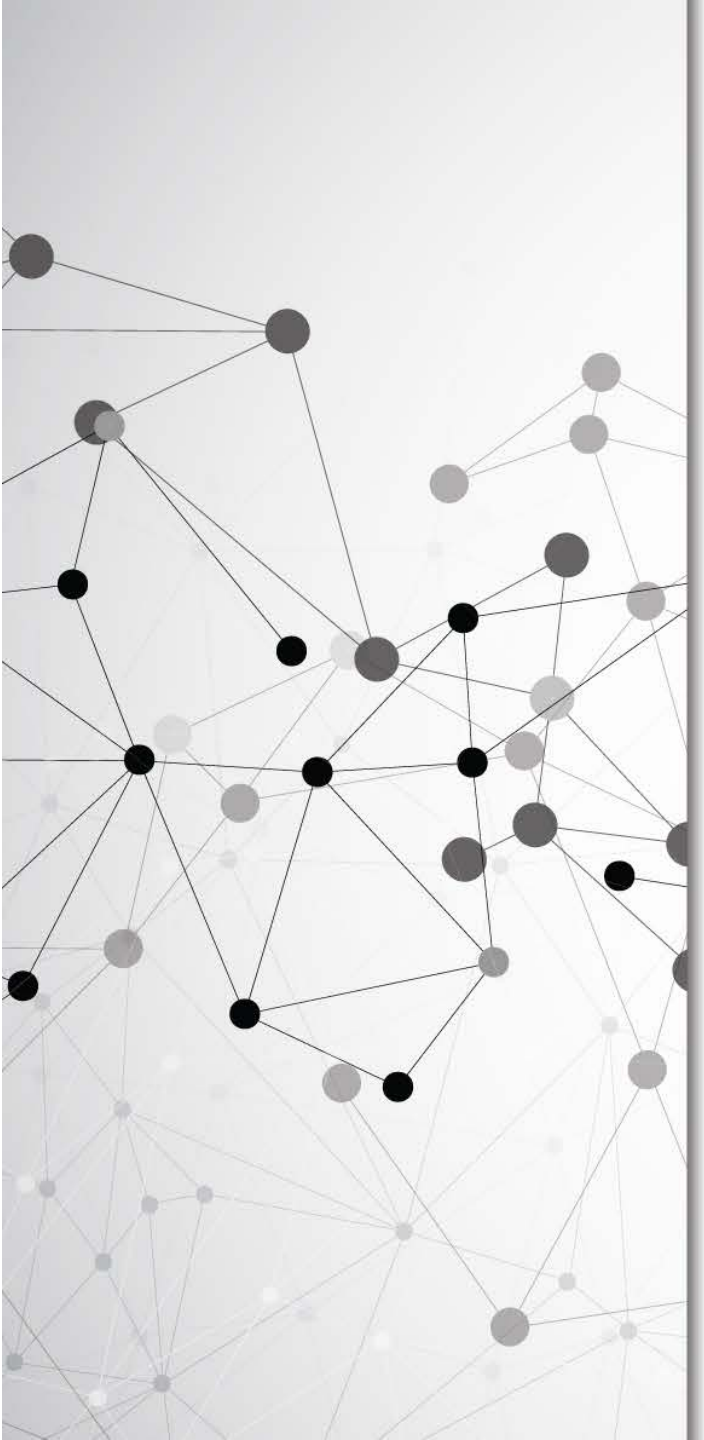


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

# Real-Time Face Detection

**EE4208**

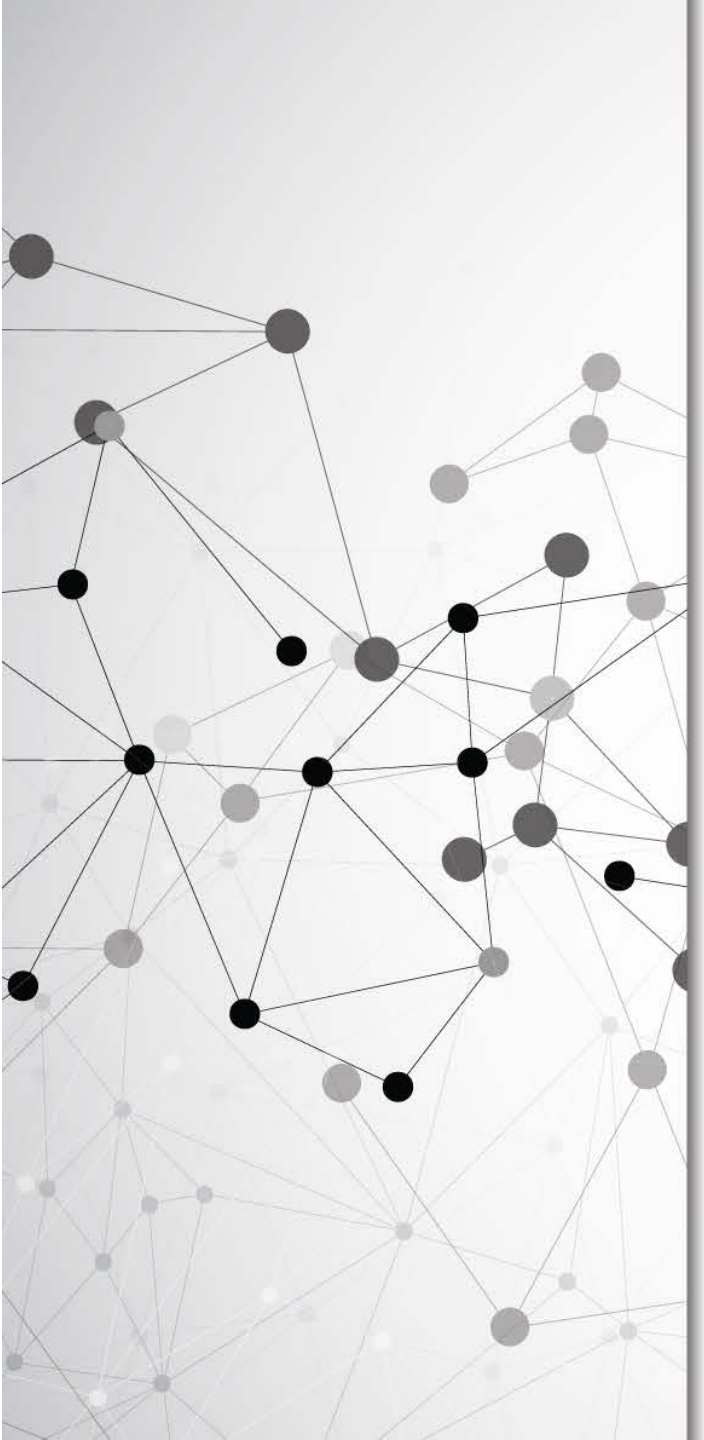
School of Electrical and Electronic Engineering

Dr. Wang Han
Office: S2-B2b-49
Email: hw@ntu.edu.sg
Phone: 6790-4506

# Learning Objectives

By the end of this topic, you should be able to:

- Undestand Haar-like features.
- Training of weak classifiers.
- Ada-boost for training of strong classfier.
- Apply a cascade of homogeneous classifiers.



# Introduction

Reference:

Viola, P., Jones, M. (2001). Robust Real-time Object Detection. *Second International Workshop on Statistical and Computational Theories on Vision – Modeling, Learning, Computing and Sampling*. Retrieved 18 September 2017, from https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-IJCV-01.pdf.

# About the Paper

- Presented in 2001 by Paul Viola and Michael Jones (published 2002 – IJCV)
- Specifically demonstrated (and motivated by) the face detection task.
- Placed a strong emphasis upon speed optimisation.
- Allegedly, was the first real-time face detection system.
- Was widely adopted and re-implemented.
- Intel distributes this algorithm in a computer vision toolkit (OpenCV).

Paul viola

Michael Jones

# Introduction

Allegedly, was the first real-time face detection system.

