PAT 498/598 (Winter 2025)

Music & AI

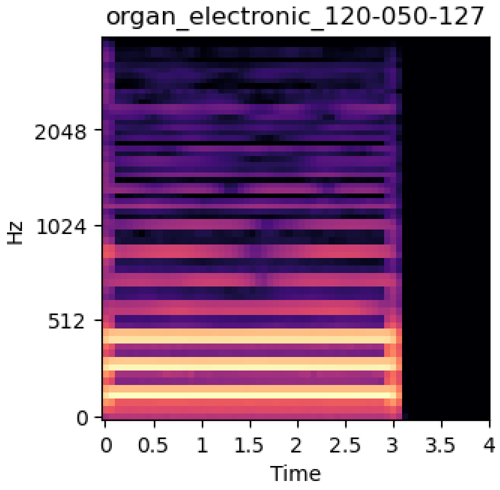
Lecture 9: Deep Learning Fundamentals III

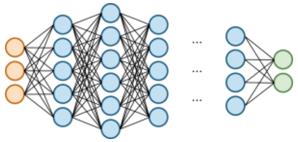
Instructor: Hao-Wen Dong

SCHOOL OF MUSIC, THEATRE & DANCE
PERFORMING ARTS TECHNOLOGY
UNIVERSITY OF MICHIGAN

Homework 3: Musical Note Classification using CNNs

- Train a CNN that can classify audio files into their **instrument families**
  - **Input**: 64x64 mel spectrogram
  - **Output**: 11 instrument classes
  - Using the **NSynth** dataset (Engel et al., 2017)



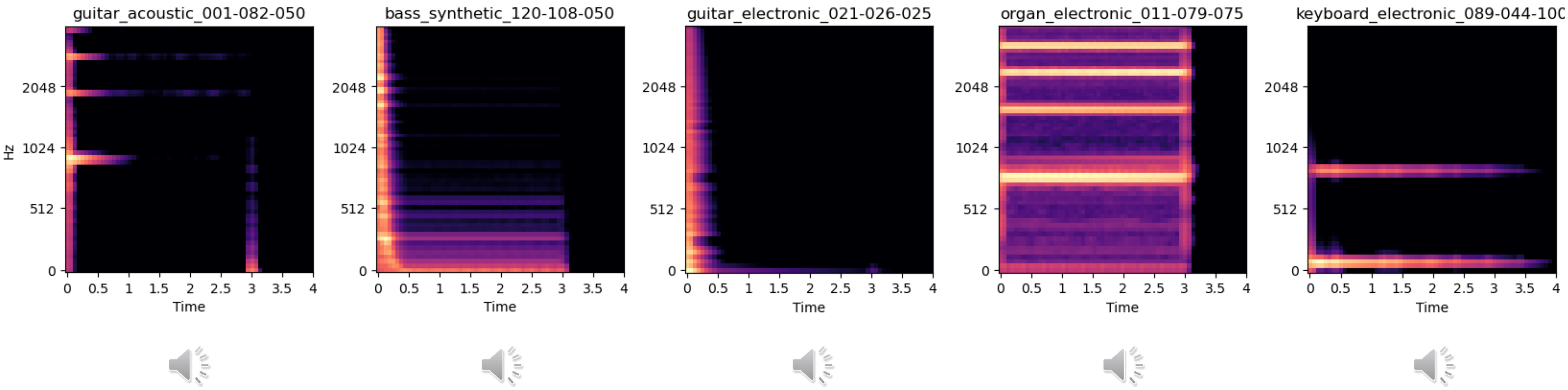


Bass
Brass
Flute
Guitar
Keyboard
Mallet
Organ
Reed
String
Synth Lead
Vocal

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi, "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders," *ICML*, 2017.

NSynth Dataset

- A collection of 305,979 **single-shot musical notes** (Engel et al., 2017)
  - Produced from 1,006 **commercial sample libraries**
  - With different **MIDI pitches** (21–108) and **velocities** (25, 50, 75, 100, 127)



Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi, "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders," *ICML*, 2017.

Homework 3: Musical Note Classification using CNNs

- Instructions will be released on Gradescope
- Due at **11:59pm ET** on **February 17**
- Late submissions: **1 point deducted per day**

(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)

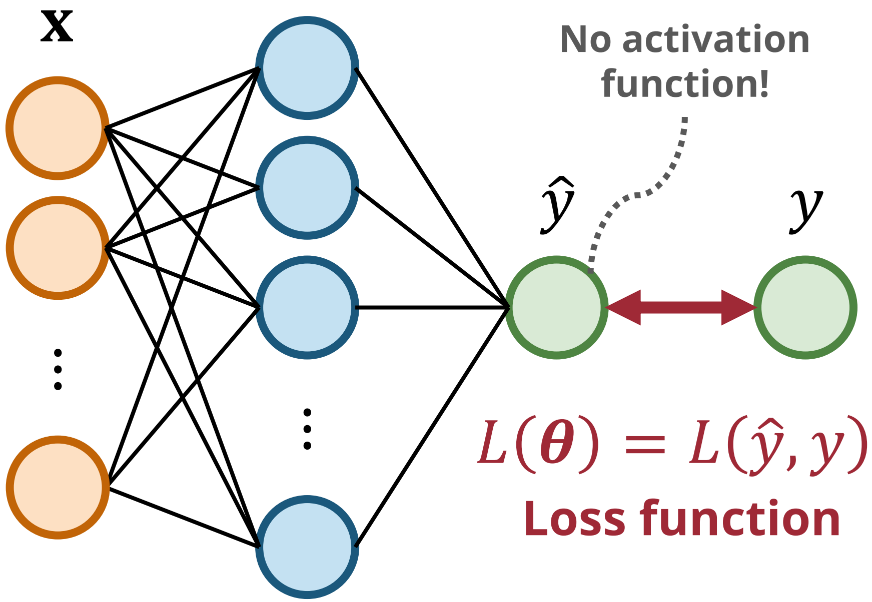
$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

↓

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

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

(Recap) Common Loss Functions for Regression



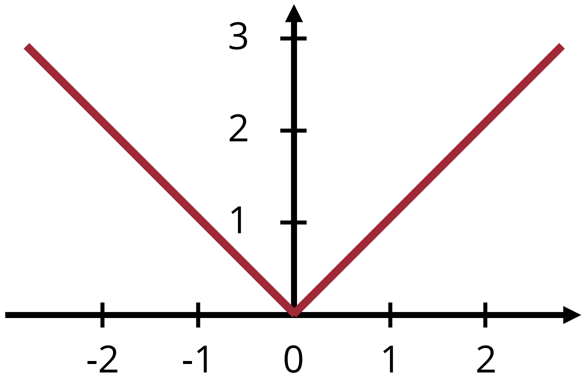
$$L(\boldsymbol{\theta}) = L(\hat{y}, y)$$

