Detecting Faces

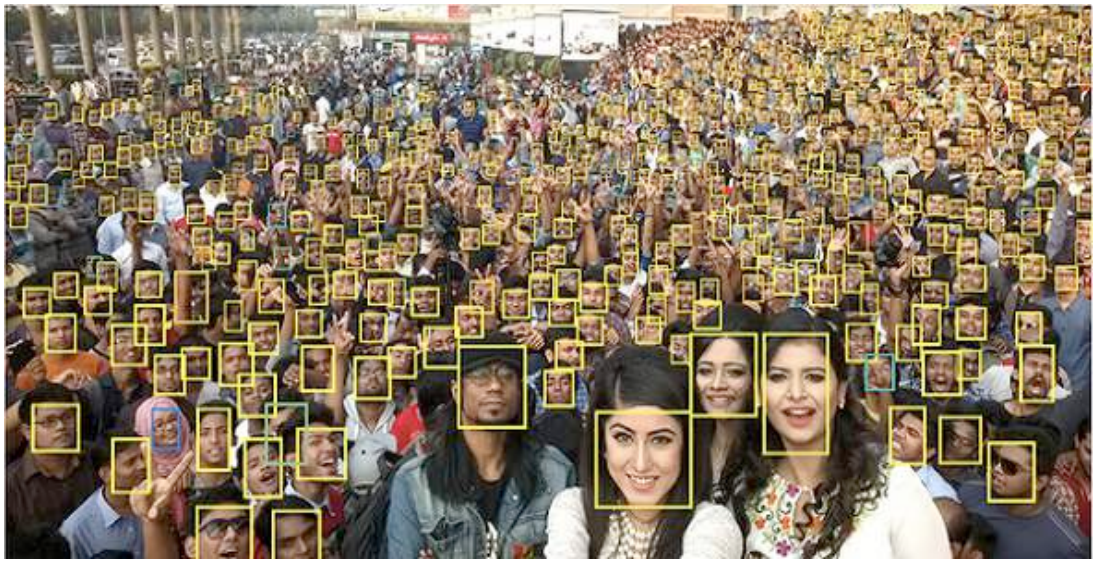
Marcello Pelillo
University of Venice, Italy

Image and Video Understanding
a.y. 2018/19

Face Detection

Identify and locate human faces in images regardless of their:

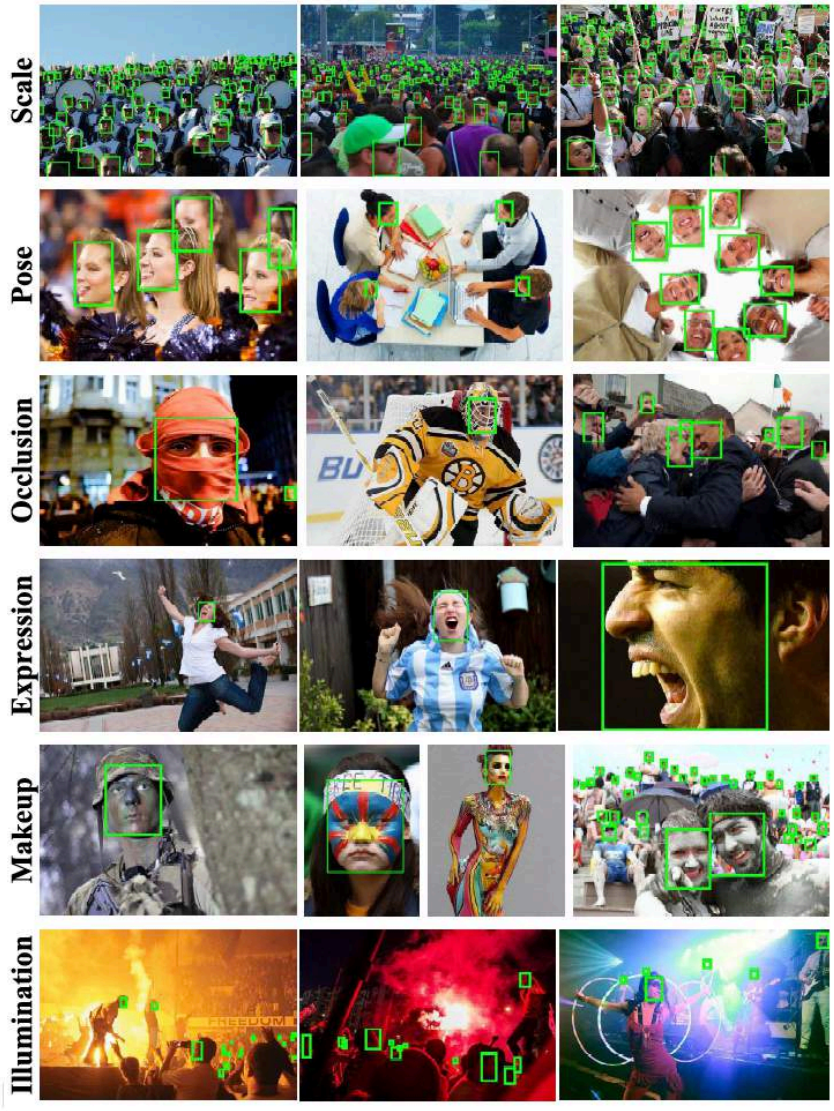
- position
- scale
- pose (out-of-plane rotation)
- orientation (in-plane rotation)
- illumination



A Few Figures

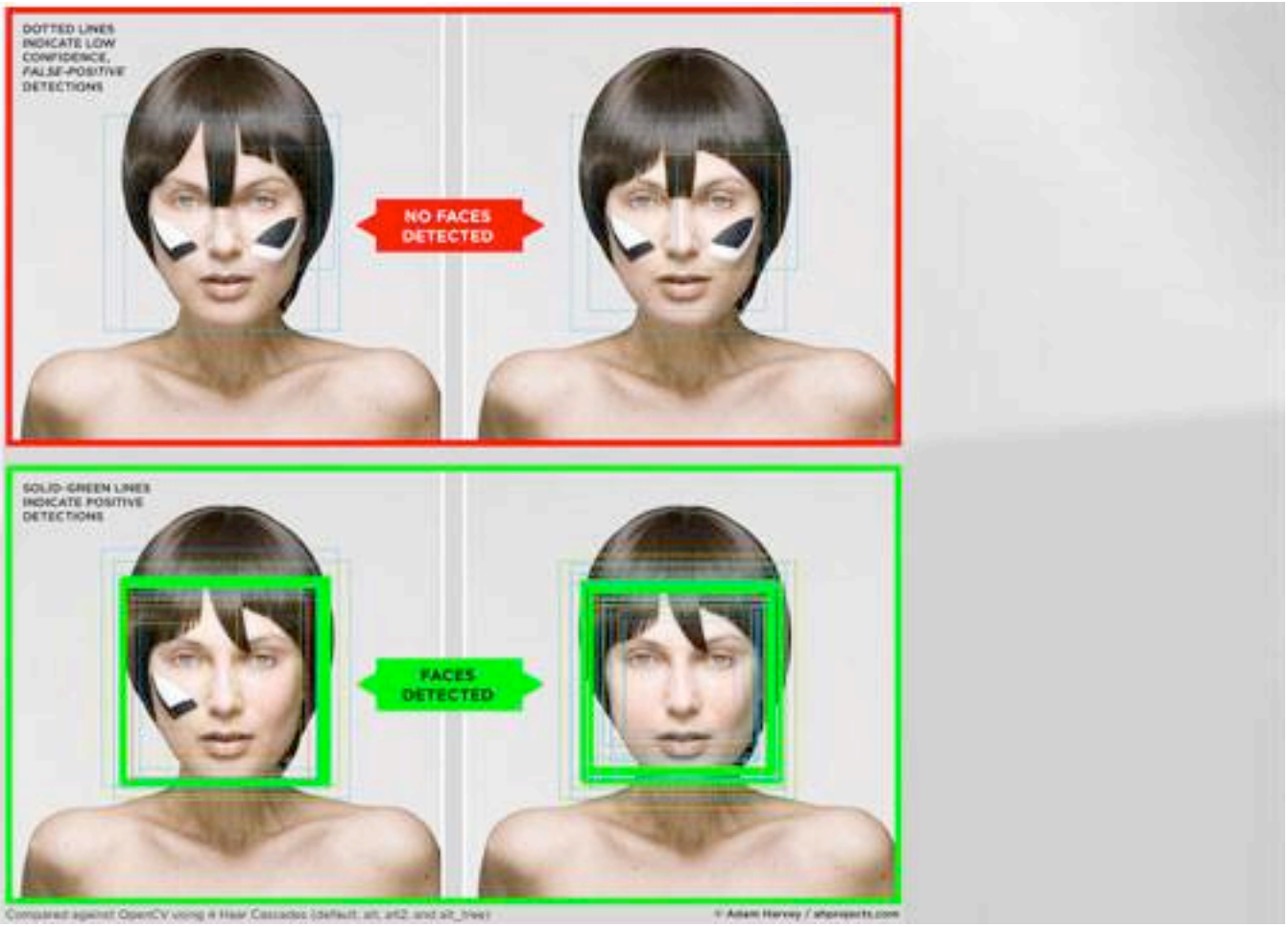
- Consider a thumbnail 19 × 19 face pattern
- 256^{361} possible combination of gray values
  - $256^{361} = 2^{8\times361} = 2^{2888}$
- Total world population (as of 2018):
  - $7{,}600{,}000{,}000 \cong 2^{33}$
- 87 times more than the world population!
- Extremely high dimensional space!

Why Is Face Detection Difficult?



Why Is Face Detection Difficult?

Fooling Face-Detection Algorithms



Fooling Face-Detection Algorithms

CVDAZZLE

Aug 22, 2017

An open source face detection toolkit is being developed to evaluate looks using haar, dlib, ssd, and yolo face detectors.