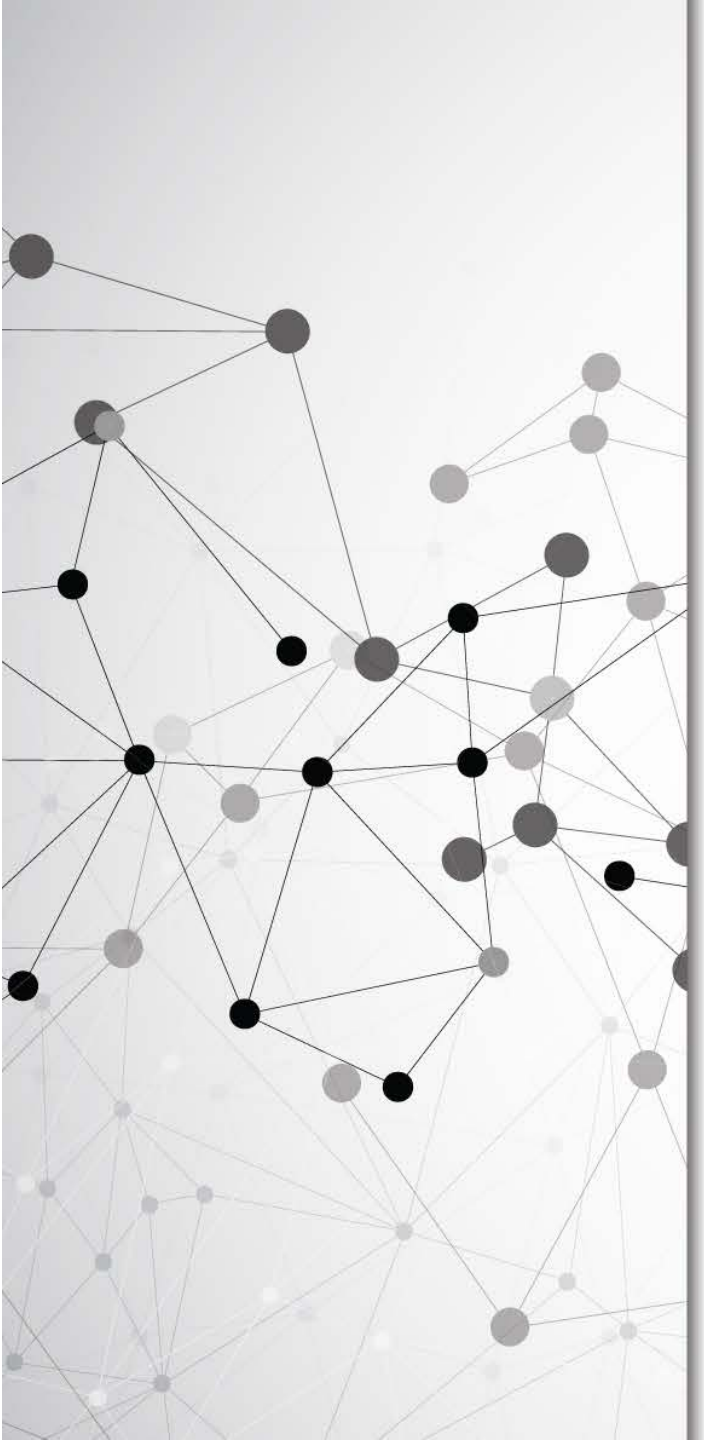
Was widely adopted and re-implemented.

Intel distributes this algorithm in a computer vision toolkit (OpenCV)

Actual output of Intel's implementation

## Requirements

- Object detection task:
  - Given a set of images, find regions in these images which contain instances of a certain kind of object.
  - Disregard various orientations, color and frame-by-frame consistency.
- Real time performances:
  - 15 fps on 384 by 288 pixel images, on a conventional 700 MHz Intel Pentium III.
- Robust (generic) learning algorithm.



# Framework scheme

# Framework scheme

## Framework consists of:

**Trainer**

Supplied with:

**Positive samples** – images containing the object.

**Negative samples** – images not containing the object.

Creates a:

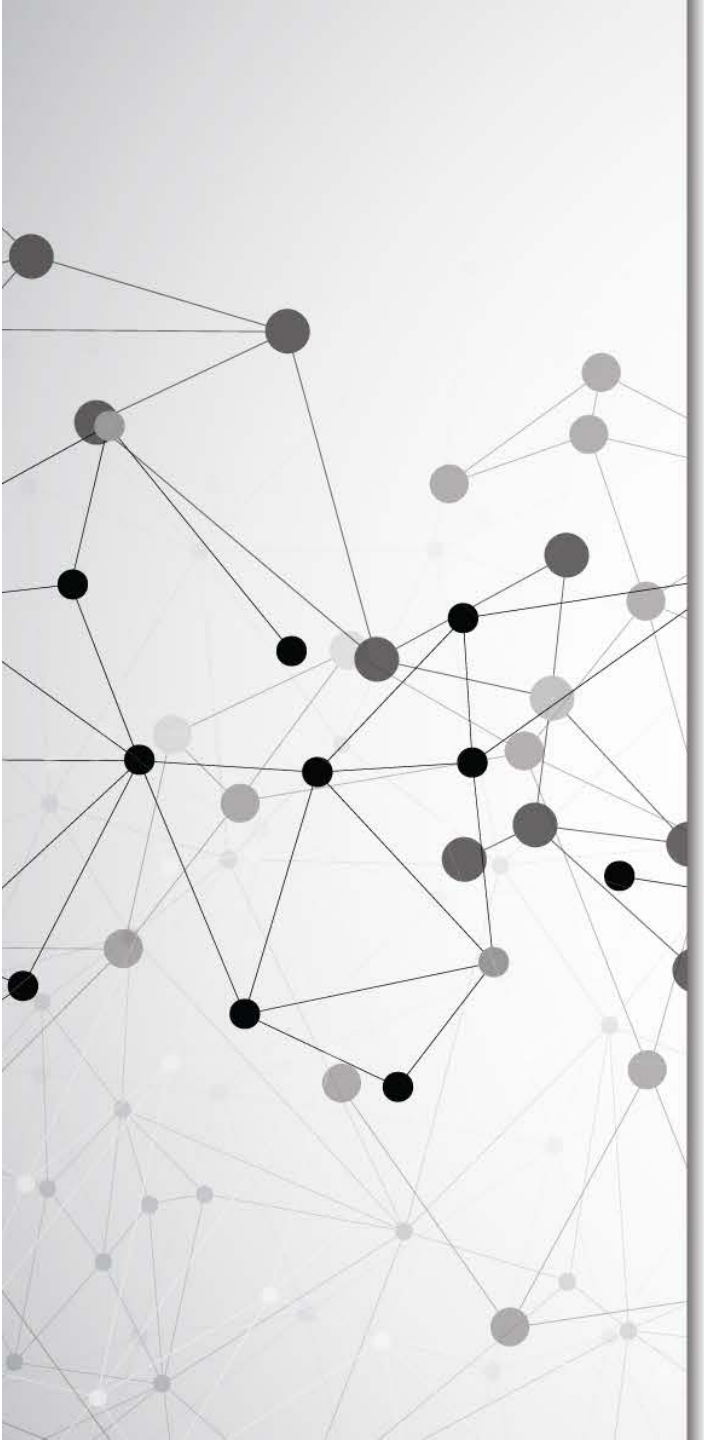
**Final classifier** –
A lengthy process, to be calculated offline.

**Detector**

Utilises the final classifier across a given input image.

# Abstract Detector

1. Iteratively sample image windows.
2. Operate Final Classifier on each window, and mark accordingly.
3. Repeat with larger window.

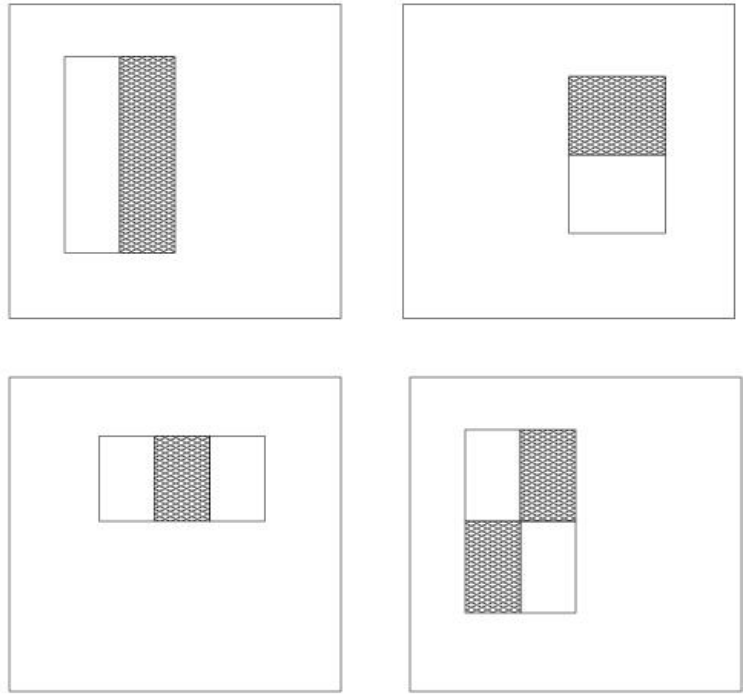


# Features

# Features

We describe an object using  simple functions also called: **Harr-like features**.

Given a sub-window, the feature function calculates a brightness differential.



For example:

The value of a two-rectangle feature is the difference between the sum of the pixels within the two rectangular regions.

