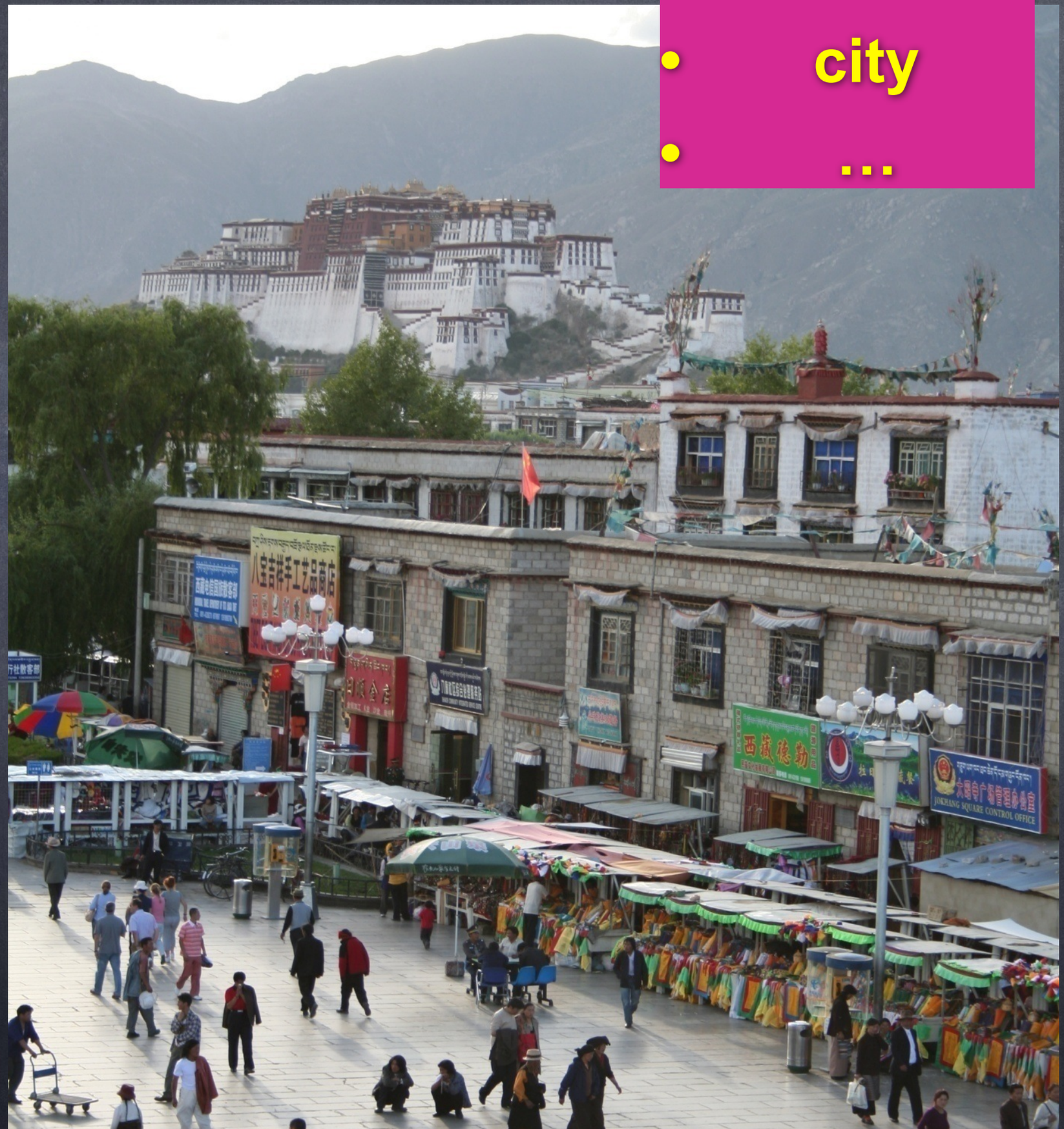
Computer Vision

So what does object recognition involve?

Scene categorization

- outdoor
- city
- …



Object detection: are there people?

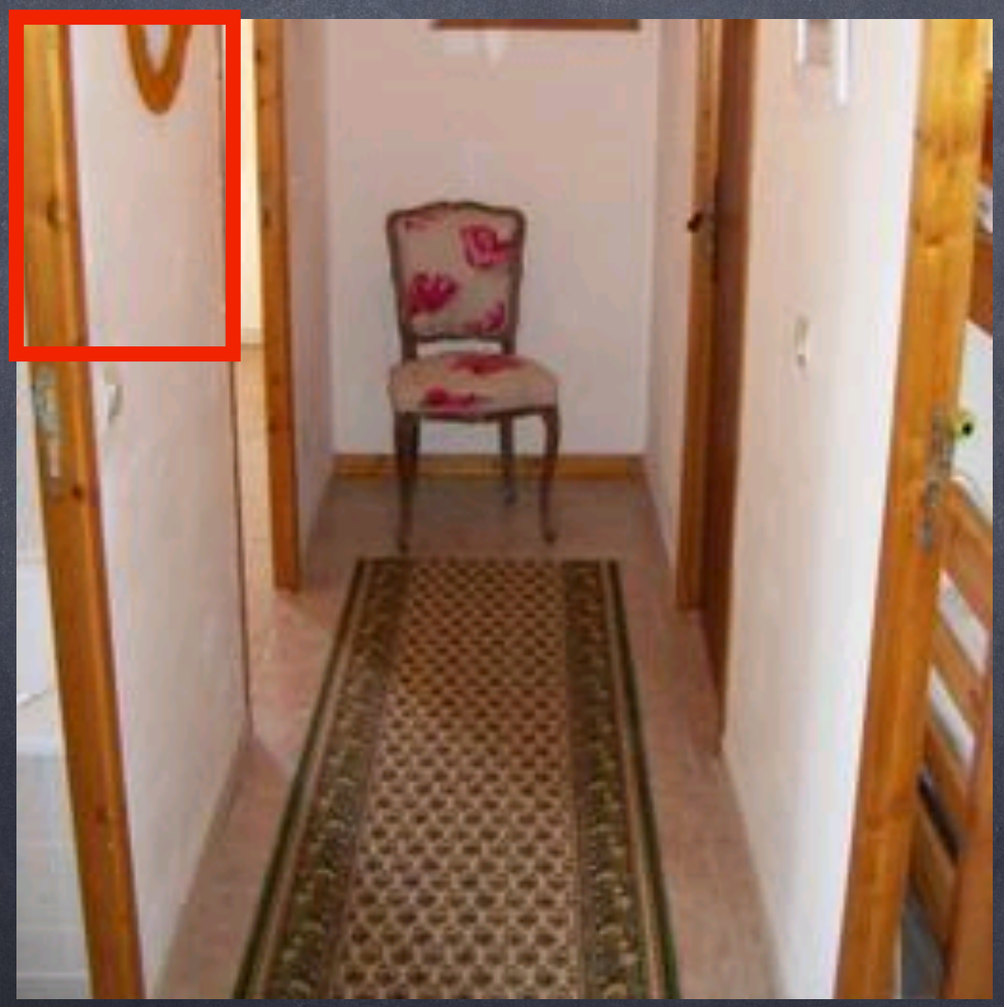
Identification: what is this structure?

Image parsing

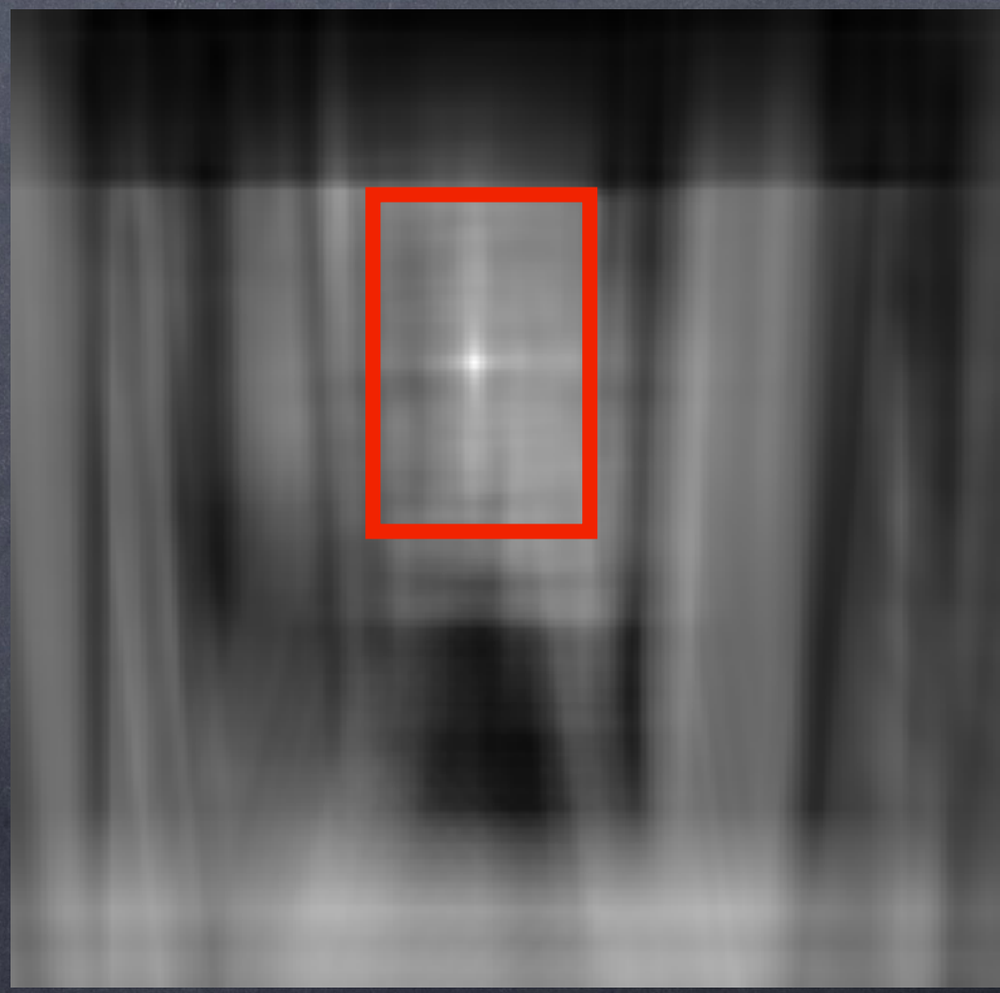
Object recognition
Is it really so hard?

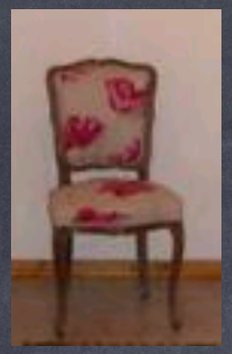
Find the chair in this image

This is a chair



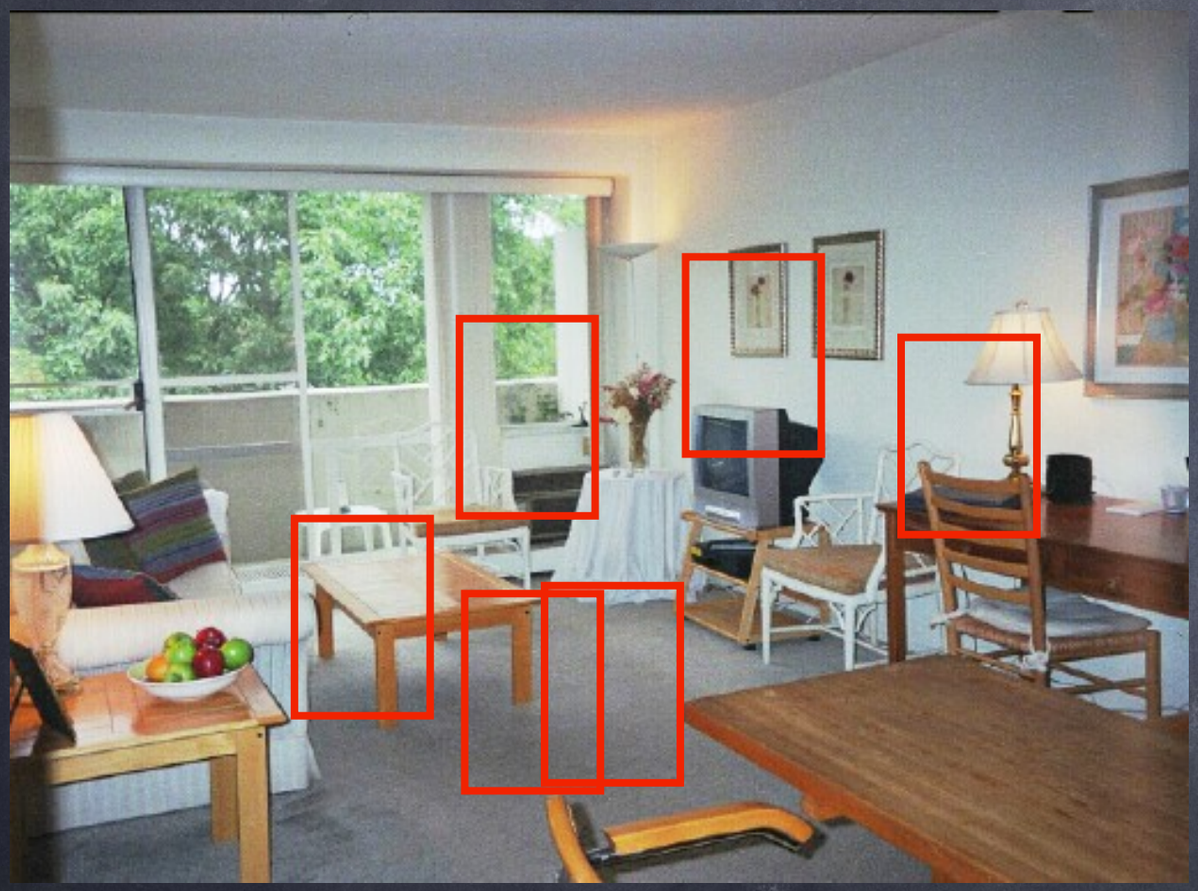
Output of normalized correlation





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

Why Object Recognition is hard?

The "Margaret Thatcher Illusion", by Peter Thompson

Why Object Recognition is hard?

The "Margaret Thatcher Illusion", by Peter Thompson

Why Object Recognition is hard?

All is Vanity, by C. Allan Gilbert, 1873-1929

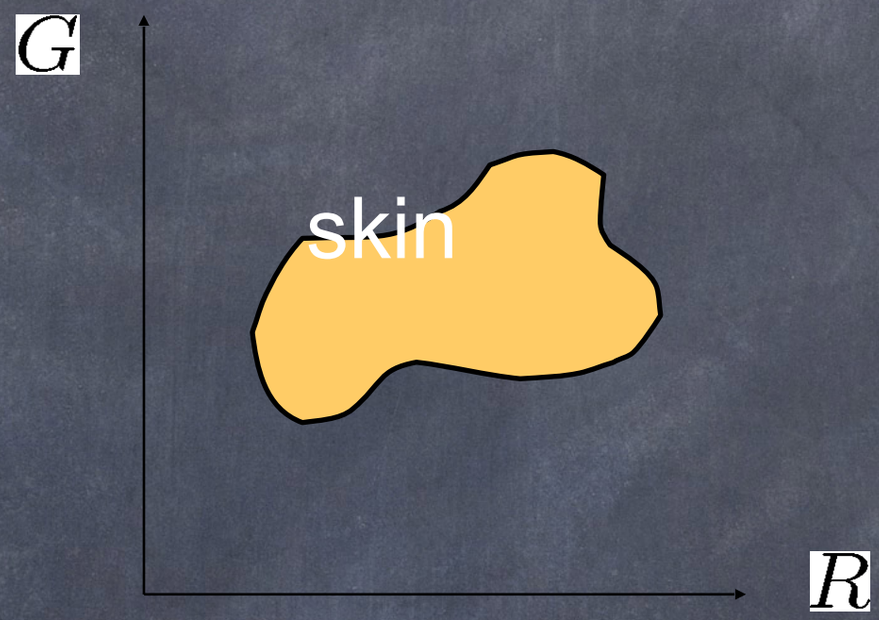
What algorithms you have studied in last lecture?

Skin color based segmentation

Skin color segmentation using Bayes

What is Bayes Rule?

Skin detection



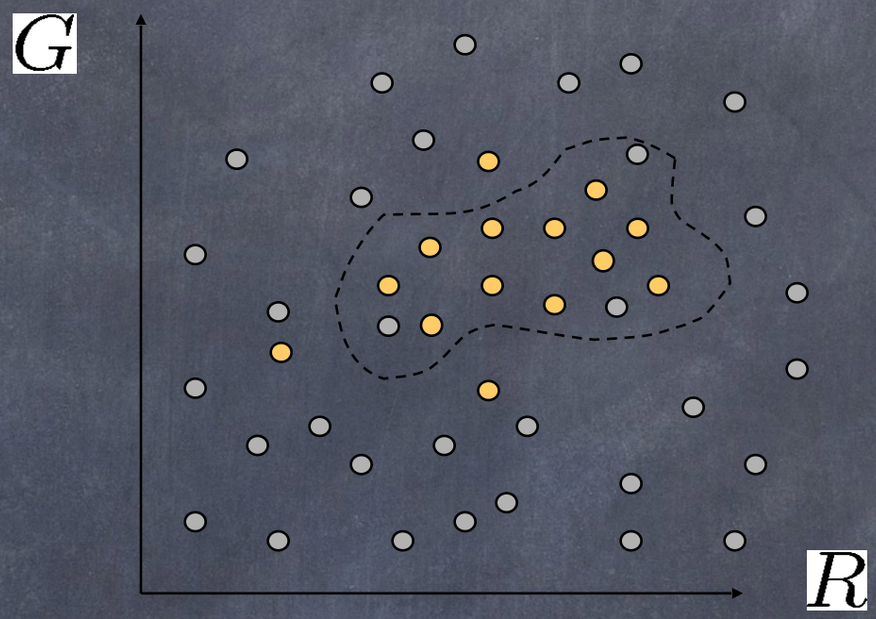
Skin pixels have a distinctive range of colors

- Corresponds to region(s) in RGB color space

Skin classifier

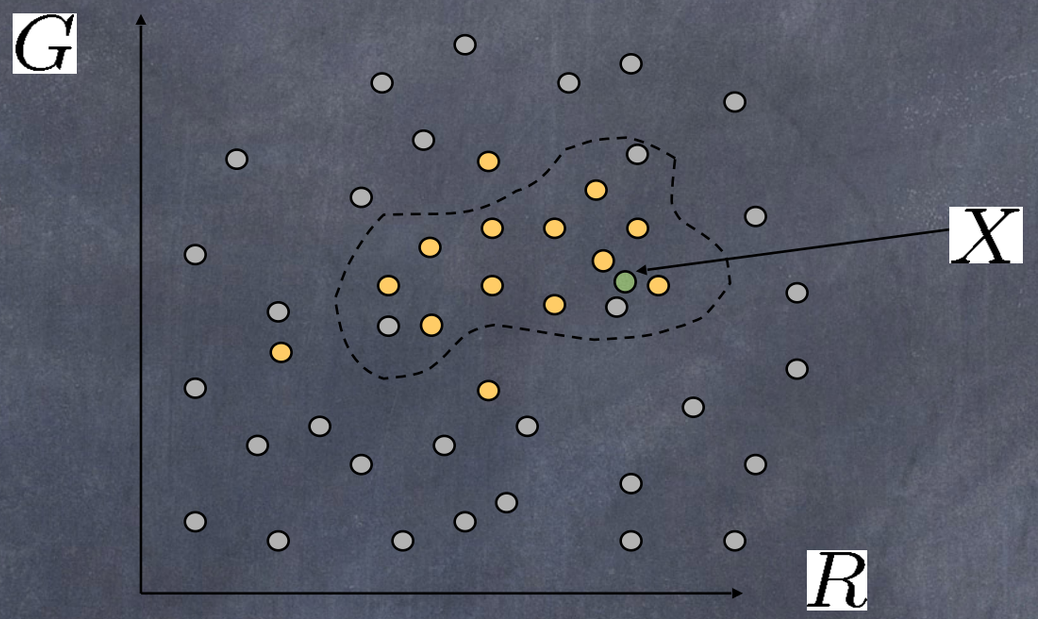
- A pixel X = (R,G,B) is skin if it is in the skin (color) region
- How to find this region?

Skin detection



Learn the skin region from examples
Manually label skin/non pixels in one or more "training images"
Plot the training data in RGB space
skin pixels shown in orange, non-skin pixels shown in gray
some skin pixels may be outside the region, non-skin pixels inside.

Skin classifier



Given X = (R,G,B): how to determine if it is skin or not?

Nearest neighbor

find labeled pixel closest to X

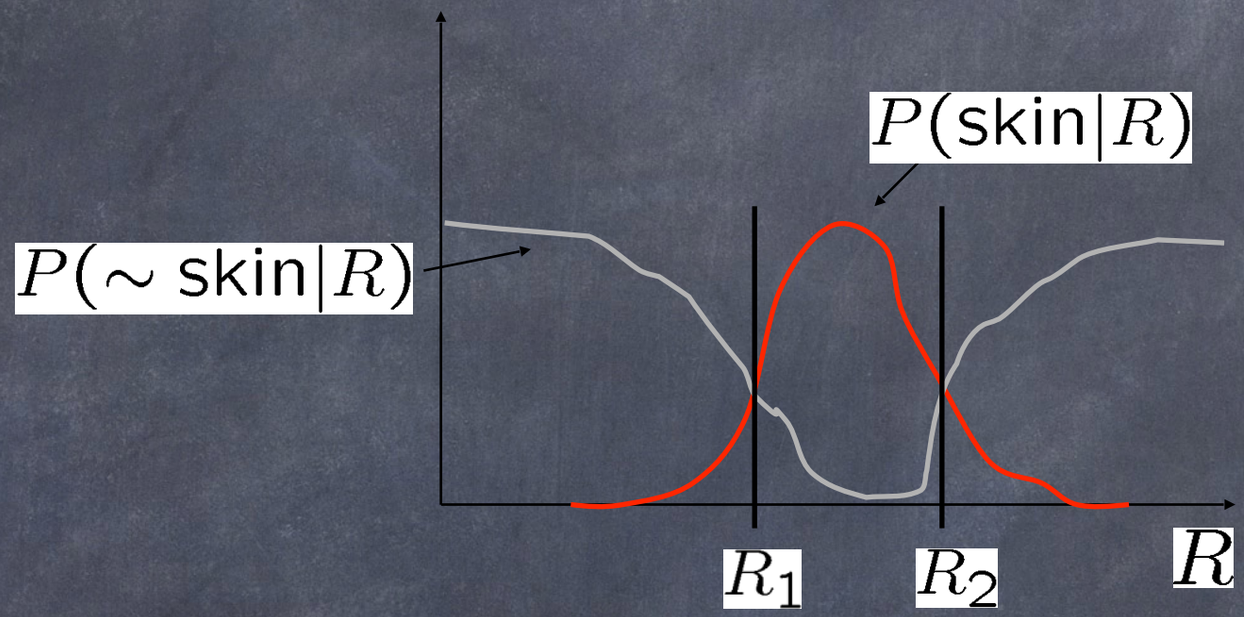
Find plane/curve that separates the two classes

popular approach: Support Vector Machines (SVM)

Data modeling

fit a probability density/distribution model to each class

Probabilistic skin classification



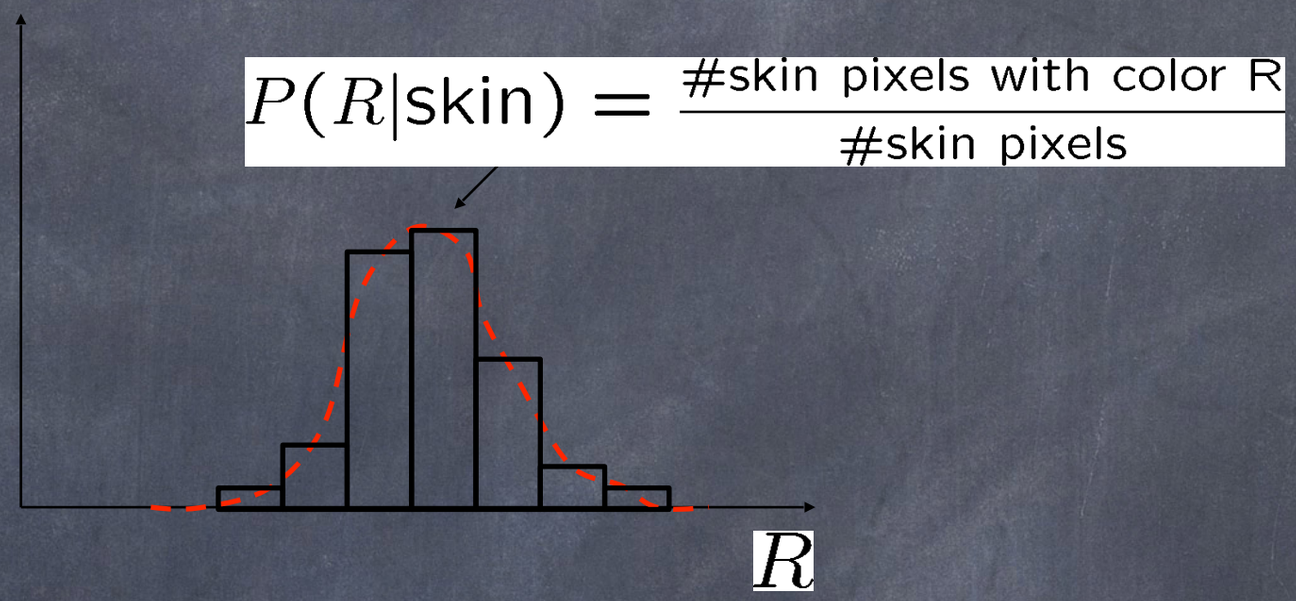
- Model PDF / uncertainty
  - Each pixel has a probability of being skin or not skin

$$P(\sim \text{skin}|R) = 1 - P(\text{skin}|R)$$

- Skin classifier
  - Given X = (R,G,B): how to determine if it is skin or not?
  - Choose interpretation of highest probability

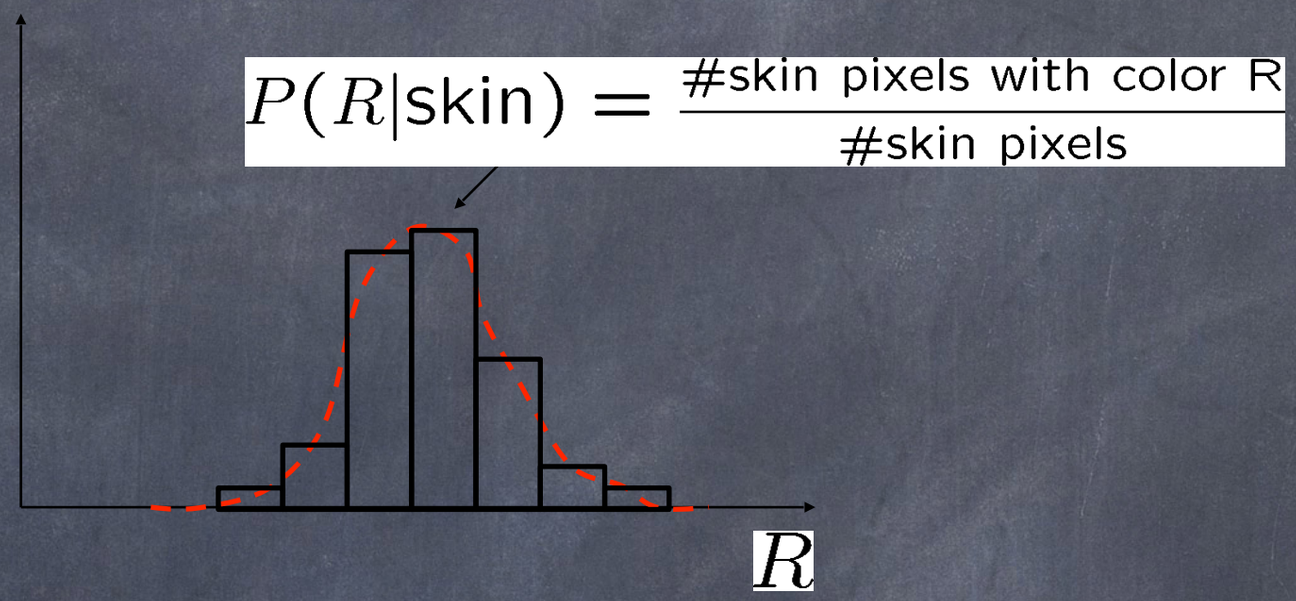
- Where do we get $P(\text{skin}|R)$ and $P(\sim \text{skin}|R)$?

Learning conditional PDF's



- We can calculate P(R | skin) from a set of training images
- It is simply a histogram over the pixels in the training images
  - each bin R_i contains the proportion of skin pixels with color R_i

Learning conditional PDF's



- We can calculate P(R | skin) from a set of training images
- But this isn't quite what we want
  - Why not? How to determine if a pixel is skin?
  - We want P(skin | R) not P(R | skin)
  - How can we get it?

Bayes rule

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

what we measure (**likelihood**)

domain knowledge (**prior**)

$$P(\text{skin}|R) = \frac{P(R|\text{skin})\ P(\text{skin})}{P(R)}$$

what we want (**posterior**)

normalization term

$$P(R) = P(R|\text{skin})P(\text{skin}) + P(R|\sim\text{skin})P(\sim\text{skin})$$

Skin detection results

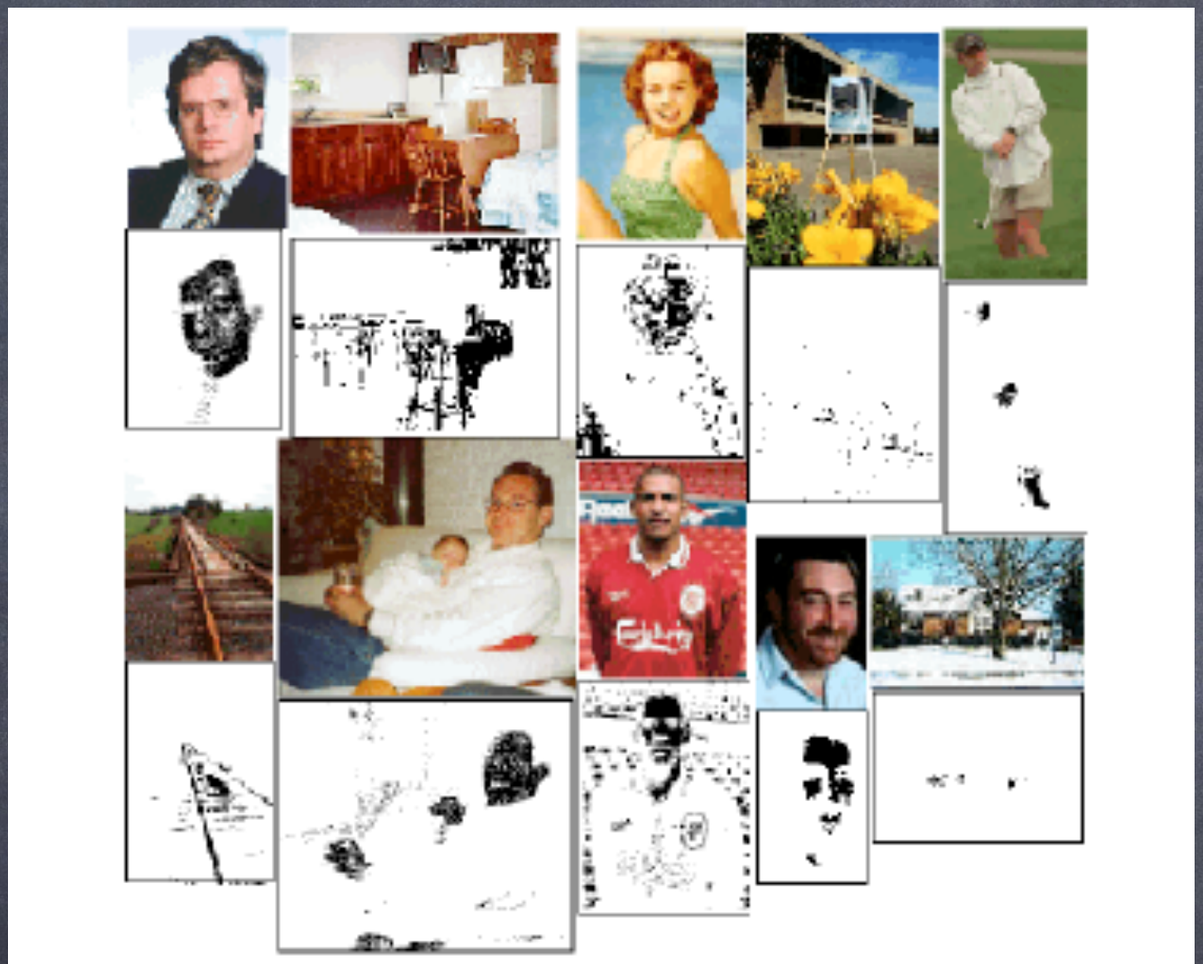


Figure 25.3. The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this process is relatively effective, and could certainly be used to focus attention on, say, faces and hands. *Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 © 1999, IEEE*

Problems with common methods

- Not effective in presence of noise and sampling artifacts (e.g. medical images).

Active Contour Models

First introduced in 1987 by Kass et al,and gained popularity since then.

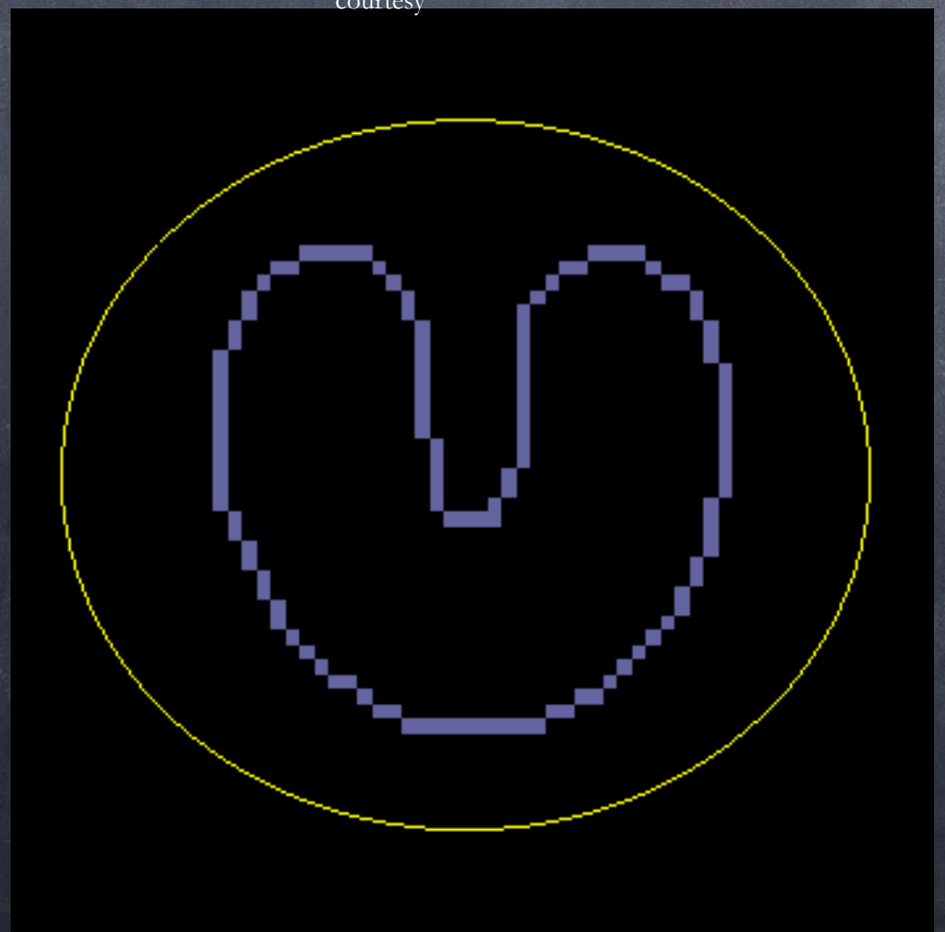
Represents an object boundary or some other salient image feature as a parametric curve.

An energy functional E is associated with the curve.

Framework for snakes

A higher level process or a user initializes any curve *close to the object boundary.*
The snake then starts *deforming* and moving towards the desired object boundary.
In the end it completely “shrink-wraps” around the object.

courtesy



(Diagram courtesy “Snakes, shapes, gradient vector flow”, Xu, Prince)

Modeling

The contour is defined in the (x, y) plane of an image as a parametric curve

$$v(s)=(x(s), y(s))$$

Contour is said to possess an energy (E_{snake}) which is defined as the sum of the three energy terms.

$$E_{snake} = E_{internal} + E_{external} + E_{constraint}$$

Internal Energy (E_{int})

Depends on the intrinsic properties of the curve.
Sum of elastic energy and bending energy.

Elastic Energy ($E_{elastic}$):
The curve is treated as an elastic rubber band possessing elastic potential energy.
It discourages stretching by introducing tension.

$$E_{elastic} = \frac{1}{2}\int_{s} \alpha(s) \mid v_s \mid^2 \, ds \qquad v_s = \frac{dv(s)}{ds}$$

Weight α(s) allows us to control elastic energy along different parts of the contour. Considered to be constant α for many applications.
Responsible for shrinking of the contour.

Bending Energy ($E_{bending}$):

The snake is also considered to behave like a thin metal strip giving rise to bending energy.

