PAT 498/598 (Fall 2024)

Special Topics: Generative AI for Music and Audio Creation

Lecture 6: Optimization

Instructor: Hao-Wen Dong

Assignment 1: AI Song Contest

- Please listen to the **ten finalists of AI Song Contest 2024** and **read the about pages** by clicking the cover arts
- **Vote for your favorites**
- **Answer the following questions** (in 10-20 sentences each)
  - Which is your favorite song? What did they do well? What can be improved?
  - What is one dimension that most finalists didn't look into or didn't do well on?
  - What tasks are easy for current AI? What are difficult?

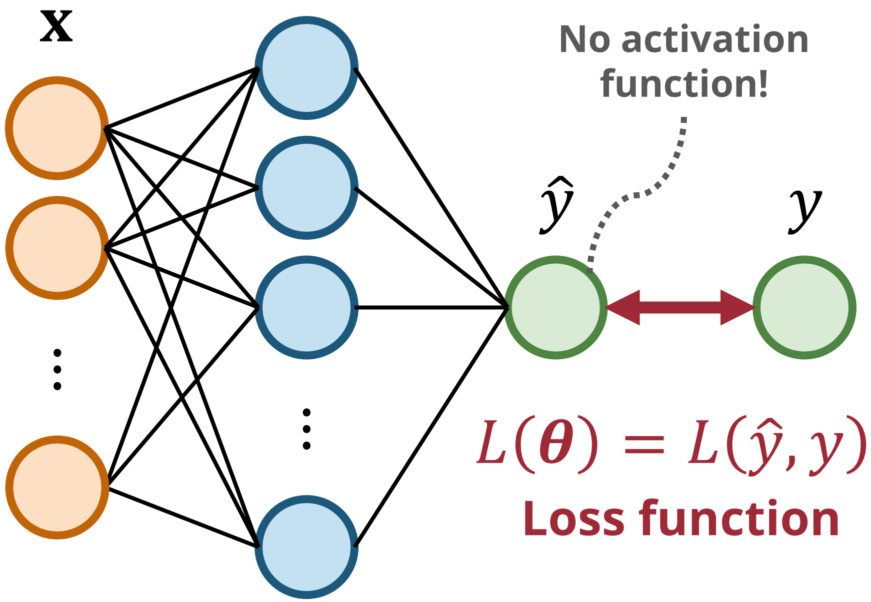
aisongcontest.com/the-2024-finalists

Assignment 1: AI Song Contest

- Instructions will be released on Gradescope
- Due at **11:59pm ET** on **September 20**
- Late submissions: **3 point deducted per day**

aisongcontest.com/the-2024-finalists

(Recap) Common Loss Functions for Regression



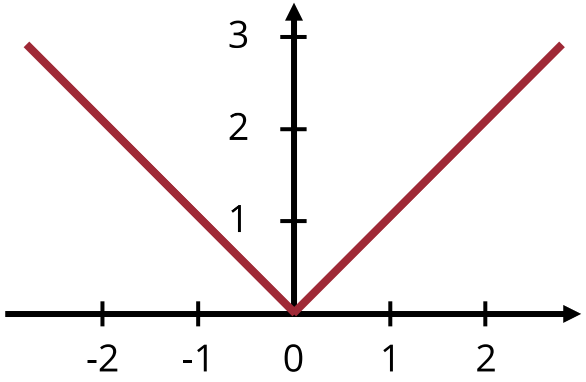
$$L(\boldsymbol{\theta}) = L(\hat{y}, y)$$

Loss function

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{y}_k, y_k)$$

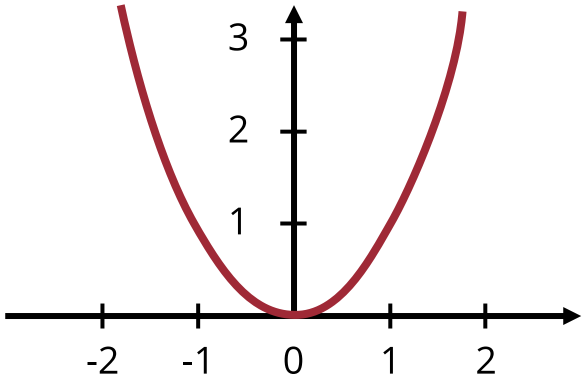
L1 loss

$$L(\hat{y}, y) = |\hat{y} - y|$$



L2 loss

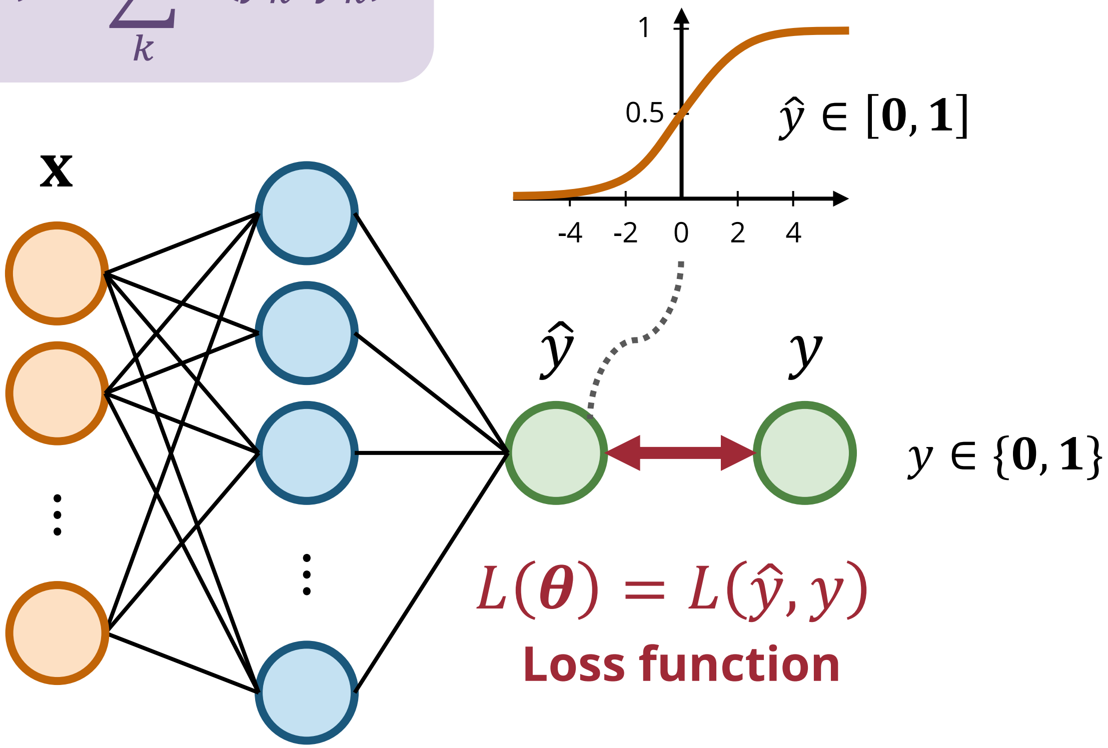
$$L(\hat{y}, y) = (\hat{y} - y)^2$$



(Recap) Binary Cross Entropy for Binary Classification

- **Logistic regression** approaches classification like regression

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{y}_k, y_k)$$

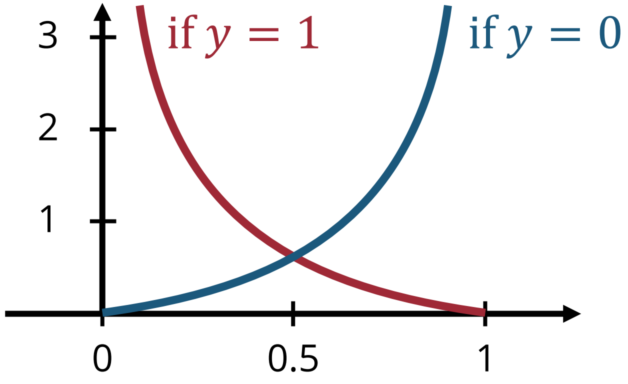


Binary cross entropy

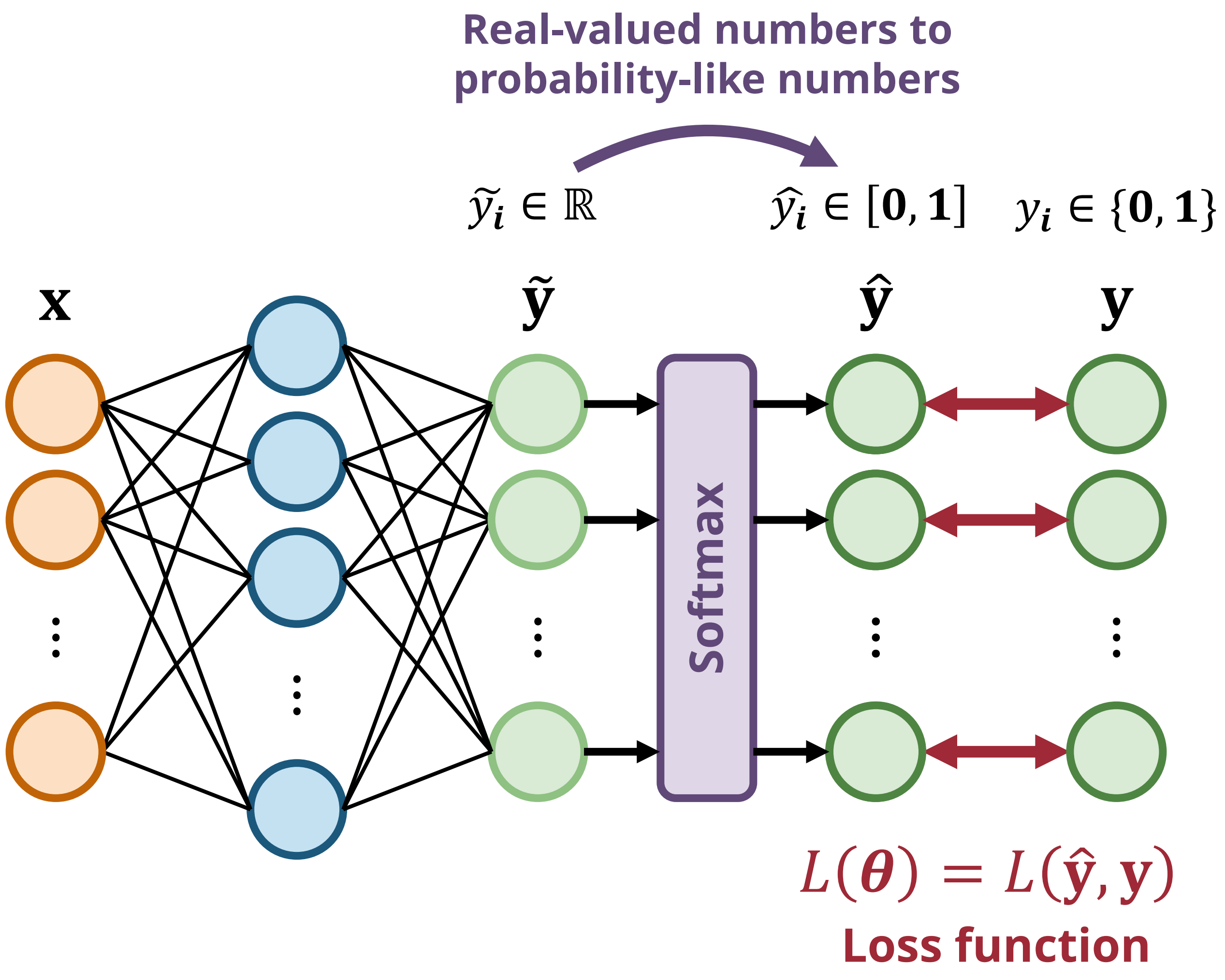
(Also called log loss)

$$L(\hat{y}, y) = \begin{cases} -\log \hat{y}, & \text{if } y = 1 \\ -\log(1-\hat{y}), & \text{if } y = 0 \end{cases}$$

$$= -y \log \hat{y} + (1-y) \log(1-\hat{y})$$



(Recap) Cross Entropy for Multiclass Classification



Softmax

$$\hat{y}_i = \frac{e^{\tilde{y}_i}}{\sum_{j=1}^{n} e^{\tilde{y}_j}}$$

Cross entropy

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i}^{n} y_i \log \hat{y}_i$$

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

(Recap) Cross Entropy for Multiclass Classification

Binary Cross Entropy

Only one of them will be one!

$$L(\hat{y}, y) = -\boxed{y} \log \hat{y} + \boxed{(1-y)} \log(1-\hat{y})$$

Cross Entropy

Only one of them will be one!

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\boxed{y_1} \log \hat{y}_1 - \boxed{y_2} \log \hat{y}_2 - \cdots - \boxed{y_i} \log \hat{y}_n$$

$$= -\underbrace{\boxed{\sum_i^n y_i \log \hat{y}_i}}_{\text{Log likelihood}}$$

(Recap) Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

↓

Define the objective
(i.e., what is good for a function)

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

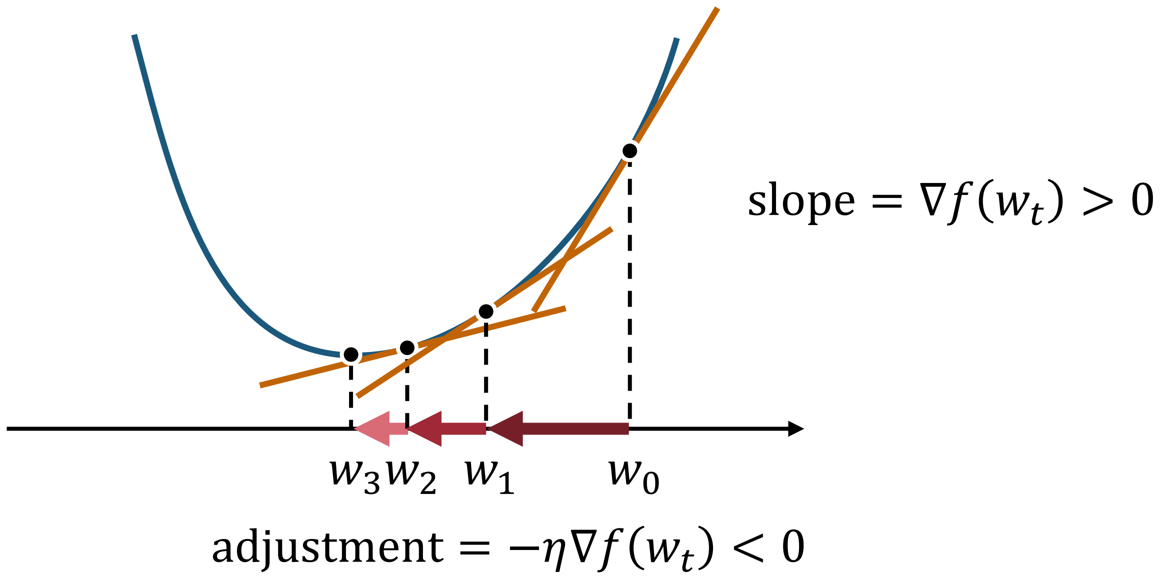
↓

Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

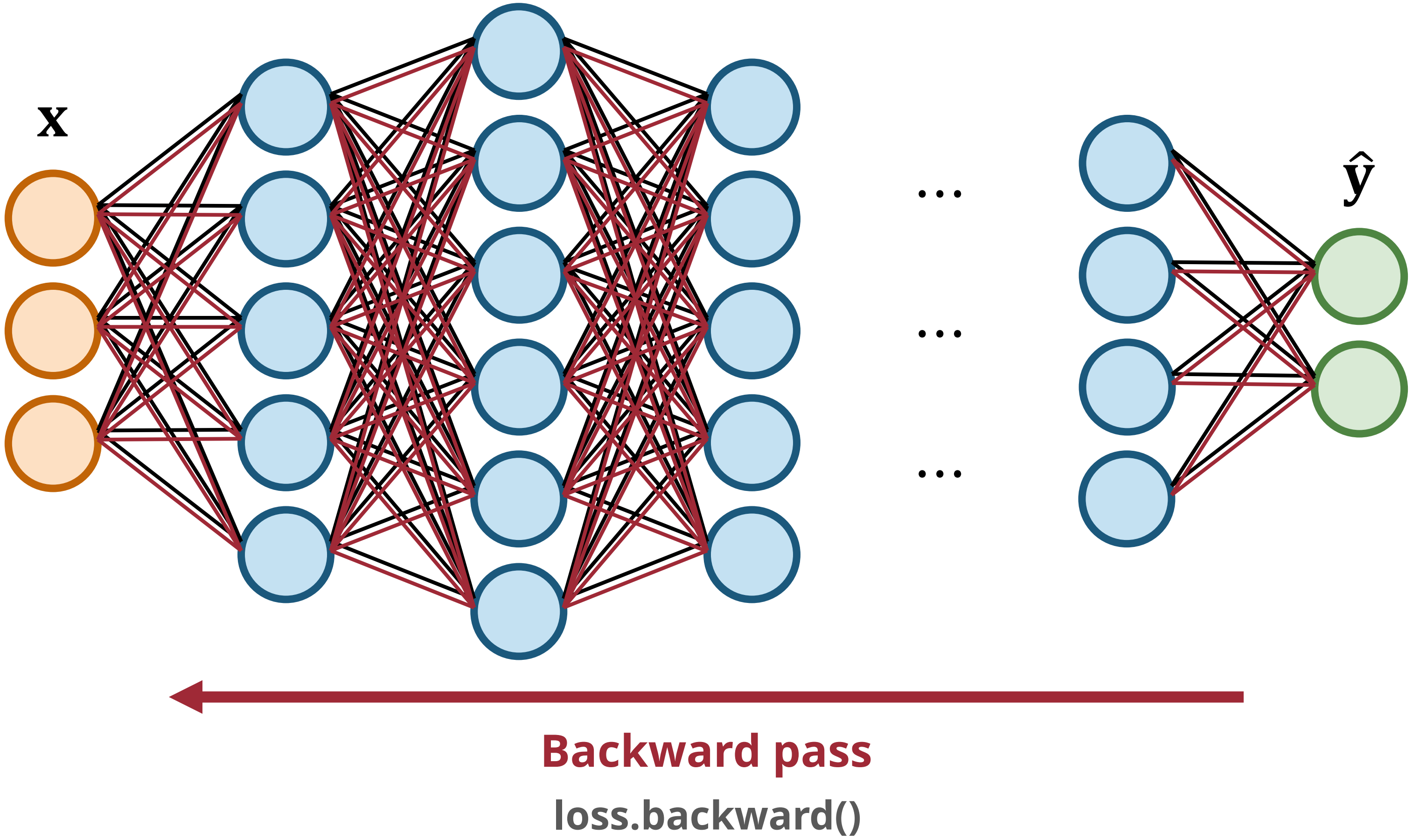
(Recap) Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



