ECC504-AI&ML

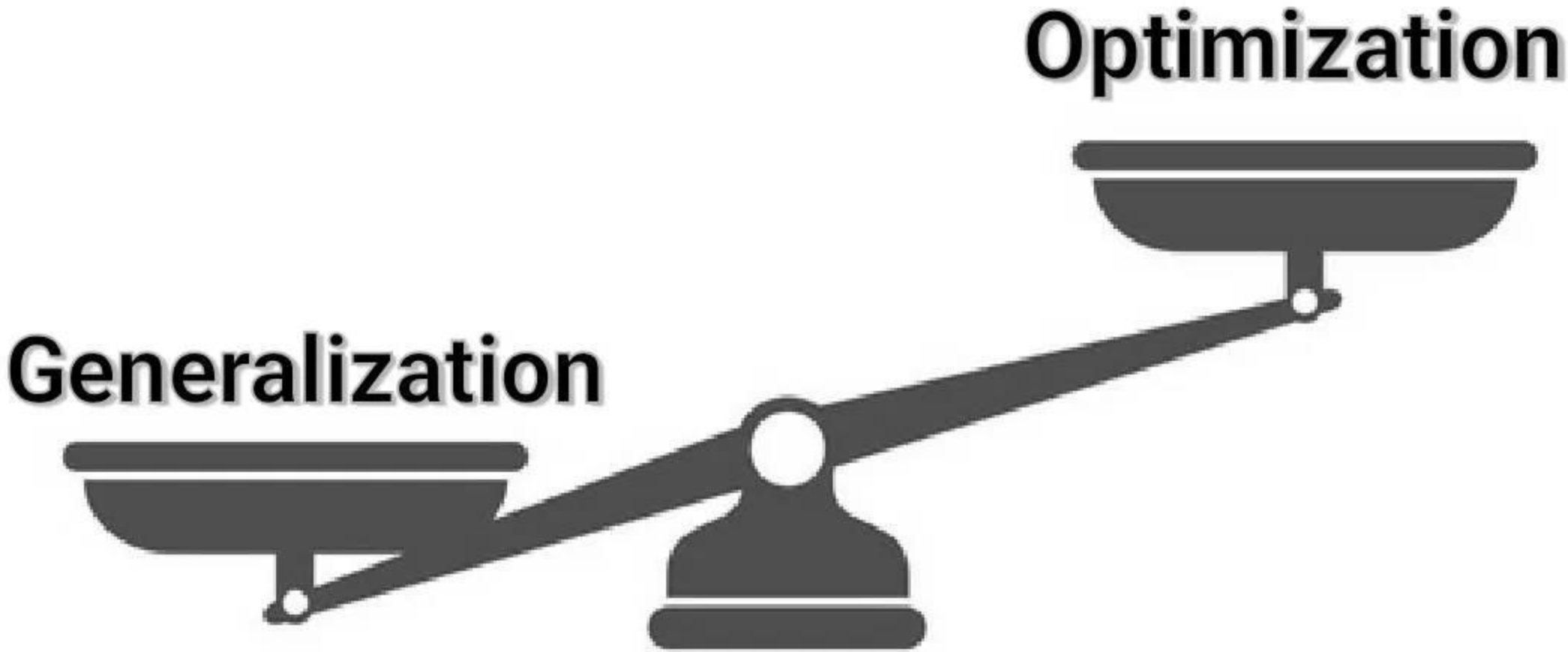
Module-1

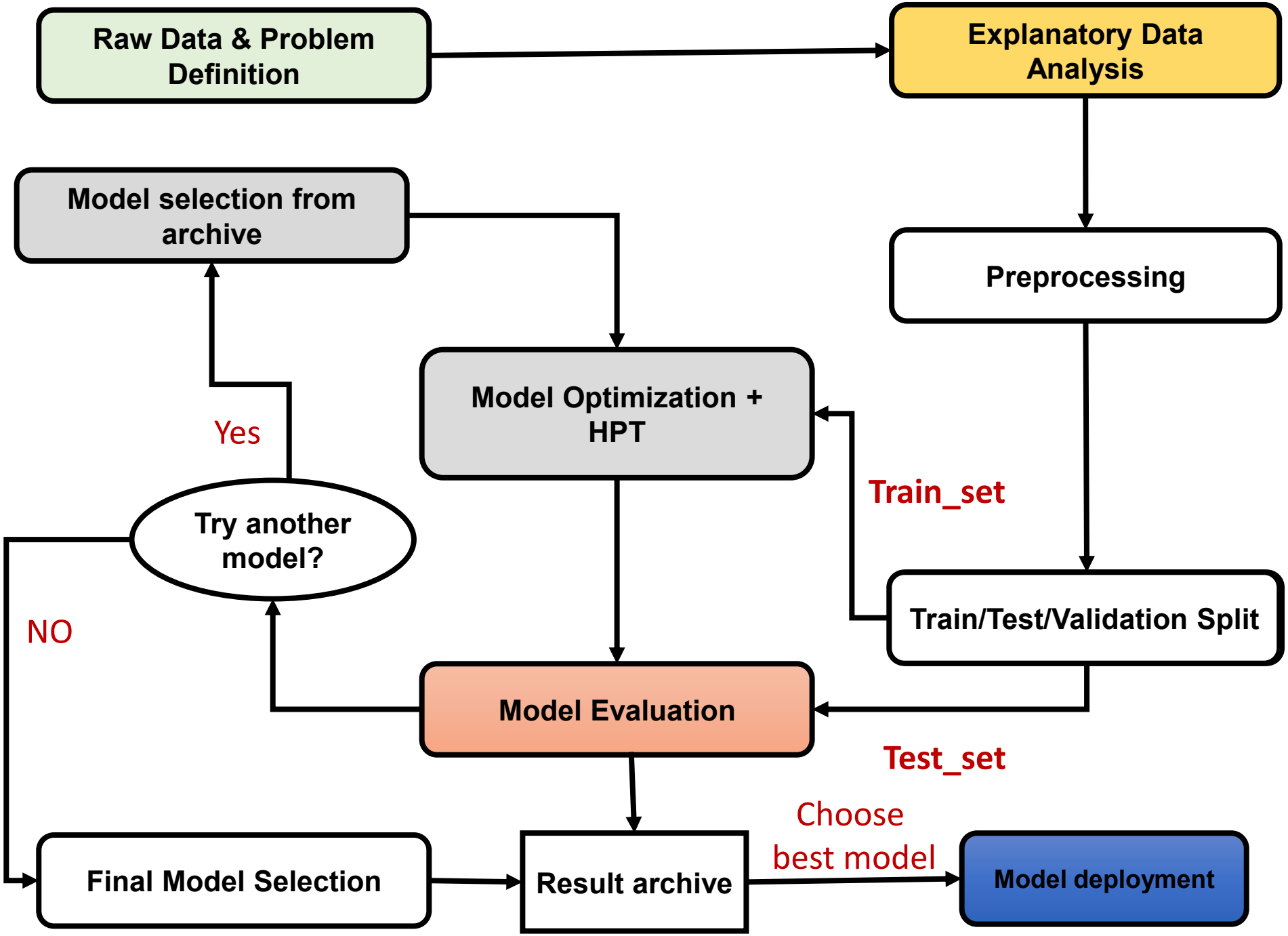
Content

- **ML pipeline**
- **Concept of generalization**
- **Case study on Heart disease prediction**
- **EDA, Model Selection**
- **Model evaluation – ROC, AUC**
- **Decision threshold tuning**
- **Evaluation metric trade-off**
- **Representation learning**
- **Case study on CIFAR-10 classification problem**
- **Concept of embedding space**
- **Shallow learning vs. Deep learning**
- **Concept of Deep Neural Network**

Key Takeaways for Machine Learning Practice

- **Good Data is Foundational**
  - High-quality, clean, and relevant data is essential for building reliable ML models.
- **Generalization is the Ultimate Goal**
  - The ability of a model to perform well on unseen data is crucial — but difficult to guarantee.
- **No Free Lunch Theorem Applies**
  - No one algorithm works best for every problem. If you're not sure, start simple and experiment wisely.
- **Use the Right Evaluation Metrics**
  - Always align your performance metrics (e.g., accuracy,MSE) with the problem context — especially in imbalanced or critical domains.
- **Model Training is the Heart of Learning**
  - Involves both **optimization** (loss minimization) and **hyperparameter tuning (HPT)** to fine-tune performance.
- **Deployment Strategy Depends on Use Case**
  - The final deployment platform (cloud, edge, web, embedded) should match application constraints like latency, memory, and connectivity.


Optimization
Generalization


Raw Data & Problem Definition
Explanatory Data Analysis
Model selection from archive
Preprocessing
Model Optimization + HPT
Yes
Train_set
Try another model?
Train/Test/Validation Split
NO
Model Evaluation
Test_set
Choose best model
Final Model Selection
Result archive
Model deployment

Heart Disease

Donated on 6/30/1988

4 databases: Cleveland, Hungary, Switzerland, and the VA Long Beach

Dataset Characteristics

Multivariate

Subject Area

Health and Medicine

Associated Tasks

Classification

- The full dataset has **76 attributes**, but only **14 features** are used in most studies.
- It is a **multivariate numerical dataset** suited for statistical/machine learning analysis.
- The most widely used subset is the **Cleveland dataset**.
- The primary goal is to **predict heart disease presence** based on patient attributes.
- The `goal` **field** (aka `num`) is an integer from **0 to 4**:
  - `0` = no heart disease
  - `1-4` = increasing severity of heart disease
- Most ML experiments treat it as a **binary classification problem**:
  - Class 0: **No disease**
  - Class 1: **Presence of disease (1, 2, 3, 4)**

