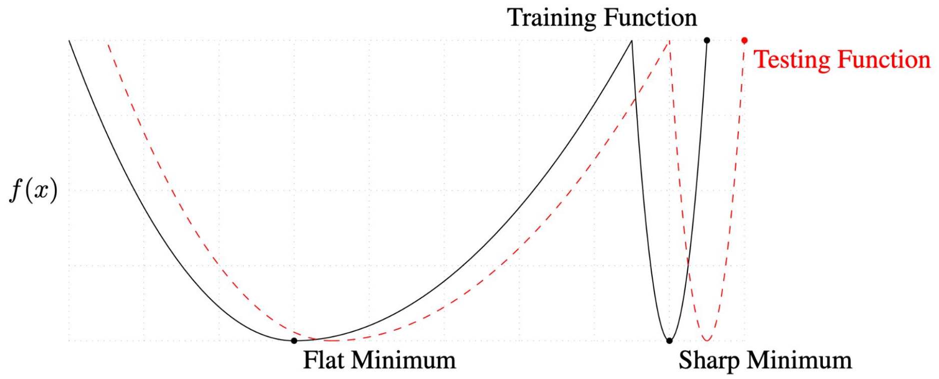


Figure 1: A Conceptual Sketch of Flat and Sharp Minima. The Y-axis indicates value of the loss function and the X-axis the variables (parameters)

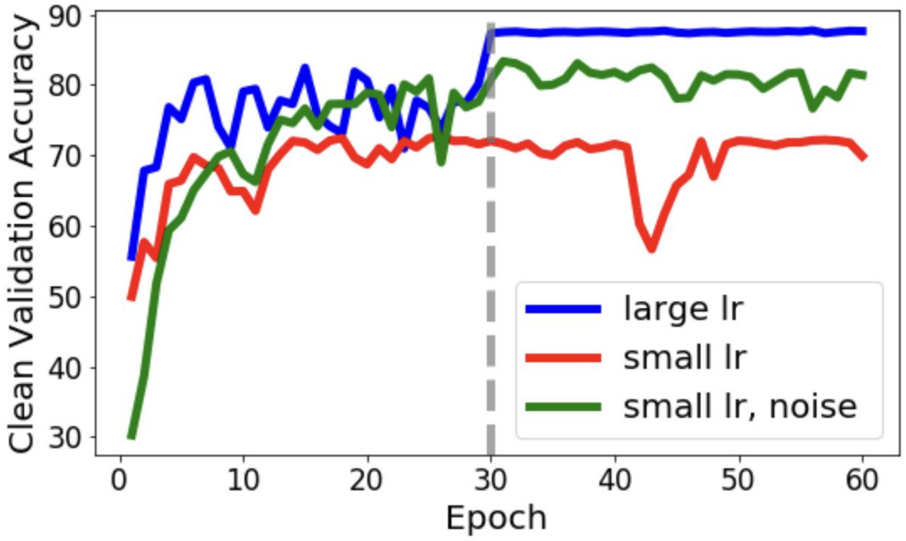
“On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima” by Keskar et al., 2017

Cornell Bowers C·IS

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)

