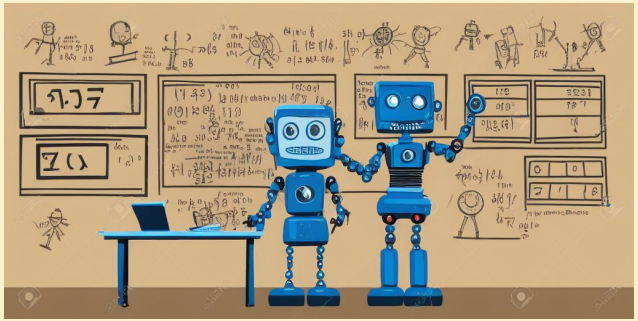


# Advanced Workshop on Machine Learning

## Lecture 5: Ensemble Methods

Danylo Vashchilenko

2022

# Agenda

**Part 1**

Why ensembles work?

Types of ensembles

Case Study: Gaussian Circles

**Part 2**

Case Study: Tesla AutoPilot

Assignment 5

**Part 3**

Course Summary

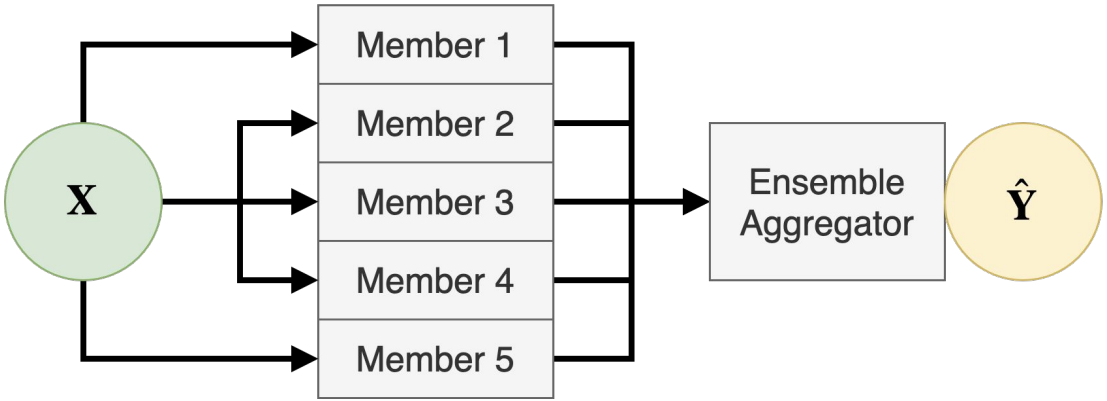
(Image by a Stable Diffusion model)

# Ensemble Inference

Ensemble is a collection of models whose output is aggregated together.

Under certain conditions, aggregated output is better than any member output.

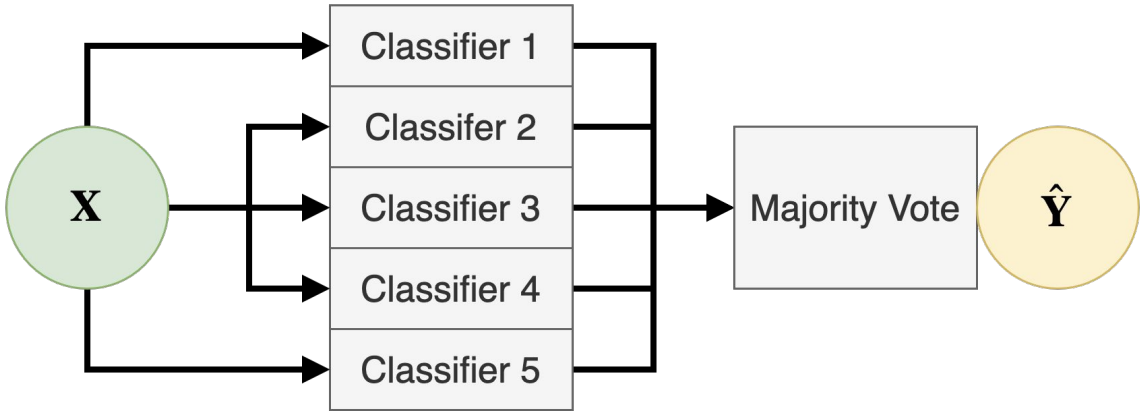
Members may or may not communicate.



# Binary Classification

Ensemble output is the class predicted by the majority of member classifiers.

In case of probabilities, the average probability will be consistent with discrete majority vote.

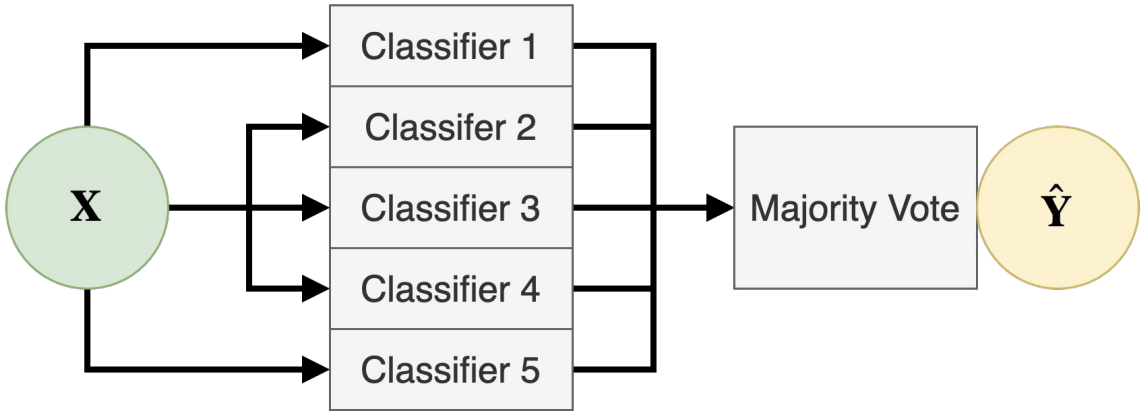


# Ensemble Properties

Ensemble performance depends on:

1. Member strength
2. Ensemble size
3. Member independence

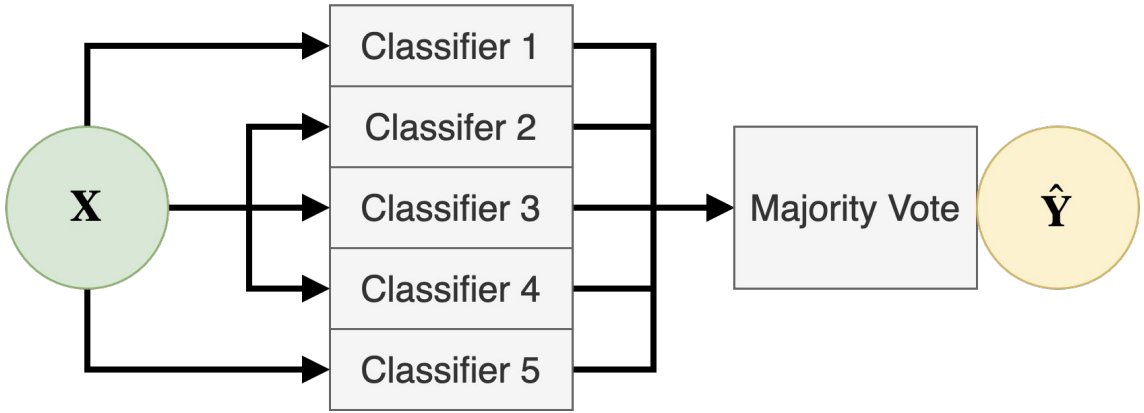
Ensemble performance may or may not be better than its best member.



# Ensemble Performance

**Method 1: Analytical**
Mathematical analysis of the statistical distribution and its properties.

**Assumptions:**
Binary classification



# Evaluation Method 1: Analytical

**Member i**

$$x \sim p_{data}$$

$$e_i(x) \in \mathbb{R}^n \rightarrow \{0, 1\}$$

$$e_i(x) \sim \text{Bernoulli}(a_i)$$

$$\mathop{\mathbb{E}}_{x}[\, e_i(x) \,] = a_i$$

**Ensemble of s members:**

$$e(x) = \lceil \sum_{i=0}^{s} \frac{e_i(x)}{s} \rfloor$$

$$\mathop{\mathbb{E}}_{x}[\, e(x) \,] = a_e$$

**Assumption**

Member independence

# Evaluation Method 2: Monte Carlo

Approximate the distribution properties
using a finite batch of random samples.

**Steps (given member accuracies)**

1. Sample the errors for each member
2. Compute the ensemble error
3. Repeat 1-2 for N iterations
4. Average the ensemble errors

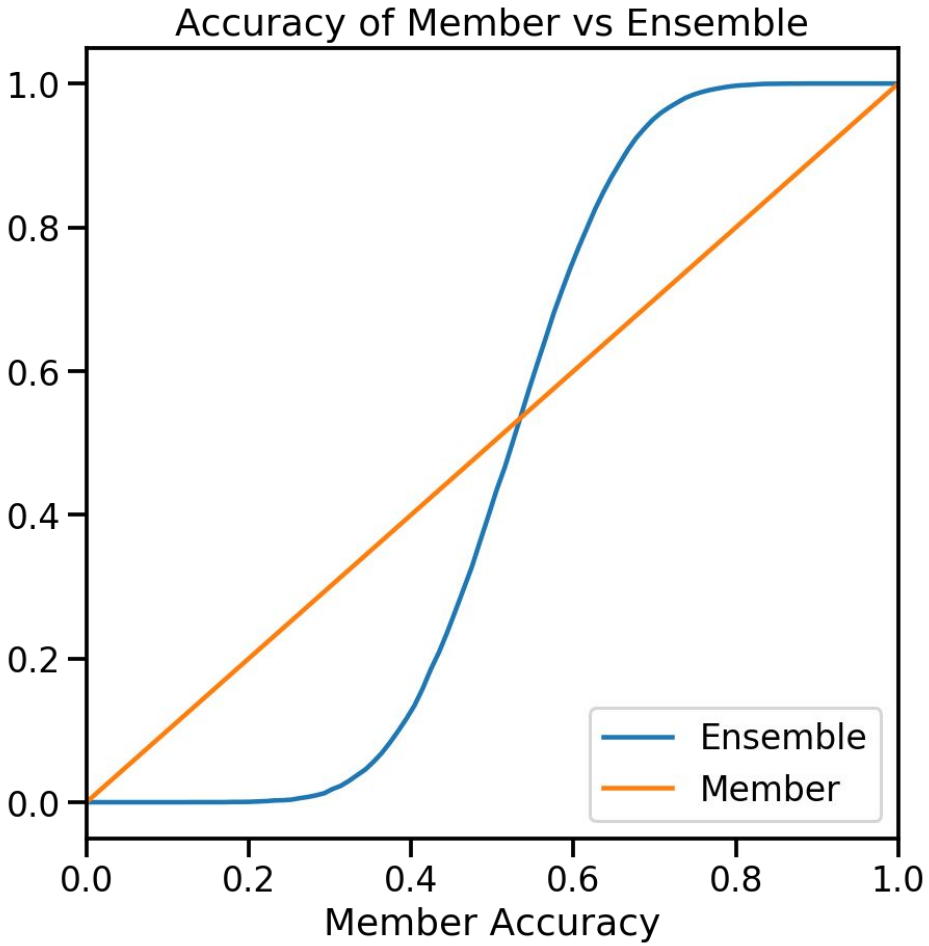
See MonteCarlo.ipynb for NumPy implementation.

# Member Strength

**Simulation**
Ensemble of 20 independent binary classifiers.

**Inference**
Ensemble strength increases with member strength.

Ensemble members must be better than random.

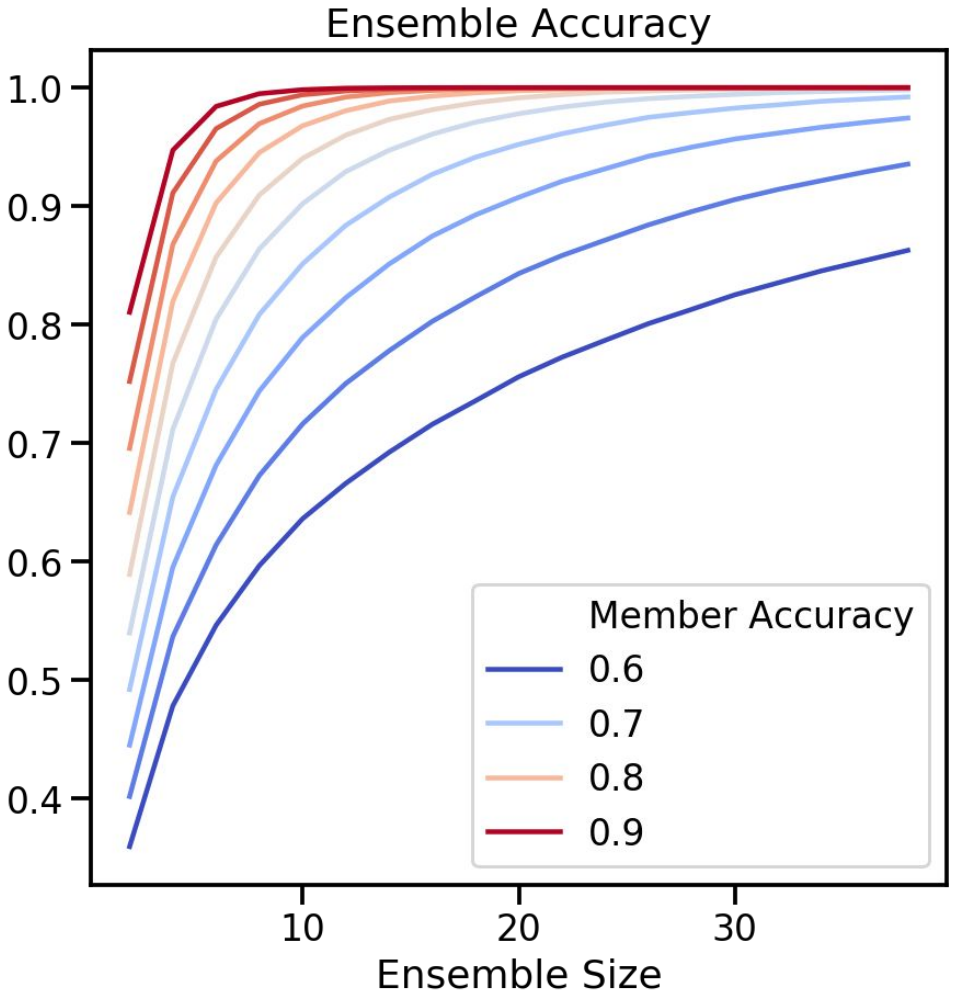


# Ensemble Size

**Simulation**
Ensemble of independent binary classifiers.

**Inference**
Ensemble strength increases with size.

Marginal value of an additional member decreases with its accuracy and ensemble size.