(Recap) Forward Pass & Backward Pass

$\frac{\partial L}{\partial \mathbf{x}}$ $\frac{\partial L}{\partial \mathbf{h}_1}$ $\frac{\partial L}{\partial \mathbf{h}_2}$ $\frac{\partial L}{\partial \mathbf{h}_3}$ $\frac{\partial L}{\partial \mathbf{h}_{L-1}}$



Optimization

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

↓

Define the objective
(i.e., what is good for a function)

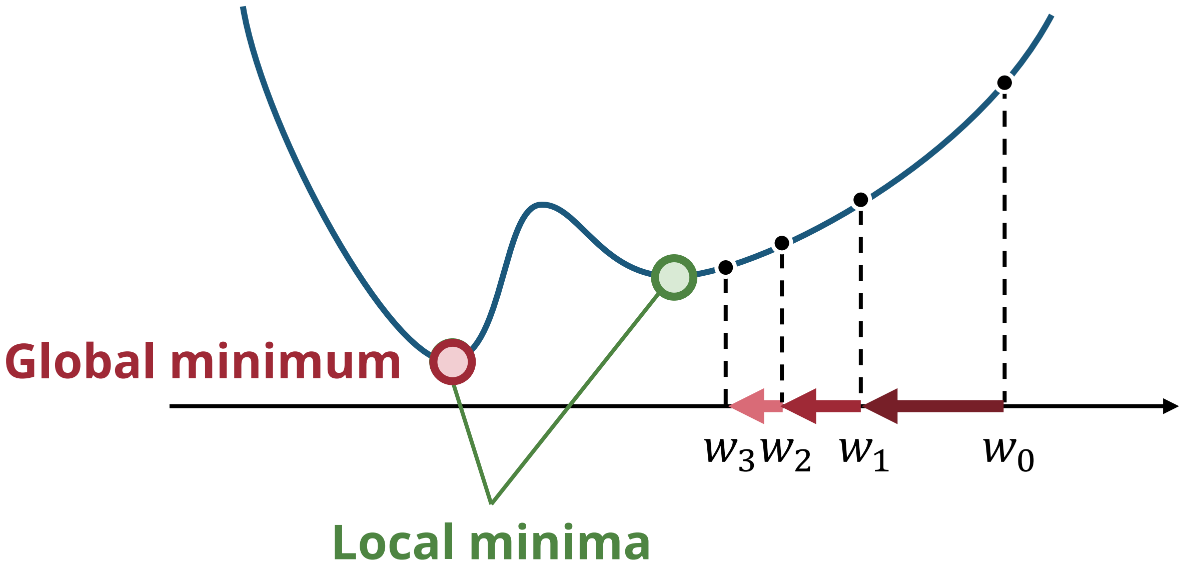
$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

↓

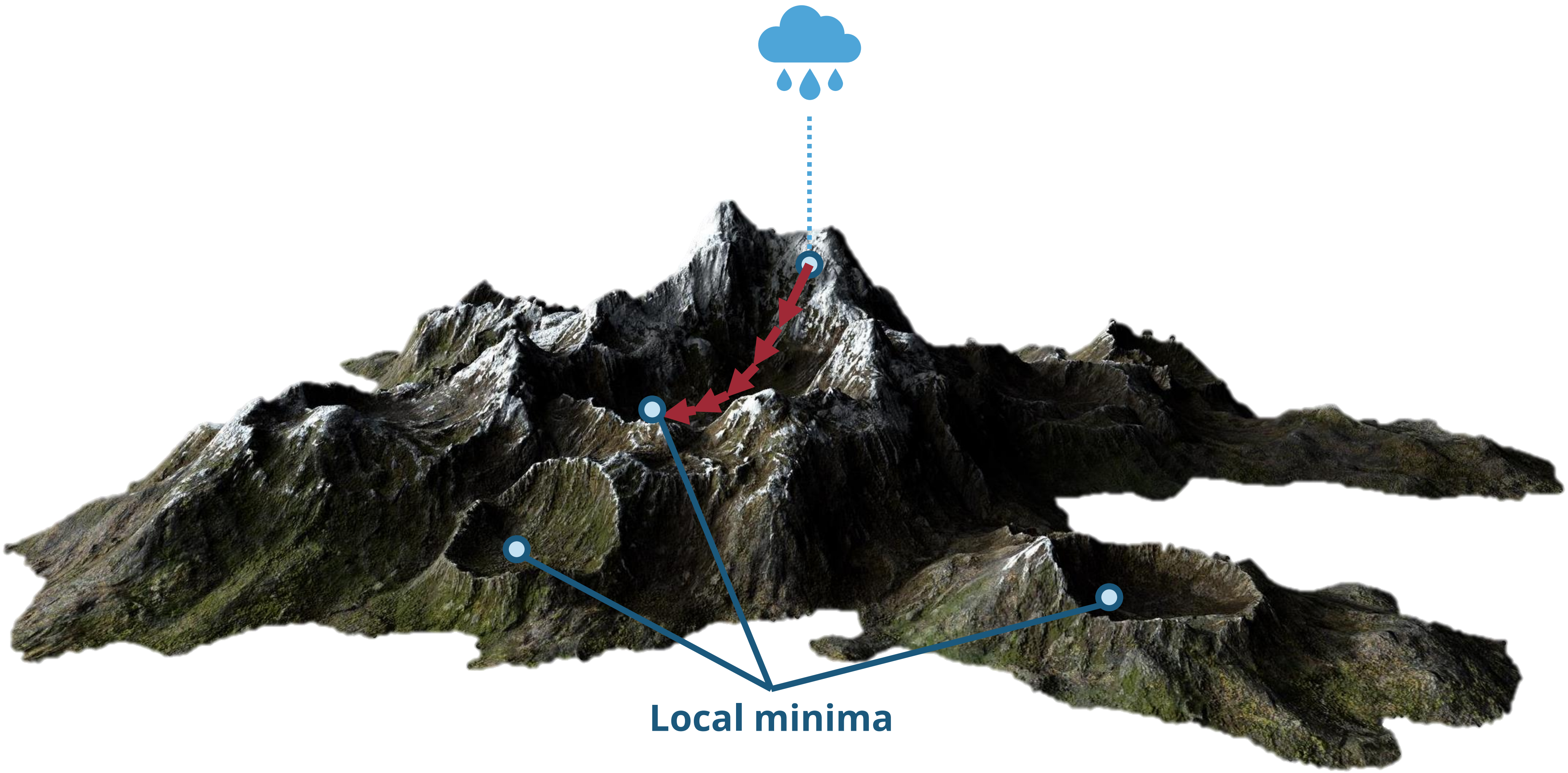
Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Gradient Descent Finds a Local Minimum

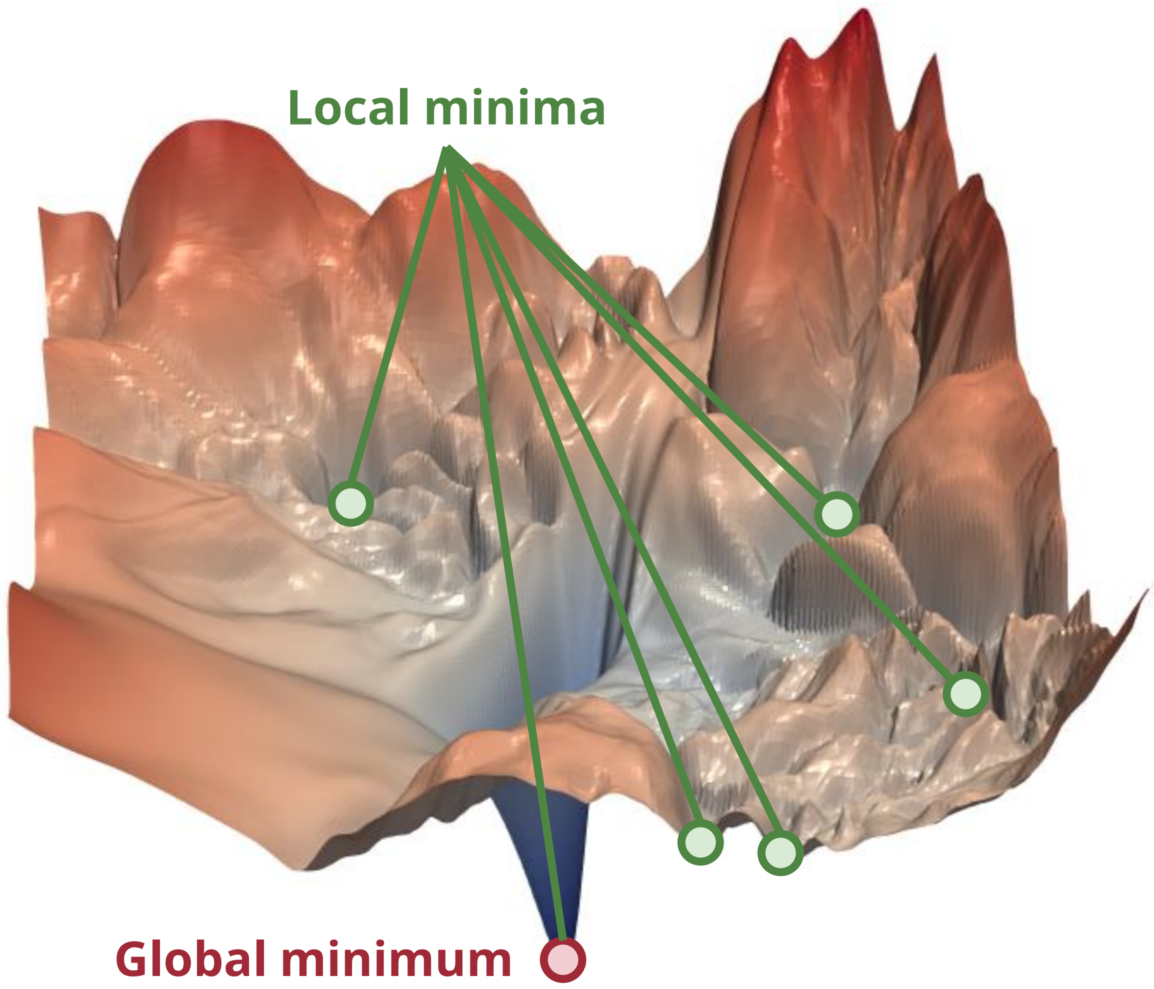


Gradient Descent Finds a Local Minimum



substance3d.adobe.com/community-assets/assets/9fe77d4279e12fe2fbd90b8f7ca18146a565f7ee

Local Minima in Complex Loss Landscape



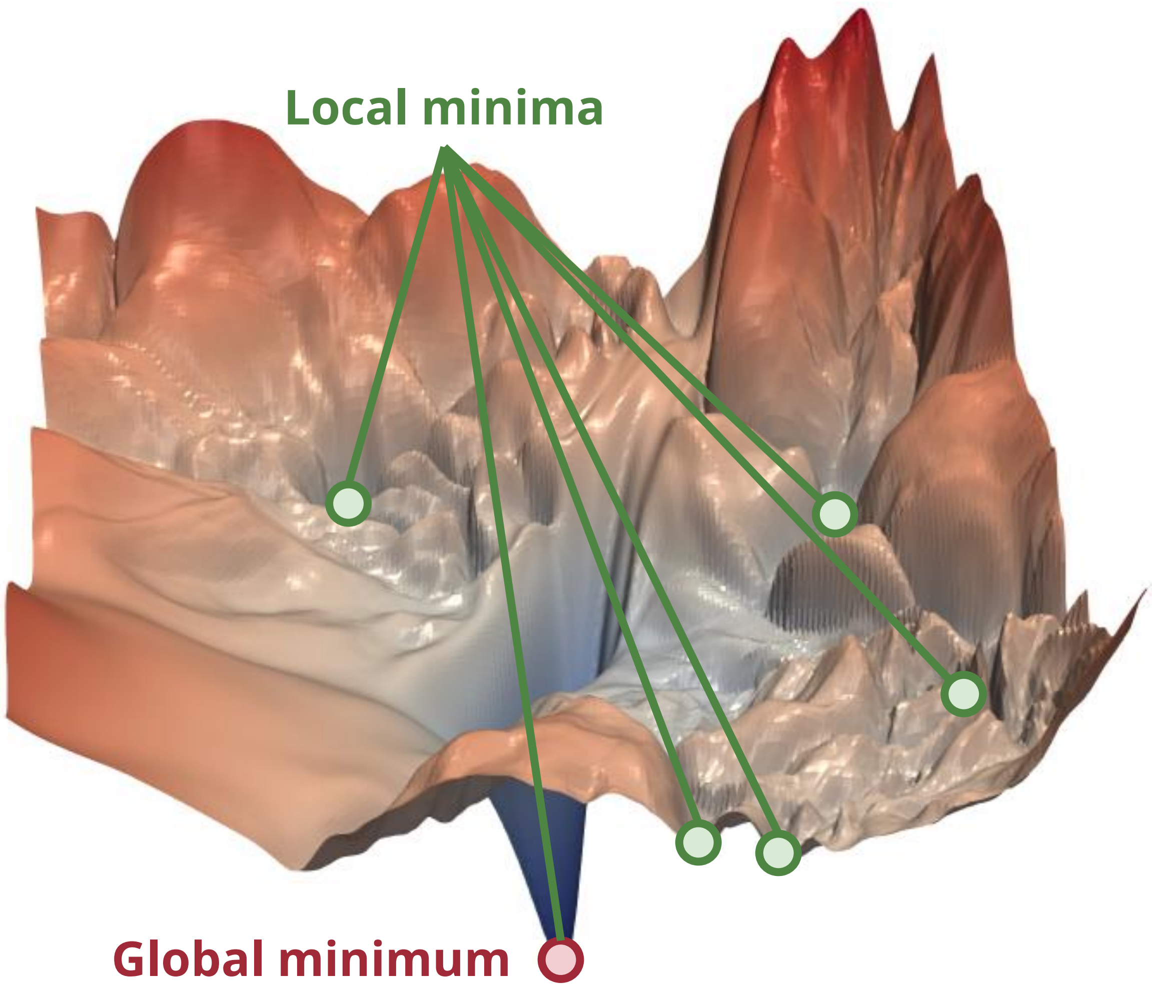
Solution 1
Use an optimizer with adaptive learning rate

Solution 2
Use a stochastic optimizer

Solution 3
Make the loss landscape smoother

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Local Minima in Complex Loss Landscape



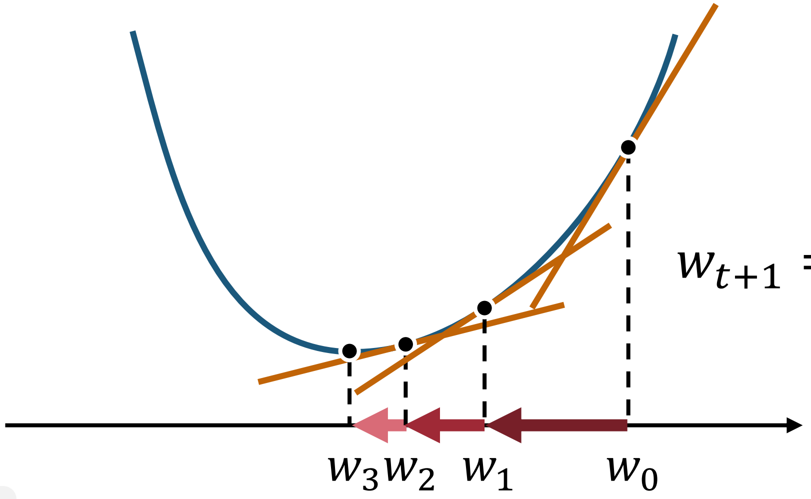
Solution 1
Use an optimizer with adaptive learning rate

Solution 2
Use a stochastic optimizer

Solution 3
Make the loss landscape smoother

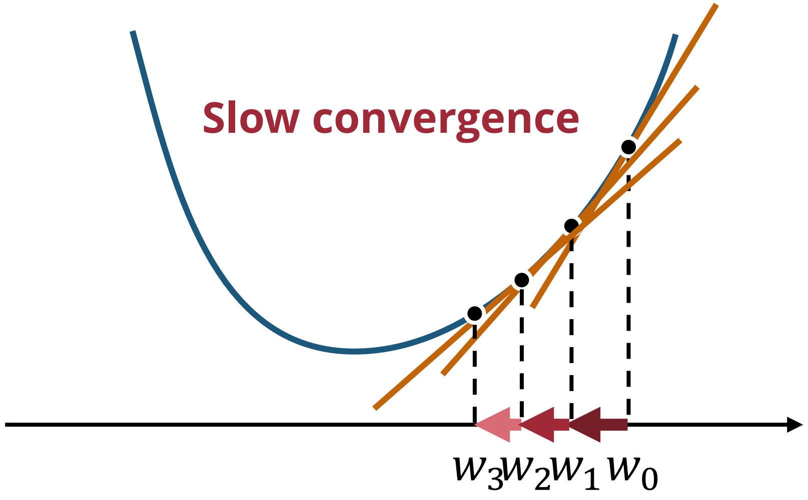
Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Learning Rate in Gradient Descent