It is defined as sum of squared curvature of the contour.

$$E_{bending} = \frac{1}{2}\int \beta(s) \mid v_{ss} \mid^2 ds$$

β(s) plays a similar role to α(s).
Bending energy is minimum for a circle.
Total internal energy of the snake can be defined as

$$E_{\text{int}} = E_{elastic} + E_{bending} = \int_s \frac{1}{2}(\alpha \mid v_s \mid^2 + \beta \mid v_{ss} \mid^2) ds$$

External energy of the contour (E_{ext})

It is derived from the image.
Define a function $E_{image}(x,y)$ so that it takes on its smaller values at the features of interest, such as boundaries.

$$E_{ext} = \int_s E_{image}(v(s))ds$$

Key rests on defining $E_{image}(x,y)$. Some examples

$$E_{image}(x,y) = -|\nabla \mathrm{I}(x,y)|^2$$

$$E_{image}(x,y) = -|\nabla(G_\sigma(x,y) * I(x,y))|^2$$

Energy and force equations

The problem at hand is to find a contour v(s) that minimize the energy functional

$$E_{snake} = \int_s \frac{1}{2}(\alpha(s)\,|\,v_s\,|^2 + \beta(s)\,|\,v_{ss}\,|^2) + E_{image}(v(s))ds$$

Using variational calculus and by applying Euler-Lagrange differential equation we get following equation

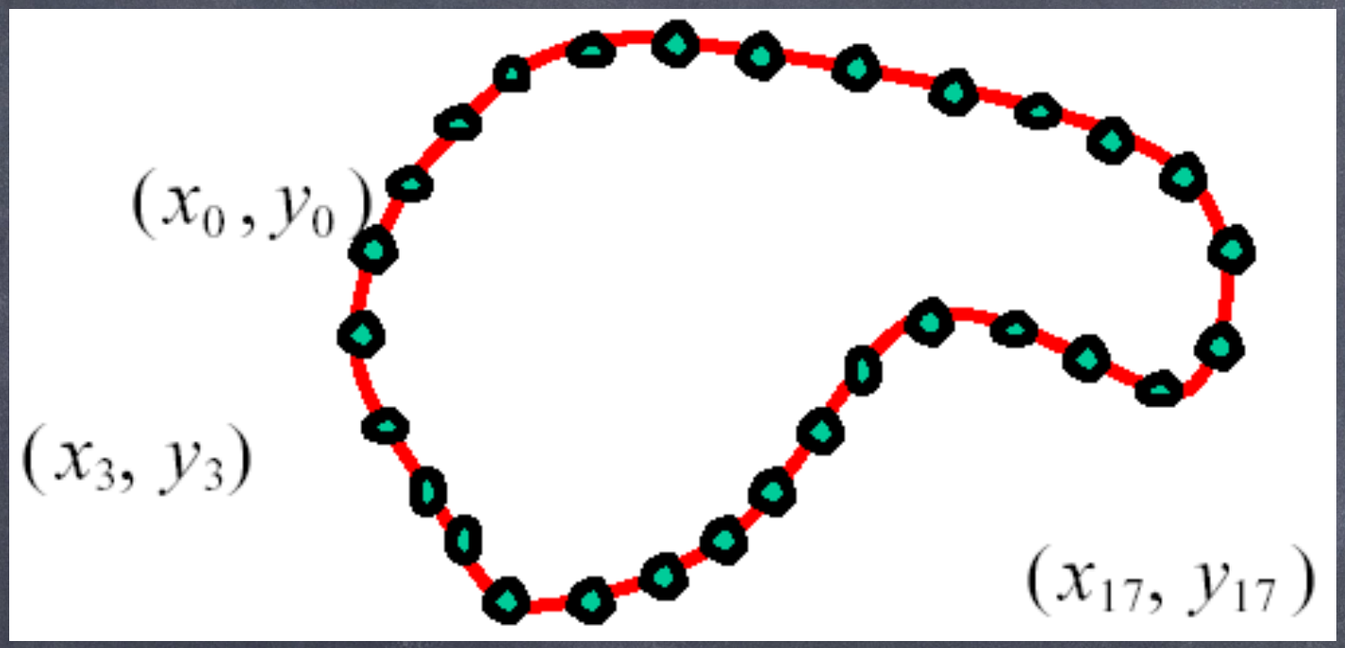
$$\alpha v_{ss} - \beta v_{ssss} - \nabla E_{image} = 0$$

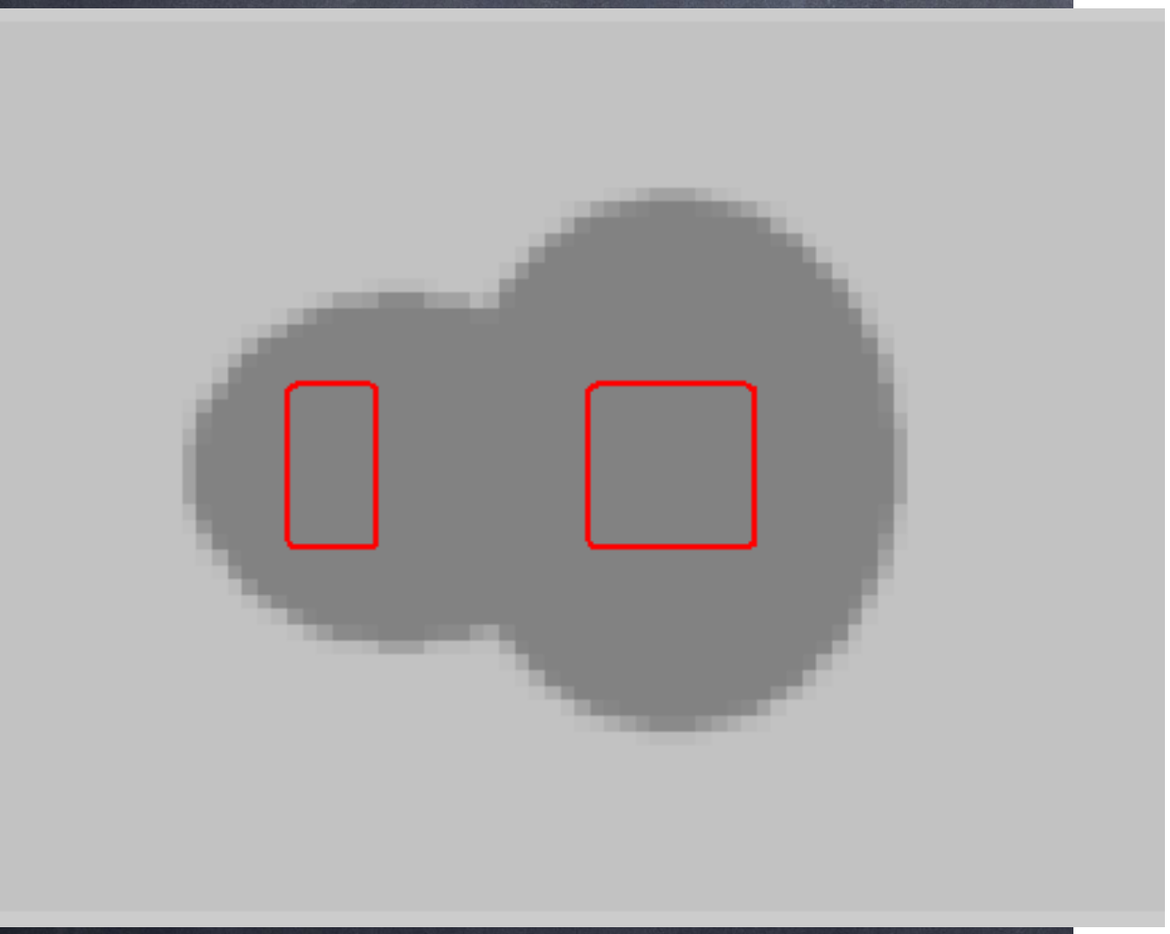
Equation can be interpreted as a force balance equation.

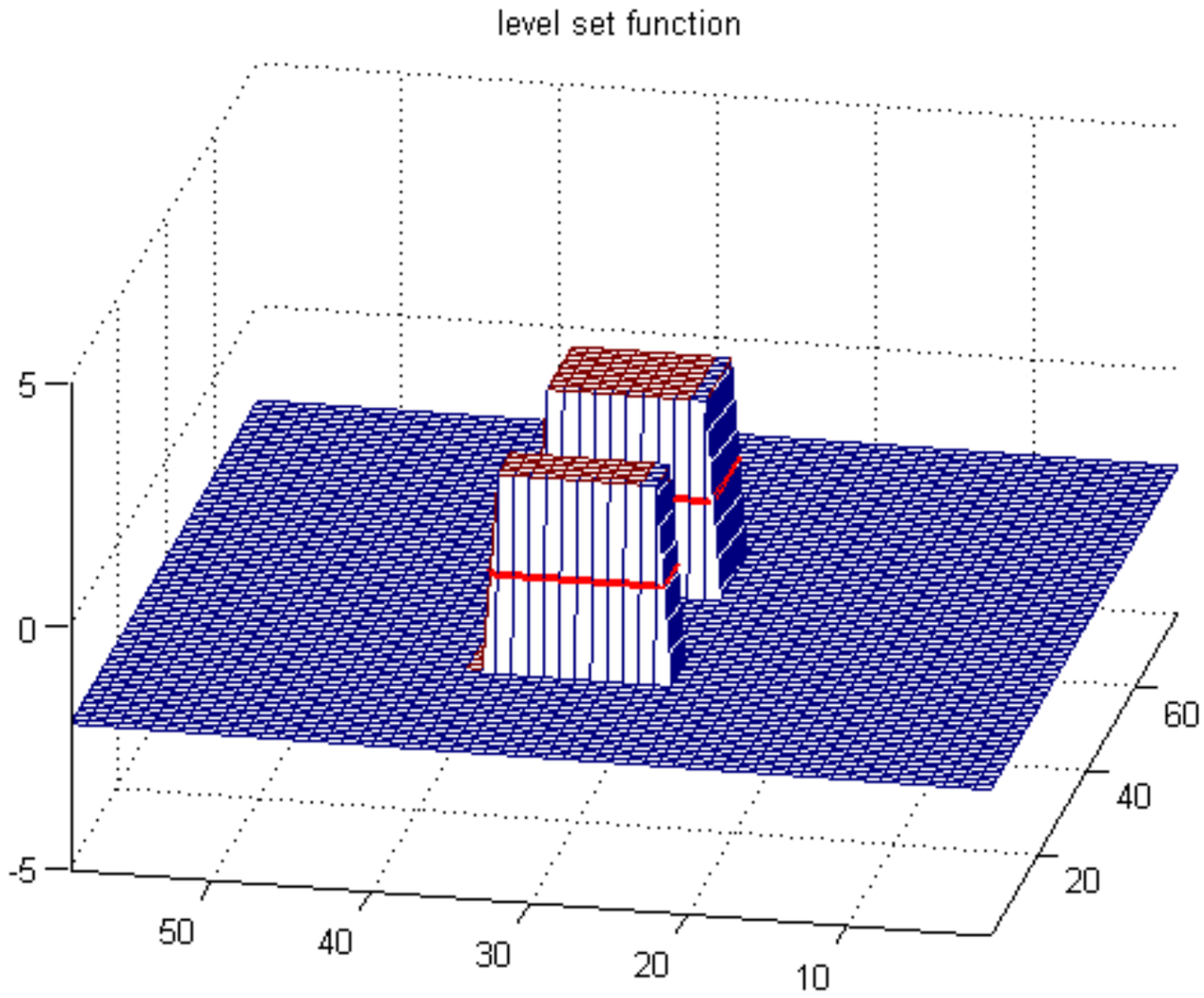
Each term corresponds to a force produced by the respective energy terms. The contour deforms under the action of these forces.

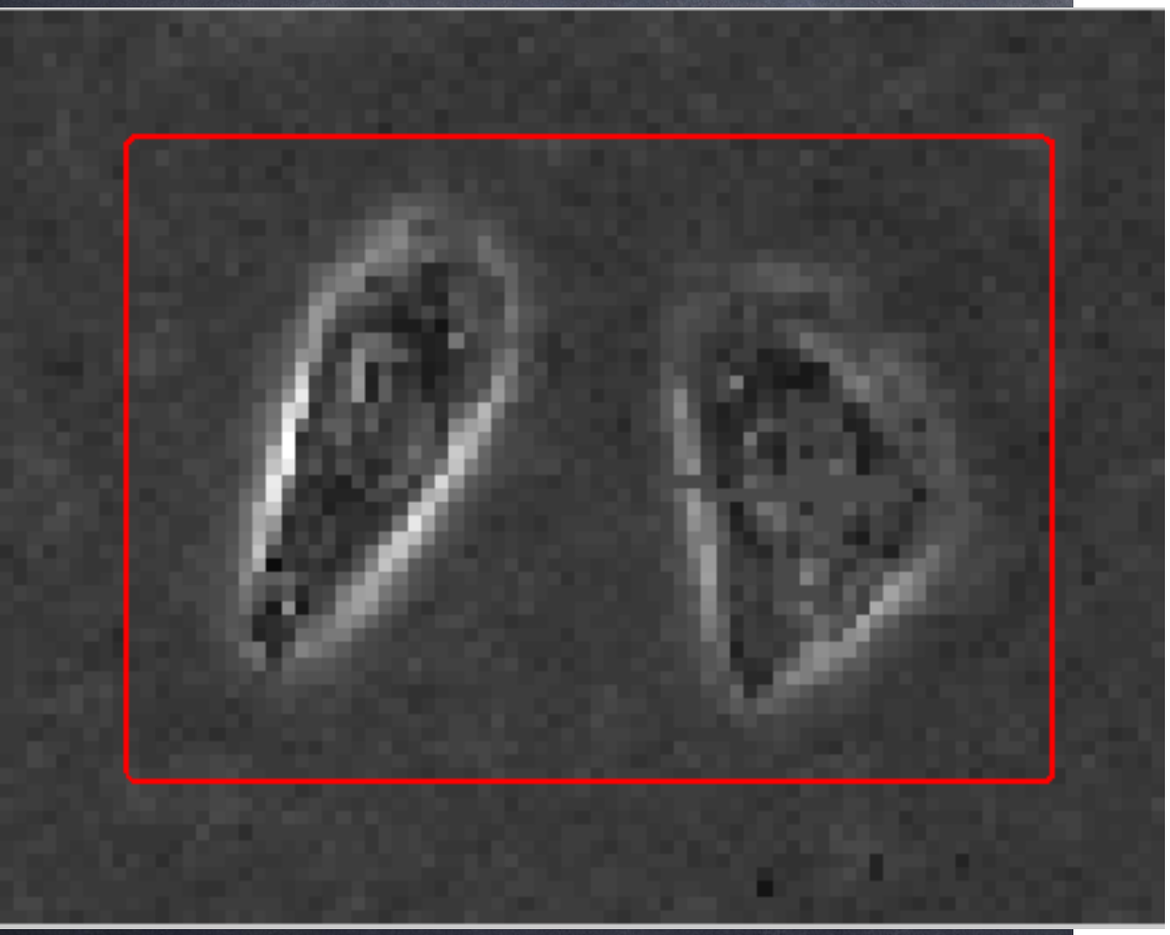
Implementation

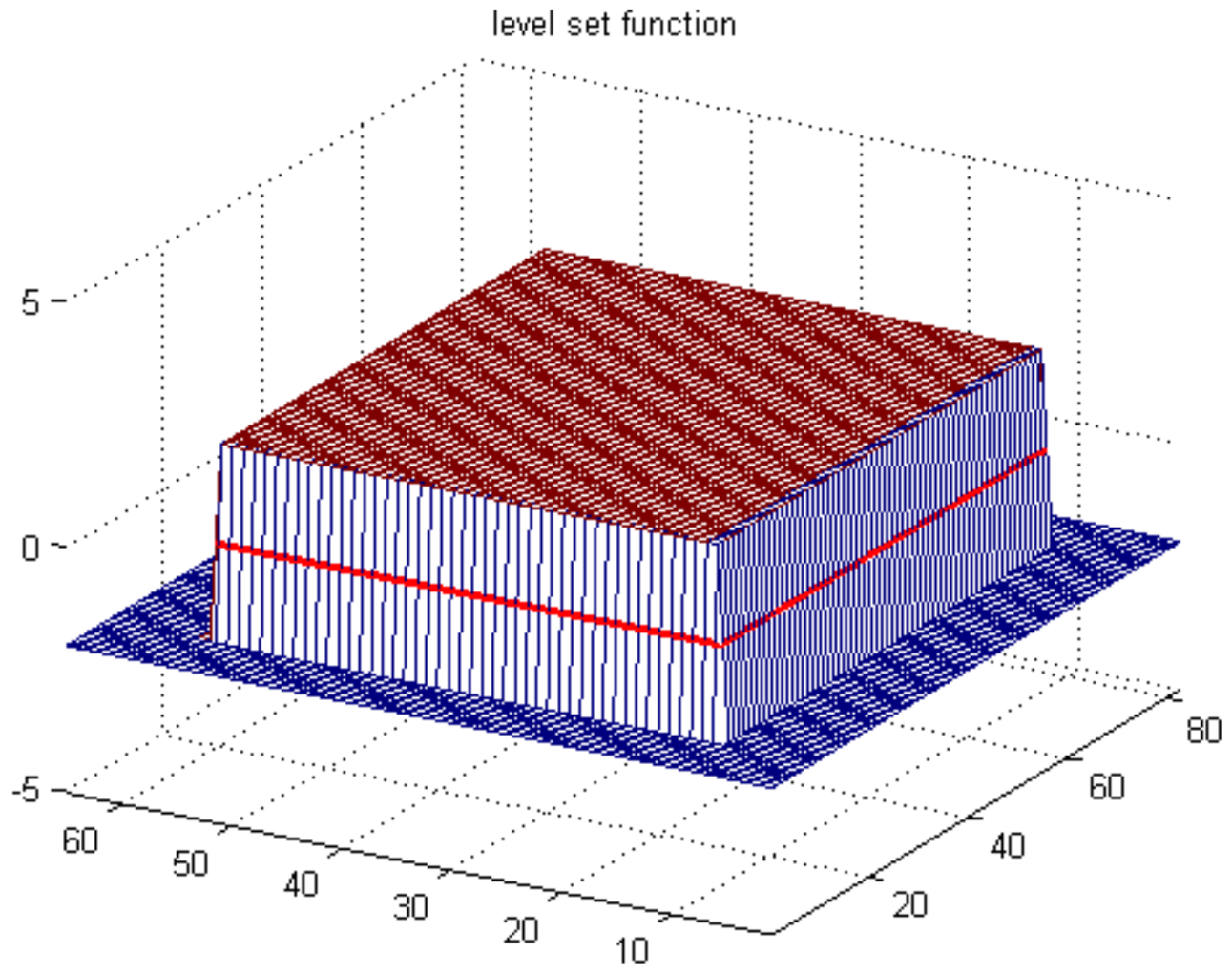
- Represent the curve with a set of n points






level set function
5
0
-5
50
40
30
20
10
20
40
60




level set function
5
0
-5
60
50
40
30
20
10
20
40
60
80

Contours in OpenCV

```
int cvFindContours( CvArr* img, CvMemStorage* storage,
    CvSeq** firstContour, int headerSize=sizeof(CvContour),
    CvContourRetrievalMode mode=CV_RETR_LIST,
    CvChainApproxMethod
    method=CV_CHAIN_APPROX_SIMPLE );
```

Advancements

There are now better approaches

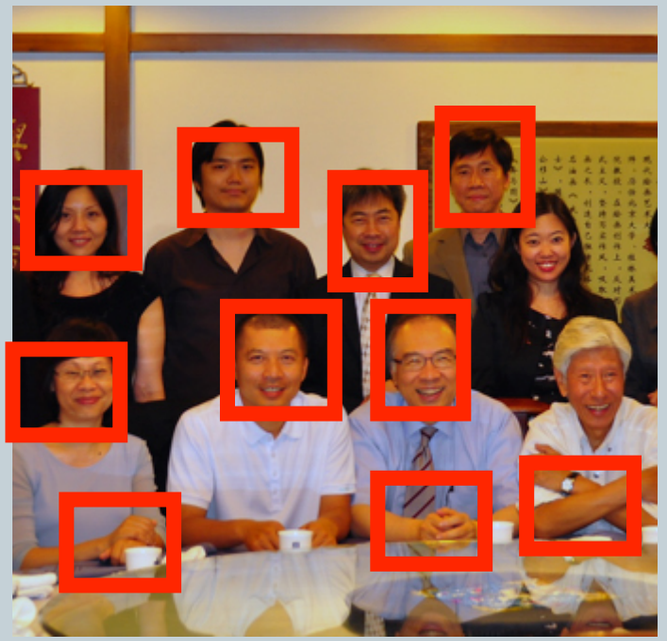
Level set

Contours without edges

Graph Cut

How to evaluate detection algorithm?

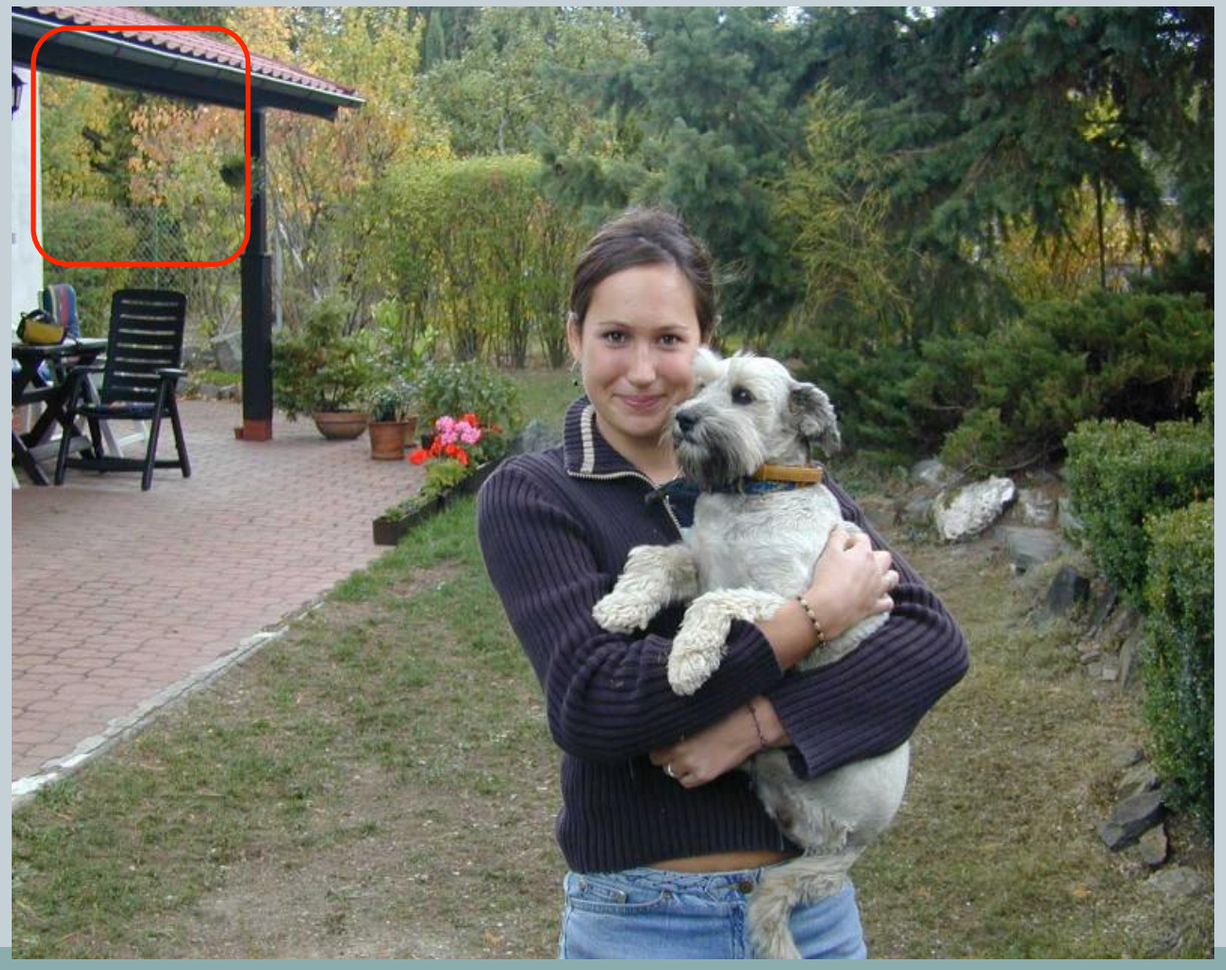
- Detected results are in red frames
- What are the detection rate and false positive rate here?



  - Answer
    - detection rate=?
    - false positive rate=?

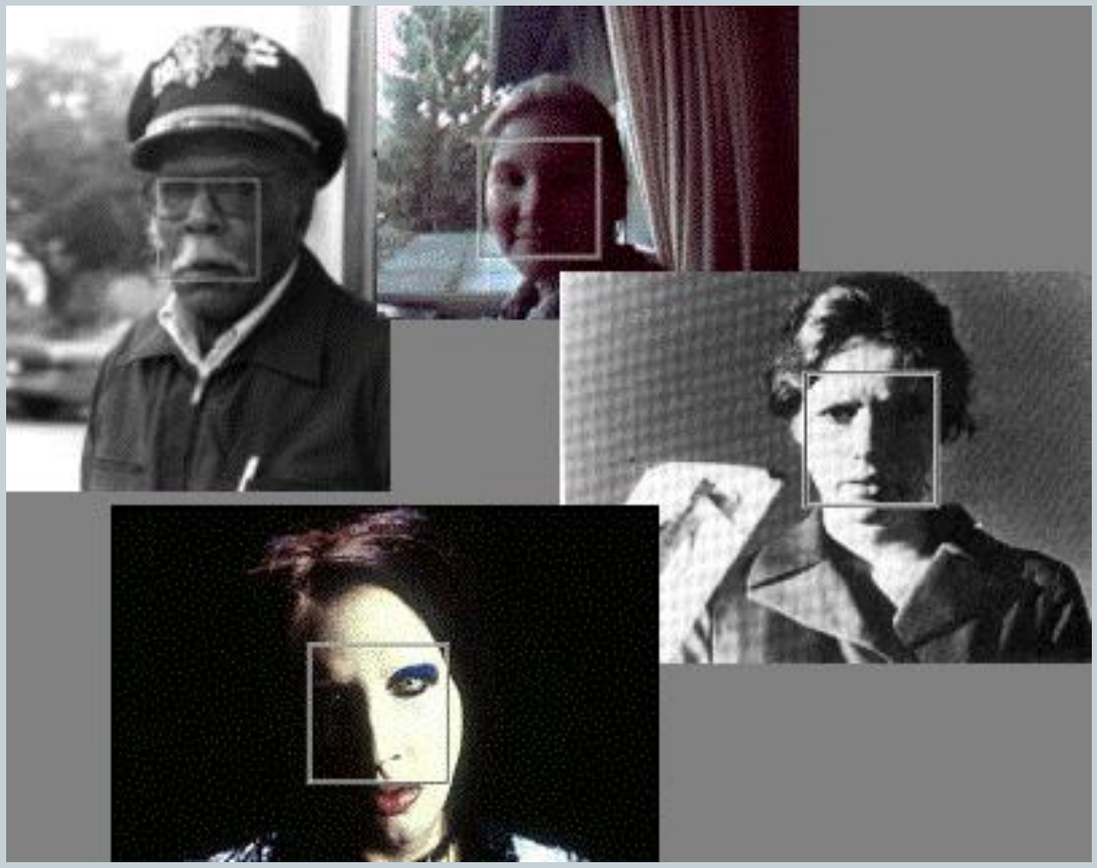
Face Detection

- Another approach: slide a window across image and evaluate a face model at every location



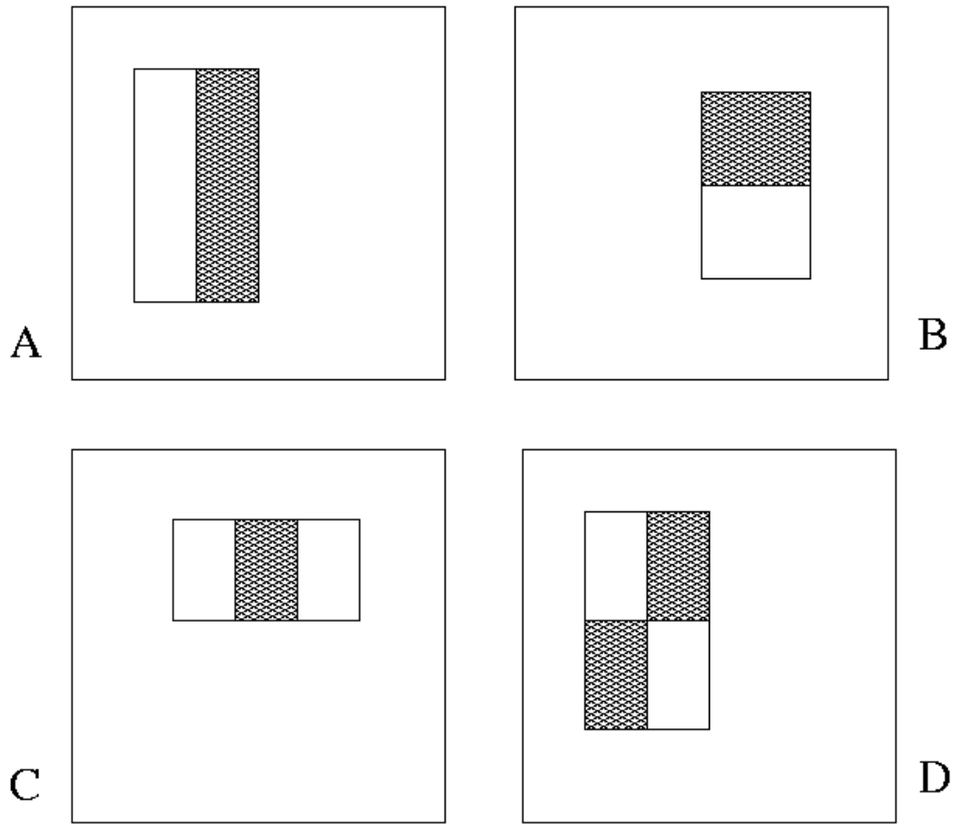
Adaboost Face Detector

- One of the most famous method for fast real time face detection



Haar/Adaboost Face Detector - Image Features

"Rectangle filters" for detecting features

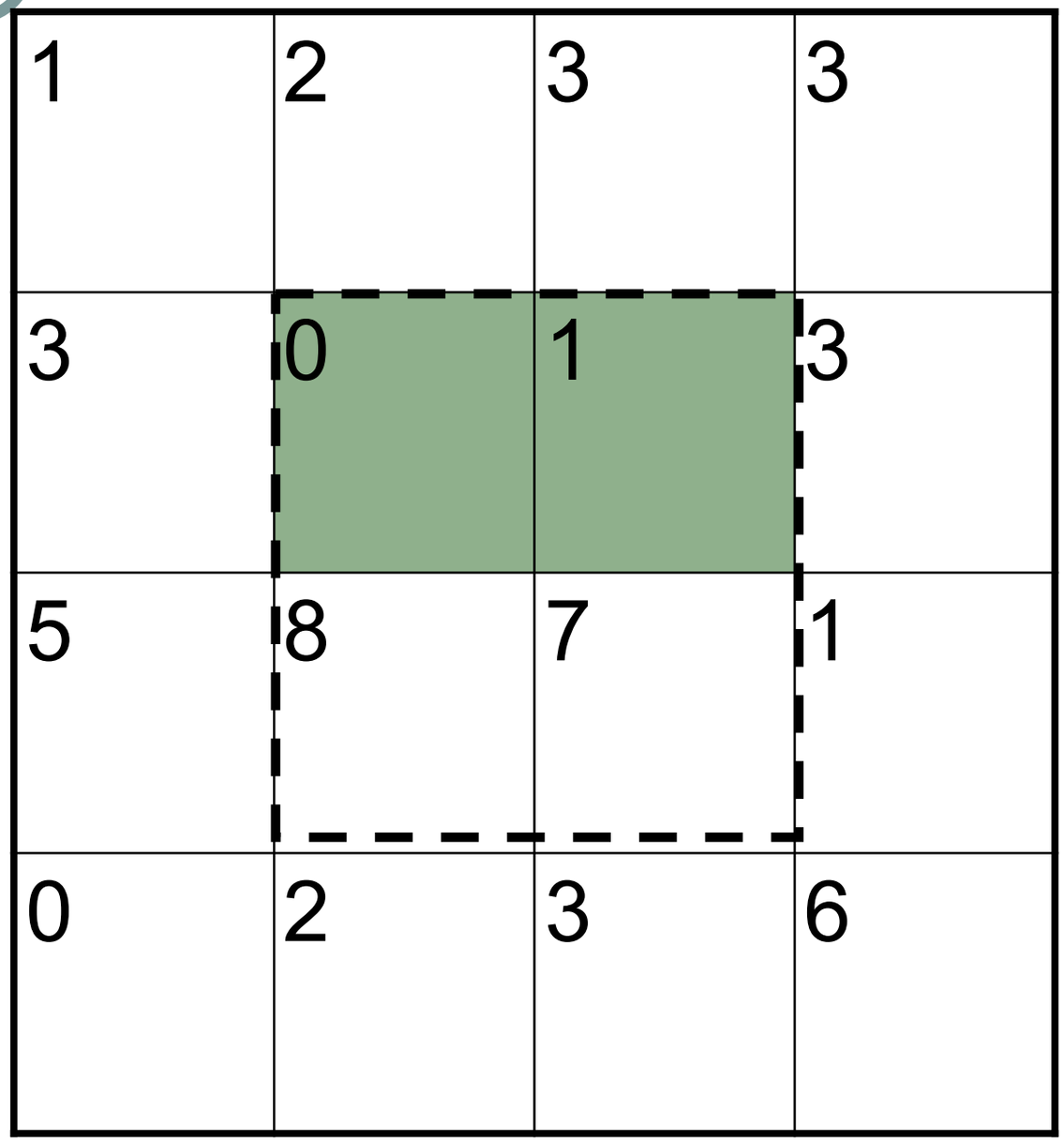


Rectangle_Feature_value f=

∑ (pixels values in white area) –
∑ (pixels values in shaded area)

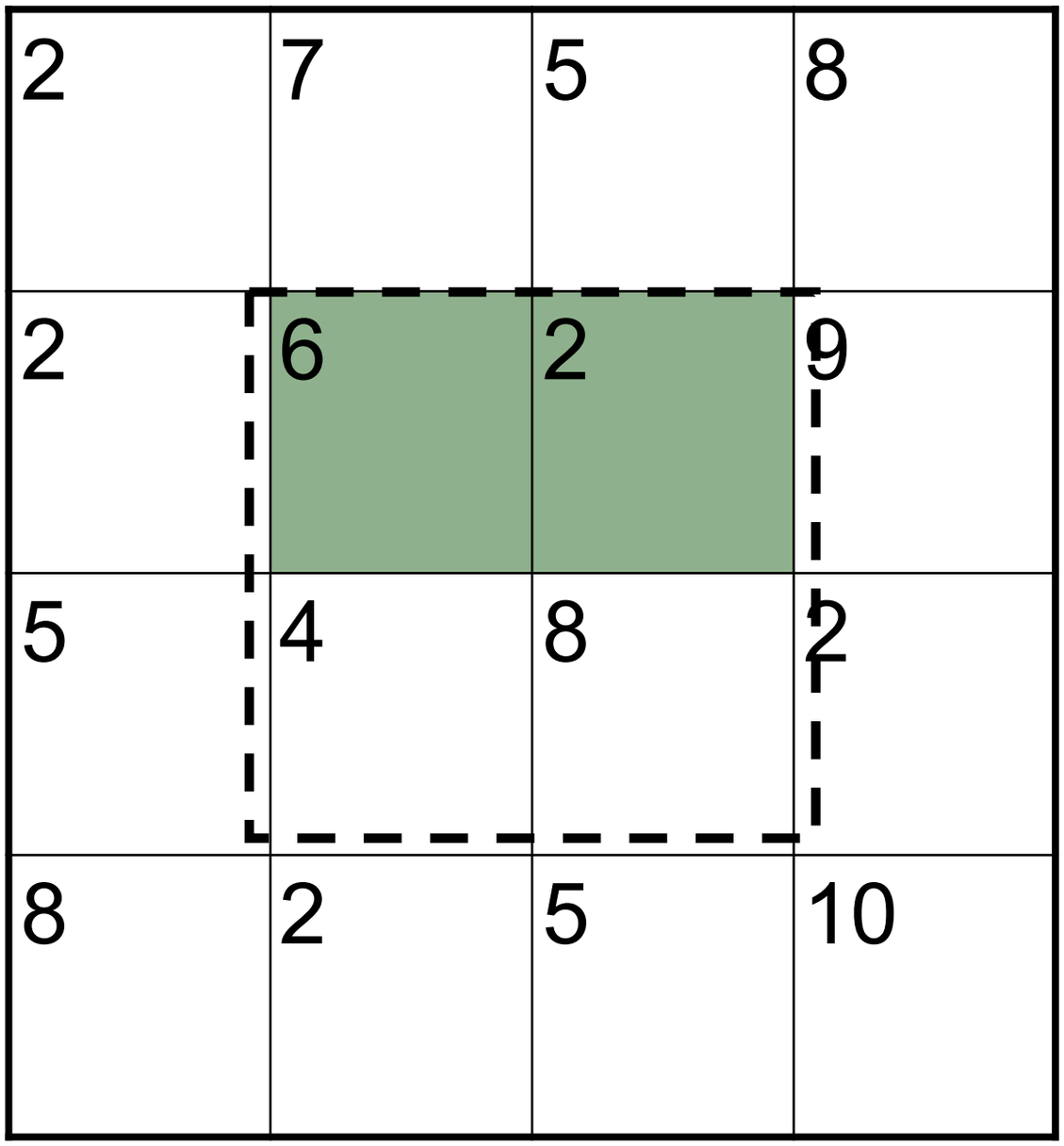
Example

- Find the *Rectangle_Feature_value* (f) of the box enclosed by the dotted line
- *Rectangle_Feature_value f=*
- *∑ (pixels values in white area) – ∑ (pixels values in shaded area)*
- *f=(8+7)-(0+1)*
- *=15-1= 14*



Example

- Find the *Rectangle_Feature_value* (f) of the box enclosed by the dotted line
- *Rectangle_Feature_value f=*
- *∑ (pixels values in white area) – ∑ (pixels values in shaded area)*
- *f=*



Example: A simple face detection method using one feature

- *Rectangle_Feature_value f*
- *f= ∑(pixels in white area) – ∑ (pixels in shaded area)*

- If (f) is large, then it is face ,i.e.
- if (f)>threshold, then
- face
- Else
- non-face

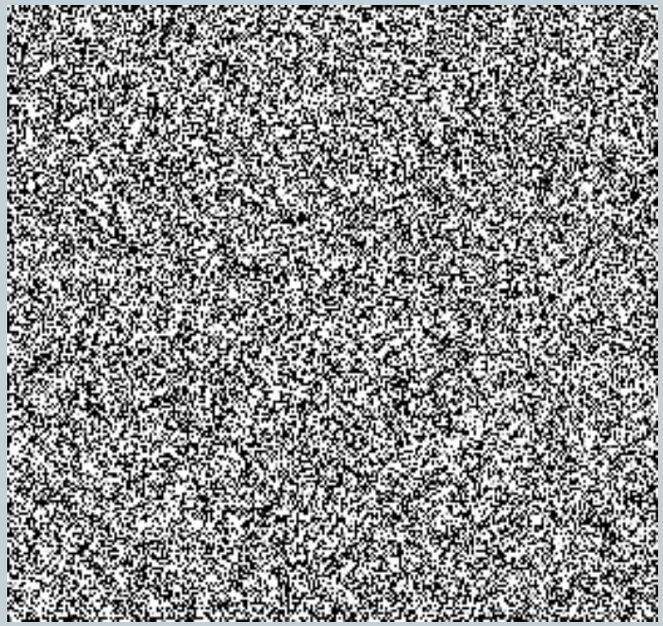
This is not a face.
Because f is small

Result

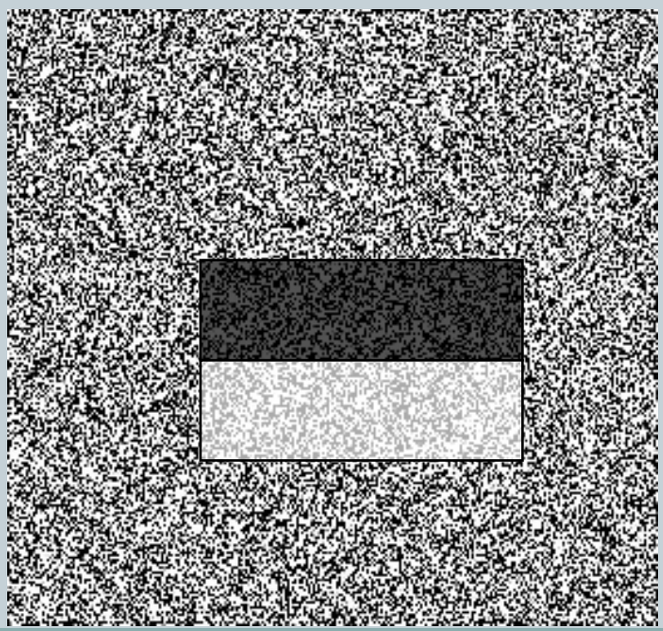
This is a face: The eye-area (shaded area)is dark, the nose-area(white area) is bright.
So f is large, hence it is face