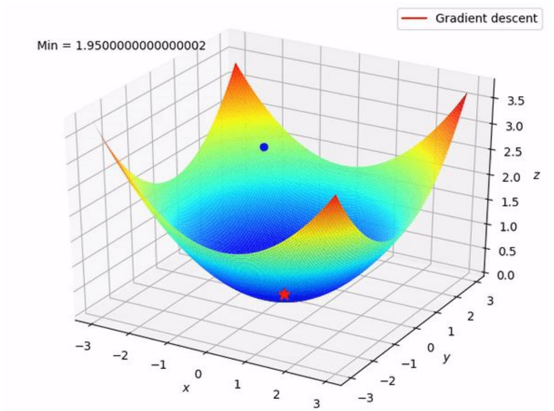


“Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks” by Li et al., 2019

Cornell Bowers C·IS

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



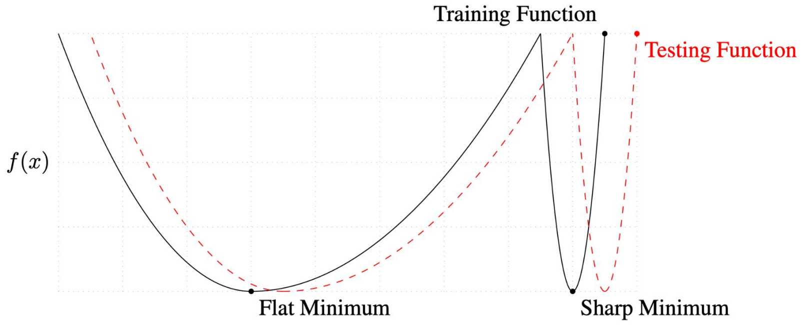


Figure 1: A Conceptual Sketch of Flat and Sharp Minima. The Y-axis indicates value of the loss function and the X-axis the variables (parameters)

Cornell Bowers C·IS

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!

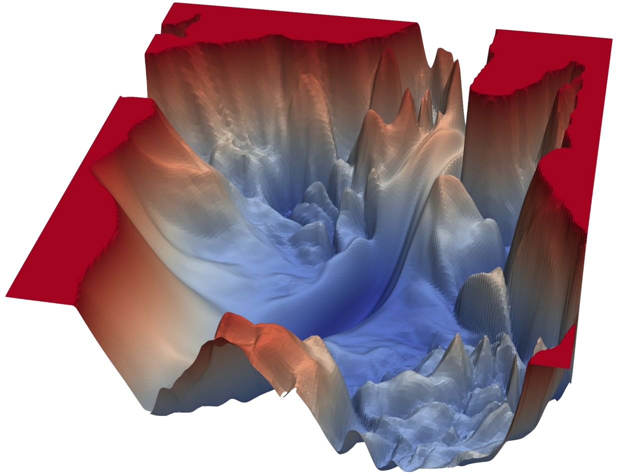
Algorithmic Regularization:

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

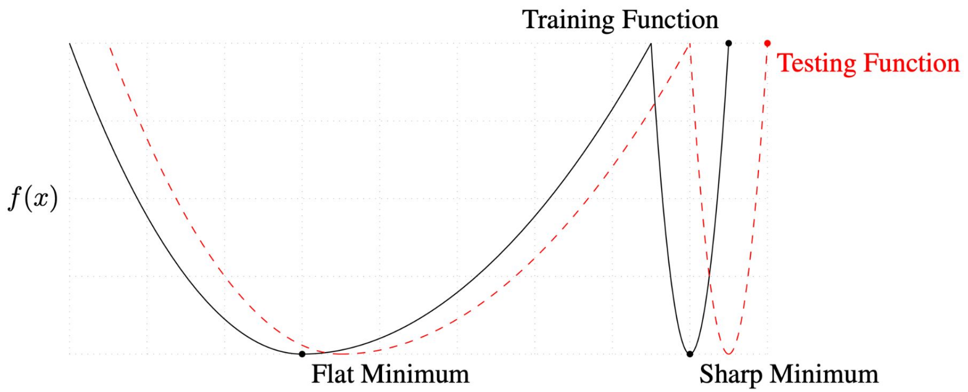
where $r_{\mathcal{A}}(\mathbf{w})$ is some measure of model complexity implicitly controlled by the learning algorithm, $\mathcal{A}$

Non-Convex Optimization

- Non-Convex optimization: Many different minima (and saddle points)
  - Different optimization algorithms will find different minima
- Training algorithms are biased towards “flatter” minima that generalize well



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

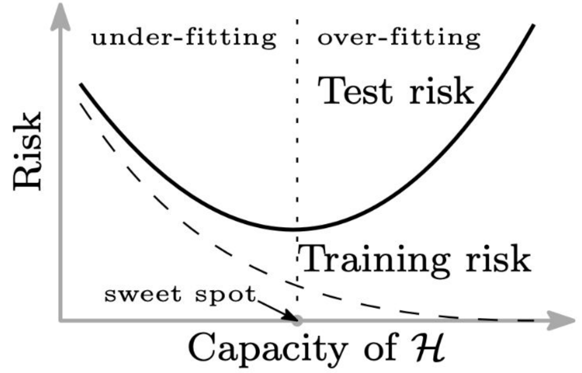


“On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima” by Keskar et al., 2017