# Member Independence

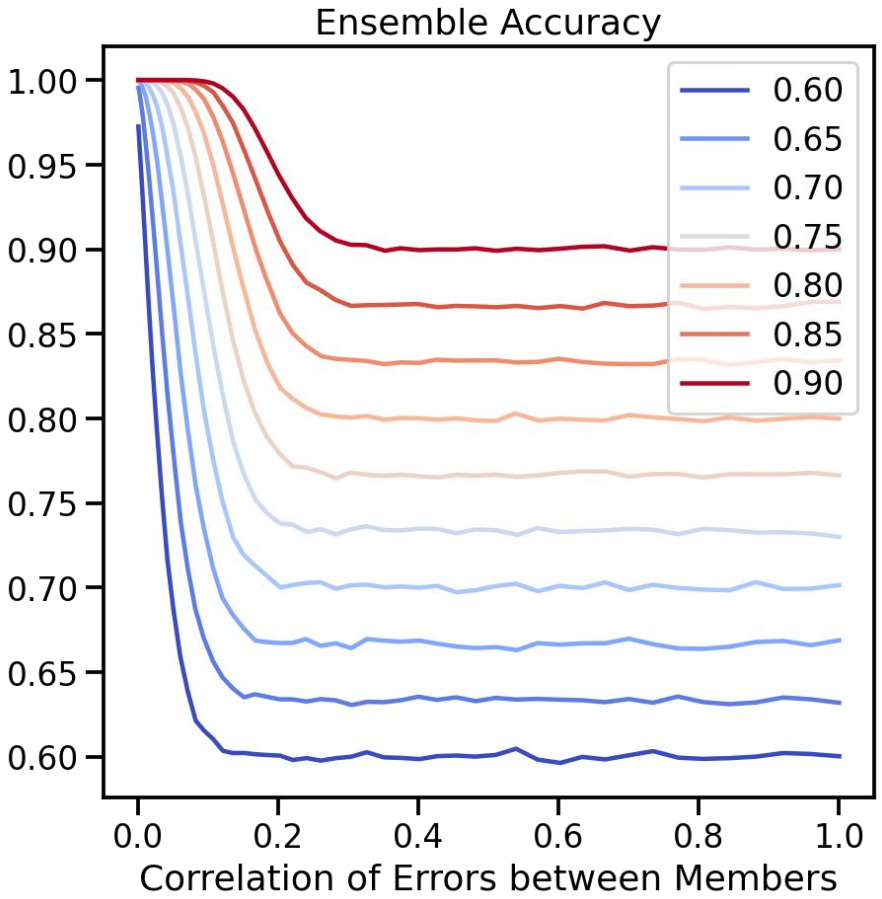
**Simulation**

Ensemble of 100 binary classifiers with correlated errors.

**Inference**

Ensemble strength decreases with correlation between members.

Sensitivity to correlation decrease with member strength.

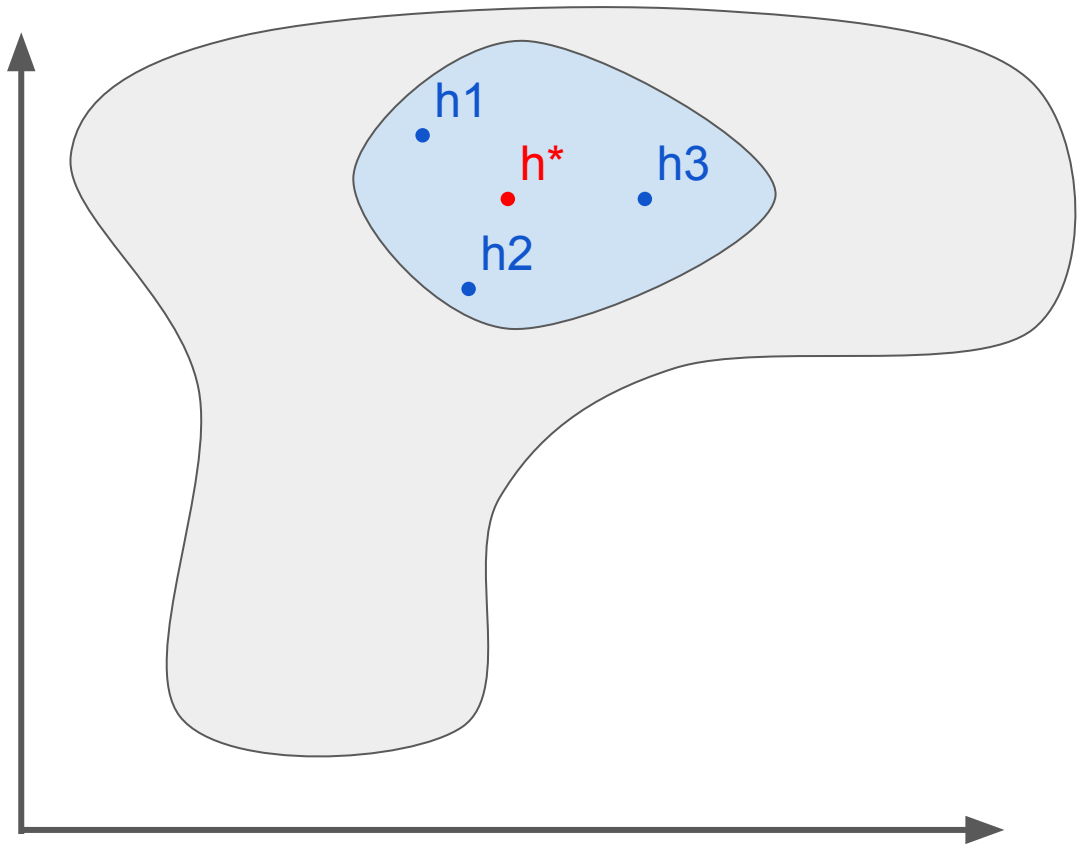



See MonteCarlo.ipynb for NumPy implementation.

# Statistical

When datasets are small relative to the hypothesis space, multiple hypotheses may minimize the estimate of the cost.

However, under all possible datasets, there should be a single optimal hypothesis.

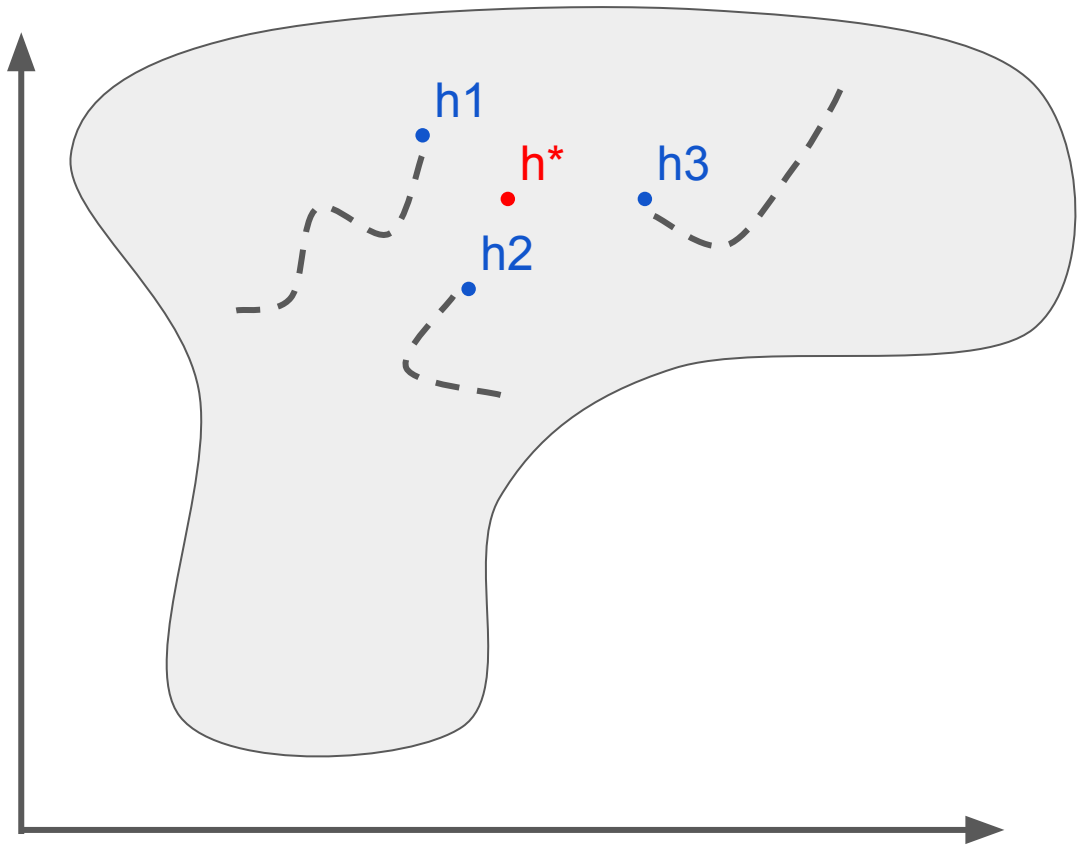



Diettrich, T. G. (2000)

# Computational

Iterative learning algorithms (e.g. gradient descent) may converge to different local minima.

An average of multiple hypotheses may be close to the global minimum.

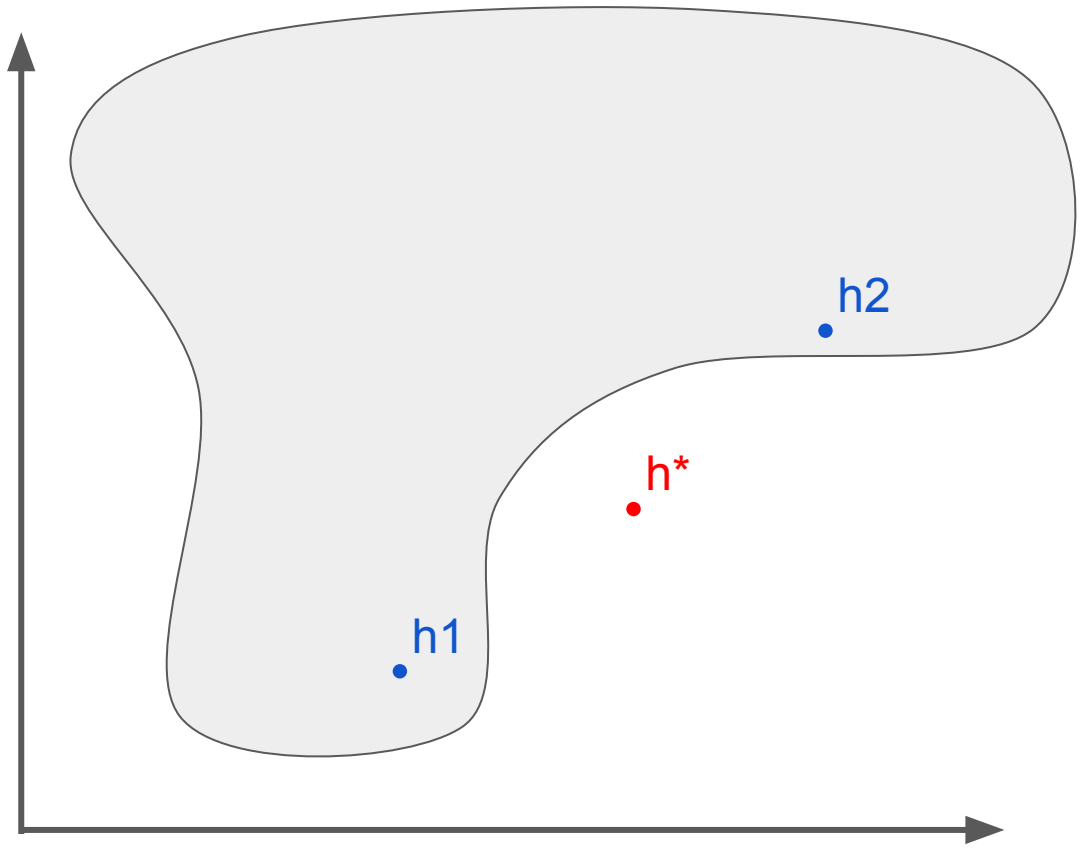



Diettrich, T. G. (2000)

# Representational

The optimal hypothesis may be outside of the effective hypothesis space that is searched over using a finite dataset.

An average of multiple hypotheses may be outside of the effective hypothesis space and closer to the optimal hypothesis.



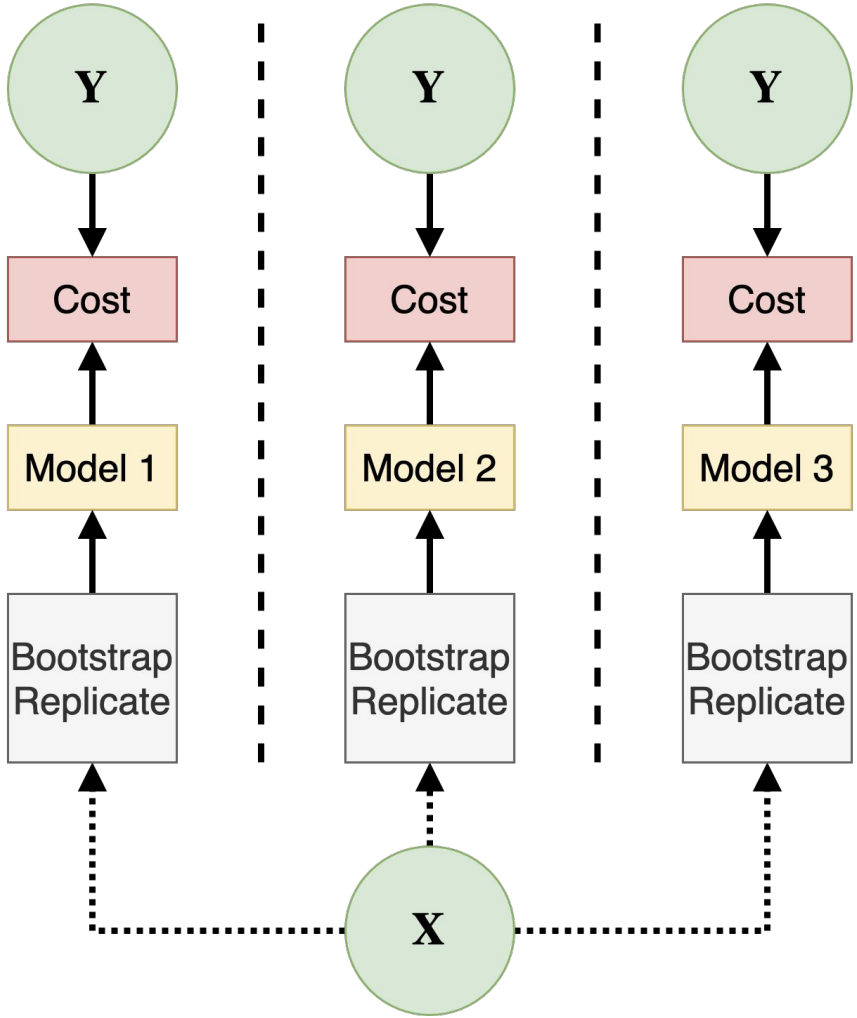

Diettrich, T. G. (2000)

# Bagging

Bootstrap replicate = random resampling of the dataset with replacement.

Bootstrap aggregation = an ensemble of models trained on different bootstrap replicates.

On average ~60% of data present in every replicate.