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        920 non-null    int64
 1   age       920 non-null    int64
 2   sex       920 non-null    object
 3   dataset   920 non-null    object
 4   cp        920 non-null    object
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object
 8   restecg   918 non-null    object
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object
 13  ca        309 non-null    float64
 14  thal      434 non-null    object
 15  num       920 non-null    int64
```

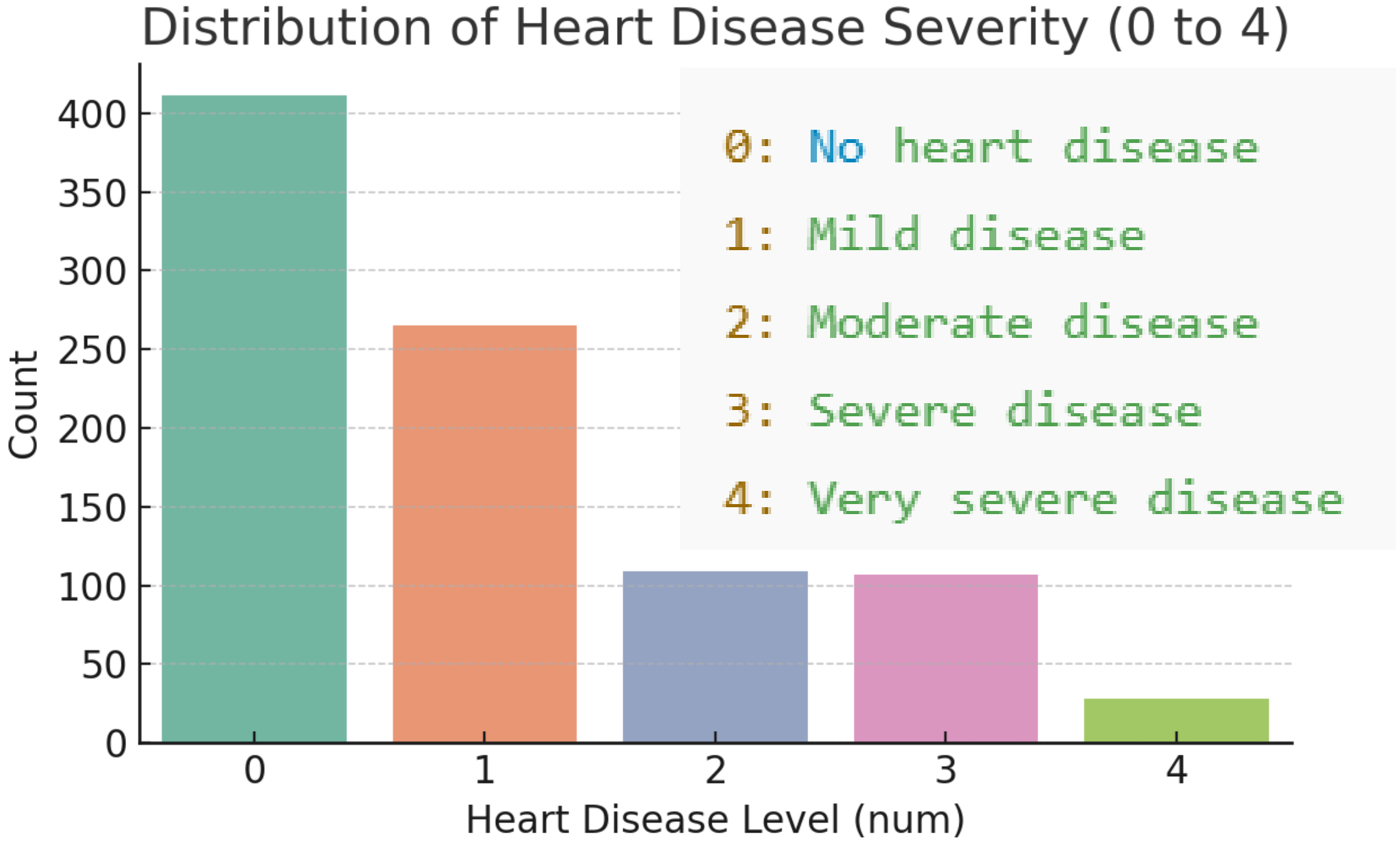
📊 Core 14 Features Commonly Used

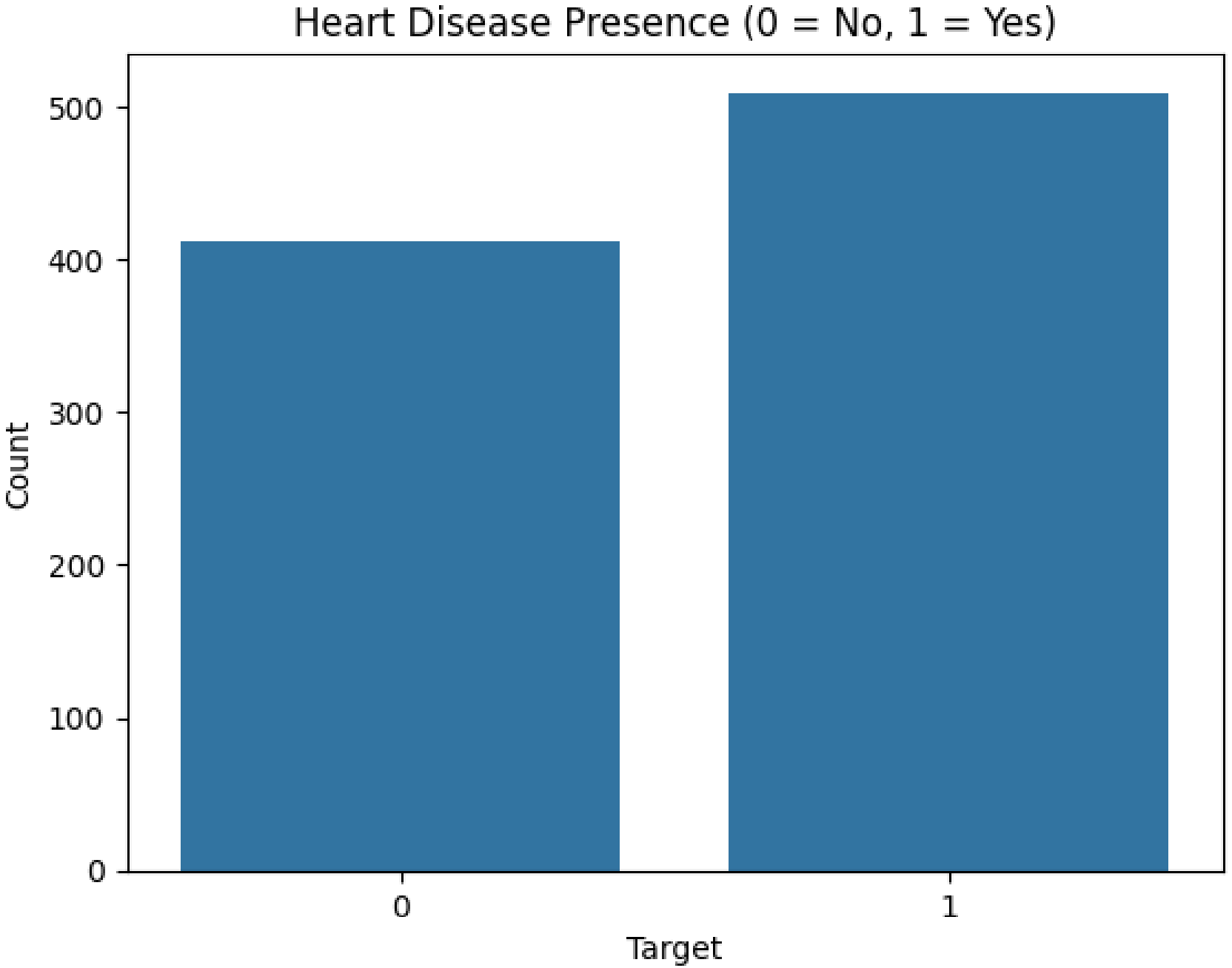
1. Age
2. Sex
3. Chest Pain Type (`cp`)
4. Resting Blood Pressure (`trestbps`)
5. Serum Cholesterol (`chol`)
6. Fasting Blood Sugar (`fbs`)
7. Resting ECG (`restecg`)
8. Max Heart Rate Achieved (`thalach`)
9. Exercise-induced Angina (`exang`)
10. ST Depression (`oldpeak`)
11. Slope of ST Segment (`slope`)
12. Number of Major Vessels Colored (`ca`)
13. Thalassemia (`thal`)
14. Target Variable (`num` or binary `target`)

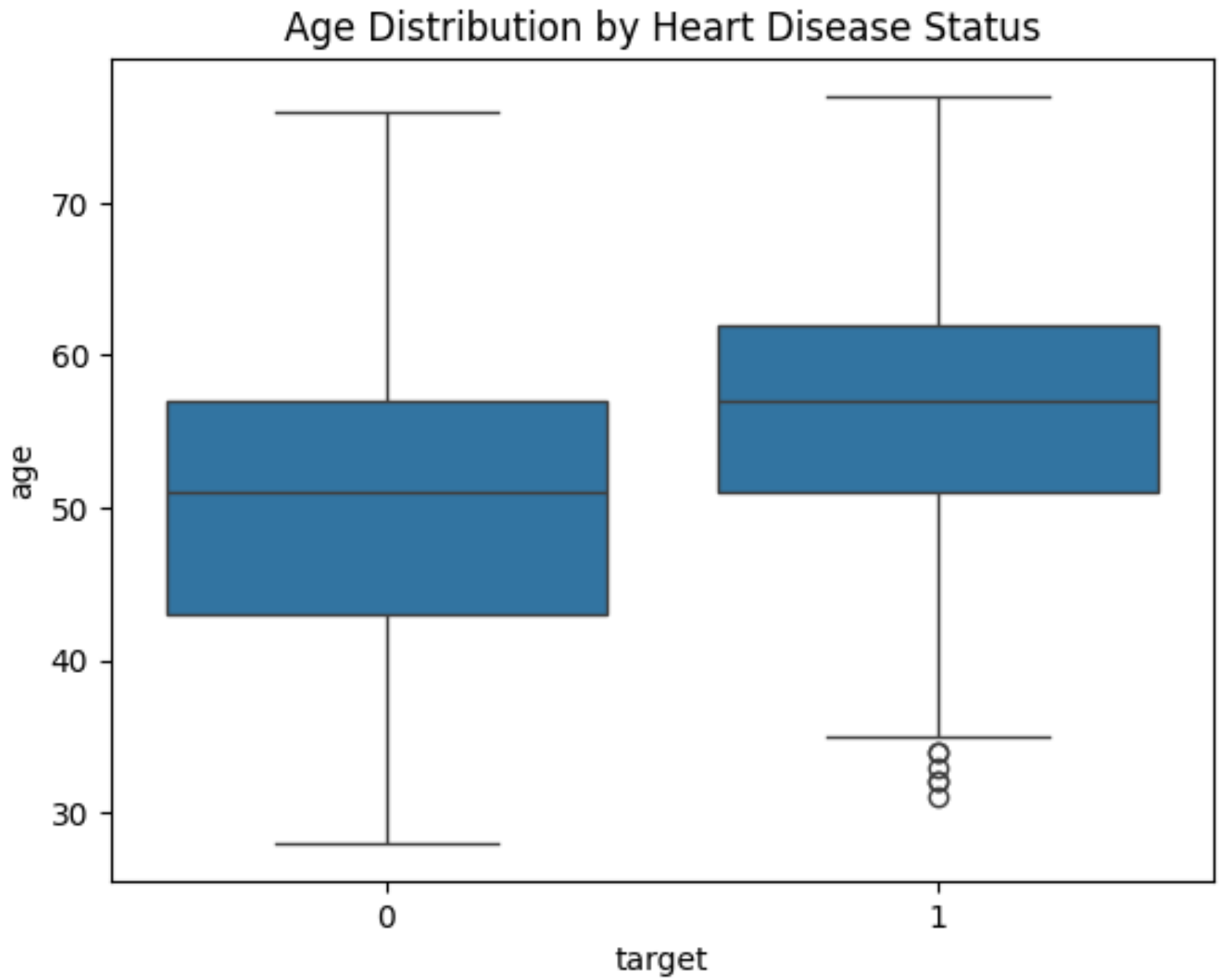
Column Name	Description
age	Age of the patient (in years)
sex	Sex of the patient (male / female)
dataset	Source dataset name (e.g., Cleveland, Hungarian, etc.)
cp	Chest pain type: typical angina, atypical angina, non-anginal, asymptomatic
trestbps	Resting blood pressure (in mm Hg)
chol	Serum cholesterol (in mg/dl)
fbs	Fasting blood sugar > 120 mg/dl (true/false)
restecg	Resting electrocardiographic results: normal, ST-T abnormality, LV hypertrophy
thalach	Maximum heart rate achieved
exang	Exercise-induced angina (true/false)
oldpeak	ST depression induced by exercise relative to rest
slope	Slope of the peak exercise ST segment: upsloping, flat, downsloping
ca	Number of major vessels (0–3) colored by fluoroscopy
thal	Thalassemia: normal, fixed defect, reversible defect
num	Heart disease diagnosis: 0 = no disease, 1–4 = presence of disease

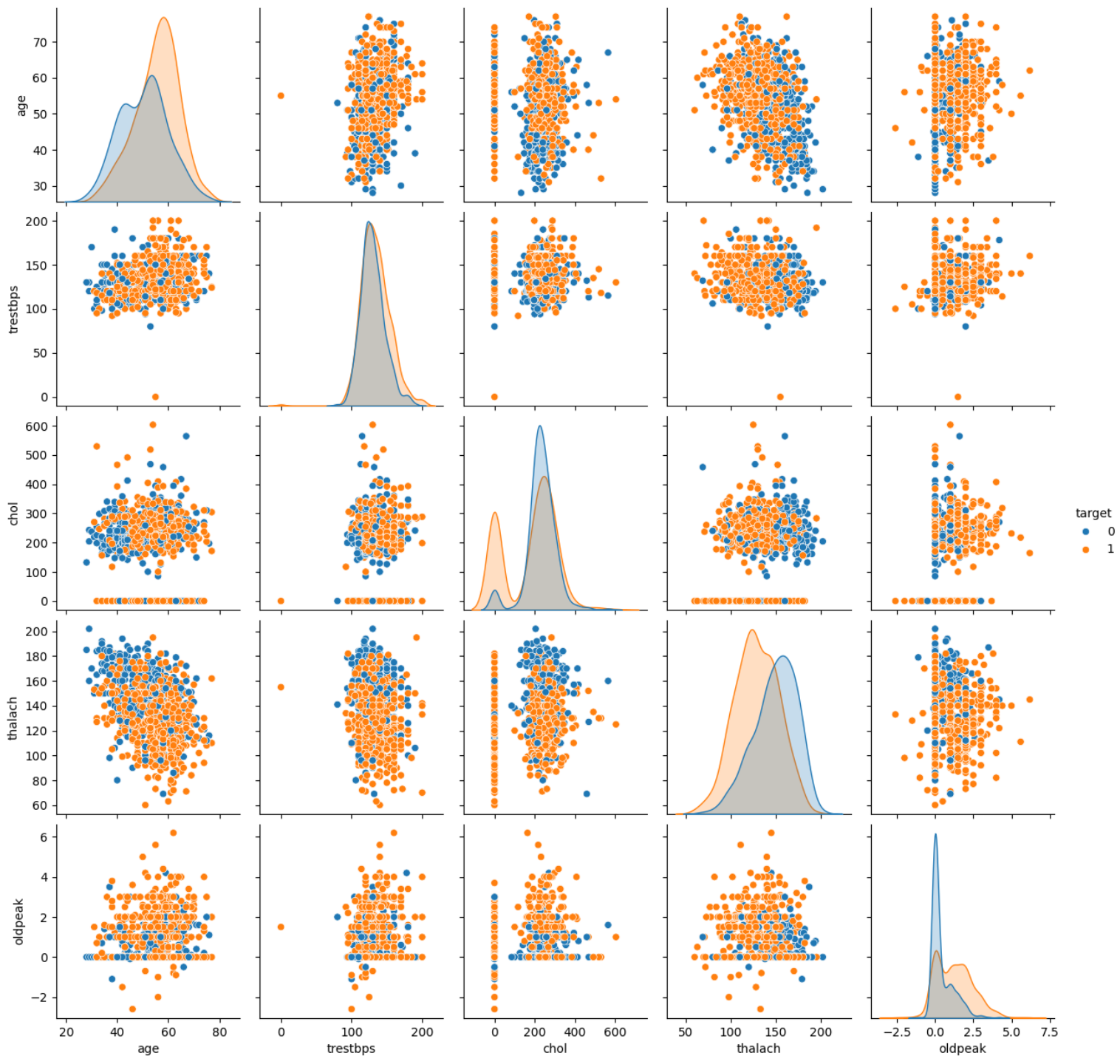
EDA Summary

	Type	Missing	Unique	Mean	Std	Min	Max	Skew
age	Numerical	0	50	53.51087	9.424685	28	77	-0.195994
trestbps	Numerical	59	61	132.132404	19.06607	0.0	200.0	0.213334
chol	Numerical	30	217	199.130337	110.78081	0.0	603.0	-0.613836
thalach	Numerical	55	119	137.545665	25.926276	60.0	202.0	-0.211119
oldpeak	Numerical	62	53	0.878788	1.091226	-2.6	6.2	1.041427
ca	Numerical	611	4	0.676375	0.935653	0.0	3.0	1.165978
target	Numerical	0	2	0.553261	0.497426	0	1	-0.214613
sex	Categorical	0	2	NaN	NaN	NaN	NaN	NaN
dataset	Categorical	0	4	NaN	NaN	NaN	NaN	NaN
cp	Categorical	0	4	NaN	NaN	NaN	NaN	NaN
fbs	Categorical	90	2	NaN	NaN	NaN	NaN	NaN
restecg	Categorical	2	3	NaN	NaN	NaN	NaN	NaN
exang	Categorical	55	2	NaN	NaN	NaN	NaN	NaN
slope	Categorical	309	3	NaN	NaN	NaN	NaN	NaN


Distribution of Heart Disease Severity (0 to 4)
0: No heart disease
1: Mild disease
2: Moderate disease
3: Severe disease
4: Very severe disease
Count
400
350
300
250
200
150
100
50
0
0
1
2
3
4
Heart Disease Level (num)


Heart Disease Presence (0 = No, 1 = Yes)
Count
500
400
300
200
100
0
0
1
Target


Age Distribution by Heart Disease Status
70
60
50
40
30
age
0
1
target

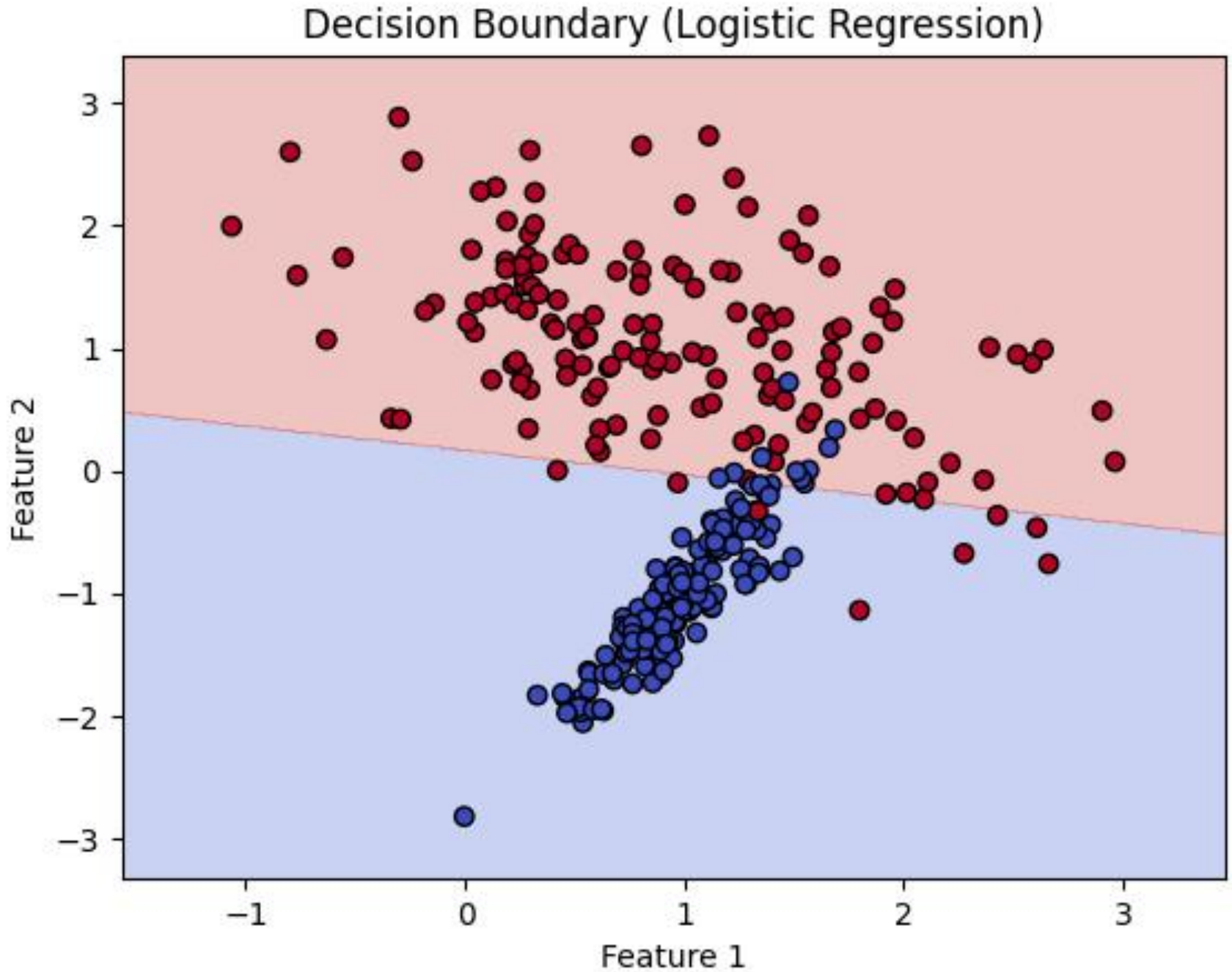

age
trestbps
chol
thalach
oldpeak
target
0
1

Class Count and Percentage Table:

	Count	
	Healthy (0)	Patient (1)
Full Dataset	411	509
Train Set	329	407
Test Set	82	102

Black-box functions

Model	Optimization Algorithm	Hyperparameter Tuning	Notes
Logistic Regression	Gradient Descent (`liblinear` , `saga`)	✅ Randomized Search	Tuned `C` , `penalty` , and `solver`
Random Forest	Ensemble of Decision Trees (Greedy Split)	✅ Randomized Search	Tuned `n_estimators` , `max_depth` , `min_samples_split` , etc.
Support Vector Machine (SVM)	Quadratic Programming (SMO)	✅ Randomized Search	Tuned `C` , `gamma` , `kernel`
XGBoost	Boosting (Gradient Tree Boosting)	✅ Randomized Search	Tuned `n_estimators` , `max_depth` , `learning_rate` , etc.


Decision Boundary (Logistic Regression)
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−1
0
1
2
3

✅ 1. Random Forest

- **Total actual patients (class 1): 102**
  - ✅ Correctly predicted as patient: **87**
  - ❌ Misclassified as healthy: **15**
- **Total actual healthy (class 0): 82**
  - ✅ Correctly predicted as healthy: **62**
  - ❌ Misclassified as patient: **20**

✅ 2. Support Vector Machine (SVM)

- **Total actual patients (class 1): 102**
  - ✅ Correctly predicted as patient: **80**
  - ❌ Misclassified as healthy: **22**
- **Total actual healthy (class 0): 82**
  - ✅ Correctly predicted as healthy: **64**
  - ❌ Misclassified as patient: **18**

