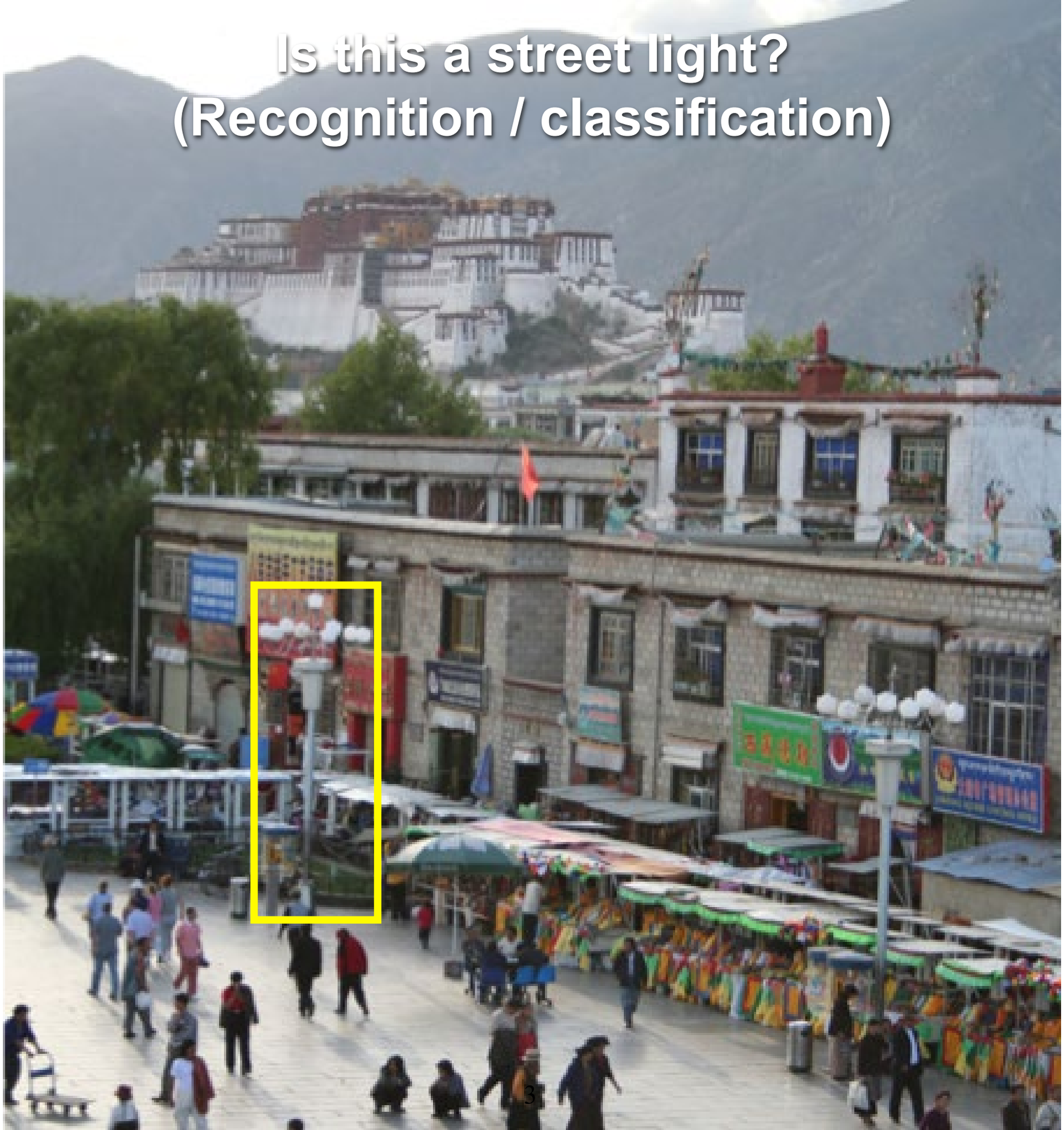
# Object Classification

<Vision System>

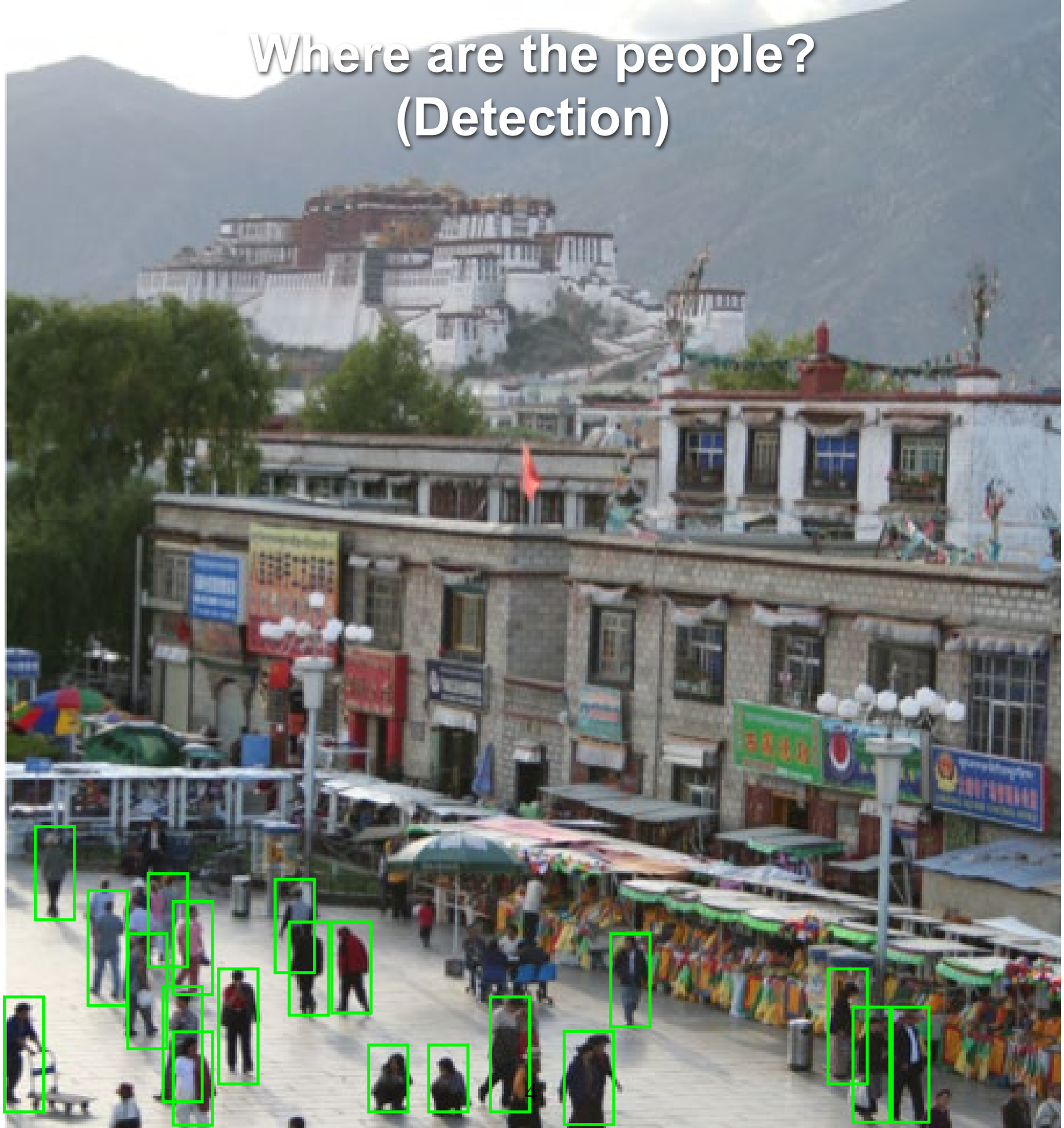
Department of Robot Engineering
Prof. Younggun Cho

# What do we mean by learning-based vision or ‘semantic vision’?

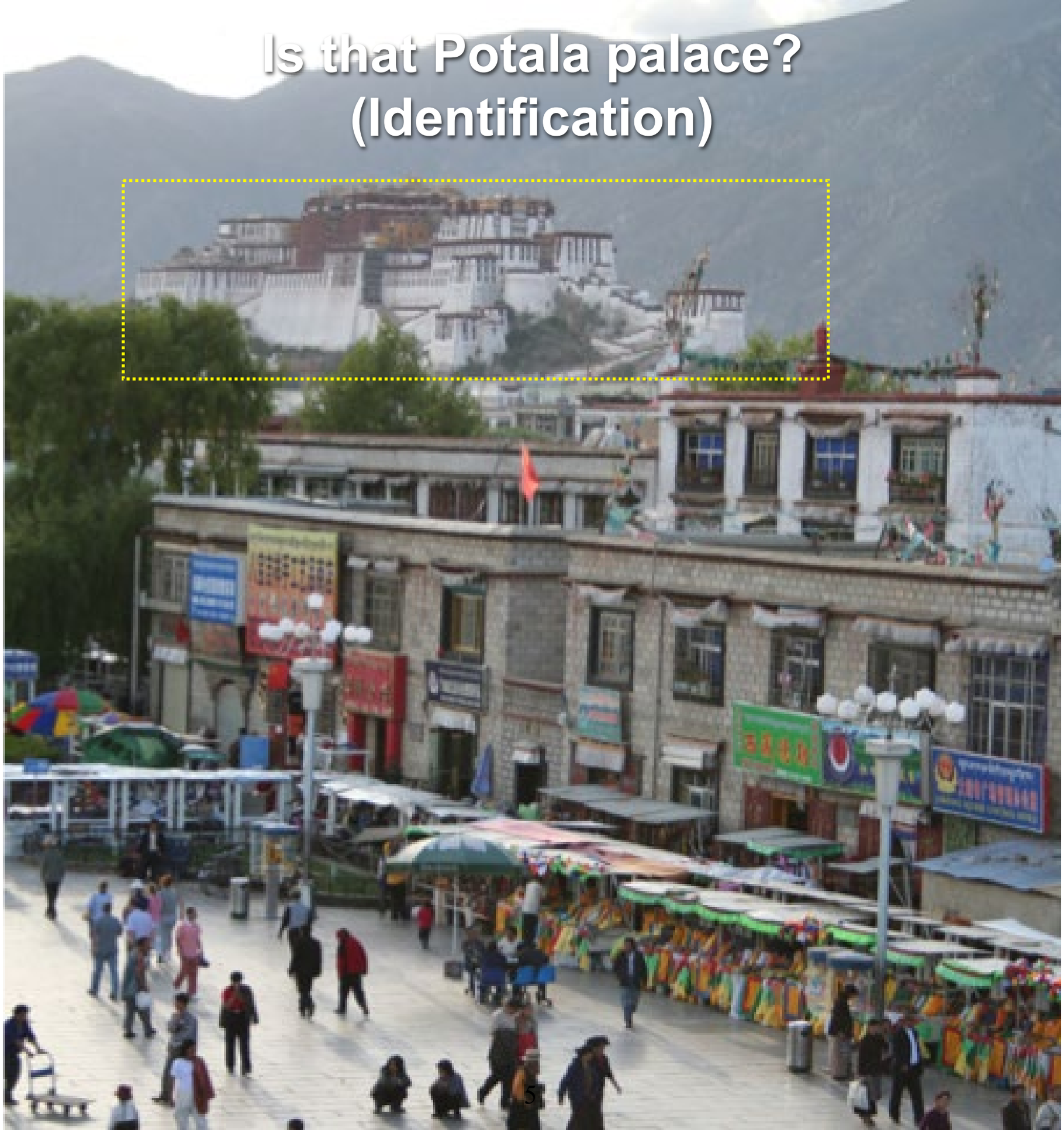
# Is this a street light?
# (Recognition / classification)



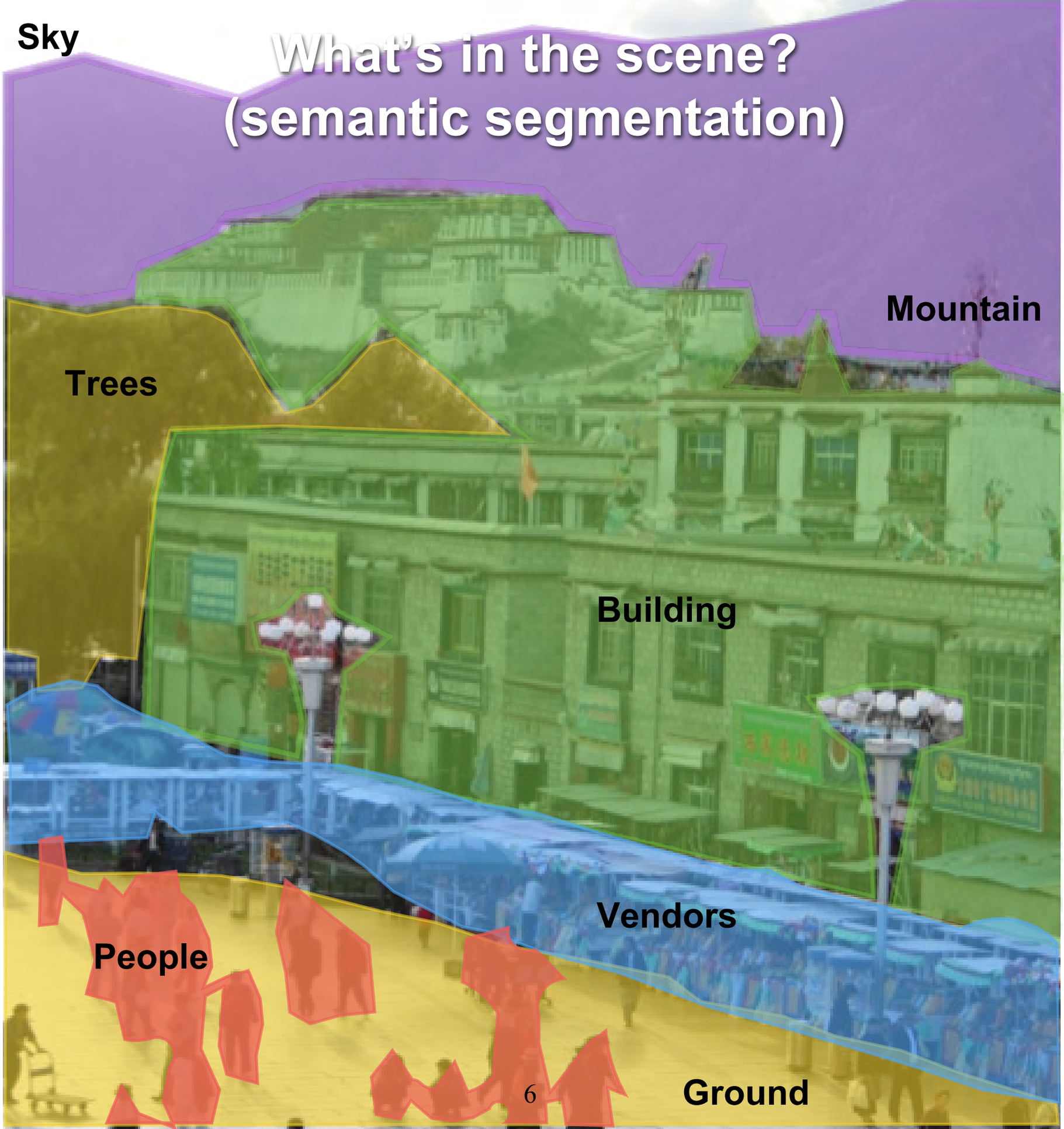
# Where are the people? (Detection)



