MAC5921 – Deep Learning

Aula 05 – 24/08/2023

Nina S. T. Hirata

ANTES de CNNs

Image classification

Bag of visual words (BOW)

Inspiration comes from **NLP**

Bag-of-words: widely used for document classification/retrieval

Vocabulary with N words

First idea: Document D represented by a vector $\mathrm{f_D}$ of size N:

$$\mathrm{f_D}[\mathrm{i}] = \text{number of occurrences of word } v_\mathrm{i} \text{ in the document}$$

Improvement: term frequency–inverse document frequency (TF-IDF)

- normalization taking into account the frequency of the words
- $\mathrm{f_D}[\mathrm{i}]$ divided by the count of occurrences of v_i over all documents

Images (Computer Vision)

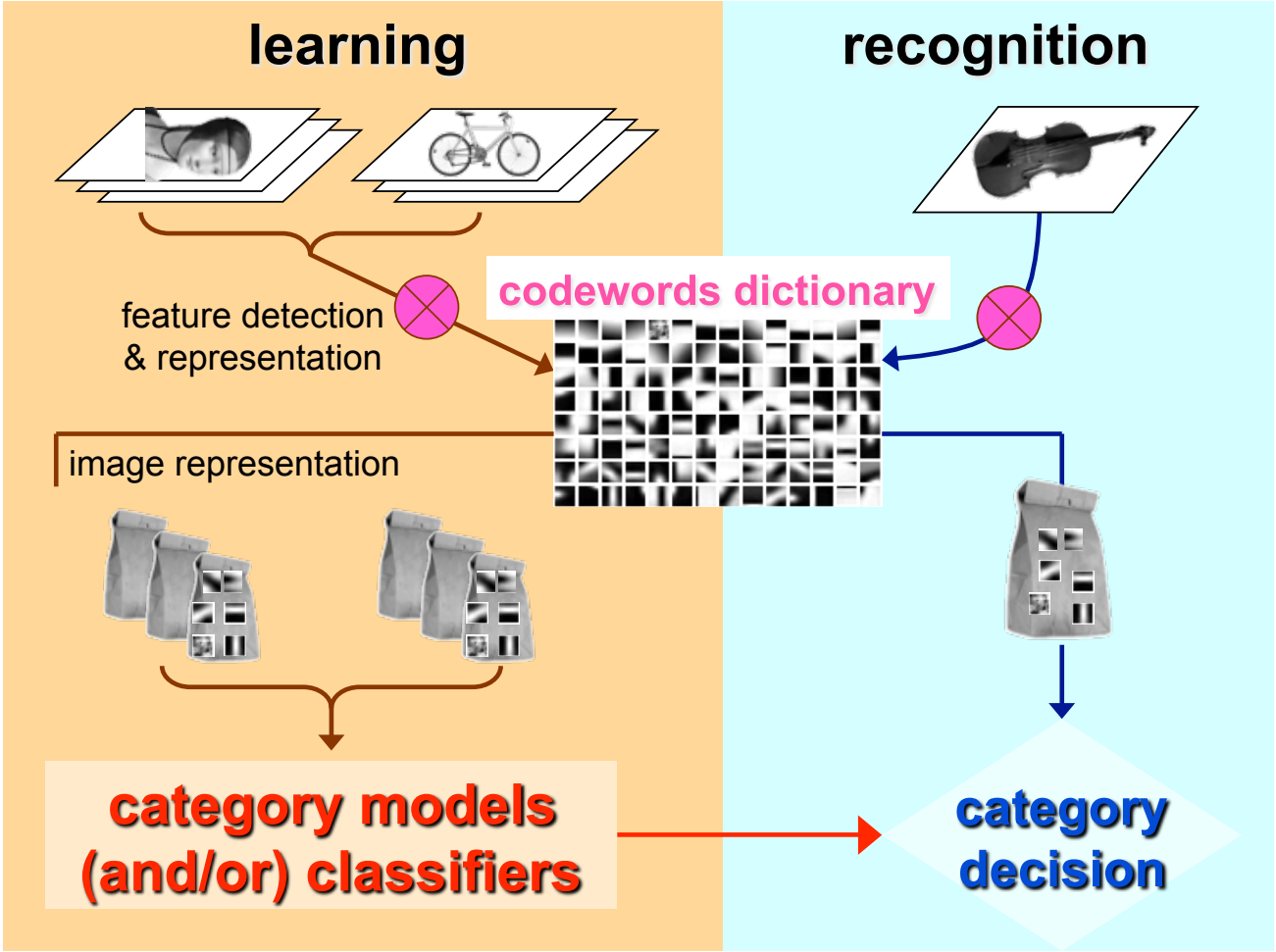
Bag of keypoints (Csurka, 2004)

1. **Create a dictionary** (visual patterns) from the set of training images

2. **Training**: compute the *bag-of-words* representation for each training image and train a classifier to discriminate object classes using the *bag-of-words* representation as features

3. **Classification**: compute the the *bag-of-words* representation of the image to be classified and apply the classifier