Loss function

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

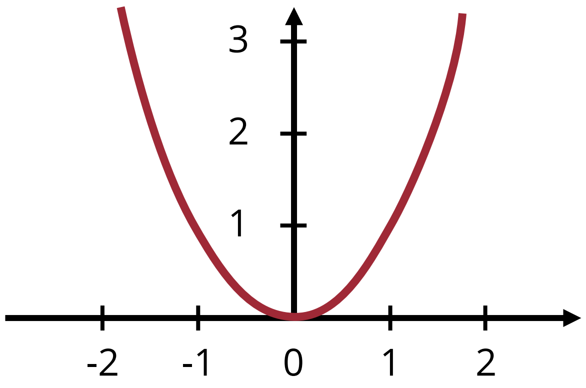
L1 loss

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



L2 loss

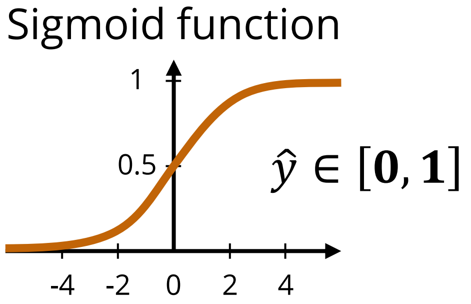
$$L(\hat{y}, y) = (\hat{y} - y)^2$$

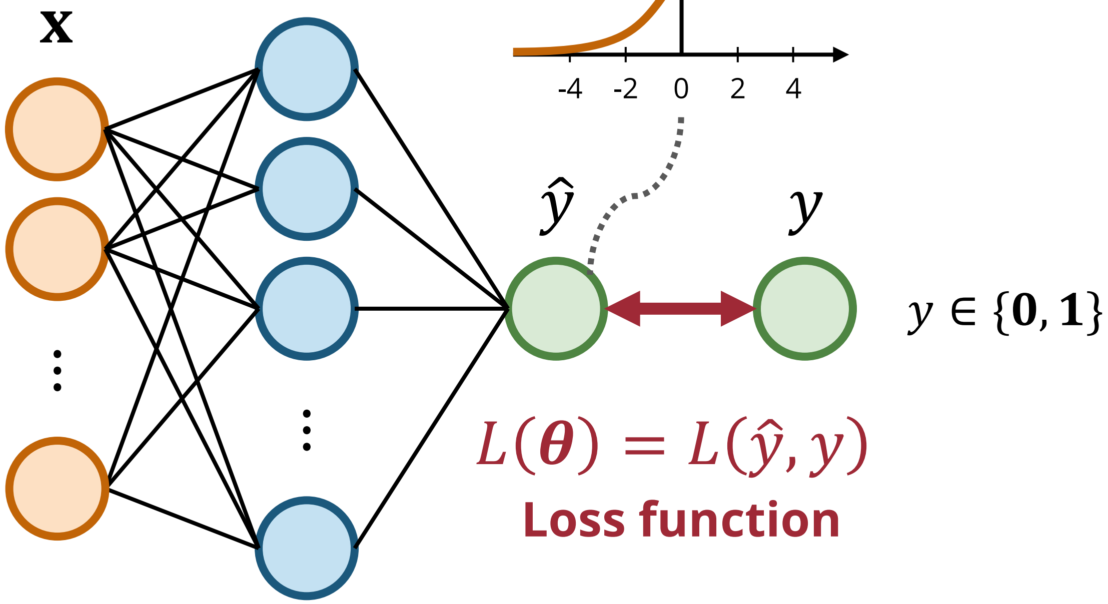


(Recap) Binary Cross Entropy for Binary Classification

- **Logistic regression** approaches classification like regression

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



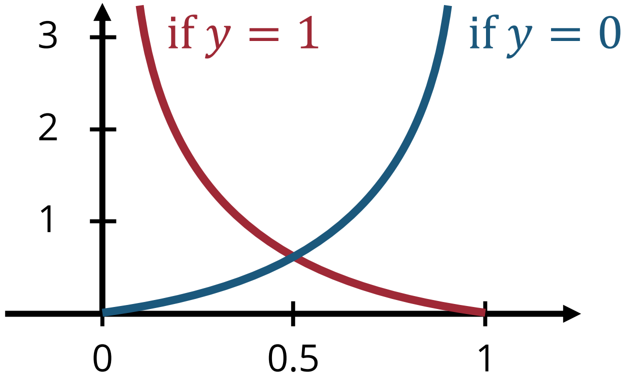


Binary cross entropy

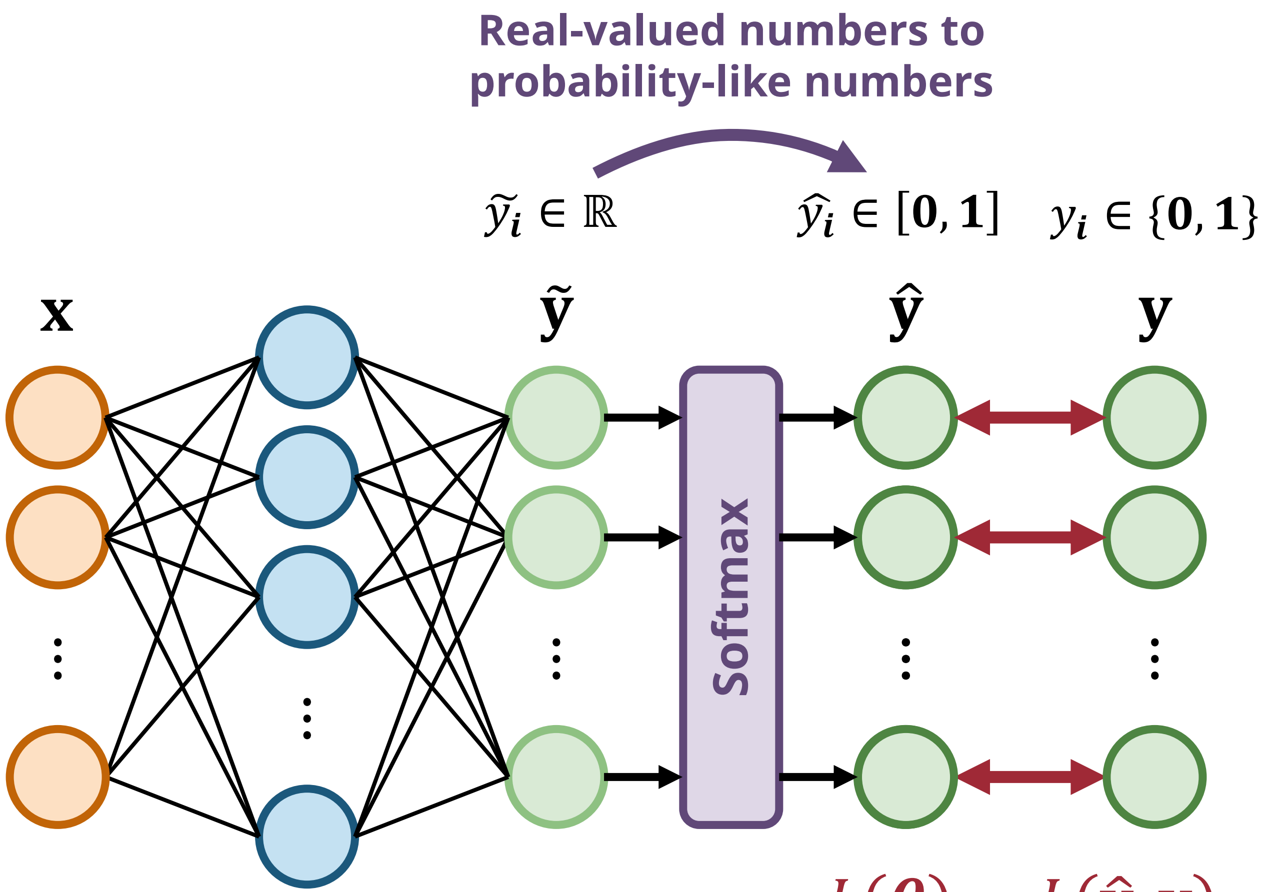
(Also called log loss)

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

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



(Recap) Cross Entropy for Multiclass Classification



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

Loss function

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!

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

Cross Entropy

Only one of them will be one!

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

$$= -\sum_{i}^{n} y_i \log \hat{y}_i$$

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)

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

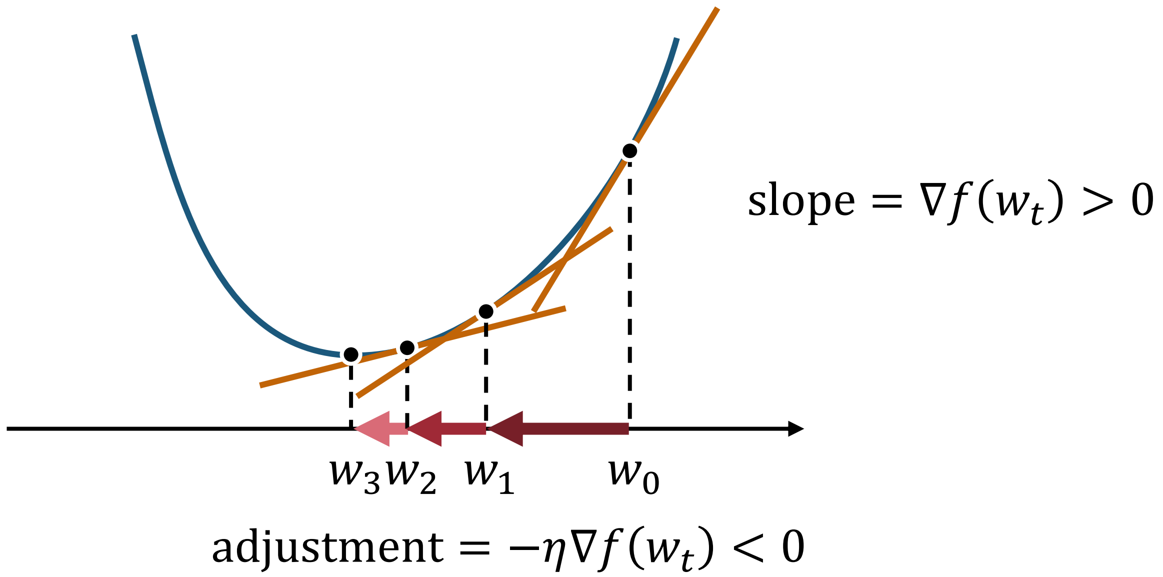
↓

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

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

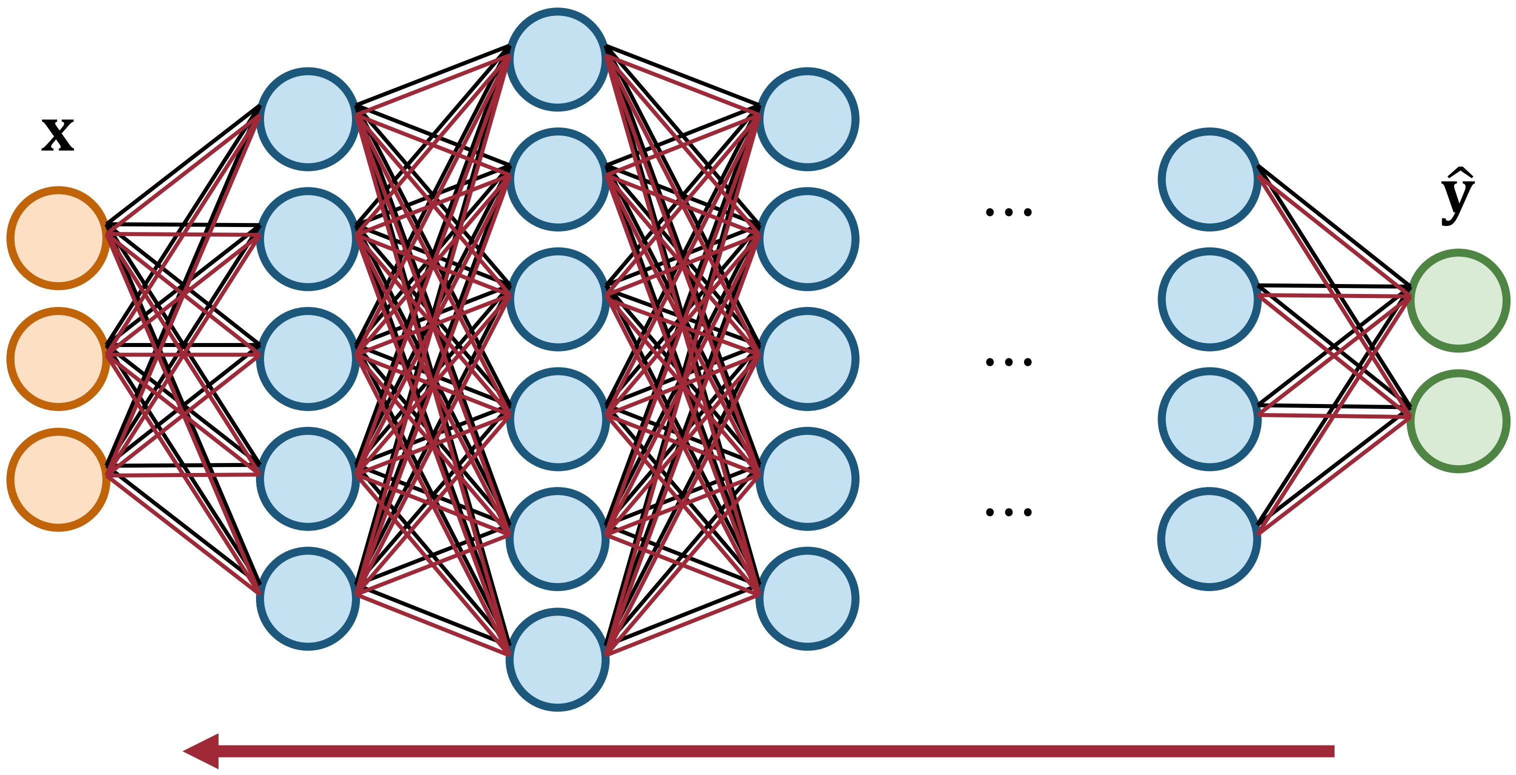
(Recap) Gradient Descent – Pseudocode

- 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

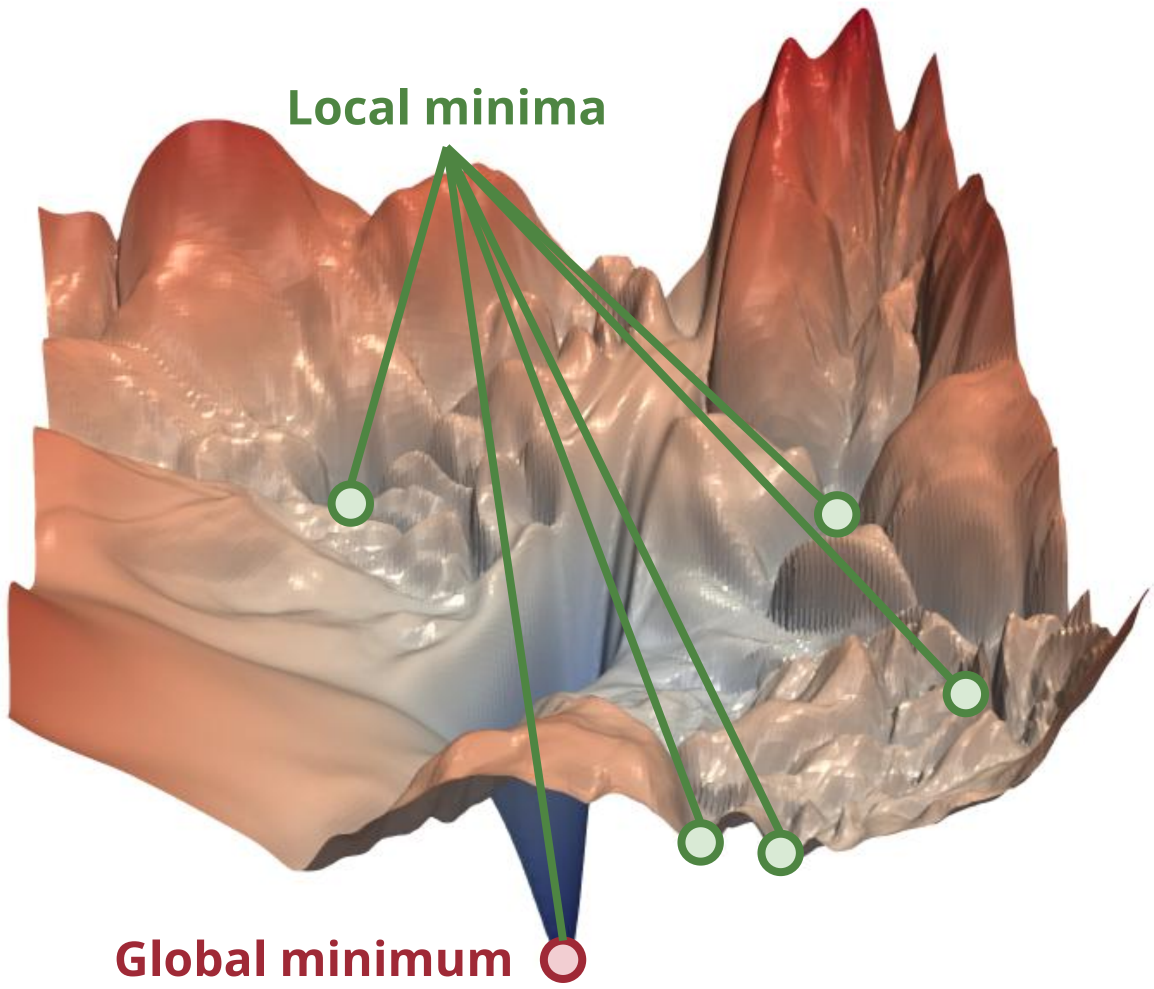
$\frac{\partial L}{\partial \mathbf{x}}$ $\frac{\partial L}{\partial \mathbf{h}_1}$ $\frac{\partial L}{\partial \mathbf{h}_2}$ $\frac{\partial L}{\partial \mathbf{h}_3}$ $\frac{\partial L}{\partial \mathbf{h}_{L-1}}$



