14.332.435/16.332.530
Introduction to Deep Learning

Lecture 2
Machine Learning Basics (1)

Yuqian Zhang

Department of Electrical and Computer Engineering

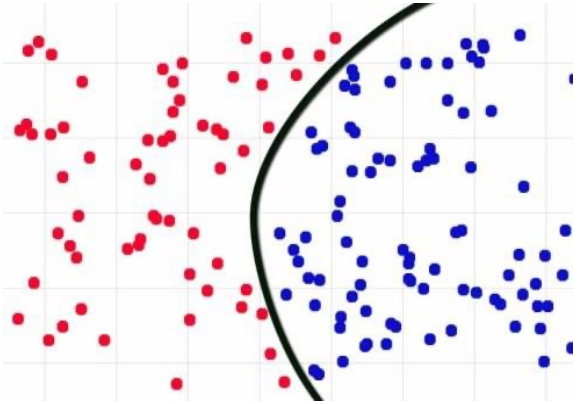
What does Learning Mean

- ML algorithm is an algorithm able to *learn* from data
- A computer program to said to learn if
  its performance P on task T can improve with experience E

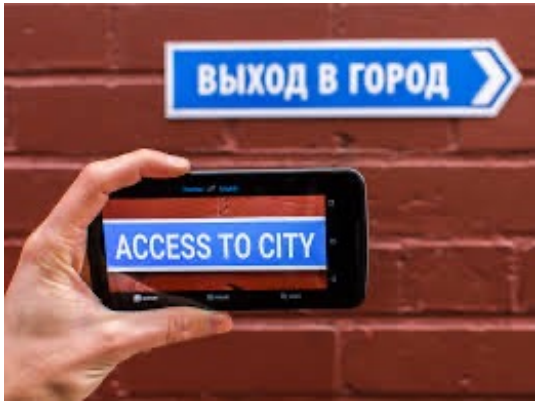
- Task: What is ML algorithm used for
- Performance: Quantitative metric to evaluate algorithm
- Experience: Effect of ML algorithm on dataset

Example Tasks of Machine Learning

Classification



Translation



Denoise

Noisy input Our result

Performance Metrics of ML

- Different tasks T have different performance measure *P*

Image Classification

TABLE II: AlexNet on ImageNet [4]. PD: Permuted Diagonal.

AlexNet	Block size (p) for PD weight matrix of FC6-FC7-FC8	Top-5 Acc.	Compression for overall FC layers
Original 32-bit float	1-1-1	80.20%	234.5MB(1×)
32-bit float with PD	10-10-4	80.00%	25.9MB(9.0×)
16-bit fixed with PD	10-10-4	79.90%	12.9MB(18.1×)

Language Translation

TABLE III: Stanford NMT (32-FC layer LSTMs) on IWSLT15 for English-Vietnamese Translation.

Stanford NMT (32-FC layer LSTMs)[5]	Block size (p) for PD weight matrix of ALL FC Layers	BLEU Points	Compression for overall FC layers
Original 32-bit float	1	23.3	419.4MB(1×)
32-bit float with PD	8	23.3	52.4MB(8×)
16-bit fixed with PD	8	23.2	26.2MB(16×)

Taxonym of Machine Learning: Different Experience

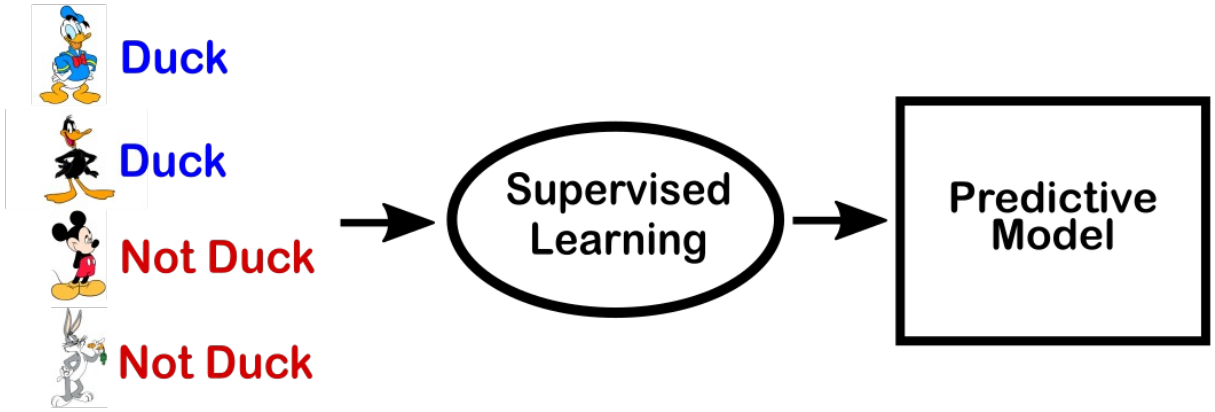
According to what kind of experience they are allowed to have during the learning process, ML algorithms can be categorized as:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning

Supervised learning

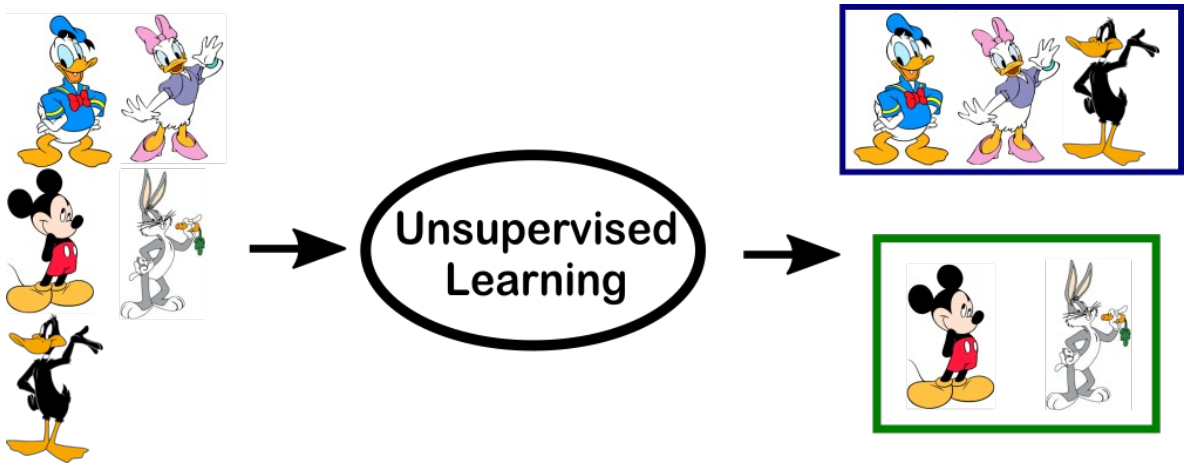
- Learn from labeled examples
- Each example is a pair consisting of an input object and a desired output value (a.k.a supervisory signal).