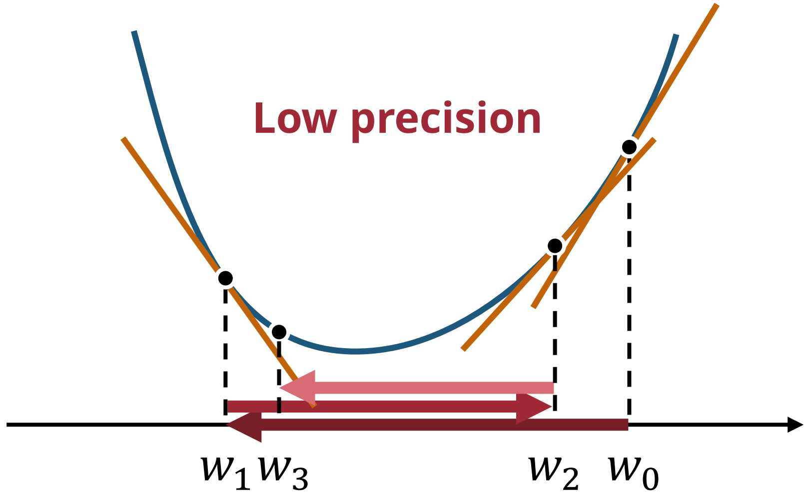


$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

Smaller learning rate



Larger learning rate



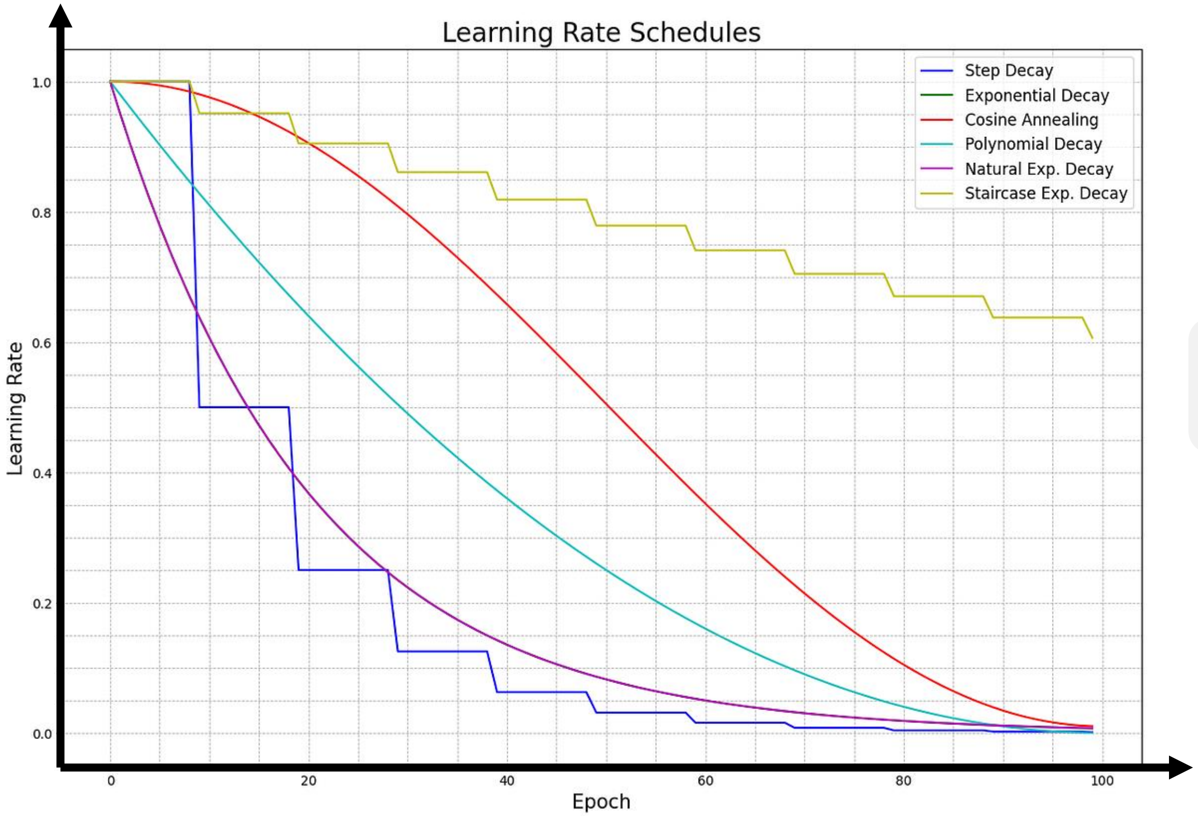
Learning Rate Decay

- We want different learning rates as the training evolves
  - At the beginning, a **large learning rate** helps us **quickly approach the target**
  - At the end, a **smaller learning rate** helps us **get a better precision**

Learning Rate Decay

- **Intuition**: **Reduce the learning rate** when we get closer to the target

Learning rate



Epoch

But how do we choose the initial value?

towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Adaptive Learning Rate

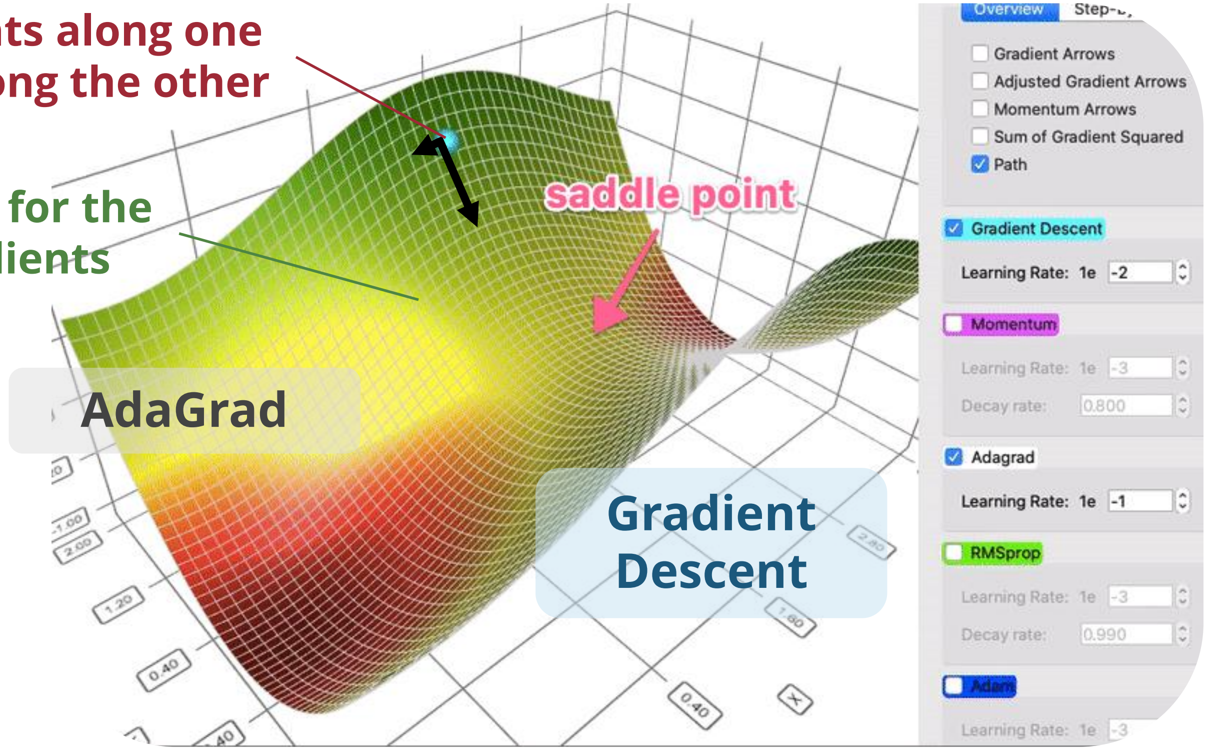
- **Key**: **Adjust the learning rate *dynamically*** based on training dynamics

Gradient-based Adaptive Learning Rate

- **Intuition**: **Compensate axis that has little progress** by comparing the current gradients to the previous gradients

Large gradients along one axis, small along the other

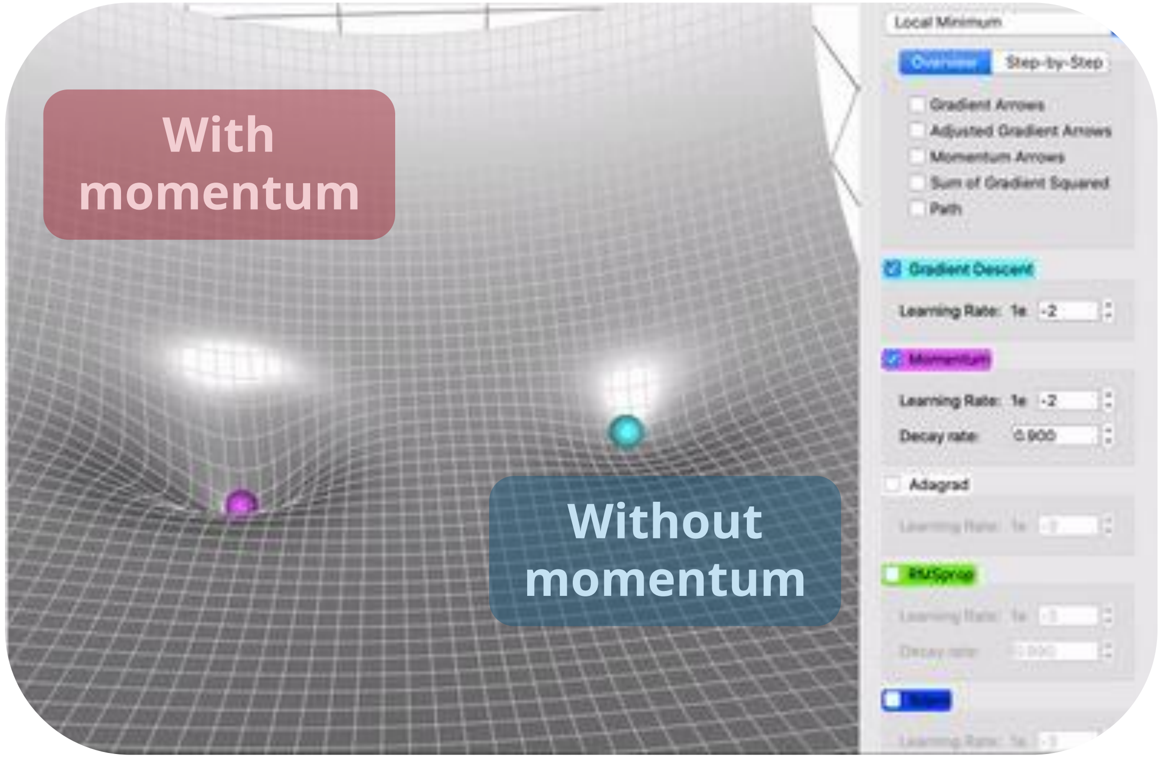
Use larger learning rate for the axis with smaller gradients



towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

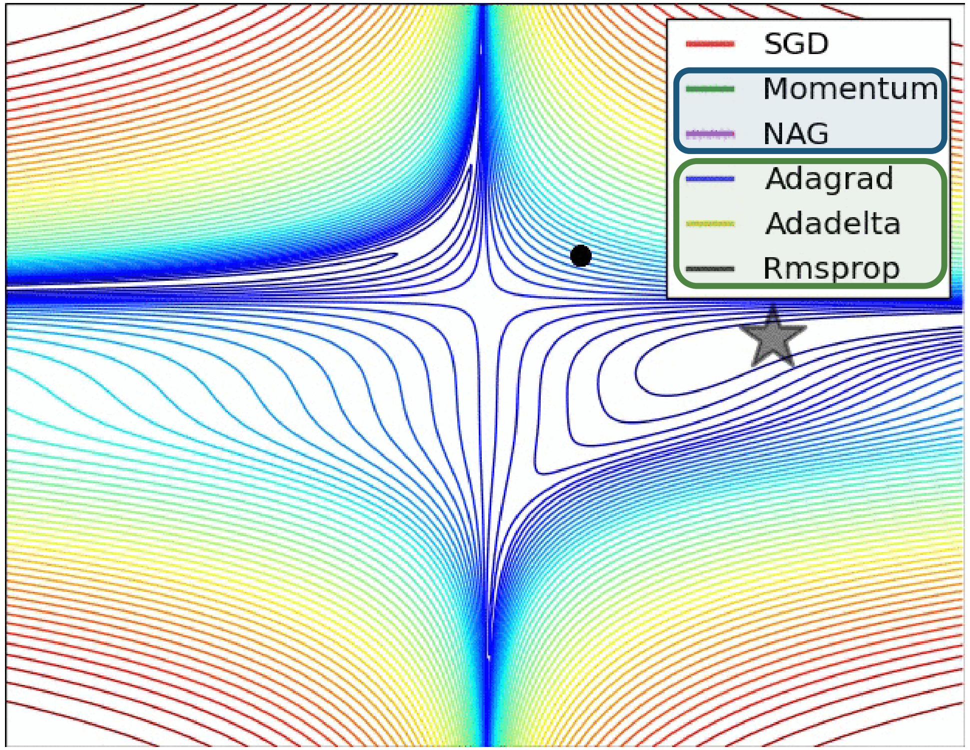
Momentum

- **Intuition**: Maintain the momentum to **escape from local minima**



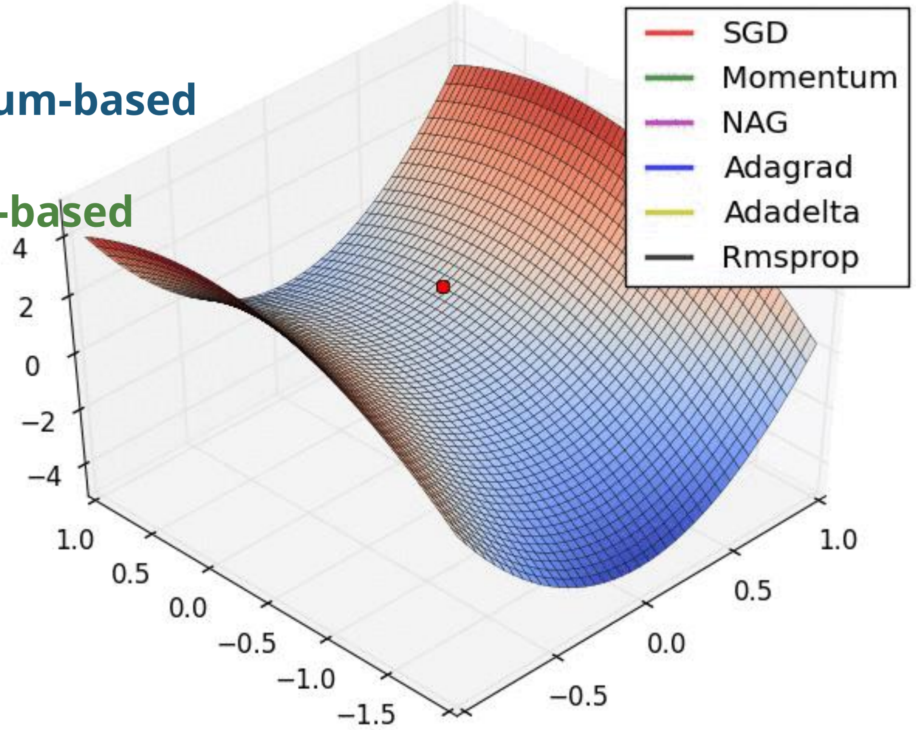
towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Comparison of Optimizers



Momentum-based

Gradient-based



Can we combine them?

analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

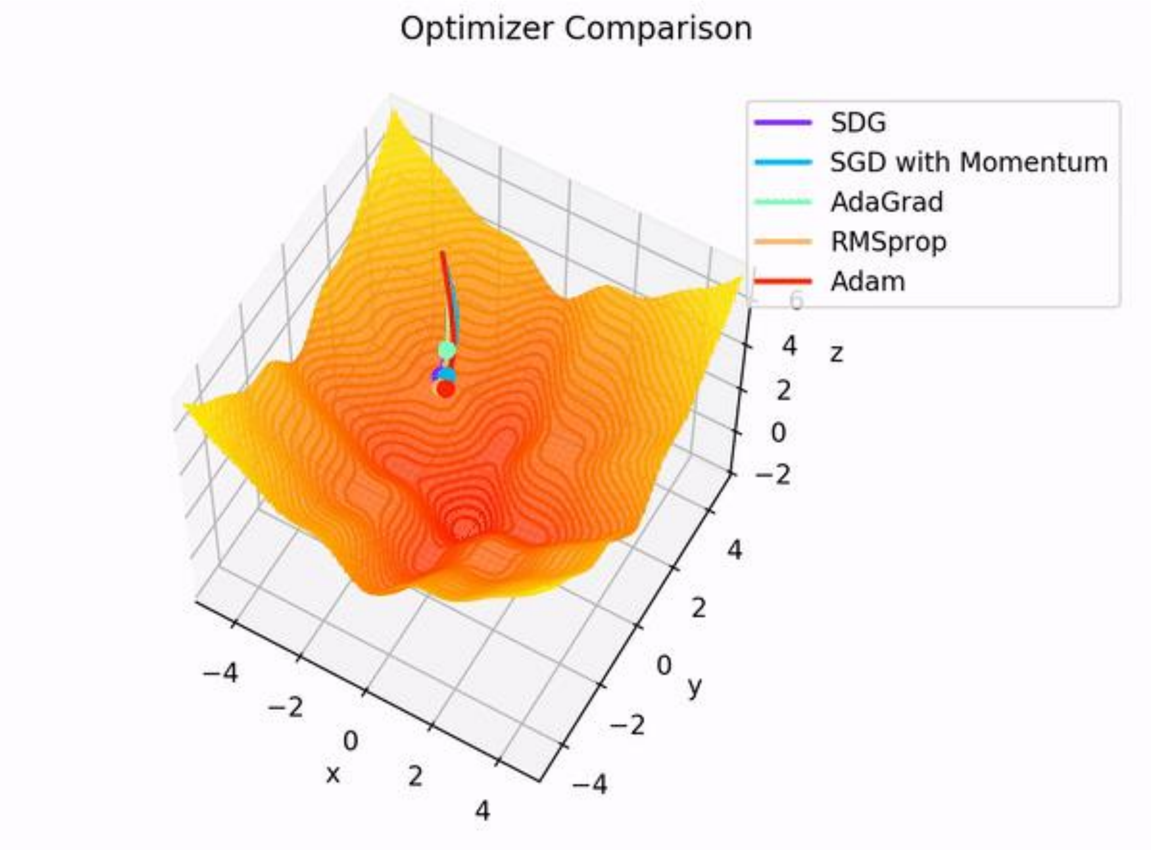
Adam Optimizer

- Combine the idea of **adaptive learning rate** and **momentum**
- Work **empirically well** in complex neural network
- The **go-to choice** for most cases

Comparison of Optimizers

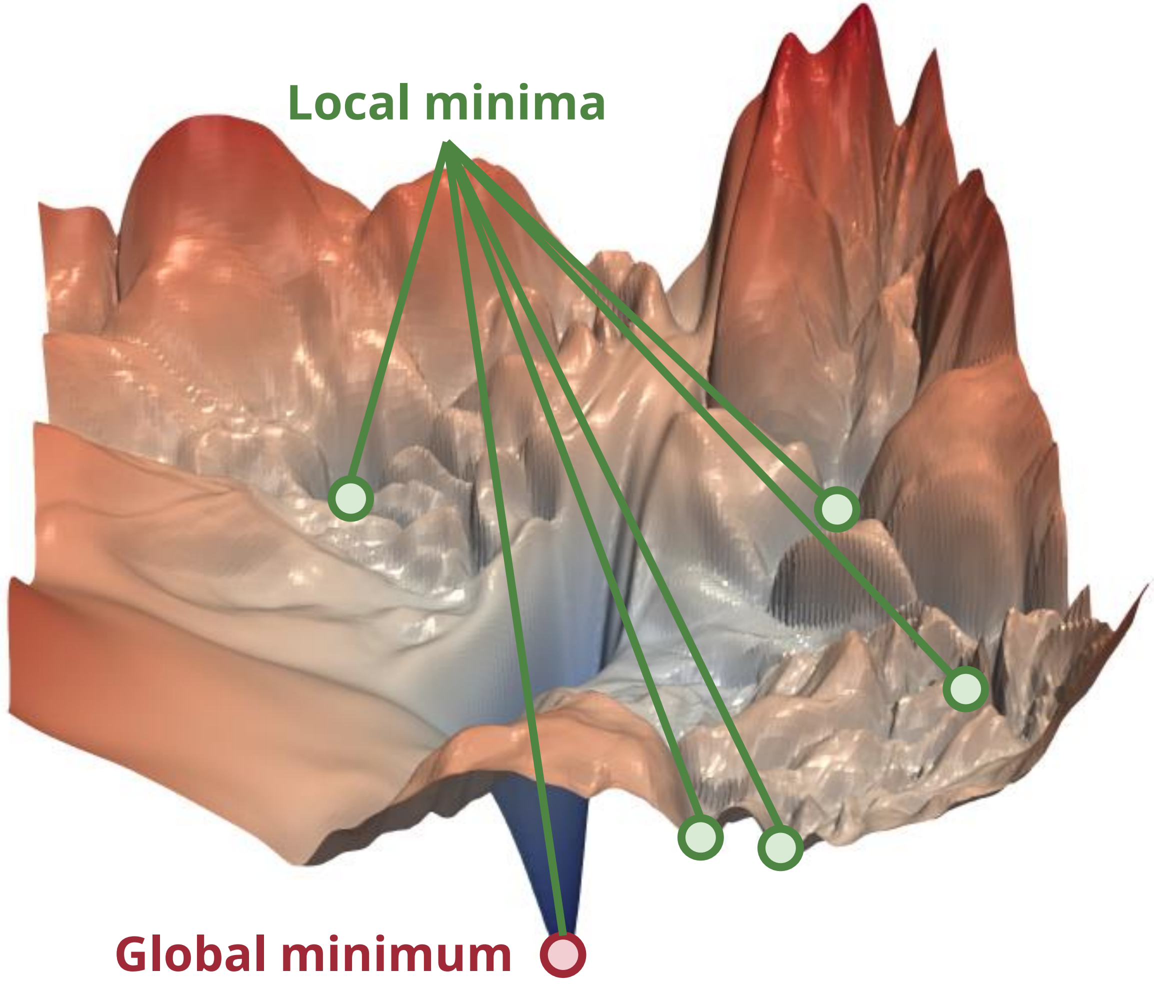
- **Momentum**
  - Gets you out of spurious local minima
  - Allows the model to explore around