Bag-of-Words Model: Overview

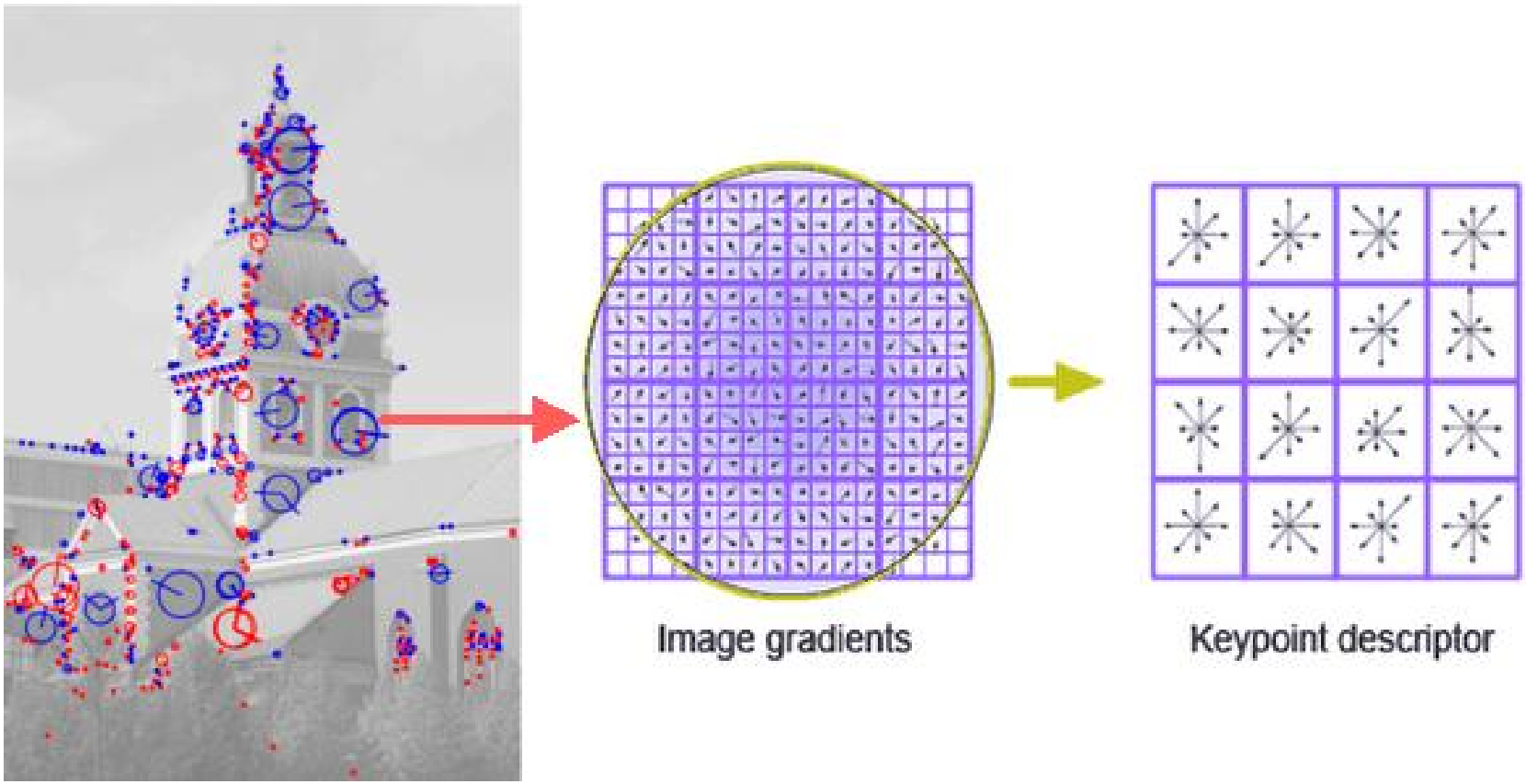


BoW – Dictionary creation

- Compute *keypoints* of the images
- Compute descriptors for each of the *keypoints*
- Csurka uses *Harris affine detector* for the detection of keypoints and SIFT for the description (feature vector of dimension 128)
- Descriptors are grouped using the K-means algorithm (parameter K is adjusted)
- The dictionary consists of the set of centers of the K clusters (**visual words**)

Example of feature detection and description

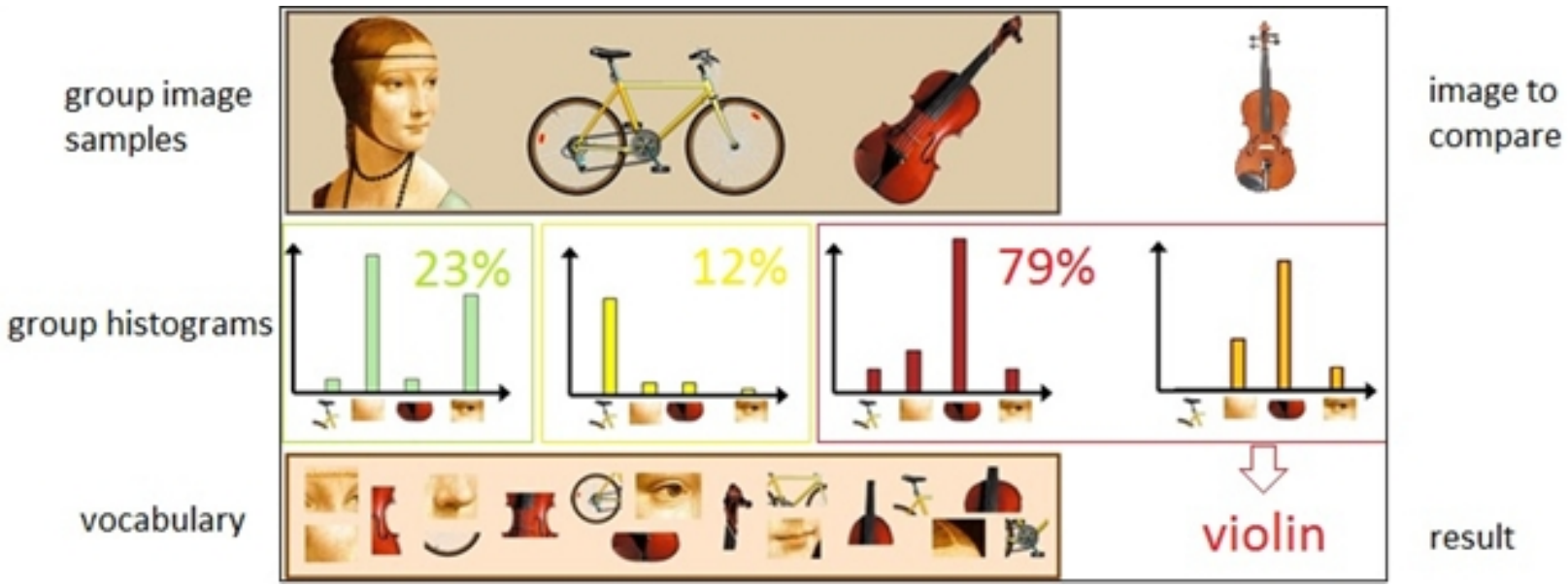
SIFT (Scale Invariant Feature Transform)



BoW – Training the classifier

- For each image, compute the set of descriptors
- Each descriptor is matched with the closest *visual word* in the dictionary
- Build a histogram of the *visual words* in the image (this is the *bag-of-words* of the image)
- The *bag-of-words* representation is the input feature used to train the classifier