Backward pass

loss.backward()

Local Minima in Complex Loss Landscape



Solution 1
Use an optimizer with adaptive learning rate

Solution 2
Use a stochastic optimizer

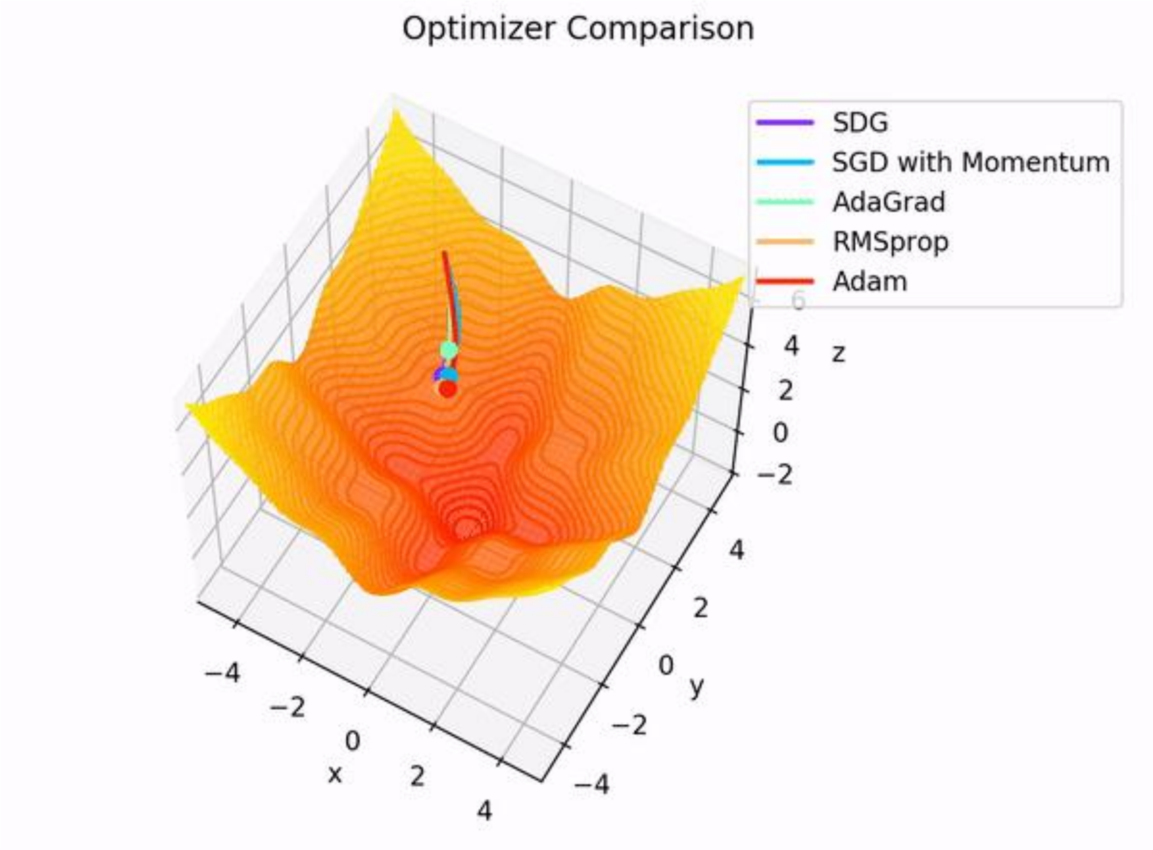
Solution 3
Make the loss landscape smoother

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

(Recap) 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

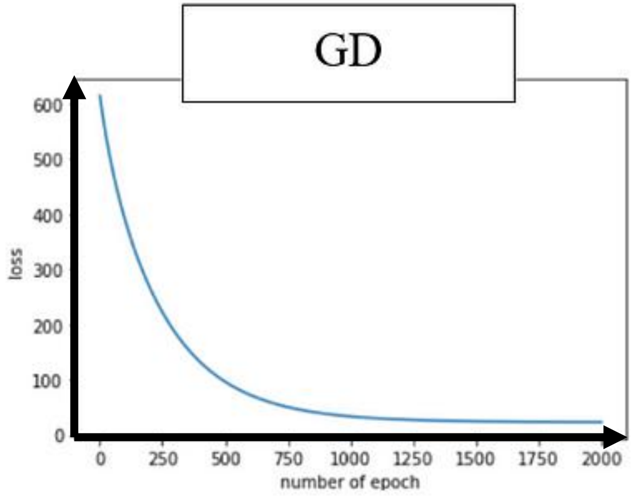


medium.com/@LayanSA/complete-guide-to-adam-optimization-1e5f29532c3d

(Recap) Mini-batch Gradient Descent

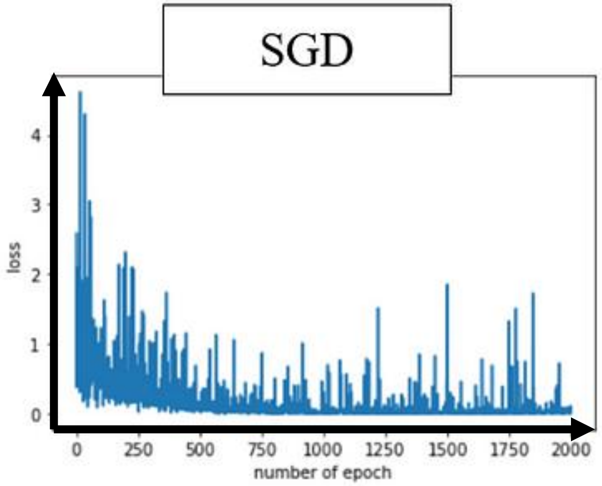
- **Intuition: Estimate** the gradient using **several random training samples**