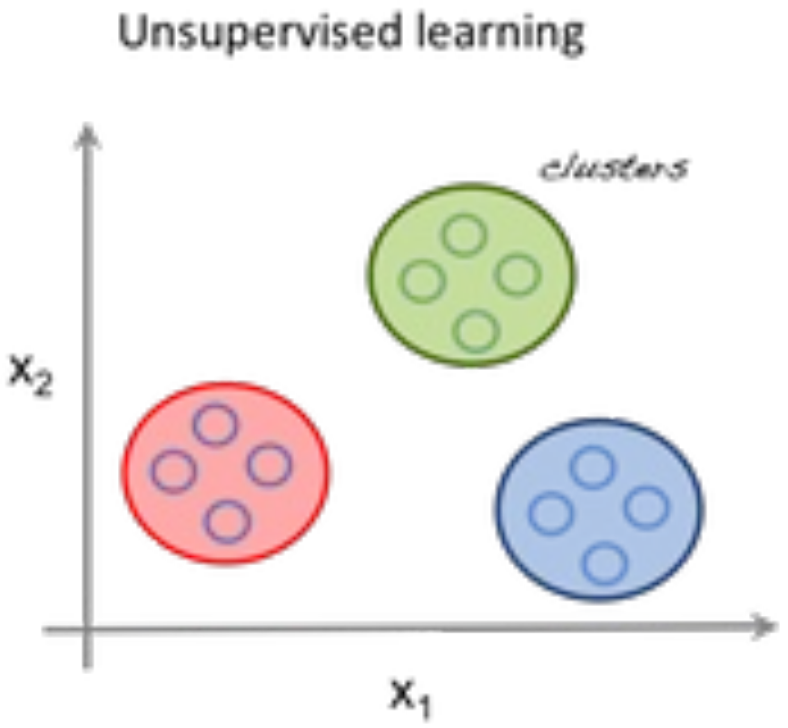
Unsupervised learning

Unsupervised learning

- Draw inferences from datasets consisting of input data **without** labeled responses
- Find hidden patterns or grouping in data



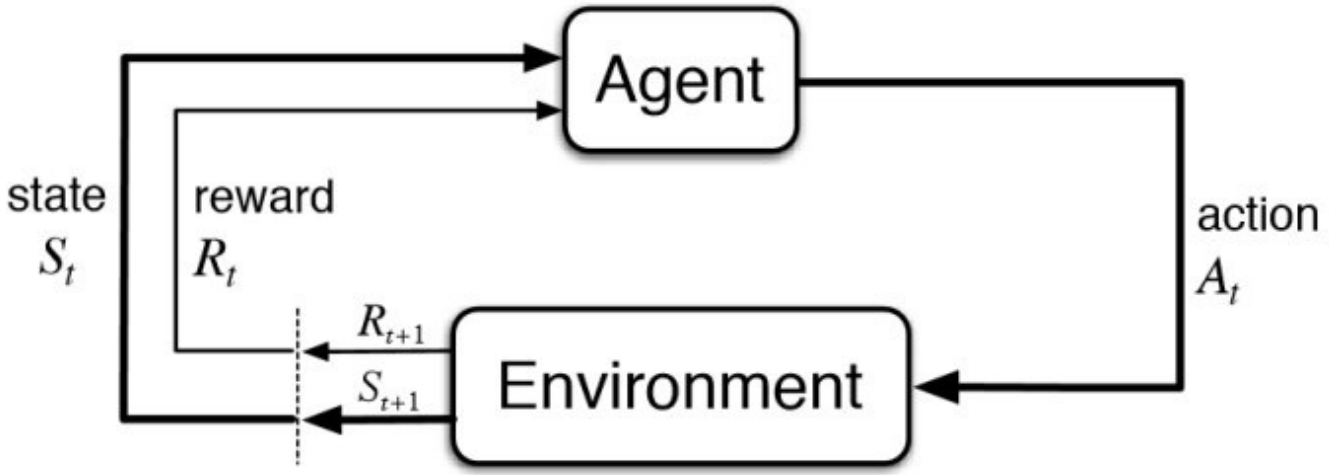
Supervised vs Unsupervised Learning



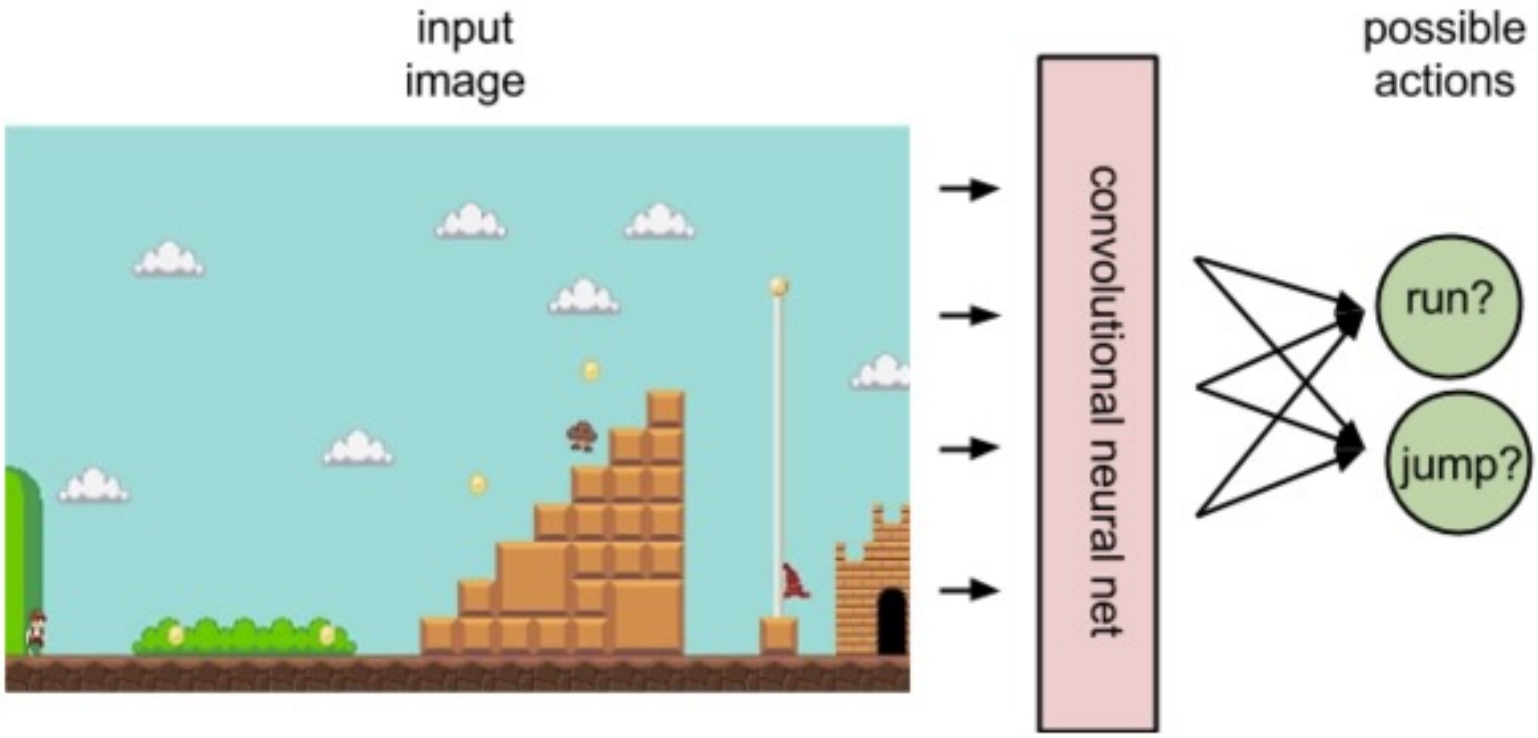
Reinforcement learning

Reinforcement learning

- Experience dataset and **interact with environment**
- **Feedback loop** between ML system and environment



