Image Classification

Most of the materials are taken from [here]

Nastaran Okati

Outline

- Introduction
- Nearest Neighbor Classifier
- Validation Sets, Cross Validation
- Pros/Cons of Nearest Neighbor

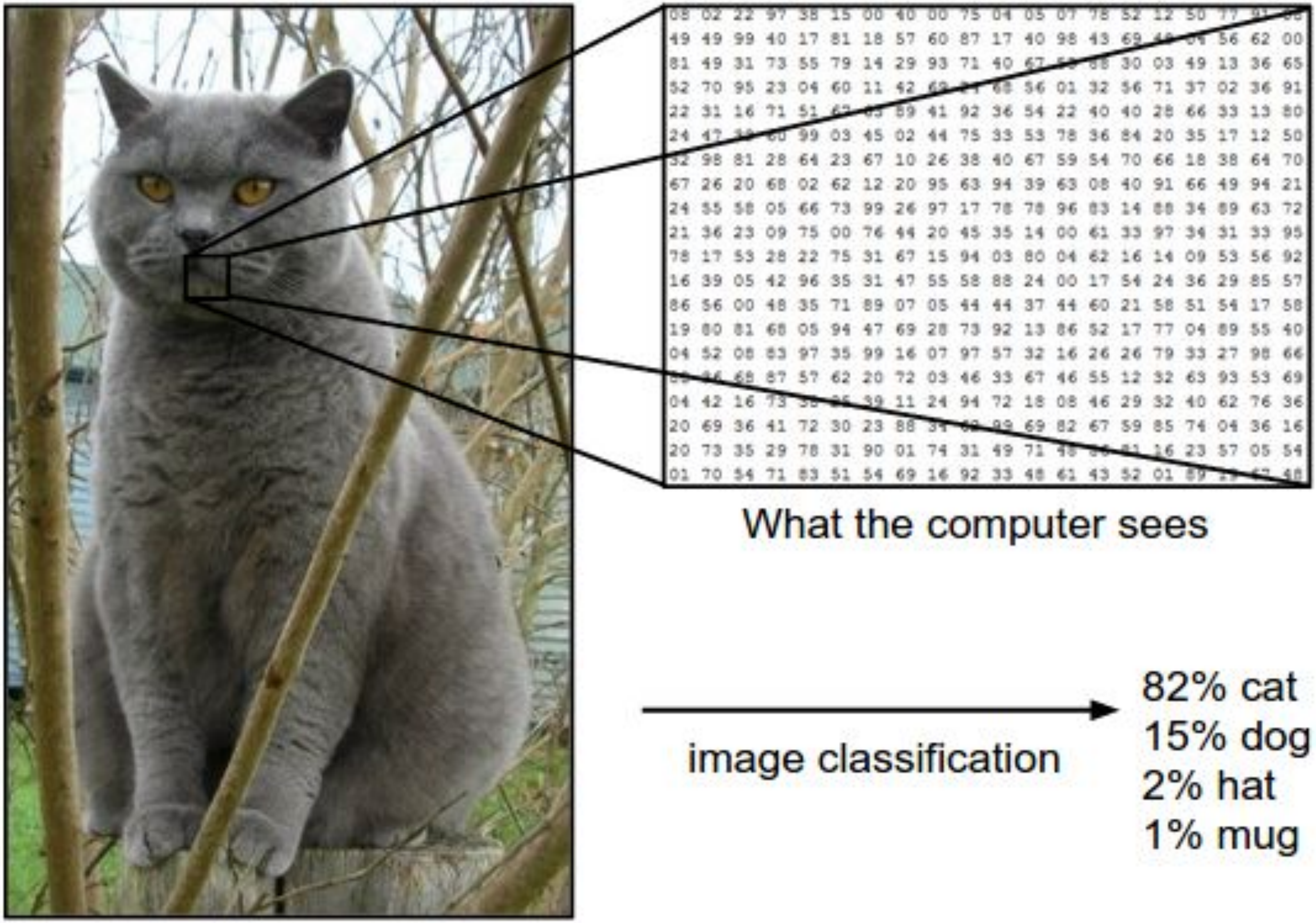
1.
Introduction

Image Classification

- The task of assigning an input image one label from a fixed set of categories

- One of the core problems in Computer Vision

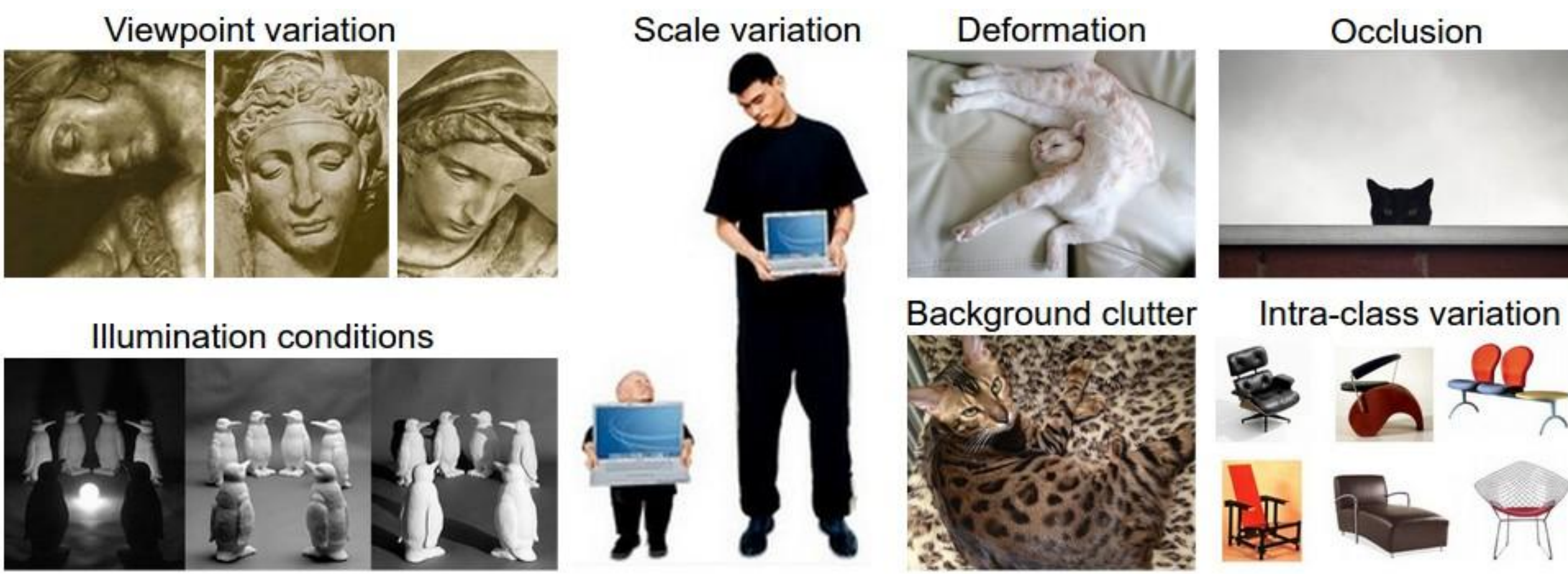
- Many other Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification


What the computer sees
image classification
82% cat
15% dog
2% hat
1% mug

Challenges

- Viewpoint variation
- Scale variation
- Deformation
- Occlusion
- Illumination conditions
- Back ground clutter
- Intra-class variation

A good image classification model must be invariant to the cross product of all these variations, while simultaneously retaining sensitivity to the inter-class variations.



Data-driven approach

Provide the computer with many examples of each class and then develop learning algorithms that look at these examples and learn about the visual appearance of each class.

