Image Processing & Vision

Lecture 07: Classification I

Hak Gu Kim

hakgukim@cau.ac.kr

Immersive Reality & Intelligent Systems Lab (IRIS LAB)

Graduate School of Advanced Imaging Science, Multimedia & Film (GSAIM)

Chung-Ang University (CAU)

1 May 2023

Topics

- ML Overview
- Classification
— K-Nearest Neighbor

***Note:** Many of these slides in this course were adapted from Convolutional Neural Networks for Visual Recognition (Stanford Univ.) and Deep Learning for Computer Vision (Univ. of Michigan)

Machine Learning (ML)

- Learn from data and make predictions on data

- **ML Problems**

	Supervised Learning	Unsupervised Learning
Discrete	Classification	Clustering
Continuous	Regression	Dimensionality Reduction

Machine Learning (ML)

- Learn from data and make predictions on data

- **ML Problems**

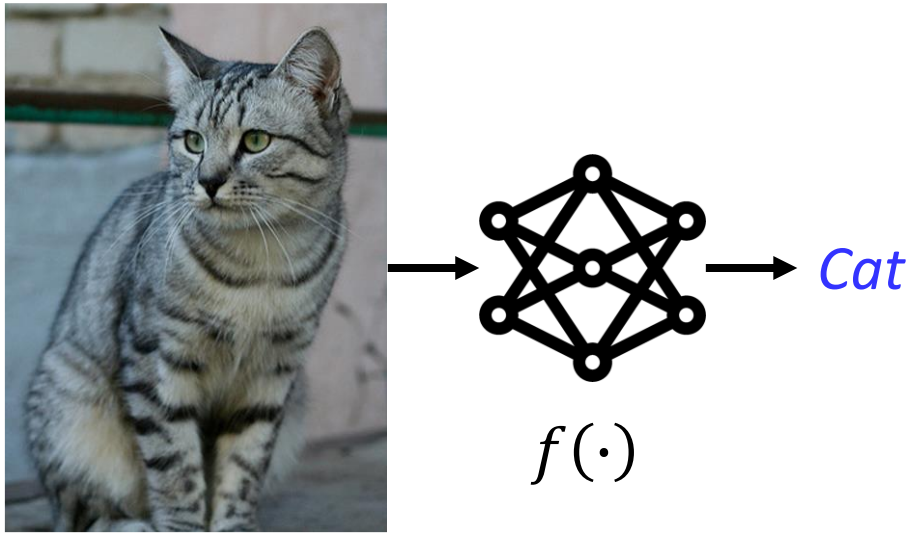
	Supervised Learning	Unsupervised Learning
Discrete	**Classification**	Clustering
Continuous	Regression	Dimensionality Reduction

Supervised Learning – Discrete: **Classification**

- Supervised learning

— The ML task of learning a function that maps an input to an output based on **example input-output pairs**

— It infers a function from *labeled* training data (a set of training examples), which is a pair consisting of an input object and a desired output value



Supervised Learning – Discrete: **Classification**

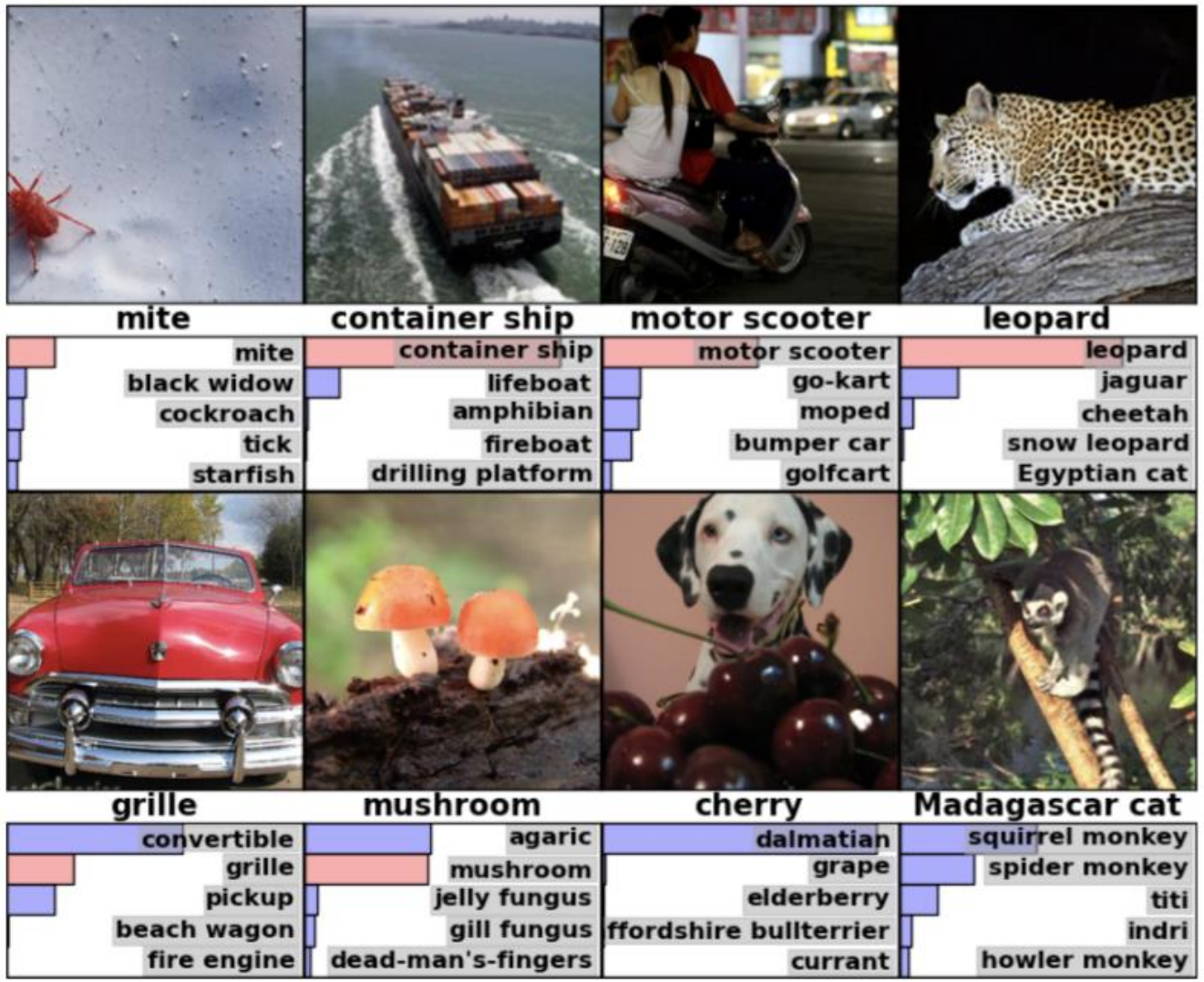
- **Classification**

— Apply the trained prediction function to a feature representation of the image to get the desired output

$$f(\text{  }) = cat$$

$$f(\text{  }) = dog$$

Supervised Learning – Discrete: **Classification**



Machine Learning (ML)

- Learn from data and make predictions on data

- **ML Problems**

	Supervised Learning	Unsupervised Learning
Discrete	Classification	Clustering
Continuous	**Regression**	Dimensionality Reduction

Supervised Learning – Continuous: **Regression**

- **Classification model for prediction**

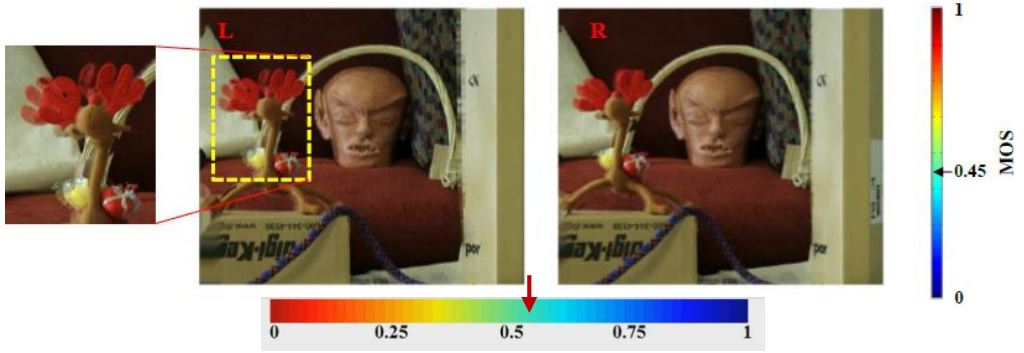
— It is the task of approximating a mapping function from input variable to discrete output variables

- **Regression model for prediction**

— It is the task of approximating a mapping function from input variable to continuous output variables

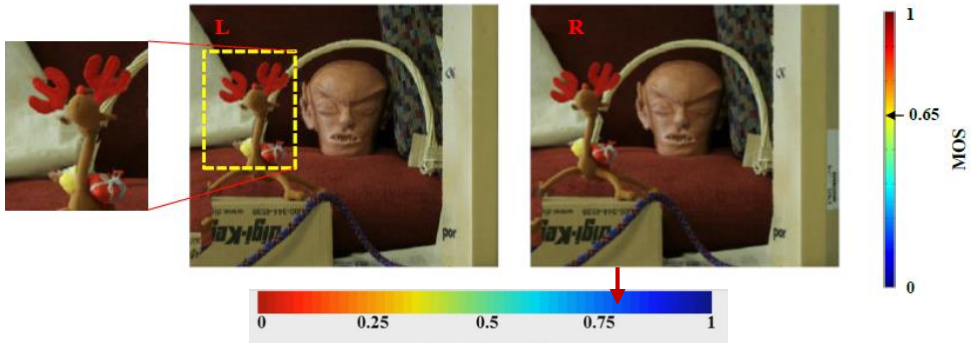
Supervised Learning – Continuous: **Regression**

- We want to find out the relationship between the image quality and subjective quality score measurements



Visual comfort score

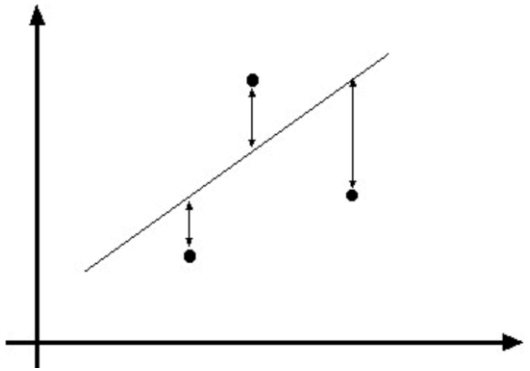
Binocular asymmetry stereo image

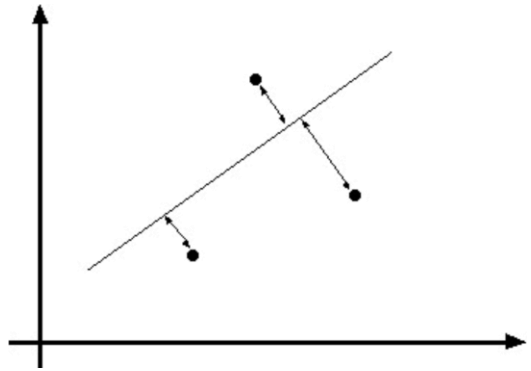


Visual comfort score

Binocular symmetry stereo image

- How can we define the linear relationship between these two measurements?