Cornell Bowers C·IS

Zhang et al. (2017) Memorization Experiment

- *"Deep neural networks easily fit random labels"* (Zhang et al., 2017)



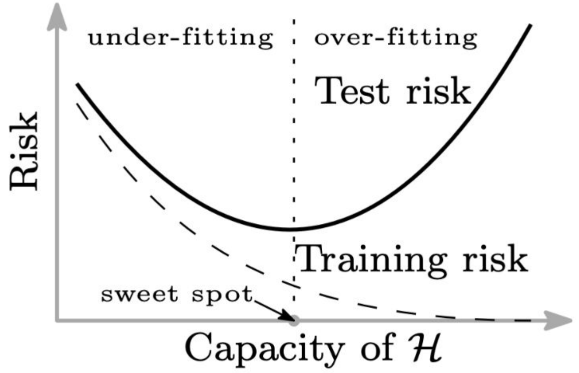
model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12

"Understanding deep learning requires rethinking generalization" by Zhang et al., 2017

Cornell Bowers C·IS

Zhang et al. (2017) Memorization Experiment

- *"Explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error."* (Zhang et al., 2017)



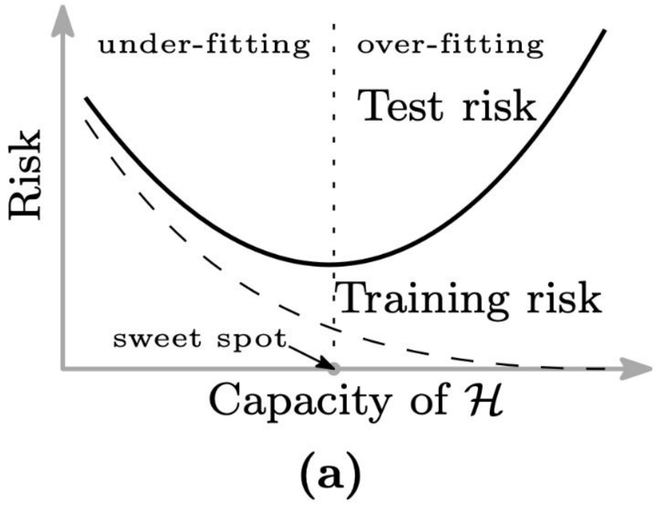
model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12

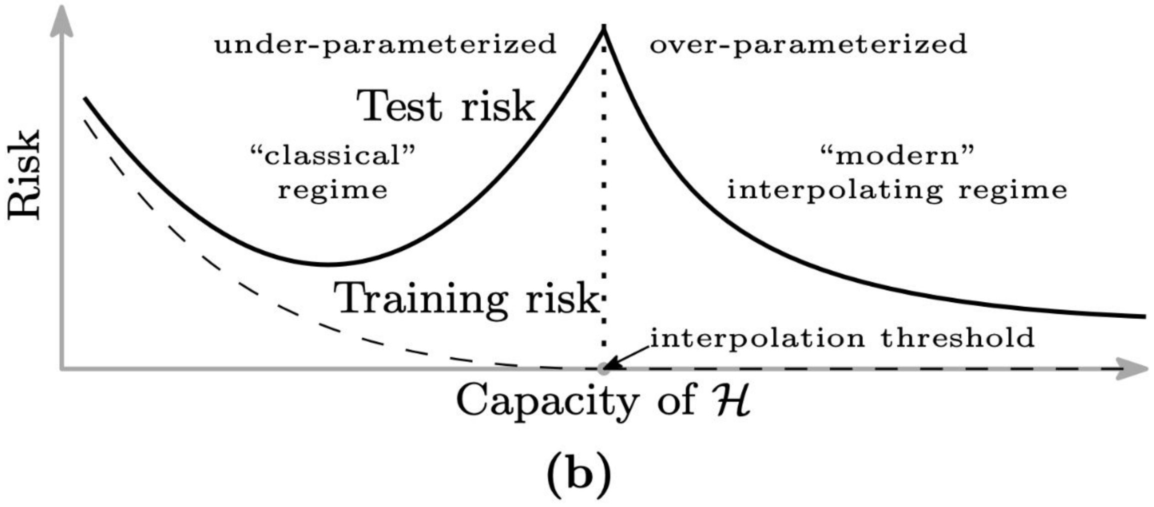
"Understanding deep learning requires rethinking generalization" by Zhang et al., 2017