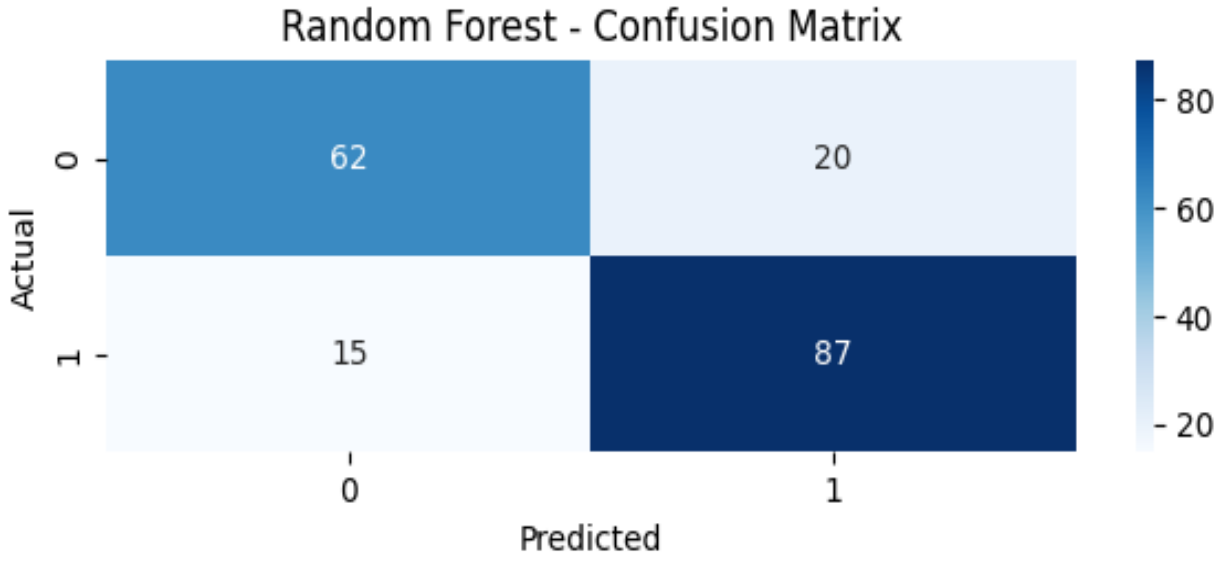
✅ 3. Logistic Regression

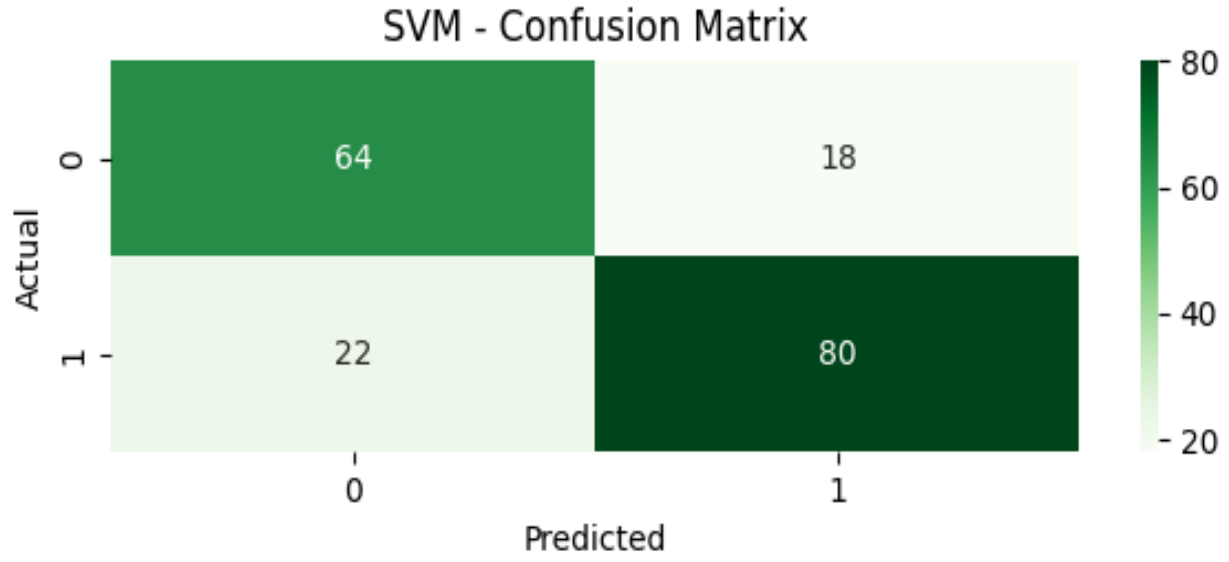
- **Total actual patients (class 1): 102**
  - ✅ Correctly predicted as patient: **82**
  - ❌ Misclassified as healthy: **20**
- **Total actual healthy (class 0): 82**
  - ✅ Correctly predicted as healthy: **65**
  - ❌ Misclassified as patient: **17**

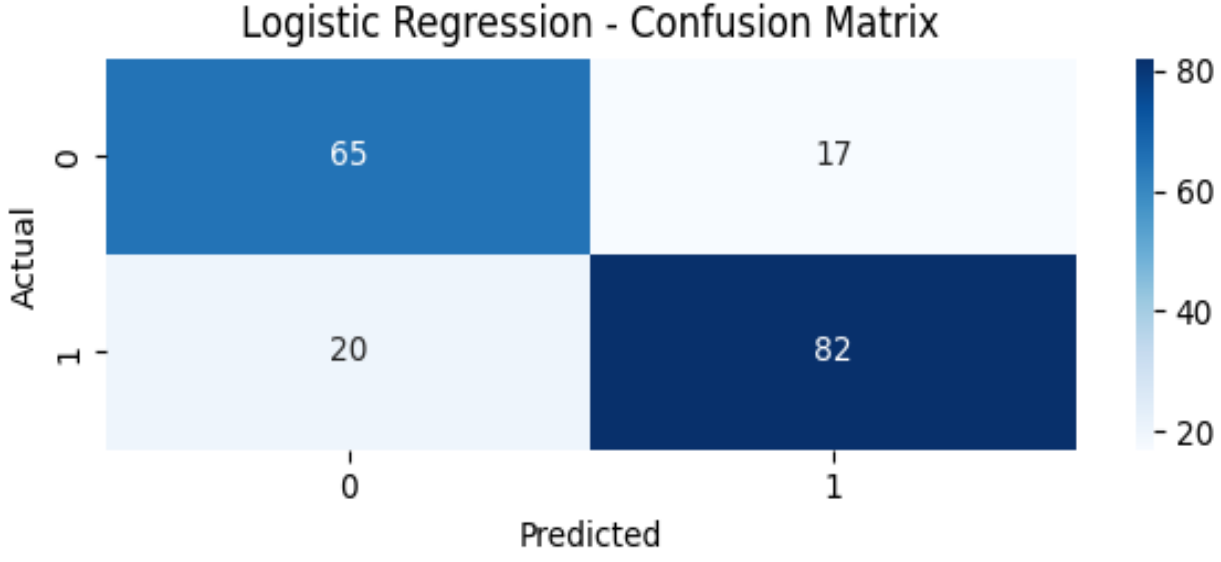
✅ 4. XGBoost

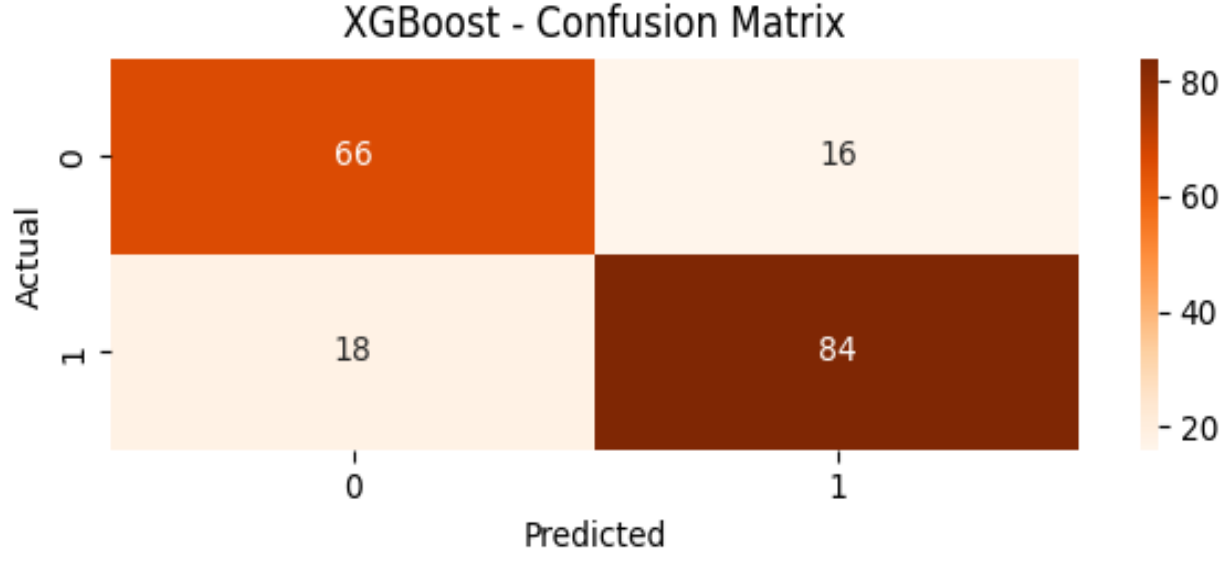
- **Total actual patients (class 1): 102**
  - ✅ Correctly predicted as patient: **84**
  - ❌ Misclassified as healthy: **18**
- **Total actual healthy (class 0): 82**
  - ✅ Correctly predicted as healthy: **66**
  - ❌ Misclassified as patient: **16**

	Classifier	Accuracy (%)
0	Logistic Regression	0.80
1	Random Forest	0.81
2	SVM	0.78
3	XGBoost	0.82


Random Forest - Confusion Matrix
Actual
Predicted
0
1
62
20
15
87
80
60
40
20


SVM - Confusion Matrix
Actual
Predicted
0
1
64
18
22
80
80
60
40
20


Logistic Regression - Confusion Matrix
Actual
Predicted
0
1
65
17
20
82
80
60
40
20


XGBoost - Confusion Matrix
Actual
Predicted
0
1
66
16
18
84
80
60
40
20

✅ **Tells us**: *Out of all predicted patients, how many were truly patients?*

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

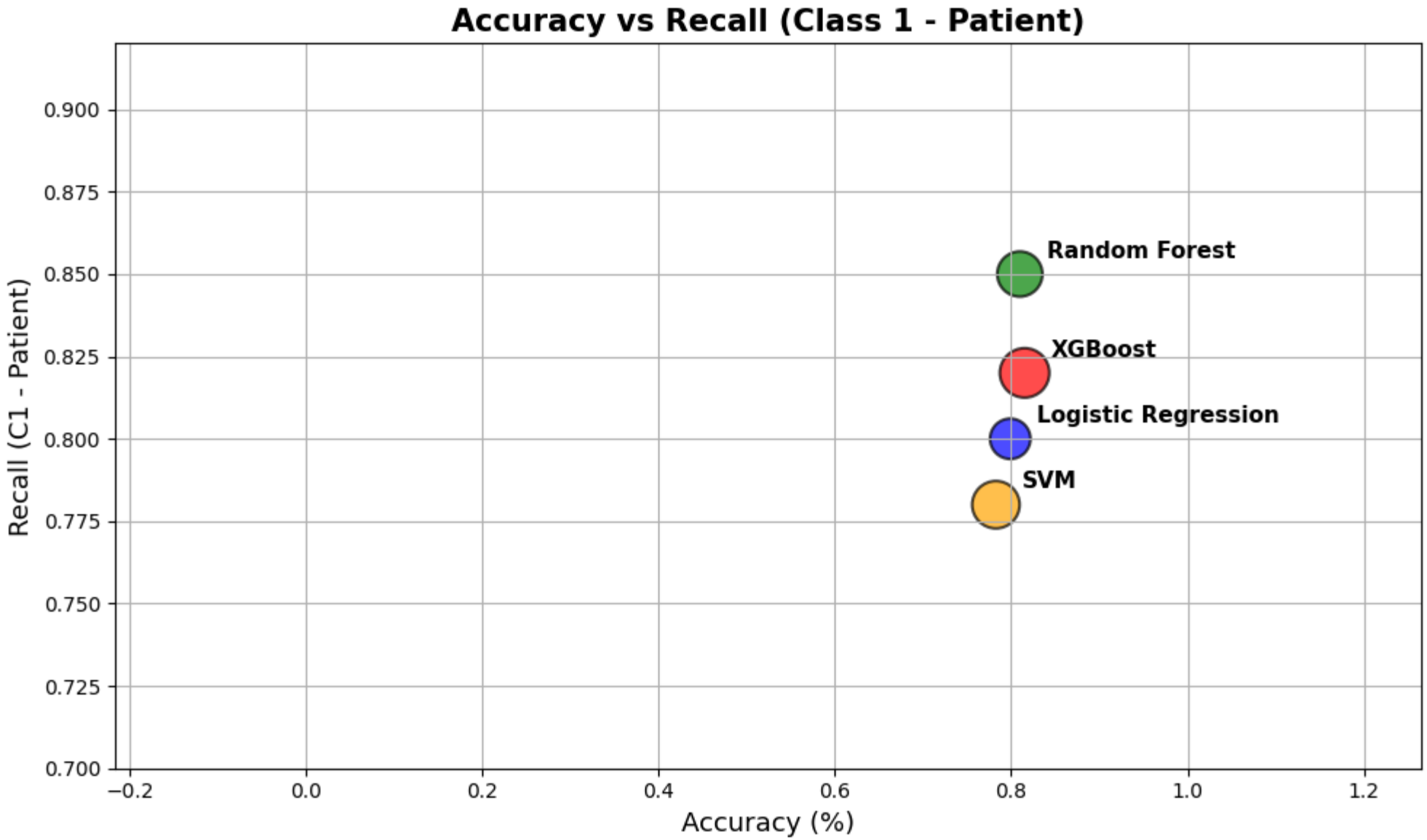
$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

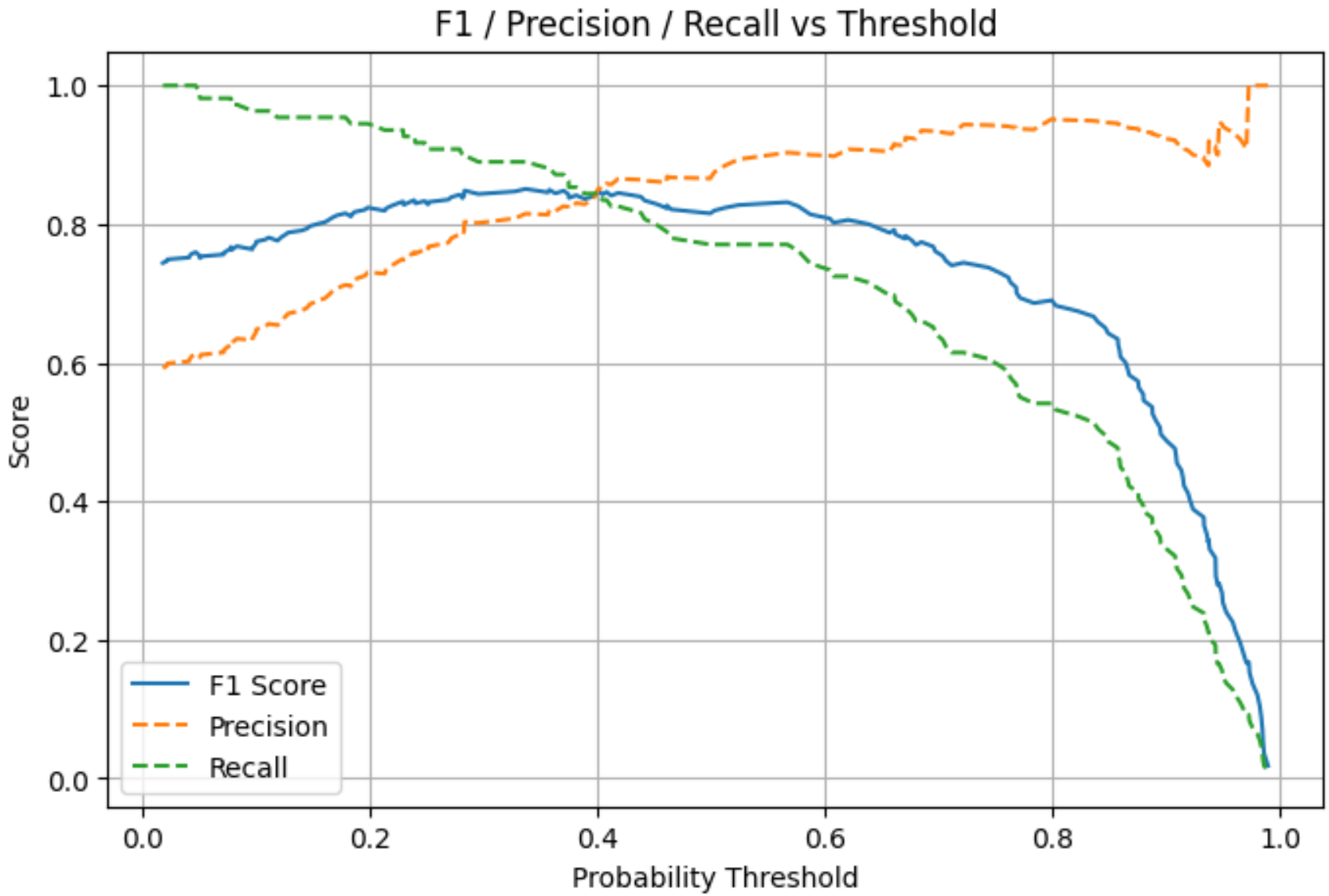
✅ **Tells us**: *Out of all actual patients, how many did we correctly identify as patients?*

⚖️ Key Trade-Off

- **High Recall**: Few false negatives → catches more patients, even if it includes some healthy people.
- **High Precision**: Few false positives → more trustworthy predictions, but might miss some real patients.

Model	Precision	Recall	F1-Score
Random Forest	0.8131	0.8529	0.8324
SVM	0.8163	0.7843	0.8000
Logistic Regression	0.8283	0.8039	0.8159
XGBoost	0.8400	0.8235	0.8317


Accuracy vs Recall (Class 1 - Patient)
Recall (C1 - Patient)
Accuracy (%)
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Random Forest
XGBoost
Logistic Regression
SVM


F1 / Precision / Recall vs Threshold
Score
Probability Threshold
F1 Score
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

Deployment

💓 Heart Disease Risk Predictor

Enter the patient information to check the risk.

Age

55

29 77

Sex

- (•) Male
- ( ) Female

Chest Pain Type

typical angina

Resting Blood Pressure (mm Hg)

150

90 200

Serum Cholesterol (mg/dl)

365

100 600

Fasting Blood Sugar > 120 mg/dl

- (•) True
- ( ) False

🔍 Prediction:

Result: ✅ No Heart Disease

Risk Probability: **38.00%**

What next?

Day 6

🩺 1. What is a Threshold in Classification?

Most classifiers (like Logistic Regression, Neural Networks) output a **probability score** between 0 and 1.

$$\text{If } P(\text{Heart Disease}) > \text{threshold} \Rightarrow \text{Predict "Disease"}$$

- **Default threshold**: 0.5
- But for **critical problems like healthcare**, **0.5 may not be optimal**.

Case study

Suppose we train a binary classifier to predict whether a person has heart disease or not.

Imagine we have tested **200 patients** using a heart disease prediction model. The outcomes are as follows:

- **40 patients** were correctly identified by the model as having heart disease.
- **10 patients** actually had heart disease, but the model incorrectly predicted them as healthy.
- **15 patients** were actually healthy, but the model wrongly predicted that they had heart disease.
- **135 patients** were correctly identified as healthy by the model.

Confusion Matrix in Heart Disease Prediction

The model gives two possible outputs:

- **Positive** → Predicts patient **has** heart disease
- **Negative** → Predicts patient **does not have** heart disease

Term	Meaning
True Positive (TP)	Model predicts **disease**, and patient **actually has** disease
False Positive (FP)	Model predicts **disease**, but patient is **healthy**
True Negative (TN)	Model predicts **no disease**, and patient is **healthy**
False Negative (FN)	Model predicts **no disease**, but patient **actually has** disease

Term	Meaning
True Positive (TP)	Model predicts **disease**, and patient **actually has** disease
False Positive (FP)	Model predicts **disease**, but patient is **healthy**
True Negative (TN)	Model predicts **no disease**, and patient is **healthy**
False Negative (FN)	Model predicts **no disease**, but patient **actually has** disease

Imagine we have tested **200 patients** using a heart disease prediction model. The outcomes are as follows:

- **40 patients** were correctly identified by the model as having heart disease.
