Lecture 15

Dimensionality reduction

Administrative

A4 is out

- Due May 23th

A5 out this week

Administrative

Recitation this friday

- Recognition review
- Jieyu Zhang

So far: visual recognition

- Apply a prediction function to a feature representation of the image to get the desired output:

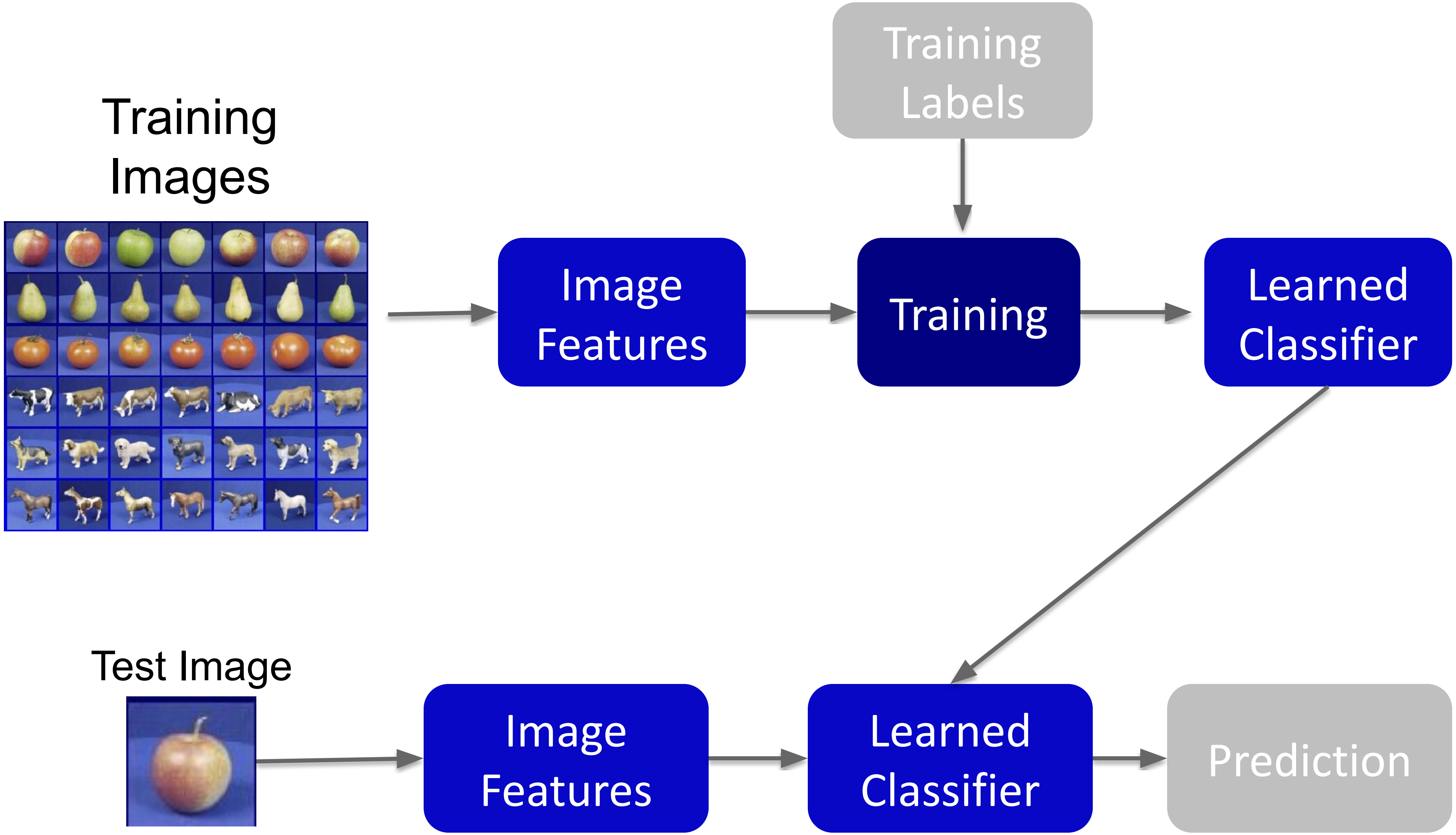
f() = "apple"

f() = "tomato"

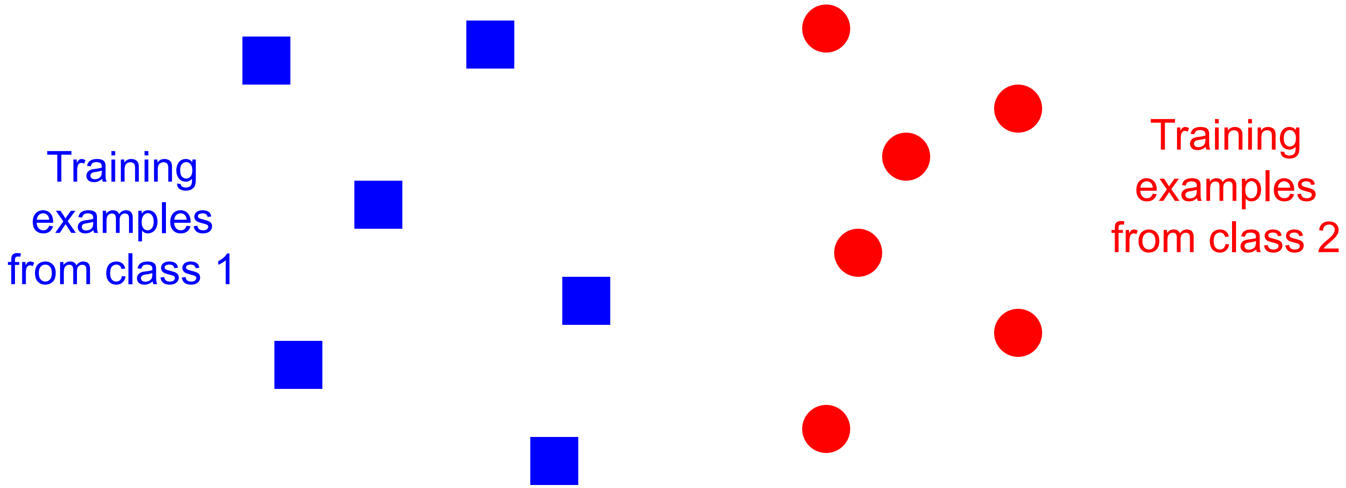
f() = "cow"

Slide credit: L. Lazebnik

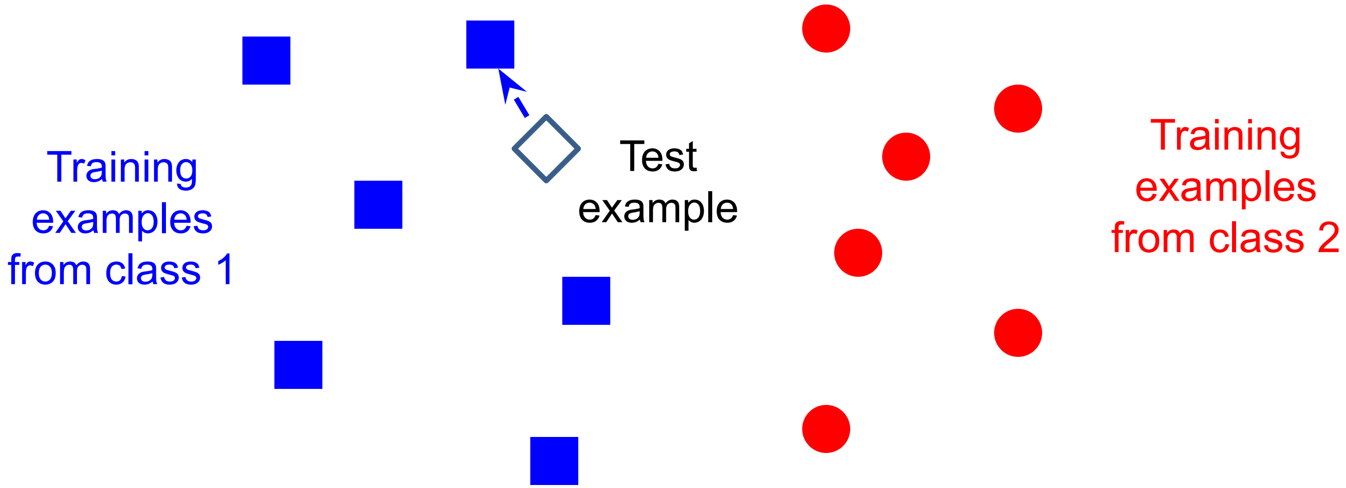
So far: A simple recognition pipeline



So far: (kNN) Nearest neighbor



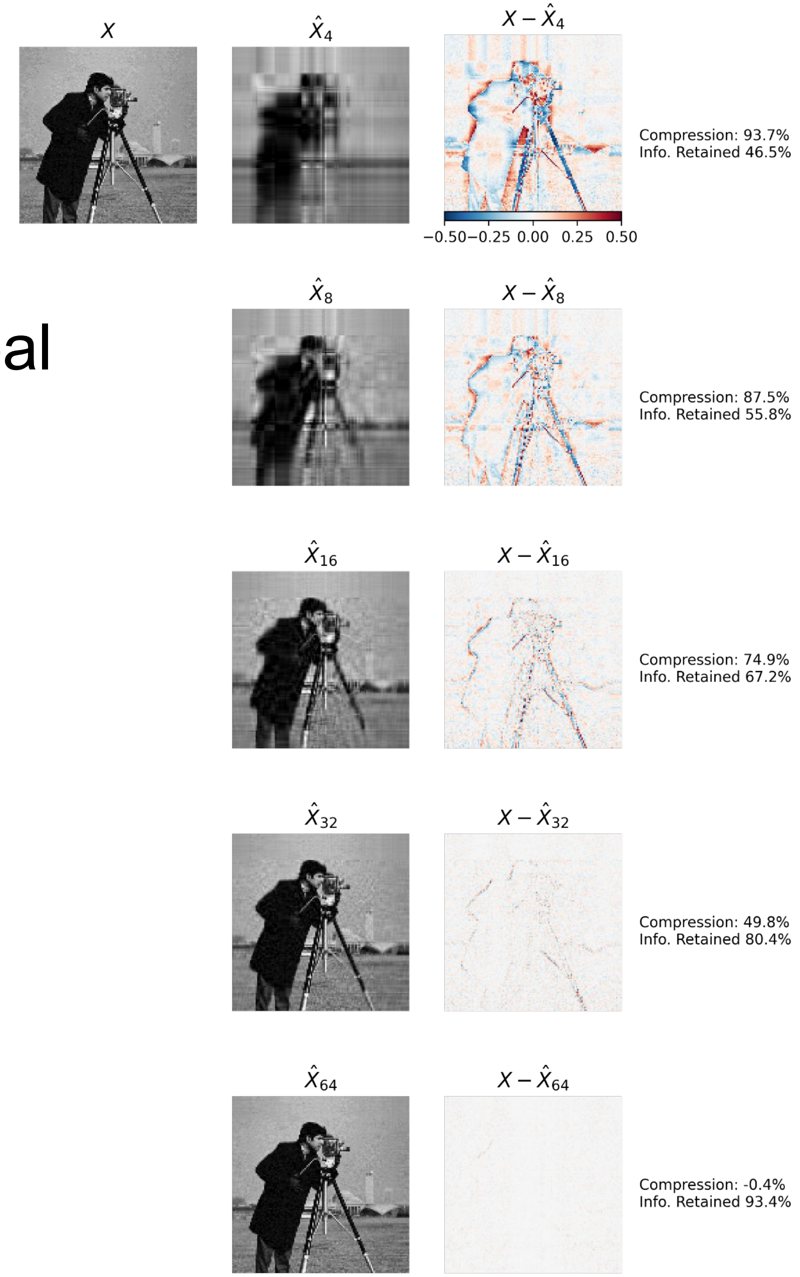
So far: (kNN) Nearest neighbor



Slide credit: L. Lazebnik

So far: Image compression with SVD

- For this image, using **only the first 16** of 300 principal components produces a recognizable reconstruction
- Using the first 64 almost perfectly reconstructs the image



Today's agenda

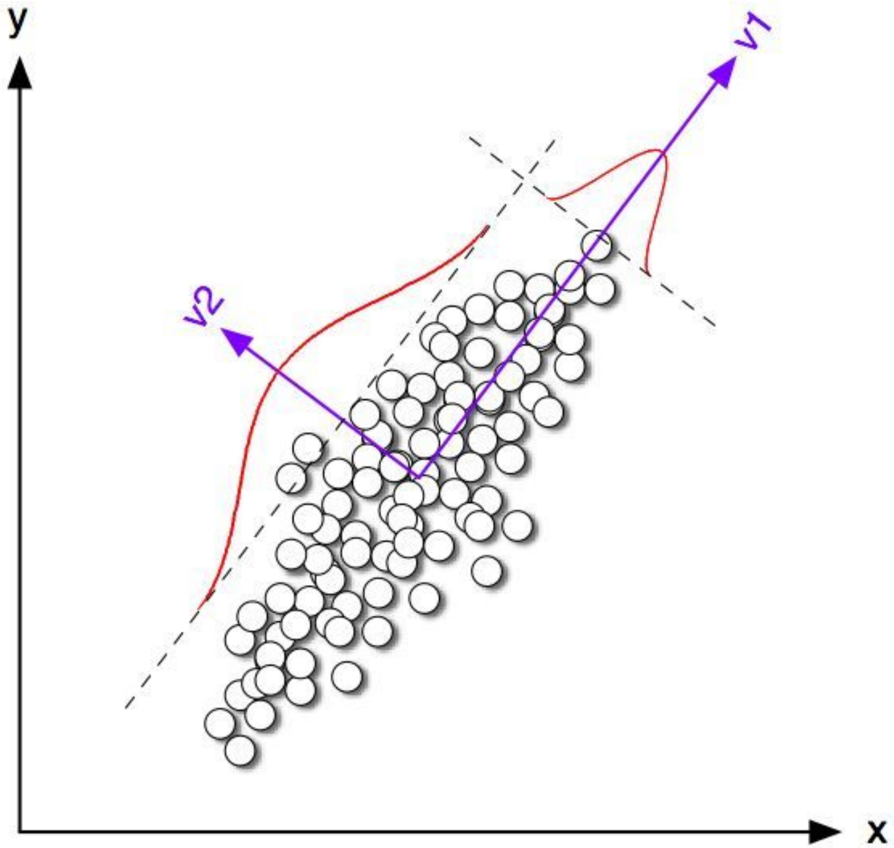
- Principal Component Analysis (PCA)
- Using PCA for computer vision: Eigenfaces
- Linear Discriminant Analysis (LDA)
- Visual bag of words (BoW)
- Spatial pyramids

Today's agenda

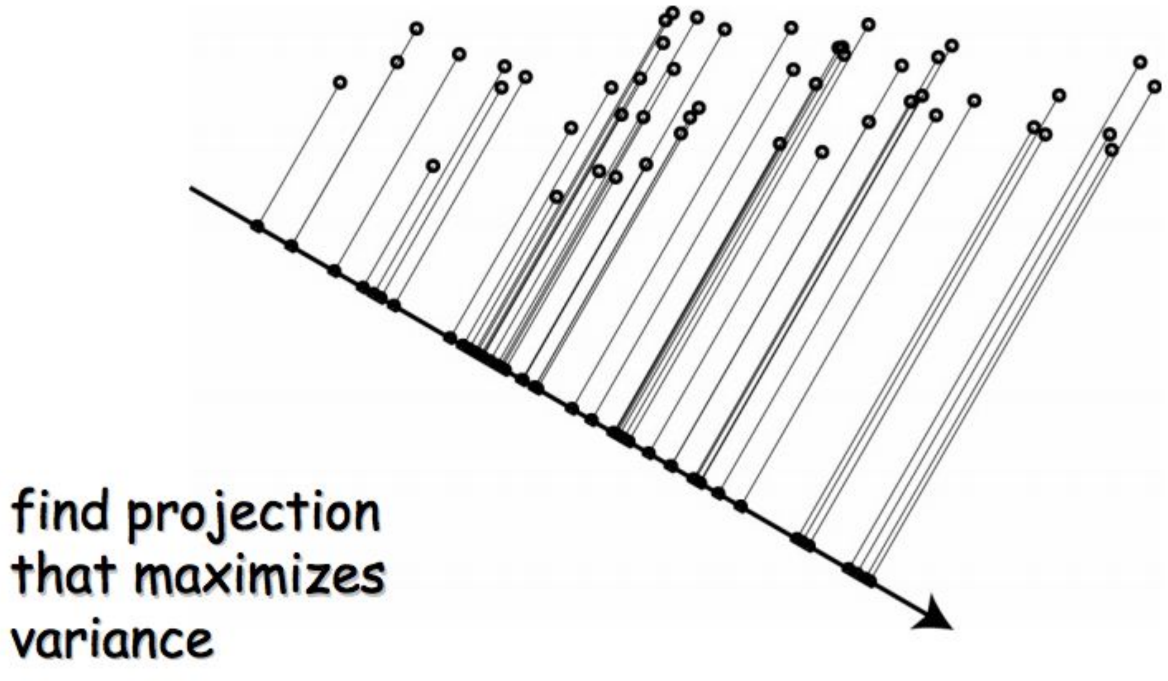
- Principal Component Analysis (PCA)
- Using PCA for computer vision: Eigenfaces
- Linear Discriminant Analysis (LDA)
- Visual bag of words (BoW)
- Spatial pyramids

Intuition behind PCA: high dimensional data usually lives in some lower dimensional space

Covariance between the two dimensions of features is high. Can we reduce the number of dimensions to just 1?

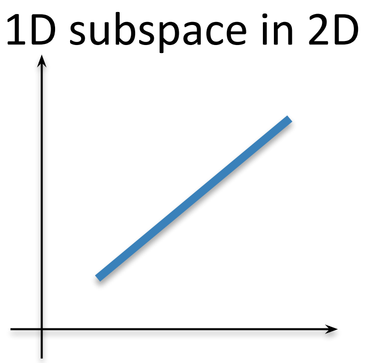


Geometric interpretation of PCA



Geometric interpretation of PCA

- Let's say we have a set of 2D data points x. But we see that all the points lie on a line in 2D.
- So, 2 dimensions are redundant to express the data. We can express all the points with just one dimension.



PCA: Principal Component Analysis

- Given a dataset of images, can we compressed them like we can compress a single image?
  - Yes, the trick is to look into the correlation between the dimensions of the image
  - The tool for doing this is called PCA

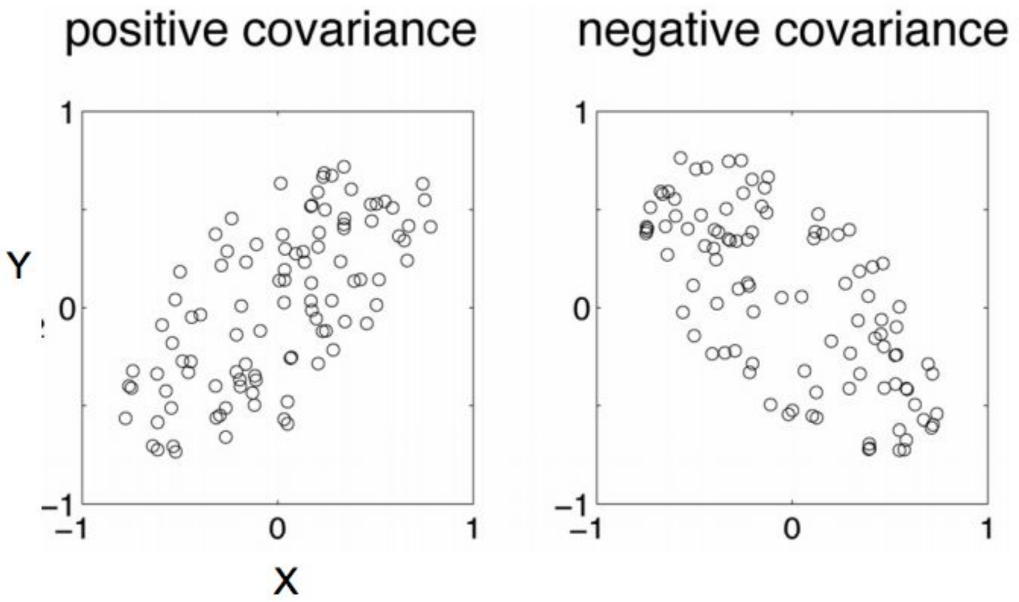
PCA can be used to compress image RGB pixel values or also be used to compress their features!

Toy example to explain covariance

- What is covariance between dimensions?
- Let’s say we have a dataset of students
  - each student is represented with 3 dimensions
  - **x**: number of hours studied for a subject
  - **y**: marks obtained in that subject
  - **z**: number of lectures attended
- covariance value between **x and y is say: 104.53**
  - what does this value mean?

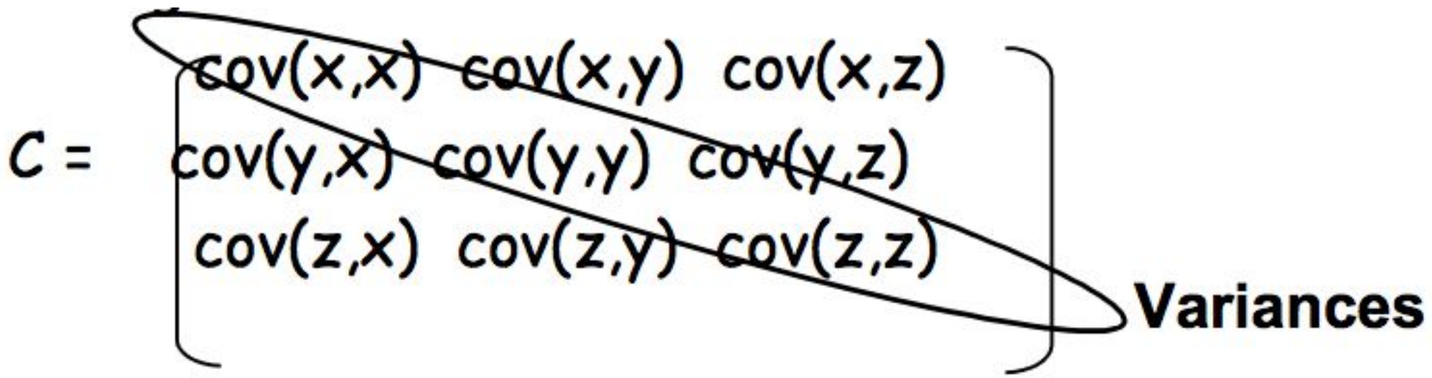
Covariance interpretation

- ○ **x**: number of hours studied for a subject
- ○ **y**: marks obtained in that subject
- covariance value between **x and y is say: 104.53**
  - ○ what does this value mean?



Visualizing this covariance matrix

- We can represent these covariance correlation numbers in a matrix
- e.g. for 3 dimensions:



- Diagonal is the **variances** of x, y and z
- **cov(x,y) = cov(y,x)** hence **C is symmetrical** about the diagonal
- N-dimensional data will result in NxN covariance matrix

Covariance interpretation

- Exact value is not as important as it's sign.
- A **positive value** of covariance indicates both dimensions increase or decrease together e.g. as the number of hours studied increases, the marks in that subject increase.
- A **negative value** indicates while one increases the other decreases, or vice-versa e.g. active social life at PSU vs performance in CS dept.
- If **covariance is zero**: the two dimensions are independent of each other e.g. heights of students vs the marks obtained in a subject

PCA by SVD

- To relate this to PCA, we consider the image (or feature) matrix

$$X = \begin{bmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{bmatrix}$$

- The sample mean of this dataset (or in plain english, the average image) is:

$$\mu = \frac{1}{n}\sum_i x_i = \frac{1}{n}\begin{bmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{bmatrix}\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \frac{1}{n}X1$$

PCA by SVD

- Center the data by subtracting the mean to each column of X
- The centered dataset matrix is

$$X_c = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix} - \begin{bmatrix} | & & | \\ \mu & \cdots & \mu \\ | & & | \end{bmatrix}$$

PCA by SVD

- The sample covariance matrix is

$$C = \frac{1}{n}\sum_i (x_i - \mu)(x_i - \mu)^T = \frac{1}{n}\sum_i x_i^c (x_i^c)^T$$

  where x_i^c is the i^{th} column of X_c

- This can be written as

$$C = \frac{1}{n}\begin{bmatrix} | & & | \\ x_1^c & \cdots & x_n^c \\ | & & | \end{bmatrix}\begin{bmatrix} - & x_1^c & - \\ & \vdots & \\ - & x_n^c & - \end{bmatrix} = \frac{1}{n} X_c X_c^T$$

PCA by SVD

- The matrix

$$X_c^T = \begin{bmatrix} - & x_1^c & - \\ & \vdots & \\ - & x_n^c & - \end{bmatrix}$$

is real (n x d). Assuming n>d it has SVD decomposition

$$X_c^T = U\Sigma V^T \qquad U^T U = I \qquad V^T V = I$$

and

$$C = \frac{1}{n} X_c X_c^T$$

Calculating covariance matrix

$$C = \frac{1}{n} X_c X_c^T$$

$$= \frac{1}{n} U \Sigma V^T (U \Sigma V^T)^T$$

$$= \frac{1}{n} U \Sigma V^T V \Sigma U^T$$

$$= \frac{1}{n} U \Sigma^2 U^T$$

PCA by SVD

$$C = \frac{1}{n} U \Sigma^2 U^T$$

- Note that U is (d x d) and orthonormal, and Σ^2 is diagonal. **This is just the eigenvalue decomposition of C**
- This means that we can calculate the eigenvectors of C using the eigenvectors of X_c
- It follows that
  - The eigenvectors of C are the columns of U
  - The eigenvalues of C are the diagonal entries of Σ^2: λ_i^2

PCA by SVD

- In summary, computation of PCA by SVD
- Given X with one image (or feature) per column
  - Create the centered data matrix

$$X_c = \begin{bmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{bmatrix} - \begin{bmatrix} | & & | \\ \mu & \dots & \mu \\ | & & | \end{bmatrix}$$

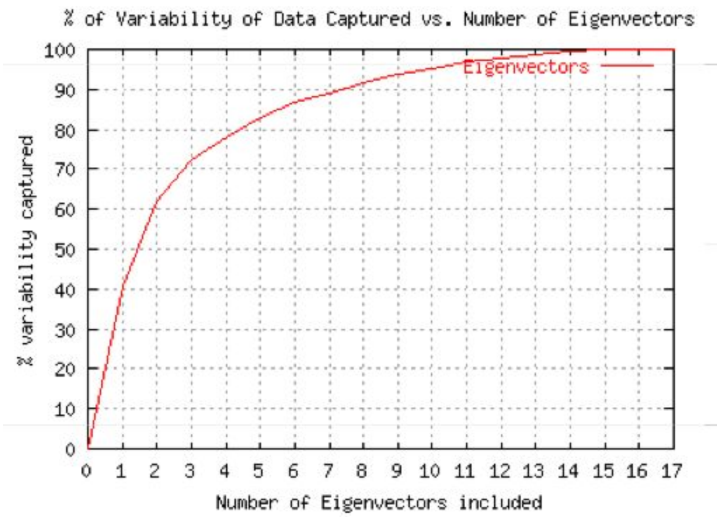
  - Compute its SVD

$$X_c^T = U\Sigma V^T$$

  - Principal components of the covariance matrix C are columns of U

To compress an image dataset, pick the largest eigenvalues and their corresponding eigenvectors