# Is that Potala palace? (Identification)

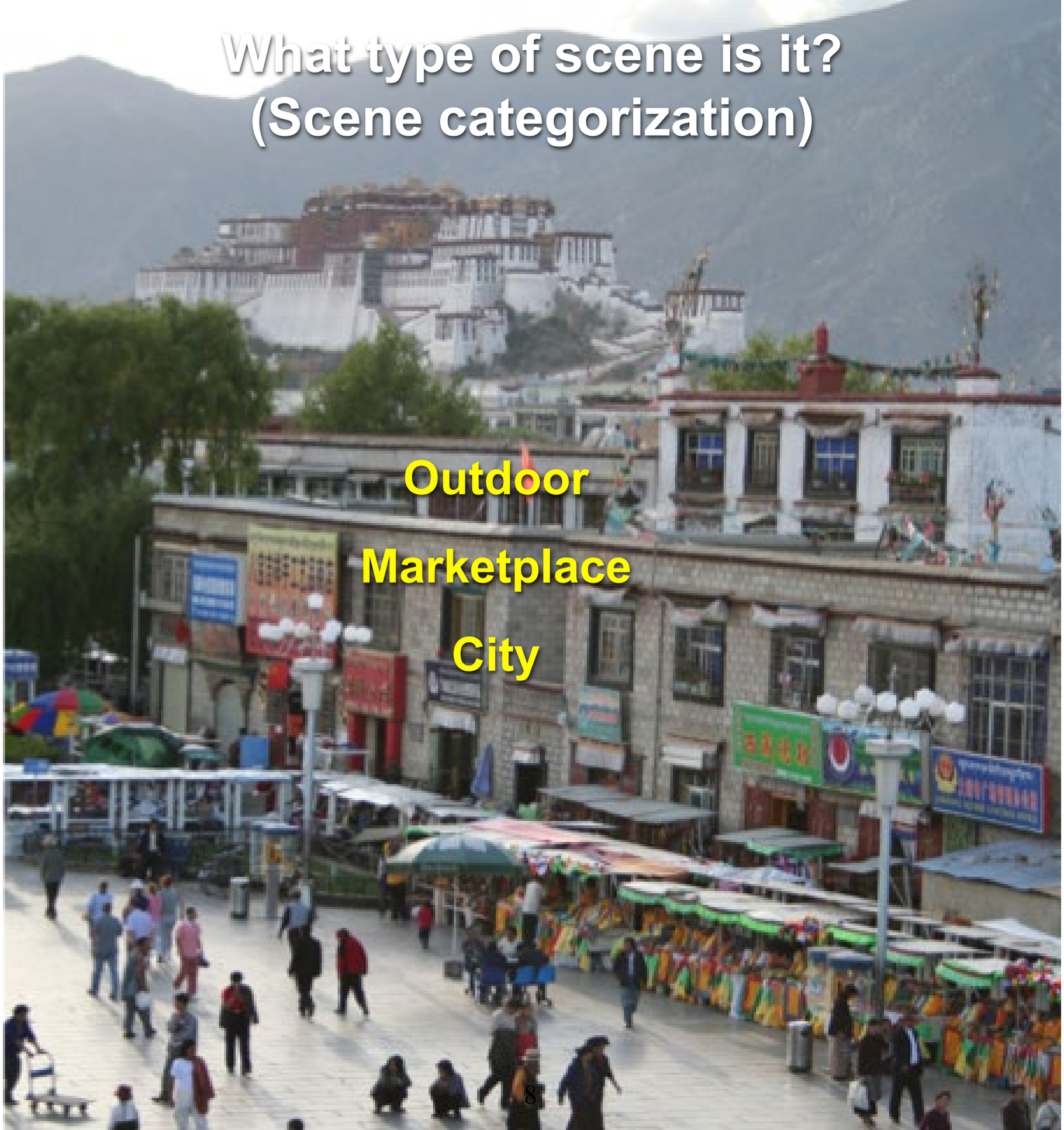


# What's in the scene? (semantic segmentation)

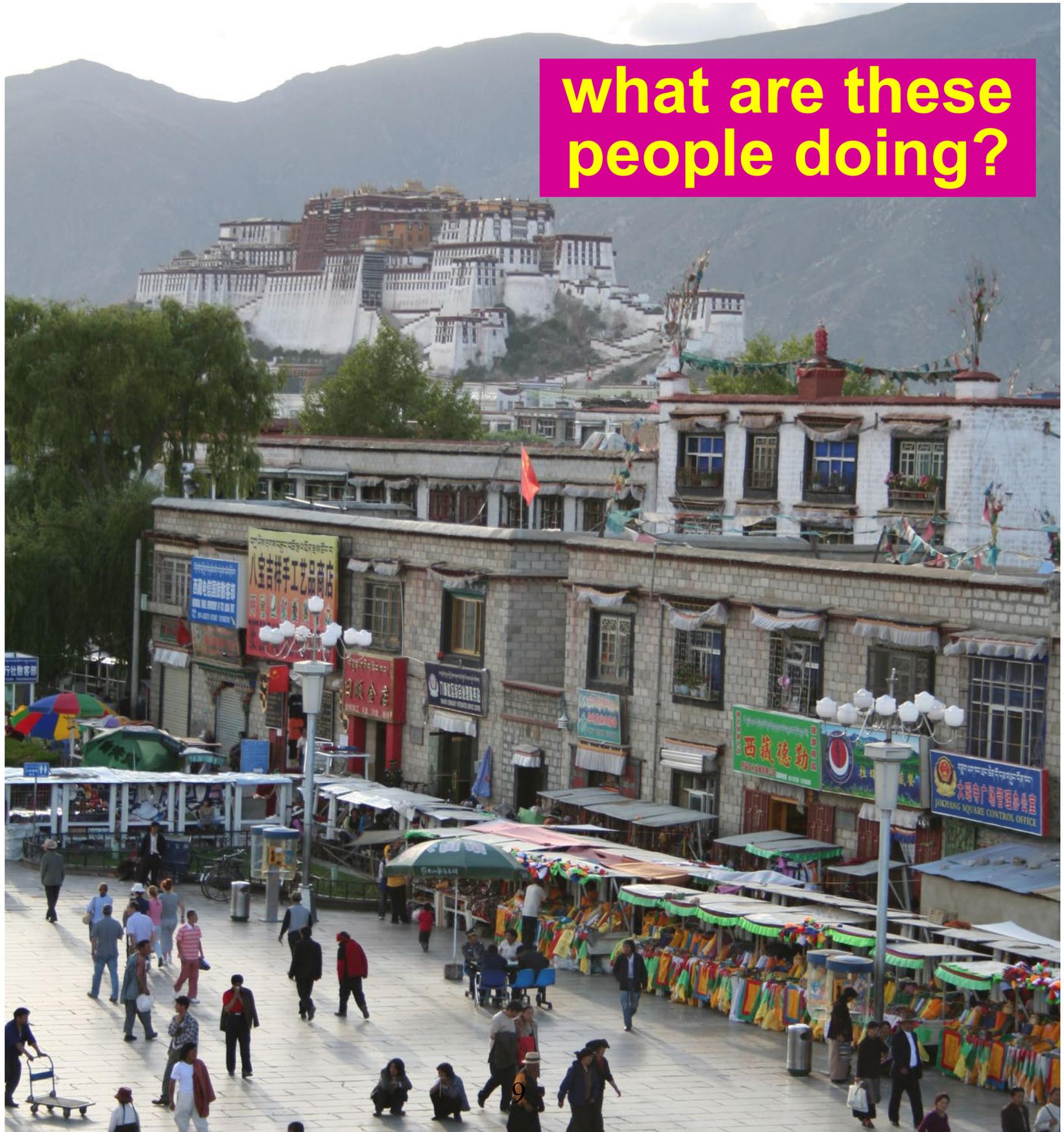


# Object categorization

# What type of scene is it? (Scene categorization)



# Activity / Event Recognition

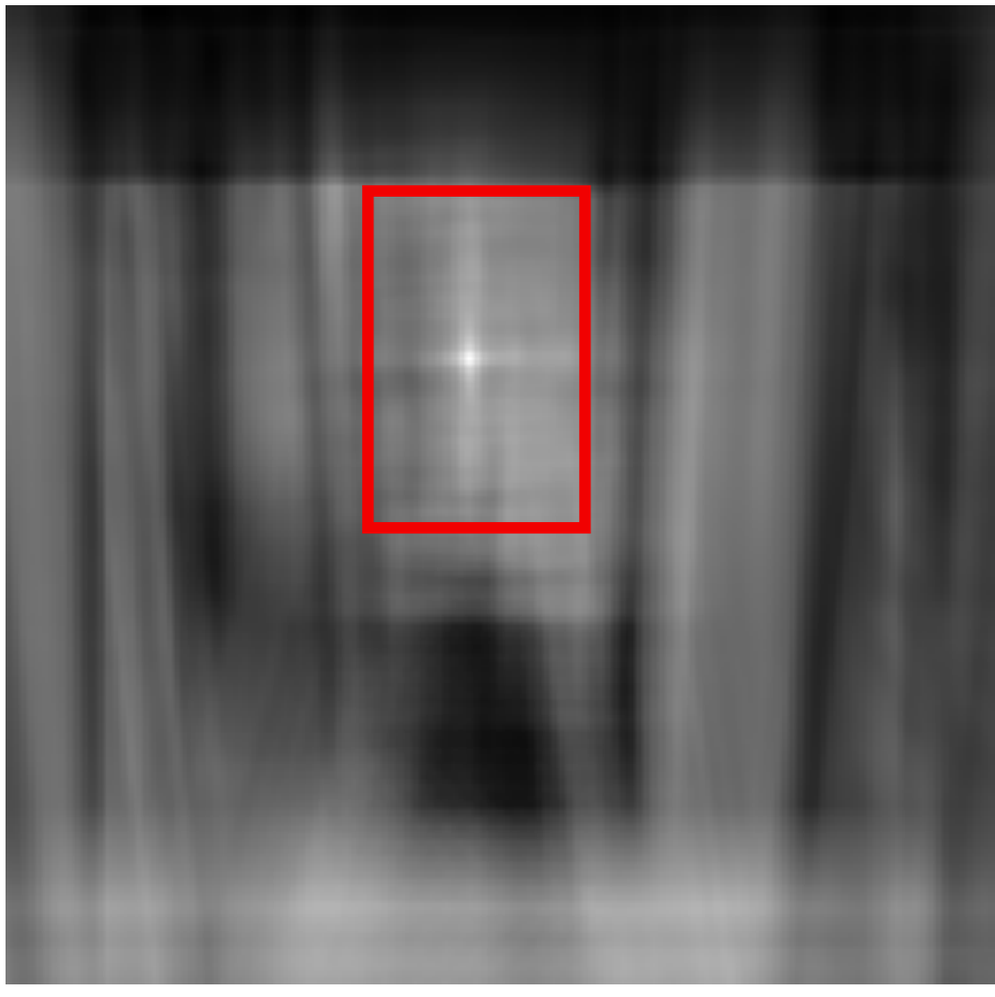


# Object recognition
# Is it really so hard?

This is a chair

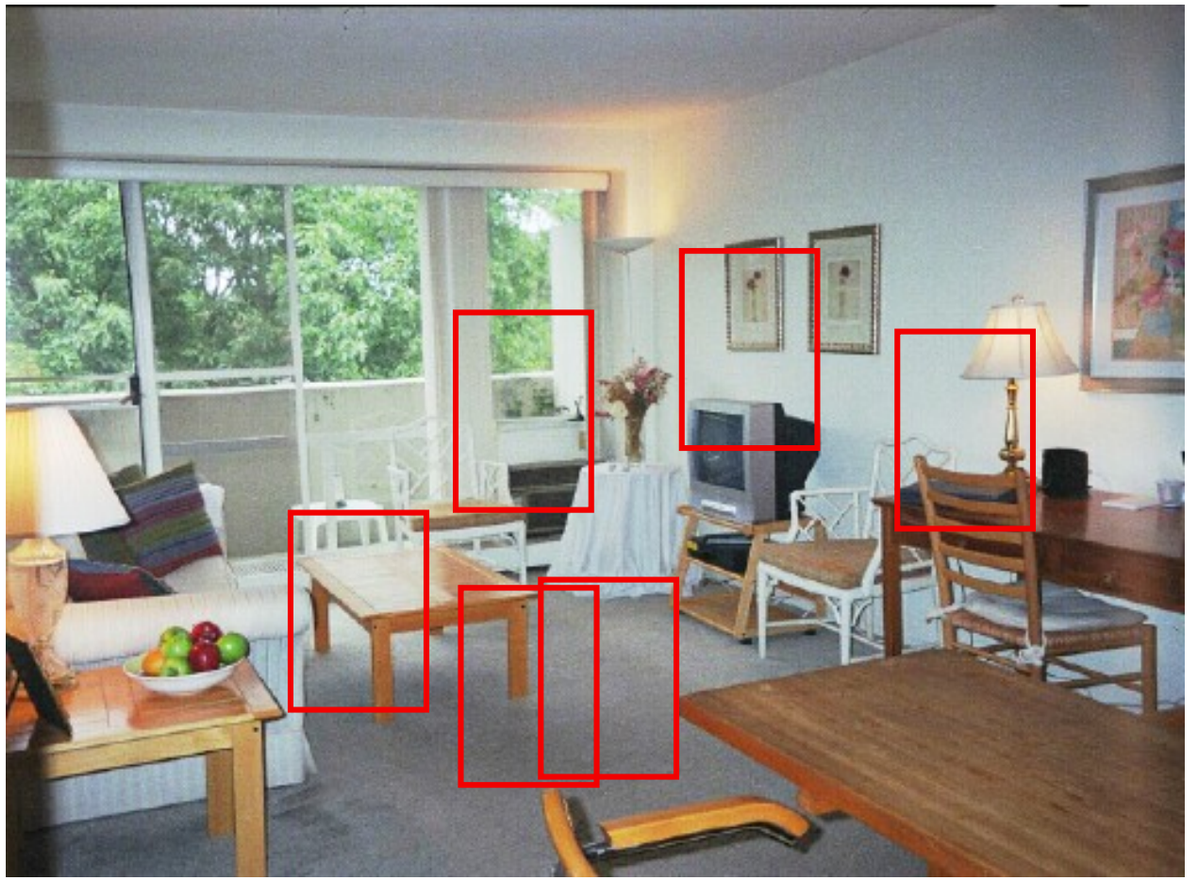
Find the chair in this image

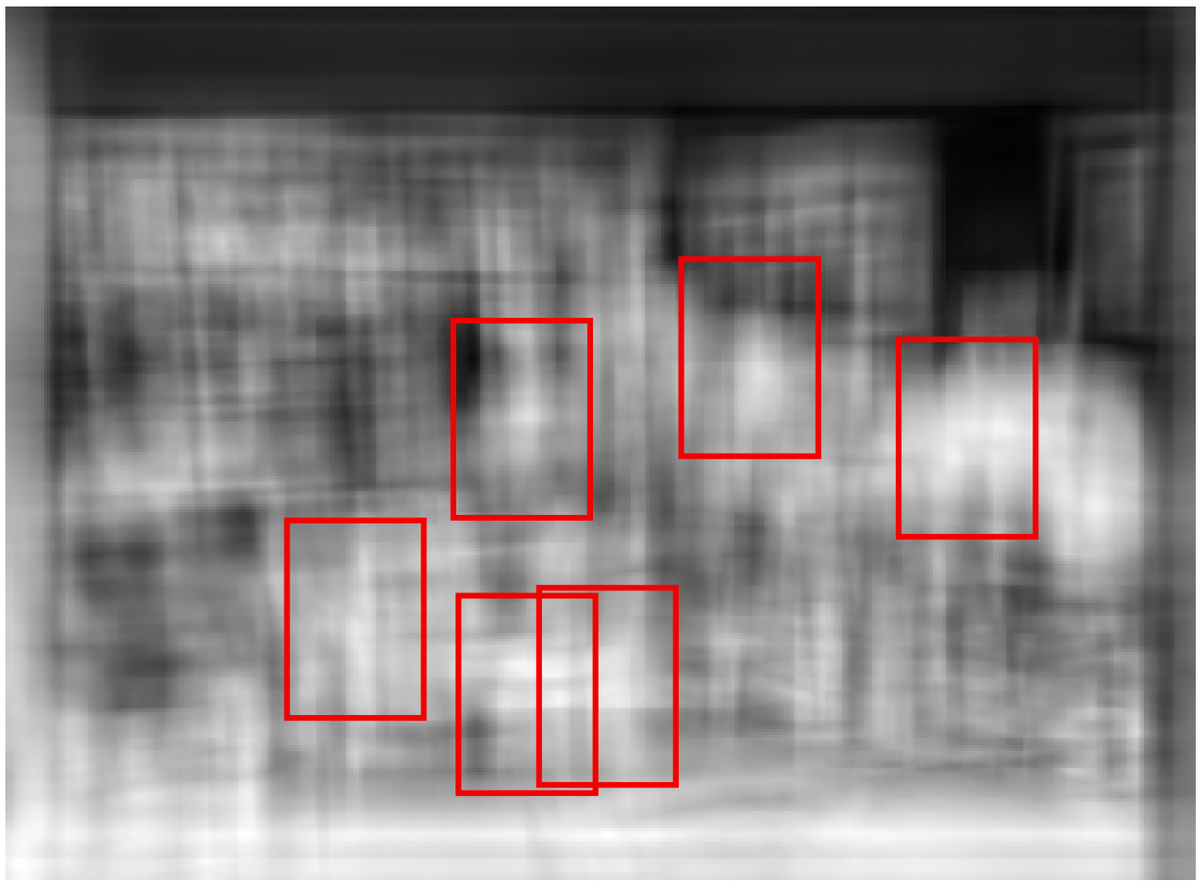
Output of normalized correlation



# Object recognition
# Is it really so hard?

Find the chair in this image





Pretty much garbage

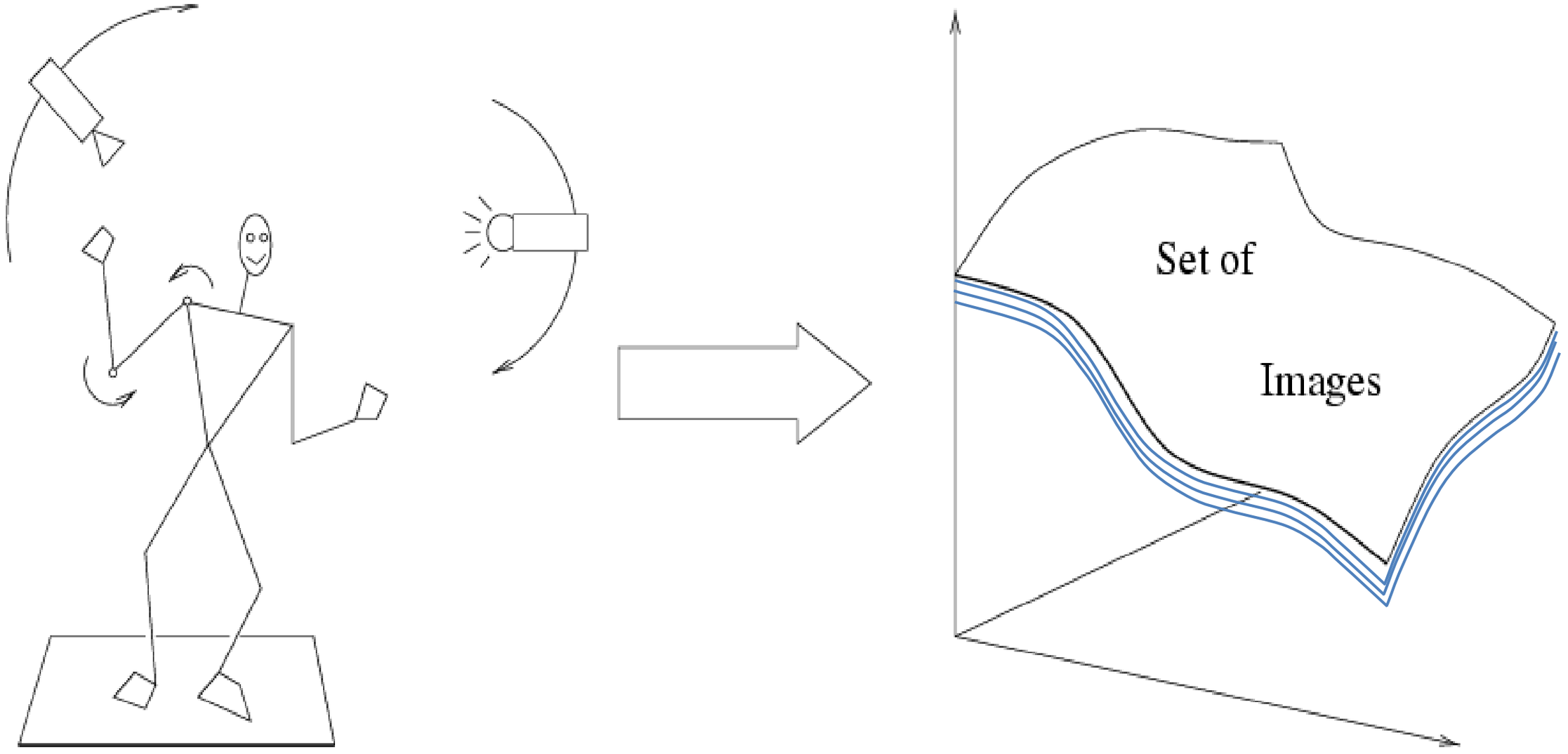
Simple template matching is not going to make it

A “popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts.” Nivatia & Binford, 1977.

# And it can get a lot harder

Brady, M. J., & Kersten, D. (2003). Bootstrapped learning of novel objects. J Vis, 3(6), 413-422