Cornell Bowers C·IS

Deep Double Descent

- Neural networks can exhibit a double descent curve in practice



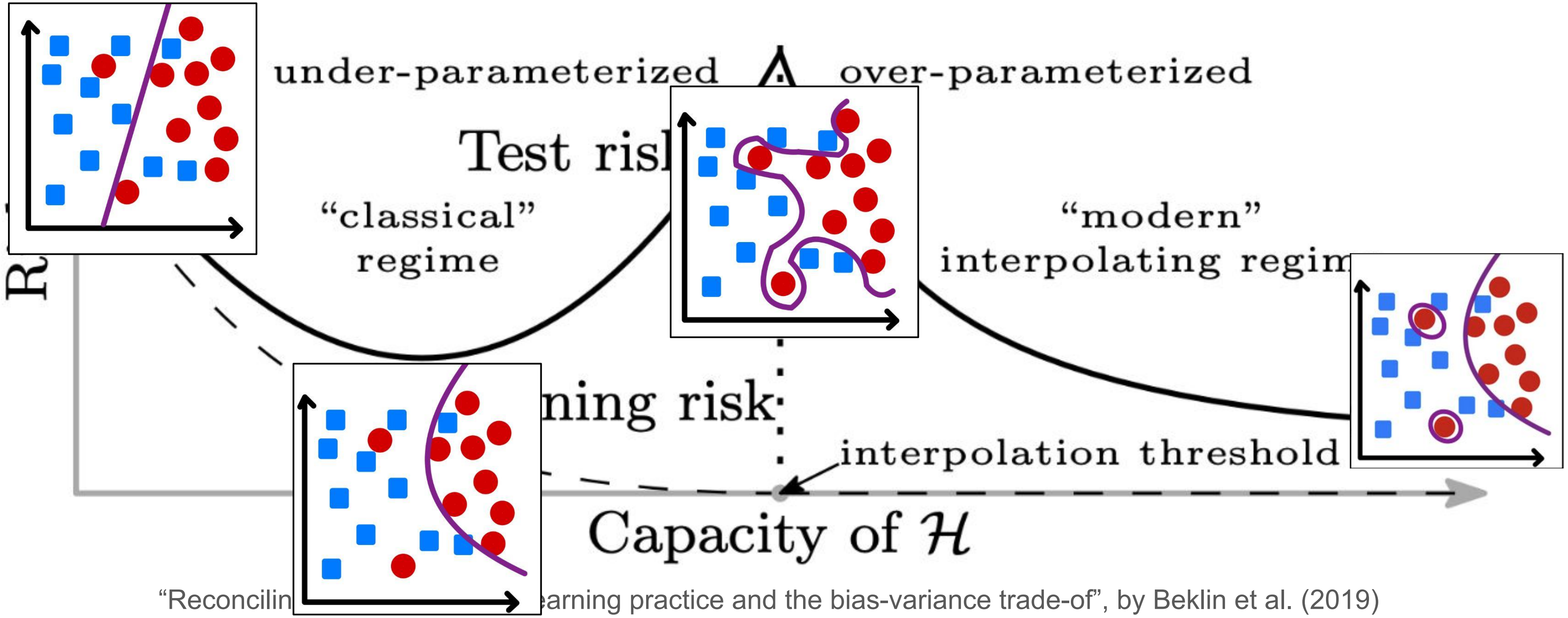


"Reconciling modern machine learning practice and the bias-variance trade-of", by Beklin et al. (2019)