- Pick the eigenvectors that explain p% of the image data variability
  - Can be done by plotting the ratio r_k as a function of k



$$r_k = \frac{\sum_{i=1}^{k} \lambda_i^2}{\sum_{i=1}^{n} \lambda_i^2}$$

  - E.g. we need k=3 eigenvectors to cover 70% of the variability of this dataset

What exactly is the covariance

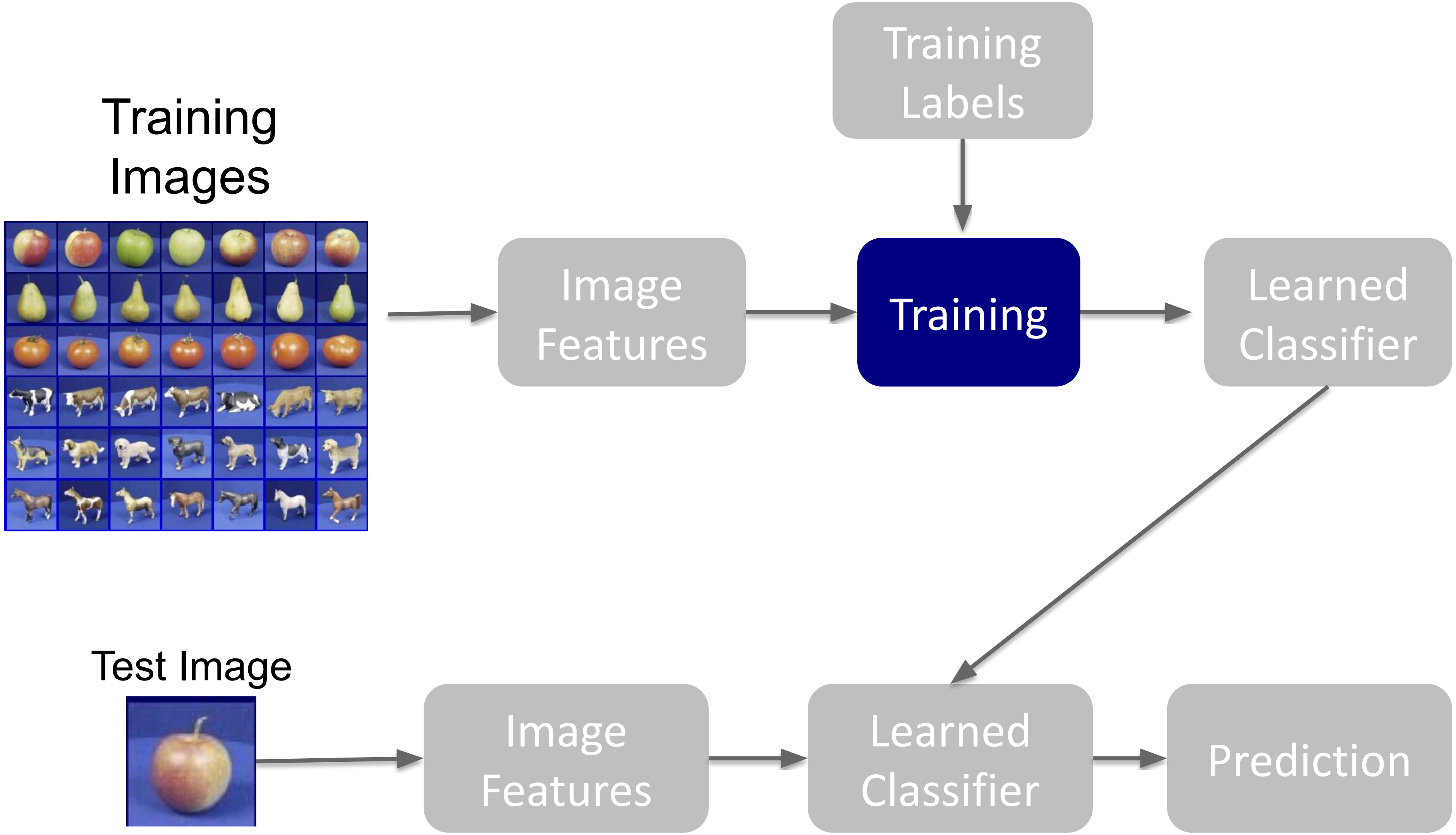
- Variance and Covariance are a measure of the **"spread"** of a set of points around their center of mass (mean)
- **Variance** – measure of the deviation from the mean for points in one dimension e.g. heights
- **Covariance** as a measure of how much each of the dimensions vary from the mean with respect to each other.
- Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions e.g. number of hours studied & marks obtained.
- The covariance between one dimension and itself is the variance

Covariance

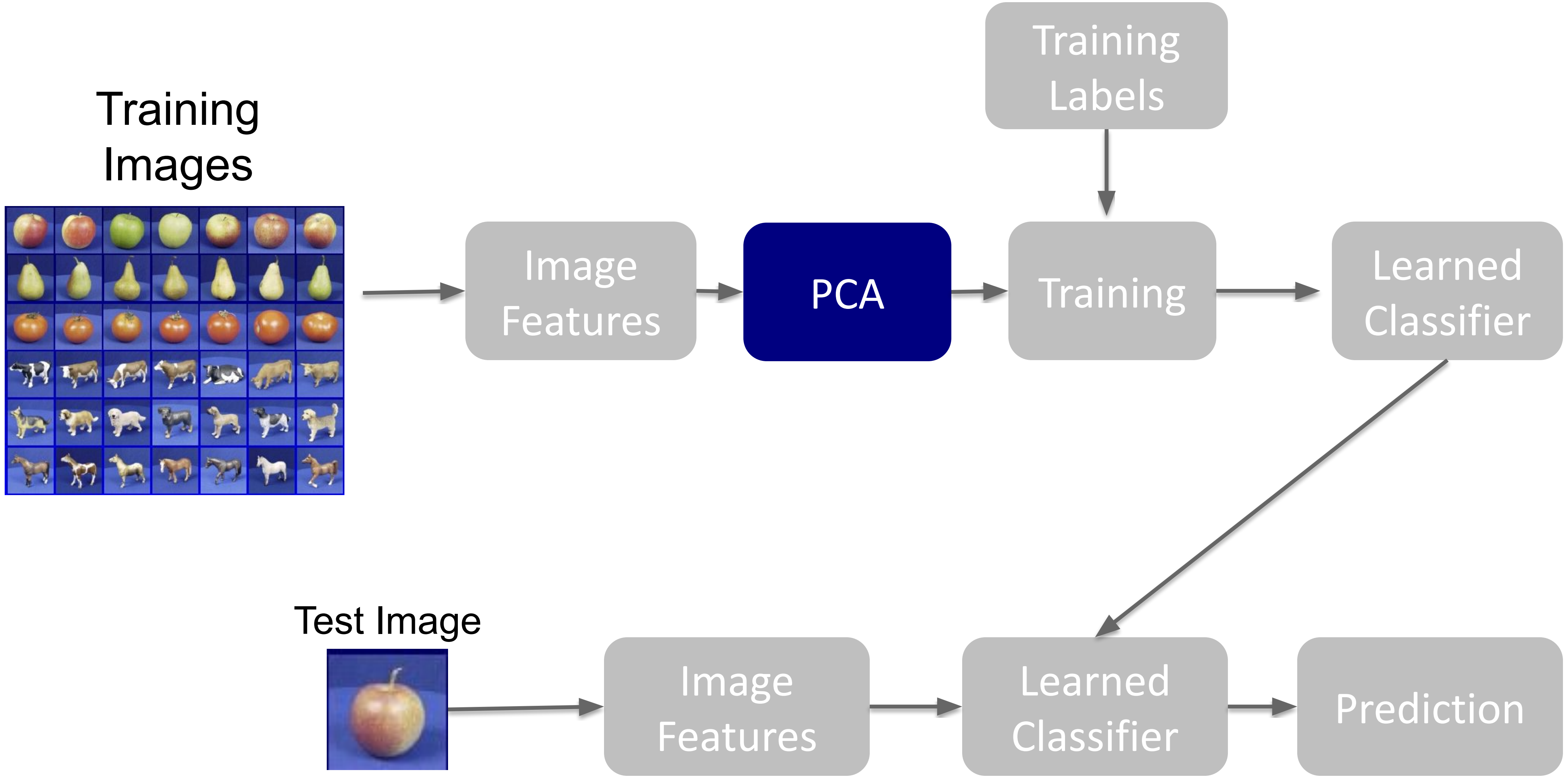
$$\text{covariance}(X,Y) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

- So, if you had a 3-dimensional data set (x,y,z), then you could measure the covariance between the x and y dimensions, the y and z dimensions, and the x and z dimensions. Measuring the covariance between x and x , or y and y , or z and z would give you the variance of the x , y and z dimensions respectively

What happens with PCA during training?



What happens with PCA during training?



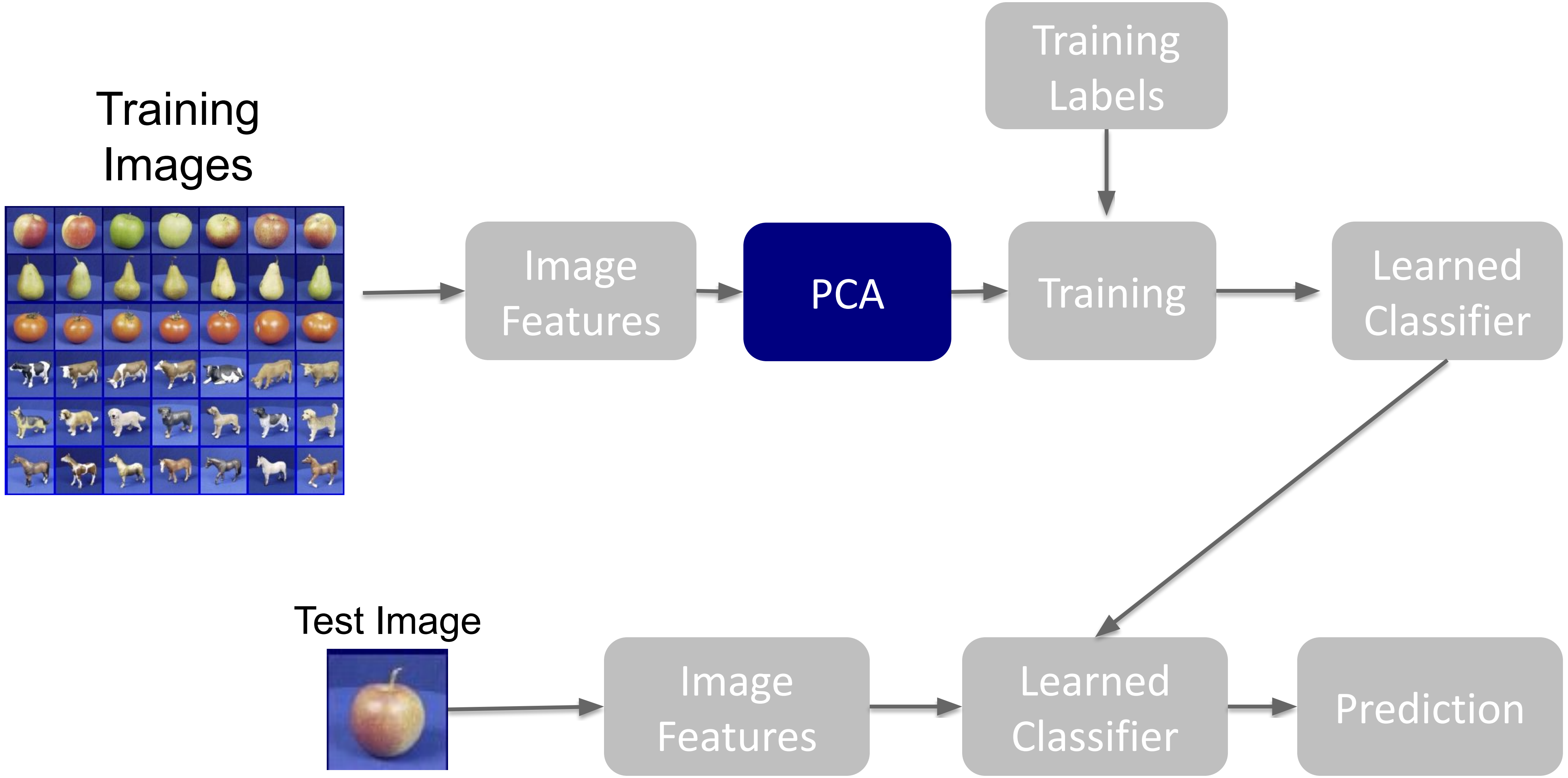
PCA during training

Let's say that we choose k top eigenvalues and their corresponding eigenvectors: $[u_1, \ldots, u_k]$

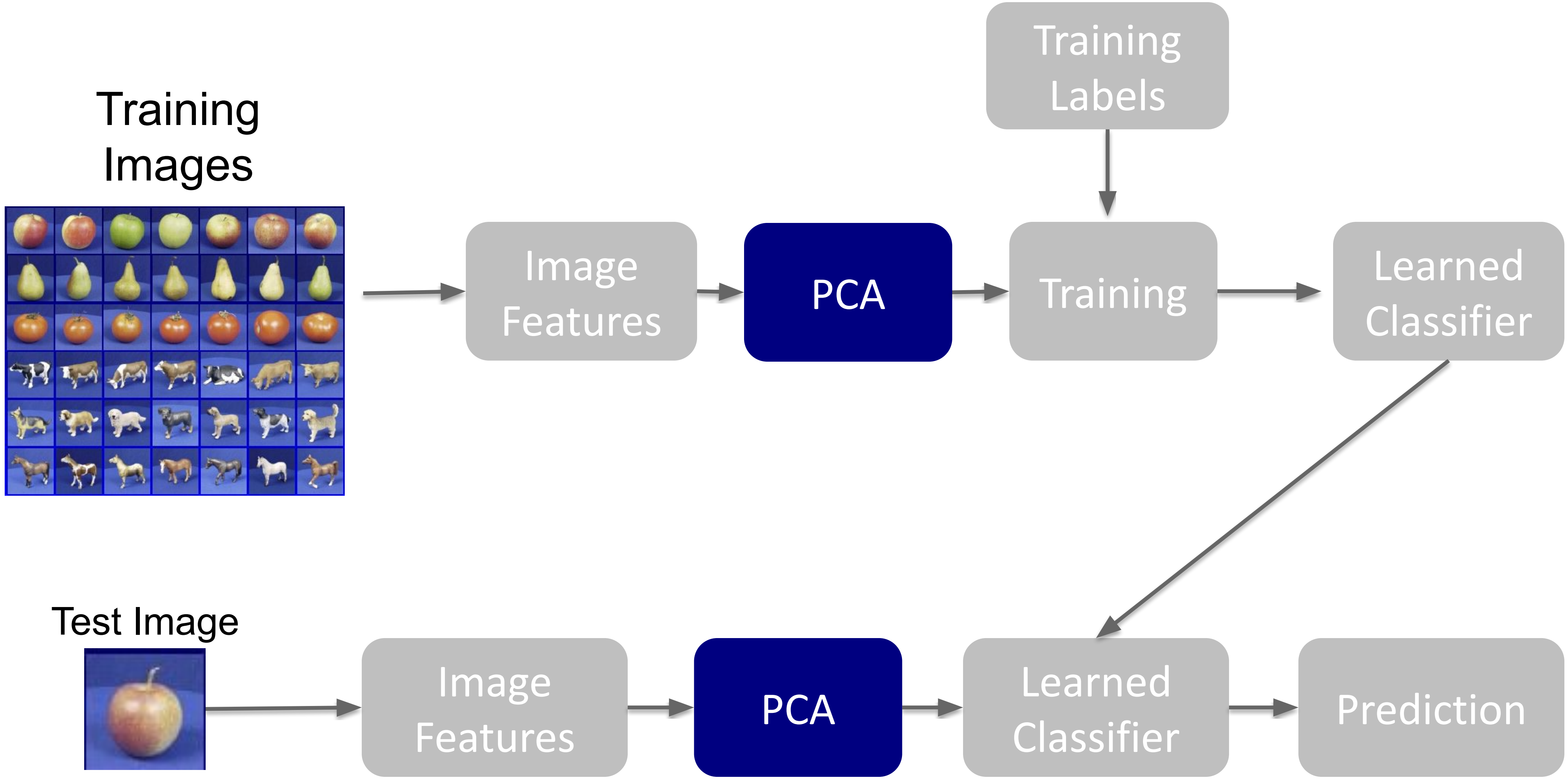
Replace all image features x with:

$$\hat{x} = \begin{bmatrix} u_1^T x \\ u_2^T x \\ \dots \\ u_k^T x \end{bmatrix}$$

What happens with PCA during testing?



What happens with PCA during testing?



Today’s agenda

- Principal Component Analysis (PCA)
- Using PCA for computer vision: Eigenfaces
- Linear Discriminant Analysis (LDA)
- Visual bag of words (BoW)
- Spatial pyramids

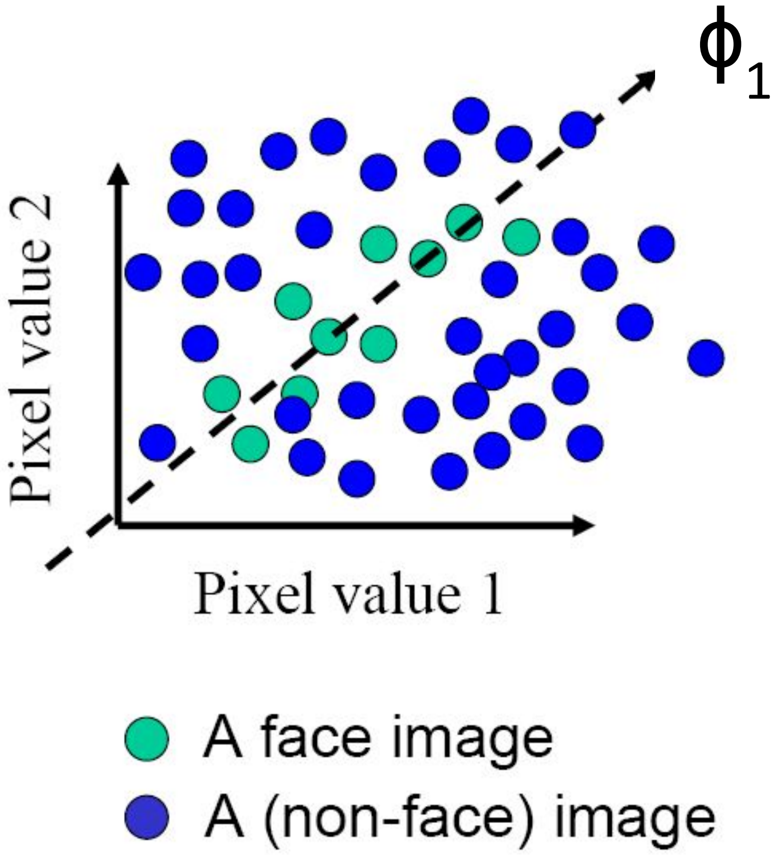
Turk and Pentland, Eigenfaces for Recognition, *Journal of Cognitive Neuroscience* **3** (1): 71–86.

How PCA was originally used in vision: To identify celebrities using their faces

- An image is a point in a high dimensional space
  - In grayscale, an N x M image is a point in R^{NM}
  - E.g. 100x100 images lives in a 10,000-dimensional space

100x100 images can contain many things other than faces!

The Space of Faces



- However, relatively few high dimensional vectors correspond to valid face images
- We want to effectively model the subspace of face images

This is where PCA comes in

Slide credit: Chuck Dyer, Steve Seitz, Nishino

Eigenfaces: an algorithm using PCA to reduce the space of faces

- Assume that most face images lie on a low-dimensional subspace determined by the first k (k<<d) eigenvectors of a dataset of faces
- To demonstrate the effectiveness of PCA for images, they called each eigenvector of a dataset "eigenfaces"
- Represent all face images in the dataset as linear combinations of eigenfaces

M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

Training images: $\mathbf{x}_1, \ldots, \mathbf{x}_N$

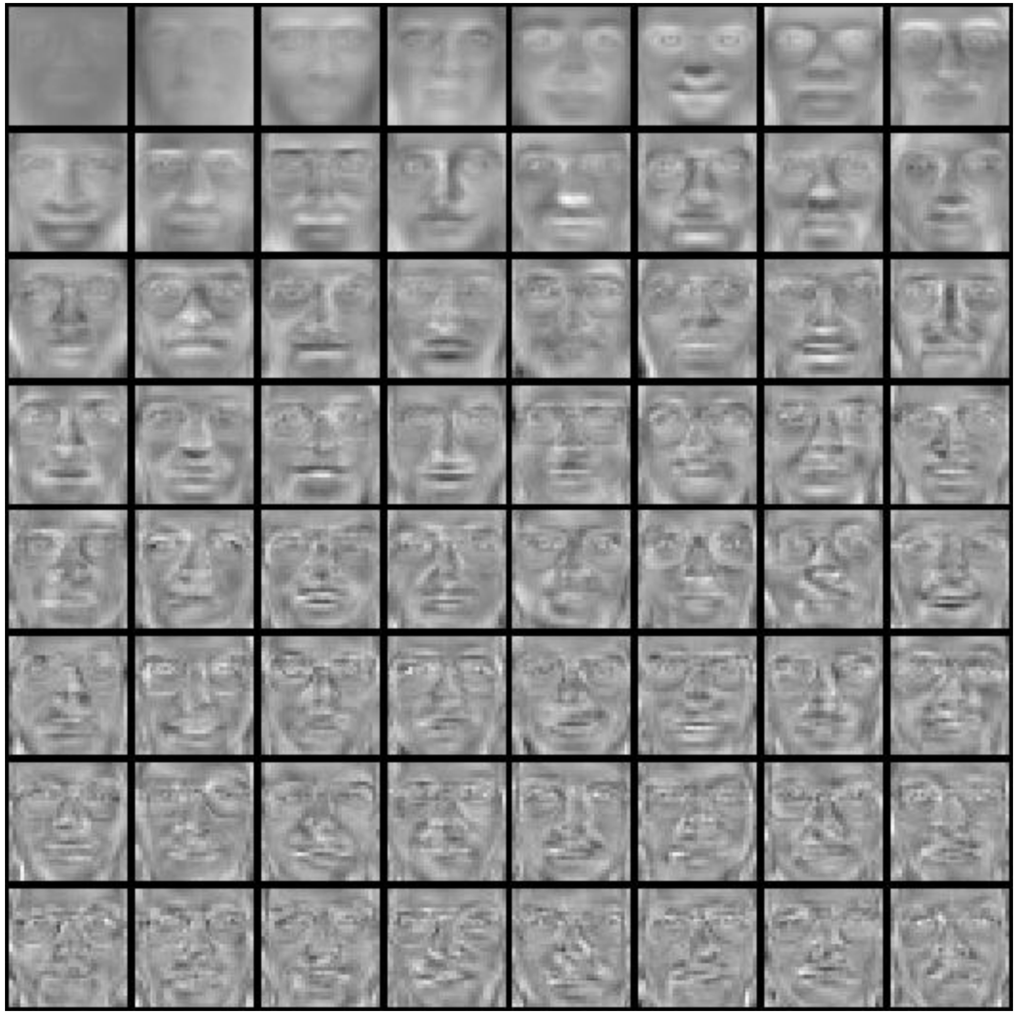
Each 100x100 image is going to be represented as a 10,000-dimensional vector