Example

Source

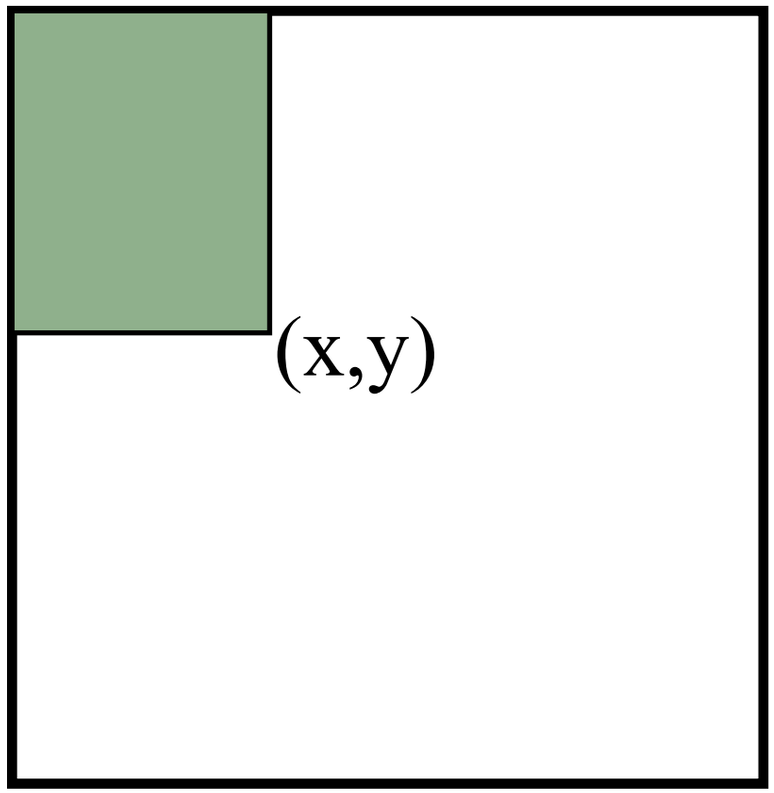


Result



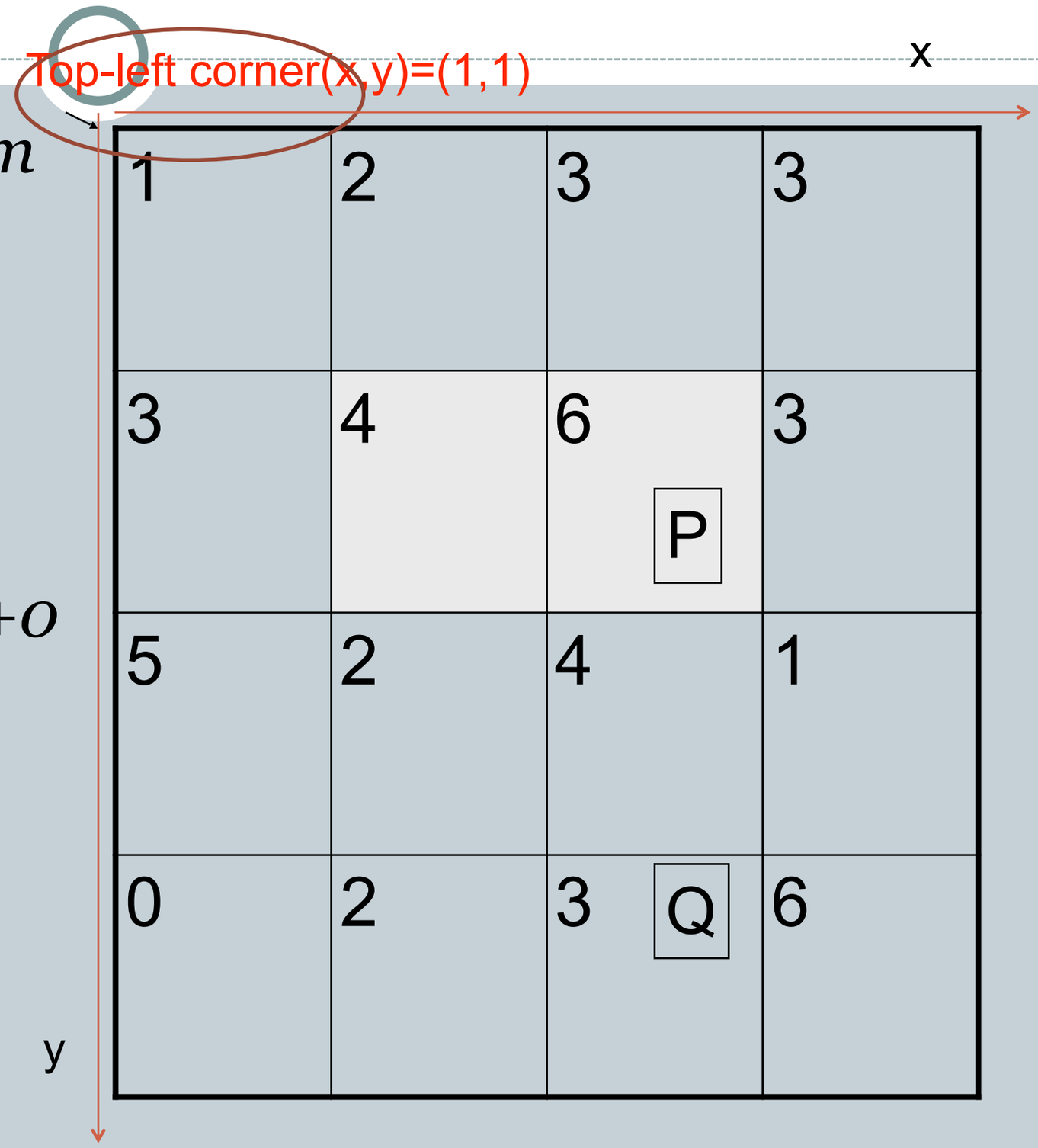
How to find features faster?
Integral images fast calculation method [Lazebnik09]

- The *integral image* = *sum of all* pixel values above and to the left of (*x*,*y*)
- Can be found very quickly



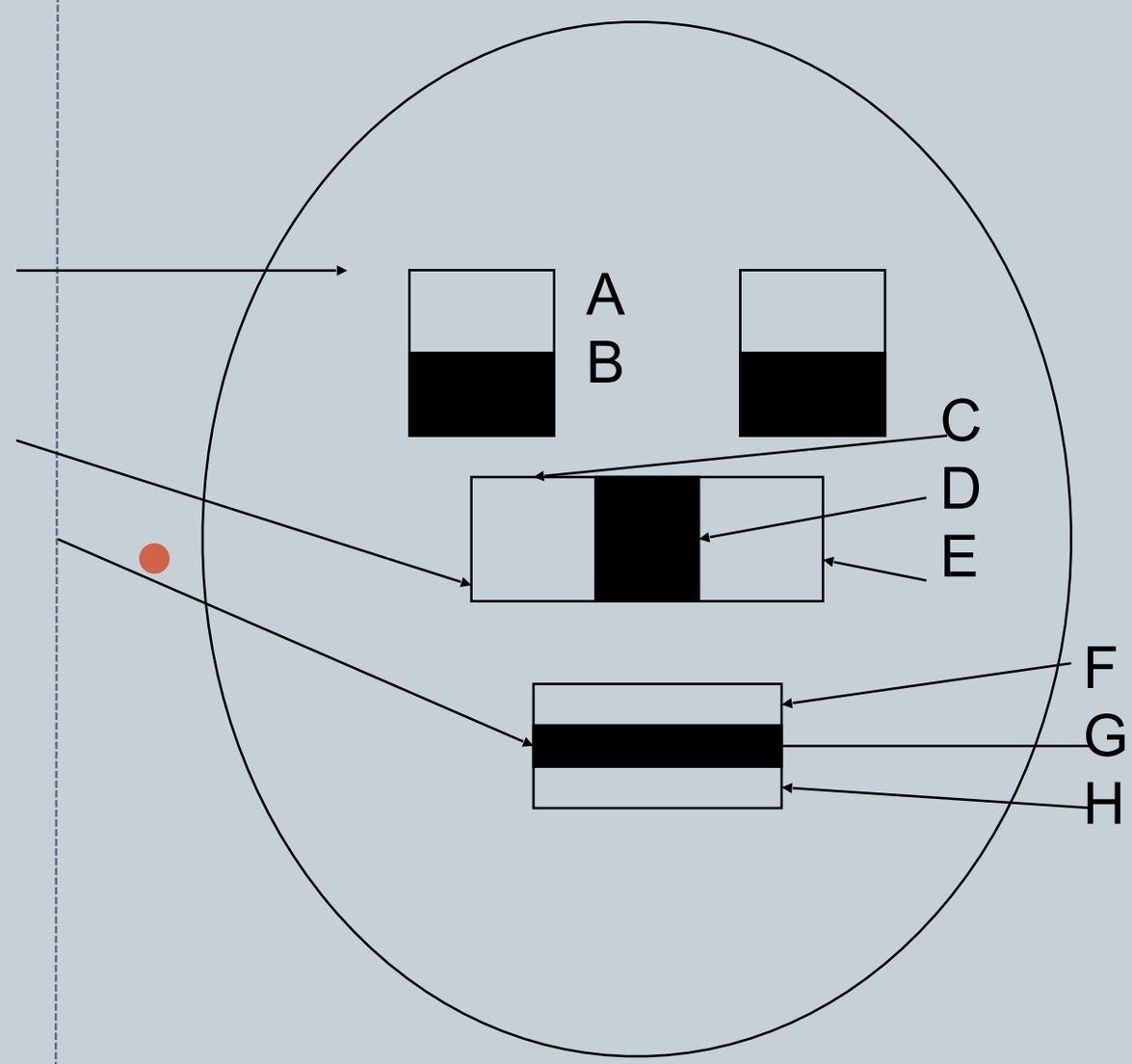
Examples

- The *integral image = sum of all* pixel values above and to the left of (*x*,*y*)
- Pixel P is at (x=3,y=2)
  - *integral image of P is =1+2+3+3+4+6*
- *integral image of Q is*
- *=1+2+3+3+4+6+5+2+4+0+2+3*



How These Features Can be Used?

- You may consider these features as face features
  - Left Eye: (Area_A-Area_B)
  - Nose :(Area_C+Area_E-Area_D)
  - Mouth:(Area_F+Area_H-Area_G)
- They can be different sizes, polarity and aspect ratios



Face Feature (simple)

Pixel values inside the areas

Shaded area	10	20	4
	7	45	7
White area	216	102	78
	129	210	111

F=Feat_val =
pixel sum in white area - pixel sum in shaded area
Example
•Pixel sum in white area=
216+102+78+129+210+111=846

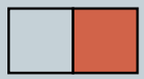
•Pixel sum in shared area= 10+20+4+7+45+7=93

Feat_val=F=846-93=753
If F>threshold,
    feature= +1
Else
    feature= -1   End if;
If we can choose threshold =700 , so feature is +1.

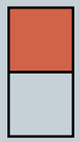
A face

4 basic types of Rectangular Features for (white_area)-(gray_area)

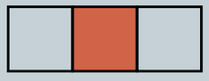
- Type n) Rows x columns
- Type 1) 1x2



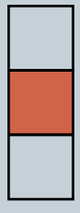
- Type 2) 2x1



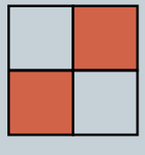
- Type 3) 1x3



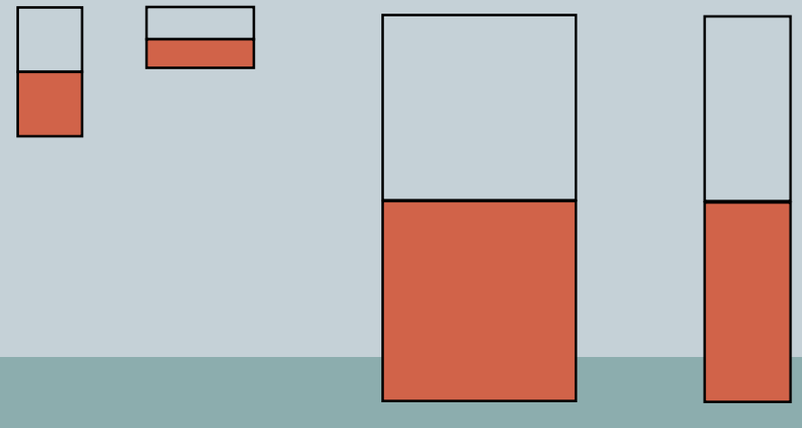
- Type 4) 3x1



- Type 5) 2x2



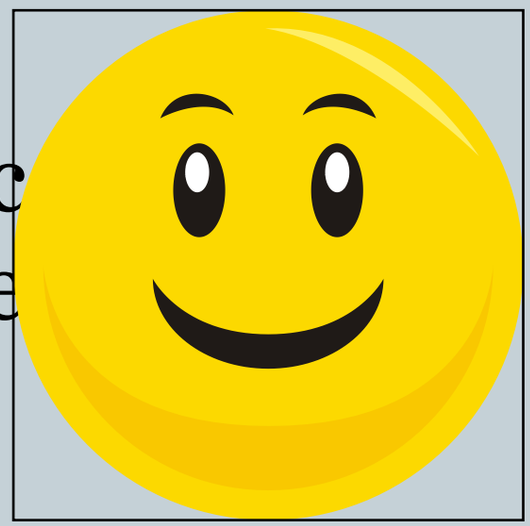
- Each basic type can have difference sizes and aspect ratios.
- I.e. the following feature windows are of the same type (Type2) even they have different sizes, or aspect ratios
- Each rectangle inside is of the same dimension



Faces can be of any sizes

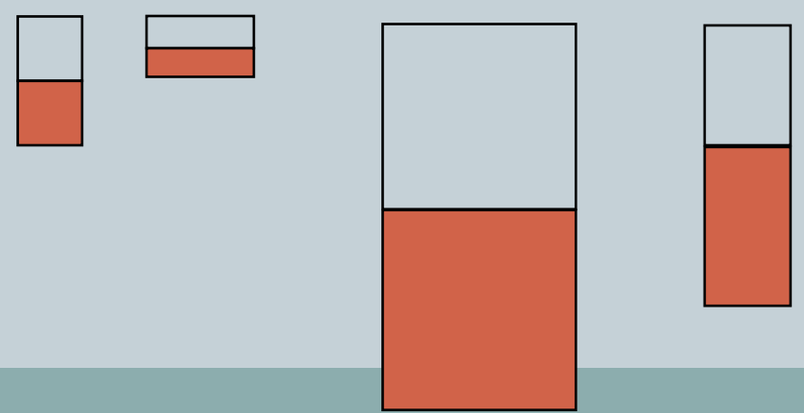
Example: a face can be big or small , from to 24 x24 to 1024x1024,

- T… fac… ze…



So, we need feature windows with different sizes.

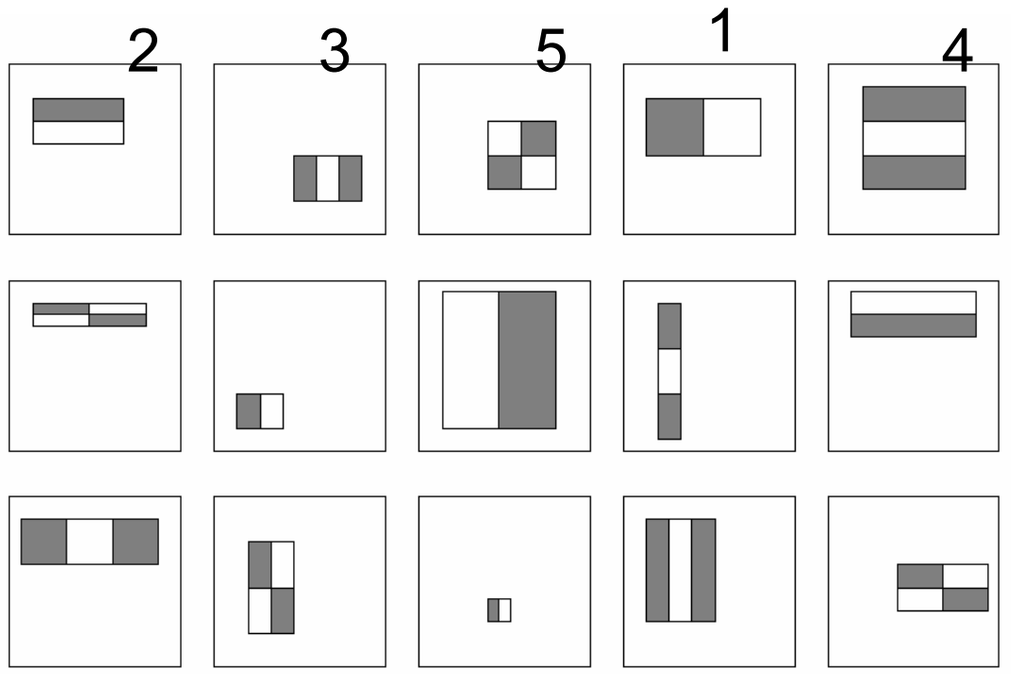
- As long as white/gray areas have the relations
- The followings are Type2 <u>Rectangular Features</u>



Many Features

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!

Some examples and their types
Fill in the types for the 2nd, 3rd rows



Standard Rectangular Feature Types

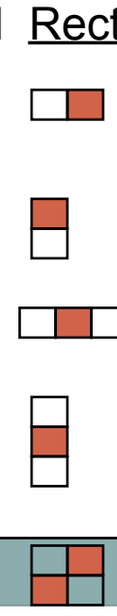
1)

2)

3)

4)

5)



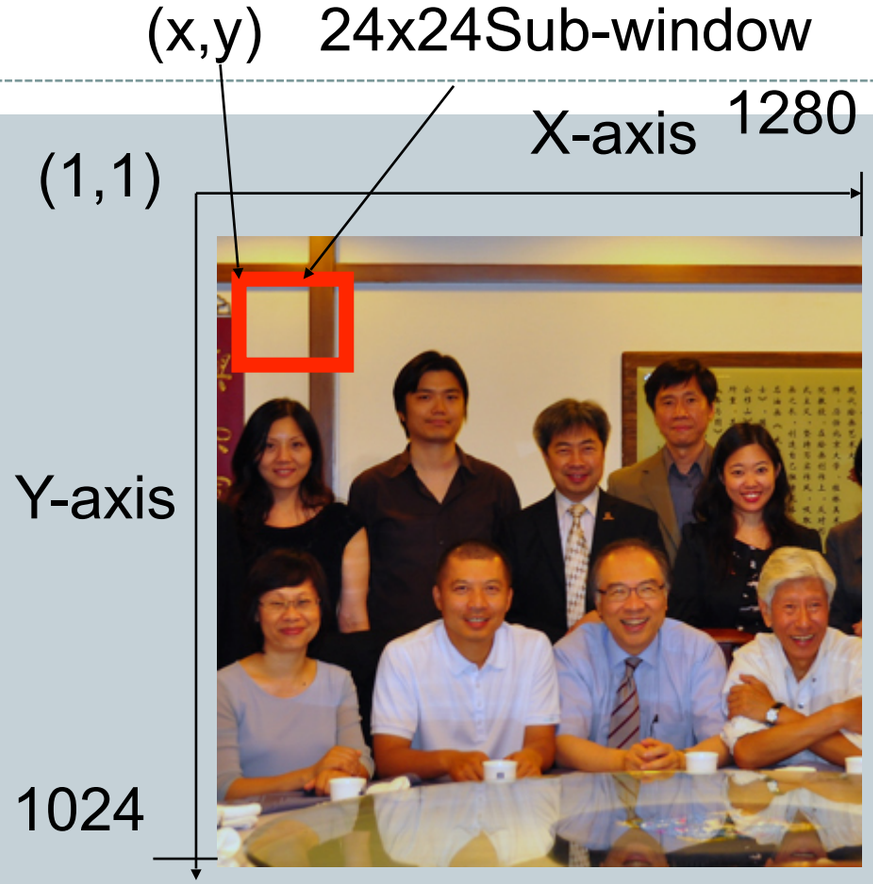
Example

- Use 24x24 base window
- For y=1;y<=1024;y++

  {For x=1;x<=1024;x++{

  - Set (x,y) = the left top corner of the 24x24 sub-window, different scales are needed to be considered too.
  - For the 24x24 sub-window, extract 162,336 features and see they combine to form a face or not.}

- }



Feature selection

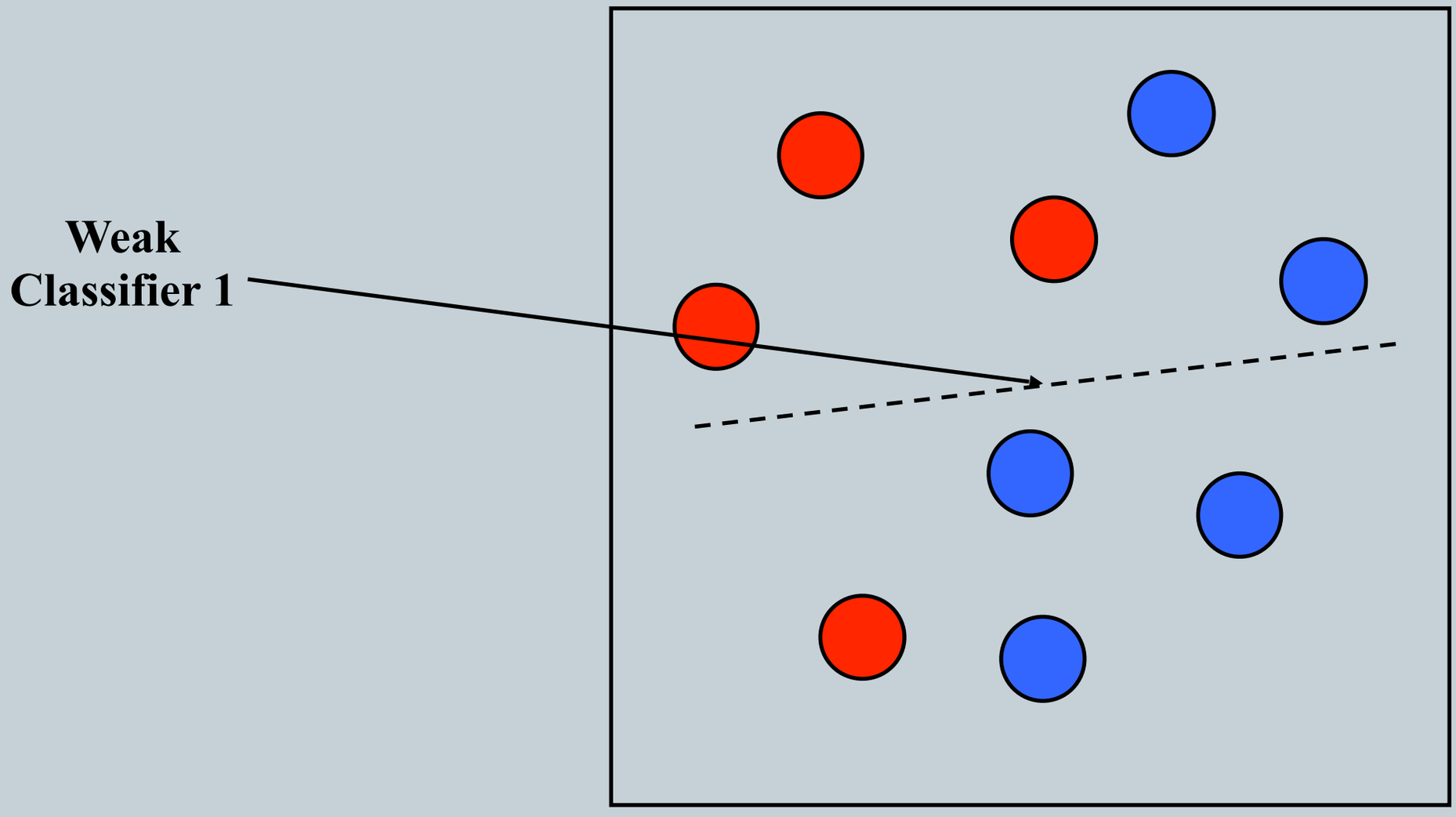
- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

Boosting

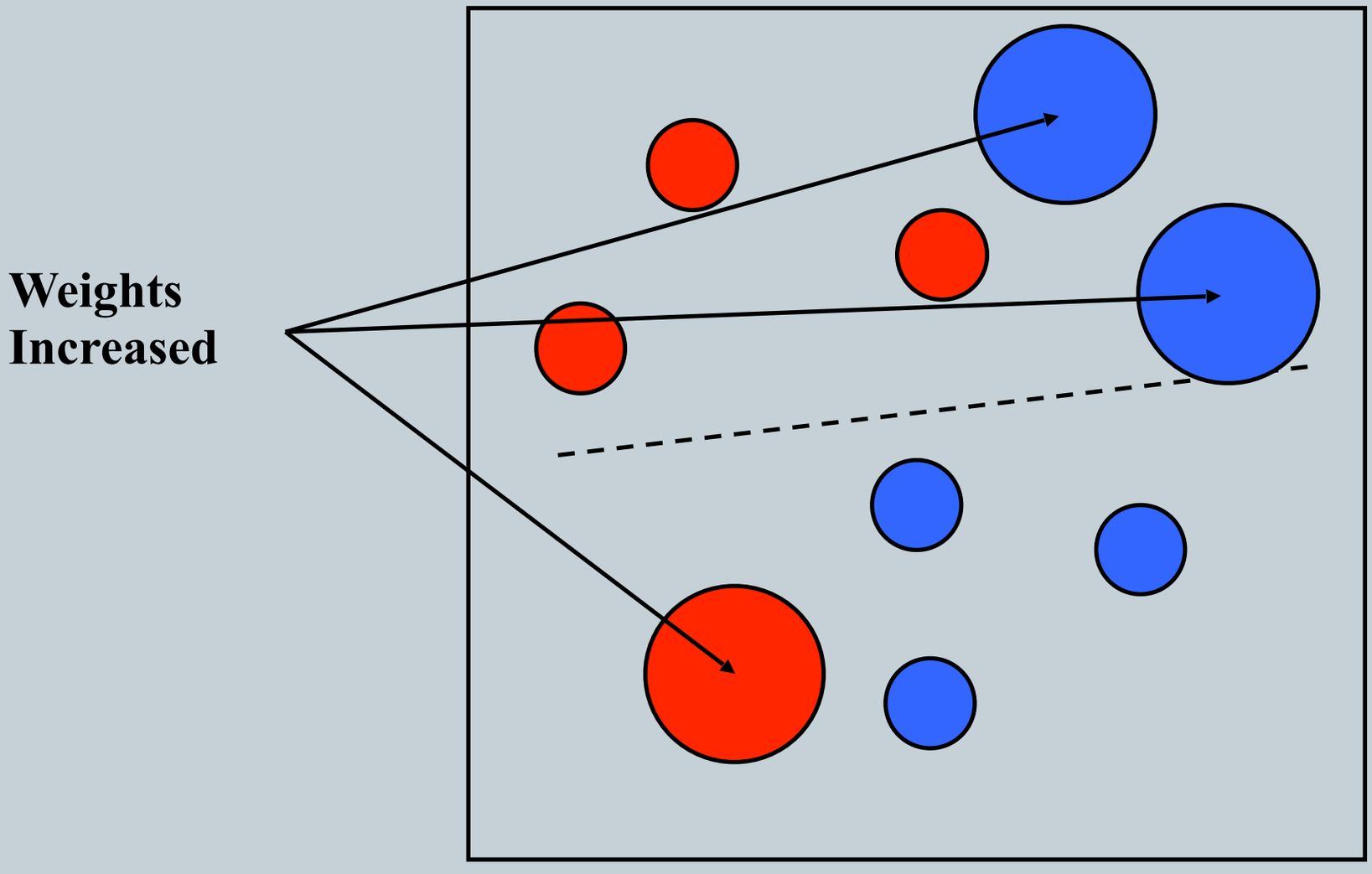
- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
  - A weak learner need only do better than chance
- Training consists of multiple *boosting rounds*
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - "Hardness" is captured by weights attached to training examples

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

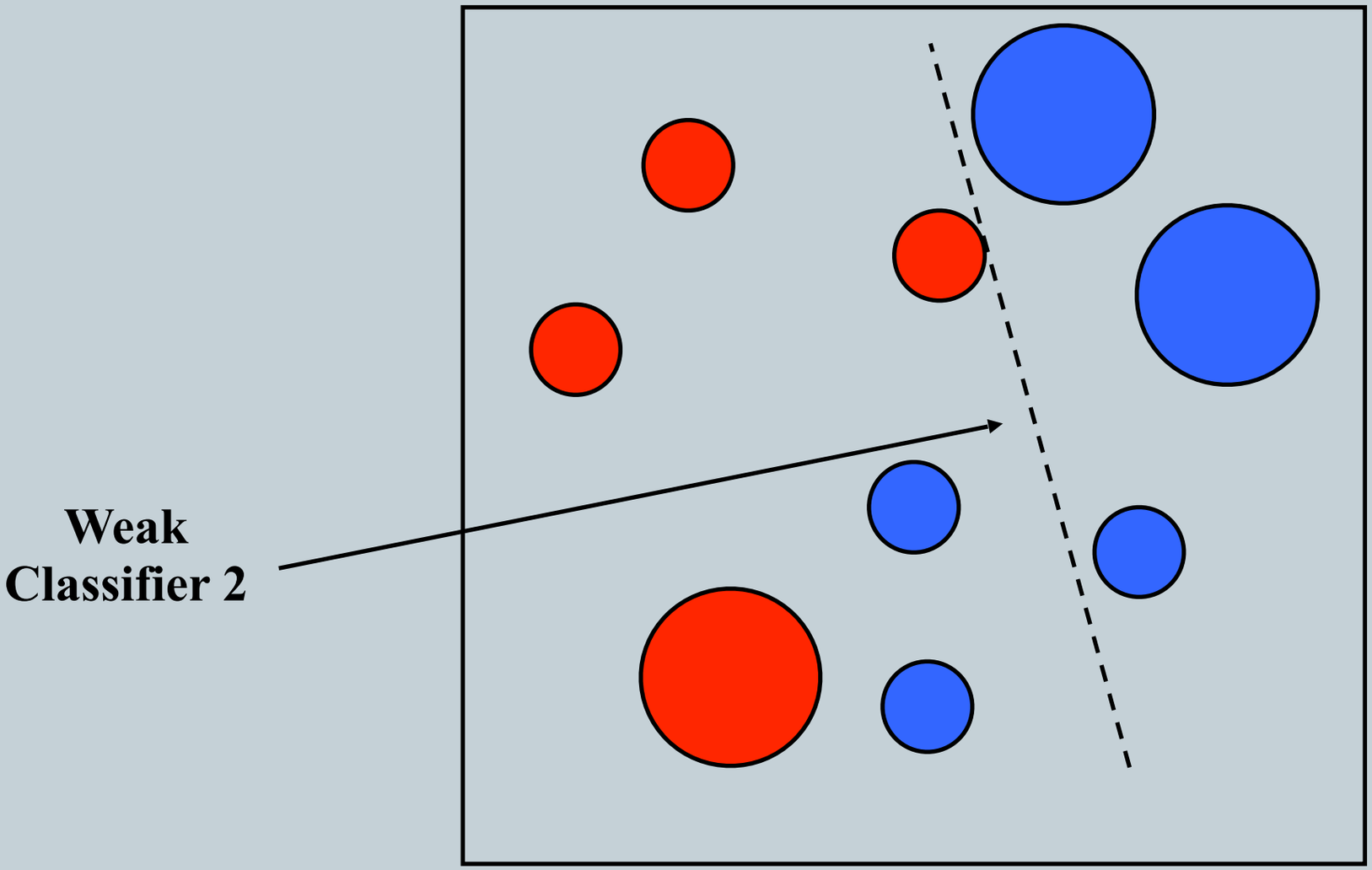
Boosting illustration



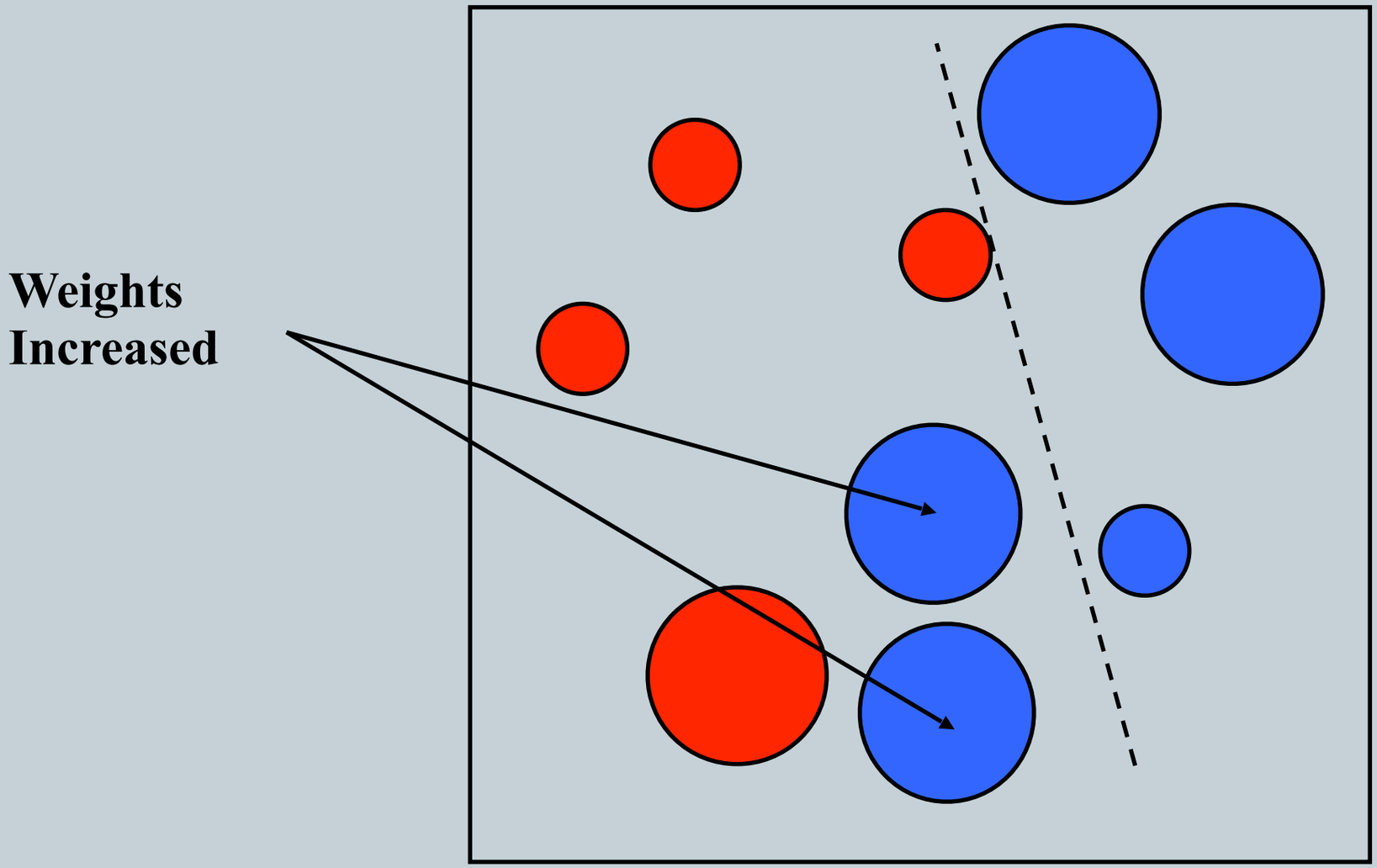
Boosting illustration



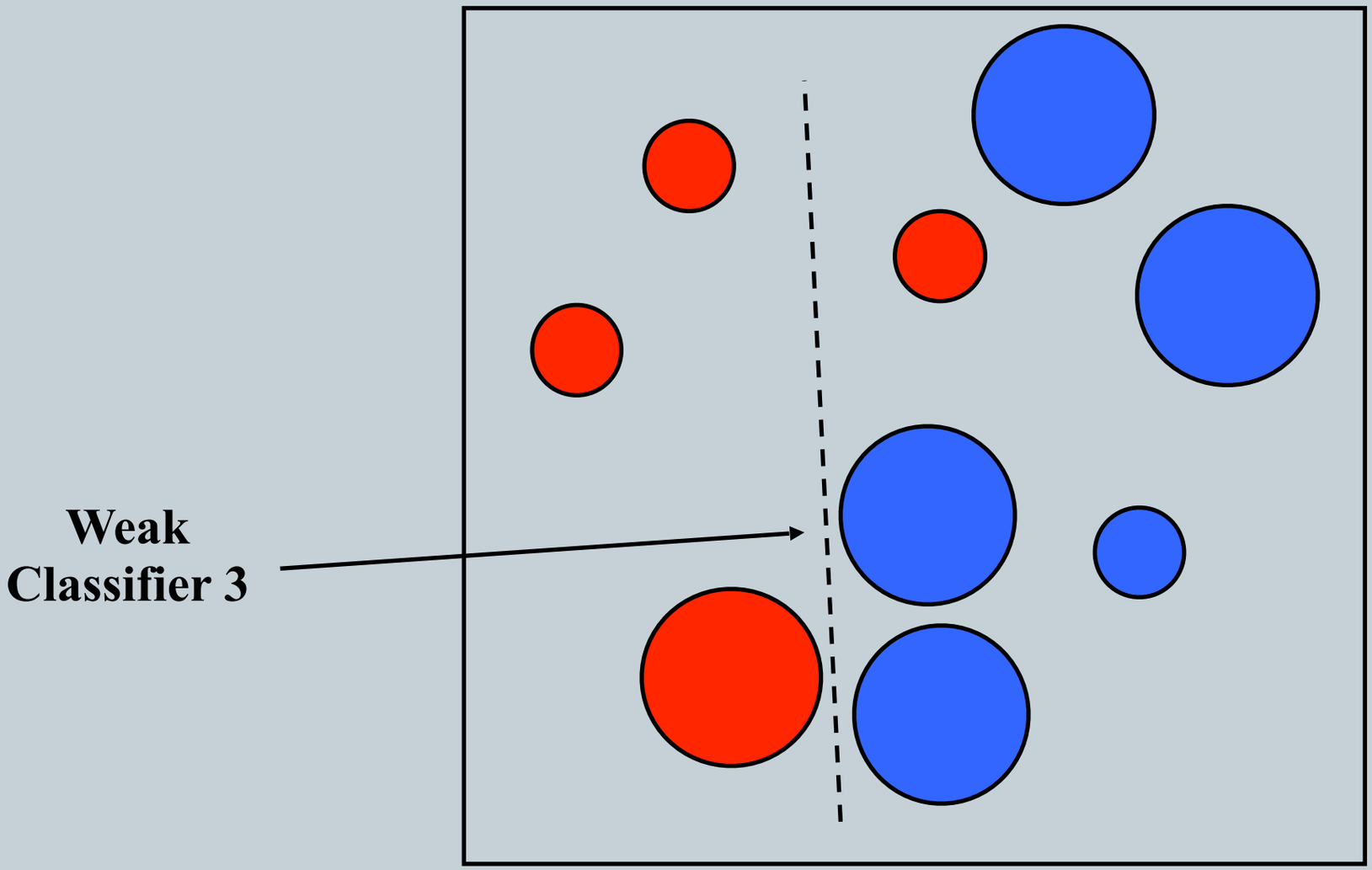
Boosting illustration



Boosting illustration

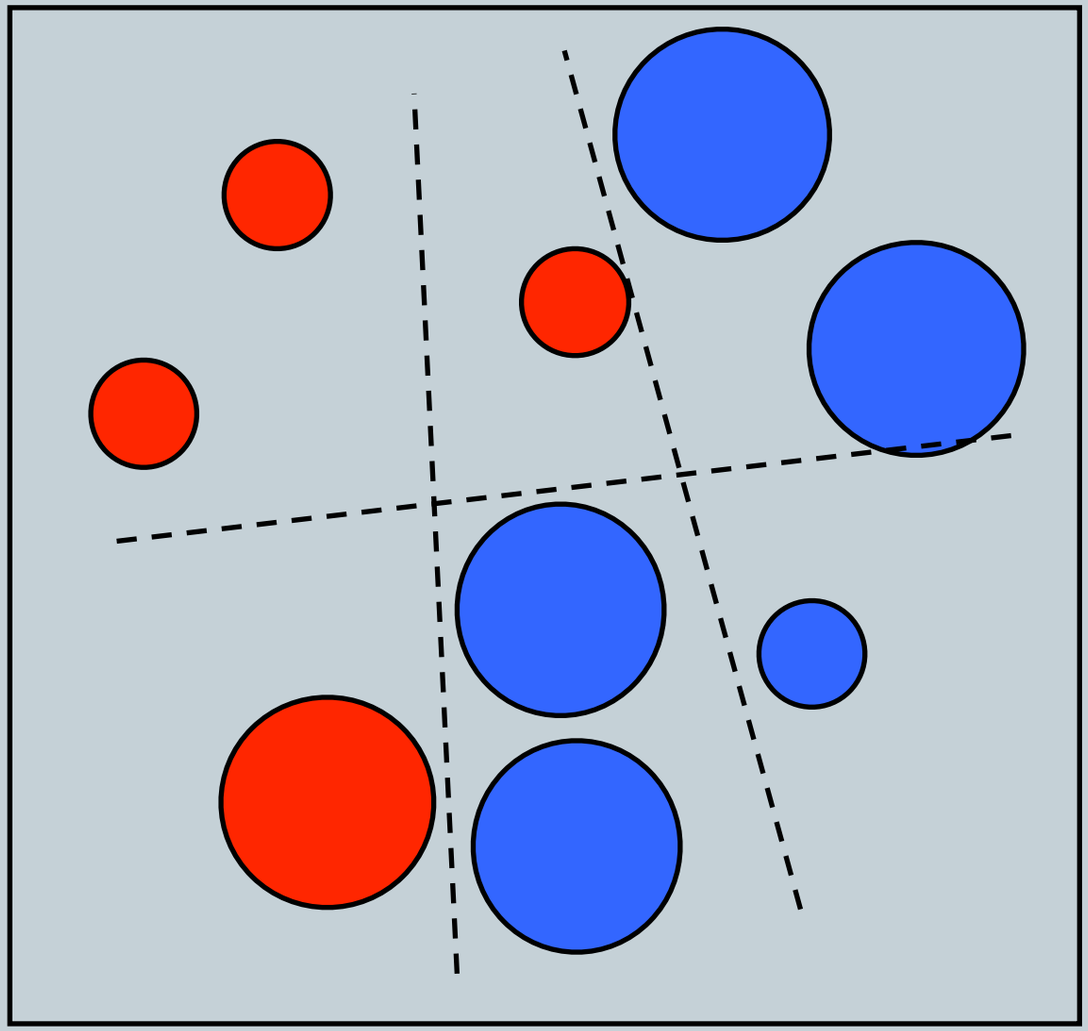


Boosting illustration



Boosting illustration

Final classifier is a combination of weak classifiers

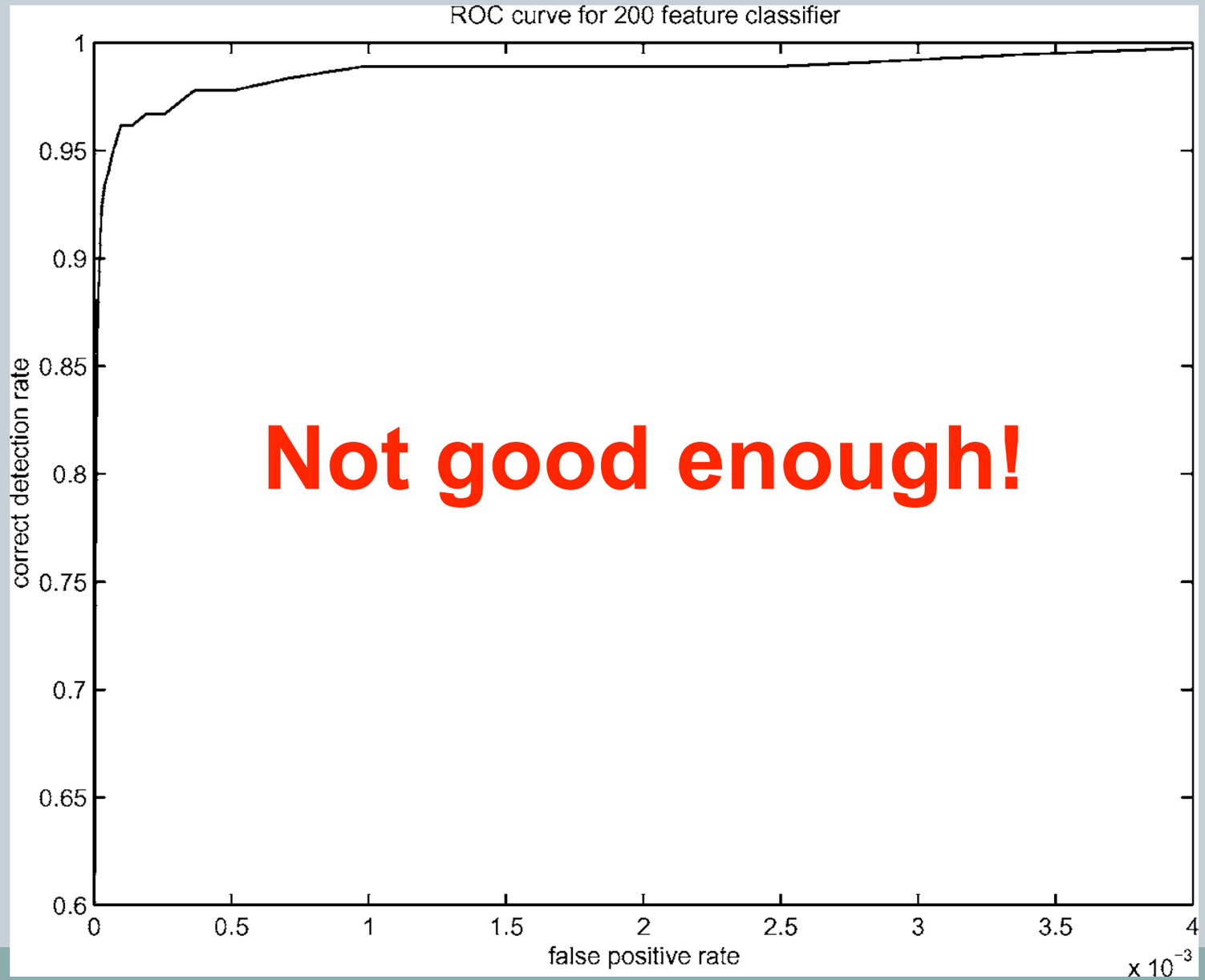


Face detection using Adaboost

- AdaBoost training
  - E.g. Collect 5000 faces, and 9400 non-faces. Different scales.
  - Use AdaBoost for training to build a strong classifier.
  - Pick suitable features of different scales and positions, pick the best few. (Take months to do , details is in [Viola 2004] paper)
- Testing
  - Scan through the image, pick a window and rescale it to 24x24,
  - Pass it to the strong classifier for detection.
  - Report face, if the output is positive