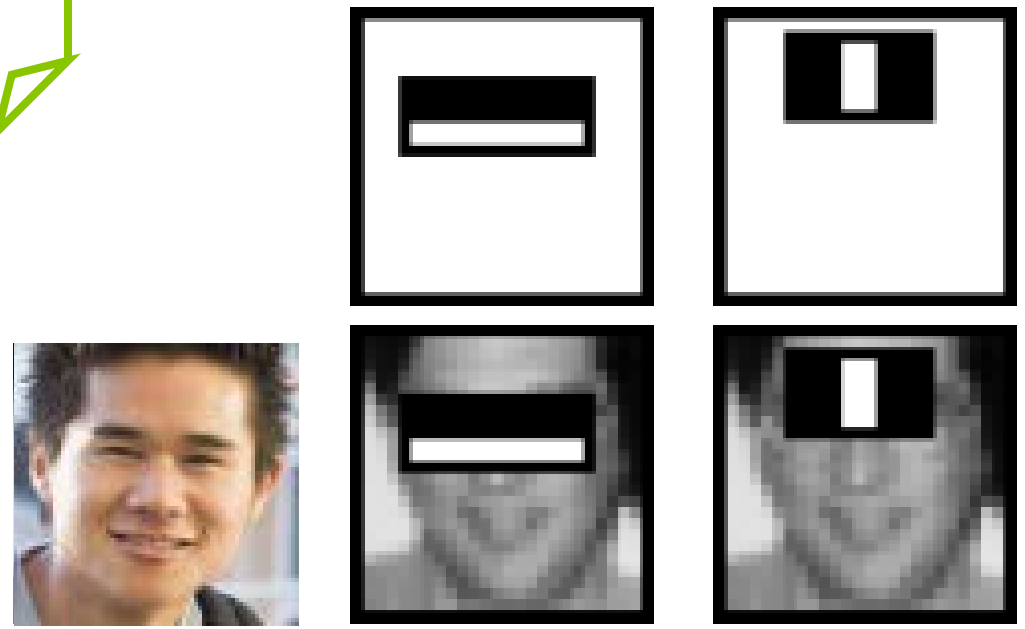
# Features: Example

Faces share many similar properties which can be represented with Haar-like features.

For example:

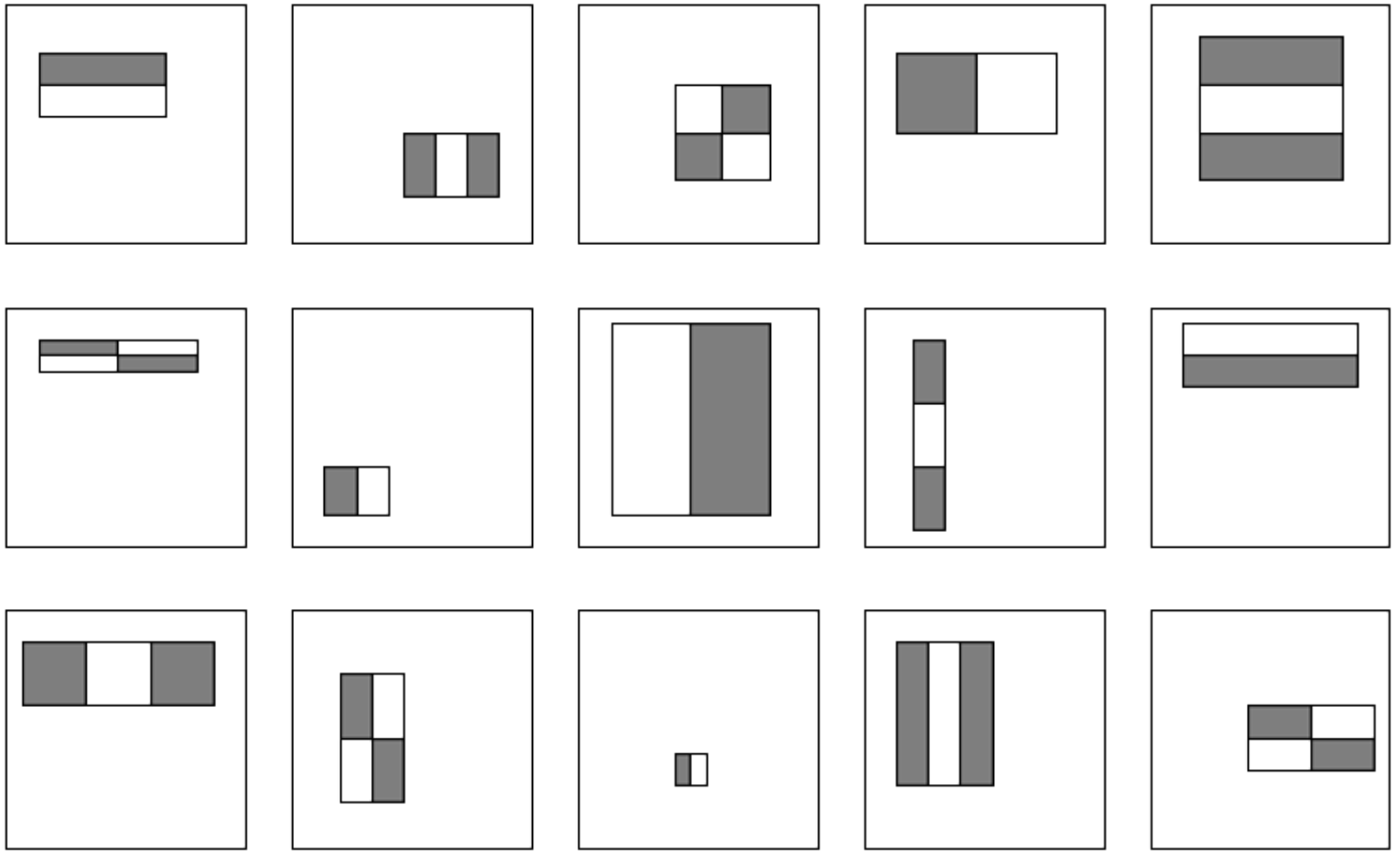
It is easy to notice that:

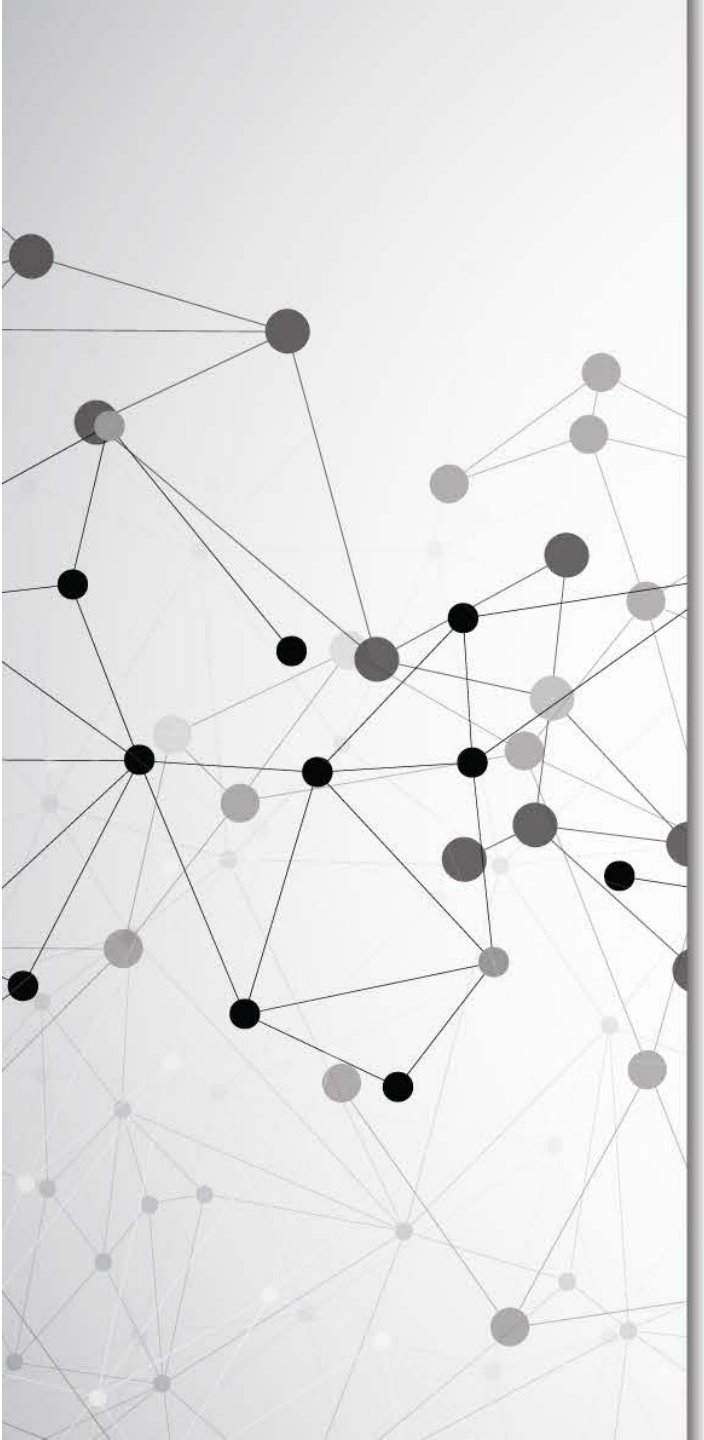
- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.