Loss



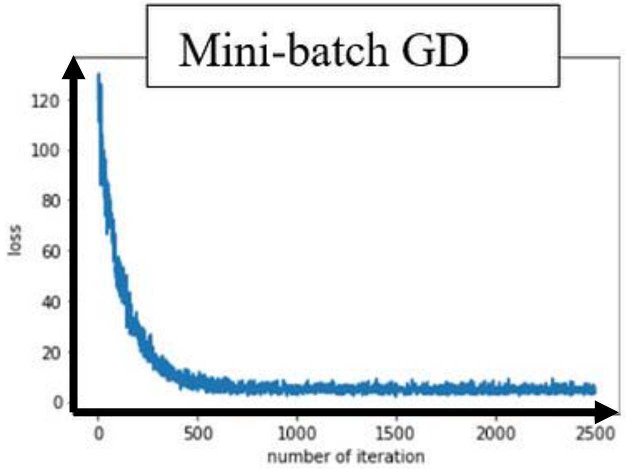
Epoch

$\text{batch size} = N$



Epoch

$\text{batch size} = 1$



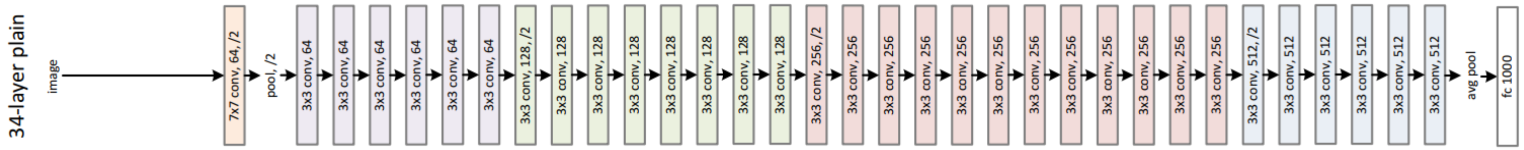
Epoch

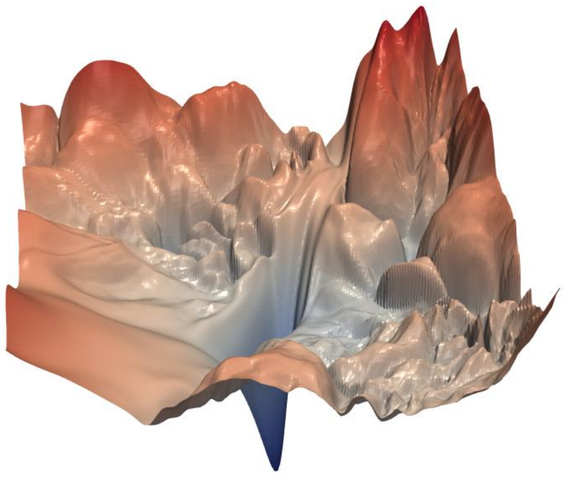
$1 < \text{batch size} < N$

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

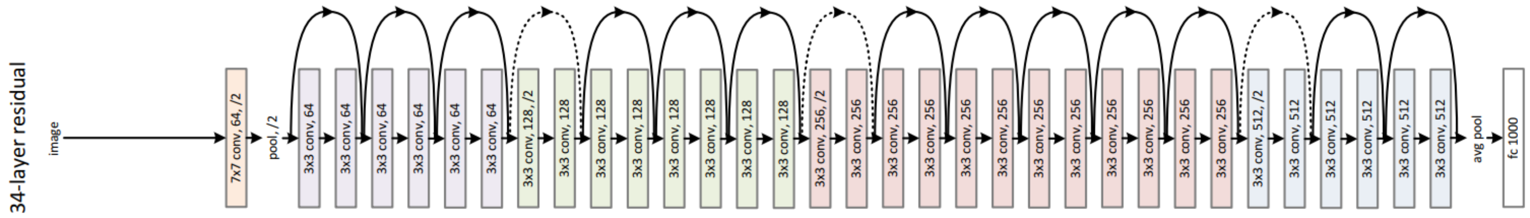
(Recap) Skip Connections

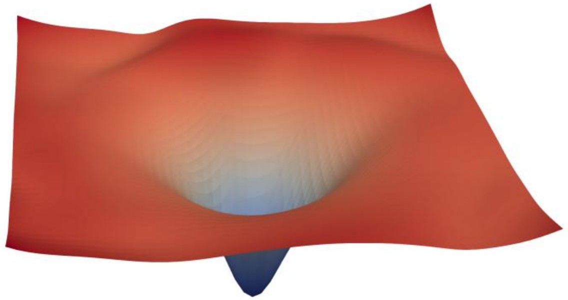
Without skip connections





With skip connections

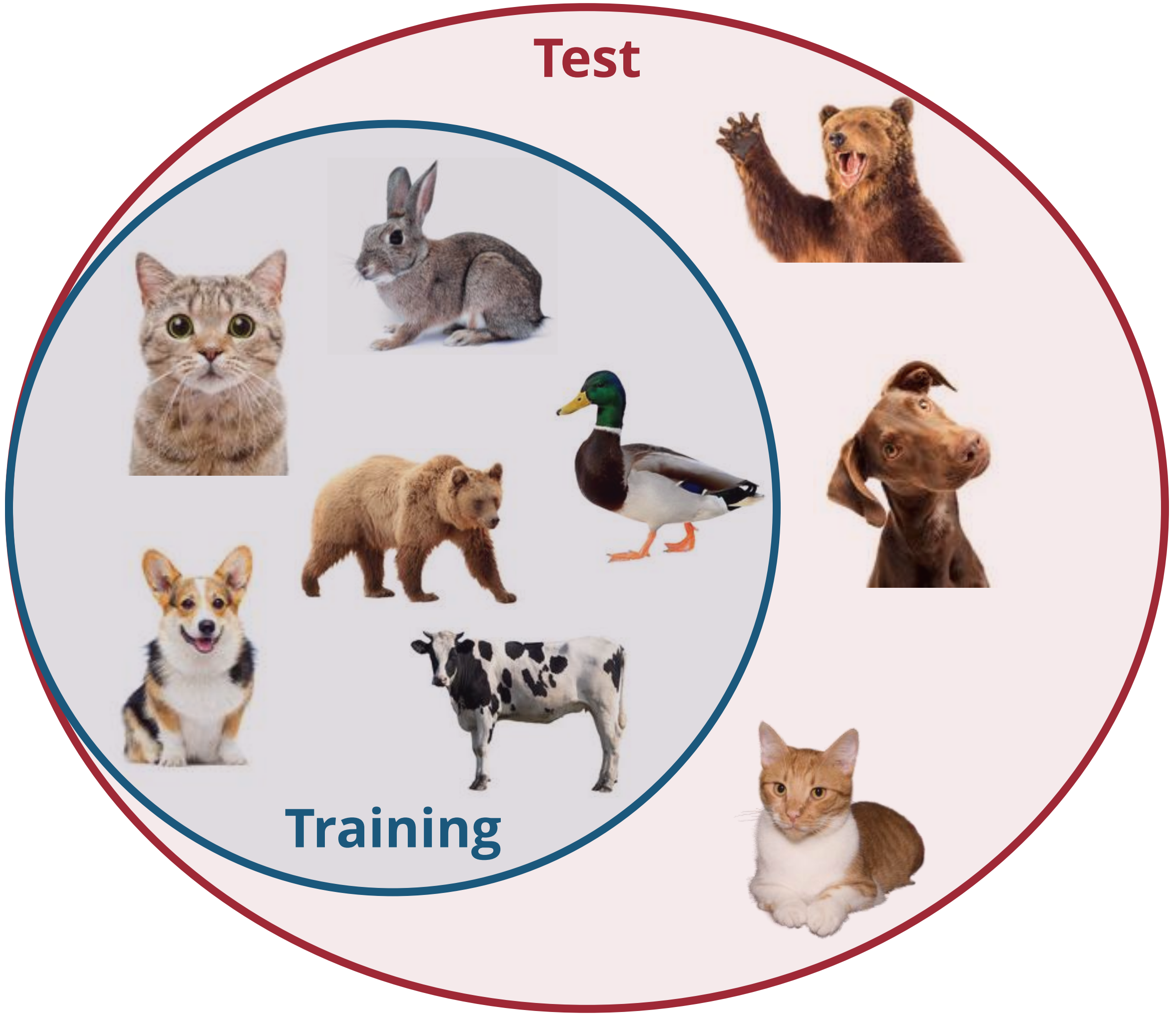




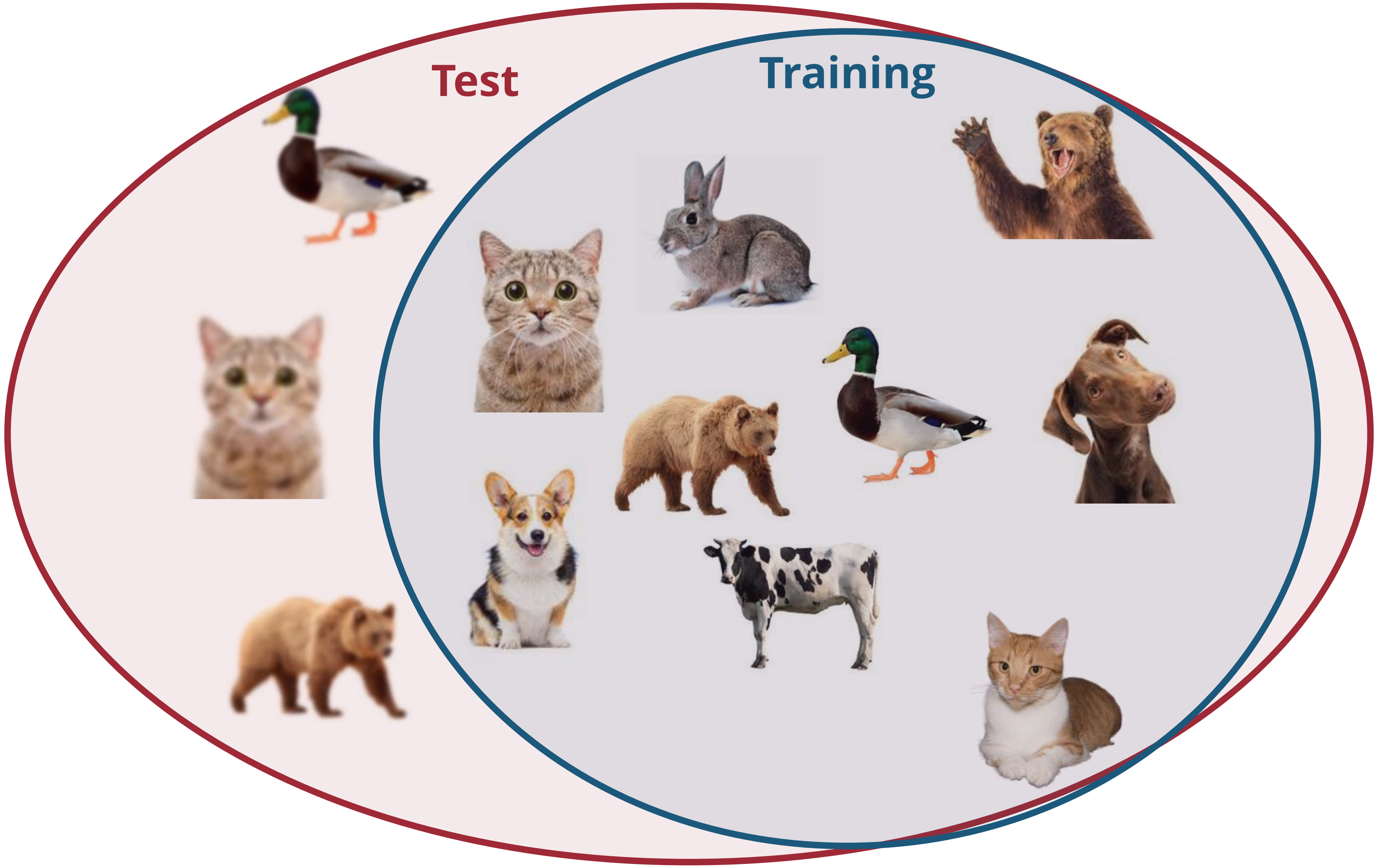
Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets," *NeurIPS*, 2018.