BoW – Training and classification process



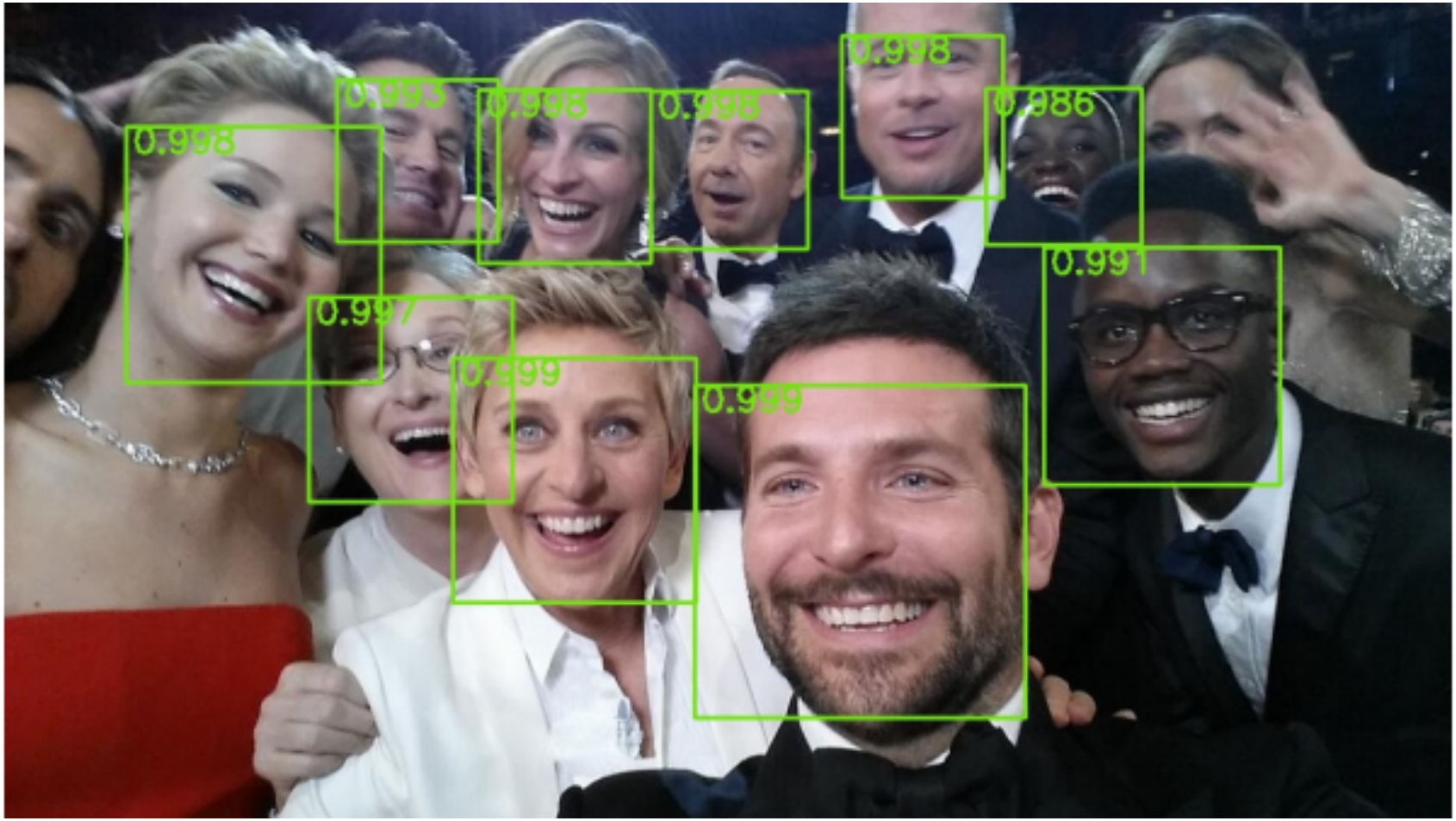
(http://vgg.fiit.stuba.sk/wp-uploads/2015/02/BOW.jpg)

Object detection

Rapid object detection using a boosted cascade of simple features, P. Viola and M. Jones, CVPR 2001

Robust Real-time Object Detection, IJCV 2004

The problem of object detection in images



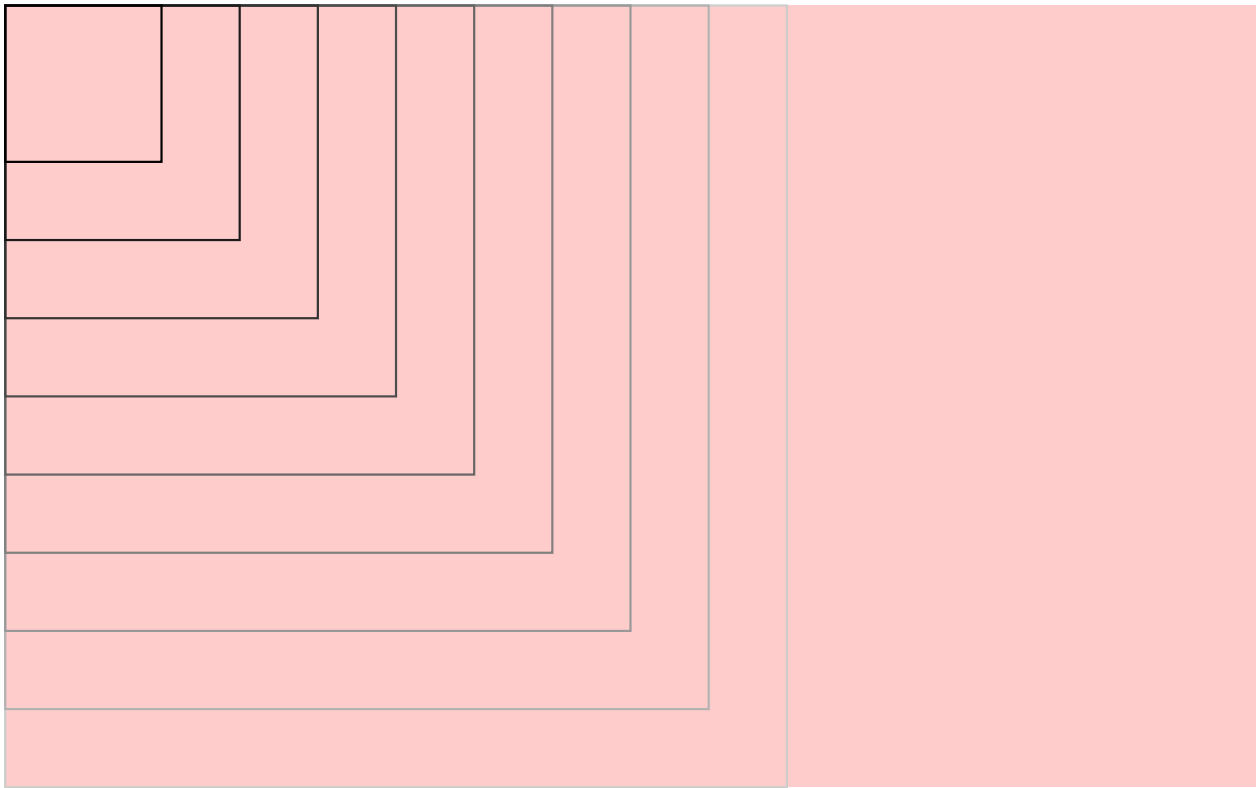
Bounding boxes are used to indicate detections

(this is the output of a Deep Learning based solution ...)

Desired characteristics for object detectors

- fast – real-time
- high detection rate
- no false positives
- robust to variations
  - localization inside the image
  - scale
  - pose
  - ilumination
  - partial occlusion