# Feature Selection

For a $24 \times 24$ detection region, the number of possible rectangle features is ~180,000!





# Challenges

# Challenges

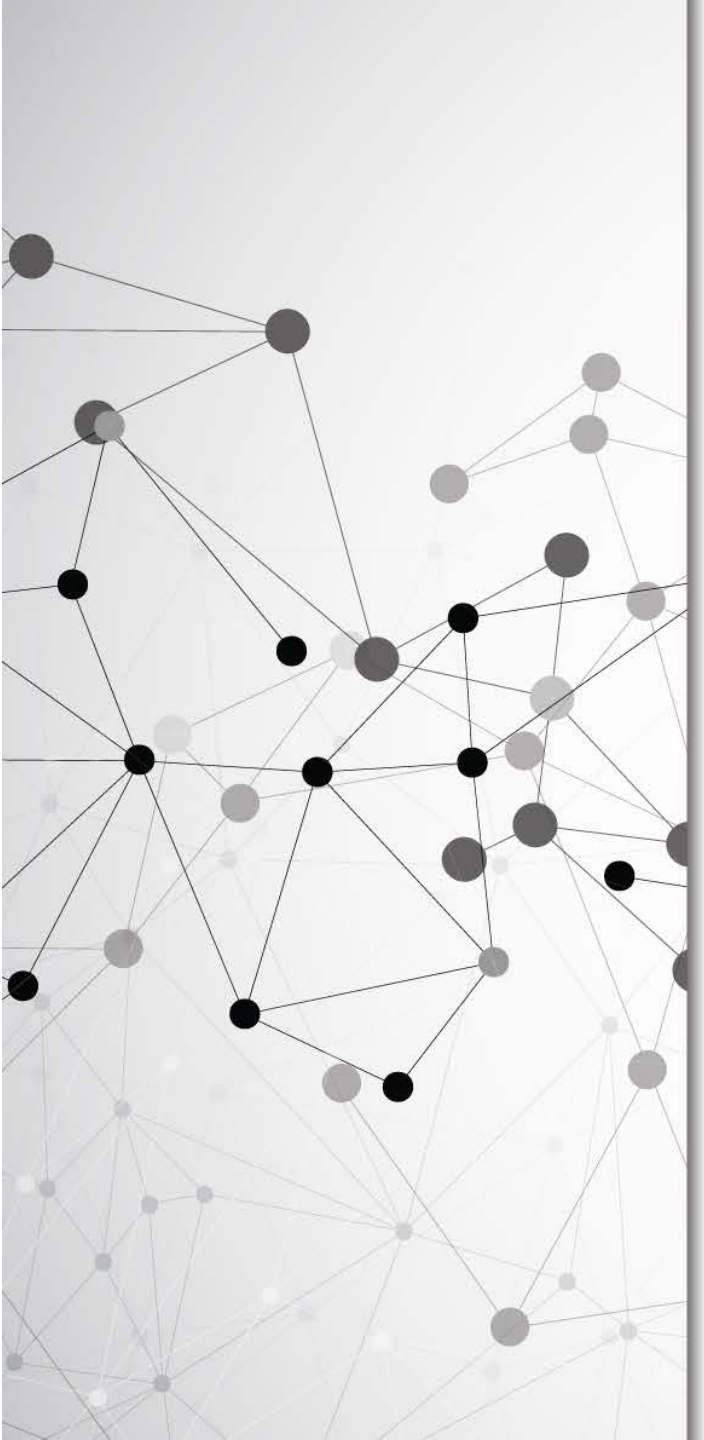
**Three challenges ahead:**

How can we evaluate features quickly?

- Feature calculation is critically frequent.
- Image scale pyramid is too expensive to calculate.

How do we obtain the best representing features possible?

How can we refrain from wasting time on image background (i.e. non-object)?



# Integral Image

# Introducing Integral Image

**Definition:**

The integral image at location $(x, y)$, is the sum of the pixel values above and to the left of $(x, y)$, inclusive.

**Note:**

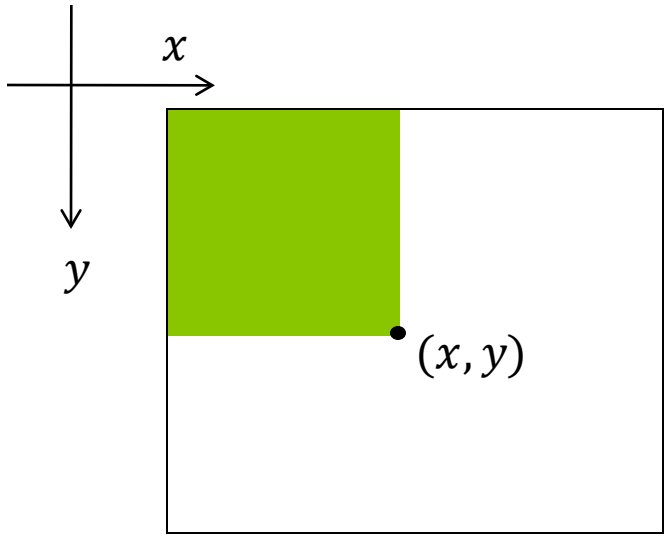
- We can calculate the integral image representation of the image in a single pass.
- $s(x, y)$ is the cumulative column sum

Formal definition:

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

Recursive definition:

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



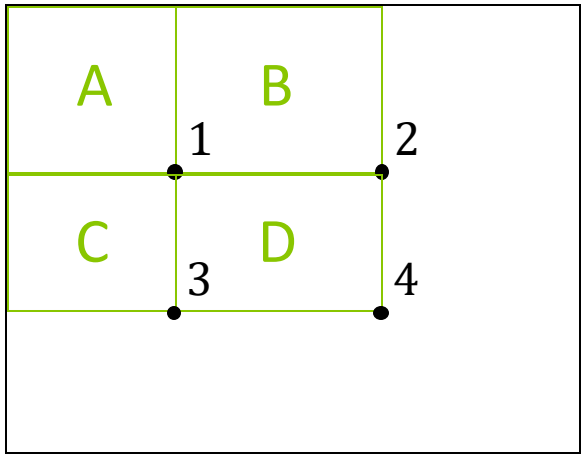
# Rapid Evaluation of Rectangular Features

Using the integral image representation one can compute the value of any rectangular sum in constant time.

For example:

The integral sum inside rectangle D can be computed as:

$$ii(4) + ii(1) - ii(2) - ii(3)$$




As a result: two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively.

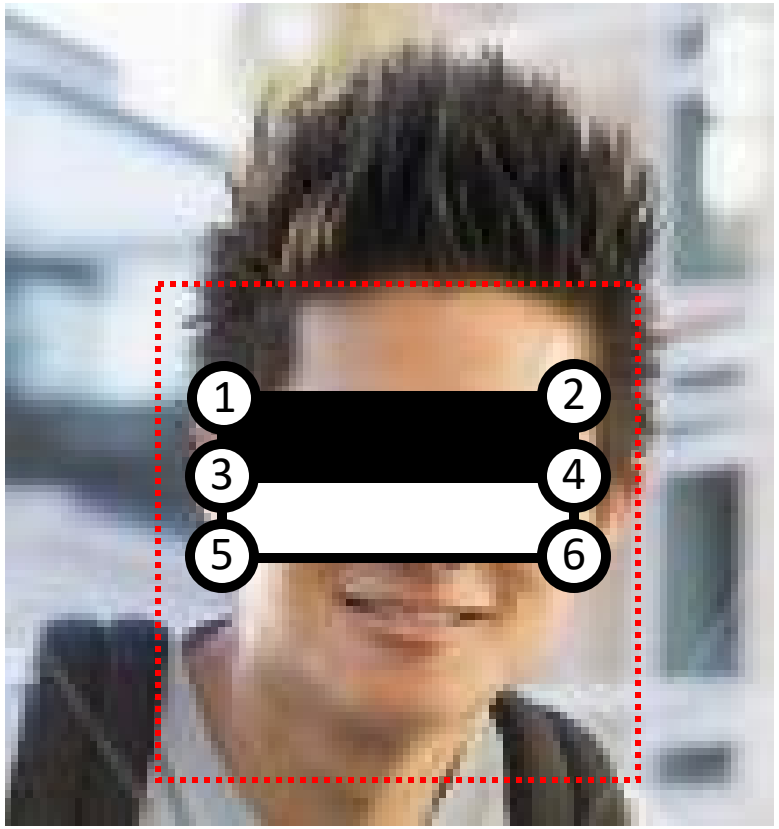
Now that's fast!