# Why is this hard?




Variability: Camera position
Illumination
Shape parameters

# How many object categories are there?

**~10,000 to 30,000**

# Challenge: variable viewpoint

Michelangelo 1475-1564

# Challenge: variable illumination

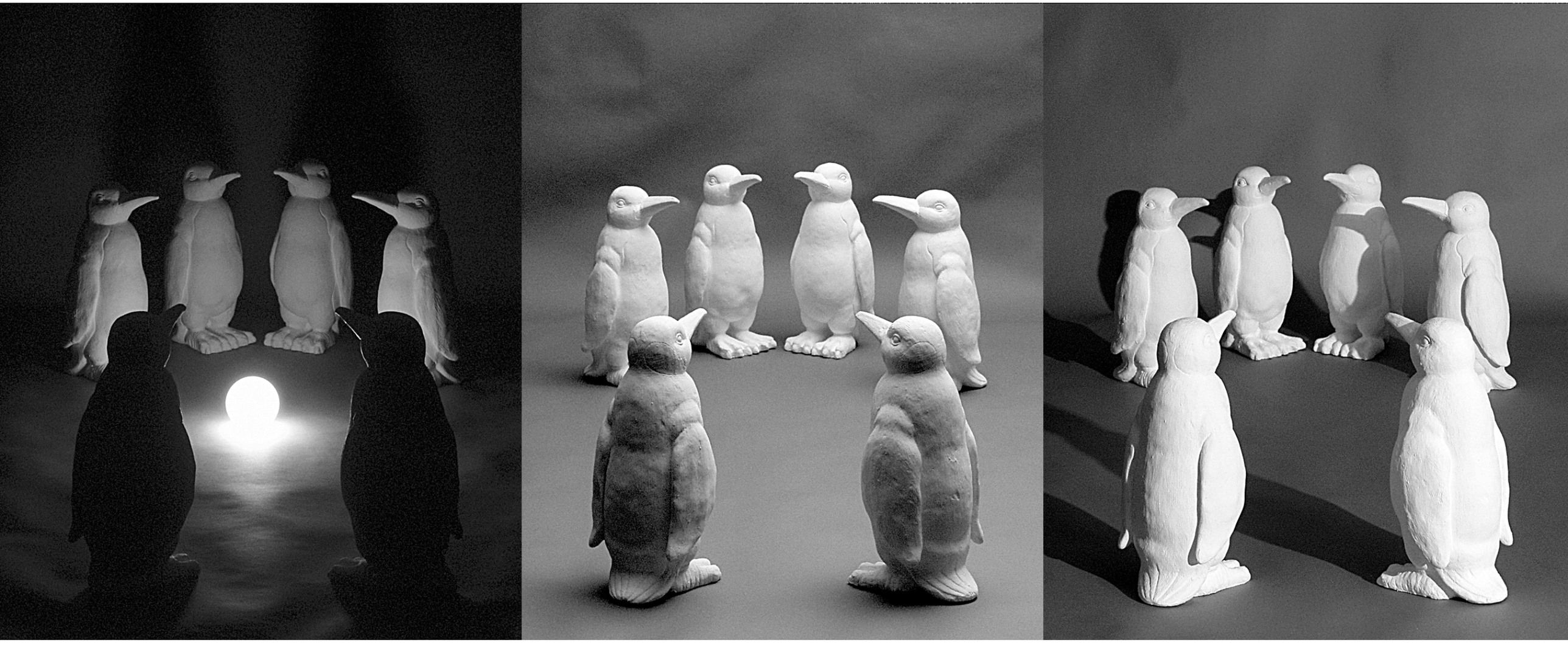


image credit: J. Koenderink


# Challenge: scale

# Challenge: deformation

# Deformation

# Challenge: Occlusion

Magritte, 1957

# Occlusion

# Challenge: background clutter

Kilmeny Niland. 1995

# Challenge: Background clutter

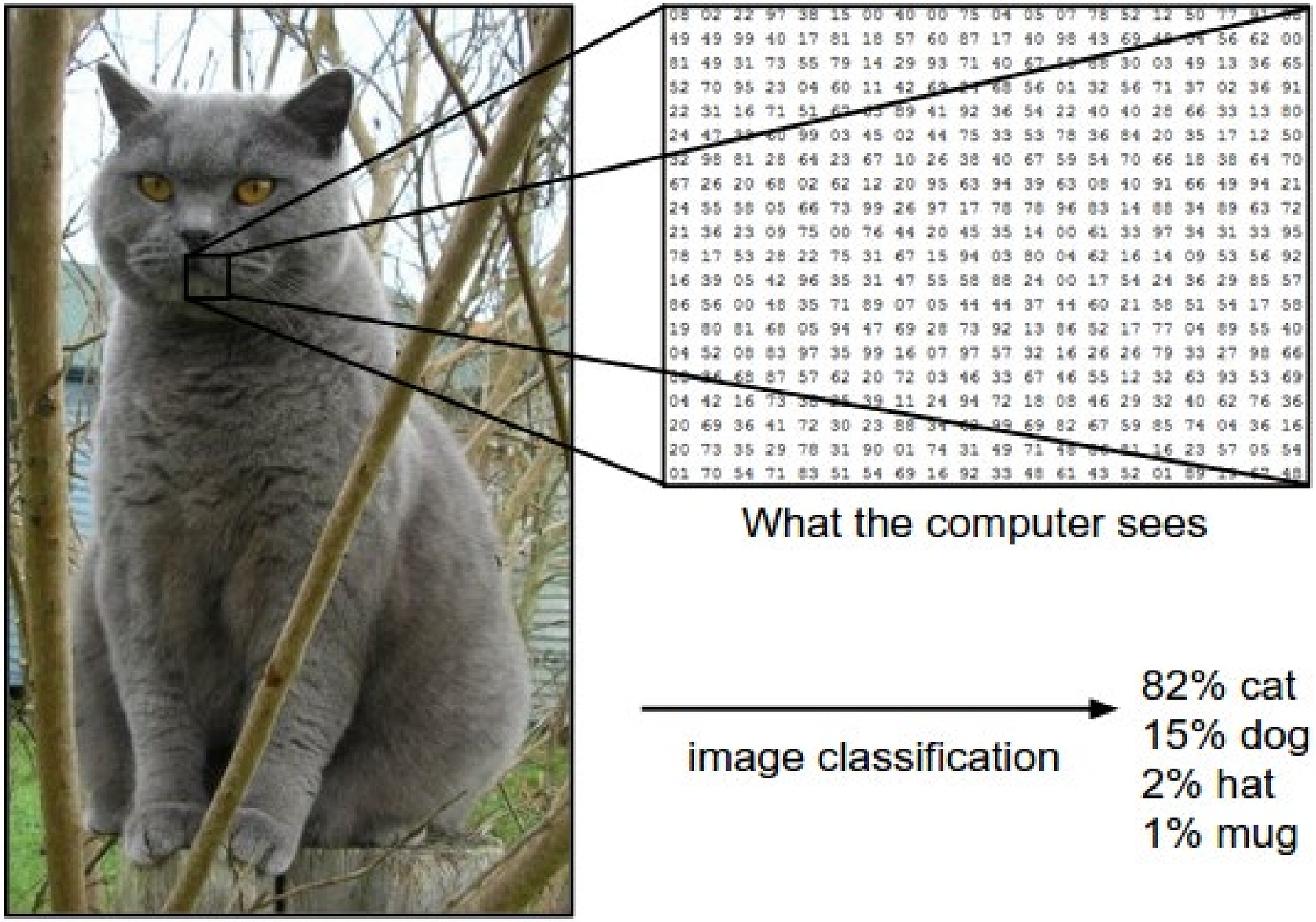
# Challenge: intra-class variations

# Image Classification

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

cat

# Image Classification: Problem



# Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

# Bag of words

# What object do these parts belong to?





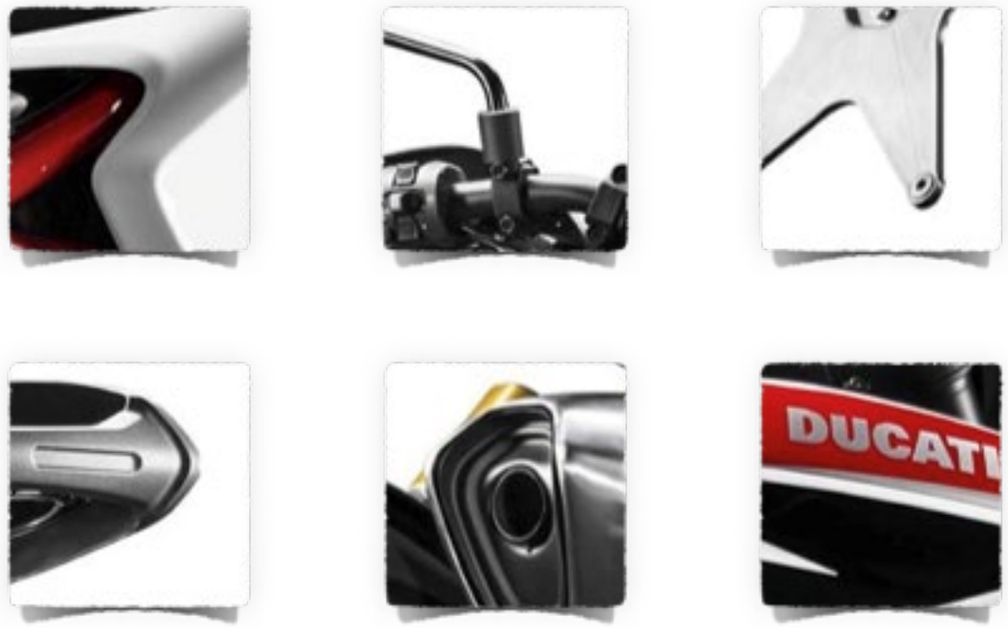









Some local feature are very informative



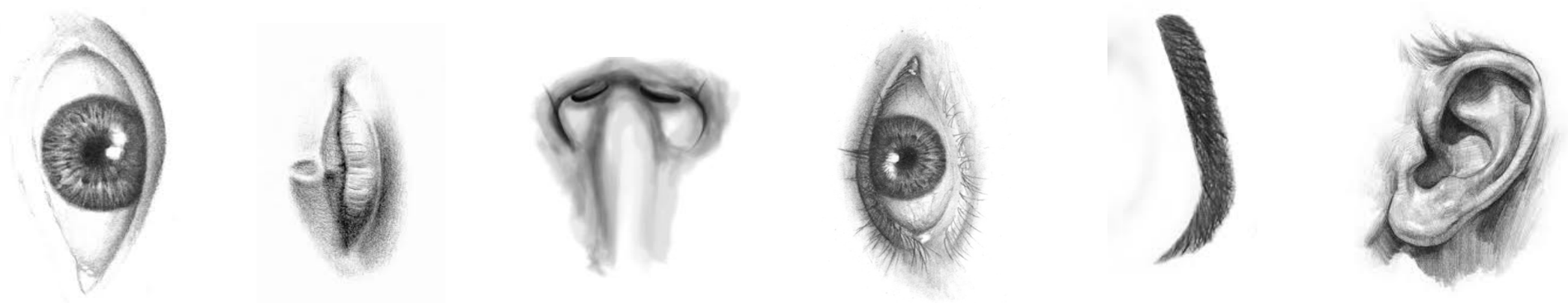
An object as


a collection of local features

(bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant

# (not so) crazy assumption



spatial information of local features
can be ignored for object recognition (i.e., verification)

# CalTech6 dataset






| class | bag of features | bag of features | Parts-and-shape model |
|---|---|---|---|
| | Zhang et al. (2005) | Willamowski et al. (2004) | Fergus et al. (2003) |
| airplanes | **98.8** | 97.1 | 90.2 |
| cars (rear) | 98.3 | **98.6** | 90.3 |
| cars (side) | **95.0** | 87.3 | 88.5 |
| faces | **100** | 99.3 | 96.4 |
| motorbikes | **98.5** | 98.0 | 92.5 |
| spotted cats | **97.0** | — | 90.0 |

Works pretty well for image-level classification

Csurka et al. (2004), Willamowski et al. (2005), Grauman & Darrell (2005), Sivic et al. (2003, 2005)

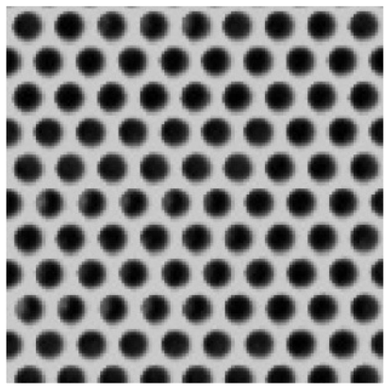
# Bag-of-features

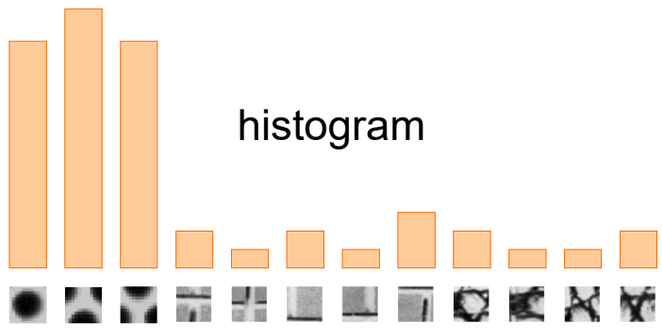
represent a data item (document, texture, image)
as a histogram over features

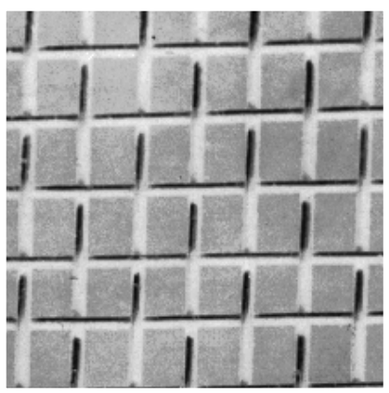
an old idea

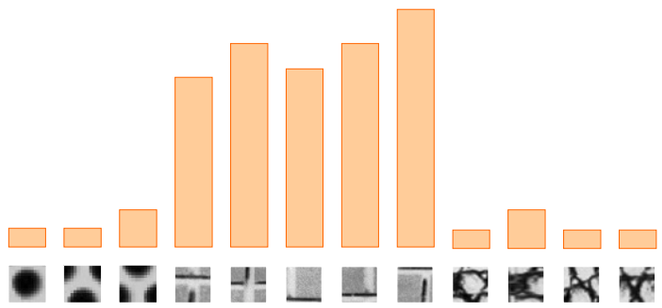
(e.g., texture recognition and information retrieval)

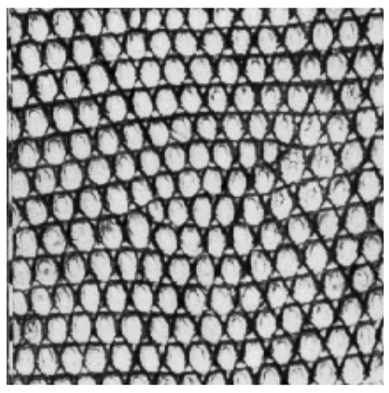
# Texture recognition

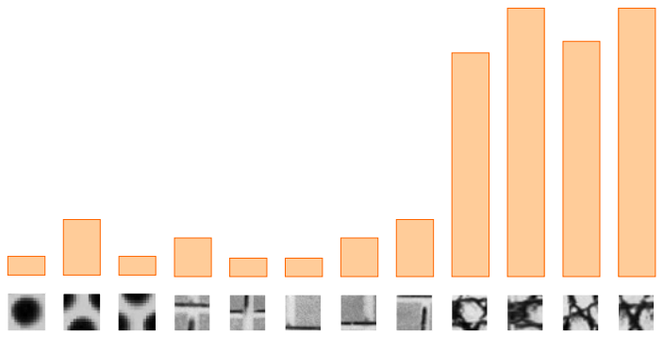




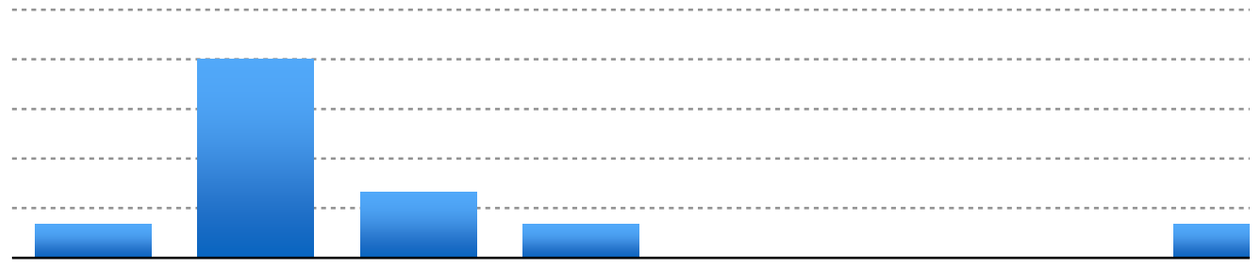






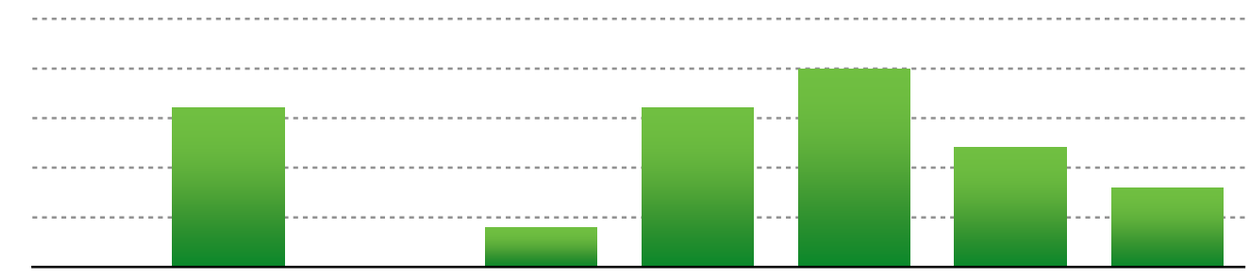
# Vector Space Model

G. Salton. 'Mathematics and Information Retrieval' Journal of Documentation,1979

| 1 | 6 | 2 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| Tartan | robot | CHIMP | CMU | bio | soft | ankle | sensor |



| 0 | 4 | 0 | 1 | 4 | 5 | 3 | 2 |
|---|---|---|---|---|---|---|---|
| Tartan | robot | CHIMP | CMU | bio | soft | ankle | sensor |



http://www.fodey.com/generators/newspaper/snippet.asp

A document (datapoint) is a vector of counts over each word (feature)

$$\boldsymbol{v}_d = [n(w_{1,d}) \quad n(w_{2,d}) \quad \cdots \quad n(w_{T,d})]$$

just a histogram over words

$n(\cdot)$ counts the number of occurrences

What is the similarity between two documents?

A document (datapoint) is a vector of counts over each word (feature)

$$\boldsymbol{v}_d = [n(w_{1,d}) \quad n(w_{2,d}) \quad \cdots \quad n(w_{T,d})]$$

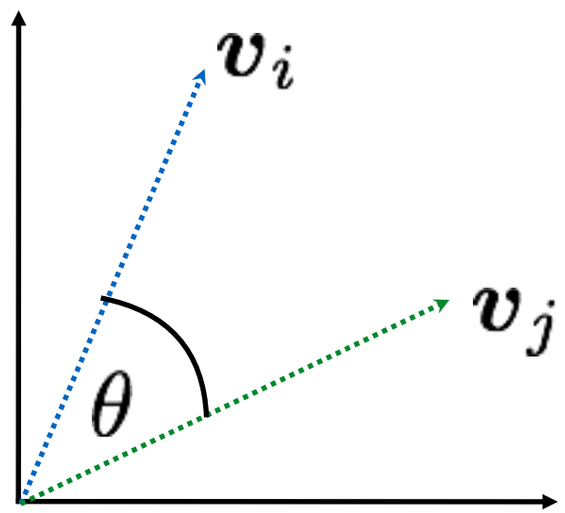
$n(\cdot)$ counts the number of occurrences

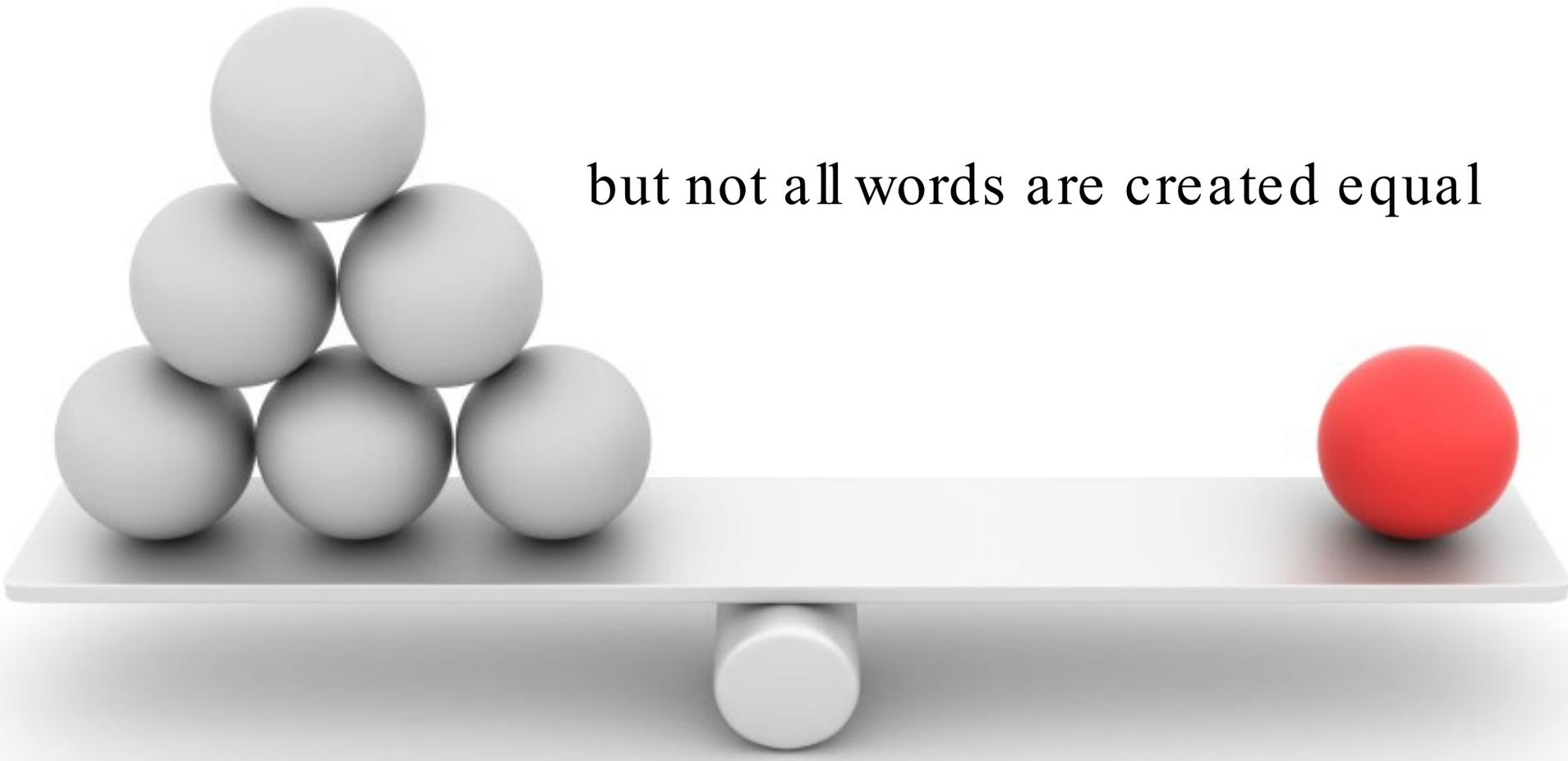
just a histogram over words


What is the similarity between two documents?

Use any distance you want but the cosine distance is fast.

$$d(\boldsymbol{v}_i, \boldsymbol{v}_j) = \cos\theta = \frac{\boldsymbol{v}_i \cdot \boldsymbol{v}_j}{\|\boldsymbol{v}_i\| \|\boldsymbol{v}_j\|}$$





but not all words are created equal

# TF-IDF

**T**erm **F**requency **I**nverse **D**ocument **F**requency

$$\boldsymbol{v}_d = [n(w_{1,d}) \quad n(w_{2,d}) \quad \cdots \quad n(w_{T,d})]$$

weigh each word by a heuristic

$$\boldsymbol{v}_d = [n(w_{1,d})\alpha_1 \quad n(w_{2,d})\alpha_2 \quad \cdots \quad n(w_{T,d})\alpha_T]$$

$$n(w_{i,d})\alpha_i = n(w_{i,d}) \log \left\{ \frac{D}{\sum_{d'} \mathbf{1}[w_i \in d']} \right\}$$

term frequency

inverse document frequency

(down-weights **common** terms)

# Standard BOW pipeline
(for image classification)

**Dictionary Learning:**

Learn Visual Words using clustering

**Encode:**

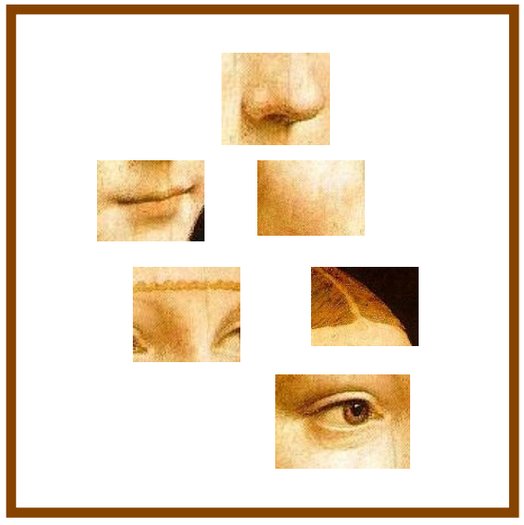
build Bags-of-Words (BOW) vectors for each image

**Classify:**

Train and test data using BOWs

# Dictionary Learning:
Learn Visual Words using clustering

1. extract features (e.g., SIFT) from images

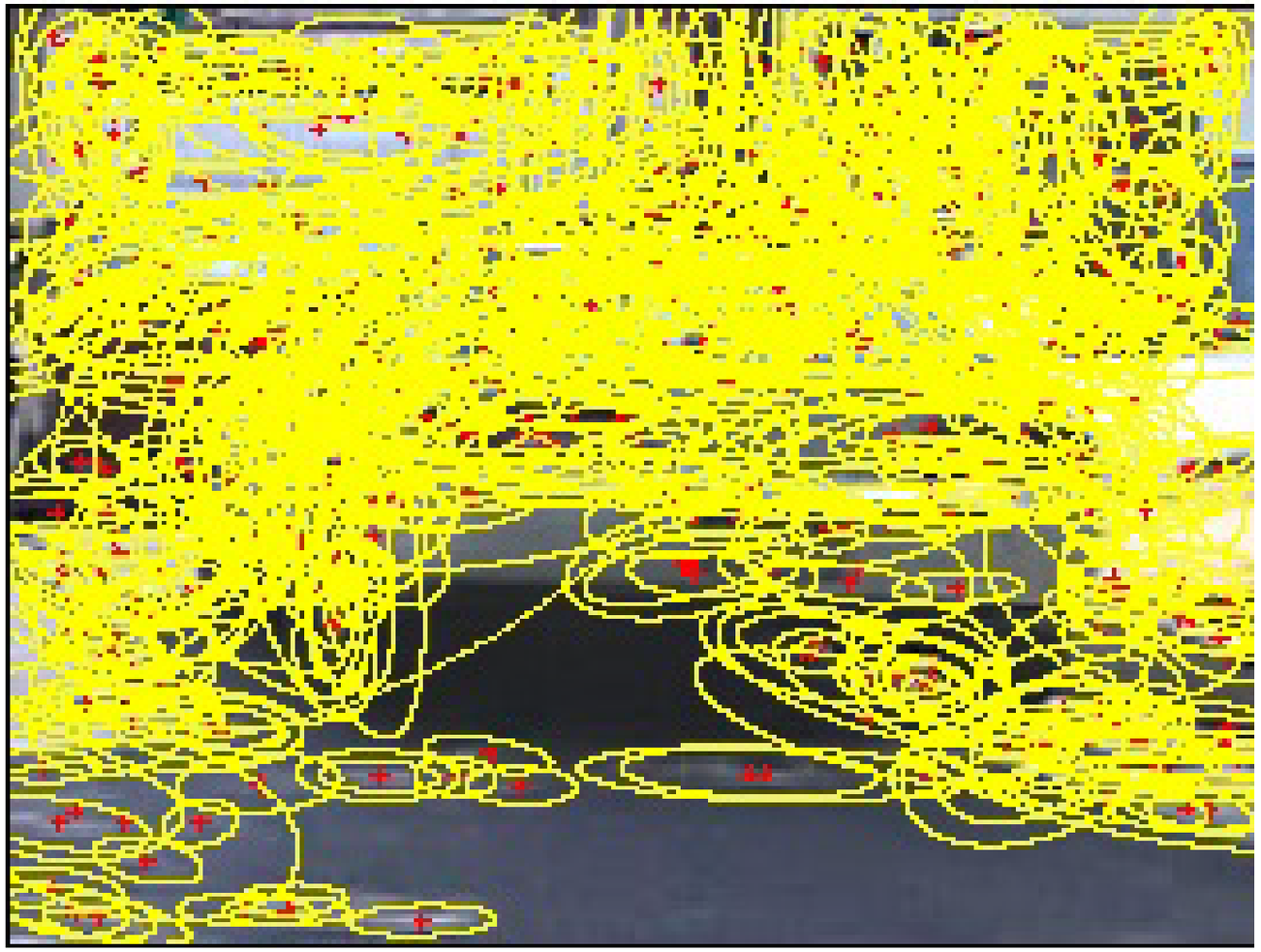


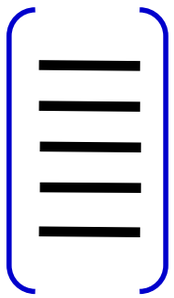
# Dictionary Learning:
Learn Visual Words using clustering