Convolutional Agent



Recall Prior Example

An image classifier

```
def classify_image(image):
  # Some magic here?
return class_label
```

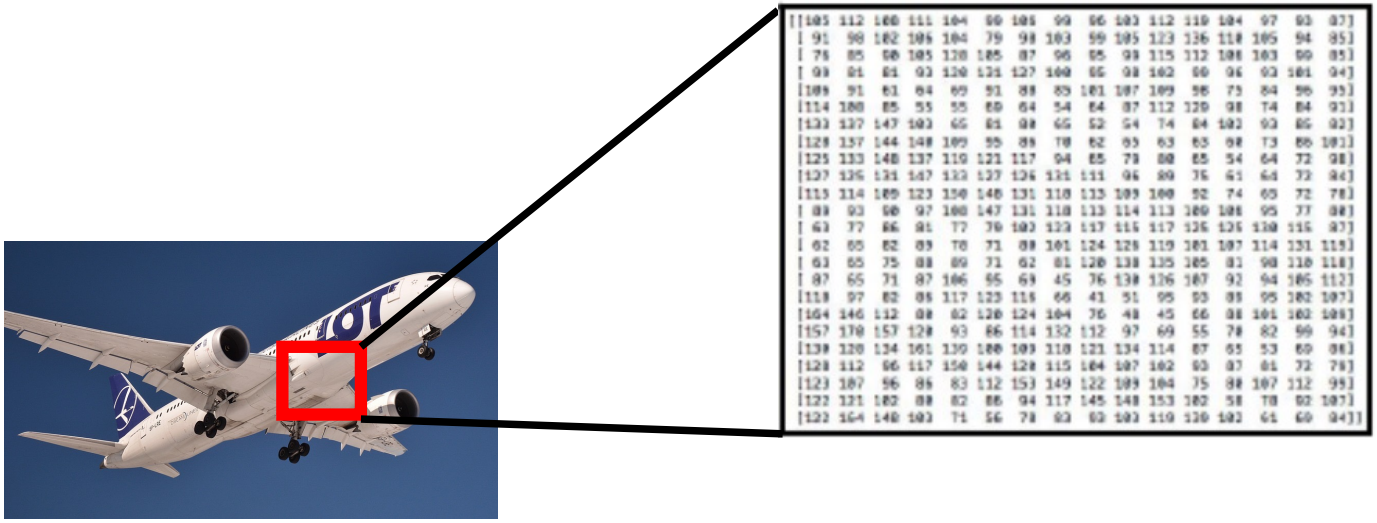
Plane

In this course we usually take image classification in computer vision as example

- Computer vision is the most successful field for DNNs
- Image classification is the core task of computer vision

Challenge: Semantic Gap

- In computer, each pixel is represented with a number
- Computer views an image as a big grid of numbers



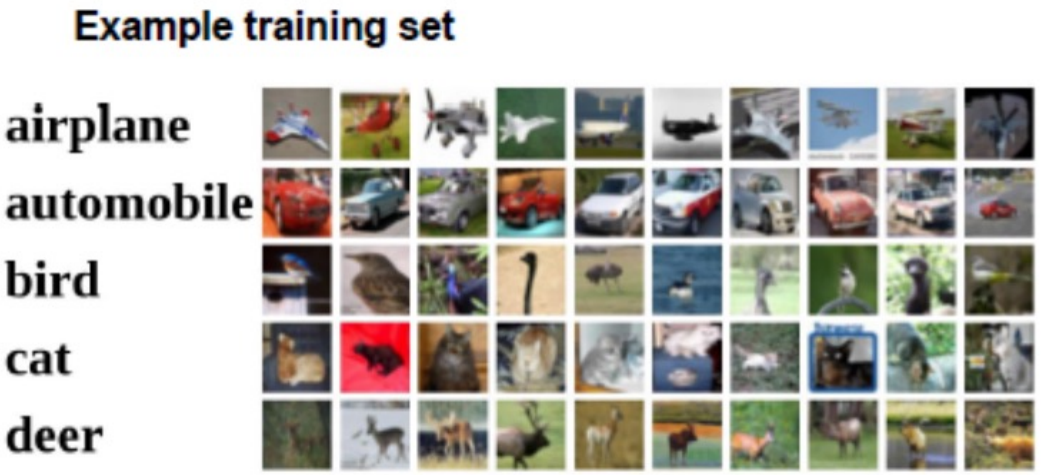
Computer vision is not human vision

Challenge: Different Sizes, Shapes, Angles…

Machine Learning: Data-driven Approach

```
def train(images, labels):
  # Machine learning!
return model
```

```
def predict(model, test_images):
  # Usemodel to predict labels
return test_labels
```



1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

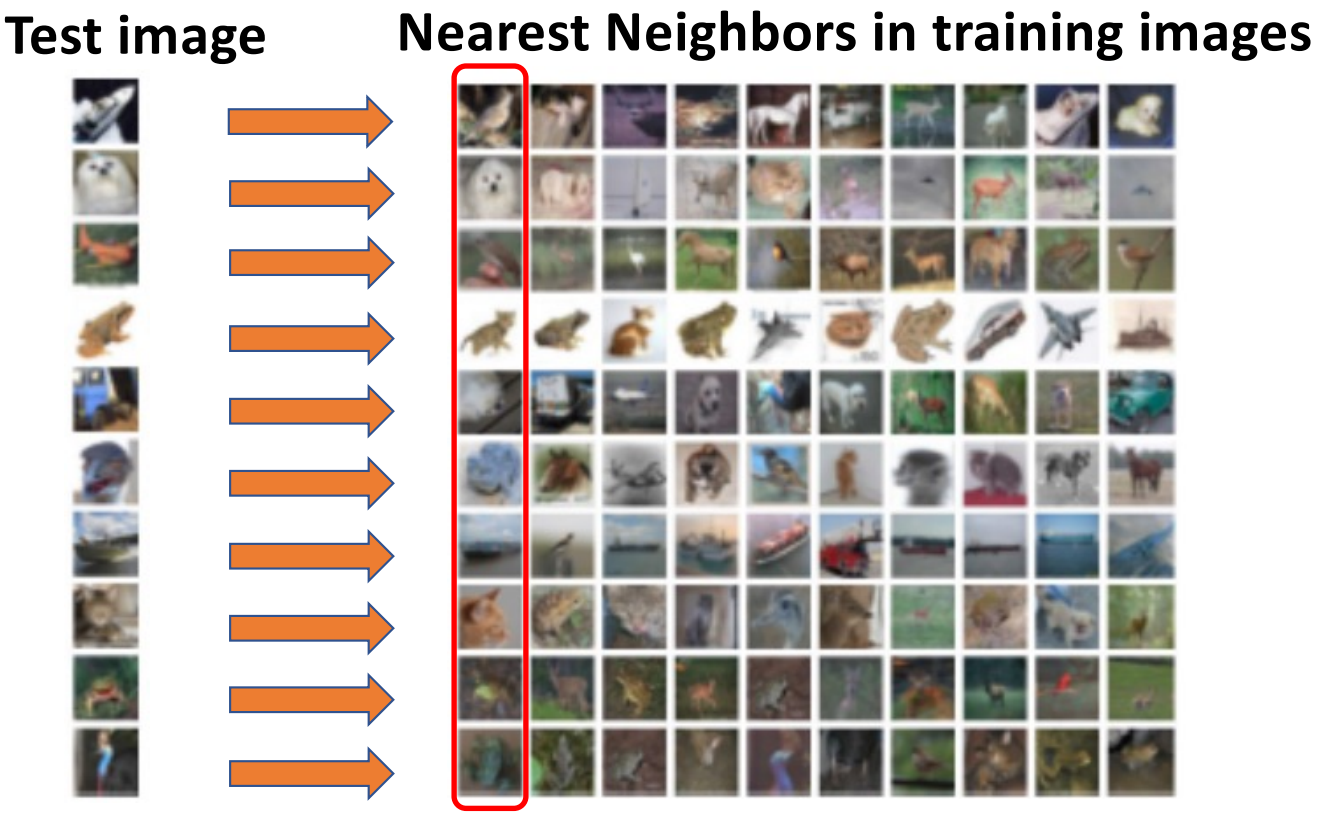
A Naive Idea: Find Similar Images

```
def train(images, labels):
  # Machine learning!
return model
```

