Neural Network Training – A Top-Down Perspective

Danna Gurari
University of Colorado Boulder
Spring 2024

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2024/

Review

- Last lecture:
  - Objective function: what to learn
  - Gradient descent: how to learn
  - Training a neural network: optimization
  - Lab assignment expectations
  - Programming tutorial

- Assignments (Canvas):
  - Lab assignment 1 due in one week

- Questions?

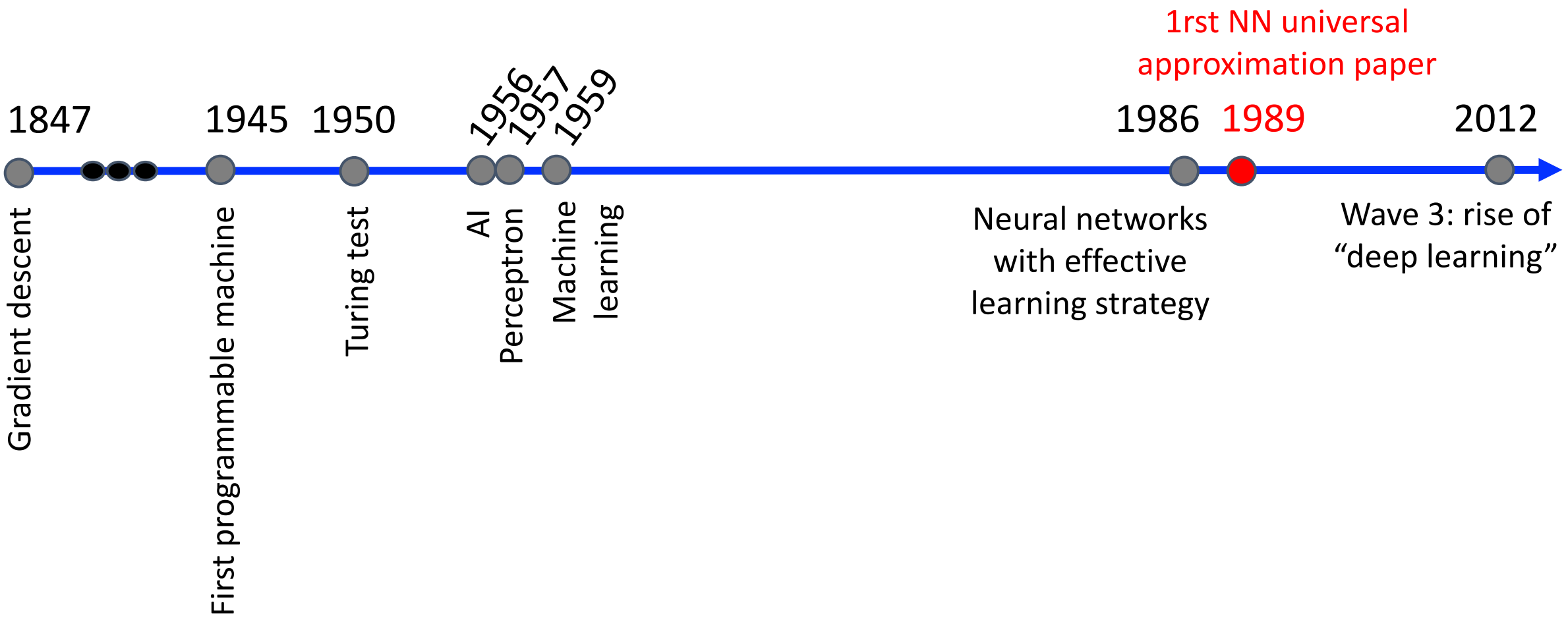
Today's Topics

- Universal approximation theorem vs No Free Lunch theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Training neural networks: hardware & software
- Programming tutorial

Today's Topics

- Universal approximation theorem vs No Free Lunch theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Training neural networks: hardware & software
- Programming tutorial

Historical Context: Universal Approximator

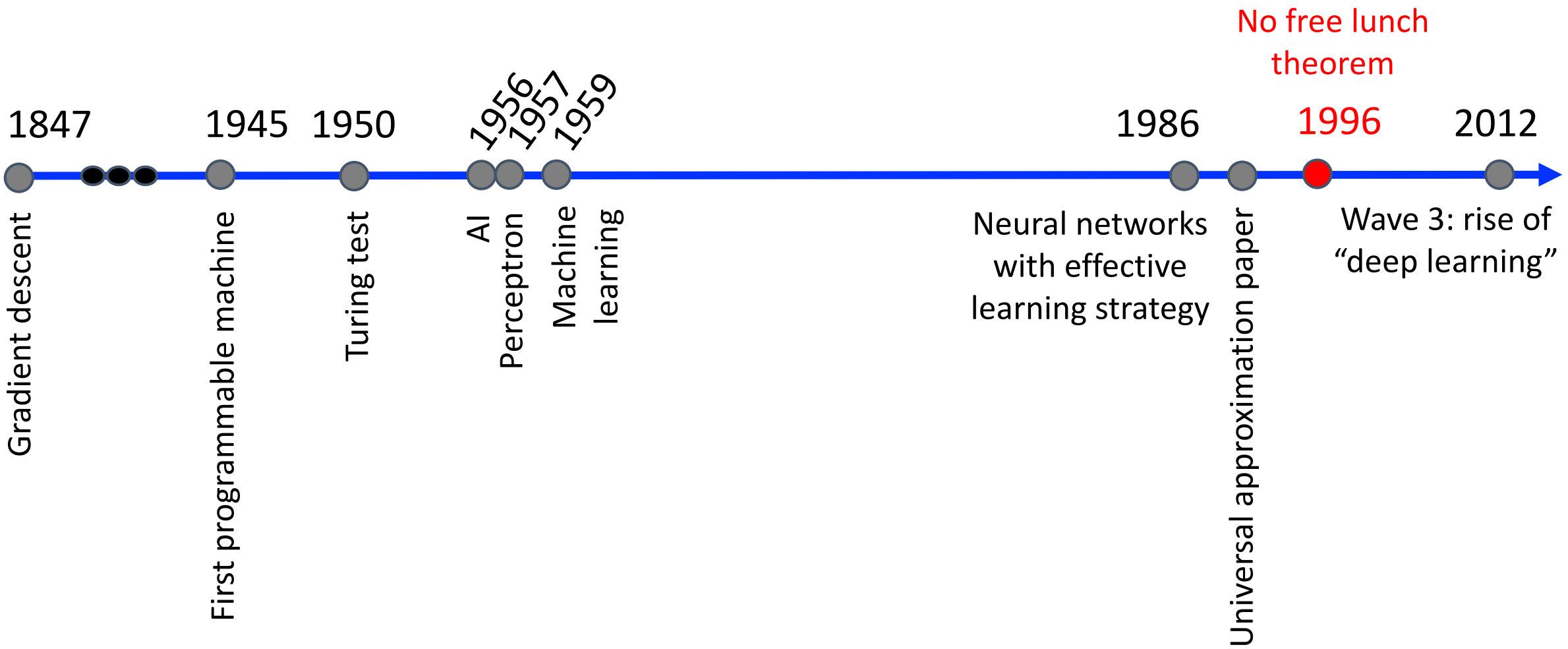


Hornik, Stinchcombe and White. Multilayer feedforward networks are universal approximators. Neural Networks, 1989

“The universal approximation theorem means that regardless of what function we are trying to learn, we know that a large MLP [multilayer perceptron] will be able to *represent* this function.”

- Ch. 6.4.1 of Goodfellow book on Deep Learning

Historical Context: Challenge



Hornik, Stinchcombe and White. Multilayer feedforward networks are universal approximators. Neural Networks, 1989

“**no free lunch theorem**… no machine learning algorithm is universally any better than any other.”

- Ch. 5.2.1 of Goodfellow book on Deep Learning

Deep Learning Challenge

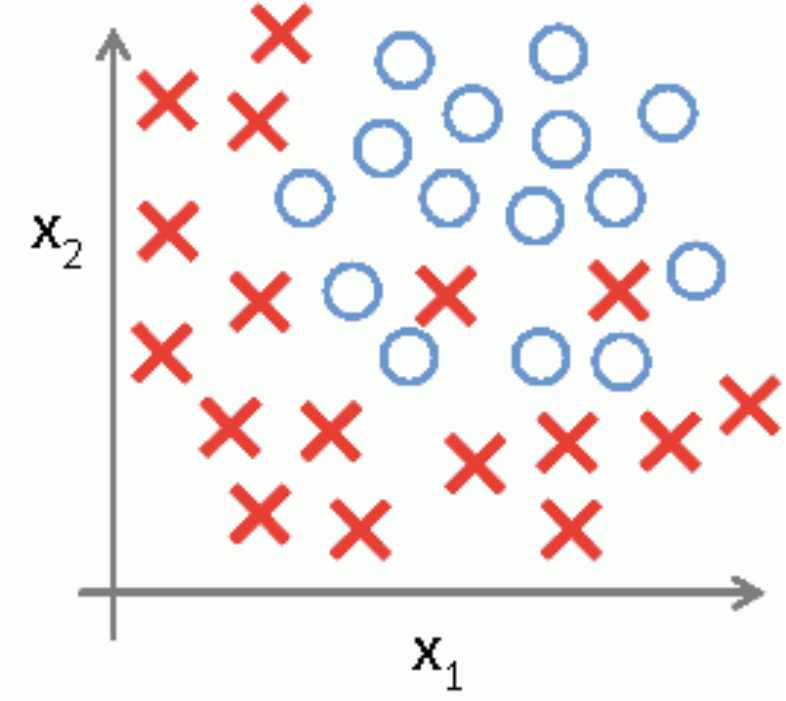
Since neural networks can in theory represent ANY function, how do we learn models that can perform well for the data generated in real world problems…

Today's Topics

- Universal approximation theorem vs No Free Lunch theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Training neural networks: hardware & software
- Programming tutorial

Recall: Class Exercise from Lecture 1

- Model-based classification approach: separate x from o



Class volunteer:
1) Draw a straight line (linear equation)
2) Draw a parabola (quadratic equation)
3) Draw any curve

Models with increasing representational capacity

Figure source: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

Model Capacity

Which model would you choose to separate x from o?

(a) (b) (c)

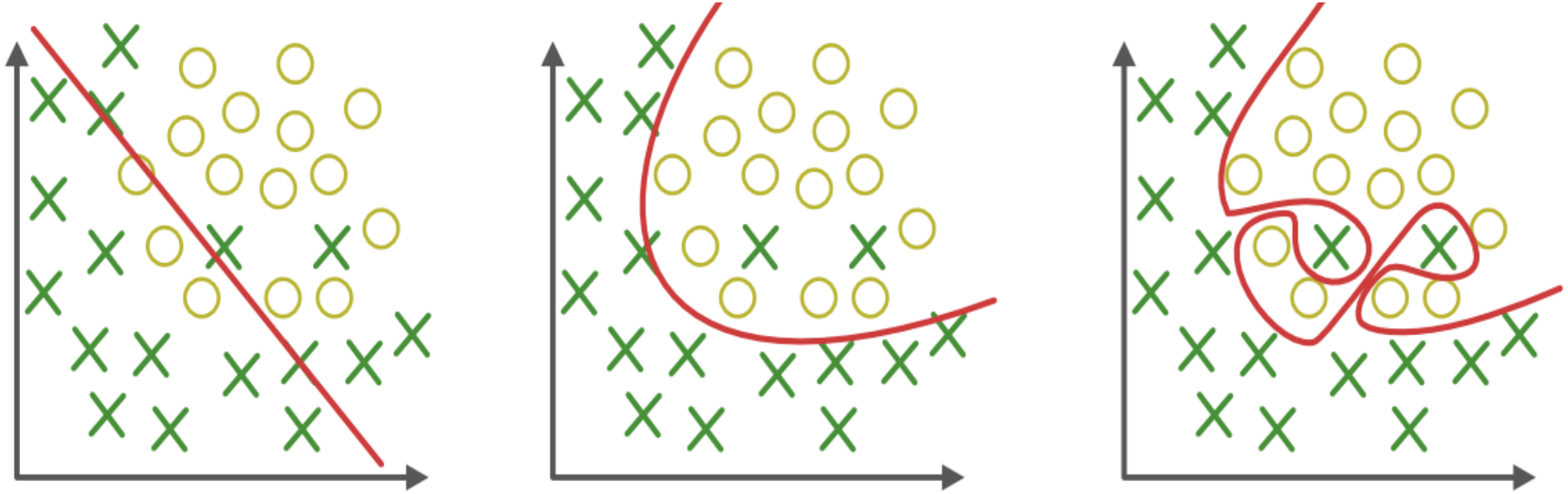


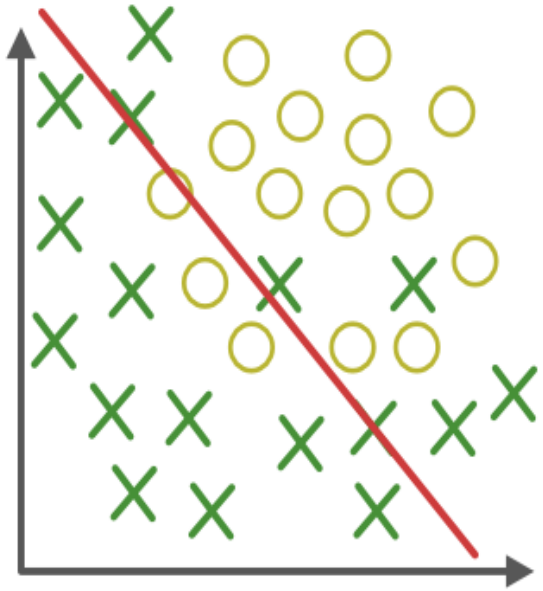
Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity

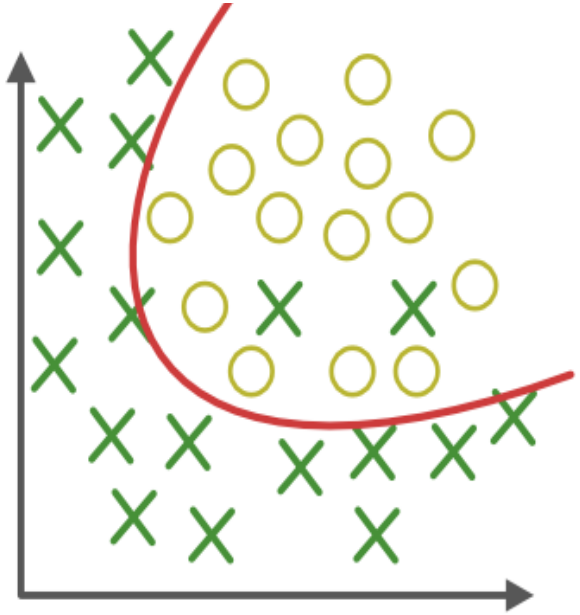
Underfits: too simple to explain the data

Overfits: too complex to generalize to a test set

(a)



(b)



(c)

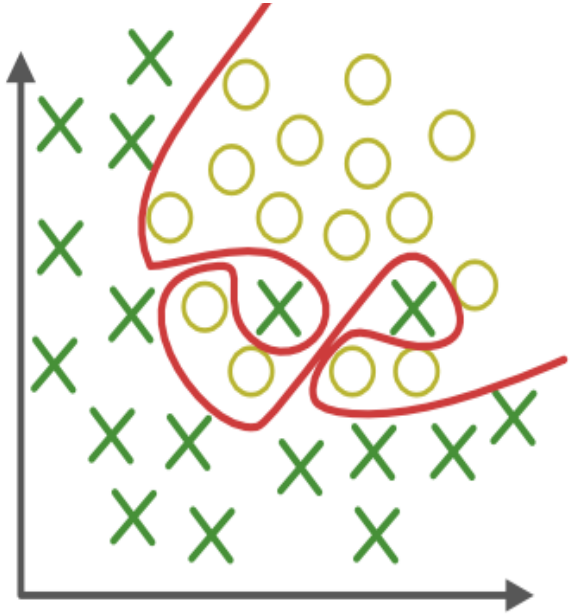


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

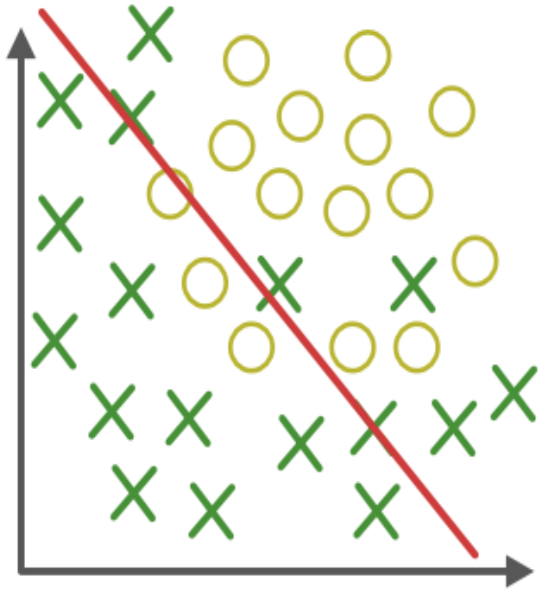
Model Capacity

Key challenge for neural networks since they have many parameters

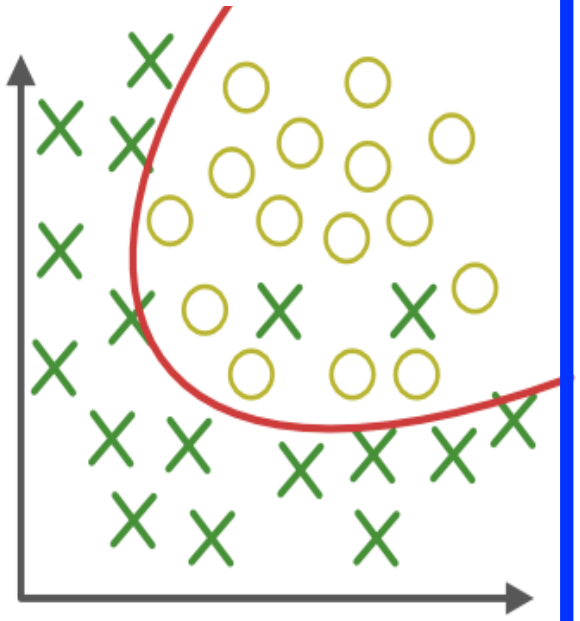
Underfits: too simple to explain the data

Overfits: too complex to generalize to a test set

(a)



(b)



(c)

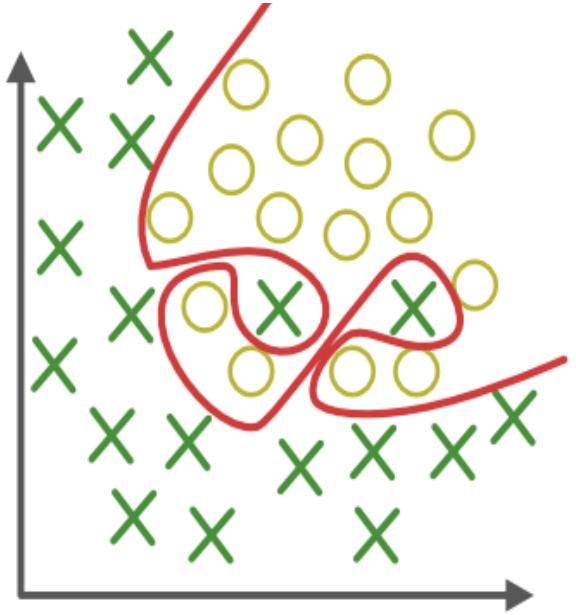


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity: Overfitting Problem

- Problem: models can learn to model **noise** and so generalize poorly to novel examples!
- What would cause noise in a dataset?
  - e.g., incorrect data entry/labeling, hardware measurement error
- Caution: some outliers are not noise and so are data points we want models to learn

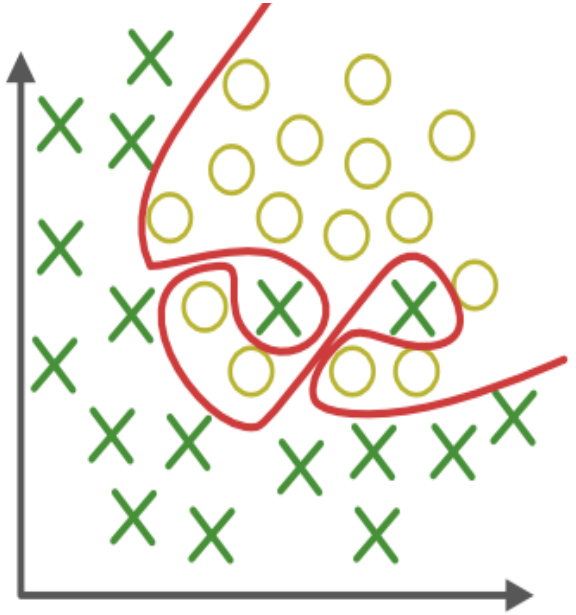
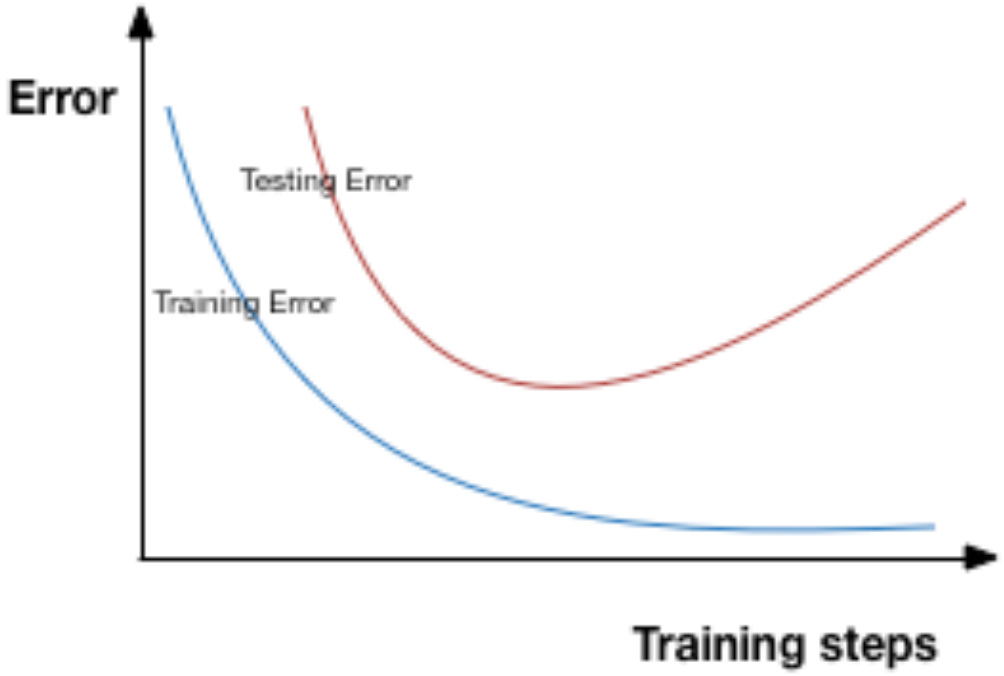


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity: Overfitting Remedy

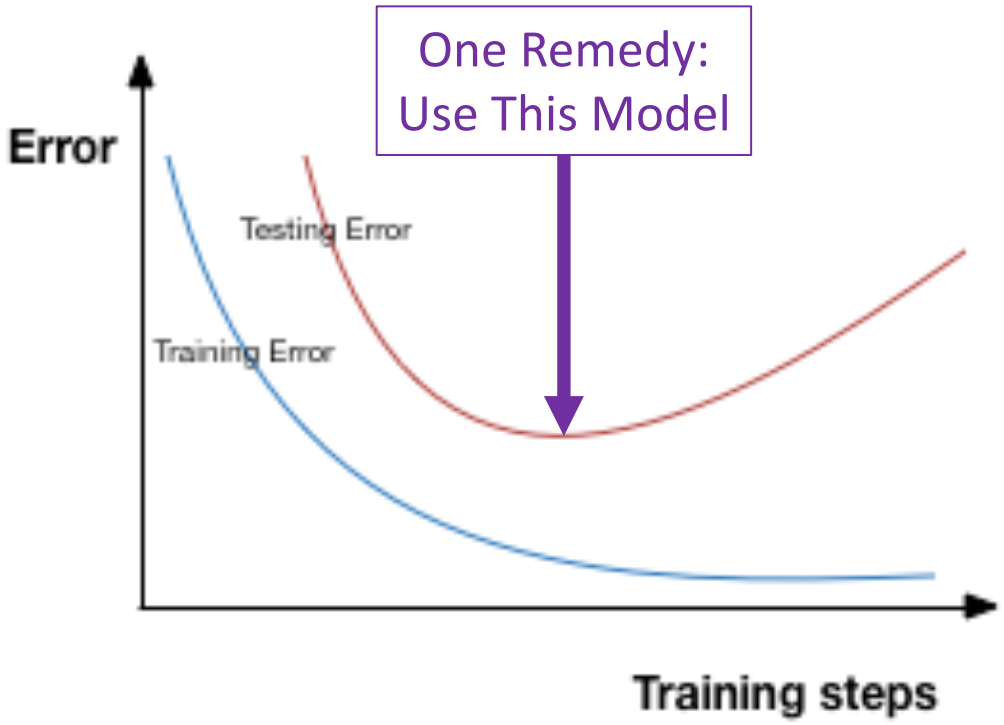


- To detect overfitting, analyze **learning curves** for models tested on training data and test data
  - What happens to training data error as number of training steps increases?
    - Error shrinks
  - What happens to test data error as number of training steps increases?
    - Error shrinks and then grows
  - Why does training error ***shrink*** and test error ***grow***?
    - Modeling ***noise*** in the training data (i.e., "overfitting") reduces training error at the expense of losing knowledge that generalizes to unobserved test data

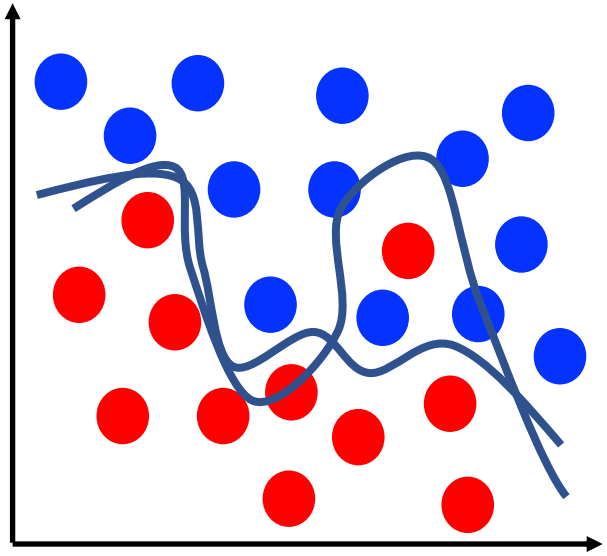
Image Source: https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

Model Capacity: How to Avoid Overfitting?