2. Learn visual dictionary (e.g., K-means clustering)

# *What kinds of features can we extract?*

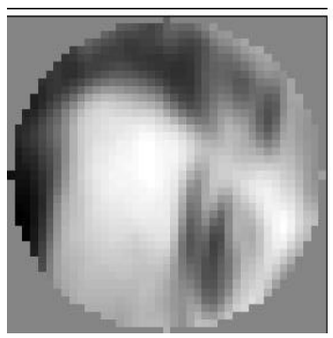
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches (Barnard et al. 2003)



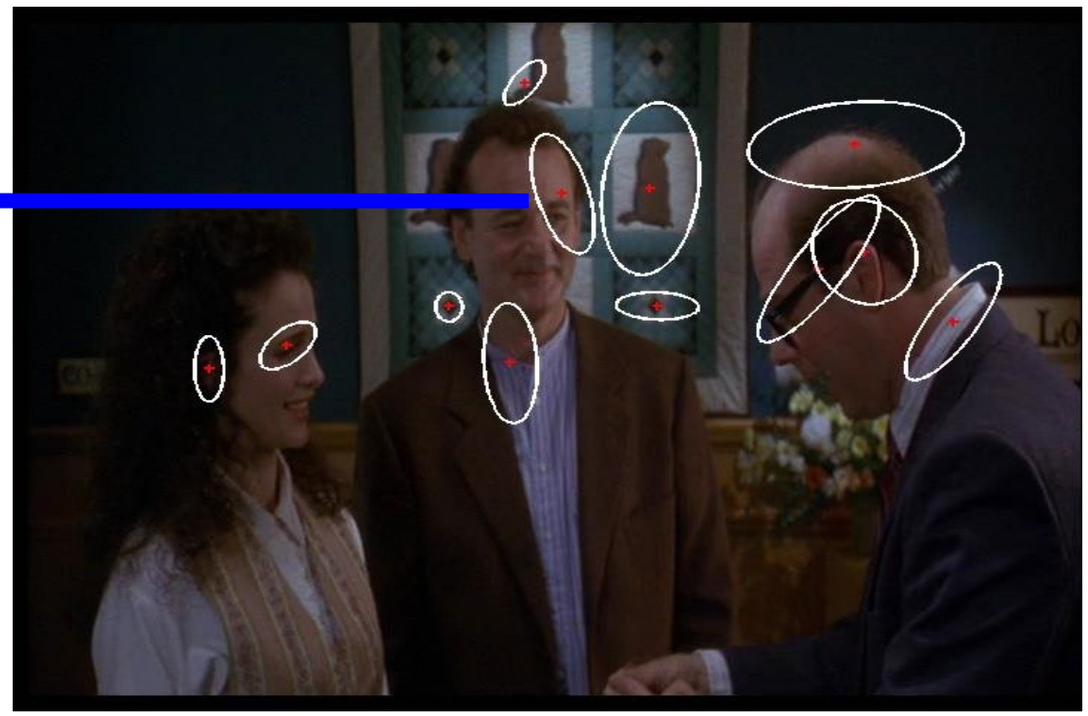


**Compute SIFT descriptor**

[Lowe'99]



**Normalize patch**

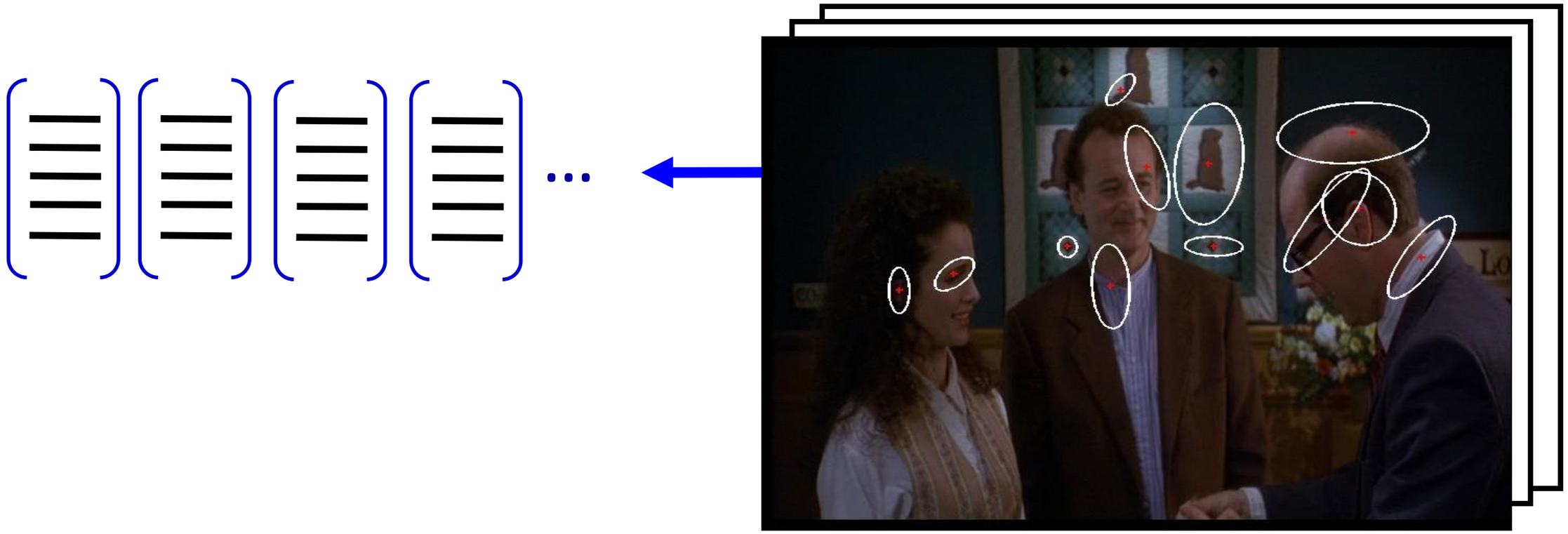


## Detect patches

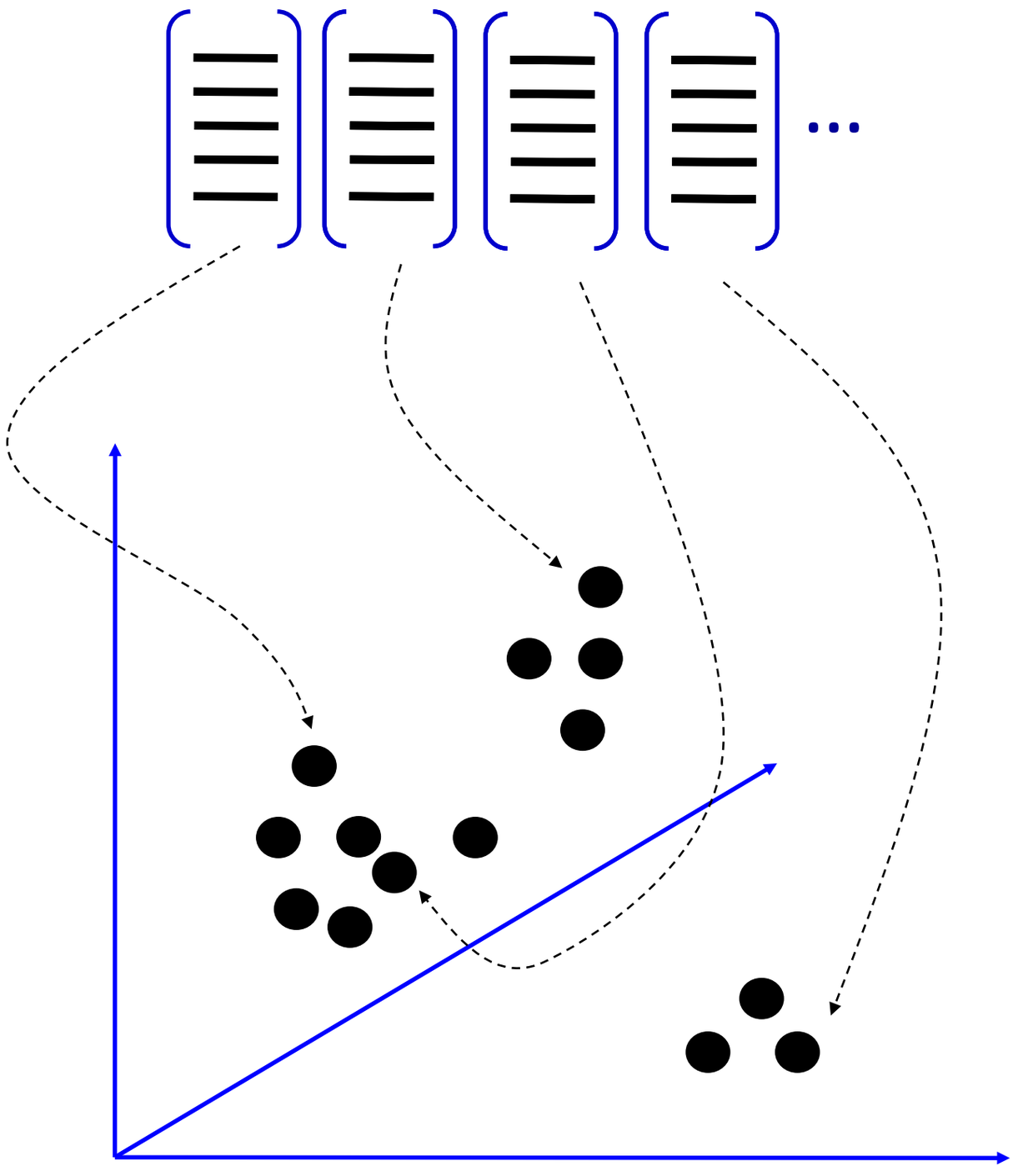
[Mikojaczyk and Schmid '02]

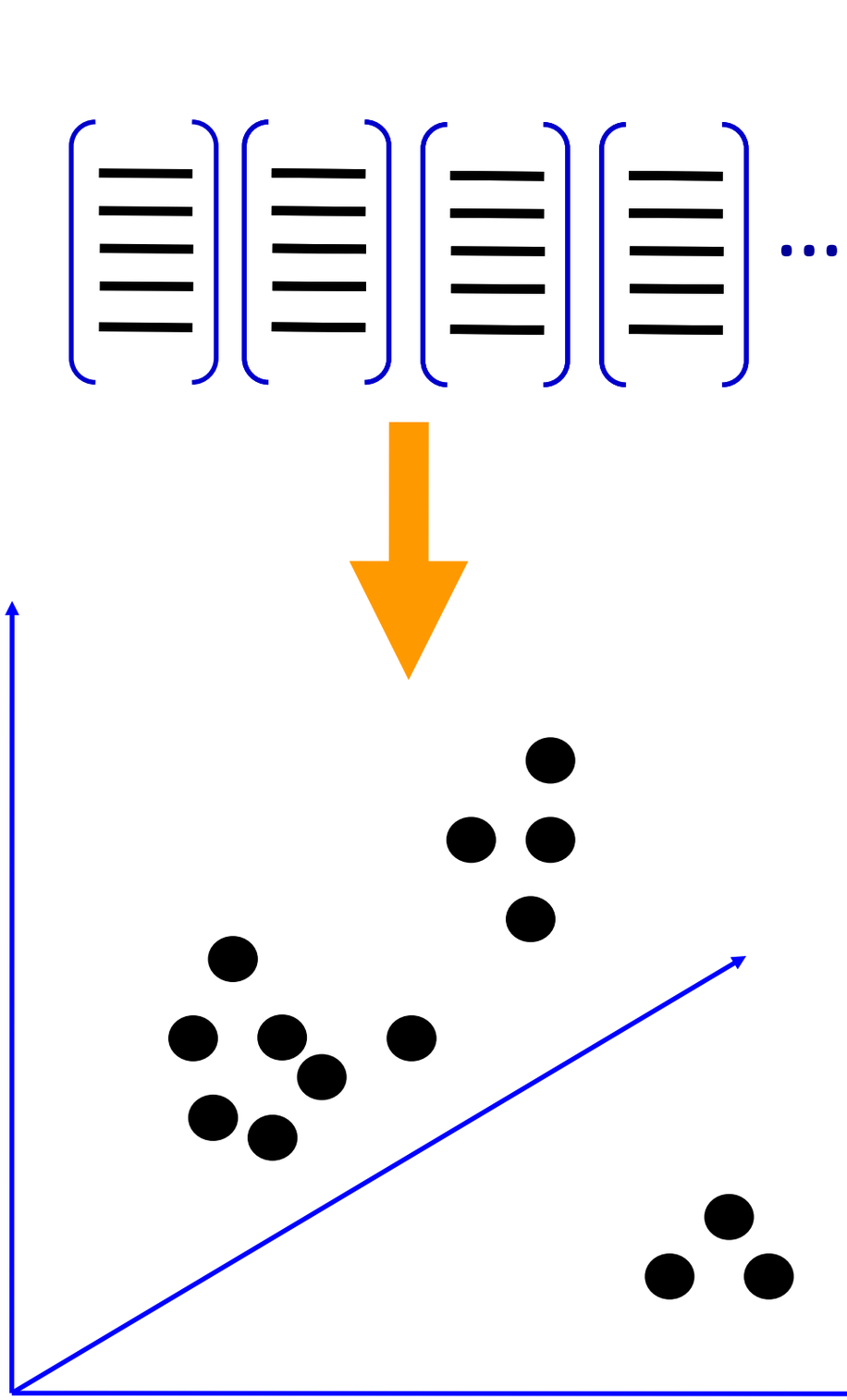
[Mata, Chum, Urban & Pajdla, '02]

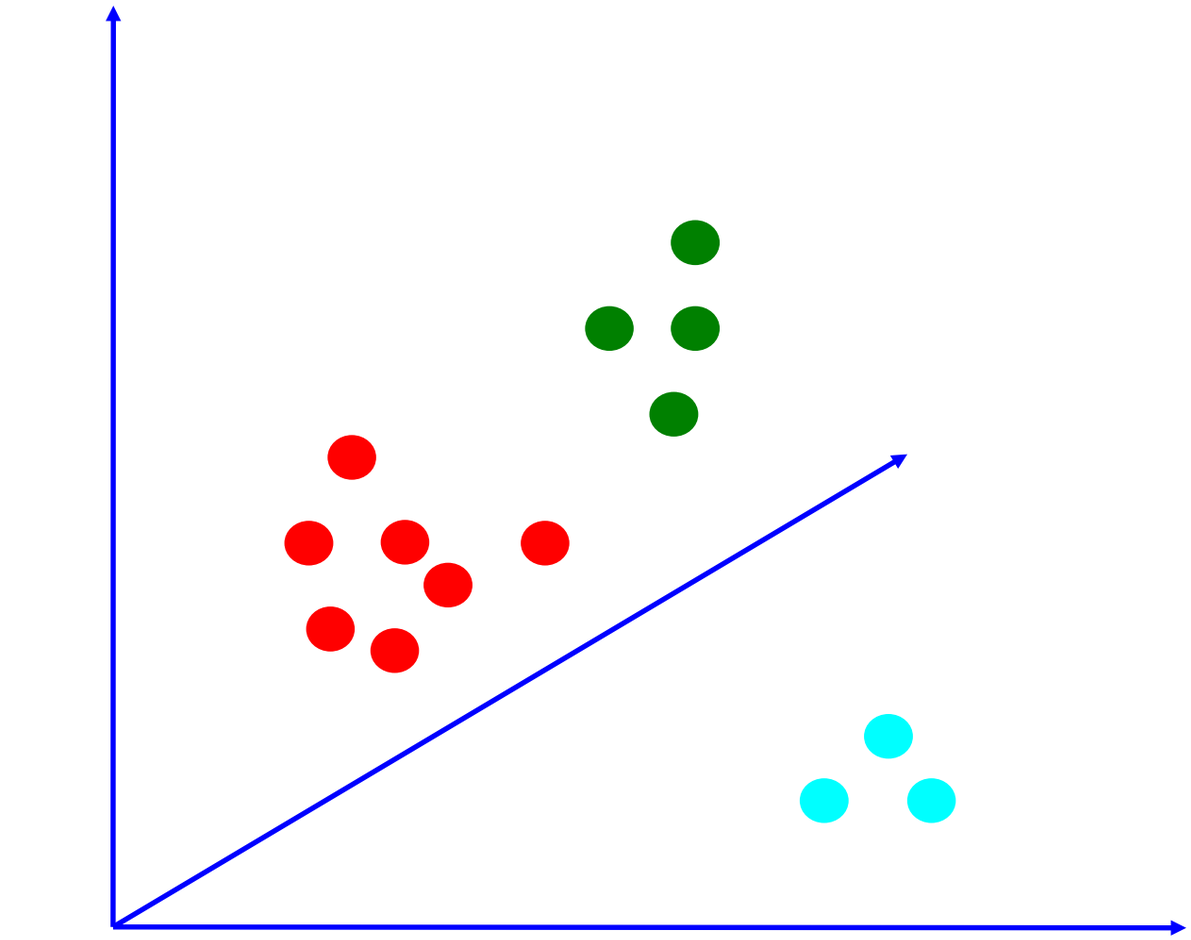
[Sivic & Zisserman, '03]