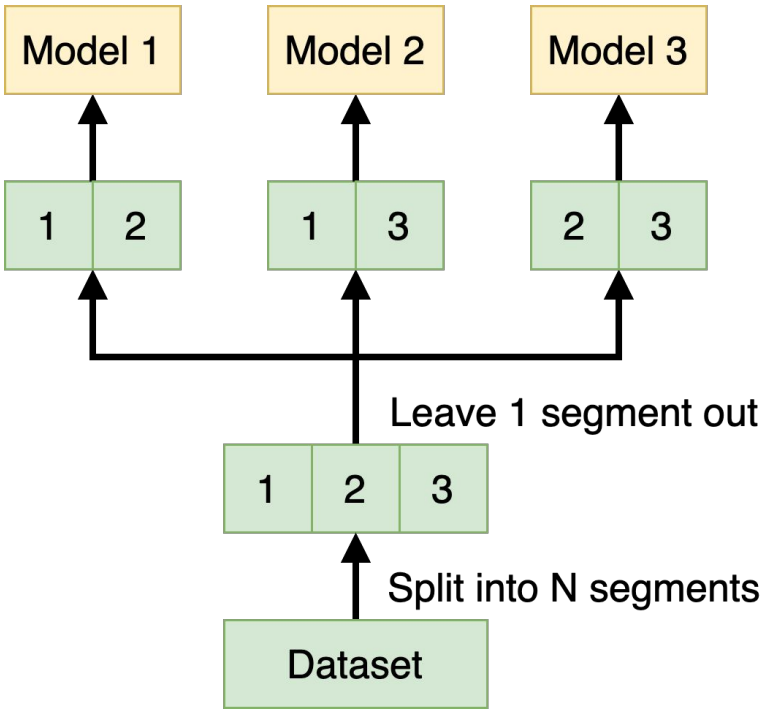



Diettrich, T. G. (2000)

# Cross-validated Committee

Segments = N disjoint subsets of the dataset.

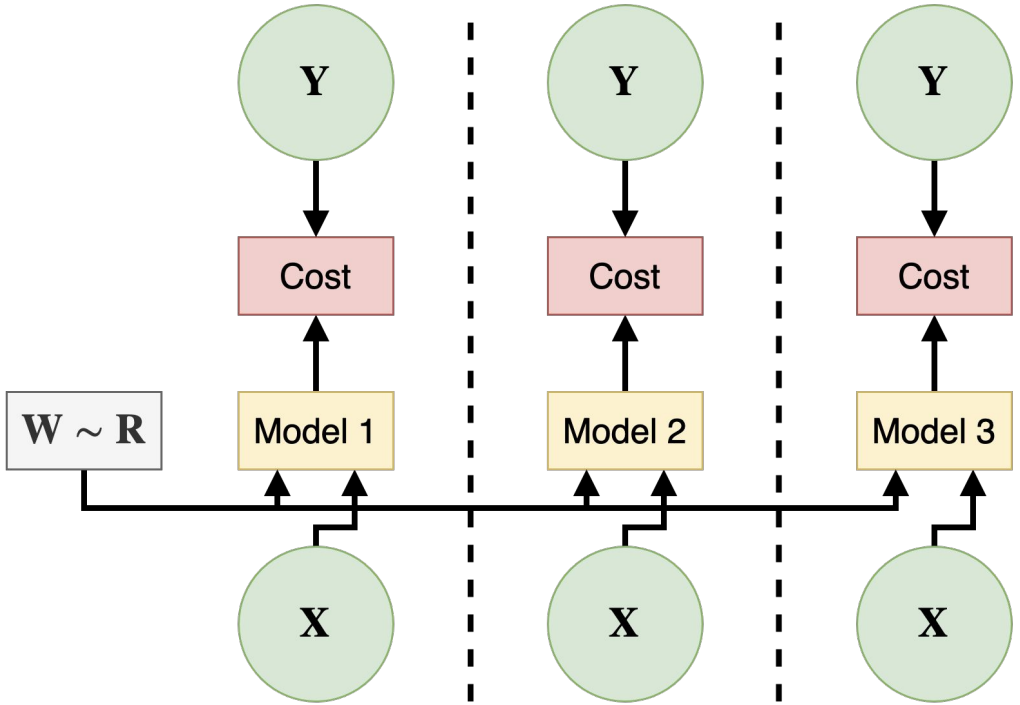
Cross-validated committee = an ensemble of N-1 models trained on the dataset with different segments missing.



# Noise Injection: Initialization

Create multiple neural networks with different initial parameters.

After training, the networks will arrive at different hypotheses.




Diettrich, T. G. (2000)

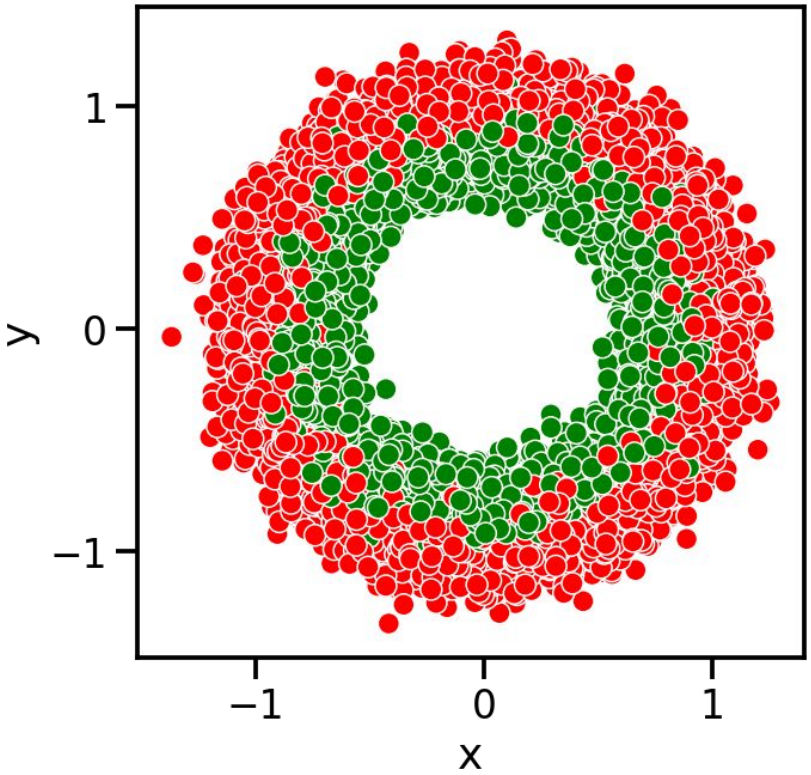
# Case Study: Gaussian Circles

**Dataset**

Collection of 10K points in 2D space arranged as two circles with Gaussian noise.

**Task**

Classify a point as either green or red given its coordinates.




See GaussianCircles.ipynb for implementation.

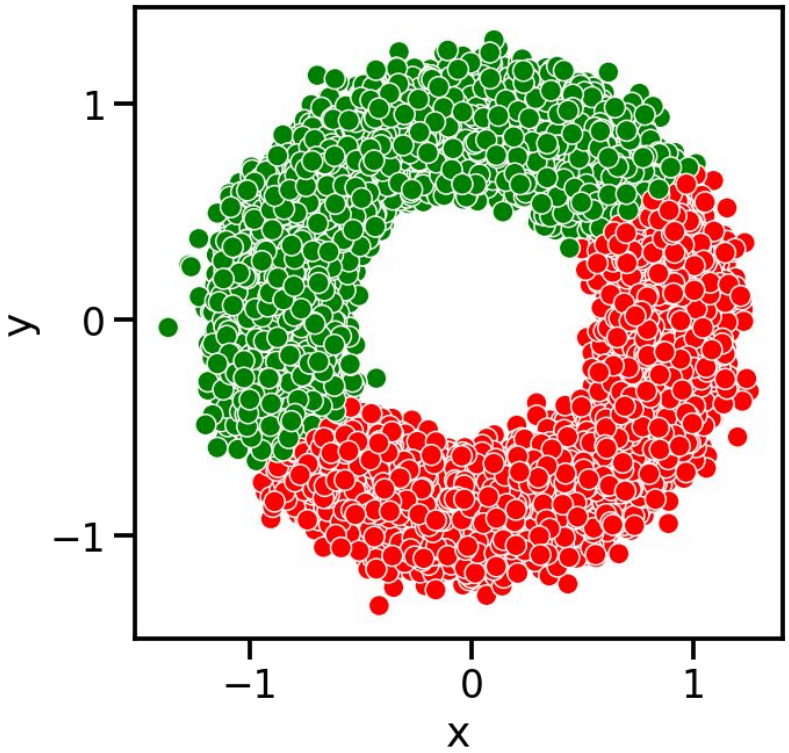
# Logistic Regression #1

**Features**
Coordinates X, Y, and constant

**Target**
Red = 0, green = 1

**Accuracy**
0.50

**Log Loss**
0.693

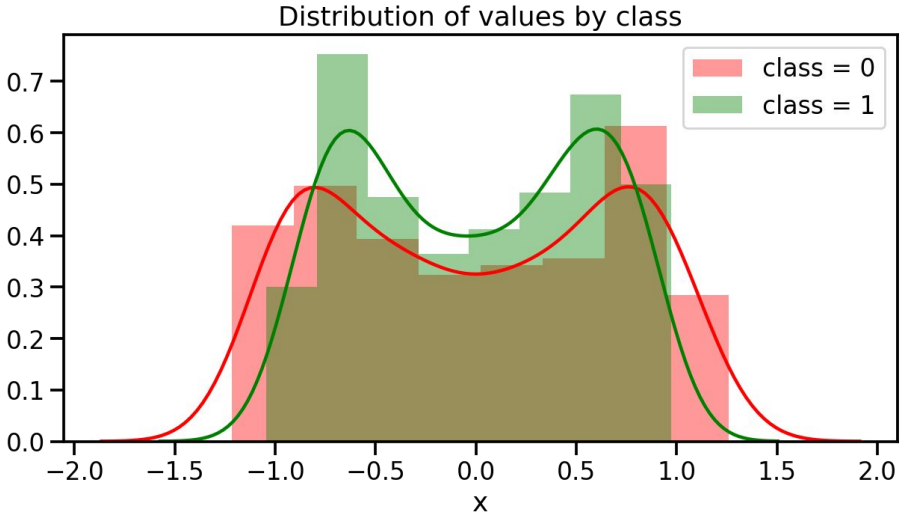


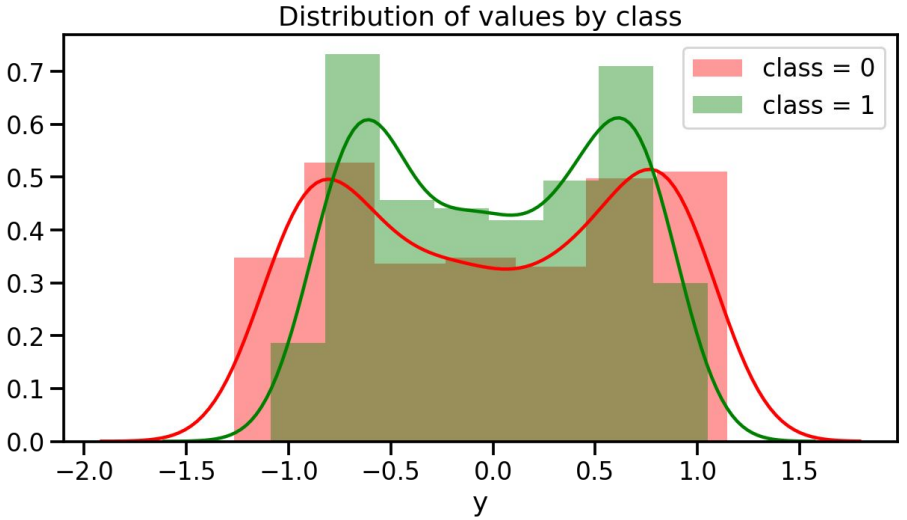

See GaussianCircles.ipynb for implementation.

# Distribution by X and Y

**Inference from Distributions**
There is no way to separate classes with a straight line in either X or Y space.

How to construct **a feature which can separate the two circles** with a line?



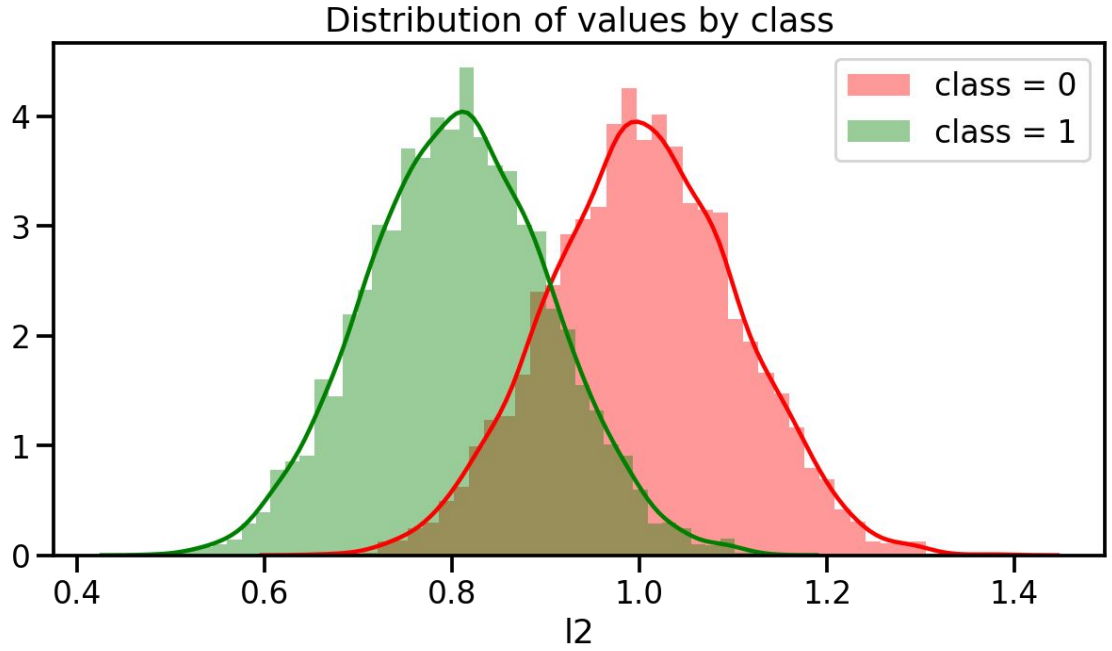



# Distribution by L2 Norm

L2 Norm of X and Y = distance from the center.

Distributions **partially separated** from each other.

Overlapping area implies that a **perfectly accurate classifier is not possible**.