Early stopping



Add training data



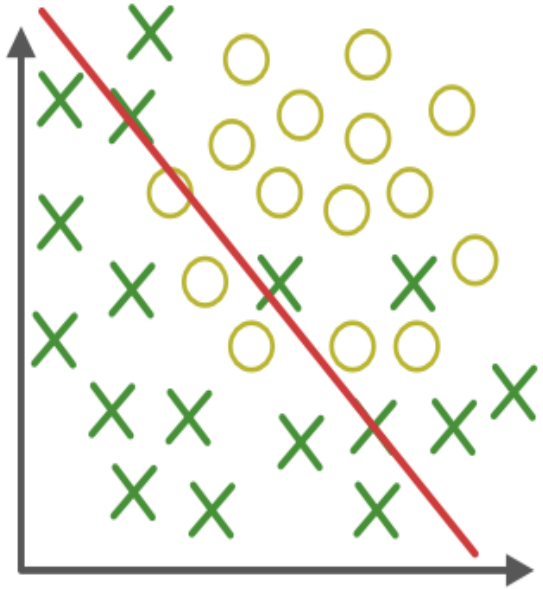
Many more techniques to be discussed in this course…

Image Source: https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

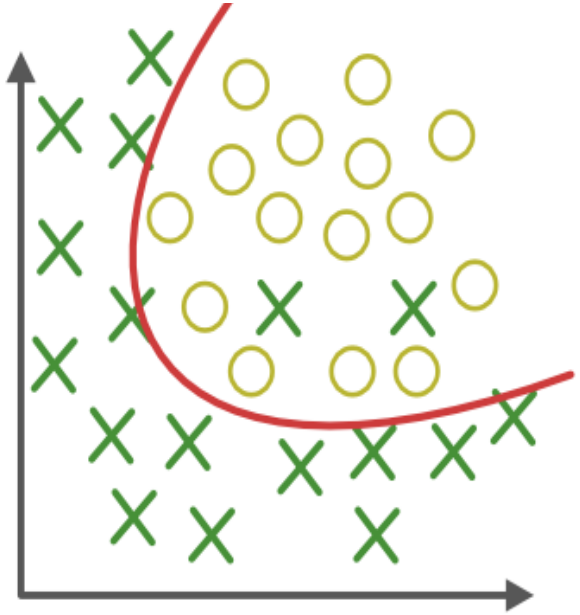
Model Capacity

Underfits: too simple to explain the data

(a)



(b)



Overfits: too complex to generalize to a test set

(c)

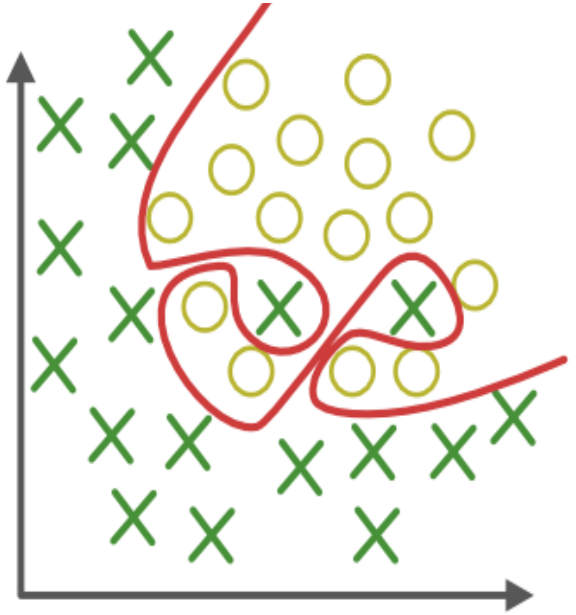
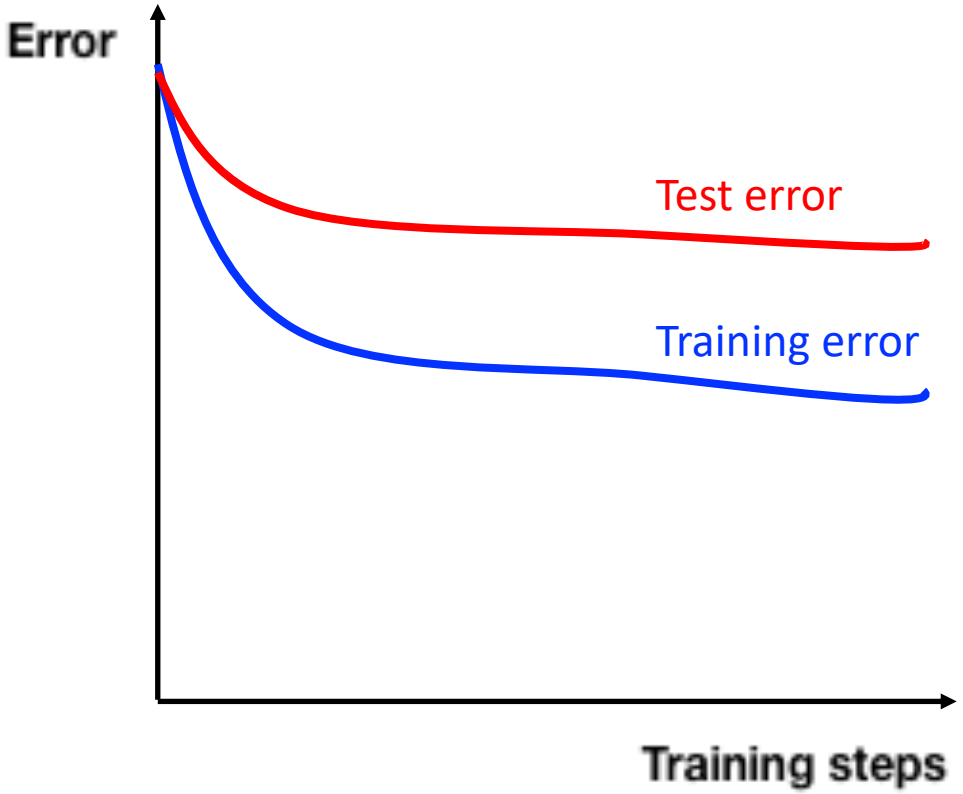


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity: Underfitting



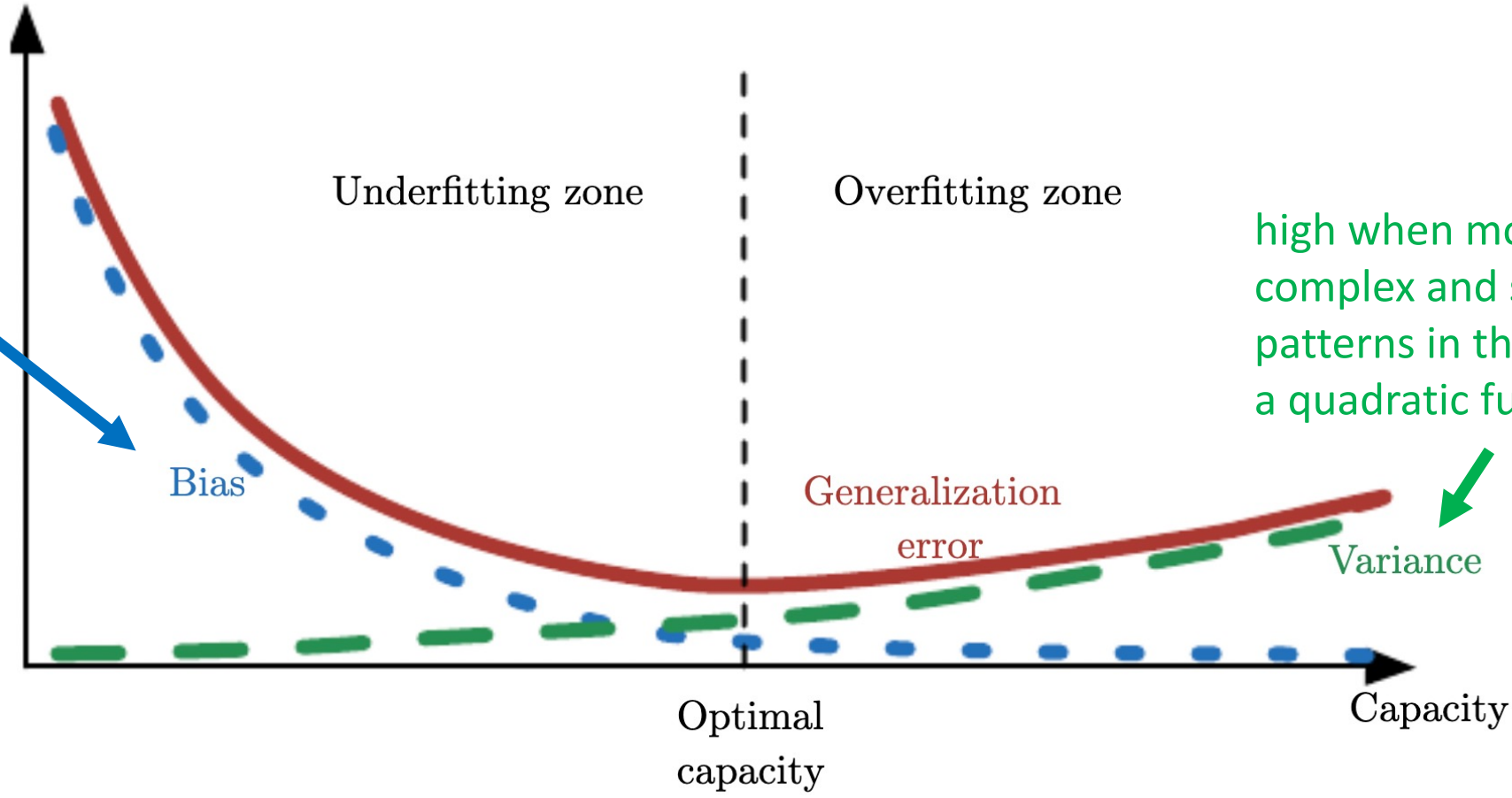
- To detect underfitting, analyze **learning curves** for models tested on training data
  - What happens to training data error as number of training steps increases?
    - Error remains high

Model Capacity: How to Avoid Underfitting?

Increase representational complexity, for example add the number of layers and/or units in a neural network

Model Capacity: Overfitting vs Underfitting

(not related to the "bias" parameter in neural networks): high when model is overly-simple and so assumes simpler patterns in the data (e.g., linear function)

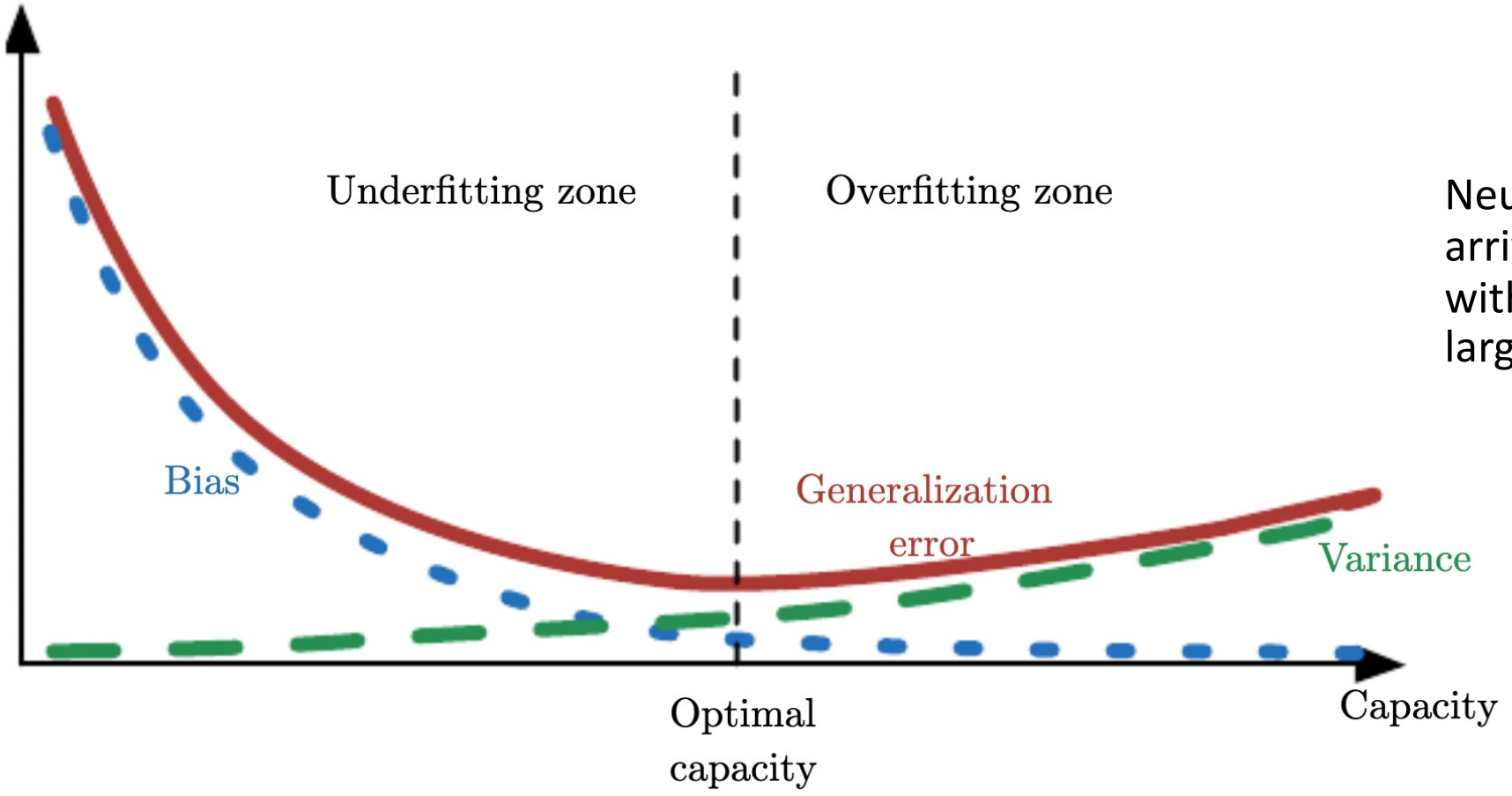


high when model is overly-complex and so misses simpler patterns in the data (e.g., does a quadratic function suffice?)