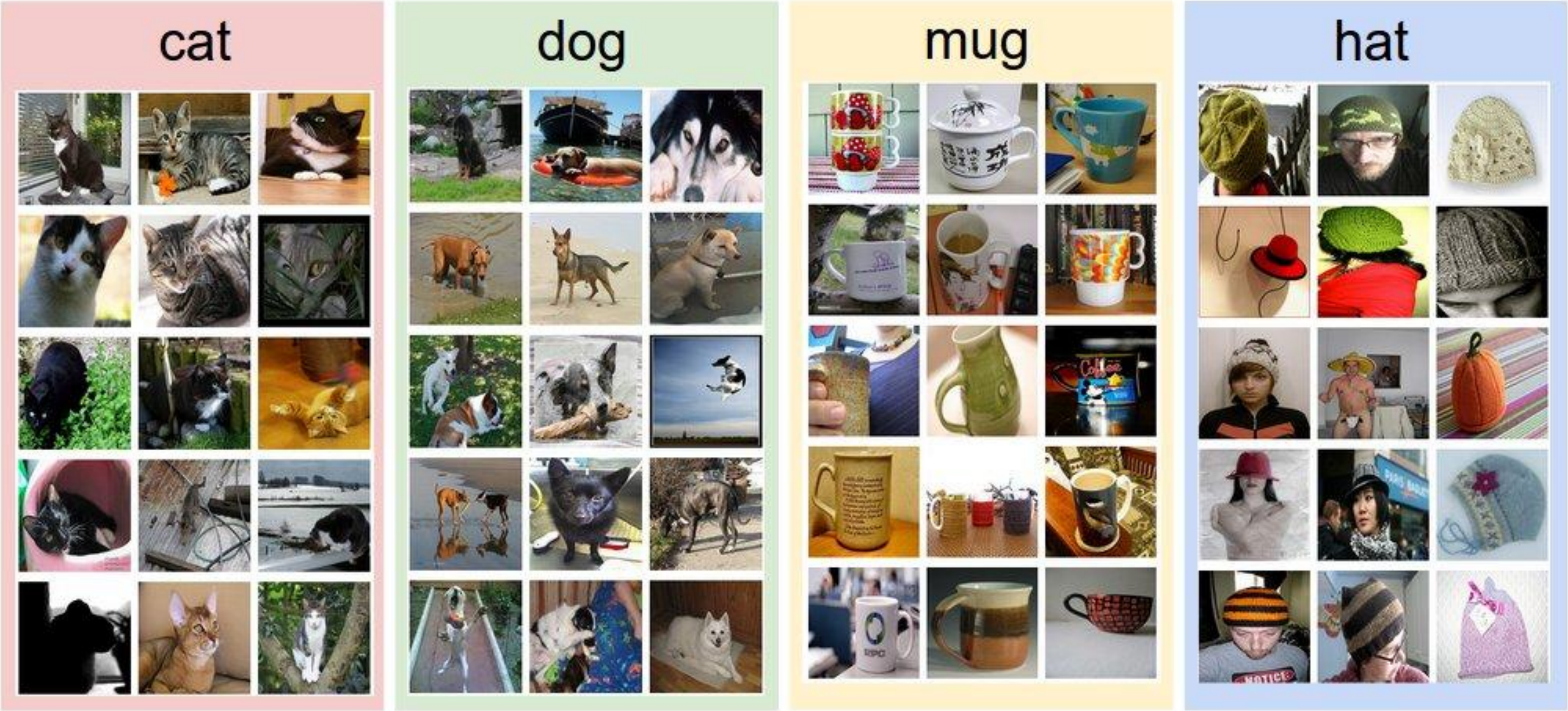

cat
dog
mug
hat

Image Classification Pipeline

- **Input**: Our input consists of a set of N images, each labeled with one of K different classes

- **Learning**: use the training set to learn what every one of the classes looks like

- **Evaluation**: evaluate the quality of the classifier by asking it to predict labels for a new set of images that it has never seen before.

2.

Nearest Neighbor Classifier

Nearest Neighbor

The nearest neighbor classifier will take a test image, compare it to every single one of the training images, and predict the label of the closest training image.

One of the simplest possibilities is to compare the images pixel by pixel and add up all the differences. In other words, given two images and representing them as vectors:

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

L2 distance

Compute the pixel-wise difference as before, but this time we square all of them, add them up and finally take the square root.

L2 distance is much more unforgiving than the L1 distance. That is, the L2 distance prefers many medium disagreements to one big one.

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

Two images are subtracted elementwise and then all differences are added up to a single number. If two images are identical the result will be zero. But if the images are very different the result will be large.