In the meantime, everything here is designed for the OpenCV haarcascade face detection algorithm

LOOKS

STYLE TIPS

RELATED PROJECTS

COLLABORATIONS

PRESS

REFERENCES

CONTACT

New Looks

by Coreana Museum of Art + Soobin Academy + G-square Model Academy

Anti Face

This face is unrecognizable to several state-of-art face detection algorithms.

Camouflage from face detection.

https://cvdazzle.com/

Related Problems

Face localization: Determine the image position of a single face (assumes input image contains only one face)

Facial feature extraction: Detect the presence and location of features such as eyes, nose, nostrils, eyebrow, mouth, lips, ears, etc

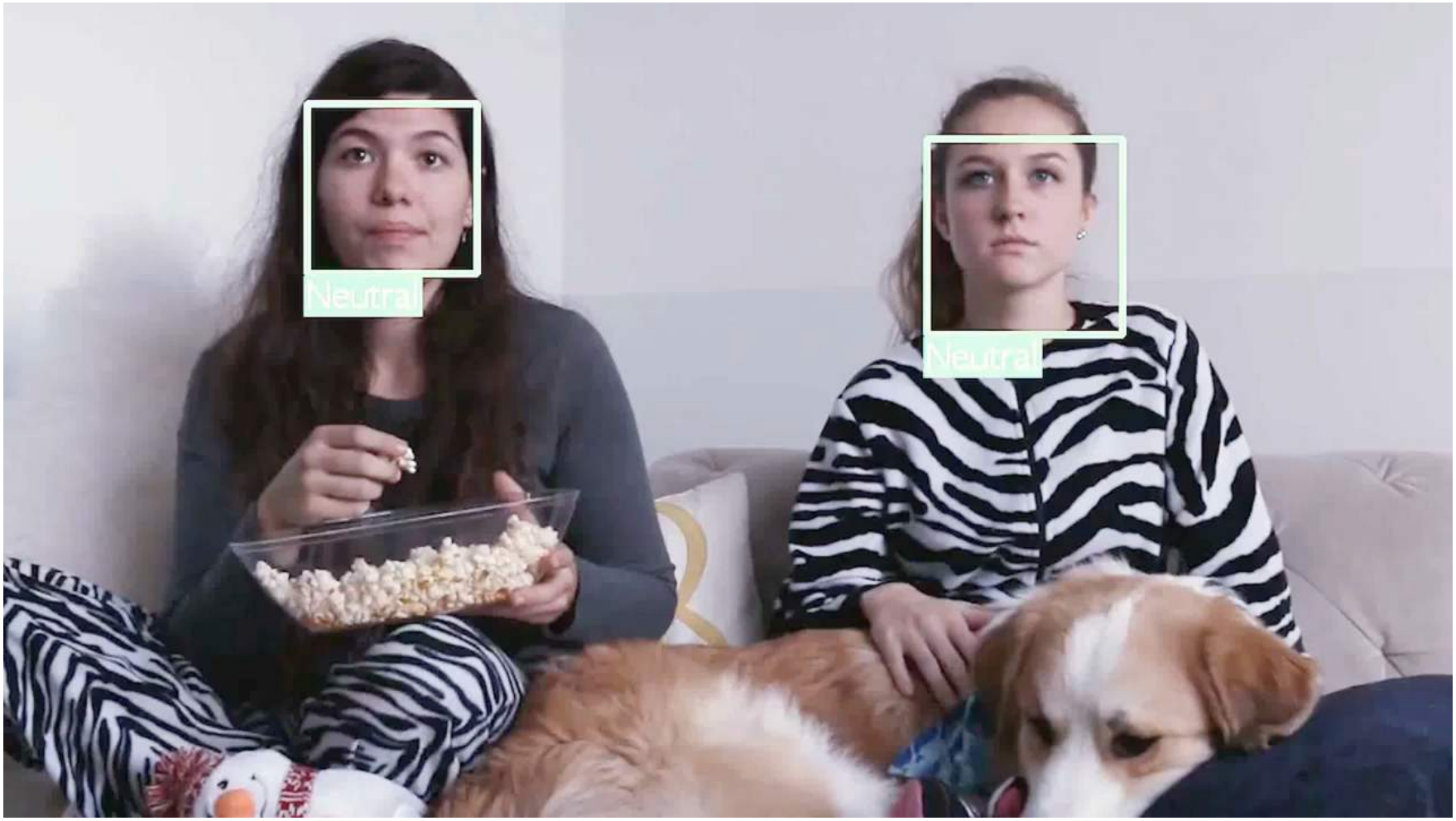
Face recognition (identification): Compare an input image (probe) against a database (gallery) and reports a match

Face authentication: verify the claim of the identity of an individual in an input image

Face tracking: Continuously estimate the location and possibly the orientation of a face in an image sequence in real time

Emotion recognition: Identifying the affective states (happy, sad, disgusted, etc.) of humans

Tracking the Emotions



Detection *vs* Recognition

Detection: concerned with a *category* of object

Recognition: concerned with *individual* identity

Face is a highly non-rigid object

Many methods can be applied to other object detection/recognition

Car detection

Pedestrian detection

Research Issues

- **Representation:** How to describe a typical face?
- **Scale:** How to deal with face of different size?
- **Search strategy:** How to spot these faces?
- **Speed:** How to speed up the process?
- **Precision:** How to locate the faces precisely?
- **Post-processing:** How to combine detection results?

Methods to Detect Faces

Knowledge-based methods

Encode human knowledge of what constitutes a typical face (usually the relationships between facial features)

Feature invariant approaches

Aim to find structural features of a face that exist even when the pose, viewpoint, or lighting conditions vary

Template matching methods

Several standard patterns stored to describe the face as a whole or the facial features separately

Appearance-based methods

The models (or templates) are learned from a set of training images which capture the representative variability of facial appearance

Knowledge-based Methods

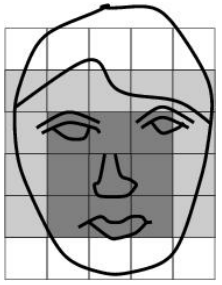
Top-down approach: Represent a face using a set of human-coded rules.

Example:

- The center part of face has uniform intensity values
- The difference between the average intensity values of the center part and the upper part is significant
- A face often appears with two eyes that are symmetric to each other, a nose and a mouth
- Use these rules to guide the search process

Knowledge-Based Method [Yang and Huang 94]

- Multi-resolution focus-of-attention approach
- **Level 1** (lowest resolution): apply the rule “the center part of the face has 4 cells with a basically uniform intensity” to search for candidates
- **Level 2:** local histogram equalization followed by edge detection
- **Level 3:** search for eye and mouth features for validation



Knowledge-Based Method [Kotropoulos & Pitas 94]

- Horizontal/vertical projection to search for candidates

$$HI(x) = \sum_{y=1}^{n} I(x, y) \qquad VI(y) = \sum_{x=1}^{m} I(x, y)$$

- Search eyebrow/eyes, nostrils/nose for validation
- Difficult to detect multiple people or in complex background

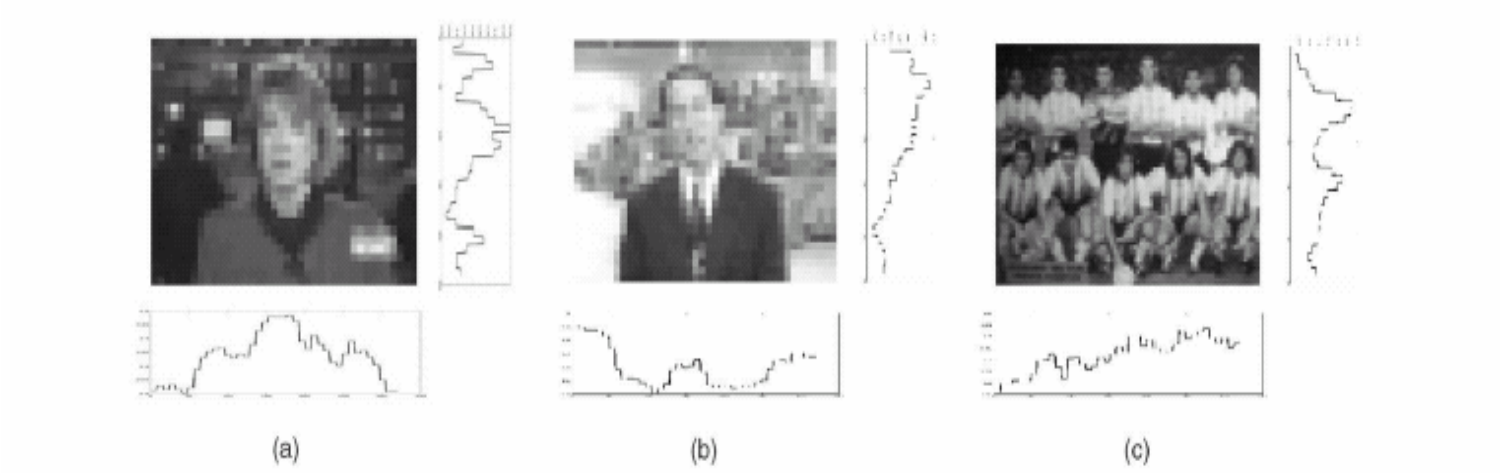


Fig. 3. (a) and (b) n = 8. (c) n = 4. Horizontal and vertical profiles. It is feasible to detect a single face by searching for the peaks in horizontal and vertical profiles. However, the same method has difficulty detecting faces in complex backgrounds or multiple faces as shown in (b) and (c).