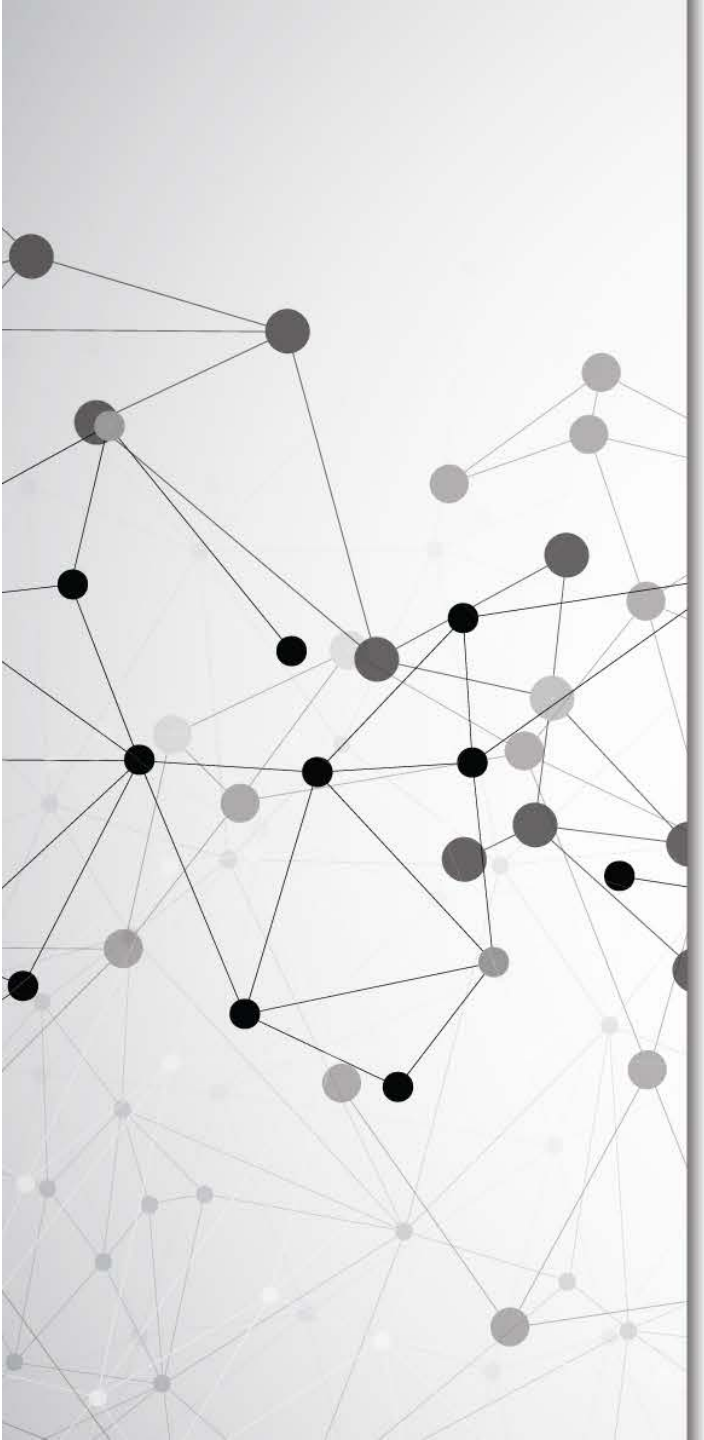
# Scaling

Integral image enables us to evaluate all rectangle sizes in constant time.

Therefore, no image scaling is necessary.

Scale the rectangular features instead!





# Boosting

# Feature Selection

Given a feature set and labeled training set of images, we create a strong object classifier.

However, we have 45,396 features associated with each image sub-window, hence the computation of all features is computationally prohibitive.

**Hypothesis:**

A combination of only a small number of **discriminant** features can yield an effective classifier.

**Variety** is the key here – if we want a small number of features – we must make sure they compensate each other's flaws.

# Boosting

Boosting is a machine learning meta-algorithm for performing supervised learning.

Creates a “strong” classifier from a set of “weak” classifiers.

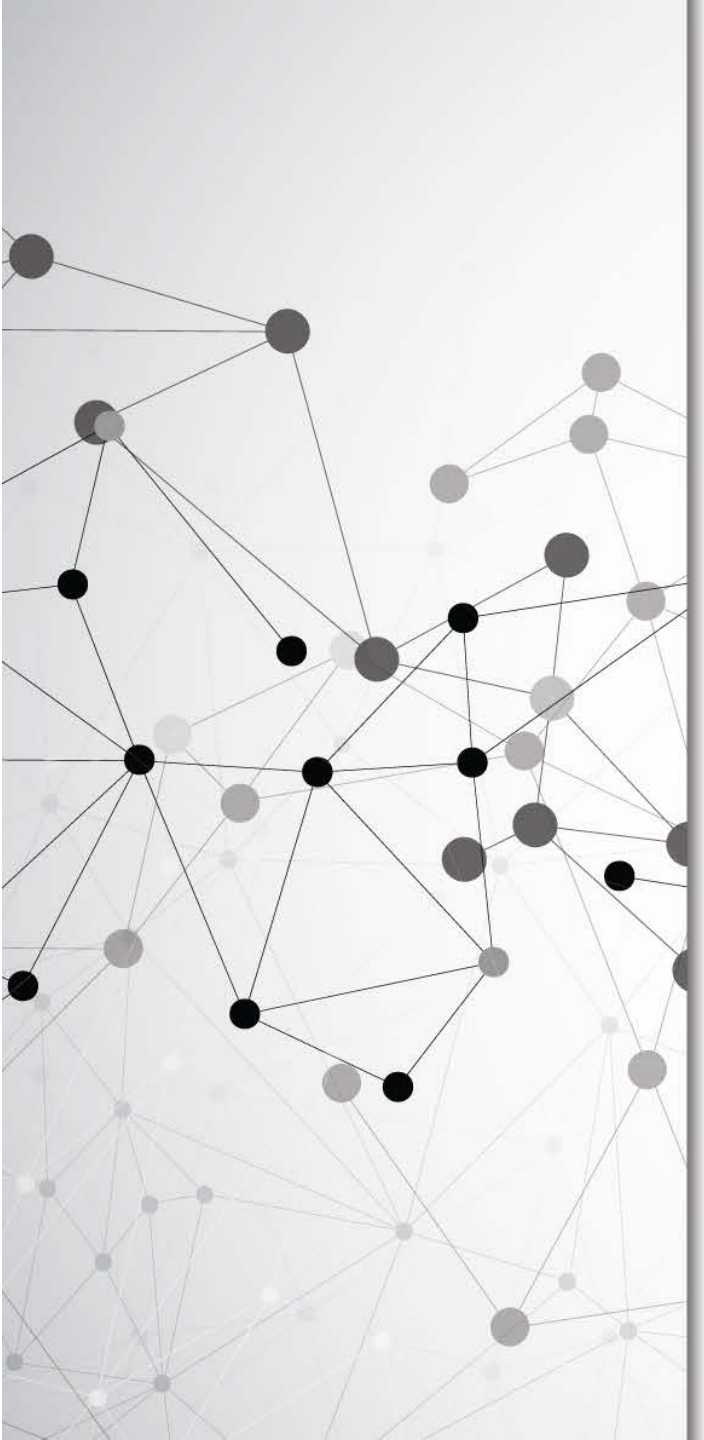
**Definitions:**

**“weak” classifier**

Has an error rate <0.5 (i.e. a better than average advice).

**“strong” classifier**

Has an error rate of ε (i.e. our final classifier).



# AdaBoost

# AdaBoost

AdaBoost is a boosting algorithm for searching out a small number of good classifiers which have **significant variety**.

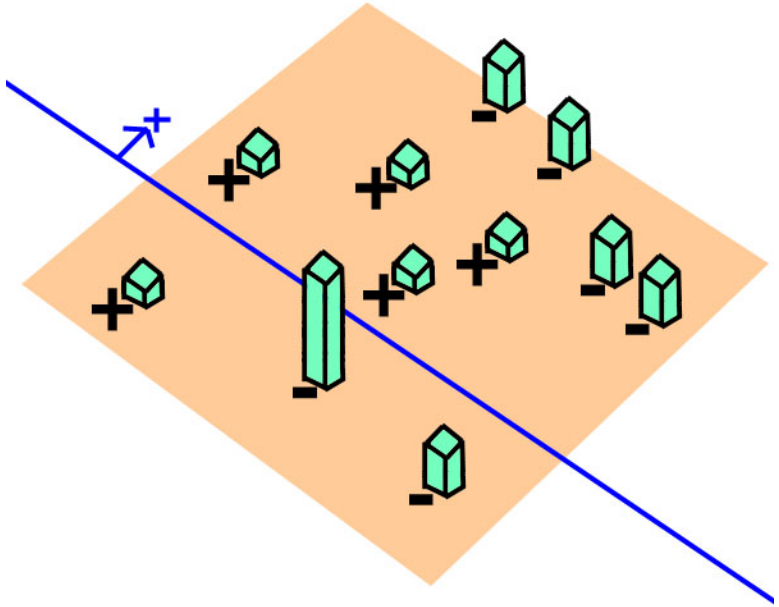
AdaBoost accomplishes this by endowing misclassified training examples with more weight (thus enhancing their chances to be classified correctly next).