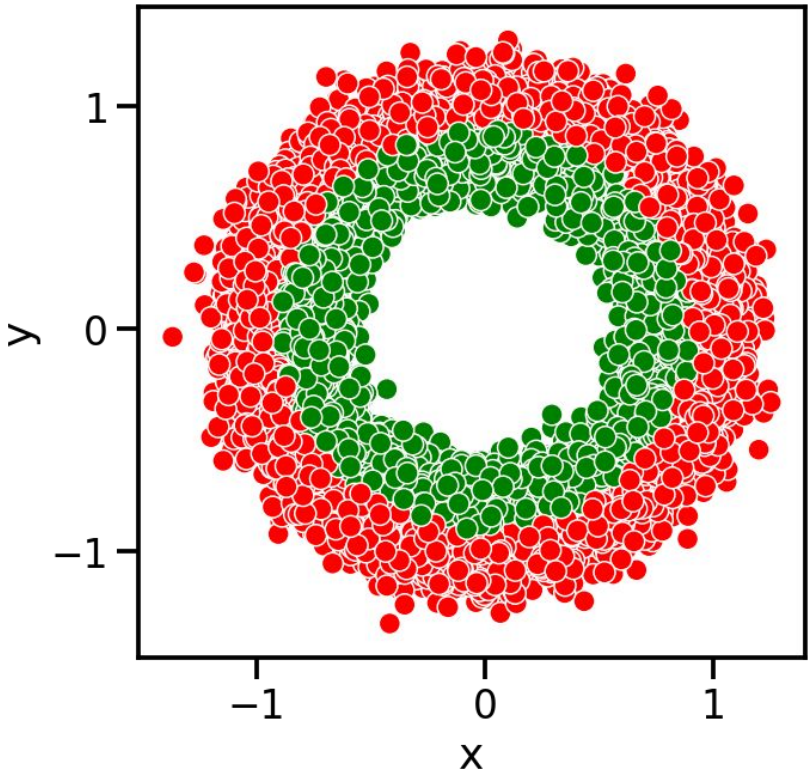
# Logistic Regression #2

**Features**
L2 norm, and constant

**Target**
Red = 0, green = 1

**Accuracy**
0.84

**Log Loss**
0.36



# Neural Network

**Layers**

1. FC with 10 nodes, tanh
2. FC with 1 node, sigmoid

**Features**

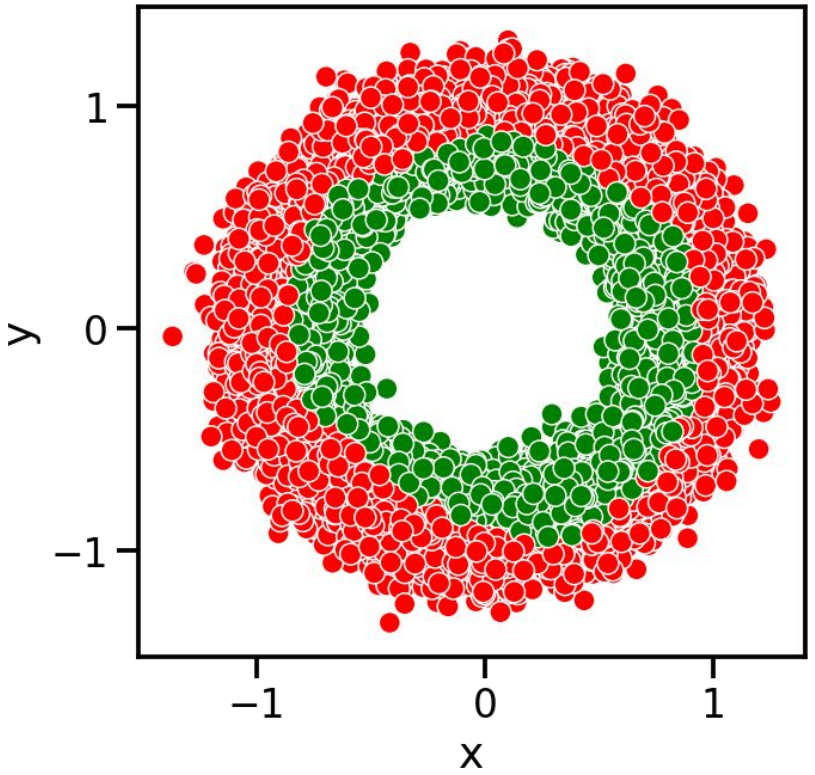
Coordinates of X and Y

**Target**

Red = 0, green = 1

**Log Loss**

0.435

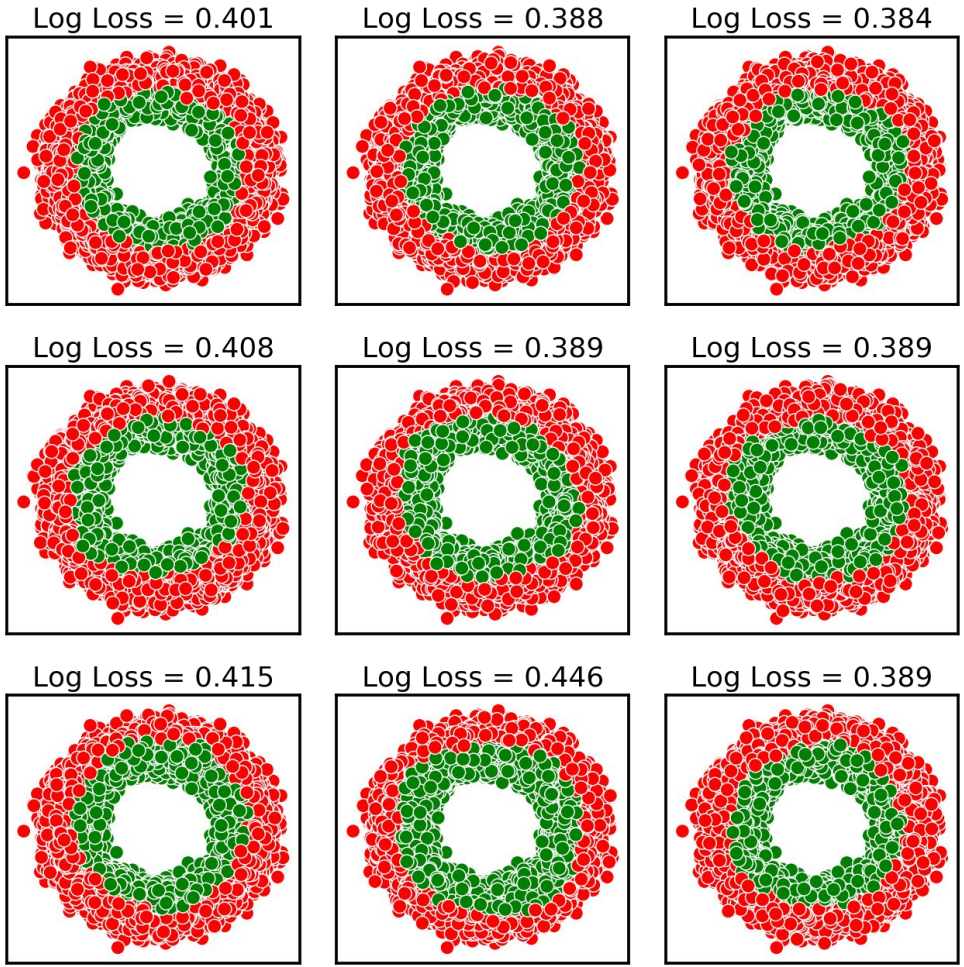


# Example: Bagging

1. Sample 20 different bootstrap replicates of the training dataset

2. Train each model for 10 epochs

3. Compare performance on the training dataset



# Demo

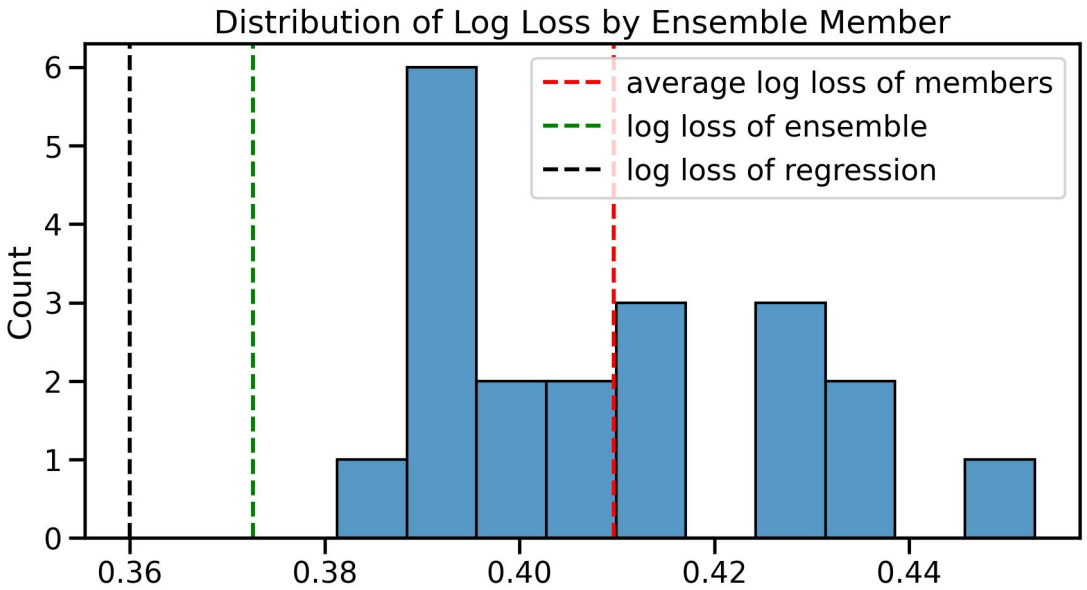
Gaussian Circles

# Ensemble: Bagging

**Members of Ensemble**
20 neural networks with different training bags

**Aggregation Method**
Average of Probability

**Log Loss of Ensemble**
0.373




See GaussianCircles.ipynb for implementation.

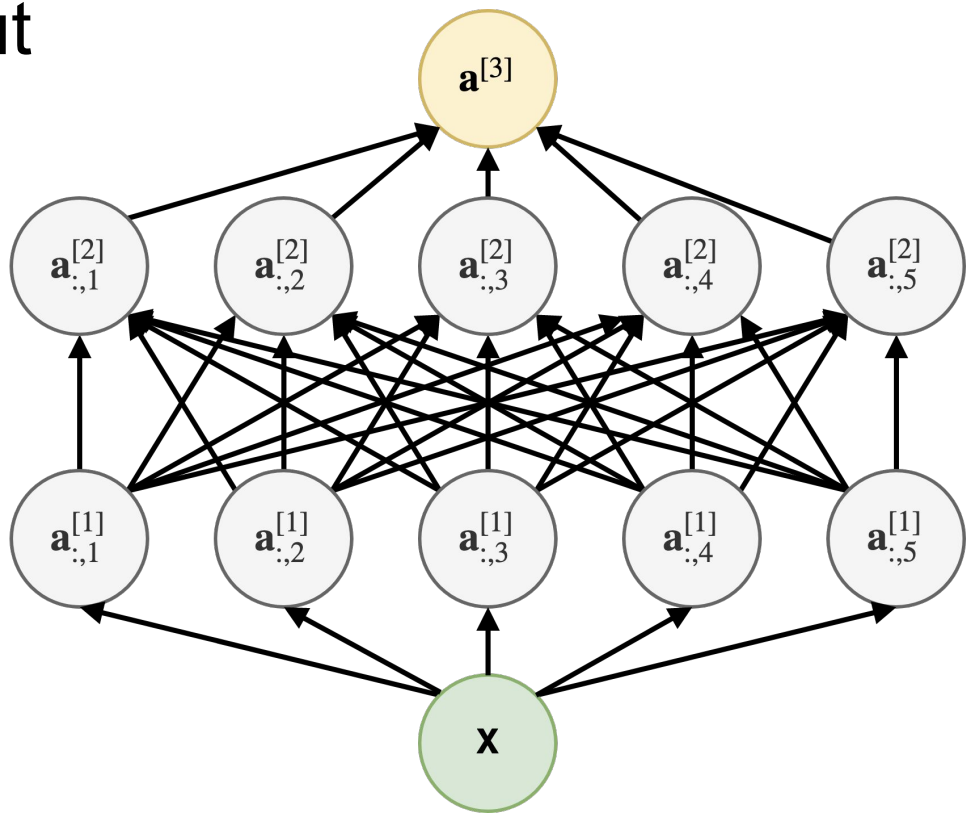
# Noise Injection: Dropout

**During training**

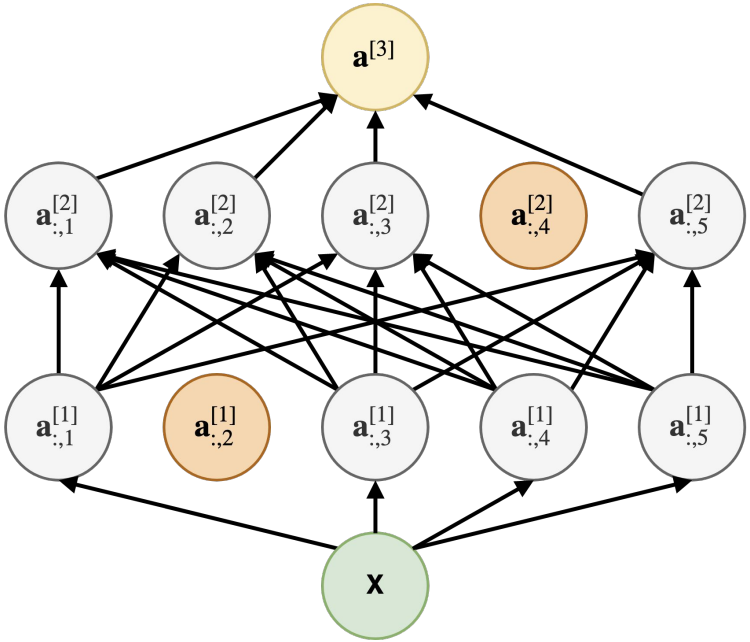
On each iteration, randomly choose and disable nodes with fixed probability.

**During inference**

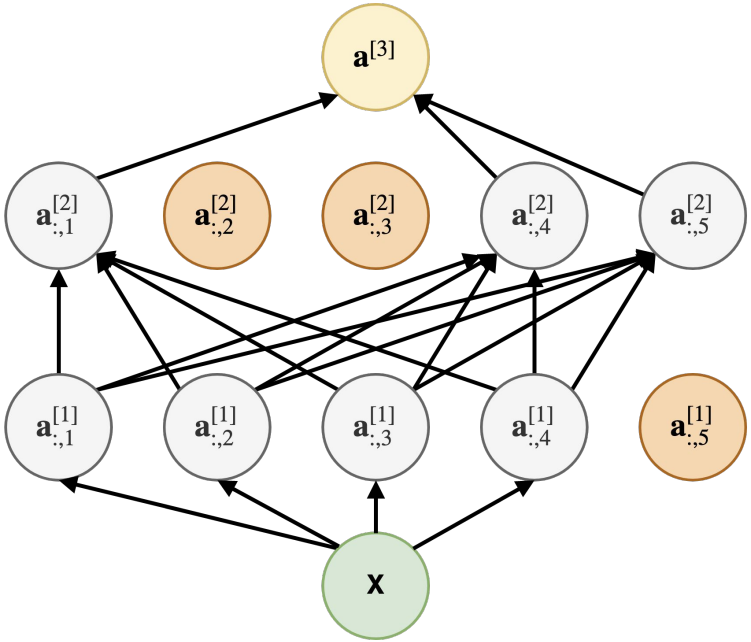
Use all nodes



# Noise Injection: Dropout




**Iteration 1**



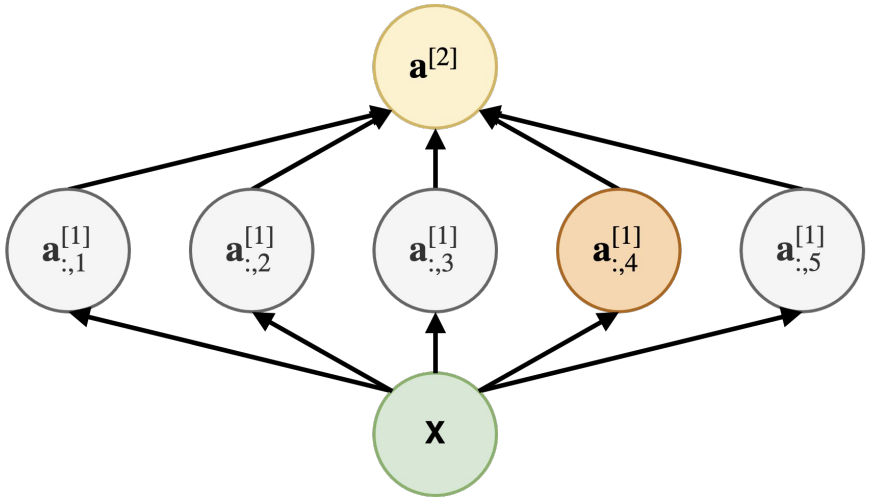

**Iteration 2**

# Noise Injection: Dropout

Let's define:

- $p$ = probability of keeping a node
- $\mathbf{d}$ = vector of random variables ~ Bernoulli($p$)

$$\tilde{\mathbf{a}}^{[1]}_{:,m} = \frac{\mathbf{a}^{[1]}_{:,m} \odot \mathbf{d}_m}{p}$$




Srivastava, N, et al. (2014)

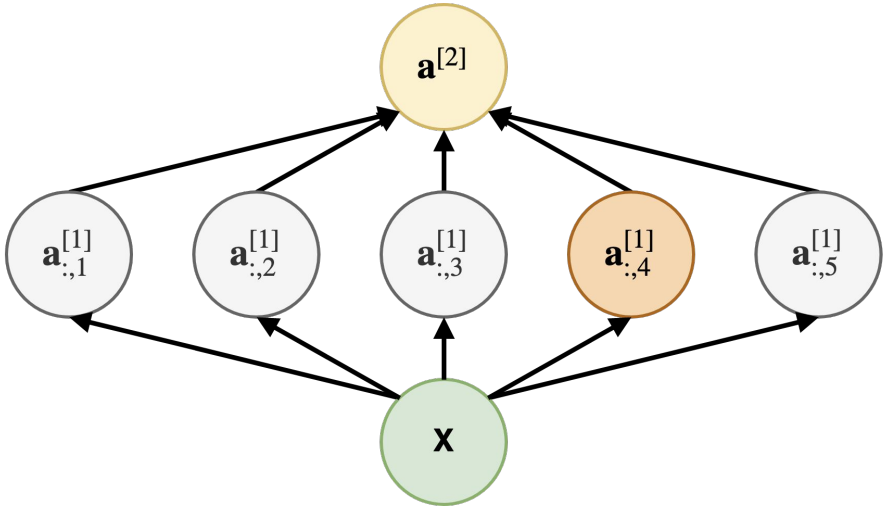
# Noise Injection: Dropout

$$\tilde{\mathbf{a}}^{[1]}_{:,m} = \frac{\mathbf{a}^{[1]}_{:,m} \odot \mathbf{d}_m}{p}$$

$$\mathbb{E}[\tilde{\mathbf{a}}^{[1]}_{s,:}] = \frac{\mathbb{E}[\mathbf{a}^{[1]}_{s,:}] \cdot \mathbb{E}[\mathbf{d}]}{p}$$

$$= \frac{\mathbb{E}[\mathbf{a}^{[1]}_{s,:}] \cdot p}{p}$$

$$= \mathbb{E}[\mathbf{a}^{[1]}_{s,:}]$$




Srivastava, N, et al. (2014)

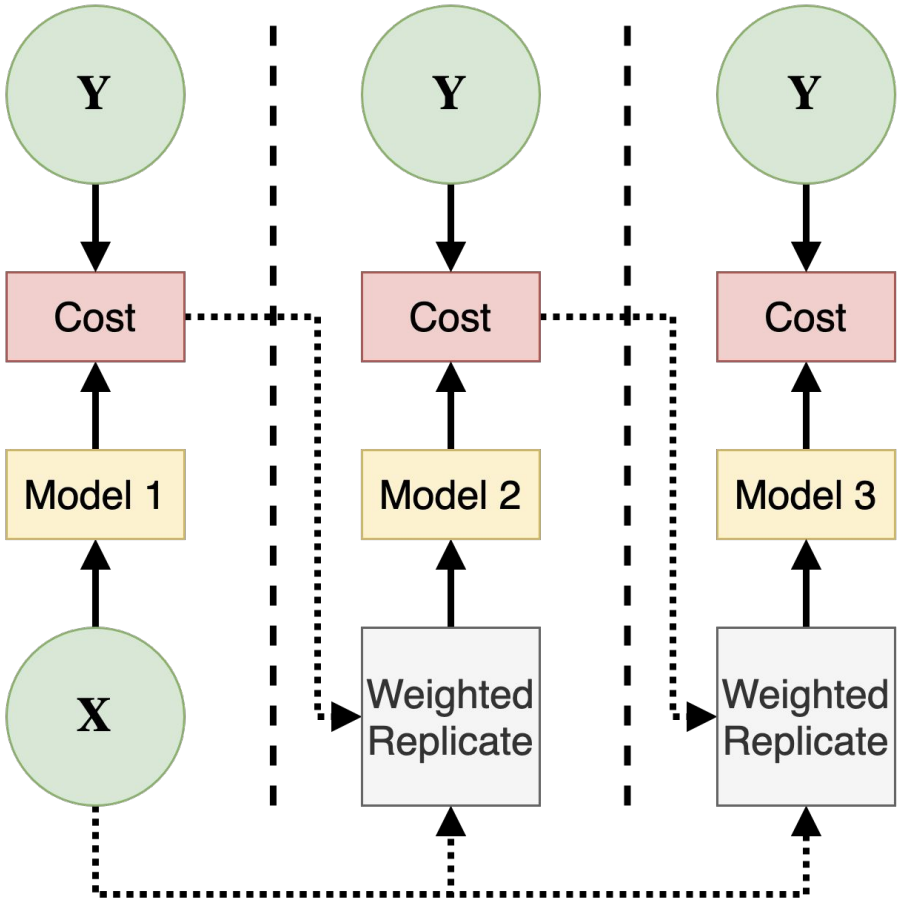
See Dropout.ipynb for discussion, and Keras implementations.

# AdaBoost: Training #1

Sequential model training on different datasets.

Each dataset resampled with probabilities depending on the per-sample cost of the previous model.

Each subsequent model is trained with higher focus on hard training samples.



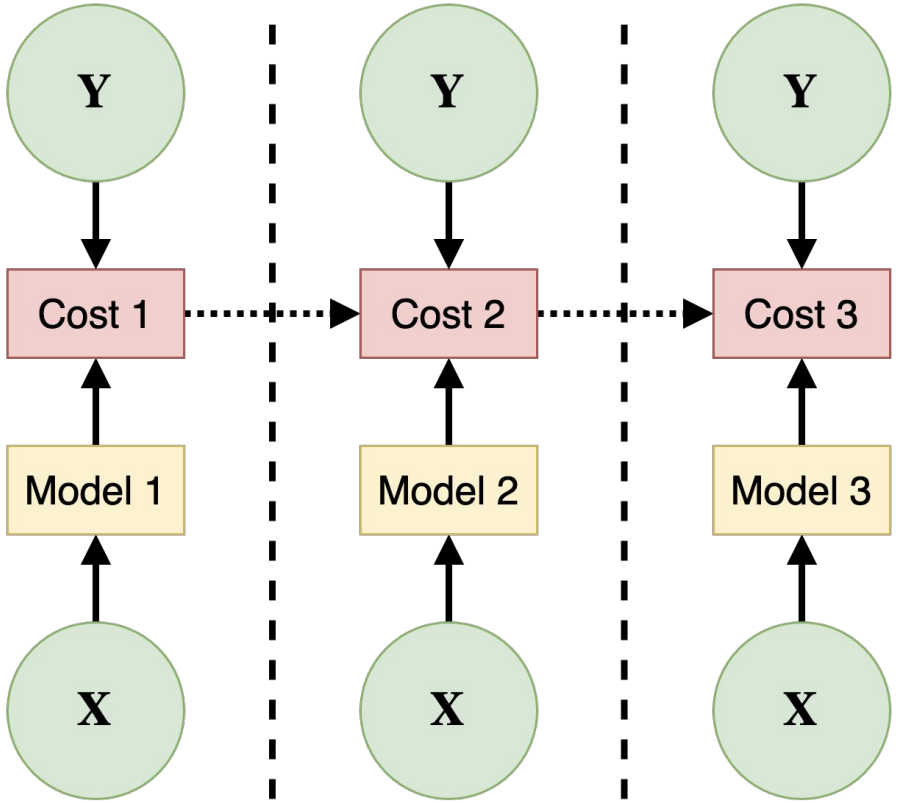

Shwenk, H., Bengio, Y. (2000)

# AdaBoost: Training #2

Sequential model training on the same dataset.

Cost = average of this model's per-example cost weighted by per-example cost from the previous model.

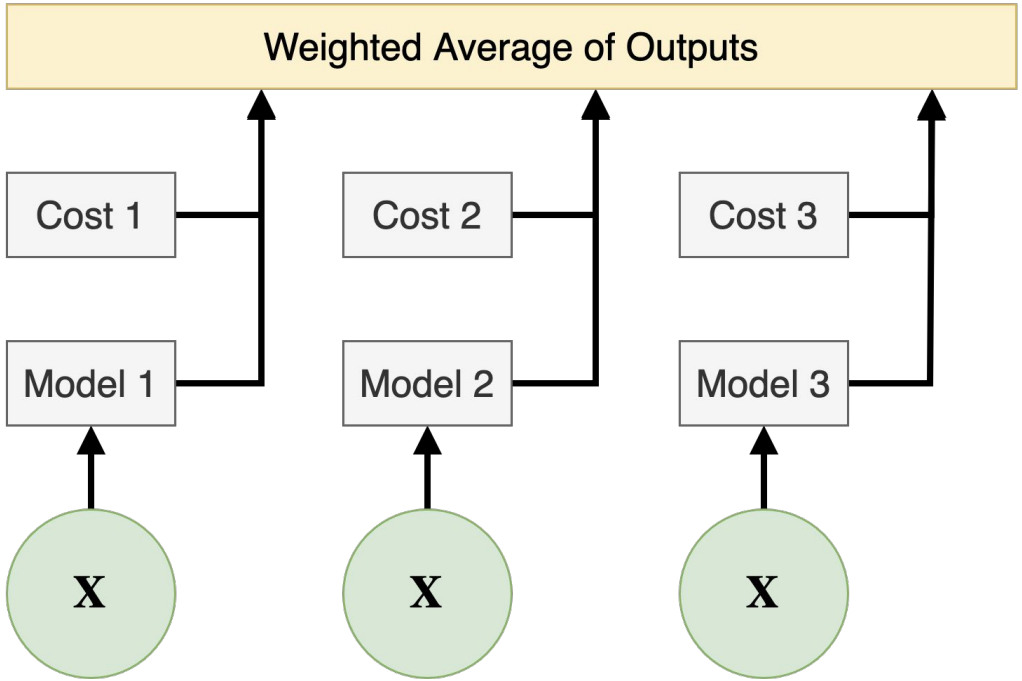
Each subsequent model is trained with higher focus on hard training examples.




Shwenk, H., Bengio, Y. (2000)

# AdaBoost: Inference

Ensemble output is the average over each model's output weighted by the performance on the training dataset.

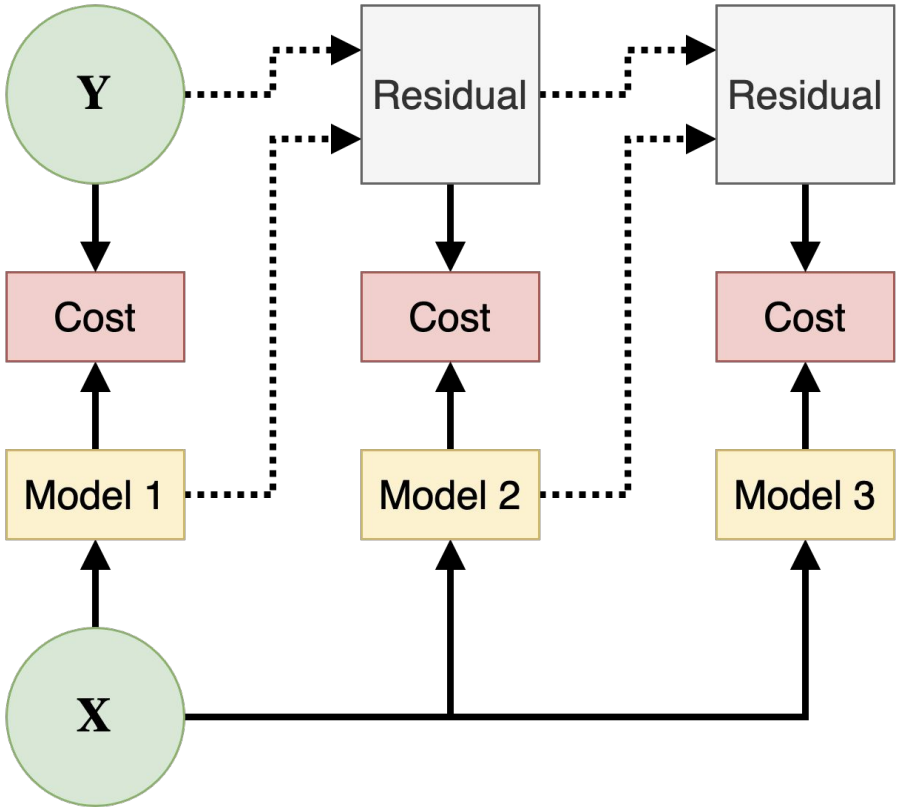



Shwenk, H., Bengio, Y. (2000)

# Gradient Boosting: Training

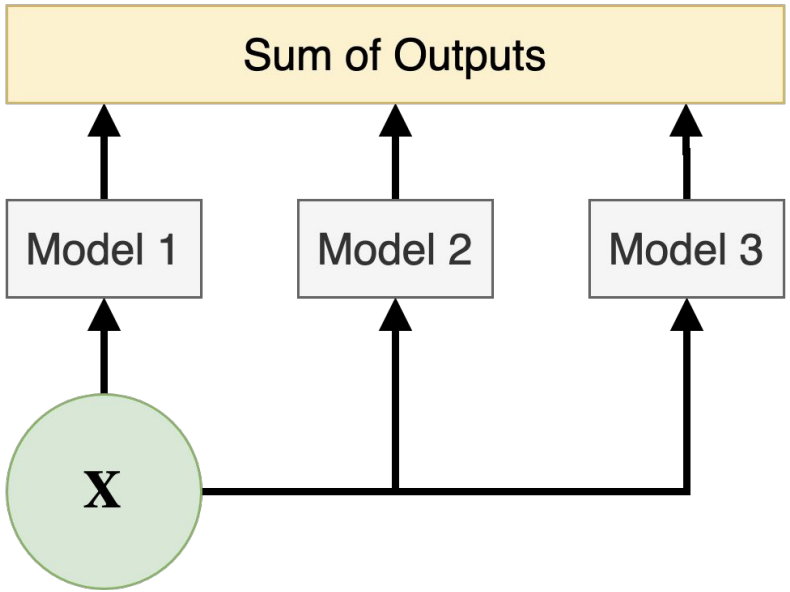
Sequential model training on the same dataset.

Each subsequent model is trained to output the residual of the previous model.