- **10 patients** actually had heart disease, but the model incorrectly predicted them as healthy.
- **15 patients** were actually healthy, but the model wrongly predicted that they had heart disease.
- **135 patients** were correctly identified as healthy by the model.

Confusion Matrix Representation

	Predicted: Disease (+)	Predicted: No Disease (−)
Actual: Disease (+)	TP	FN
Actual: No Disease (−)	FP	TN

	Predicted: Disease (+)	Predicted: No Disease (−)
Actual: Disease (+)	40 (TP)	10 (FN)
Actual: No Disease (−)	15 (FP)	135 (TN)

Accuracy?

Why This Matters in Healthcare

- **False Negatives (FN)** are critical → a patient with disease goes undetected!
- **False Positives (FP)** can cause stress or unnecessary tests.

Why Accuracy Alone Doesn’t Tell the Whole Story?

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

It doesn’t tell the type of errors — and that’s where it fails.

Heart Disease Prediction

Let’s say: **(200 Patients)**

- 190 patients are healthy
- 10 patients have heart disease
- The model predicts **“no disease” for all 200**

	Predicted: Disease	Predicted: No Disease
Actual: Disease (10)	0	10 (all missed)
Actual: No Disease (190)	0	190

🧮 Accuracy = 190 / 200 = **95%**

But it detected **0 cases of heart disease!**

> 👉 **High accuracy, zero usefulness.**

Better Metrics to Use

Metric	Why it Helps
Precision	How many predicted positives are correct
Recall	How many actual positives were detected
F1 Score	Balance between precision and recall

Email Spam Classifier

Confusion Matrix Example (100 emails):

	Predicted Spam	Predicted Not Spam
Actual Spam	35 (TP)	5 (FN)
Actual Not Spam	10 (FP)	50 (TN)

- **Accuracy** = (TP + TN) / Total = (35 + 50) / 100 = **85%**
- **Precision** = TP / (TP + FP) = 35 / (35 + 10) = **77.78%**
- **Recall** = TP / (TP + FN) = 35 / (35 + 5) = **87.5%**
- **FPR** = FP / (FP + TN) = 10 / (10 + 50) = **16.67%**

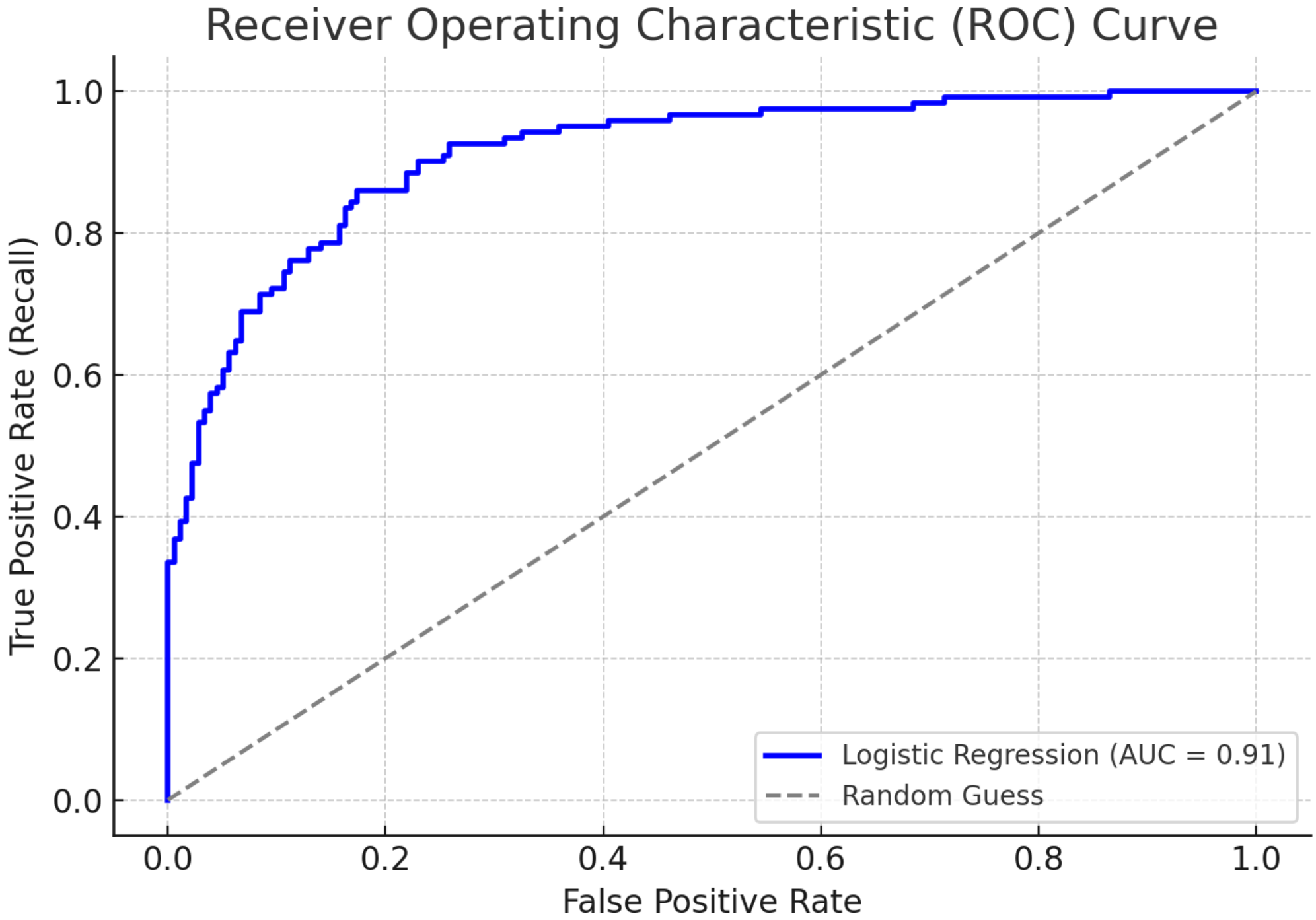
Netflix Churn Prediction

- Predicting whether a user will cancel the subscription in the next month

Confusion Matrix Example (300 users):

	Predicted Churn	Predicted Stay
Actual Churn	60 (TP)	20 (FN)
Actual Not Churn	30 (FP)	190 (TN)

- **Accuracy** = (60 + 190) / 300 = **83.3%**
- **Precision** = 60 / (60 + 30) = **66.7%**
- **Recall** = 60 / (60 + 20) = **75%**
- **FPR** = 30 / (30 + 190) = **13.6%**

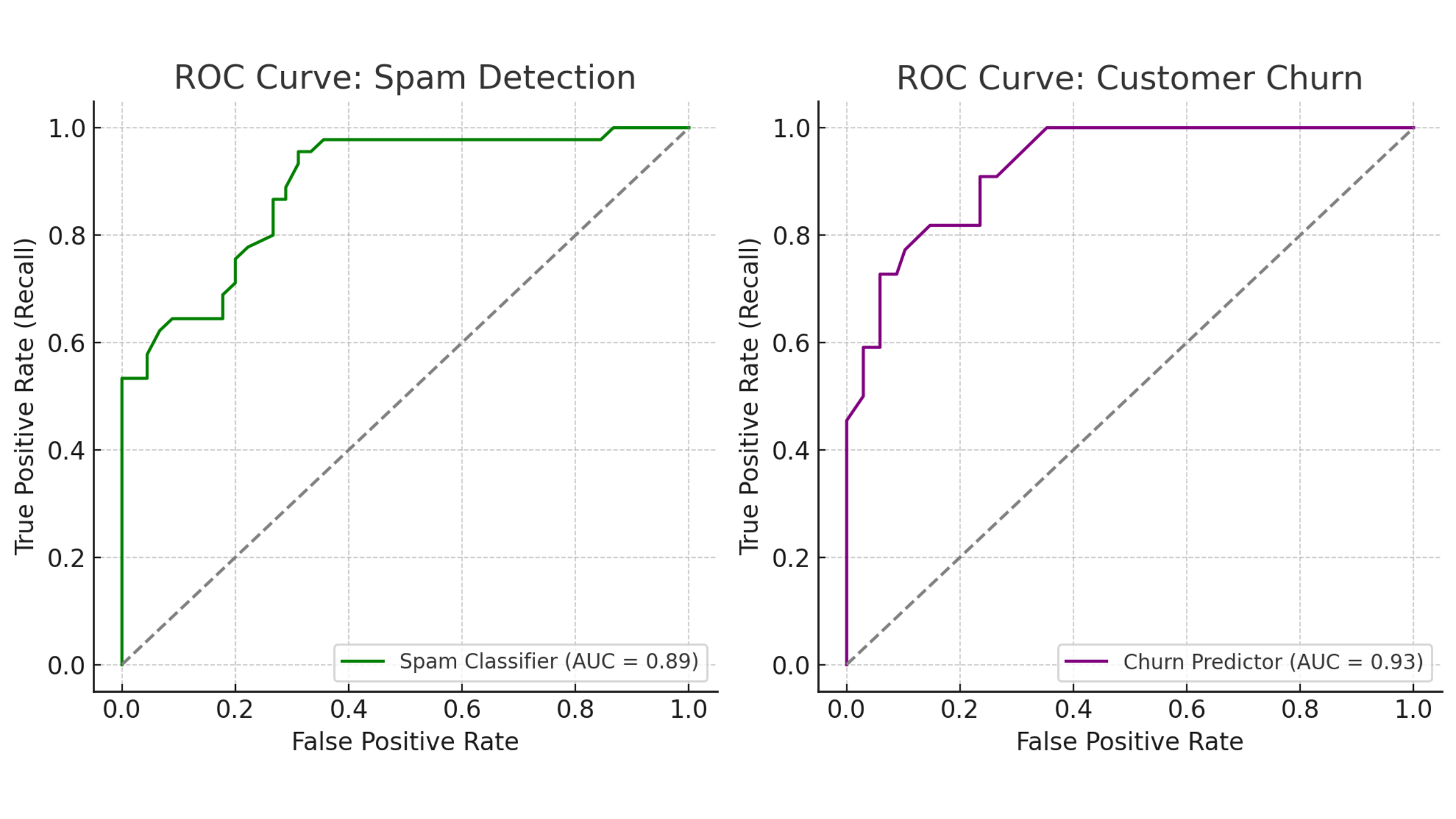

Receiver Operating Characteristic (ROC) Curve
True Positive Rate (Recall)
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression (AUC = 0.91)
Random Guess

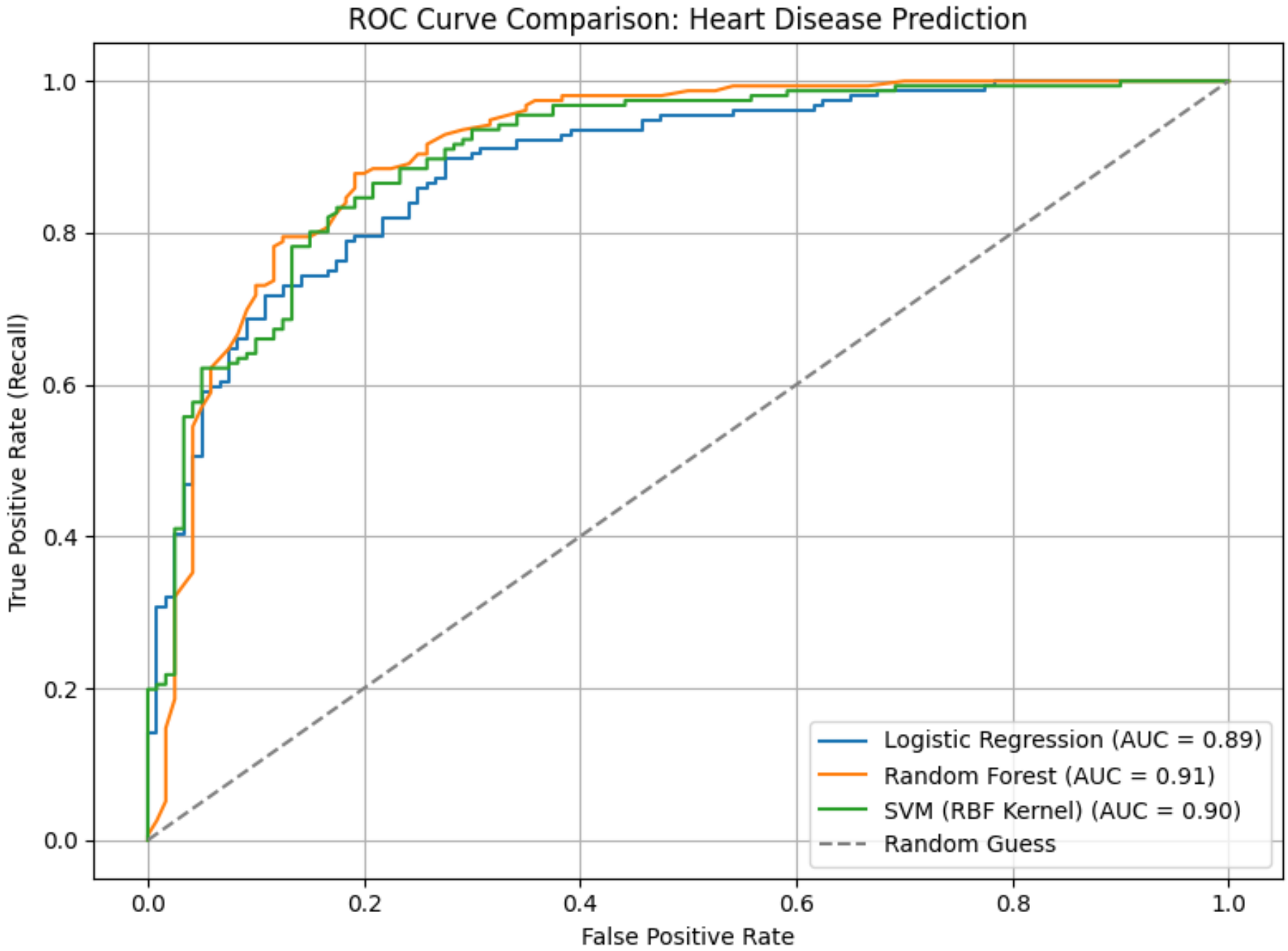
TPR → Benefit

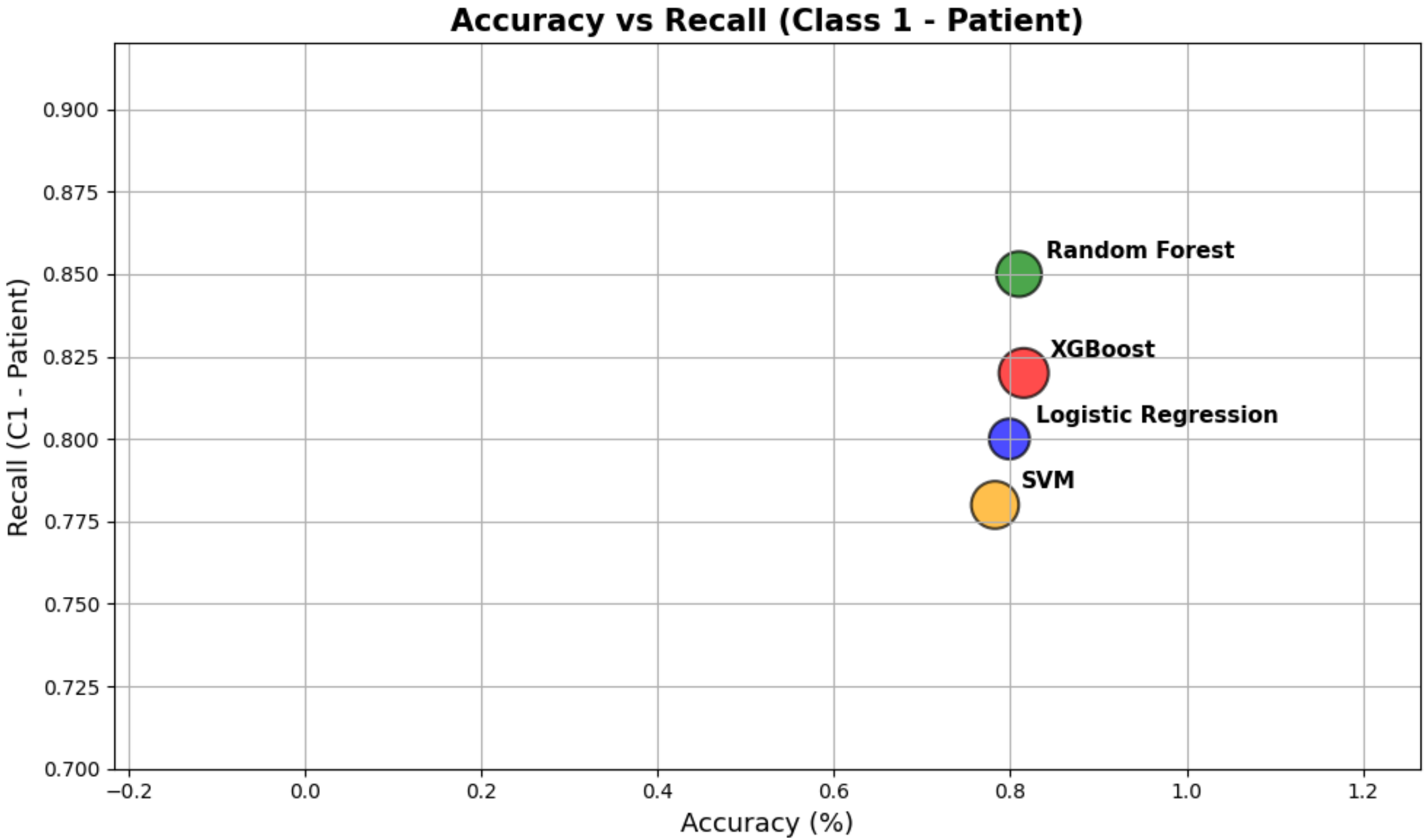
FPR → Cost

- **ROC Curve:**
  - The **x-axis** is the **False Positive Rate (FPR)**:
  - The **y-axis** is the **True Positive Rate (Recall)**
  - The **blue curve** shows how the model performs at various thresholds.
  - The **gray dashed line** is a baseline: random guessing.

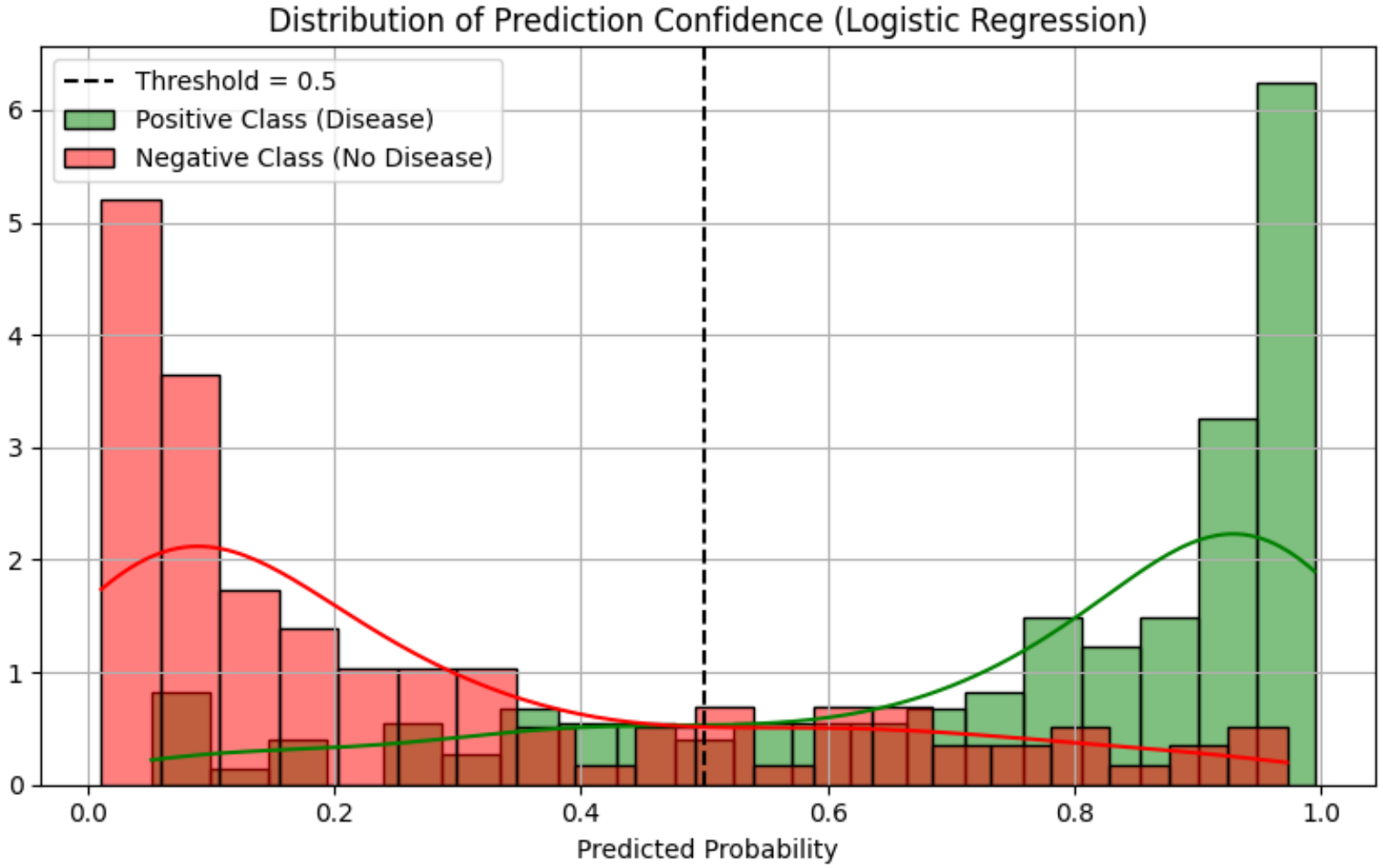
- **AUC (Area Under the Curve):**
  - AUC = 1.0 → perfect classifier
  - AUC = 0.5 → random guessing
  - AUC > 0.8 → good model


ROC Curve: Spam Detection
True Positive Rate (Recall)
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Spam Classifier (AUC = 0.89)
ROC Curve: Customer Churn
True Positive Rate (Recall)
False Positive Rate
Churn Predictor (AUC = 0.93)

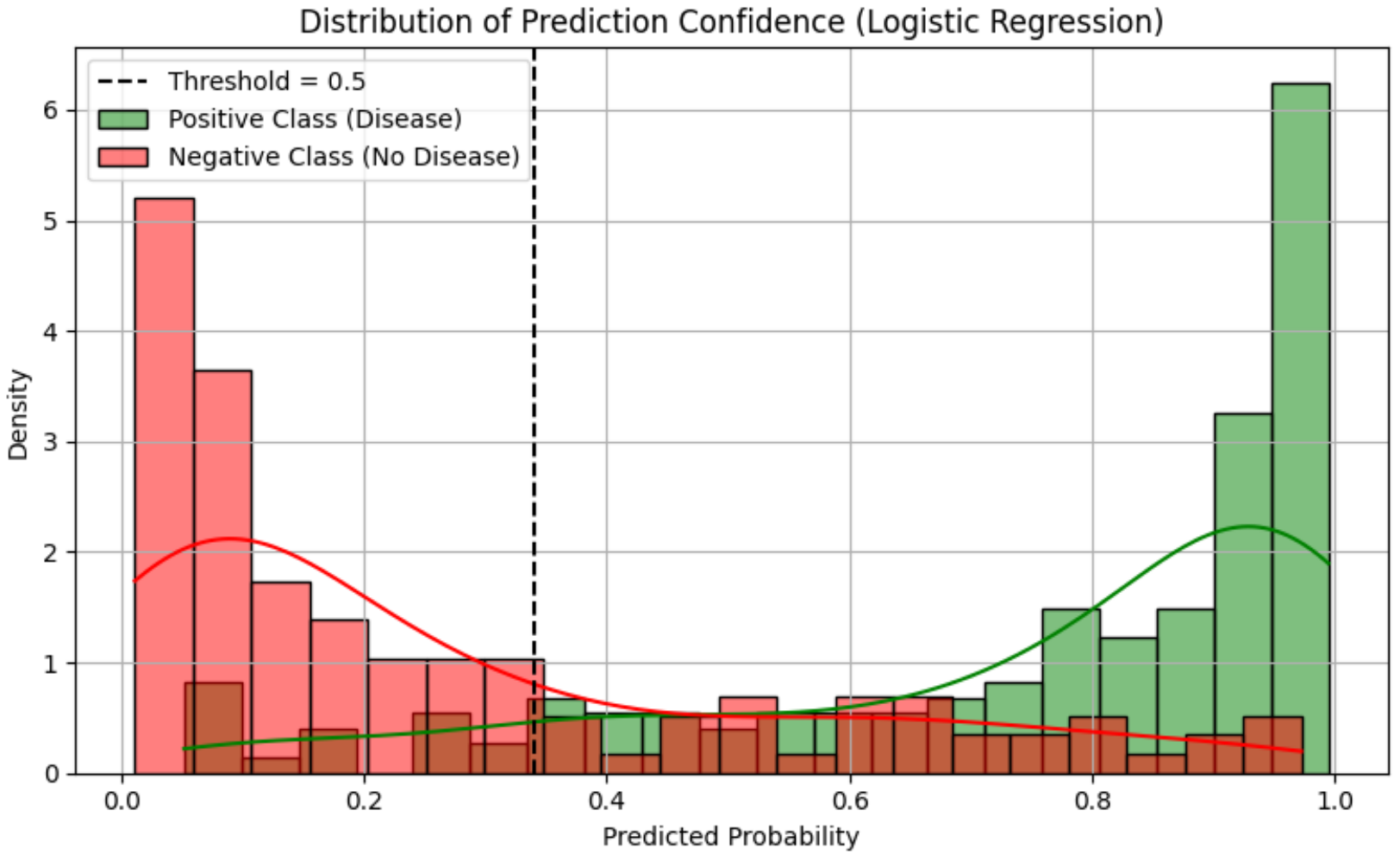

ROC Curve Comparison: Heart Disease Prediction
True Positive Rate (Recall)
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression (AUC = 0.89)
Random Forest (AUC = 0.91)
SVM (RBF Kernel) (AUC = 0.90)
Random Guess


Accuracy vs Recall (Class 1 - Patient)
Recall (C1 - Patient)
Accuracy (%)
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Random Forest
XGBoost
Logistic Regression
SVM

Threshold = 0.5
True Positives (TP): 126
False Positives (FP): 26
False Negatives (FN): 30
True Negatives (TN): 94



Threshold = 0.34
True Positives (TP): 140
False Positives (FP): 33
False Negatives (FN): 16
True Negatives (TN): 87



Summary

- First define the objective and then design a model
- For any model, HPT will affect the performance
- By default, Threshold = 0.5 accuracy may not be optimal for all classification problems
- To set threshold, use TPR, FPR, F1-score (Precision and Recall) as per requirement.

- Optimizing model hyperparameters may give you best AUC. But AUC doesn't tell you which threshold to use.
- A sub-optimal threshold could give poor recall (missed patients) even for a high-AUC model.
- Jointly tuning threshold + hyperparams ensures best performance for application-specific goals.