Boosting for face detection

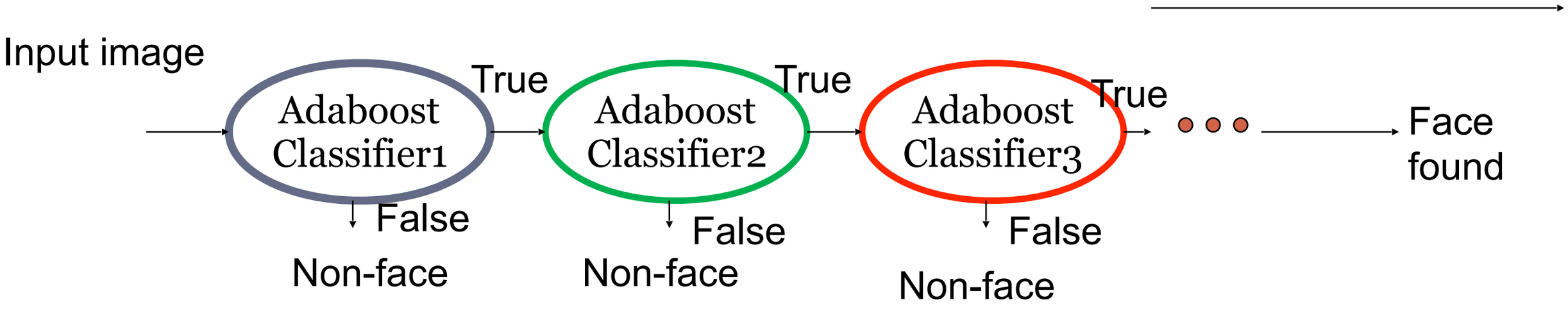
- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084



Receiver operating characteristic (ROC) curve

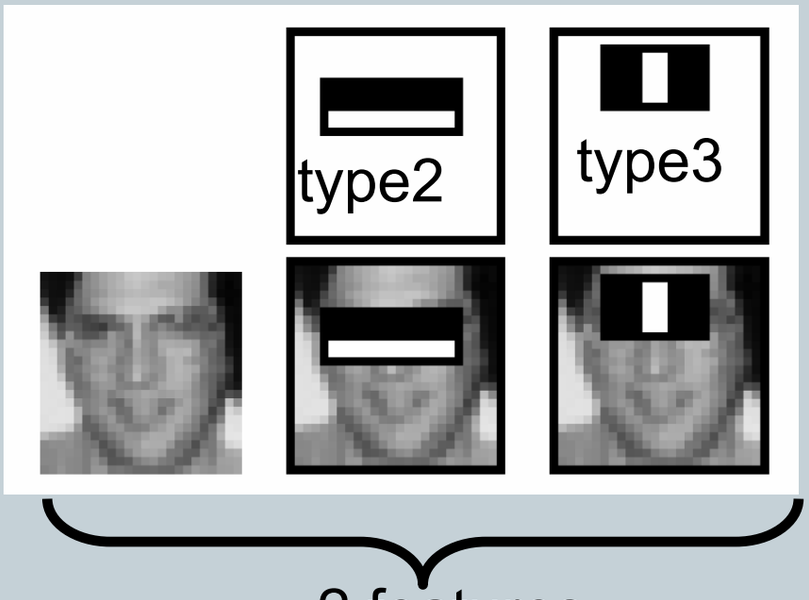
To improve false positive rate: Attentional cascade

- Cascade of many AdaBoost strong classifiers.
- Begin with simple classifiers to reject many negative sub-windows.
- Many non-faces are rejected at the first few stages.
- Hence the system is efficient enough for real time processing.



Example

- More features for later stages in the cascade [viola2004]



2 features

10 features

25 features

50 features…

Input image

Adaboost Classifier1

True

Adaboost Classifier2

True

Adaboost Classifier3

True

Face found

False

Non-face

False

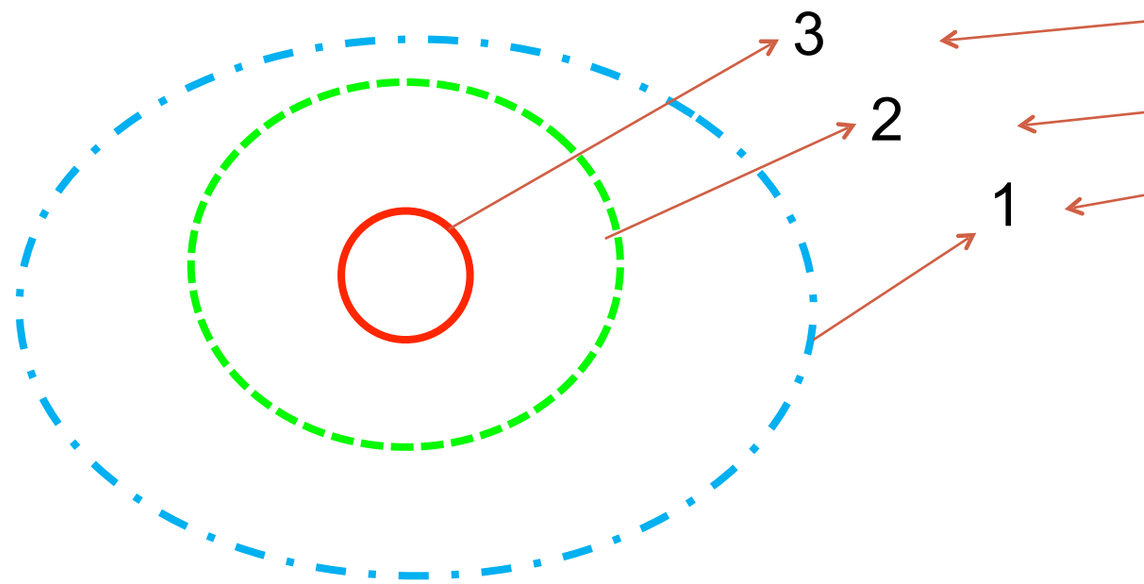
Non-face

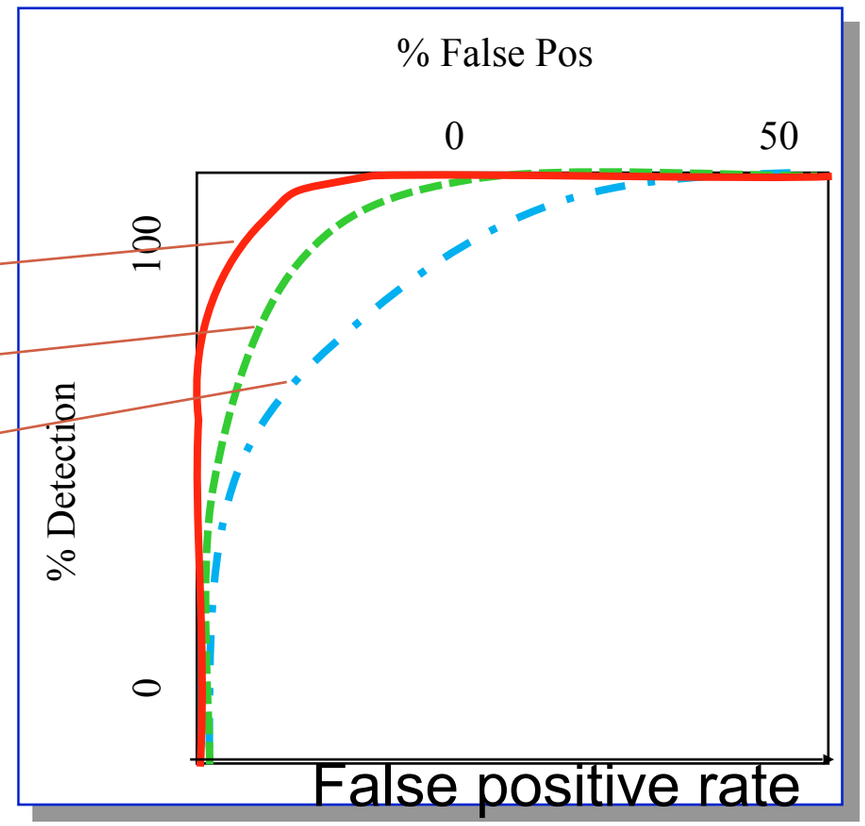
False

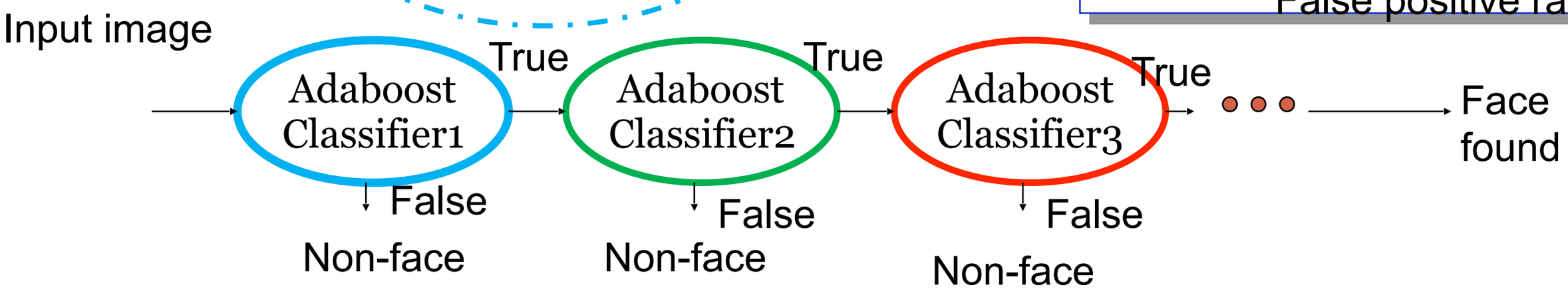
Non-face

Example

- Chain classifiers are progressively more complex and have lower false positive rates:







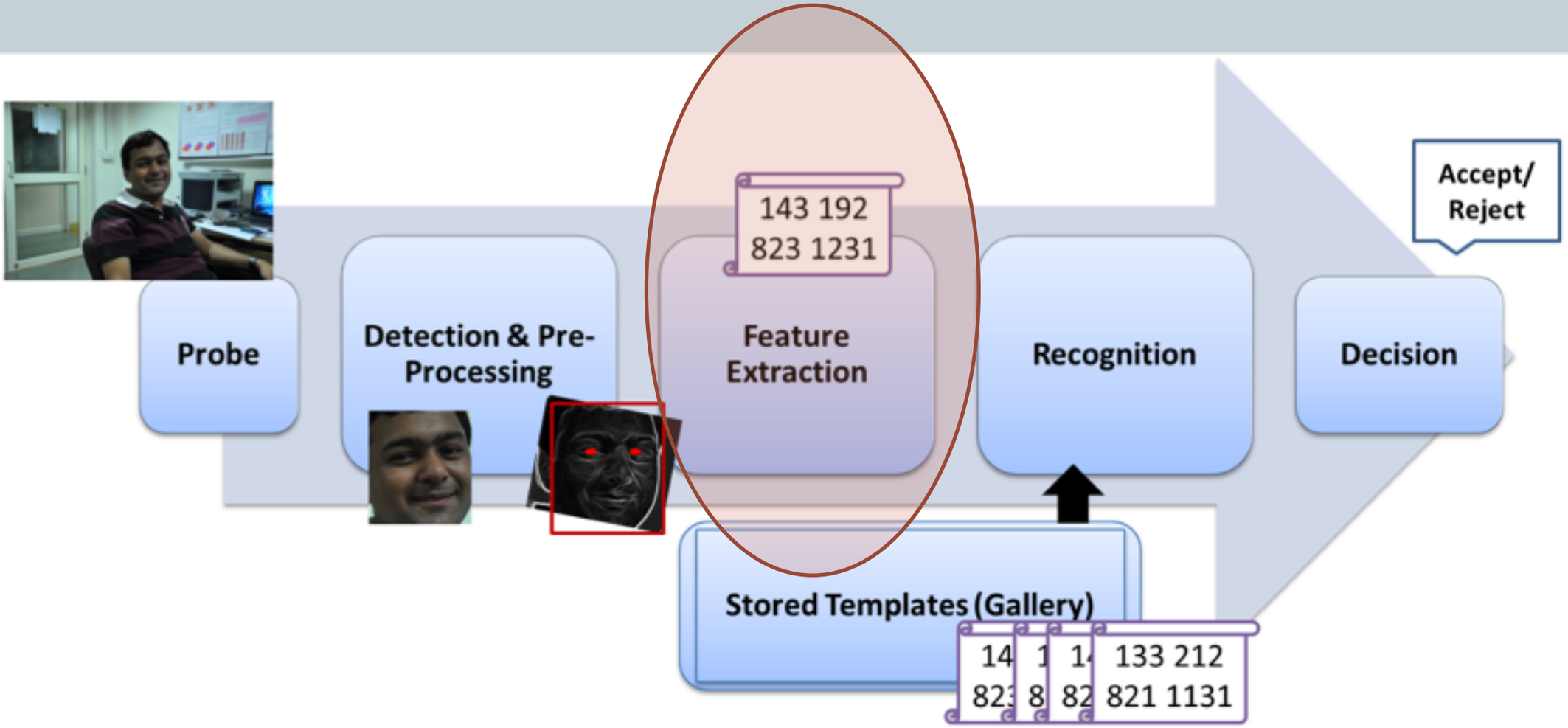
CNN based Object Detection

We will come back to it when we discuss CNN based object recognition

Object Recognition

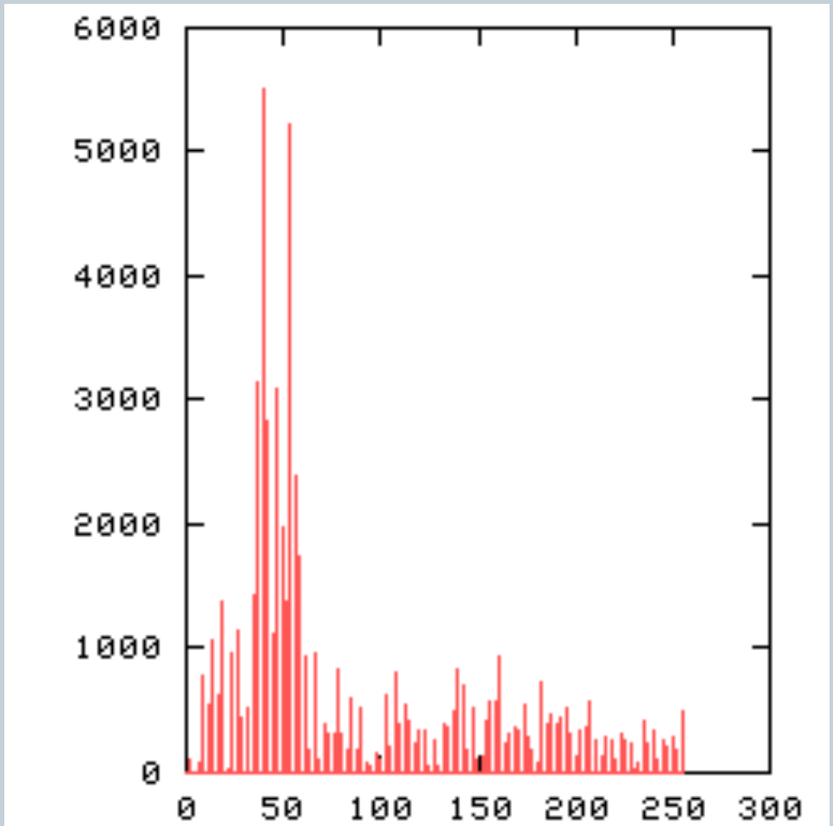
- Once object is detected, what's next?

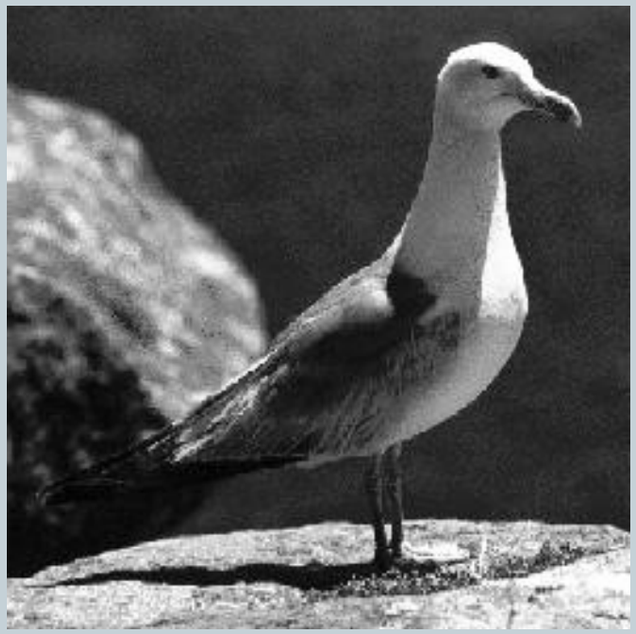
Let's take a look at the pipeline



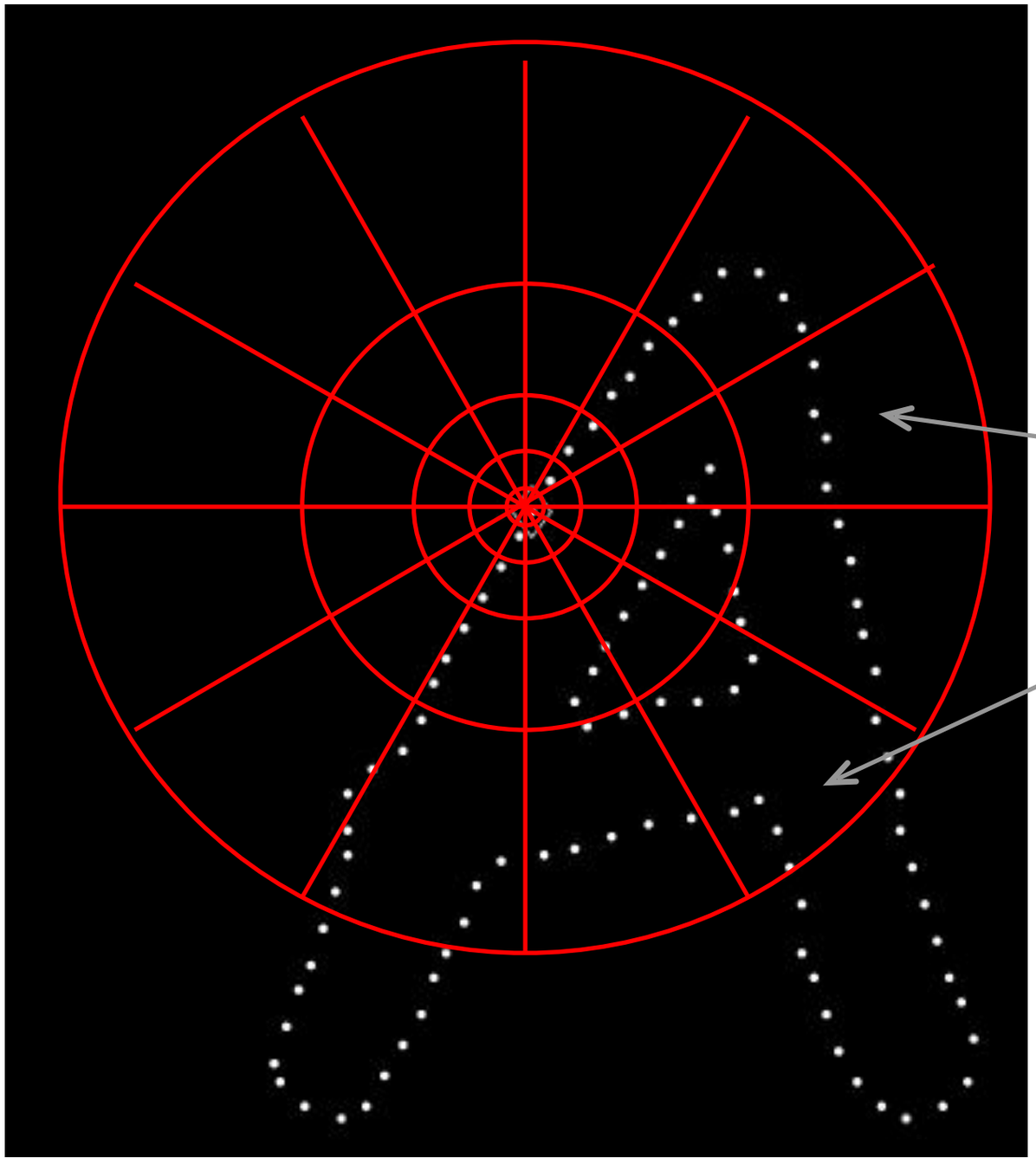
Feature Extraction

- Features are “representations” of an image





Local Descriptors: Shape Context



Count the number of points inside each bin, e.g.:

Count = 4

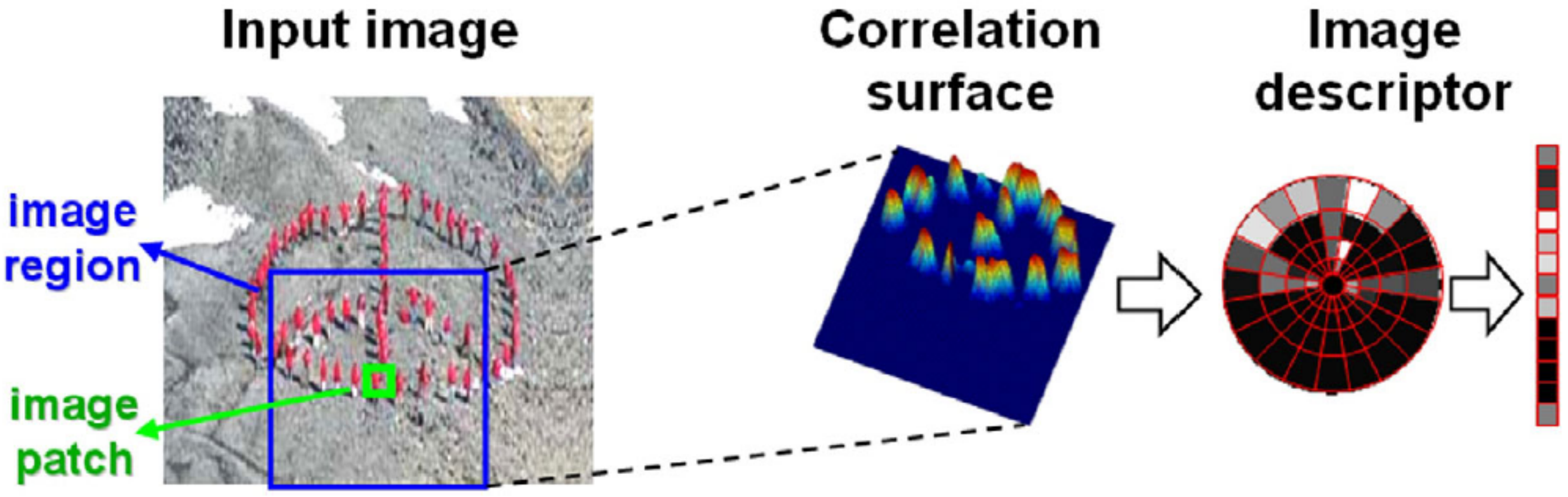
⋮

Count = 10

Log-polar binning: more precision for nearby points, more flexibility for farther points.

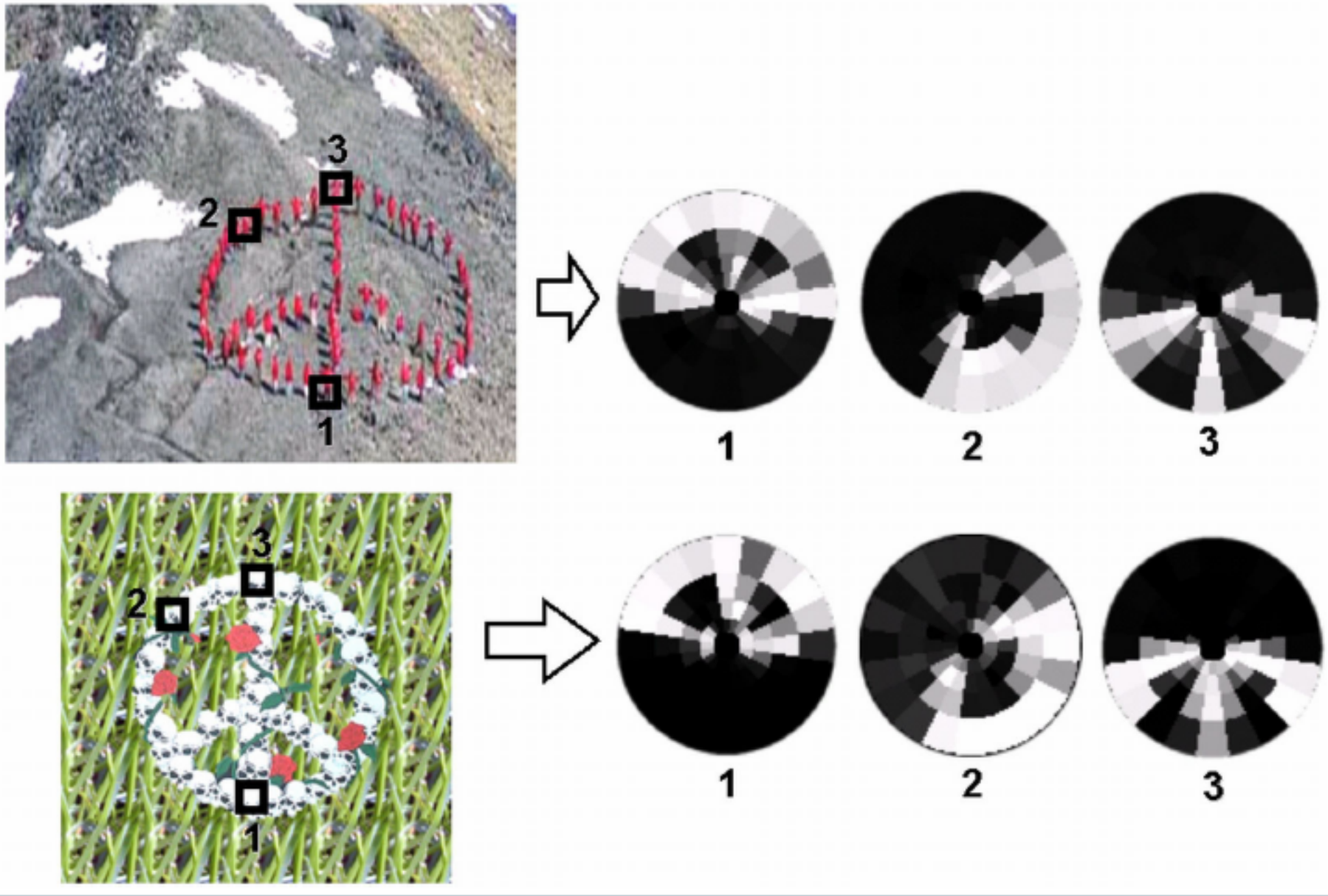
Belongie & Malik, ICCV 2001

Self-similarity Descriptor



Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

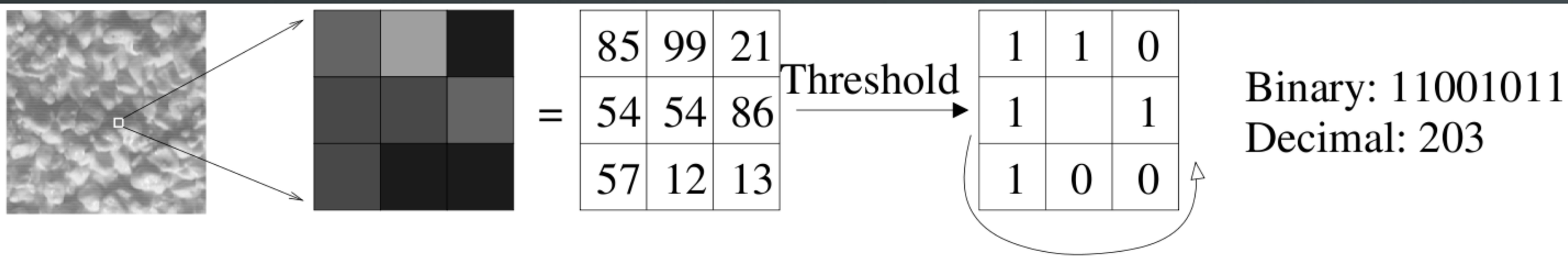
Self-similarity Descriptor



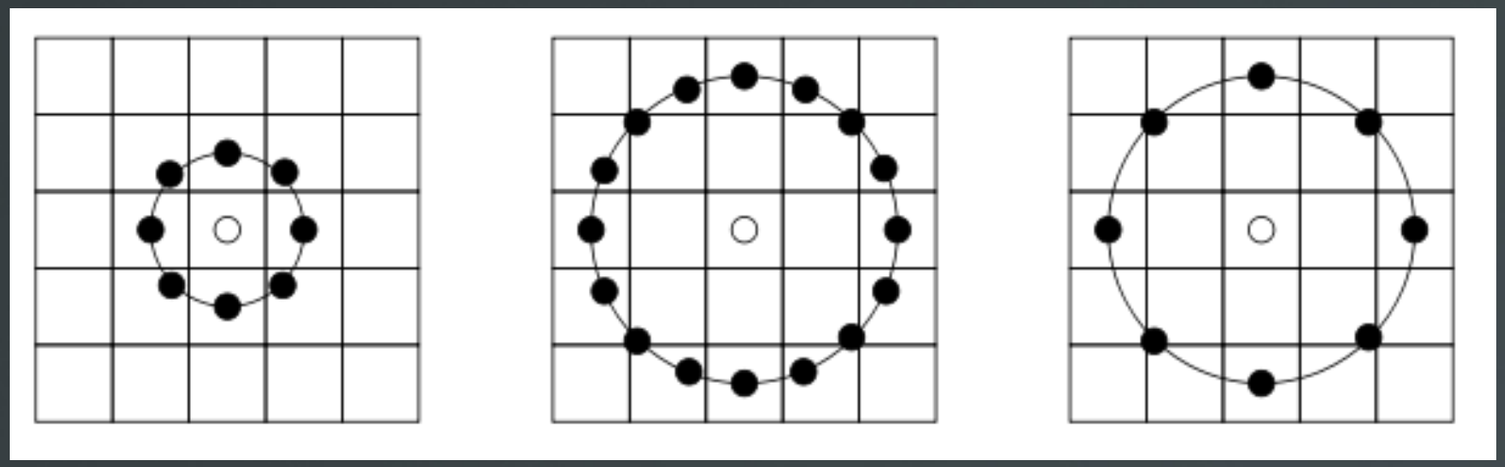
Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Texture

- Consider each pixel to have some texture information.
- This information is encoded using the LBP operator.

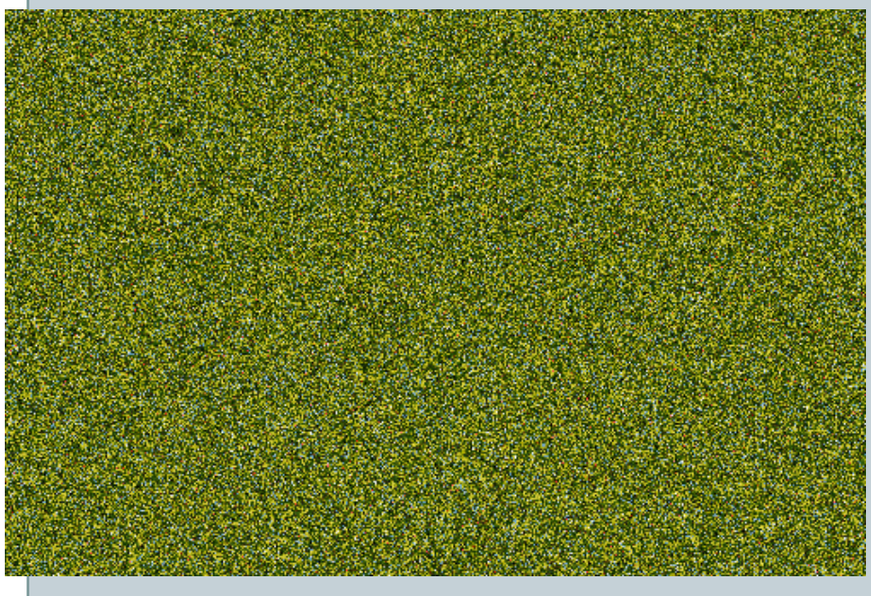


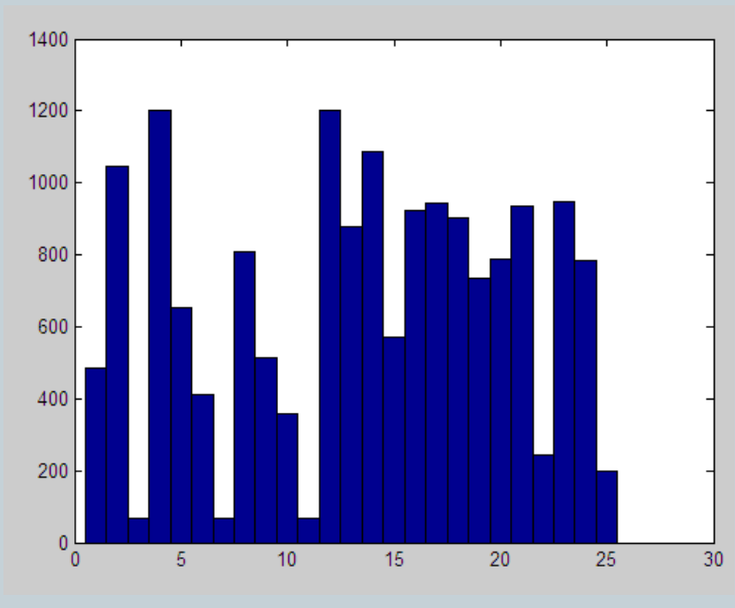
Other LBP operators

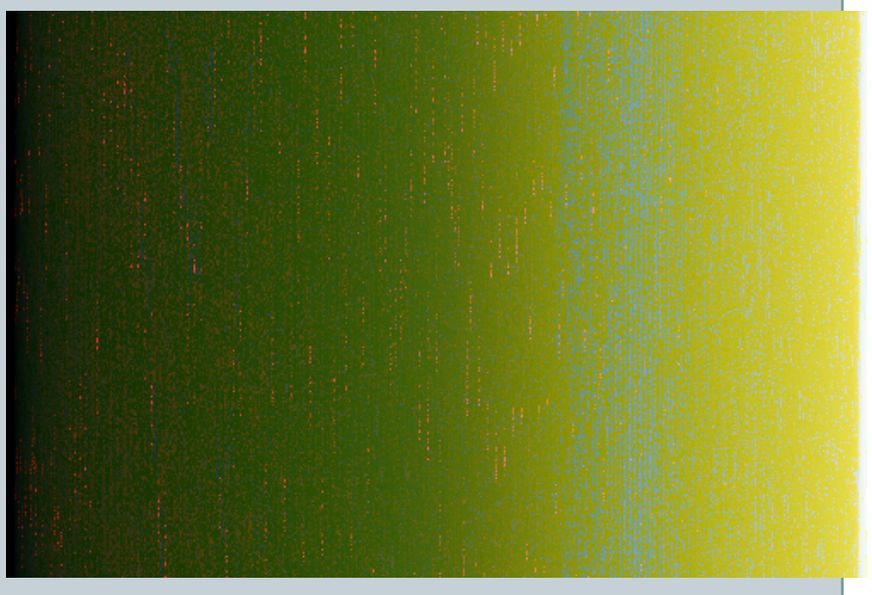


What does pattern image look like?

But what about layout?