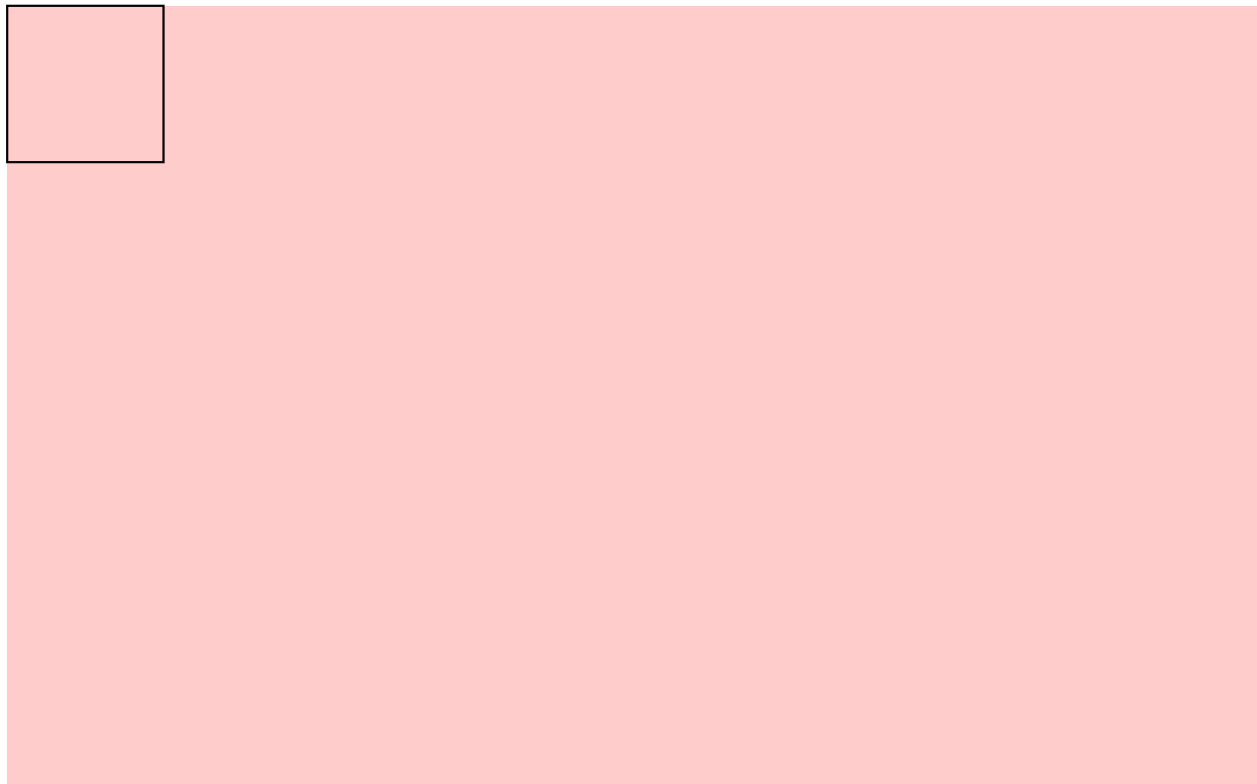
Scale



At each position in the image, the window is scaled by a constant factor

Multiple image patches

Localization



The window is shifted over the image by a step Δ

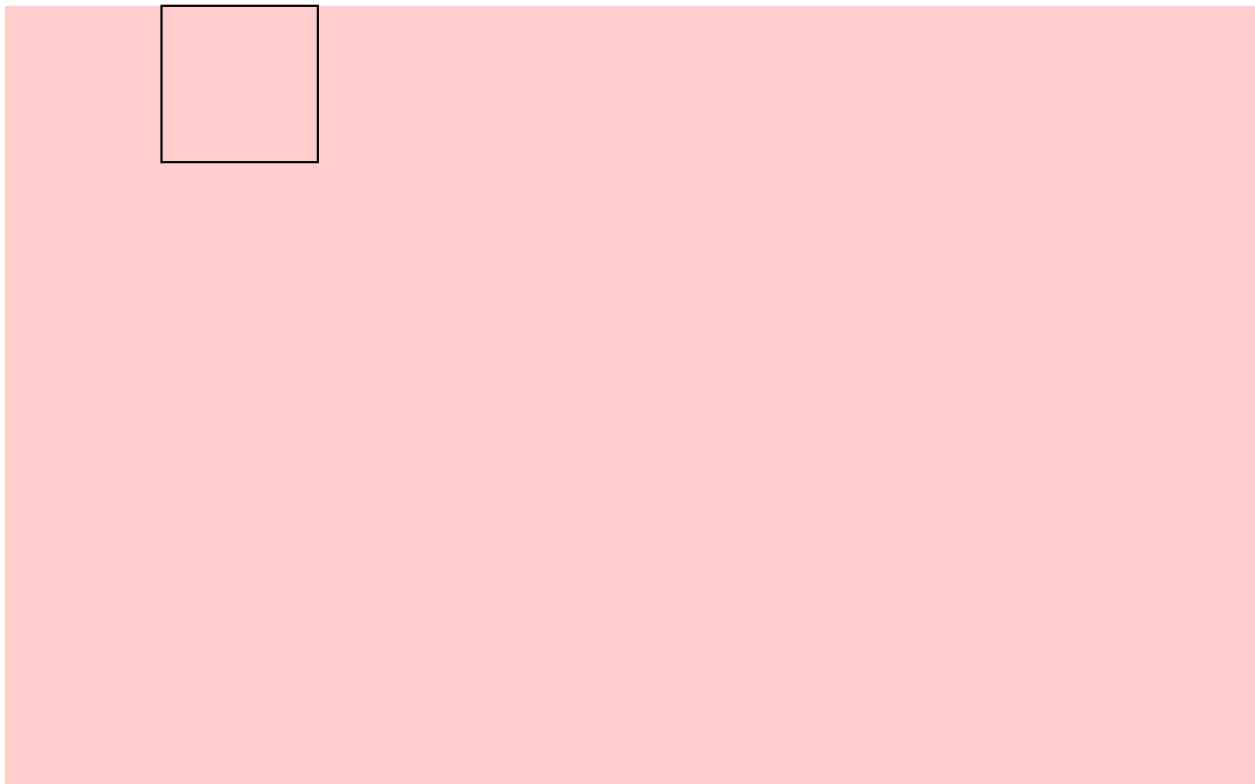
Position invariant

Localization

The window is shifted over the image by a step Δ

Position invariant

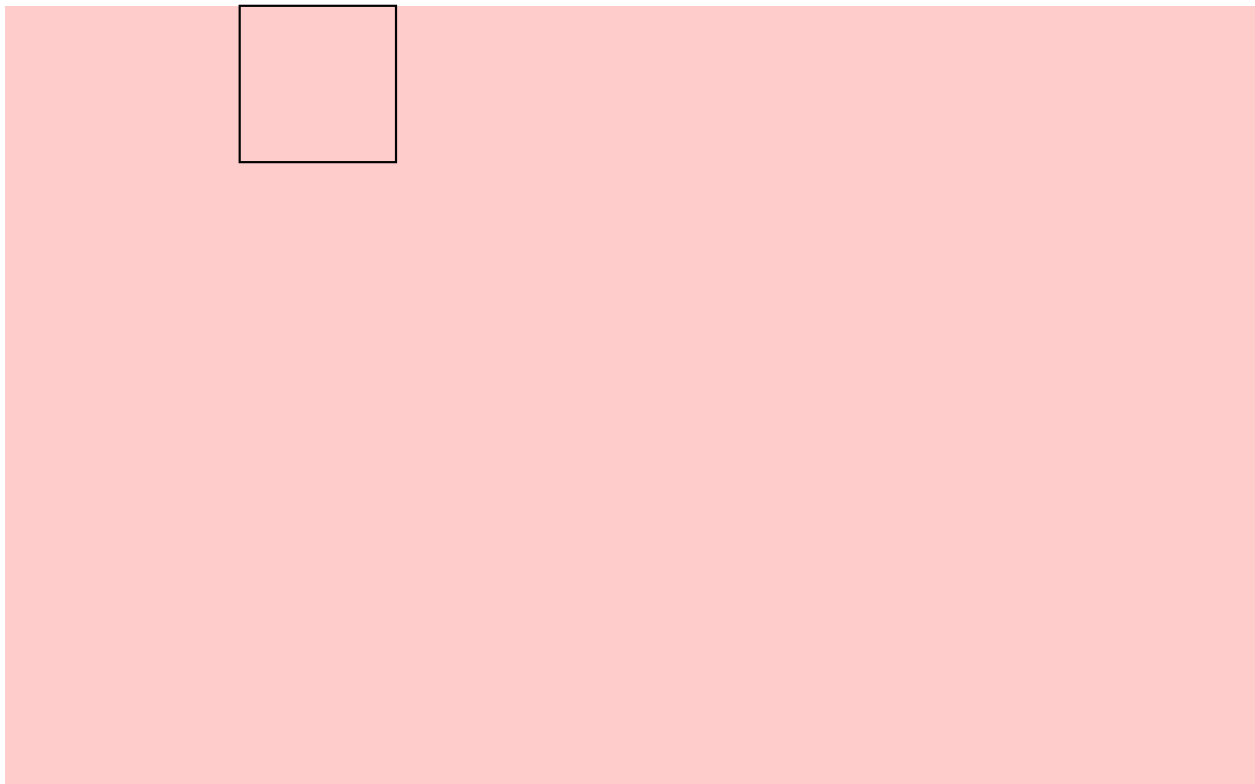
Localization



The window is shifted over the image by a step Δ

Position invariant

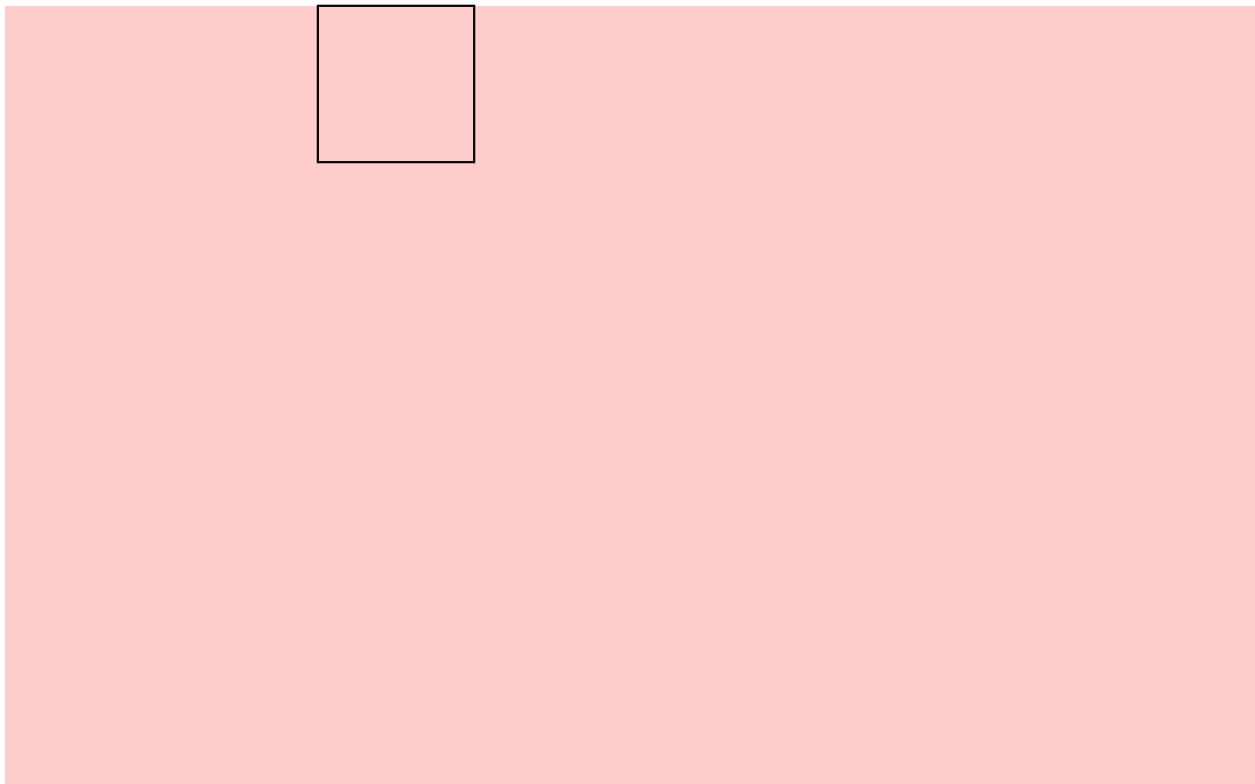