Machine Learning (ML)

- Learn from data and make predictions on data

- **ML Problems**

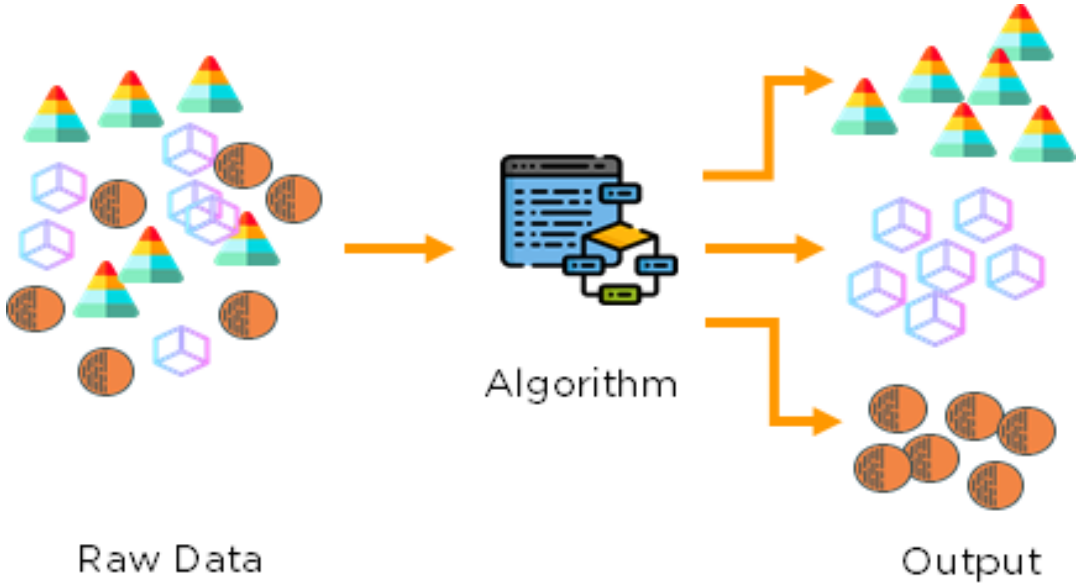
	Supervised Learning	Unsupervised Learning
Discrete	Classification	**Clustering**
Continuous	Regression	Dimensionality Reduction

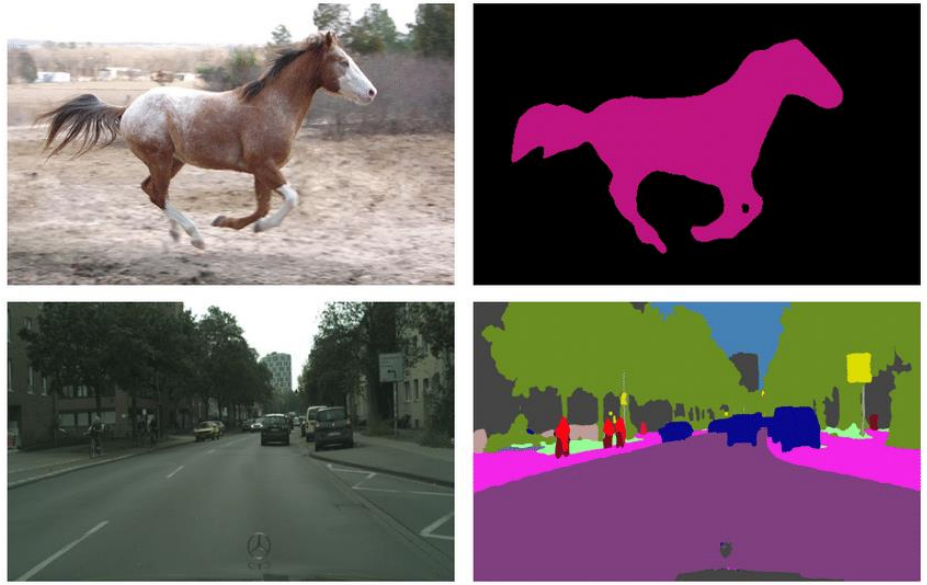
Unsupervised Learning – Discrete: **Clustering**

- Unsupervised learning

— The ML task of learning a function that analyzes and clusters unlabeled (untargeted) datasets

— It discovers hidden patterns or data groupings without the need for human intervention





Unsupervised Learning – Discrete: **Clustering**

Machine Learning (ML)

- Learn from data and make predictions on data

- **ML Problems**

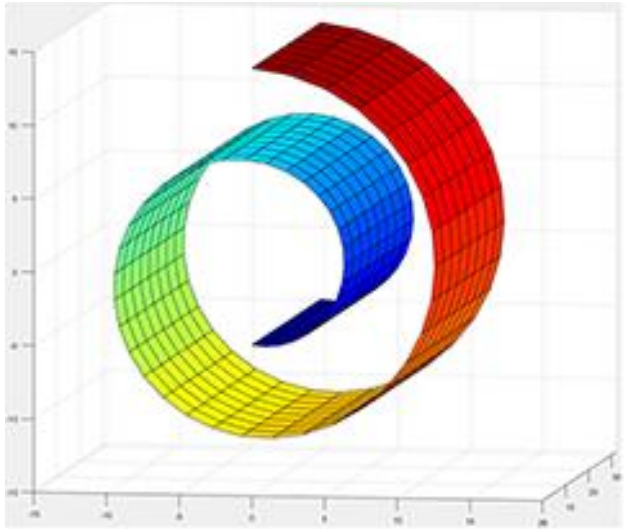
	Supervised Learning	**Unsupervised Learning**
Discrete	Classification	Clustering
Continuous	Regression	**Dimensionality Reduction**

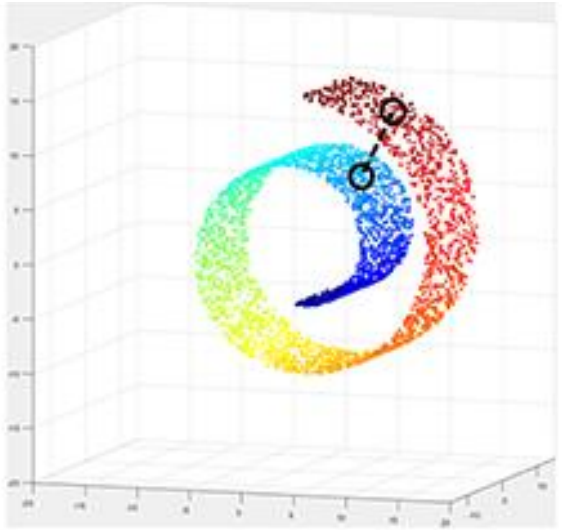
Unsupervised – Continuous: **Dimensionality Reduction**

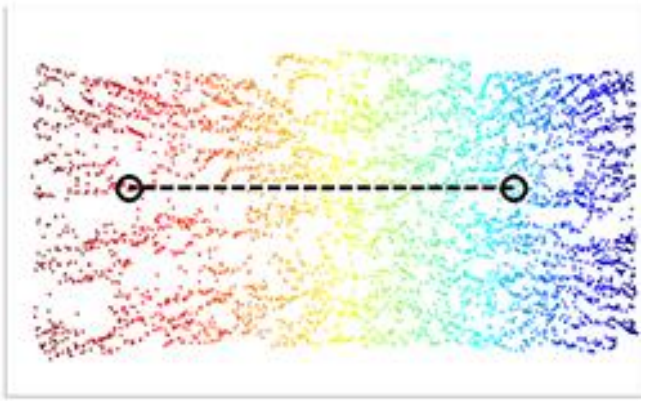
- Dimensionality reduction (Dimension reduction)

— It is the transformation of data from a high-dimensional space into a low-dimensional space

— The low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension





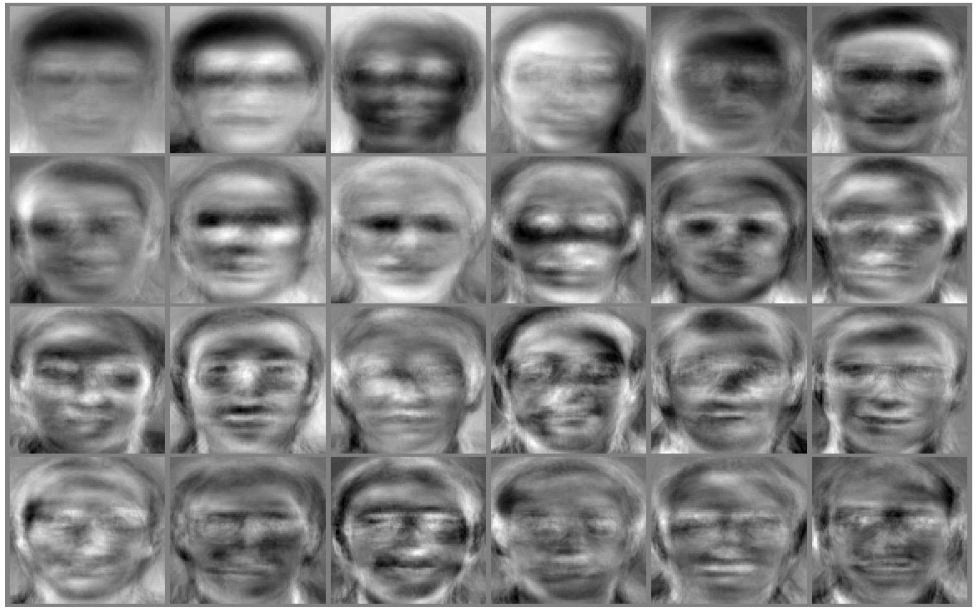


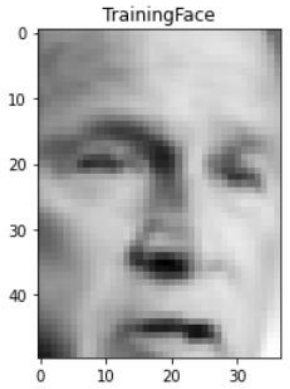
Unsupervised – Continuous: **Dimensionality Reduction**

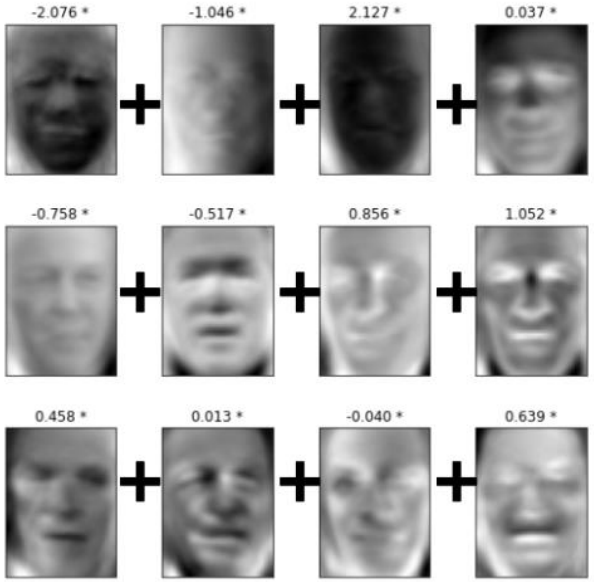
- Eigenfaces

— The eigenfaces themselves form a basis set of all images used to construct the covariance matrix

— This produces dimension reduction by allowing the smaller set of basis images to represent the original training images







ML Framework **in Supervised Learning**

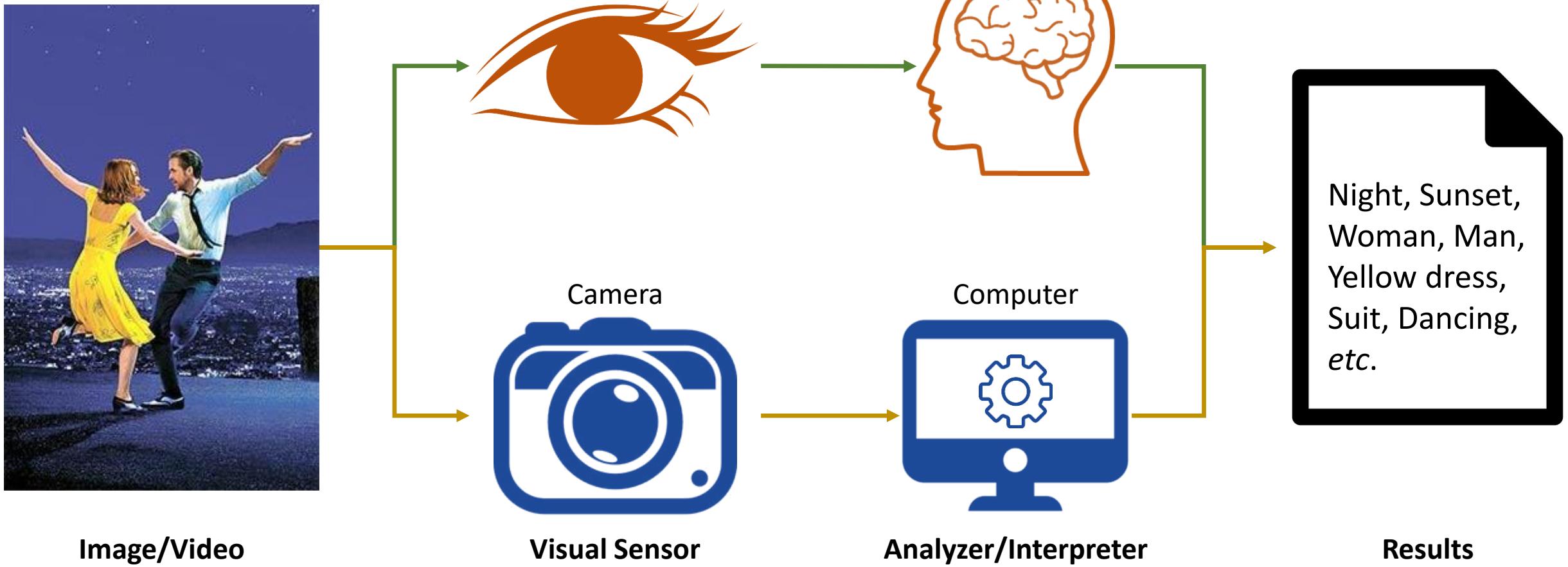
- **Training**

— Given a training set of labeled examples: $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set