$$X = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix}$$

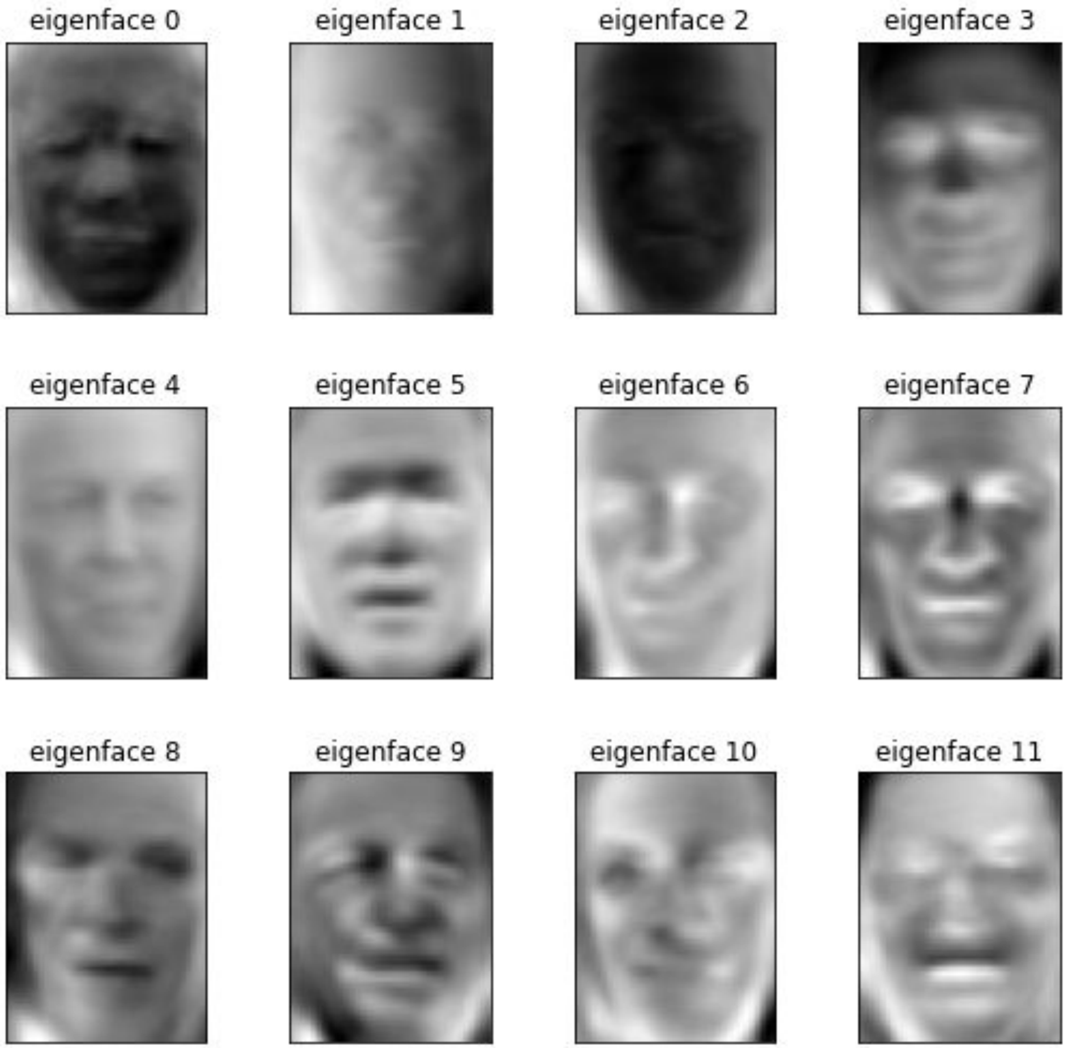
Top eigenvectors: $U_1,\ldots,U_k$

Mean: μ

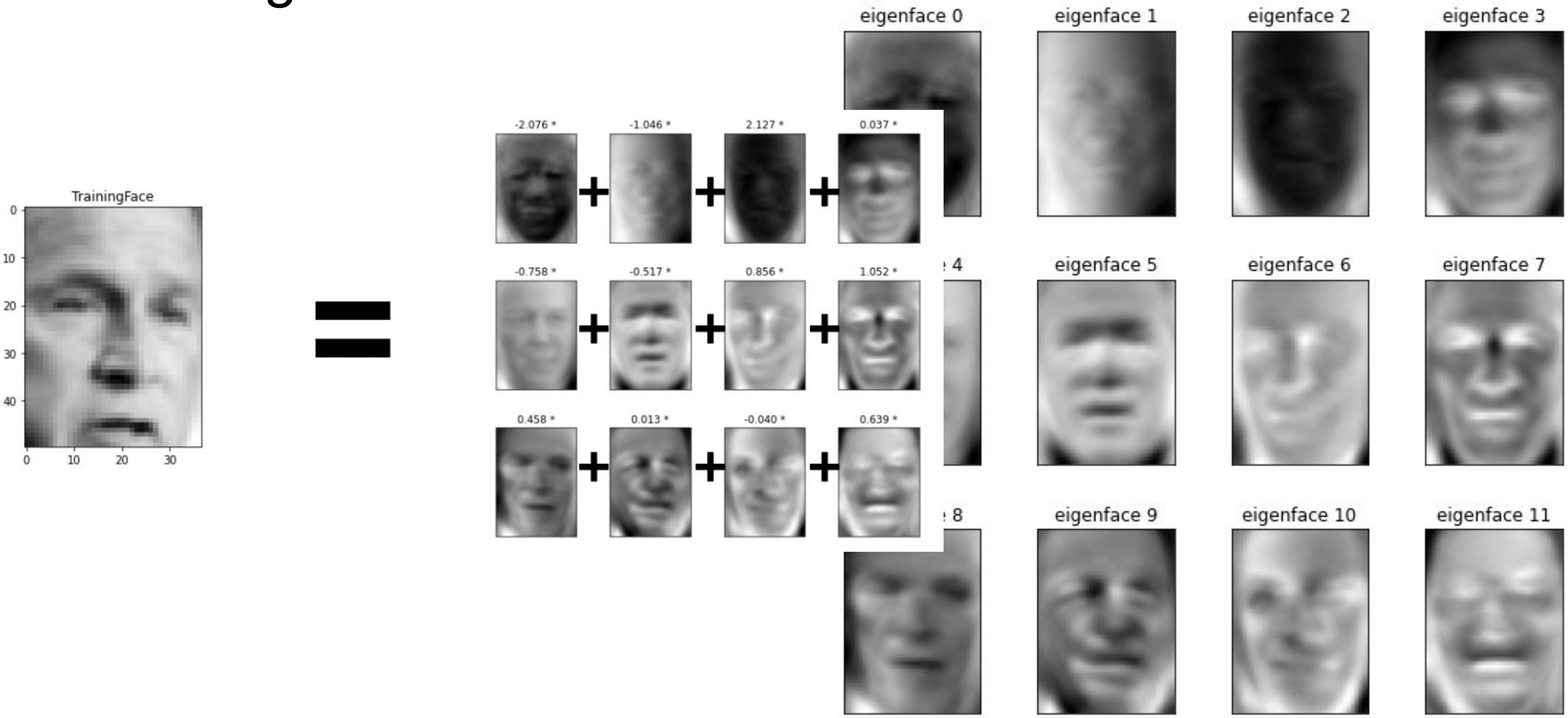
$$\mu = \frac{1}{n}\sum_i x_i$$



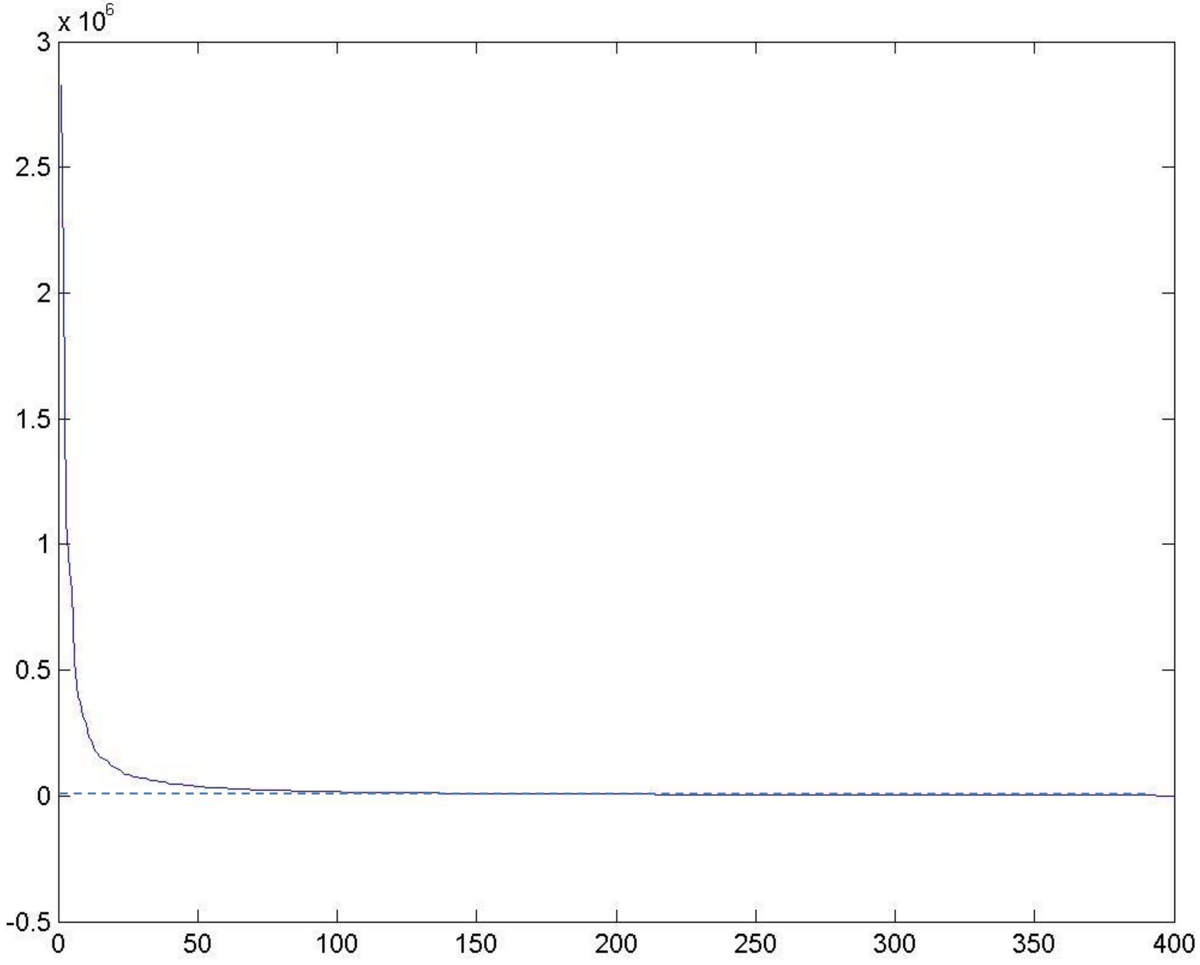
Calculate its SVD and visualize its top eigenvectors



Every image can be reconstructed as a linear combination of these eigenvectors



Error rate when reconstructing a face decreases as you use more eigenvectors



Reconstruction and Errors

- Fewer eigenfaces result in more information loss, and hence less discrimination between faces.

Using PCA for classifying faces

- Training
  1. Place all training images $x_1, x_2, \ldots, x_N$ into a matrix
  2. Compute average face
  3. Compute the difference image (the centered data matrix)

$$X_c = \begin{bmatrix} | & & | \\ x_1 & \ldots & x_n \\ | & & | \end{bmatrix} - \begin{bmatrix} | & & | \\ \mu & \ldots & \mu \\ | & & | \end{bmatrix}$$

  4. Use SVD to find the eigenvectors of the covariance matrix

$$X_c^T = U \Sigma V^T$$

  5. Keep the top-K eigenvalues and their eigenvectors
  6. Compute each training image x_i 's new projected features:

$$\hat{x} = \begin{bmatrix} u_1^T x \\ u_2^T x \\ \ldots \\ u_k^T x \end{bmatrix}$$

Using PCA for classifying faces

- Testing
  1. Given a test image x_{test}
  2. Project x into this new space into eigenface space:

$$\hat{x}_{test} = \begin{bmatrix} u_1^T x_{test} \\ u_2^T x_{test} \\ \dots \\ u_k^T x_{test} \end{bmatrix}$$

  3. Run your classifier on this new space.
     - For example, use k-NN using distance measures (Euclidean) in this new space

Shortcomings

- Requires carefully curated training data:
  - All faces centered in frame
  - All faces have to be the same size
  - Some sensitivity to angle (ideally all faces are facing front)
- Alternative:
  - “Learn” one set of PCA vectors for each angle
  - Use the one with lowest error
- Method is completely knowledge free
  - (sometimes this is good!)
  - Doesn’t know that faces 2D projections of 3D heads
  - But it also makes no effort to preserve what makes a “face” a “face”

Summary for Eigenface

Pros

- Non-iterative, globally optimal solution

Cons:

- PCA projection is **optimal for reconstruction** from a low dimensional basis, but **may NOT be optimal for recognition**
- Is there a better dimensionality reduction?

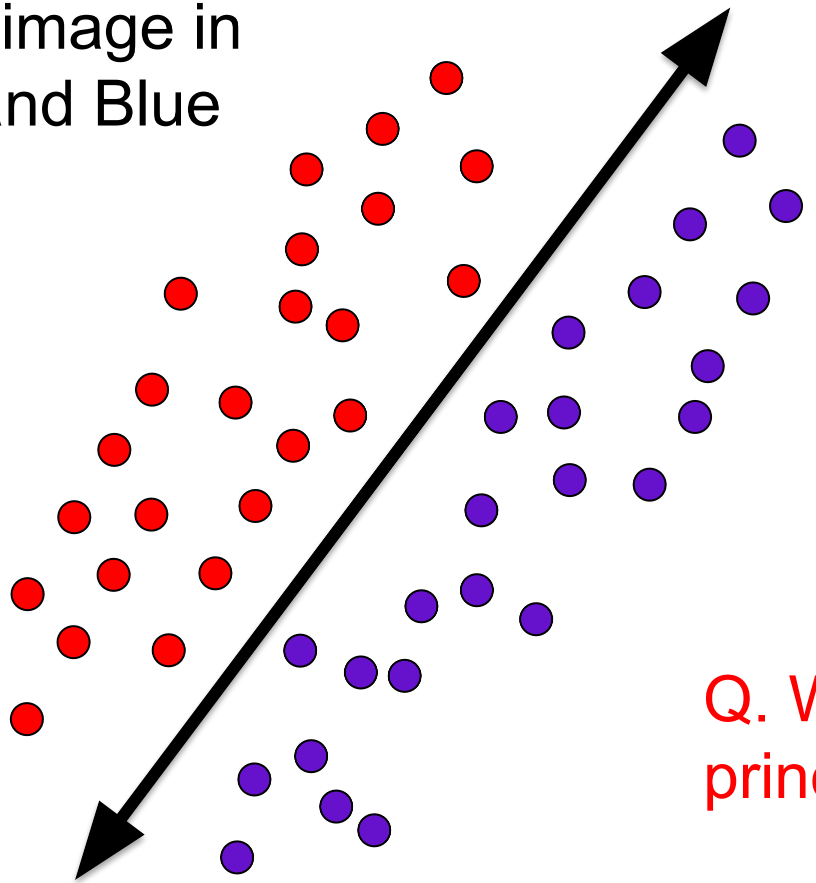
Today’s agenda

- Principal Component Analysis (PCA)
- Using PCA for computer vision: Eigenfaces
- **Linear Discriminant Analysis (LDA)**
- Visual bag of words (BoW)
- Spatial pyramids

P. Belhumeur, J. Hespanha, and D. Kriegman. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection". *IEEE Transactions on pattern analysis and machine intelligence* **19** (7): 711. 1997.

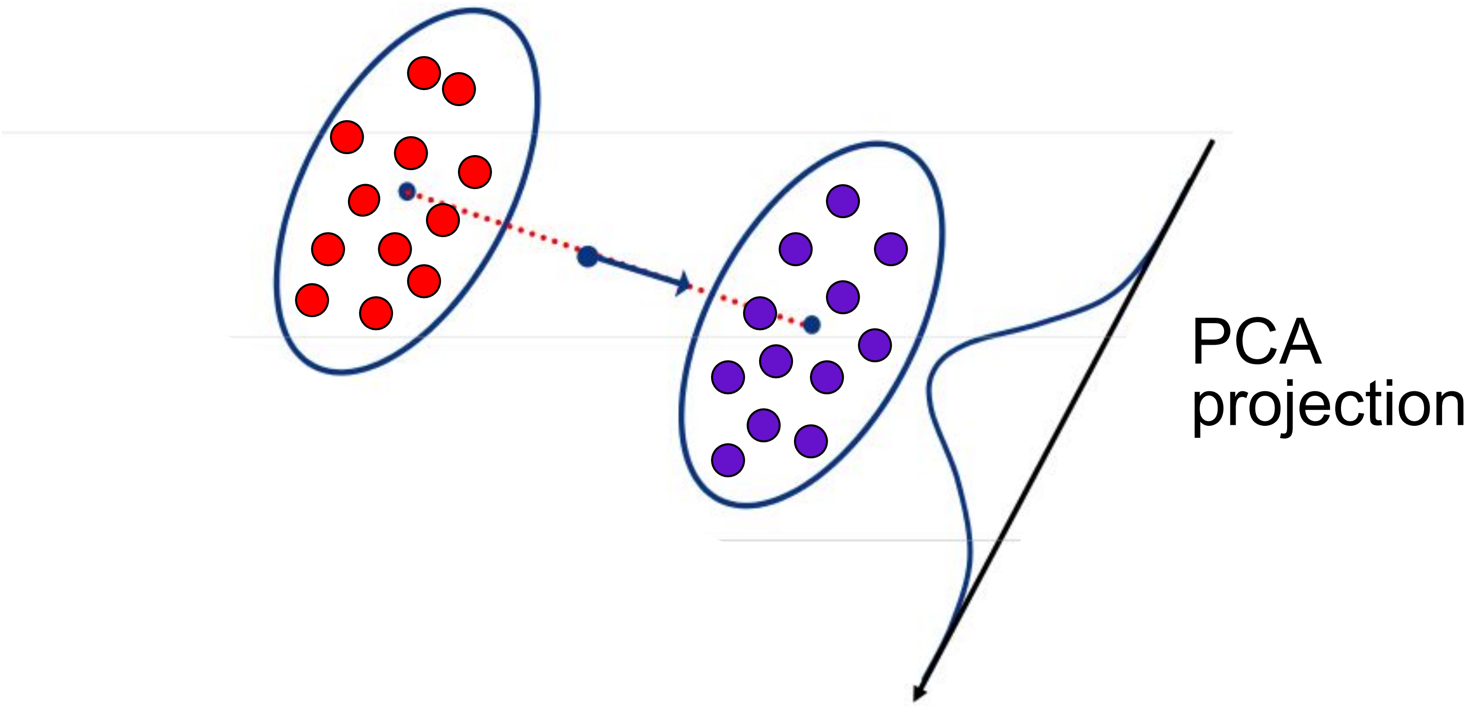
Let's say that we this hypothetical 2-dimensional feature space.

Here I am showing each image in this feature space. Red and Blue are the two classes.

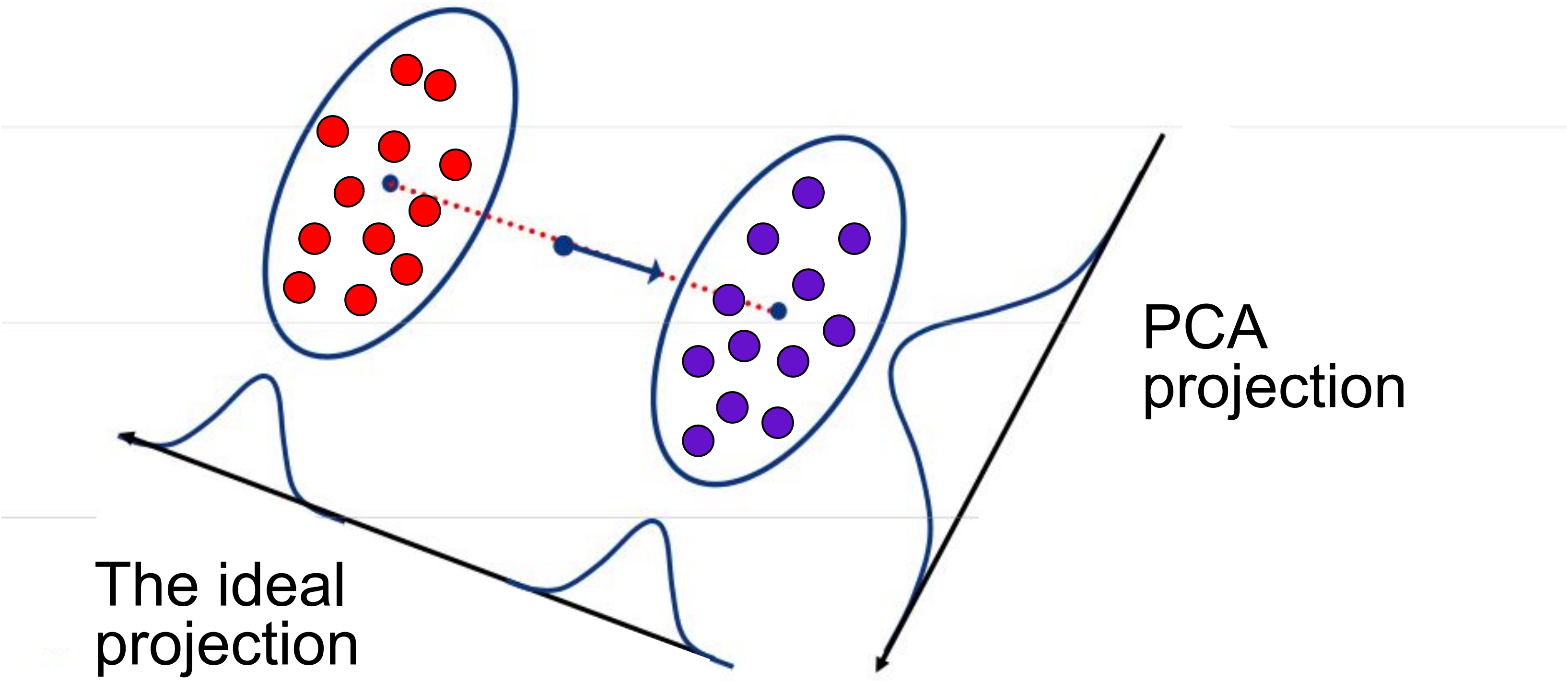


Q. Which direction will is the first principle component?

PCA can project the data such that it will become harder to separate the two classes

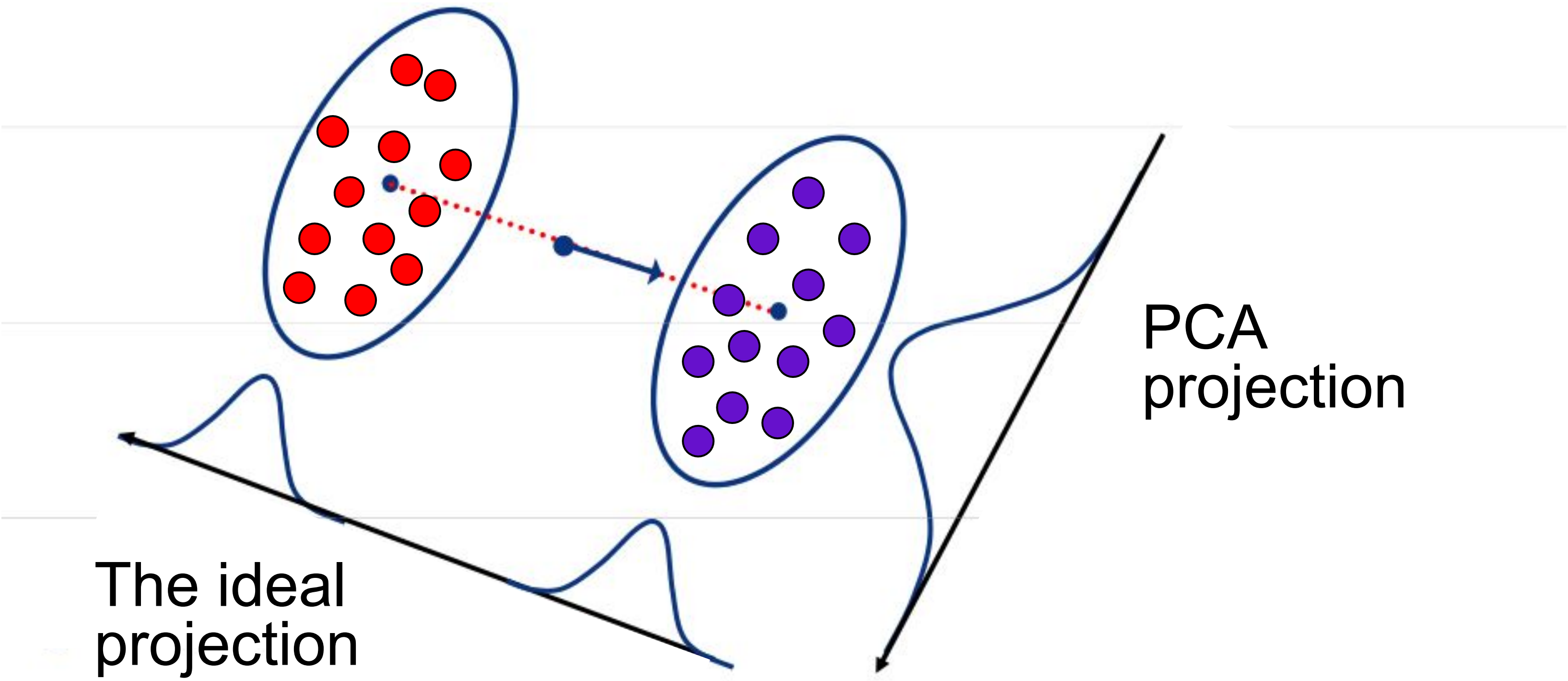


The ideal projection should make it easy to differentiate between images from two classes



Fischer's Linear Discriminant Analysis (LDA)

- Goal: find the best separation between two classes



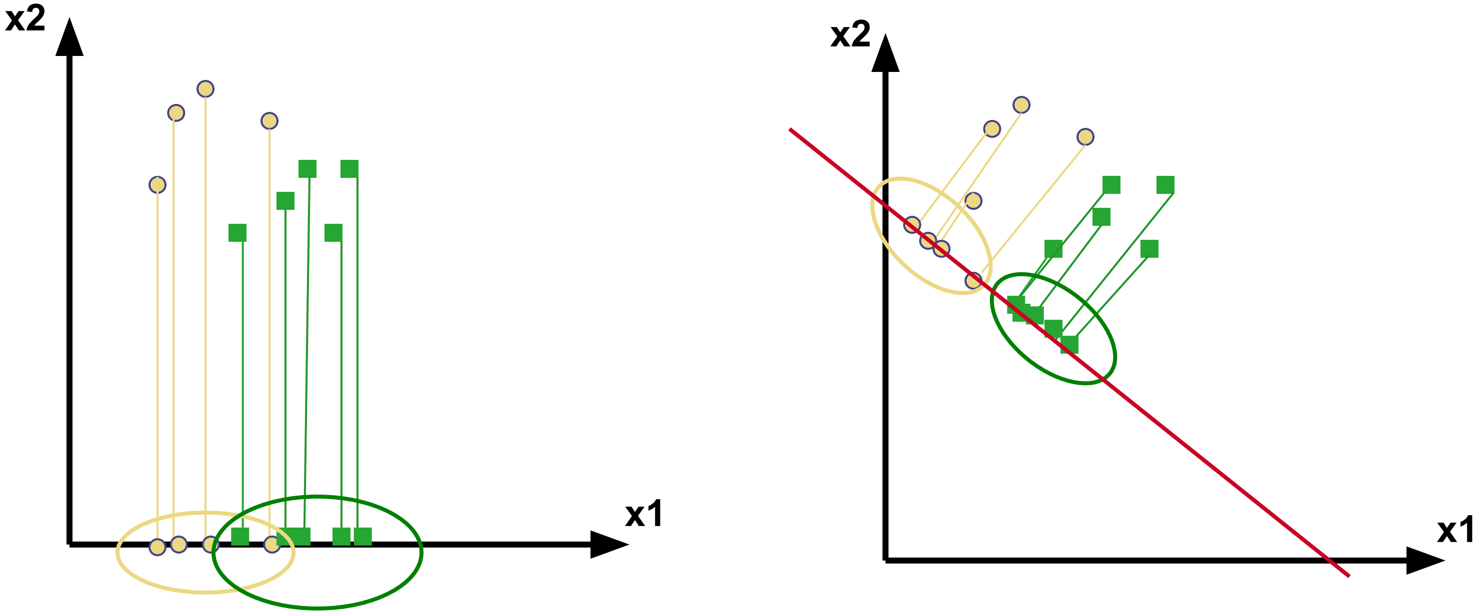
Difference between PCA and LDA

- PCA preserves maximum variance
    - PCA maximizes our ability to reconstruct each image
    - Doesn’t help us find the best projection for classification

- LDA preserves discrimination (difference between categories)
    - Find projection that **maximizes scatter** between classes and **minimizes scatter** within classes