Knowledge-based Methods

Pros:

- Easy to come up with simple rules to describe the features of a face and their relationships
- Based on the coded rules, facial features in an input image are extracted first, and face candidates are identified
- Work well for face localization in uncluttered background

Cons:

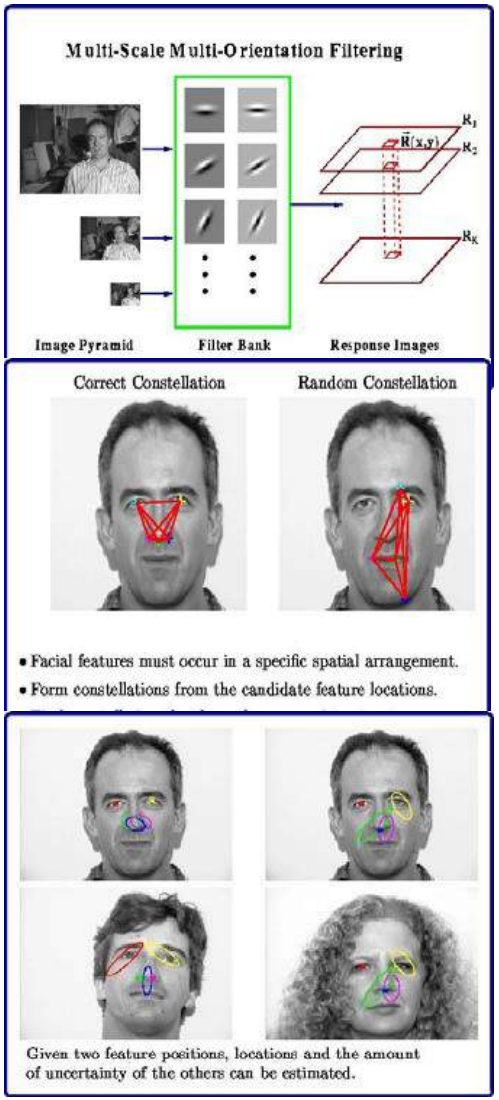
- Difficult to translate human knowledge into rules precisely: detailed rules fail to detect faces and general rules may find many false positives
- Difficult to extend this approach to detect faces in different poses: implausible to enumerate all the possible cases

Feature-based Methods

- Bottom-up approach: Detect facial features (eyes, nose, mouth, etc) first
- Facial features: edge, intensity, shape, texture, color, etc
- Aim to detect invariant features
- Group features into candidates and verify them

Random Graph Matching [Leung et al. 95]

- Formulate as a problem to find the correct geometric arrangement of facial features
- Facial features are defined by the average responses of multi-scale filters
- Graph matching among the candidates to locate faces



Feature-Based Methods

Pros:

- Features are invariant to pose and orientation change

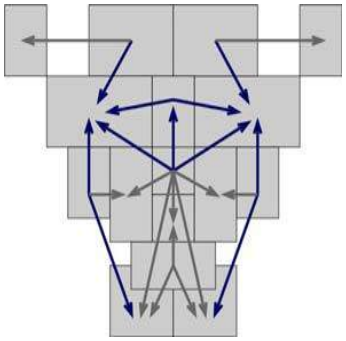
Cons:

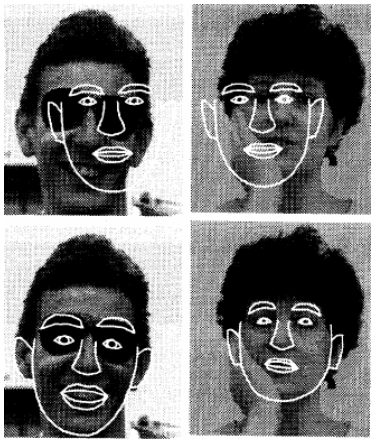
- Difficult to locate facial features due to several corruption (illumination, noise, occlusion)
- Difficult to detect features in complex background

Template Matching Methods

- Store a template
  - Predefined: based on edges or regions
  - Deformable: based on facial contours (e.g., Snakes)
- Templates are hand-coded (not learned)
- Use correlation to locate faces

Face Templates





Template-Based Methods

Pros

- Simple

Cons

- Templates needs to be initialized near the face images
- Difficult to enumerate templates for different poses (similar to knowledge-based methods)

Appearance-based Methods

General idea

1. Collect a large set of (resized) face and non-face images and train a classifier to discriminate them.

2. Given a test image, detect faces by applying the classifier at each position and scale of the image.

Sung and Poggio (1994)

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 20, NO. 1, JANUARY 1998

Example-Based Learning for View-Based Human Face Detection

Kah-Kay Sung and Tomaso Poggio

Abstract—We present an example-based learning approach for locating vertical frontal views of human faces in complex scenes. The technique models the distribution of human face patterns by means of a few view-based "face" and "nonface" model clusters. At each image location, a difference feature vector is computed between the local image pattern and the distribution-based model. A trained classifier determines, based on the difference feature vector measurements, whether or not a human face exists at the current image location. We show empirically that the distance metric we adopt for computing difference feature vectors, and the "nonface" clusters we include in our distribution-based model, are both critical for the success of our system.

Index Terms—Face detection, object detection, example-based learning, example selection, pattern recognition, view-based recognition, density estimation, Gaussian mixture model.

Originally published as an MIT Technical Report in 1994

System Overview

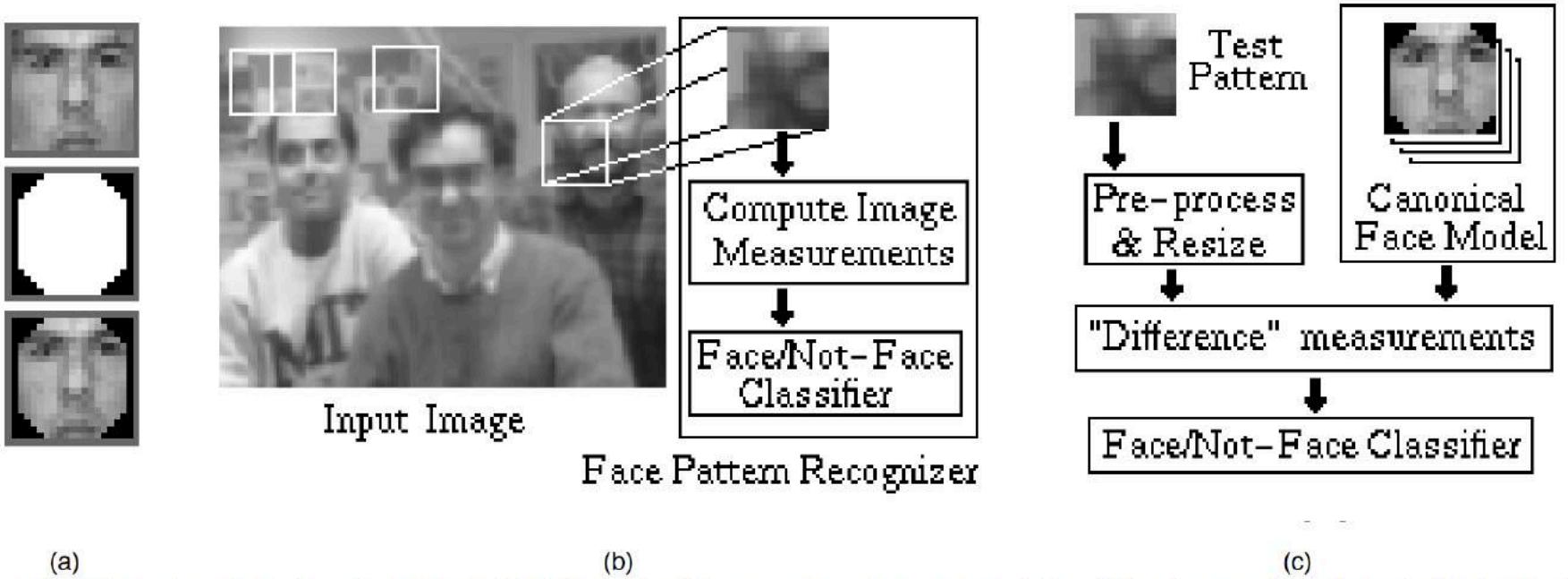


Fig. 1. Overview of our face-detection system. (a) A "canonical" face pattern, a 19 × 19 mask for eliminating near-boundary pixels of canonical face patterns, and the resulting "canonical" face pattern after applying the mask. (b) At each scale, the image is divided into many possibly overlapping windows. Each window pattern gets classified as either "a face" or "not a face," based on a set of local image measurements. (c) The key components of the *Face Pattern Recognizer* block in greater detail.

Pre-processing

Resizing: resizes all image patterns to 19x19 pixels

Masking: reduce the unwanted background noise in a face pattern

Illumination gradient correction: find the best fit brightness plane and then subtracted from it to reduce heavy shadows caused by extreme lighting angles

Histogram equalization: compensates the imaging effects due to changes in illumination and different camera input gains