Localization



The window is shifted over the image by a step Δ

Position invariant

Localization

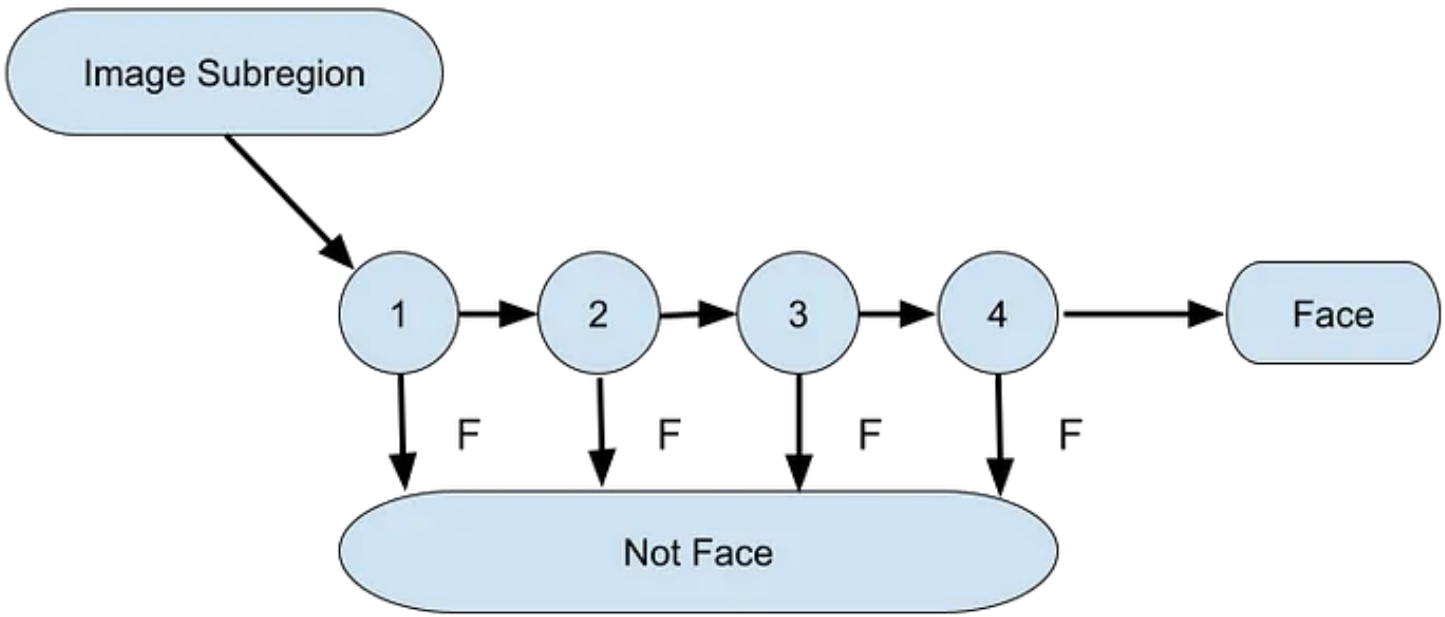


The window is shifted over the image by a step Δ

Position invariant

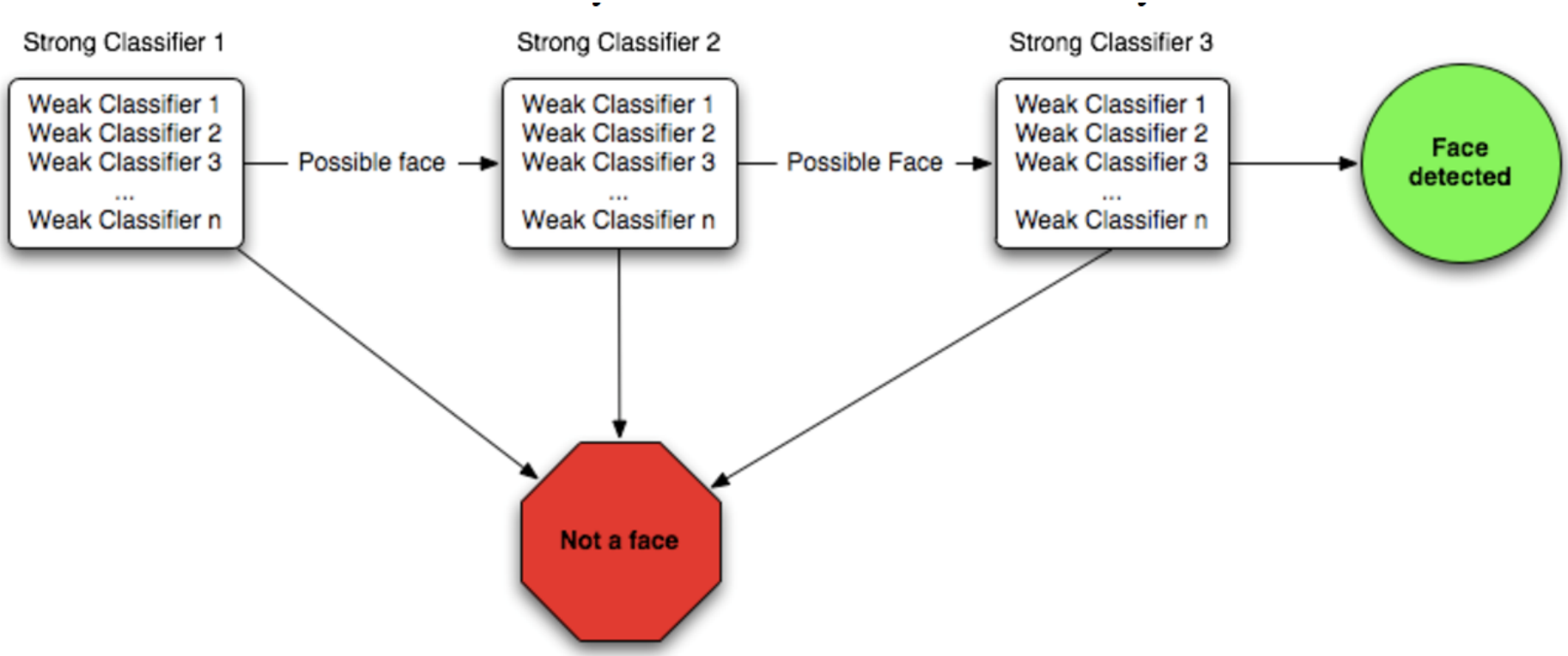
Para cada patch (diferentes escalas e localizações)

Transformar para um tamanho padrão fixo



Viola-Jones – based on Adaboost and cascade of classifiers

Image patch classifier structure

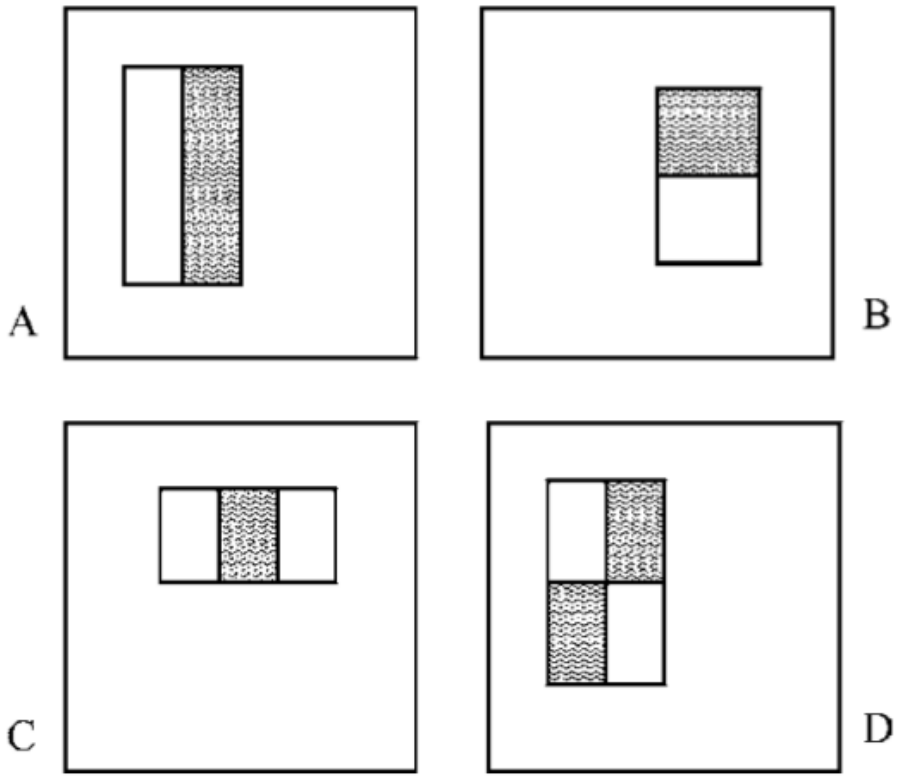


(https://www.quora.com/How-can-I-understand-Haar-like-feature-for-face-detection)

Viola & Jones – Four key ideas

1. **Haar-like** features
2. **Integral image** for fast computation of features
3. **Classifier based on Adaboost**, using Haar-like features, to classify image patches
4. **Cascade of classifiers** for rejecting false *patches* as soon as possible

Haar-like features



patch size: 24×24

(sum of intensities under white region) - (sum of intensities under dark region)

[or vice-versa]

