Part VII: Classification

Nicola Conci
nicola.conci@unitn.it

Pattern recognition problems

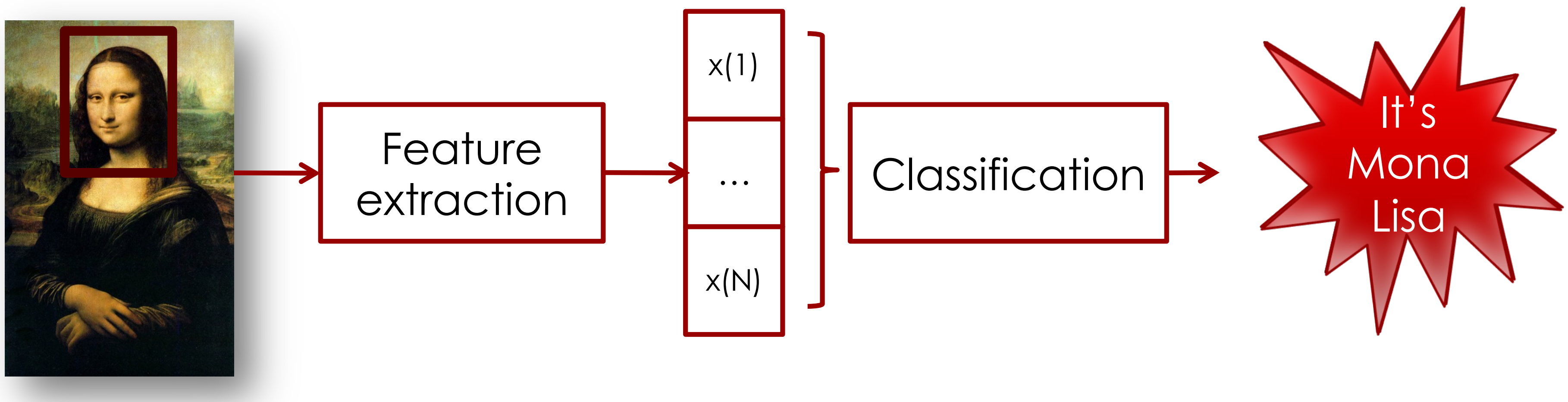
- When you have to **make a decision** about a content of an image
- When you have to determine **what the object** present in the picture **represents**
- Need to use the **features of the appearance** of the object to determine the category it belongs to
- Examples:
  - In a supermarket implement a system that recognizes vegetables
  - At the gate of your house, restrict the access to specific subjects or cars
  - At the entrance of a parking lot determine the category of the entering vehicle
  - On a robot, look for objects (doors, humans, obstacles)

Classes

- Objects, patterns, textures are arranged in classes
- A **class is made up of descriptions** provided by a number of examples
- Classes contain items that share **similar properties**
- Goal of a classifier is to take an object as input and to output the class label it belongs to

The classification process

- Starting from an input image the steps are:
  - Segment the image (if necessary)
  - Extract the features
  - Use the feature vector to feed a classifier
  - Output a label