Cornell Bowers C·IS

Deep Double Descent

- Neural networks can exhibit a double descent curve in practice

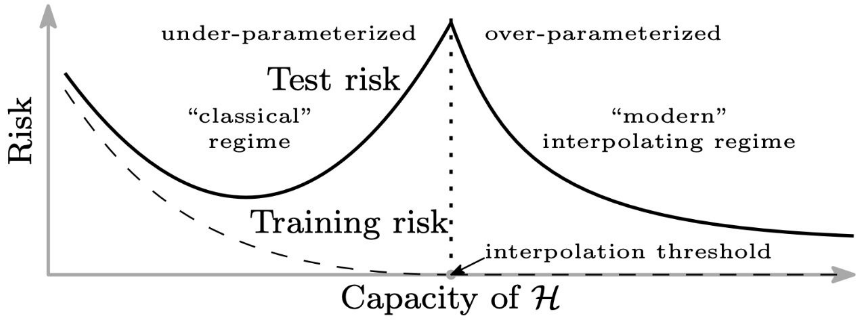


"Reconcilin earning practice and the bias-variance trade-of", by Beklin et al. (2019)

Cornell Bowers C·IS

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 ($\mathcal{L}(h) \approx 0$):

$$h = \arg\min_{h \in \mathcal{H}} \mathcal{L}(h) + \lambda \cdot r(h) \approx \arg\min_{h \in \{h : \mathcal{L}(h) \approx 0\}} r(h)$$

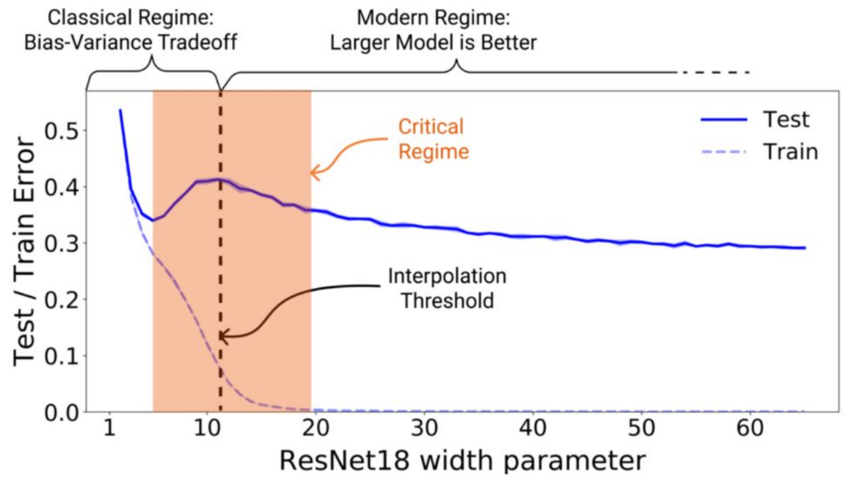
where $r(h)$ is some measure of complexity

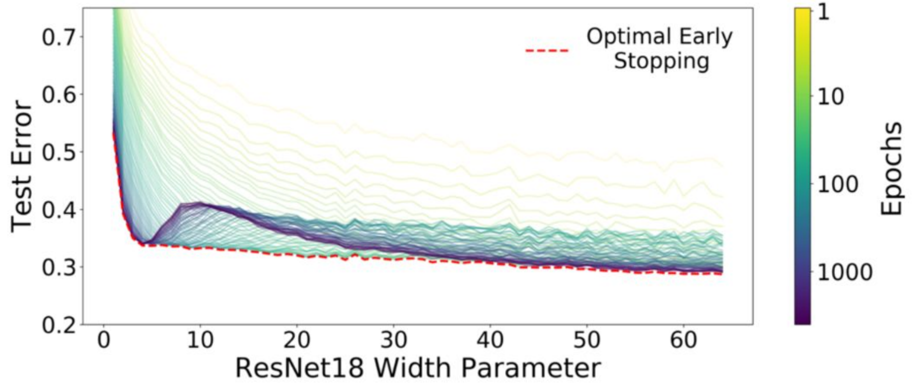
“Reconciling modern machine learning practice and the bias-variance trade-of”, by Beklin et al. (2019)