Integral image

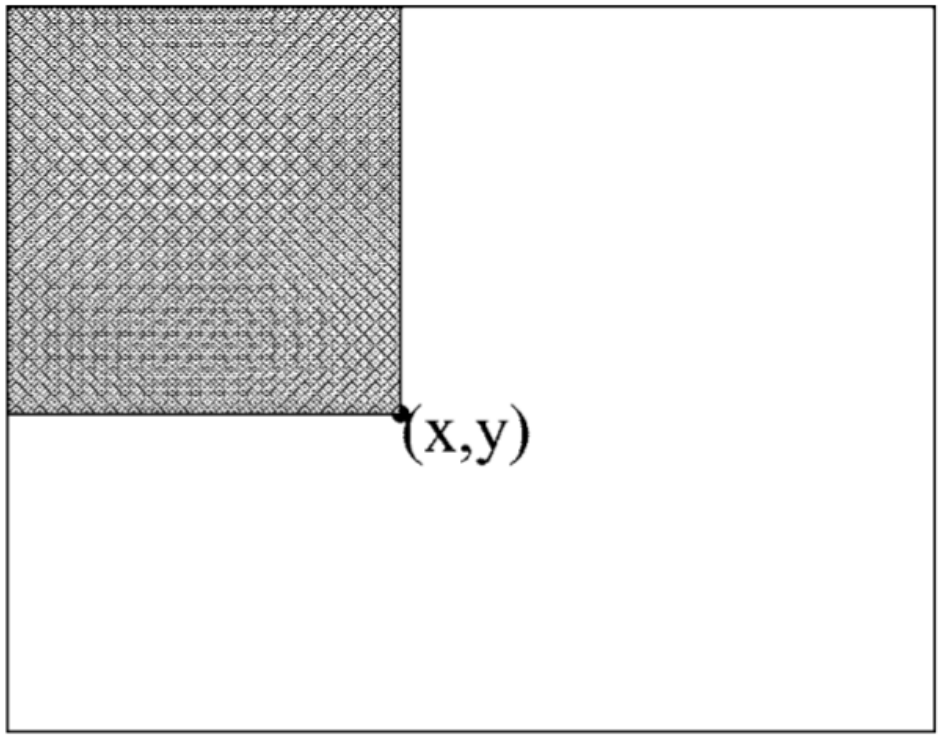


Figure 2. The value of the integral image at point (x, y) is the sum of all the pixels above and to the left.

Feature computation

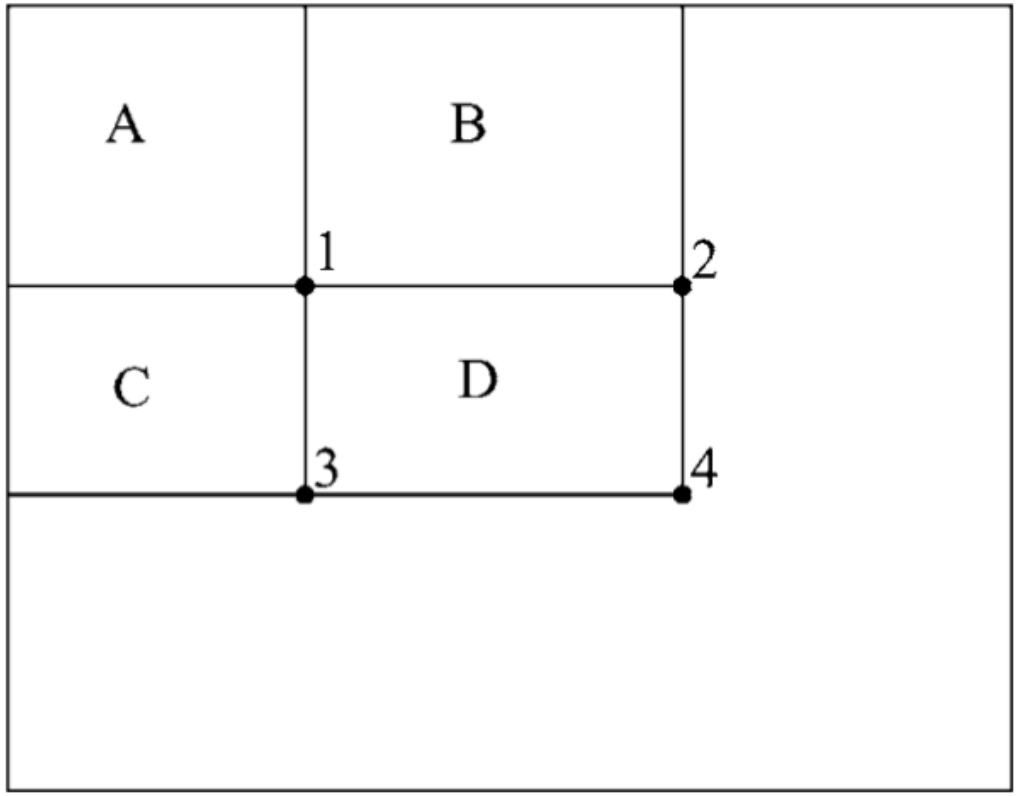


Figure 3. The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

Adaboost

Weighted combination of *weak classifiers*

A weak classifier $(h(x, f, p, \theta))$ thus consists of a feature (f), a threshold (θ) and a polarity (p) indicating the direction of the inequality:

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & \text{otherwise} \end{cases}$$

Here x is a 24×24 pixel sub-window of an image.

Adaboost

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

- For $t = 1, \ldots, T$:
  1. Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  2. Select the best weak classifier with respect to the weighted error

     $$\epsilon_t = \min_{f,p,\theta} \sum_i w_i \, |h(x_i, f, p, \theta) - y_i| \, .$$

     See Section 3.1 for a discussion of an efficient implementation.
  3. Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t, and θ_t are the minimizers of ϵ_t.
  4. Update the weights:

     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Training details

4916 faces 24×24 + vertical flipping $\longrightarrow$ 9832
10000 non-face images

At each level of the cascade, 10000 new false positives are collected from images that do not contain faces