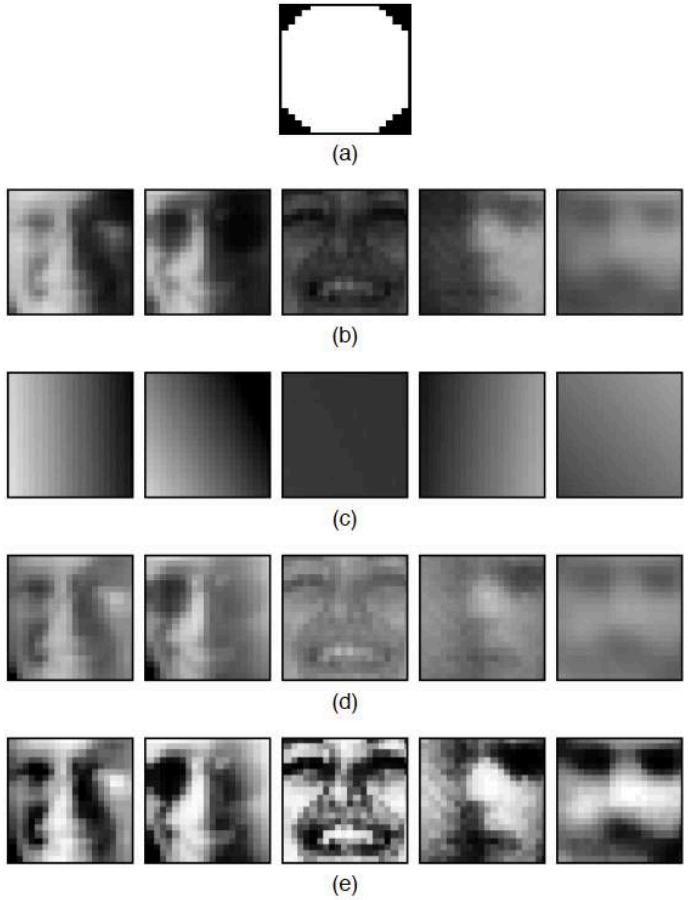
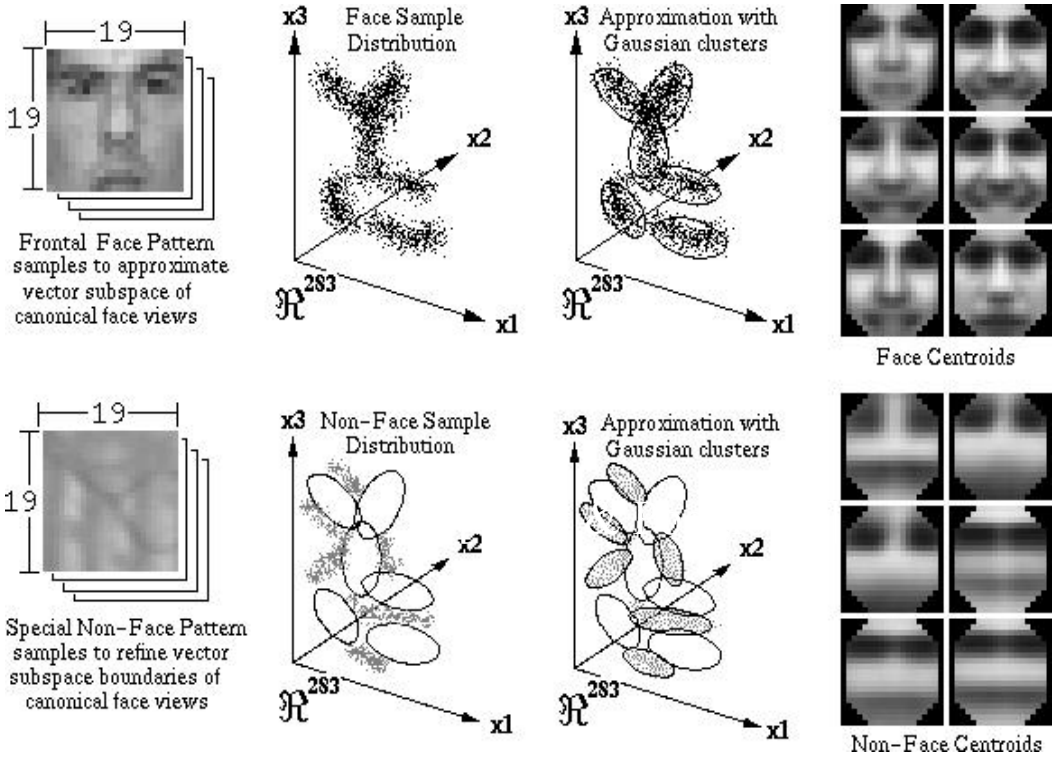


Fig. 2. The steps in preprocessing a window. First, a linear function is fit to the intensity values in the window, and then subtracted out, correcting for some extreme lighting conditions. Then, histogram equalization is applied, to correct for different camera gains and to improve contrast. For each of these steps, the mapping is computed based on pixels inside the oval mask, and then applied to the entire window. (a) Oval mask for ignoring background pixels. (b) Original window. (c) Best fit linear function. (d) Lighting corrected window (linear function subtracted). (e) Histogram equalized window.

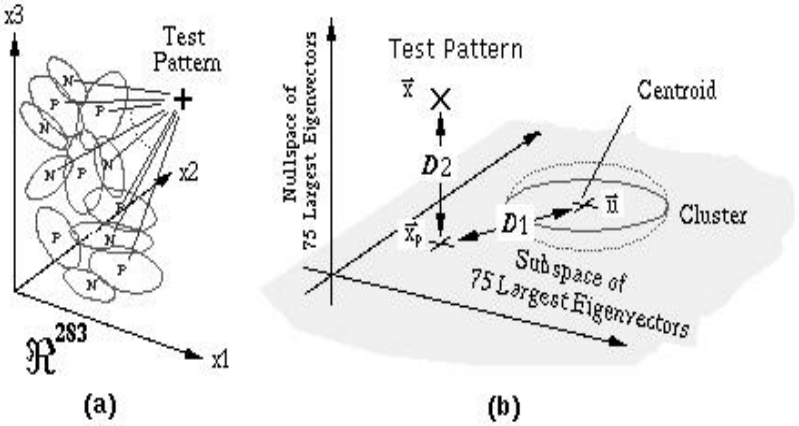
Distribution of Face Patterns

- Cluster face and non-face samples into a few (i.e., 6) clusters using K-means algorithm
- Each cluster is modeled by a multi-dimensional Gaussian with a centroid and covariance matrix
- Approximate each Gaussian covariance with a subspace (i.e., using the largest eigenvectors)



Distance Metrics

- Compute distances of a sample to all the face and non-face clusters
- Each distance has two parts:
  - Within subspace distance (D_1): Mahalanobis distance of the projected sample to cluster center
  - Distance to the subspace (D_2): distance of the sample to the subspace
- Feature vector: Each face/non-face samples is represented by a vector of these distance measurements
- Train a multilayer neural network using the feature vectors for face detection



- 6 face clusters
- 6 non-face clusters
- 2 distance values per cluster
- 24 measurements

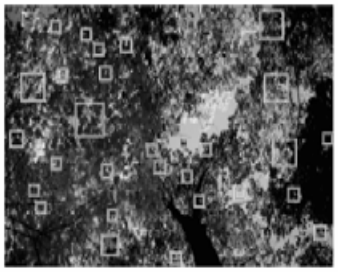
Face and Non-faces Examples

Positive examples

- Get as much variation as possible
- Manually crop and normalize each face image into a standard size (e.g., 19 × 19 pixels)
- Creating virtual examples

Negative examples

- Fuzzy idea
- Any images that do not contain faces
- A large image subspace
- Bootstraping

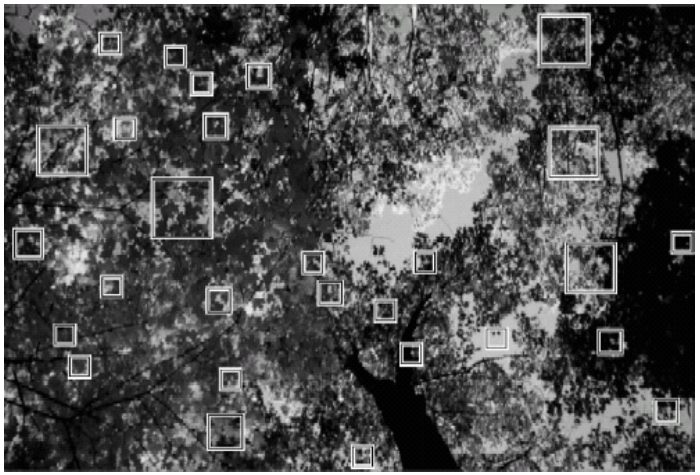


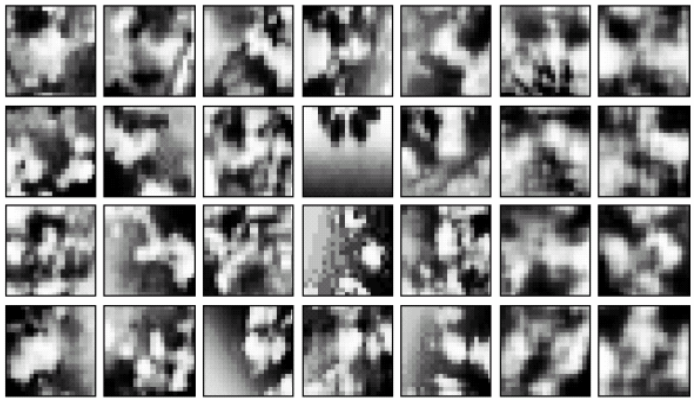
Creating Virtual Positive Examples

- Simple and very effective method
- Randomly mirror, rotate, translate and scale face samples by small amounts
- Increase number of training examples
- Less sensitive to alignment error