# Gradient Boosting: Inference

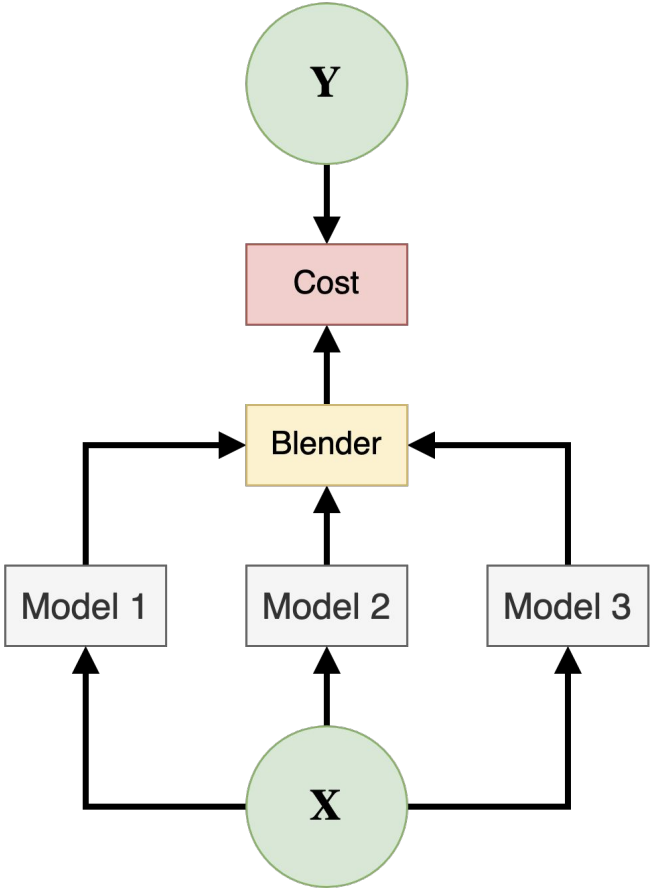
Ensemble output is the sum of outputs of each of the models.



# Stacking: Forward Pass

Instead of taking a simple average, the aggregation step can be an independent model taking outputs of ensemble members as input.

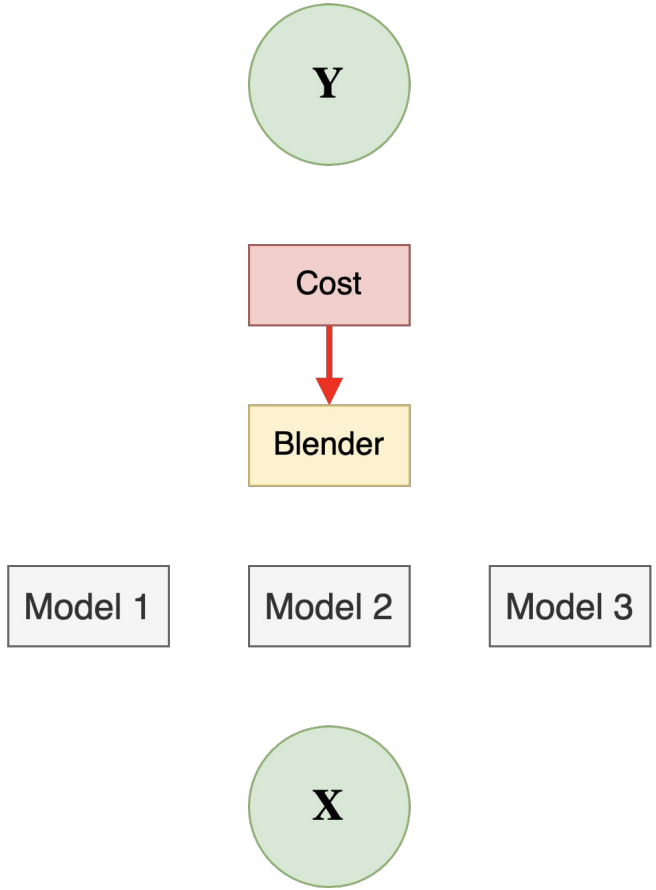
Blender is trained after the ensemble models have been trained.



# Stacking: Backward Pass

Instead of taking a simple average, the aggregation step can be an independent model taking outputs of ensemble members as input.

Blender is trained after the ensemble models have been trained.

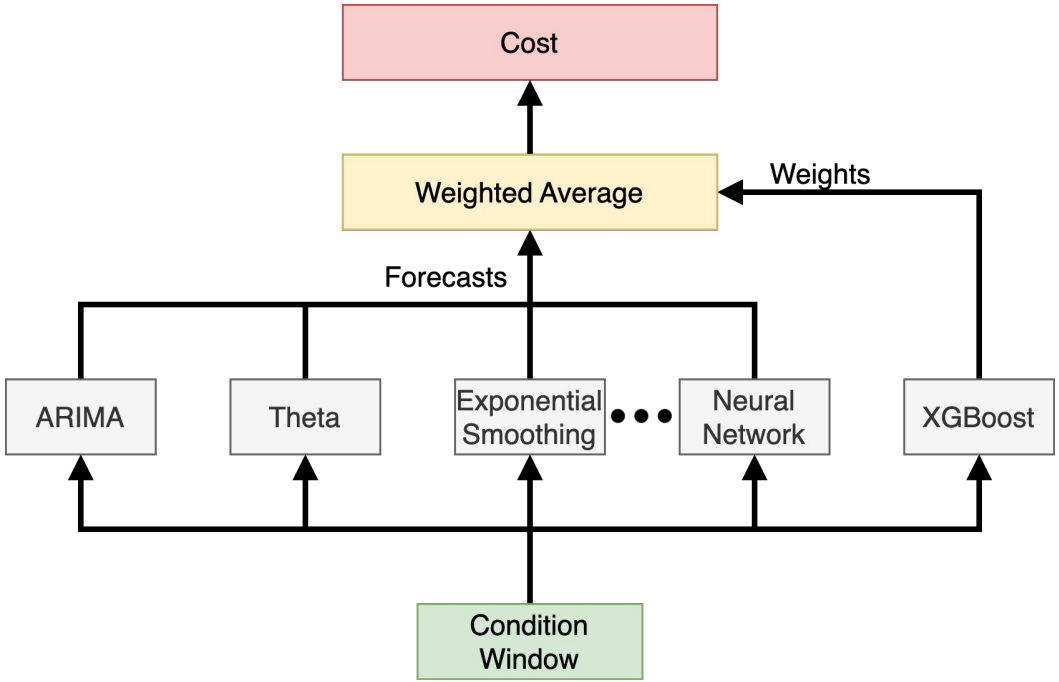


# Case Study: FFORMA

Top-2 in M4 Competition

**Blender**
Weighted average
with weights predicted by XGBoost

**XGBoost Features**
Training-time errors by each member
Length of time-series
Strength of trend
Curvature
Autocorrelation




Source: FFORMA: Feature-based forecast model averaging (Monterto-Manso, P., et al, 2020)

# Examples from Previous Papers

| Model | Diversity Source |
| --- | --- |
| GoogLeNet | Dropout + Batch seed |
| ResNet | Network depth |
| VGGNet | Dropout + Image size |
| AlexNet | Dropout + ? |

# Part 2

## CarNet

(Image by a Stable Diffusion model)

# Case Study: Tesla AutoPilot

**Source**

Tesla AI Day 2022

**Neural Network sub-tasks**

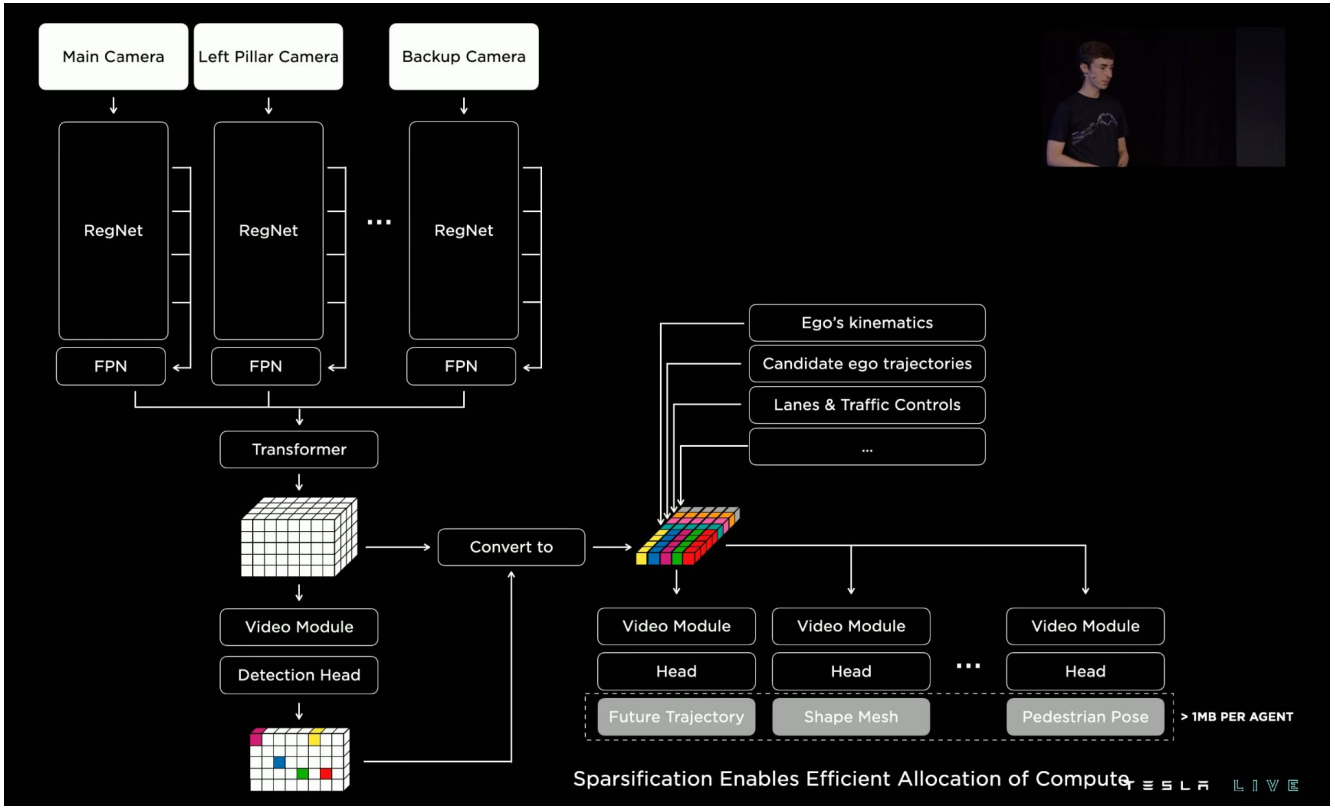
Object detection

Occupancy detection

Lane detection

# AutoPilot - Object Detection

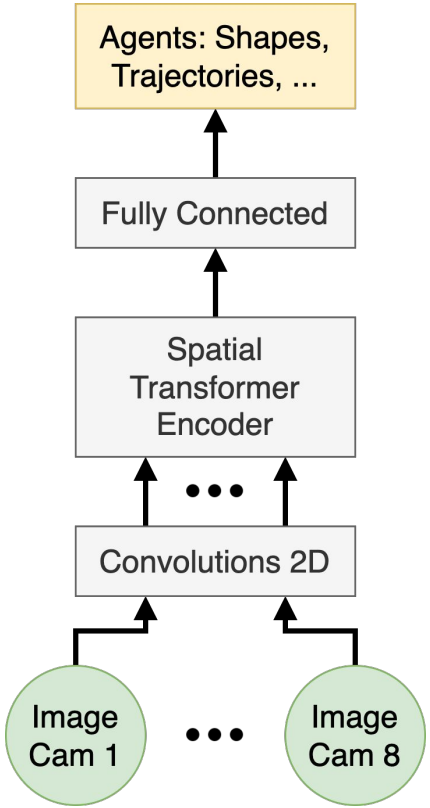
Source: Tesla AI Day 2022



# AutoPilot - Object Detection

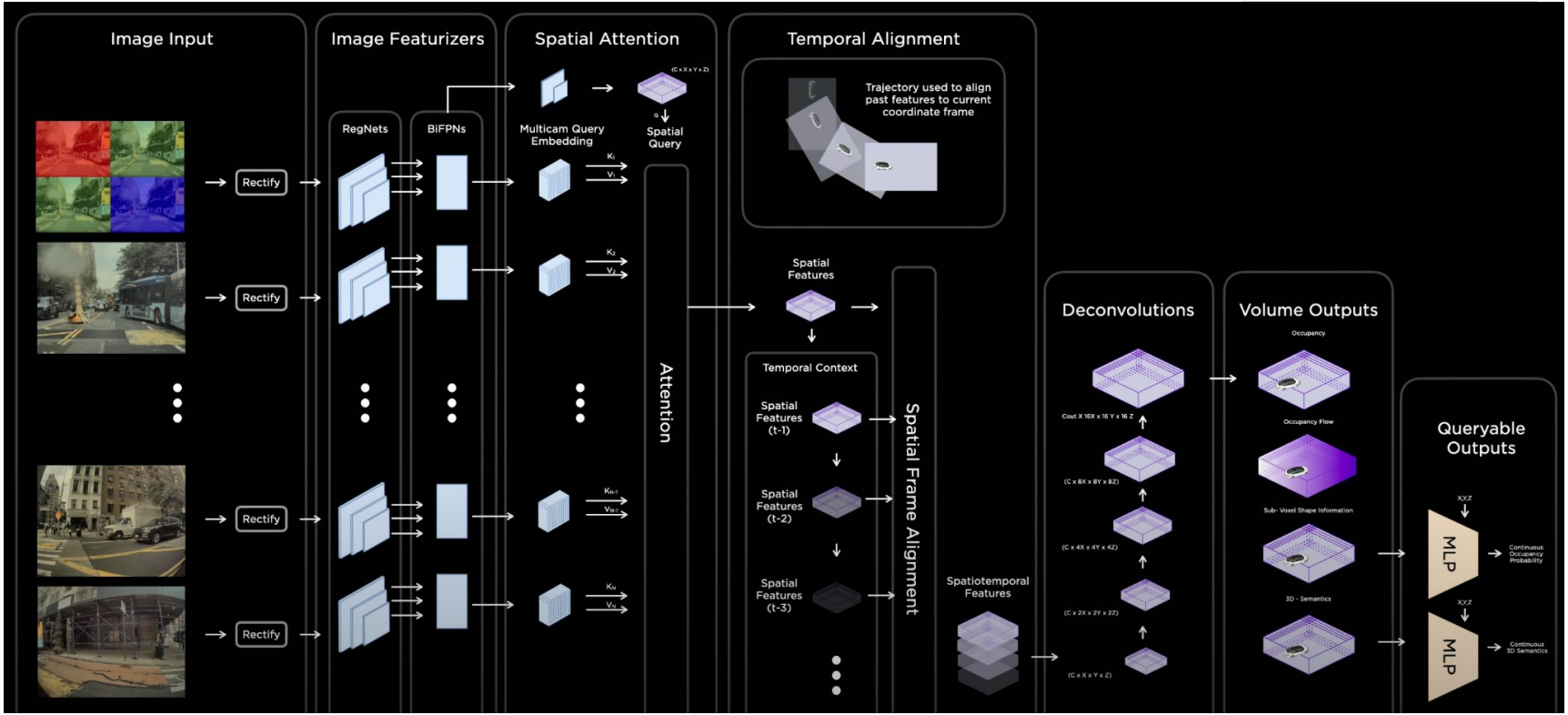
**Task**
Detect agents: vehicles, people, etc.
Per-agent attributes: velocity, shape, etc.

Other parts of the system make decisions based on the detected objects and attributes.