- **Gradient-based adaption**
  - Maintains steady improvement
  - Allows faster convergence



medium.com/@LayanSA/complete-guide-to-adam-optimization-1e5f29532c3d

Local Minima in Complex Loss Landscape



Solution 1
Use an optimizer with adaptive learning rate

Solution 2
Use a stochastic optimizer

Solution 3
Make the loss landscape smoother

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Batch Gradient Descent

- How to aggregate the gradients obtained from different training samples?
- Batch gradient descent computes the mean gradients **over the whole training set**

MSE loss

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_{k}^{N} \sum_{i}^{n} \left(\hat{y}_i^{(k)} - y_i^{(k)} \right)^2$$

Binary cross entropy

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{y}, y) = \sum_{k}^{N} -y \log \hat{y} + (1 - y) \ \log(1 - \hat{y})$$

Cross entropy

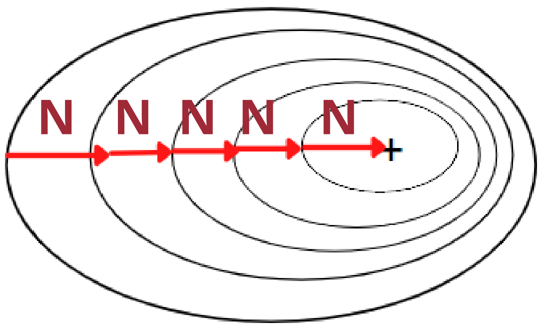
$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k}^{N} \sum_{i}^{n} y_i \log \hat{y}_i$$

Stochastic Gradient Descent (SGD)

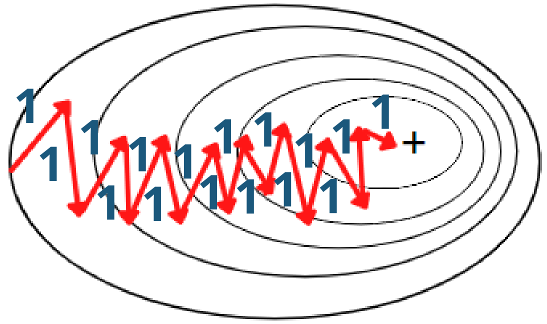
- **Intuition: Estimate** the gradient using **one random training sample**
- Benefits
  - **Speed up** the computation of the gradient **N computations → 1 computation**
  - Add some **randomness** to the gradient descent algorithm **Help escape spurious local minima**

Gradient descent

5N gradient computations



Stochastic gradient descent

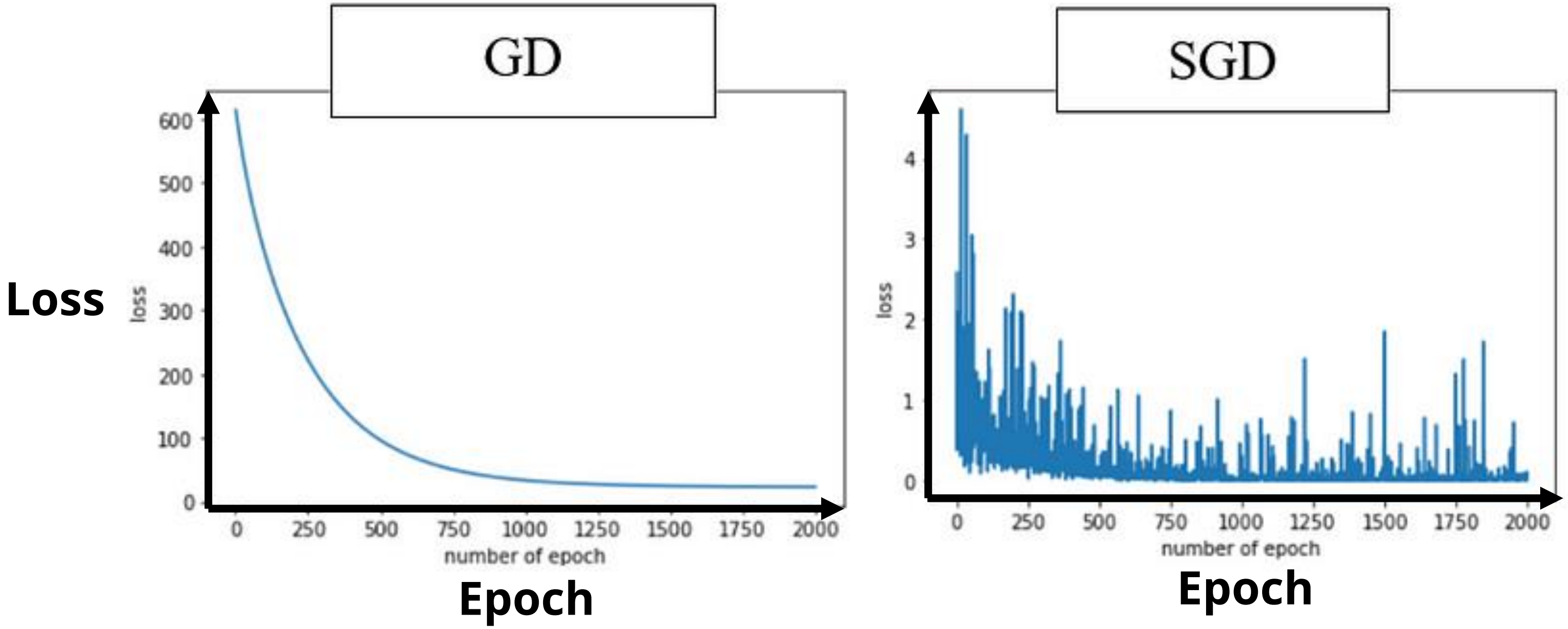


16 gradient computations

analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Stochastic Gradient Descent is Noisy and Unstable

- Gradient estimate using one single sample can be unreliable

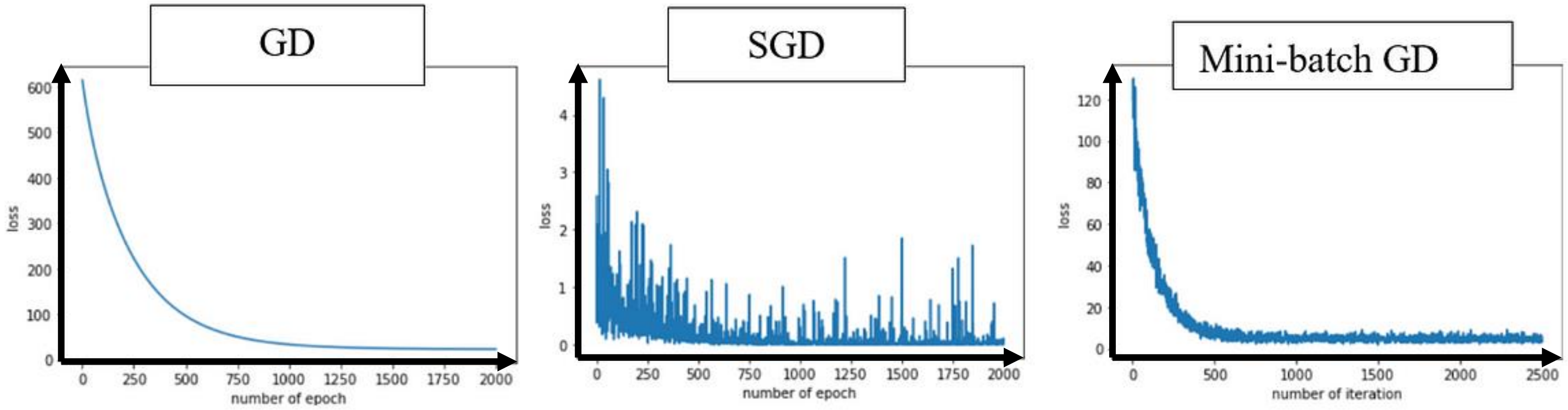


How about we use more samples to estimate the gradient?

Mini-batch Gradient Descent

- **Intuition:** **Estimate** the gradient using **several random training samples**

Loss



Epoch Epoch Epoch

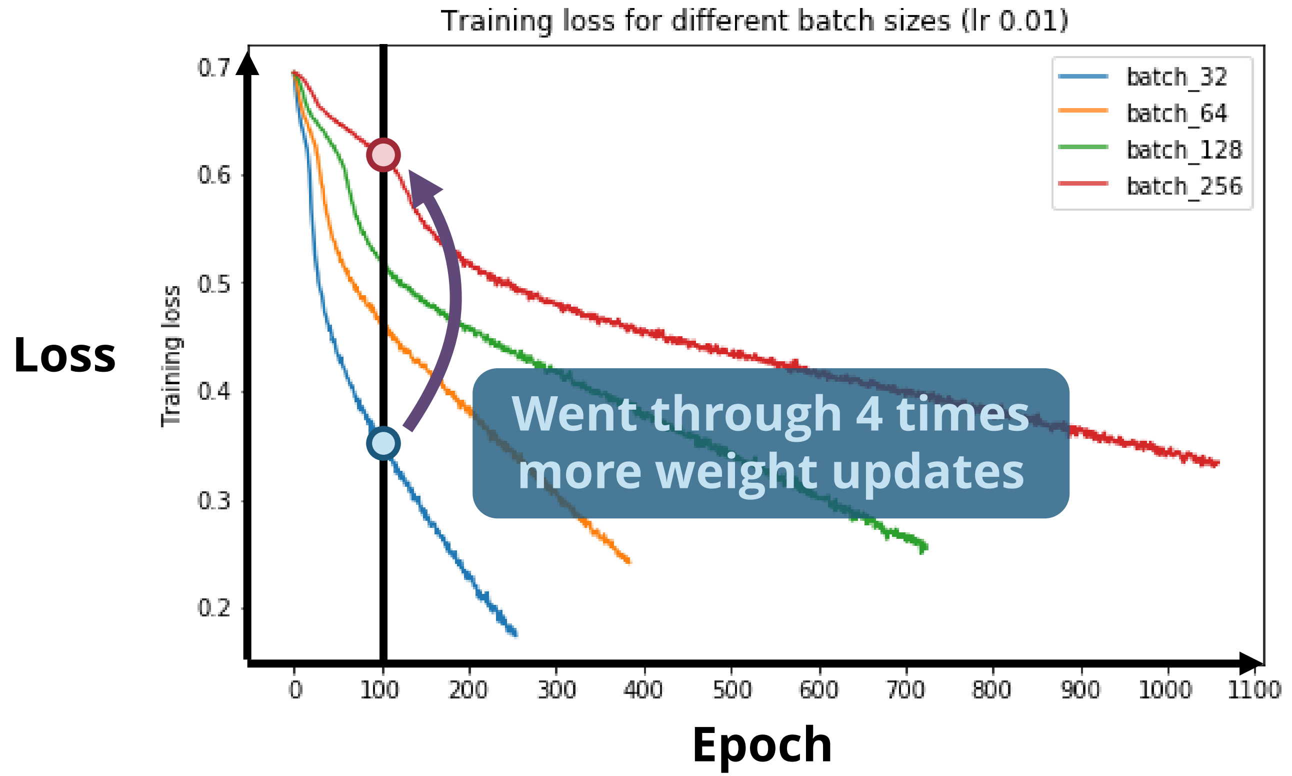
$\text{batch size} = N$ $\text{batch size} = 1$ $1 < \text{batch size} < N$

analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Effects of Batch Size

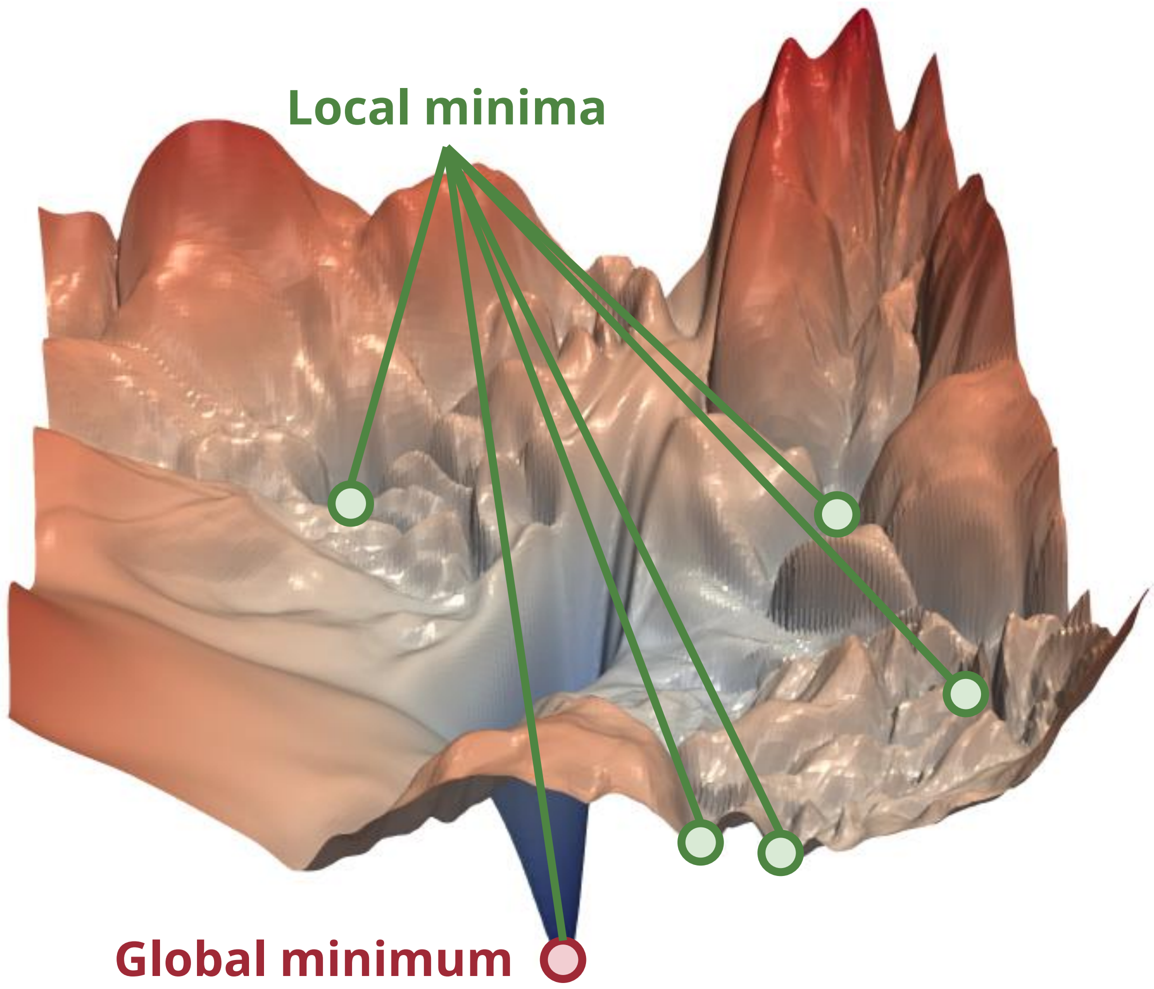
- An **epoch** is a full run of the whole dataset
- Steps per epoch depends on the batch size

$$\#(\text{steps}) = \frac{\#(\text{training samples})}{\text{batch size}}$$



medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57

Local Minima in Complex Loss Landscape



Solution 1
Use an optimizer with adaptive learning rate

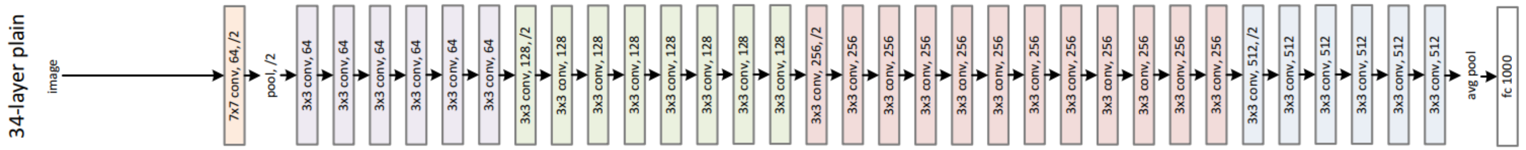
Solution 2
Use a stochastic optimizer

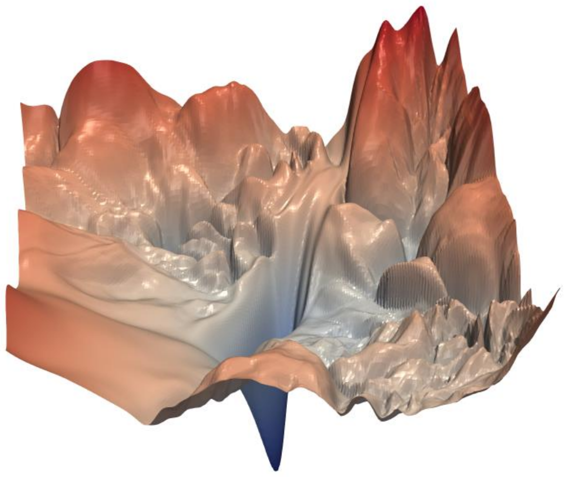
Solution 3
Make the loss landscape smoother