The weights tell the learning algorithm the importance of the example.

Stands for "Adaptive boost".

# AdaBoost example

- Adaboost starts with a uniform distribution of “weights” over training examples.
- Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)
- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.



$$h_{\text{strong}}(\mathbf{x}) = \begin{cases} 1 & \alpha_1 h_1(\mathbf{x}) + \ldots + \alpha_n h_n(\mathbf{x}) \geq \frac{1}{2}\left(\alpha_1 + \ldots + \alpha_n\right) \\ 0 & \text{otherwise} \end{cases}$$

*Slide taken from a presentation by Qing Chen, Discover Lab, University of Ottawa*

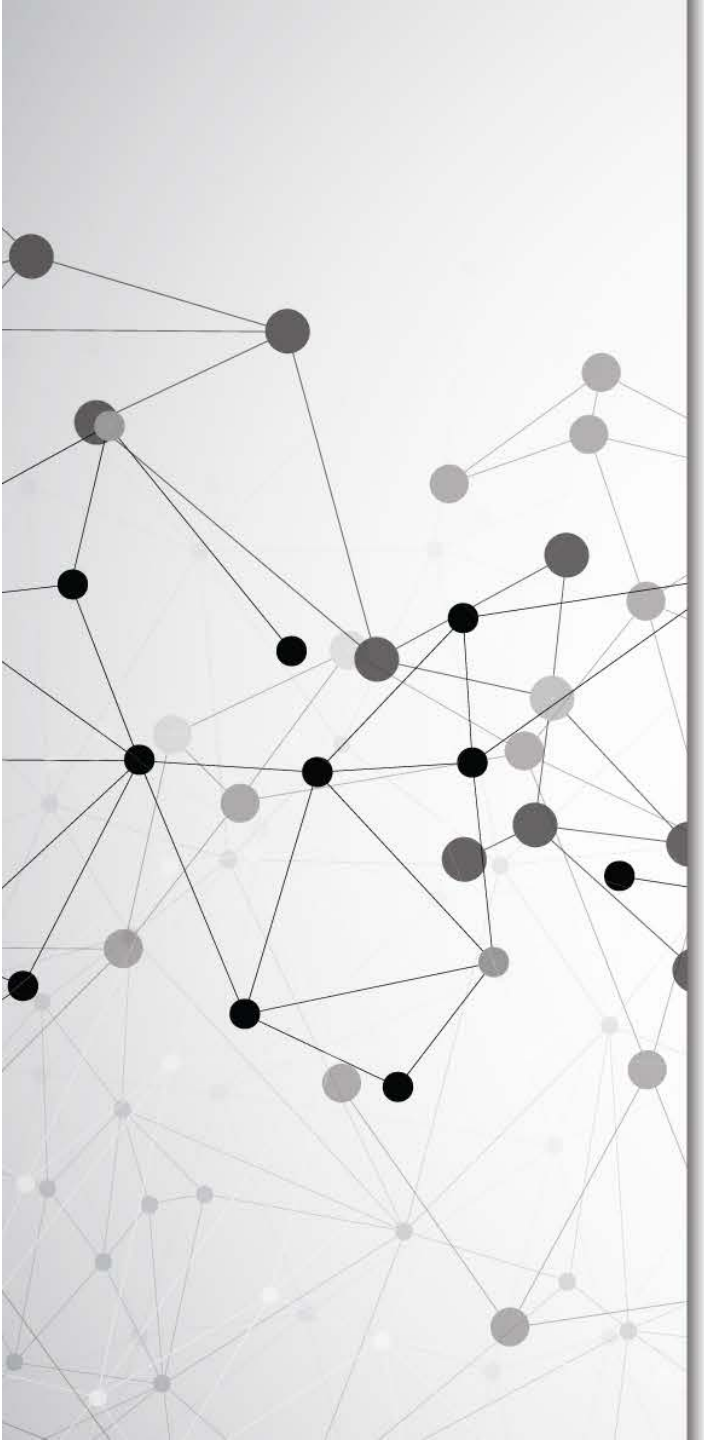
# Back to Feature Selection

We use a variation of AdaBoost for aggressive feature selection.

Basically similar to the previous example.

Our training set consists of positive and negative **images**.

Our simple classifier consists of a single **feature**.



# Simple Classifier

# Simple Classifier

A simple classifier depends on a single feature.

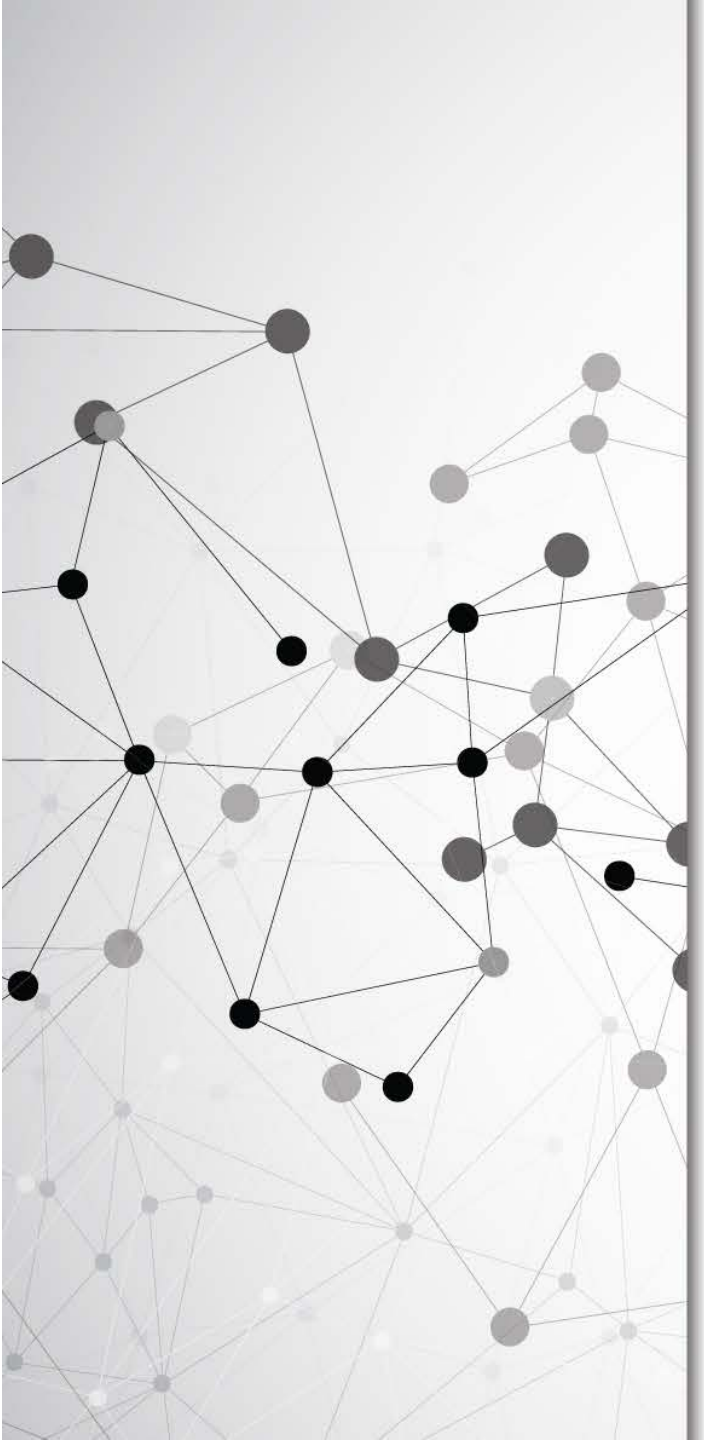
Hence, there are 45,396 classifiers to choose from.