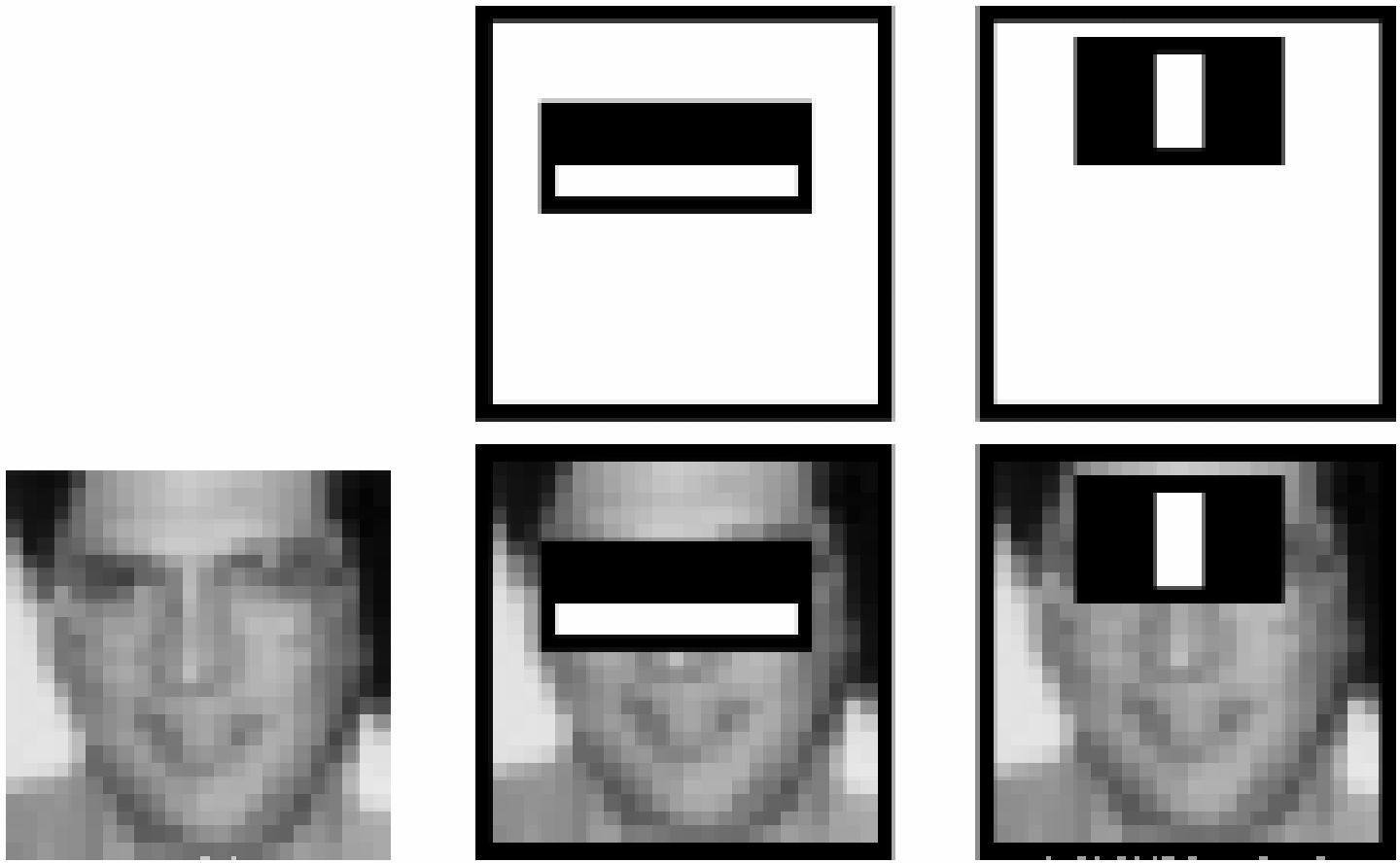
Each classifier in the cascade is trained using the Adaboost algorithm, with threshold adjusted to minimize false-negatives

The closer to the end of the cascade, the harder is the classification task

Final: 38 levels in the cascade, total of 6061 features

Exemples of face images usend in training

Result: The two main features

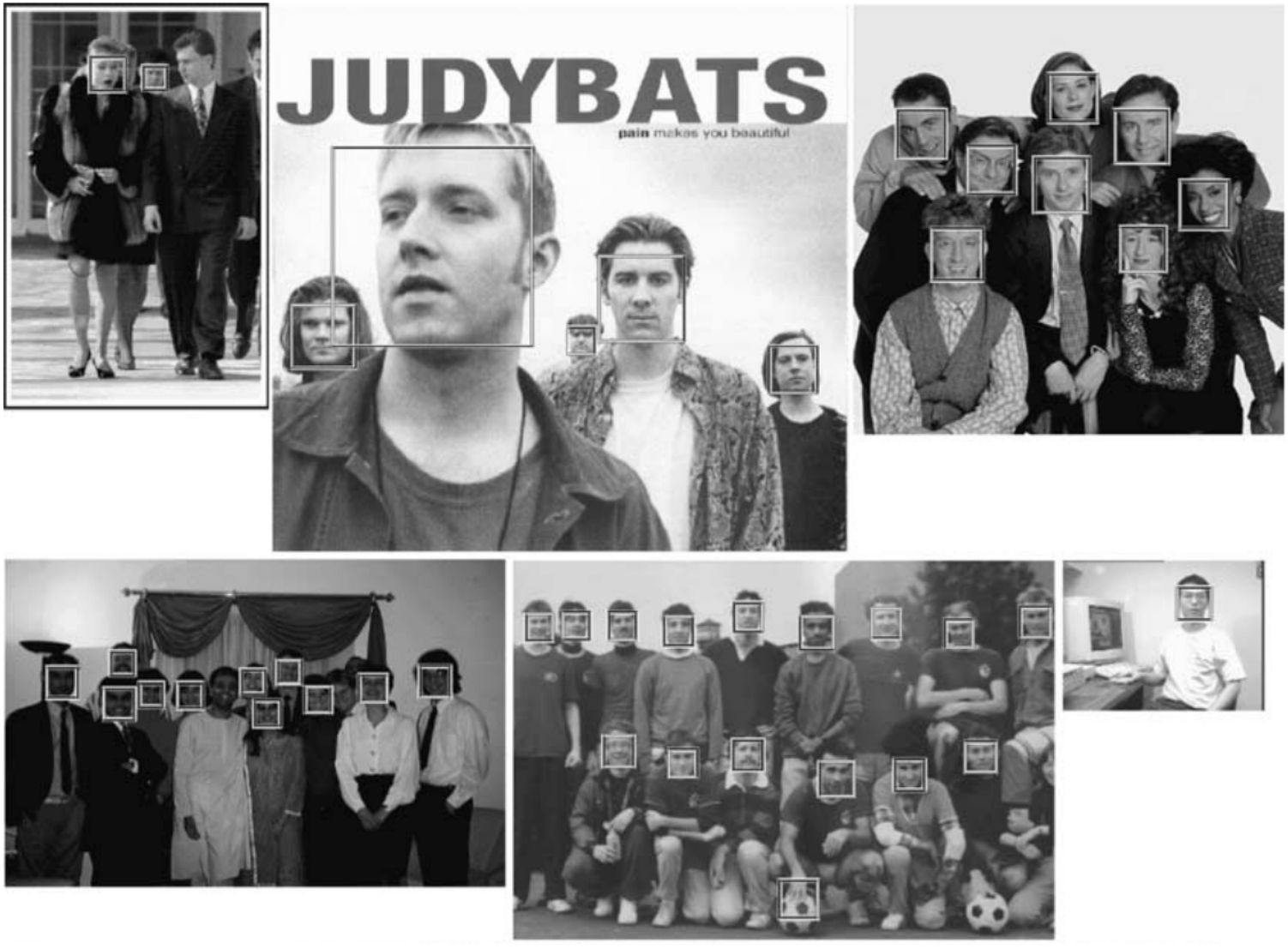


(https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html)

Results

Table 3. Detection rates for various numbers of false positives on the MIT + CMU test set containing 130 images and 507 faces.

	False detections							
Detector	10	31	50	65	78	95	167	422
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%	94.1%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2%	93.7%	–
Rowley-Baluja-Kanade	83.2%	86.0%	–	–	–	89.2%	90.1%	89.9%
Schneiderman-Kanade	–	–	–	94.4%	–	–	–	–
Roth-Yang-Ahuja	–	–	–	–	(94.8%)	–	–	–


JUDYBATS
pain makes you beautiful