1. Understand the differences between shallow and deep learning models.
2. Apply shallow learning models to handcrafted features from images/audio.
3. Understand feature extraction limitations.
4. Apply deep learning models directly on raw/unstructured data.

Day 7

Recap

- End-to-End-ML pipeline (discriminative learning)

Data→FE→Model training→Evaluate (training pipeline) → Deploy

Test_data→Trained_model→check_prediction (Inference)

- Familiar with Different Terms
- How to compare models? In terms of Evaluation metrics
- How to set "decision threshold" or "classification threshold" in machine learning
- **Case study: Heart disease prediction**
  - **-- Tabular: Structured in csv form**
  - **-- Data size: Small**

What is Computer Vision?

Computer Vision (CV) is a field of Artificial Intelligence (AI) that enables computers to **see, interpret, and understand visual information** from the world — just like humans do.

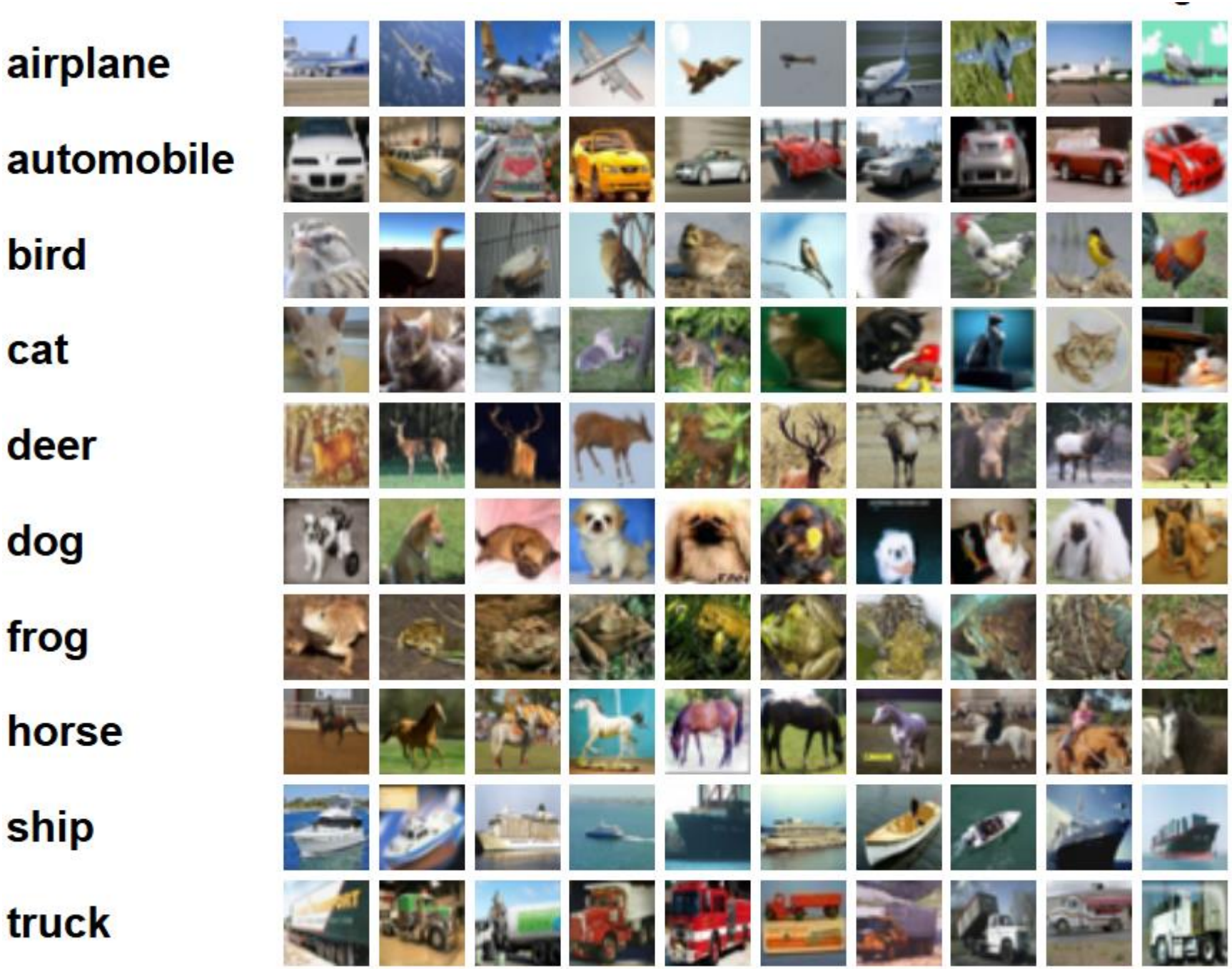
Task	Description
Image Classification	What is in the image? (e.g., dog, car, tree)
Object Detection	Where are the objects in the image?
Face Recognition	Who is in the image?
Image Segmentation	Which pixels belong to which object?

CV-Problem Statement

Design and evaluate image classification models that can accurately classify small natural images into 10 distinct categories.

Dataset Description: CIFAR-10

Name	CIFAR-10 (Canadian Institute for Advanced Research - 10 classes)
Source	Collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton
Dataset Type	Image classification
Image Size	32 × 32 pixels, RGB color images


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

Total Images	60,000
Training Images	50,000
Test Images	10,000
Number of Classes	10

Characteristics

- Balanced dataset: 6,000 images per class
- Highly challenging due to small image size and inter-class similarity
- Real-world scenes with high intra-class variation and background clutter

What's Next?

CIFAR-10 image classification problem, which involves assigning an input image to one of ten object categories.

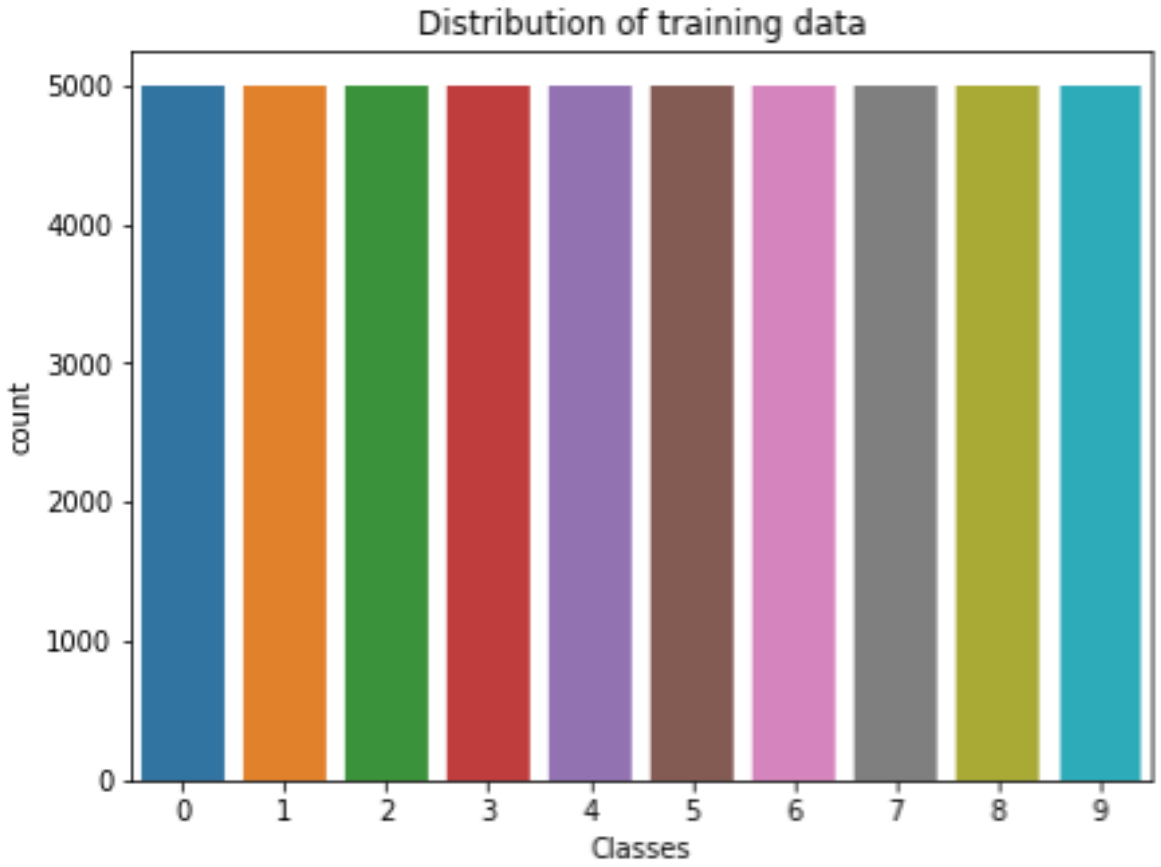
We begin with a machine learning approaches, using handcrafted features obtained from domain knowledge of Image Processing

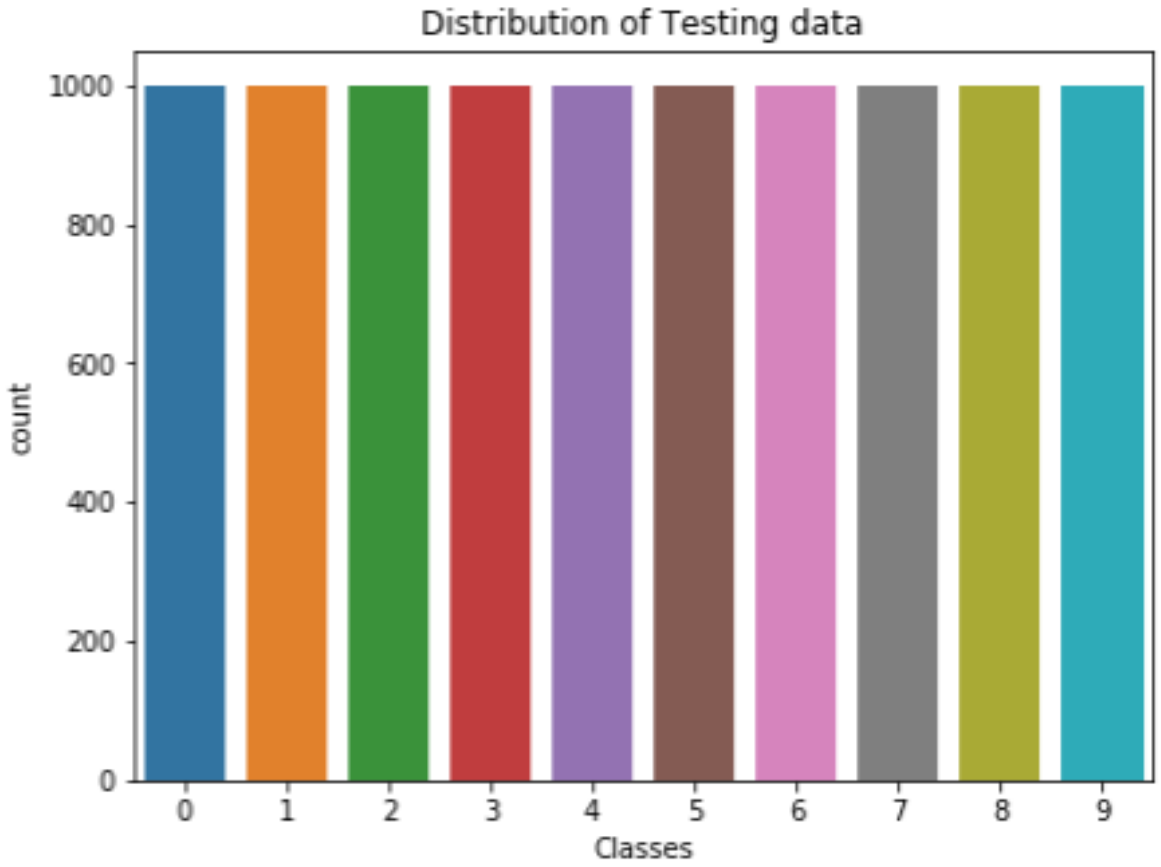
- Feature: Histogram of Oriented Gradients (HOG), gray/color histograms.
- Models: SVM and Random Forest are trained on these features.

What is the HOG?

HOG stands for Histogram of Oriented Gradients — a handcrafted feature descriptor used in traditional computer vision for object detection and image classification.

HOG captures the shape and edge information in an image by analyzing the directions of gradients (edges) in small regions of the image.


Distribution of training data
count
Classes
0
1000
2000
3000
4000
5000
0 1 2 3 4 5 6 7 8 9


Distribution of Testing data
count
Classes
0
200
400
600
800
1000
0 1 2 3 4 5 6 7 8 9

HOG does this by:

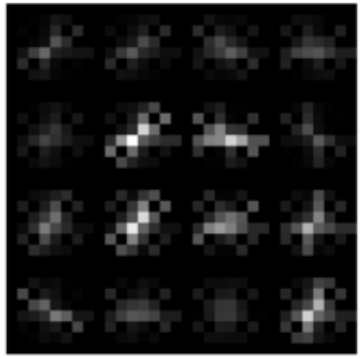
1. **Dividing** the image into small patches (called **cells**).
2. **Computing edge directions** (gradients) in each cell.
3. **Making a histogram** of edge orientations in each patch.
4. **Normalizing** them across neighboring cells (called **blocks**) for contrast invariance.

**You can use also

LBP, SIFT, Gabor filter

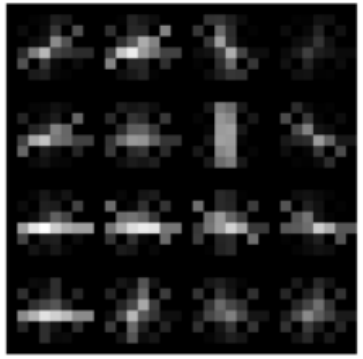
Original Image

HOG Features



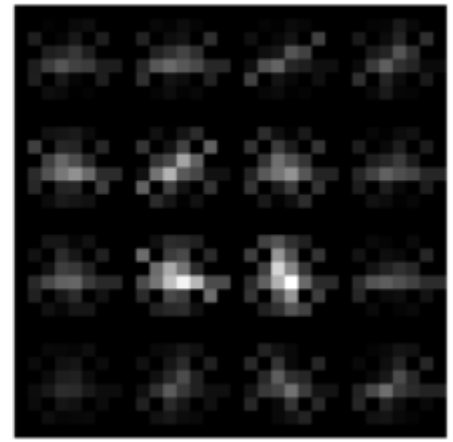
Original Image

HOG Features



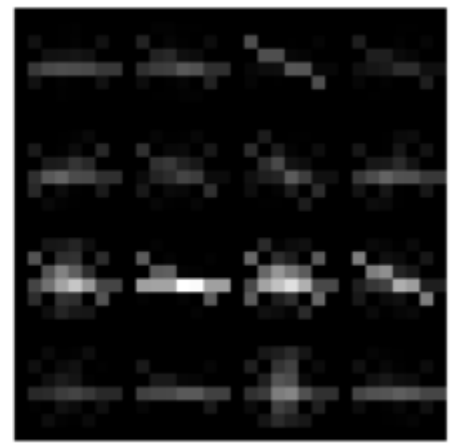
Original Image

HOG Features



Original Image

HOG Features



HOG Feature vector length per image : 324

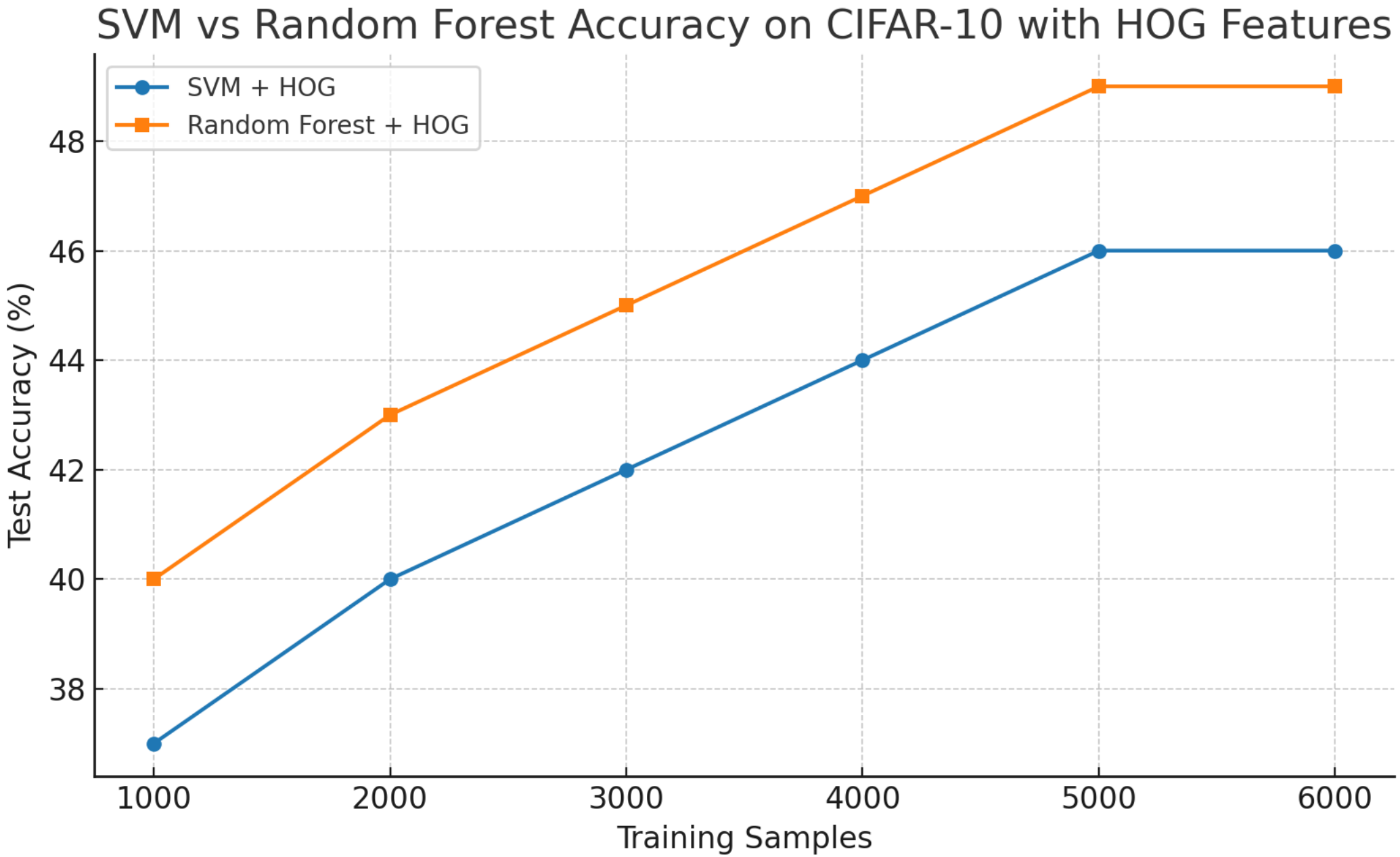
Breaks down in terms of sizes and shapes — both for the original image data and the HOG feature vectors.

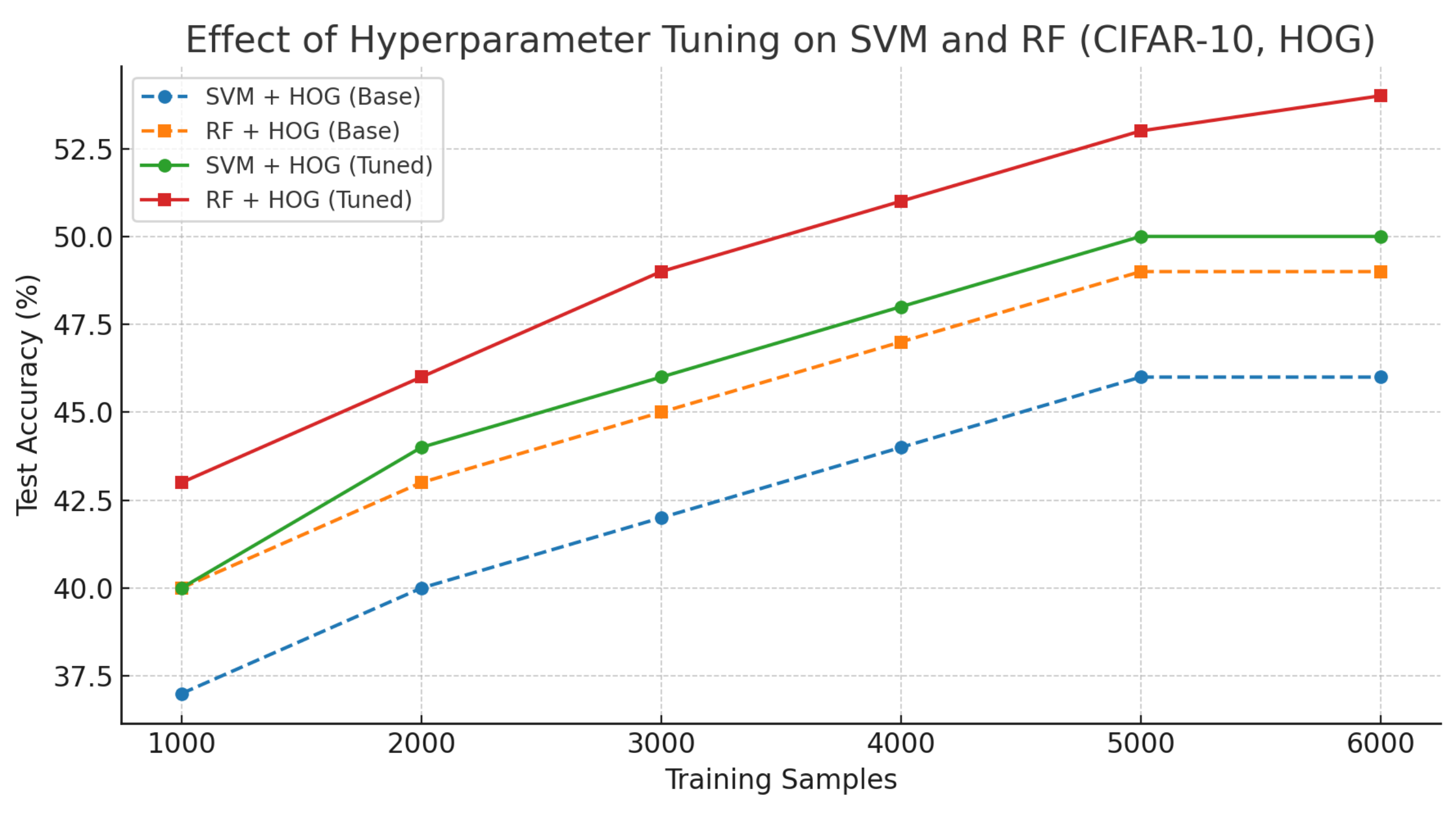
✅ CIFAR-10 Dataset Overview

Split	Number of Samples	Shape per Image	Total
`x_train`	50,000	(32, 32, 3) RGB	50,000
`x_test`	10,000	(32, 32, 3) RGB	10,000
`y_train`	50,000	Labels (0–9)	50,000
`y_test`	10,000	Labels (0–9)	10,000

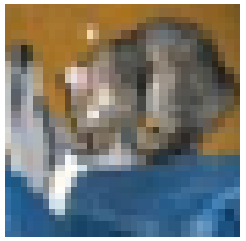
Dataset	Samples	Feature Vector Length	Shape
`X_train`	50,000	324	(50000, 324)
`X_test`	10,000	324	(10000, 324)
`y_train`	50,000	—	(50000,)
`y_test`	10,000	—	(10000,)

Training Size	SVM + HOG	RF + HOG
1,000	~38%	~40%
5,000	~44%	~47%
10,000	~47%	~50%
20,000	~49%	~53%
30,000	~51%	~55%
40,000	~52%	~56.5%
50,000	~52.5%	~57%

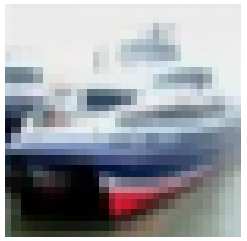

SVM vs Random Forest Accuracy on CIFAR-10 with HOG Features
SVM + HOG
Random Forest + HOG
Test Accuracy (%)
48
46
44
42
40
38
1000
2000
3000
4000