Often discussed with respect to a **bias-variance** trade-off

Source: Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016

Model Capacity: Overfitting vs Underfitting



Neural networks can simultaneously arrive at low bias and low variance, with large neural networks and large amounts of training data

Source: Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016

Summary: Model Capacity

- Goal: learn model with capacity that is neither too small nor too large so it generalizes well when predicting on previously unseen test data

- Challenges: choosing…
  - **Architecture** (i.e., number of layers, number of units per layer)
  - **Training algorithm** (e.g., training duration too brief/long)
  - **Training dataset** (e.g., insufficient training data)

Today's Topics

- Universal approximation theorem vs No Free Lunch theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Training neural networks: hardware & software
- Programming tutorial

Model Design Decisions

Model hyperparameters (selected); e.g.,

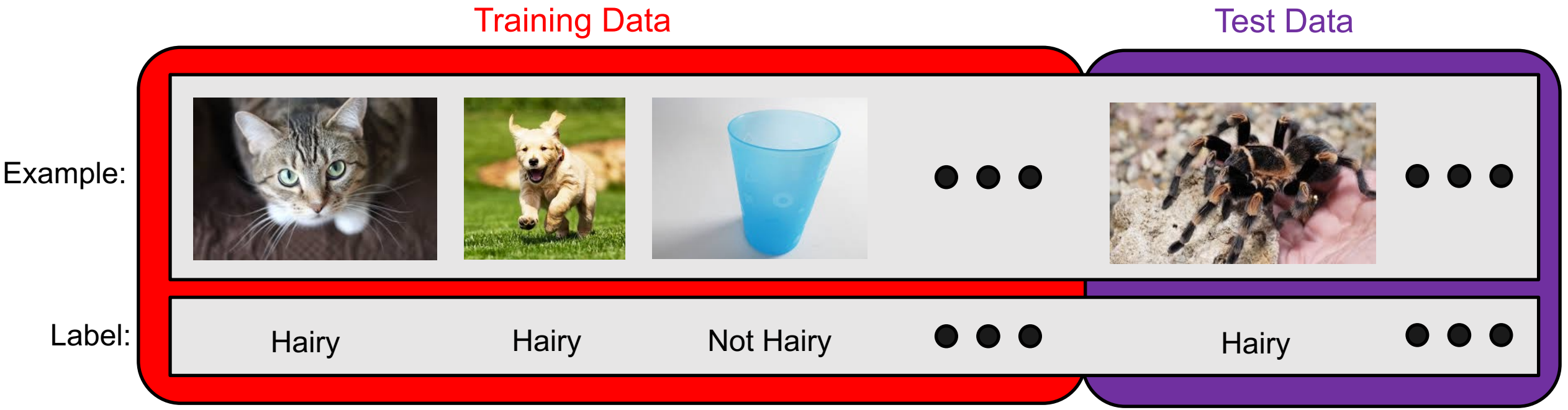
- Number of layers
- Number of units in each layer
- Activation function
- Batch size
- …

Model parameters (learned)

- Weights
- Biases
- …

Key Challenge: how to design a model without repeatedly observing the test data (which leads to overfitting)?

Recall: Our Goal is to Design Models that **Generalize** Well to New, Previously Unseen Examples (Test Data)

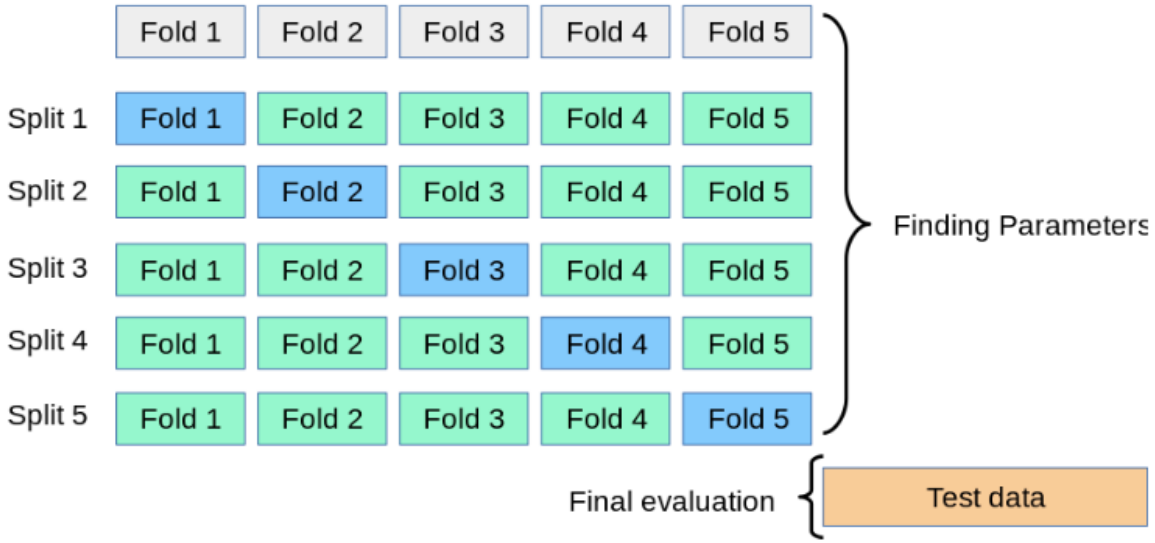


Key Challenge: how to design a model without repeatedly observing the test data (which leads to overfitting)?

Hyperparameter Tuning: Split Training Set So It Can Be Used to Test Different Hyperparameters

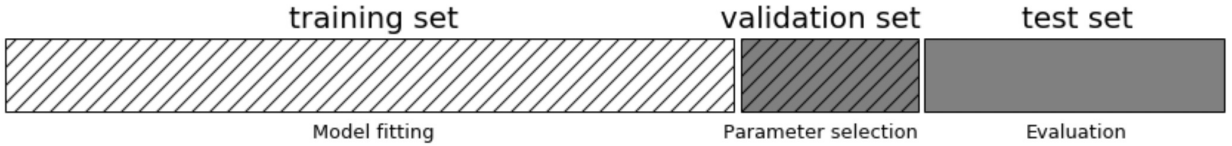
For statistically strong results:

Small training dataset: cross validation



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

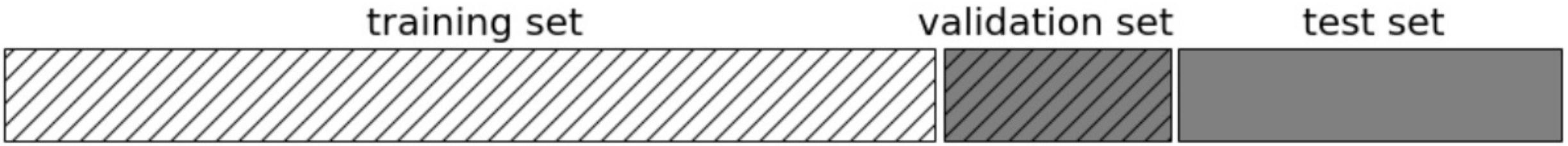
Else: train/validation split



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Train/Validation/Test Split

- Split dataset into 3 sets: “train”, “validation”, and “test” splits
  - e.g., 60%/20%/20% train/val/test split



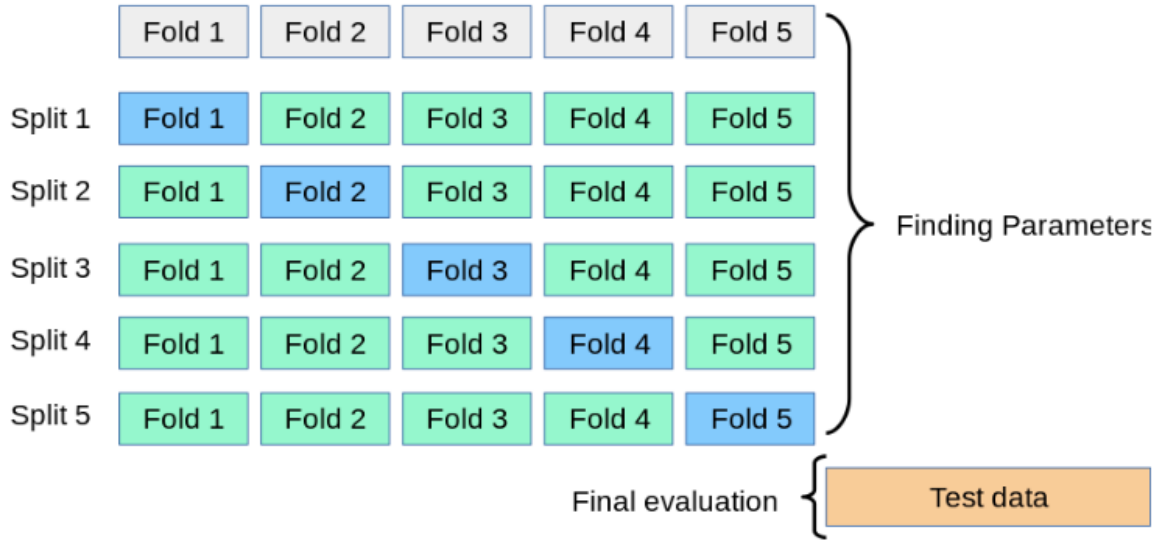
- **Hyperparameter selection**: test variants on validation set to identify best set of hyperparameters
- **Final model**: train a new model on data in the training AND validation splits using the best hyperparameters from hyperparameter selection

https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Hyperparameter Tuning: Split Training Set So It Can Be Used to Test Different Hyperparameters

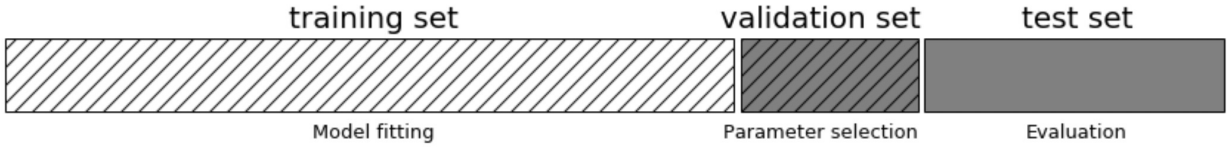
For statistically strong results:

Small training dataset: cross validation



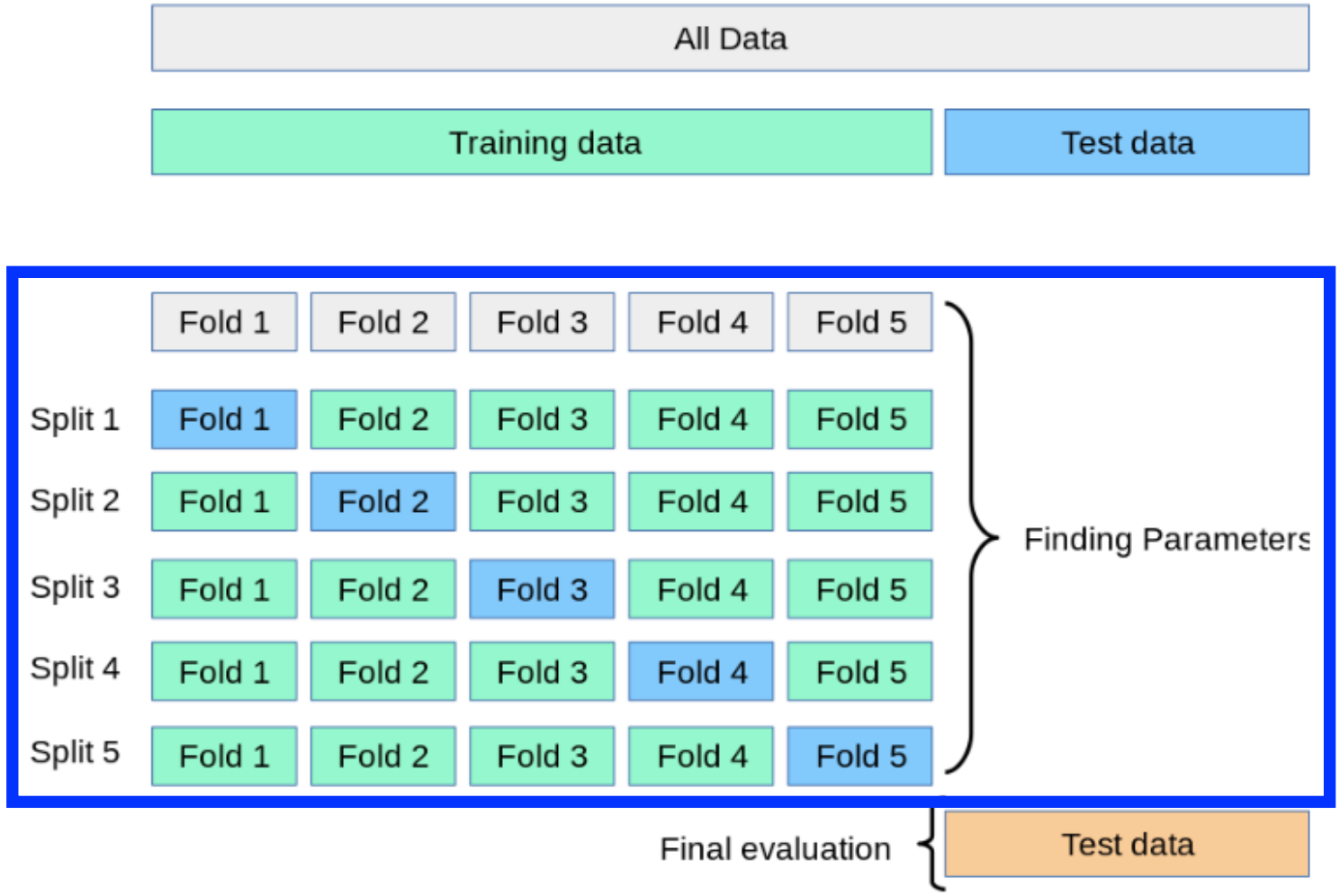
https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Else: train/validation split