- **Testing**

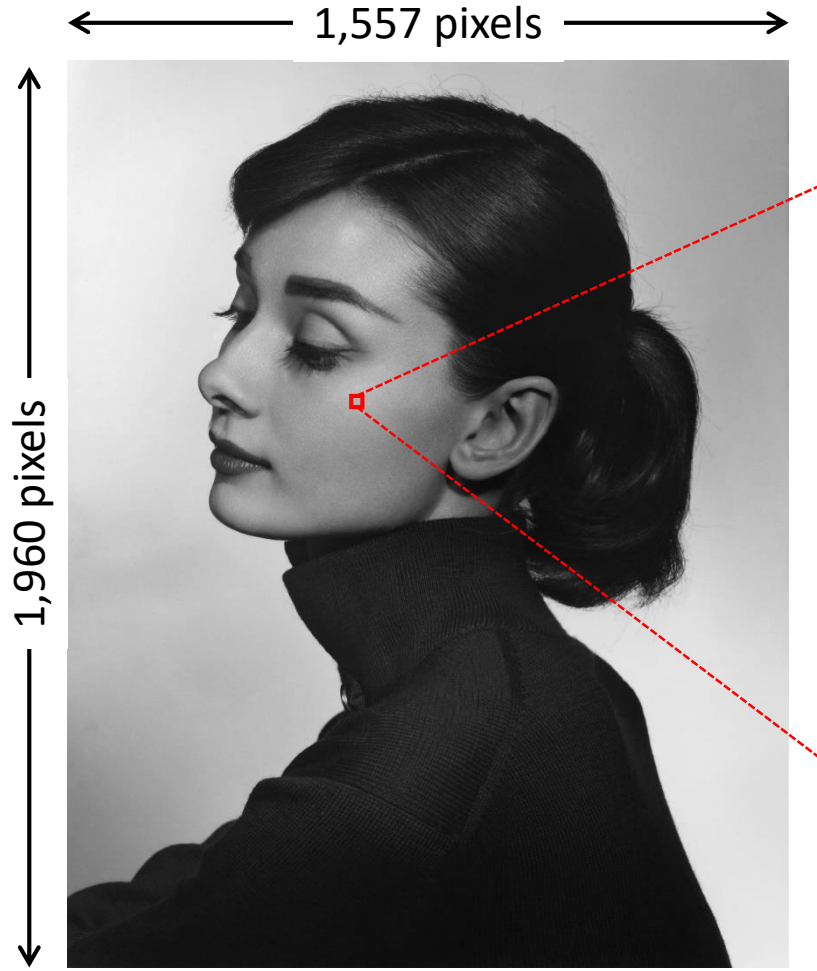
— Apply the trained prediction function f to a unseen test example $\mathbf{x}_{test}$ and output the predicted value $\hat{y}_{test} = f(\mathbf{x}_{test})$ to classify $\mathbf{x}_{test}$

Human Vision vs. Computer Vision



Goal of Computer Vision

- To bridge the gap between “Pixels” and “Meaning”

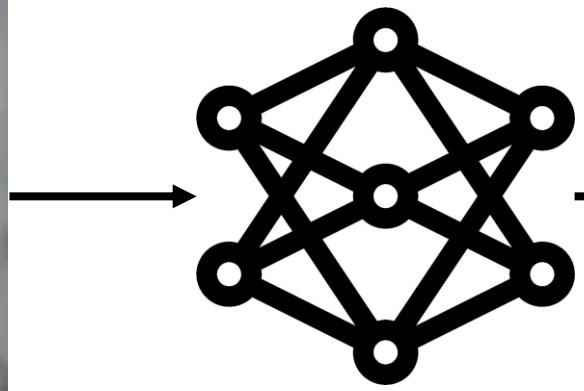


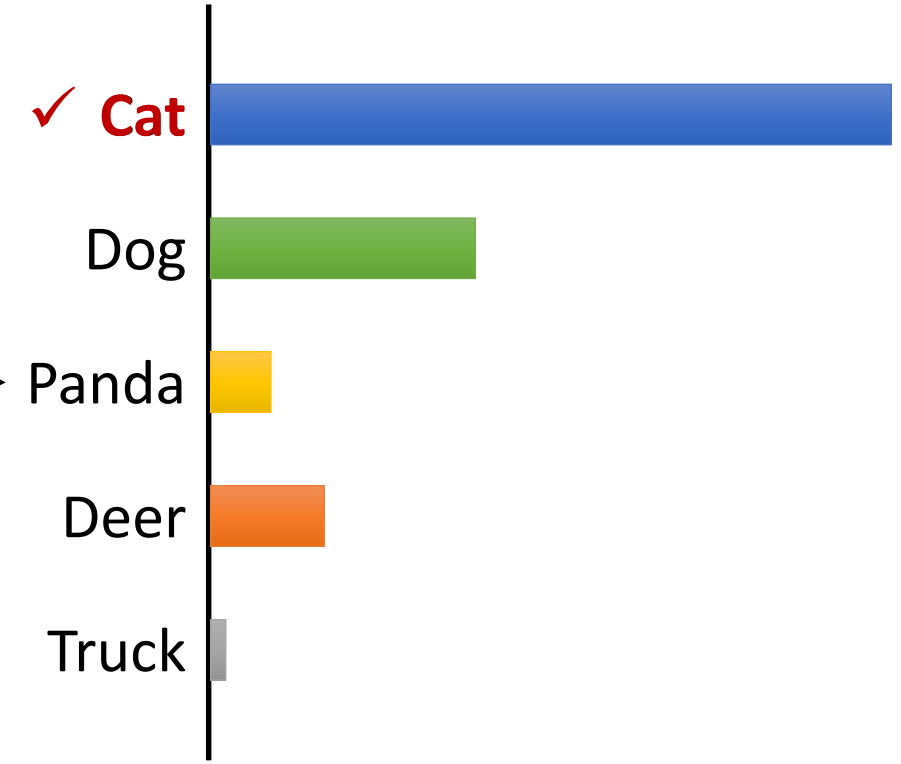
What we see

161	165	167	168	169	172	172	170
165	170	173	172	172	174	175	173
165	171	174	171	168	169	171	172
167	173	181	175	169	168	172	174
174	179	181	175	169	168	172	174
176	180	179	173	168	164	166	165
173	175	172	166	163	164	166	165
172	172	168	163	160	163	164	162

What a computer sees

Image Classification in Computer Vision



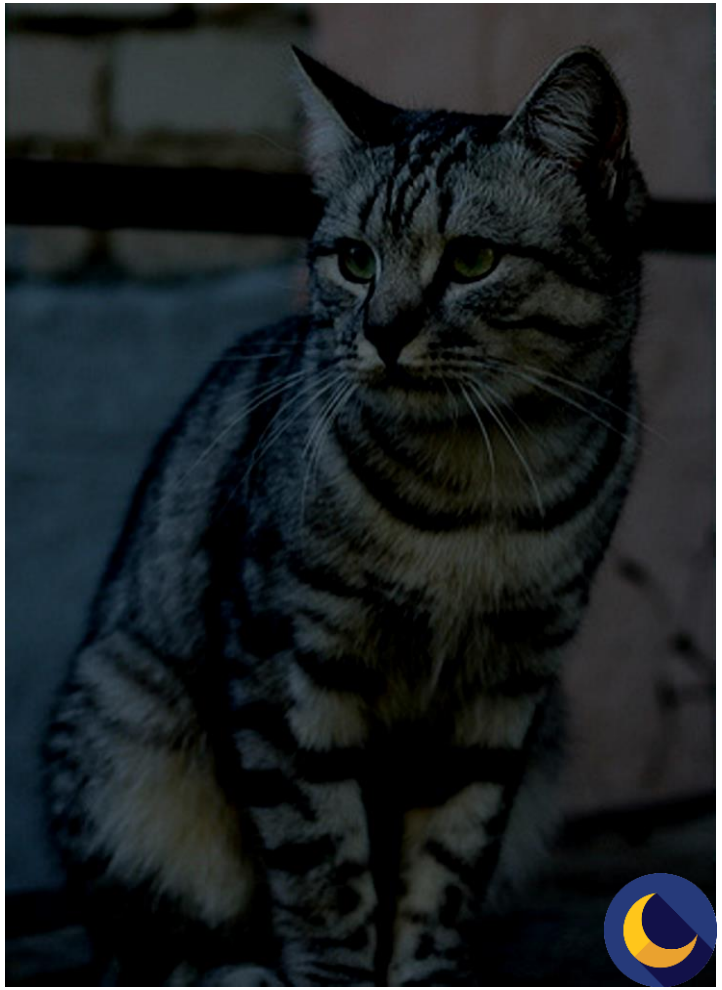


Input: Image

Output: Category (Object class)

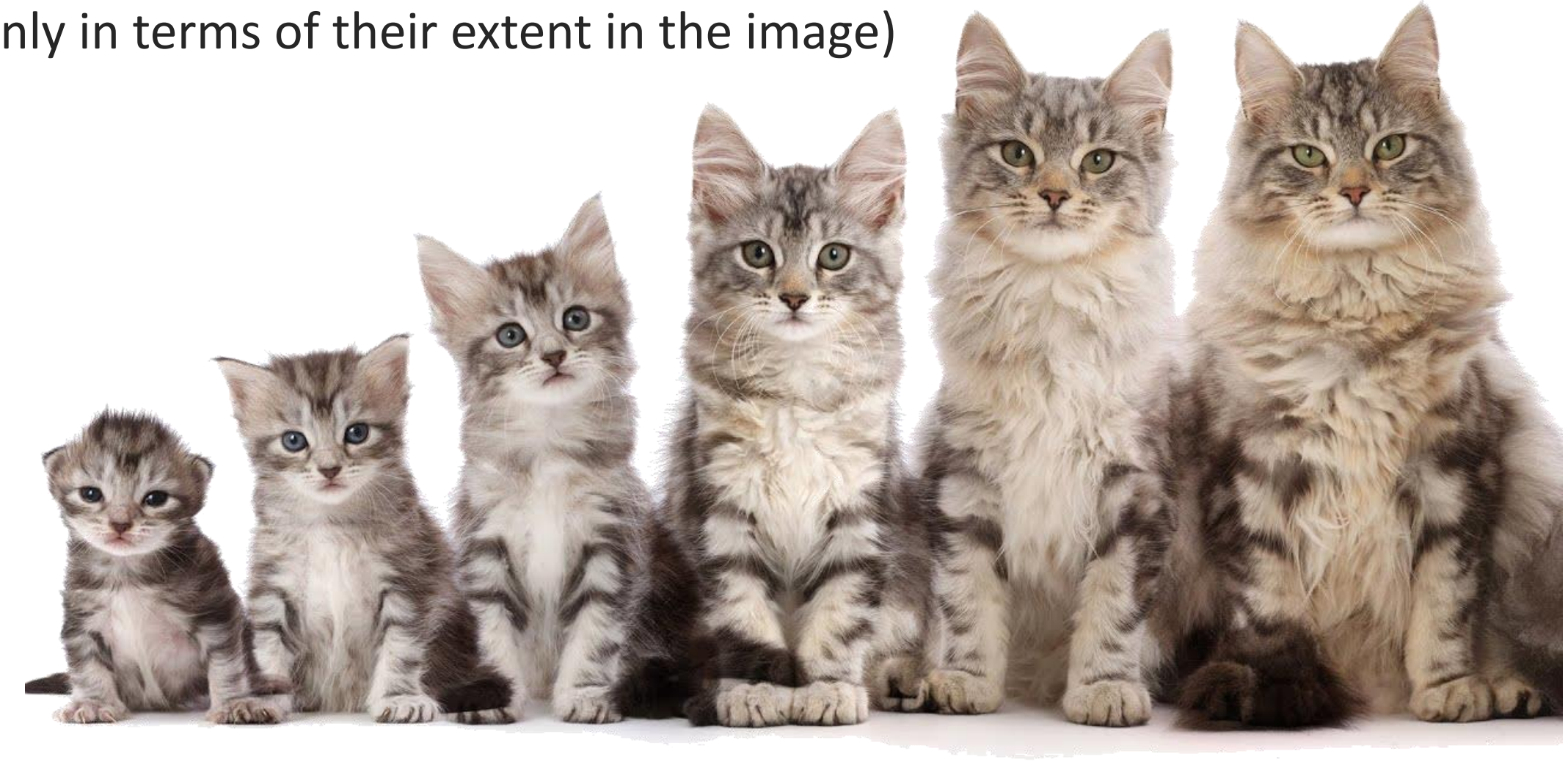
Challenges: **Illumination Variation**

- The effects of illumination are drastic on the pixel level



Challenges: **Scale Variation**

- Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image)