Memorize all data and labels

```
def predict(model, test_images):
  # Usemodel to predict labels
return test_labels
```

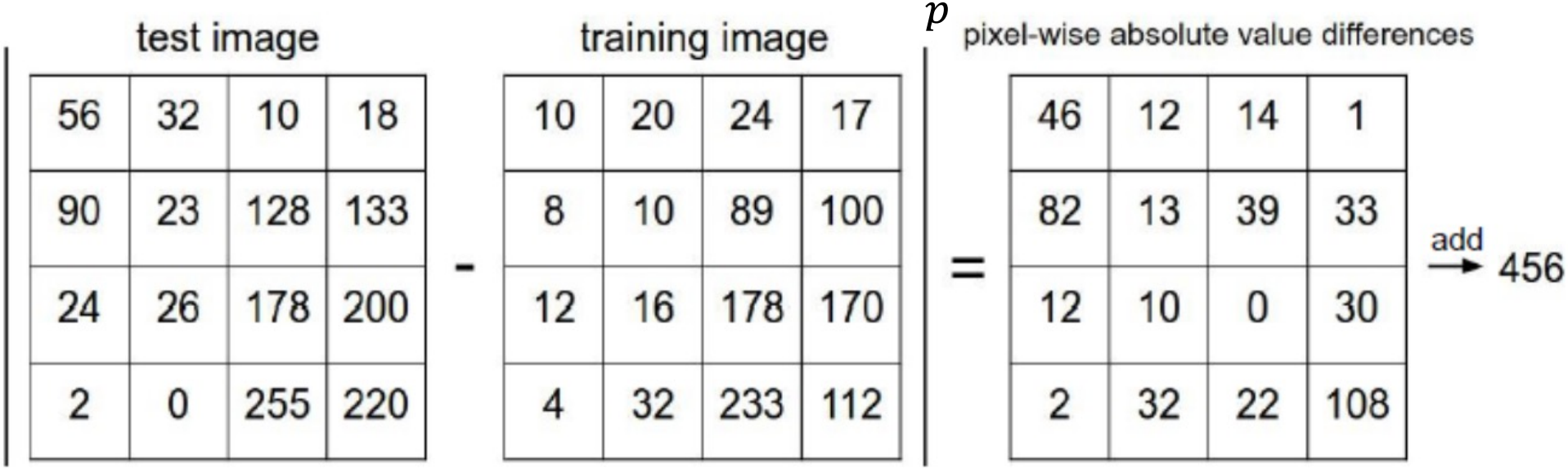
Predict the label of the most similar training images


Test image
Nearest Neighbors in training images

How to Determine the Nearest?

L-distance is good metric:

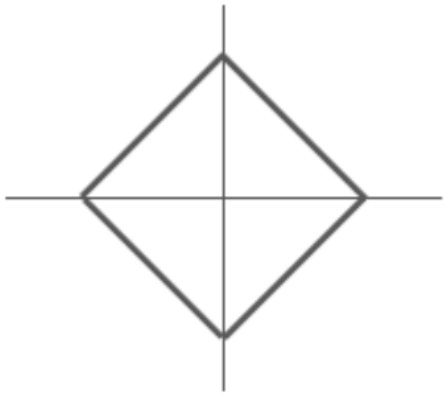
e.g.: L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$ ***p*: dimension**



Distance Metric

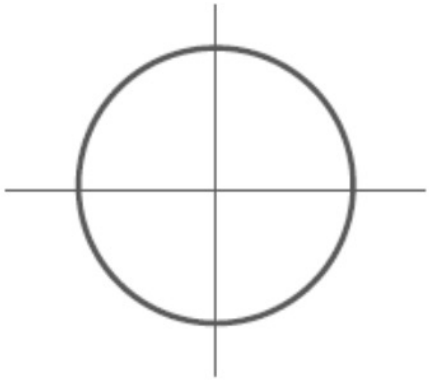
L1(Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2(Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