How LDA reduces dimentionality

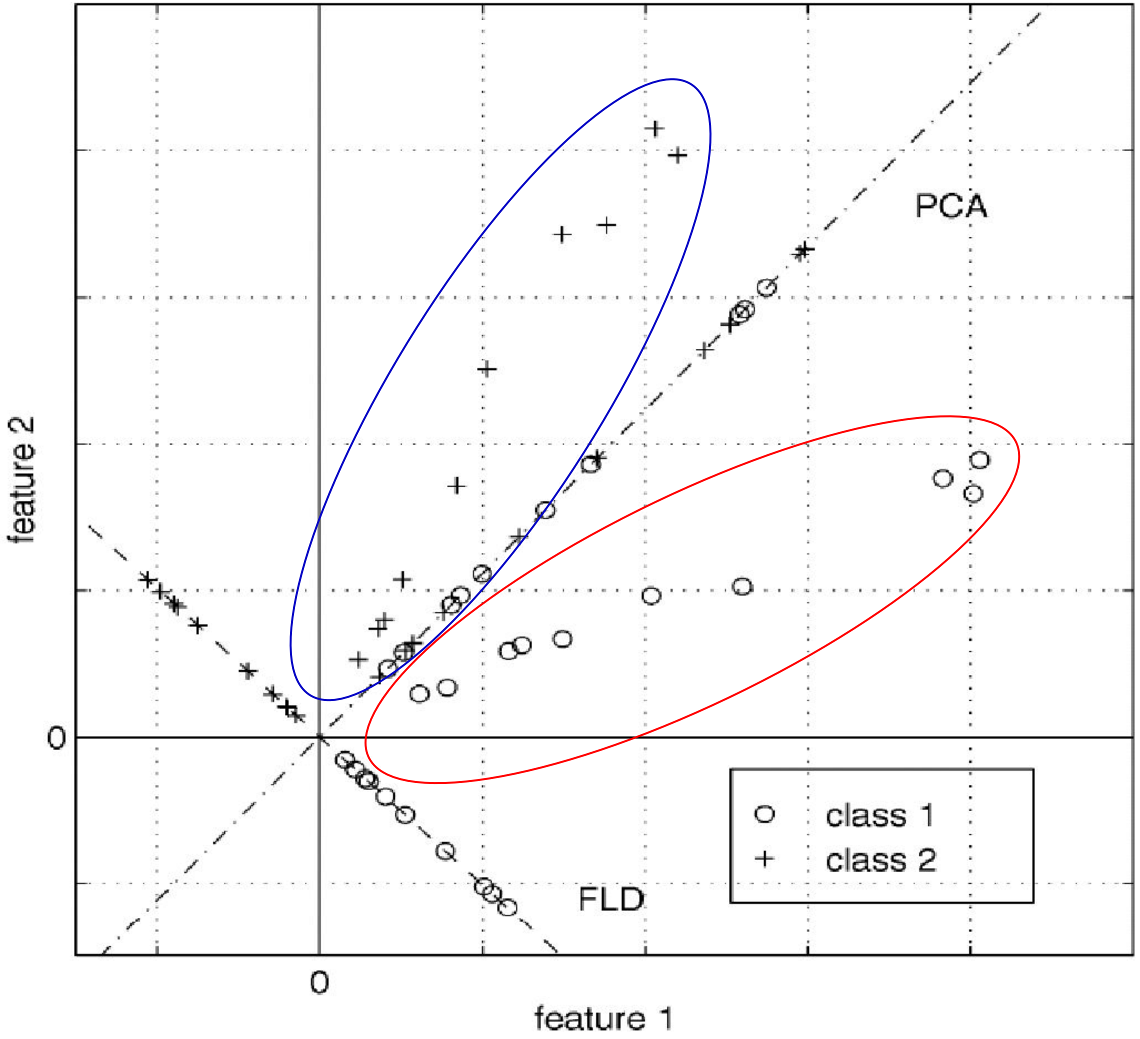
- Using two classes as example:



Poor Projection

Good

Basic intuition: PCA vs. LDA



First, let's calculate the per category statistics

- We want to learn a dimension reduction **projection W** such that the projection converts all image features **x** to a lower dimensional space:

$$z = w^T x \qquad z \in \mathbf{R}^m \quad x \in \mathbf{R}^n$$

- First, let's calculate the **per class** means be:

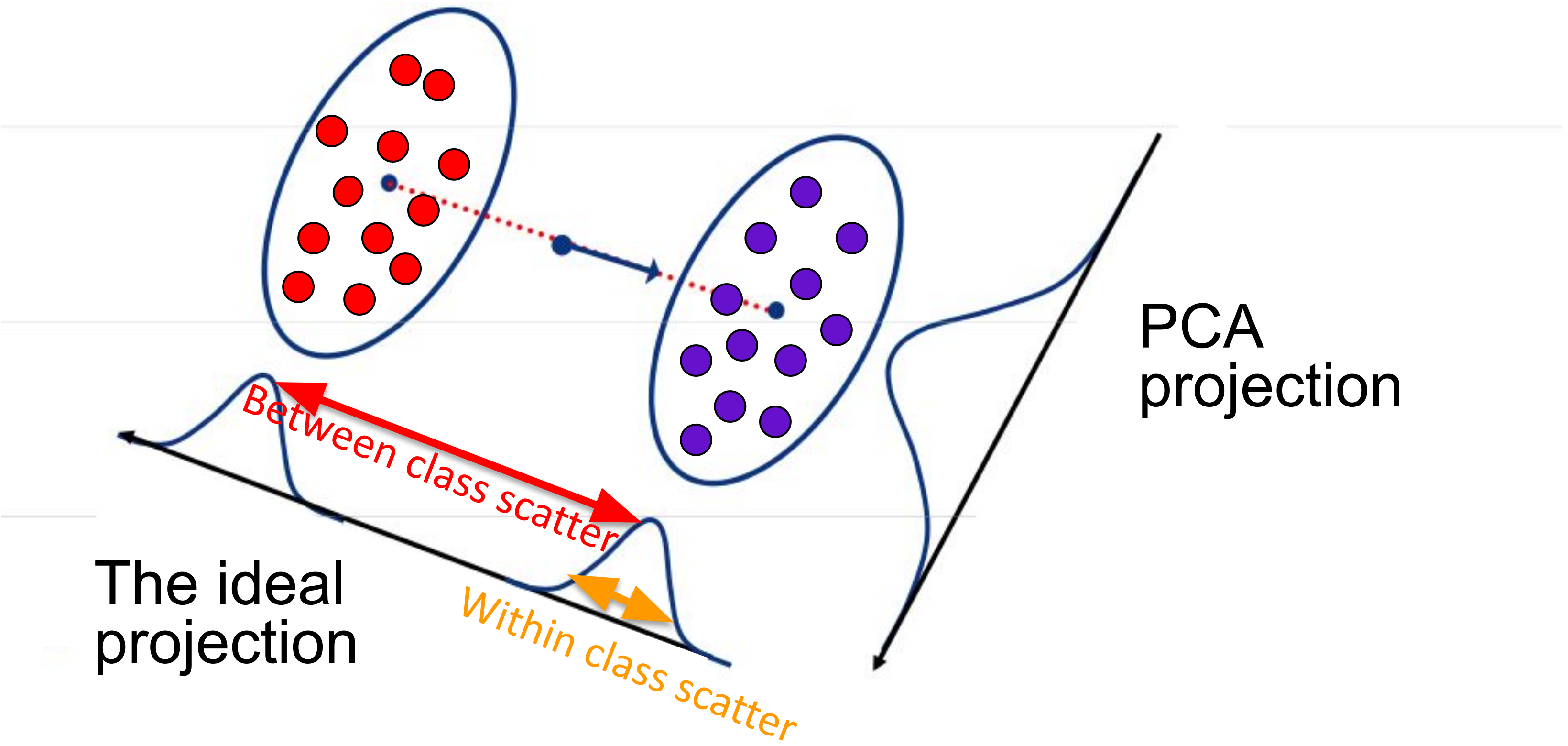
$$\mu_i = E_{X|Y}[X|Y = i]$$

- And the **per class** covariance matrices are:

$$C_i = [(X_i - \mu_i)(X_i - \mu_i)^T | Y = i]$$

Using the per class means and covariance, we want to minimize the following objective:

We want a projection that maximizes: $J(w) = \max \frac{between\ \ class\ \ scatter}{within\ \ class\ \ scatter}$



What does J(w) look like when we only have 2 classes

The following objective function:

$$J(w) = \frac{\textit{between class scatter}}{\textit{within class scatter}}$$

Can be written as

$$J(w) = \frac{|E_{Z|Y}[Z|Y=1] - E_{Z|Y}[Z|Y=0]|^2}{\text{var}[Z|Y=1] + \text{var}[Z|Y=0]}$$

LDA with 2 variables

- **Numerator:** We can write the **between** class scatter as:

$$\begin{aligned} |E_{Z|Y}[Z|Y=1] - E_{Z|Y}[Z|Y=0]|^2 &= |w^T(\mu_1 - \mu_0)|^2 \\ &= w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w \end{aligned}$$

- **Each part of Denominator:** Also, the **within** class scatter becomes:

$$\begin{aligned} var[Z|Y=i] &= E_{Z|Y}[w^T(x-\mu_i)^2|Y=i] \\ &= E_{Z|Y}[w^T(x-\mu_i)(x-\mu_i)^T w|Y=i] \\ &= w^T C_i w \end{aligned}$$

LDA with 2 variables

- We can plug in these scatter values to our objective function:

$$J(w) = \frac{w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T C_1 w + w^T C_0 w}$$

LDA with 2 variables

- We can plug in these scatter values to our objective function:

$$J(w) = \frac{w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T C_1 w + w^T C_0 w}$$

$$= \frac{w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T (C_1 + C_0) w}$$

LDA with 2 variables

- We can plug in these scatter values to our objective function:

$$J(w) = \frac{w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T C_1 w + w^T C_0 w}$$

$$= \frac{w^T (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T (C_1 + C_0) w}$$

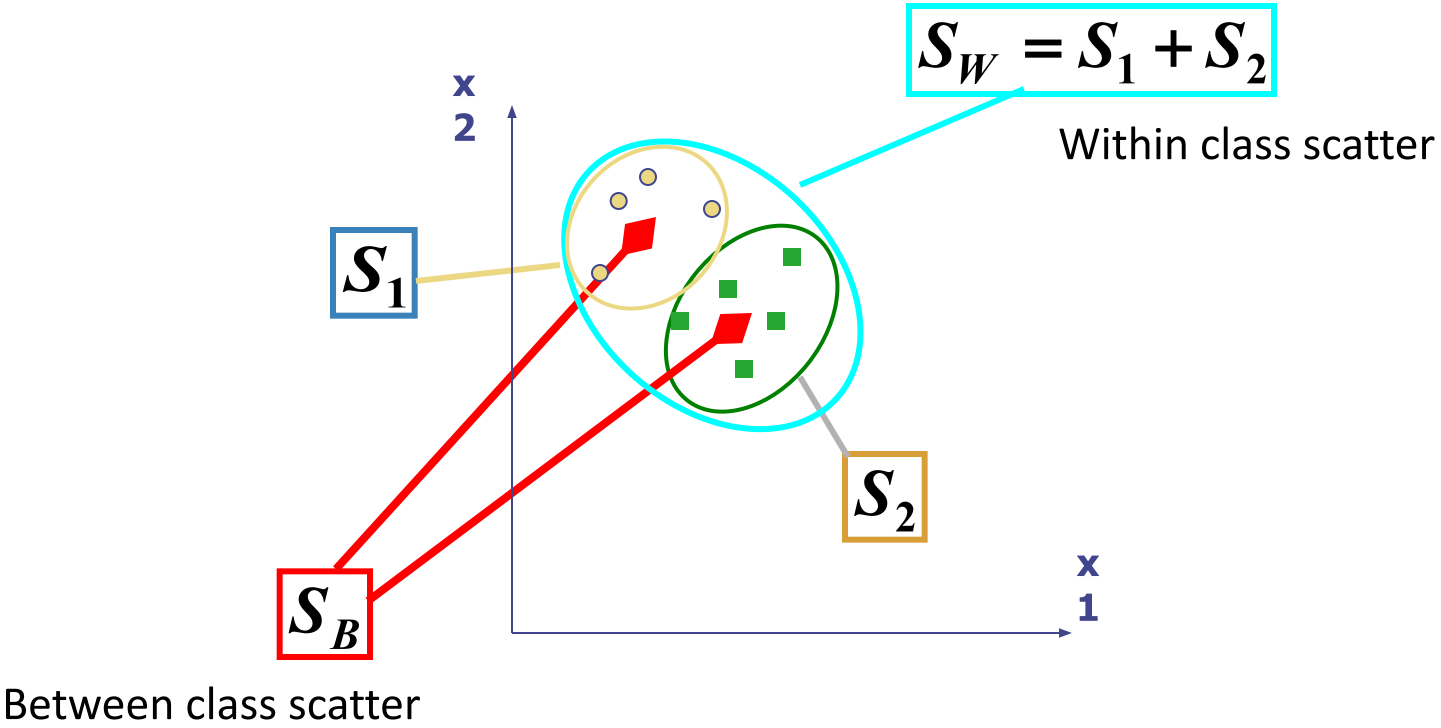
$$S_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T$$

Between class scatter

$$S_W = (C_1 + C_0)$$

Inwithin class scatter

Visualizing S_w and S_B



Linear Discriminant Analysis (LDA)

- Maximizing the ratio

$$J(w) = \frac{w^T S_B w}{x^T S_W w}$$

Linear Discriminant Analysis (LDA)

- Maximizing the ratio

$$J(w) = \frac{w^T S_B w}{x^T S_W w}$$

- Is equivalent to maximizing the numerator while keeping the denominator constant, i.e.

$$\max_w w^T S_B w \quad \text{subject to} \quad w^T S_W w = K$$

Linear Discriminant Analysis (LDA)

- Maximizing the ratio

$$J(w) = \frac{w^T S_B w}{x^T S_W w}$$

- Is equivalent to maximizing the numerator while keeping the denominator constant, i.e.

$$\max_w w^T S_B w \qquad \text{subject to} \qquad w^T S_W w = K$$

- And can be accomplished using Lagrange multipliers, where we define the Lagrangian as

$$L = w^T S_B w - \lambda\left(w^T S_W w - K\right)$$

- And maximize with respect to both w and λ

Linear Discriminant Analysis (LDA)

- Setting the gradient of $L = w^T\left(S_B - \lambda S_W\right)w + \lambda K$

- Taking the derivative respect to **w** to find the maximum:

$$\nabla_w L = 2\left(S_B - \lambda S_W\right)w = 0$$

Linear Discriminant Analysis (LDA)

- Setting the gradient of $L = w^T(S_B - \lambda S_W)w + \lambda K$

- Taking the derivative respect to **w** to find the maximum:

$$\nabla_w L = 2(S_B - \lambda S_W)w = 0$$

- This is maximized when $S_B w = \lambda S_W w$

Linear Discriminant Analysis (LDA)

- Setting the gradient of $L = w^T(S_B - \lambda S_W)w + \lambda K$

- Taking the derivative respect to **w** to find the maximum:

$$\nabla_w L = 2(S_B - \lambda S_W)w = 0$$

- This is maximized when $S_B w = \lambda S_W w$

- The solution is easy when S_w has an inverse: $S_W^{-1} = (C_1 + C_0)^{-1}$

Linear Discriminant Analysis (LDA)

$$S_B w = \lambda S_W w$$

If an inverse for S_W exists:

$$S_W^{-1} S_B w = \lambda w$$

We want to find the optimal w.

Q. What does this look like?

Linear Discriminant Analysis (LDA)

$$S_B w = \lambda S_W w$$

If an inverse for S_W exists:

$$S_W^{-1} S_B w = \lambda w$$

The solution is the eigenvector of $S_W^{-1} S_B$ **corresponding to the largest eigenvalue**

LDA with C classes

Same as when C=2. Except S_W and S_B now include all classes.

$$S_W = \sum_i C_i$$

$$S_B = \sum_i \sum_{j \neq i} (\mu_i - \mu_j)^2$$

PCA vs. LDA

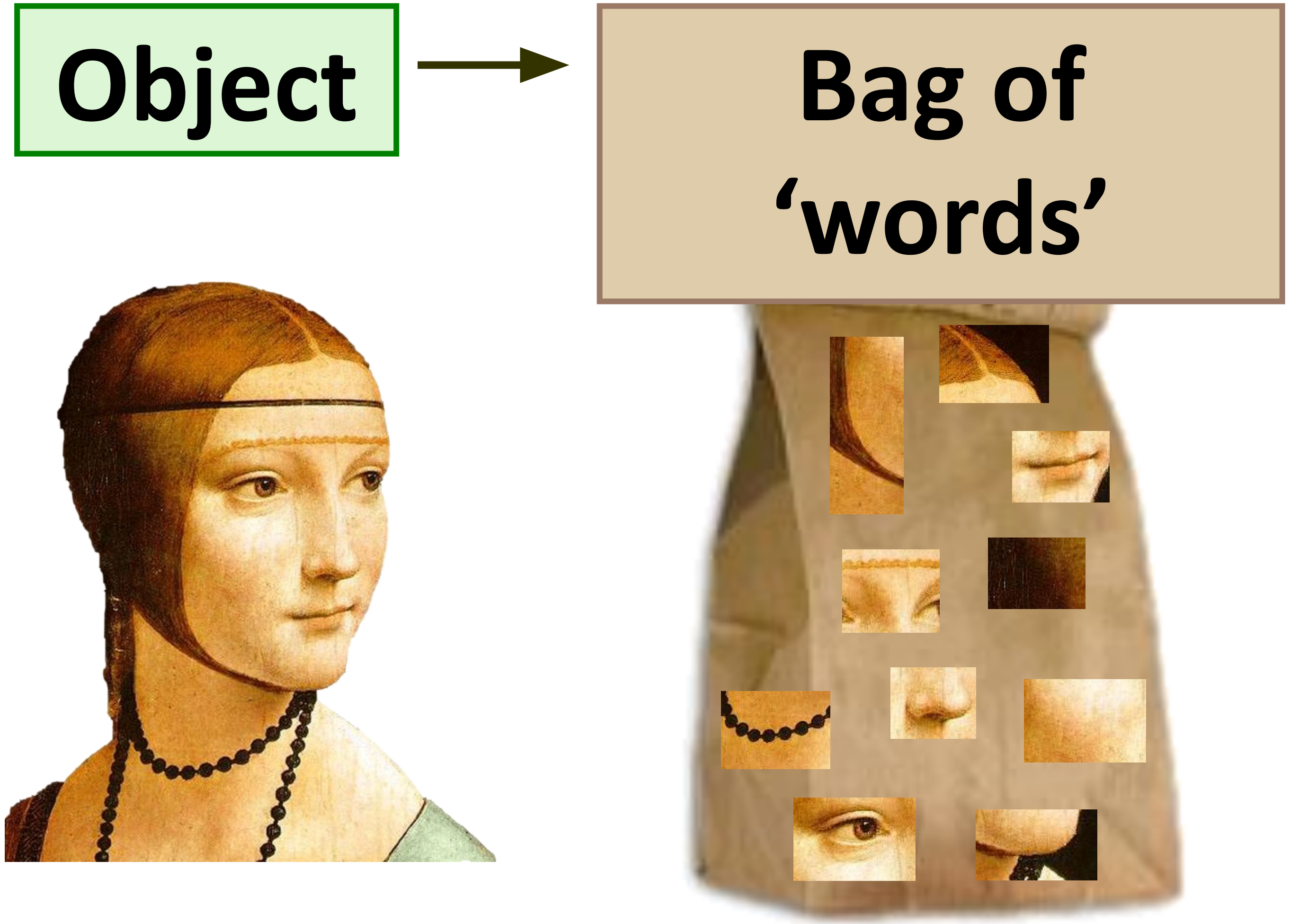
- PCA exploits the max scatter of the training images in face space
- LDA attempt to maximise the **between class scatter**, while minimising the **within class scatter**.

Today's agenda

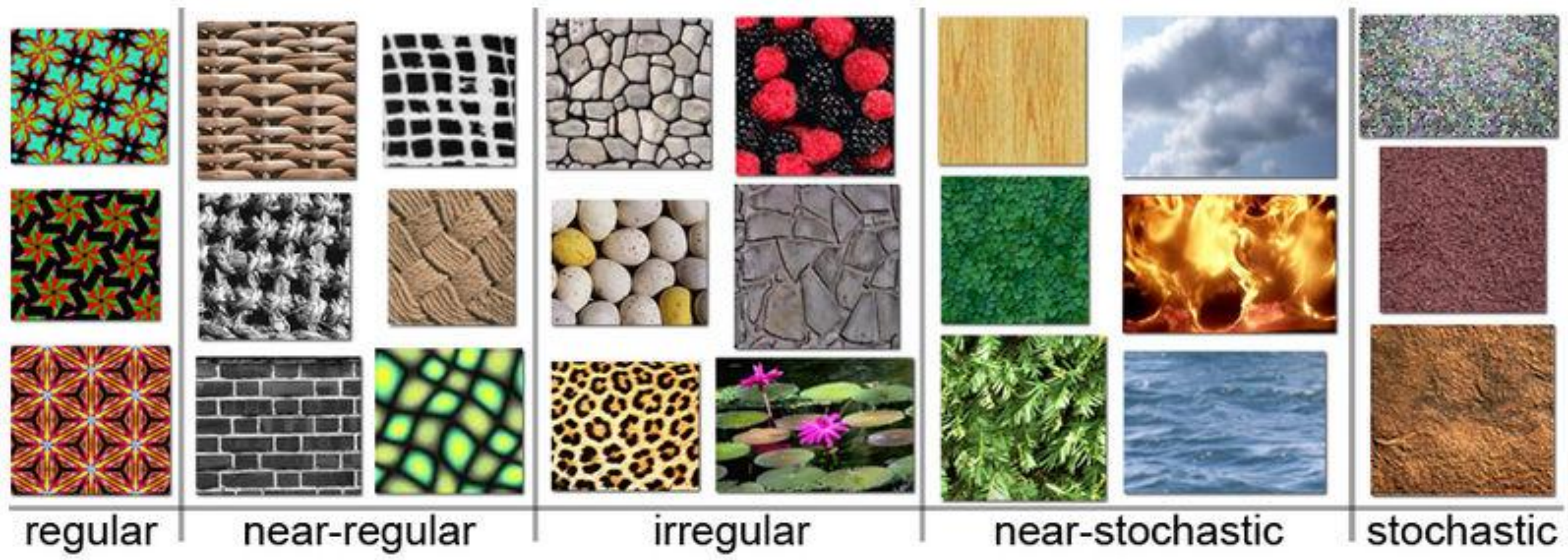
- Principal Component Analysis (PCA)
- Using PCA for computer vision: Eigenfaces
- Linear Discriminant Analysis (LDA)
- Visual bag of words (BoW)
- Spatial pyramids

Main idea: create a vocabulary of filters that would be able to recognize patches of specific objects

The size of the vocabulary will determine the size of the feature dimension.