Training–Validation–Test

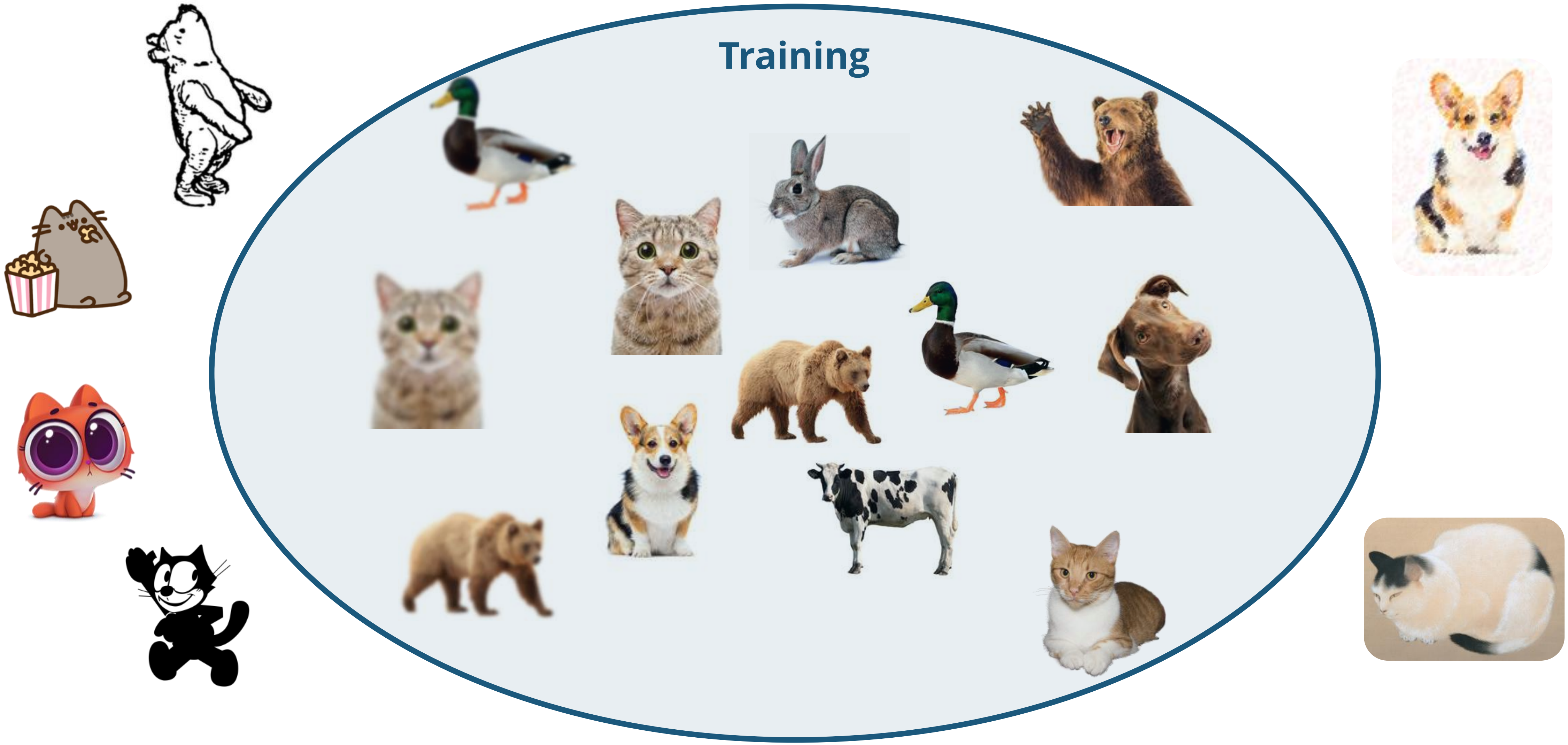
In-distribution vs Out-of-distribution



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

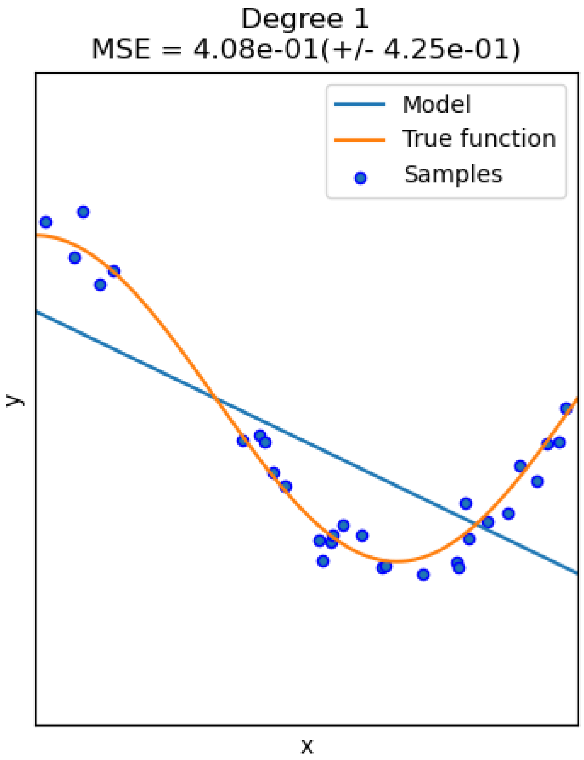
In-distribution vs Out-of-distribution

- 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

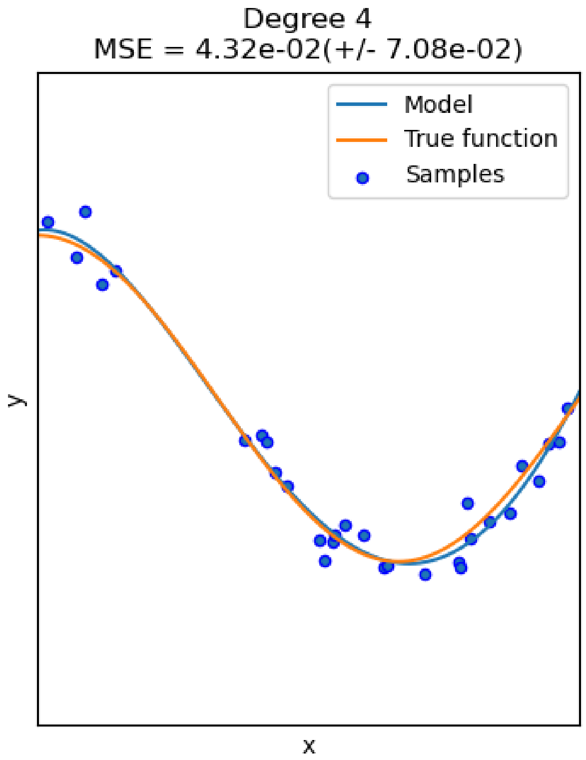
Overfitting & Underfitting

Underfitting

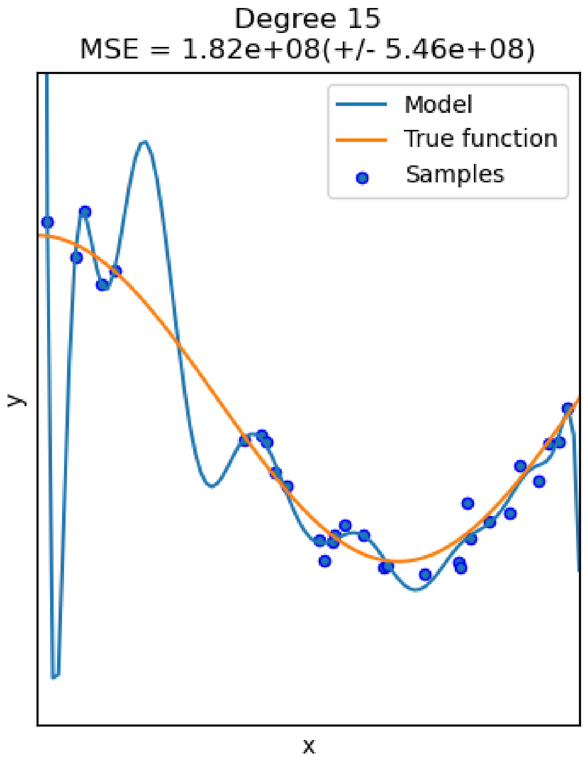


Model too inexpressive

Good fit!



Overfitting

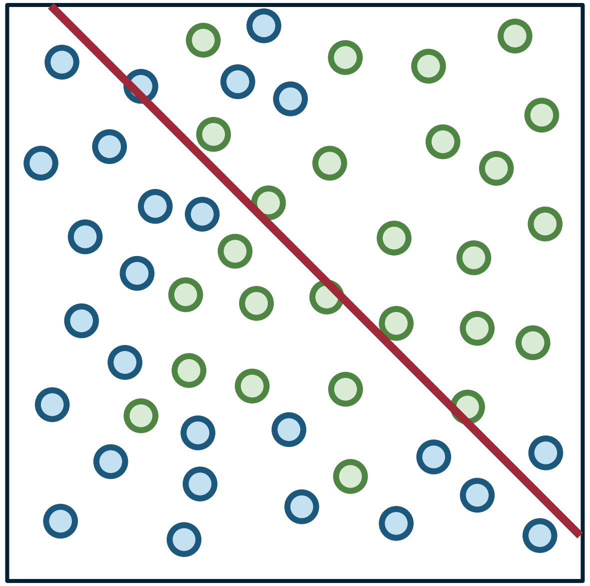


Model too expressive

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

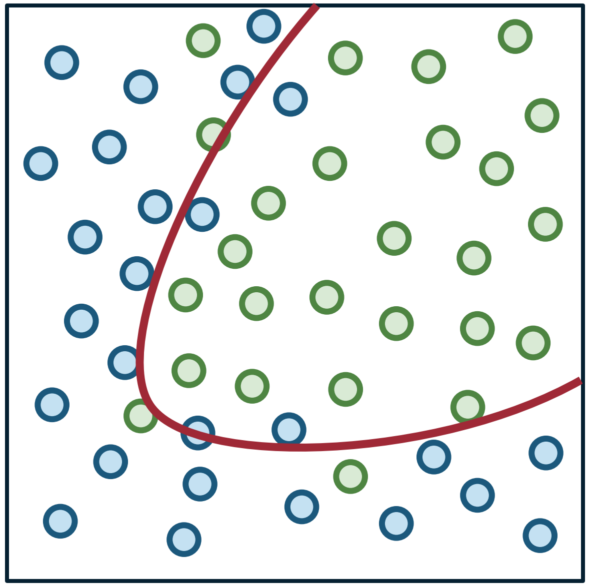
Overfitting & Underfitting

Underfitting

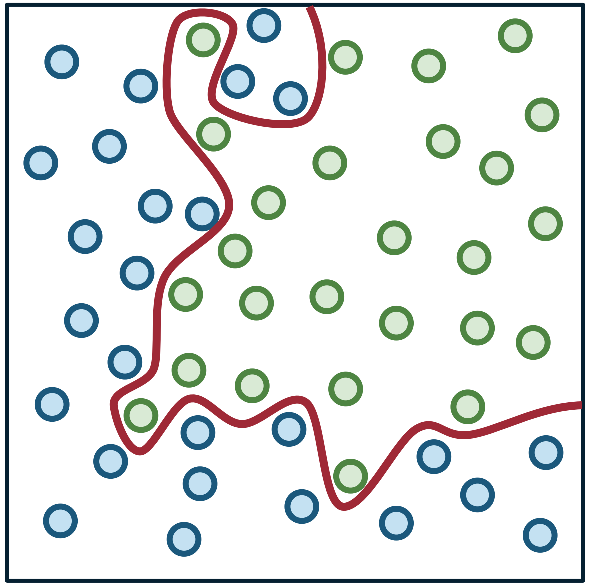


Model too inexpressive

Good fit!



Overfitting



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**

Training

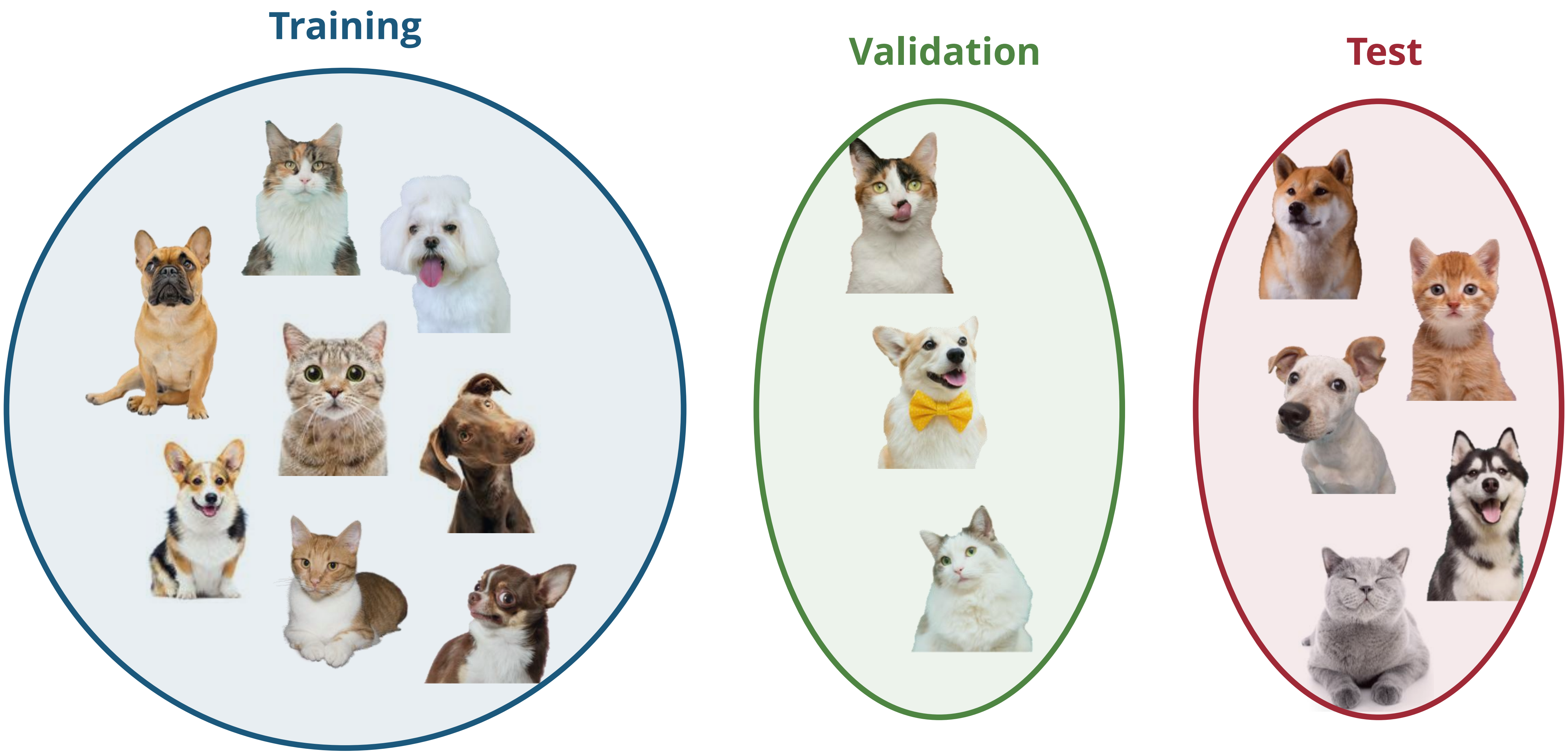
Test

Test Set is an Estimation of the Test Distribution

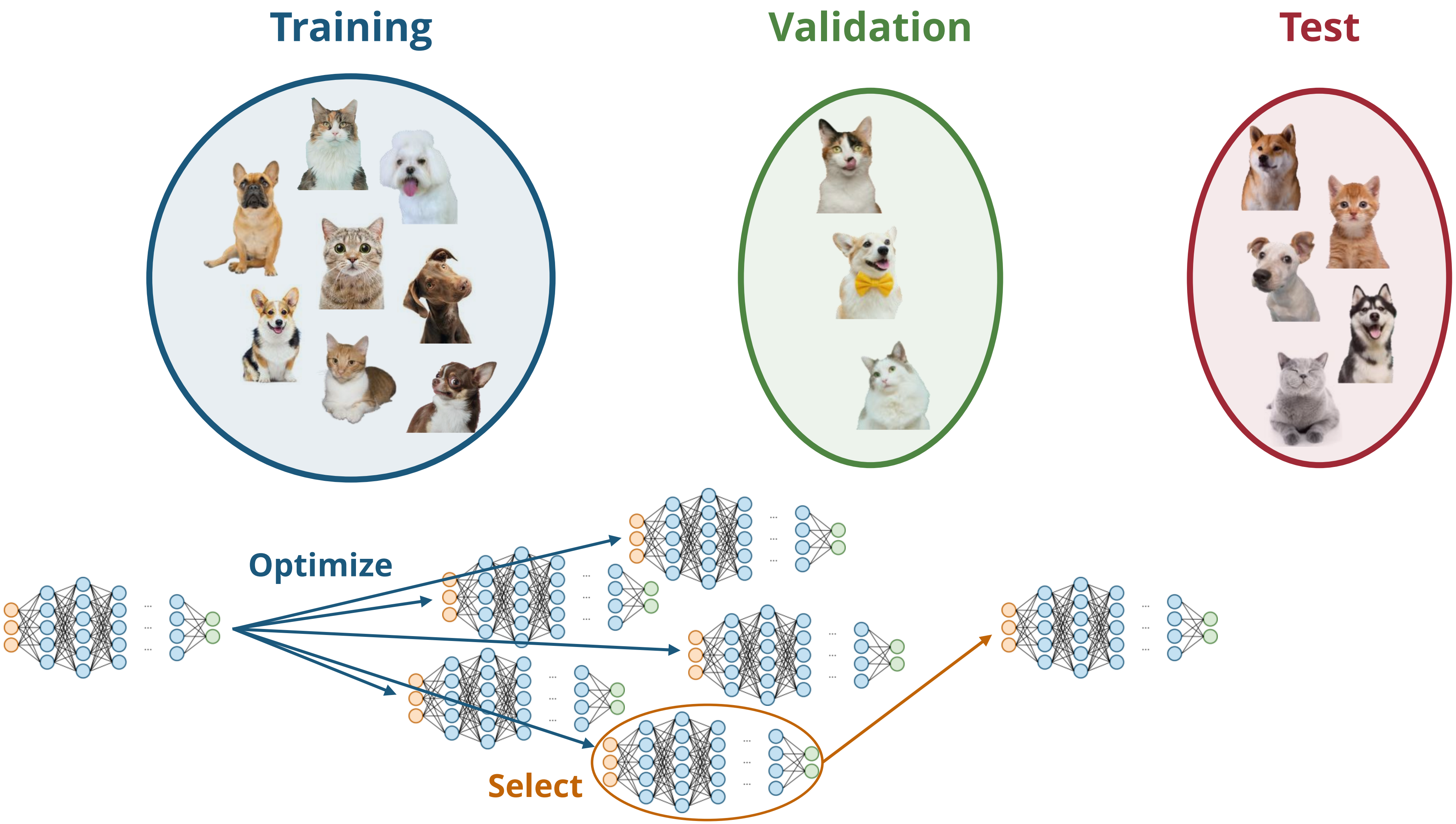
- We create a test set because we want to **estimate the performance when the model is applied to an interested distribution**