Cornell Bowers C·IS

Deep Double Descent

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



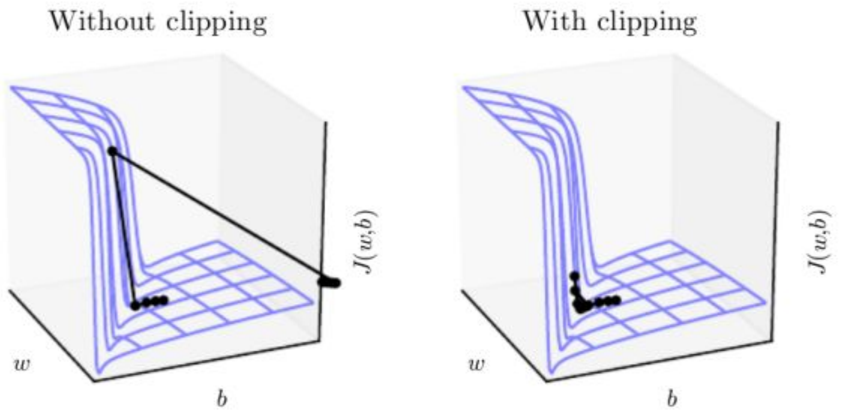


“Deep Double Descent: Where Bigger Models and More Data Hurt”, by Nakkiran et al., 2019

Cornell Bowers C·IS

Gradient Clipping

- Exploding gradients result in unstable training
- Optimization is hard when you have very large gradients



Gradient clipping algorithm:

```
if  $\|\mathbf{g}\| > \tau$ :
    $\mathbf{g}' = \frac{\tau}{\|\mathbf{g}\|}\mathbf{g}$
else:
    $\mathbf{g}' = \mathbf{g}$
```

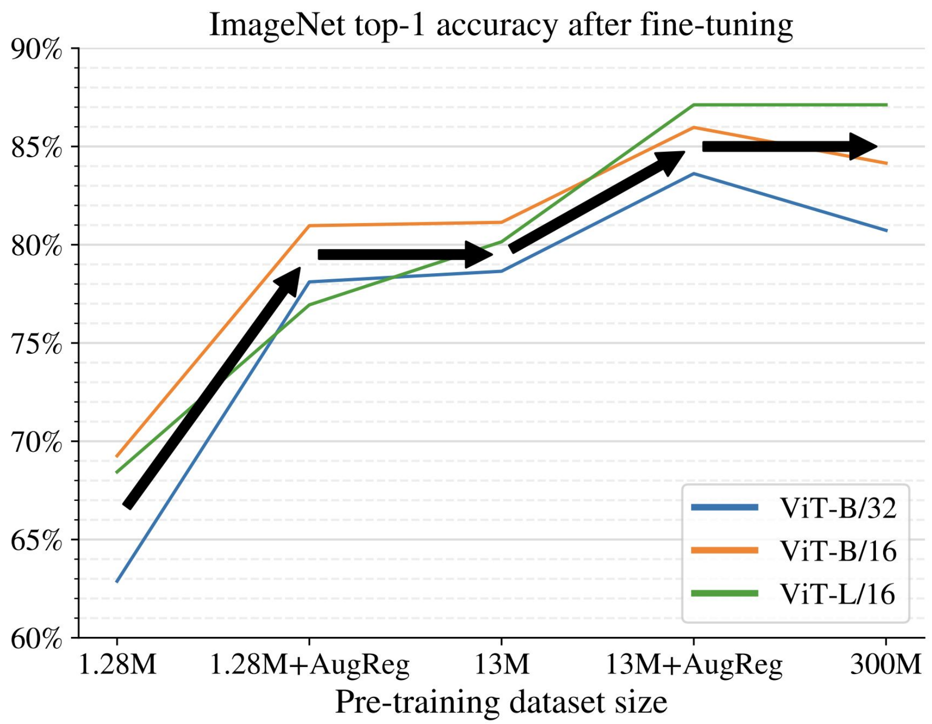
τ: Max gradient norm

https://neptune.ai/blog/understanding-gradient-clipping-and-how-it-can-fix-exploding-gradients-problem

Cornell Bowers C·IS

Regularization and Data Augmentation

- 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, Steiner et al. 2022

Recap

- Use a combination of various regularization techniques to improve generalization
    - L1/L2 regularization, dropout, etc.

- The training algorithm itself (e.g. SGD) is a critical regularizer in deep learning

- Neural networks are expressive enough to memorize the training data and fail to generalize
    - Generalize extremely well in practice

Cornell Bowers C·IS

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!

Cornell Bowers C·IS

Thanks!