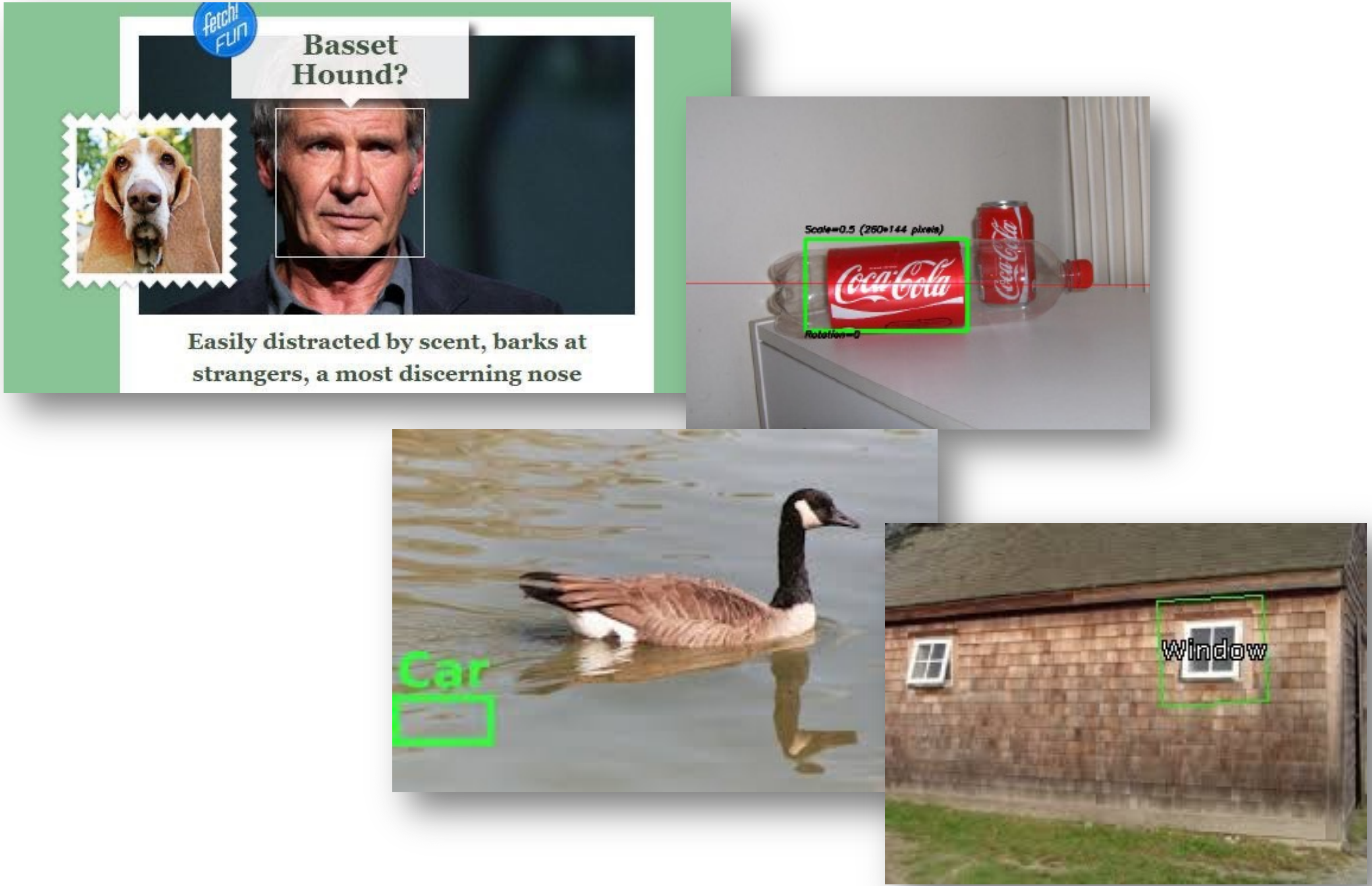
Basic principles

- Children learn new things by examples
- In order to distinguish a dog from a cat, they rely on observing samples of dogs and cats
- We realize they have learned the concept by the time they see a new sample and they're able to recognize it
- Teaching a computer works the same way