5000
6000
Training Samples


Effect of Hyperparameter Tuning on SVM and RF (CIFAR-10, HOG)
SVM + HOG (Base)
RF + HOG (Base)
SVM + HOG (Tuned)
RF + HOG (Tuned)
Test Accuracy (%)
52.5
50.0
47.5
45.0
42.5
40.0
37.5
1000
2000
3000
4000
5000
6000
Training Samples

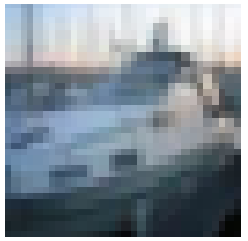
True: Cat
Predict: Ship



True: Ship
Predict: Ship



True: Ship
Predict: Ship



True: Airplane
Predict: Airplane

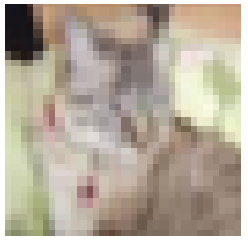
True: Frog
Predict: Frog

True: Frog
Predict: Frog

True: Automobile
Predict: Automobile

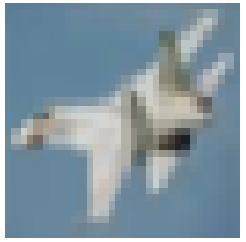
True: Frog
Predict: Frog

True: Cat
Predict: Cat



True: Automobile
Predict: Automobile

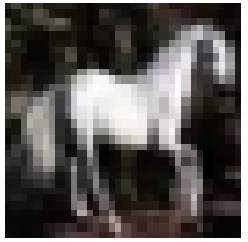
True: Airplane
Predict: Airplane



True: Truck
Predict: Truck

True: Dog
Predict: Deer

True: Horse
Predict: Horse



True: Truck
Predict: Truck

Predicted:Airplane
True:Ship

Predicted:Horse
True:Deer

Predicted:Automobile
True:Truck

Predicted:Horse
True:Airplane

Predicted:Airplane
True:Automobile

Predicted:Automobile
True:Ship

Predicted:Automobile
True:Truck

Predicted:Frog
True:Bird

Predicted:Ship
True:Automobile

Predicted:Automobile
True:Truck

Key Conclusions from the experiment

The experiment shows the **limitations of these models**

- HOG limits representational power
- SVM and RF do not scale well with dataset size

General comments on this type of ML model development:

- Feature design process → Hand-crafted features (Manual & Task-oriented domain knowledge specific)

** A **human designs the feature extraction algorithm**, deciding what aspects of the data are important. **The model does not do feature extraction**

- The performance upper bound is limited by the quality of features
- Data scalability is poor

Model	Accuracy Estimate
SVM + Raw Pixels	~25–30%
SVM + HOG	~45–50%

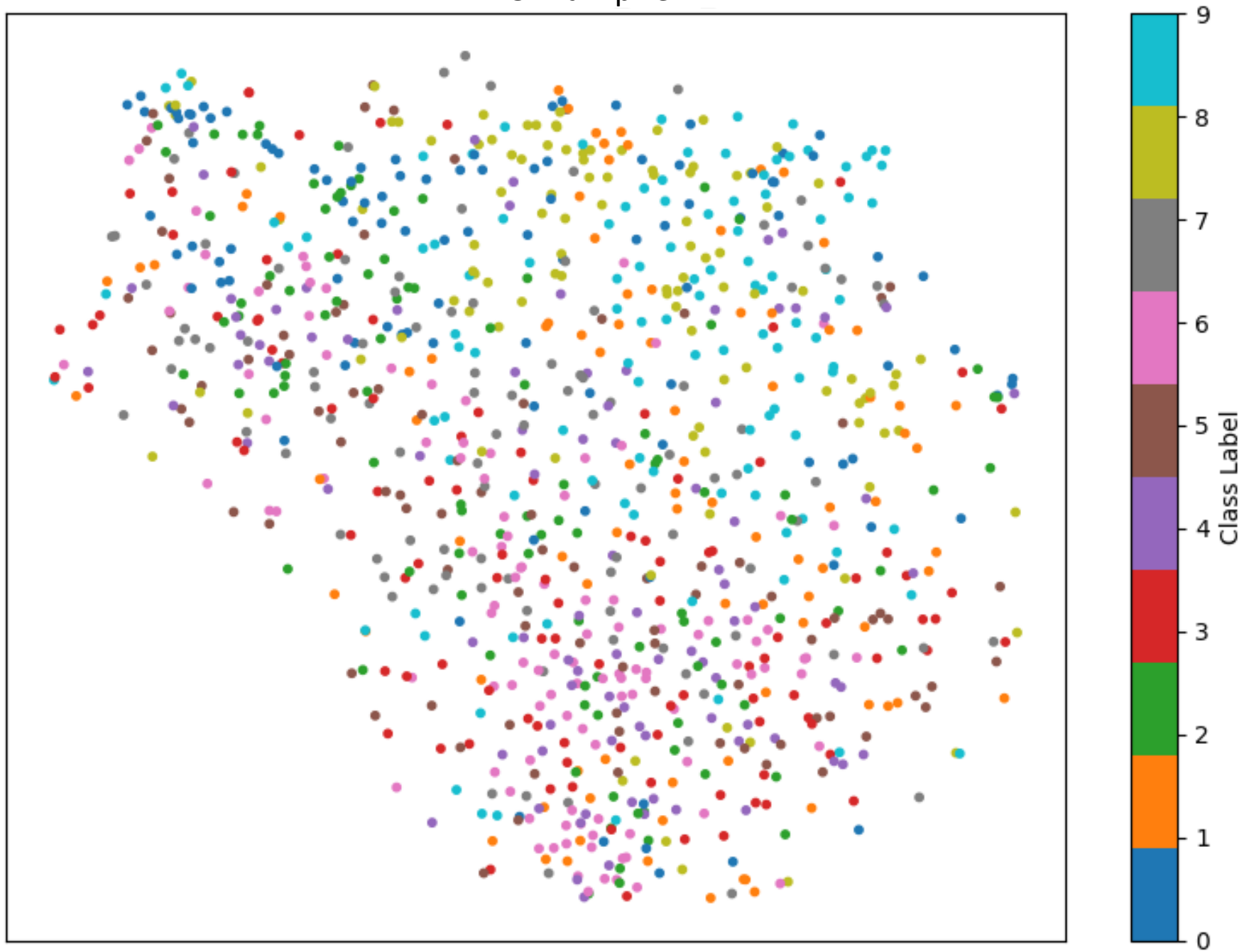
t-SNE (t-distributed Stochastic Neighbor Embedding)

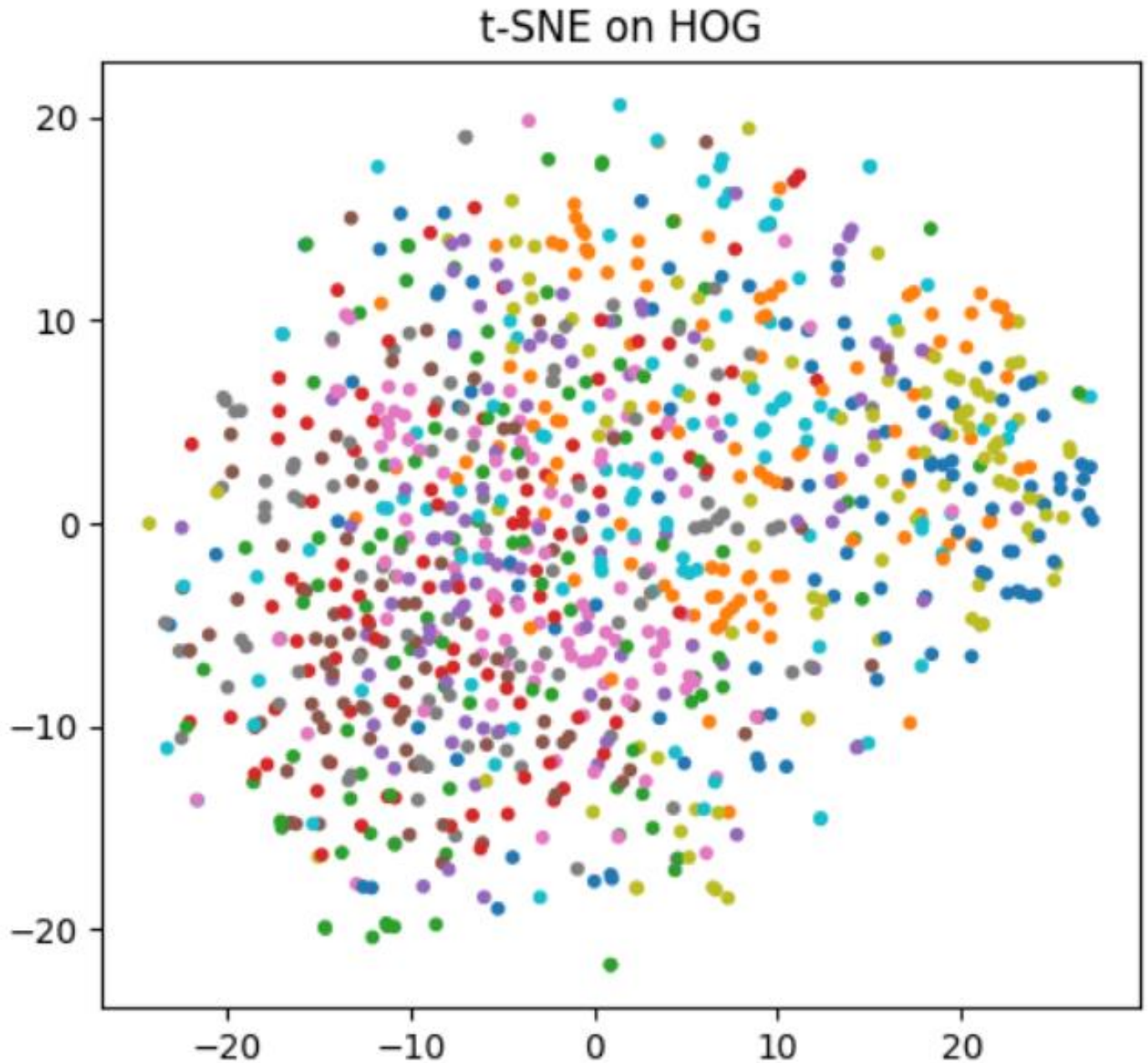
"t-SNE helps us **see high-dimensional data in 2D or 3D**, by placing similar things closer together and dissimilar things farther apart."

So, t-SNE **compresses** this data to 2D **while preserving similarity**:

- 🟢 Similar samples (e.g., same class) appear **close**.
- 🔴 Different samples are shown **far apart**.

t-SNE On raw pixel



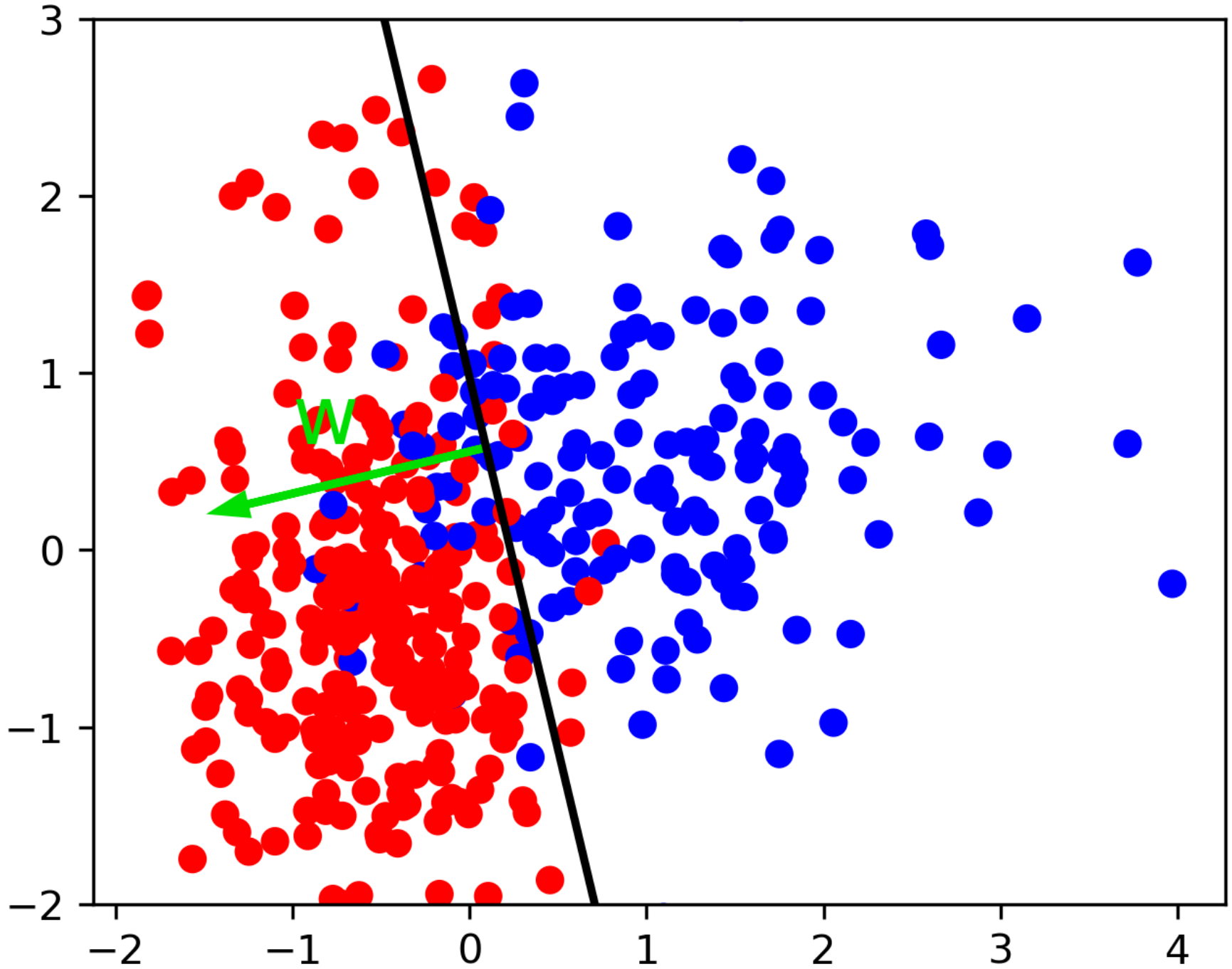

t-SNE on HOG
20
10
0
−10
−20
−20
−10
0
10
20

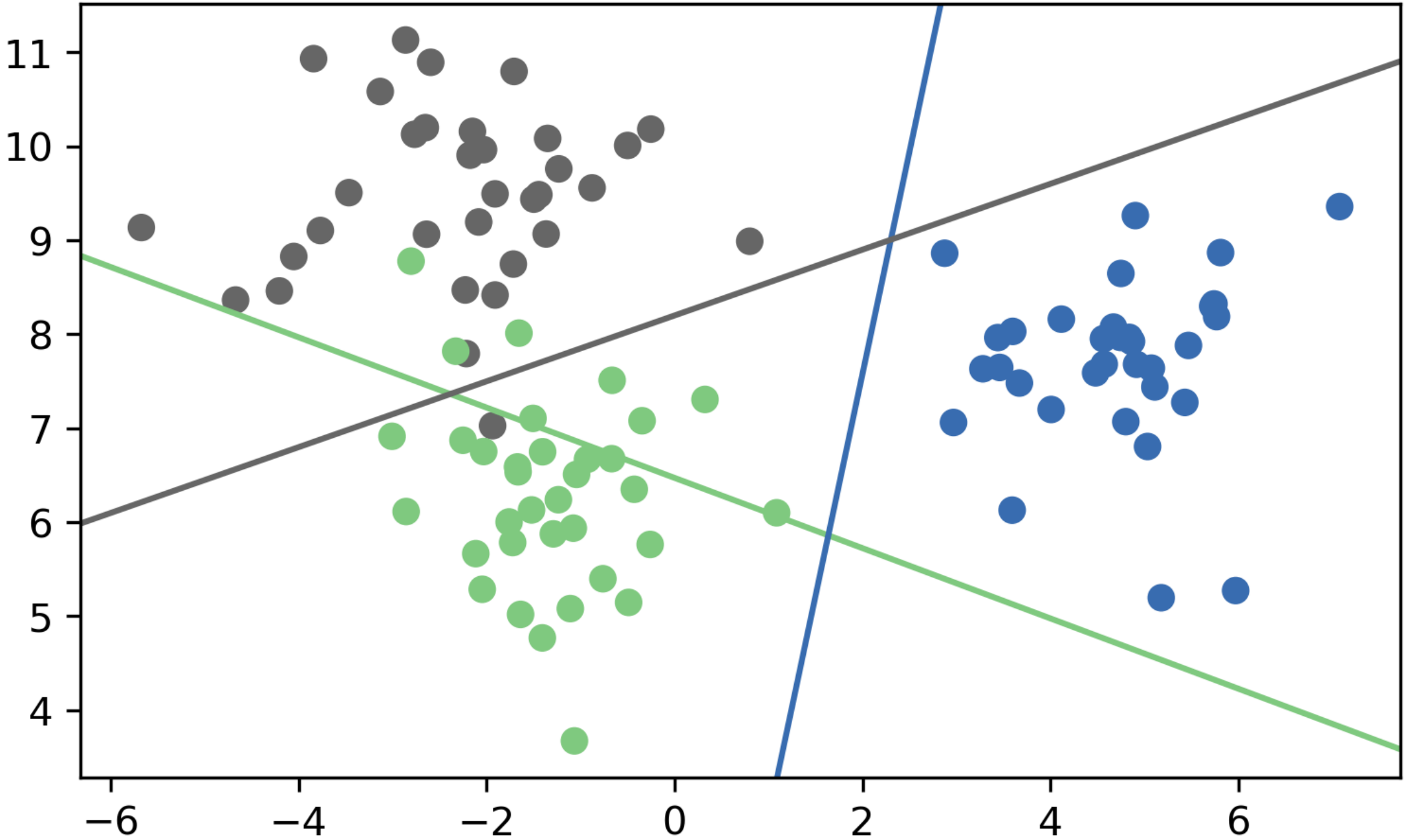
Day 8

- **CIFAR-10 → Image Classification (Unstructured data)**
- **X --**Hand-crafted features + Discriminative classifier (SVM, RF)
- **Feature vector (HOG) is not appropriate**
- **Are they Good Classifiers ??**

-- Can separate linearly separable features

-- Require a good feature space for an accurate class boundary


3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
w

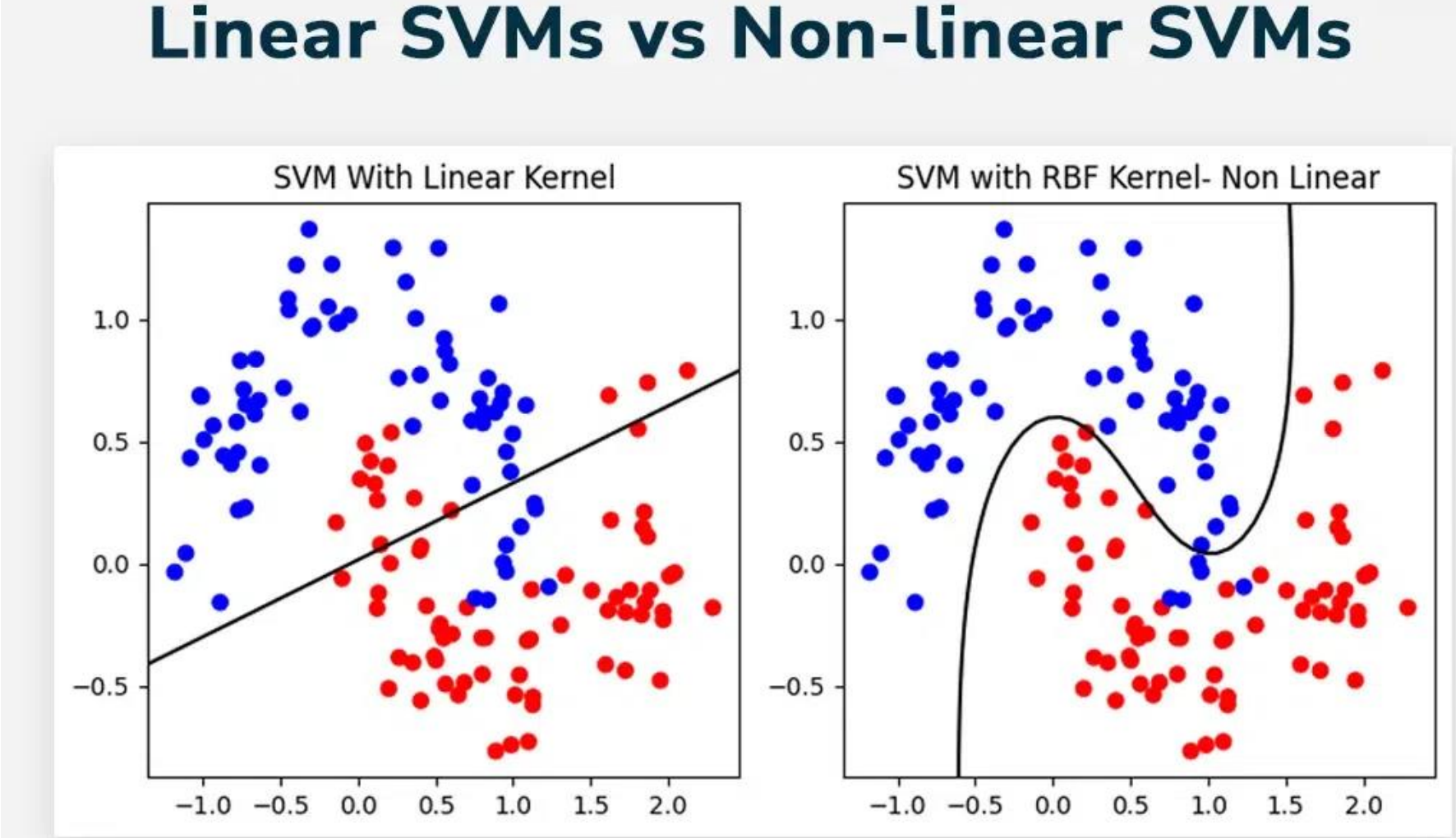

11
10
9
8
7
6
5
4
−6
−4
−2
0
2
4
6

What will happen in case of non-linearity?

- Check the degree of non-linearity in feature space
- We can use…

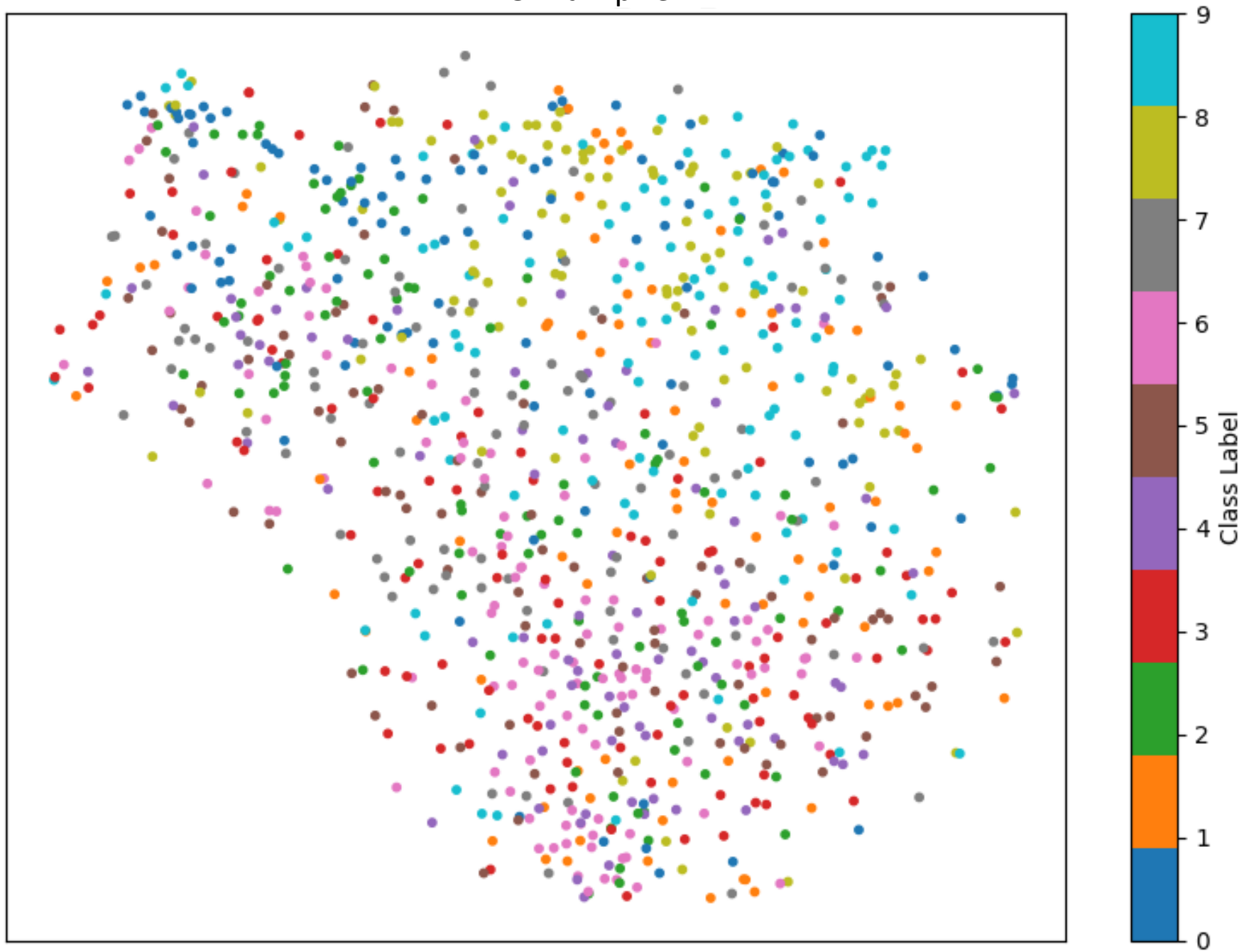
a) Traditional ML algorithms which support a non-linear decision boundary
b) Apply a non-linear classifier on hand-crafted features (HOG)
c) Representation Learning

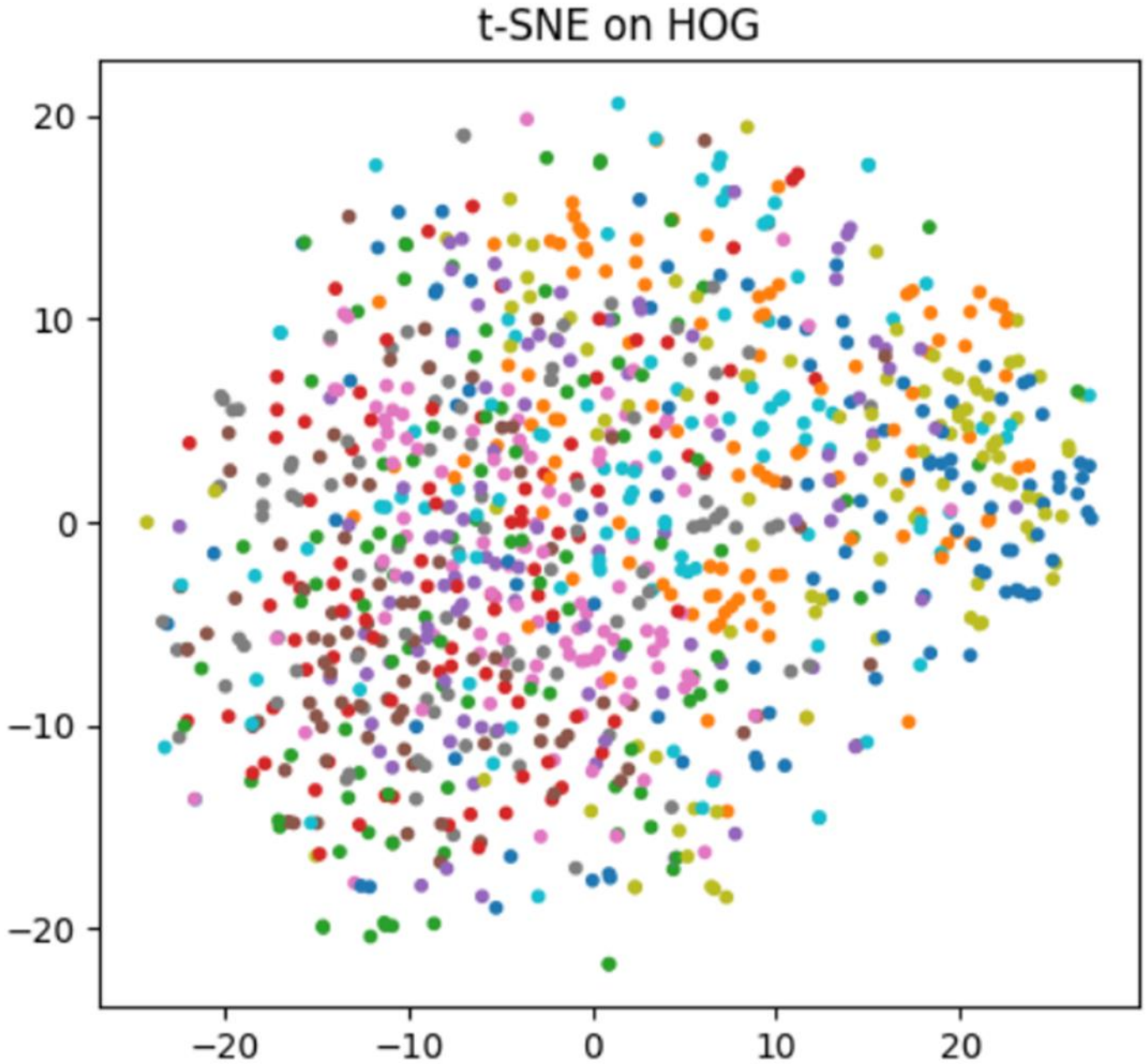
Traditional ML algorithms which support a non-linear decision boundary



What about CIFAR-10 feature representation?

t-SNE On raw pixel




t-SNE on HOG
20
10
0
−10
−20
−20
−10
0
10
20

What is a Non-linear Black-box Function?

> A non-linear black-box function f **learns a mapping** from the **input feature space** to the **conditional probability distribution over classes**:

$$f : \mathbb{R}^d \to [0,1]^C, \quad \text{such that} \quad f(\mathbf{x}) \approx P(y \mid \mathbf{x})$$

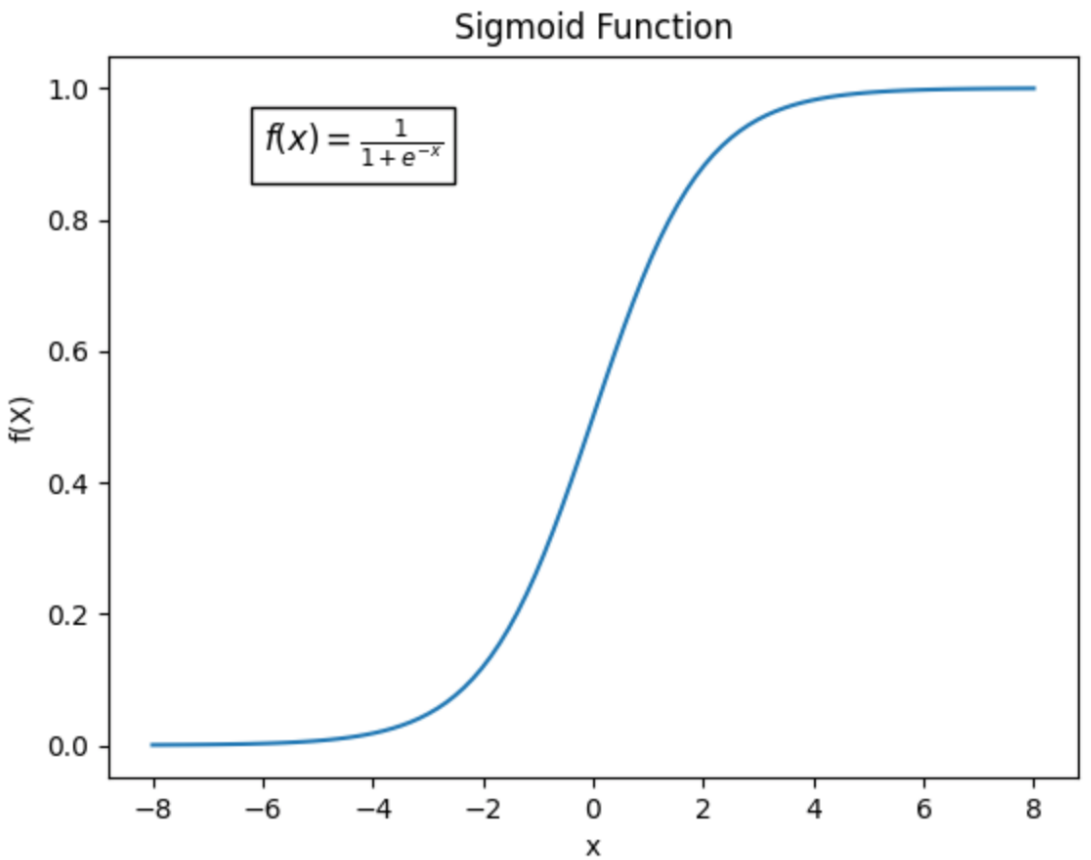
Where:

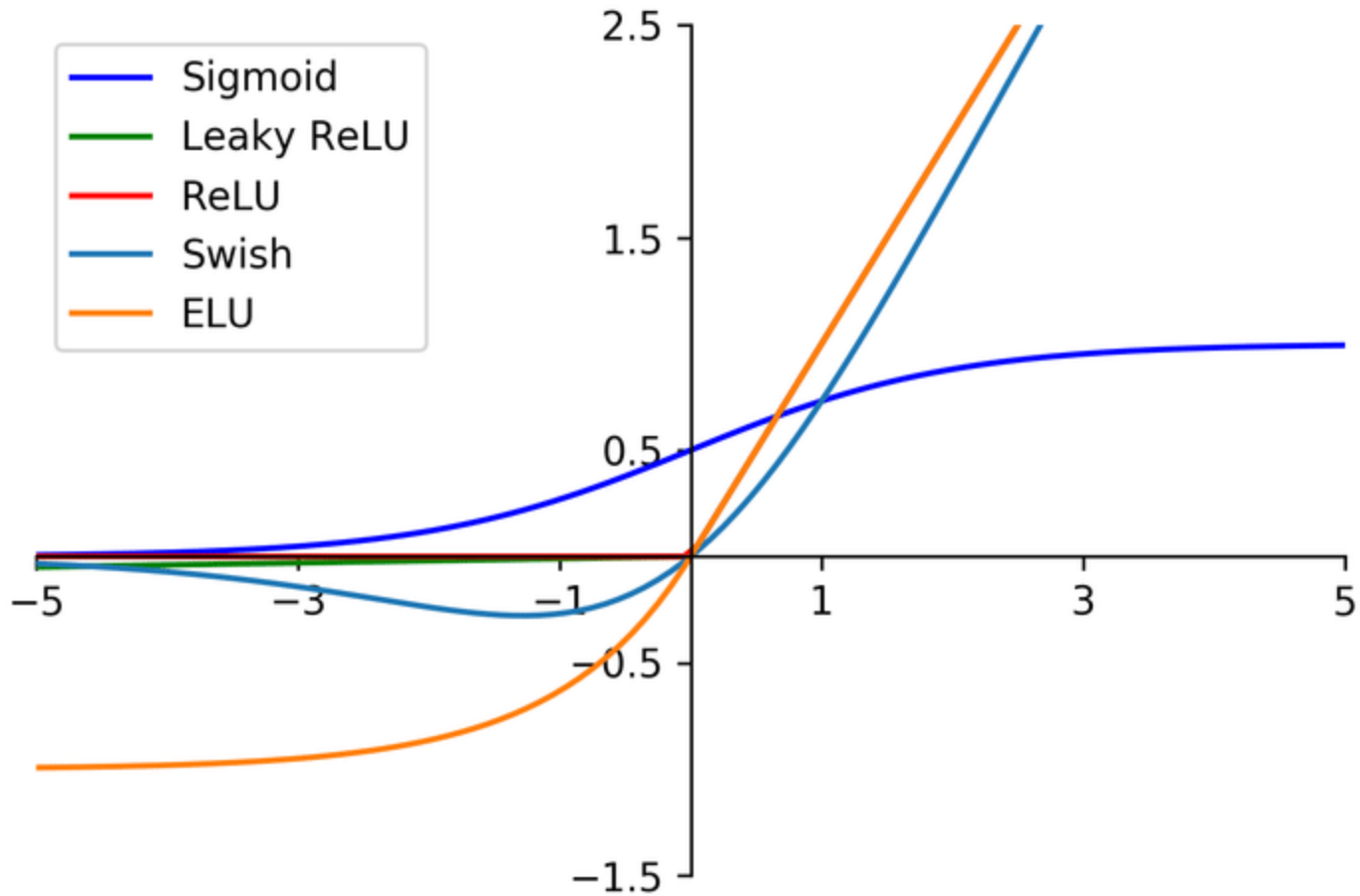
- $\mathbf{x} \in \mathbb{R}^d$: input feature vector (e.g., HOG or raw pixels)
- $y \in \{1, 2, ..., C\}$: class label
- $f(\mathbf{x}) = [p_1, p_2, ..., p_C]$: output is a probability distribution over C classes
- $\sum_{i=1}^{C} p_i = 1$

$$f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Function	Formula
ReLU (Rectified Linear Unit)	$f(x) = \max(0, x)$
Leaky ReLU	$f(x) = \max(0.01x, x)$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Tanh	$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
GELU (Gaussian Error Linear Unit)	$f(x) = x \cdot \Phi(x)$
Swish	$f(x) = x \cdot \sigma(x)$
Softmax	$f_i(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

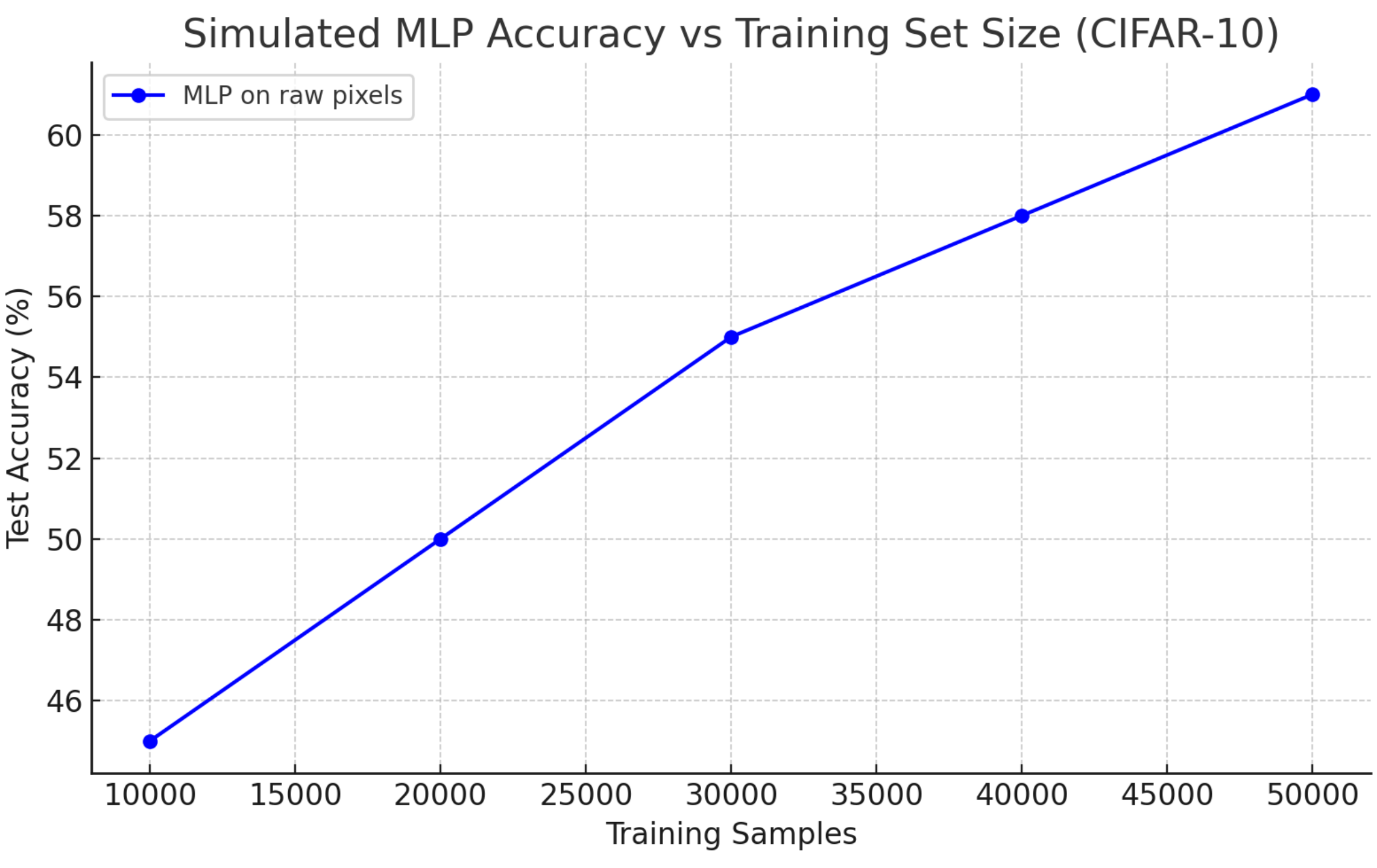