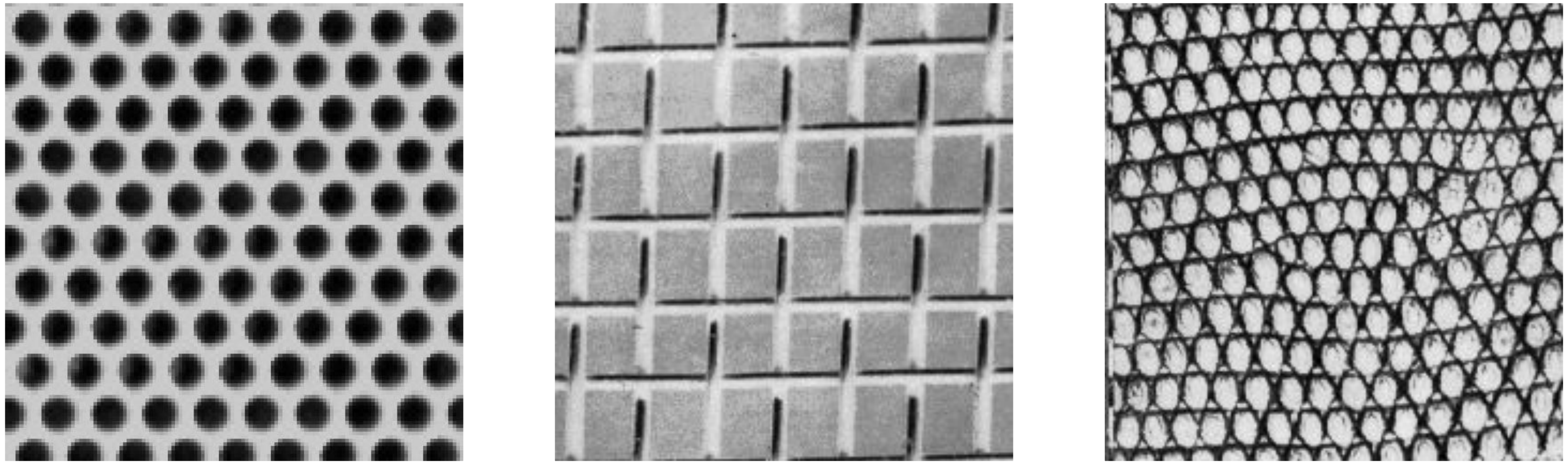
The idea originated from: **Texture Recognition**

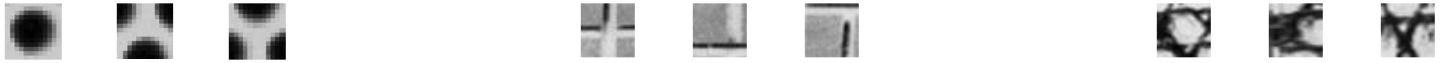


Example textures (from Wikipedia)

The idea originated from: Texture Recognition

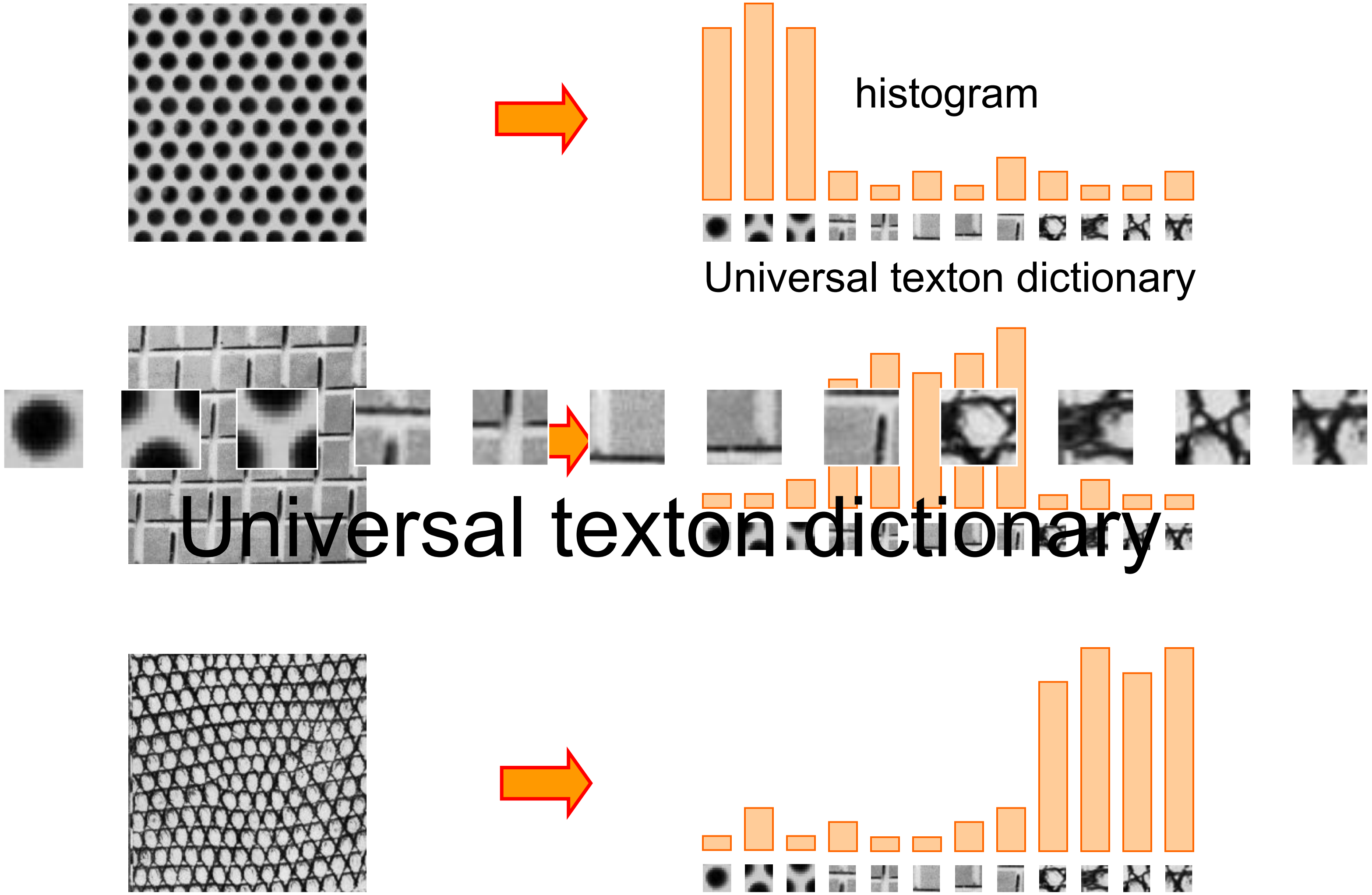
- Texture is characterized by the repetition of certain patches



Vocabulary: 

Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

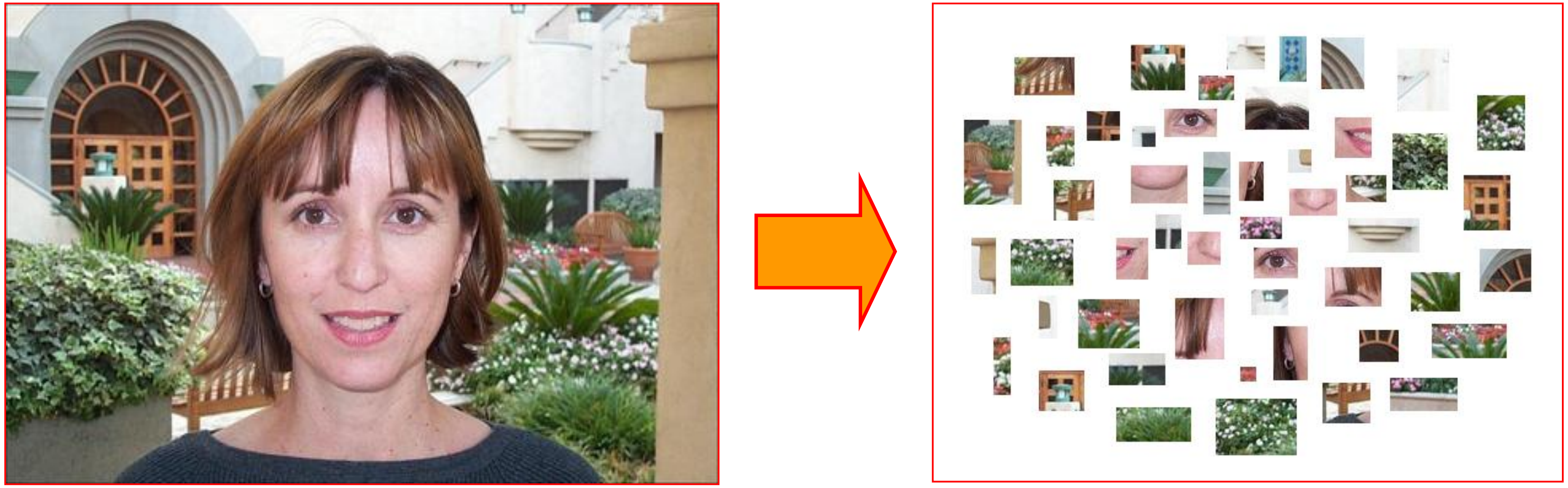
Every image is represented as fixed sized histogram of the number of times a patch appears



A similar idea is also used in natural language processing and called: Bag-of-words models

- Every word document is represented as the frequencies of words from a fixed vocabulary Salton & McGill (1983)

Visual bag of words for object recognition



face, flowers, building

- Works pretty well for recognition and for enabling image retrieval

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

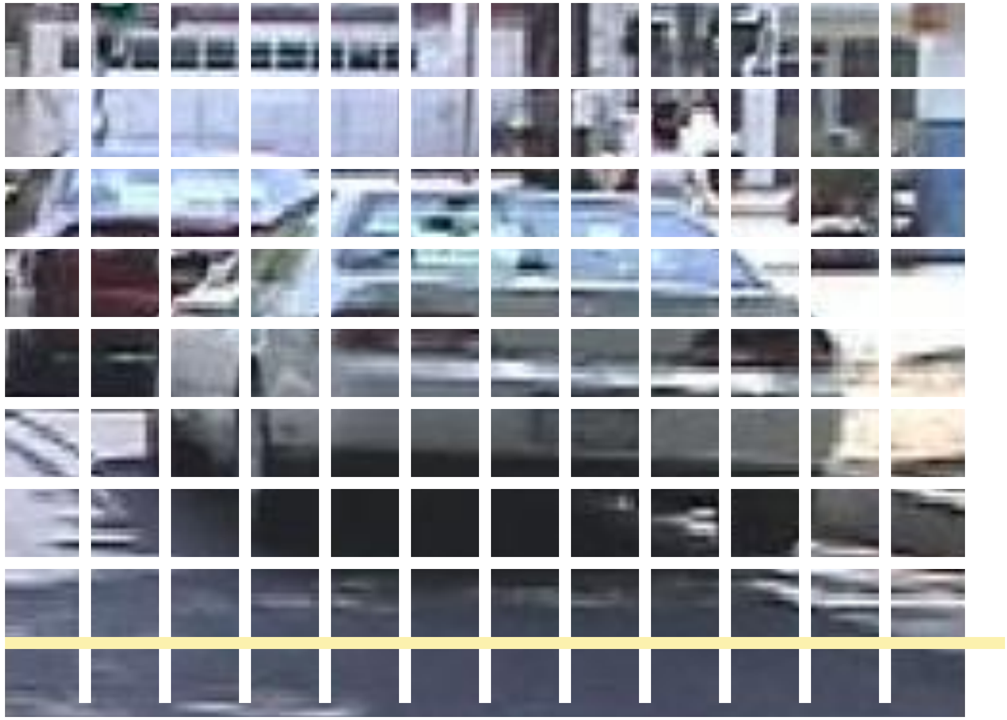
Bag of features

- First, take a bunch of images, extract features, and build up a "visual vocabulary" – a list of common features

- Given a new image, extract features and build a histogram of visual bag of words
  - for each patch in the image, find the closest visual word in the vocabulary and increment its corresponding value in the histogram

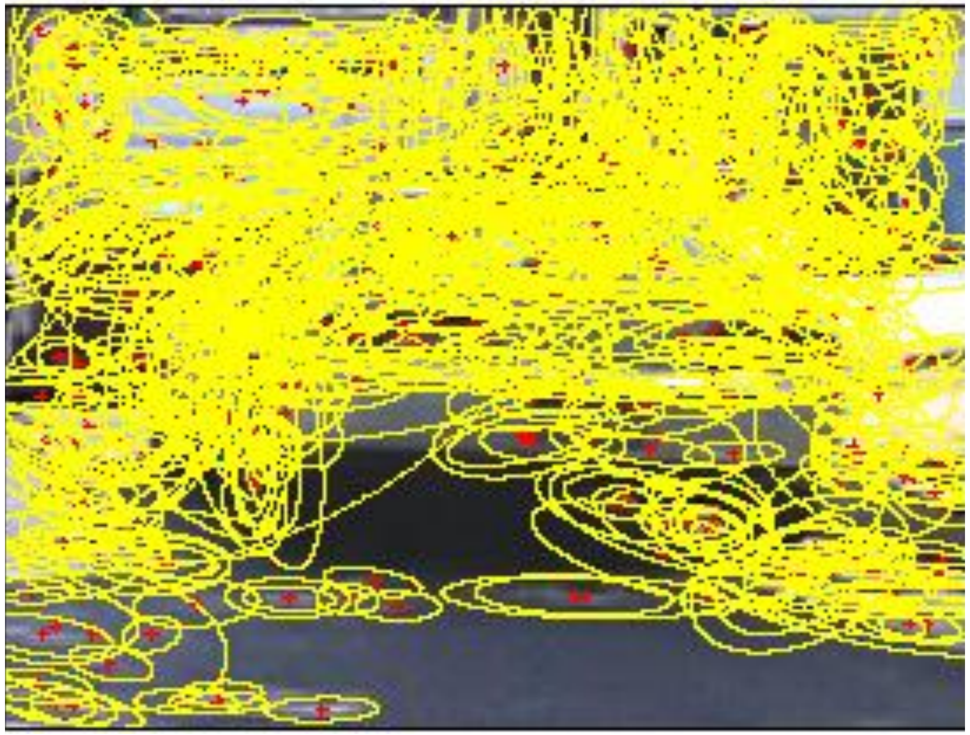
Step 1. Choose patches in a training dataset of images

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005



Step 1. Choose patches in a training dataset of images

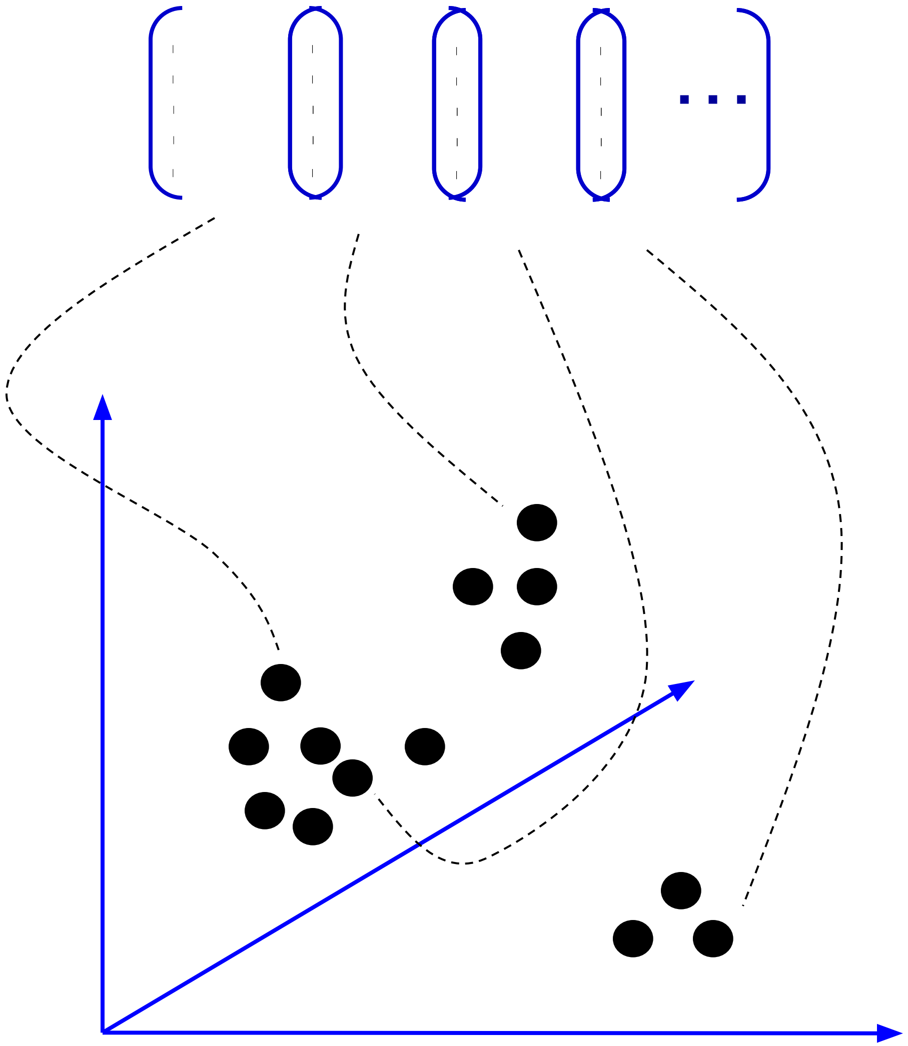
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005



Step 1. Choose patches in a training dataset of images

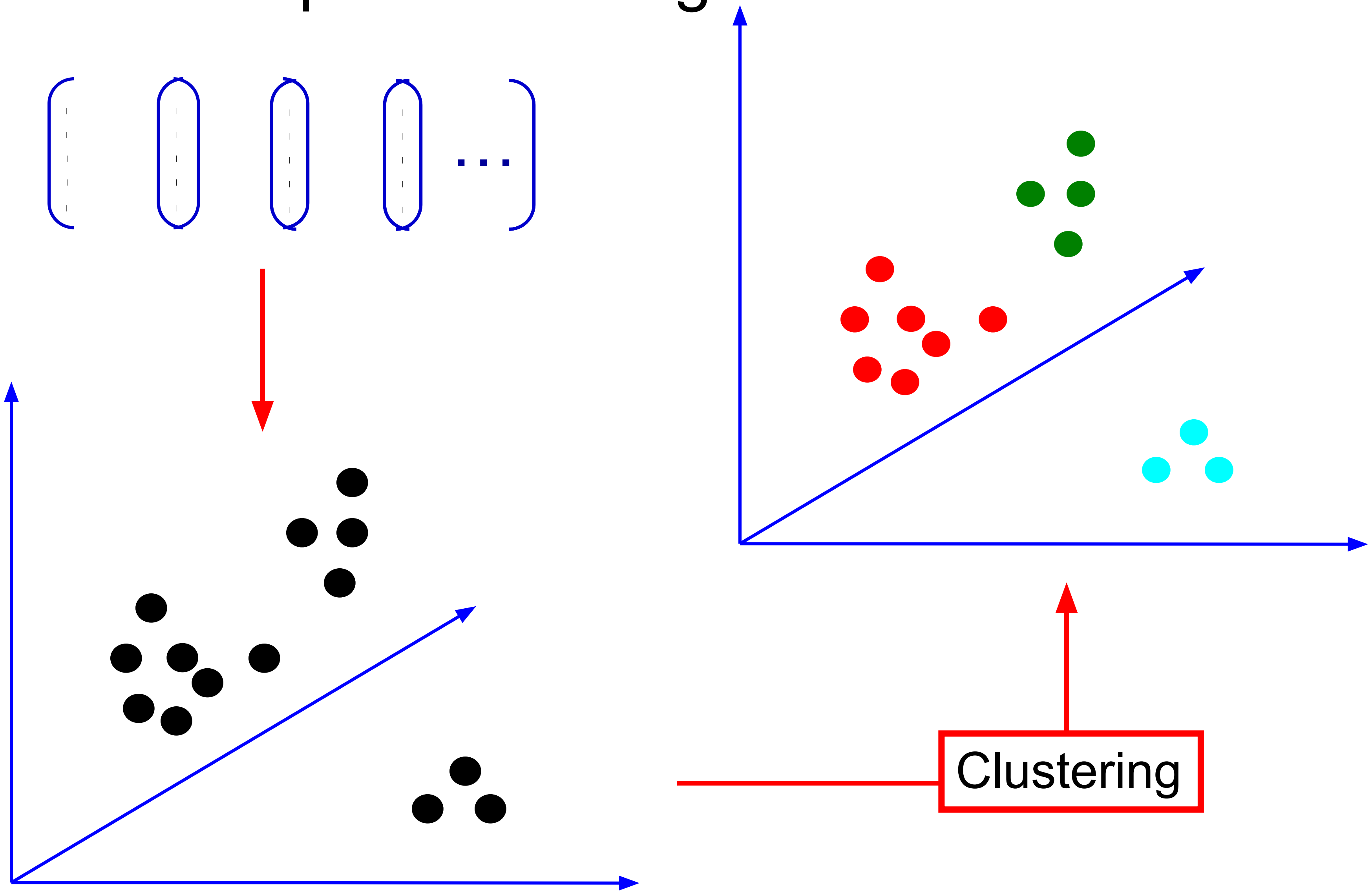
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches (Barnard et al. 2003)

Step 2. Cluster the patches using k-means

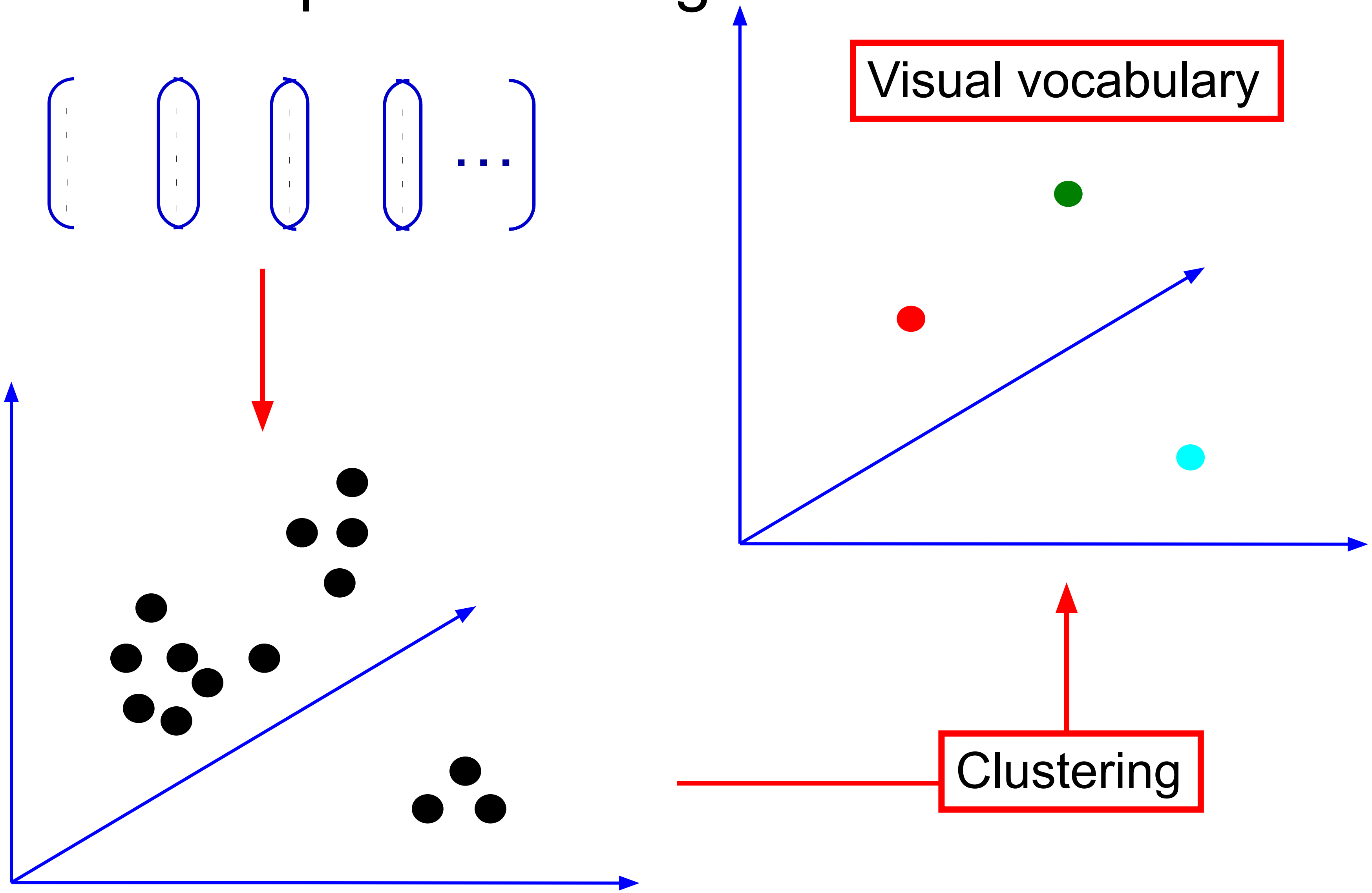


The k in k-means is the size of the vocabulary. It will determine the size of the features

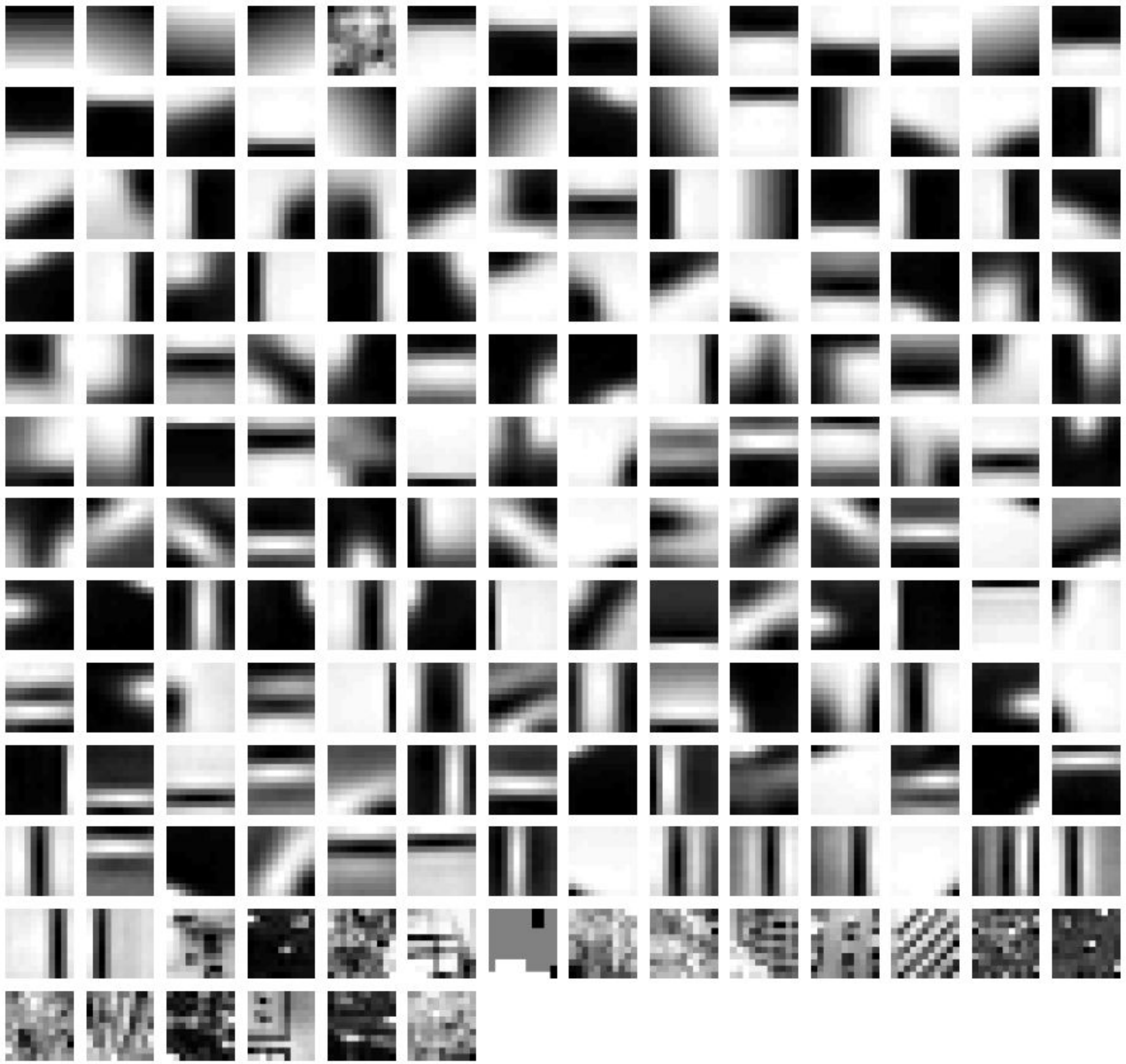
Step 2. Cluster the patches using k-means



Step 2. Cluster the patches using k-means

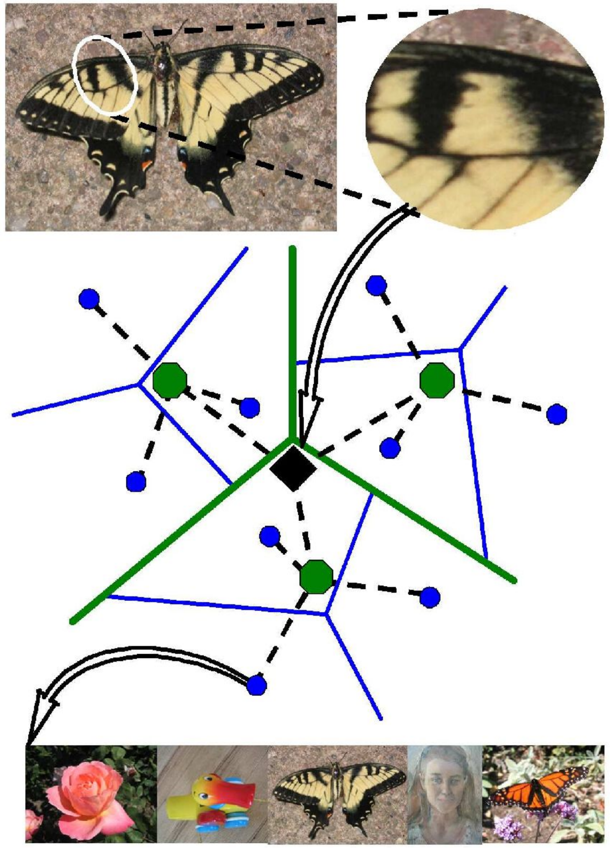


Example visual vocabulary



Visual vocabularies: Issues

- How to choose vocabulary size?
  - **Too small**: Most patches are just noisy and not useful
  - **Too large**: overfits to training images and doesn't generalize
- **Computational efficiency**
  - Try to choose as small of a vocabulary size as possible to reduce curse of dimensionality



Step 3. Convert every image into a histogram

- Every image now becomes a k-dimensional histogram representation.
- We can use these features for any recognition task.