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

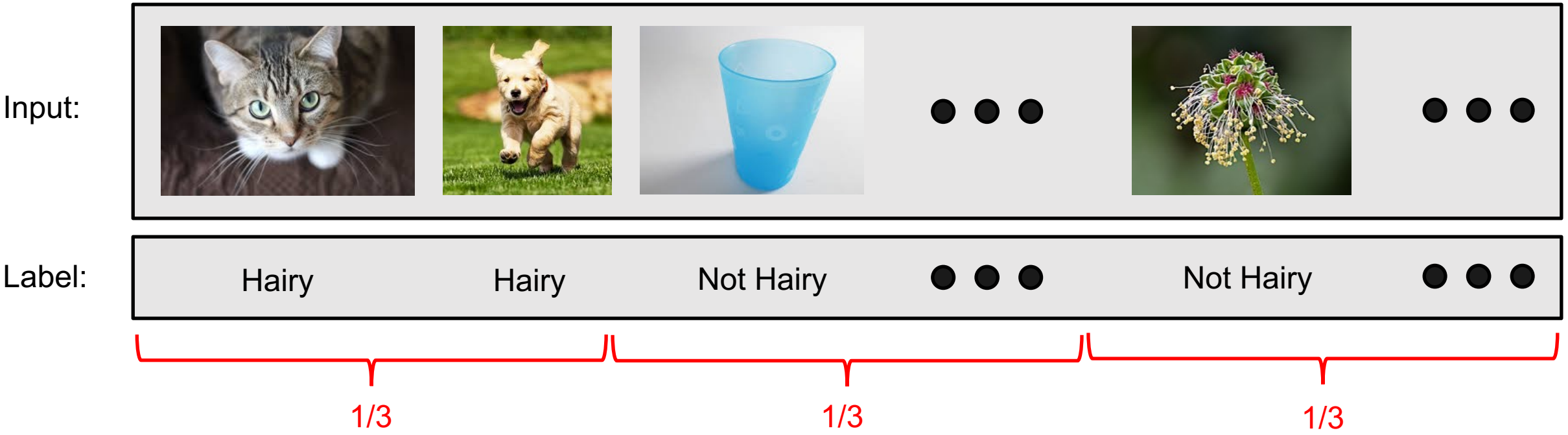
Cross-Validation: Limit Influence of Dataset Split



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Cross-Validation: Limit Influence of Dataset Split

e.g., 3-fold cross-validation on training data

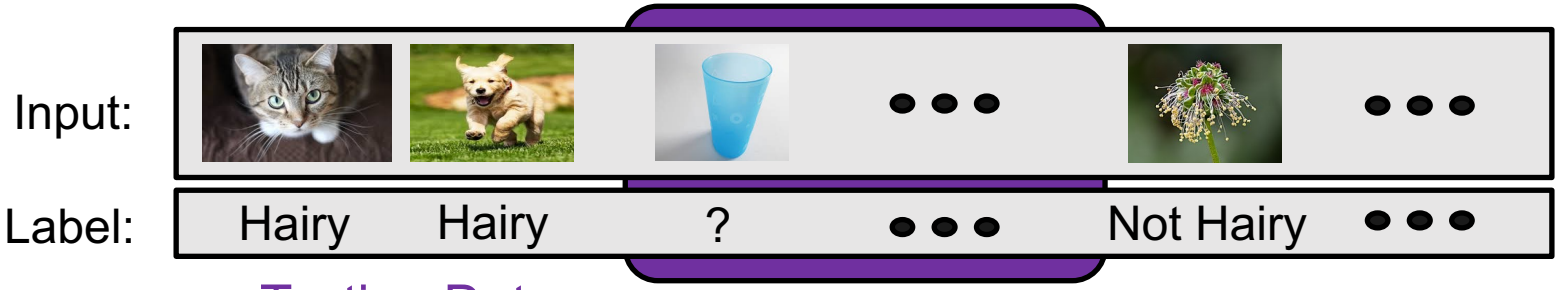


Cross-Validation: Limit Influence of Dataset Split

e.g., 3-fold cross-validation on training data

Fold 1:
- train on k-1 partitions
- test on k partitions

Fold 2:
- train on k-1 partitions
- test on k partitions

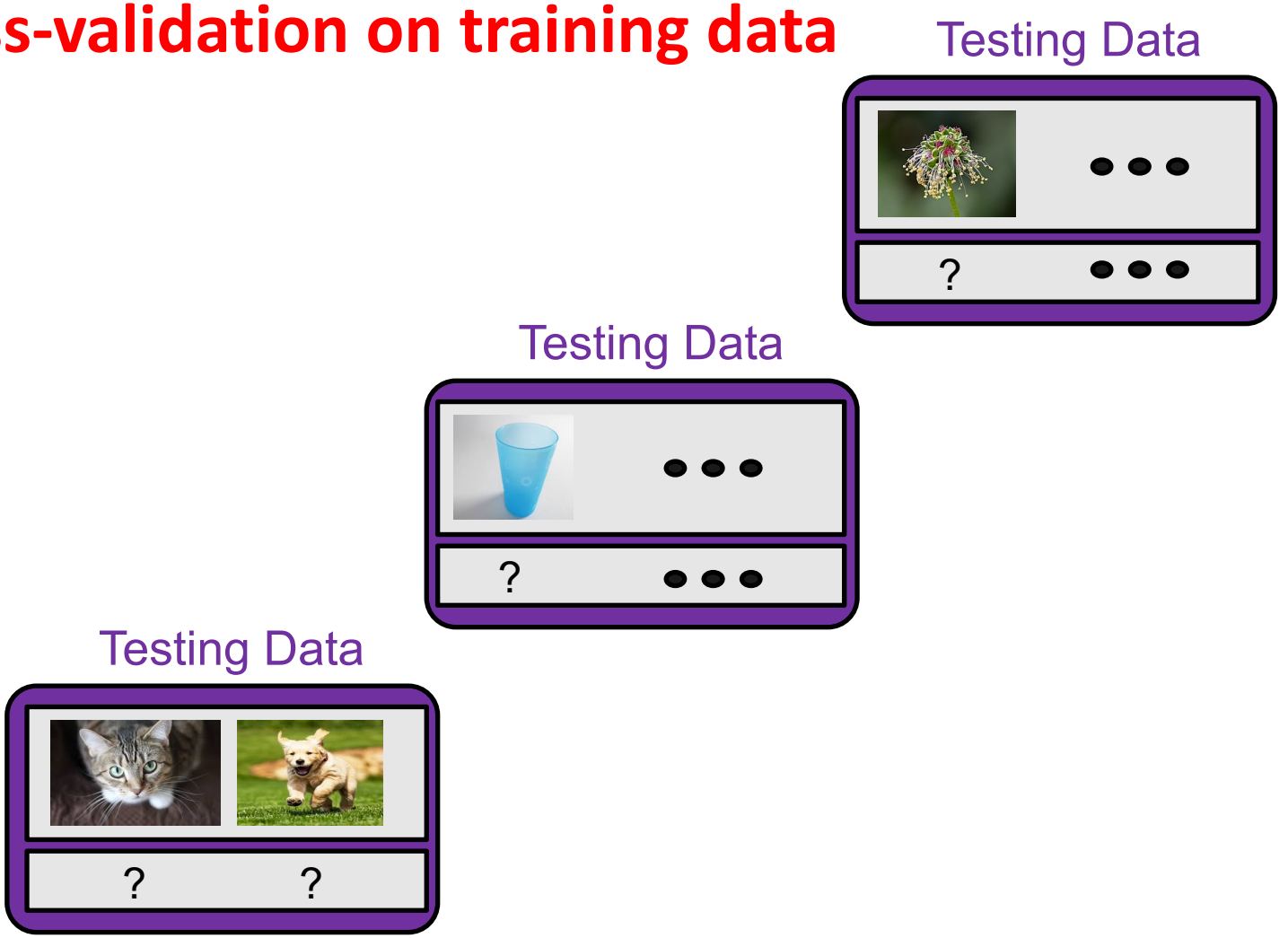


Fold 3:
- train on k-1 partitions
- test on k partitions

Cross-Validation: Limit Influence of Dataset Split

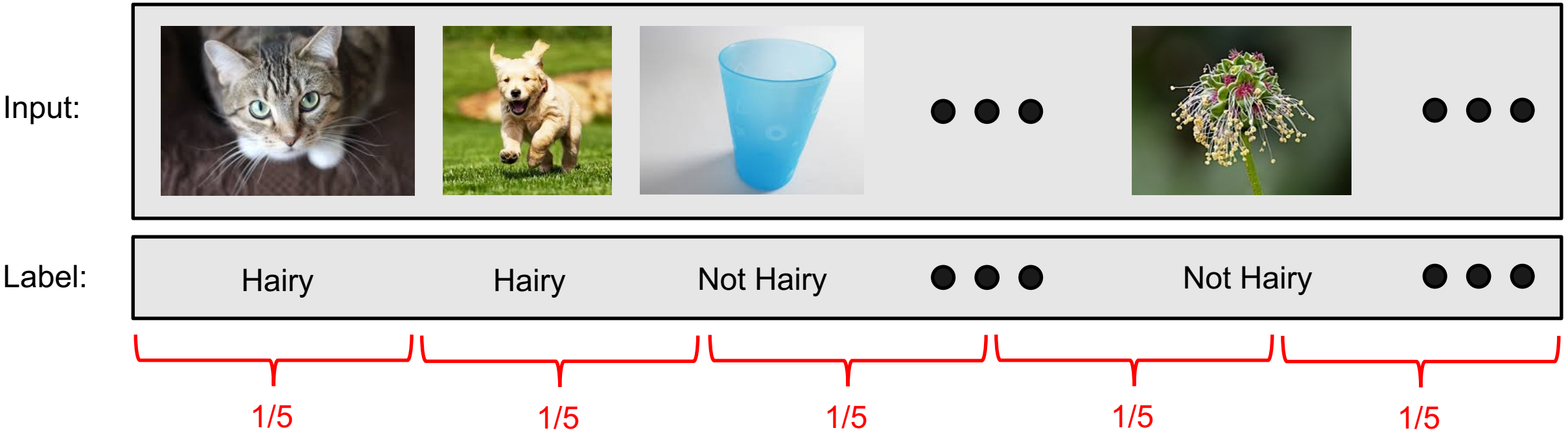
e.g., 3-fold cross-validation on training data

Model performance:
performance across all folds of “test” data



Cross-Validation: Limit Influence of Dataset Split

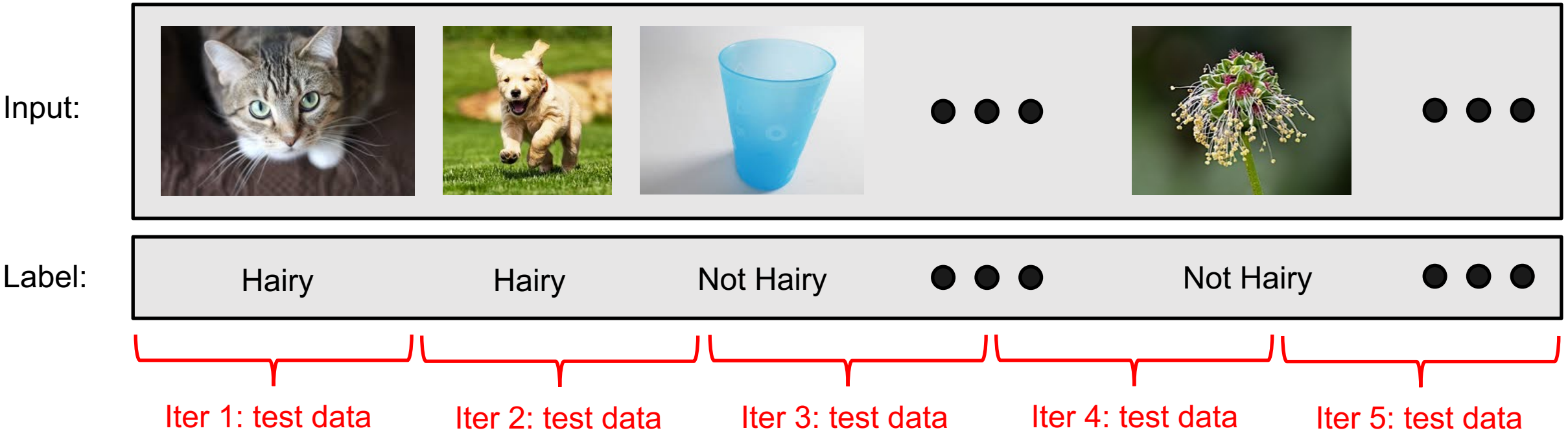
e.g., 5-fold cross-validation on training data



How many partitions of the data to create?