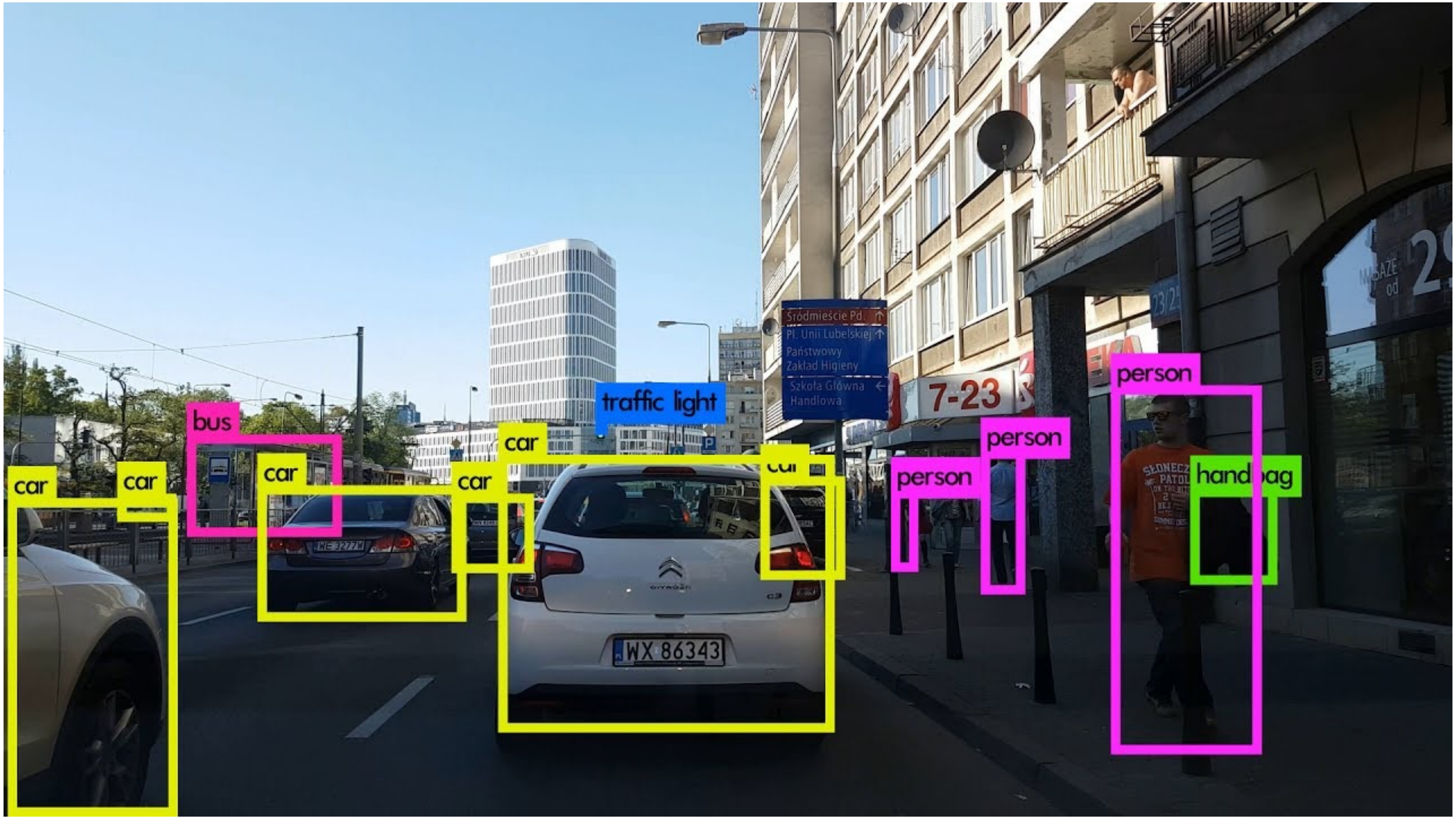
Subject to failures

- Classification is a tough job
- It usually requires **A LOT** of annotated data
- Machines may take days or weeks in order to learn concepts
- Still they are likely to fail due to:
  - Lack of data
  - Wrong annotations
  - Occlusions
  - Perspective
  - Similarity among classes

Examples of Failures…



… and also very good results



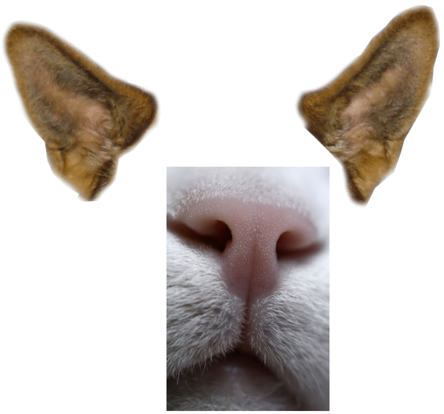
The pipeline

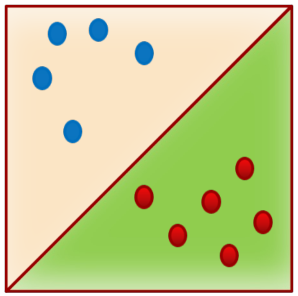
Define the task

e.g. Dogs vs cats

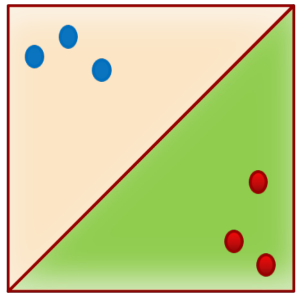
Collect data

Design features





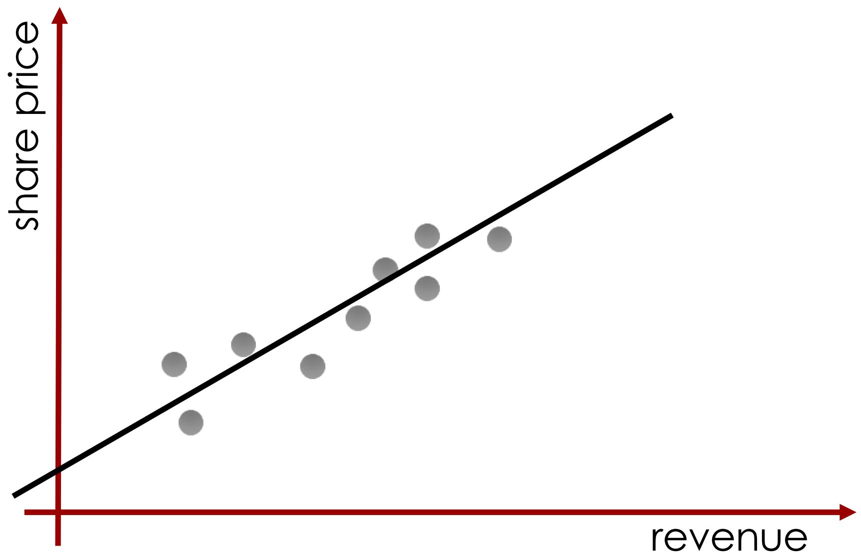
Train model



Test model

Simple problem: regression

- Use case: predict the share price of a company that is about to go public
- Training data coming from similar companies
- Features: companies revenue and share price



- Define loss function (our score to be minimized)
- Find the parameters of the line (slope, intercept)
- Use the line for testing

In our problem:

- We use the training to determine the line
- In testing, given the revenue we can predict the share price

More formally

- We have a set of training samples D={Z_1, Z_2, …, Z_n}
- We have an unknown process P(Z)
- And a loss function L(*f*,Z) where f is the decision function
- In **supervised** learning each example is a pair Z(X,Y) where
  - X contains the features for that sample
  - Y is the known/expected output
  - *f* takes X as an argument and outputs something in the range of Y
- Loss function defined as:

$$L\big(f,(X,Y)\big) = \|f(X) - Y\|^2$$

In case of multi-class

- Y is a class (finite integer) and the loss function is the negative conditional log-likelihood
- $f_i(X)$ estimates $P(Y{=}i \mid X)$

$$L\big(f, (X, Y)\big) = -\log f_Y(X)$$

- With the constraint that

$$\mathrm{f_Y(X)} \geq 0, \sum_i f_i(X) = 1$$

Unsupervised learning

- Learn f to characterize $P(Z)$
- Through clustering the space is partitioned in regions centered around a prototype (centroid)
  - K-Means (hard partitioning)
  - GMM (soft partitioning), each Z has a probability