Example Python Codes

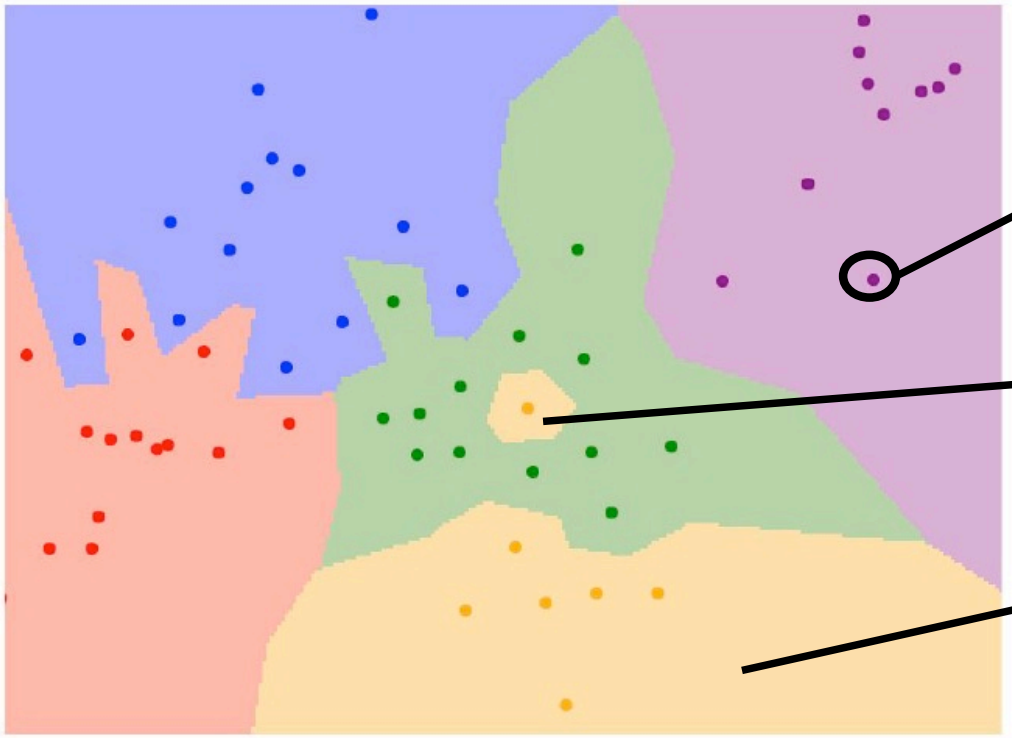
```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

With N training examples, time complexity for
Train is O(1) ----> Quick
Prediction is O(N) ----> Slow

Memorize training data

For each test image
Find nearest train image
Predict label of nearest image

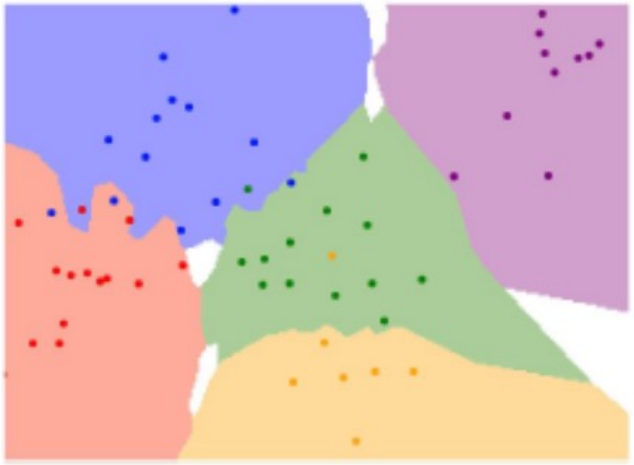
Disadvantage If Only Considering Nearest



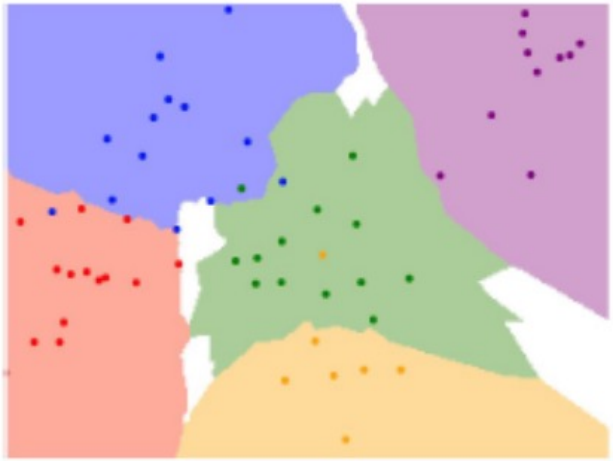
Improvement: Using K Nearest Neighbors (KNN)

- KNN: Take **majority vote** from *K* closest points

K = 1



K = 3



K = 5

Hyperparameter

- When setting up KNN, you can choose two parameters:

  What is the best value of k to use for voting?

  What is the best distance to use for measuring?

- These are hyperparameters, which are not adapted by the machine learning algorithm itself.