Cross-Validation: Limit Influence of Dataset Split

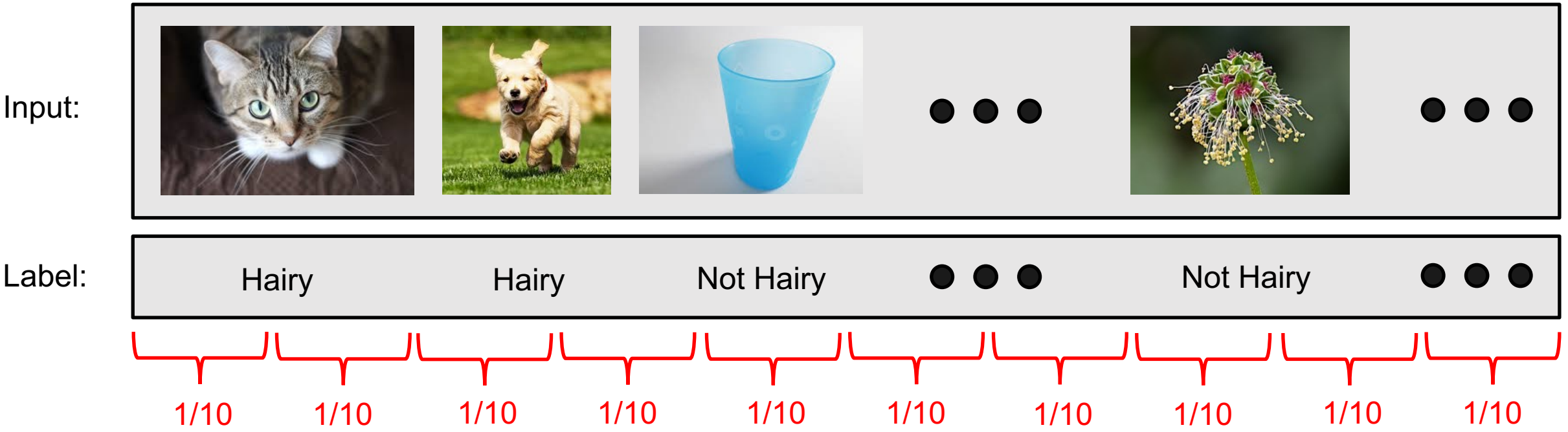
e.g., 5-fold cross-validation on training data



Run 5 iterations of train & test

Cross-Validation: Limit Influence of Dataset Split

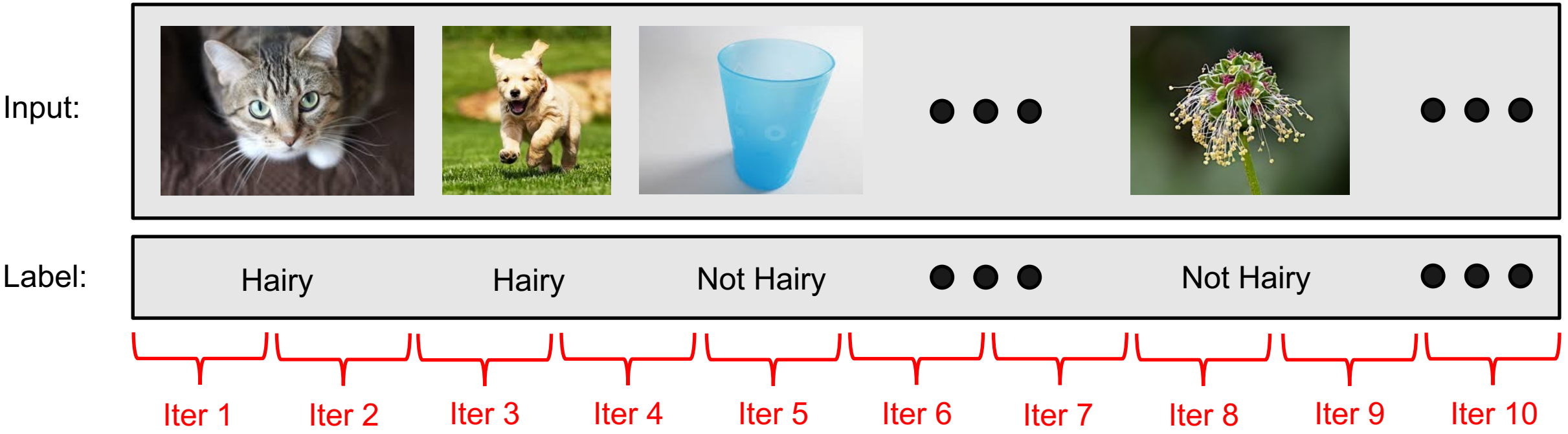
e.g., 10-fold cross-validation on training data



How many partitions of the data to create?

Cross-Validation: Limit Influence of Dataset Split

e.g., 10-fold cross-validation on training data



Run 10 iterations of train & test

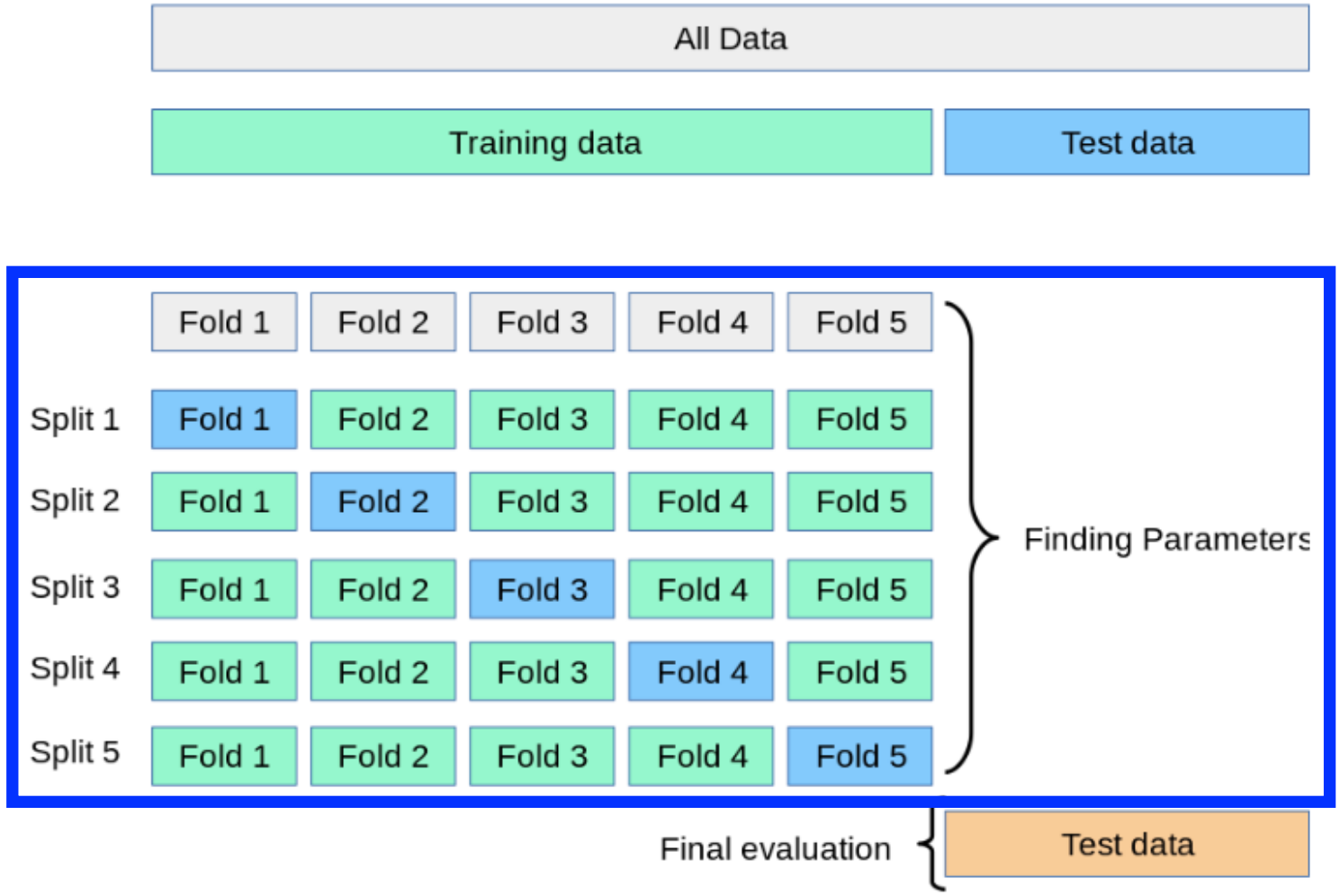
Cross-Validation: Limit Influence of Dataset Split

e.g., k-fold cross-validation on training data

What are the (dis)advantages of using larger values for "k"?

Cross-Validation: Limit Influence of Dataset Split

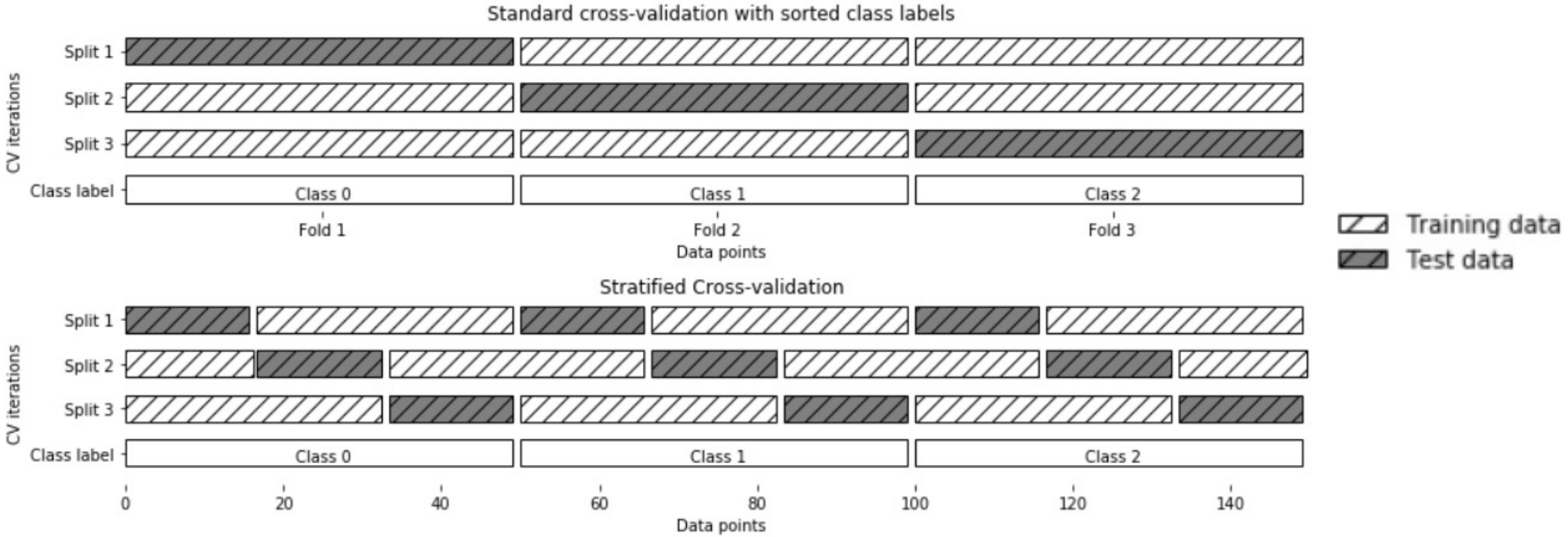
Select the hyperparameters that lead to the best results overall across all the folds



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Stratified Dataset Splits

- For imbalanced datasets, preserve frequencies of each category in each split; e.g.,

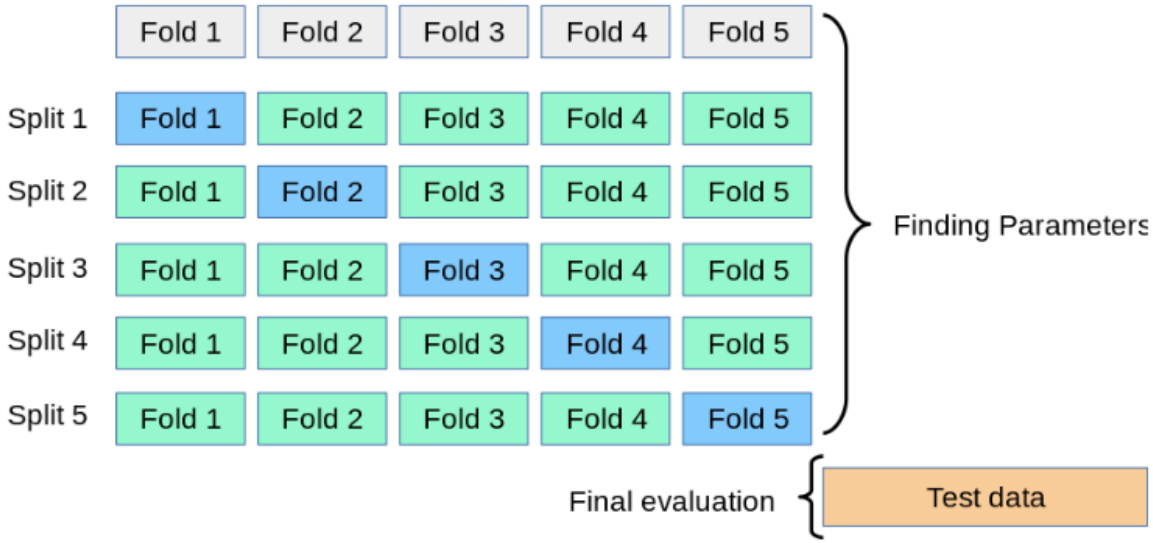


https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Summary: Hyperparameter Tuning Approaches