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

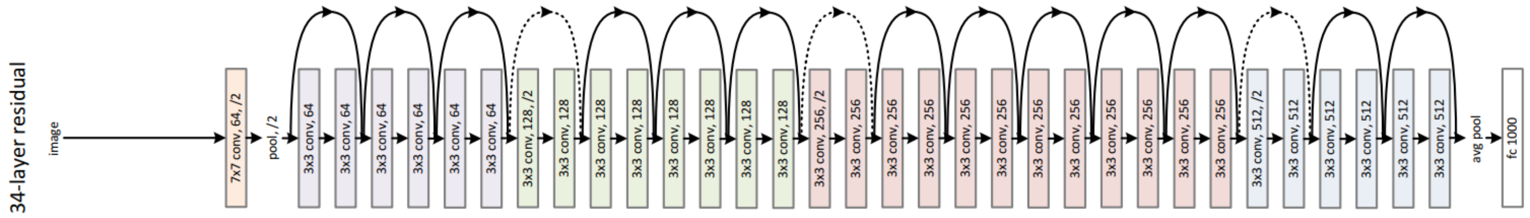
Skip Connections

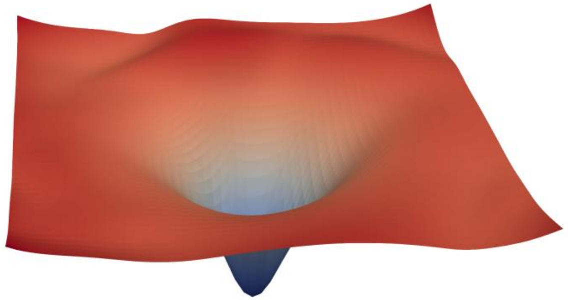
Without skip connections





With skip connections

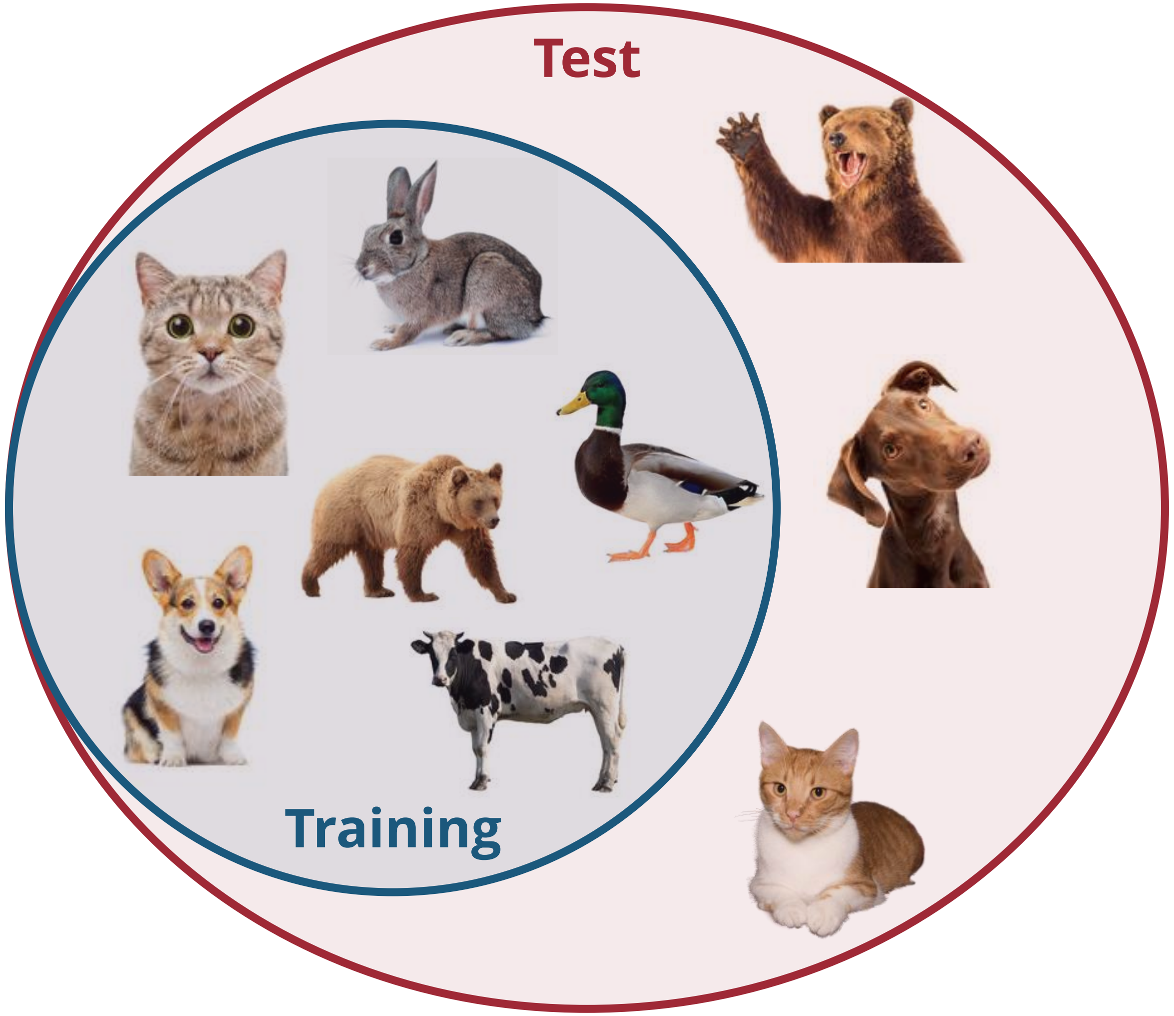




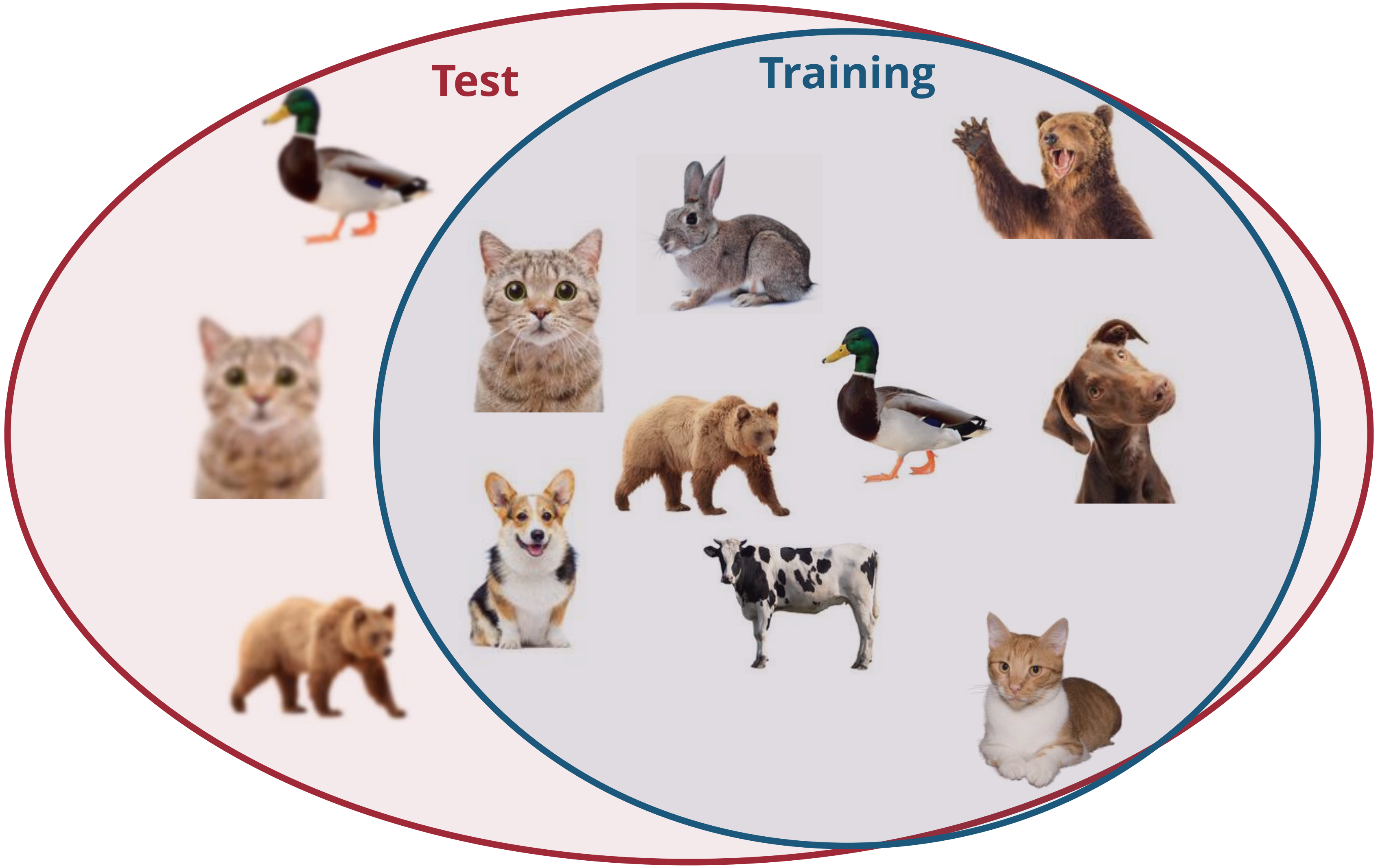
Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Training–Validation–Test

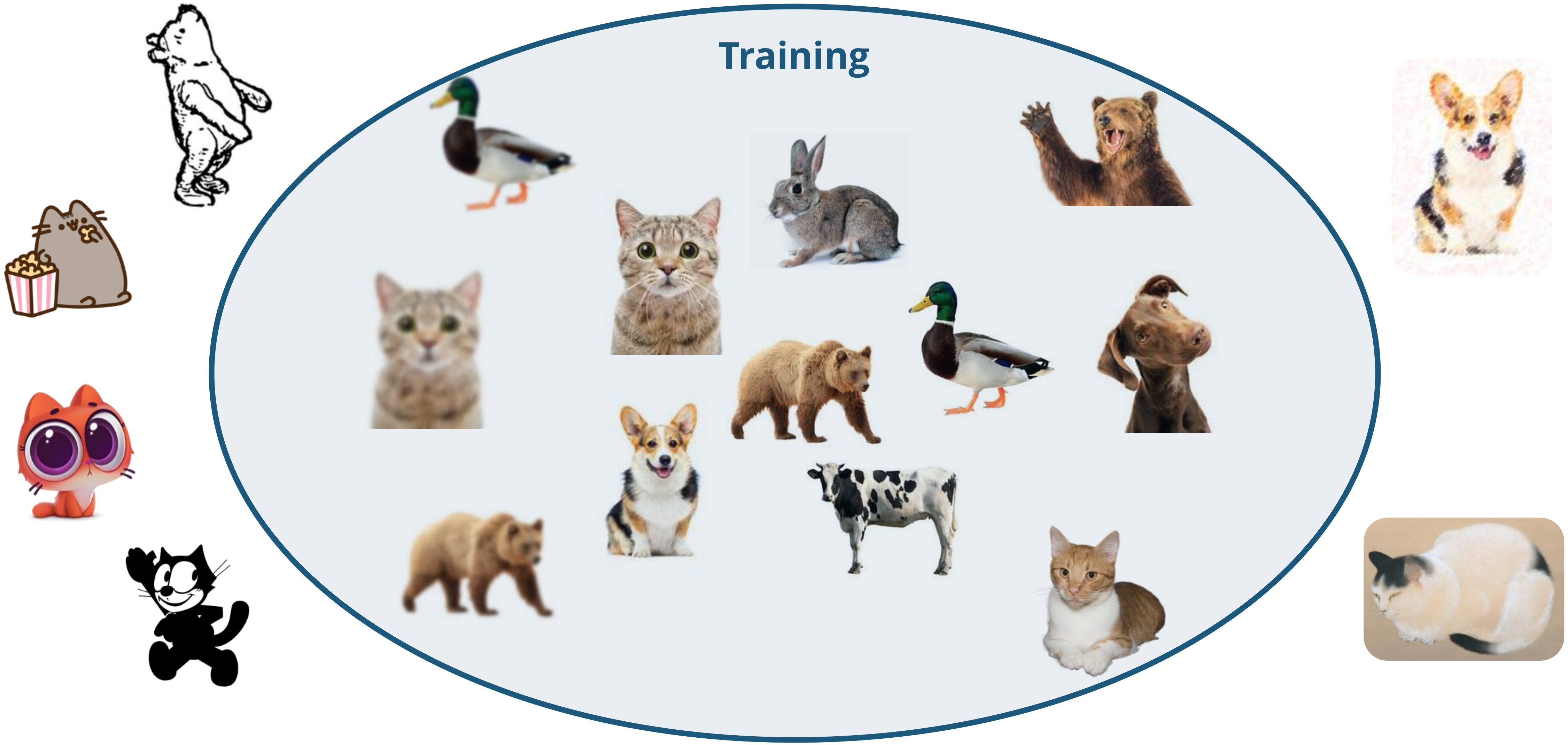
In-distribution vs Out-of-distribution



In-distribution vs Out-of-distribution



In-distribution vs Out-of-distribution



In-distribution vs Out-of-distribution

- **Key**: Make the training distribution closer to the target distribution
- First, we need to **define our target distribution**
- Then, we can try to
  - Collect a **diverse** dataset covering that covers different parts of the target distribution
  - Apply **data augmentation** to fill the gaps in the distribution

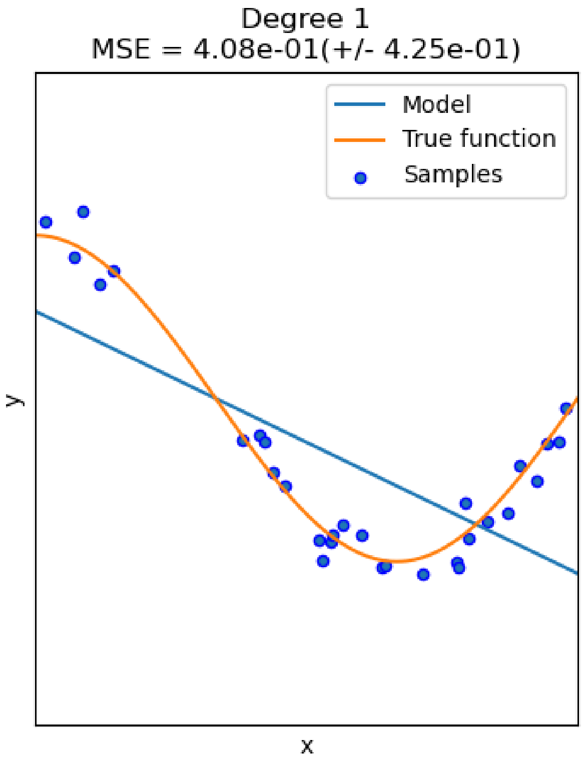
In-distribution vs Out-of-distribution

- What do we really want?
  - Good performance on the **training samples** **We already have their answers**
  - Good performance on **unseen samples in the target distribution** **Yep, we can do this!**
  - Good performance on **out-of-distribution samples** **Hopefully, but not guaranteed**

How to achieve good performance on
unseen samples in the target distribution

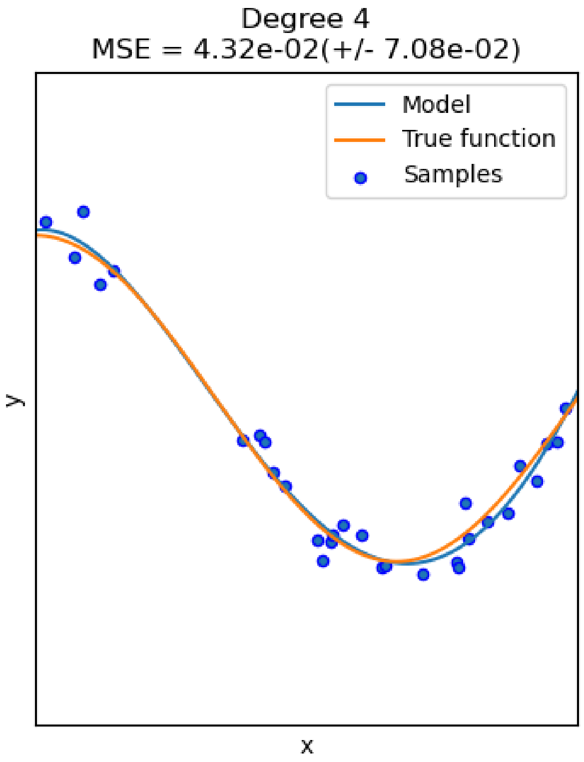
Overfitting & Underfitting

Underfitting

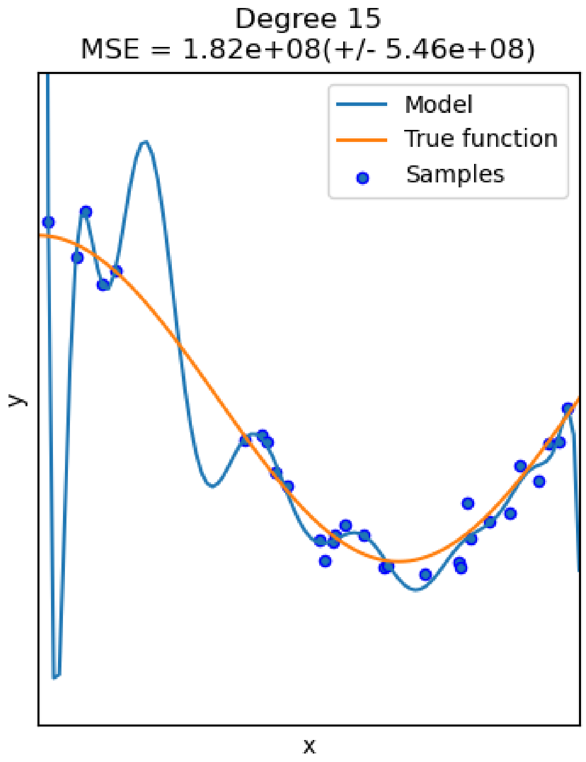


Model too inexpressive

Good fit!