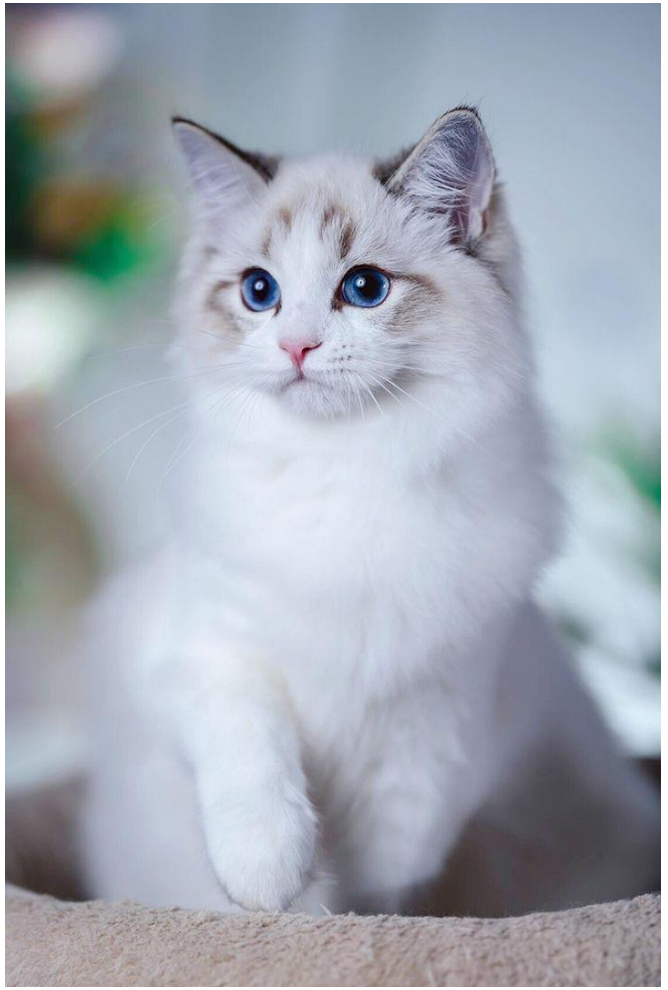


Challenges: **Intra-class Variation**

- The classes of interest can often be relatively broad, such as *cat*
- There are many different types of these objects, each with their own appearance

Challenges: **Fine-grained Categories**

- It is difficult to differentiate between hard-to distinguish object classes



Challenges: **View-point Variation**

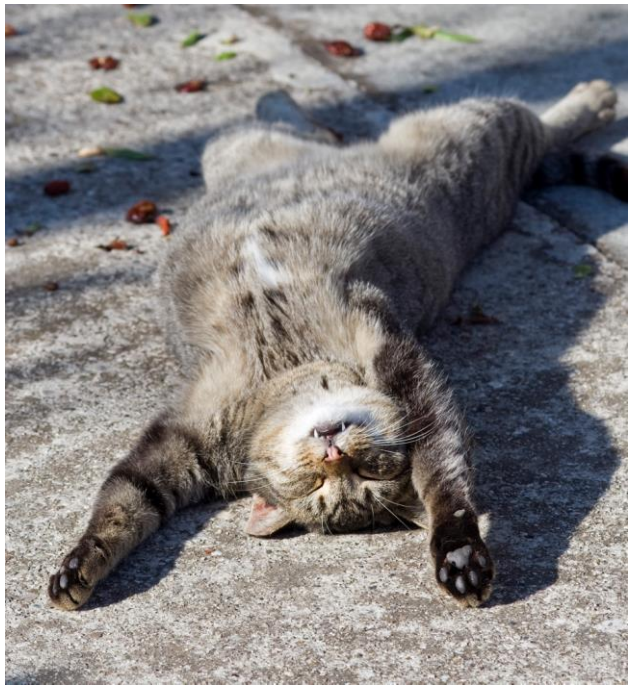
- A single instance of an object can be oriented in many ways with respect to the camera

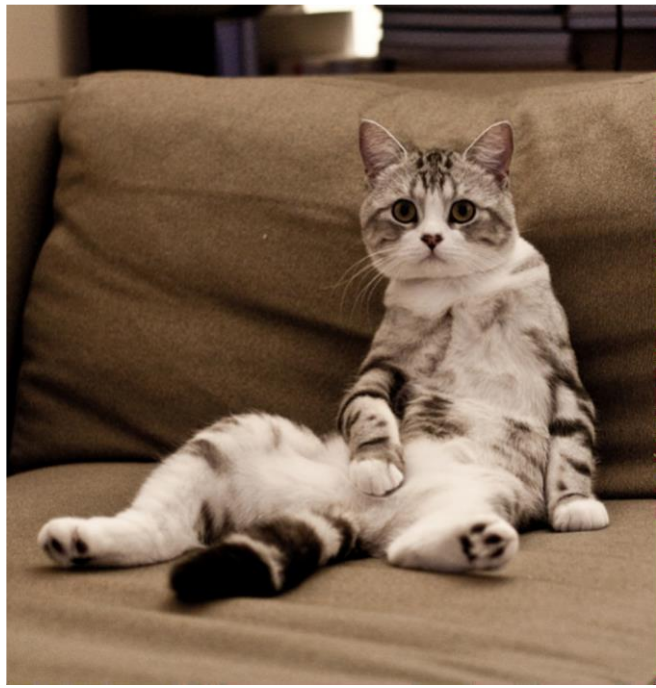
Challenges: **Occlusion**

- The objects of interest can be occluded
- Sometimes only a small portion of an object (as little as few pixels) could be visible

Challenges: **Deformation**

- Many objects of interest are not rigid bodies and can be deformed in extreme ways



Challenges: **Background Clutter**

- The objects of interest may blend into their environment, making them hard to identify

Approach: **Handcraft Feature-based**

Feature Extraction

Color features

Edges

Corner

Classification

Approach: **Data-driven (Machine Learning)**

- **Machine Learning Approach:**

① Collect a large-scale dataset including images and the corresponding labels

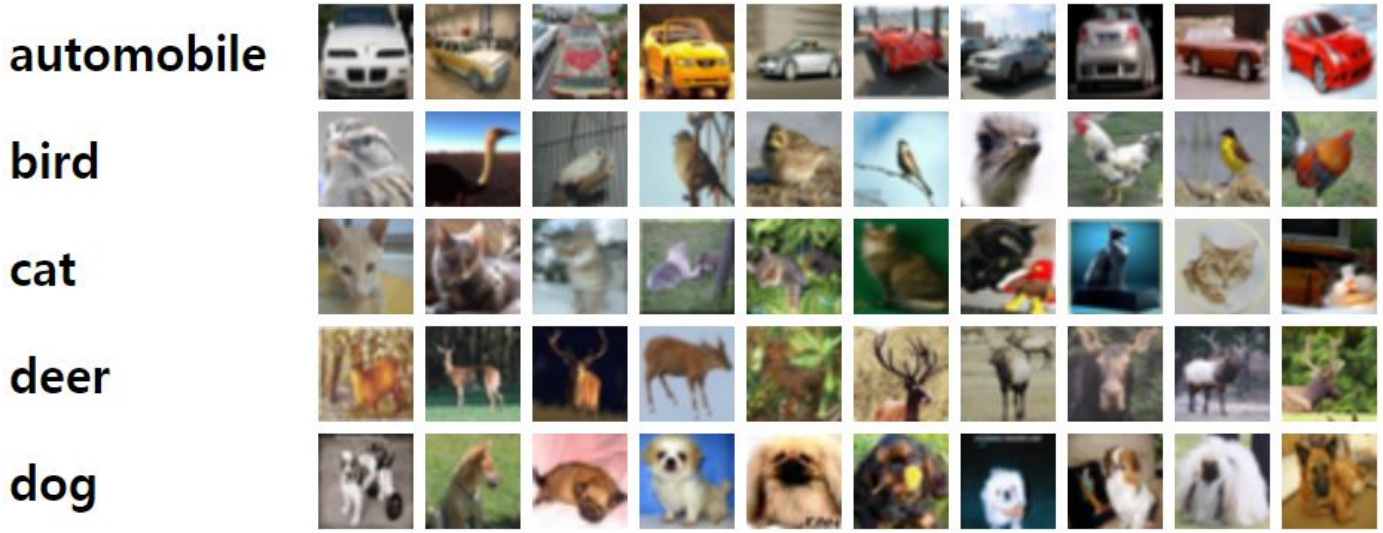
② Train a classifier with the training dataset by minimizing the loss

③ Evaluate the performance (e.g., classification accuracy) of the trained classifier on the test dataset

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

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

Input & output for training and evaluation



Example image classification dataset: CIFAR-10

Image Classification Datasets: **MNIST**

— **Class:** 10 (0 to 9 digits)

— **# of Training:** 50,000 (50K)

— **# of Testing:** 10,000 (10K)

— **Image Format:** 28x28 grayscale images

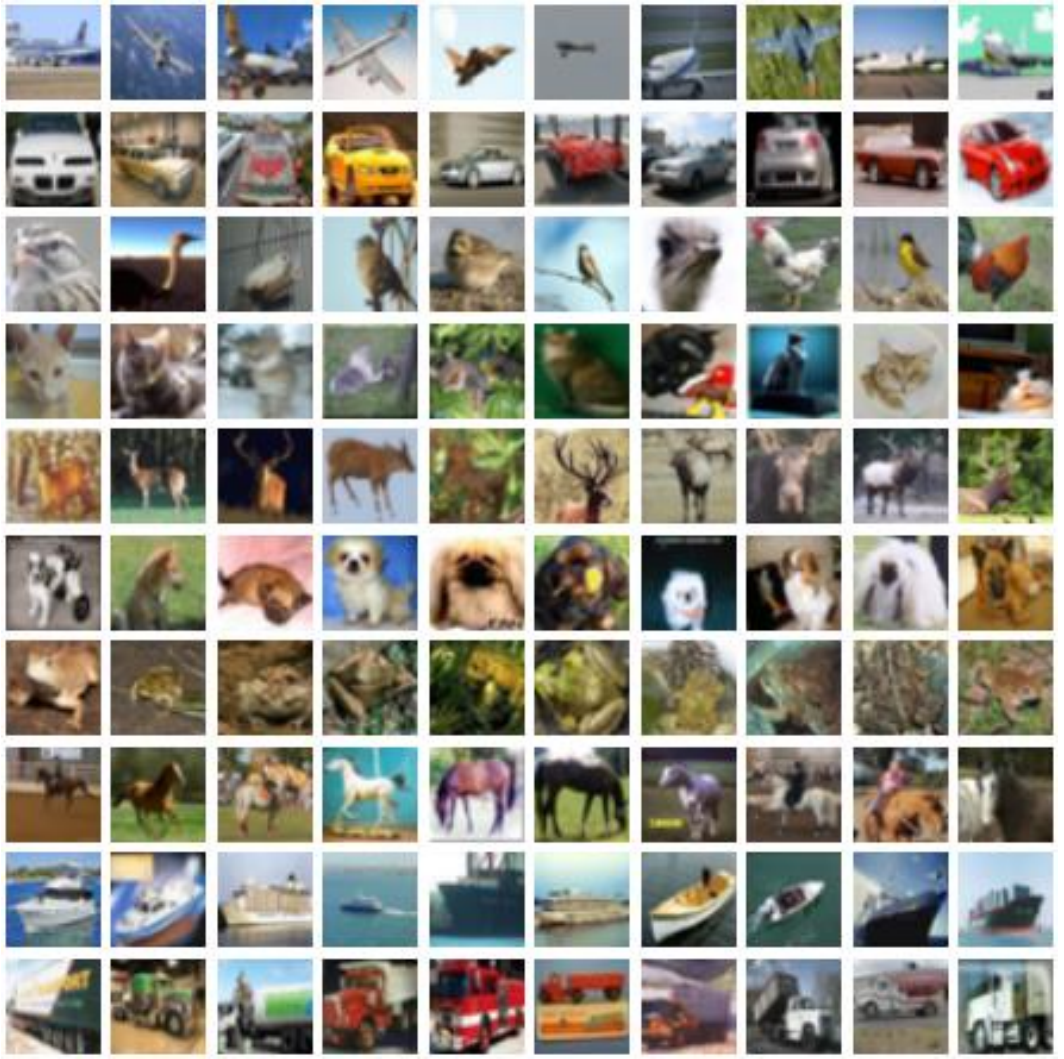
Image Classification Datasets: **CIFAR-10**