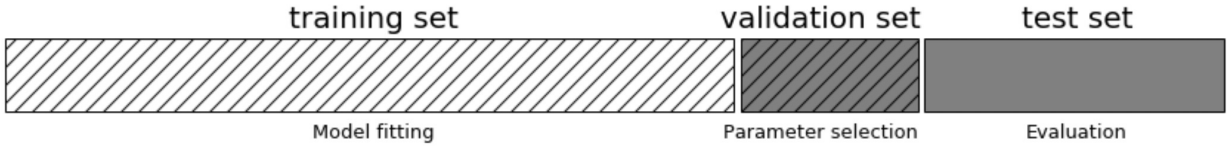
For statistically strong results:

Small training dataset: cross validation



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Else: train/validation split



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Today's Topics

- Universal approximation theorem vs No Free Lunch theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Training neural networks: hardware & software
- Programming tutorial

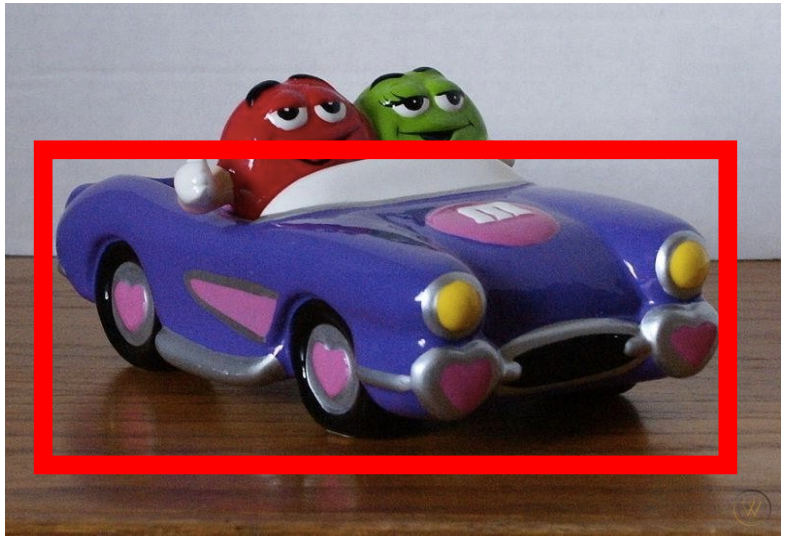
Recall: Key Ingredients for Success With Neural Networks

An **algorithm** learns from **data**
on a **processor** the patterns that
will be used to make a prediction

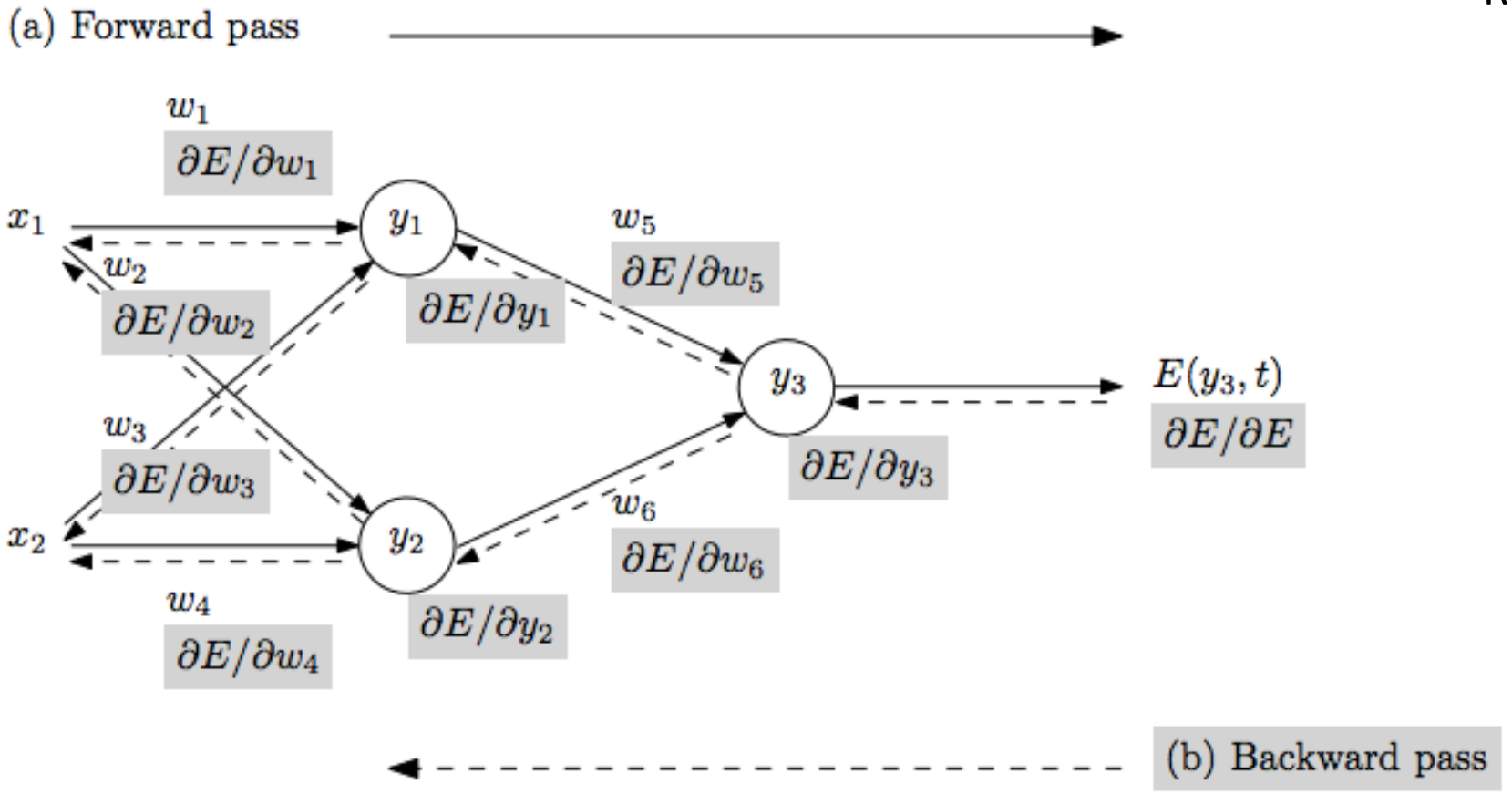
Analogous to a Love Story of Partnering Up and Road Tripping Somewhere

Recall: Key Ingredients for Success With Neural Networks

Key Issue: How Long Will It Take to Get There?



Challenge: Training Neural Network **Requires Many Computations** (e.g., millions of model parameters)

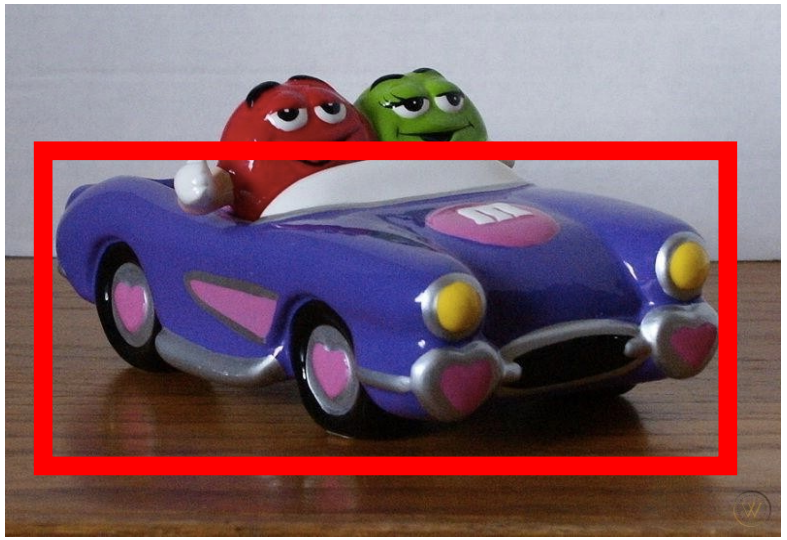


- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

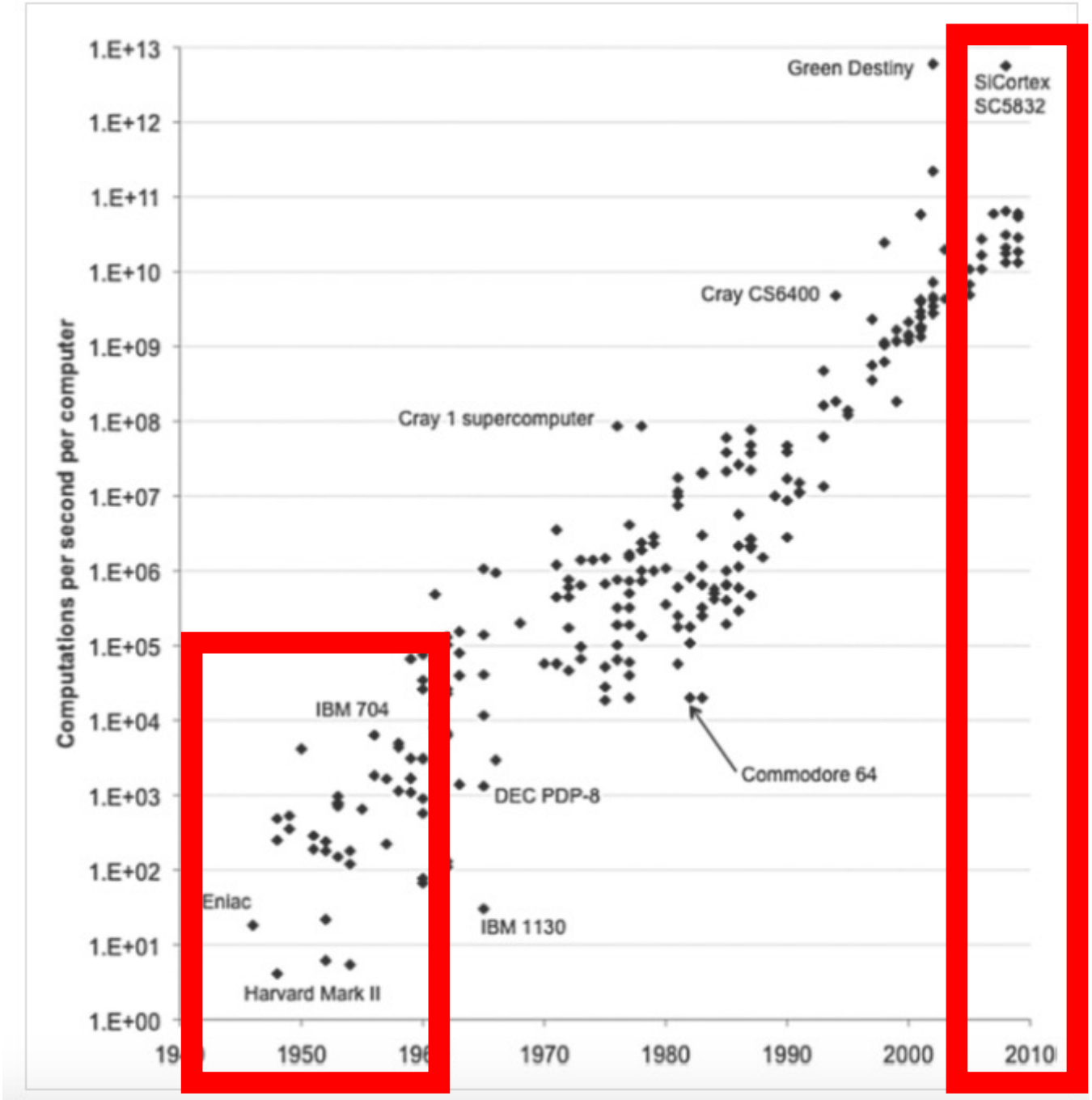
Idea: Better Hardware

Idea: Train Algorithms Using
GPUs (think Porsche) Instead of CPUs (think Golf Cart)