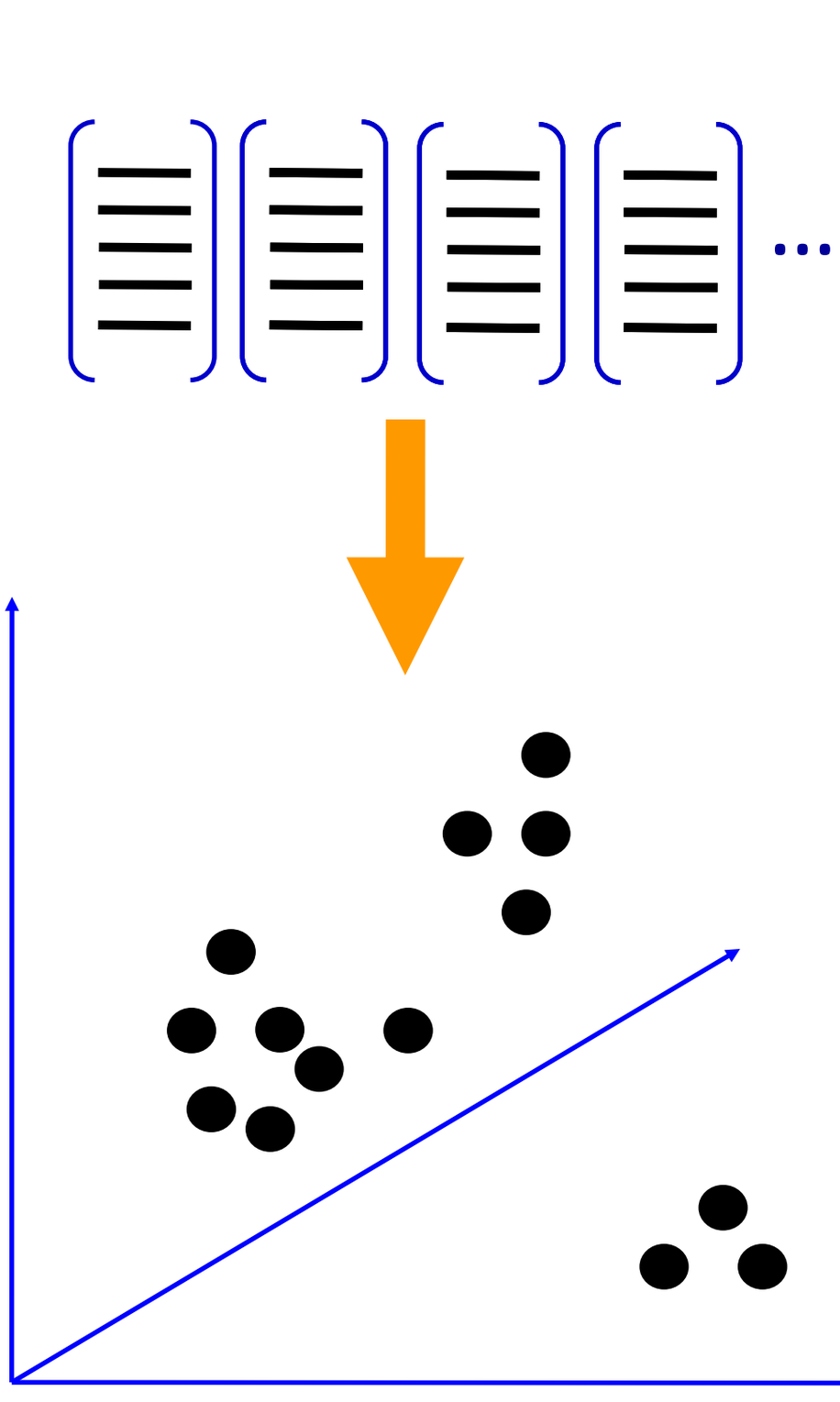
# *How do we learn the dictionary?*

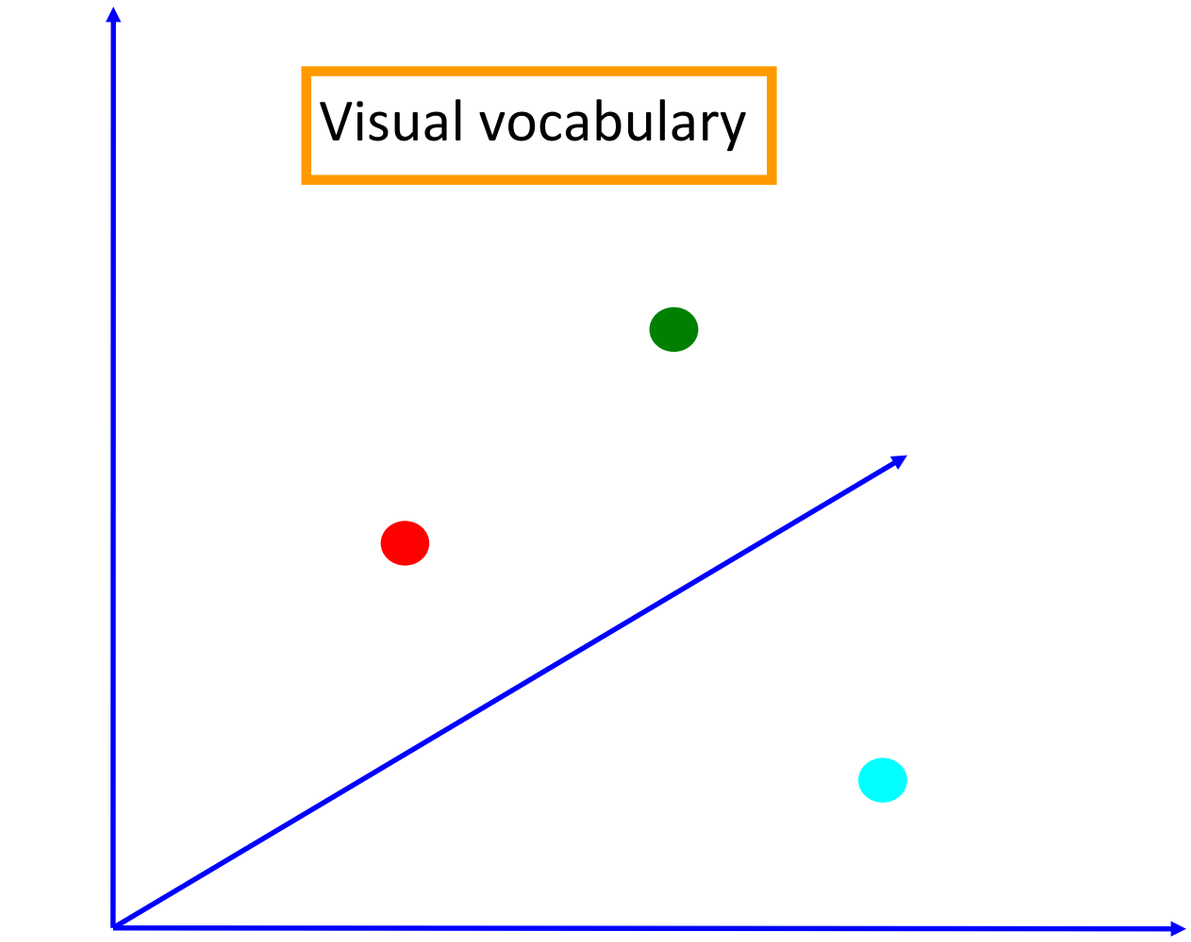






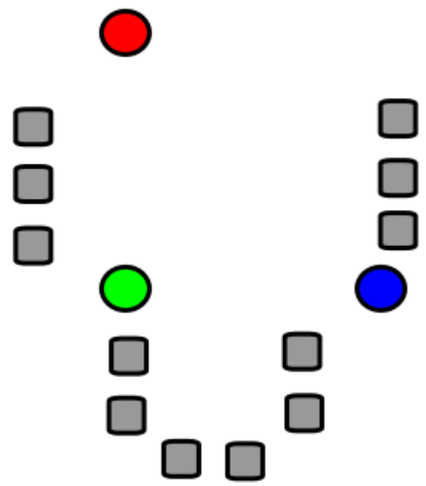
Clustering



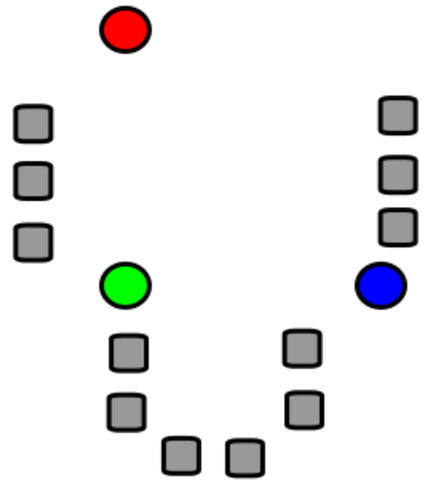

Visual vocabulary


Clustering

# K-means clustering

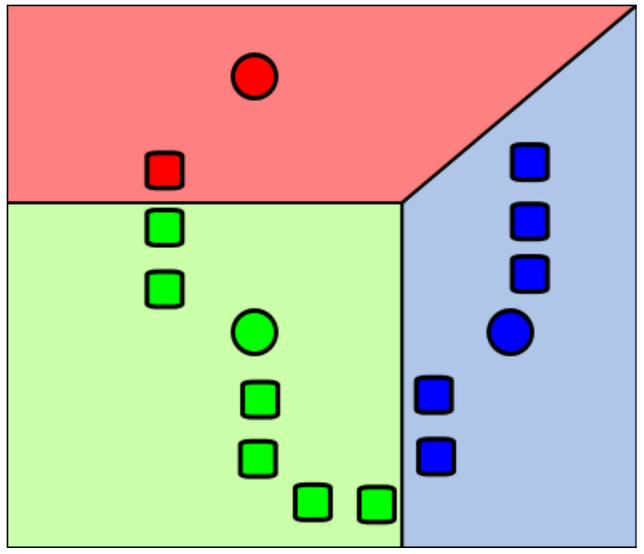


1. Select initial centroids at random

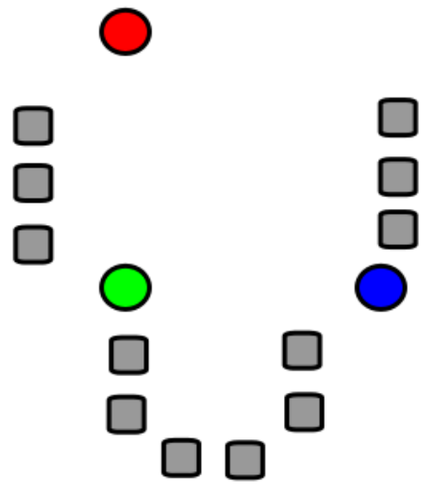


1. Select initial centroids at random

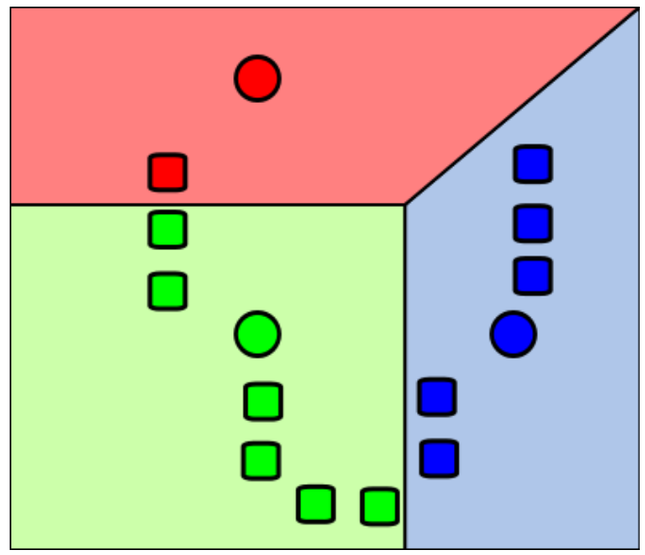




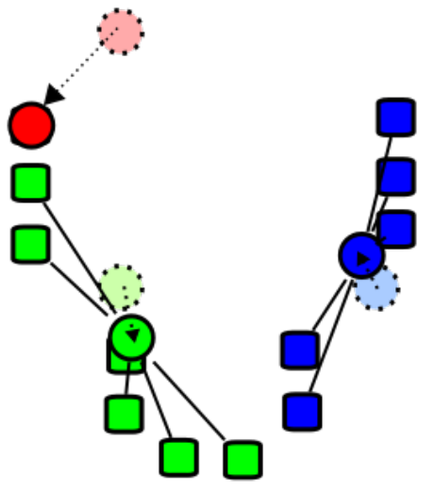
2. Assign each object to the cluster with the nearest centroid.



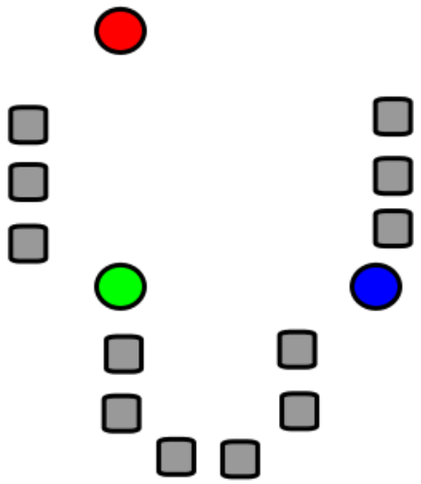
1. Select initial centroids at random



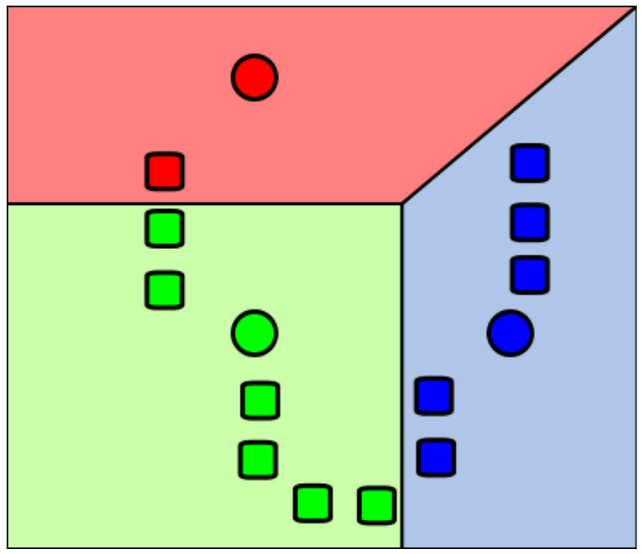
2. Assign each object to the cluster with the nearest centroid.



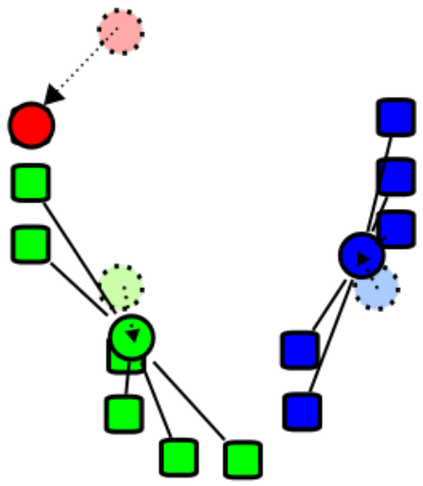
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



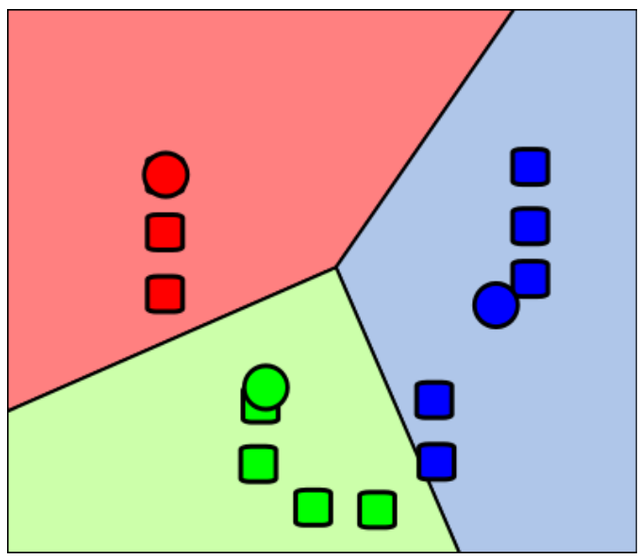
1. Select initial centroids at random



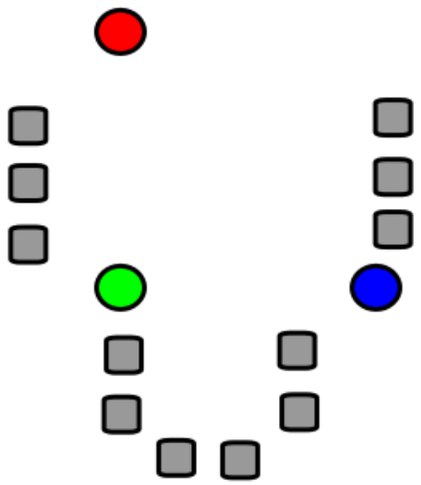
2. Assign each object to the cluster with the nearest centroid.



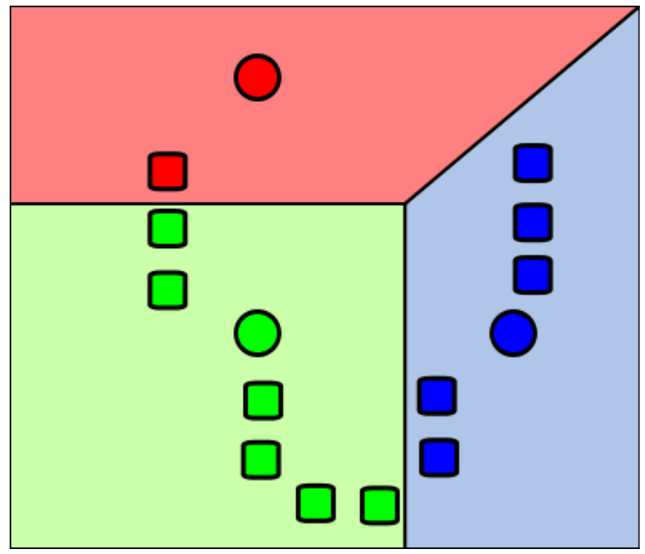
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



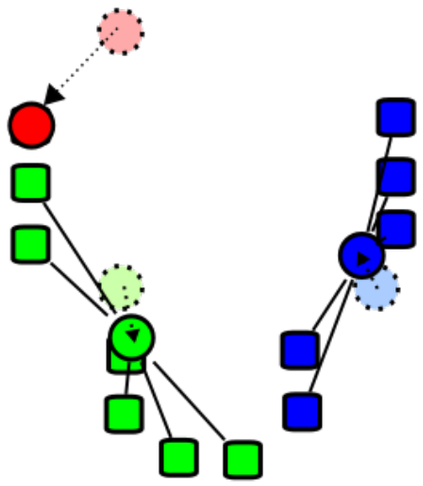
2. Assign each object to the cluster with the nearest centroid.



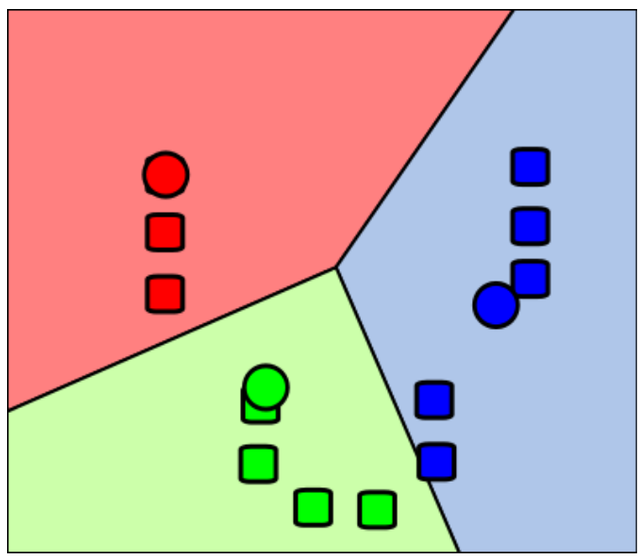
1. Select initial centroids at random



2. Assign each object to the cluster with the nearest centroid.



3. Compute each centroid as the mean of the objects assigned to it (go to 2)



2. Assign each object to the cluster with the nearest centroid.



Repeat previous 2 steps until no change

# K-means Clustering

Given k:

1.Select initial centroids at random.

2.Assign each object to the cluster with the nearest centroid.

3.Compute each centroid as the mean of the objects assigned to it.

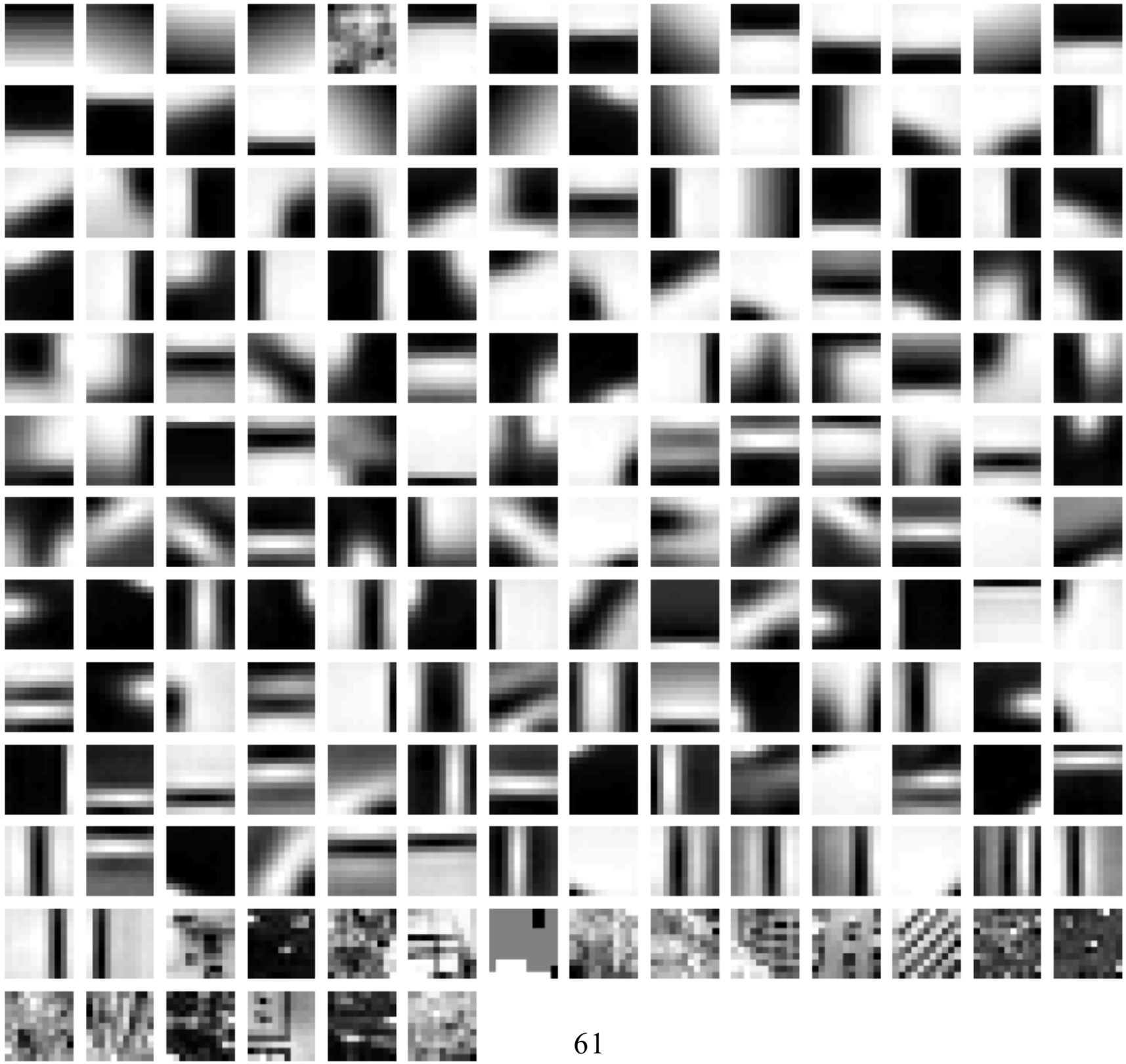
4.Repeat previous 2 steps until no change.