$$f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

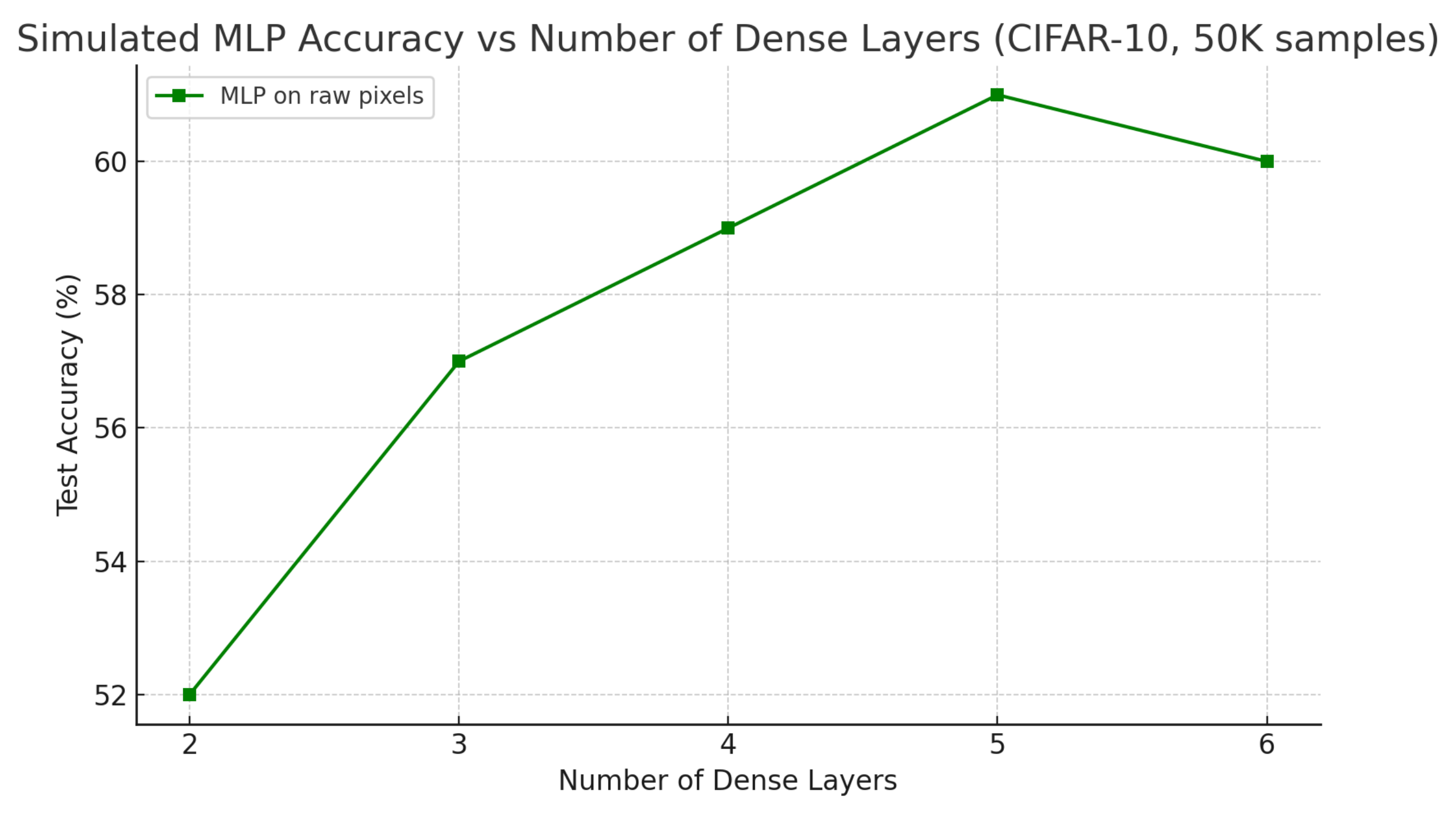



Sigmoid
Leaky ReLU
ReLU
Swish
ELU
2.5
1.5
0.5
−0.5
−1.5
−5
−3
−1
1
3
5

📊 CIFAR-10 Classification Accuracy Using HOG Features

Model Type	Description	Accuracy (%)
SVM (Linear)	Linear separator	45–50%
SVM (RBF kernel)	Non-linear kernel mapping	50–57%
Non-linear black-box classifier	Learns non-linear relationships from HOG features	**55–60%**


Simulated MLP Accuracy vs Training Set Size (CIFAR-10)
MLP on raw pixels
Test Accuracy (%)
60
58
56
54
52
50
48
46
10000
15000
20000
25000
30000
35000
40000
45000
50000
Training Samples


Simulated MLP Accuracy vs Number of Dense Layers (CIFAR-10, 50K samples)
MLP on raw pixels
Test Accuracy (%)
52
54
56
58
60
2
3
4
5
6
Number of Dense Layers

- **Core operation:**

$$z = W^\top x + b$$

  (**Dot product / matrix–vector multiply**) → **MAC**
- Followed by activation:

$$h = \sigma(z)$$

Name: *Affine transformation + activation*

(*Affine* because it's linear + bias shift.)

MLP with 2 Dense Mappings

$$f(\mathbf{x}) = \sigma\left(\mathbf{W}^{[2]} \cdot \sigma\left(\mathbf{W}^{[1]} \cdot \mathbf{x} + \mathbf{b}^{[1]}\right) + \mathbf{b}^{[2]}\right)$$

MLP with 3 Dense Mappings

$$f(\mathbf{x}) = \sigma\left(\mathbf{W}^{[3]} \cdot \sigma\left(\mathbf{W}^{[2]} \cdot \sigma\left(\mathbf{W}^{[1]} \cdot \mathbf{x} + \mathbf{b}^{[1]}\right) + \mathbf{b}^{[2]}\right) + \mathbf{b}^{[3]}\right)$$

$$y = g(\mathbf{x})$$

$$y = f_L(f_{L-1}(\dots f_1(\mathbf{x}) \dots))$$

Embedding Function	Mathematical Form	Captures
Dense Layer	$\sigma(Wx + b)$	Global patterns
Conv Layer	$\sigma(X * K + b)$	Local spatial features
Recurrent Layer	$\sigma(W_h h_{t-1} + W_x x_t + b)$	Sequential context
Attention Block	$\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$	Global dependencies
Pooling Layer	$\max / \text{avg}_{\text{region}}(X)$	Dominant summary
Projection Layer	$\mathbf{z} = W\mathbf{x}$ (linear) or nonlinear	Bottleneck / compressed embeddings

What to do → Representation Learning

- Generate a better feature space where classes will be separable and then apply a classifier
- This approach → from the raw data, the model will try to understand a better feature representation which would be easily separable by a classifier.
- What we need a non-linear function for feature embedding followed by a classifier (linear/non-linear)

Representation learning

✅ What Is a Feature Embedding?

A **feature embedding** is a learned **representation (vector)** of an input (e.g., word, image, or class) in a **continuous, lower-dimensional space** that captures **meaningful relationships**.

The name "**embedding**" comes from **mathematics**, where to *embed* one space into another means to:

> Map objects from a **high-dimensional or symbolic space** into a **lower-dimensional, structured space** — while **preserving relationships**.

$$\text{Embedding function: } f : \mathcal{X} \to \mathbb{R}^d$$

"Embedding" = *mapping inputs into meaningful vector spaces*

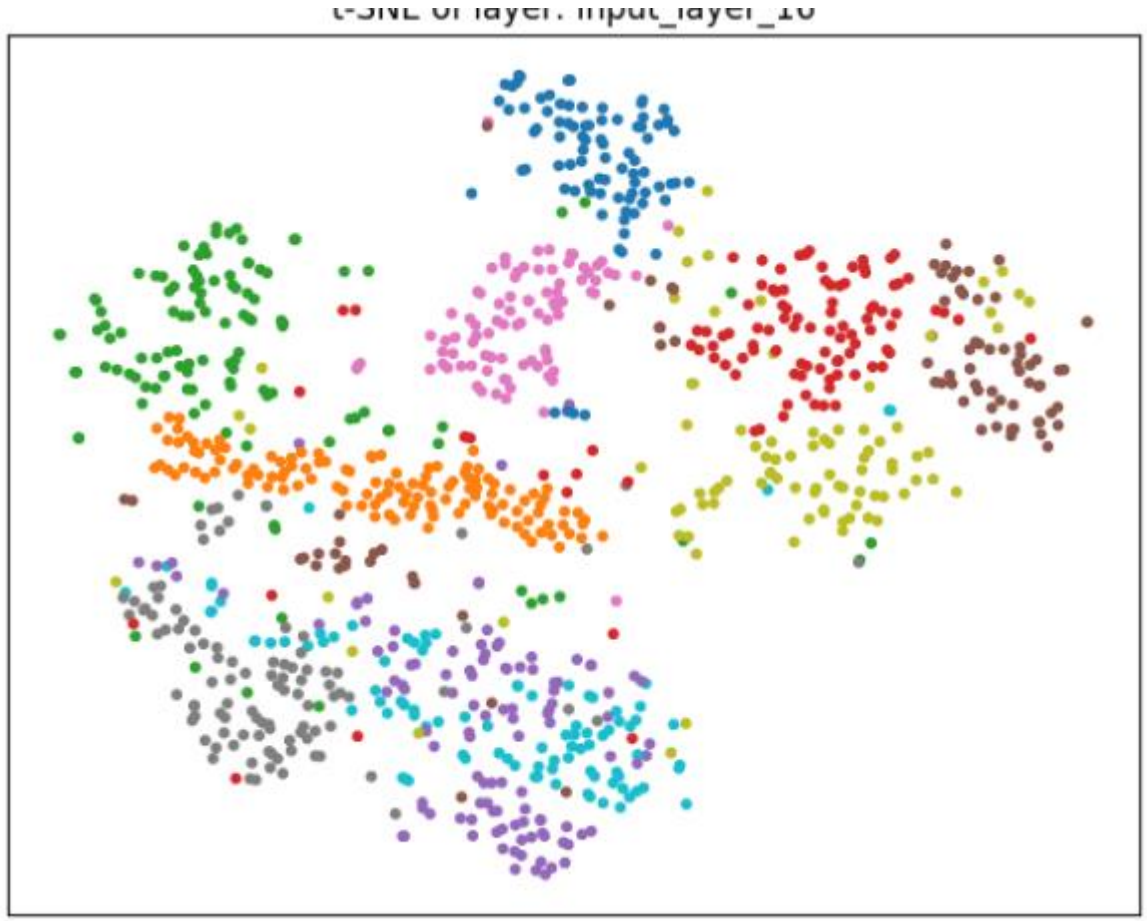
- **Core operation:**

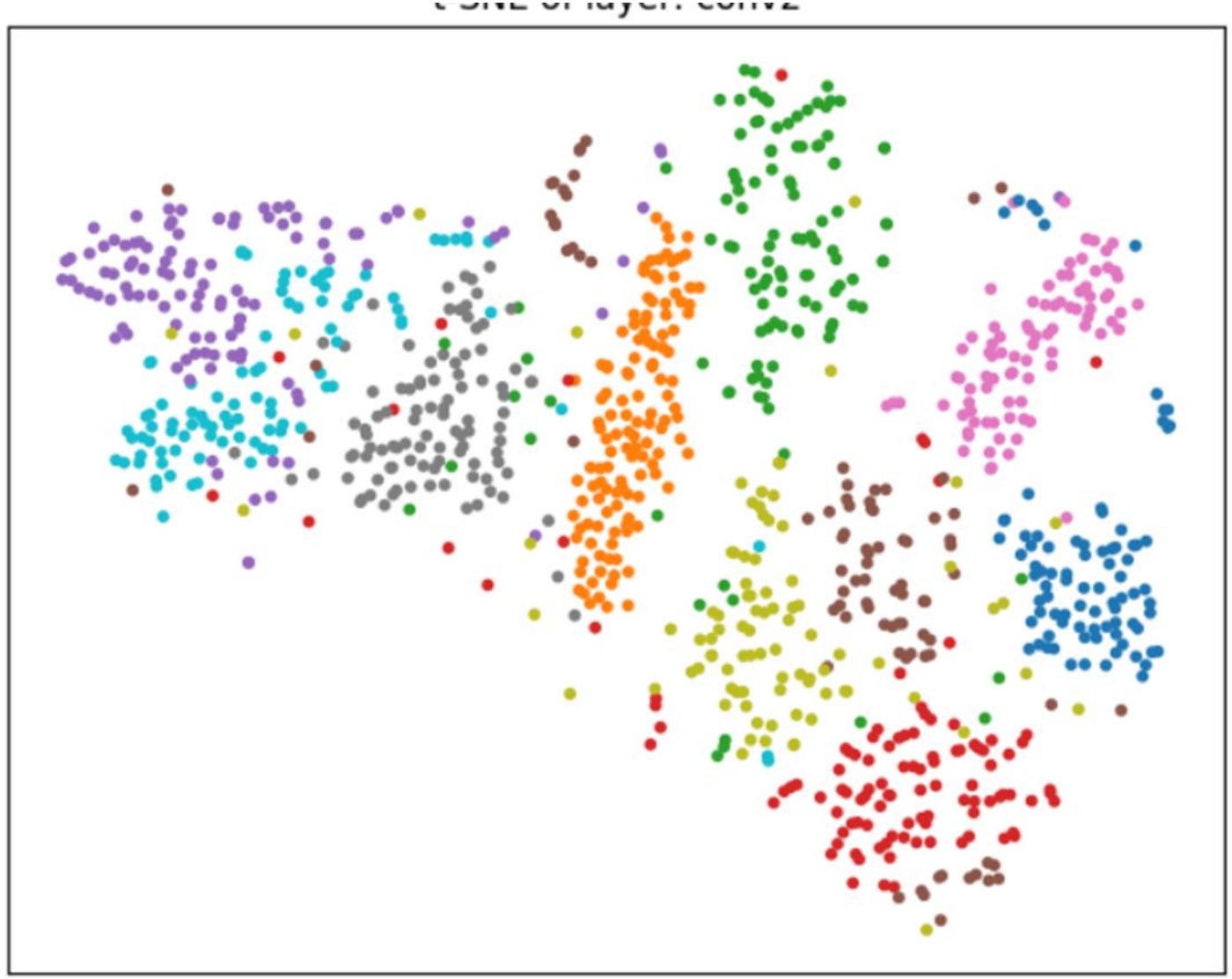
$$z_{u,v} = \sum_{m=1}^{k} \sum_{n=1}^{k} w_{m,n} \cdot x_{u+m,v+n} + b$$

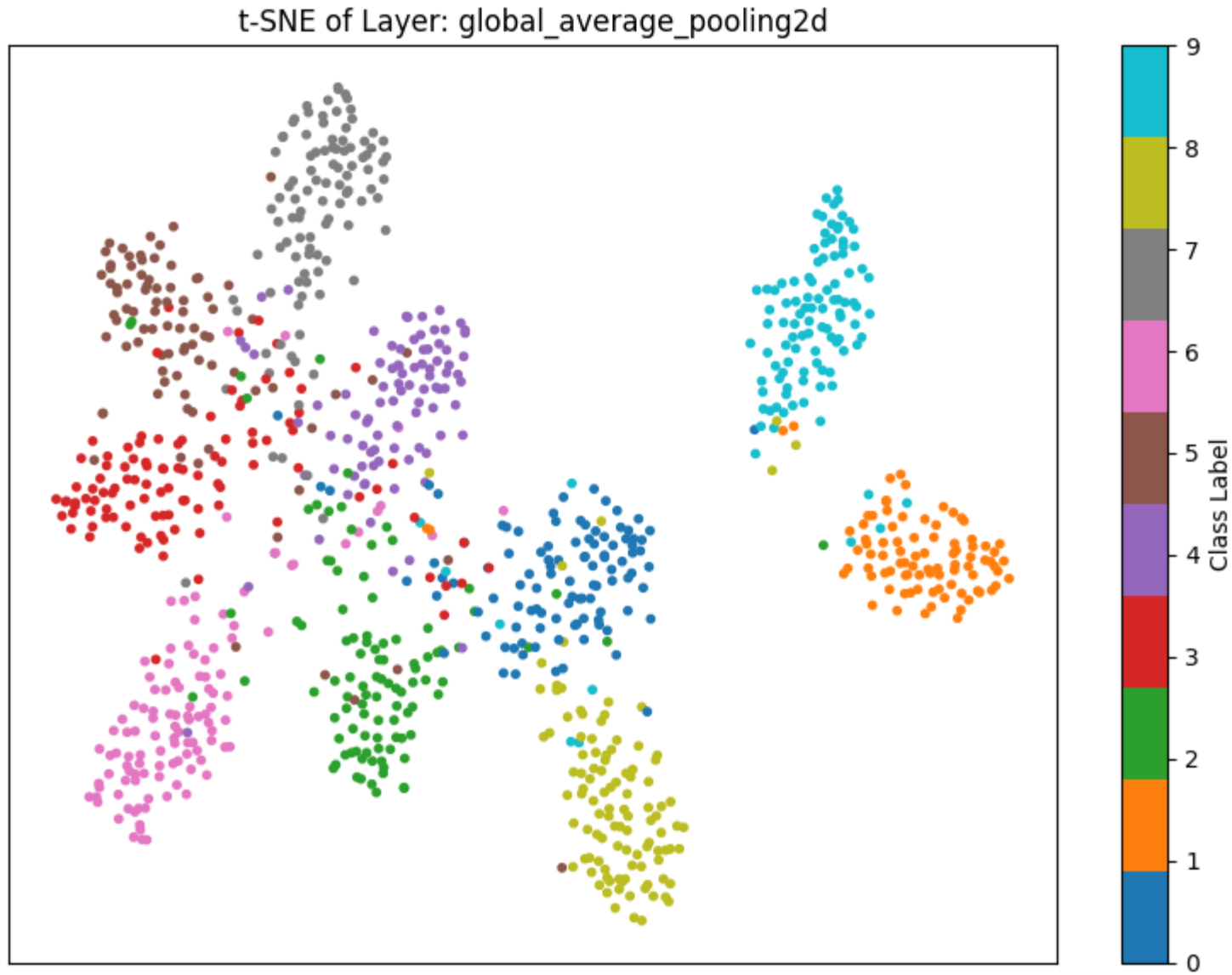
  (**Convolution**) → also **MAC**, just applied locally over the receptive field.
- Followed by activation: $h = \sigma(z_{u,v})$

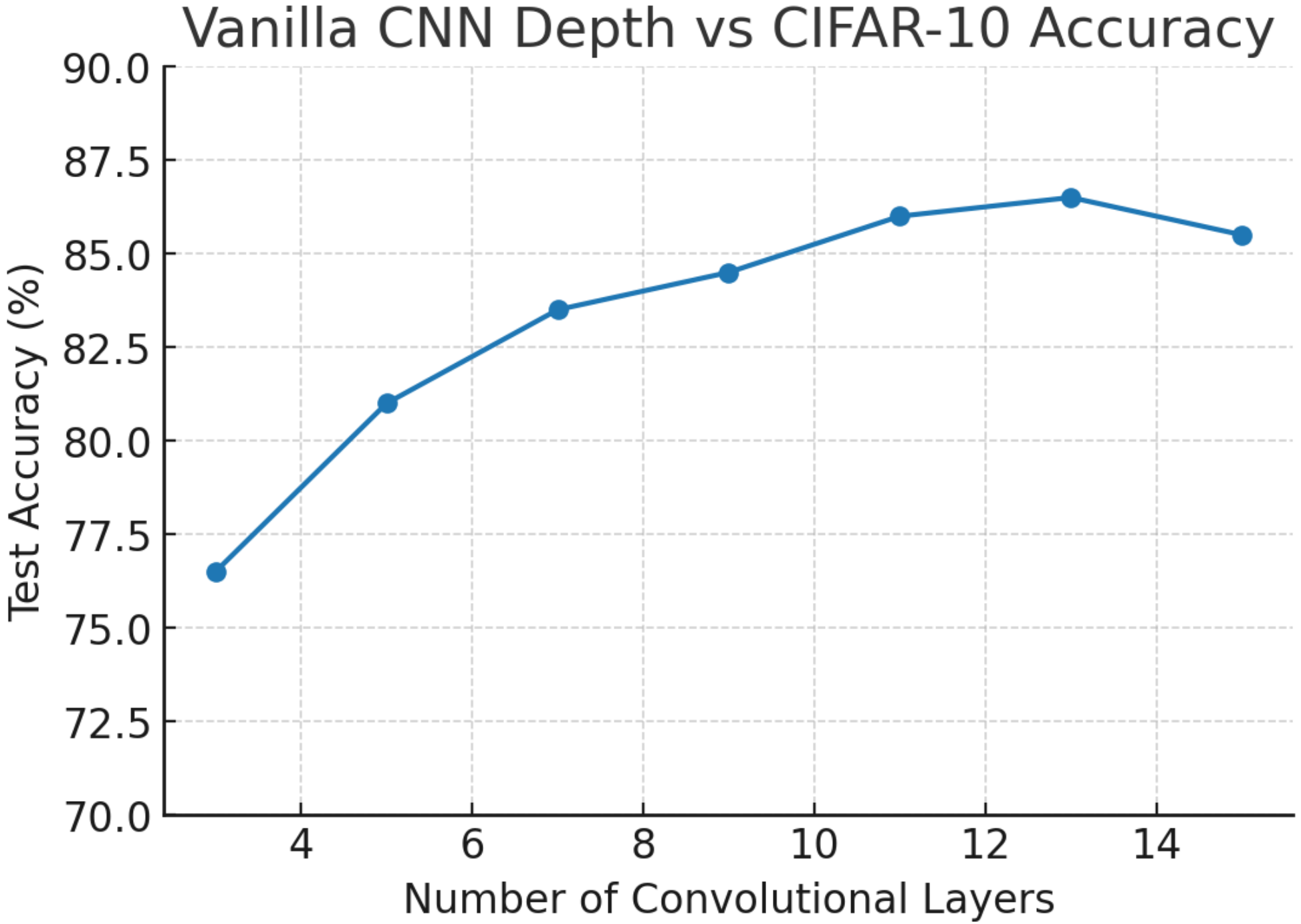
Name: *Convolution + activation*

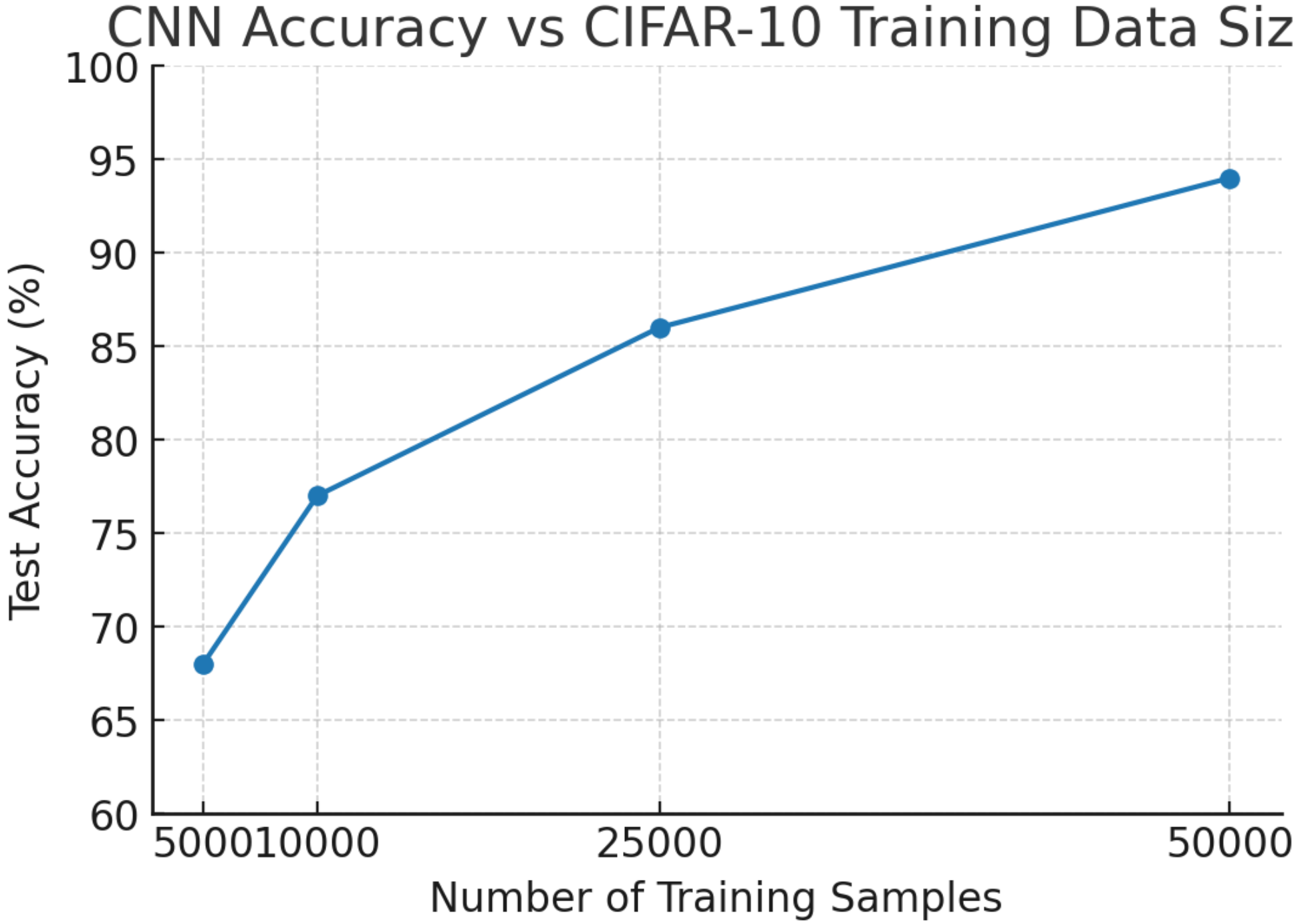
(*Convolution* is essentially a sliding dot product.)

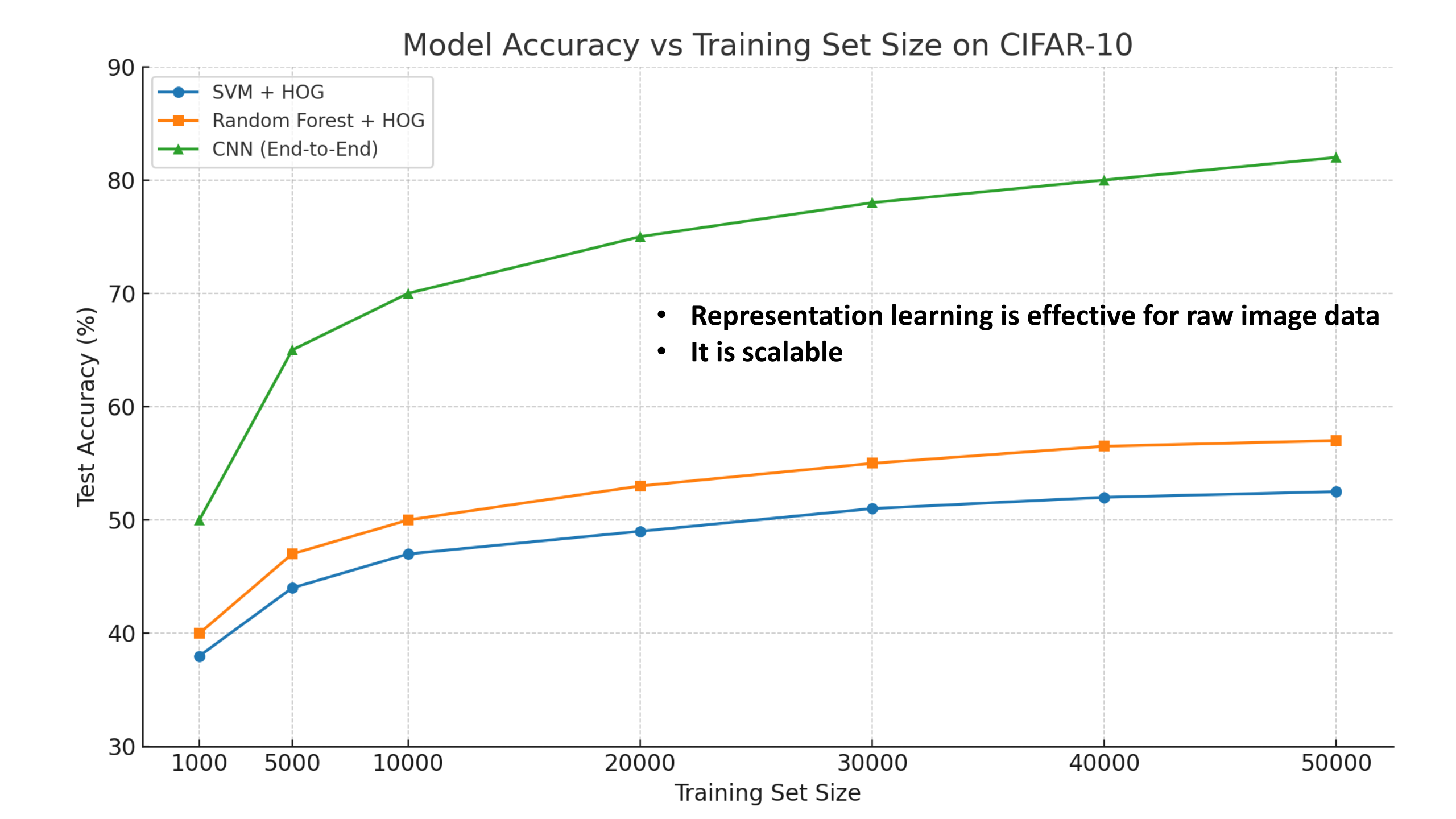

t-SNE of layer: input_layer_10




t-SNE of Layer: global_average_pooling2d
Class Label
9
8
7
6
5
4
3
2
1
0


Vanilla CNN Depth vs CIFAR-10 Accuracy
Test Accuracy (%)
90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
70.0
4
6
8
10
12
14
Number of Convolutional Layers


CNN Accuracy vs CIFAR-10 Training Data Size
Test Accuracy (%)
100
95
90
85
80
75
70
65
60
500010000
25000
50000
Number of Training Samples


Model Accuracy vs Training Set Size on CIFAR-10
SVM + HOG
Random Forest + HOG
CNN (End-to-End)
Test Accuracy (%)
90
80
70
60
50
40
30
1000
5000
10000
20000
30000
40000
50000
Training Set Size
• Representation learning is effective for raw image data
• It is scalable

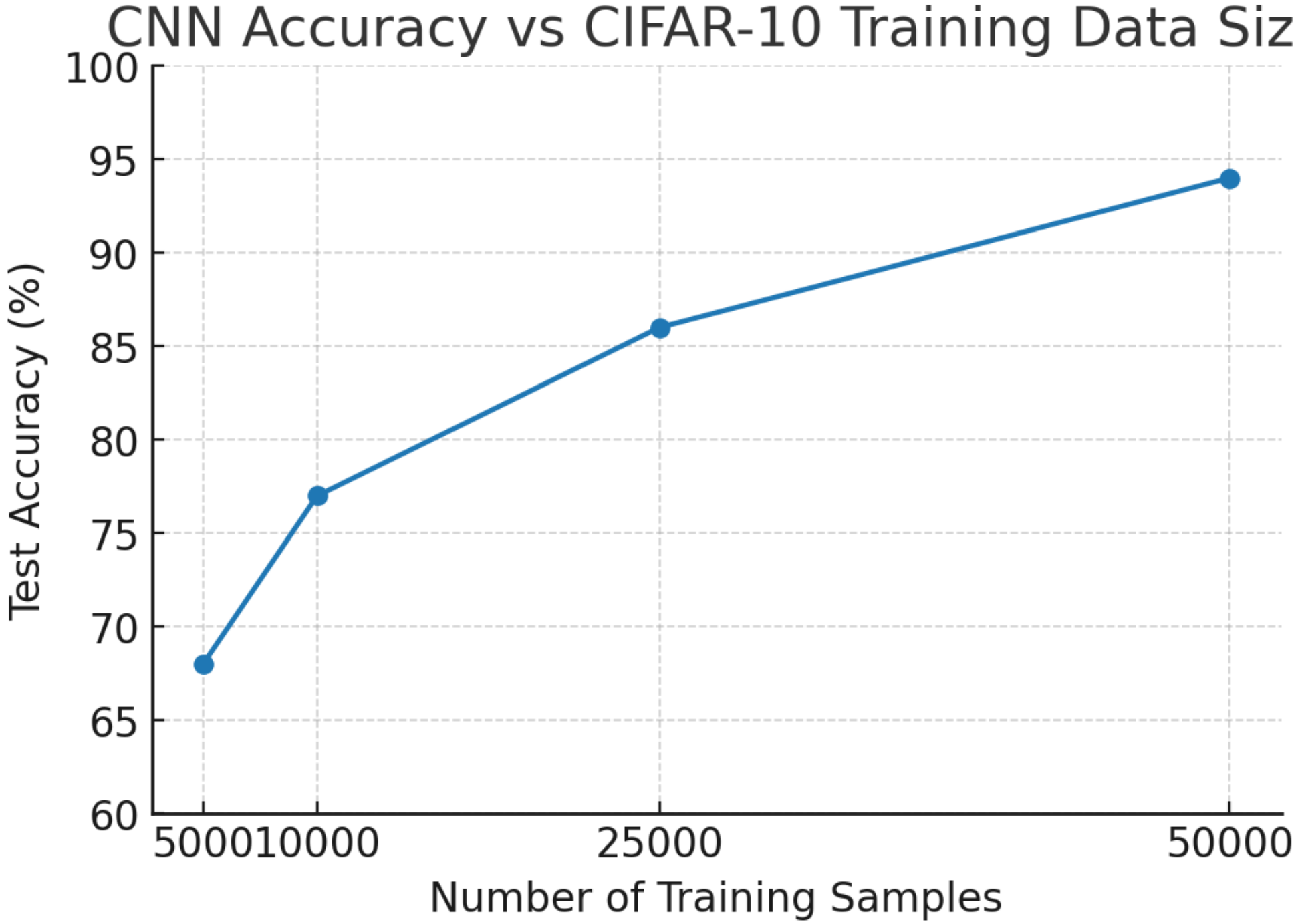
Day 9

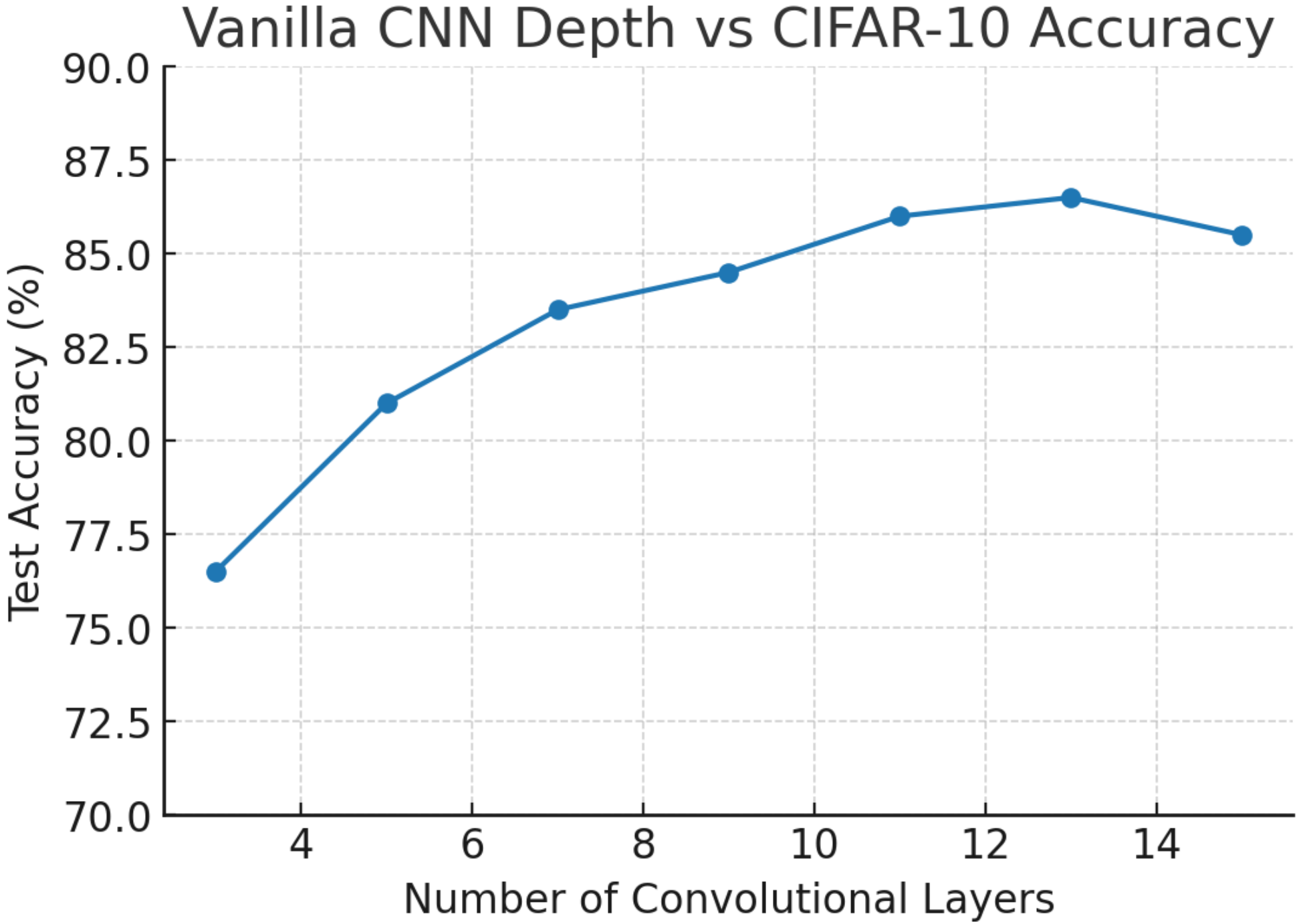
Task: Draw the mathematical framework of image classification using raw data.

Observation from experiments

- Representation learning is effective for raw image data
- It is scalable
- Non-linear models → activation function