Classification output - evaluation

- As an output you may have:
  - Correct detection
    - The item is associated to the correct class
  - Wrong detection
    - An item is associated to the wrong class
  - In a binary classification problem:
    - True positive (hit)
    - True negative
    - False positive
    - False negative (miss)

Reject option

- As we have seen the classification is subject to errors
- Errors tend to arise in the presence of those samples for which largest of the posterior probability $P(Y=i|X)<<1$

  → *intuitively, this means that the class with the highest likelihood is still very low.*
- In such cases the system can implement a so-called *reject option*, or *reject class*
- This helps avoiding taking decisions for those samples that are difficult to classify

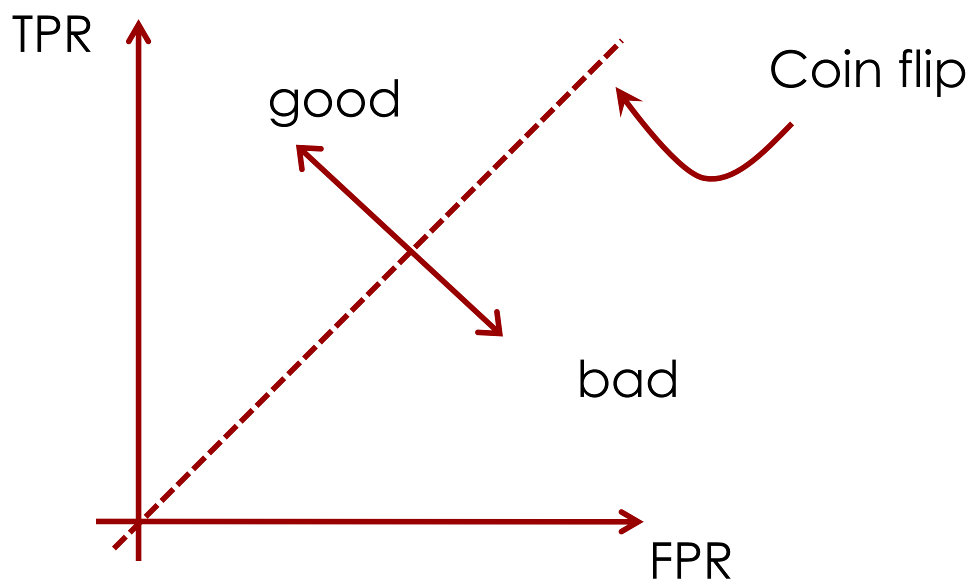
A simple example

- Let us take as a very simple example a binary classification problem
- The output can only be true or false
- Ex: Does a patient have a disease?

		Current value	
		p	n
Test Result	p	TP	FP
	n	FN	TN

The ROC curve

- Receiver Operating Characteristic
- Plot of TPR vs FPR
- Each setup of the system is a point in the ROC space



Precision and Recall

- The system is usually targeted at finding elements that satisfy a query
  - **Precision**: TP/(TP+FP) = hit/(all retrieved)
  - **Recall**: TP/(TP+FN) = hit/(hit+miss)
- In terms of probability it means:
  - Precision: probability that a randomly selected elements is relevant
  - Recall: probability that a randomly relevant element is retrieved in the search

Precision and Recall in practice

- Assume we want to find all pictures that contain faces
- We have 200 images and only 100 contain faces
- The classifier returns only 75 (TP) out of 100 but also additional 30 images (FP).
- Precision is 75/(75+30) = 71%
- Recall is 75/100 =75%