Historical Perspective

- Better Hardware
  - e.g., faster processing -- GPUs



https://ourworldindata.org/technological-progress

Historical Perspective

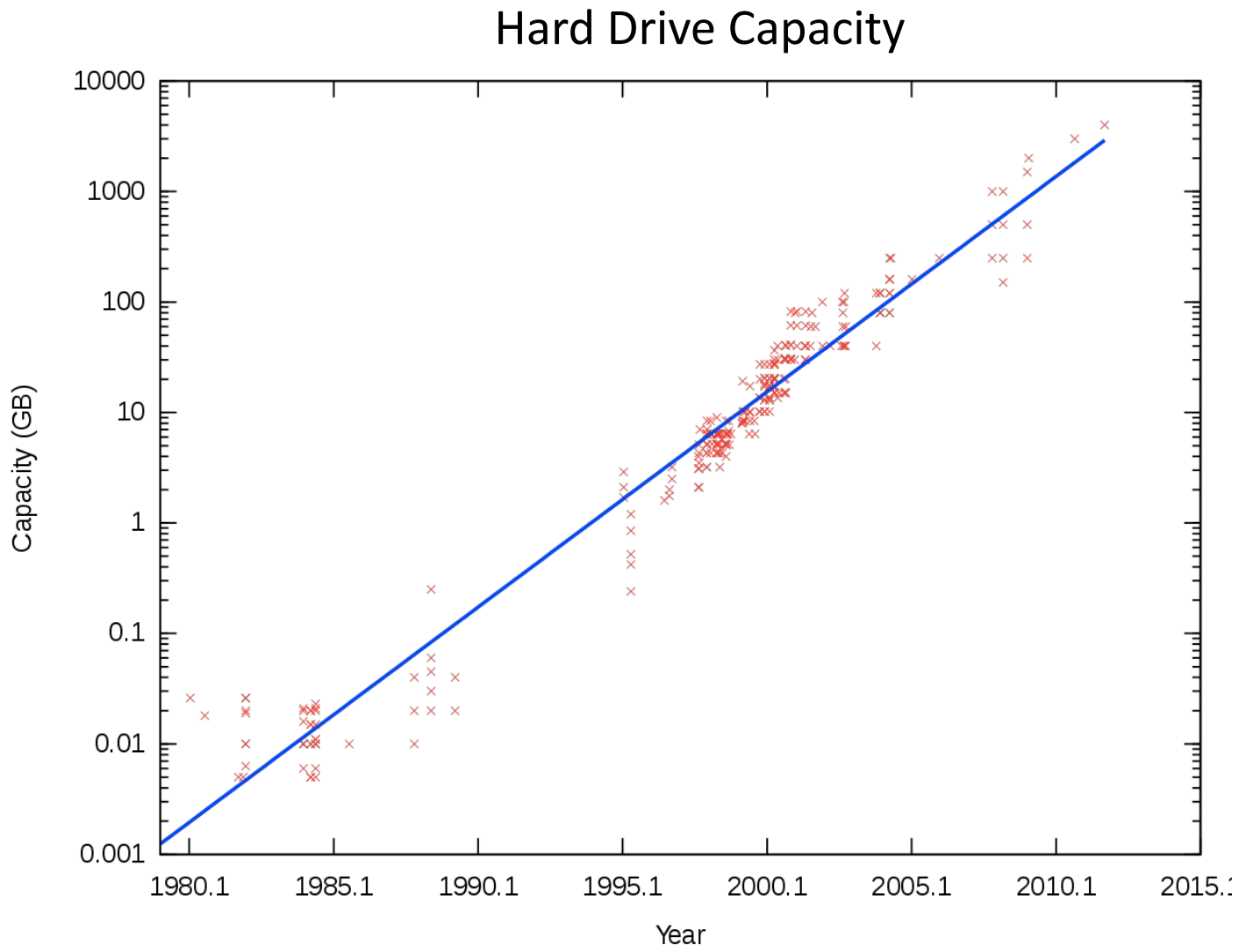
- Better Hardware
  - e.g., faster processing -- GPUs
  - e.g., more data storage

The IBM Model 350 disk file with a storage space of 5MB fro 1956 nd a Micro SD Card

https://ourworldindata.org/technological-progress

Historical Perspective

- Better Hardware
  - e.g., faster processing -- GPUs
  - e.g., more data storage



https://ourworldindata.org/technological-progress

Hardware: CPU versus GPU

Spot the CPU!

(central processing unit)

This image is licensed under CC-BY 2.0

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture08.pdf

Hardware: CPU versus GPU

Spot the GPUs!

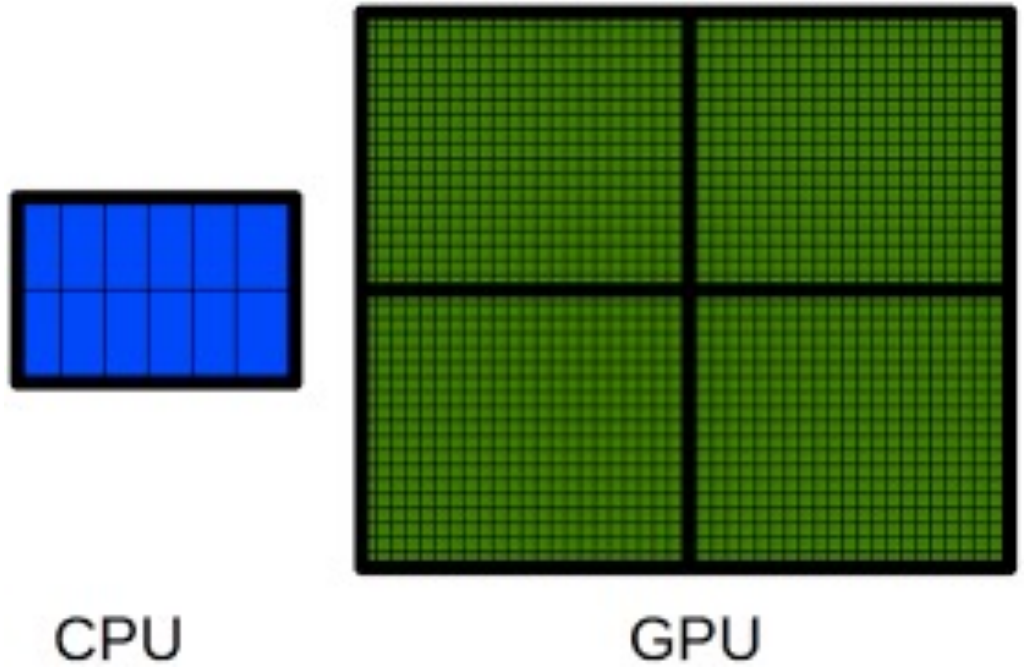
(graphics processing unit)

This image is in the public domain

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture08.pdf

Hardware: CPU versus GPU

- Graphical Processing Units: accelerates computational workloads due to MANY more processing cores



https://www.researchgate.net/figure/The-main-difference-between-CPUs-and-GPUs-is-related-to-the-number-of-available-cores-A_fig7_273383346

Hardware: Training Models with GPUs

If you aren't careful, training can bottleneck on reading data and transferring to GPU!

Solutions:

- Read all data into RAM
- Use SSD instead of HDD
- Use multiple CPU threads to prefetch data

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture08.pdf

GPU Machines: Rent Versus Buy?

Rent from Cloud

(e.g., Microsoft Azure):

Instance	Core(s)	RAM	Temporary storage	GPU	Pay as you go with AHB
ND96asr A100 v4	96	900 GiB	6,500 GiB	8x A100 (NVlink)	$27.197/hour

Buy:

Lambda Bare Metal

- 4-8x NVIDIA A100 SXM4 GPUs
- Install in your Datacenter or Lambda Colo
- Customize CPU, RAM, Storage & Network
- Delivered in 2-4 weeks

Starting at

$ 89,283.00

Rise of “Deep Learning” Open Source Platforms

Motivation:

Can run on GPUs:

OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]

Simplifies using popular neural network architectures:

Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

Rise of “Deep Learning” Open Source Platforms

torch (Collobert et al.)

theano (Bastien et al.)

Caffe (Jia et al.)

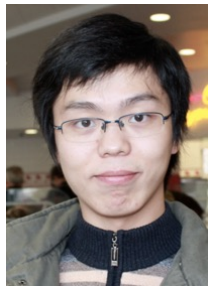
Keras

TensorFlow™ (Abadi et al.)

PYTORCH (Paszke et al.)

2011 2012 2013 2014 2015 2016 2017

Université de Montréal



idiap RESEARCH INSTITUTE

Berkeley UNIVERSITY OF CALIFORNIA

Google

Popular Options Today

Rise of “Deep Learning” Open Source Platforms

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)	Actively Developed
roNNie.ai	Kevin Lok	MIT license	Yes	Linux, macOS, Windows	Python	Python			Yes		Yes	Yes	Yes			
BigDL	Jason Dai	Apache 2.0	Yes	Apache Spark	Scala	Scala, Python			No		Yes	Yes	Yes			
Caffe	Berkeley Vision and Learning Center	BSD	Yes	Linux, macOS, Windows[2]	C++	Python, MATLAB, C++	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?	
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	On roadmap[5]	Yes[6][7]	Computational Graph	Yes[8]	Yes	Yes	Yes	Yes[9]	
Chainer	Preferred Networks	MIT license	Yes	Linux, macOS, Windows		Python	No	No[10][11]	Yes	Yes	Yes	Yes	Yes			
Darknet	Joseph Redmon	Public Domain	Yes	Cross-Platform	C	C, Python	Yes	No[12]	Yes	Yes						
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
DataMelt (DMelt)	S.Chekanov	Freemium	Yes	Cross-Platform	Java	Java	No	No	No	No	No	No	No	No	No	
DyNet	Carnegie Mellon University	Apache 2.0	Yes	Linux, macOS, Windows		C++, Python		No[13]	Yes	Yes	Yes					
Intel Data Analytics Acceleration Library	Intel	Apache License 2.0	Yes	Linux, macOS, Windows on Intel CPU[14]	C++, Python, Java	C++, Python, Java[14]	Yes	No	No	Yes	No		Yes		Yes	
Intel Math Kernel Library	Intel	Proprietary	No	Linux, macOS, Windows on Intel CPU[15]		C[16]	Yes[17]	No	No	Yes	No	Yes[18]	Yes[18]		No	
Keras	François Chollet	MIT license	Yes	Linux, macOS, Windows	Python	Python, R	Only if using Theano as backend	Can use Theano or Tensorflow as backends	Yes	Yes	Yes[19]	Yes	Yes	Yes	Yes[20]	
MATLAB + Neural Network Toolbox	MathWorks	Proprietary	No	Linux, macOS, Windows	C, C++, Java, MATLAB	MATLAB	No	No	Train with Parallel Computing Toolbox and generate CUDA code with GPU Coder[21]	No	Yes[22][23]	Yes[22]	Yes[22]	No	With Parallel Computing Toolbox[24]	
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[25]	Yes	Windows, Linux[26] (macOS via Docker on roadmap)	C++	Python (Keras), C++, Command line,[27] BrainScript[28] (.NET on roadmap[29])	Yes[30]	No	Yes	Yes	Yes[31]	Yes[32]	Yes[32]	No[33]	Yes[34]	
Apache MXNet	Apache Software Foundation	Apache 2.0	Yes	Linux, macOS, Windows,[35][36] AWS, Android,[37] iOS, JavaScript[38]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap[39]	Yes	Yes[40]	Yes[41]	Yes	Yes	Yes	Yes[42]	
Neural Designer	Artelnics	Proprietary	No	Linux, macOS, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?	
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	Yes	?	?	No	No	No	?	
PaddlePaddle	Baidu	Apache License	Yes	Linux, macOS, Windows	C++, Python	Python	No	Yes	Yes	Yes	Yes	Yes	Yes	?	Yes	
PlaidML	Vertex.AI	AGPL3	Yes	Linux, macOS, Windows	C++, Python	Keras, Python, C++, C	No	Yes	Yes	Yes		Yes	Yes	?	Yes	
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan	BSD	Yes	Linux, macOS, Windows	Python, C, CUDA	Python	Yes	Via separately maintained package[43][44][45]	Yes	Yes	Yes	Yes	Yes		Yes	
Apache SINGA	Apache Incubator	Apache 2.0	Yes	Linux, macOS, Windows	C++	Python, C++, Java	No	No	Yes	?	Yes	Yes	Yes	Yes	Yes	
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, macOS, Windows,[46] Android	C++, Python, CUDA	Python (Keras), C/C++, Java, Go, R[47], Julia, Swift	No	On roadmap[48] but already with SYCL[49] support	Yes	Yes[50]	Yes[51]	Yes	Yes	Yes	Yes	
TensorLayer	Hao Dong	Apache 2.0	Yes	Linux, macOS, Windows,[52] Android	C++, Python,	Python	No	On roadmap[48] but already with SYCL[49] support	Yes	Yes[53]	Yes[54]	Yes	Yes	Yes	Yes	
Theano	Université de Montréal	BSD	Yes	Cross-platform	Python	Python (Keras)	Yes	Under development[55]	Yes	Yes[56][57]	Through Lasagne's model zoo[58]	Yes	Yes	Yes	Yes[59]	No
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD	Yes	Linux, macOS, Windows,[60] Android,[61] iOS	C, Lua	Lua, LuaJIT,[62] C, utility library for C++/OpenCL[63]	Yes	Third party implementations[64][65]	Yes[66][67]	Through Twitter's Autograd[68]	Yes[69]	Yes	Yes	Yes	Yes[70]	
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, macOS, Linux, Cloud computing	C++, Wolfram Language, CUDA	Wolfram Language	Yes	No	Yes	Yes	Yes[71]	Yes	Yes	Yes	Under Development	
VerAI	VerAI	Proprietary	No	Linux, Web-based	C++,Python, Go, Angular	Graphical user interface, cli	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

Excellent comparison: https://skymind.ai/wiki/comparison-frameworks-dl4j-tensorflow-pytorch

Excellent comparison: https://arxiv.org/pdf/1511.06435.pdf

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

Today's Topics

- Universal approximation theorem vs No Free Lunch theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Training neural networks: hardware & software
- Programming tutorial

Today's Topics

- Universal approximation theorem vs No Free Lunch theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Training neural networks: hardware & software
- Programming tutorial

The End