- In DNN, many hyperparameters

  # of layers/neuron, learning rate, weight decay……

Train, Validation and Test Dataset

- Training set

  -- a set of data used to discover predictive relationships

  -- simple words: help to learn the model

- Test set

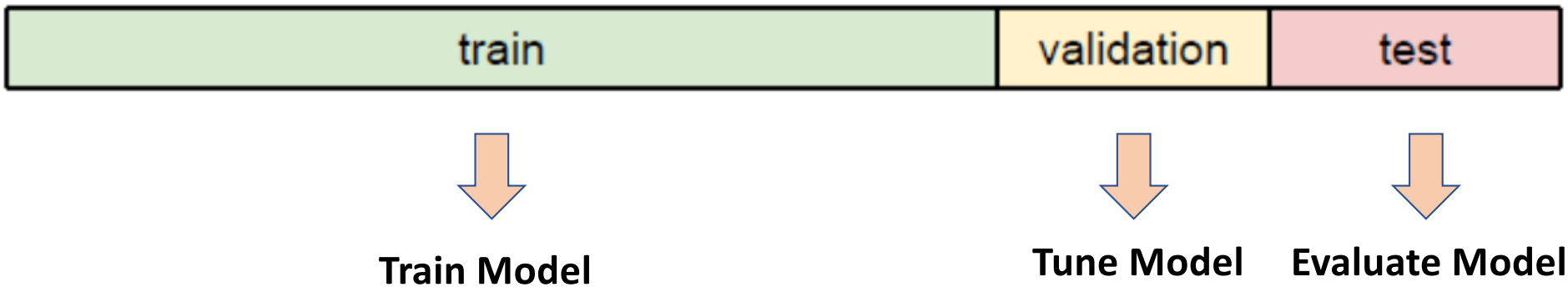
  -- a set of data used to assess the strength and utility of a predictive relationship

  -- simple words: help to evaluate a learned model

- Validation set

  -- a set of examples used to tune the hyperparameters

Train, Validation and Test Dataset

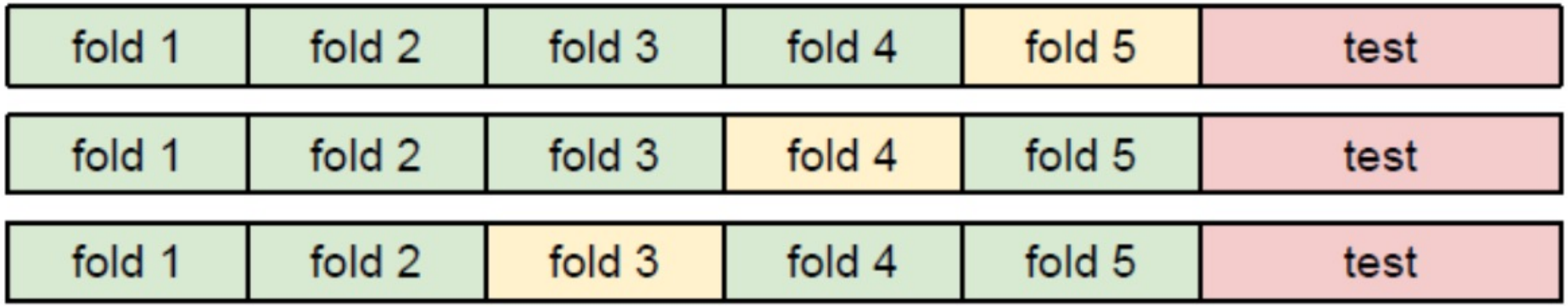


Never use test data to train model!

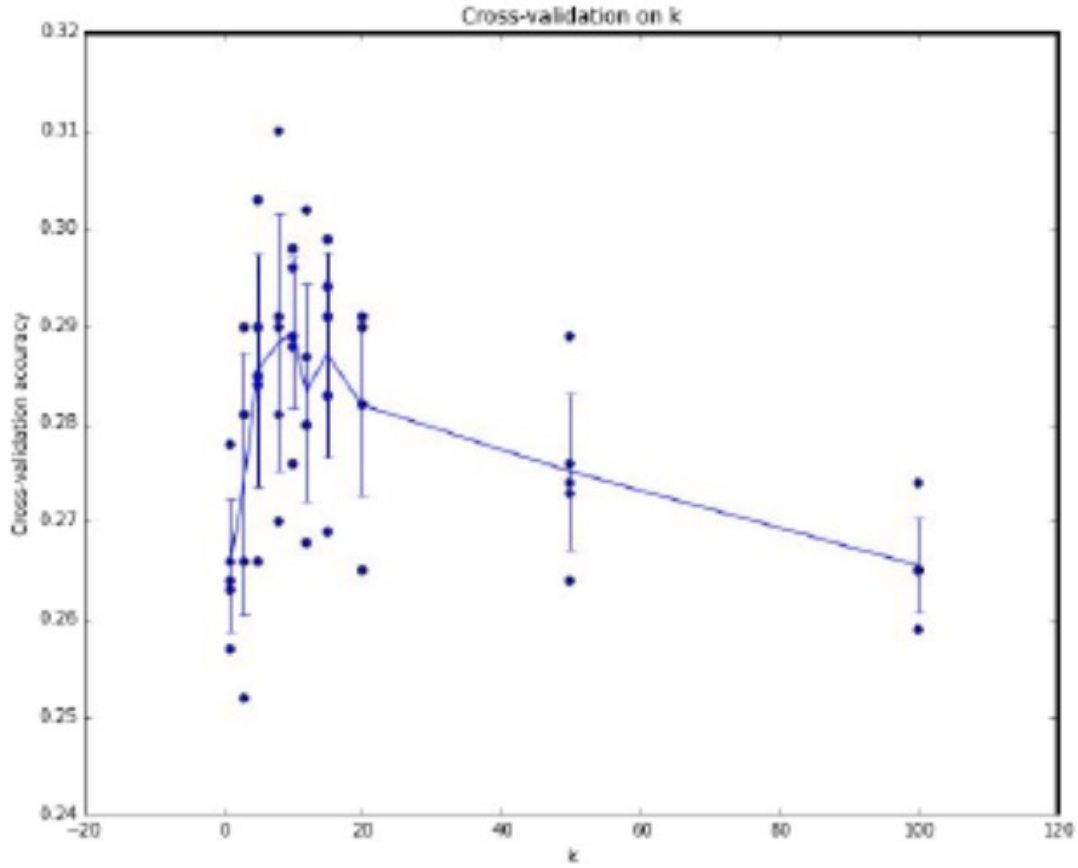
Cross-validation

When dataset is small

- Split data, try each fold as validation, and average
- Not too frequent in deep learning