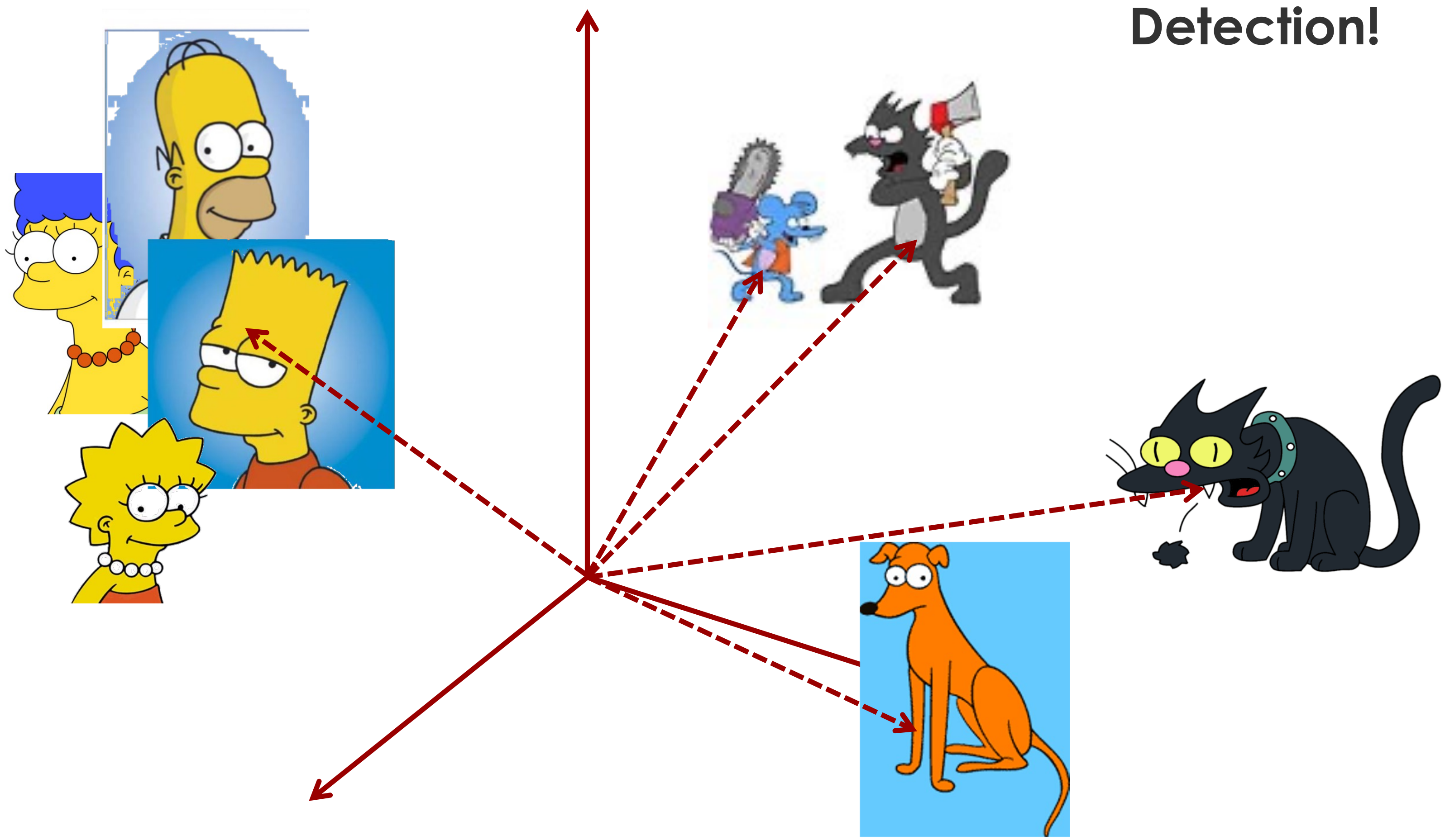
The confusion matrix

- Used in case we have multiple classes
- In vegetables: Potatoes – Lettuce –Tomatoes
- The simple true/false matrix is not sufficient anymore
- Construct a matrix with the correct classification on the diagonal and the false positives in the other cells