*From what **data** should I learn the dictionary?*

*From what **data** should I learn the dictionary?*

- Dictionary can be learned on separate training set
- Provided the training set is sufficiently representative, the dictionary will be “universal”

# Example visual dictionary



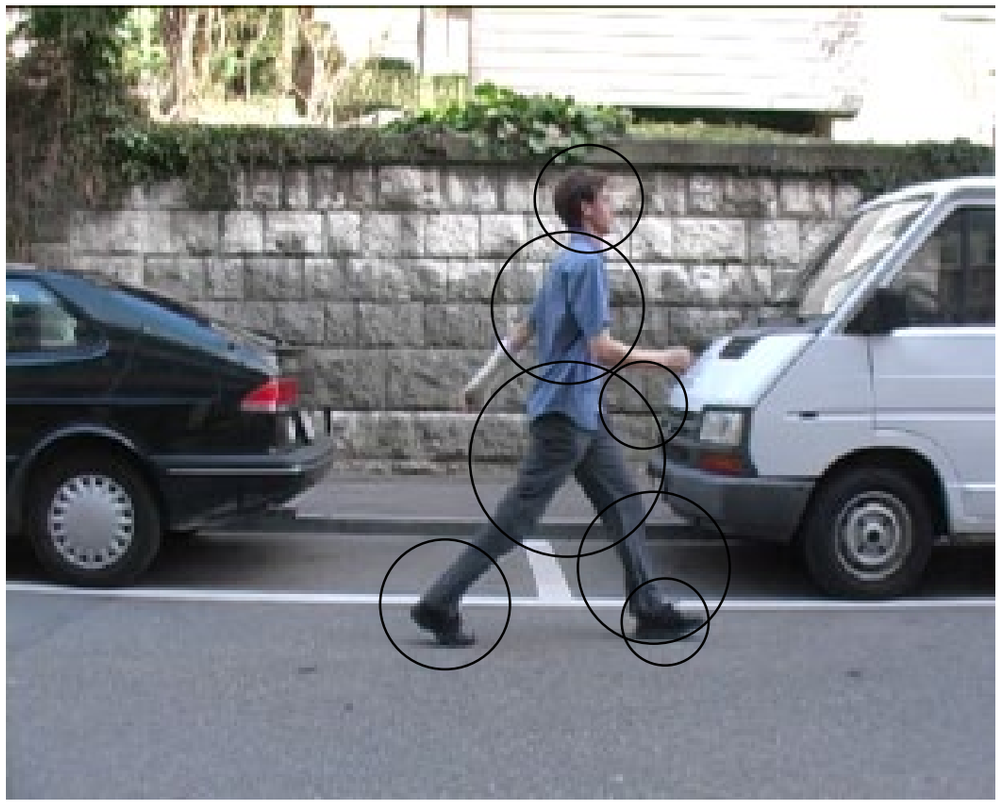
# Example dictionary


Appearance codebook



Source: B. Leibe

# Another dictionary




Appearance codebook



Source: B. Leibe

**Dictionary Learning:**

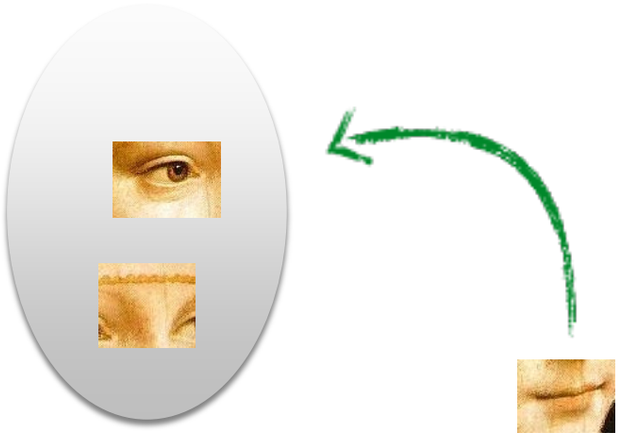
Learn Visual Words using clustering

**Encode:**

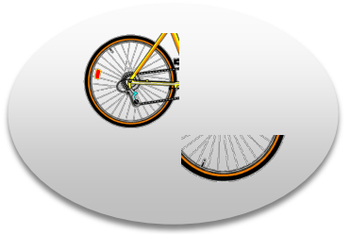
build Bags-of-Words (BOW) vectors for each image

**Classify:**

Train and test data using BOWs

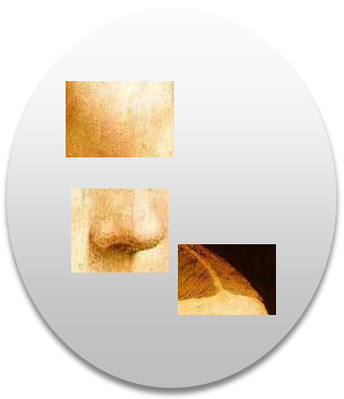


1. Quantization: image features gets associated to a visual word (nearest cluster center)



## Encode:

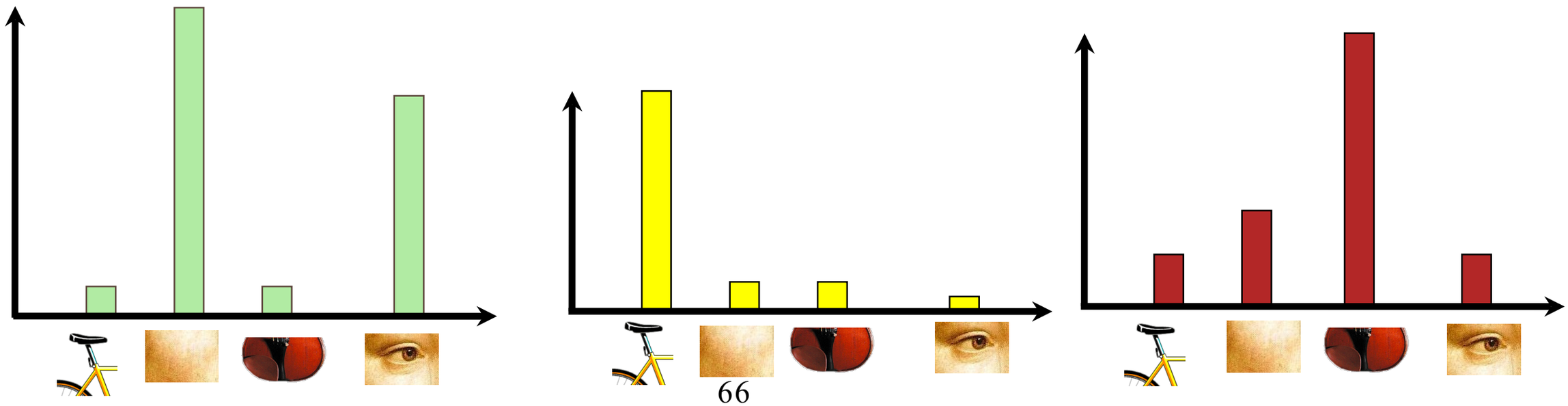
build Bags-of-Words (BOW) vectors for each image

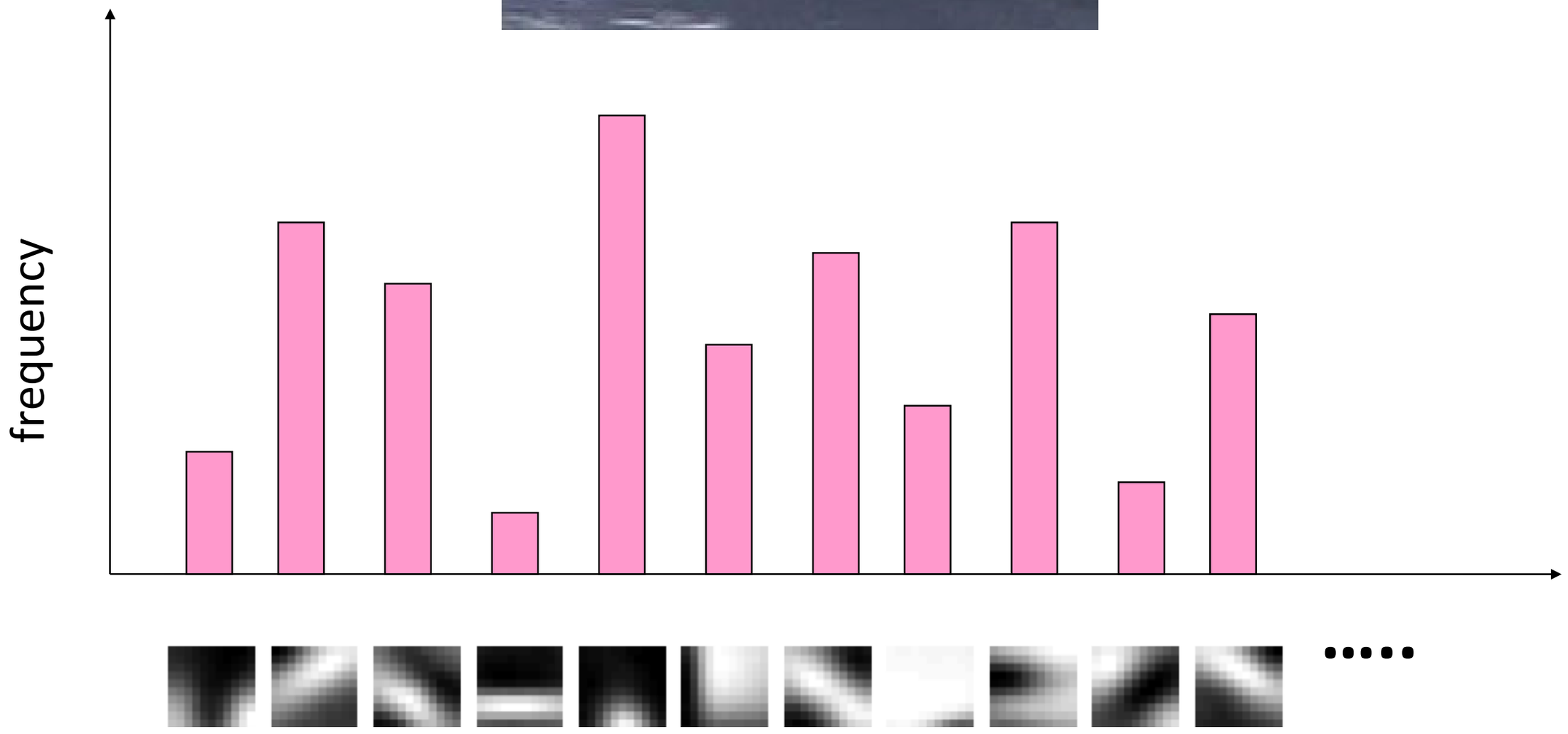


# Encode:

build Bags-of-Words (BOW) vectors for each image

2. Histogram: count the number of visual word occurrences




frequency
codewords

**Dictionary Learning:**

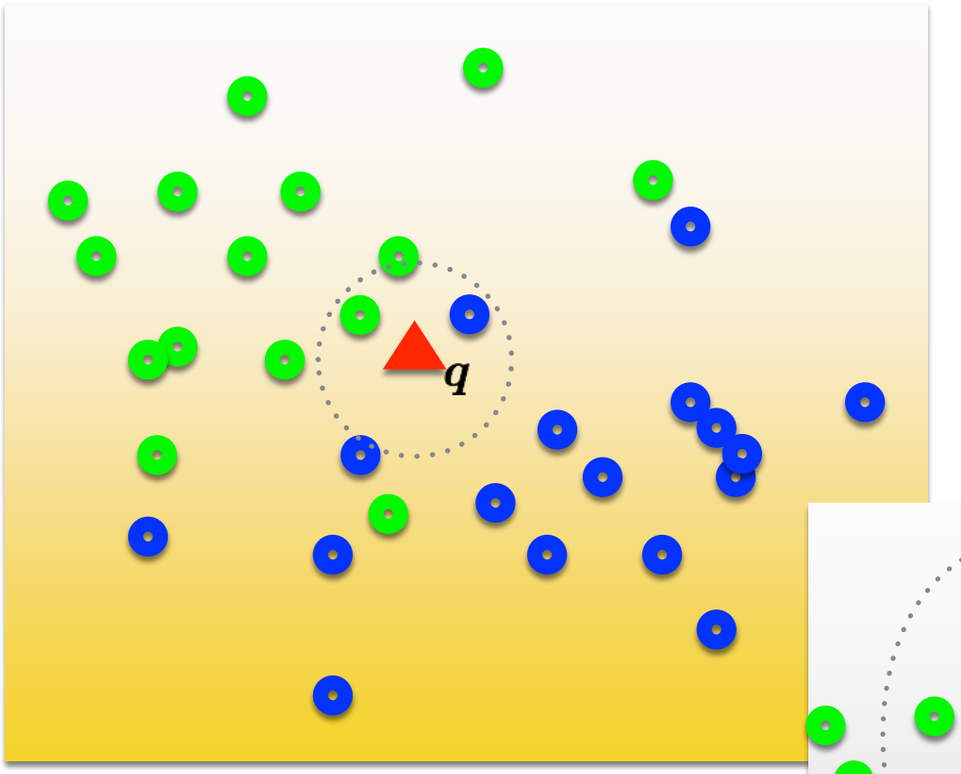
Learn Visual Words using clustering

**Encode:**

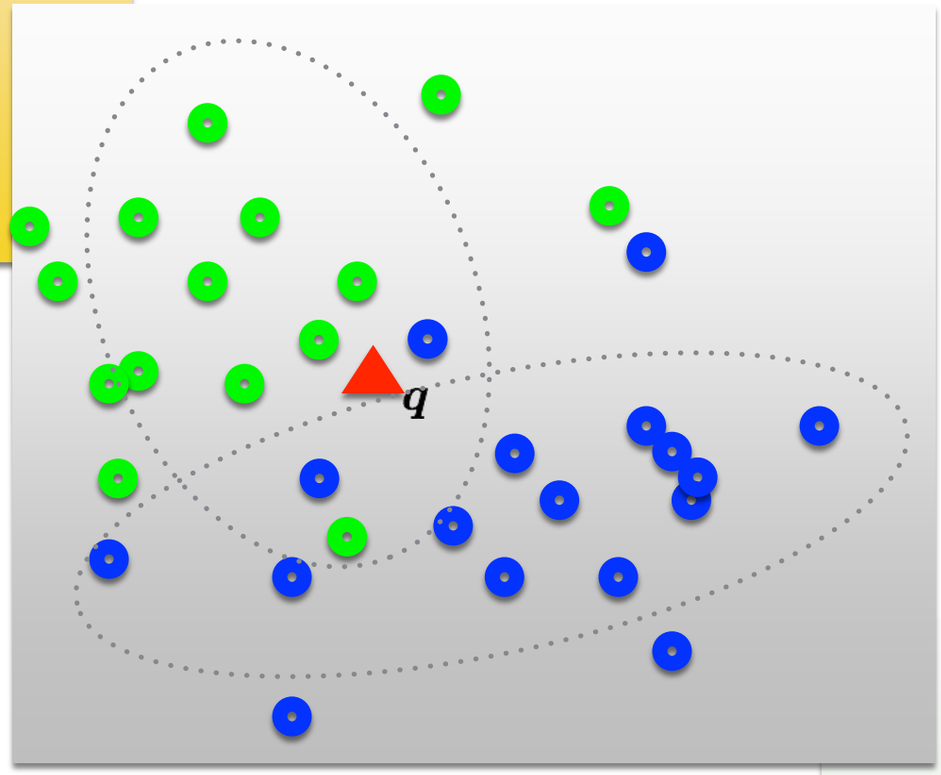
build Bags-of-Words (BOW) vectors for each image