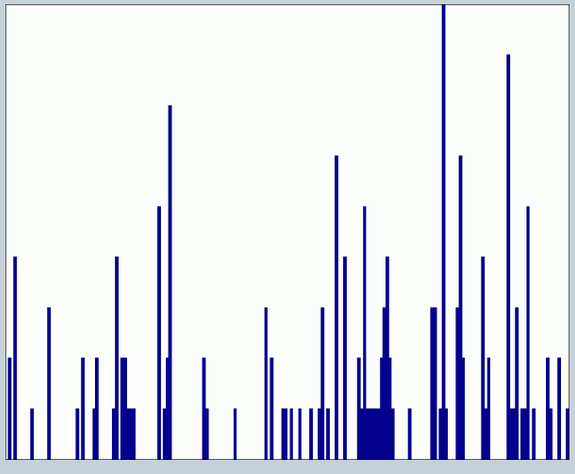
All of these images have the same color histogram

Spatial pyramid

Compute histogram in each spatial bin

Spatial pyramid representation

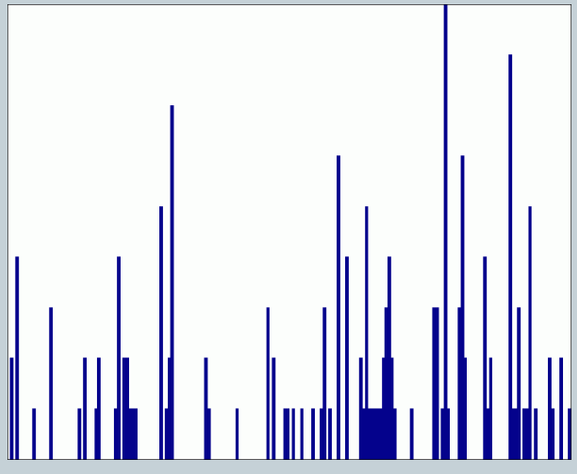
- Locally orderless representation at several levels of resolution



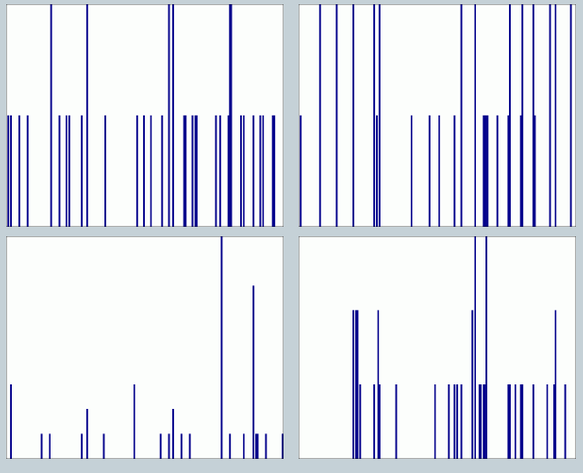
level 0

Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramid representation



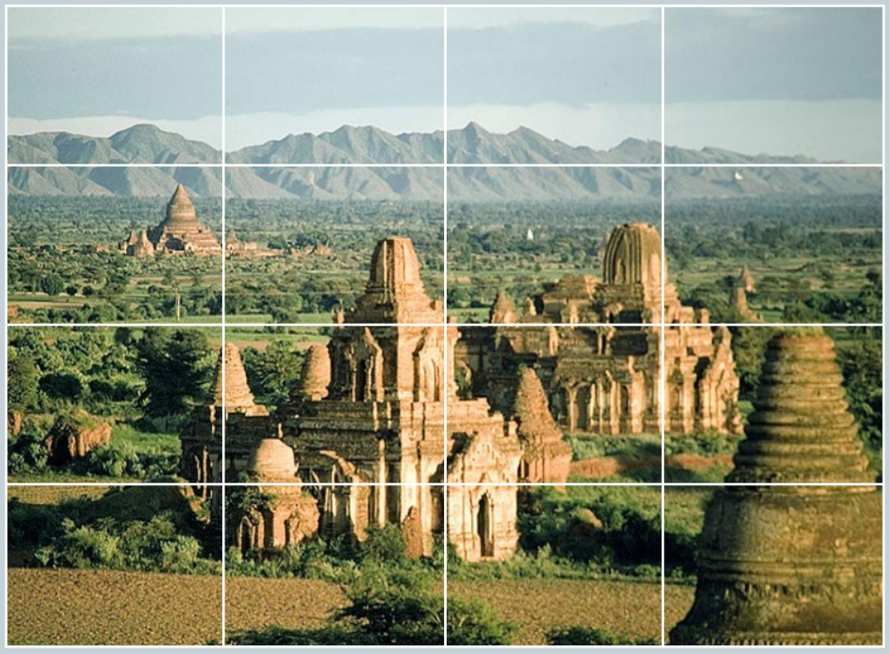
level 0

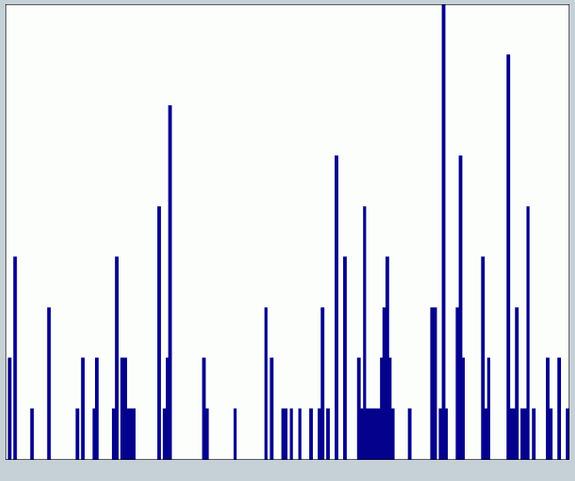


level 1

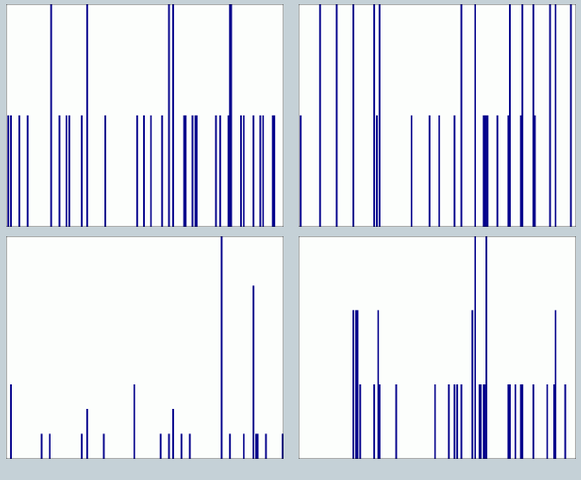
Lazebnik, Schmid & Ponce (CVPR 2006)

Spatial pyramid representation

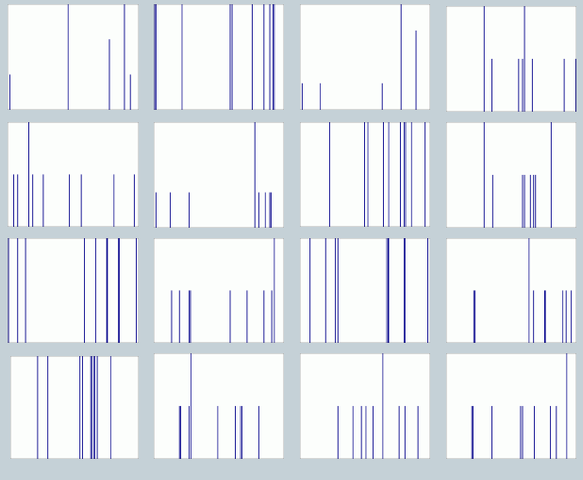




level 0



level 1



level 2

Lazebnik, Schmid & Ponce (CVPR 2006)

Distance Measures

$$d(x, y) \geq 0$$

The distance of two objects x and y can't be less than zero.

$$d(x, y) = 0 \iff x = y$$

Two perfectly similar objects have distance zero.

$$d(x, y) = d(y, x)$$

The distance between x and y is the same as between y and x — it doesn't matter which way you go.

$$d(x, z) \leq d(x, y) + d(x, z)$$

If you take a "detour" via y on your way from x to z, your path can't be shorter than if you had taken the direct route. This is called the triangle inequality.

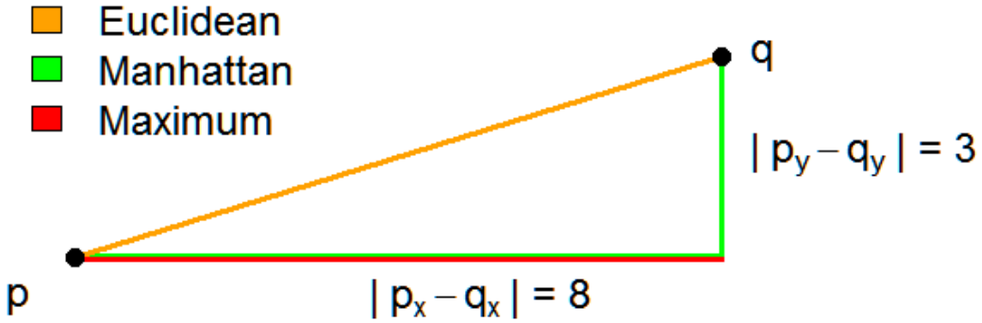
Distance Measures

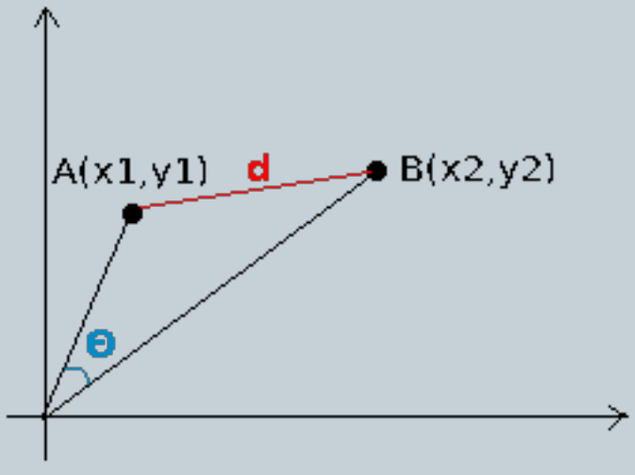
$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$\sum_{i=1}^{n}|x_i - y_i|$$

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}}$$

$$\cos\theta = \frac{x \cdot y}{\|x\|\|y\|}$$

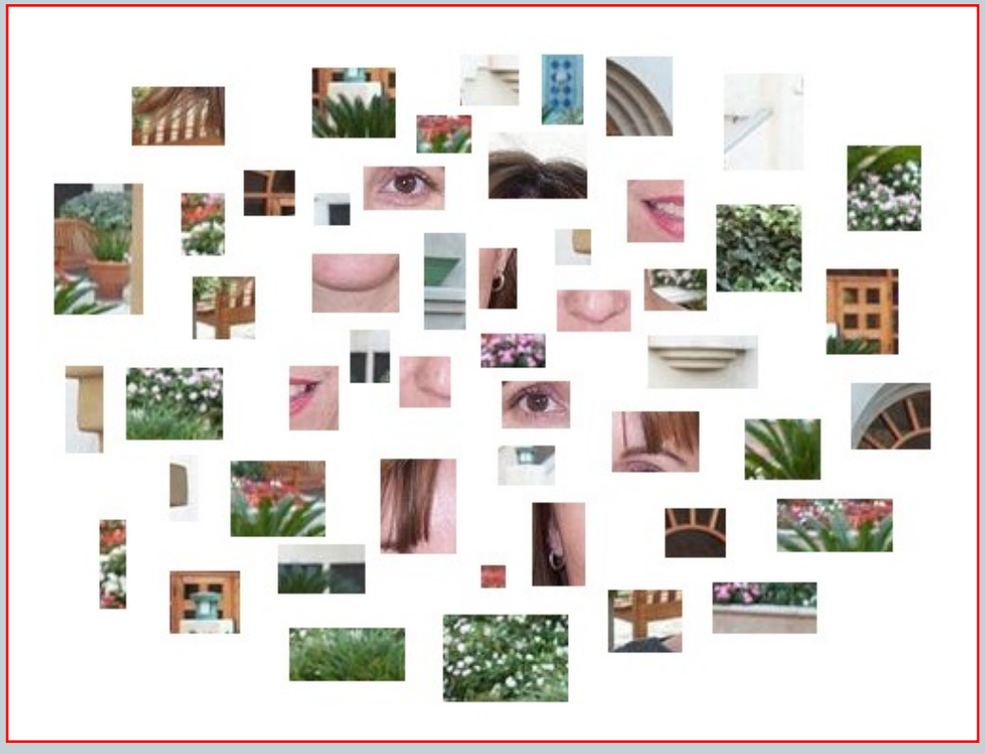




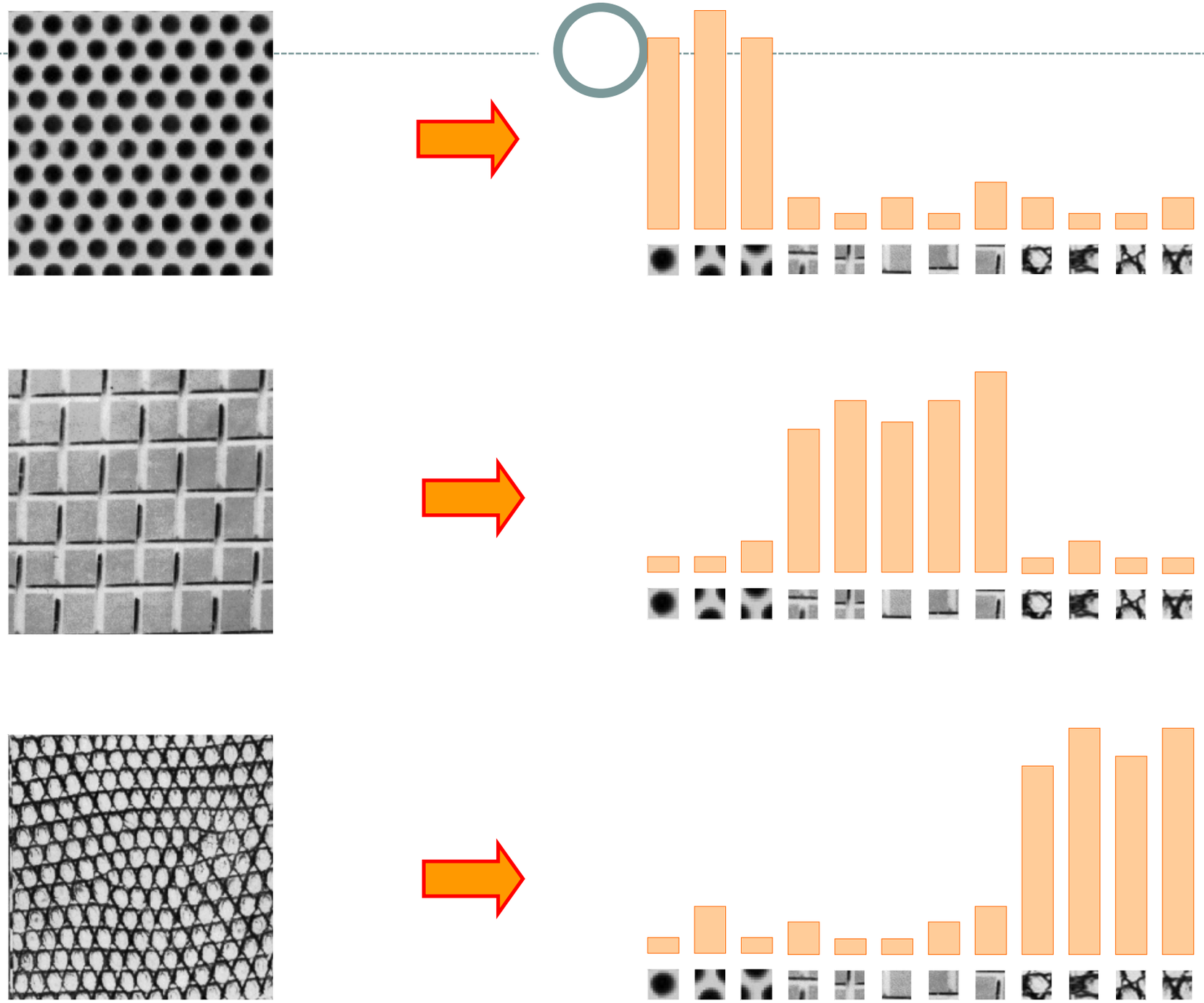
Bag of Word Approach

- Have you heard “bag of word”?

Bag-of-features models



Origin 1: Texture recognition



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

Origin 2: Bag-of-words models

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

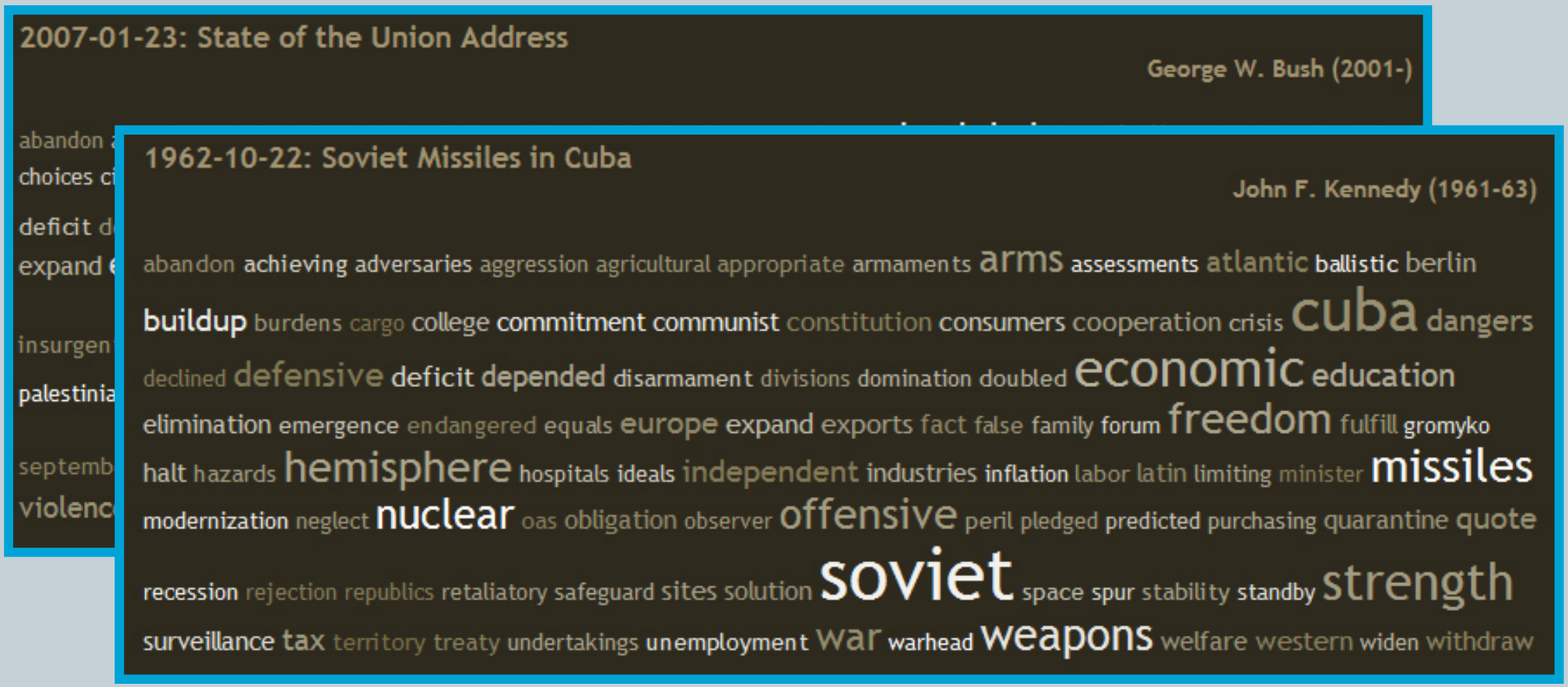
Origin 2: Bag-of-words models

- 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

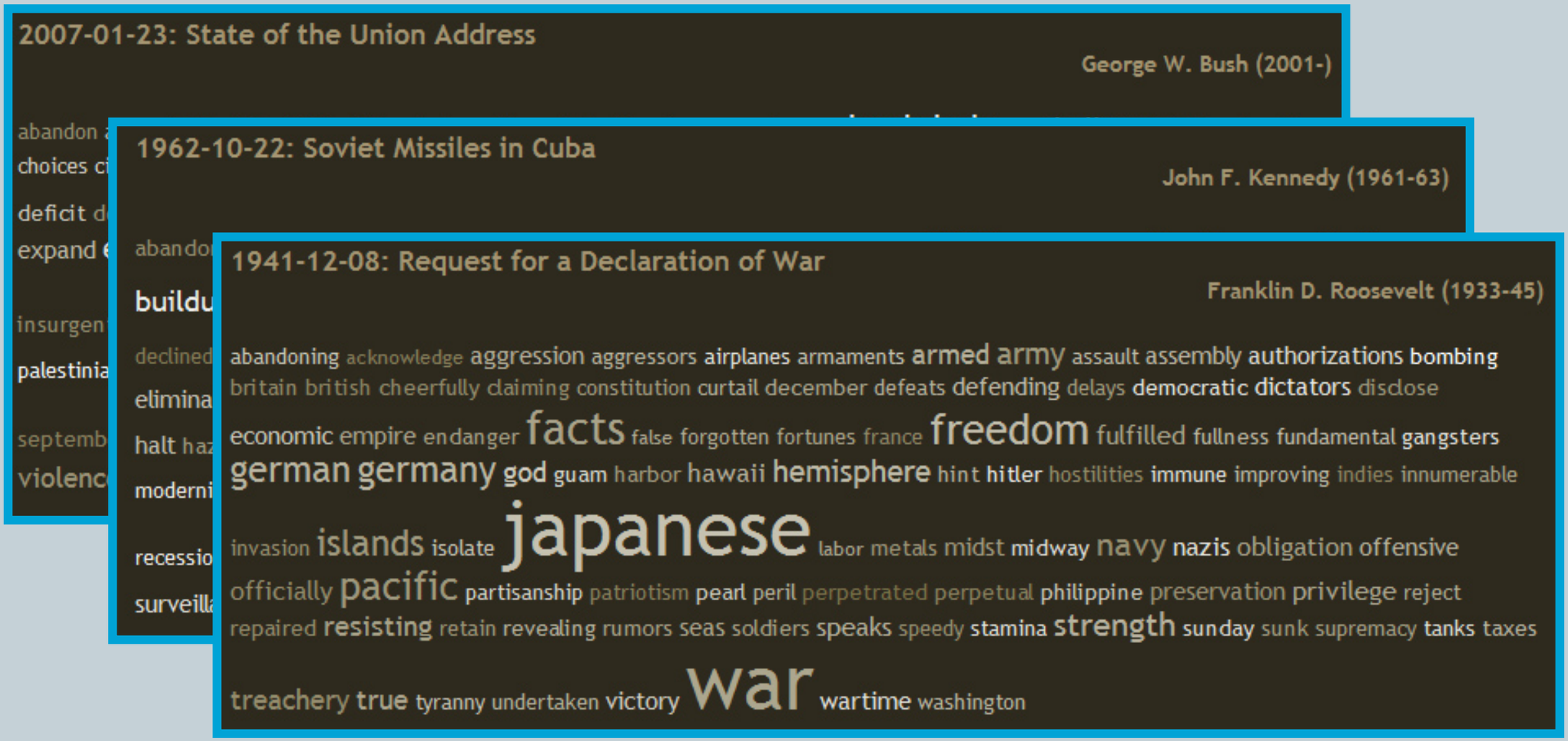
- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)



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

Origin 2: Bag-of-words models

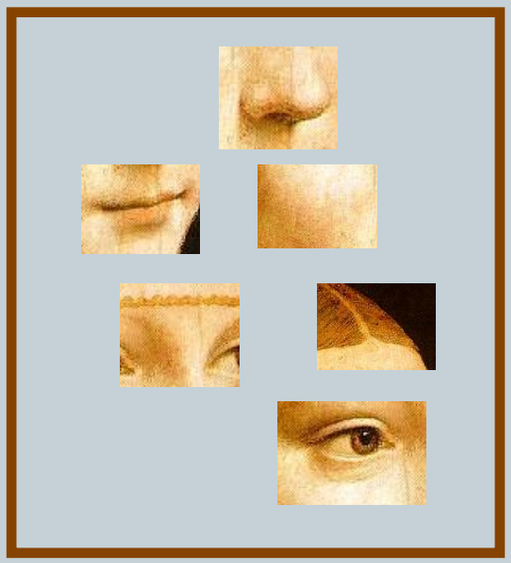
- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

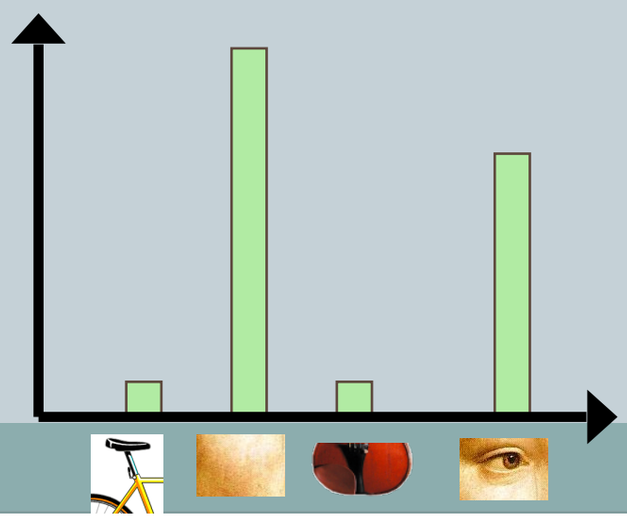


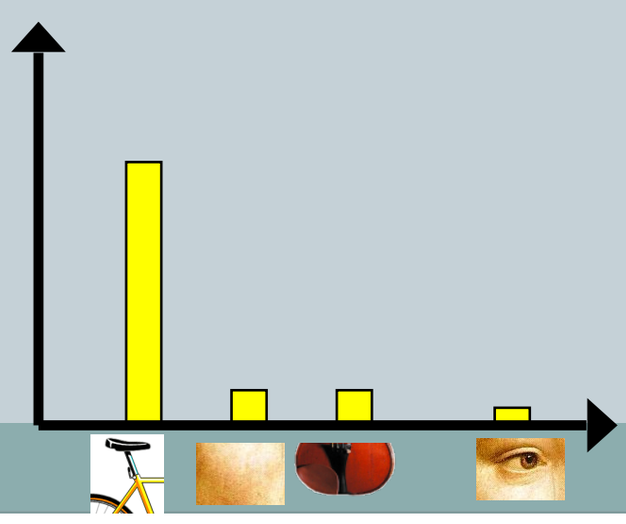
US Presidential Speeches Tag Cloud
http://chir.ag/phernalia/preztags/