— **Class:** 10 (airplane; automobile; bird; cat; deer; dog; frog; horse; ship; truck)

— **# of Training:** 50,000 (5K / class)

— **# of Testing:** 10,000 (1K / class)

— **Image Format:** 32x32 RGB images



https://www.cs.toronto.edu/~kriz/cifar.html

Image Classification Datasets: **CIFAR-100**

— **Class:** 100 (20 superclasses with 5 classes)
  - Aquatic mammals – beaver; dolphin; otter; seal
  - Trees – Maple; oak; palm; pine; willow

— **# of Training:** 50,000 (500 / class)

— **# of Testing:** 10,000 (100 / class)

— **Image Format:** 32x32 RGB images

https://www.cs.toronto.edu/~kriz/cifar.html

Image Classification Datasets: **ImageNet**

— **Class:** 1000

— **# of Training:** ~1,300,000 (~1.3K / class)

— **# of Testing:** 100,000 (100 / class)

— **# of Validation:** 50,000 (50 / class)

— **Image Format:** ~469x387 RGB

- Crop or resize to 256x256 or 224x224

Image Classification Datasets: **Places365**

— **Class:** 365 (365 difference scenes)

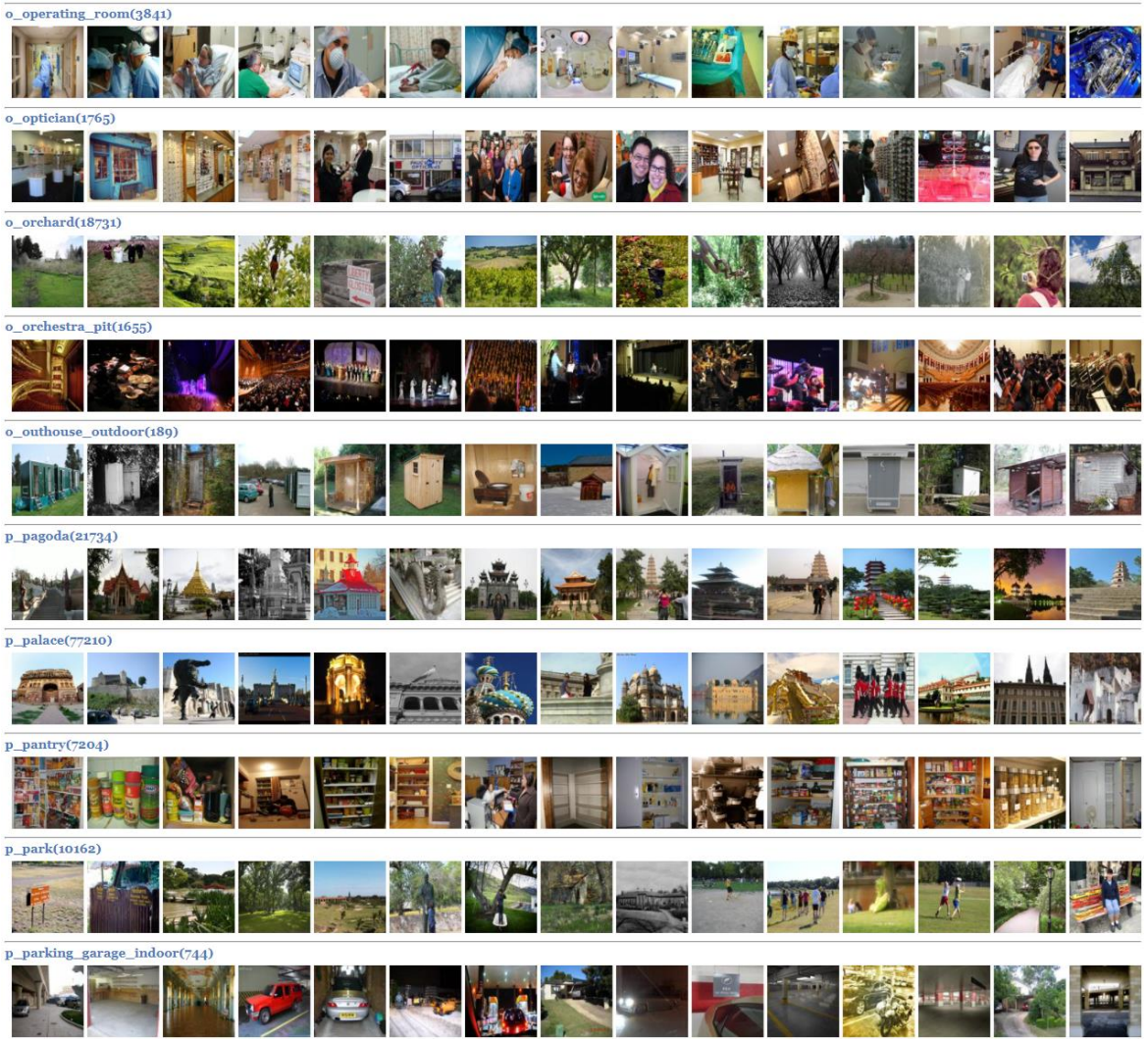
— **# of Training:** ~8,000,000 (8M)

— **# of Testing:** 328,500 (900 / class)

— **# of Validation:** 18,250 (50 / class)

— **Image Format:** variable image size

- Resize to 256x256 for training



Classifier: Nearest Neighbor

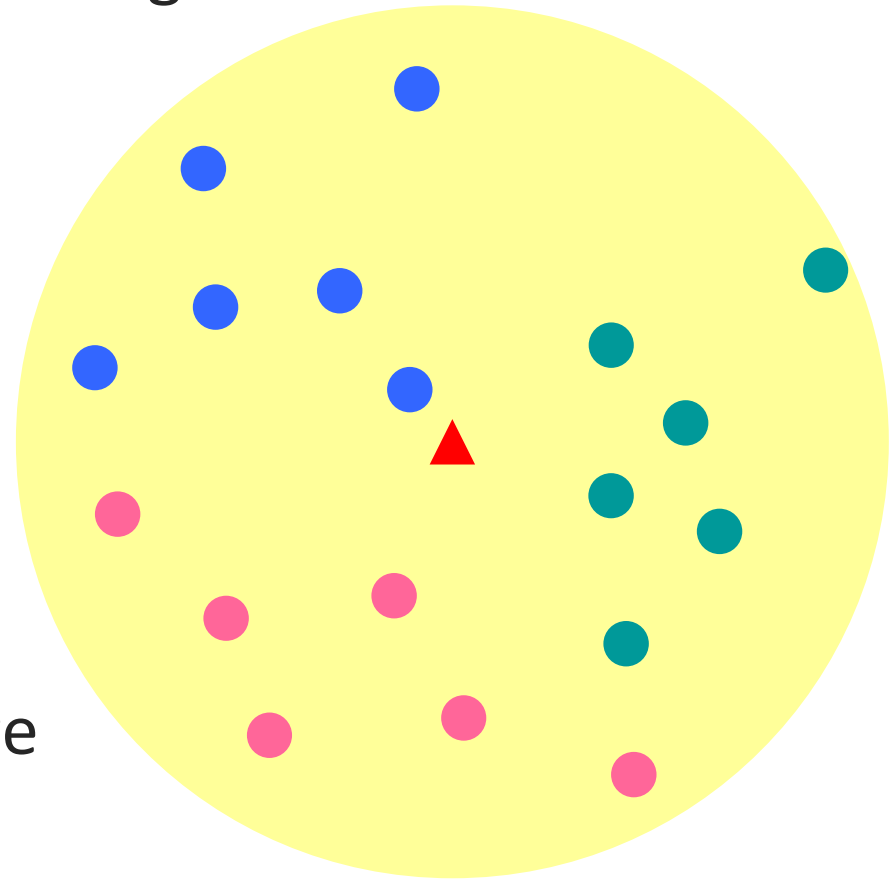
- It is the optimization problem of finding the point in a given set that is closest (or most similar) to a given point

- **Training Stage:**

— Memorize all data and labels

- **Testing Stage:**

— Predict the label of the most similar training image



Classifier: Nearest Neighbor

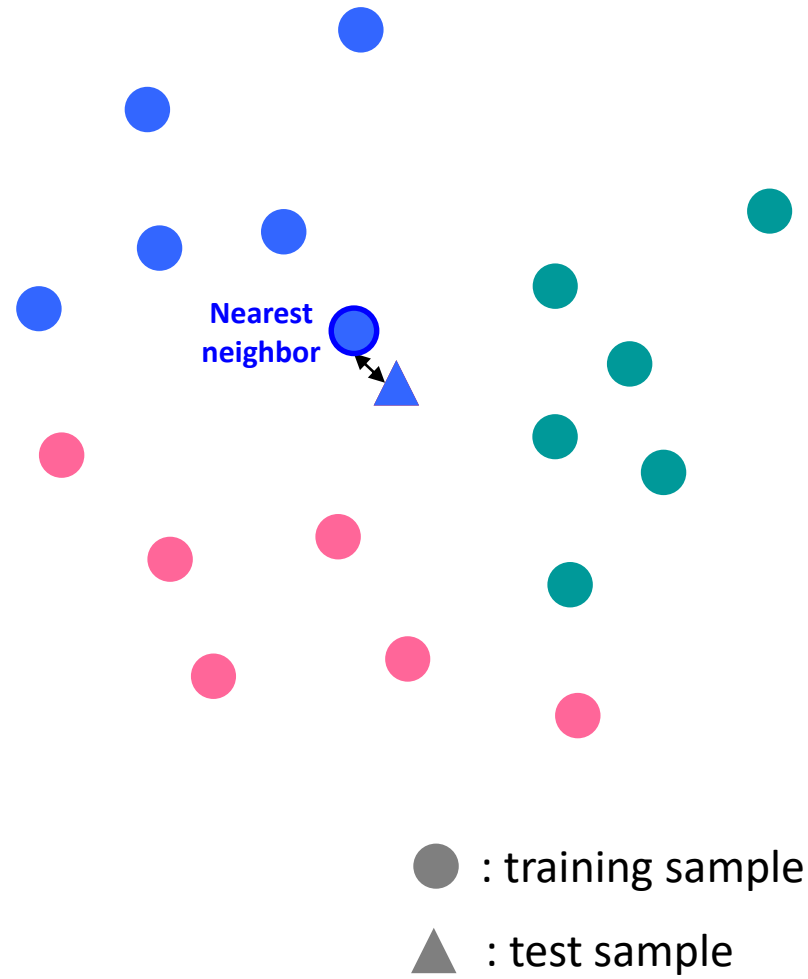
- It is the optimization problem of finding the point in a given set that is closest (or most similar) to a given point

- **Training Stage:**

— Memorize all data and labels

- **Testing Stage:**

— Predict the label of the most similar training image



Distance Metric in Nearest Neighbor

- **L_1 distance (ℓ_1 norm; Manhattan distance):** The distance between two points is **the sum of the absolute differences** of their Cartesian coordinates

$$d_1(I_{test}, I_{train}) = \sum_{i=1}^{W} \sum_{j=1}^{H} |I_{test}(i,j) - I_{train}(i,j)|$$

$$\left| \begin{array}{|c|c|c|c|} \hline 56 & 32 & 10 & 18 \\ \hline 90 & 23 & 128 & 133 \\ \hline 24 & 26 & 178 & 200 \\ \hline 2 & 0 & 255 & 220 \\ \hline \end{array} \right. - \left. \begin{array}{|c|c|c|c|} \hline 10 & 20 & 24 & 17 \\ \hline 8 & 10 & 89 & 100 \\ \hline 12 & 16 & 178 & 170 \\ \hline 4 & 32 & 233 & 112 \\ \hline \end{array} \right| = \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

Test image, I_{test} Training image, I_{train}

Distance Metric in Nearest Neighbor

- **L_1 distance (ℓ_1 norm; Manhattan distance):** The distance between two points is **the sum of the absolute differences** of their Cartesian coordinates

$$d_1(I_{test}, I_{train}) = \sum_{i=1}^{W} \sum_{j=1}^{H} |I_{test}(i,j) - I_{train}(i,j)|$$