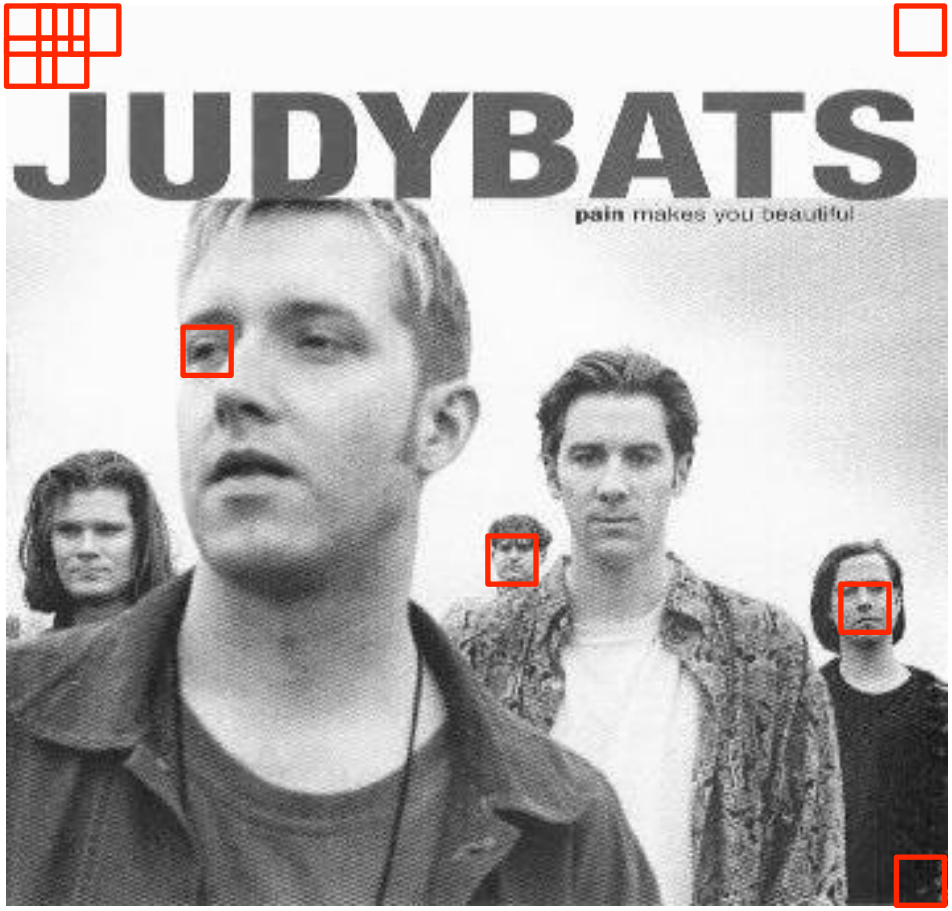
Bootstrapping

1. Start with a small set of non-face examples in the training set
2. Train a neural network classifier with the current training set
3. Run the learned face detector on a sequence of random images.
4. Collect all the non-face patterns that the current system wrongly classifies as faces (i.e., false positives)
5. Add these non-face patterns to the training set
6. Got to Step 2 or stop if satisfied

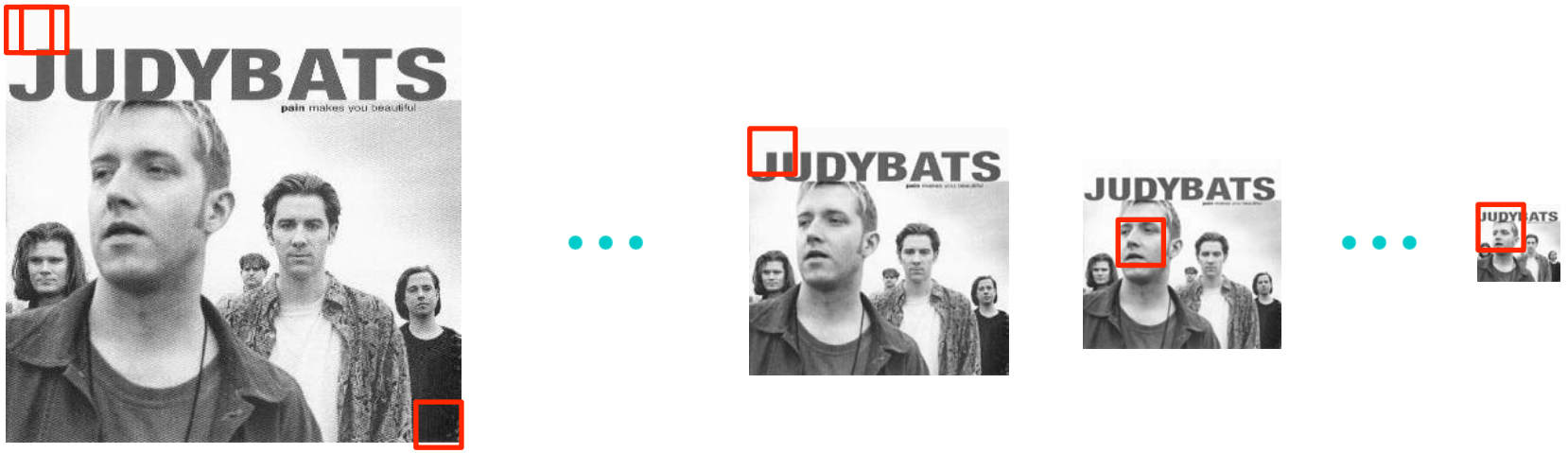




Search over Space and Scale



Search over Space and Scale



Continue to downsample the input image and search until the image size is too small

Some Results

Rowley-Baluja-Kanade (1996/98)

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 20, NO. 1, JANUARY 1998

Neural Network-Based Face Detection

Henry A. Rowley, *Student Member, IEEE,* Shumeet Baluja, and Takeo Kanade, *Fellow, IEEE*

Abstract—We present a neural network-based upright frontal face detection system. A retinally connected neural network examines small windows of an image and decides whether each window contains a face. The system arbitrates between multiple networks to improve performance over a single network. We present a straightforward procedure for aligning positive face examples for training. To collect negative examples, we use a bootstrap algorithm, which adds false detections into the training set as training progresses. This eliminates the difficult task of manually selecting nonface training examples, which must be chosen to span the entire space of nonface images. Simple heuristics, such as using the fact that faces rarely overlap in images, can further improve the accuracy. Comparisons with several other state-of-the-art face detection systems are presented, showing that our system has comparable performance in terms of detection and false-positive rates.

Index Terms—Face detection, pattern recognition, computer vision, artificial neural networks, machine learning.

Originally presented at CVPR 1996

Rotation Invariant Neural Network-Based Face Detection*

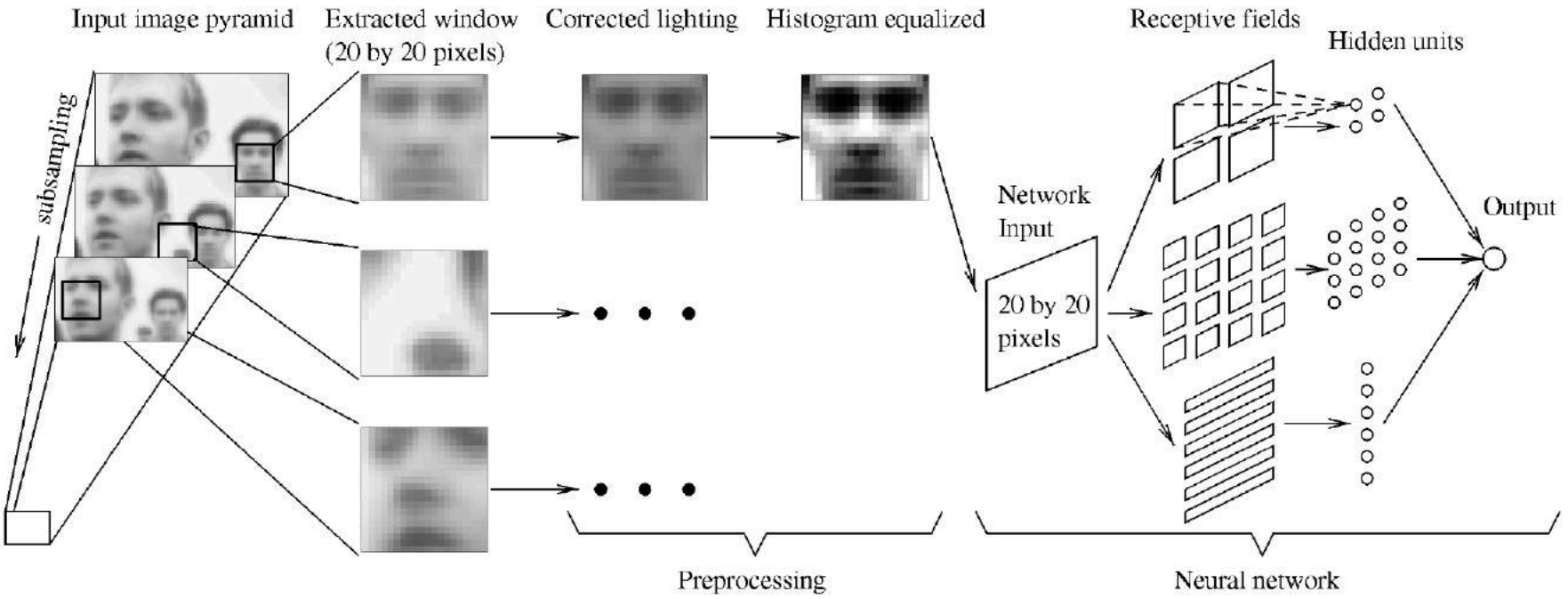
Henry A. Rowley[1] har@cs.cmu.edu
Shumeet Baluja[2,1] baluja@jprc.com
Takeo Kanade[1] tk@cs.cmu.edu

[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
[2] Justsystem Pittsburgh Research Center, 4616 Henry Street, Pittsburgh, PA 15213

Features

- Similar to Sung and Poggio
- 20x20 instead of 19x19
- Same technique for bootstrapping, preprocessing, etc.
- Neural network (with different receptive fields) applied directly to the image
- Different heuristics
- Faster than Sung and Poggio (but still far from real-time)