Source: Tesla AI Day 2022

# Tesla AutoPilot - Occupancy Detection




Source: Tesla AI Day 2022

# AutoPilot - Occupancy Detection

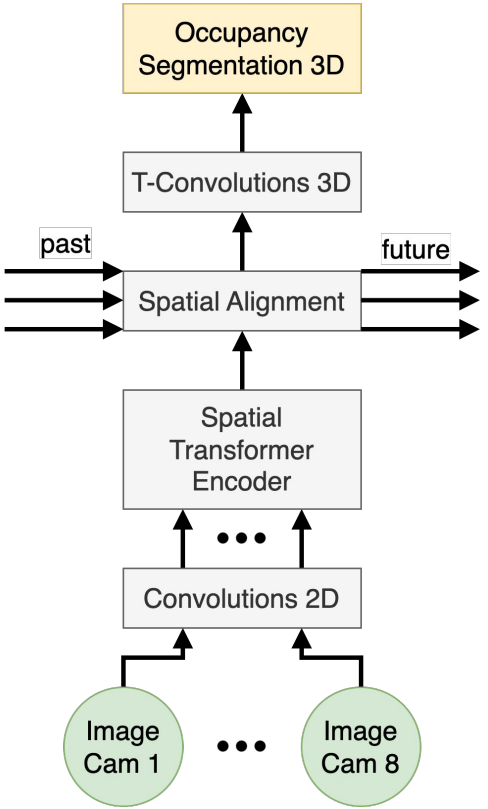
**Task**

Segmentation in 3D space

Per-voxel classification:

- Occupied by a static object
- Occupied by a moving object
- Free space

Voxel = a pixel in 3D




Source: Tesla AI Day 2022

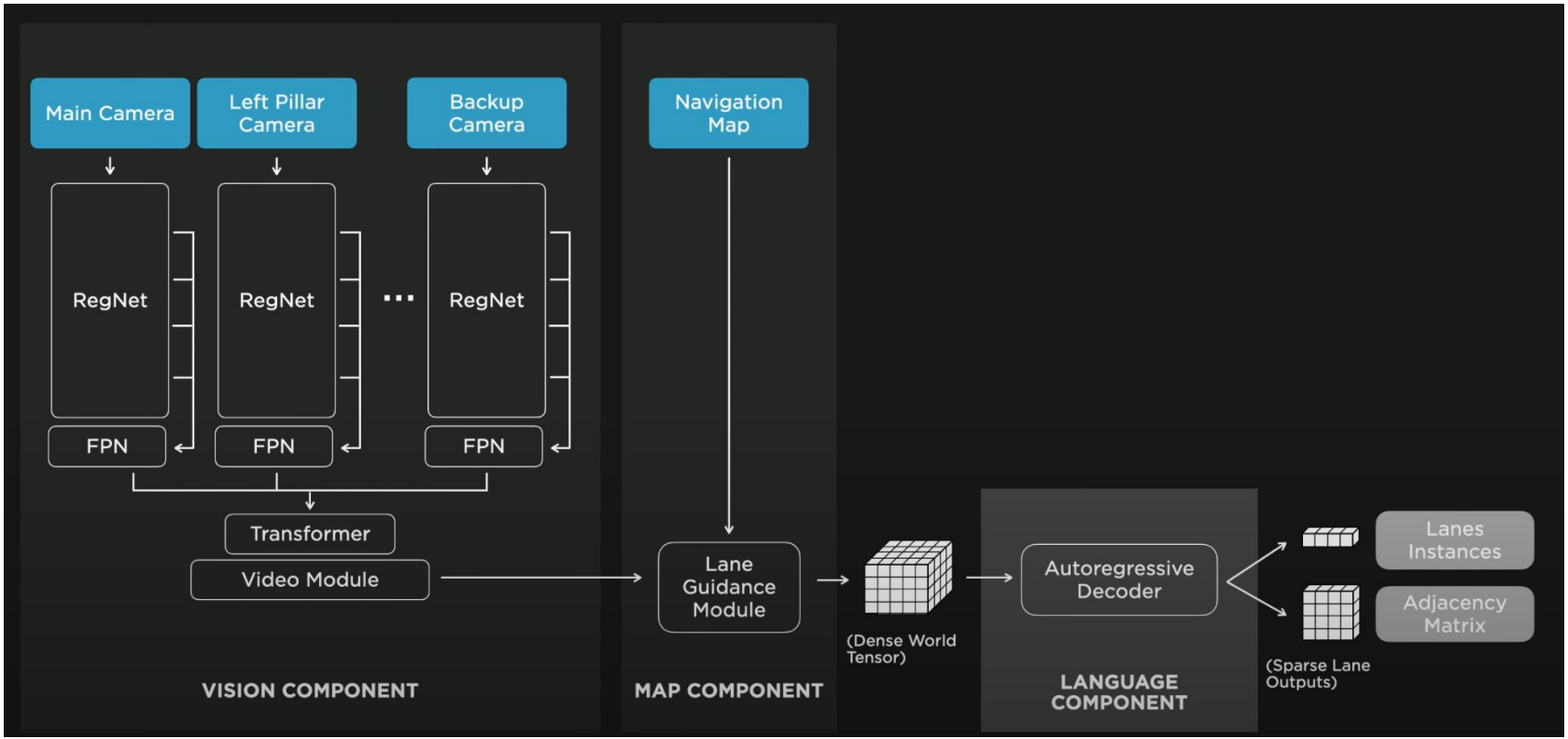
Bus in 2D on Camera
Bus in 3D in Occupancy Detection

Car

Source: Tesla AI Day 2022

# AutoPilot - Lane Detection

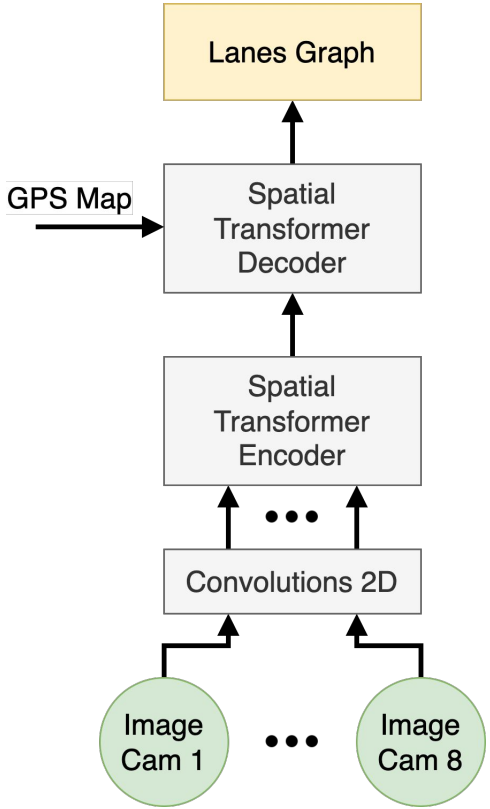
Source: Tesla AI Day 2022



# AutoPilot - Lane Detection

**Task**
Detect lane instances
and connections between lanes.

**Details**
Utilizes a navigation map and GPS location.
Graph is encoded with a “language of lanes”.




Source: Tesla AI Day 2022

# Assignment 5: CarNet

**Dataset**

30K images

228 x 228 x 3

**Task**

1. Count signals (traffic lights, street signs)
2. Count vehicles (cars, planes, bikes, …)

**Loss**

Average MSE =

½ MSE(signals) + ½ MSE(vehicles)


# CarNet: Baseline MSE

| backbone | signal | vehicle | average |
|---|---|---|---|
| vanilla cnn | 1.81 | 9.65 | 5.73 |
| mobile net | 1.41 | 9.77 | 5.59 |
| mean model | 1.91 | 13.23 | 7.57 |


# CarNet: Grading

Any neural network architecture, training procedure, and ensemble method.

**Grading**

**15 points** = percentile of Average MSE

**5 points** = description of the solution

**Deadline**

EOD October 30th


# Demo

## Assignment Template

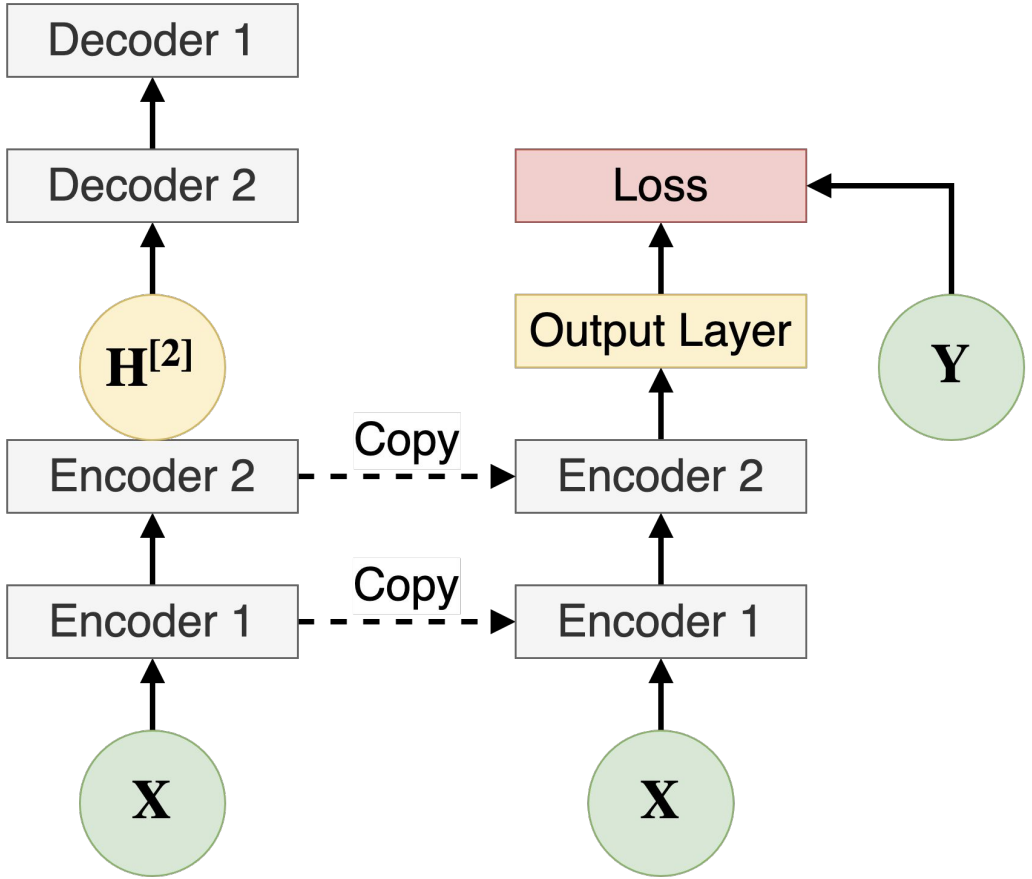
# Part 3

Course Summary

# Managerial Economics of Deep Learning

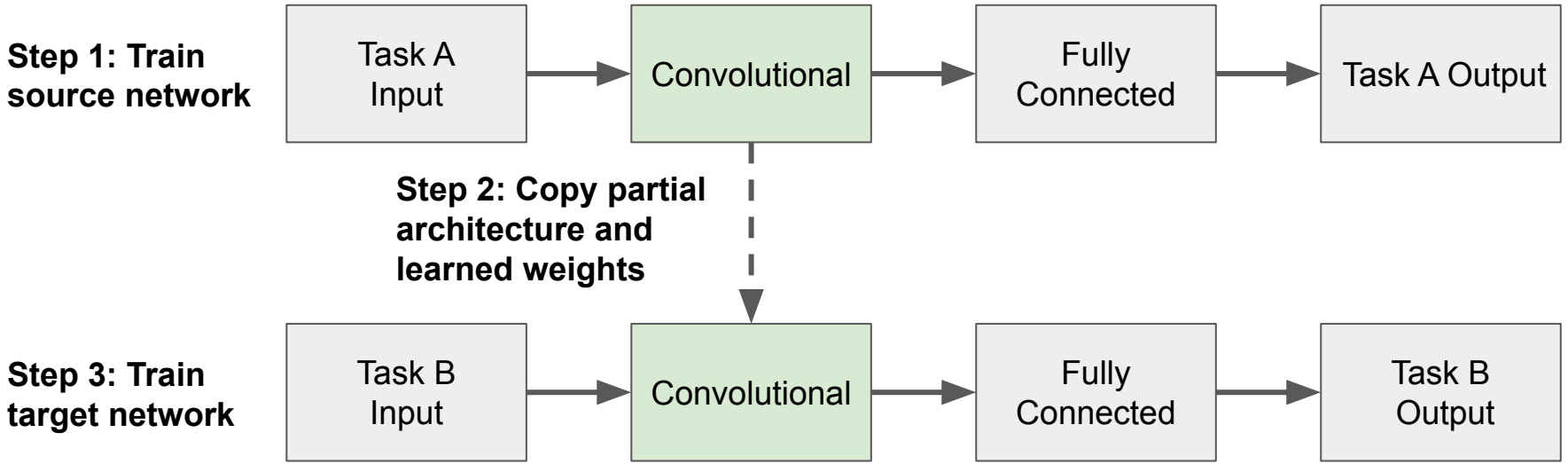
| Project Methodology | Project Duration | Compute Resource | Required Training |
|---|---|---|---|
| Design new architecture and train from scratch | Months | Expensive | Machine Learning Researcher |
| Use existing architecture and/or transfer learning | Weeks | Cheaper | Machine Learning Engineer |
| Find a pre-trained model and/or buy Inference as a Service API | Days | Almost Free | Automation Software Engineer |

# Lecture 1: Unsupervised Pre-training




Bengio, Y., et al. (2007)

# Lecture 2: Transfer of Supervised Learning




Tan, C., et al. (2018)