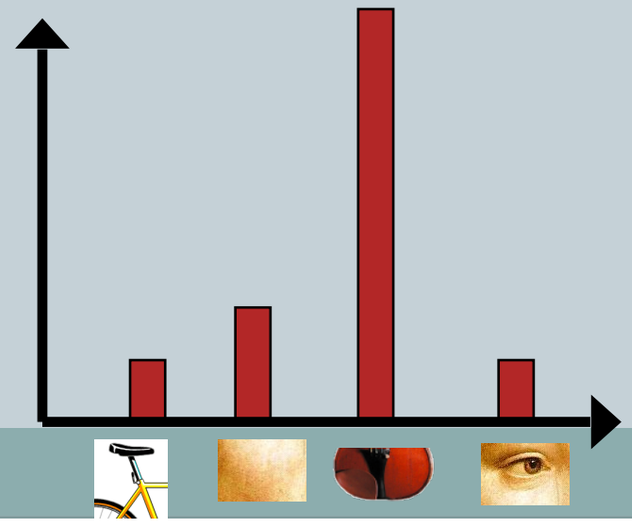
Bag-of-features steps

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





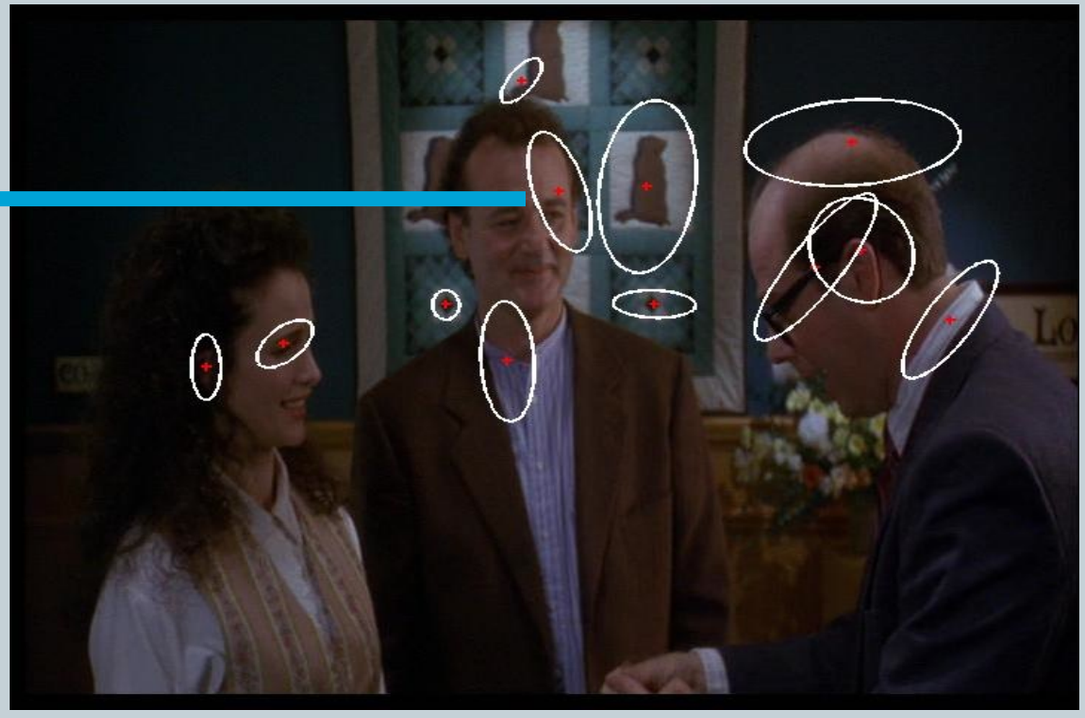




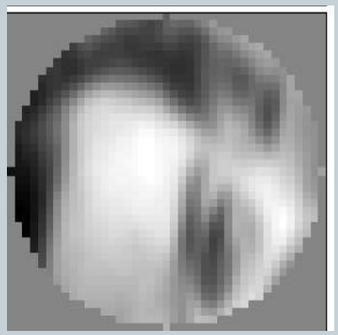
1. Feature extraction

- Regular grid or interest regions

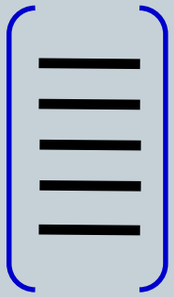
1. Feature extraction



Detect patches



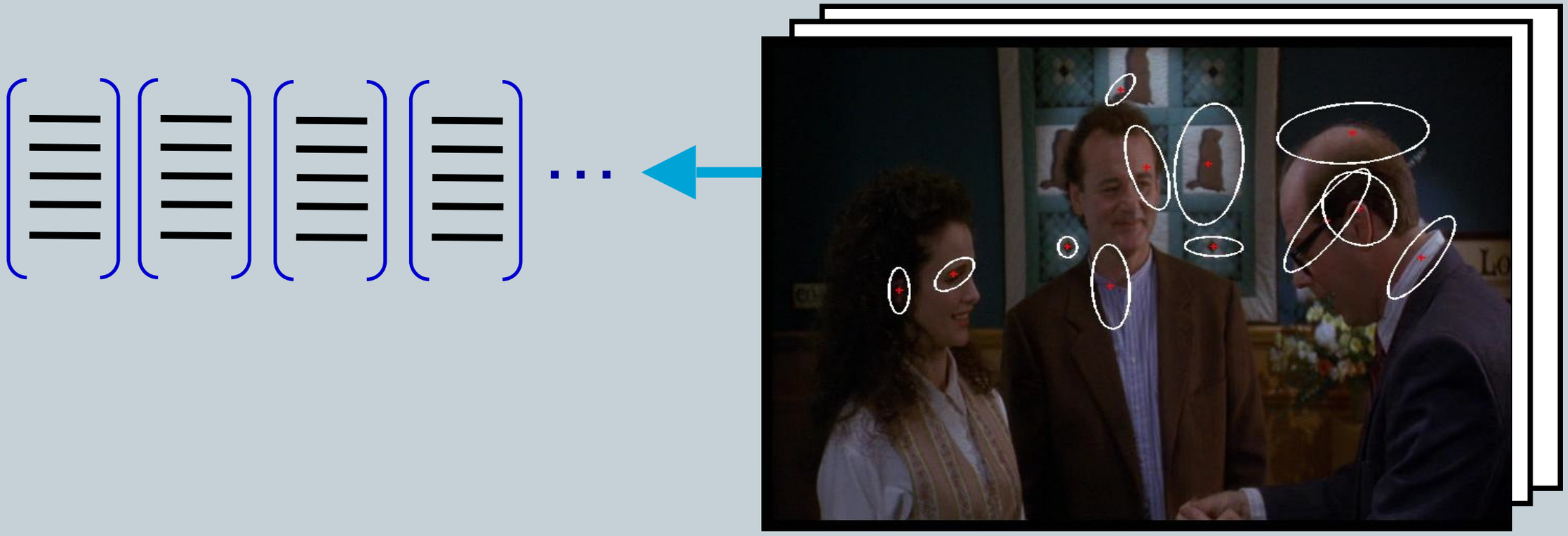
Normalize patch



Compute descriptor

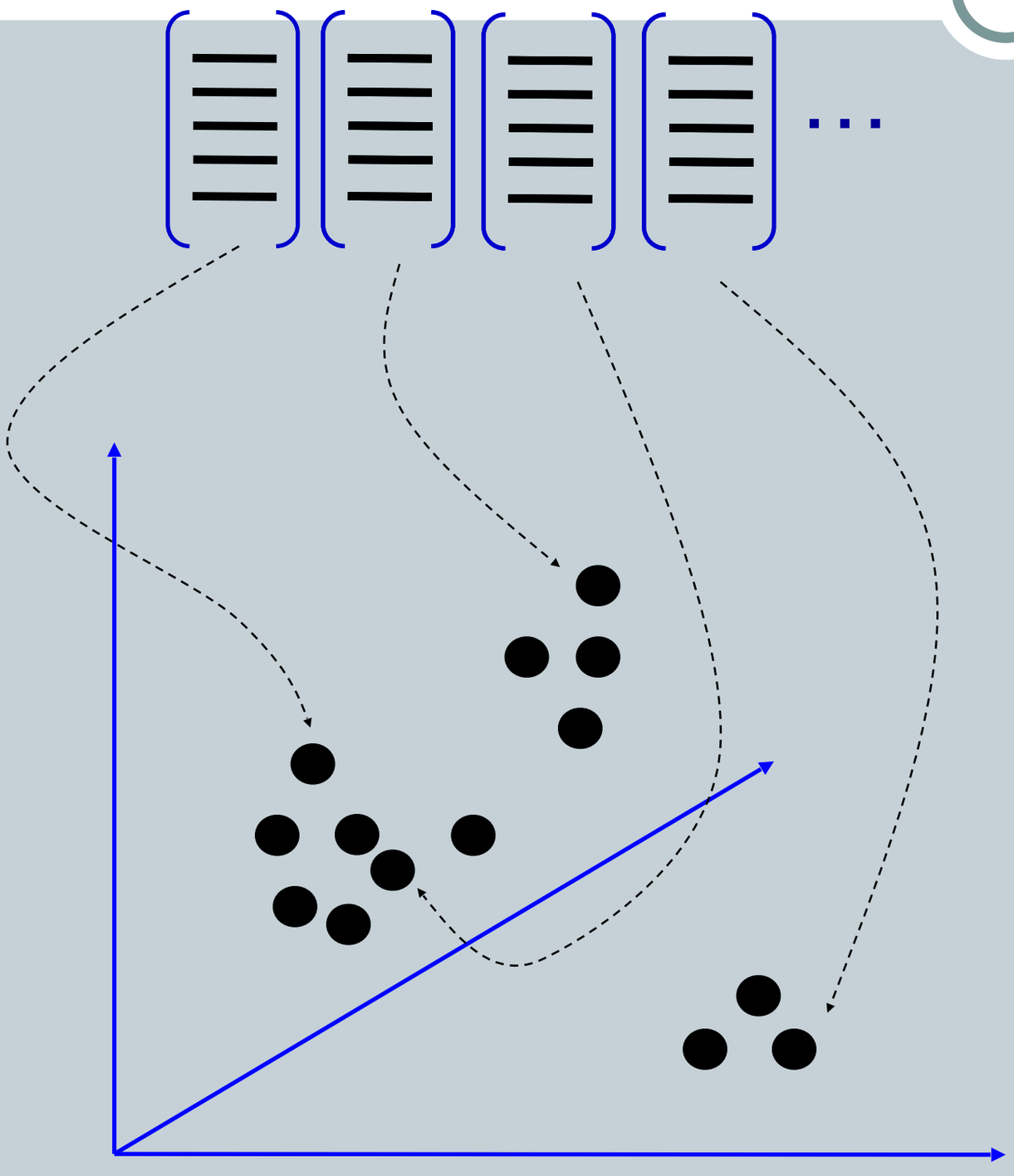
Slide credit: Josef Sivic

1. Feature extraction



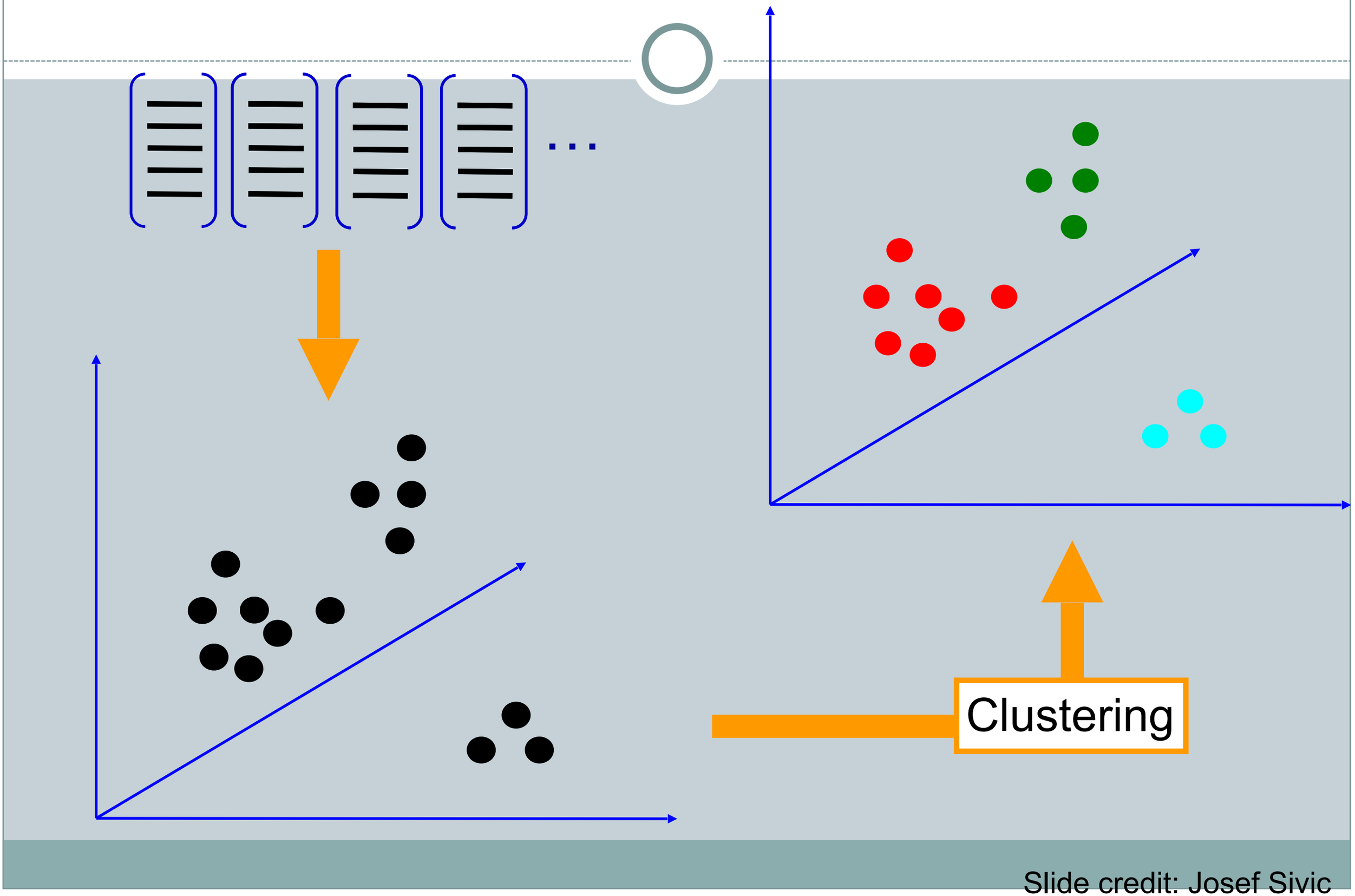
Slide credit: Josef Sivic

2. Learning the visual vocabulary



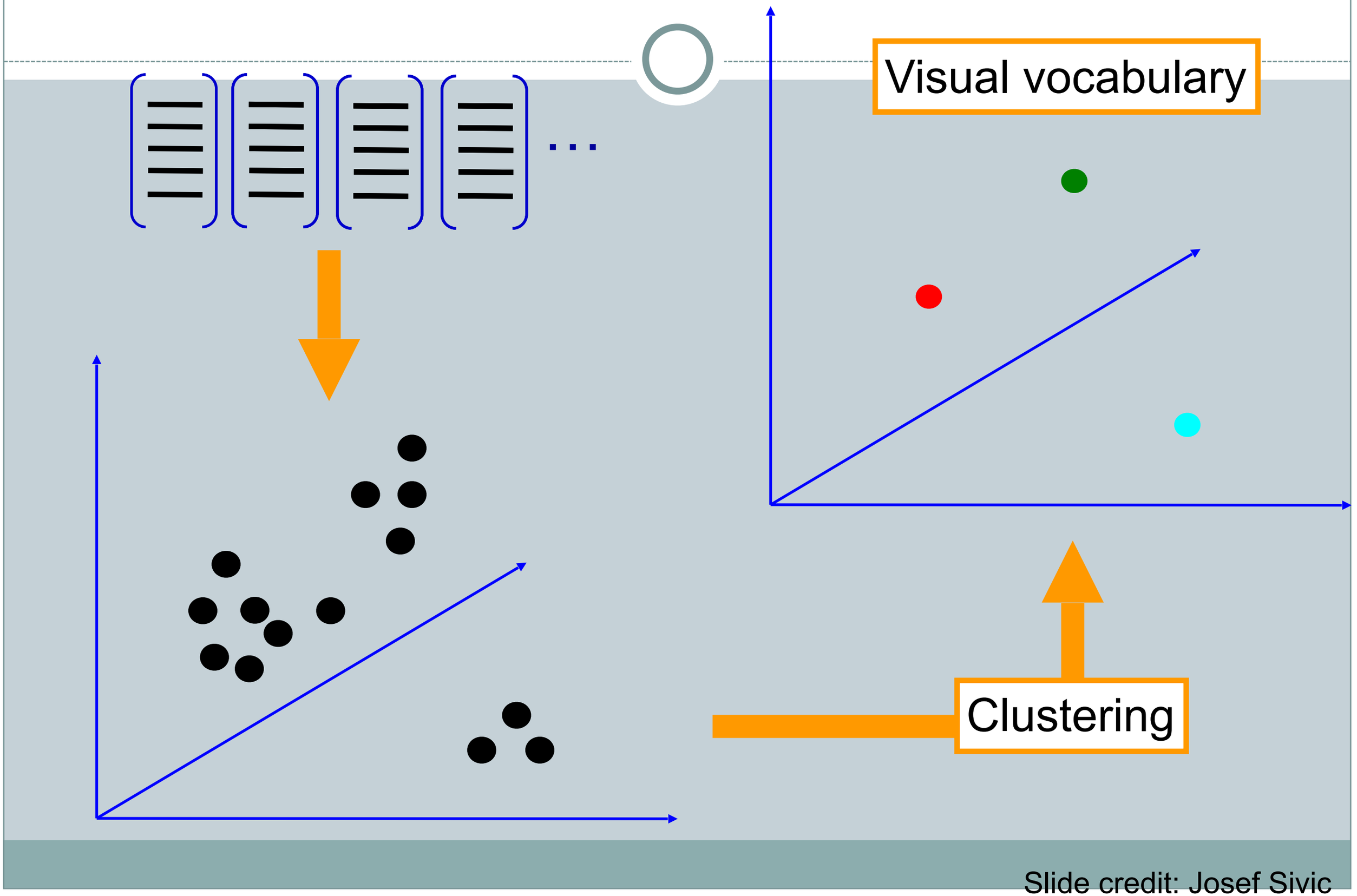
Slide credit: Josef Sivic

2. Learning the visual vocabulary



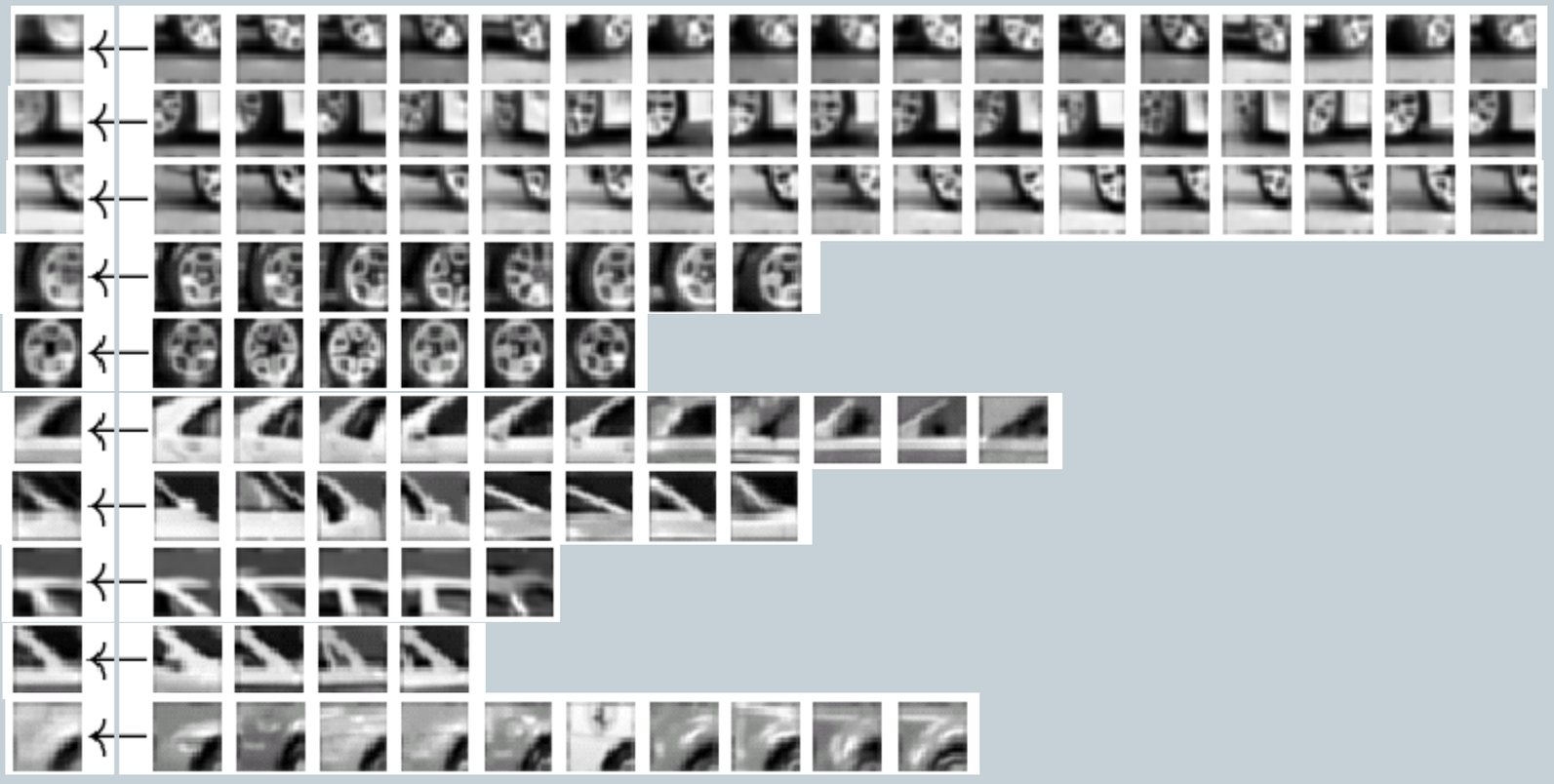
Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

Example codebook (visual vocabulary)

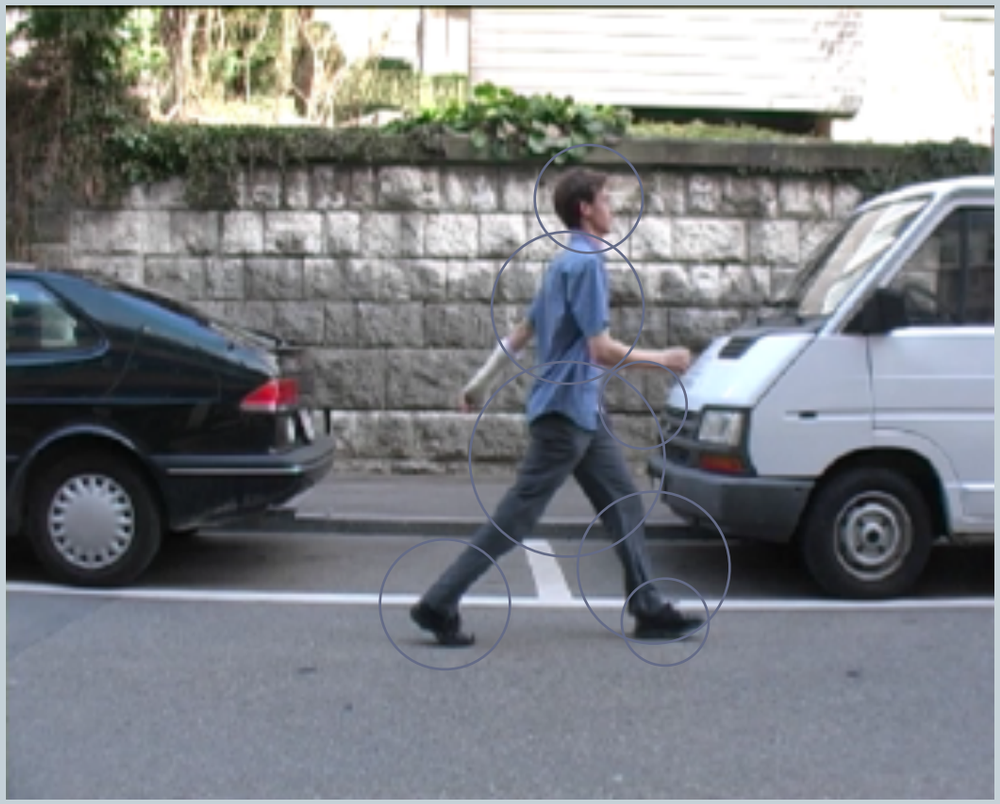


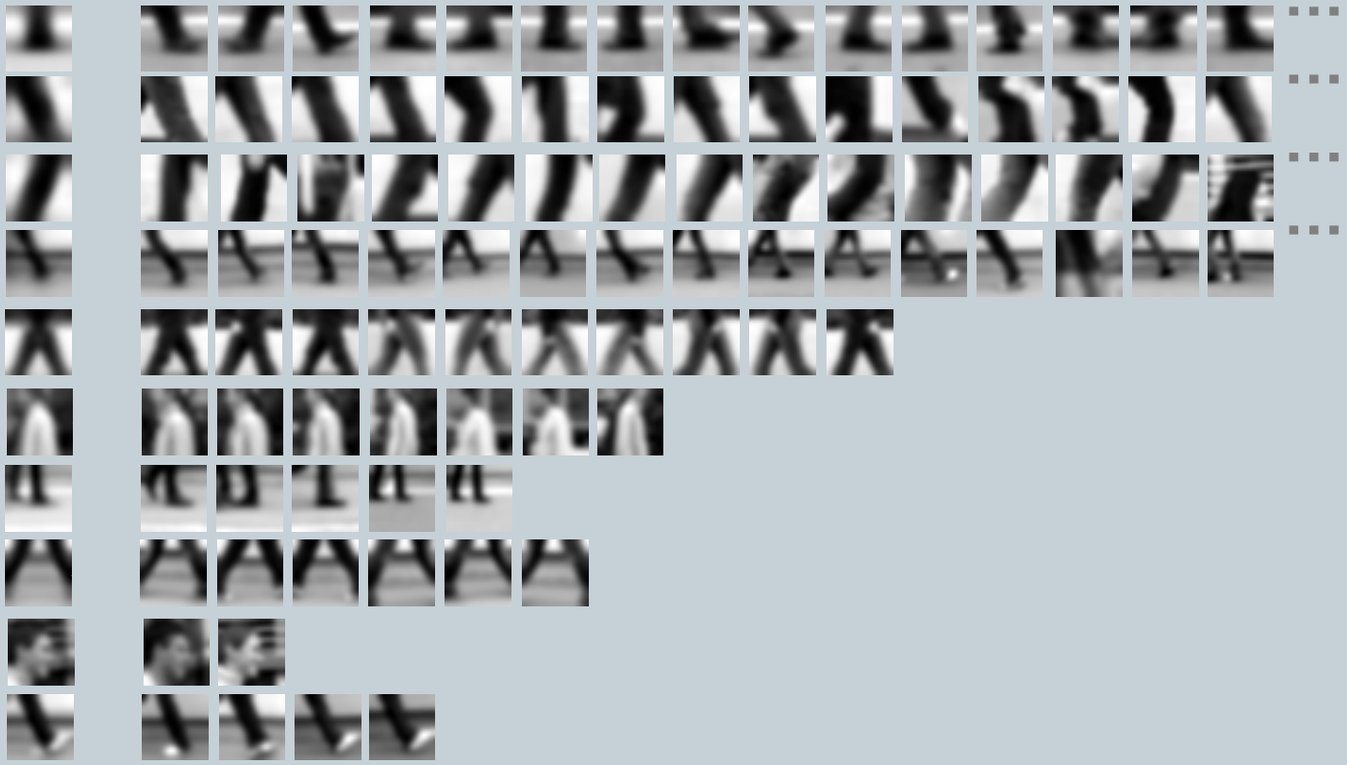
...

Appearance codebook

Source: B. Leibe

Another codebook





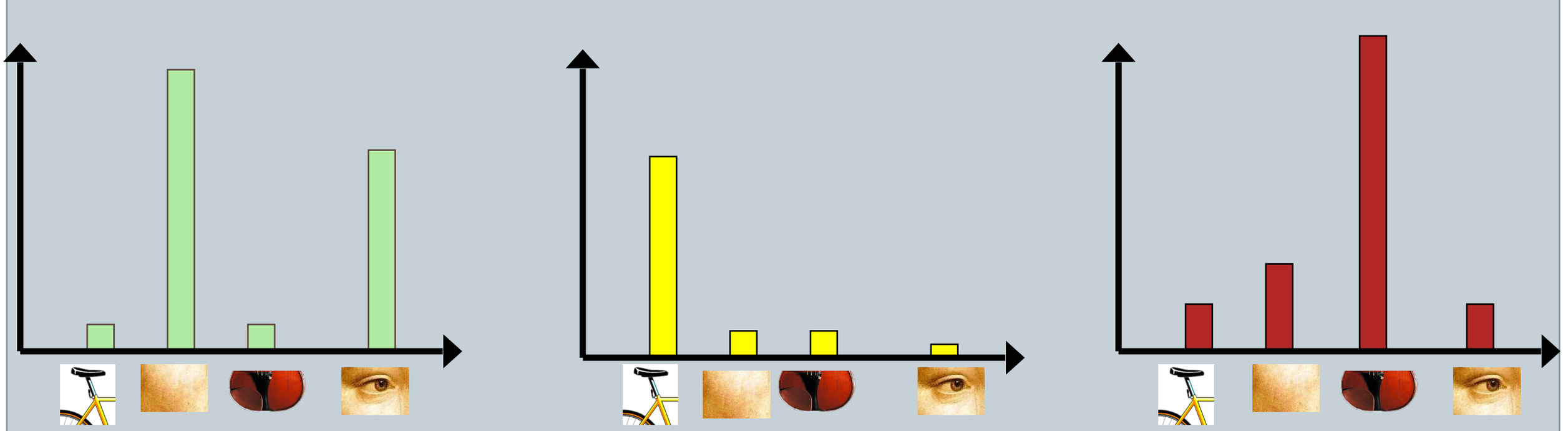
...

Appearance codebook

Source: B. Leibe

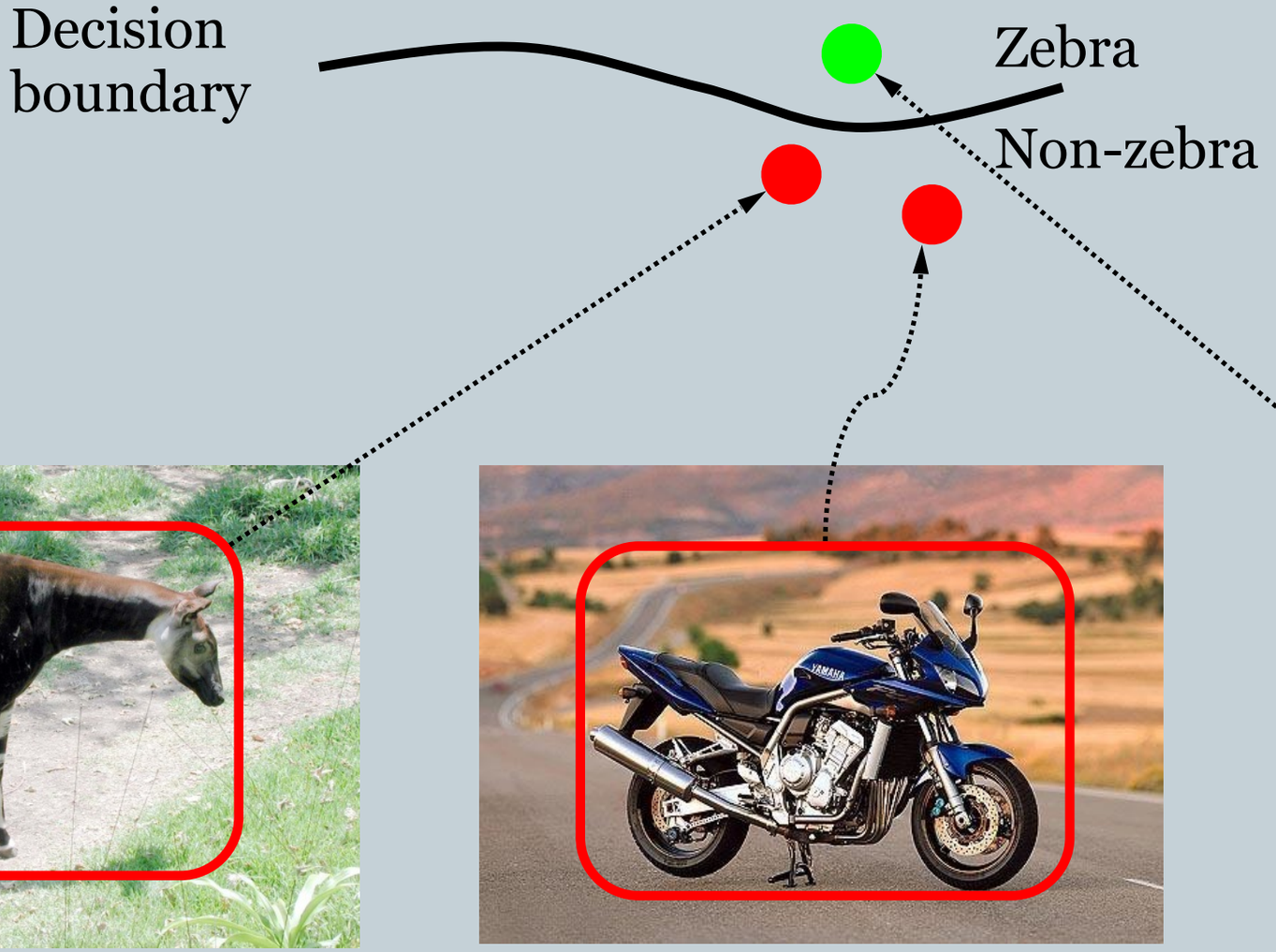
Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



Discriminative methods

- Learn a decision rule (classifier) assigning bag-of-features representations of images to different classes

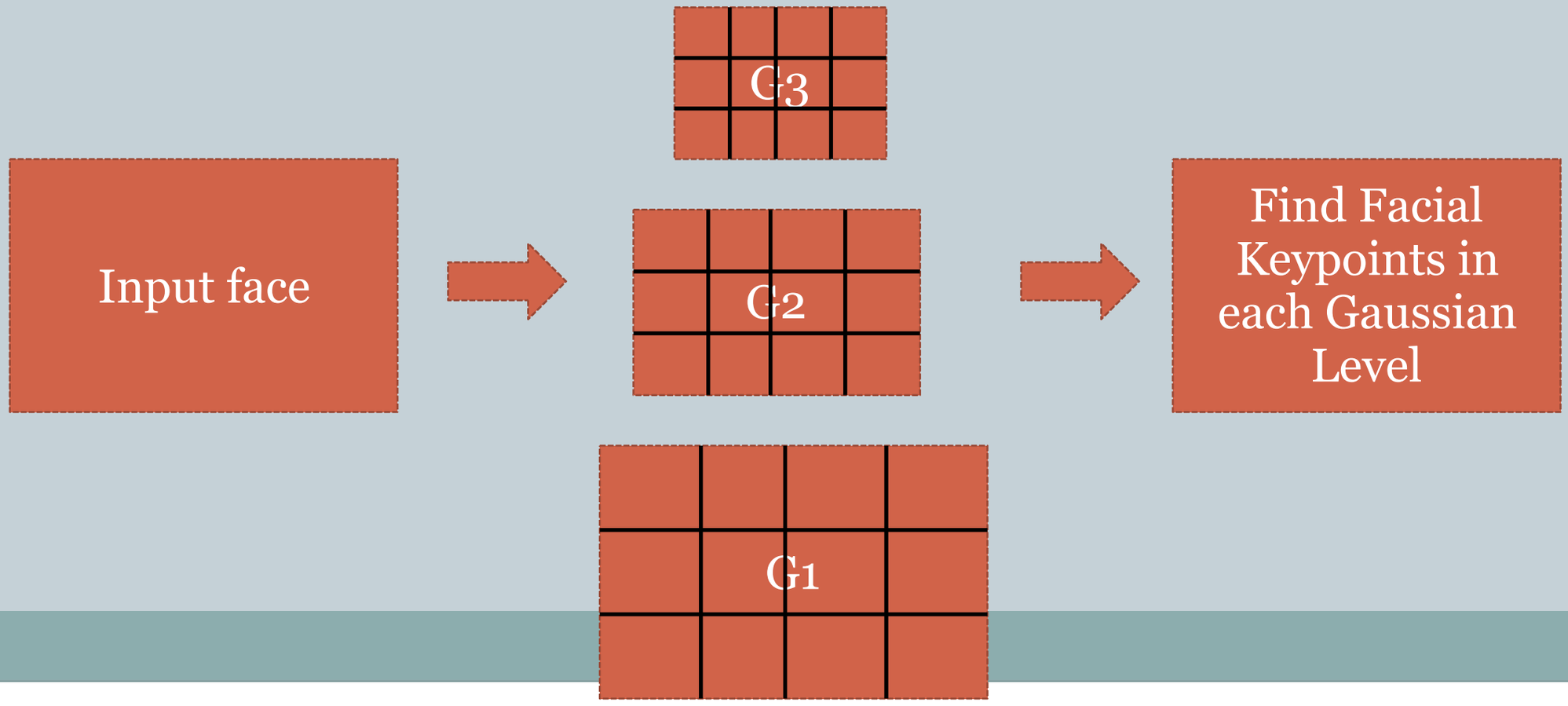


How we can apply BoW model in face or object recognition?

- Let us develop an algorithm for face recognition that uses some of the concept learnt so far
- Let us assume we have training sets and testing sets with no-overlap
- Database 1 for training and Database 2 for testing
- We can do two experiments:
  - Database 1 for training and Database 2 for testing
  - Database 2 for training and Database 1 for testing

Training: Step 1 (Database 1)

- Perform face detection and pre-processing and then compute three levels of Gaussian Pyramid
- In G1, G2, and G3, find the facial keypoints – you have to be imaginative in this step



Training: Step 2 (Database 1)

- For each keypoint, take n x n window and compute SSD
- Create Bag of words (features) for G1, G2, and G3

BoW for G1

BoW for G2

BoW for G3

Training: Step 3 (Database 2)

- Split Database 2 into training set and testing set
- Database 2 training set is used to train the classifier
- Database 2 testing set is used for performance evaluation
- Database 2 training set is processed as in Step 1 (Database 1) and them
  1. Extract features from Database 2 training set
  2. Represent images by frequencies of “visual words”

Training: Step 4 (Database 2)

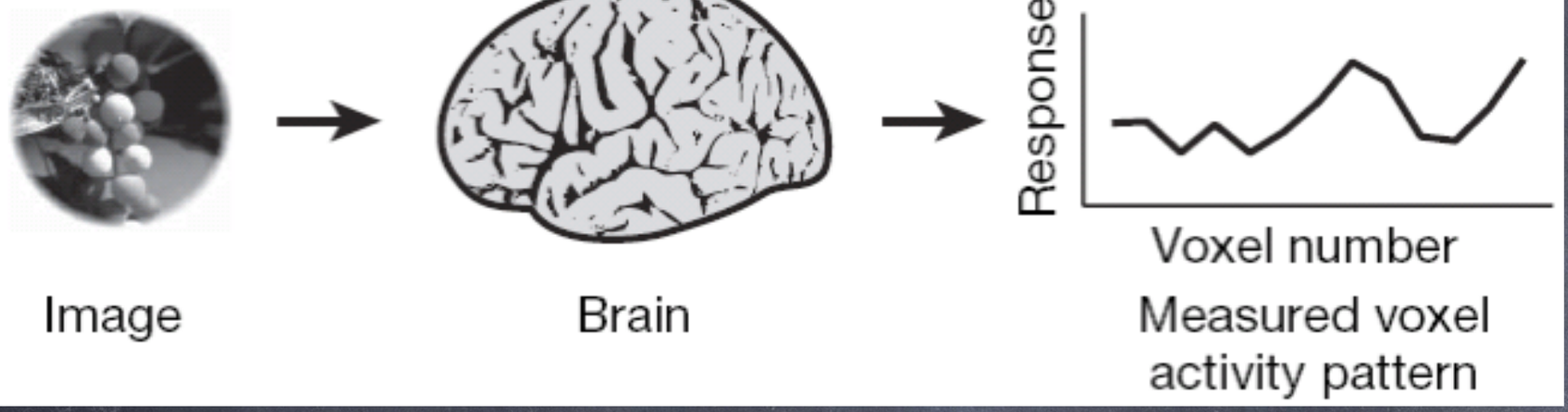
- Once you compute the histograms (frequencies of “visual words”) of all training images, train the classifiers for either 2-class classification (verification) or n-class classification (identification)
- Note that we will have to do this process for reach Gaussian levels; so, we will have three features/scores/decisions
- Combination can be done in many ways,
  - Concatenate three histograms (feature level fusion)
  - Apply Sum rule using the scores obtained
  - Majority voting at decision level – but what about n-class classification?

Testing Step

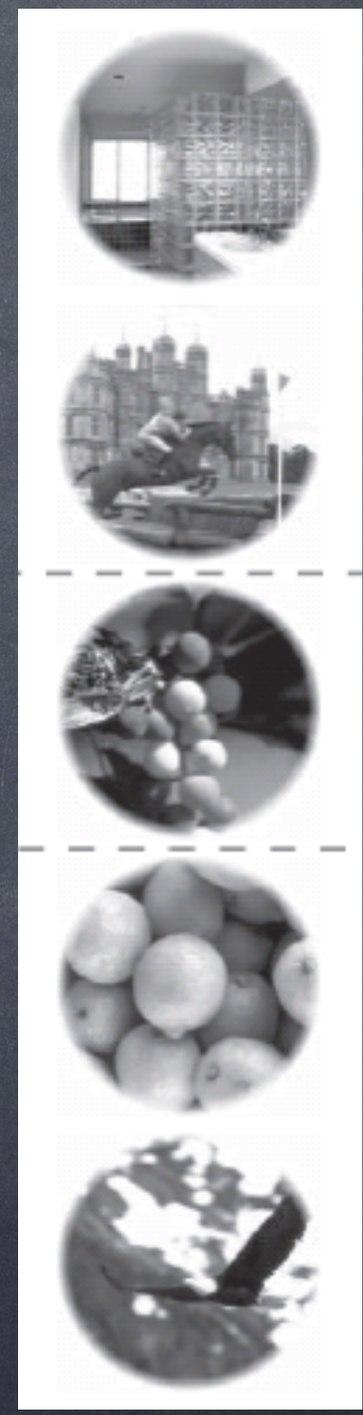
- Using training on Database 1, we have created three BoW
- Using training on Database 2 (train set), we have learnt a classifier
- Now using test set of Database 2, perform testing
  - In verification mode, compute genuine match scores and impostor match scores to compute ROC
  - In identification mode, fix your gallery for each subject and then compute CMC
- Perform cross validation with Database 2 – train and test splits
- Repeat the same experiment when Database 2 is used to create BoW and Database 1 is used to train classifier and testing

Gabor Features

Let us now look at "Human Visual System"



?



Identifying natural images from human brain activity

Kay, K.N., Naselaris, T., Prenger, R.J., & Gallant, J.L. (2008). Identifying natural images from human brain activity. Nature, 452, 352-355.

(1) Measure brain activity for an image

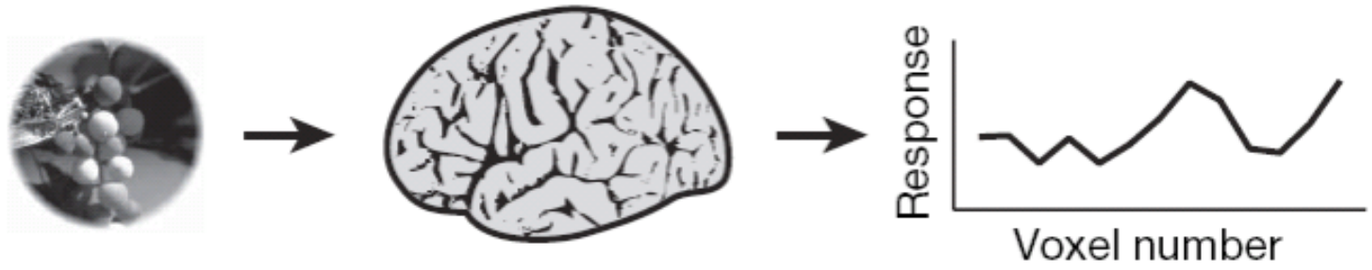
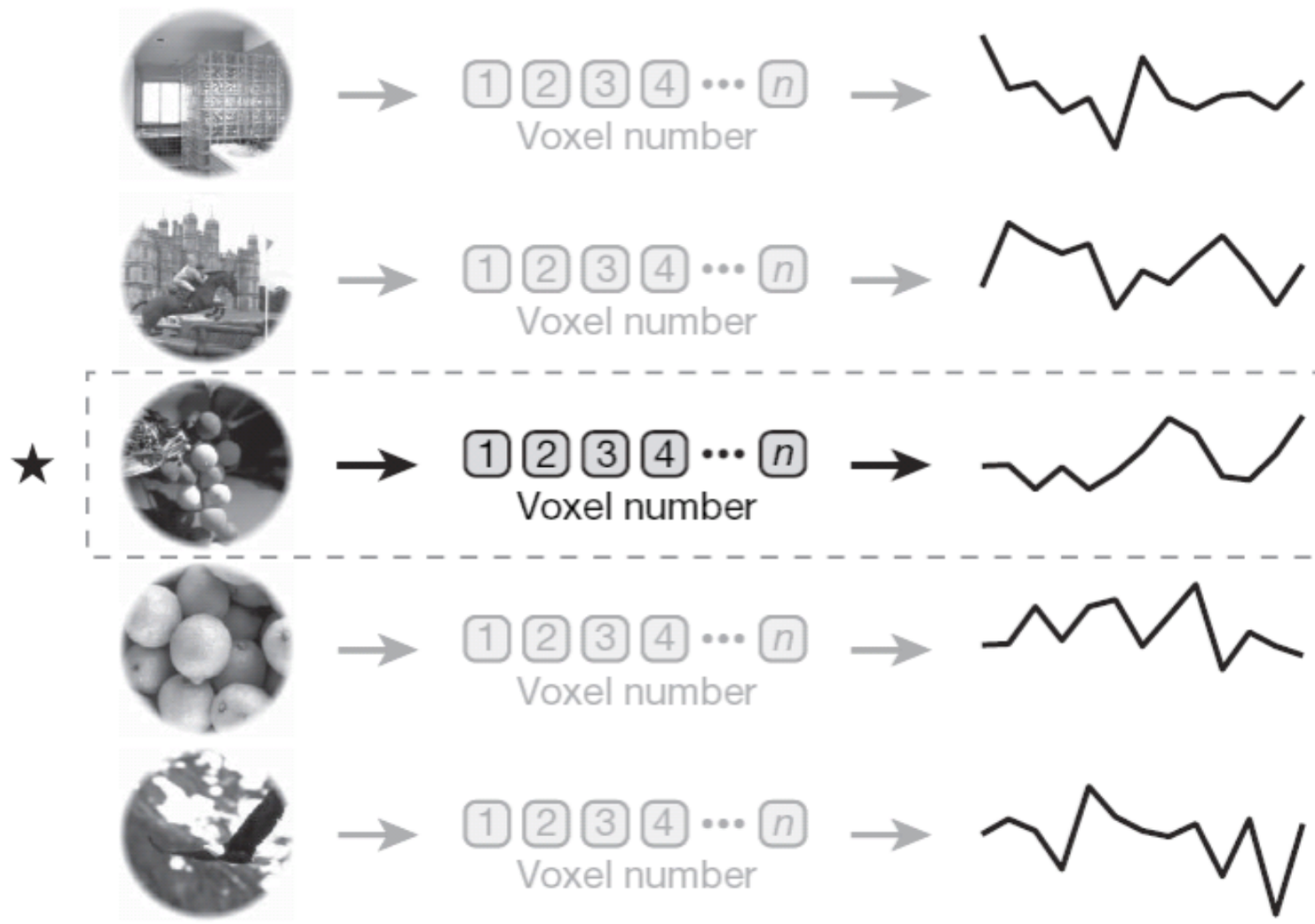


Image Brain Measured voxel activity pattern

(2) Predict brain activity for a set of images using receptive-field models

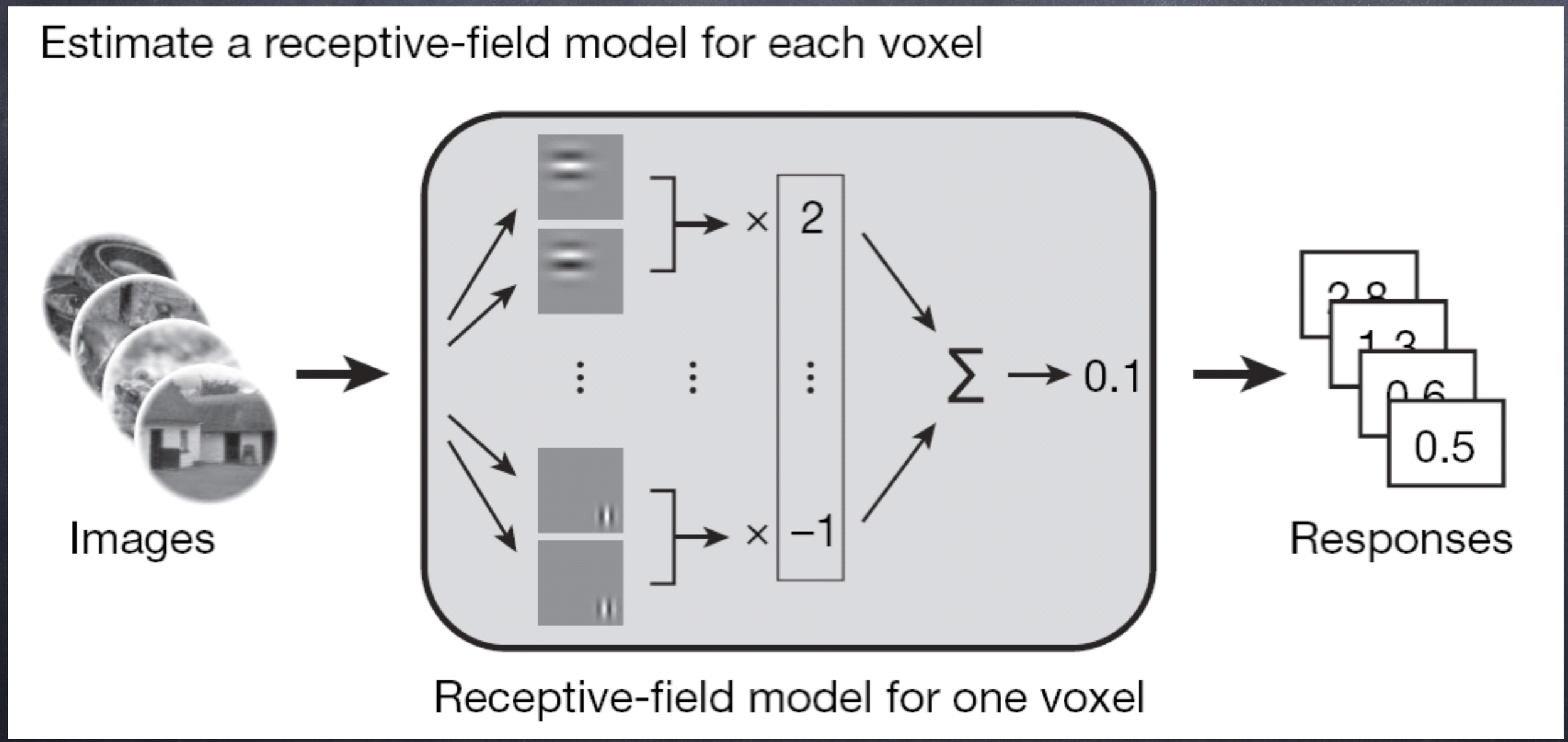


Set of images Receptive-field models for multiple voxels Predicted voxel activity patterns

(3) Select the image (★) whose predicted brain activity is most similar to the measured brain activity

Voxel Activity Model

Goal: to predict the image seen by the observer out of a large collection of possible images. And to do this for new images:



Kay, K.N., Naselaris, T., Prenger, R.J., & Gallant, J.L. (2008). Identifying natural images from human brain activity. Nature, 452, 352-355.

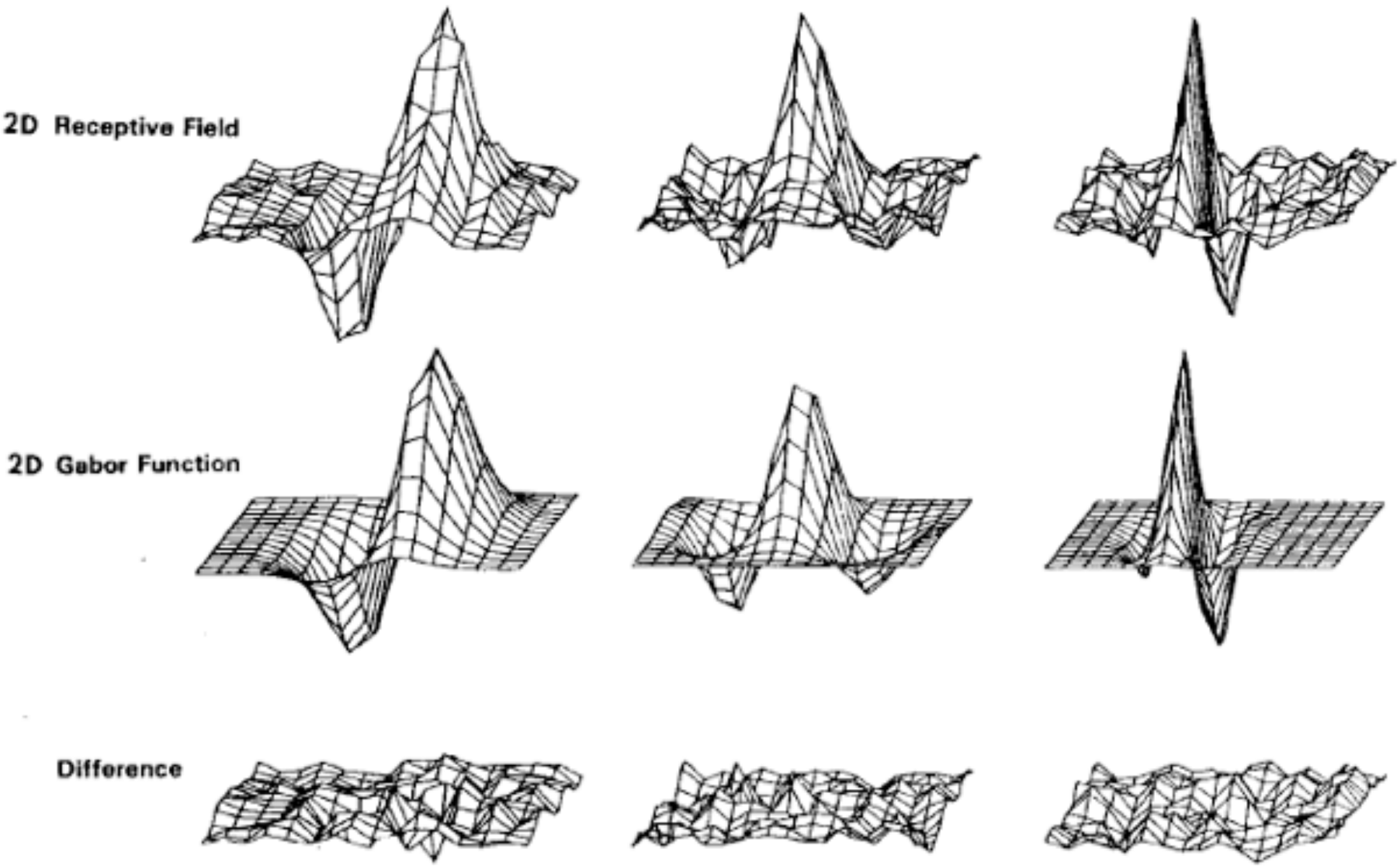


Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of the fit, indistinguishable from random error in the Chi-squared sense for 97 percent of the cells studied.