Train–Validation–Test Split

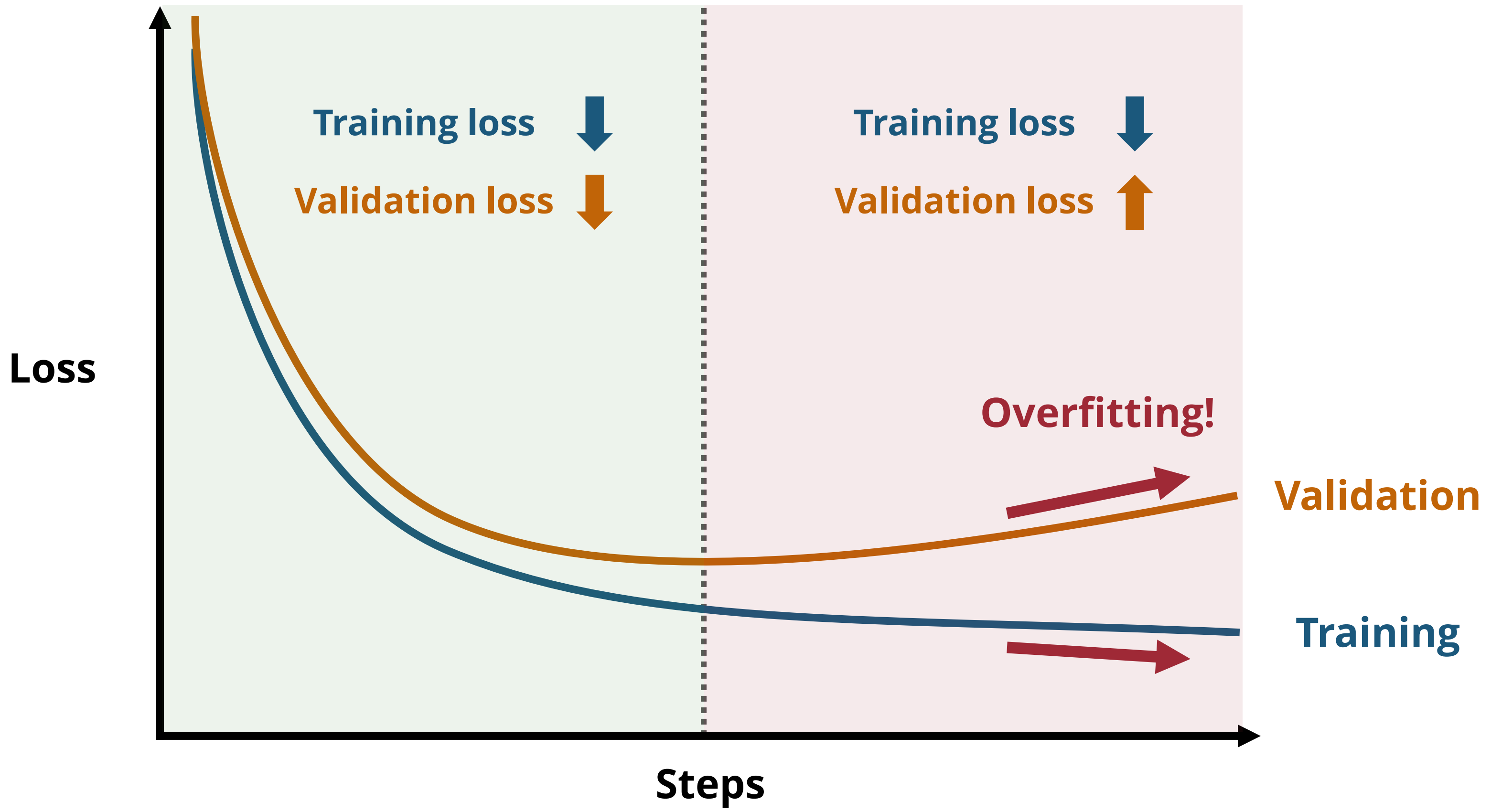
Train–Validation–Test Split



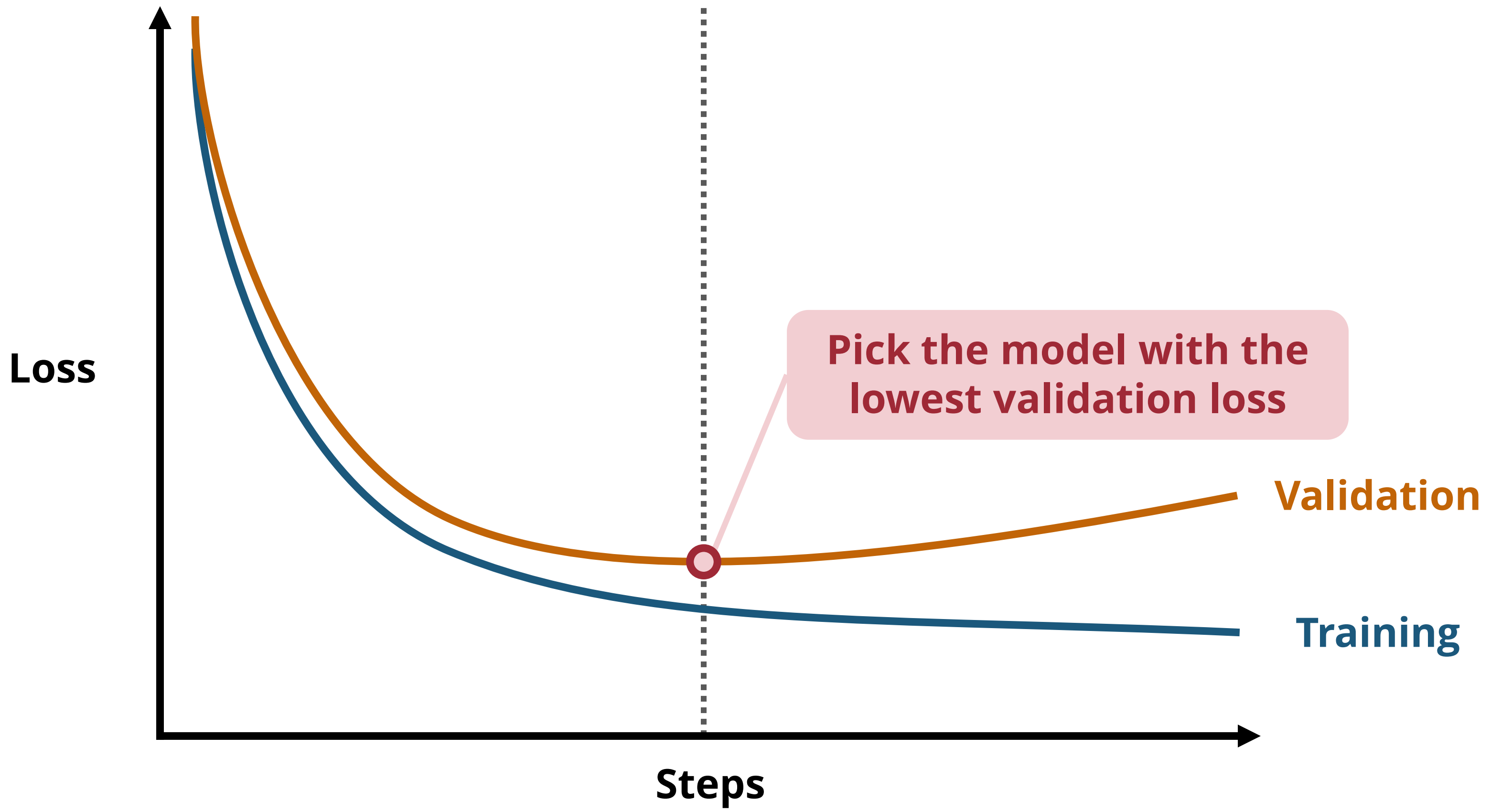
Training–Validation–Test Pipeline



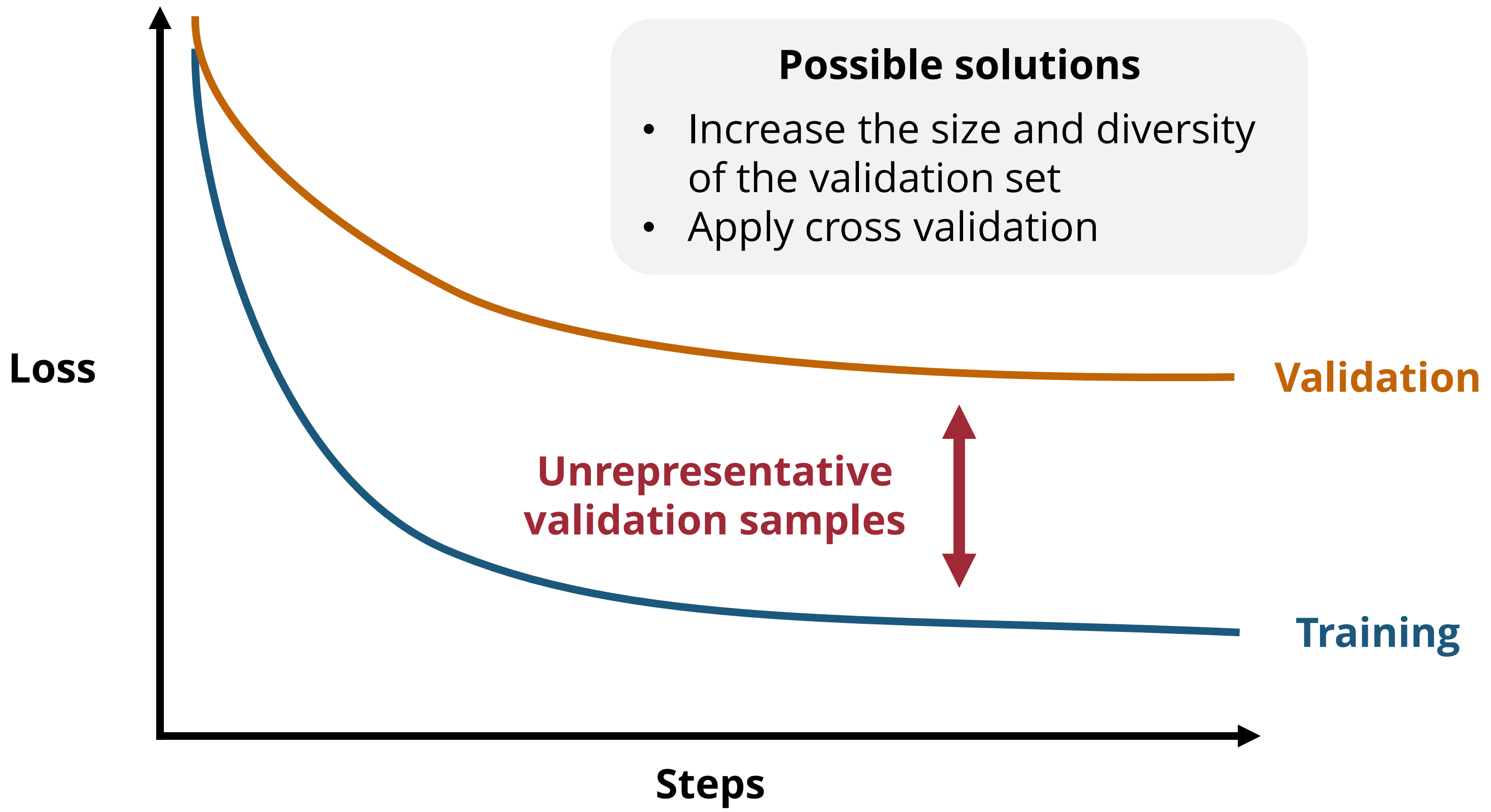
Training vs Validation Losses



Training vs Validation Losses



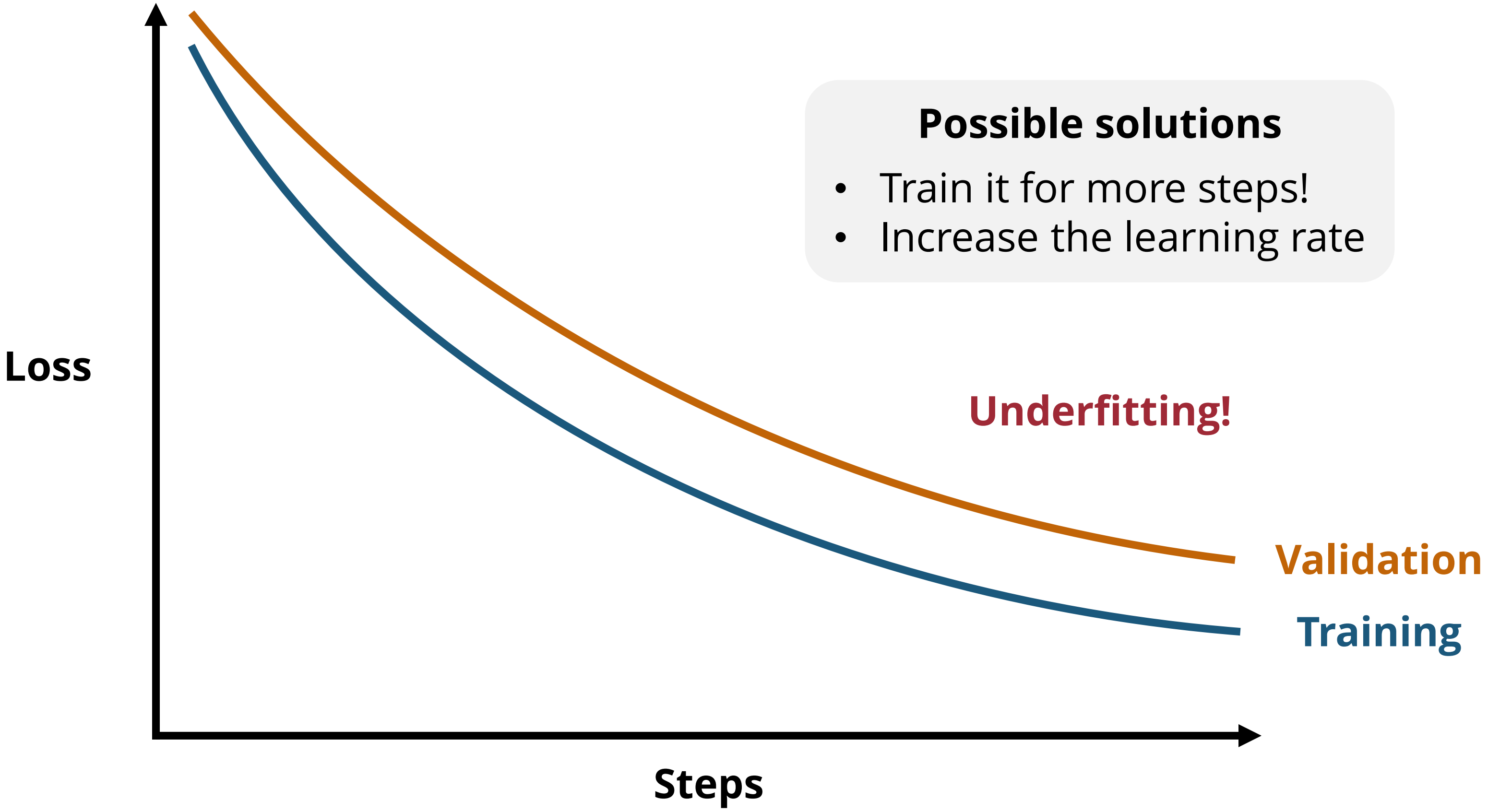
Training vs Validation Losses



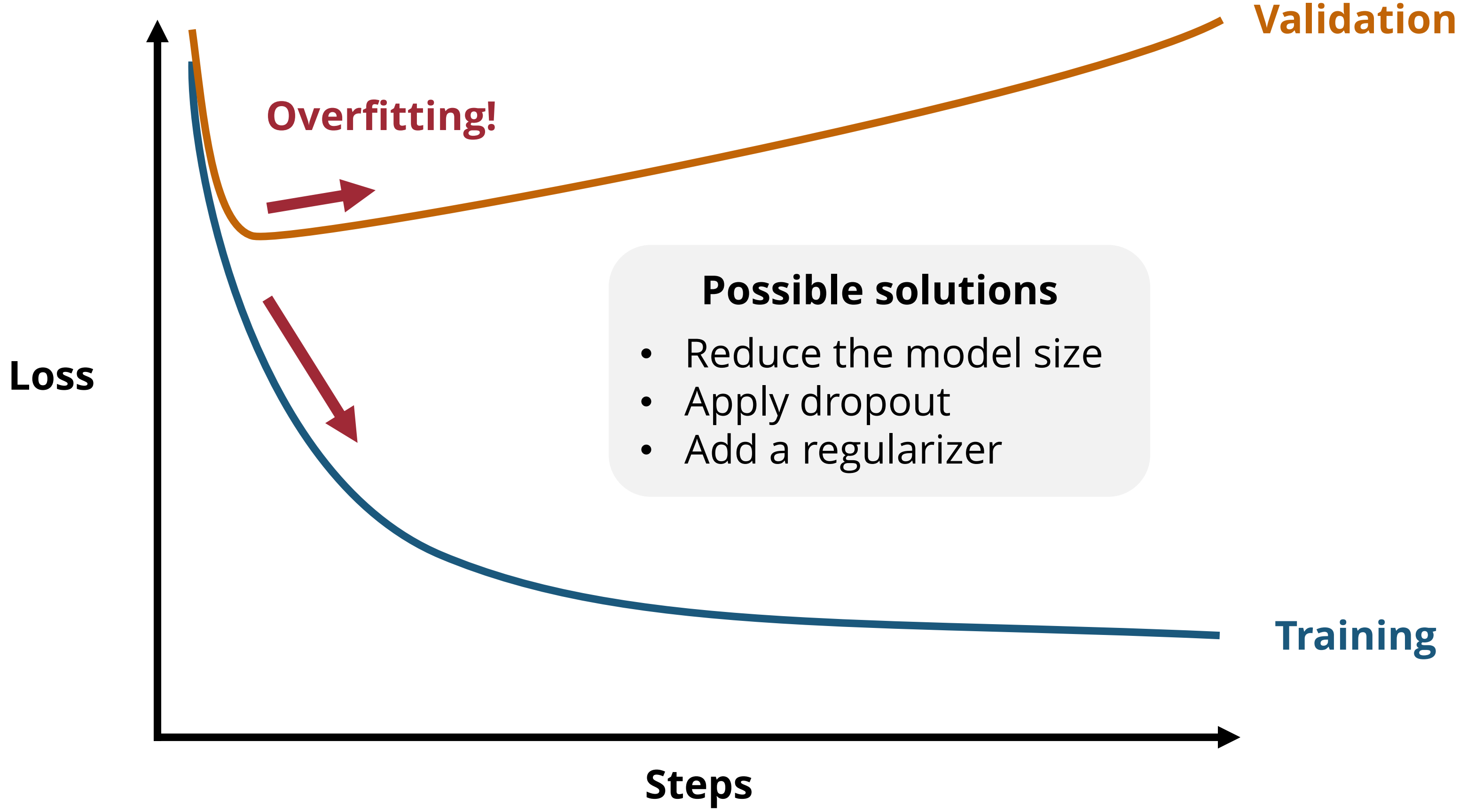
Training vs Validation Losses