Overfitting

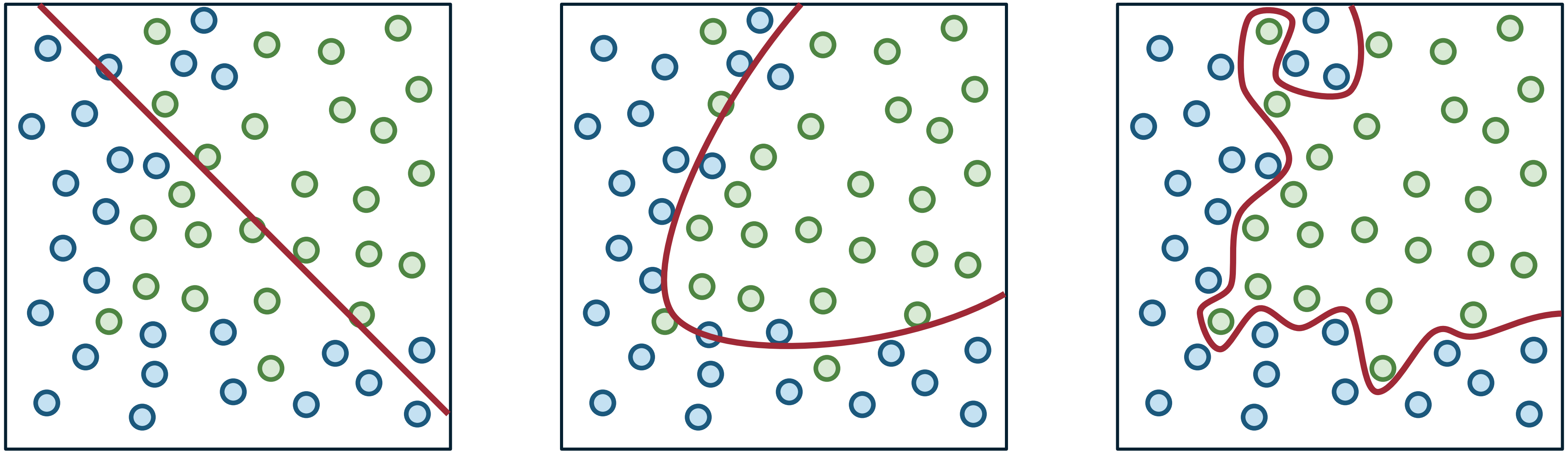


Model too expressive

scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Overfitting & Underfitting

Underfitting | **Good fit!** | **Overfitting**



Model too inexpressive

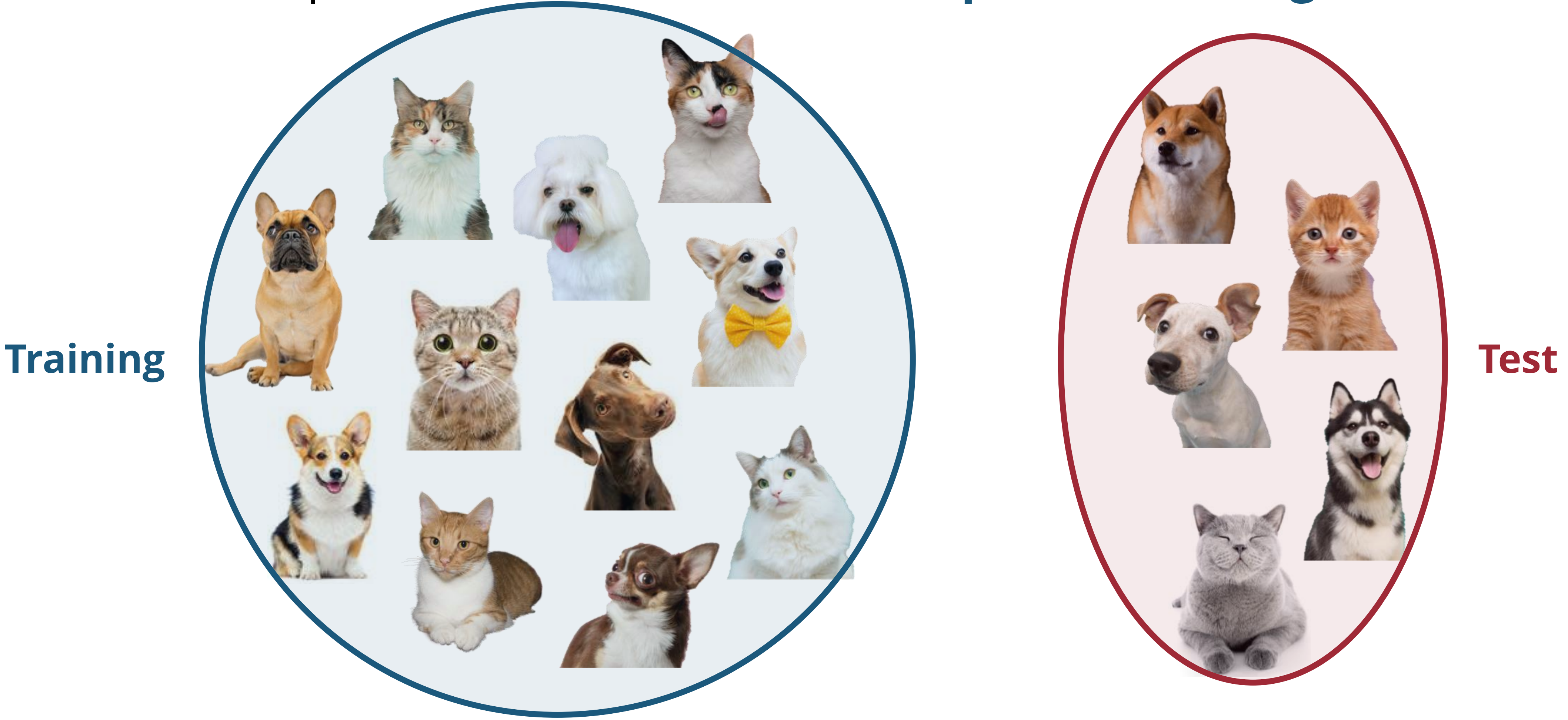
Model too expressive

Train–Test Split

- **Goal**: Good performance on **unseen samples in the target distribution**

Train–Test Split

- **Goal**: Good performance on **unseen samples in the target distribution**

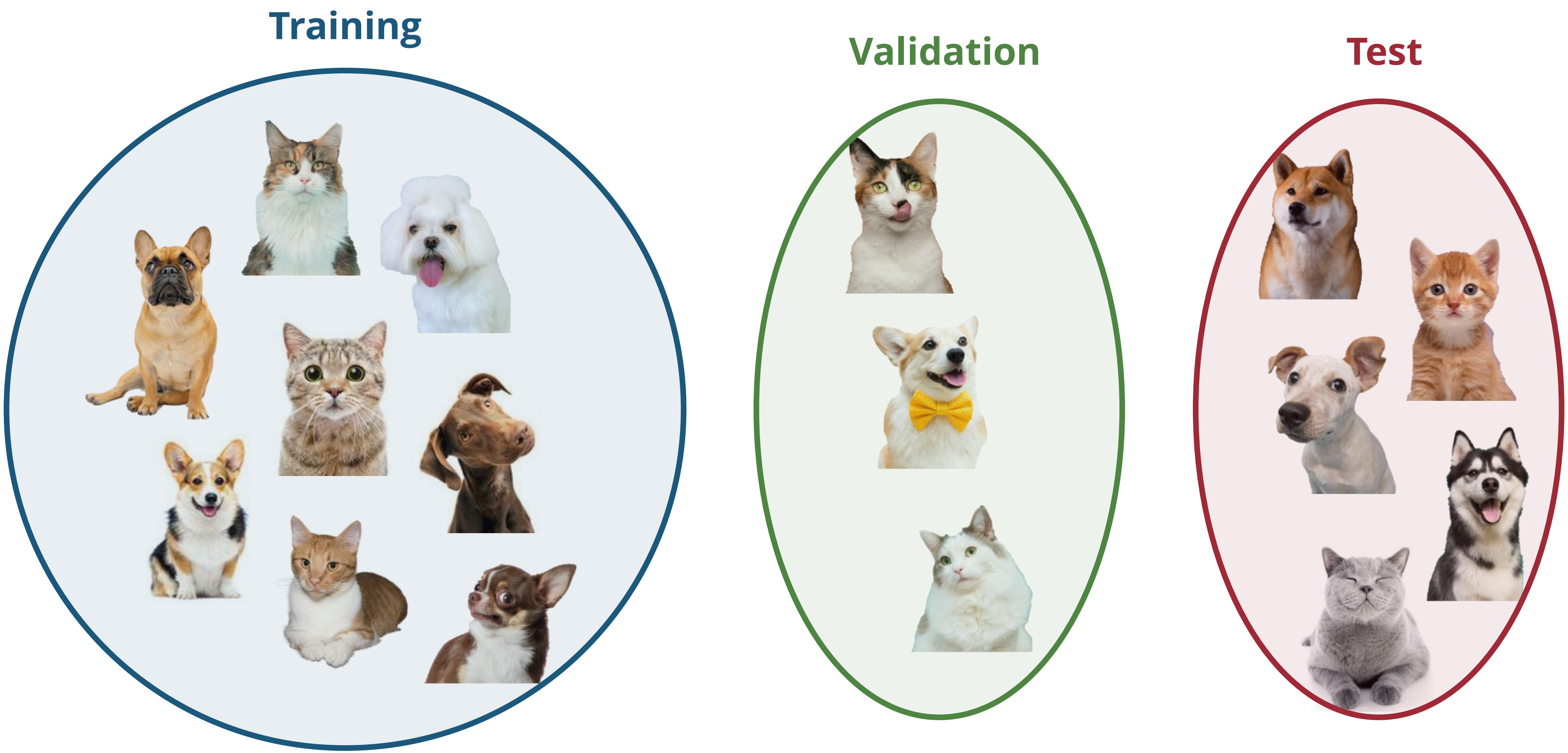


Test Set is an Estimation of the Test Distribution

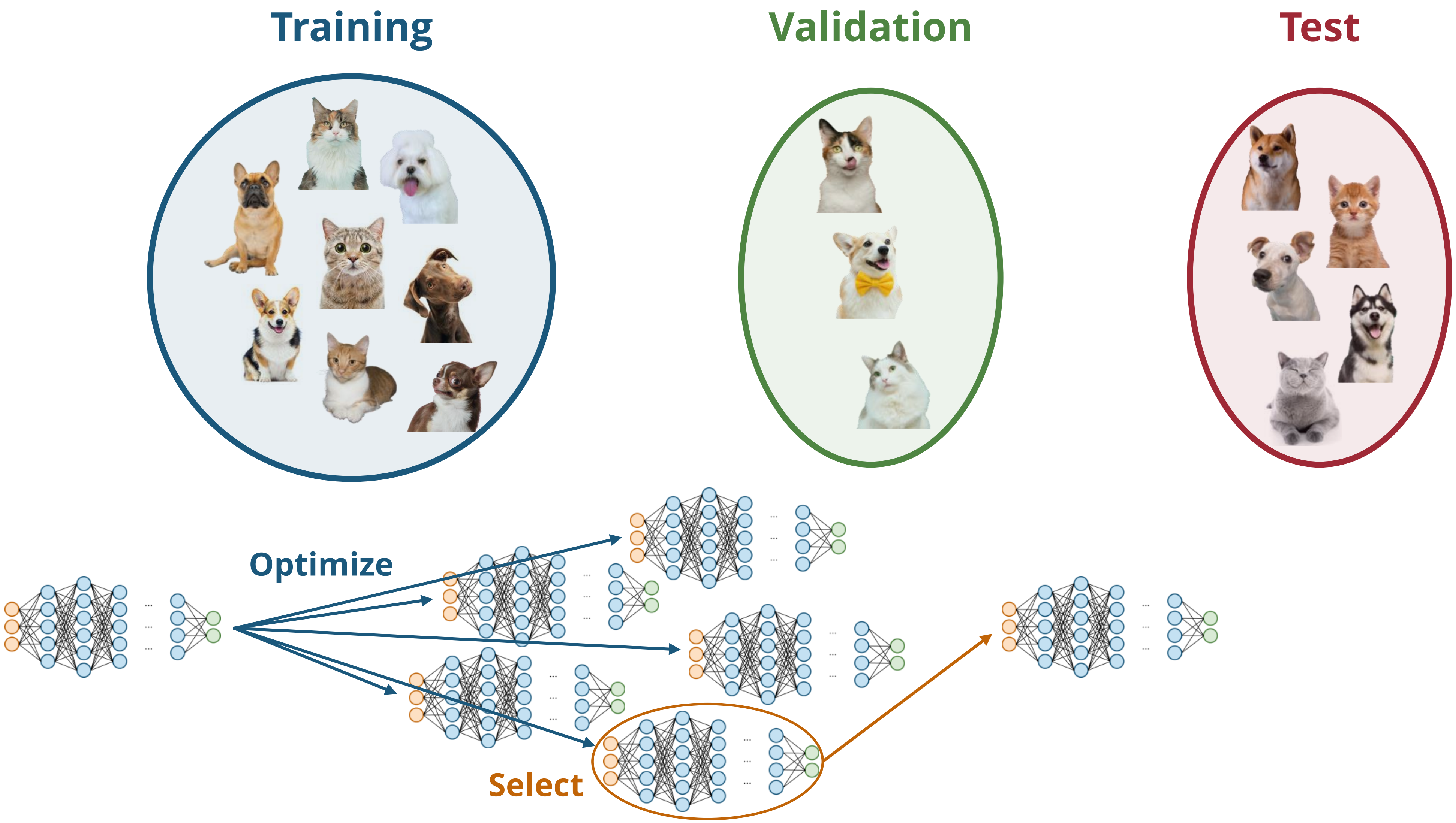
- We create a test set because we want to **estimate the performance when the model is applied to an interested distribution**