The Architecture

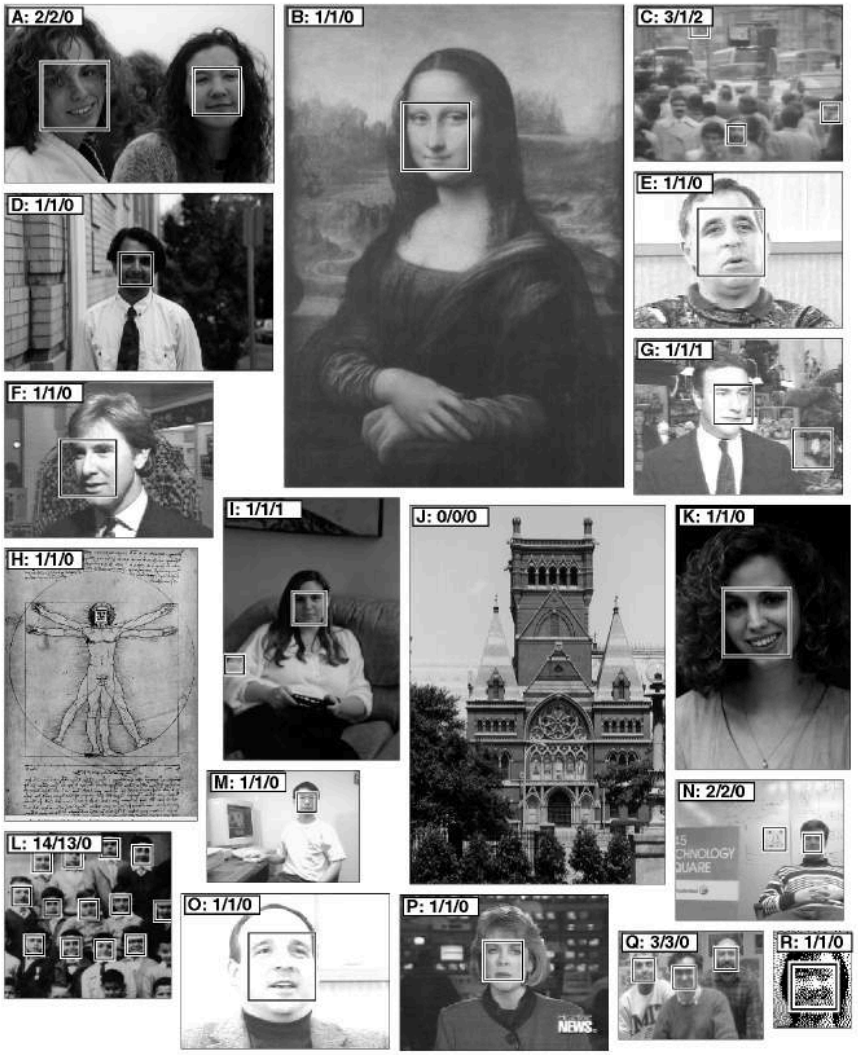


Trained using standard back-propagation with momentum

Some Results

The label in the upper left corner of each image (D/T/F) gives the number of faces detected (D), the total number of faces in the image (T), and the number of false detections (F).

Some Results

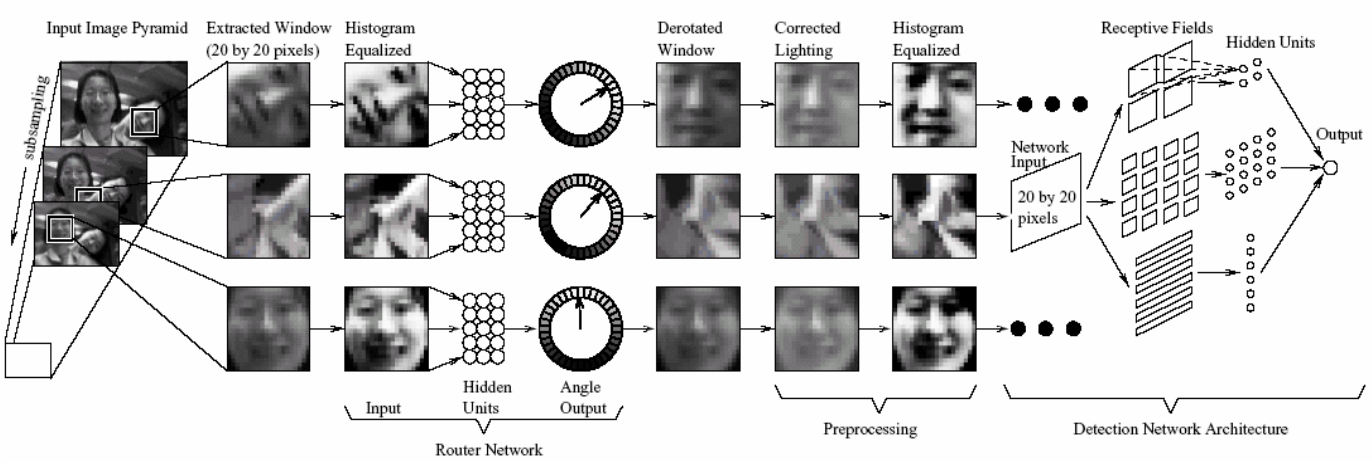


Detecting Rotated Faces

A router network is trained to estimate the angle of an input window

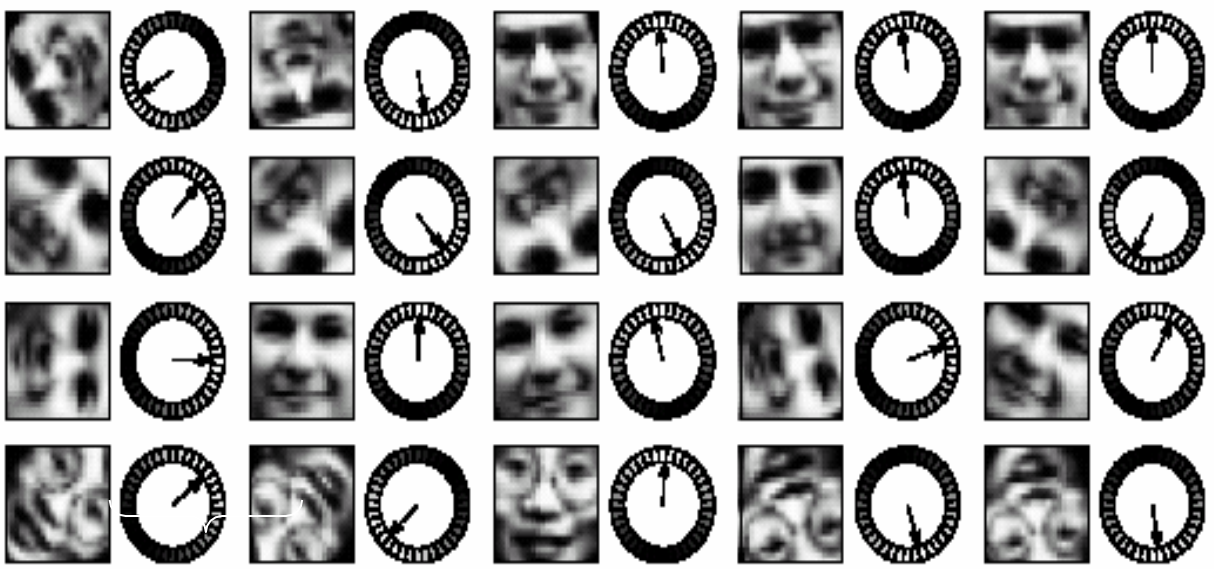
- If it contains a face, the router returns the angle of the face and the face can be rotated back to upright frontal position
- Otherwise the router returns a meaningless angle

The de-rotated window is then applied to a detector (previously trained for upright frontal faces)



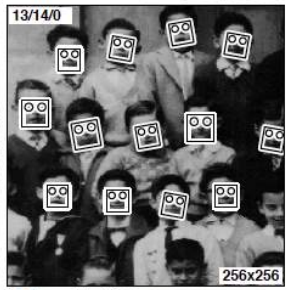
Router Network

- Rotate a face sample at 10 degree increment
- Create virtual examples (translation and scaling) from each sample
- Train a multilayer neural network with input-output pair



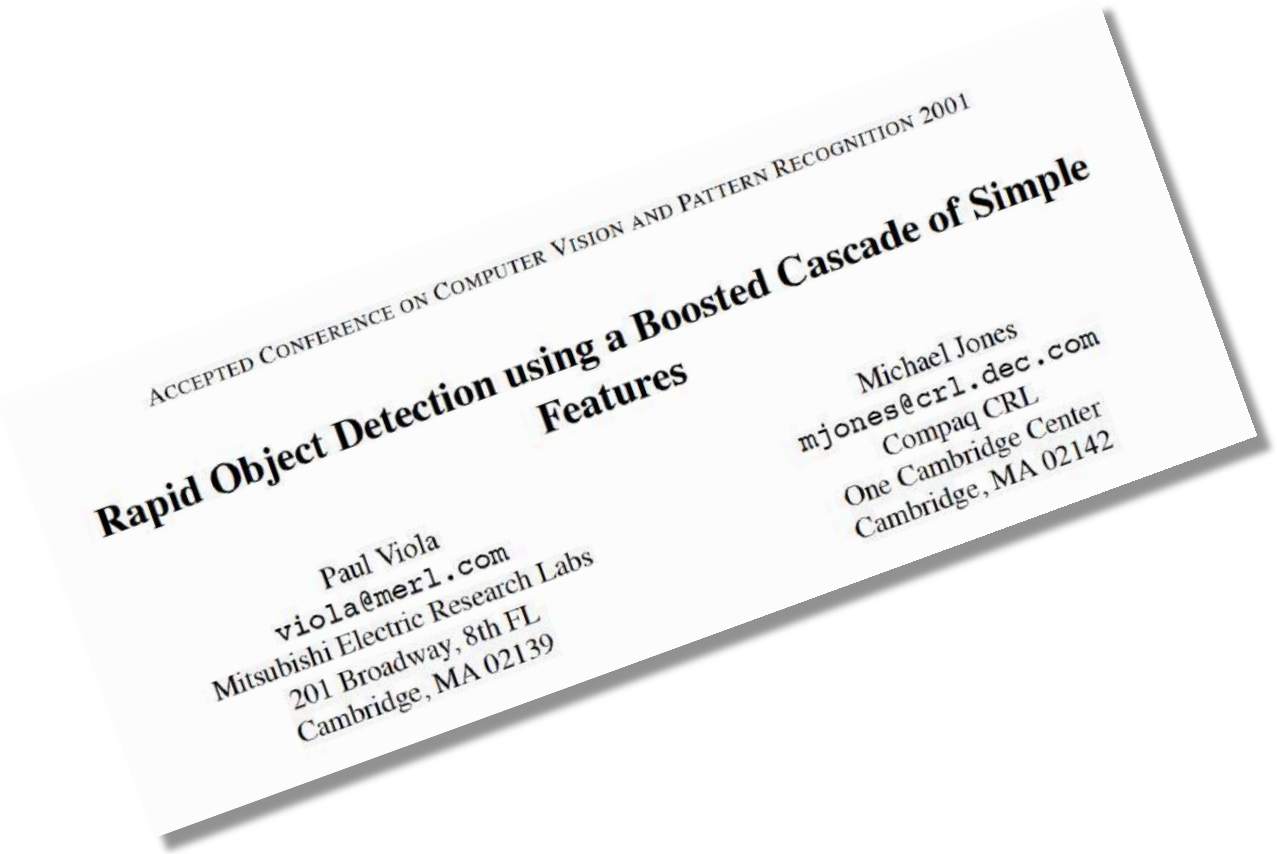
Input-output pair to train a router network