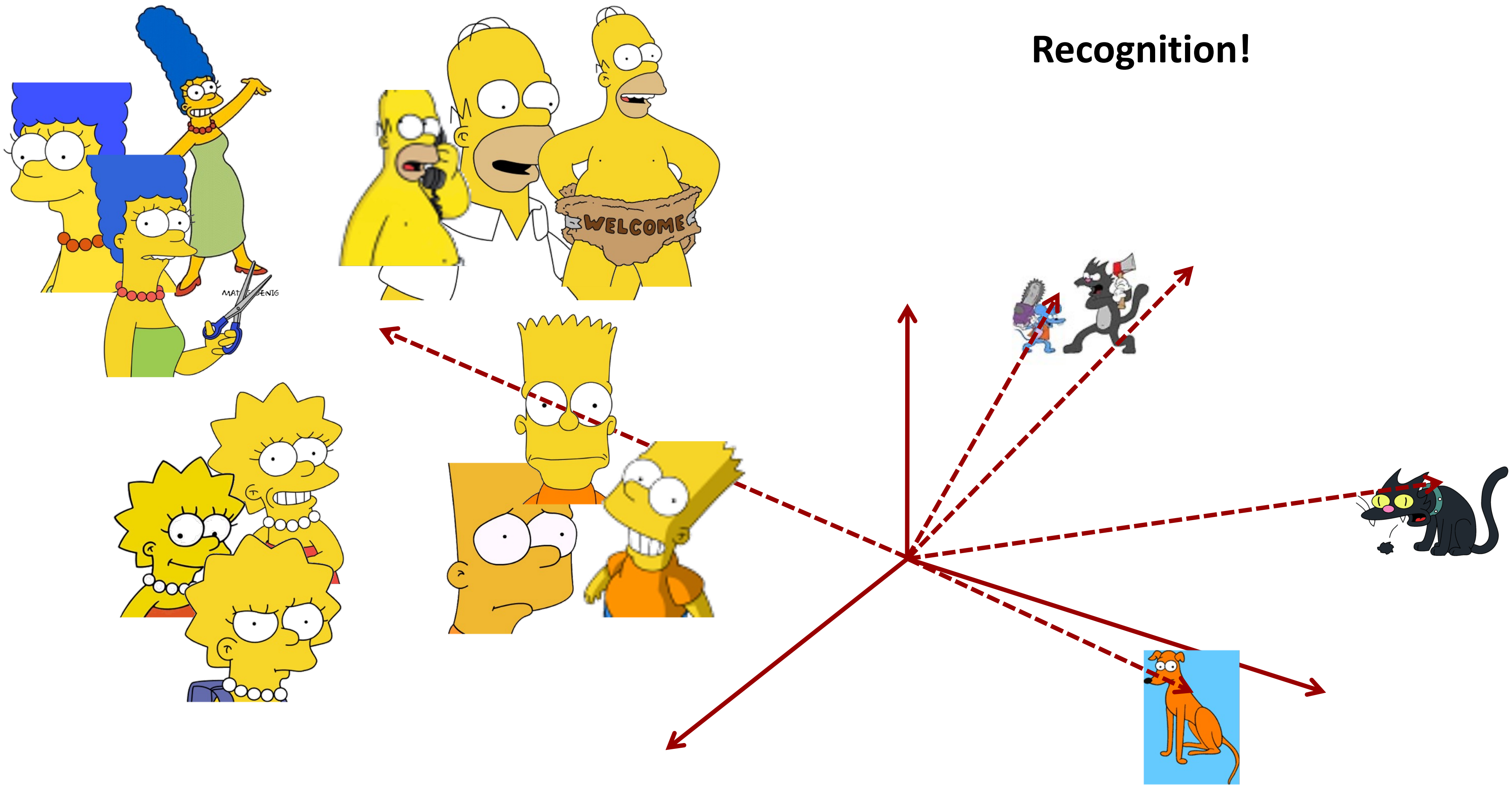
The face detection problem

- Once I have identified the significant features for the object that I’m looking for, I can take some samples and train the classifier
- If I’m looking for faces the problem can be:
  - Find all faces (binary - detection)
  - Find Bart – Lisa – Homer – Marge (recognition)
- The two scenarios are very different and involve the application of different algorithms

The classification problem



The classification problem

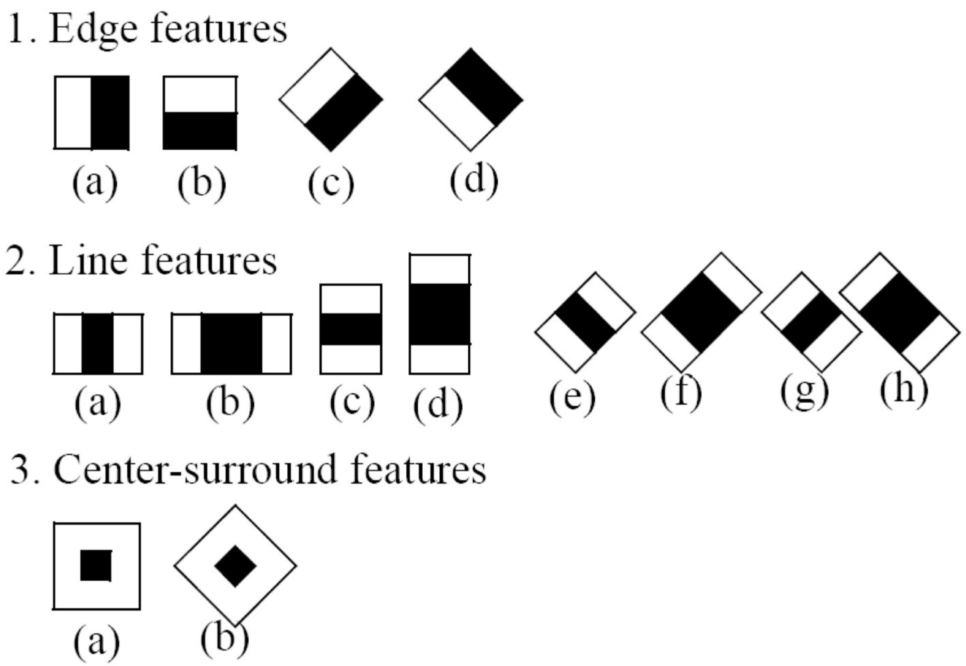


The Viola-Jones algorithm

- The Viola-Jones algorithm probably the most widespread face detector
- Available in OpenCV
- Goal:
  - Implement a robust classifier using simple features
  - Based on binary features

Features

- 14 Haar-like features
- Pretty simple features: vertical, horizontal, and diagonal