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

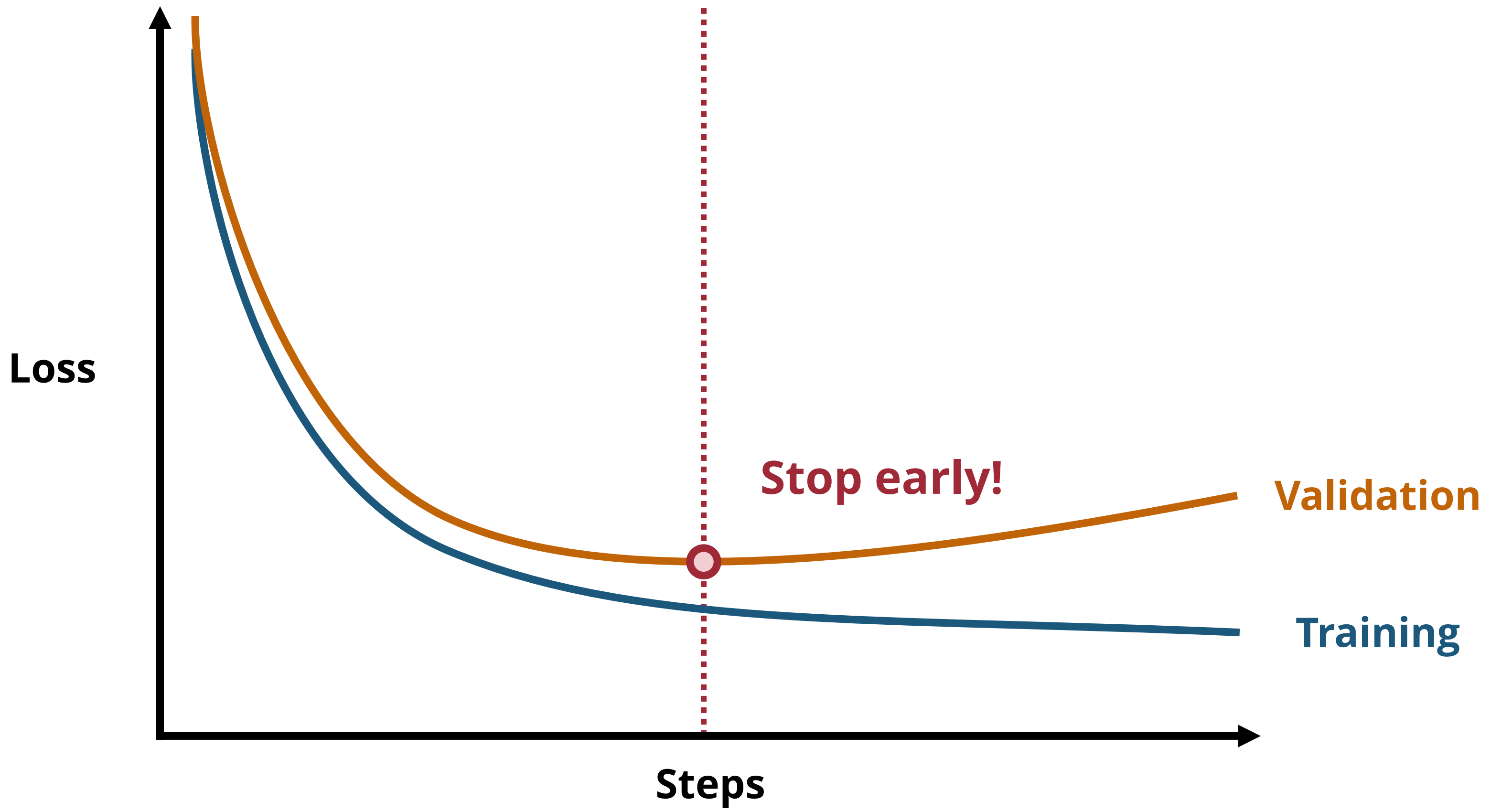
Train–Validation–Test Split

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

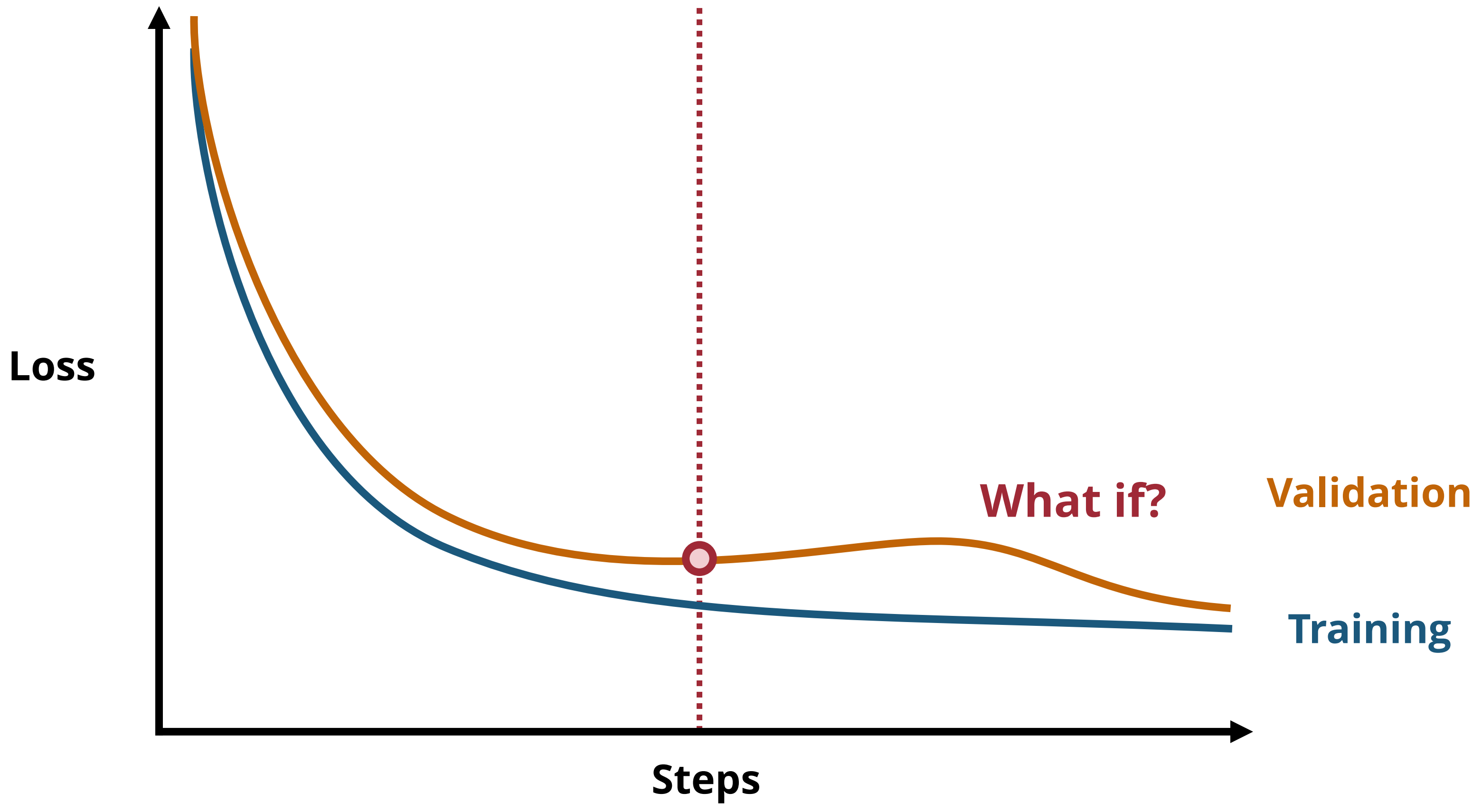
- What's the **best ratio**?
  - Most common: **8:1:1** or 9:0.5:0.5
  - For smaller dataset, you might even want 6:2:2

Overcoming Overfitting

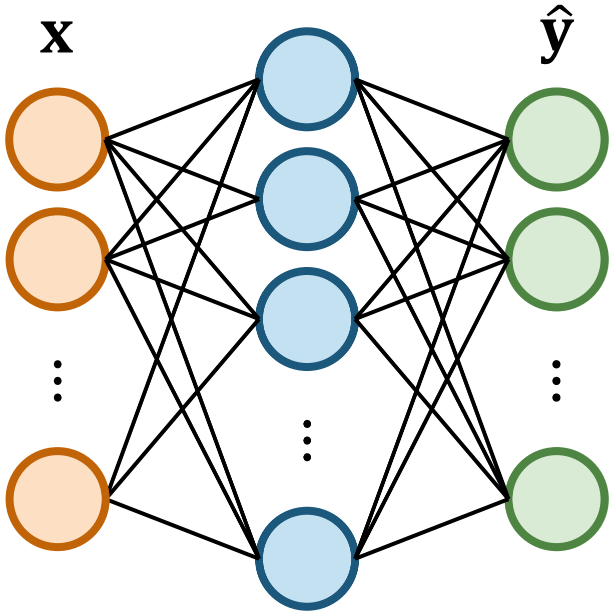