Some Results



The label in the upper left corner of each image (D/T/F) gives the number of faces detected (D), the total number of faces in the image (T), and the number of false detections (F). The label in the lower right corner of each image gives its size in pixels

Viola and Jones (2001)



ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001

Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola
viola@merl.com
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
mjones@crl.dec.com
Compaq CRL
One Cambridge Center
Cambridge, MA 02142

Abstract

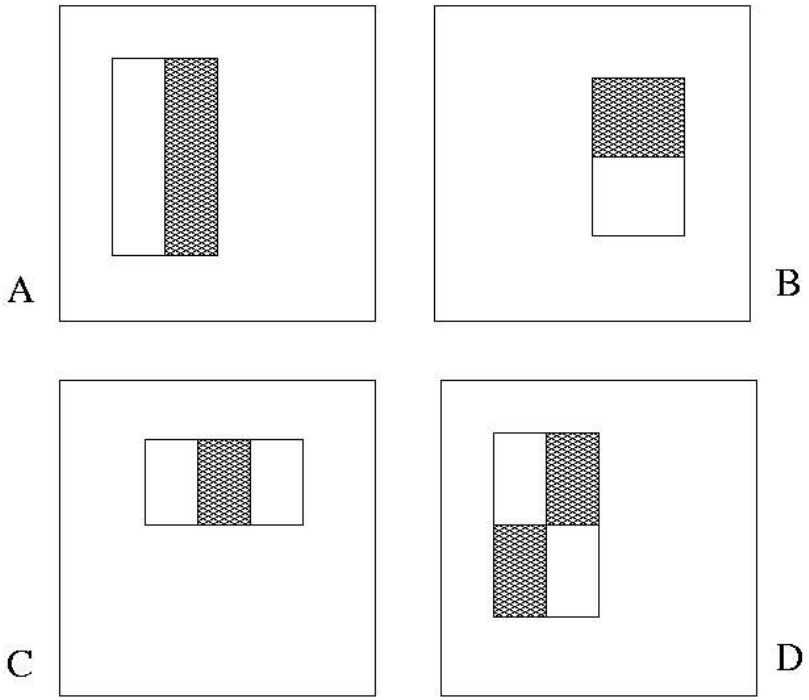
This paper describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions. The first is the introduction of a new image representation called the "Integral Image" which allows the features used by our detector to be computed very quickly. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers[6]. The third contribution is a method for combining increasingly more complex classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. The cascade can be viewed as an object specific focus-of-attention mechanism which unlike previous approaches provides statistical guarantees that discarded regions are unlikely to contain the object of interest. In the domain of face detection the system yields detection rates comparable to the best previous systems. Used in real-time applications, the detector runs at 15 frames per second without resorting to image differencing or skin color detection.

Journal version: P. Viola and M. Jones. Robust real-time face detection. *Int. J. Computer Vision* (2004).

The Viola-Jones Face Detector

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

Rectangular Image Features



Value = ∑ (pixels in white area) – ∑ (pixels in black area)

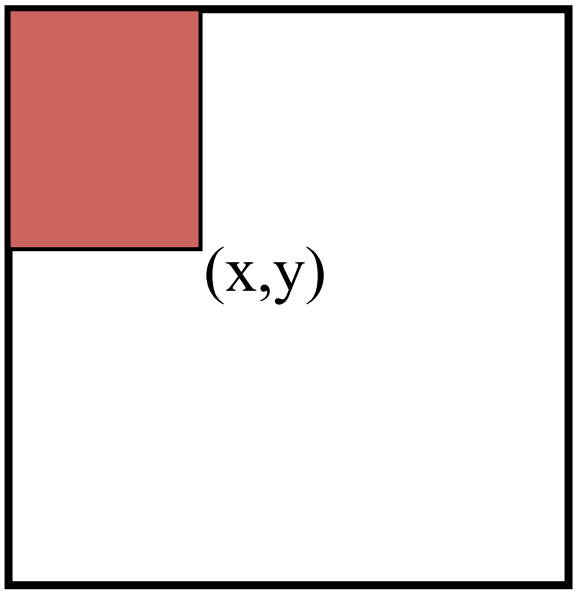
Rectangular Image Features

Forehead, eye features can be captured

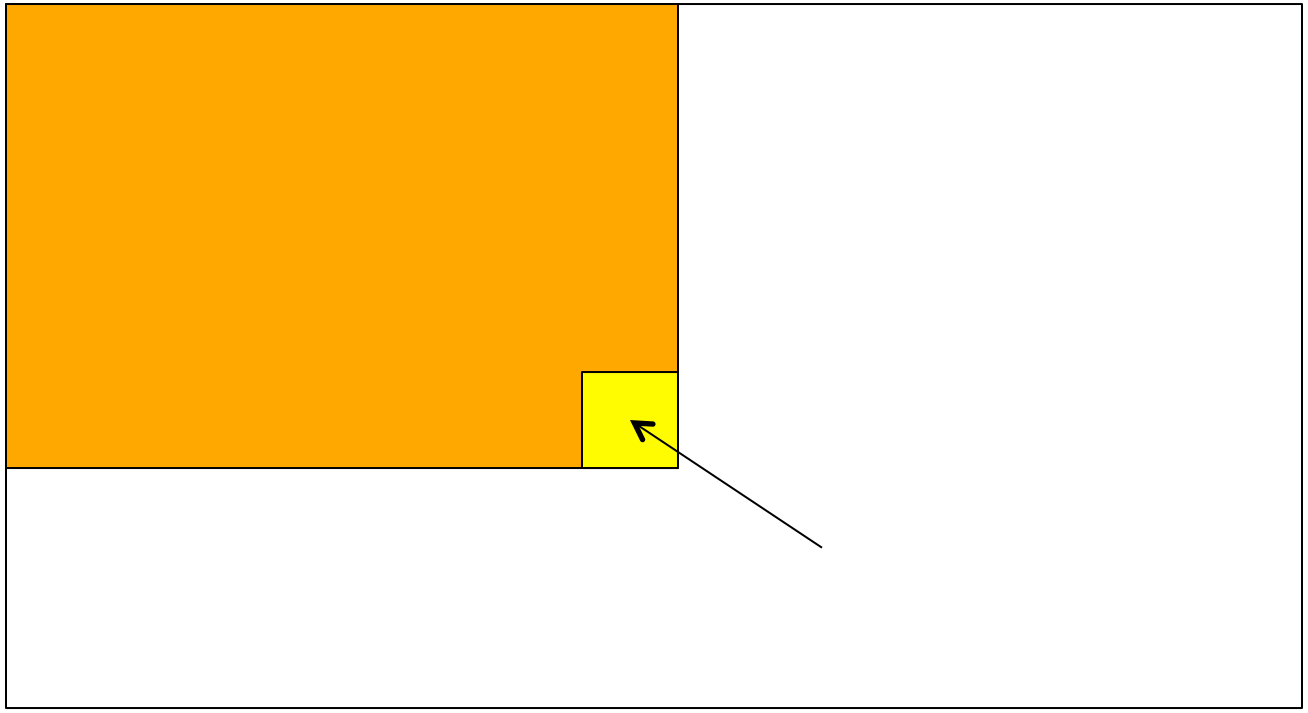
Fast Computation with Integral Images

- The *integral image* computes a value at each pixel (*x*,*y*) that is the sum of the pixel values above and to the left of (*x*,*y*), inclusive

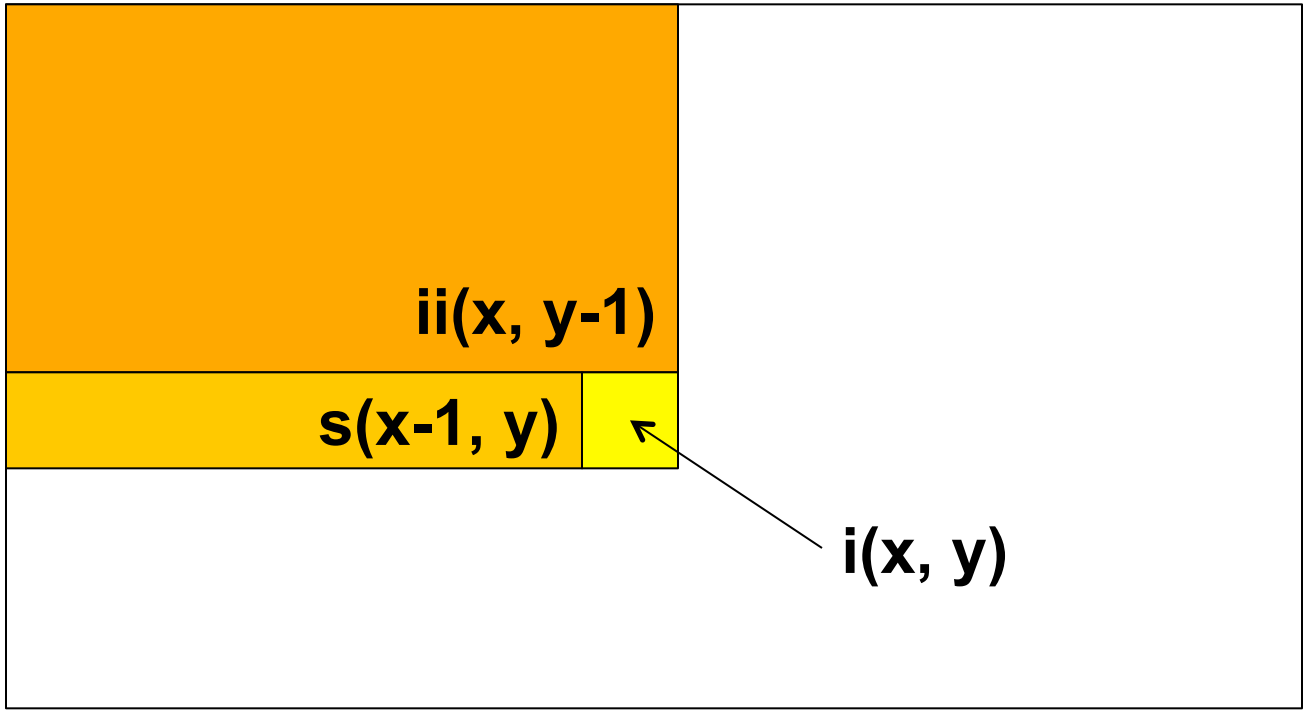
- This can quickly be computed in one pass through the image