test image

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

-

training image

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

=

pixel-wise absolute value differences

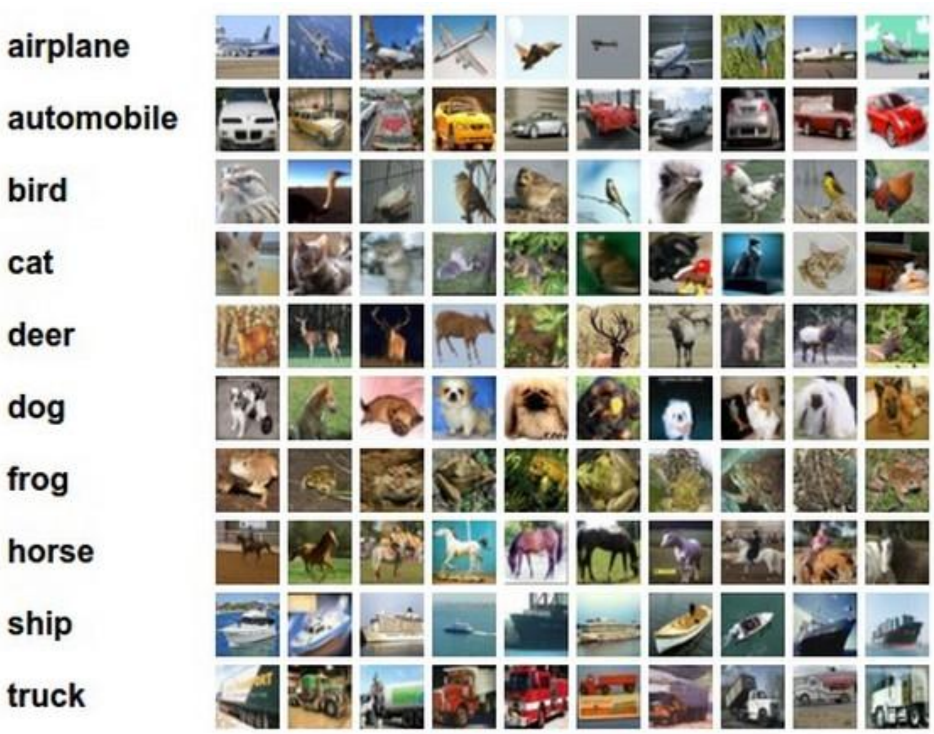
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

Cifar-10 Dataset

- Consists of 60,000 tiny images that are 32 pixels high and wide
- Each image is labeled with one of 10 classes
- These 60,000 images are partitioned into a training set of 50,000 images and a test set of 10,000 images.

Cifar-10 Dataset



Coding time!

1. Implement the NN classifier and then classify Cifar-10 test set using your classifier.

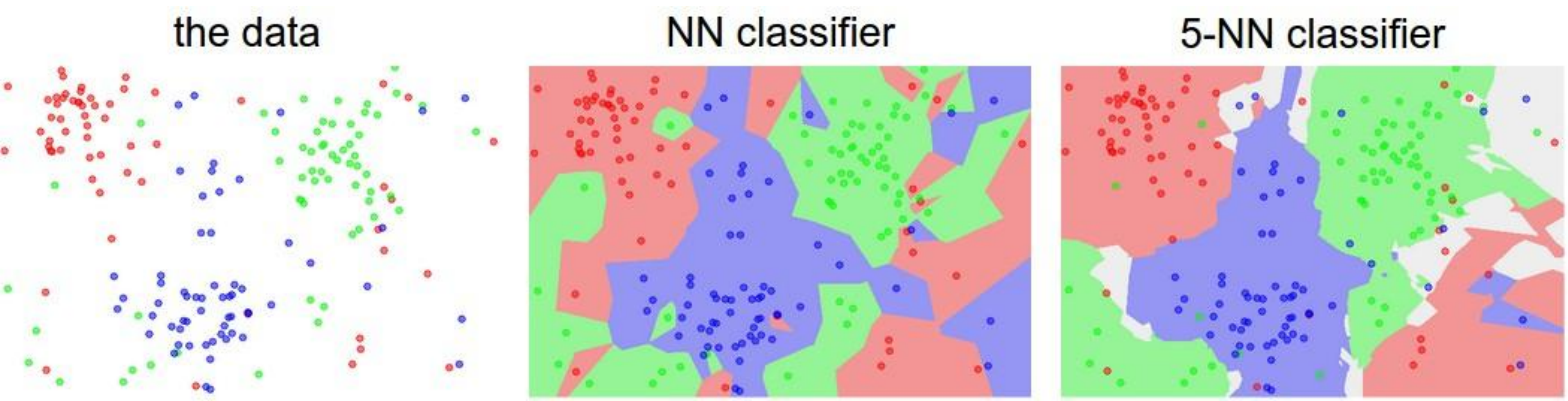
2. Try with the L2 distance!

3.

Validation Sets

k - Nearest Neighbor Classifier

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.



Hyperparameter Tuning

- What number of K works best?
- L1 or L2?