**Classify:**

Train and test data using BOWs

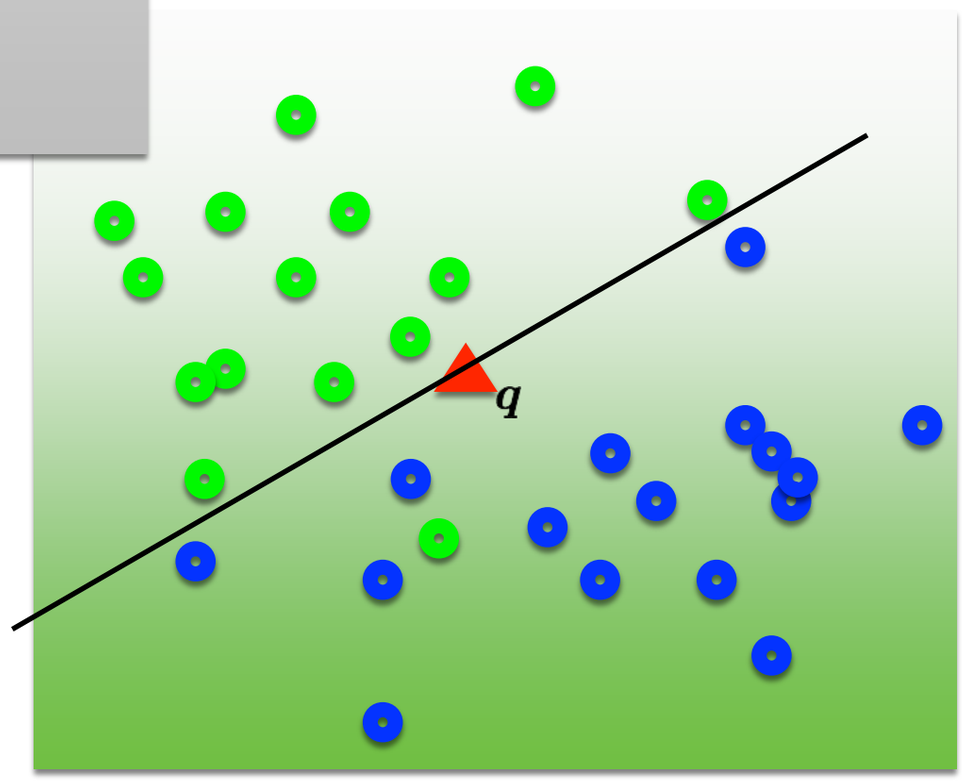



K nearest neighbors



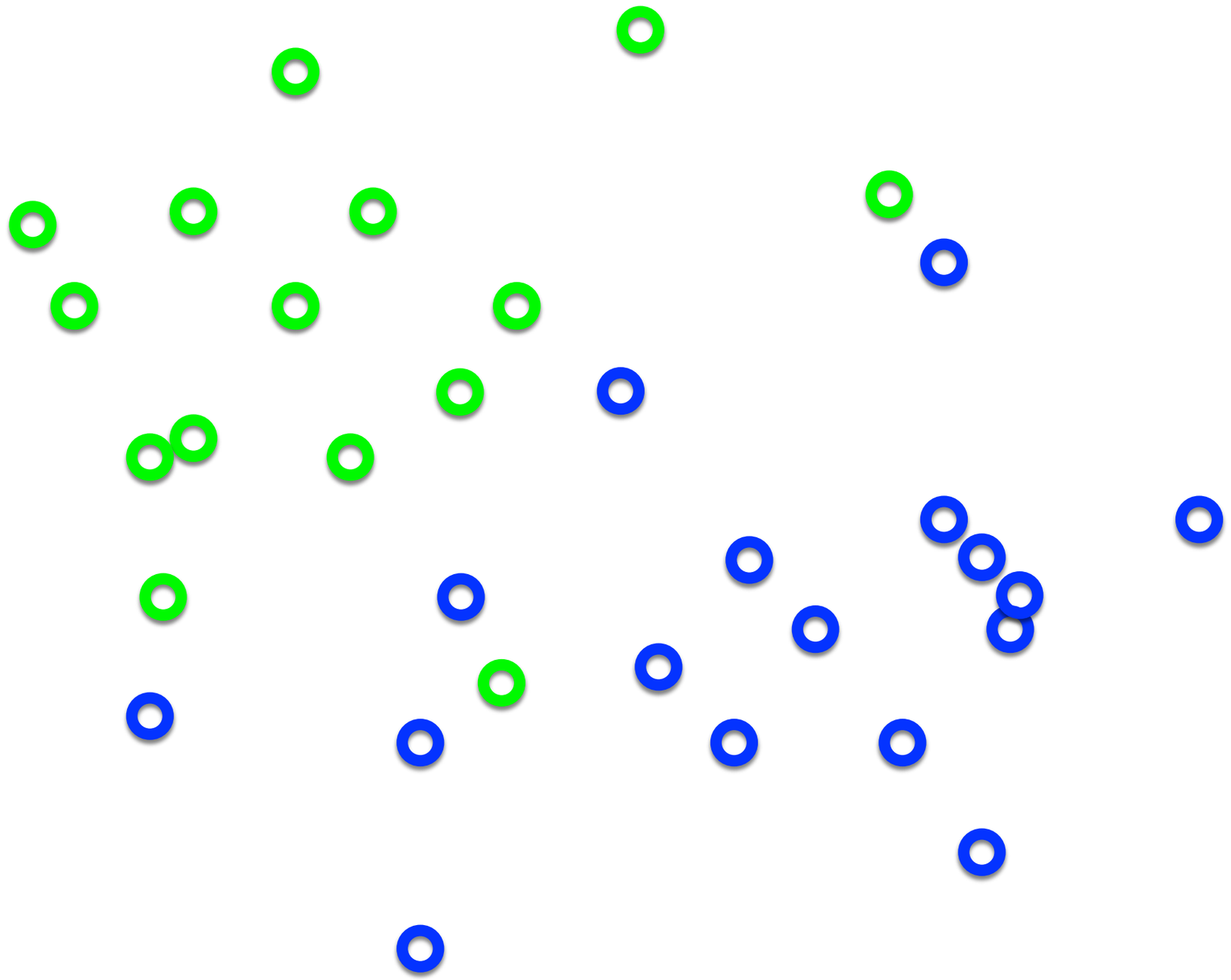

Naïve Bayes

Support Vector Machine

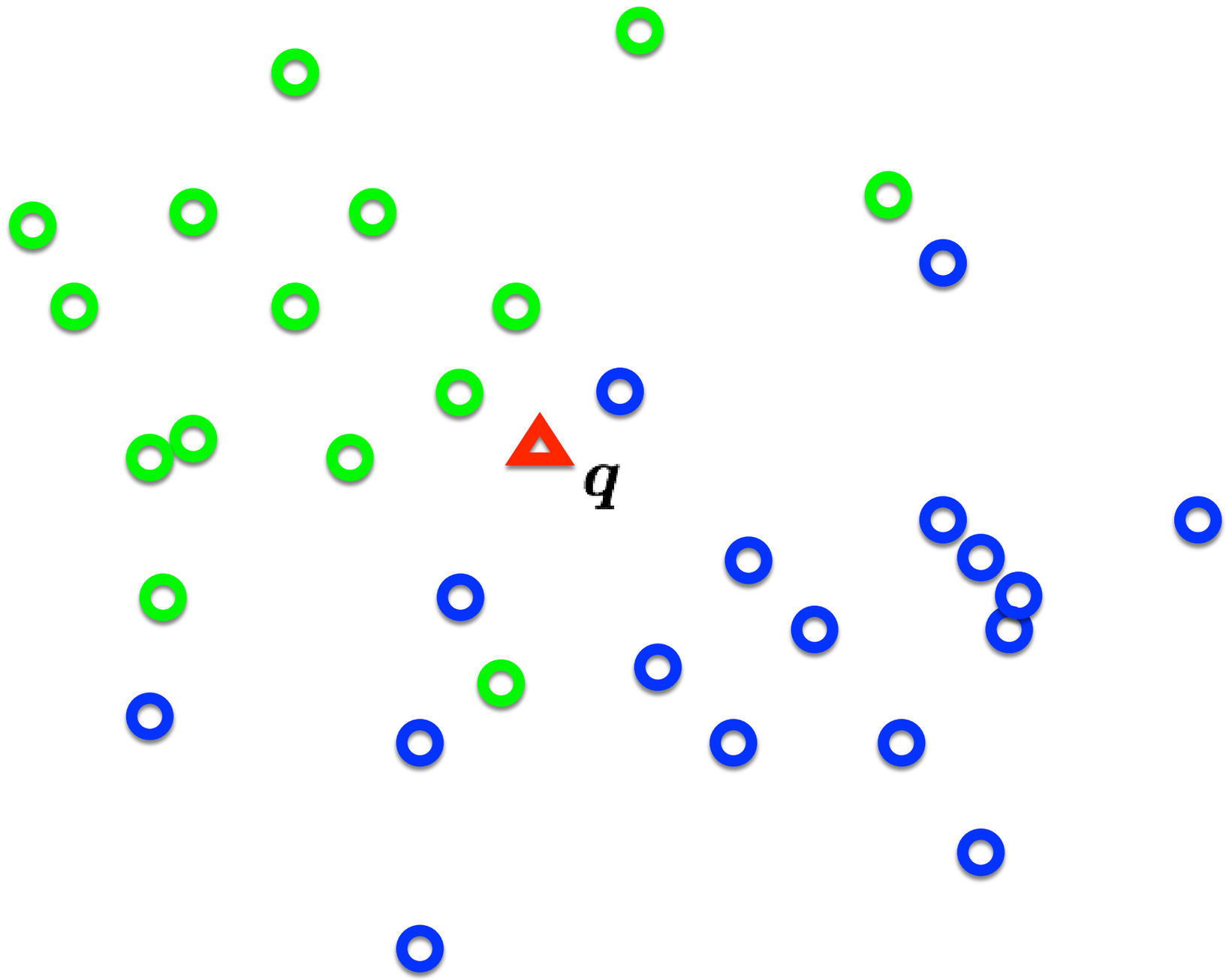


# K nearest neighbors

# Distribution of data from two classes

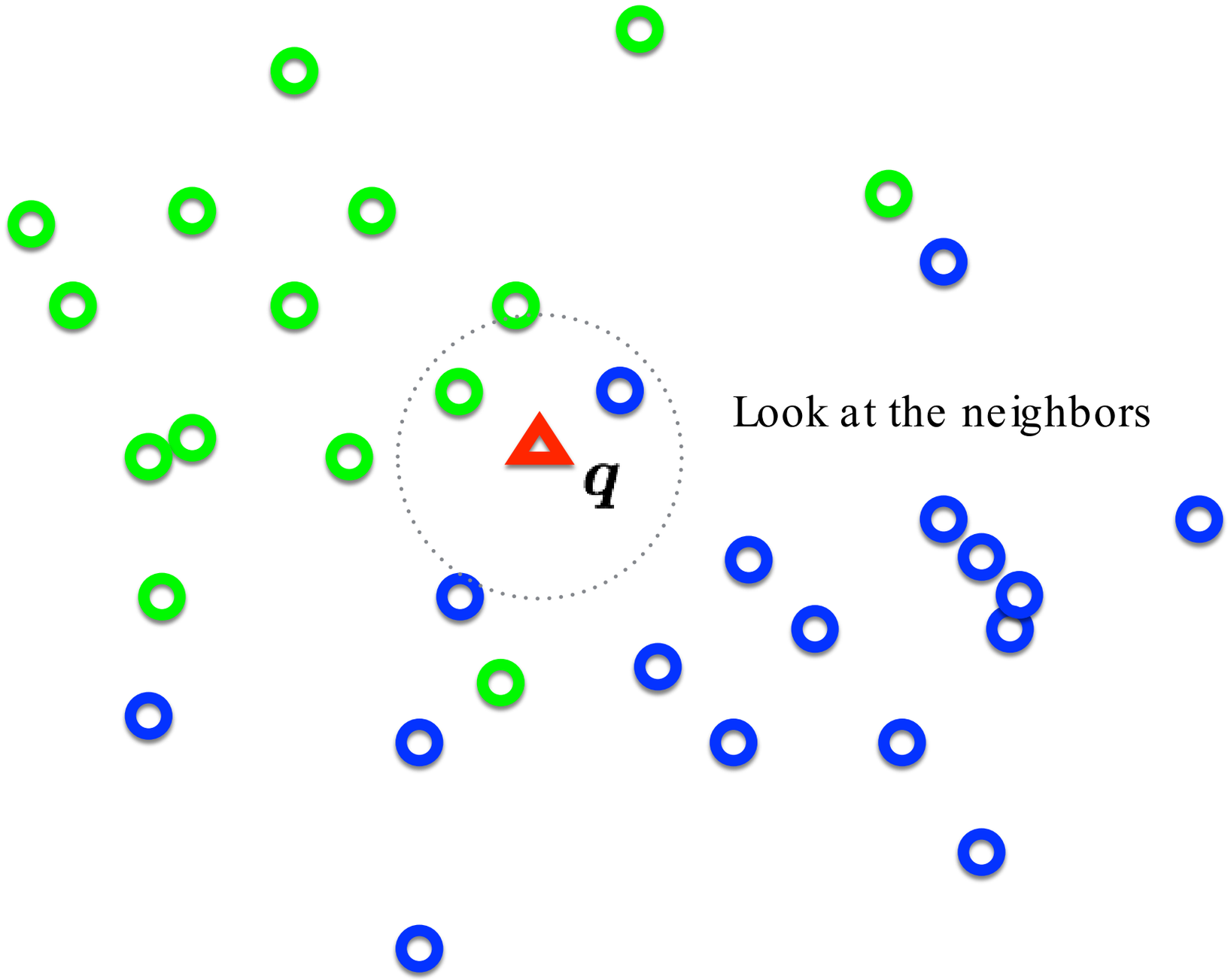


# Distribution of data from two classes

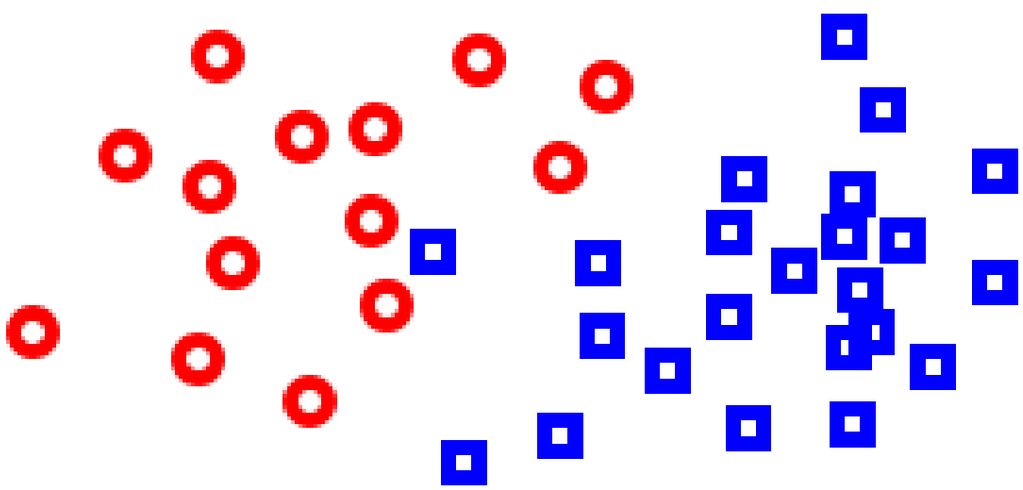



*Which class does q belong too?*

# Distribution of data from two classes



# K-Nearest Neighbor (KNN) Classifier

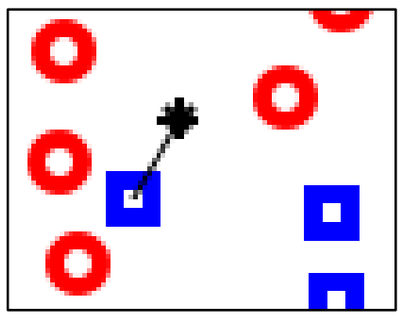


Non-parametric pattern classification approach

Consider a two class problem where each sample consists of two measurements (x,y).

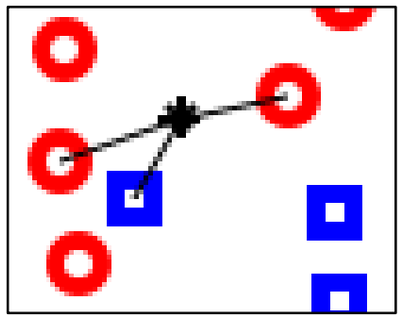
For a given query point q, assign the class of the nearest neighbor

k = 1



Compute the k nearest neighbors and assign the class by majority vote.

k = 3



# Nearest Neighbor is competitive

**MNIST Digit Recognition**

- Handwritten digits
- 28x28 pixel images: d = 784
- 60,000 training samples
- 10,000 test samples

Yann LeCunn

| | Test Error Rate (%) |
|---|---|
| Linear classifier (1-layer NN) | 12.0 |
| K-nearest-neighbors, Euclidean | 5.0 |
| K-nearest-neighbors, Euclidean, deskewed | 2.4 |
| K-NN, Tangent Distance, 16x16 | 1.1 |
| K-NN, shape context matching | 0.67 |
| 1000 RBF + linear classifier | 3.6 |
| SVM deg 4 polynomial | 1.1 |
| 2-layer NN, 300 hidden units | 4.7 |
| 2-layer NN, 300 HU, [deskewing] | 1.6 |
| LeNet-5, [distortions] | 0.8 |
| Boosted LeNet-4, [distortions] | 0.7 |

## What is the best distance metric between data points?

- Typically Euclidean distance
- Locality sensitive distance metrics
- Important to normalize.
  Dimensions have different scales

## How many K?

- Typically k=1 is good
- Cross-validation (try different k!)

# Distance metrics

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_N - y_N)^2}$$ Euclidean

$$D(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|} = \frac{x_1 y_1 + \cdots + x_N y_N}{\sqrt{\sum_n x_n^2} \sqrt{\sum_n y_n^2}}$$ Cosine

$$D(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \sum_n \frac{(x_n - y_n)^2}{(x_n + y_n)}$$ Chi-squared

# Choice of distance metric

- Hyperparameter

L1 (Manhattan) distance

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

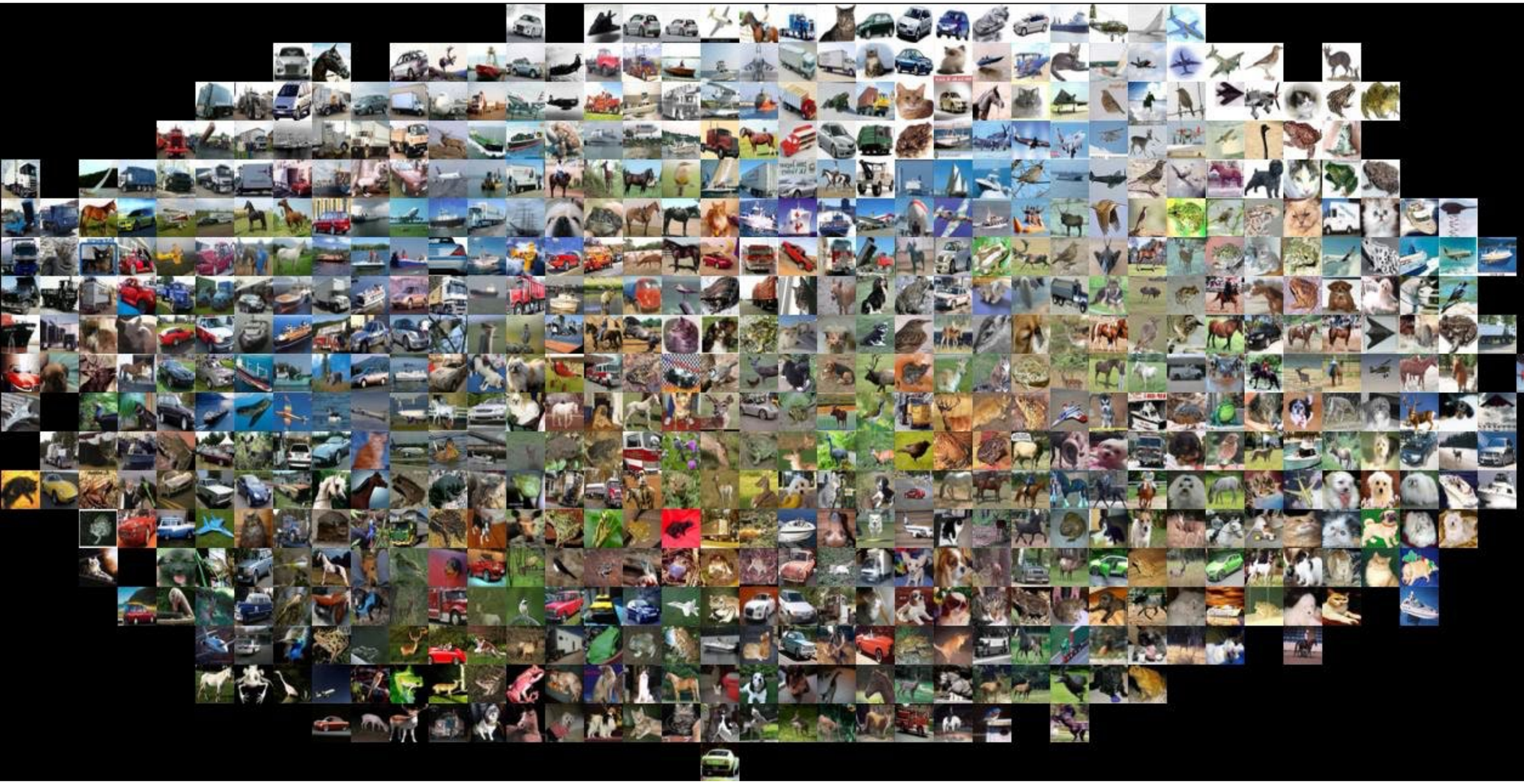
L2 (Euclidean) distance

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

- Two most commonly used special cases of p-norm

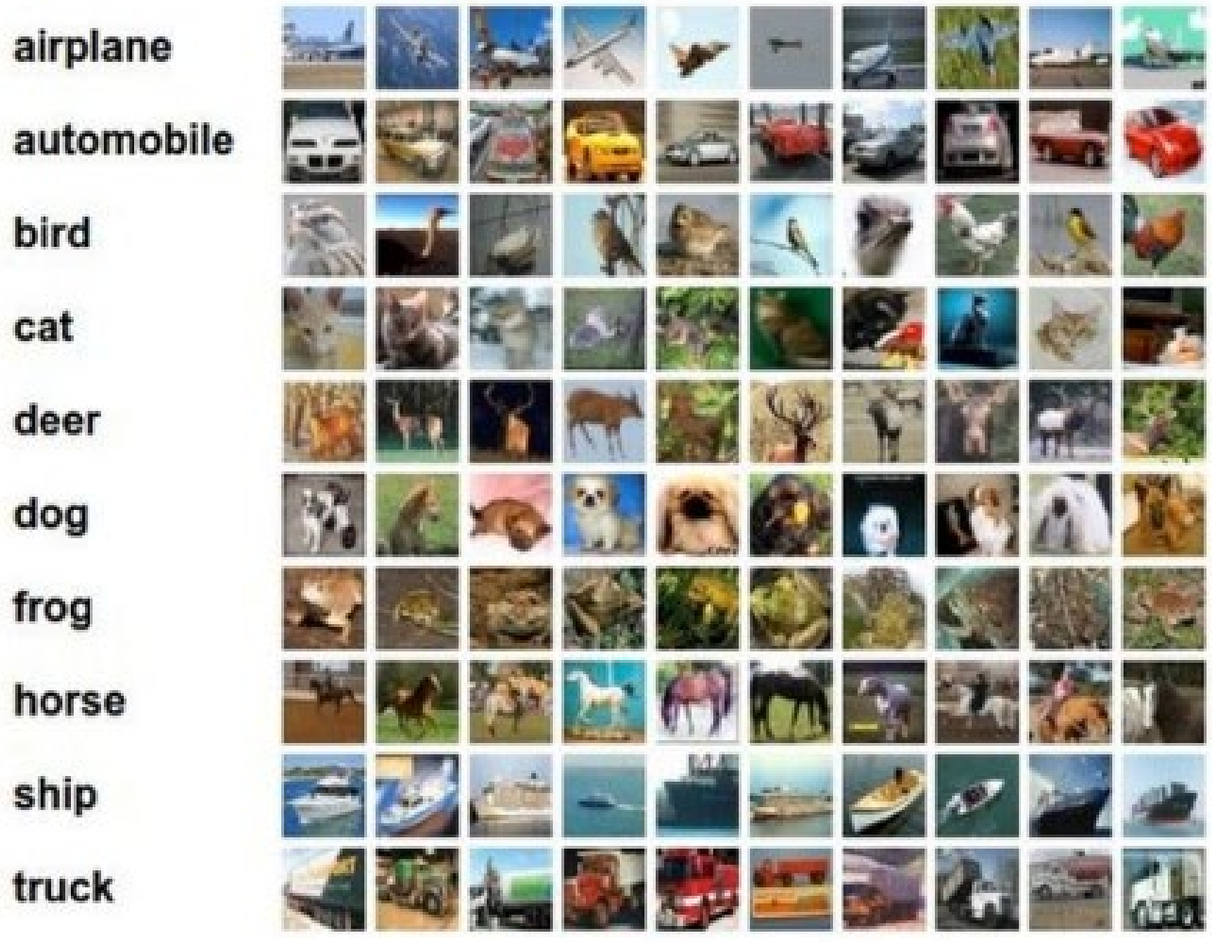
$$||x||_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \qquad p \geq 1, x \in \mathbb{R}^n$$

# Visualization: L2 distance



# CIFAR-10 and NN results

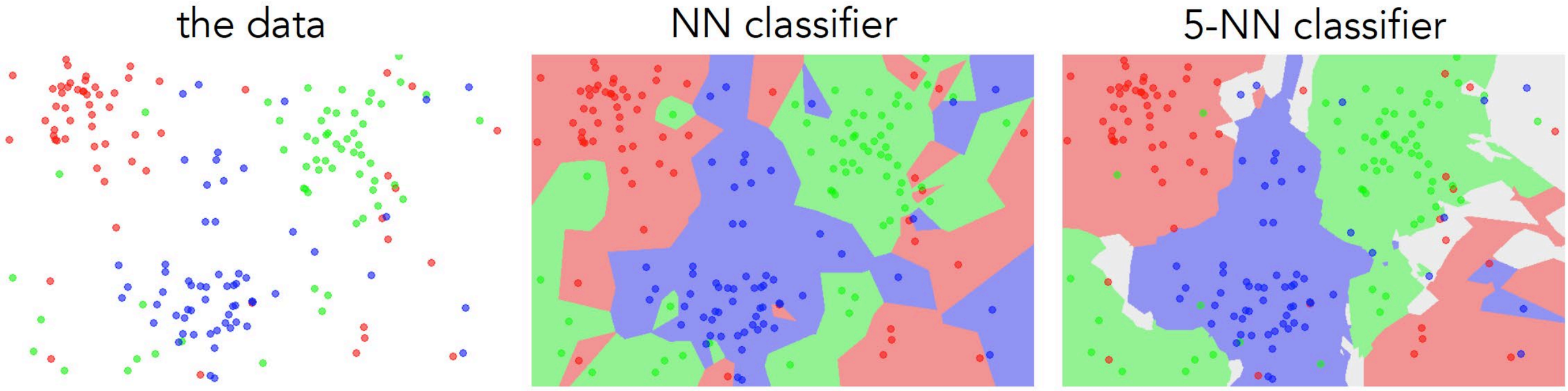
Example dataset: **CIFAR-10**
**10** labels
**50,000** training images
**10,000** test images.