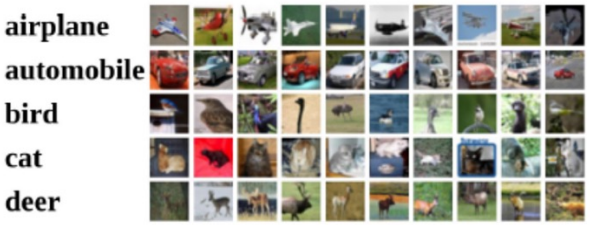


5-fold Cross-validation for KNN



Summary of Non-parametric KNN

Training Examples



ML Model
($f(x_{train})$)

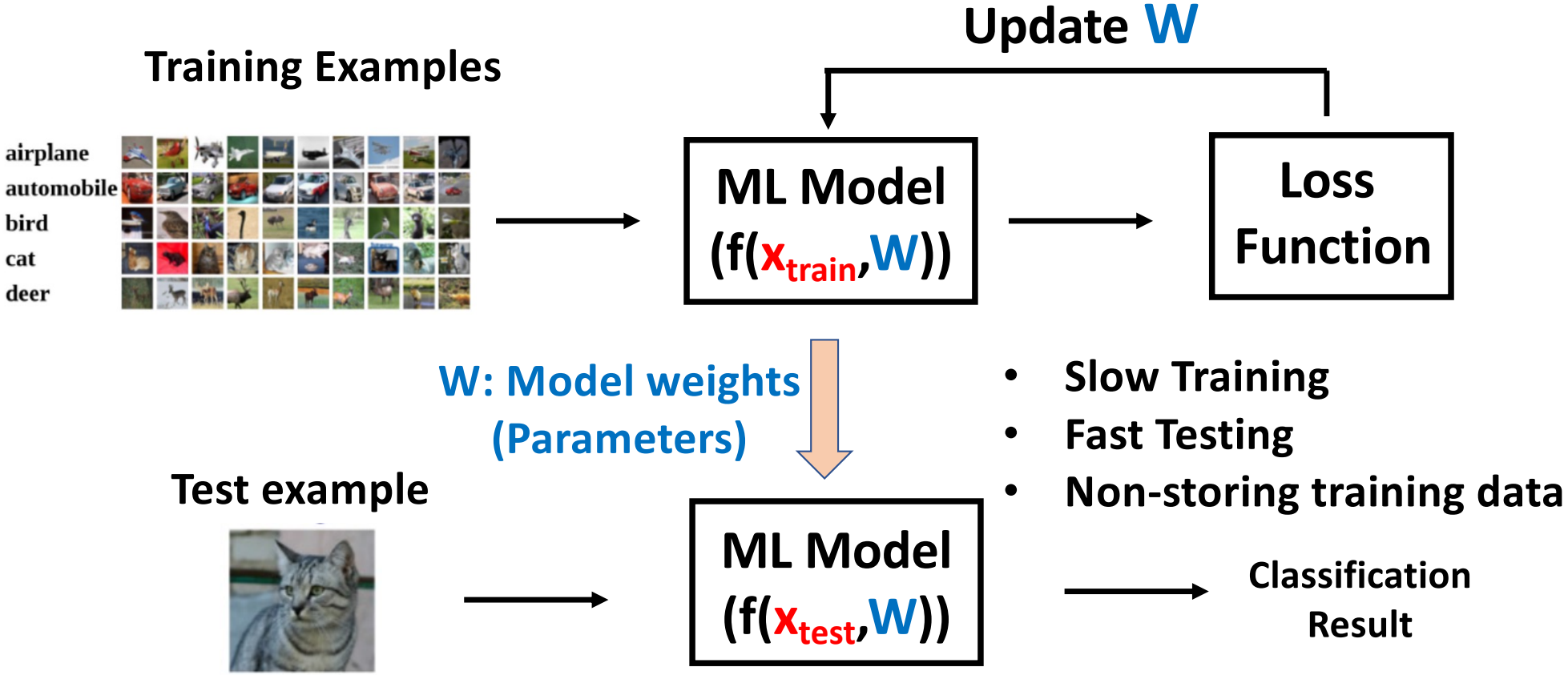
- **Pros:** Fast Training
  Easy to implement
- **Cons:** Slow Testing
  Store all training data

Test example

ML Model
($f(x_{test}, x_{train})$)

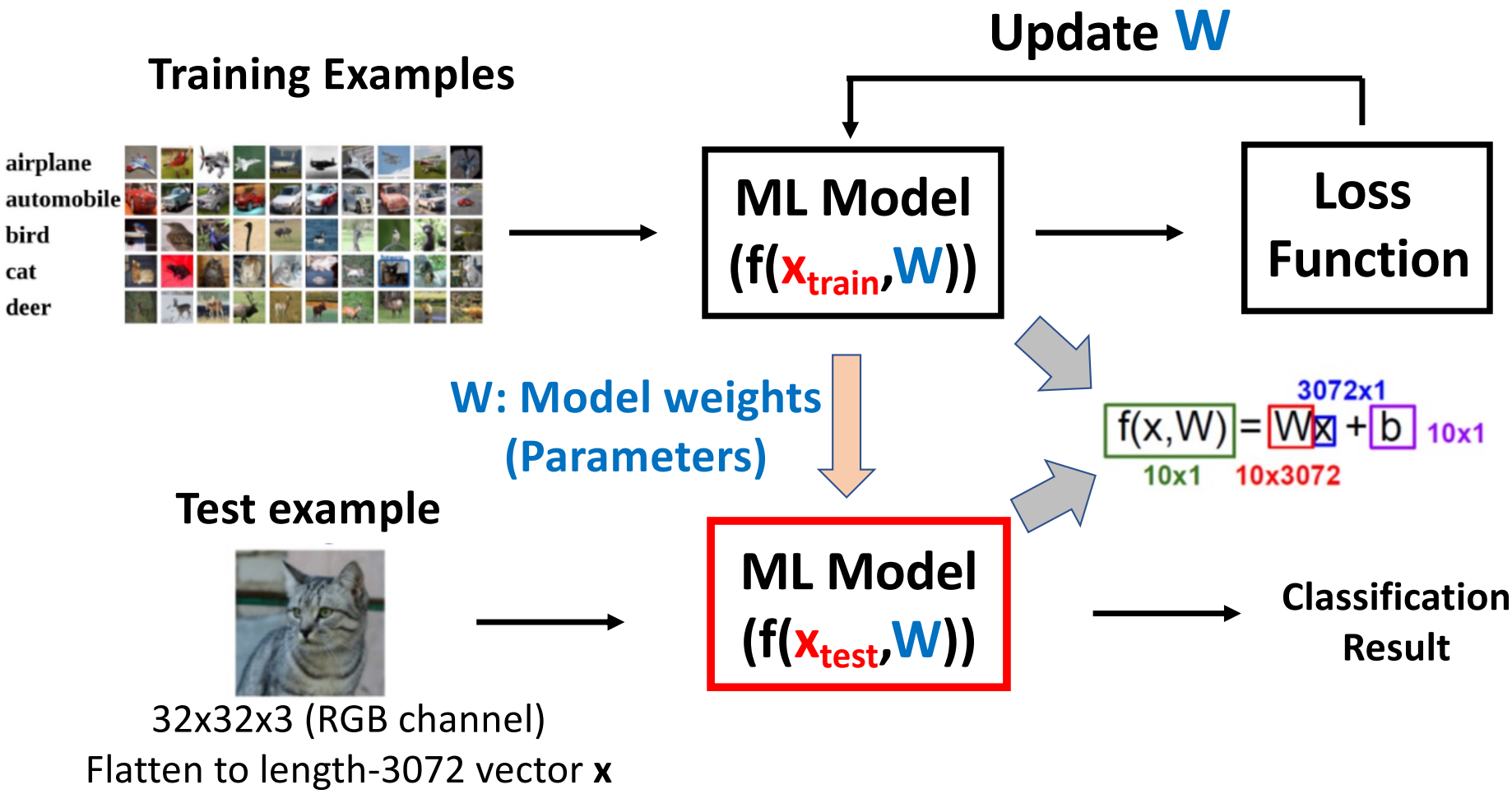
Classification Result

Parametric Machine Learning Model

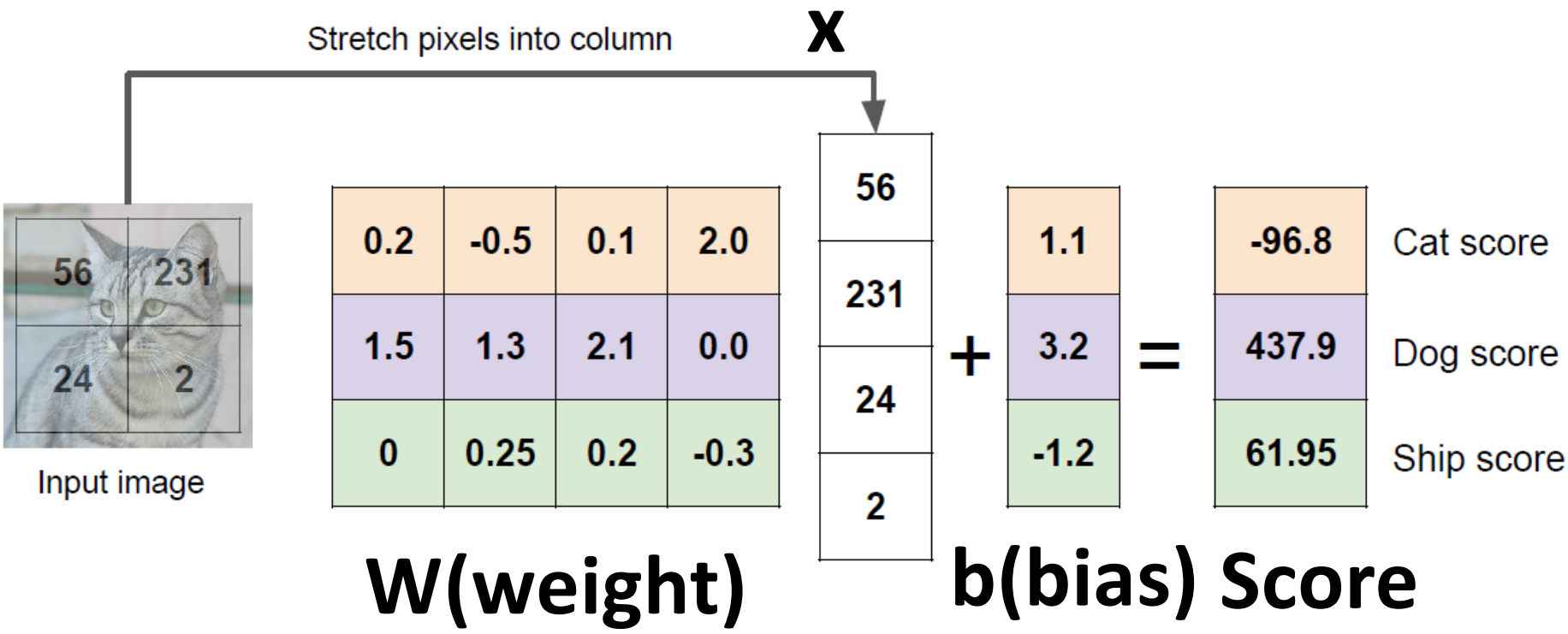


- Slow Training
- Fast Testing
- Non-storing training data

Linear Classifier

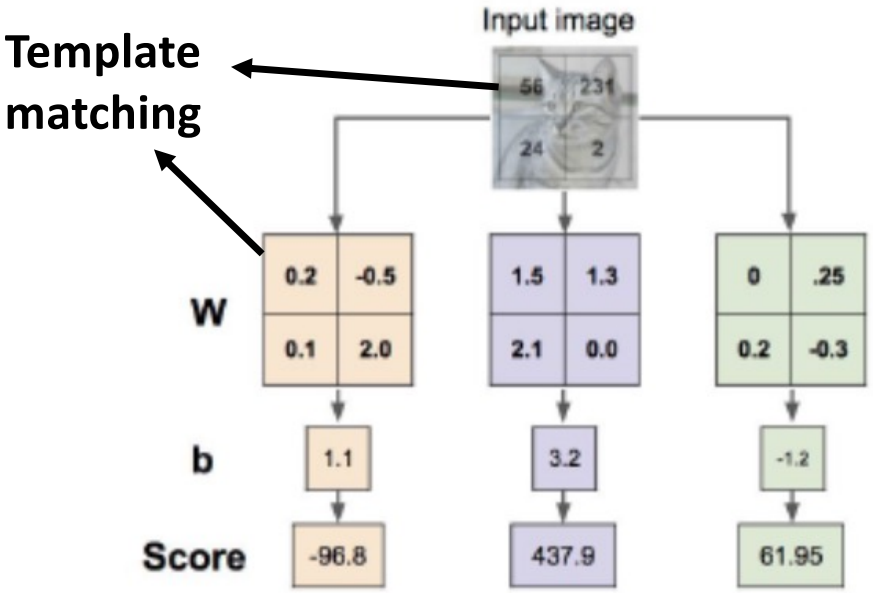


Vectorized Example



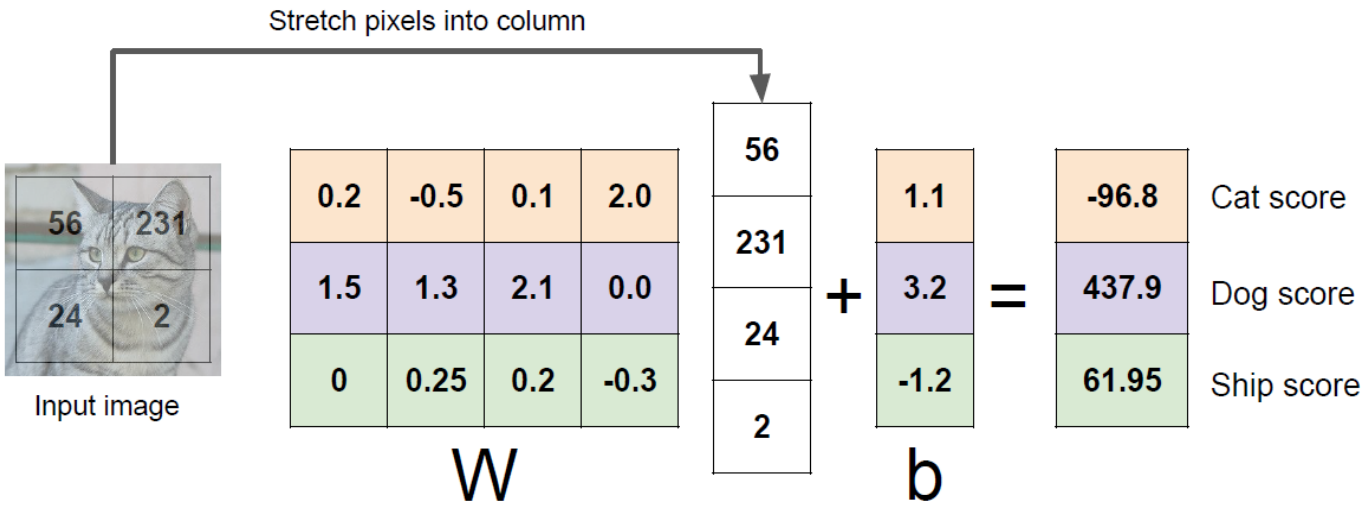
Q: Is it a well-trained model?

Visual Viewpoint of a Linear Classifier



Why We Need Bias

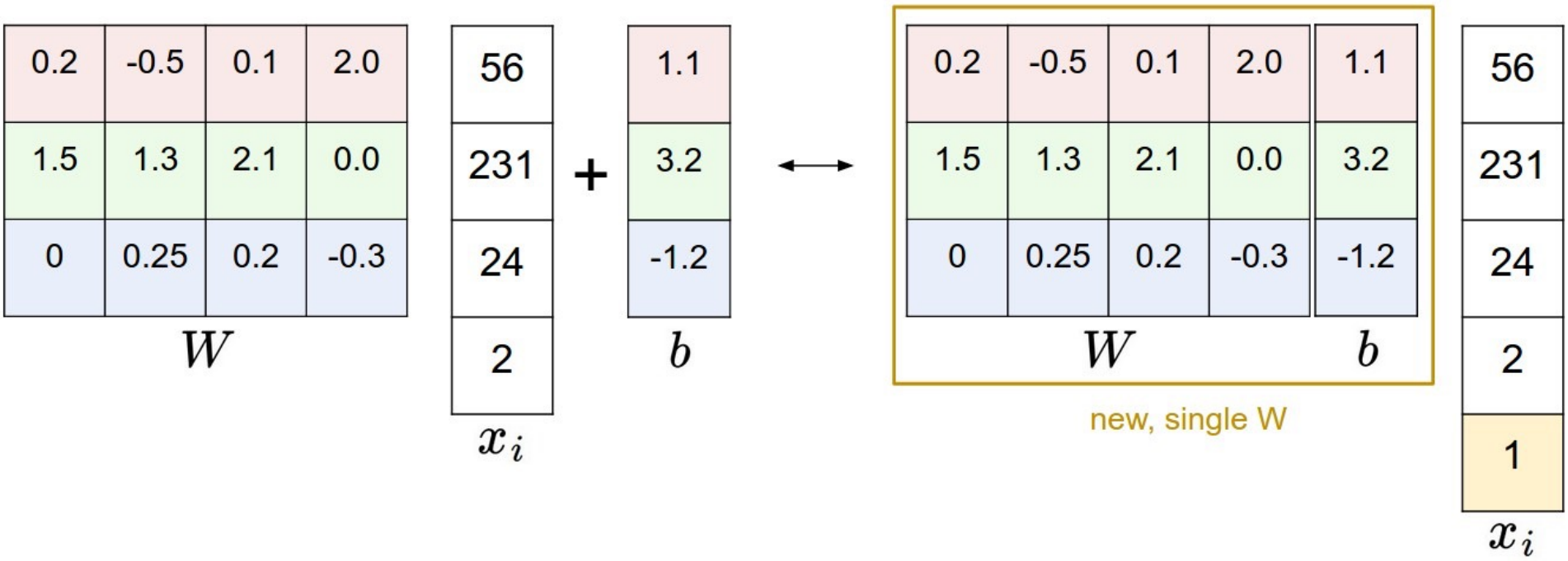
- What happens if inputs are all zeros without bias?



- Viewpoint of function approximation

--Linear classifier is to approximate an (assumed) linear relationship

Weights and Bias can be Unified



Acknowledgement

Many materials of the slides of this course are adopted and re-produced from several deep learning courses and tutorials.

- Prof. Fei-fei Li, Stanford, CS231n: Convolutional Neural Networks for Visual Recognition (online available)
- Prof. Andrew Ng, Stanford, CS230: Deep learning (online available)
- Prof. Yanzhi Wang, Northeastern, EECE7390: Advance in deep learning
- Prof. Jianting Zhang, CUNY, CSc G0815 High-Performance Machine Learning: Systems and Applications
- Prof. Vivienne Sze, MIT, “Tutorial on Hardware Architectures for Deep Neural Networks”
- Pytorch official tutorial https://pytorch.org/tutorials/