These choices are called hyperparameters and they come up very often in the design of many Machine Learning algorithms that learn from data. It’s often not obvious what values one should choose.

Overfitting to the test set

- Don’t use the test set for the purpose of tweaking hyperparameters!
- Test set should ideally never be touched until one time at the very end.
- the very real danger is that you may tune your hyperparameters to work well on the test set, but if you were to deploy your model you could see a significantly reduced performance.

Validation set

Split the training set in two: a slightly smaller training set, and what we call a validation set.

This validation set is essentially used as a fake test set to tune the hyper-parameters.

Coding time!

Implement KNN and test it with several values of K and find which one works best.

Cross Validation

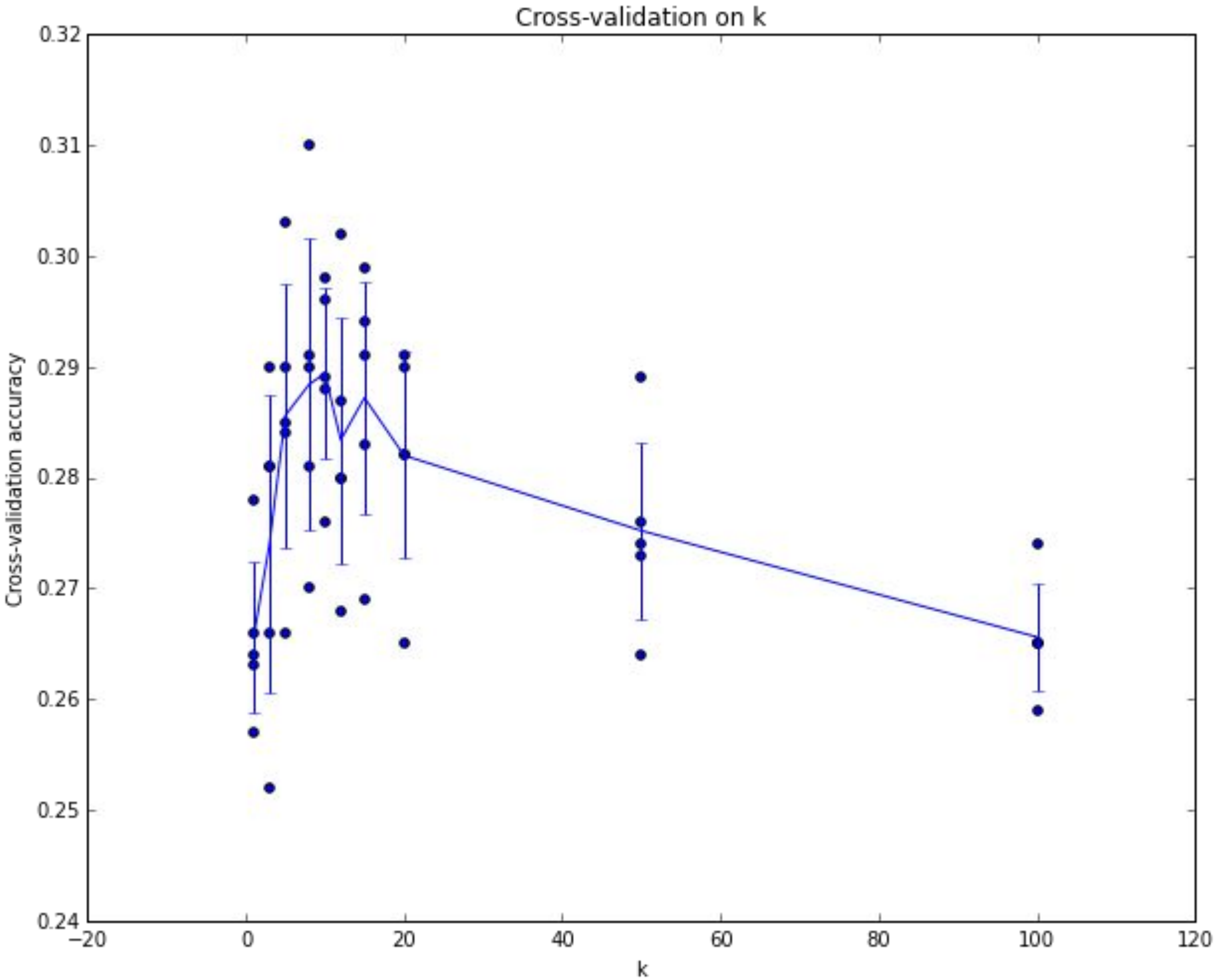
Instead of arbitrarily picking the first 1000 data points to be the validation set and rest training set, you can get a better and less noisy estimate of how well a certain value of k works by iterating over different validation sets and averaging the performance across these.