For every test image (first column),
examples of nearest neighbors in rows

# k-nearest neighbor

- Find the k closest points from training data
- Labels of the k points “vote” to classify



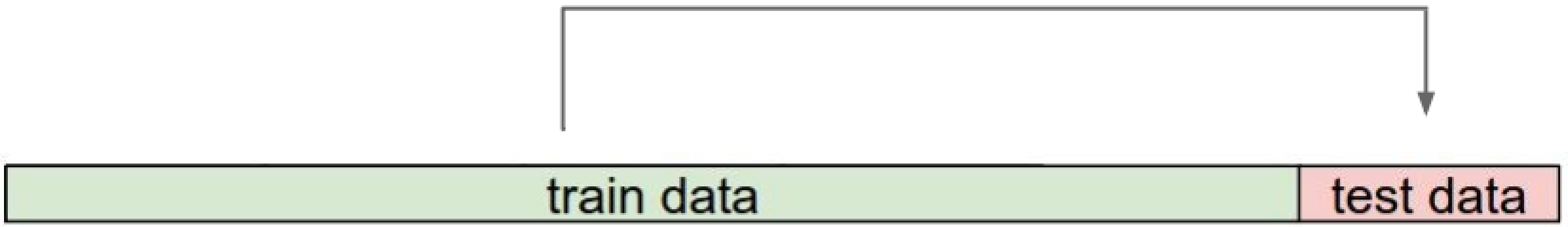
# Hyperparameters

- What is the best distance to use?
- What is the best value of k to use?

- i.e., how do we set the hyperparameters?

- Very problem-dependent
- Must try them all and see what works best

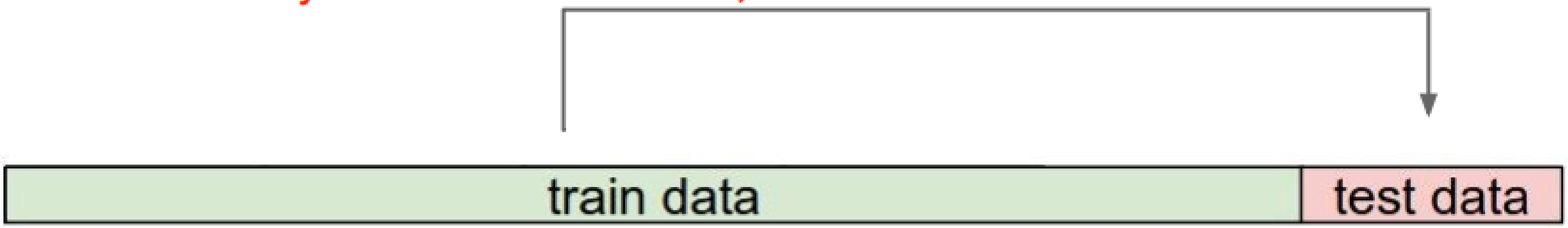
Try out what hyperparameters work best on test set.



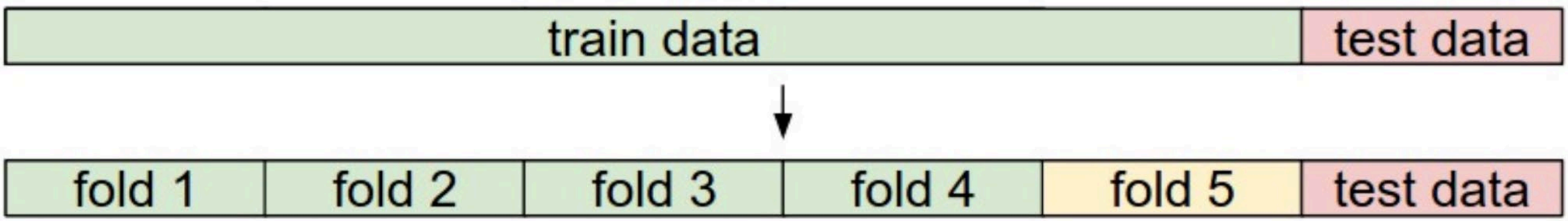
Trying out what hyperparameters work best on test set:

Very bad idea. The test set is a proxy for the generalization performance!

Use only **VERY SPARINGLY**, at the end.



# Validation

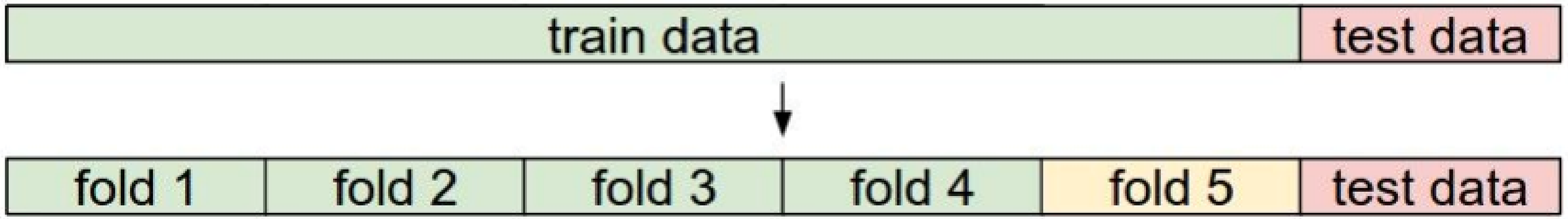



Validation data

use to tune hyperparameters

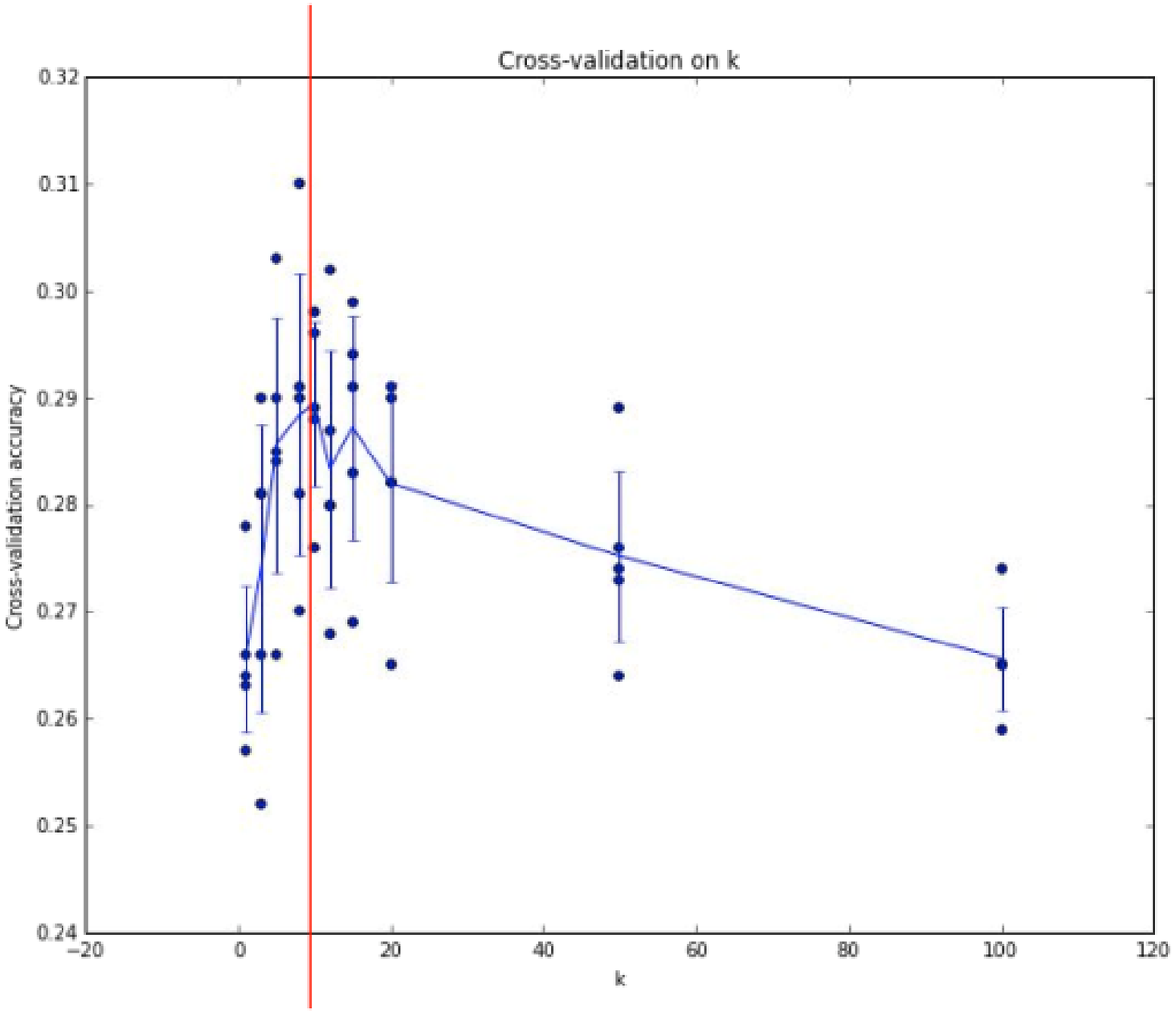
evaluate on test set ONCE at the end

# Cross-validation




**Cross-validation**

cycle through the choice of which fold is the validation fold, average results.




Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

# How to pick hyperparameters?

- Methodology
  - Train and test
  - Train, validate, test

- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

## Pros

- simple yet effective

## Cons

- search is expensive (can be sped-up)
- storage requirements
- difficulties with high-dimensional data

# Timeline of recognition

Before deep learning

- 1965-late 1980s: alignment, geometric primitives
- Early 1990s: invariants, appearance-based methods
- Mid-late 1990s: sliding window approaches
- Late 1990s: feature-based methods
- Early 2000s: parts-and-shape models
- 2003 – 2010: bags of features

After deep learning

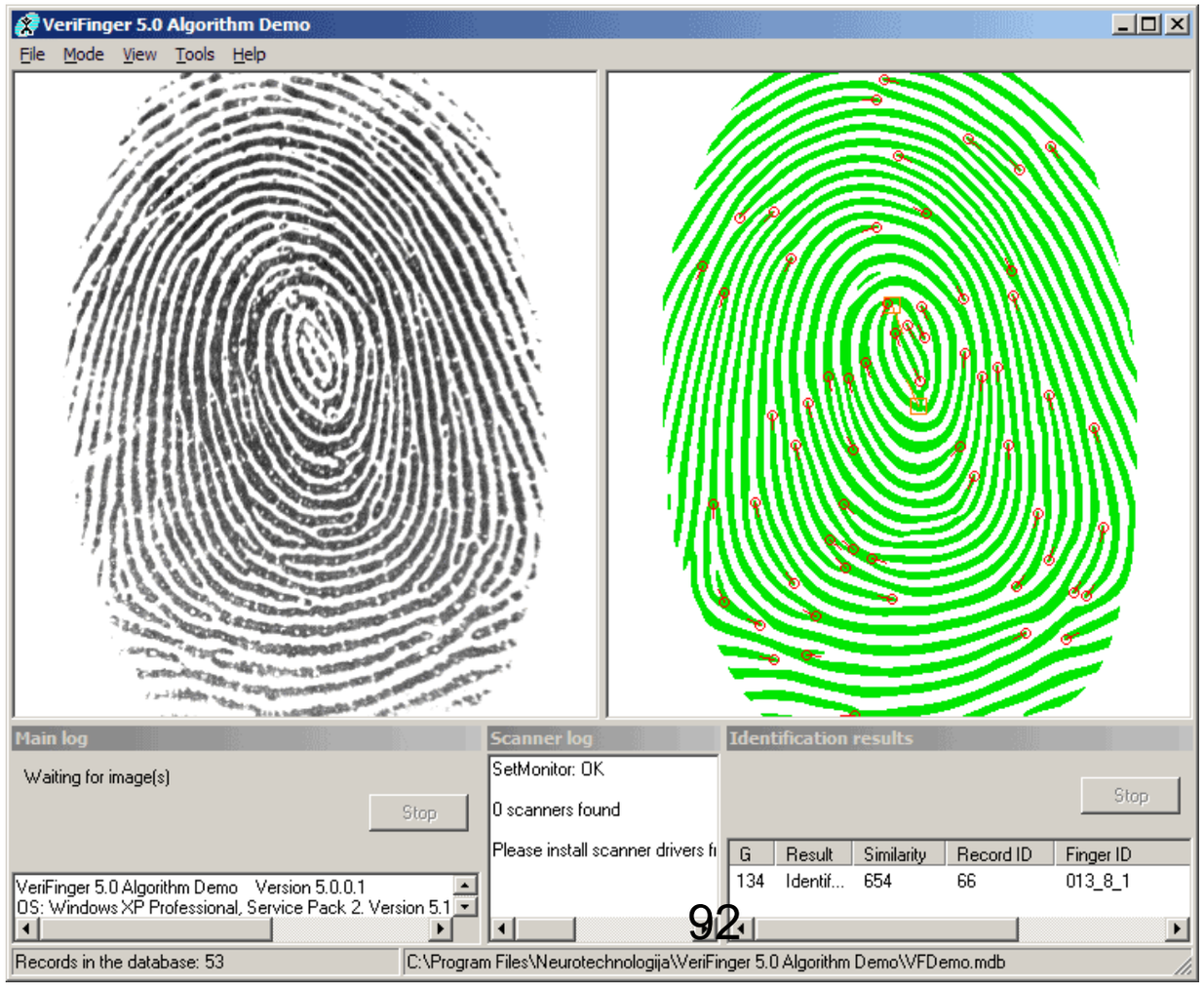
- Present trends: combination of local and global methods, modeling context, integrating recognition and segmentation, deep learning

# What “works” today

- Reading license plates, zip codes, checks

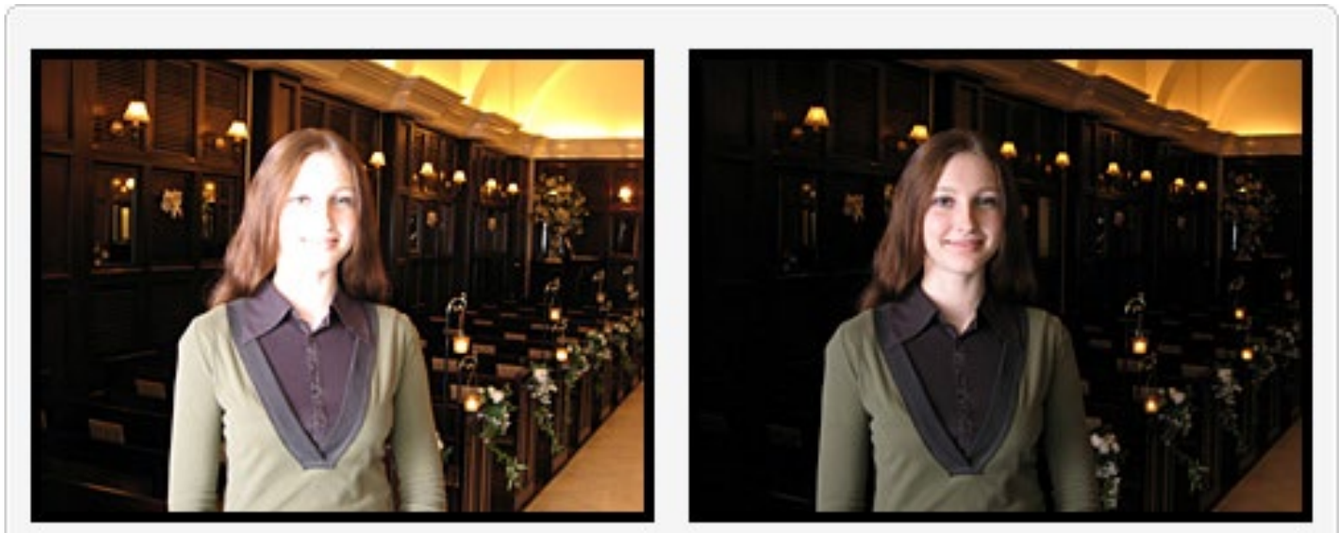
# What “works” today

- Reading license plates, zip codes, checks
- Fingerprint recognition



# What “works” today

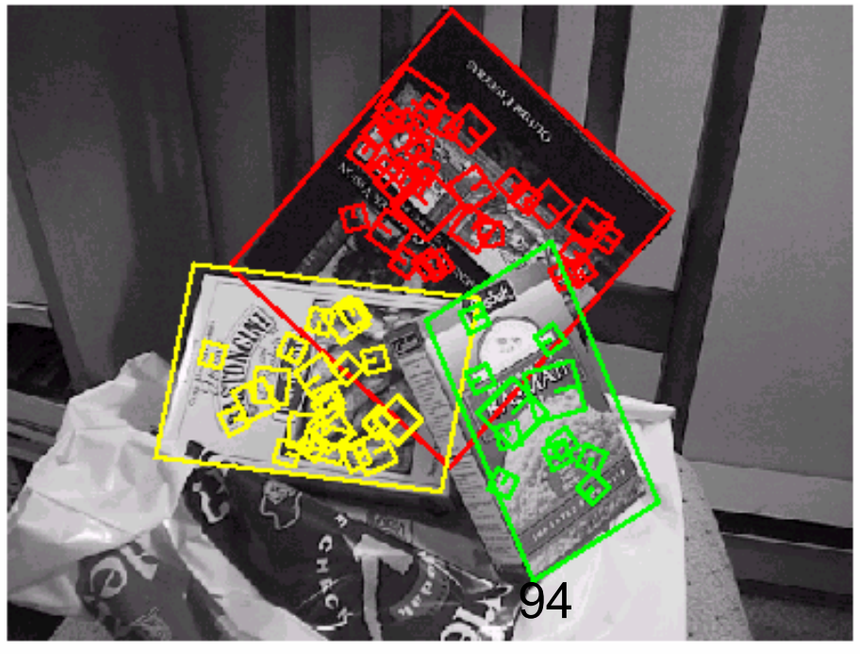
- Reading license plates, zip codes, checks
- Fingerprint recognition
- Face detection

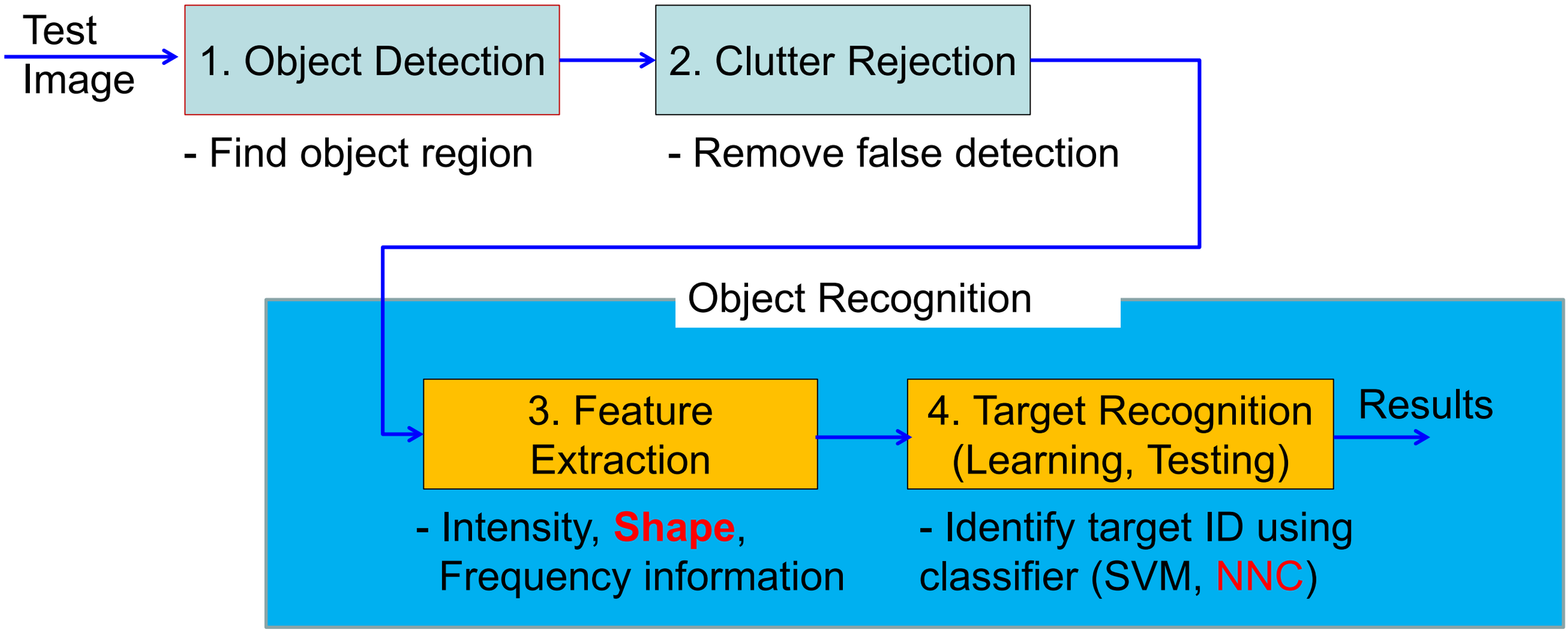
[Face priority AE] When a bright part of the face is too bright

# What “works” today

- Reading license plates, zip codes, checks
- Fingerprint recognition
- Face detection
- Recognition of flat textured objects (CD covers, book covers, etc.)



# Previous Object Detection/Recognition Flow



# What Matters in Recognition?

- Learning Techniques
  - E.g. choice of classifier or inference method
- Representation
  - Low level: SIFT, HoG, GIST, edges
  - Mid level: Bag of words, sliding window, deformable model
  - High level: Contextual dependence
  - **Deep learned features**
- **Data**
  - More is always better (as long as it is good data)
  - Annotation is the hard part