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

Test image, I_{test}

$-$

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

Training image, I_{train}

$=$

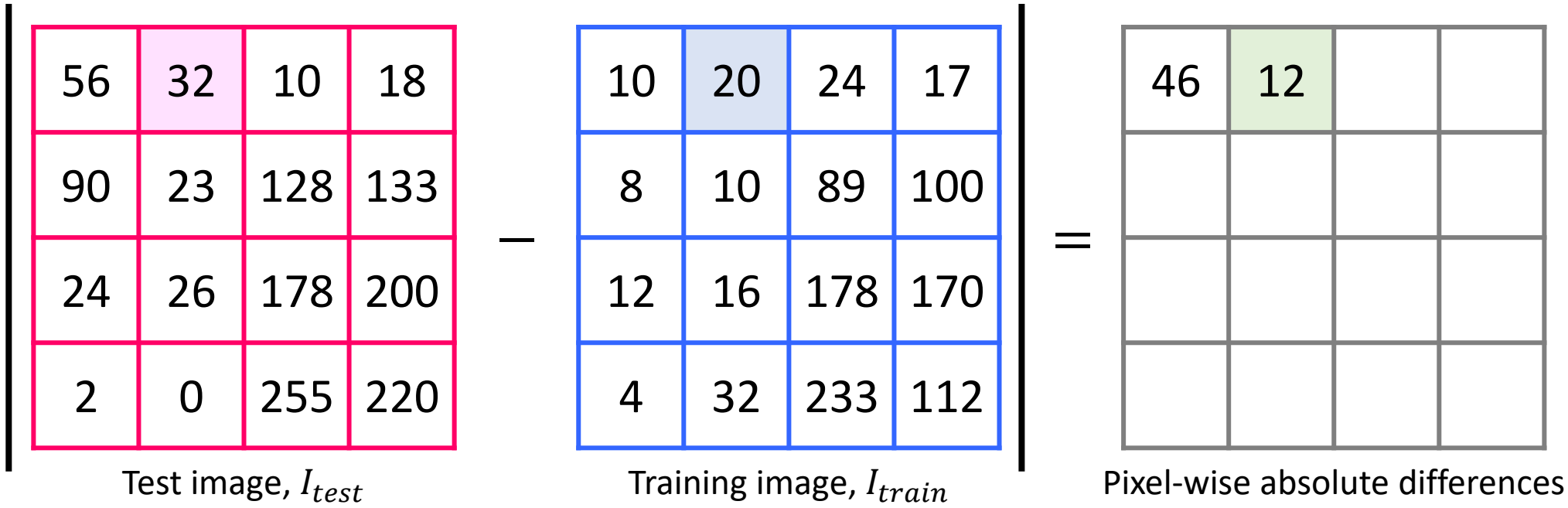
46			

Pixel-wise absolute differences

Distance Metric in Nearest Neighbor

- **L_1 distance (ℓ_1 norm; Manhattan distance):** The distance between two points is **the sum of the absolute differences** of their Cartesian coordinates

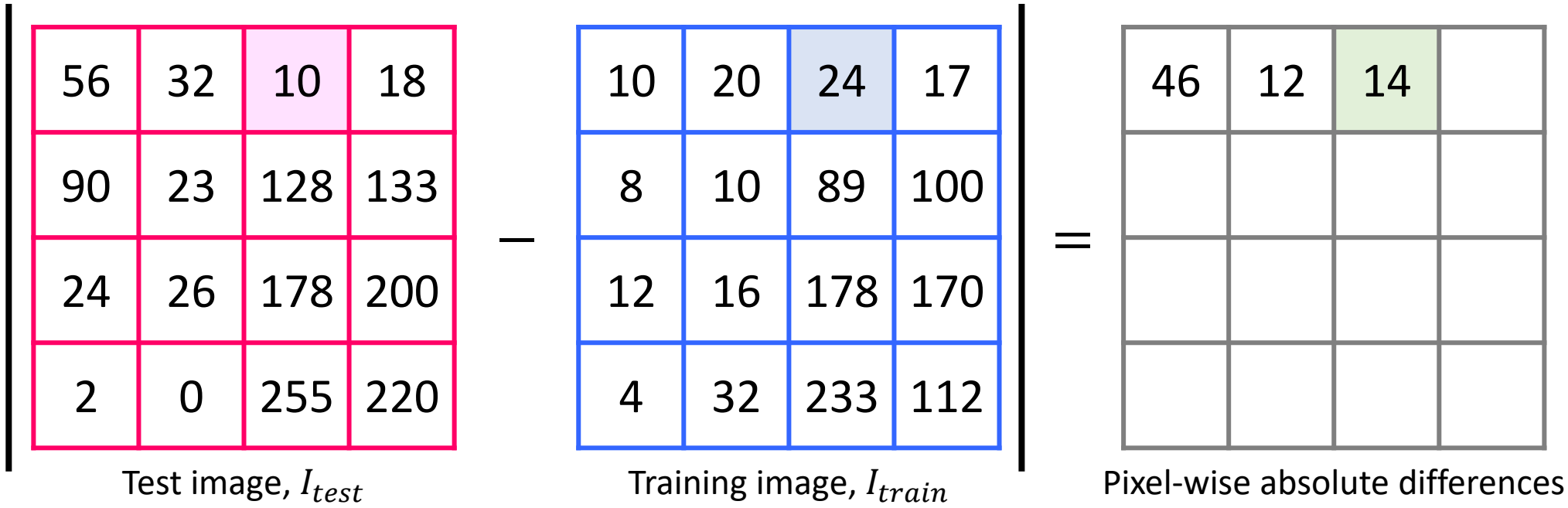
$$d_1(I_{test}, I_{train}) = \sum_{i=1}^{W} \sum_{j=1}^{H} |I_{test}(i,j) - I_{train}(i,j)|$$



Distance Metric in Nearest Neighbor

- **L_1 distance (ℓ_1 norm; Manhattan distance):** The distance between two points is **the sum of the absolute differences** of their Cartesian coordinates

$$d_1(I_{test}, I_{train}) = \sum_{i=1}^{W} \sum_{j=1}^{H} |I_{test}(i,j) - I_{train}(i,j)|$$



Test image, I_{test} Training image, I_{train} Pixel-wise absolute differences

Distance Metric in Nearest Neighbor

- **L_1 distance (ℓ_1 norm; Manhattan distance):** The distance between two points is **the sum of the absolute differences** of their Cartesian coordinates

$$d_1(I_{test}, I_{train}) = \sum_{i=1}^{W} \sum_{j=1}^{H} |I_{test}(i,j) - I_{train}(i,j)|$$

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

Test image, I_{test}

$-$

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

Training image, I_{train}

$=$

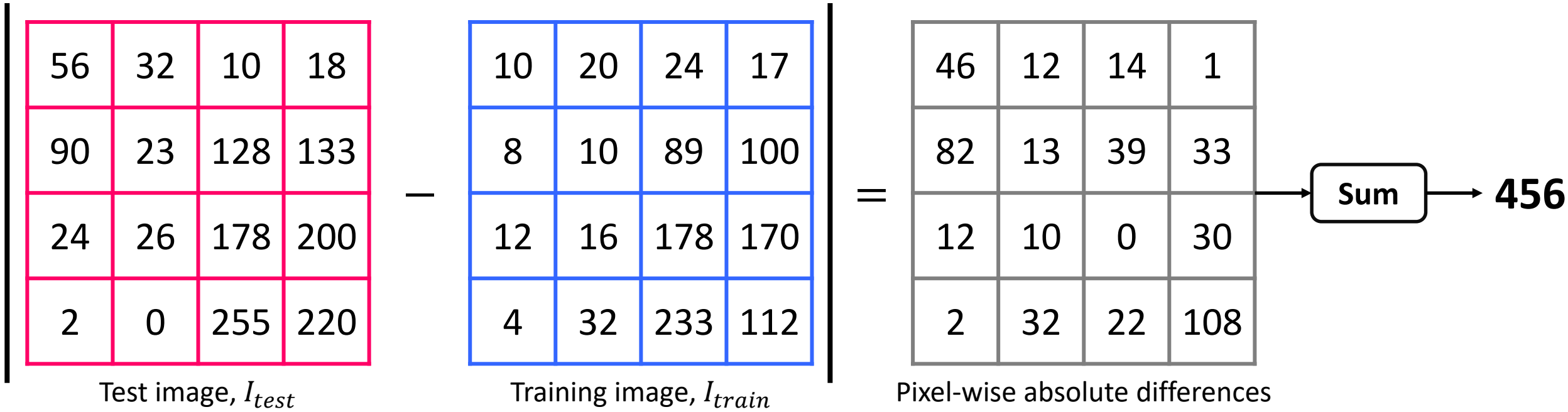
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

Pixel-wise absolute differences

Distance Metric in Nearest Neighbor

- **L_1 distance (ℓ_1 norm; Manhattan distance):** The distance between two points is **the sum of the absolute differences** of their Cartesian coordinates

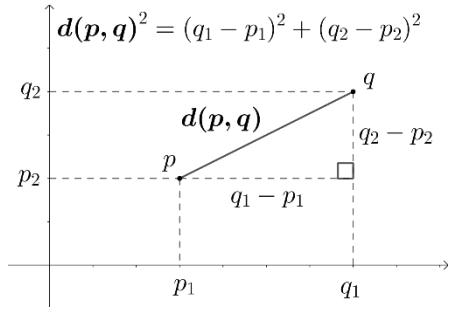
$$d_1(I_{test}, I_{train}) = \sum_{i=1}^{W}\sum_{j=1}^{H} |I_{test}(i,j) - I_{train}(i,j)|$$



Distance Metric in Nearest Neighbor

- **L_2 distance (ℓ_2 norm; Euclidean distance):** The distance between two points is **the length of a line segment** between the two points

$$d_2(I_{test}, I_{train}) = \sum_{i=1}^{W}\sum_{j=1}^{H}\sqrt{\left(I_{test}(i,j) - I_{train}(i,j)\right)^2}$$



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

$$\left[\begin{array}{cccc} 56 & 32 & 10 & 18 \\ 90 & 23 & 128 & 133 \\ 24 & 26 & 178 & 200 \\ 2 & 0 & 255 & 220 \end{array}\right] - \left[\begin{array}{cccc} 10 & 20 & 24 & 17 \\ 8 & 10 & 89 & 100 \\ 12 & 16 & 178 & 170 \\ 4 & 32 & 233 & 112 \end{array}\right]^2 = \left[\begin{array}{cccc} 46^2 & 12^2 & 14^2 & 1^2 \\ 82^2 & 13^2 & 39^2 & 33^2 \\ 12^2 & 10^2 & 0^2 & 30^2 \\ 2^2 & 32^2 & 22^2 & 108^2 \end{array}\right] \rightarrow \boxed{\text{Sum \& Root}} \rightarrow \mathbf{162.1}$$

Test image, I_{test} — Training image, I_{train} — Pixel-wise absolute differences

Example Code: Nearest Neighbor Classifier

- **Training Stage:**

— Memorize all data and labels

— Time: $\mathcal{O}(1)$ (with N training samples)

```python
import numpy as np

class NearestNeighbor(object):
  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 range(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
```

Training

Example Code: Nearest Neighbor Classifier

- **Training Stage:**

— Memorize all data and labels

— Time: $\mathcal{O}(1)$ (with N training samples)

- **Testing Stage:**

— Predict the label of the most similar training image

— Time: $\mathcal{O}(N)$ (with N training samples)

  - With 1.3M images (ImageNet dataset), how fast is testing?

```python
import numpy as np

class NearestNeighbor(object):
  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
```

Training

```python
  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 range(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
```