Early Stopping



Early Stopping

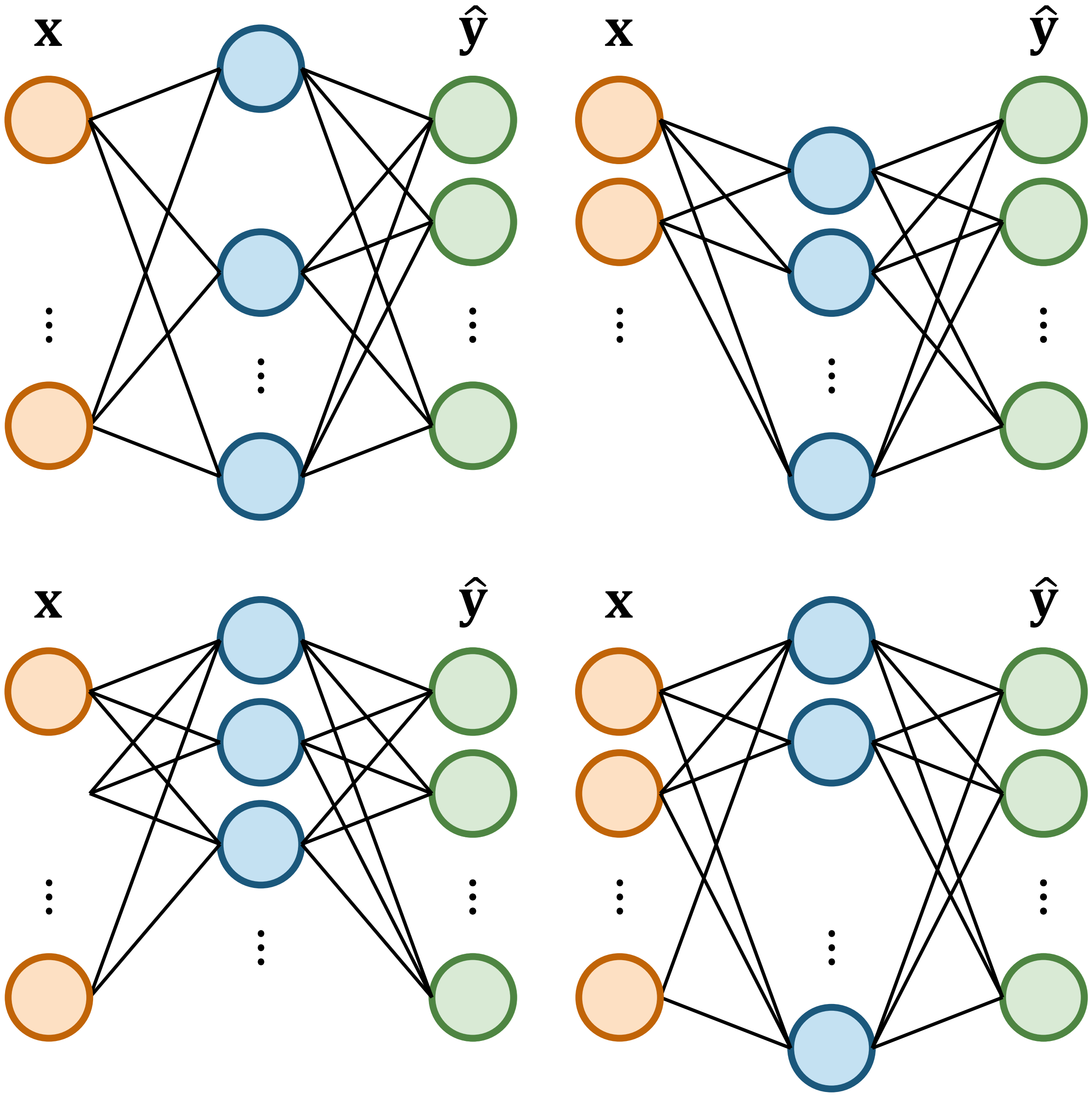


Dropout

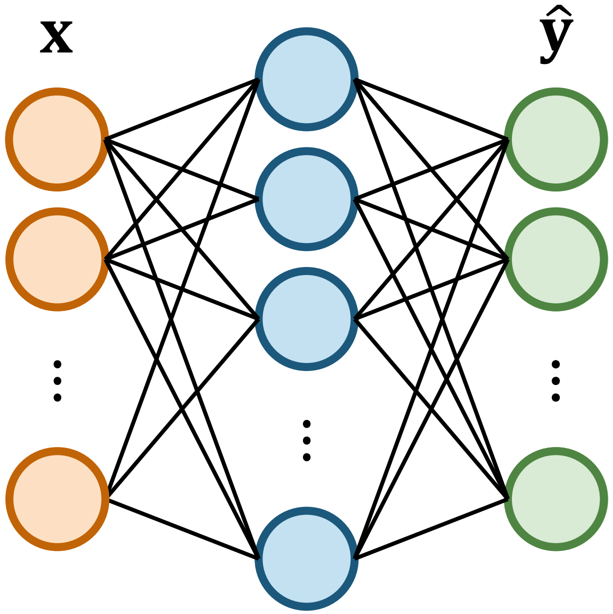


Each neuron may be removed with probability p during training

Dropout rate

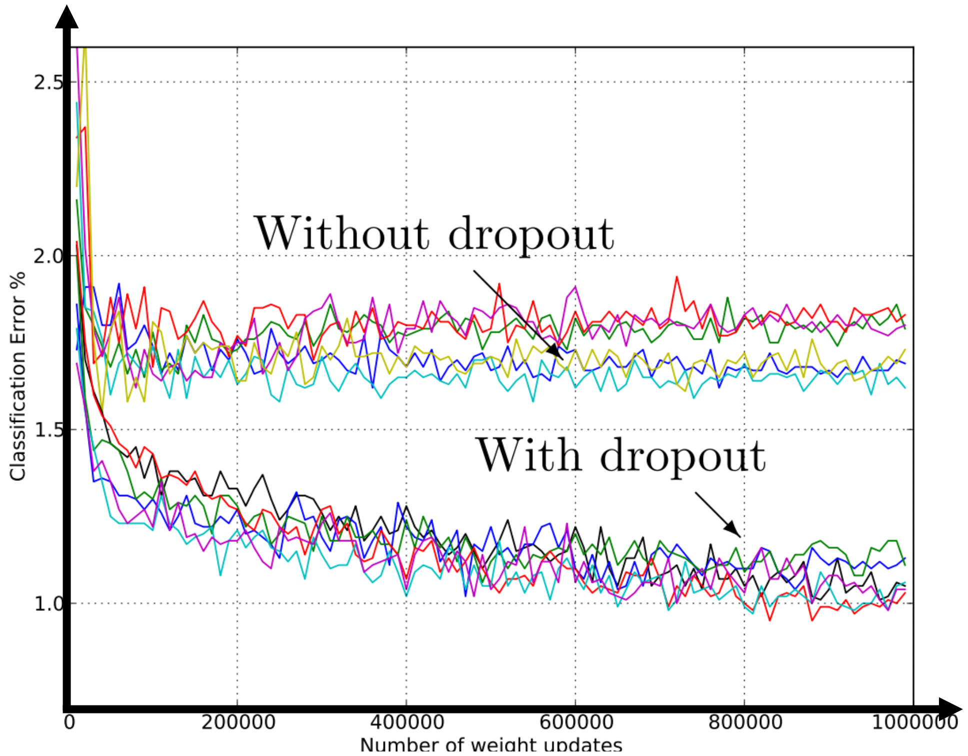


Dropout



Each neuron may be removed with probability p during training

Test error rate



Weight updates

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