# Lecture 3: Zero-Shot Prediction

**Source Task**

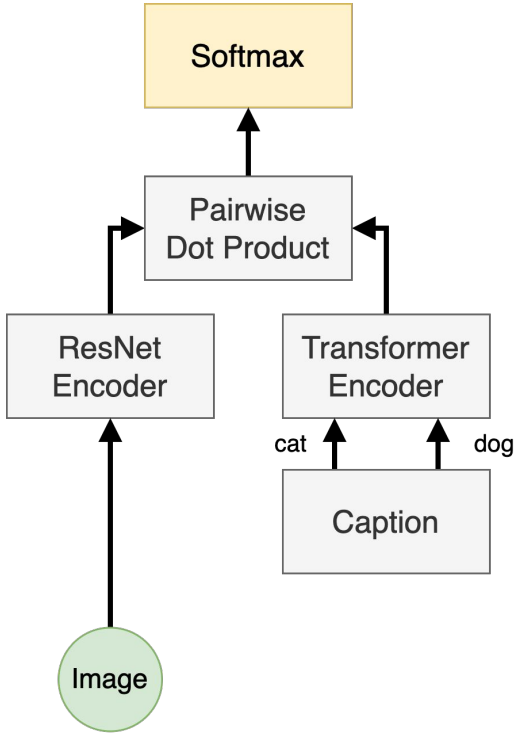
Image-caption similarity

**Target Task**

Image classification

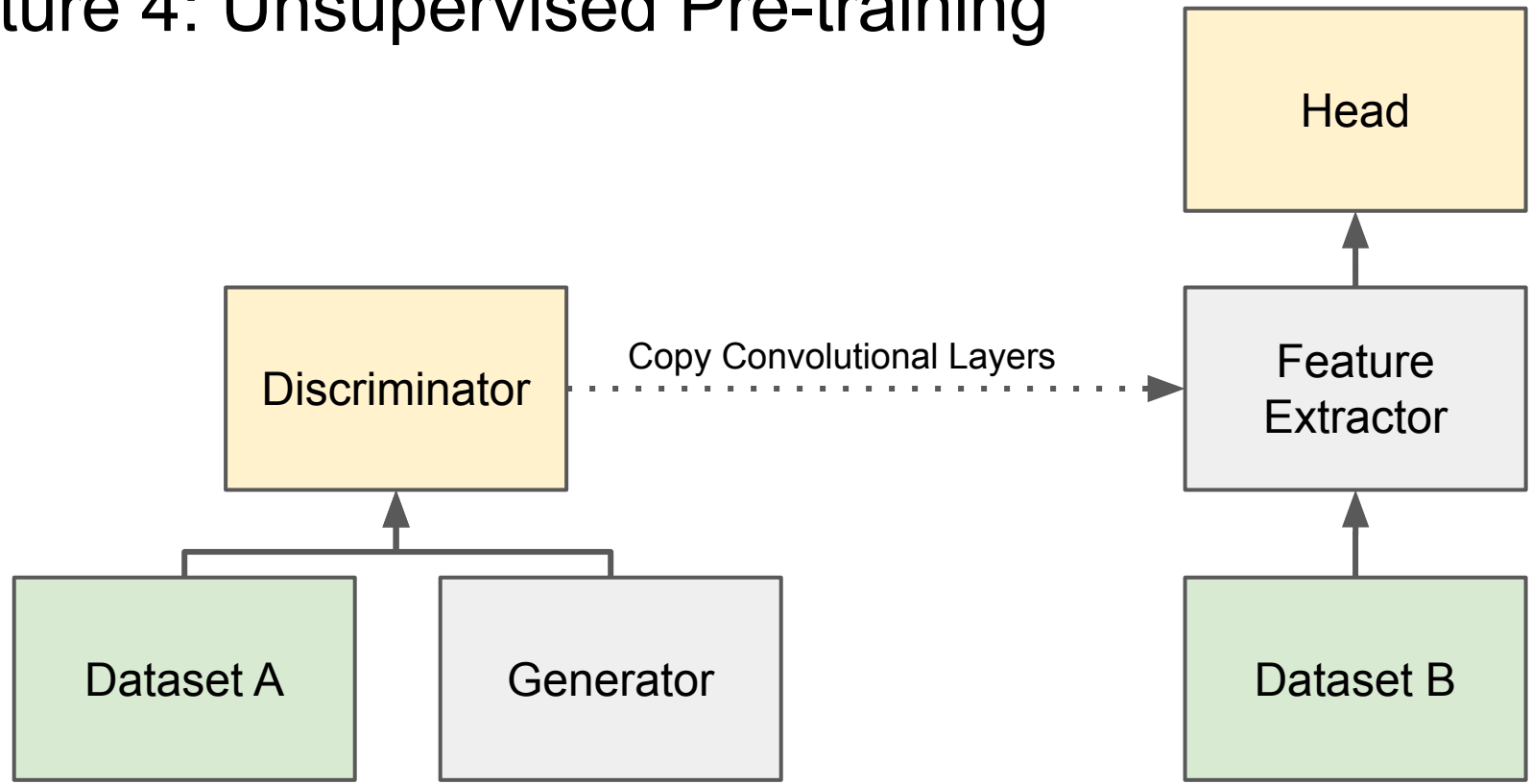
1. Create per-class English captions
2. Pass similarity scores to softmax

No training!




Source: Learning Transferable Visual Models From Natural Language Supervision (Radford, A., et al., 2021)

# Lecture 4: Unsupervised Pre-training




Radford, A., et al. (2016)

# Model Stores

**In this Course**

Keras Applications

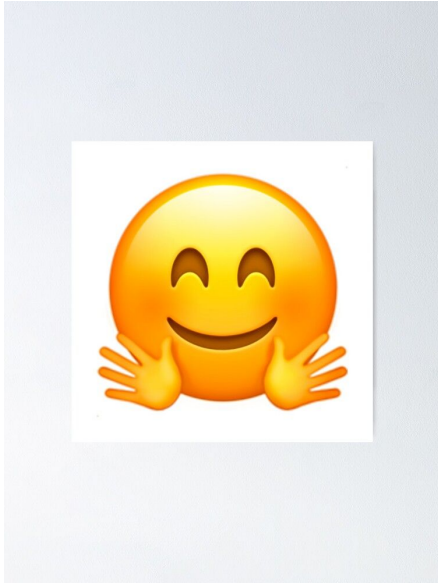
TensorFlow Hub

HuggingFace Hub

**Other**

PapersWithCode.com

TensorFlow Model Garden



# AWS Rekognition

**Product**

~ $0.001 / image

Available as a convenient API

**Features**

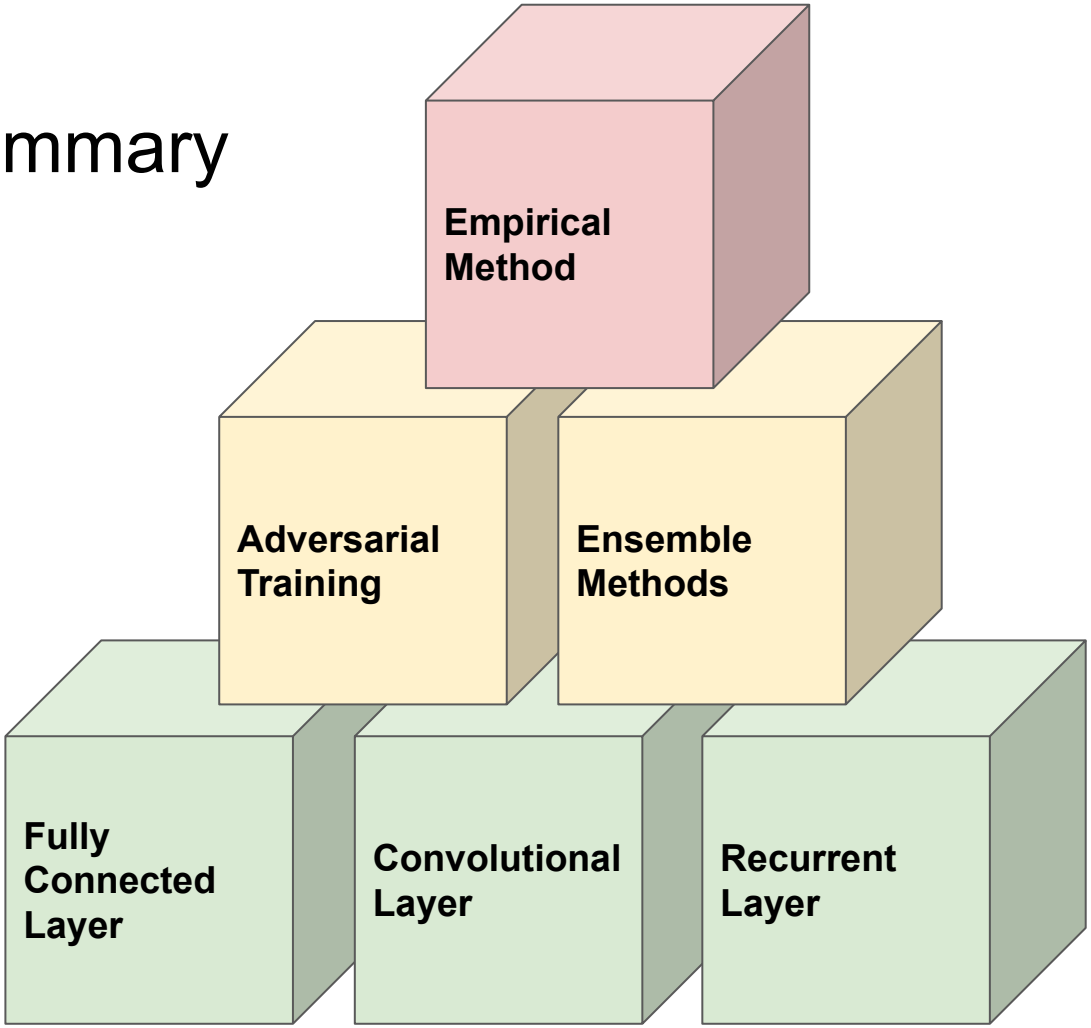
Object/scene detection

Face recognition and analysis

Both photo and video input

Image to text

# Course Summary



# BodyX

Input:

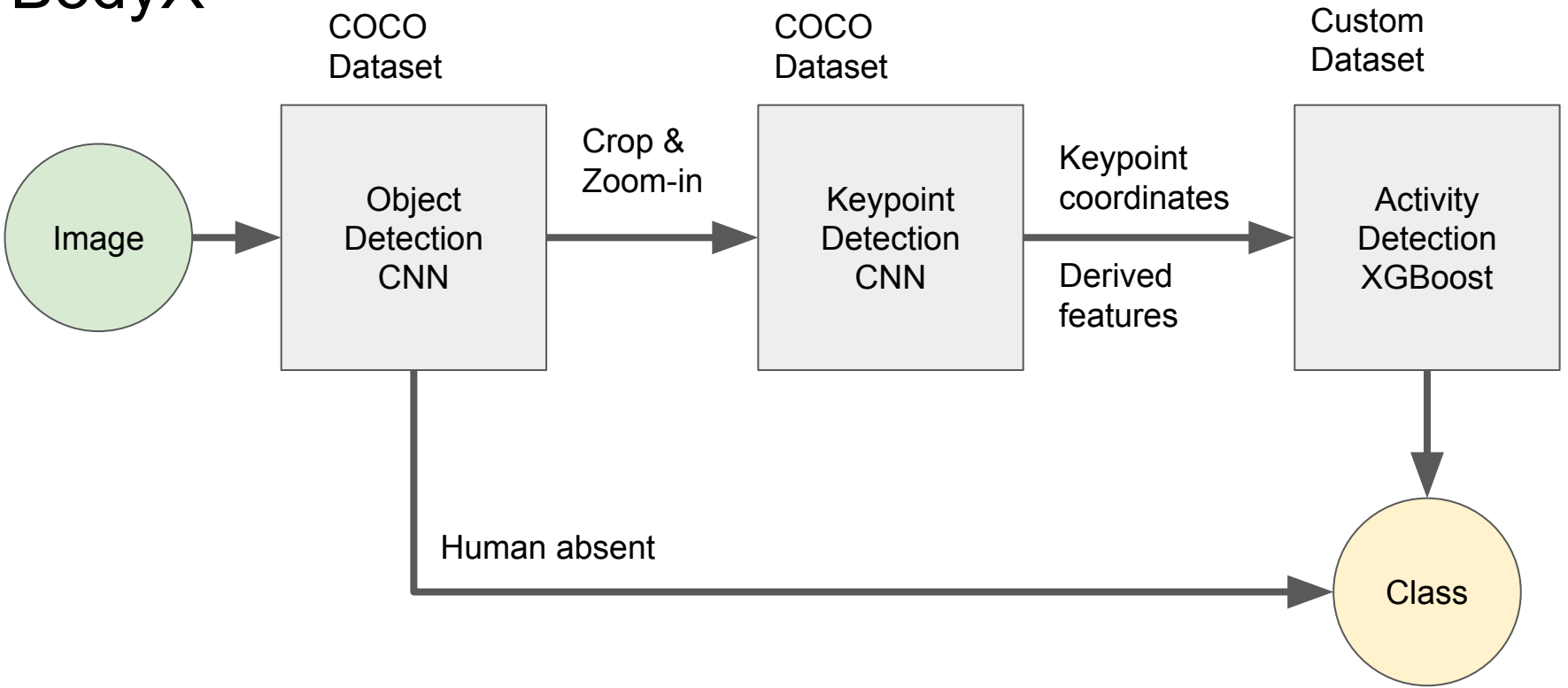
- Image RGB

Output:

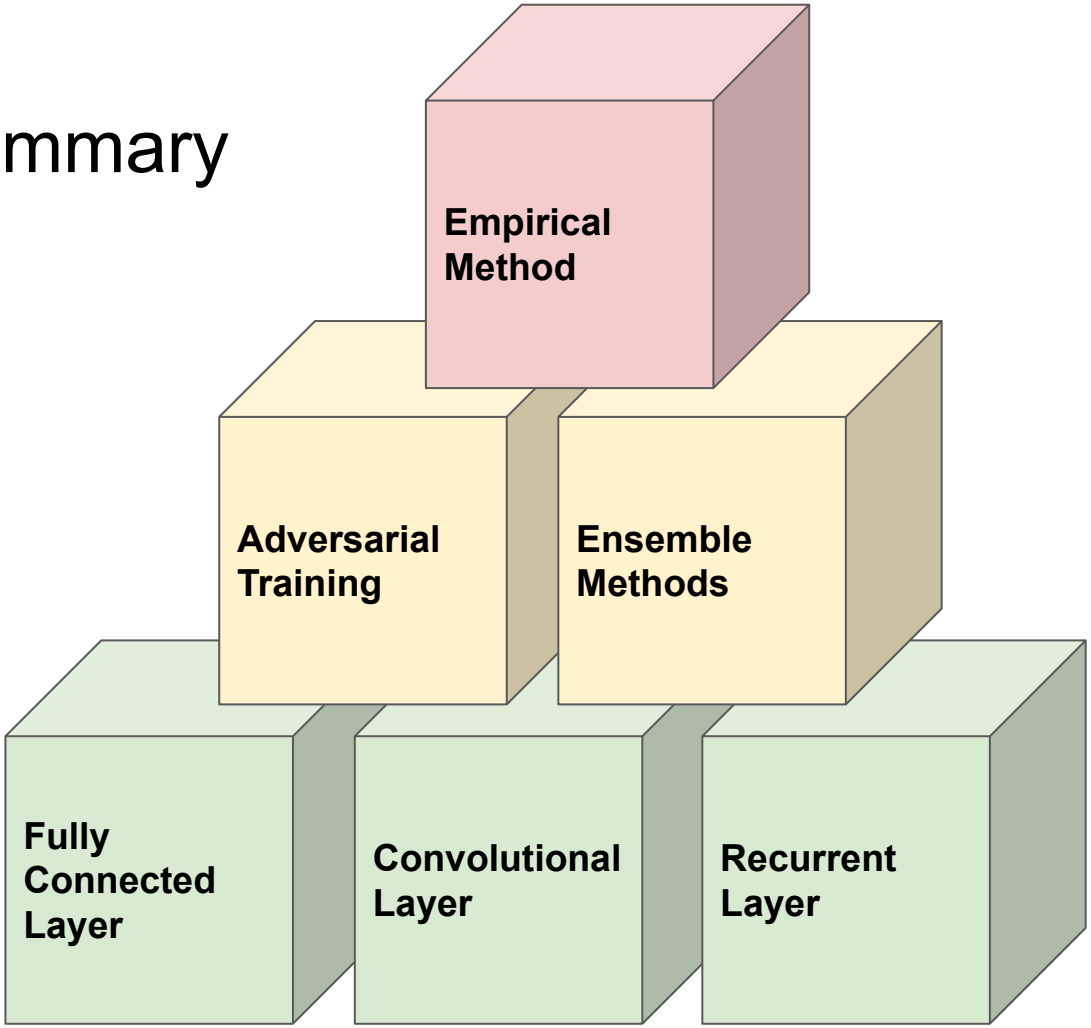
- 4-way classification
  - Human absent
  - Human present, sitting
  - Human present, standing
  - Human present, unknown pose

Present & Sitting

# BodyX


COCO
Dataset
COCO
Dataset
Custom
Dataset
Image
Object
Detection
CNN
Crop &
Zoom-in
Keypoint
Detection
CNN
Keypoint
coordinates
Derived
features
Activity
Detection
XGBoost
Human absent
Class

# Course Summary



# References


- A Short Introduction to Boosting (Fruend, Y., and Schapire, R. E., 1999)
- Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Srivastava, N., et al., 2014)
- Ensemble Methods in Machine Learning (Diettrich, T. G., 2000)
- FFORMA: Feature-based forecast model averaging (Monterto-Manso, P., et al, 2020)
- Tesla AI Day 2022: https://youtube.com/watch?v=ODSJsviD_SU

# Thank you

(Really) Final Q&A Time