N-fold cross-validation

For example, in 5-fold cross-validation, we would split the training data into 5 equal folds, use 4 of them for training, and 1 for validation.

We would then iterate over which fold is the validation fold, evaluate the performance, and finally average the performance across the different folds.

5-fold Cross Validation



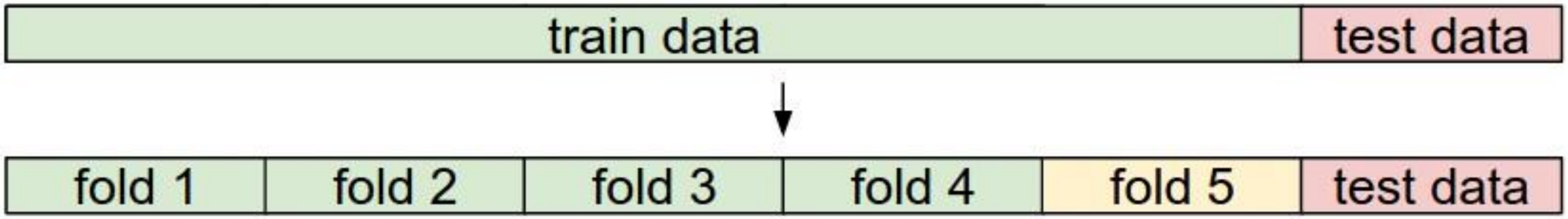
In Practice:

- People prefer to avoid cross-validation in favor of having a single validation split, since cross validation can be computationally expensive.

- The splits people tend to use is between 50%-90% of the training data for training and rest for validation.

- Typical number of folds you can see in practice would be 3-fold, 5-fold or 10-fold cross-validation.

Cross-validation



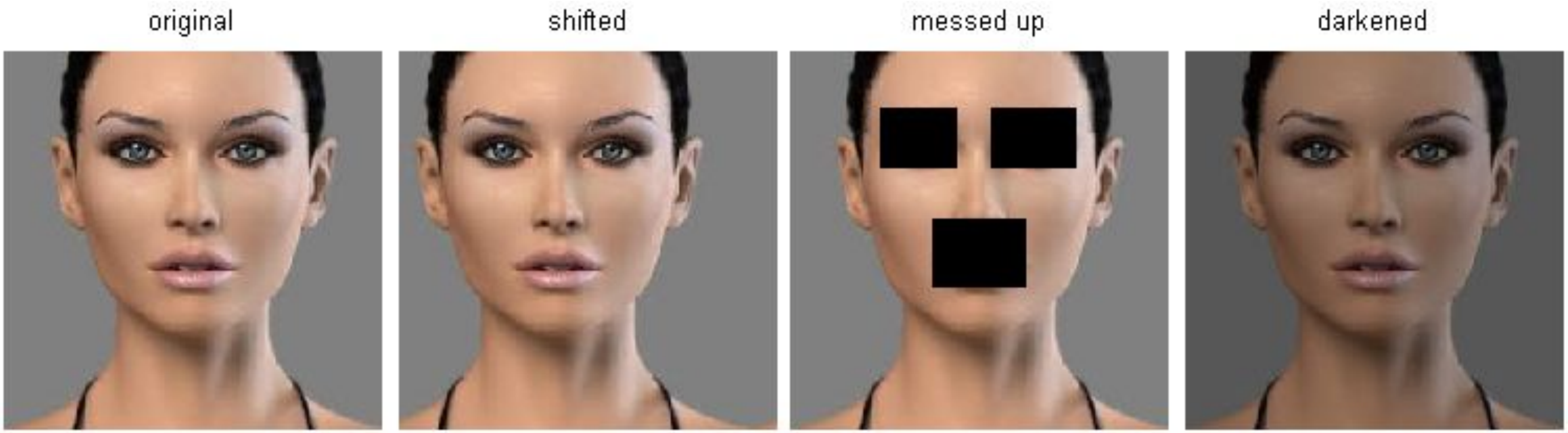
4.