Train–Validation–Test Split

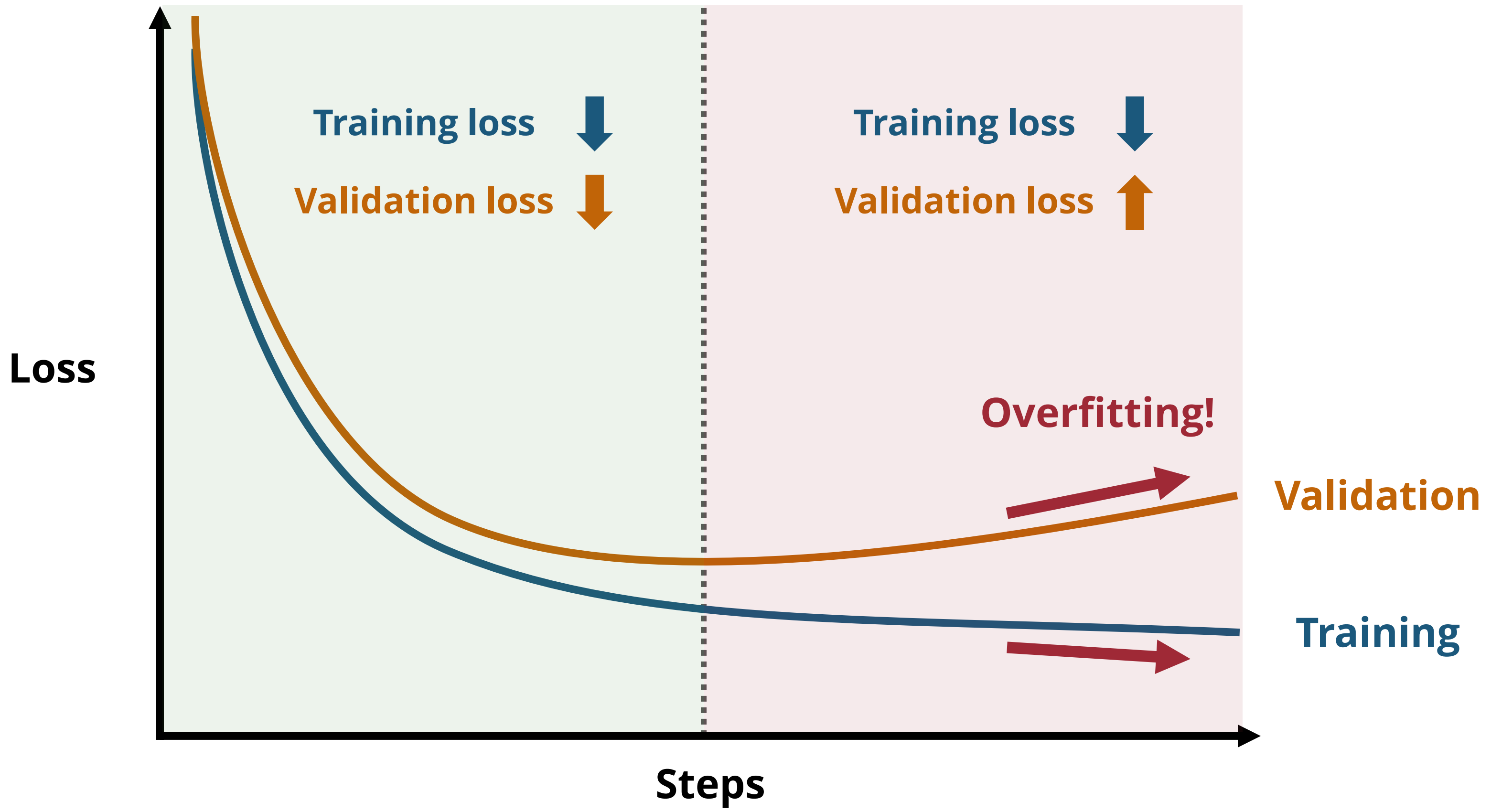
Train–Validation–Test Split



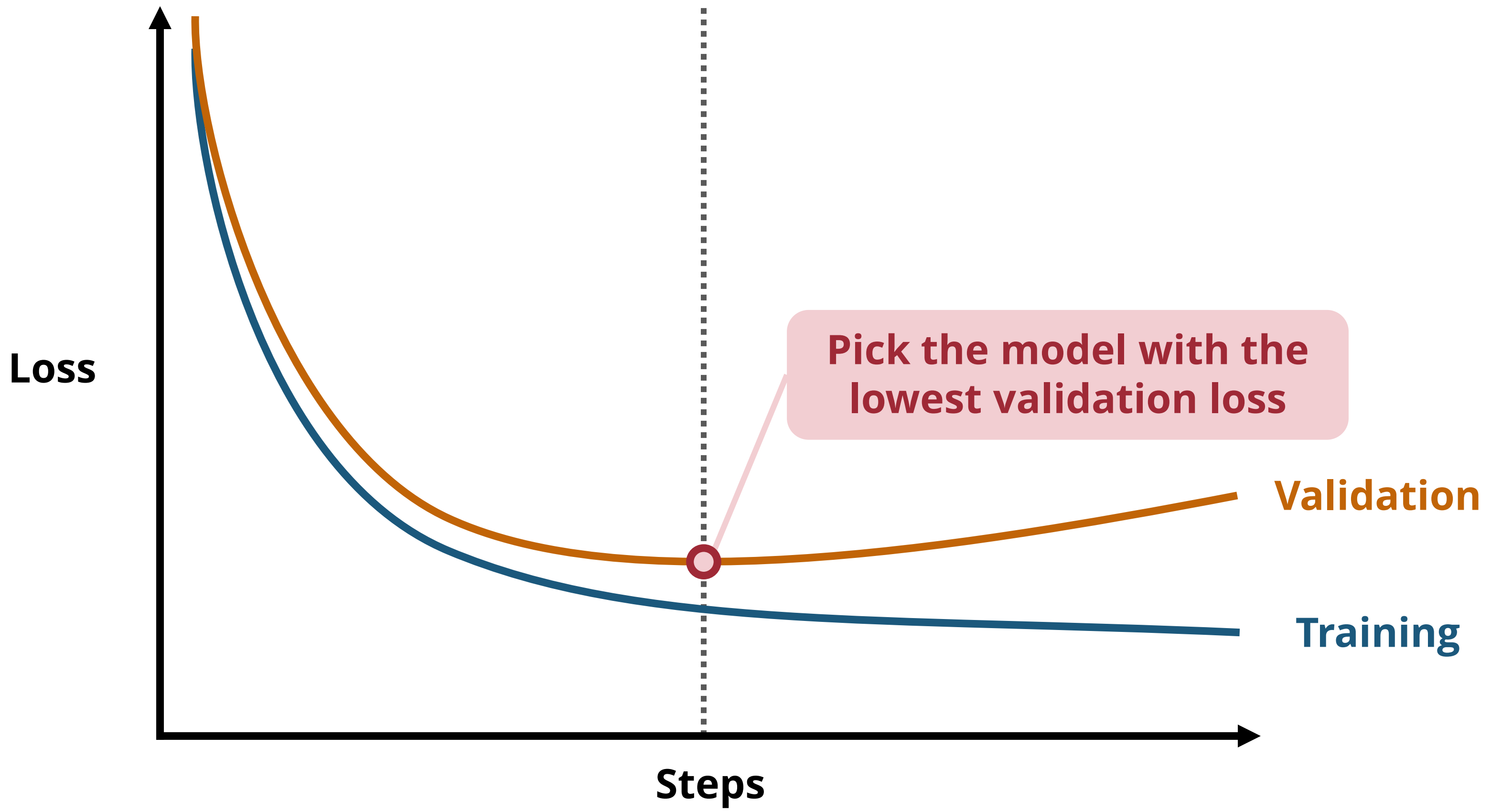
Training–Validation–Test Pipeline



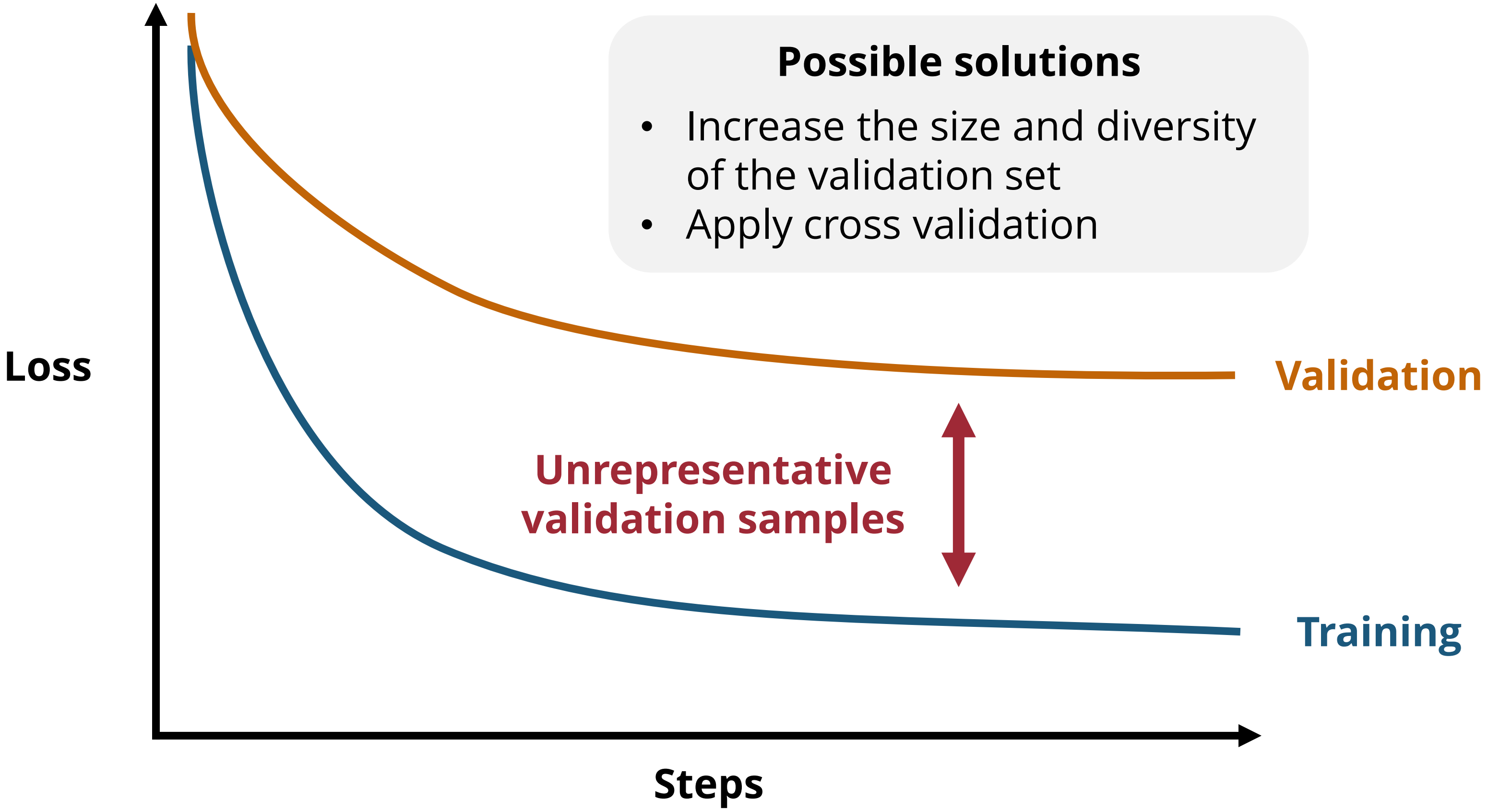
Training vs Validation Losses



Training vs Validation Losses



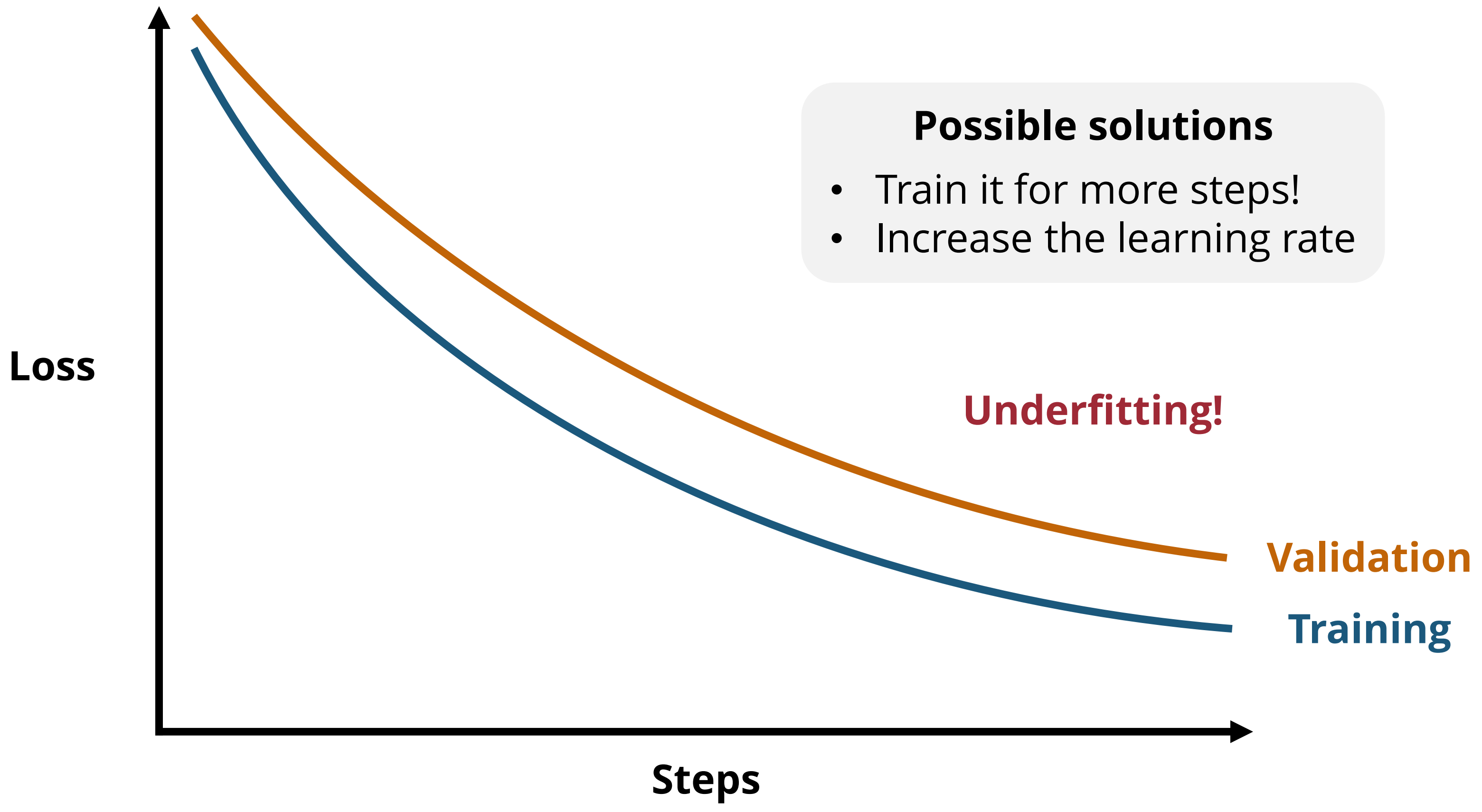
Training vs Validation Losses



Possible solutions

- Increase the size and diversity of the validation set
- Apply cross validation

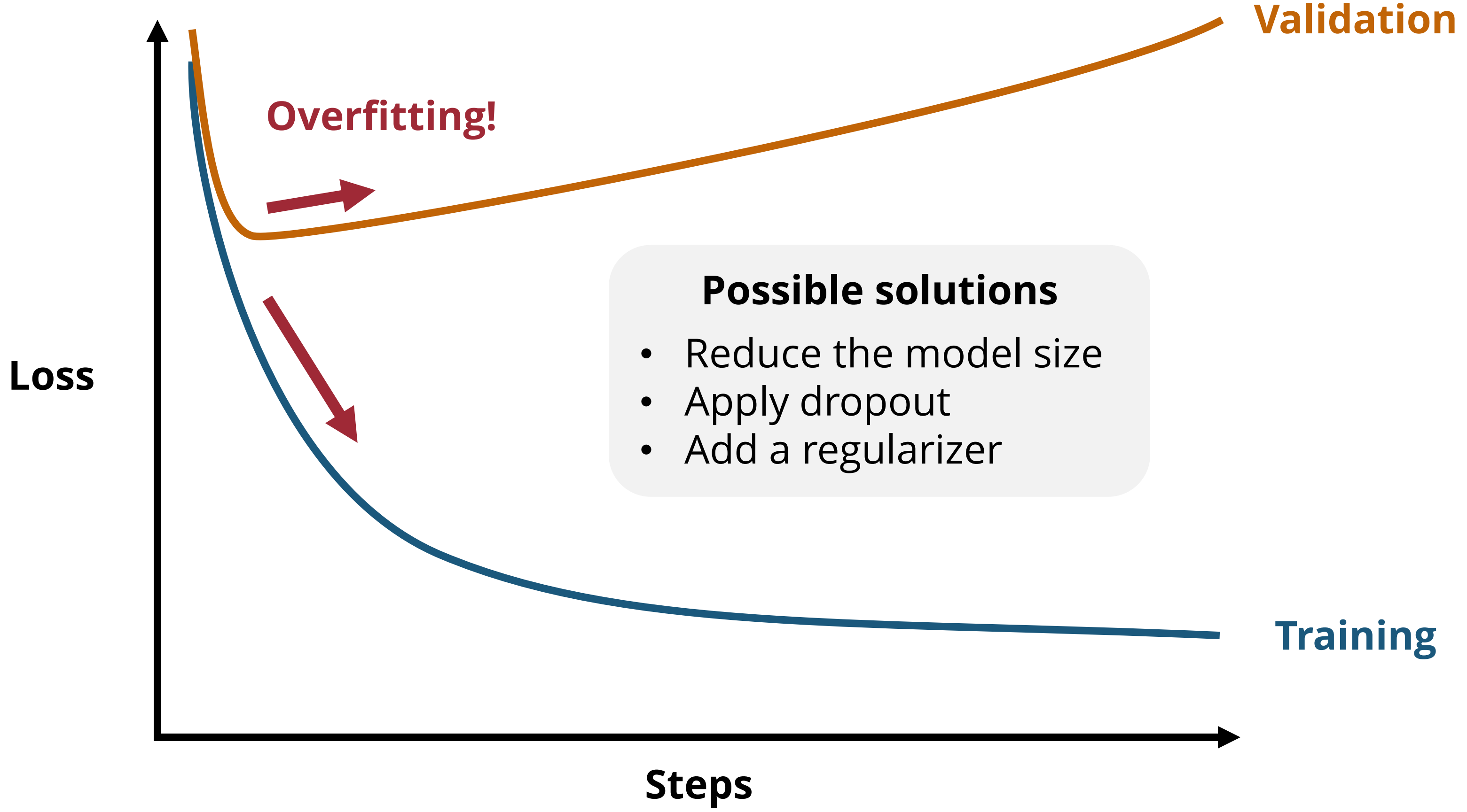
Training vs Validation Losses



Possible solutions

- Train it for more steps!
- Increase the learning rate

Training vs Validation Losses



Possible solutions

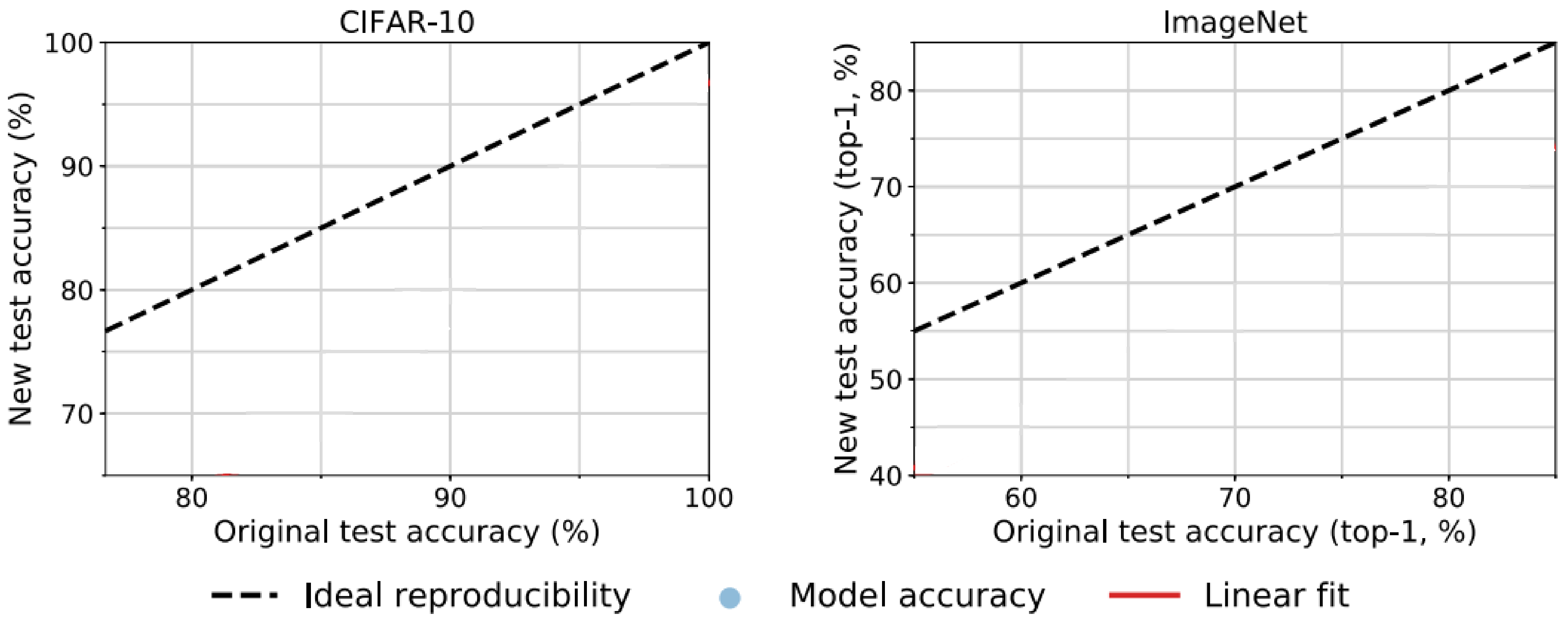
- Reduce the model size
- Apply dropout
- Add a regularizer

Train–Validation–Test Split

- **Keys**
  - **Never train or select your model on test samples!**
  - Don't over-select your model on the validation set

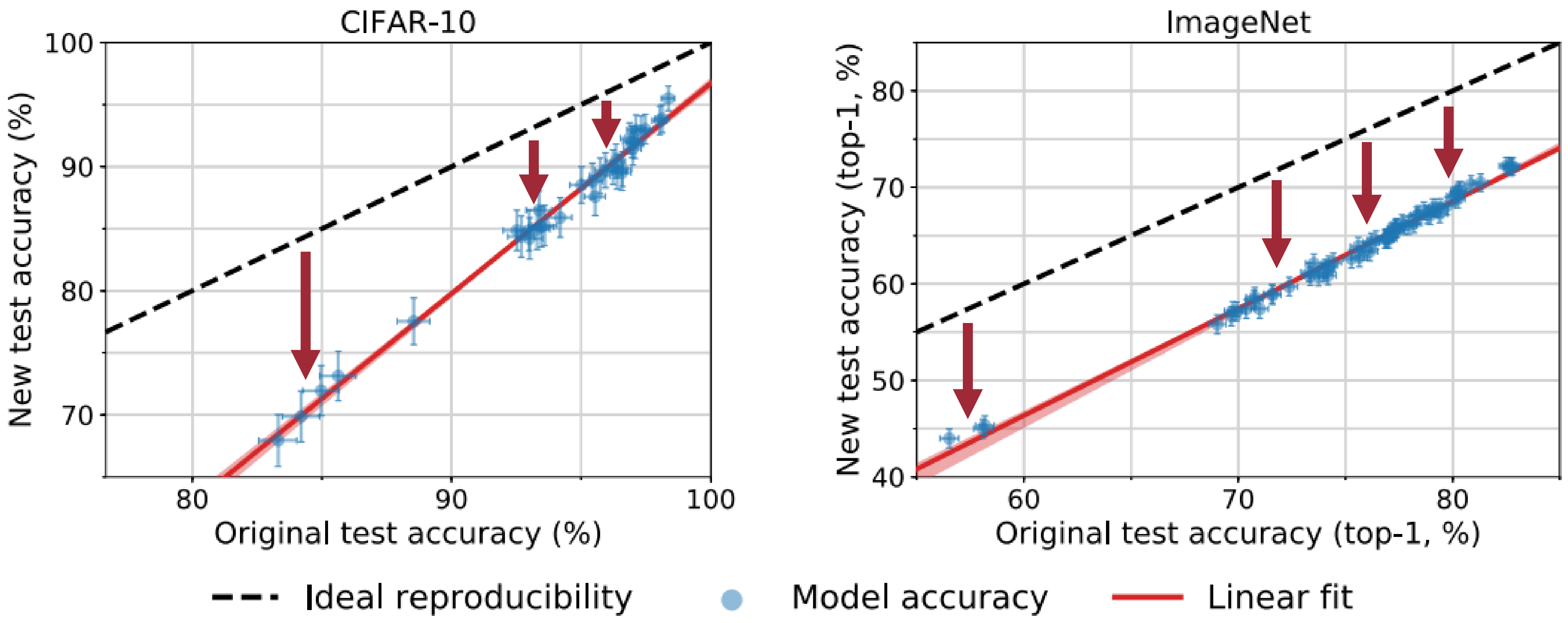
- What's the **best ratio**?
  - Most common: **8:1:1** or 9:0.5:0.5
  - For smaller dataset, you might even want 6:2:2

Validation Set



Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar, "Do ImageNet Classifiers Generalize to ImageNet?," *ICML*, 2019.