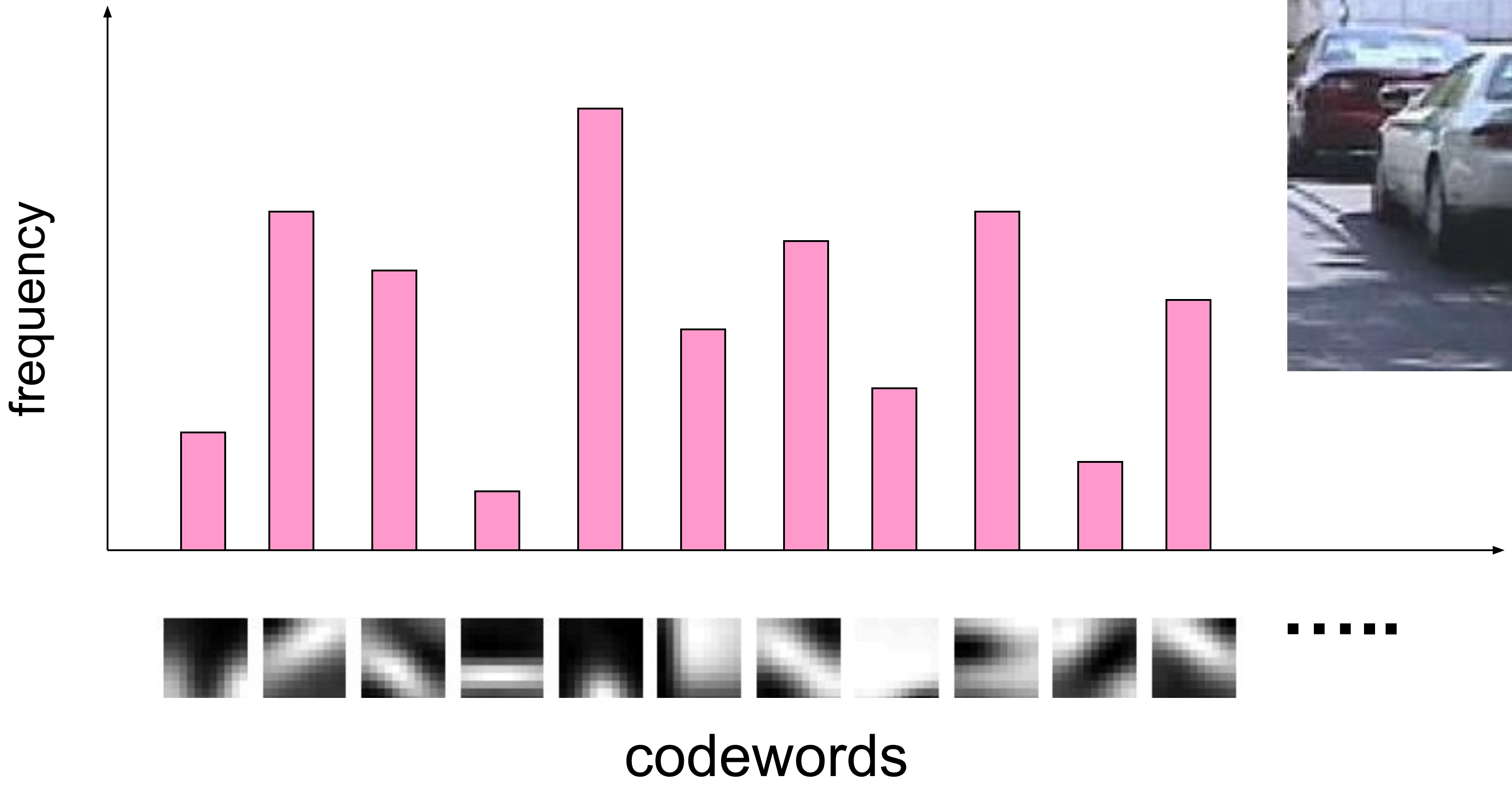
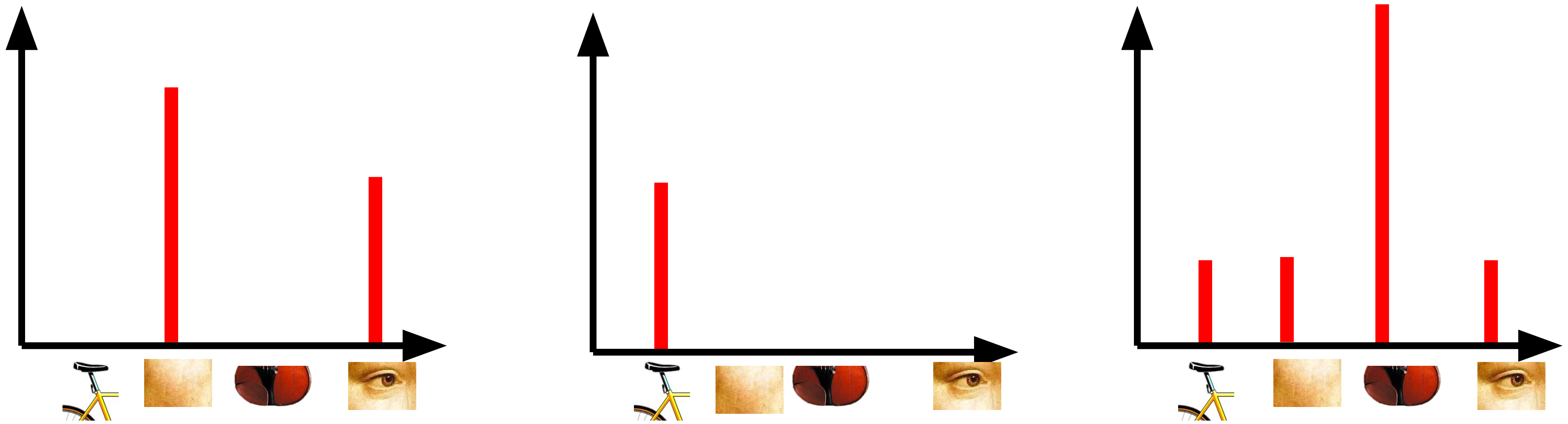


Image classification

- A histogram of bag-of-words features are very good at distinguishing between different categories.
- E.g., first image is a face, second is a bike, third is an instrument



Uses of BoW representation

- Treat as feature vector for standard classifier
  - e.g k-nearest neighbors

Visual bag of words works quite well for a fixed set of categories

class	bag of features Zhang et al. (2005)	bag of features Willamowski et al. (2004)	Parts-and-shape model 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

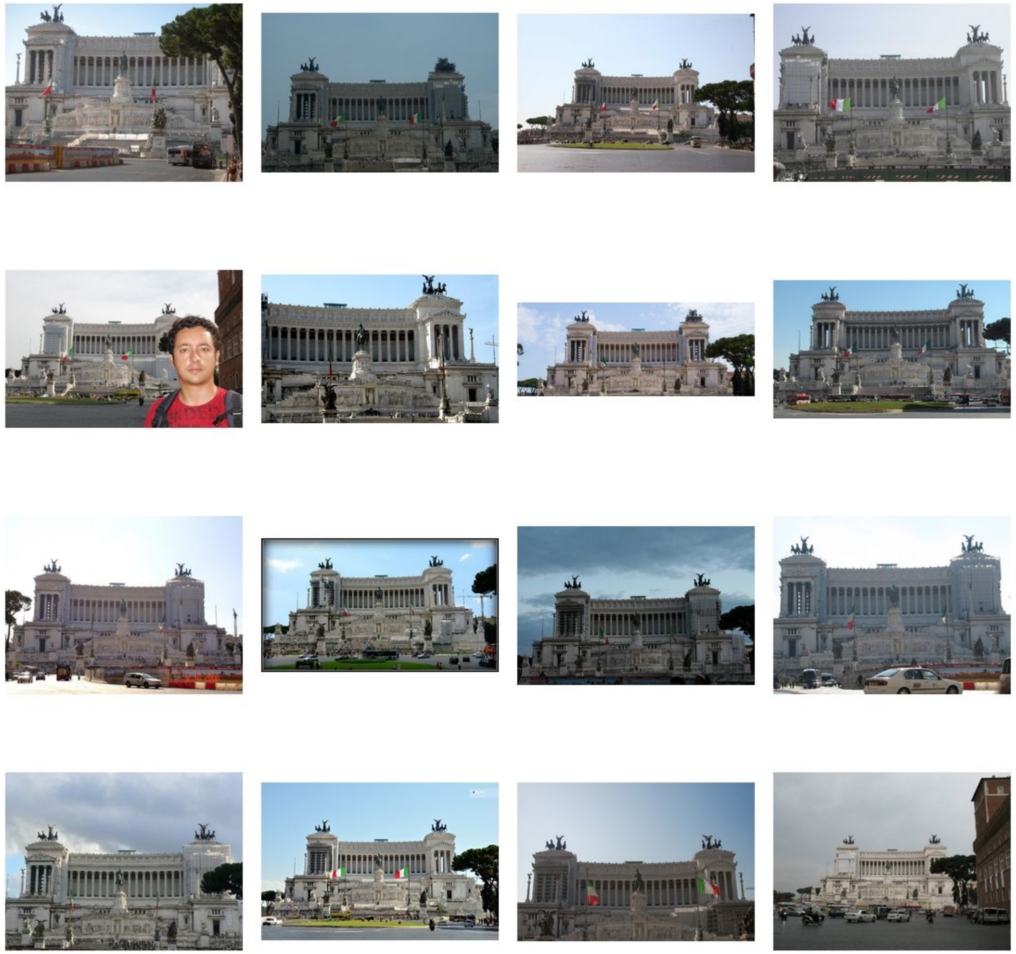
Bag of words can also enable search

query image

top 6 results

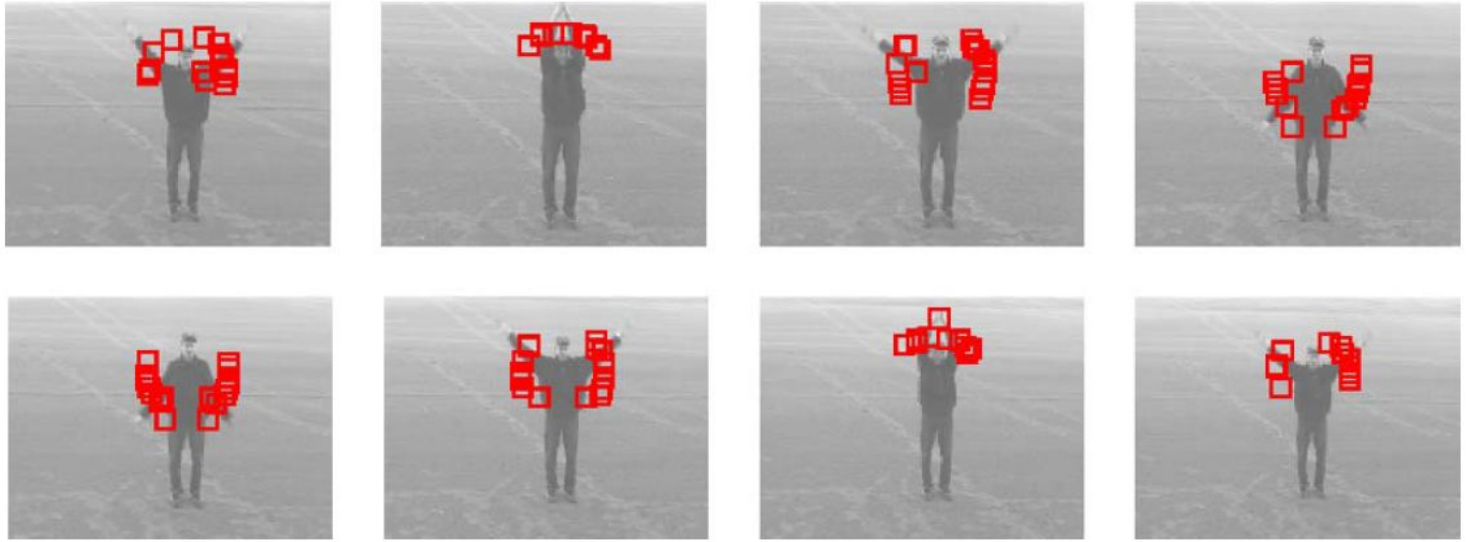
- Cons:
  - performance degrades as the database grows

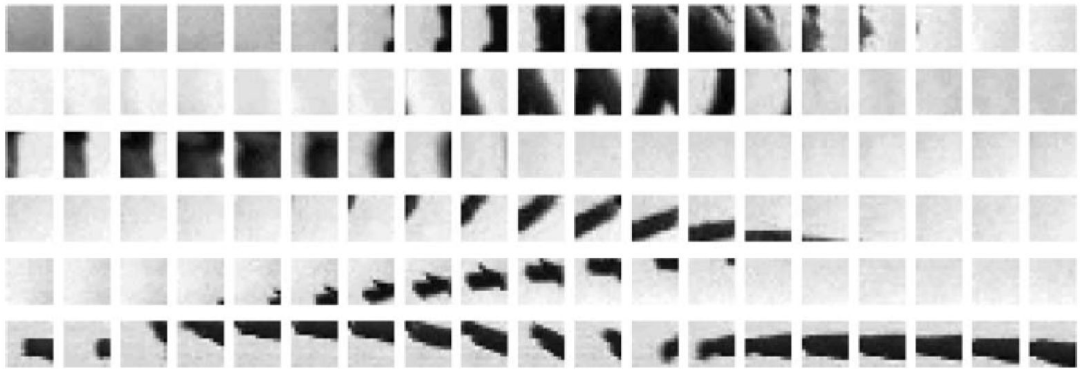
Example bag-of-words matches



Example bag-of-words matches

Bags of words in videos





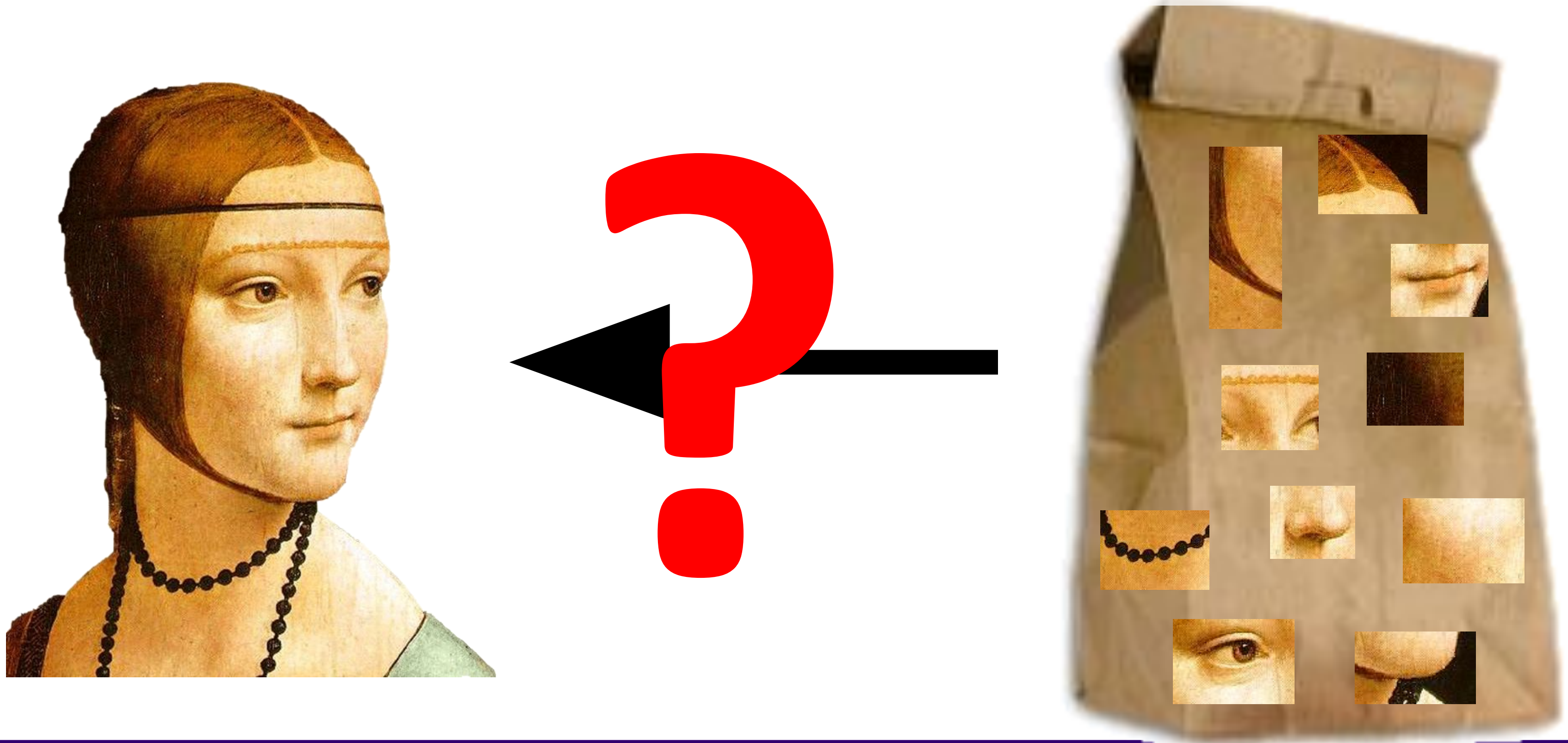
Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei, **Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words**, IJCV 2008.

Today's agenda

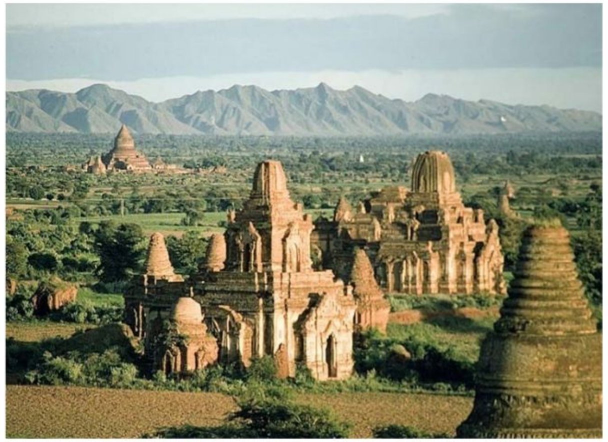
- Principal Component Analysis (PCA)
- Using PCA for computer vision: Eigenfaces
- Linear Discriminant Analysis (LDA)
- Visual bag of words (BoW)
- Spatial pyramids

How do we choose the size of the patches?

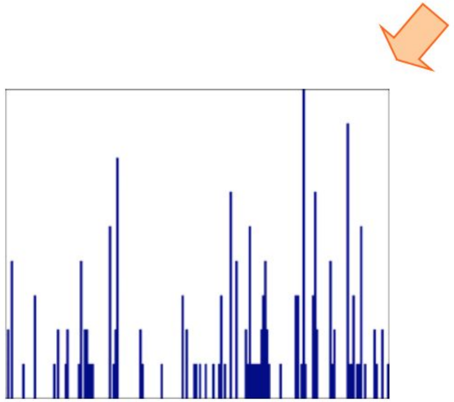
- If the object is close to the camera, larger patches are better
- If the object is really far away, smaller patches are better for finding it.



Bag of words + pyramids

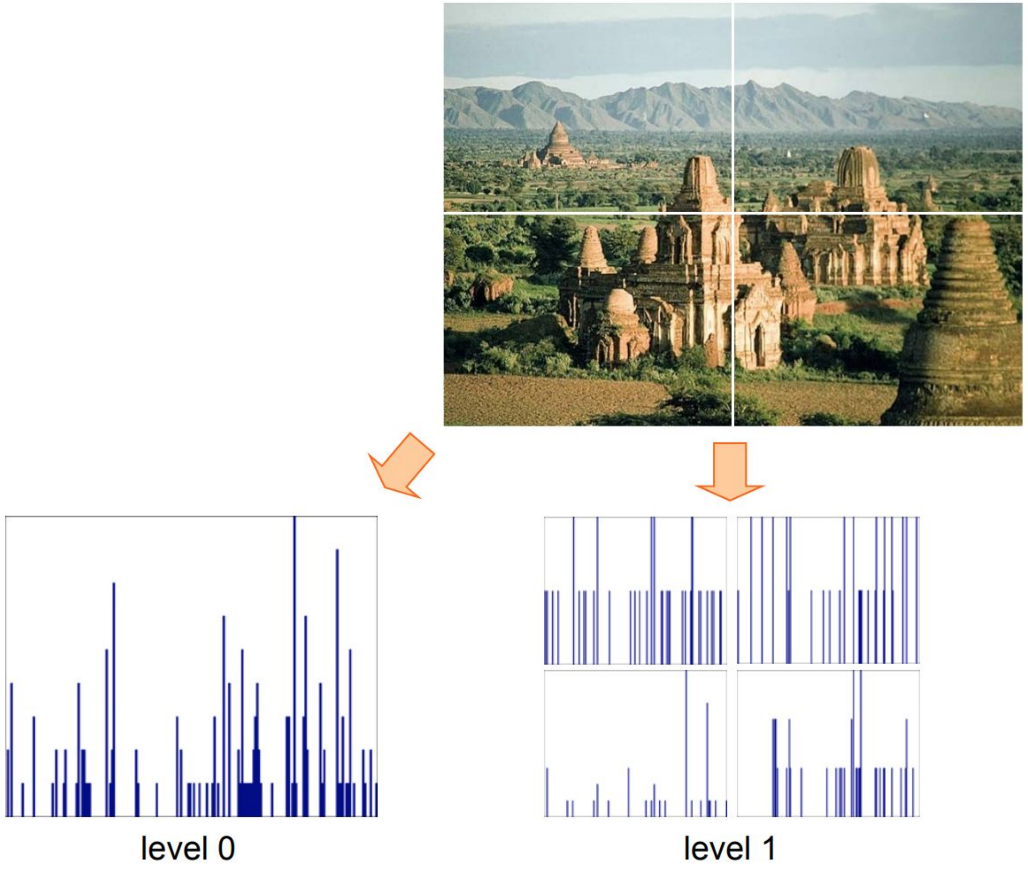


Locally orderless representation at several levels of spatial resolution



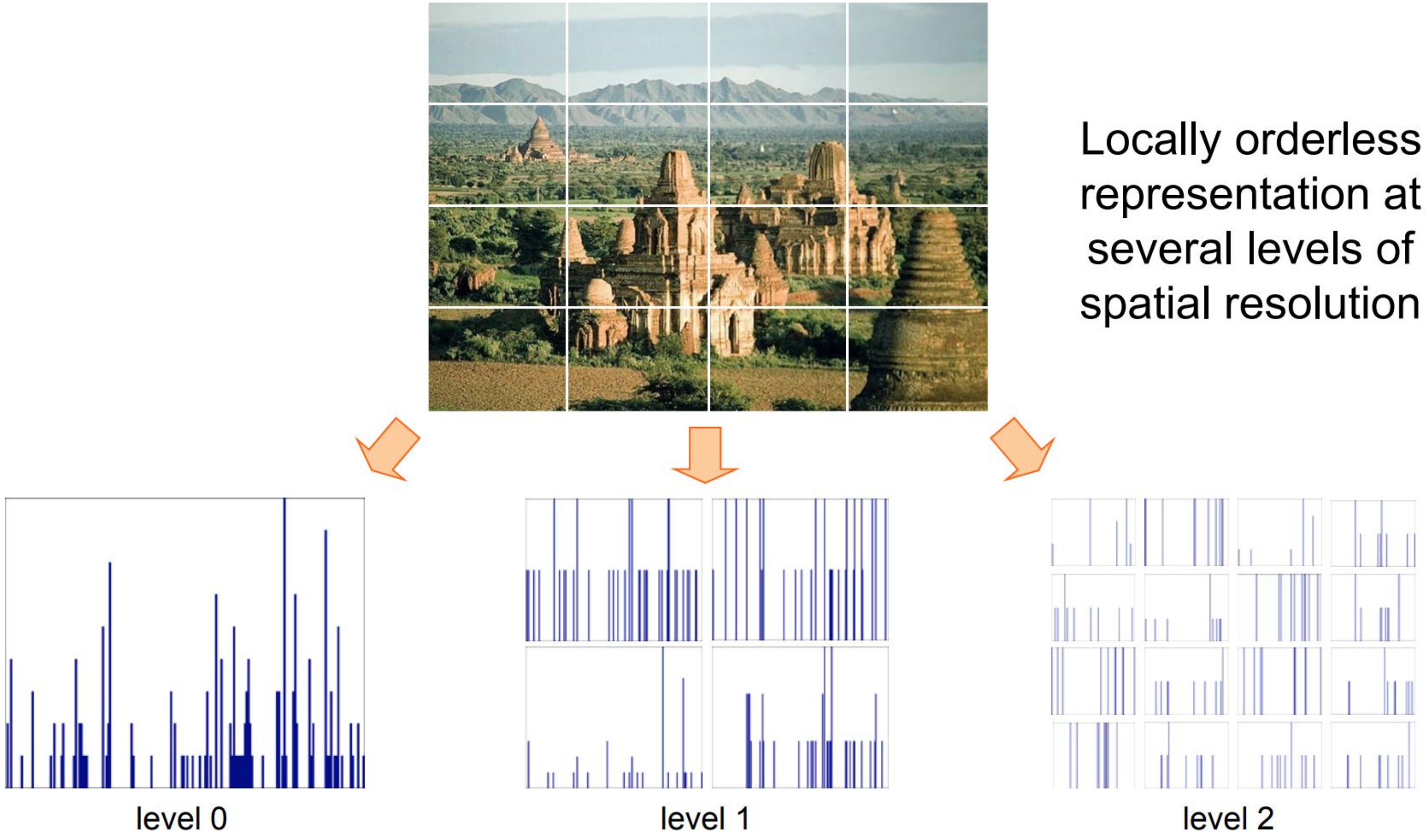
level 0

Bag of words + pyramids



Locally orderless representation at several levels of spatial resolution

Bag of words + pyramids



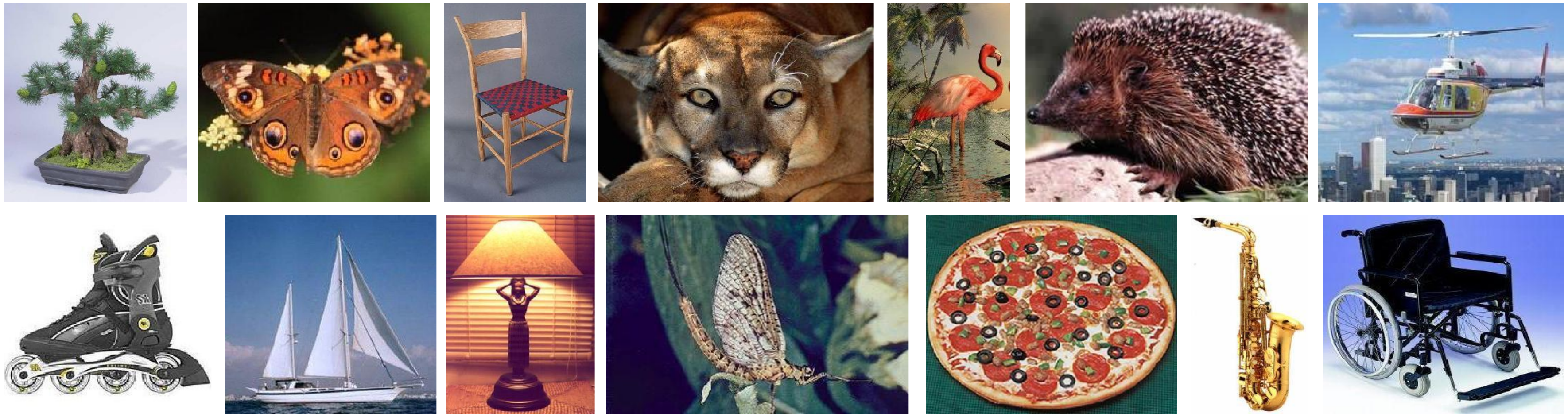
Locally orderless representation at several levels of spatial resolution

Pyramids are a general idea that is used in all vision models today (including swin transformers)

- Very useful for representing images.
- Pyramid is built by using multiple copies of image.
- Each level in the pyramid is 1/4 of the size of previous level.