Testing

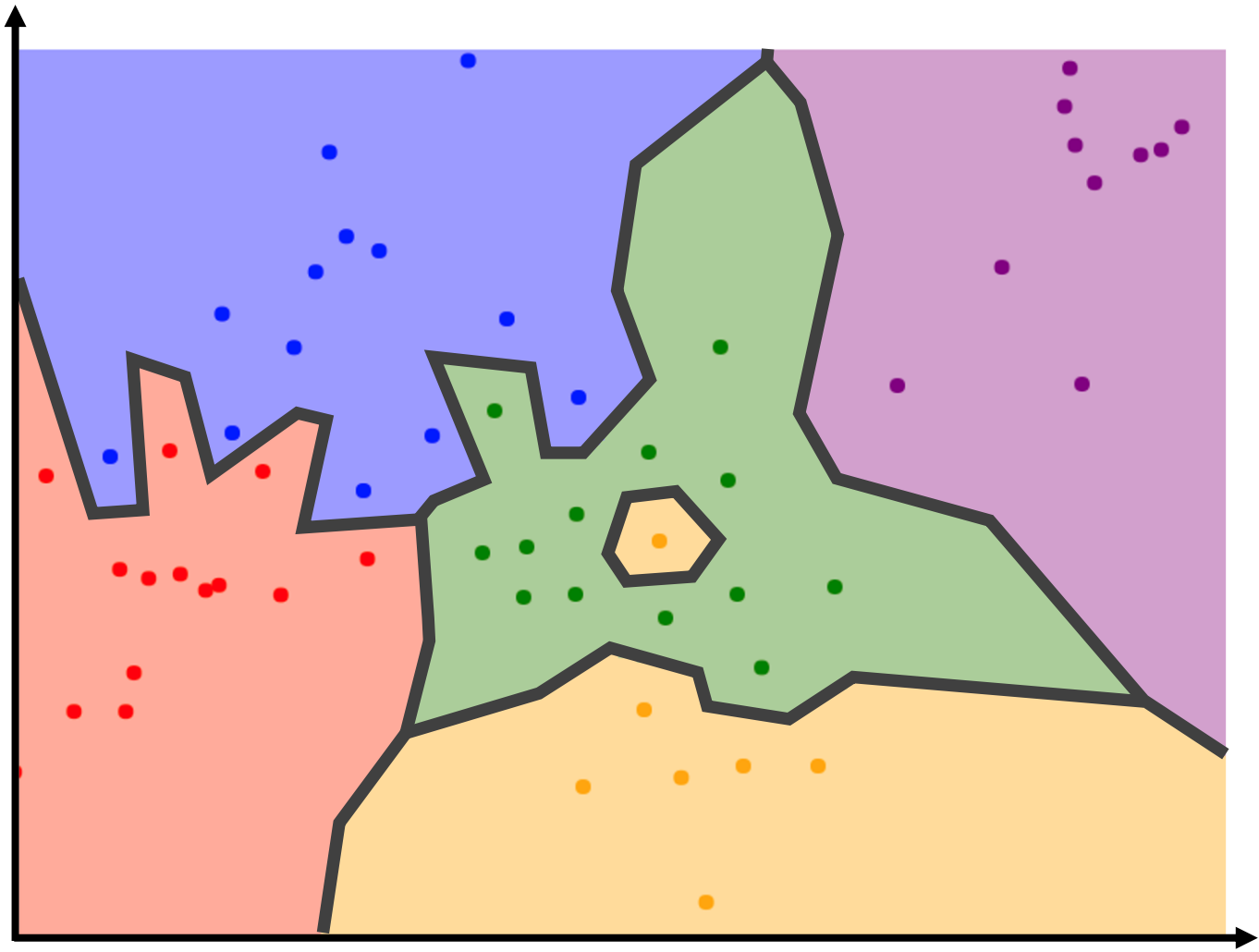
Nearest Neighbor: **Decision Boundaries**

— **Points:** Training samples

— **Colors:** Training labels (classes)

— **Background colors:** The category a test sample would be assigned

— **Dark gray lines:** The decision boundary between two classification regions



An example of the Nearest Neighbor (NN) classifier using 2-dimensional points and 5 classes

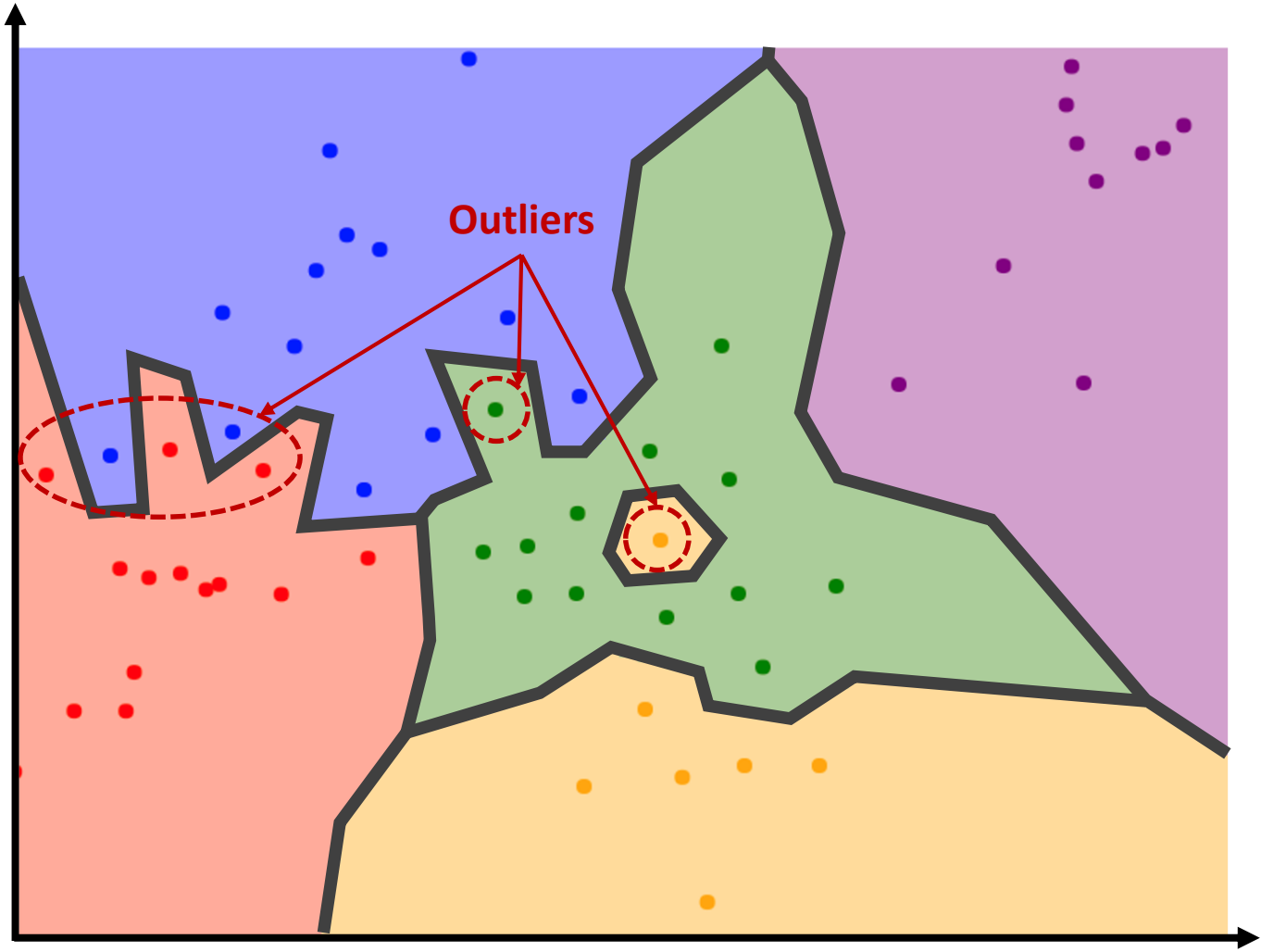
Nearest Neighbor: **Decision Boundaries**

— **Outliers:** A data point that **differs significantly from other** observations

- Outliers in the training dataset make **the decision boundary noisy**

- How to smooth out decision boundaries?

— Use more neighbors



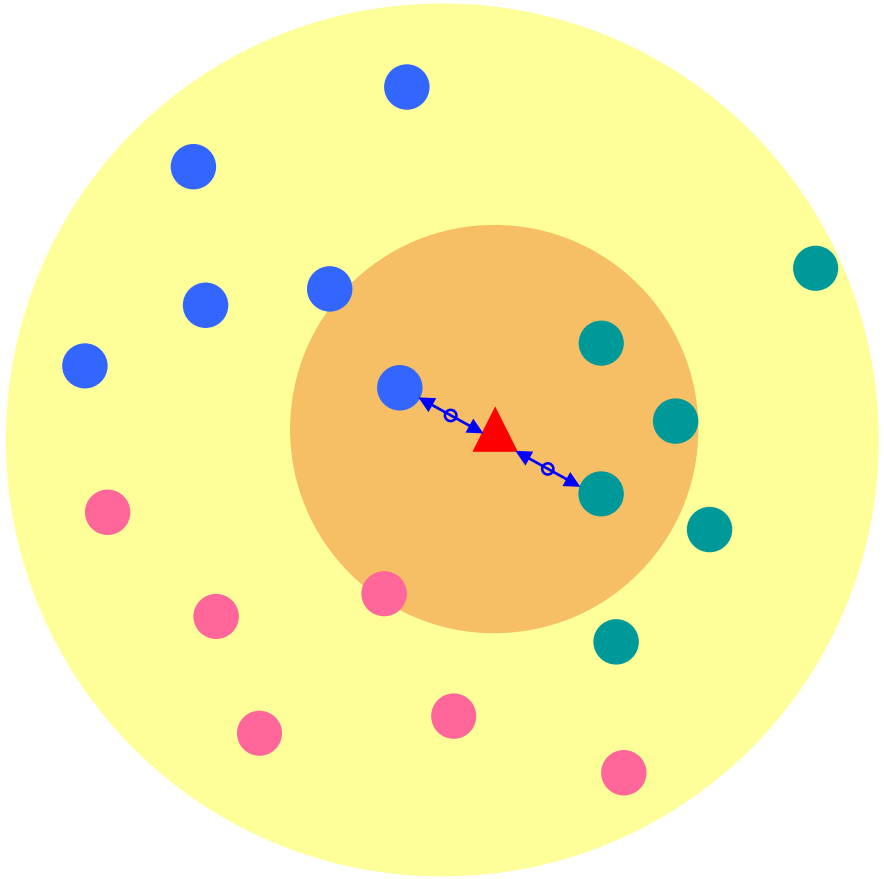
An example of the Nearest Neighbor (NN) classifier using 2-dimensional points and 5 classes

K-Nearest Neighbors (K-NN)

- Instead of finding the single closest image in the training set, we will **find the top k closest images**, and have them vote on the label of the test image

— NN method is the K-NN method where $K = 1$

- **Higher values of k** have a smoothing effect that makes **the classifier more resistant** to outliers

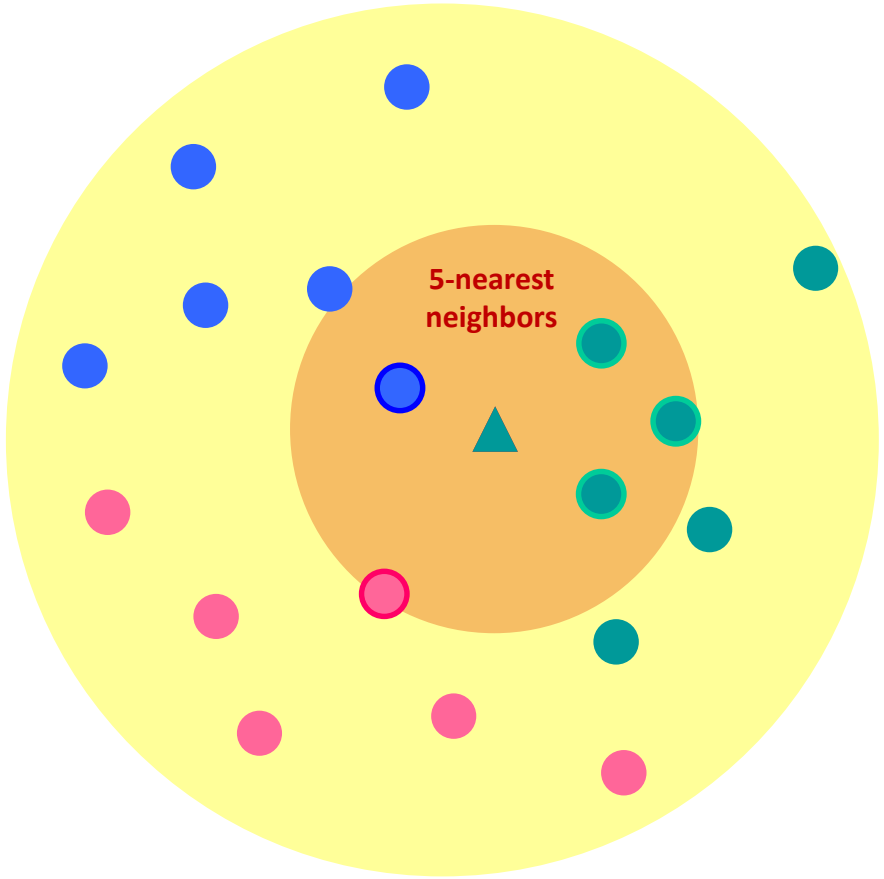


K-Nearest Neighbors (K-NN)