John Daugman, 1988

Gabor Filter

$$h(x,y) = s(x,y)g(x,y)$$

$$s(x,y) = e^{-j2\pi(u_0x+v_0y)}$$

$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})}$$

$$h(x,y) = e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})} e^{-j2\pi(u_0x+v_0y)}$$

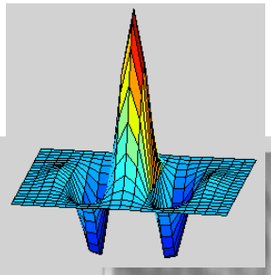
$$\Rightarrow H(u,v) = 2\pi\sigma_x\sigma_y[e^{-2\pi^2[(u-u_0)^2\sigma_x^2+(v-v_0)^2\sigma_y^2]}]$$

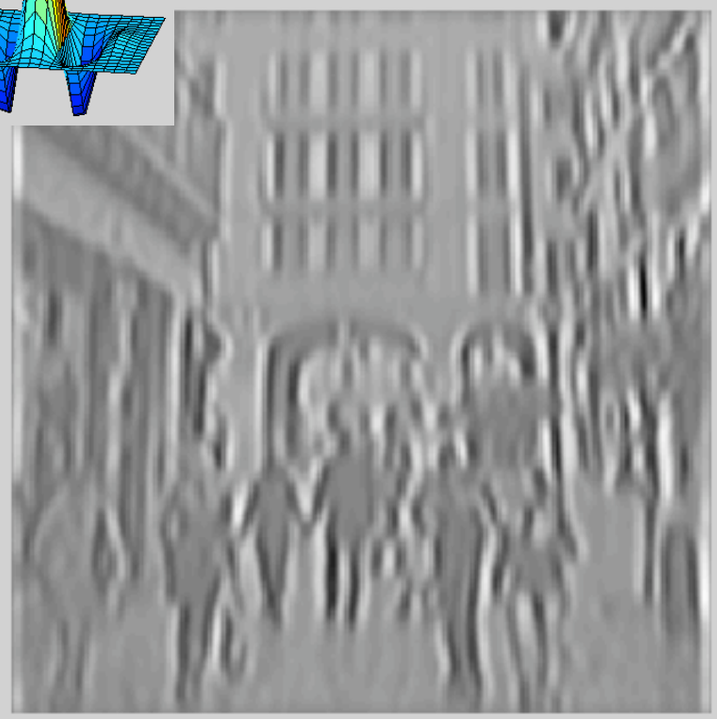
Gabor Filter

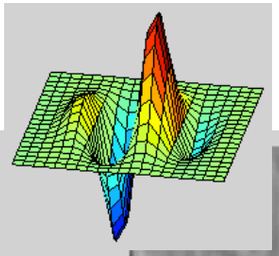
$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

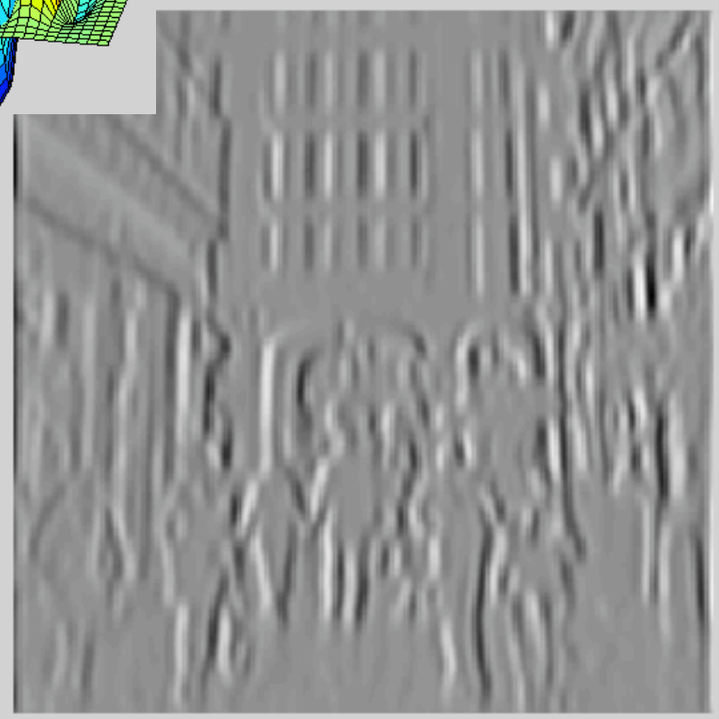
$$x' = x\cos\theta + y\sin\theta$$

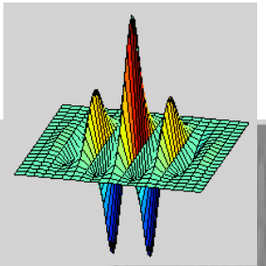
$$y' = -x\sin\theta + y\cos\theta$$

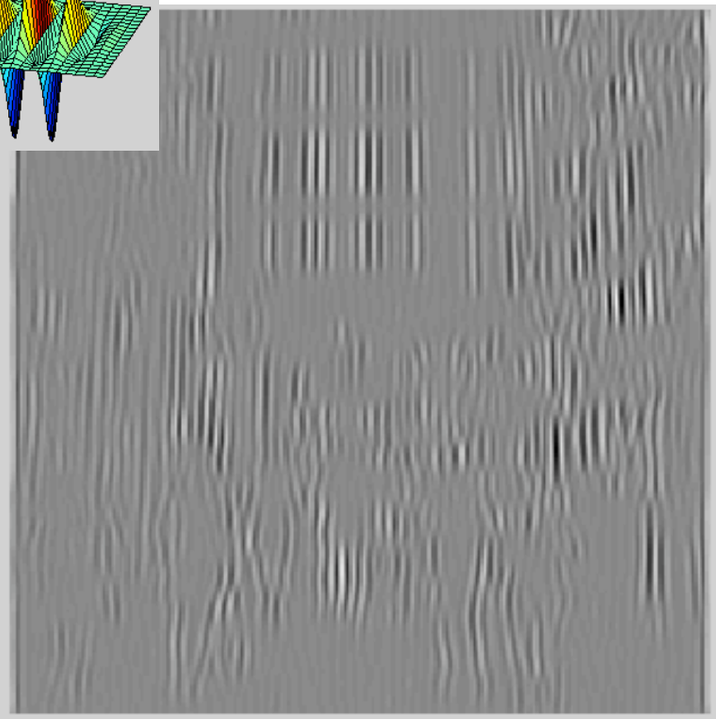


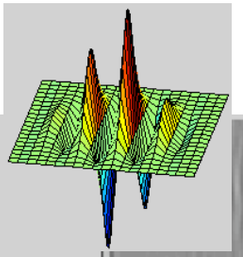


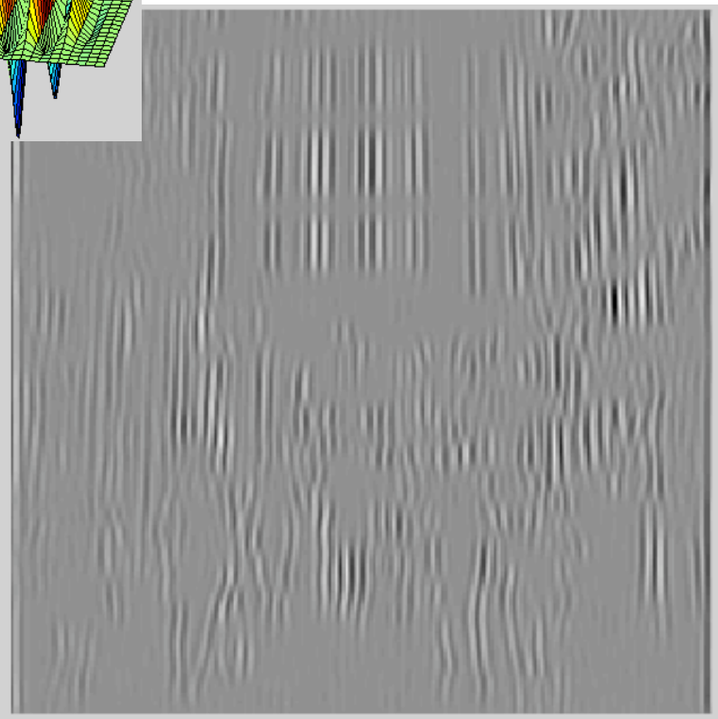






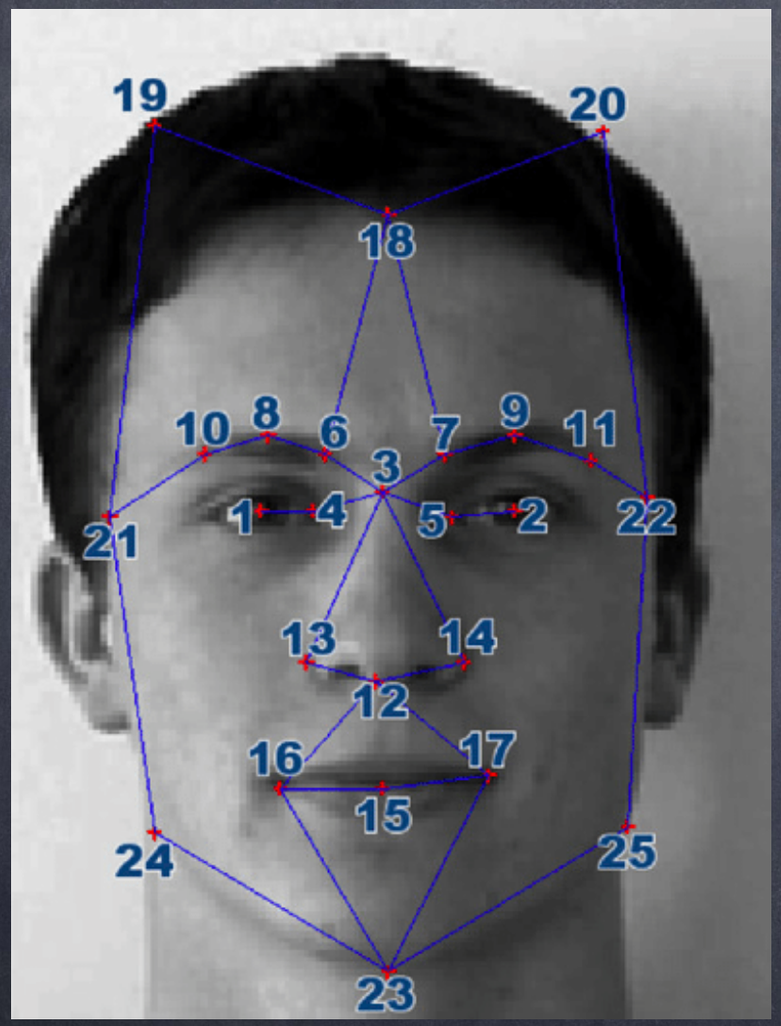


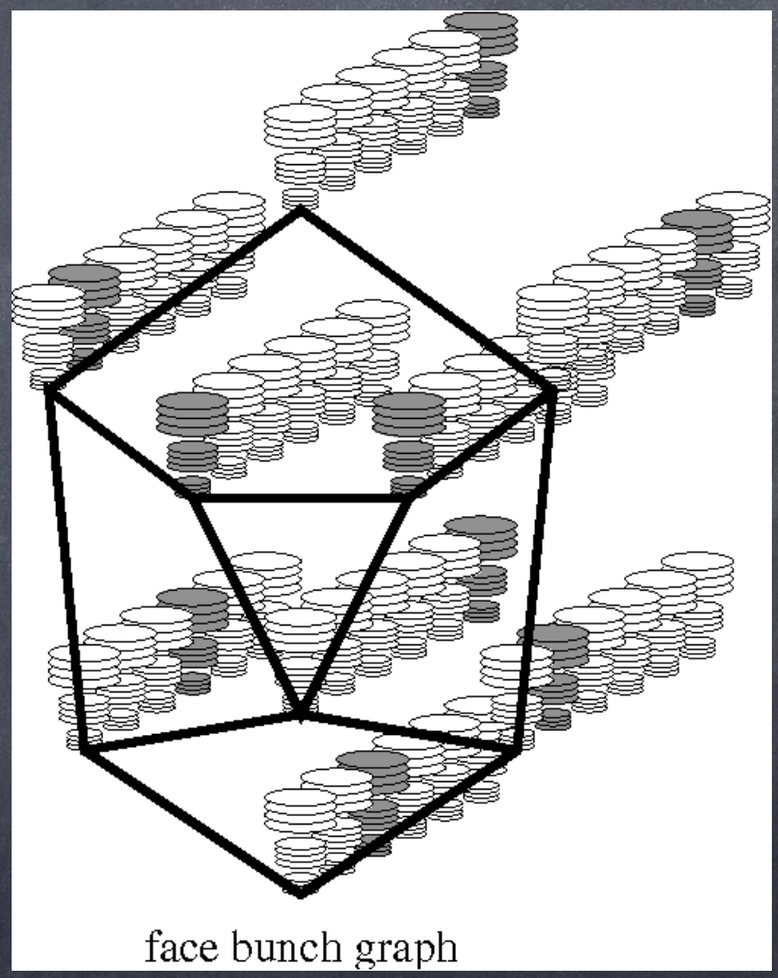




Can we build object/face recognition using Gabor Filter?

Can we build object/face recognition using Gabor Filter?

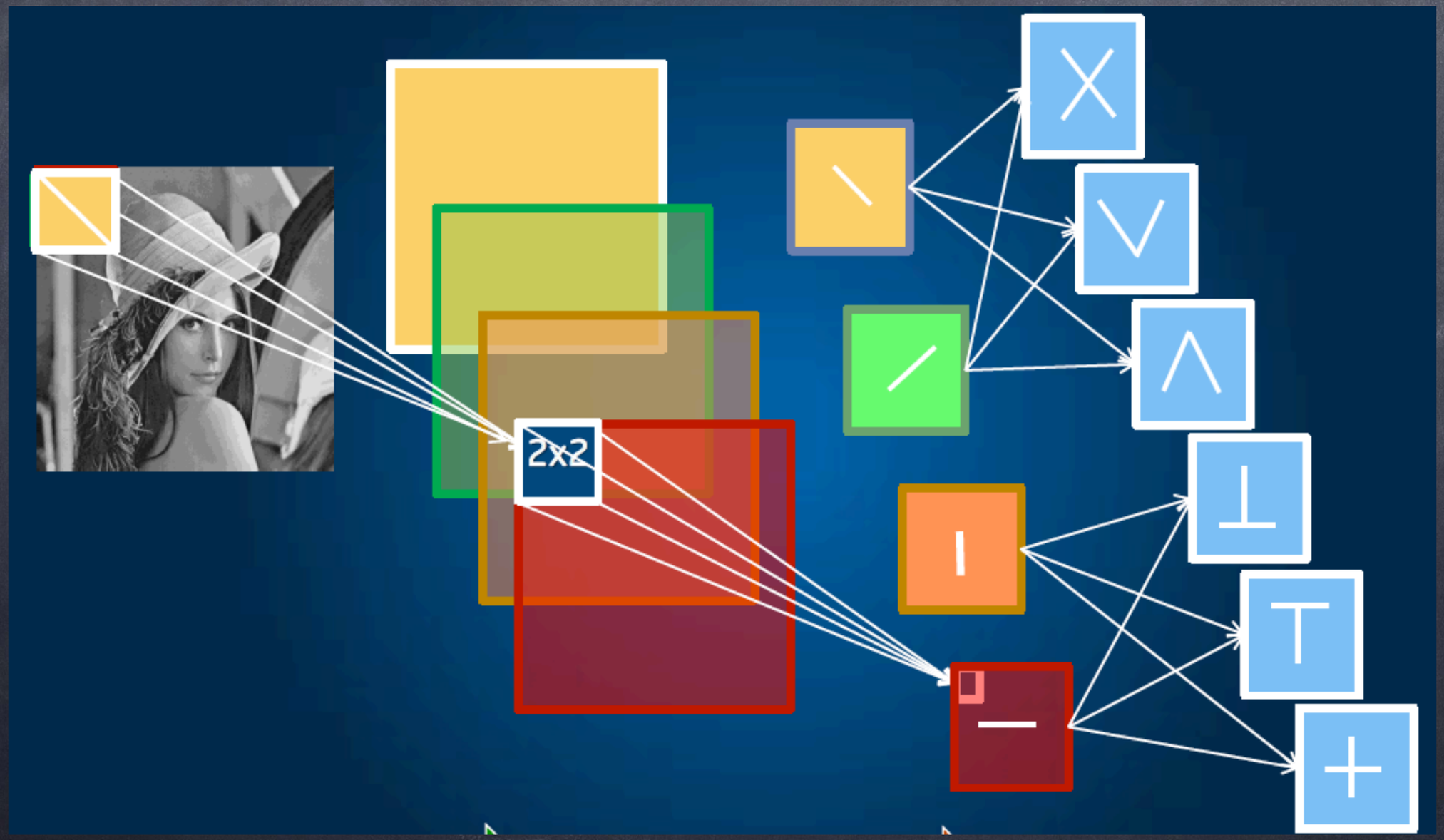



face bunch graph

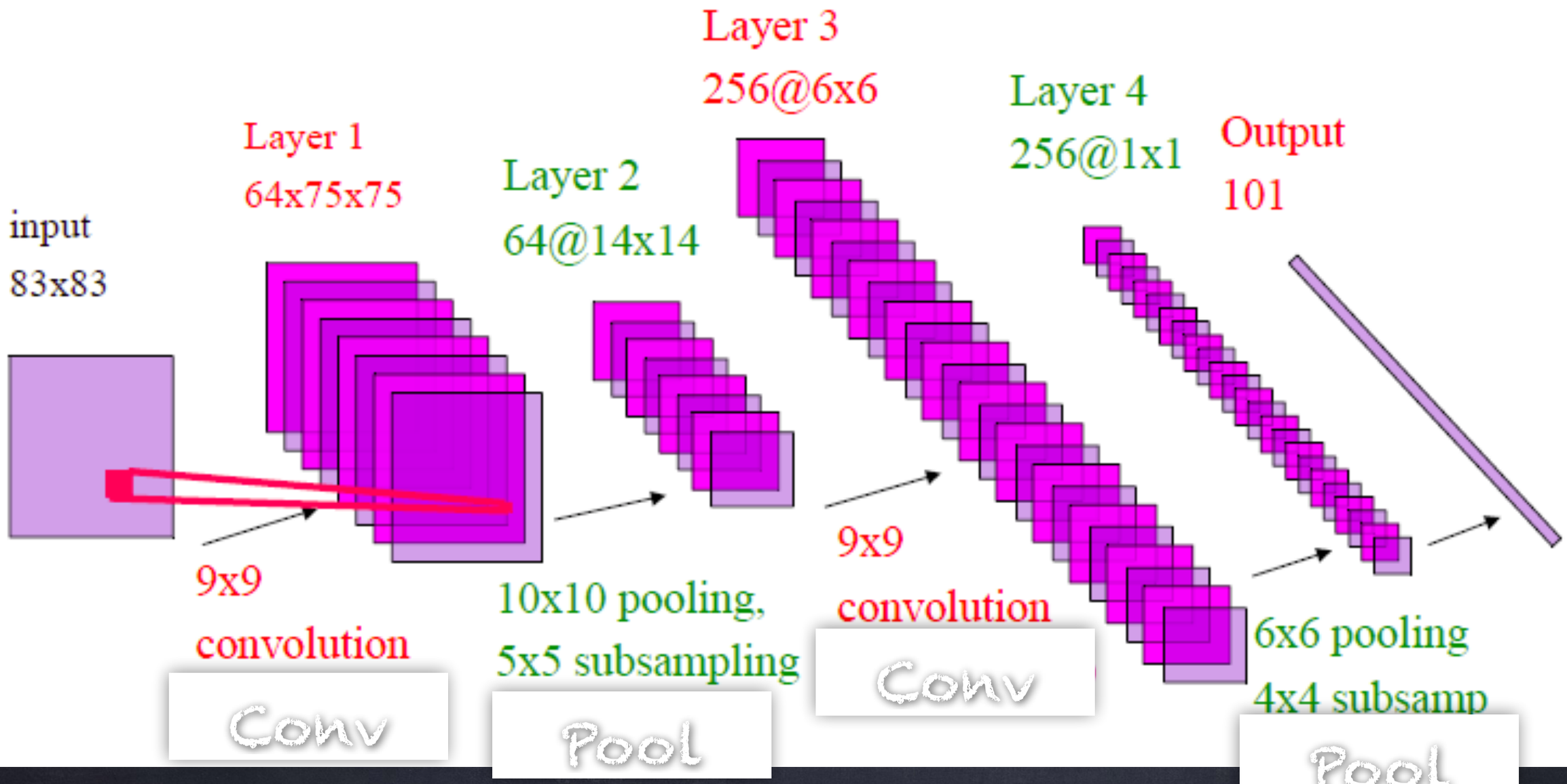
Object Recognition

Using CNNs

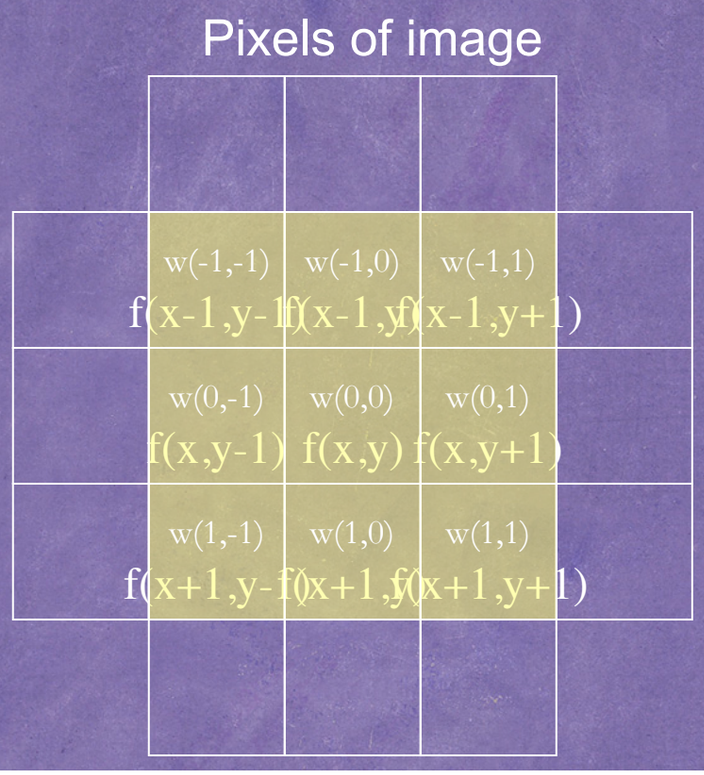
DNN in Visual Understanding - CNN



Convolutional Neural Network



Recall "filtering"



The result is the sum of products of the mask coefficients with the corresponding pixels directly under the mask

Mask coefficients

w(-1,-1)	w(-1,0)	w(-1,1)
w(0,-1)	w(0,0)	w(0,1)

$$f(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + w(-1,1)f(x-1,y+1) + w(0,-1)f(x,y-1) + w(0,0)f(x,y) + w(0,1)f(x,y+1) + w(1,-1)f(x+1,y-1) + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1)$$

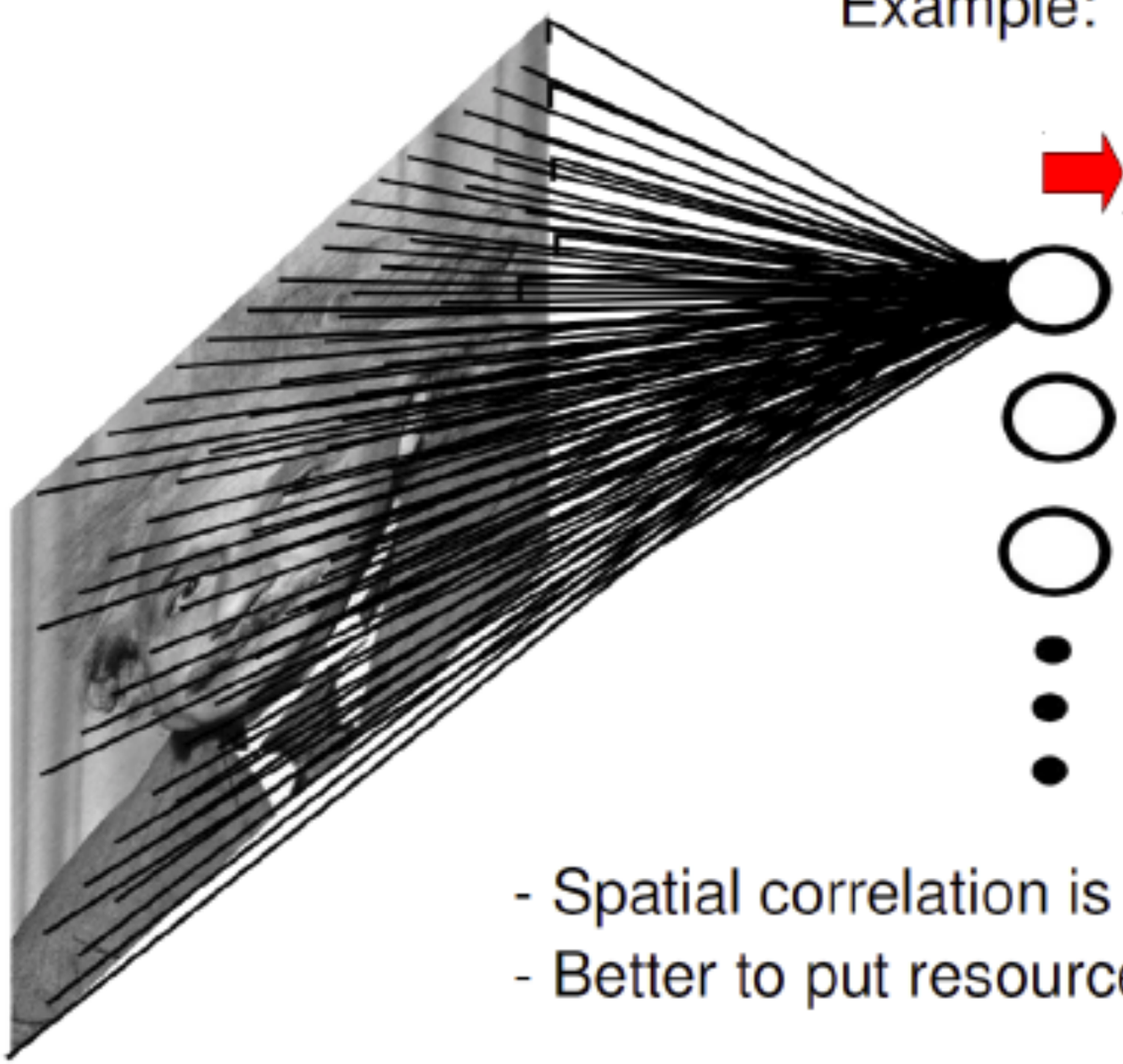
CNN

FULLY CONNECTED NEURAL NET

Example: 1000x1000 image

1M hidden units

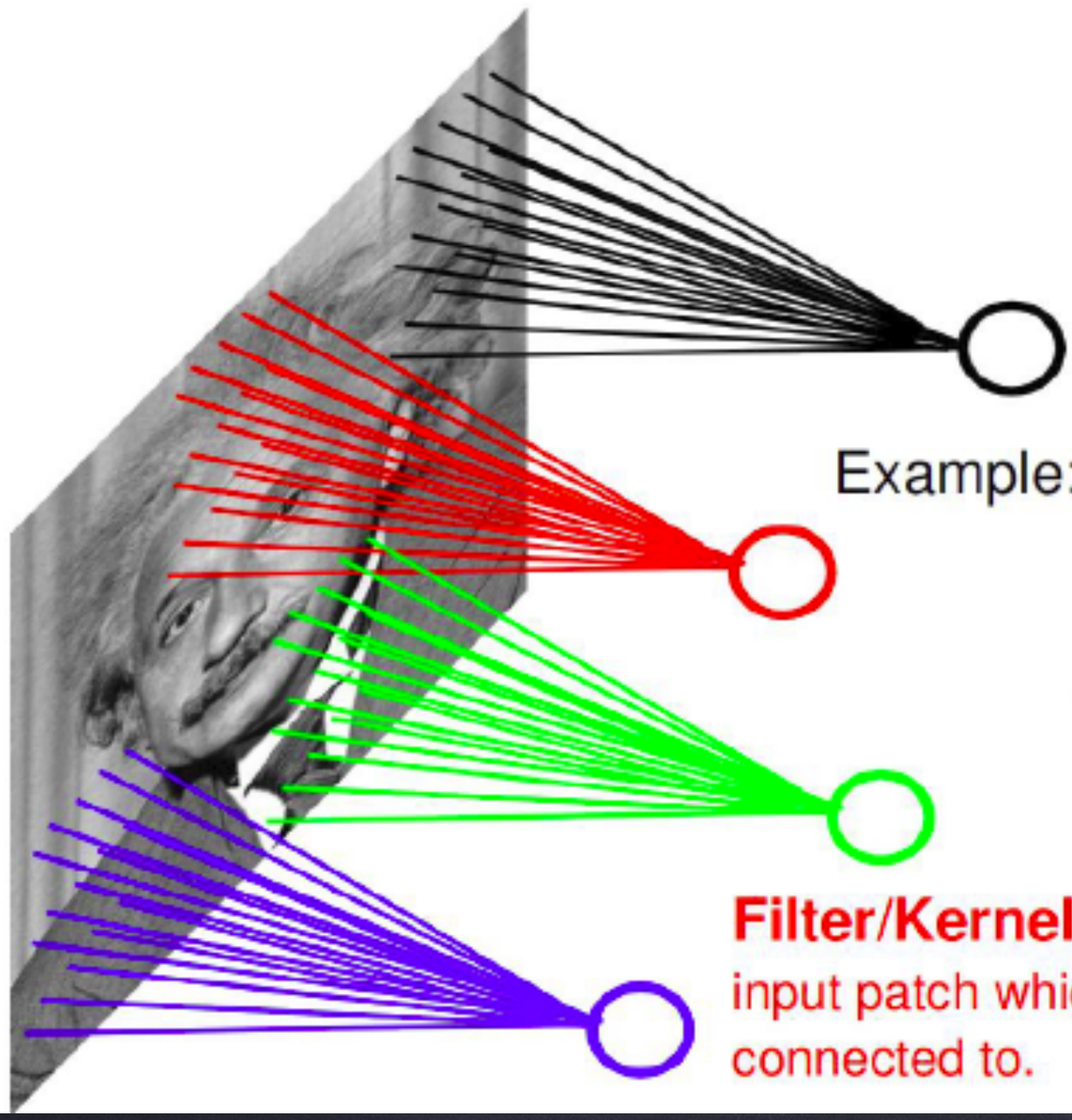
10^12 parameters!!!



- Spatial correlation is local
- Better to put resources elsewhere!

CNN

LOCALLY CONNECTED NEURAL NET

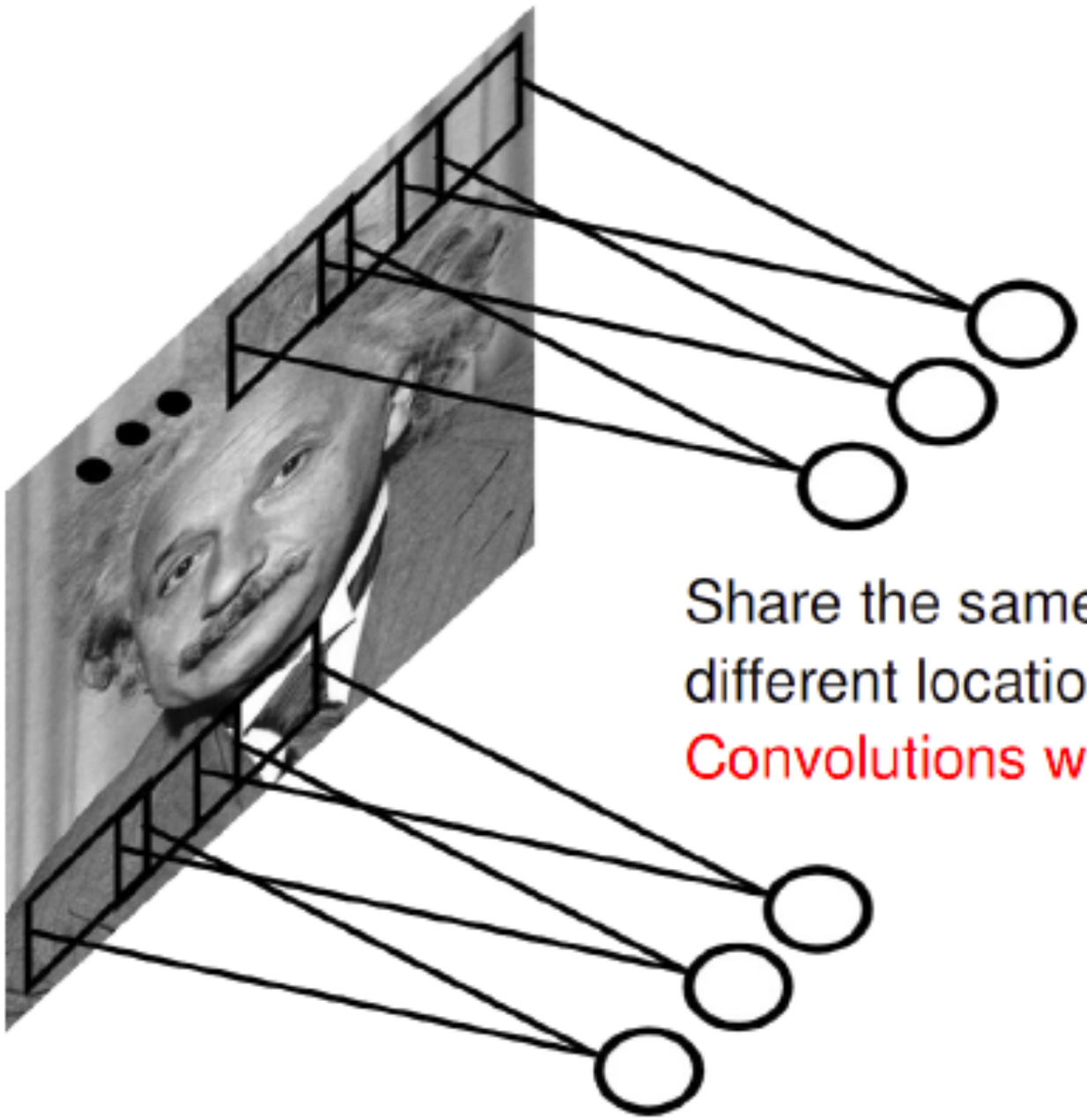


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Filter/Kernel/Receptive field:
input patch which the hidden unit is connected to.