Caltech101 dataset

Multi-class classification results (30 training images per class)



Level	Single-level	Pyramid	Single-level	Pyramid
0	15.5 ±0.9		41.2 ±1.2	
1	31.4 ±1.2	32.8 ±1.3	55.9 ±0.9	57.0 ±0.8
2	47.2 ±1.1	49.3 ±1.4	63.6 ±0.9	**64.6** ±0.8
3	52.2 ±0.8	**54.0** ±1.1	60.3 ±0.9	64.6 ±0.7

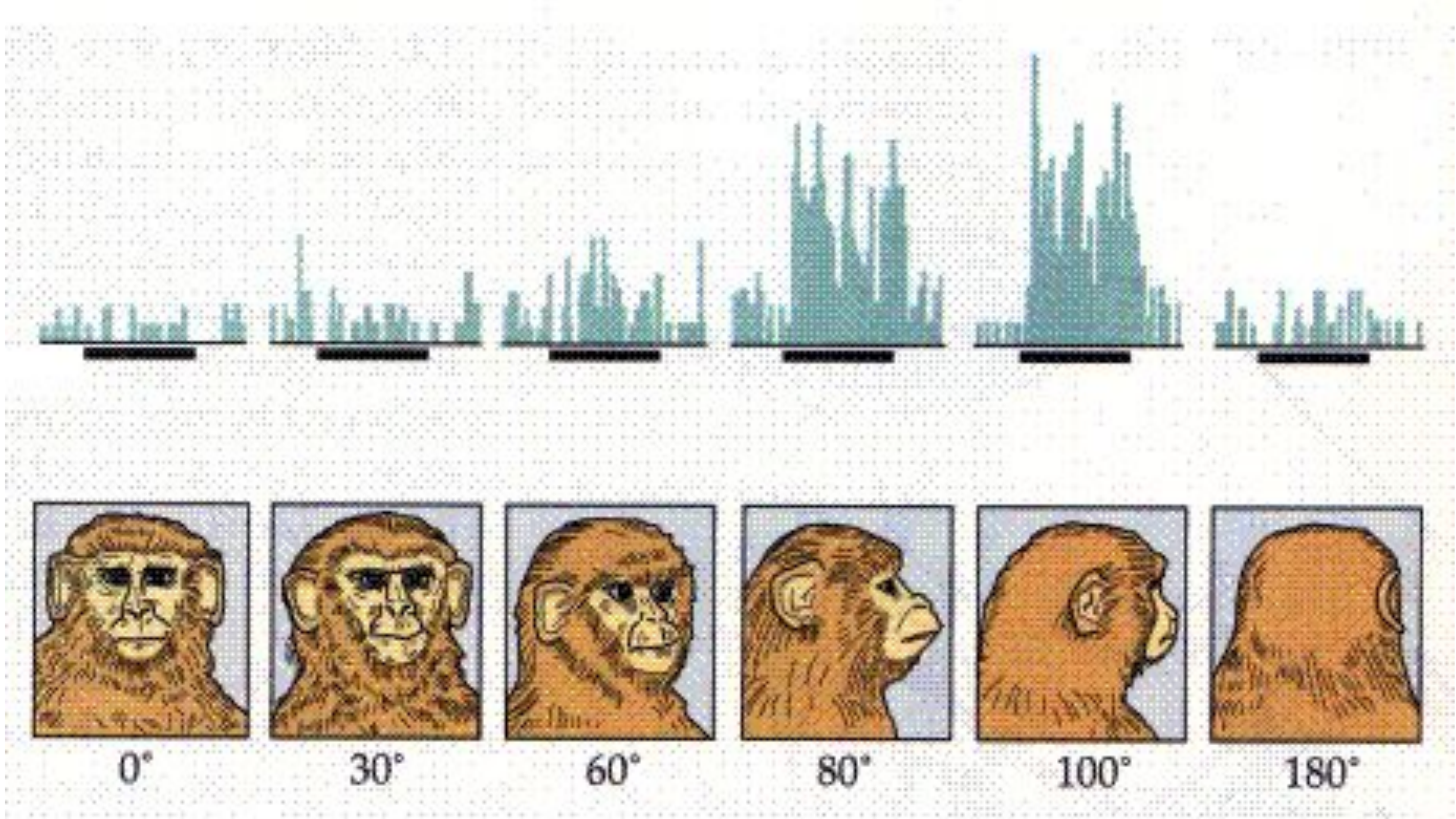
Today’s agenda

- Principal Component Analysis (PCA)
- Using PCA for computer vision: Eigenfaces
- Linear Discriminant Analysis (LDA)
- Visual bag of words (BoW)
- Spatial pyramids

Next lecture

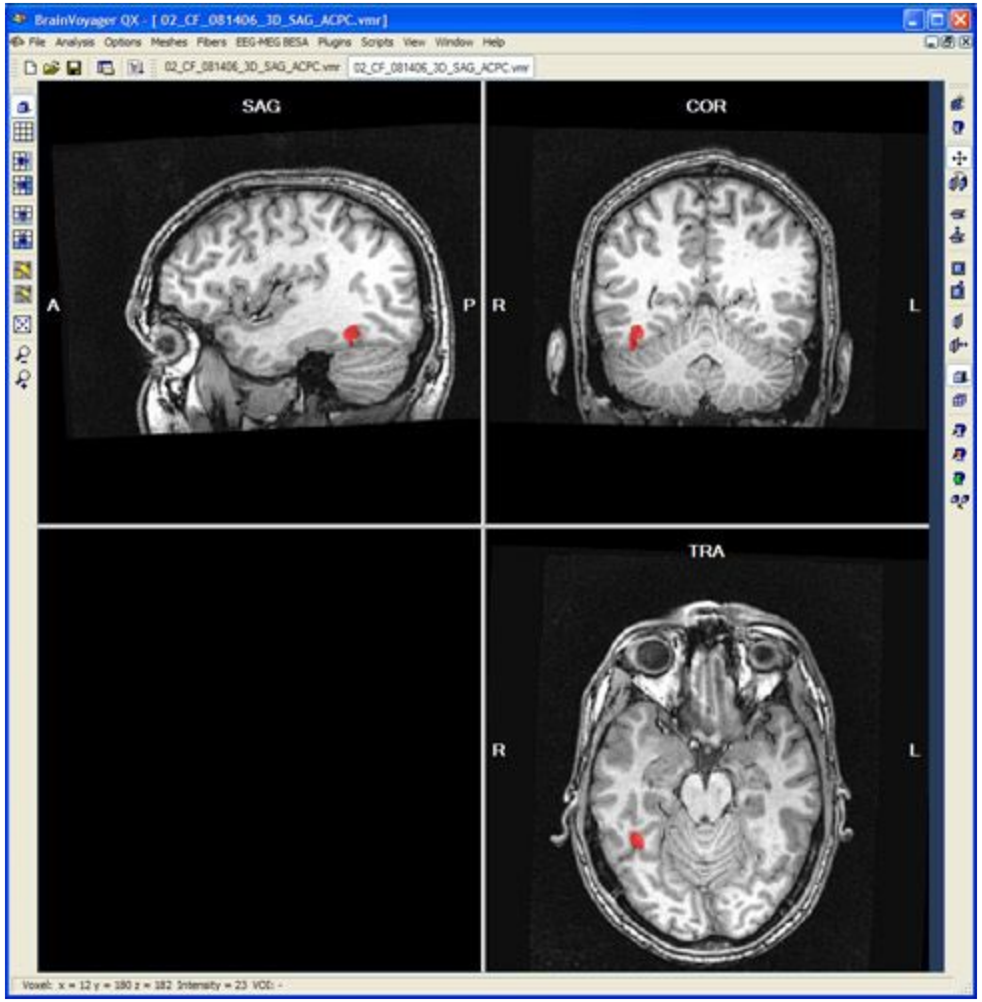
Object detection

“Faces” in the brain



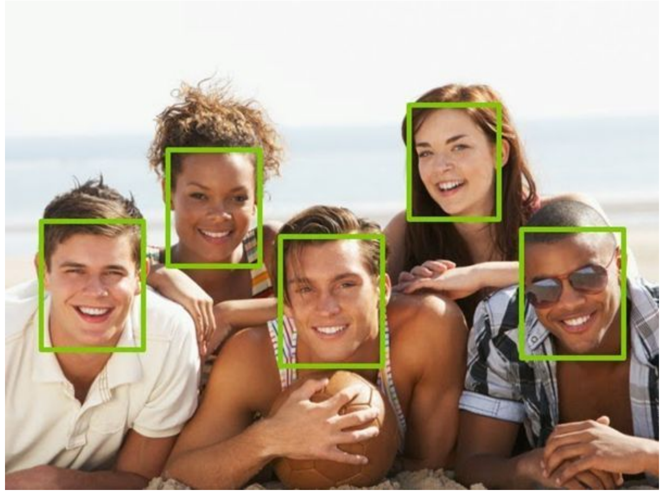
Courtesy of Johannes M. Zanker

“Faces” in the brain fusiform **face area**

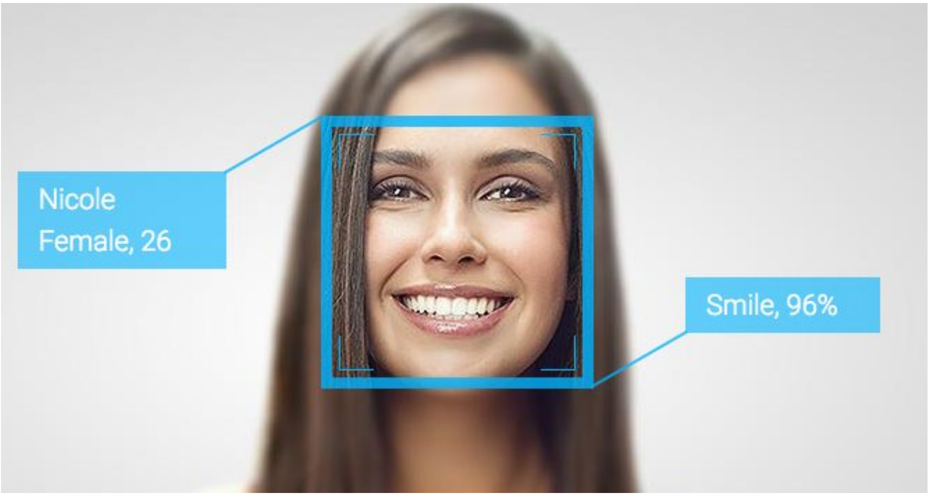


Kanwisher, et al. 1997

Detection versus Recognition



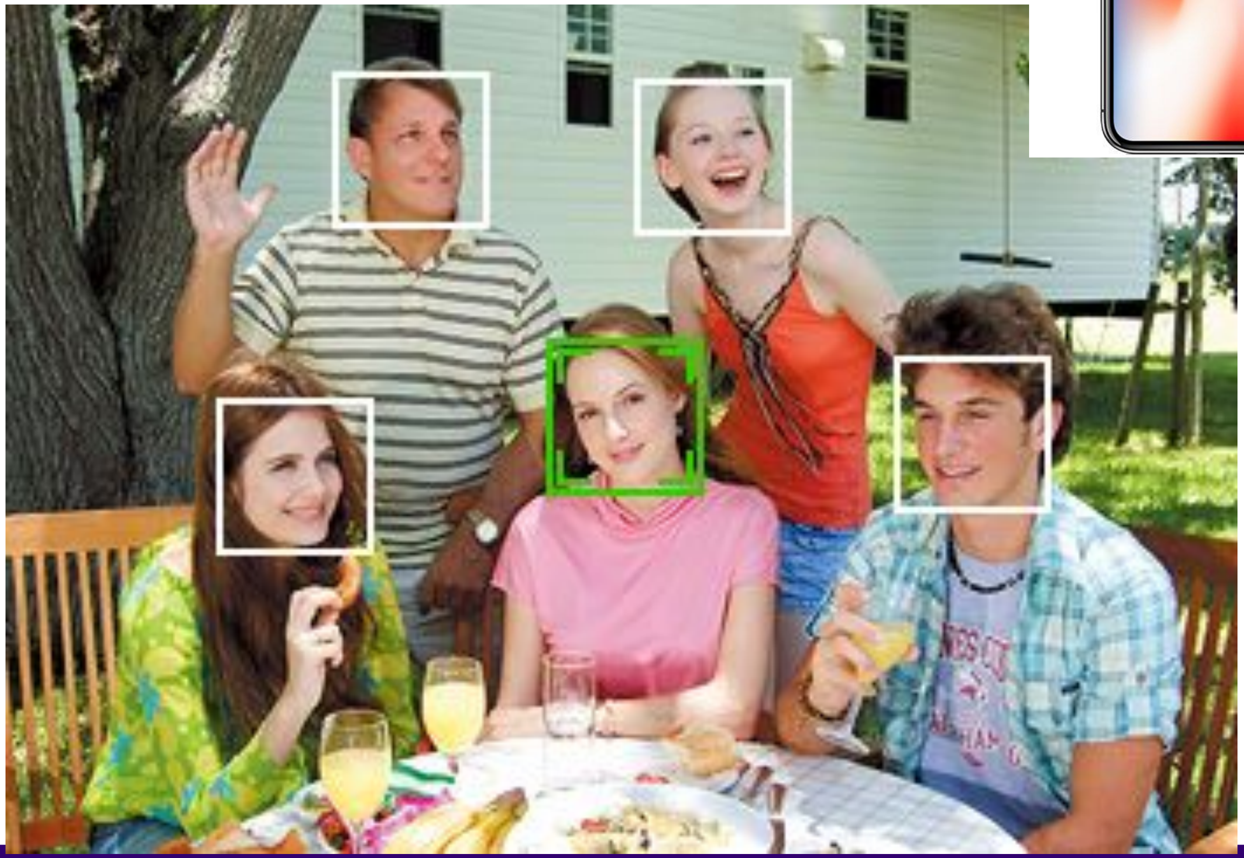
Detection finds the faces in images



Recognition recognizes WHO the person is

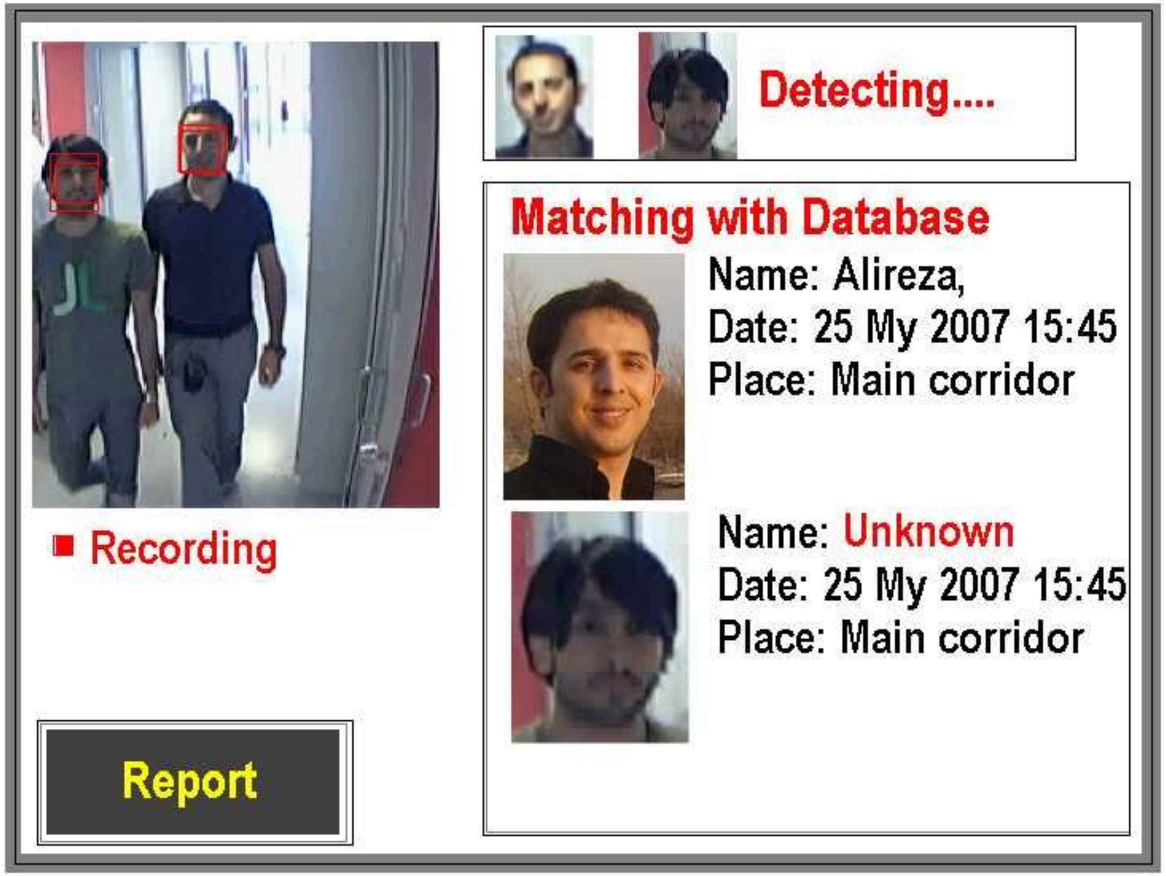
Face Recognition

- Digital photography



Face Recognition

- Digital photography
- Surveillance

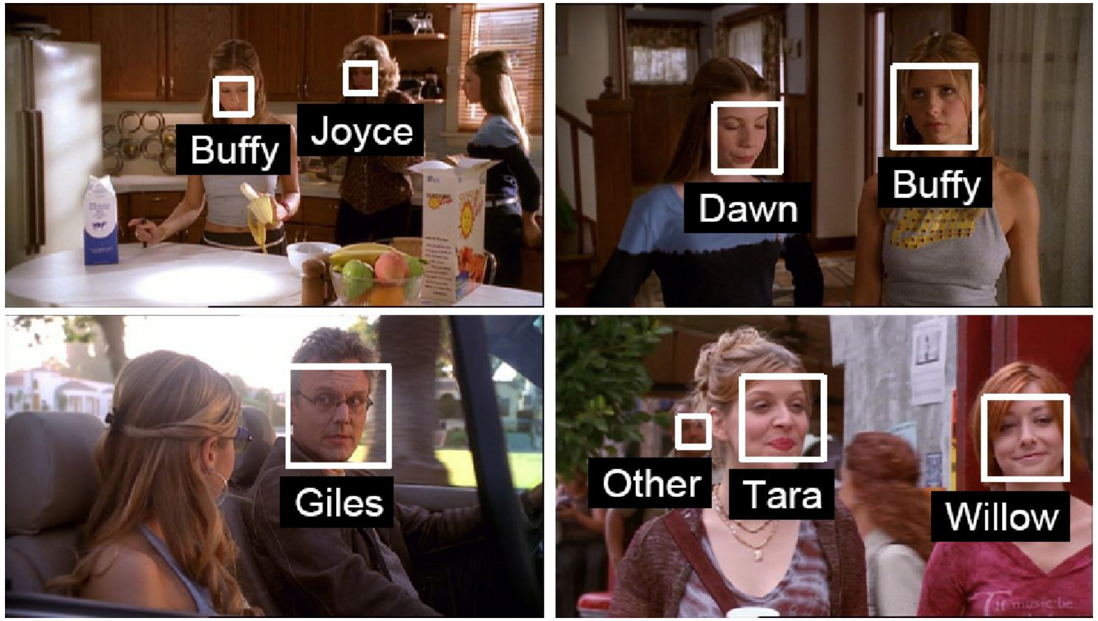


Face Recognition

- Digital photography
- Surveillance
- Album organization

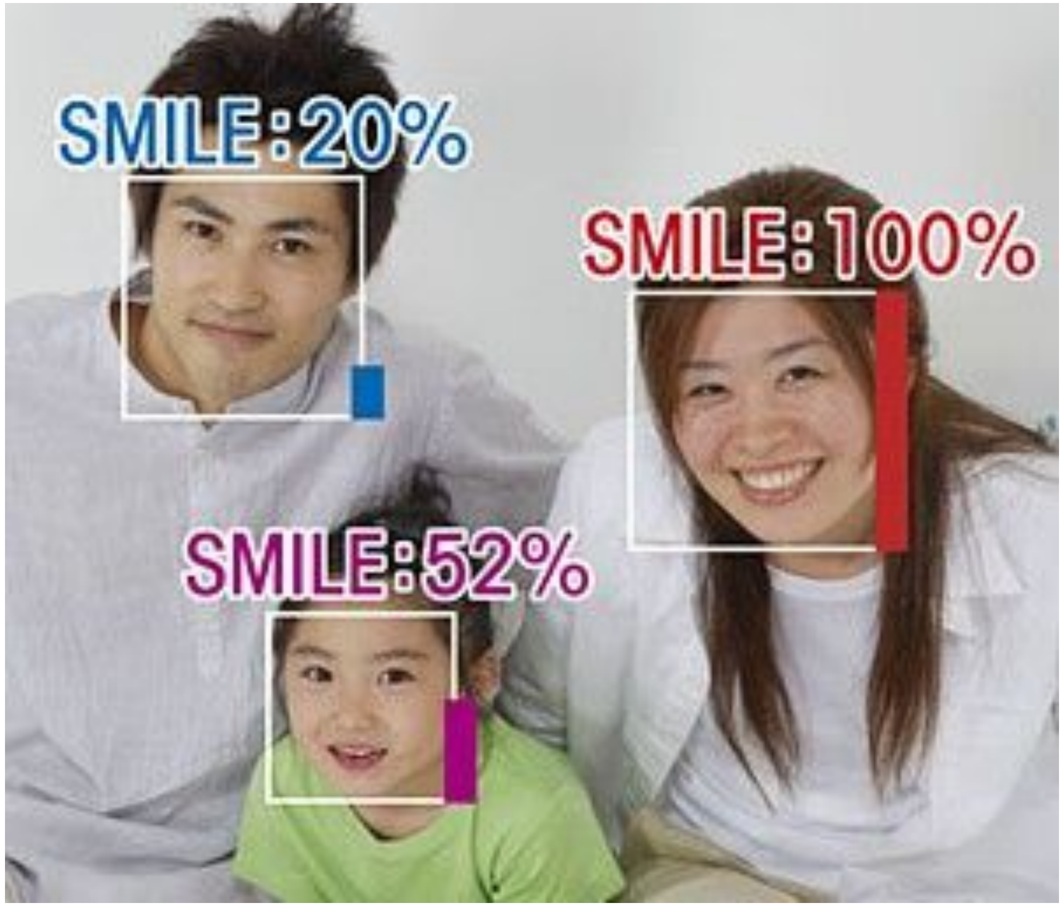
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.



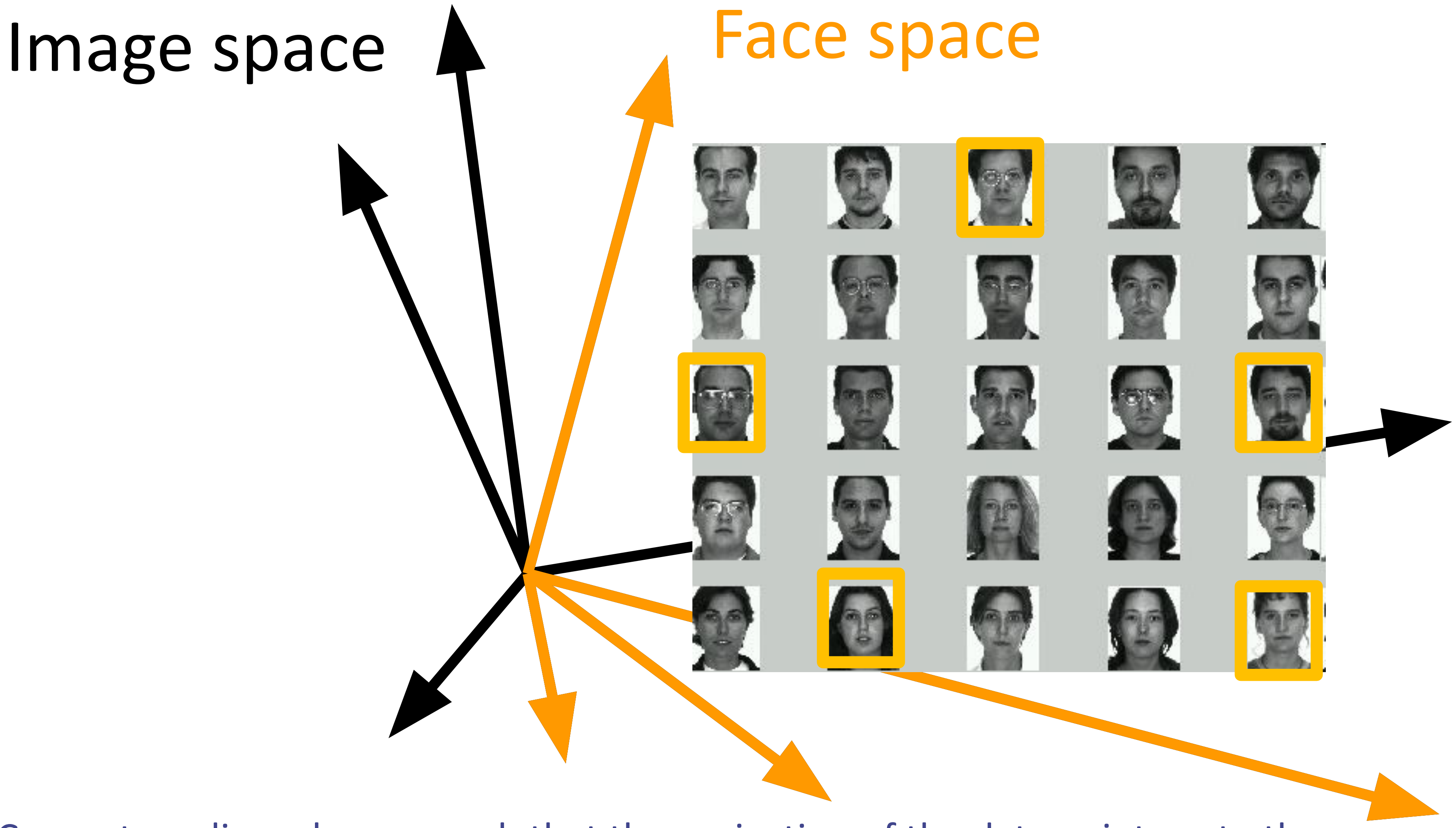
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.
- Emotions and expressions



Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.
- Emotions and expressions
- Security/warfare
- Tele-conferencing
- Etc.