Features

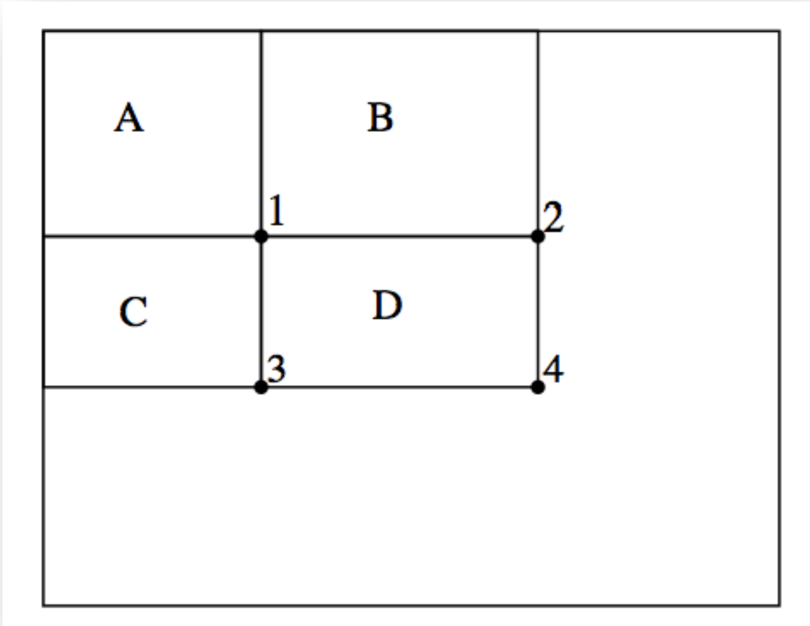
- Two/Three/Four-rectangle features
- The sum of the pixels within the white rectangles are subtracted from the sum of pixels in the black rectangles
- Rectangle features can be computed easily using an intermediate representation for the image called the *integral image*

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y')$$

- The *integral image* contains the sum of the above and left pixels w.r.t the current location $i(x,y)$

The integral image

- The sum of the values in D can be computed using four references points:
- The value in 2 is A+B
- At 3 it is A+C
- At 4 A+B+C+D
- Sum in D is 4+1-(2+3)



Recursions

- Using the following recursions, the integral image can be computed over the entire image in one single pass

$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

- s is the cumulative row sum

Classifiers

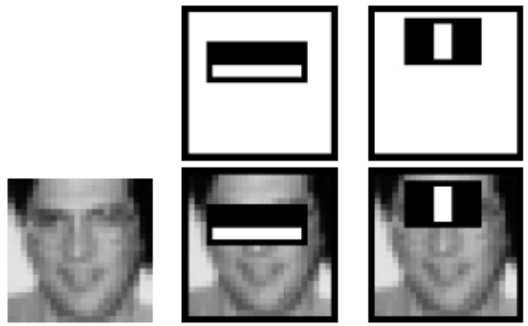
- In literature a lot of different classification algorithms have been proposed
  - Artificial neural networks
    - Feedforward
    - Radial Basis Function
    - Multi Layer Perceptron
  - Support Vector Machines

  - **Boosting**

AdaBoost

- It consists of the implementation of a strong classifier that relies on a combination of weak classifiers

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$



- *h(x)* is the weak classifier, a feature evaluated in the image
- H(x) = *sign(f(x))* is the strong classifier

$$h_i(x) = \begin{cases} 1 & \text{if } f_i > \text{threshold} \\ -1 & \text{if } f_i < \text{threshold} \end{cases}$$

$$f_i = \mathrm{Sum}(\mathrm{r}_{\mathrm{i,\,white}}) - \mathrm{Sum}(\mathrm{r}_{\mathrm{i,\,black}})$$

AdaBoost

- Given a set of points $(\mathbf{x}_i, y_i)$, $i=1 \ldots m$, with $y=\pm 1$
- Initialize weights $D_1(i) = 1/m$
- Evaluate the feature
- Find $$h_t = \underset{h_j \in H}{\operatorname{argmin}}\, \varepsilon_j = \sum_{j=1}^{m} D_t(i) I(y_i \neq h_j(x_i))$$
- I is the *Indicator function* (binary 1 or 0)
- If error **>0.5** stop
- Choose α_t

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$

It represents the sum of the weights of the misclassified samples

AdaBoost

- Update weights

$$D_{(t+1)}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

Z is a normalization factor

$$Z_t = \sum_i D_t(i)e^{-\alpha_t y_i h_t(x_i)}$$

Samples that are more difficult to classify will have higher weight in the next iteration

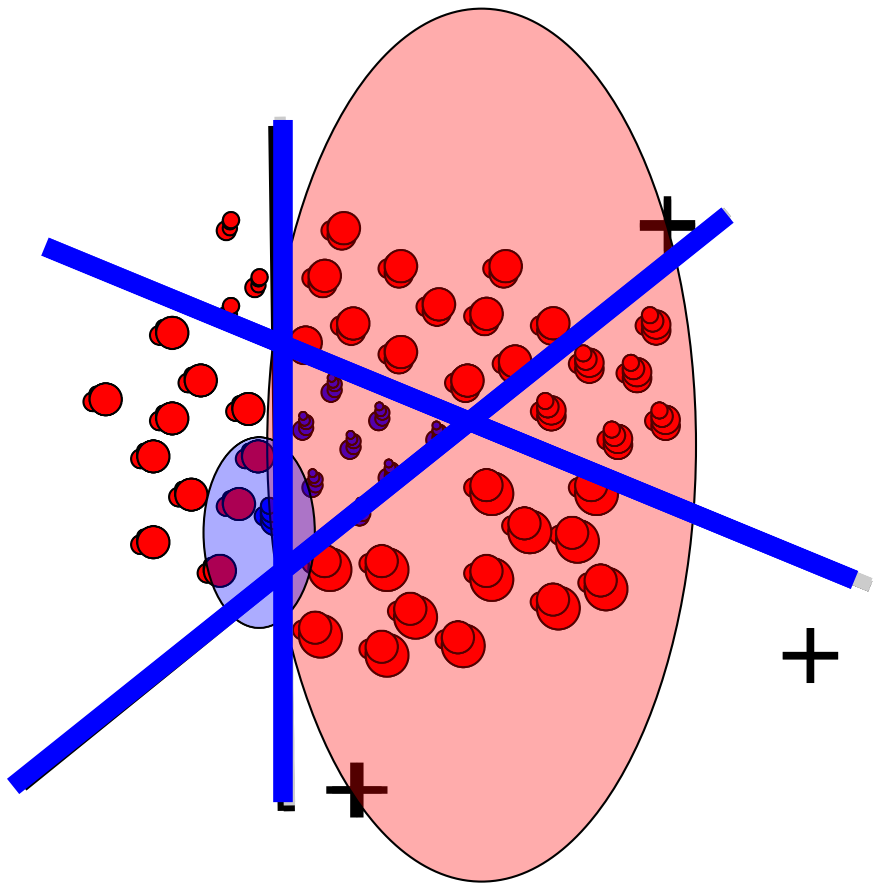
$$-\alpha_t y_i h_t(x_i) \begin{cases} < 0, y(i) = h_t(x_i) \\ > 0, y(i) \neq h_t(x_i) \end{cases}$$