Pros and Cons of NN

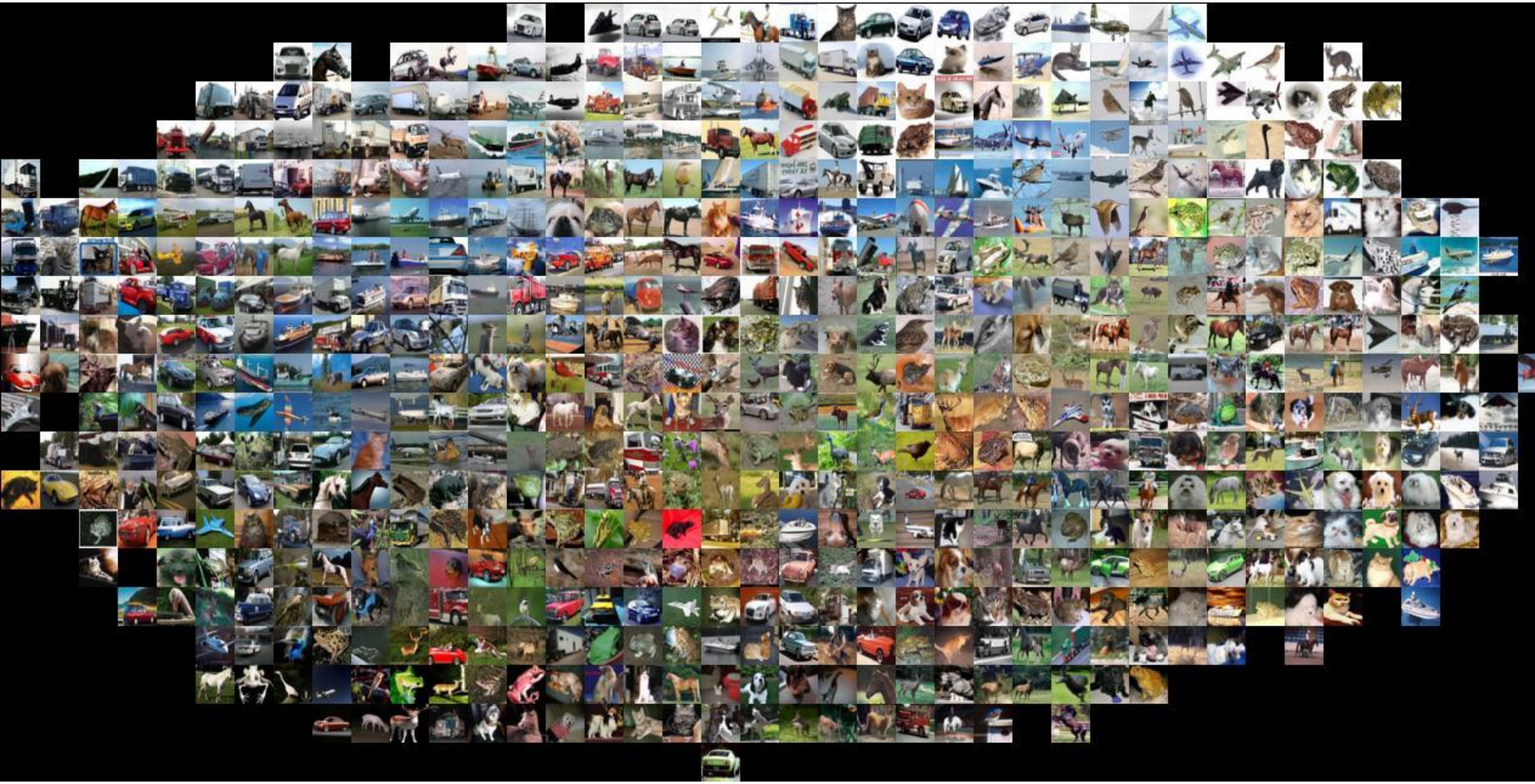
Pros and Cons of Nearest Neighbor classifier

- very simple to implement and understand
- The classifier takes no time to train, however, we pay that computational cost at test time
- Images are high-dimensional objects and distances over high-dimensional spaces can be very counter-intuitive.

An original image (left) and three other images next to it that are all equally far away from it based on L2 pixel distance. Clearly, the pixel-wise distance does not correspond at all to perceptual or semantic similarity.



Strong effect of background rather than semantic class differences



Thanks!

Any questions?