- Compute n-dim subspace such that the projection of the data points onto the subspace has the largest variance among all n-dim subspaces.
- Maximize the scatter of the training images in face space

Key Idea

- So, compress them to a low-dimensional subspace that captures key appearance characteristics of the visual DOFs.

- USE PCA for estimating the sub-space (dimensionality reduction)

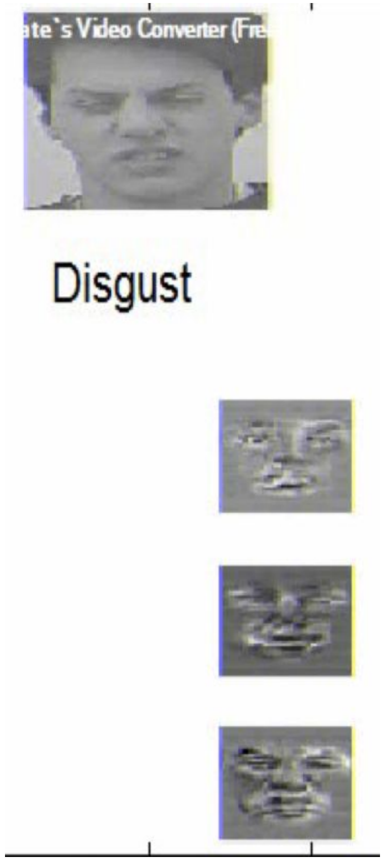
- Compare two faces by projecting the images into the subspace and measuring the EUCLIDEAN distance between them.

Besides face recognitions, we can also do Facial expression recognition

Happiness subspace (method A)

Disgust subspace (method A)

Facial Expression Recognition Movies (method A)



Variables

- N Sample images: $\{x_1, \cdots, x_N\}$
- C classes: $\{Y_1, Y_2, \ldots, Y_c\}$
- Average of each class: $\mu_i = \frac{1}{N_i} \sum_{x_k \in Y_i} x_k$
- Average of all data: $\mu = \frac{1}{N} \sum_{k=1}^{N} x_k$

Scatter Matrices

- Scatter of class i: $$S_i = \sum_{x_k \in Y_i} (x_k - \mu_i)(x_k - \mu_i)^T$$
- Within class scatter: $$S_W = \sum_{i=1}^{c} S_i$$
- Between class scatter: $$S_B = \sum_{i=1}^{C} \sum_{j \neq i} (\mu_i - \mu_j)(\mu_i - \mu_j)^T$$

Mathematical Formulation

- Recall that we want to learn a projection W such that the projection converts all the points from x to a new space z:

$$z = w^T x \qquad z \in \mathbf{R}^m \quad x \in \mathbf{R}^n$$

- After projection:
  - Between class scatter
  - Within class scatter

$$\tilde{S}_B = W^T S_B W$$
$$\tilde{S}_W = W^T S_W W$$

- So, the objective becomes:

$$W_{opt} = \arg\max_{\mathbf{W}} \frac{\left|\tilde{S}_B\right|}{\left|\tilde{S}_W\right|} = \arg\max_{\mathbf{W}} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

Mathematical Formulation

$$W_{opt} = \arg\max_{W} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

- Solve generalized eigenvector problem:

$$S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

Mathematical Formulation

- Solution: Generalized Eigenvectors

$$S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

- Rank of $\boldsymbol{W_{opt}}$ is limited
  - Rank(S_B) <= |C|-1
  - Rank(S_W) <= N-C

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary   Salton & McGill (1983)

US Presidential Speeches Tag Cloud
`http://chir.ag/phernalia/preztags/`

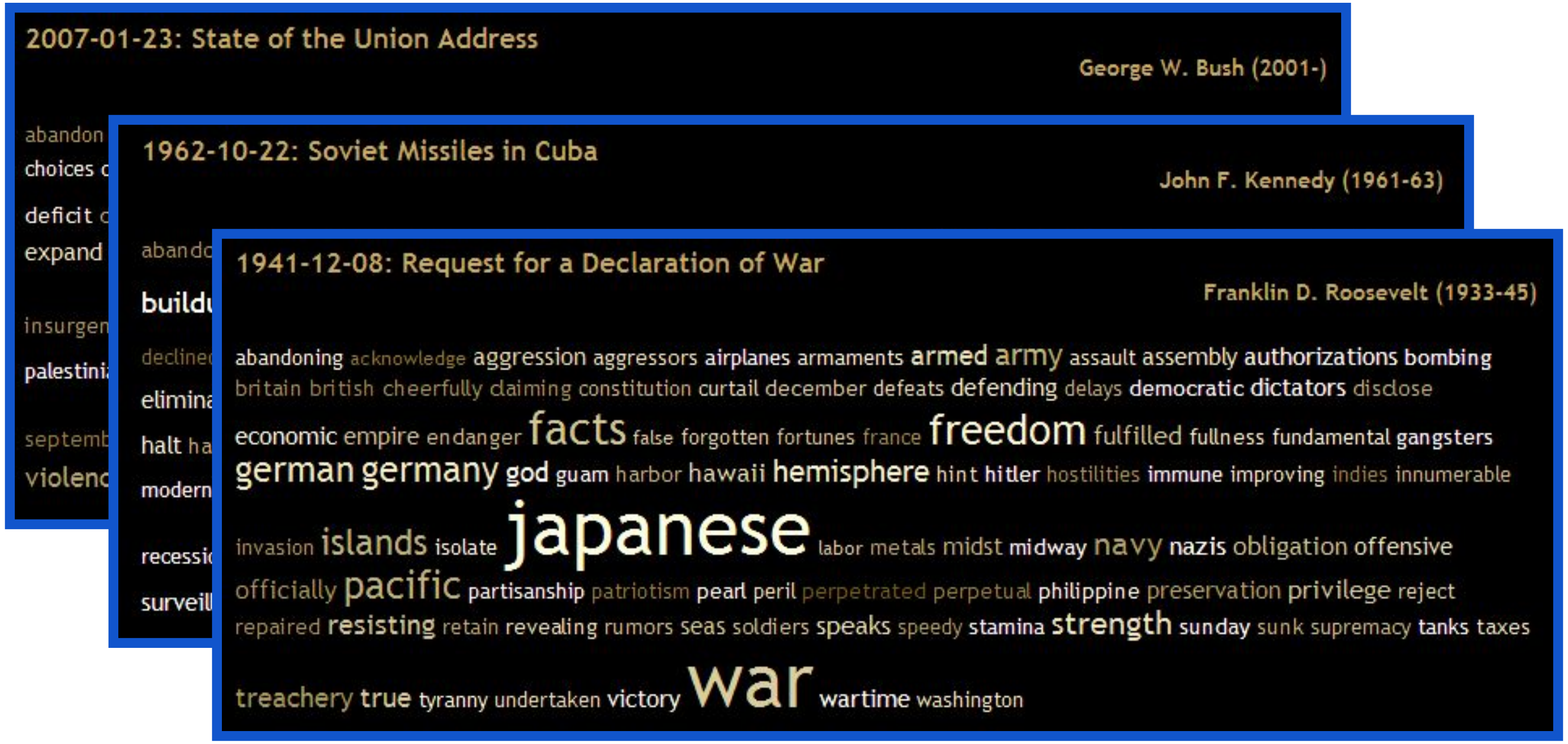
Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

US Presidential Speeches Tag Cloud
`http://chir.ag/phernalia/preztags/`

Origin 2: Bag-of-words models

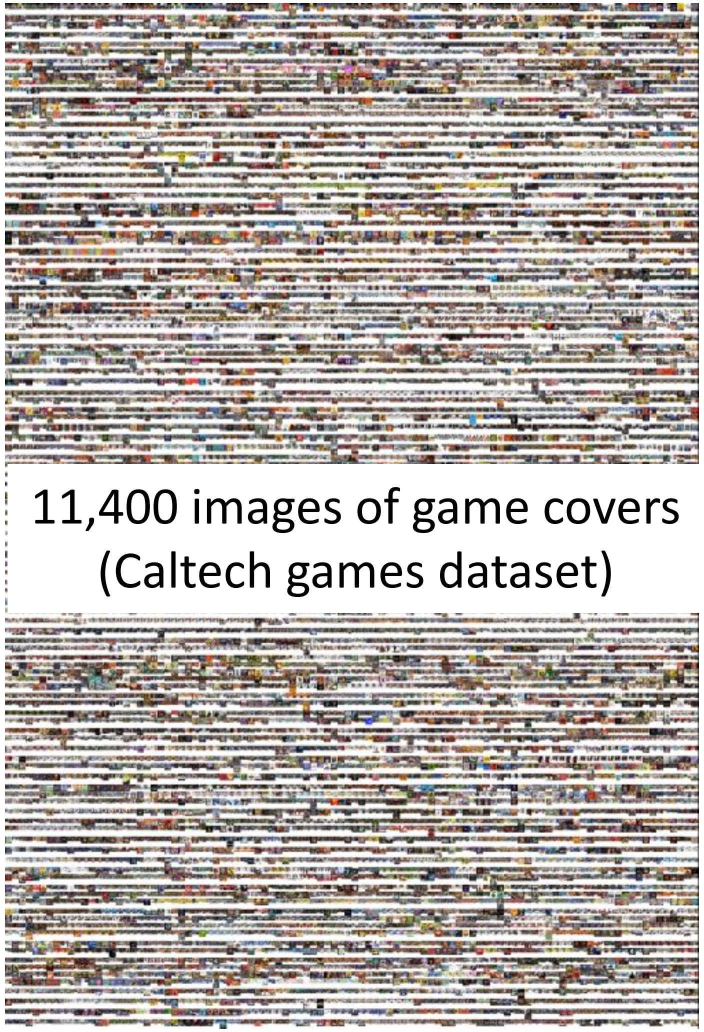
- Orderless document representation: frequencies of words from a dictionary   Salton & McGill (1983)



US Presidential Speeches Tag Cloud
`http://chir.ag/phernalia/preztags/`

Large-scale image matching

- Bag-of-words models have been useful in matching an image to a large database of object *instances*



11,400 images of game covers
(Caltech games dataset)

how do I find this image in the database?

Large-scale image search

Build the database:

- Extract features from the database images
- Learn a vocabulary using k-means (typical k: 100,000)
- Compute *weights* for each word
- Create an inverted file mapping words images

Weighting the words

- Just as with text, some visual words are more discriminative than others

the, and, or vs. ***cow, AT&T, Cher***

- the bigger fraction of the documents a word appears in, the less useful it is for matching
    - e.g., a word that appears in *all* documents is not helping us

TF-IDF weighting

- Instead of computing a regular histogram distance, we'll weight each word by it's *inverse document frequency*

- inverse document frequency (IDF) of word *j* =

$$\log \frac{\text{number of documents}}{\text{number of documents in which } j \text{ appears}}$$

TF-IDF weighting

- To compute the value of bin *j* in image *I*:

term frequency of *j* in *I* **x** *inverse document frequency* of *j*

Inverted file

- Each image has ~1,000 features
- We have ~100,000 visual words

  each histogram is extremely sparse (mostly zeros)

- Inverted file
  - mapping from words to documents

```
"a":      {2}
"banana": {2}
"is":     {0, 1, 2}
"it":     {0, 1, 2}
"what":   {0, 1}
```

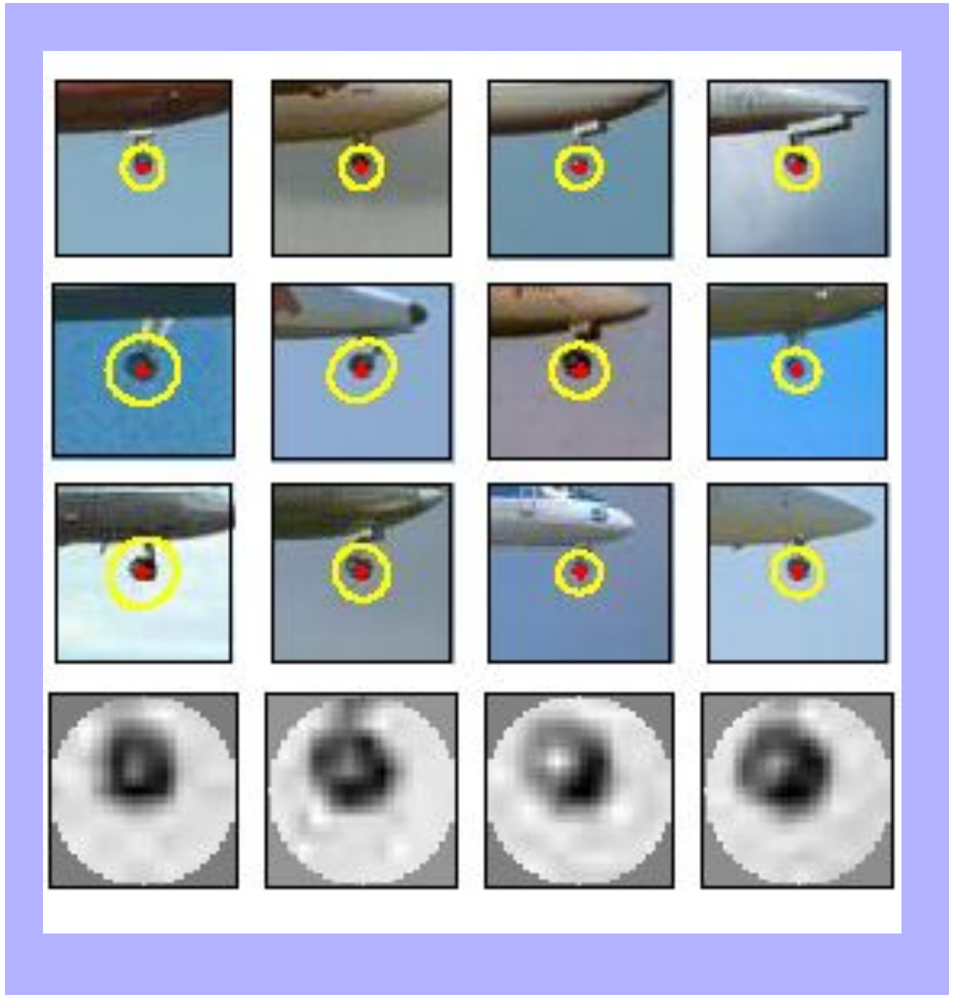
Inverted file

- Can quickly use the inverted file to compute similarity between a new image and all the images in the database
  - Only consider database images whose bins overlap the query image

Matching Statistics

Dataset	Size	Matches possible	Matches Tried	Matches Found	Time
Dubrovnik	58K	1.6 Billion	2.6M	0.5M	5 hrs
Rome	150K	11.2 Billion	8.8M	2.7M	13 hrs
Venice	250K	31.2 Billion	35.5M	6.2M	27 hrs

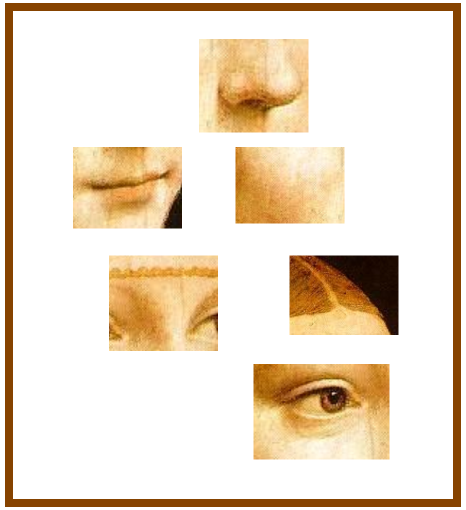
Image patch examples of visual words

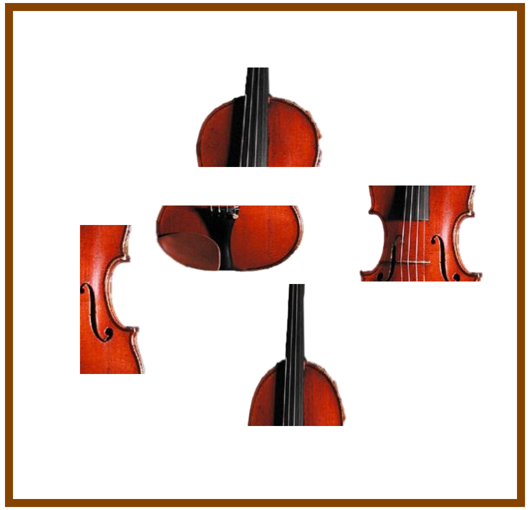


Sivic et al. 2005

Bag of features: outline

1. Extract features





Bag of features: outline

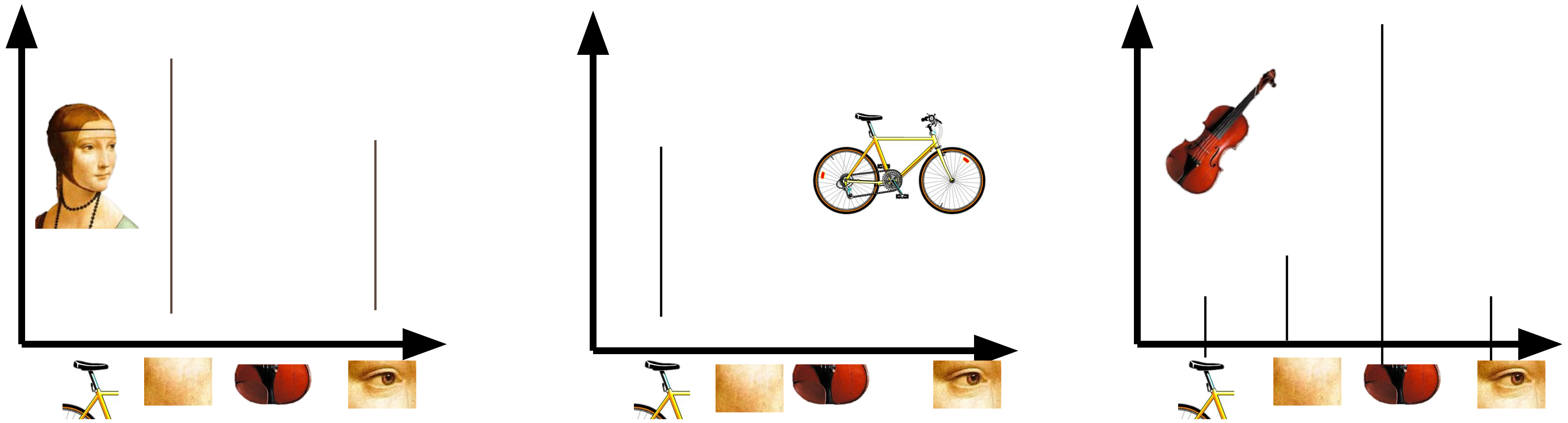
1. Extract features
2. Learn "visual vocabulary"

Bag of features: outline

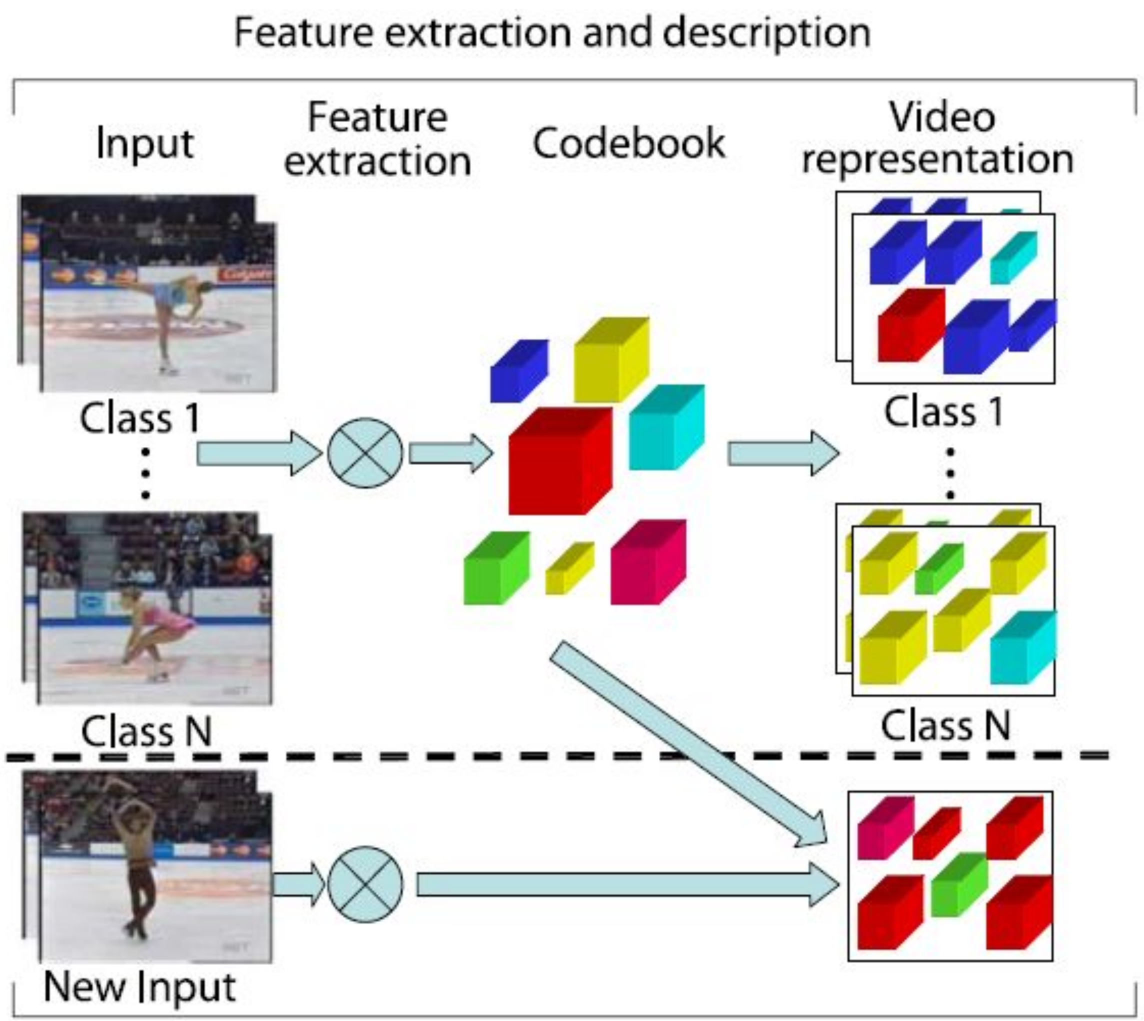
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

Bag of features: outline

1. Extract features
2. Learn "visual vocabulary"
3. Quantize features using visual vocabulary
4. Represent images by frequencies of "visual words"



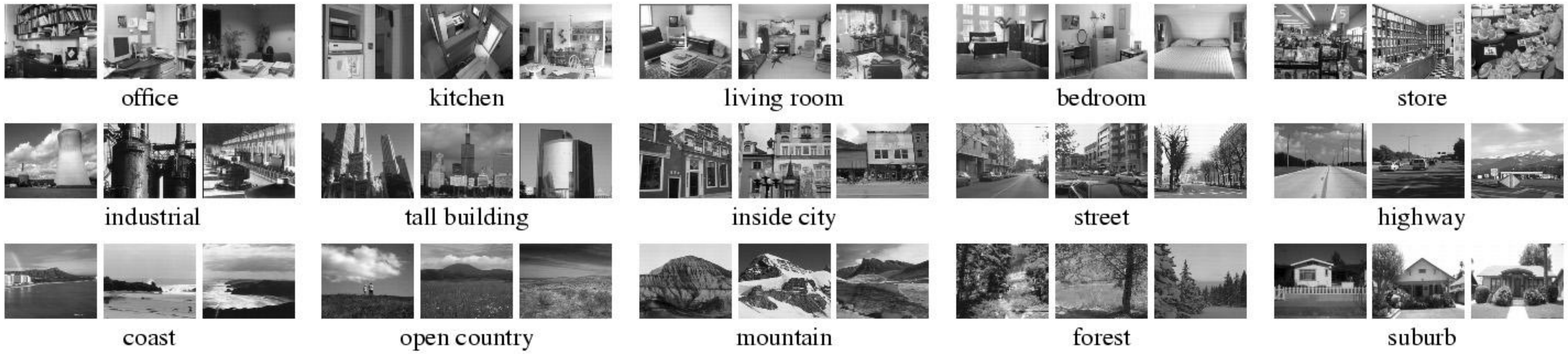
Bags of features for action recognition



Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei, **Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words**, IJCV 2008.

Scene category dataset

Multi-class classification results
(100 training images per class)



	(vocabulary size: 16)		(vocabulary size: 200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0 (1 × 1)	45.3 ±0.5		72.2 ±0.6	
1 (2 × 2)	53.6 ±0.3	56.2 ±0.6	77.9 ±0.6	79.0 ±0.5
2 (4 × 4)	61.7 ±0.6	64.7 ±0.7	79.4 ±0.3	**81.1** ±0.3
3 (8 × 8)	63.3 ±0.8	**66.8** ±0.6	77.2 ±0.4	80.7 ±0.3

Lazebnik, Schmid & Ponce (CVPR 2006)

Slide credit: Svetlana Lazebnik