- The final output of the classifier becomes:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

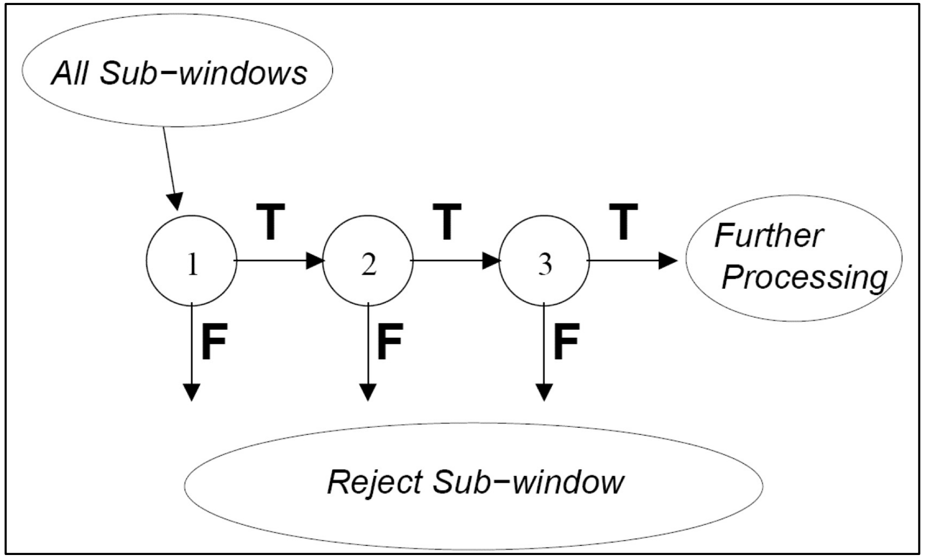
AdaBoost

- For example



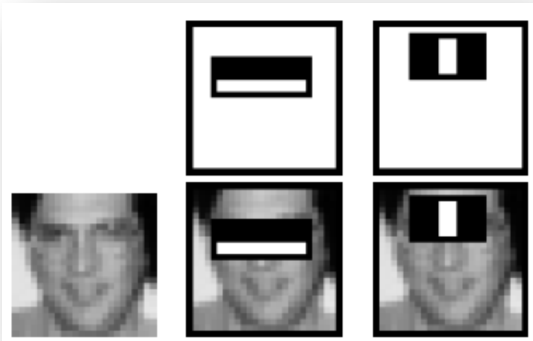
Cascade of classifiers

- In AdaBoost the combination of the weak classifiers improves the speed of the classification process
- In the algorithm proposed by Viola and Jones, classifiers are used in cascade, further reducing the computational complexity



Cascade of classifiers

- Goal: quickly remove false negatives and focus on the positive samples
- In face detection, basic features can be used to exclude non-faces



- The output of the first classifier is used to trigger the second one
- Each stage is trained using AdaBoost
- At each stage the negatives are rejected

Training and testing

- In the cascade, binary features are evaluated on training images of equal size (24x24pix)
- Classification:
  - During the training stage, the boundaries of the classifiers are learned
  - During the testing phase, the learned classifiers are used to evaluate *unknown* samples
- Images are not scaled at 24x24, so a multi-scale analysis must be conducted
- Scaling is performed at the feature level, not image → fast by considering the type of detector [binary + integral image]
- In case of multiple detections at different scales, windows are averaged

Complexity and concluding remarks

- Complexity depends on :
  - the number of classifiers in the cascade (mainly in training)
  - The number of levels in the multi-scale analysis
  - The sampling distance between the windows
- Images used for training/testing could be acquired in different environmental conditions, so light is in general different
- A normalization would be highly desirable
- Here variance normalization in the sub-window

$$\sigma^2 = m^2 - \frac{1}{M}\sum_{x \in M} x^2$$