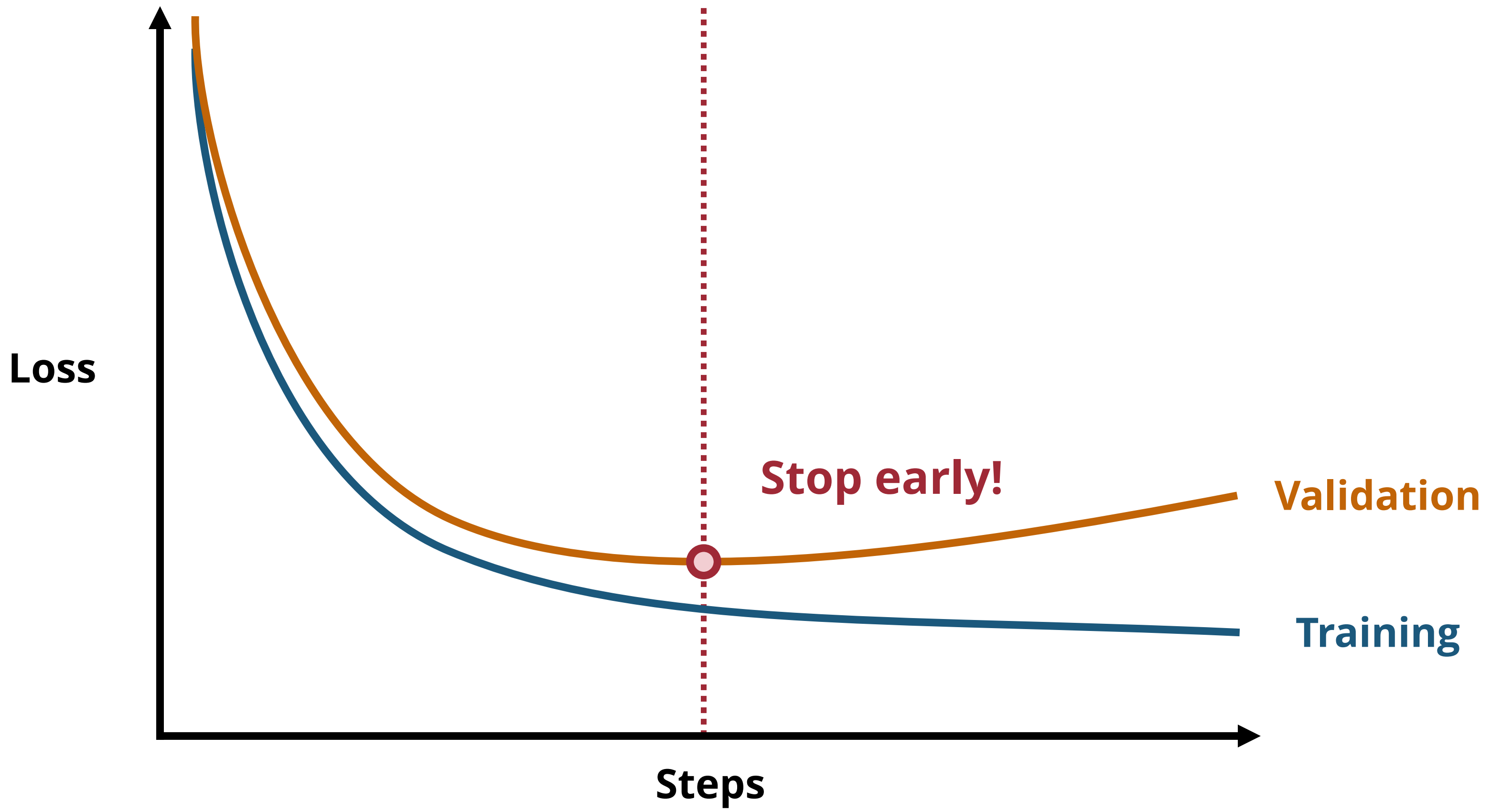
Validation Set



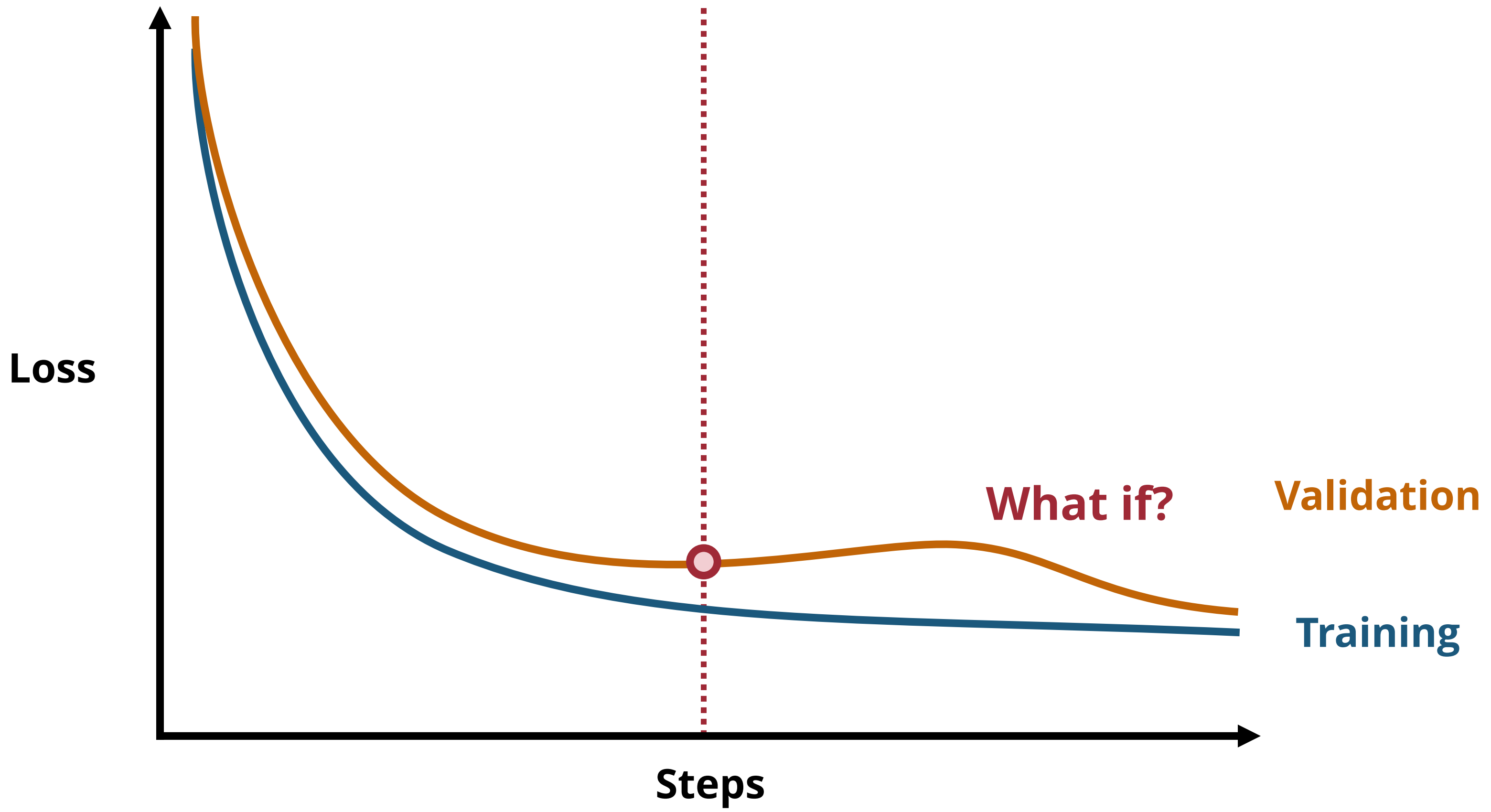
Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar, "Do ImageNet Classifiers Generalize to ImageNet?," *ICML*, 2019.

Overcoming Overfitting

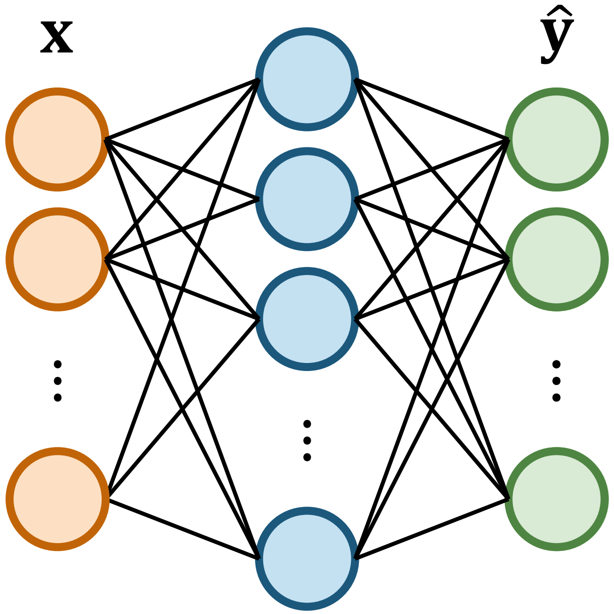
Early Stopping



Early Stopping

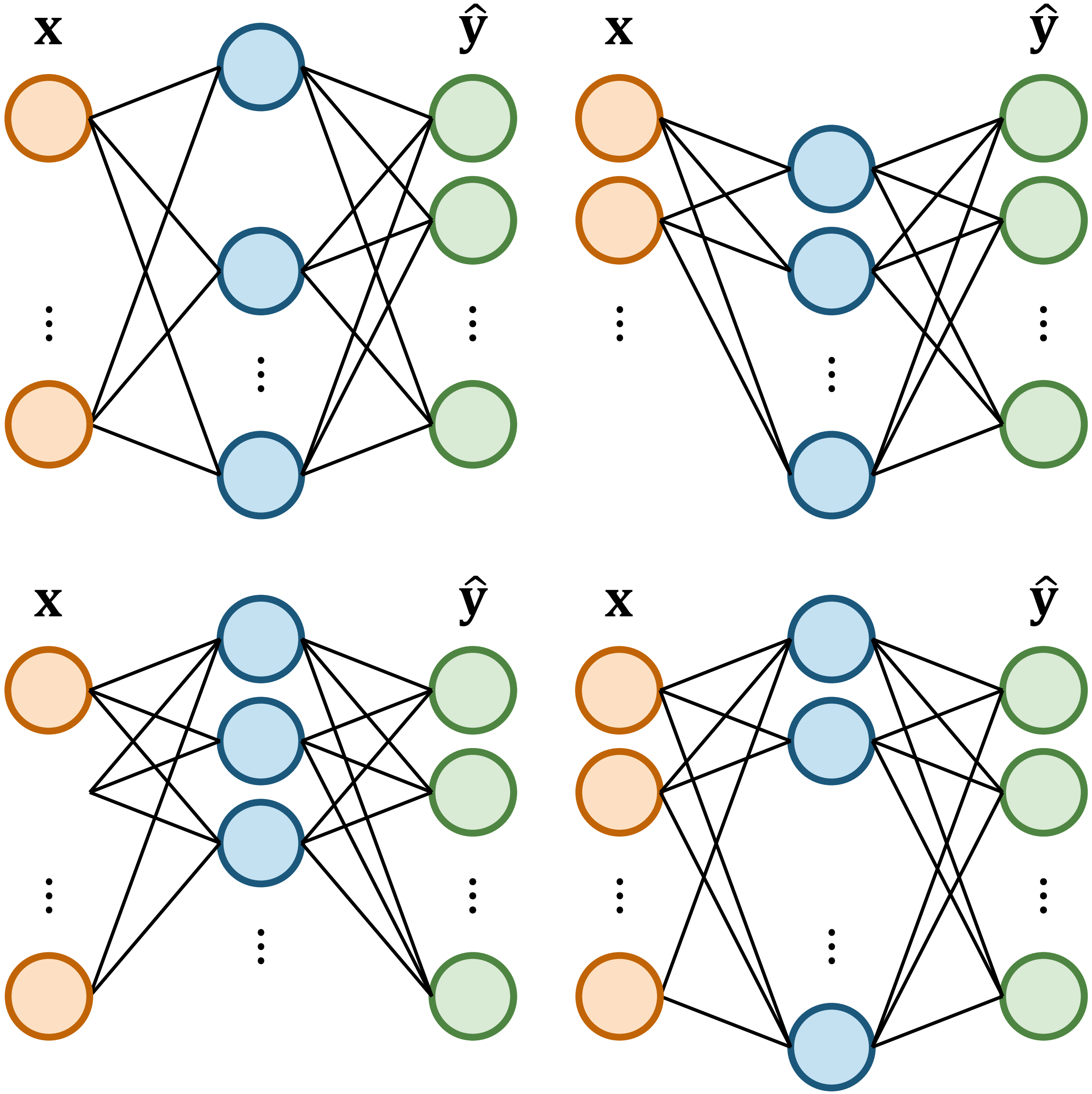


Dropout

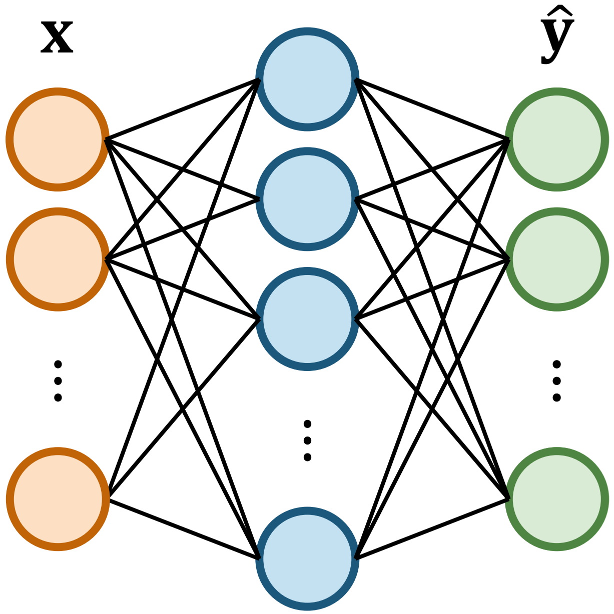


Each neuron may be removed with probability p during training

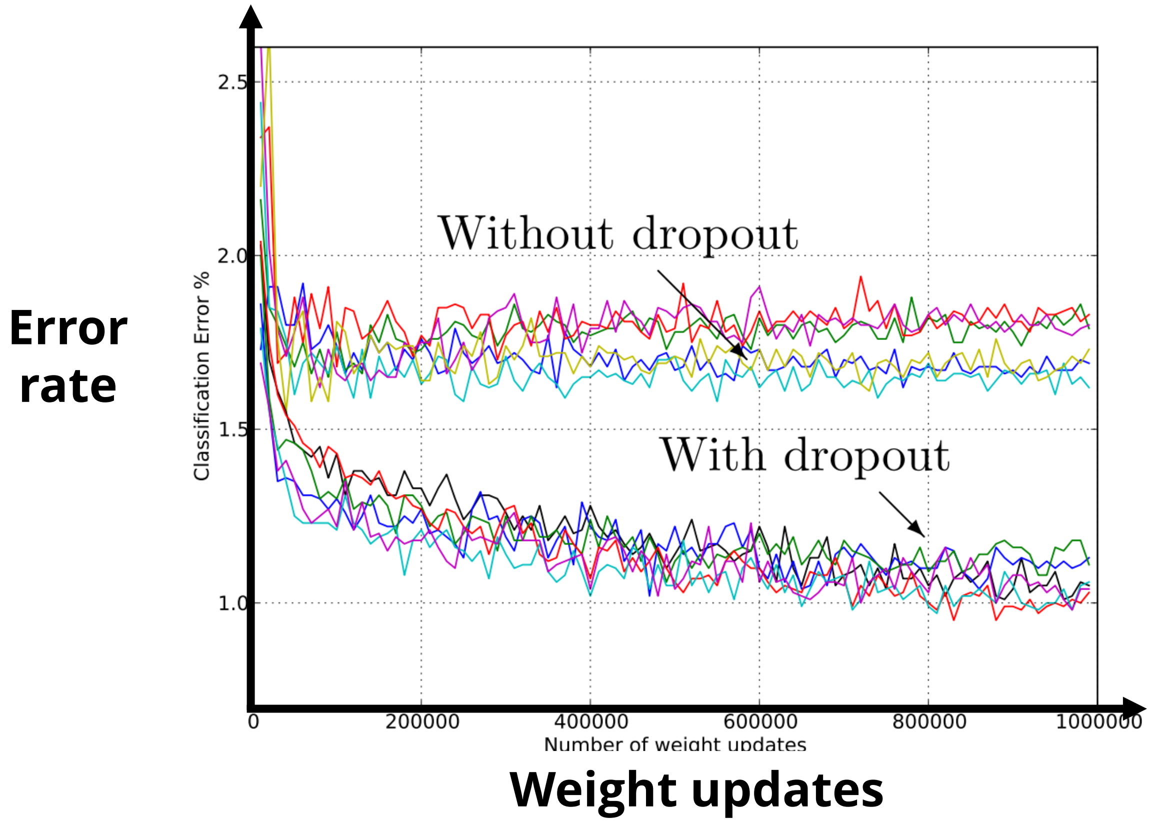
Dropout rate



Dropout



Each neuron may be removed with probability p during training



Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *JMLR*, 2014.

Regularization Term

- A regularization term can help alleviate overfitting
  - **L1 regularization** (LASSO)

$$L' = L + \lambda(|w_1| + |w_2| + \cdots + |w_K|)$$

  - **L2 regularization** (ridge regression)

$$L' = L + \lambda\left(w_1^2 + w_2^2 + \cdots + w_K^2\right)$$

Both L1 and L2 regularization encourage smaller weights