- Instead of finding the single closest image in the training set, we will **find the top k closest images**, and have them vote on the label of the test image

— NN method is the K-NN method where $K = 1$

- **Higher values of k** have a smoothing effect that makes **the classifier more resistant** to outliers

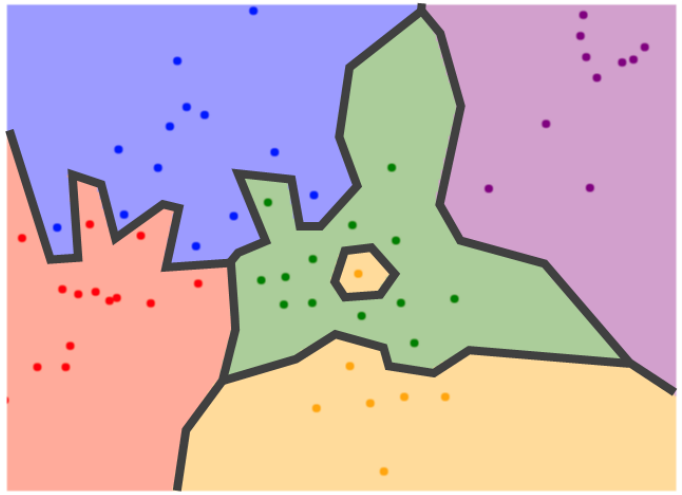


K-Nearest Neighbors (K-NN)

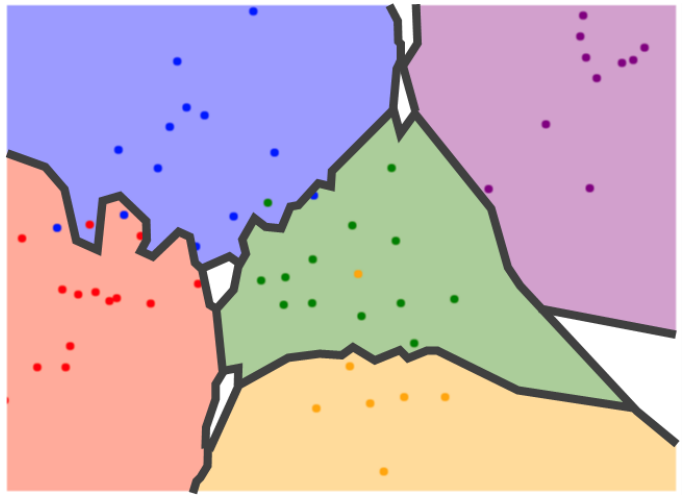
- Instead of finding the single closest image in the training set, we will **find the top k closest images**, and have them vote on the label of the test image

— NN method is the K-NN method where $K = 1$

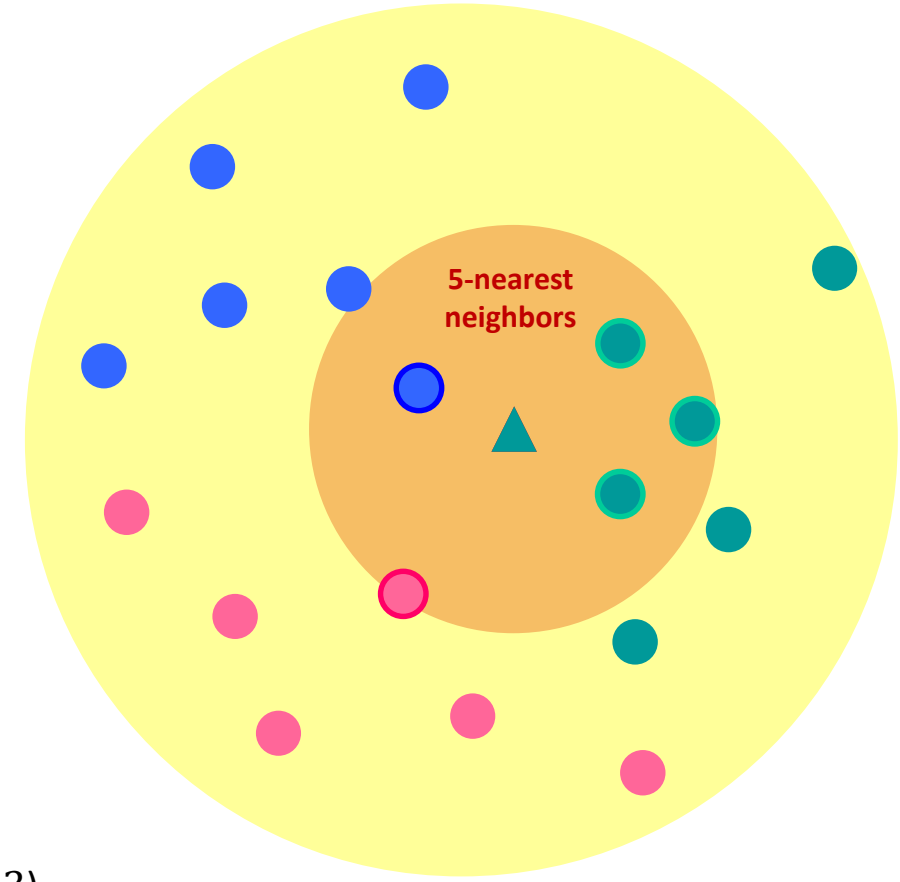
- **Higher values of k** have a smoothing effect that makes **the classifier more resistant** to outliers



Decision boundaries of NN classifier



Decision boundaries of K-NN classifier ($K = 3$)



Validation Set for Hyperparameter Tuning

- A hyperparameter is a parameter whose value is used **to control the learning process**
- It is often not obvious what values or settings one should choose

- In K-NN Classification:

— What is the best value of $\boldsymbol{K}$?

— What is the best **distance metric**?

- In particular, we **cannot use the test set for the purpose of tweaking hyperparameters**

Setting Hyperparameters

① Choose hyperparameters that work best **on the entire data – Bad**

- $K = 1$ always works perfectly on training data

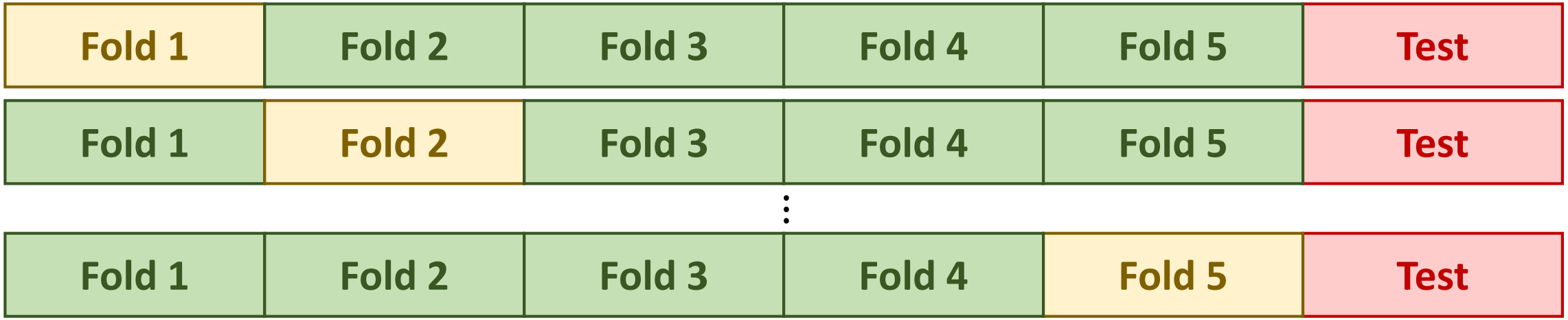
② Split data into train and test; Choose hyperparameters that work best **on test set – Bad**

- No idea how the algorithm will perform on new data (over-fit to test set)

Setting Hyperparameters

③ Split data in to **train**, **validation**, and **test**; Choose hyperparameters on validation set and evaluate them on test set – **Better**

④ **Cross-validation:** Split data into K-folds, **try each fold** as a validation set and **average** the results – Useful for small datasets



Limitations of K-Nearest Neighbor

- Distance metrics on pixel domain is not informative
- The image below illustrates the point that these three different images have **the same L_2 similarities,** but very **different from our perceptual similarities**

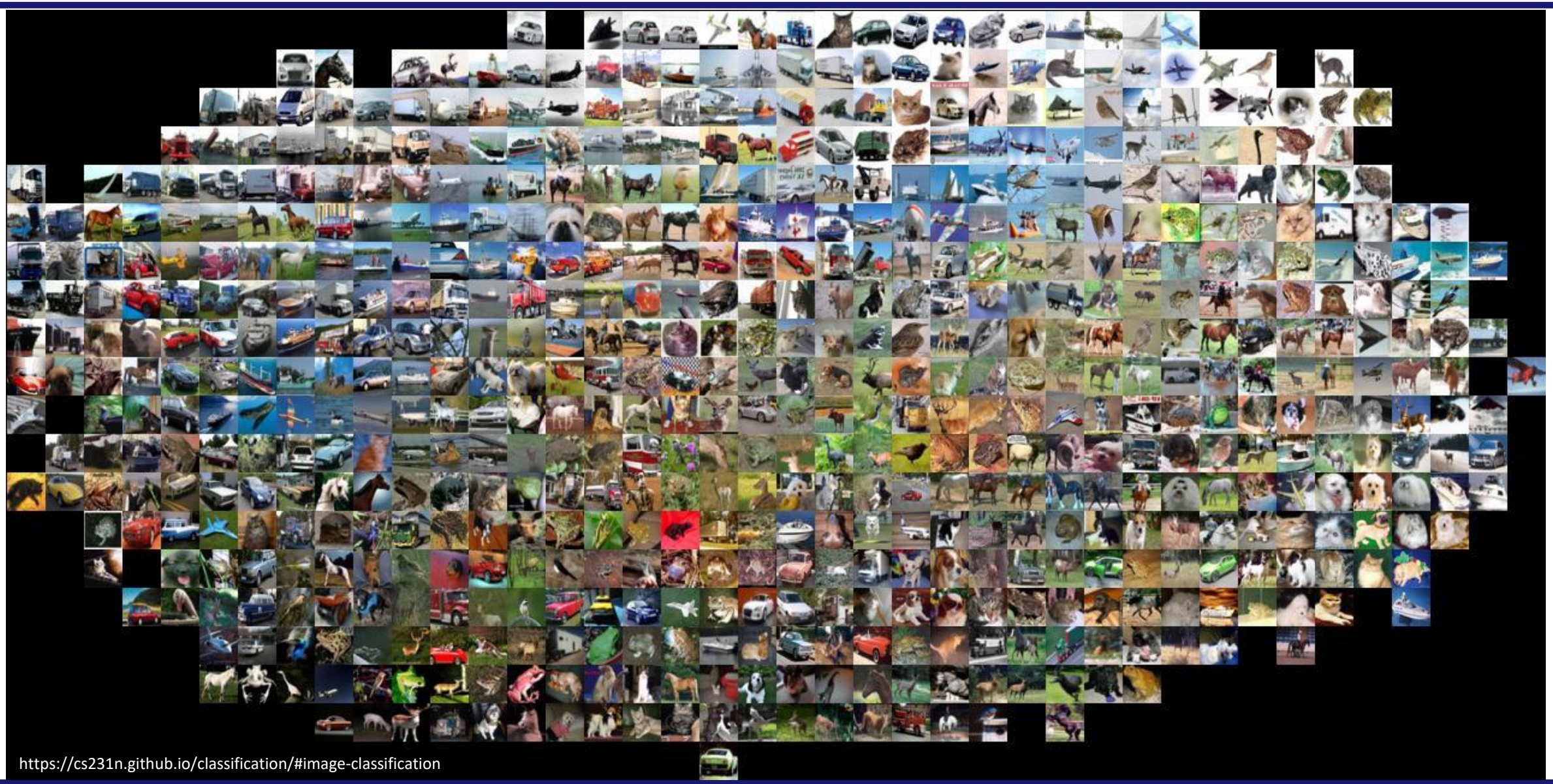
Original image

Masked image

Shifted image

Tinted image

Limitations of K-Nearest Neighbor



https://cs231n.github.io/classification/#image-classification

Summary

- **Advantages** of the Nearest Neighbor classifier:

— It is very **simple** to implement and understand

— Additionally, the classifier takes **no time to train**, since all that is required is to store and possibly index the training data

- **Disadvantages** of the Nearest Neighbor classifier:

— Computational cost is **expensive at test** time, since classifying a test example requires a **comparison to every single training example**

  - This is backwards, since in practice we often care about the test time efficiency much more than the efficiency at training time