CNN

CONVOLUTIONAL NET

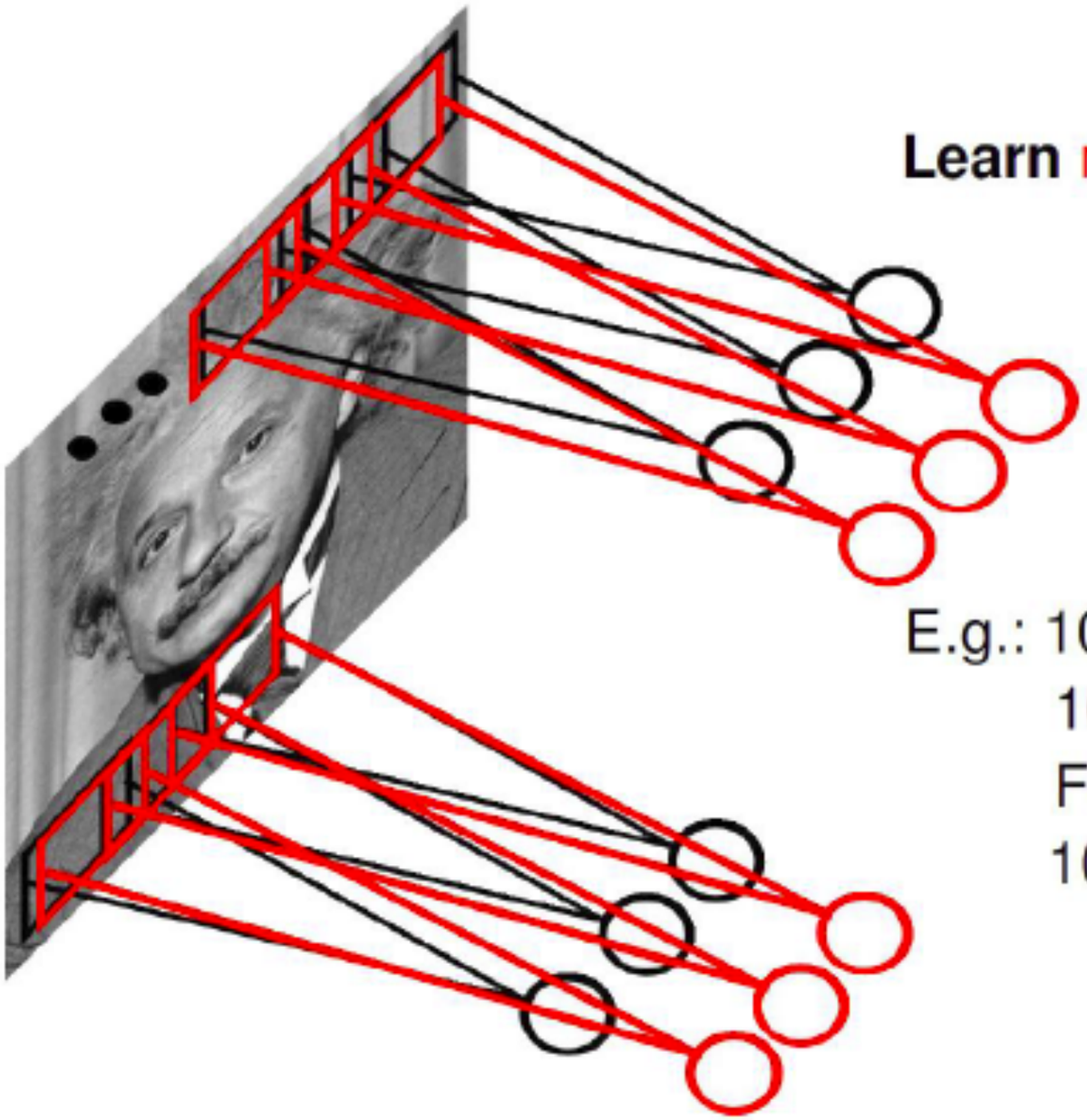


Share the same parameters across different locations:

Convolutions with learned kernels

CNN

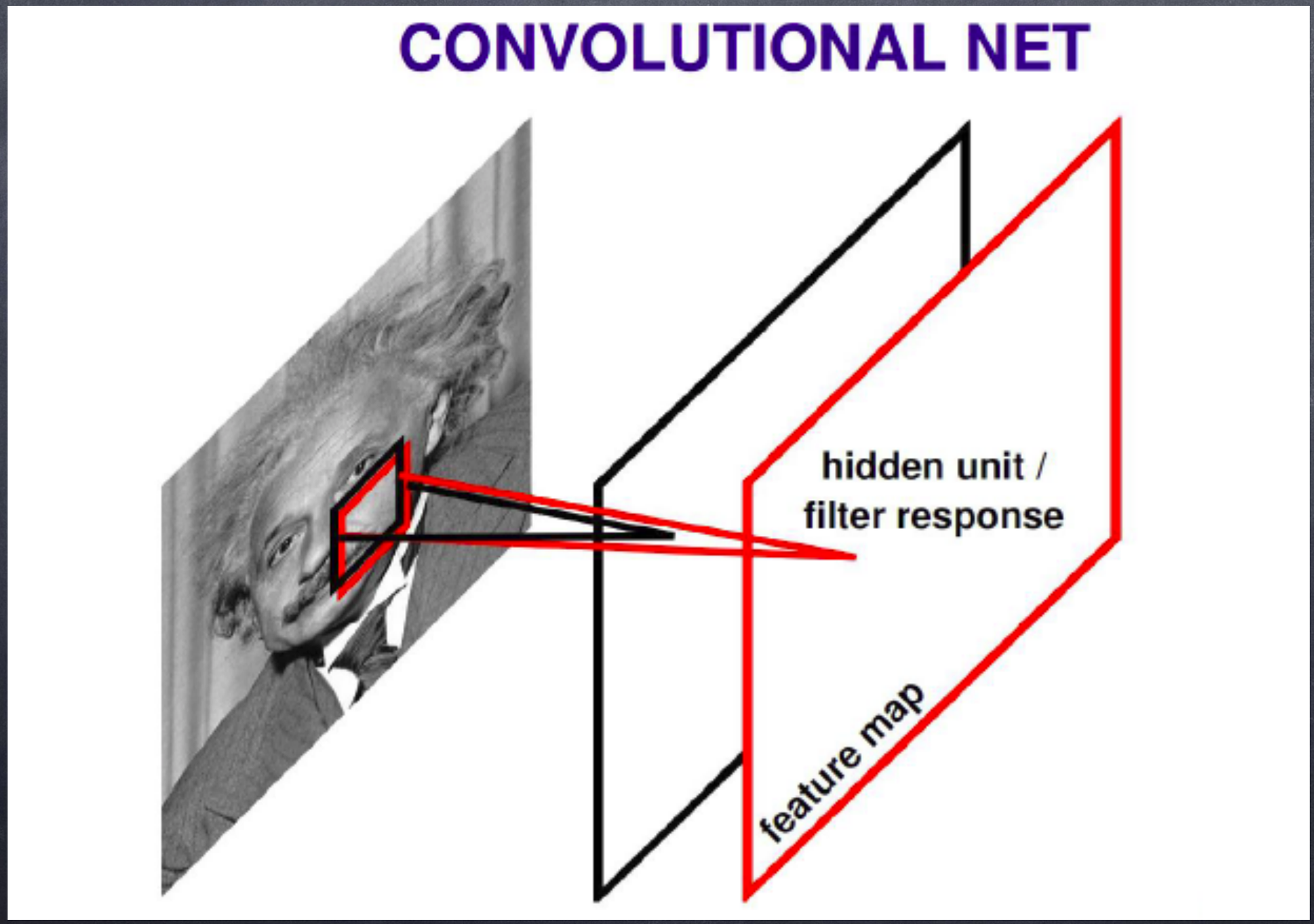
CONVOLUTIONAL NET

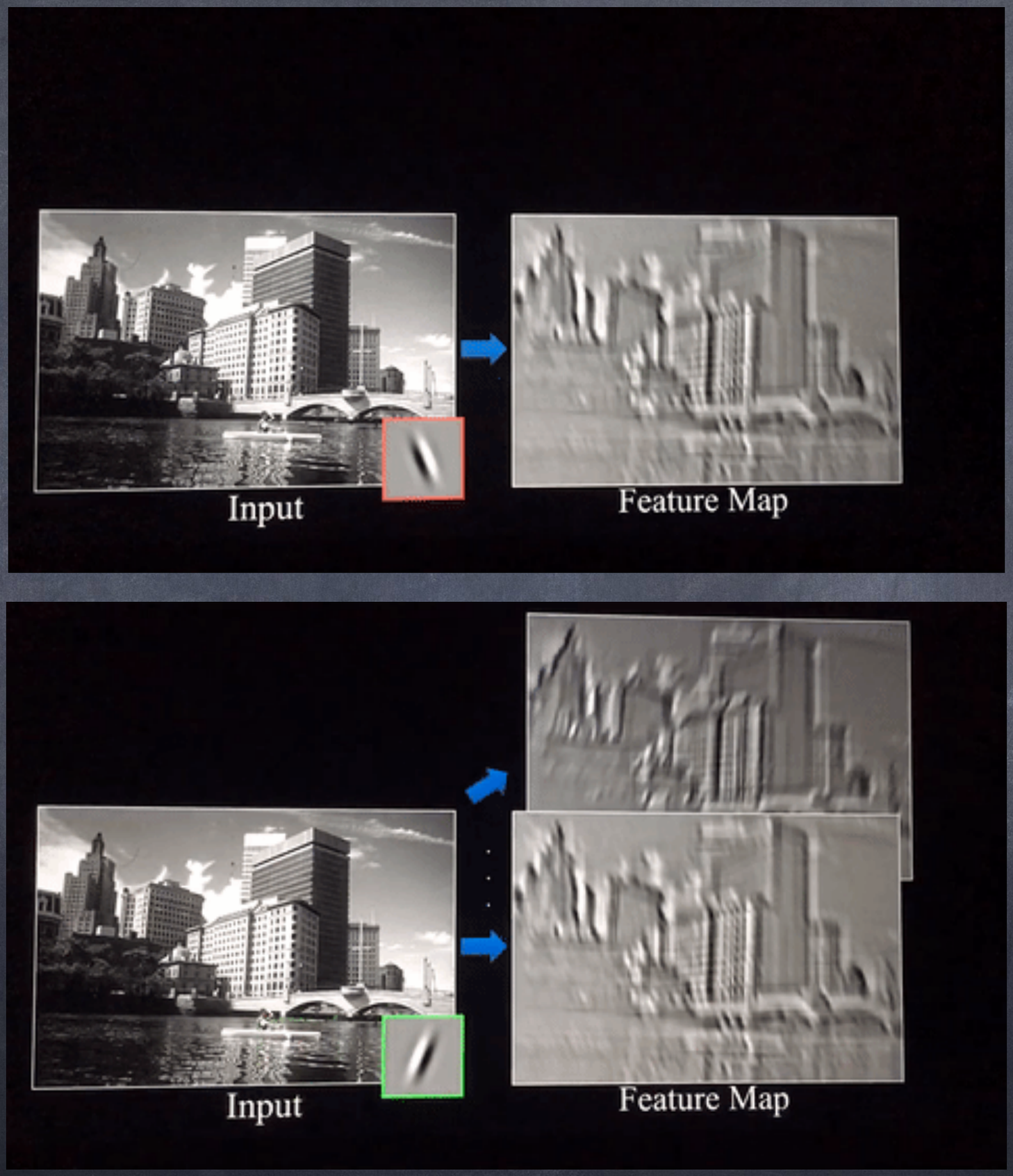


Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

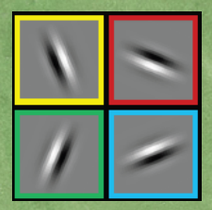
CNN


CONVOLUTIONAL NET
hidden unit /
filter response
feature map

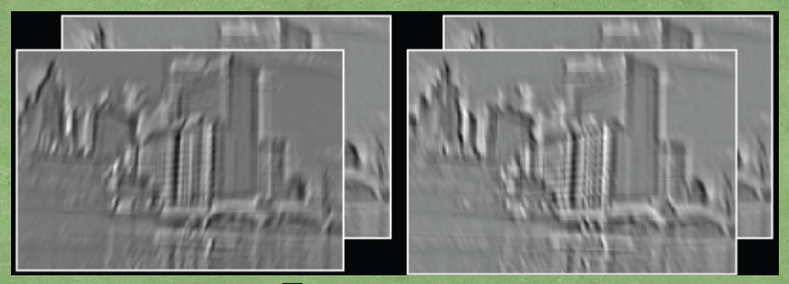

Input
Feature Map
Input
Feature Map

An Example of Filtering

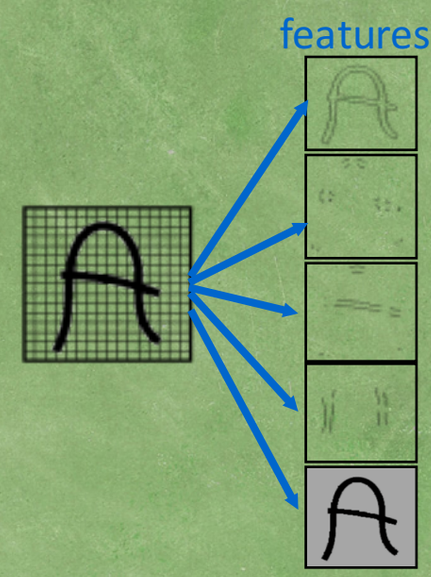
Input



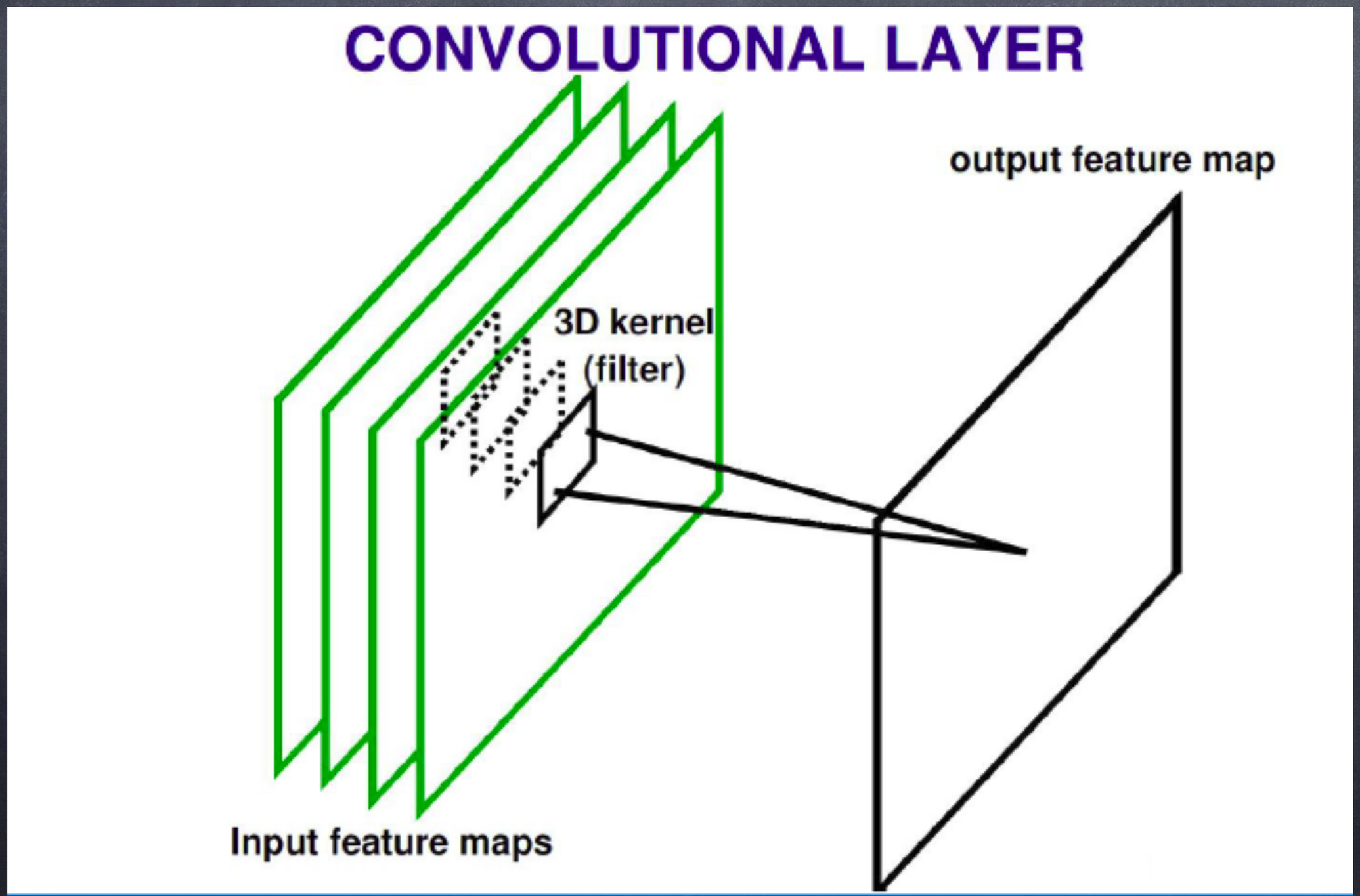
Filter
(kernel)



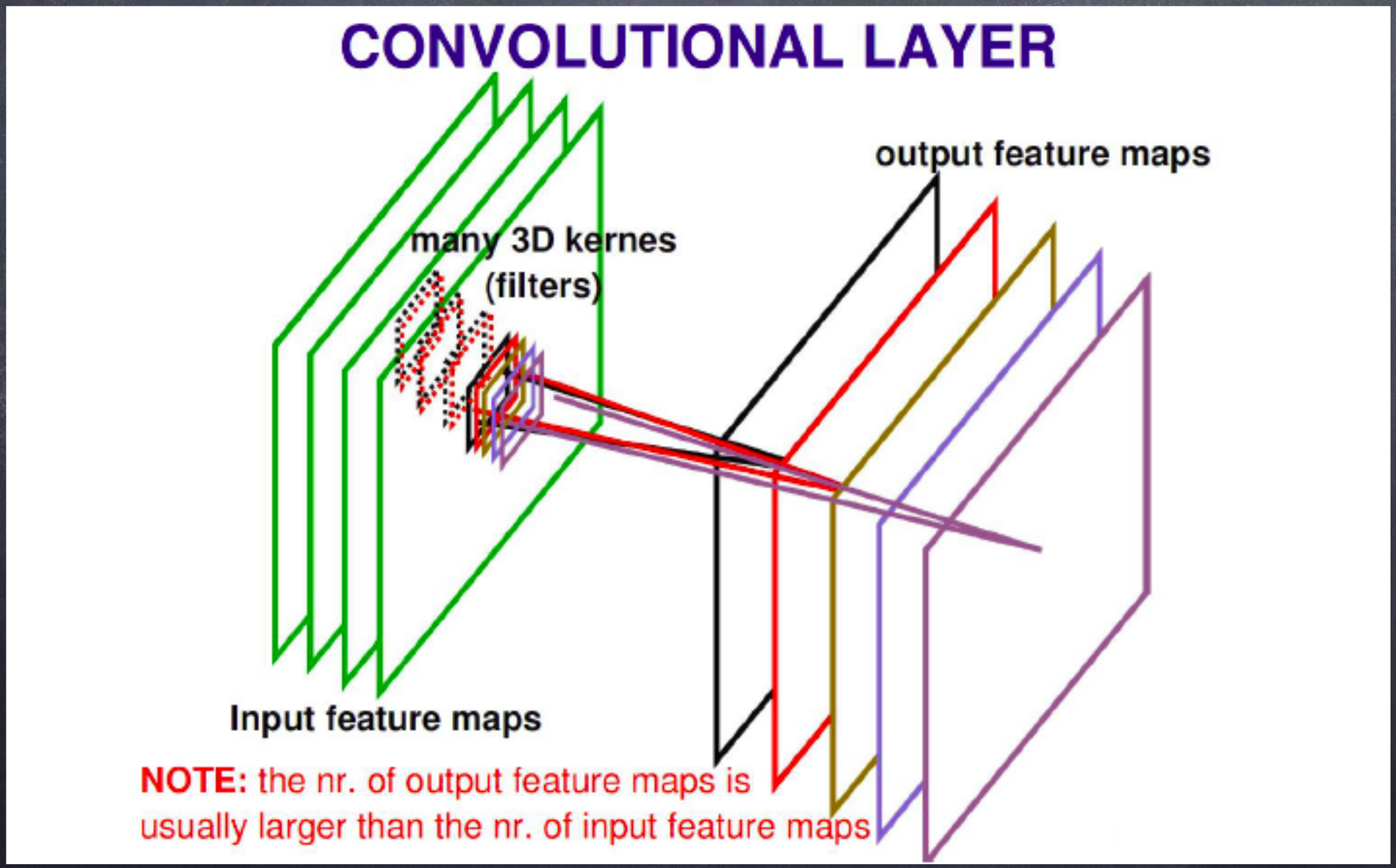
Feature map



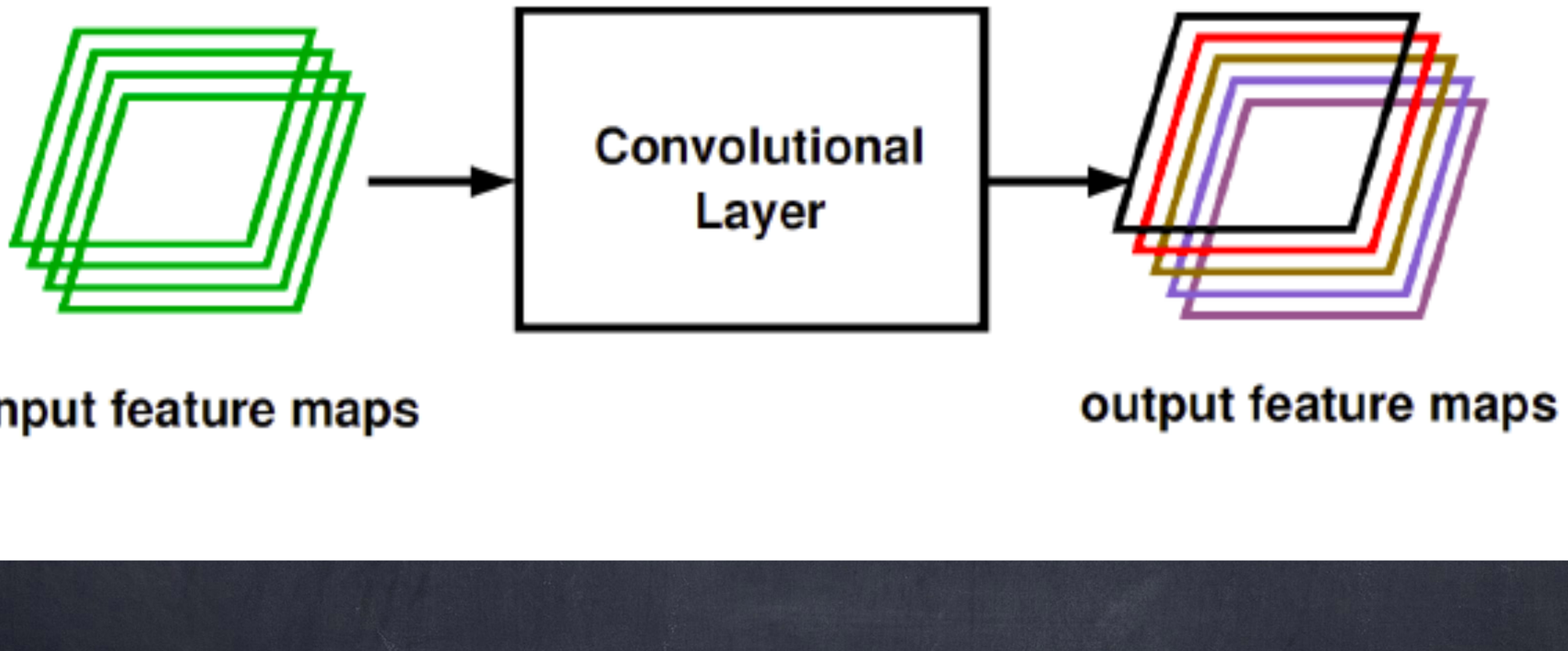
CNN


CONVOLUTIONAL LAYER
output feature map
3D kernel
(filter)
Input feature maps

CNN

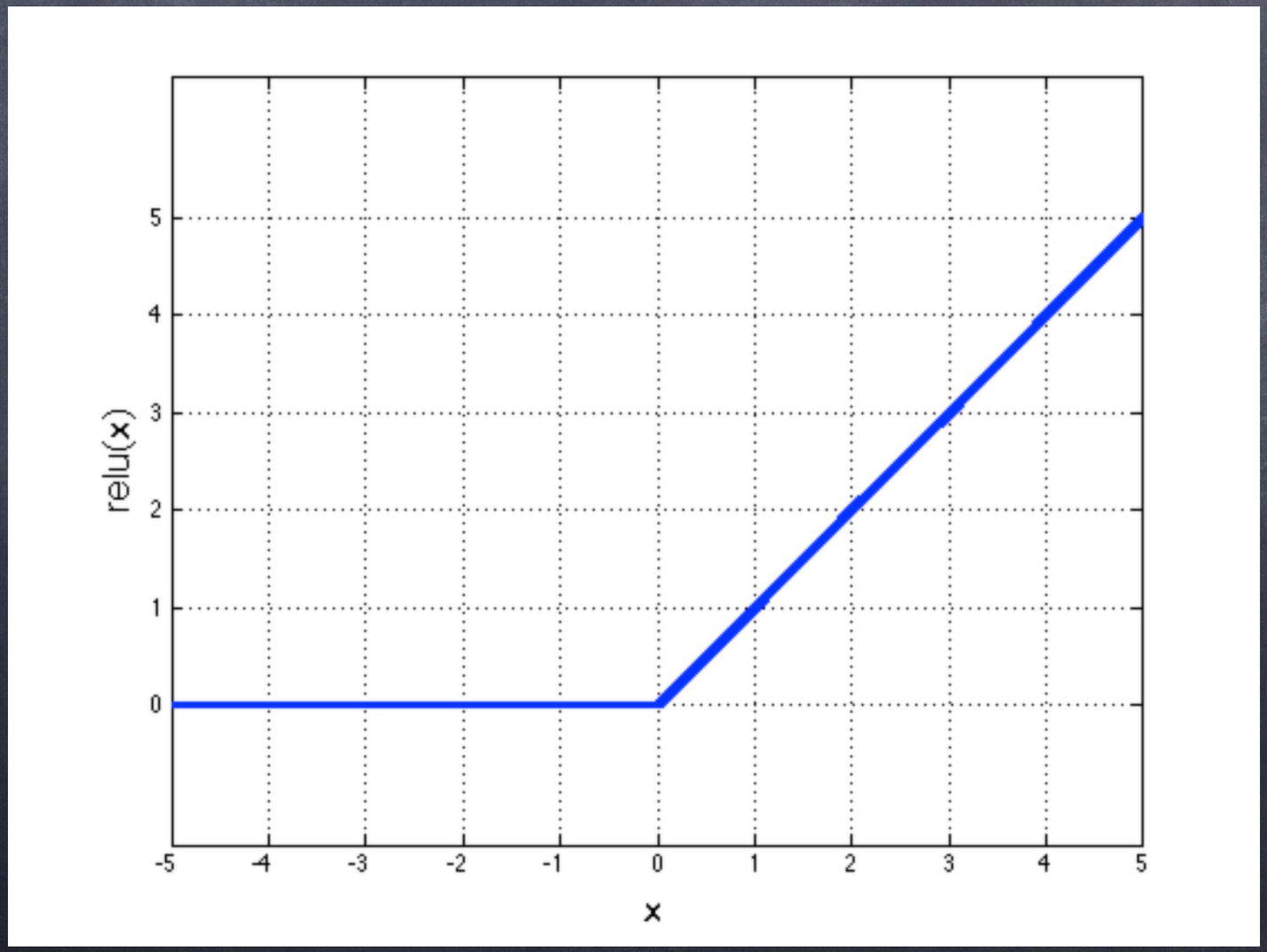

CONVOLUTIONAL LAYER
output feature maps
many 3D kernes
(filters)
Input feature maps
NOTE: the nr. of output feature maps is
usually larger than the nr. of input feature maps

CNN


Convolutional Layer
input feature maps
output feature maps

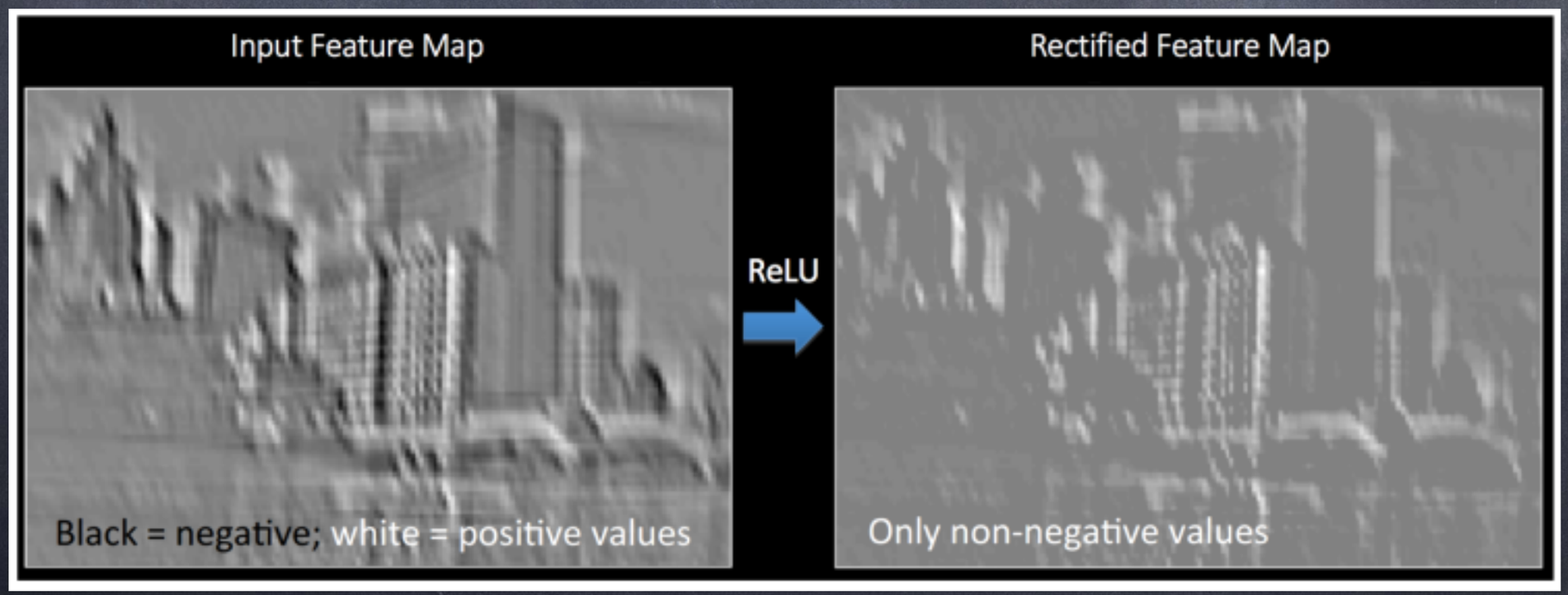
Non Linearity

- Rectified Linear (ReLU) : max(0,x)



ReLU

- Easy to implement and backprop

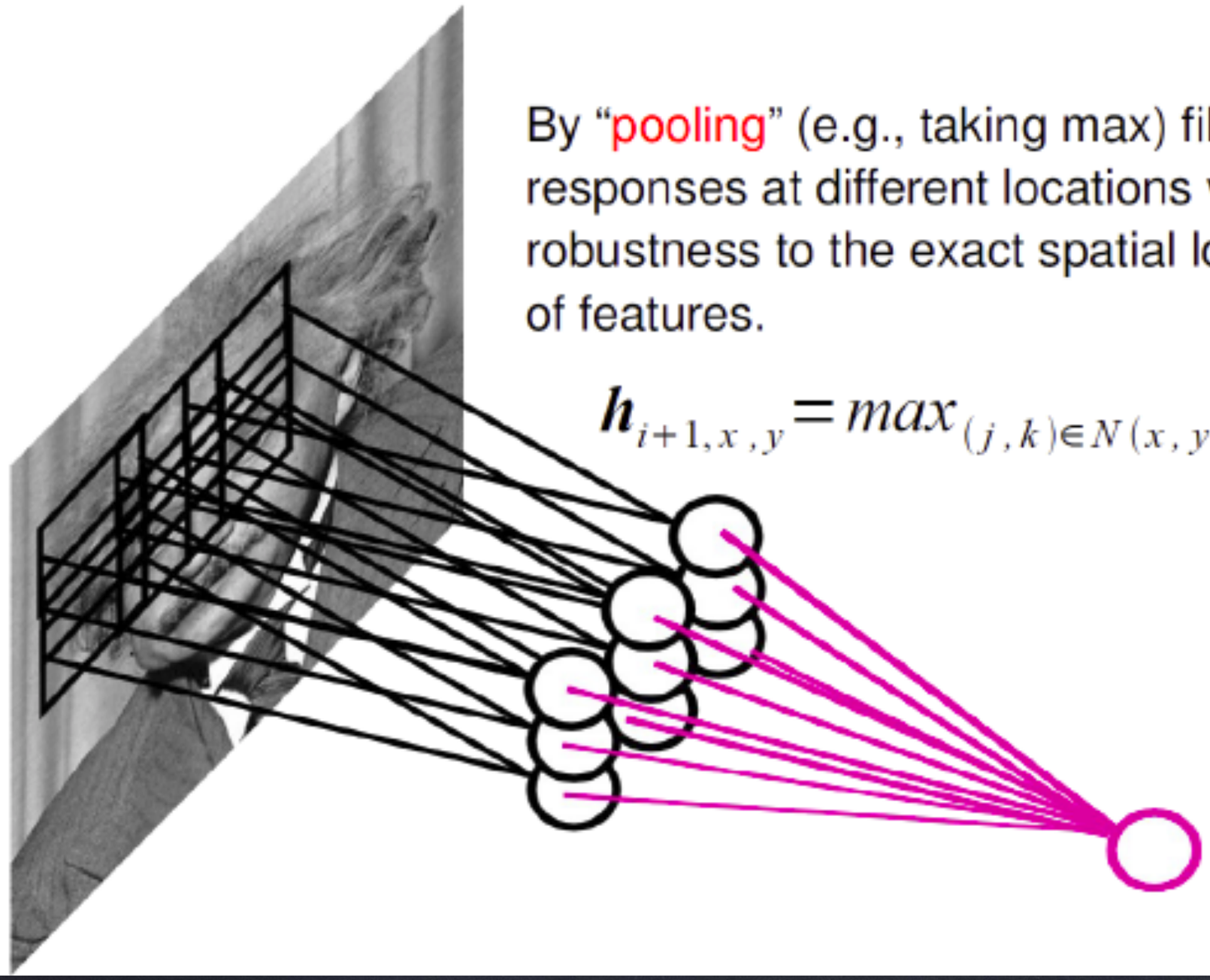


CNN

POOLING

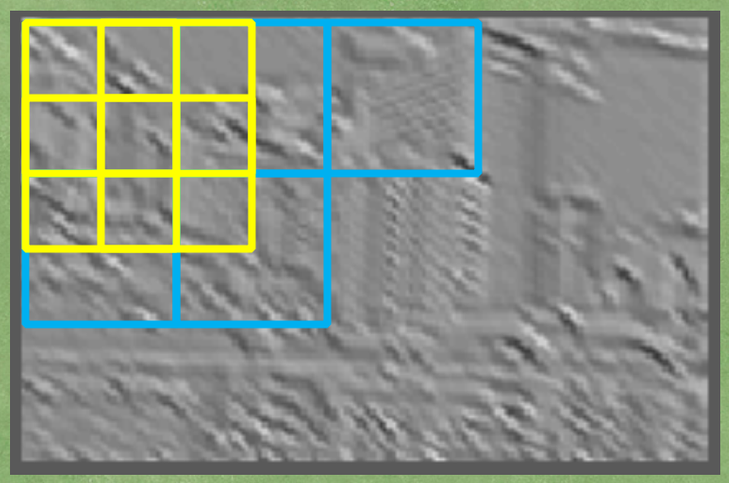
By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

$$h_{i+1,x,y} = max_{(j,k)\in N(x,y)} h_{i,j,k}$$

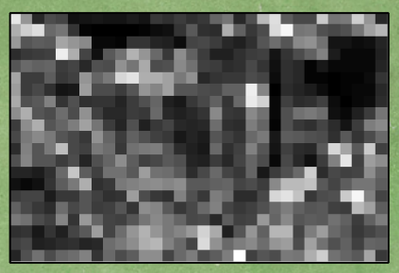


CNN

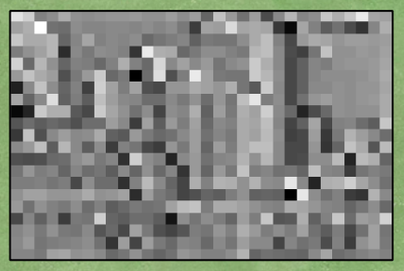
- Spatial Pooling
  - Non-overlapping / overlapping regions
  - Sum or max
  - Invariance to small transformations
  - Larger receptive fields
    (see more of input)



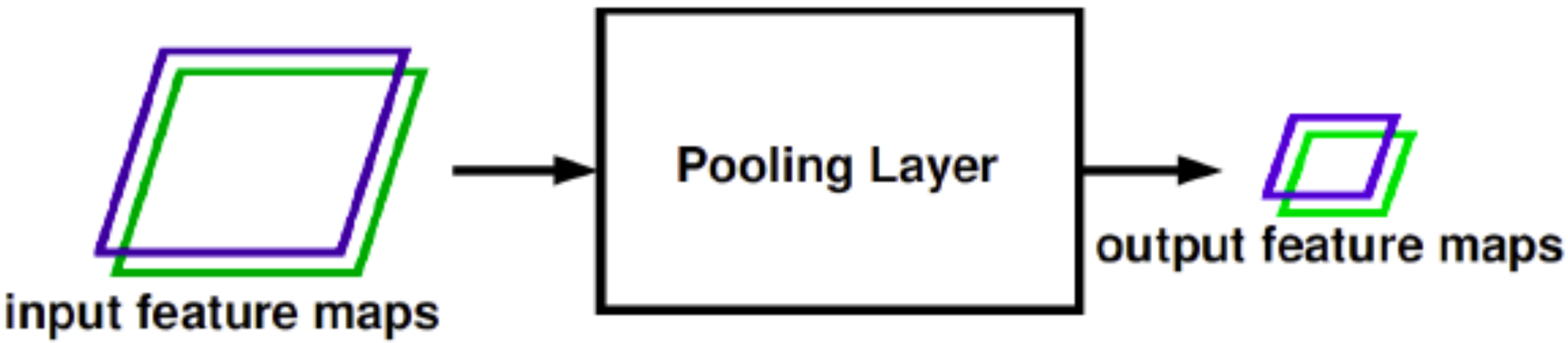
Max



Sum

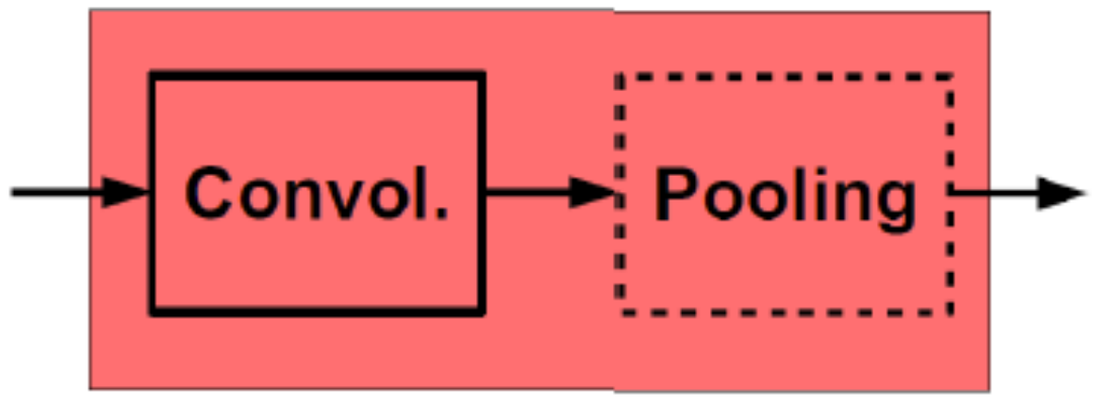


CNN

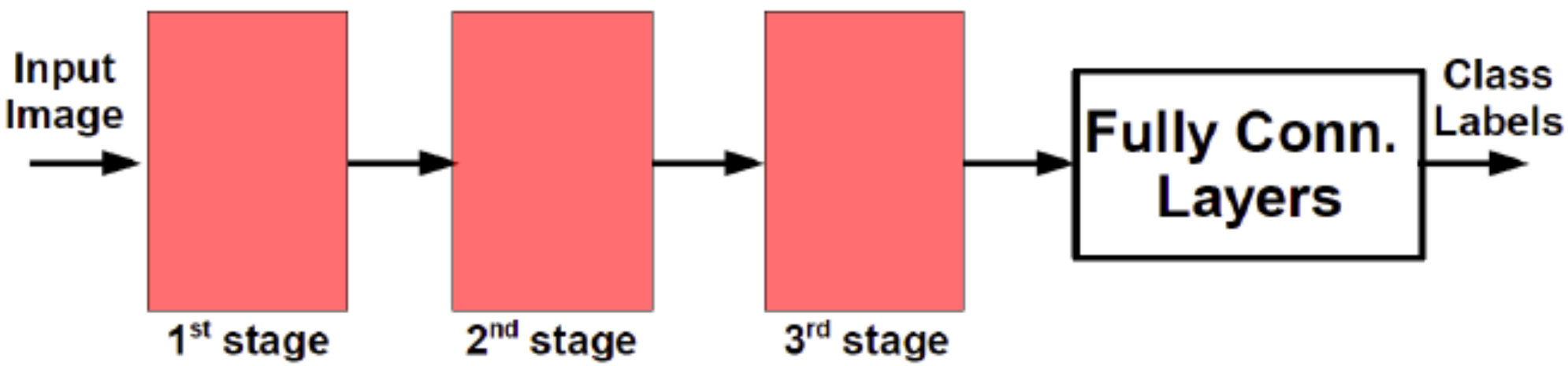


CNN

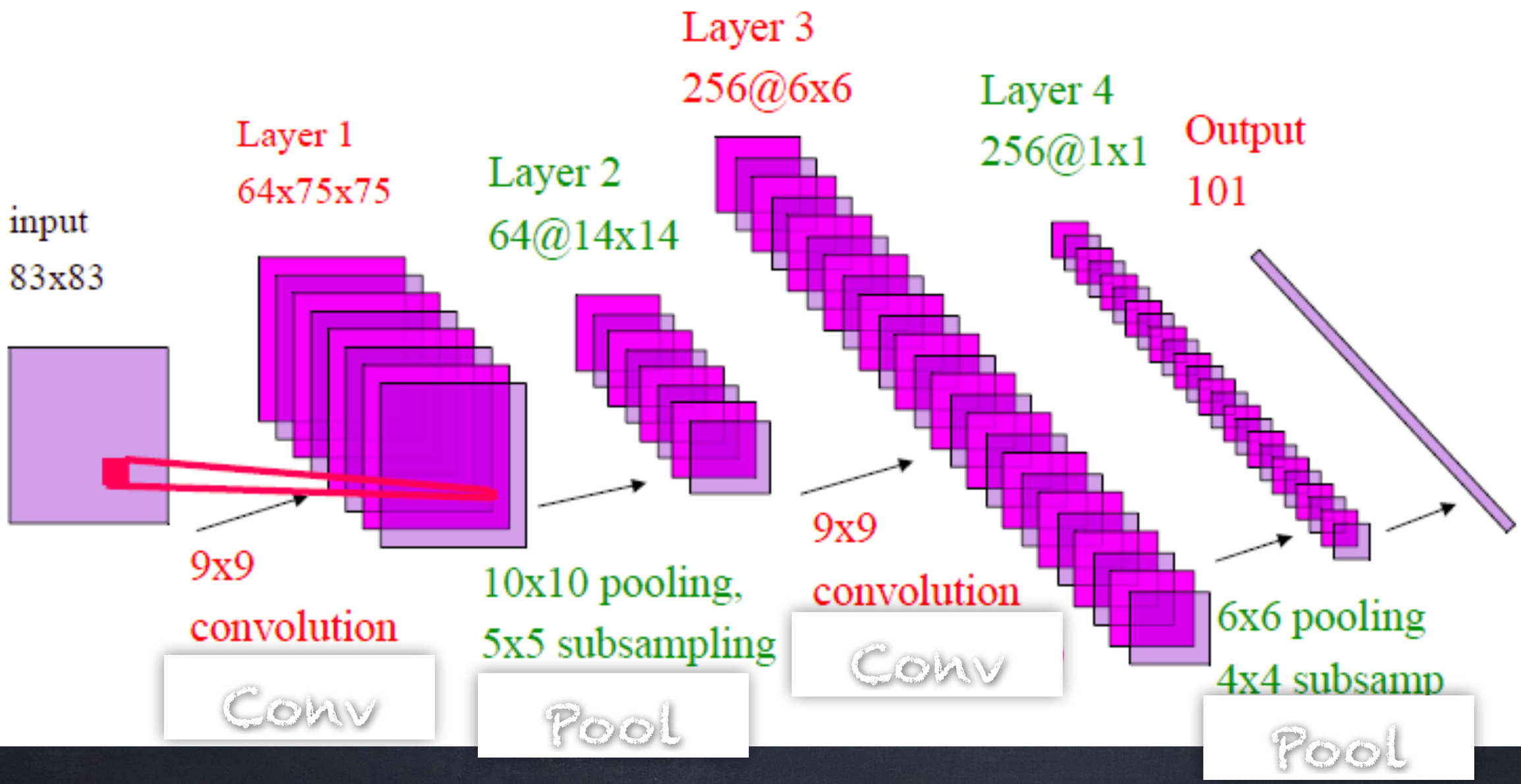
One stage (zoom)



Whole system



LeNet Architecture



How do we use CNNs for Object Detection and Recognition?