For each classifier, we set an **optimal threshold** such that the minimum number of examples are misclassified.

value of rectangle feature

$$h_j = \begin{cases} -1, f_j(x) < \theta_j \\ +1, else; \end{cases}$$

$h_j$ : simple classifier
$f_j$ : feature
$\theta_j$ : threshold, selected optimally from training samples



# Feature Selection Pseudo-Code

# Feature Selection Pseudo-Code

Given example images $(x_1, y_1), \ldots, (x_m, y_m)$ where $y_1 = \{-1, +1\}$

For example: $(x_1, -1) \rightarrow$ negative example, $(x_9, +1) \rightarrow$ positive example

- Initialise $D_1(i) = 1/m, i = 1, \ldots, m$
- For $t = 1, \ldots, T$: ($T$ is the total weak classifiers)
  - Find the classifier $h_t$ that minimizes the error with respect to the distribution $D_t$:

$$h_t = \arg\min \varepsilon_j \quad \text{where} \quad \varepsilon_j = \sum_{i=1}^{m} D_t(i)\left[y_i \neq h_j(x_i)\right]$$

($\varepsilon_j$ is the sum of $D_t(i)$ for all mismatched $h_i$; $h_t$ is selected with the smallest $\varepsilon_j$.)

  - If $\varepsilon_t \geq 0.5$ then stop;

# Feature Selection Pseudo-Code

o Choose $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$

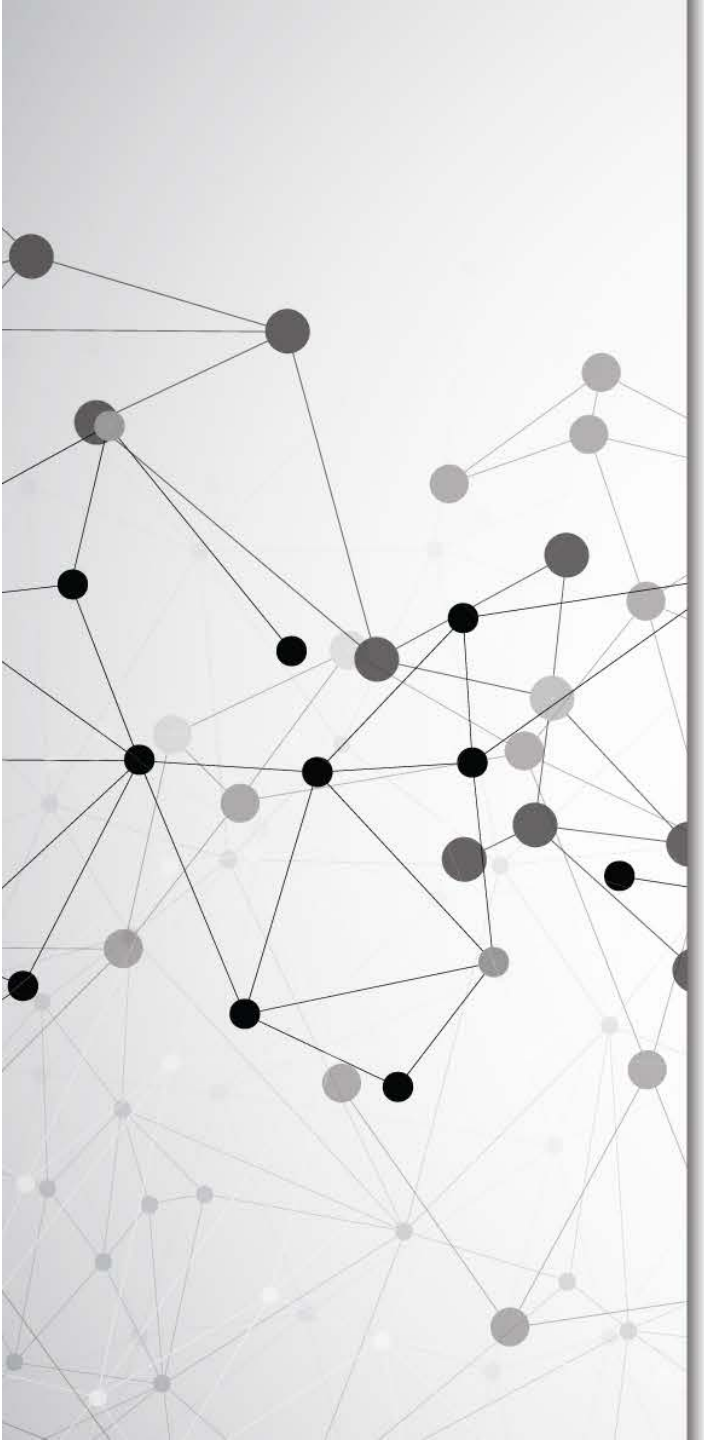
typically where $\varepsilon_t$ is the weighted error rate of classifier $h_t$.

o Update $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t \cdot y_i \cdot h_t(x_i))}{Z_t}$

where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a probability distribution, i.e. sum one over all $x$).

- Output the final classifier

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



# 200 Feature Face Detector

# 200 Feature Face Detector

We can now train a classifier as accurate as we desire.

By increasing the number of features per classifier, we:

- Increase detection accuracy.
- Decrease detection speed.

Experiments showed that a 200 feature classifier makes a good face detector:

- Takes 0.7 seconds to scan an 384 by 288 pixel image.