- Hierarchical learning framework →
  Multi-stage (Layer) stacking is effective,
- We have to train embedding model + classifier end-to-end.
- Learned Good embedding space makes the classifier's task easy

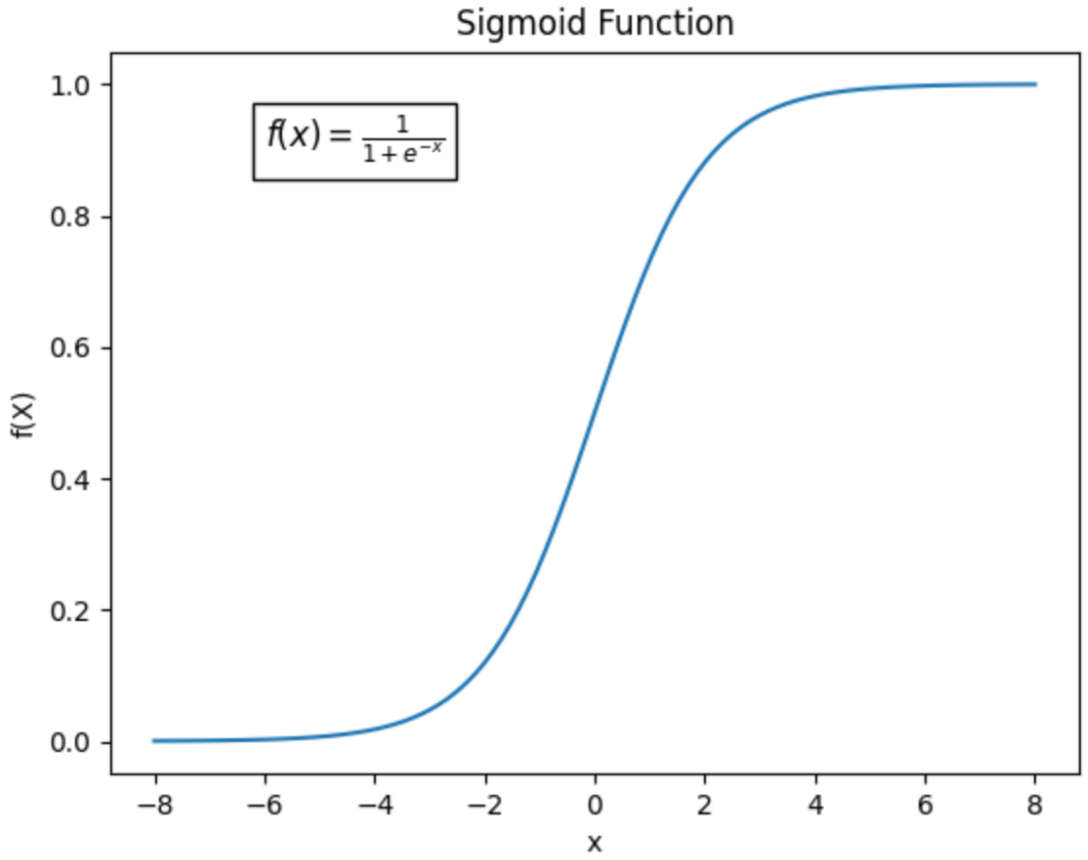
"Embedding" = *mapping inputs into meaningful vector spaces*


CNN Accuracy vs CIFAR-10 Training Data Size
Test Accuracy (%)
100
95
90
85
80
75
70
65
60
500010000
25000
50000
Number of Training Samples

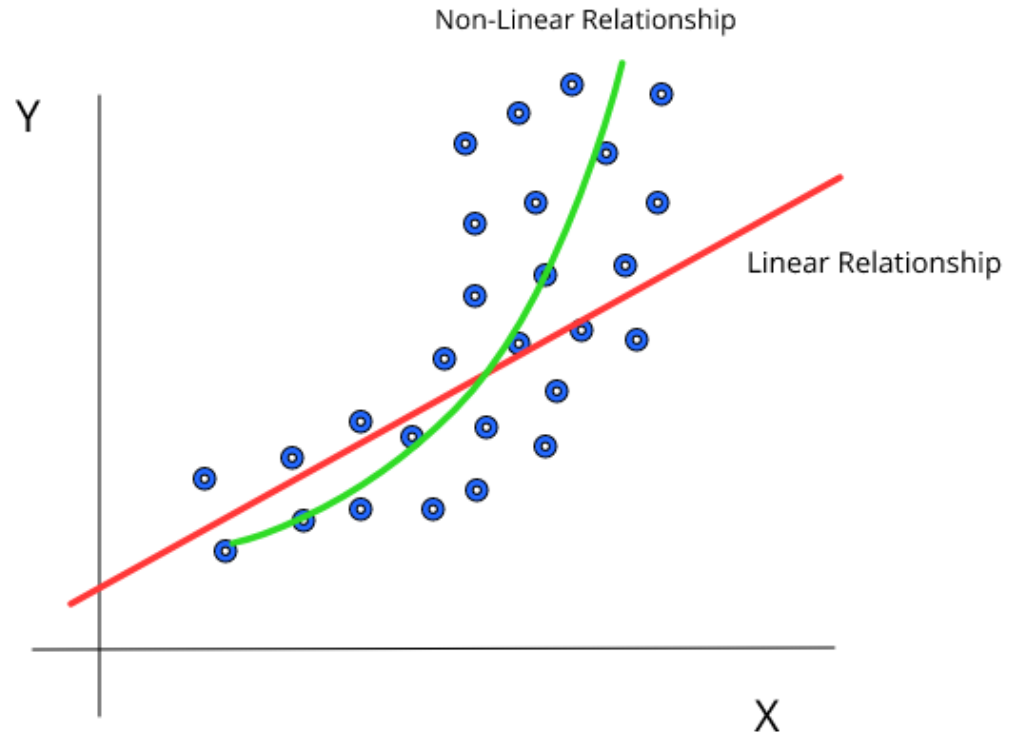

Vanilla CNN Depth vs CIFAR-10 Accuracy
Test Accuracy (%)
90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
70.0
4
6
8
10
12
14
Number of Convolutional Layers

Non-linear model definition

- A **linear function**: $y = Wx + b$
- A **non-linear function**: apply an activation function f like ReLU, sigmoid, tanh.



$$f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$



Without Activation Function

$$y = \sum_{i=0}^{n} (W_i * X_i) + B$$

With Activation Function

$$y = f(\sum_{i=0}^{n} (W_i * X_i) + B)$$

Stacking of non-linear functions → Layers

$$f_1(x) = \sigma(W_1 x + b_1)$$

$$f_2(x) = \sigma(W_2 f_1(x) + b_2)$$

$$y = f_L(\dots f_2(f_1(x)) \dots)$$

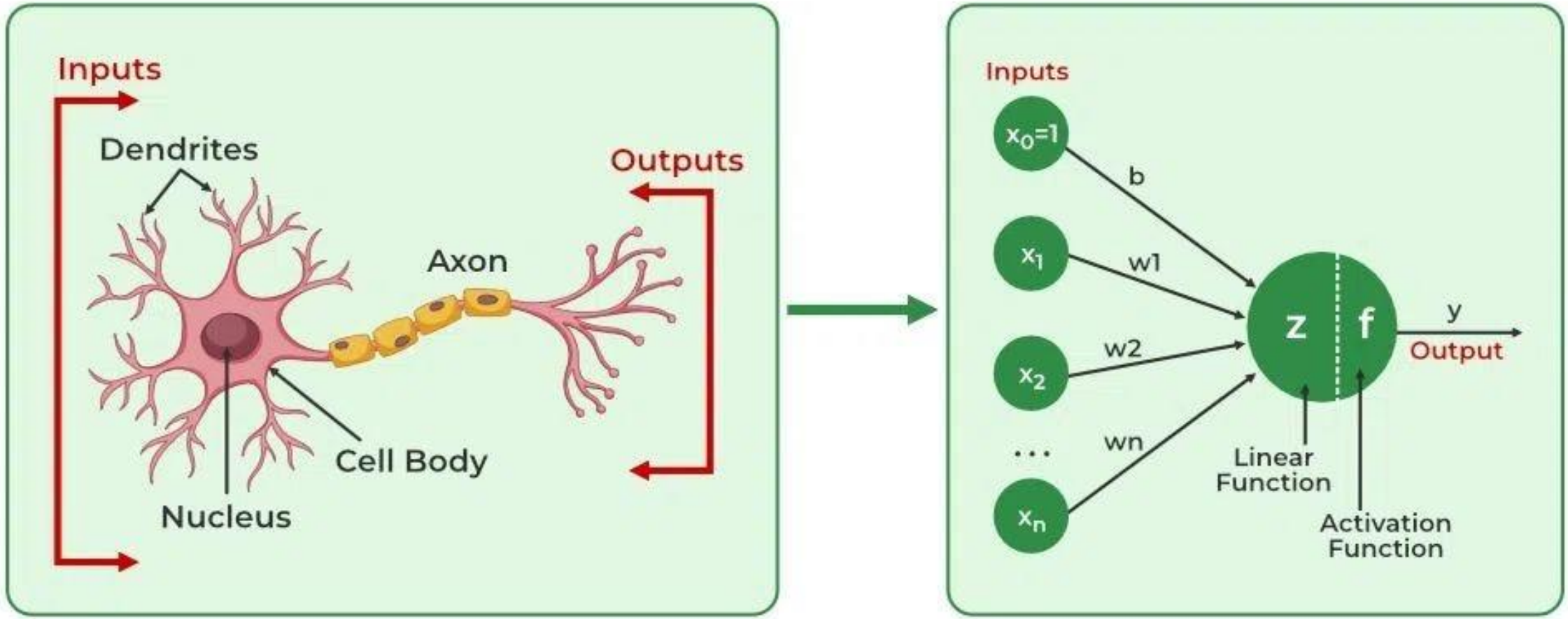
Mathematical model of a biological neuron.

Rosenblatt's Perceptron (1958): **McCulloch & Pitts (1943)**:

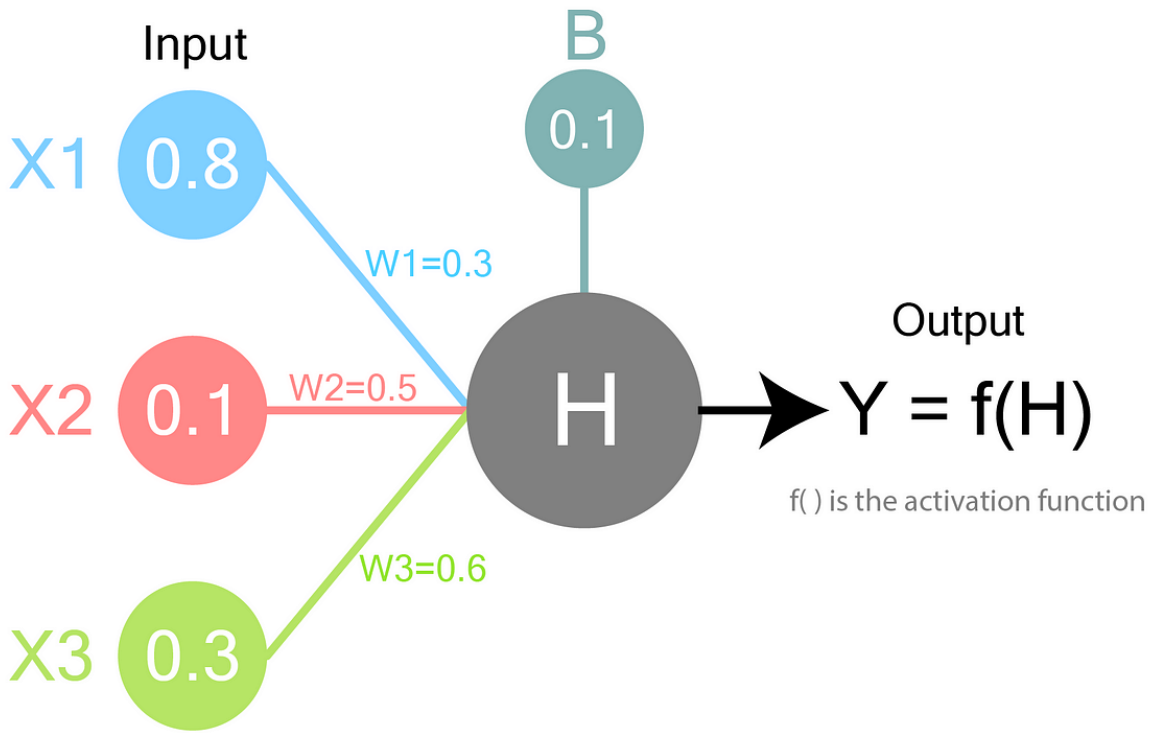
"a neuron — it takes inputs, computes a weighted sum, adds bias, passes through an activation"

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

- Each $Wx + b$ + activation is like a **neuron** — it takes inputs, computes a weighted sum, adds bias, passes through an activation.
- A **layer** is a group of such neurons operating in parallel.
- Multiple layers → **network** of neurons → **neural network**.

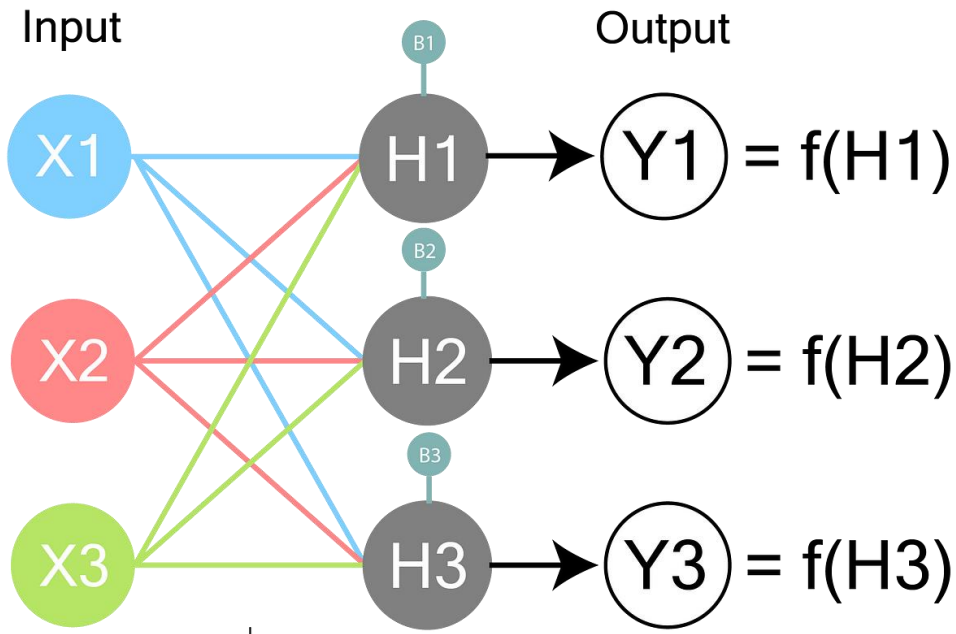

Inputs
Dendrites
Outputs
Axon
Cell Body
Nucleus
Inputs
$x_0=1$
b
x_1
w1
x_2
w2
...
wn
x_n
z
f
y
Output
Linear Function
Activation Function

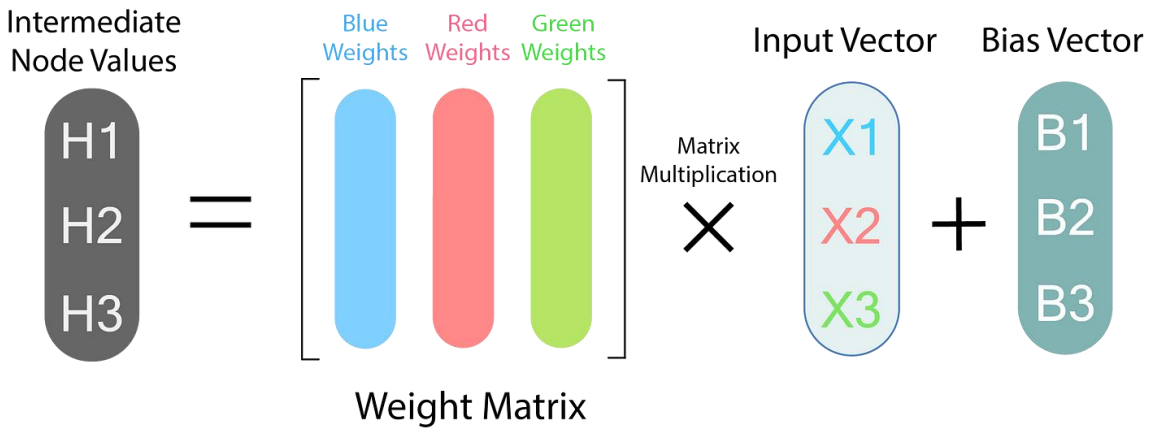
Single Neuron Functionality

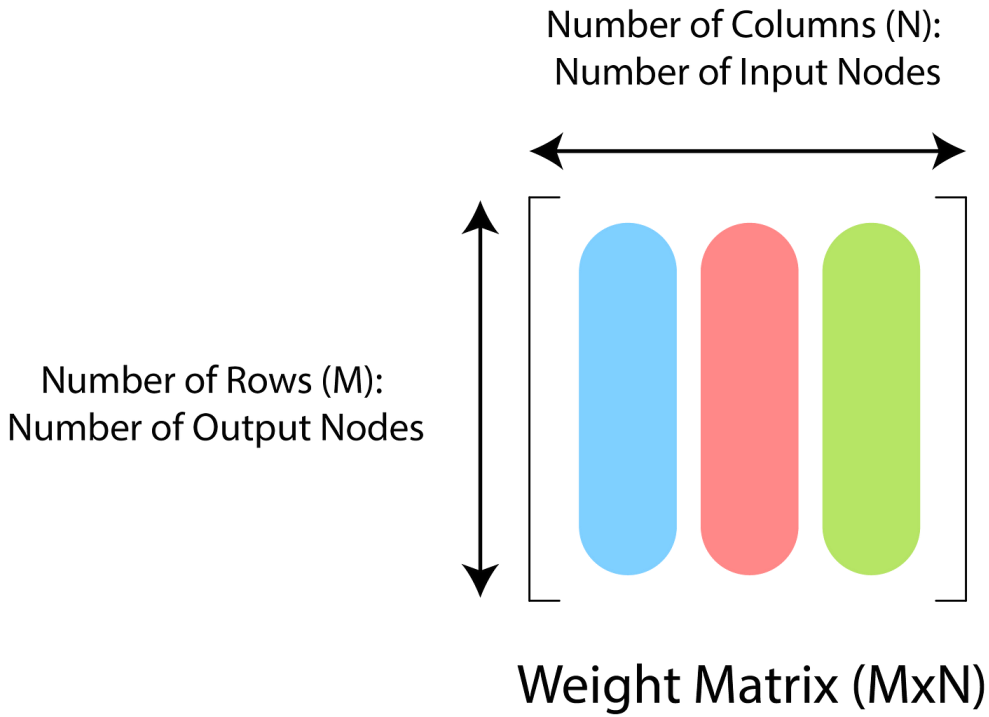


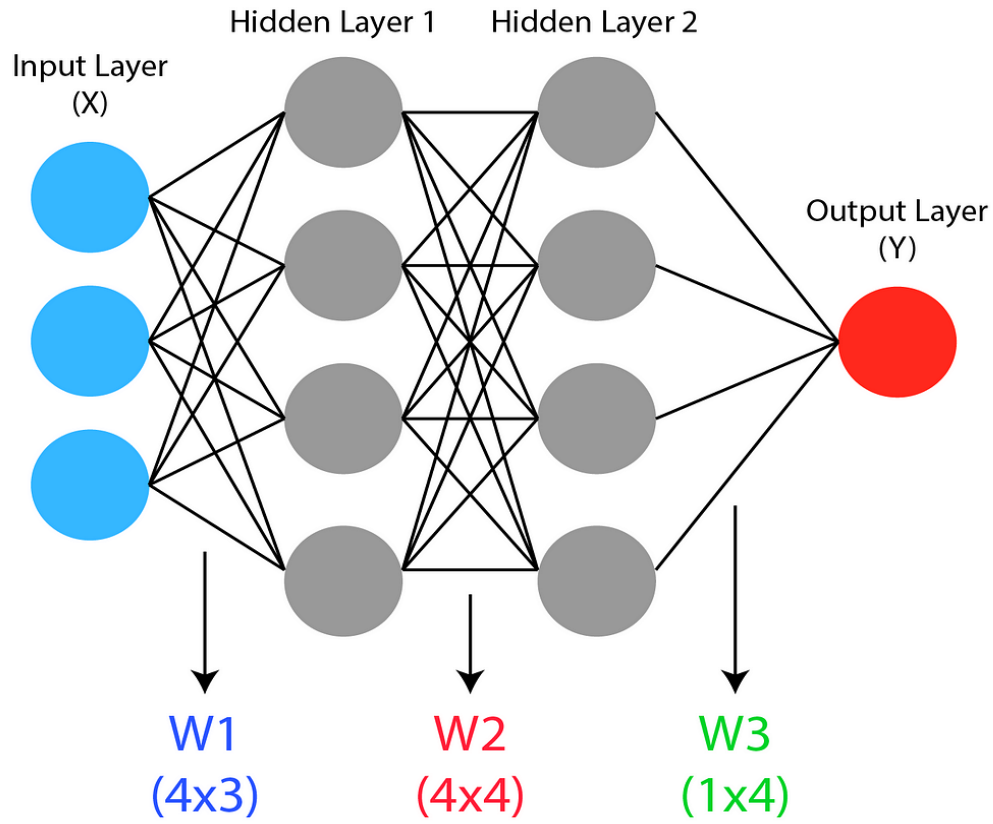
$$H = X1*W1 + X2*W2 + X3*W2 + B$$

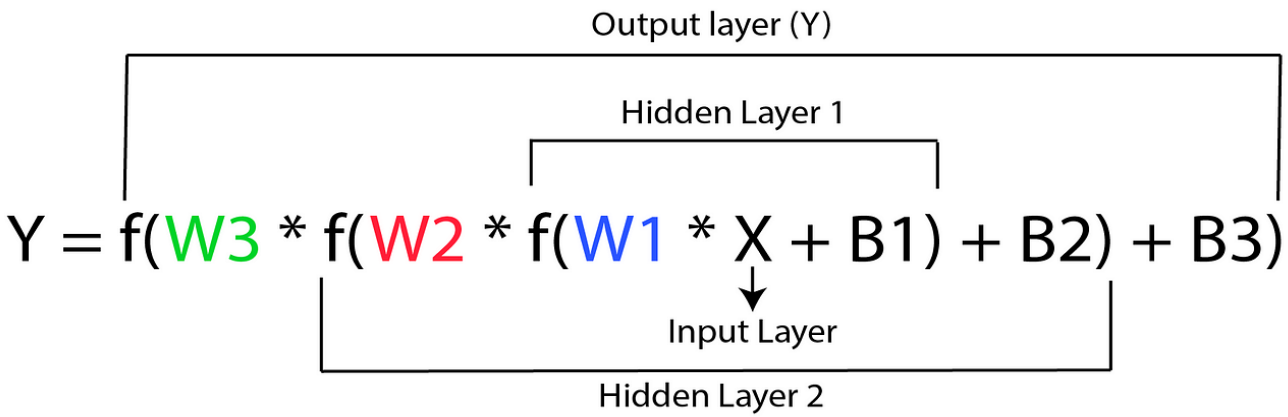
$$Y = f(H)$$


Input
Output
B1
X1
H1
Y1 = f(H1)
B2
X2
H2
Y2 = f(H2)
B3
X3
H3
Y3 = f(H3)


Intermediate Node Values
H1
H2
H3
=
Blue Weights
Red Weights
Green Weights
Weight Matrix
Matrix Multiplication
×
Input Vector
X1
X2
X3
+
Bias Vector
B1
B2
B3


Number of Columns (N):
Number of Input Nodes
Number of Rows (M):
Number of Output Nodes
Weight Matrix (MxN)


Hidden Layer 1
Hidden Layer 2
Input Layer
(X)
Output Layer
(Y)
W1
(4x3)
W2
(4x4)
W3
(1x4)


Output layer (Y)
Hidden Layer 1
Y = f(W3 * f(W2 * f(W1 * X + B1) + B2) + B3)
Input Layer
Hidden Layer 2

“Non-linear function/model” = mathematical view

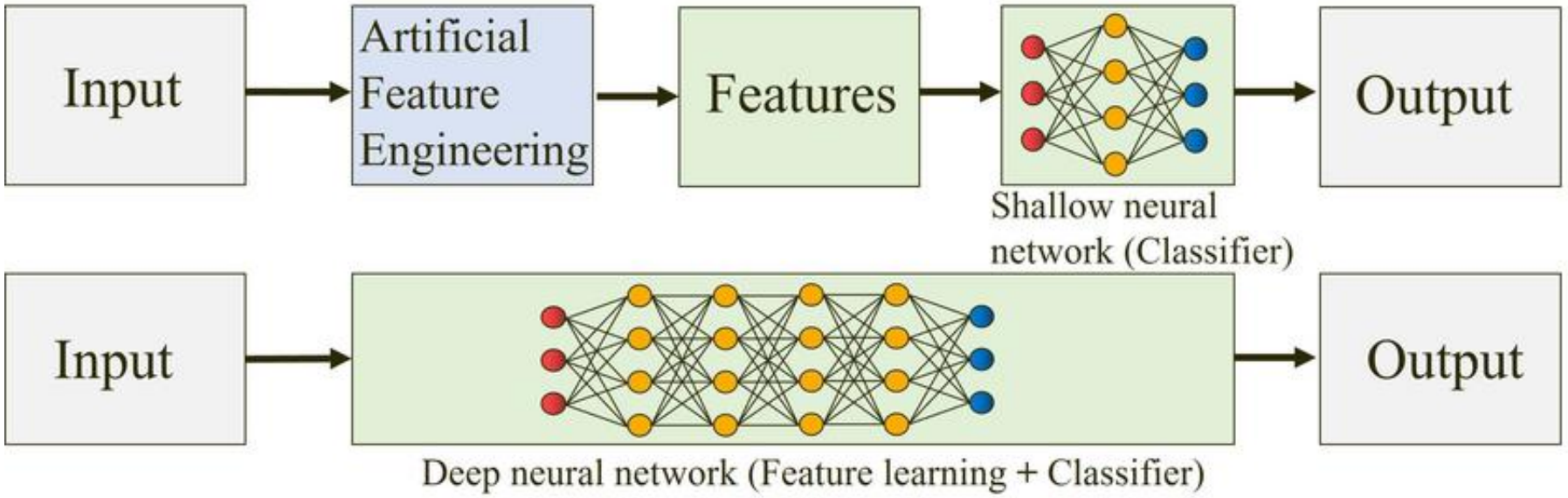
“Neural network” = structural/biological inspiration view

Neuron/Processor/Node → Computational unit

What is deep learning?

- Representation learning using stacks of layers → Deep Learning
- Hand-crafted features + Classifier (Traditional learning) → Shallow learning

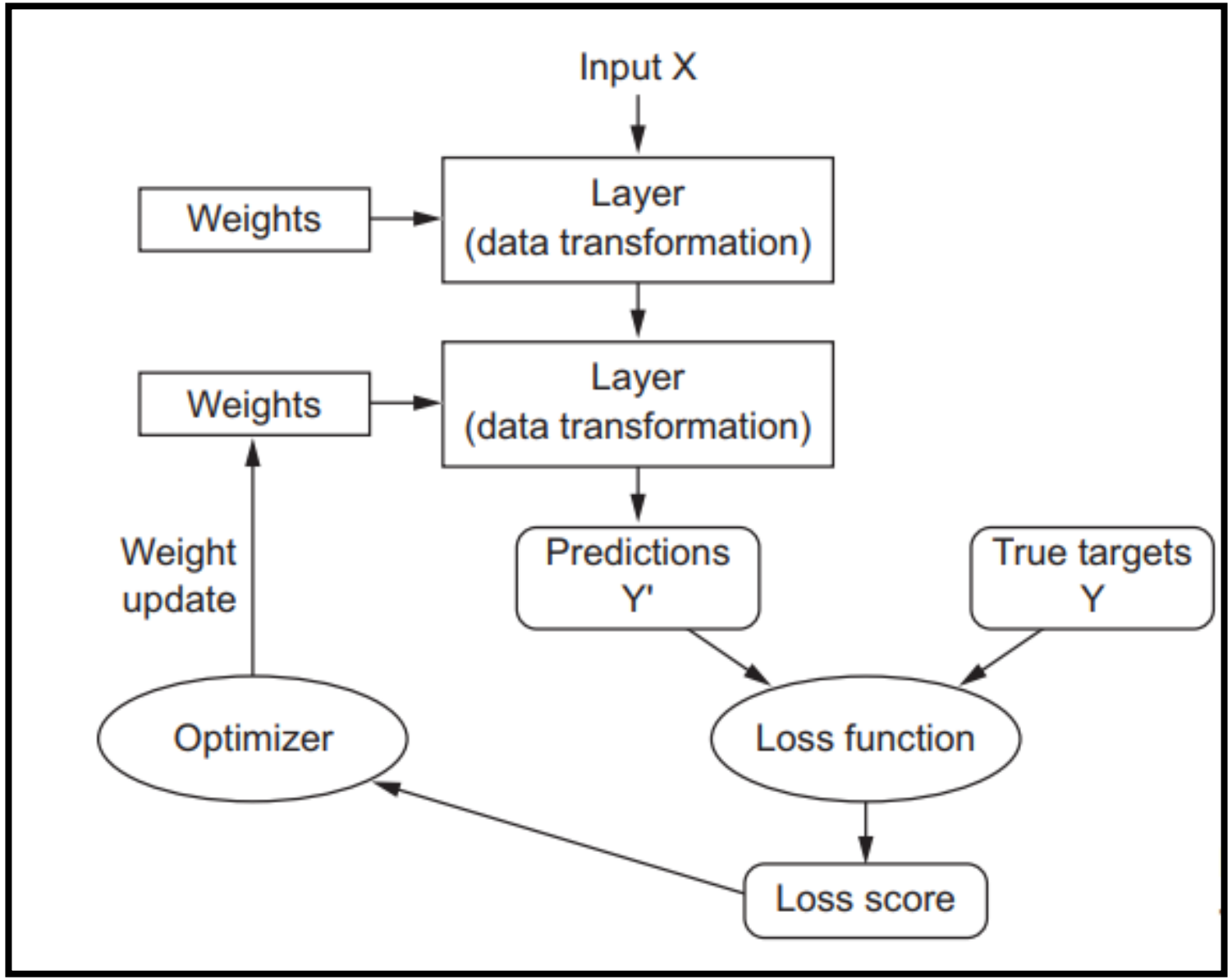
Shallow learning relies heavily on manual feature design, whereas deep learning automatically learns multi-level, transferable representations from raw data, making it far more powerful for representation learning.

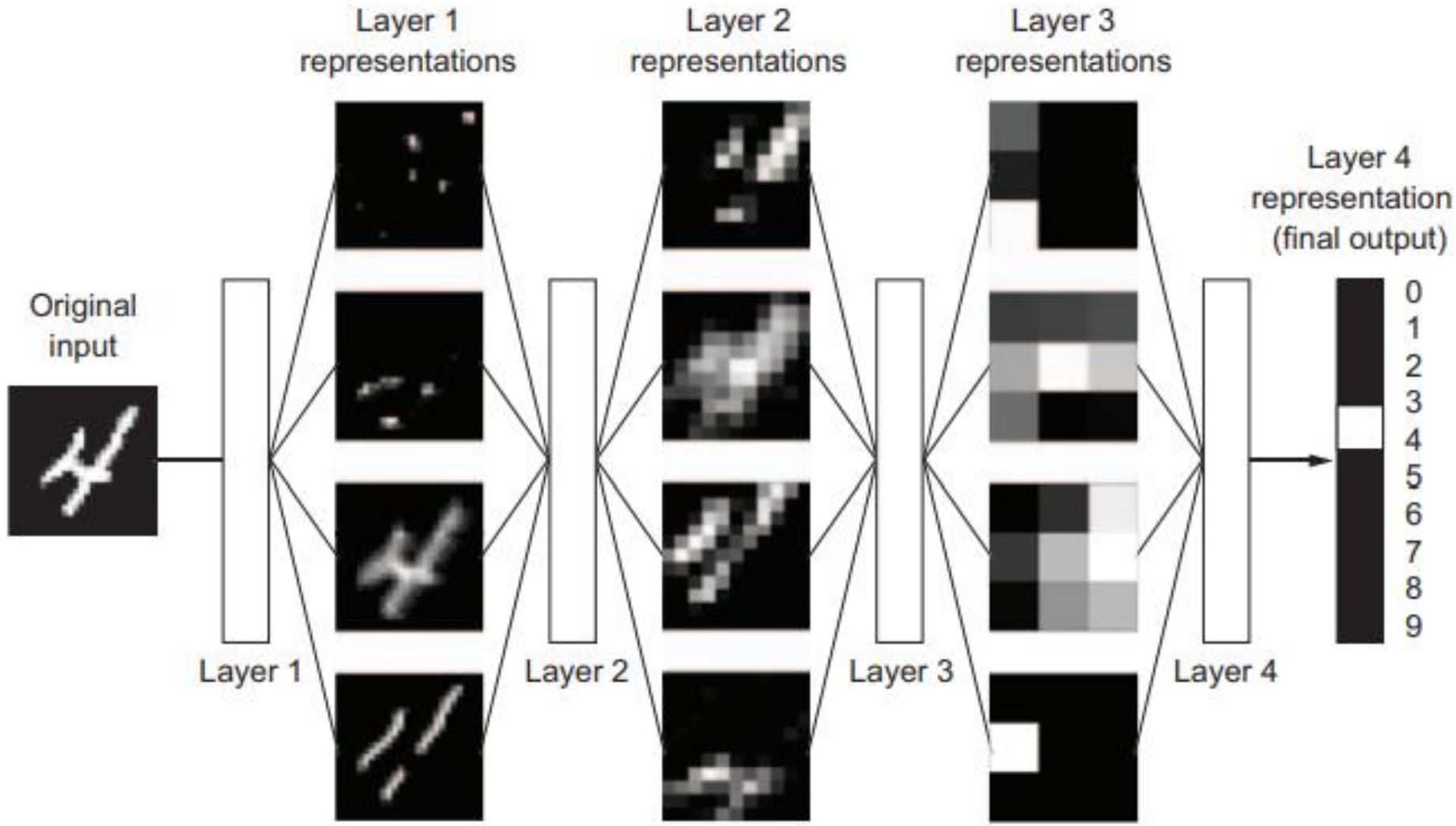


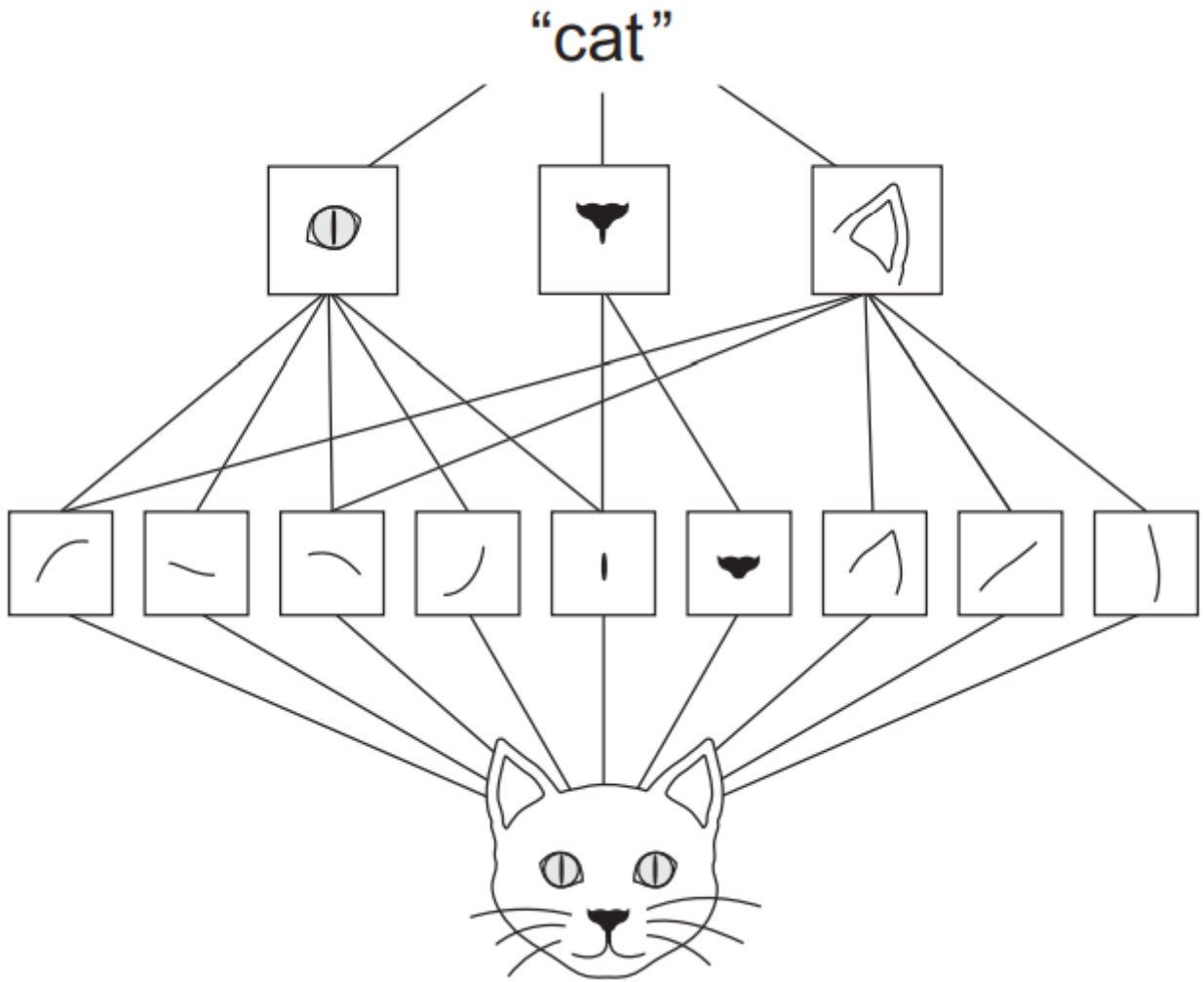
- Deep-learning models are not models of the brain. There's no evidence that the brain implements anything like the learning mechanisms used in modern deep-learning models.
- You may come across pop-science articles proclaiming that deep learning works like the brain or was modelled after the brain, but that isn't the case. It would be confusing and counterproductive for newcomers to the field to think of deep learning as being in any way related to neurobiology;
- Deep learning is just a mathematical framework for learning representations from data; it is not a model of the brain.

"Deep Learning with Python" — François Chollet (Manning, 2nd ed.)

Mathematical framework of deep learning




Layer 1 representations
Layer 2 representations
Layer 3 representations
Layer 4 representation (final output)
Original input
Layer 1
Layer 2
Layer 3
Layer 4
0
1
2
3
4
5
6
7
8
9


“cat”

- Shallow:

$$f(\mathbf{x}) = \text{sign}(w^T \mathbf{x} + b)$$

- Deep:

$$f(\mathbf{x}) = \text{softmax}(W^{(L)} \cdots \sigma(W^{(1)} \mathbf{x} + b^{(1)}))$$

- Ensemble:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m \cdot \text{sign}(w_m^T \mathbf{x} + b_m)$$