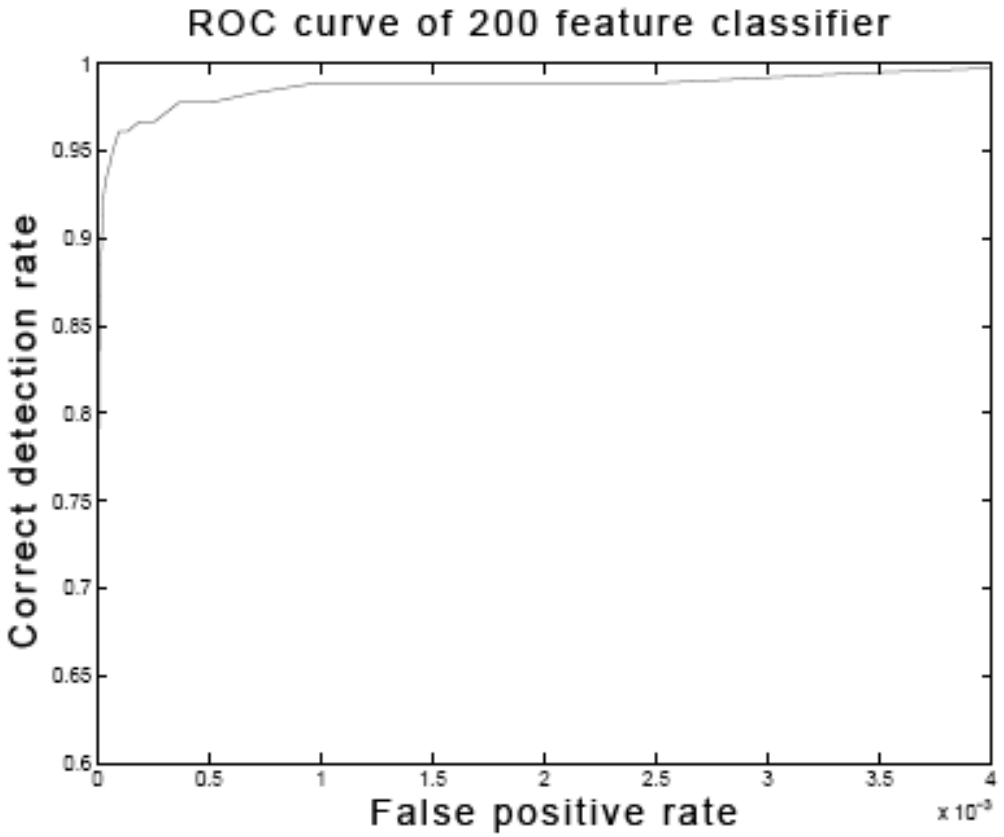
**Problem:**

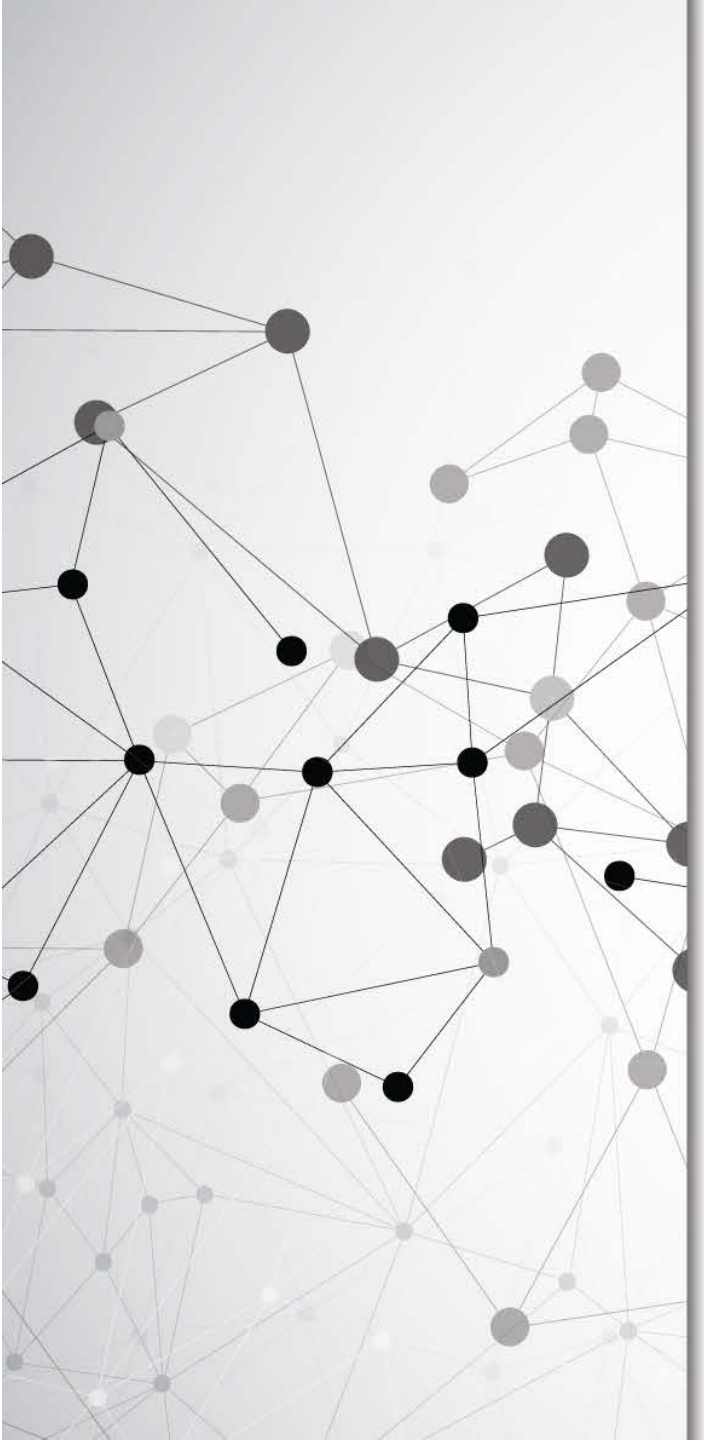
Not real time! (At most 0.067 seconds needed).

# Performance of 200 Feature Face Detector

- The Receiver Operating Characteristic (ROC) curve of the constructed classifies indicates that a reasonable detection rate of 0.95 can be achieved while maintaining an extremely low false positive rate of approximately $10^{-4}$.

- By varying the threshold of the final classifier one can construct a two-feature classifier which has a detection rate of 1 and a false positive rate of 0.4.

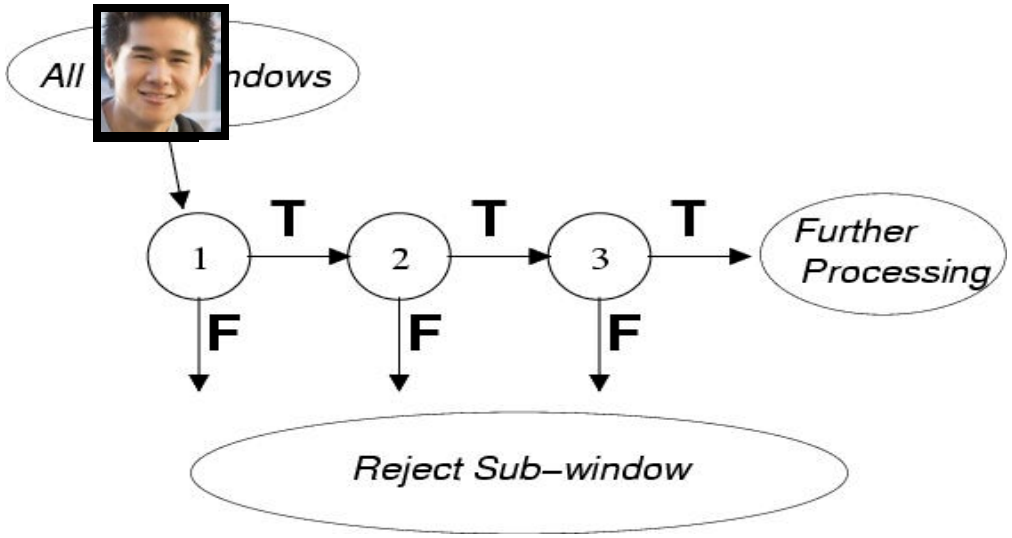




# Cascaded Classifier

# The attentional cascade

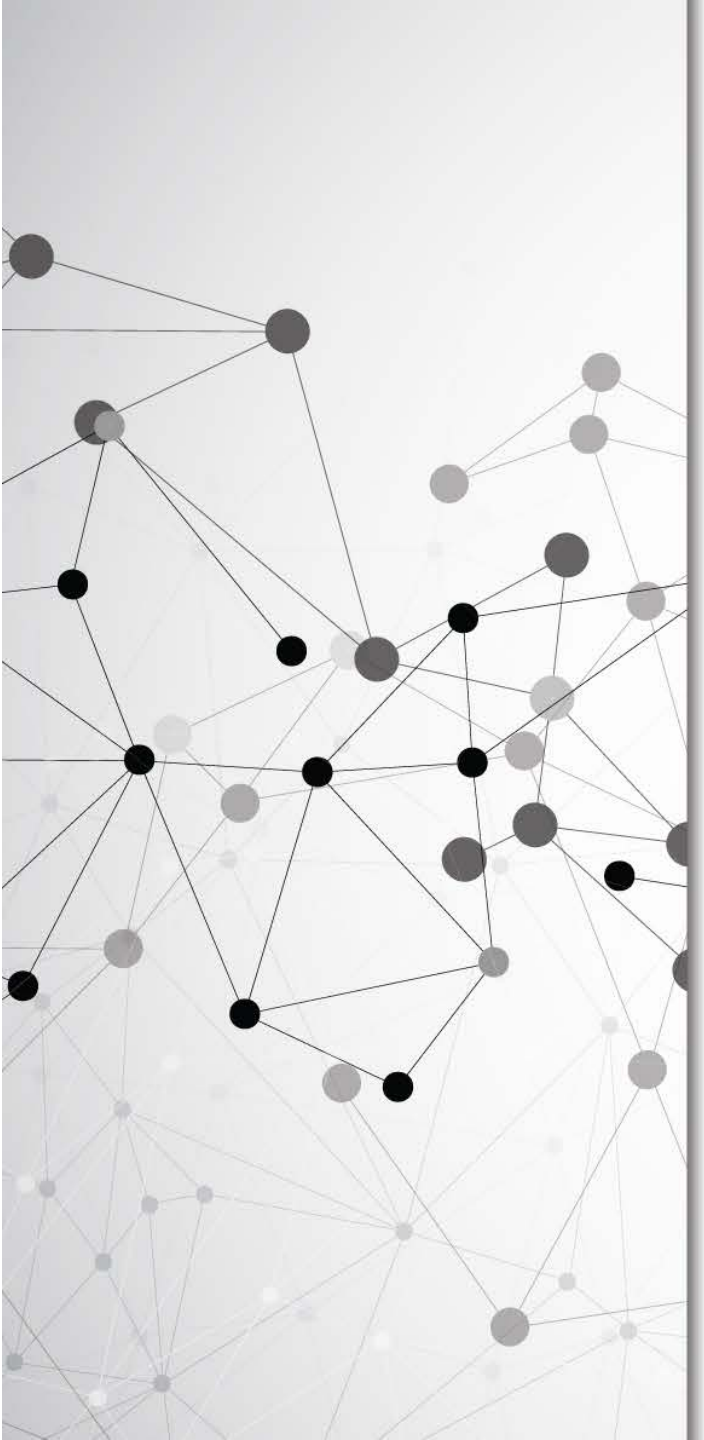
- Overwhelming majority of windows are in fact negative.
- Simpler, boosted classifiers can reject many of negative sub-windows while detecting all positive instances.
- A cascade of gradually more complex classifiers achieves good detection rates.
- Consequently, on **average**, much fewer features are calculated per window.



# Experiments - Dataset for training

4916 positive training example were hand picked, aligned, normalised, and scaled to a base resolution of $24 \times 24$.

10,000 negative examples were selected by randomly picking sub-windows from 9500 images which did not contain faces.



# Summary

# Summary

Key points discussed in this topic:

- The face detection task is a general object detection method.
- Using the integral image representation and simple rectangular features eliminate the need of expensive calculation of multi-scale image pyramid.
- Simple modification to AdaBoost gives a general technique for efficient feature selection.
- A general technique for constructing a cascade of homogeneous classifiers is presented, which can reject most of the negative examples at early stages of processing thereby significantly reducing computation time.