Computing the Integral Image



Computing the Integral Image



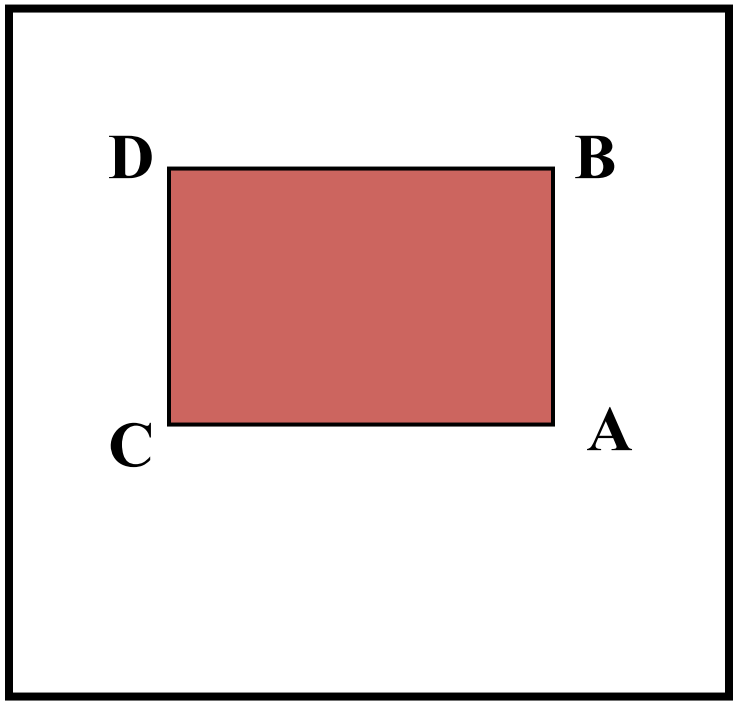
Cumulative row sum: $s(x, y) = s(x-1, y) + i(x, y)$

Integral image: $ii(x, y) = ii(x, y-1) + s(x, y)$

Computing Sum within a Rectangle

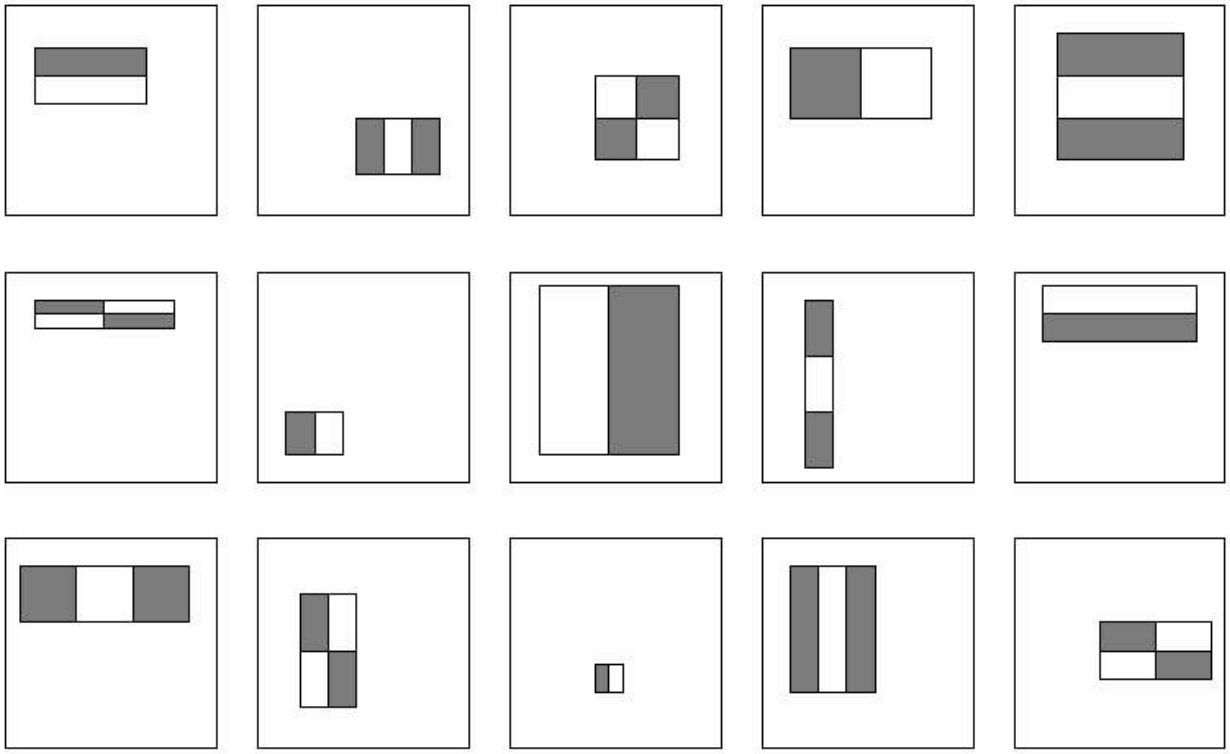
- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!

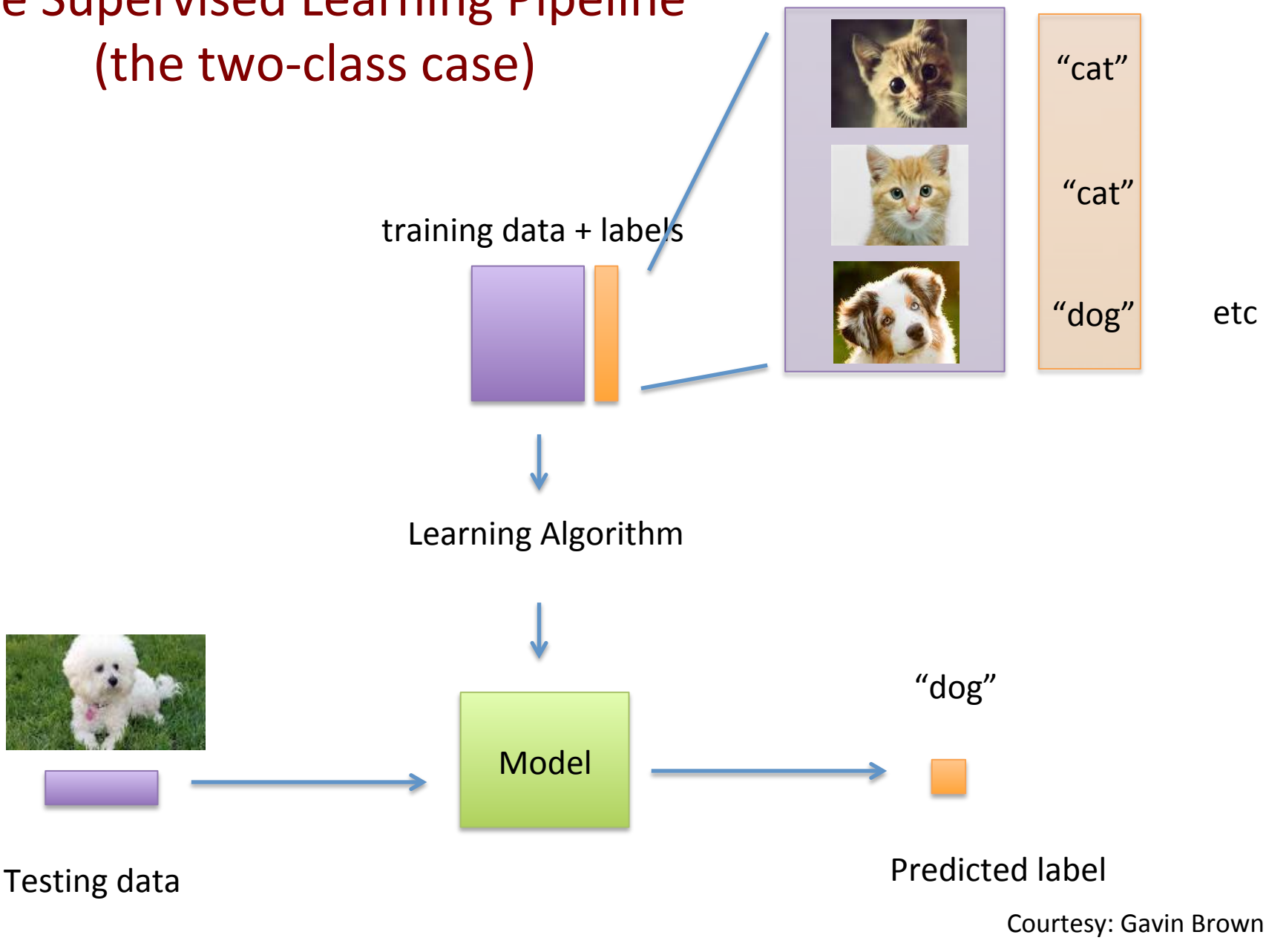


Feature selection

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

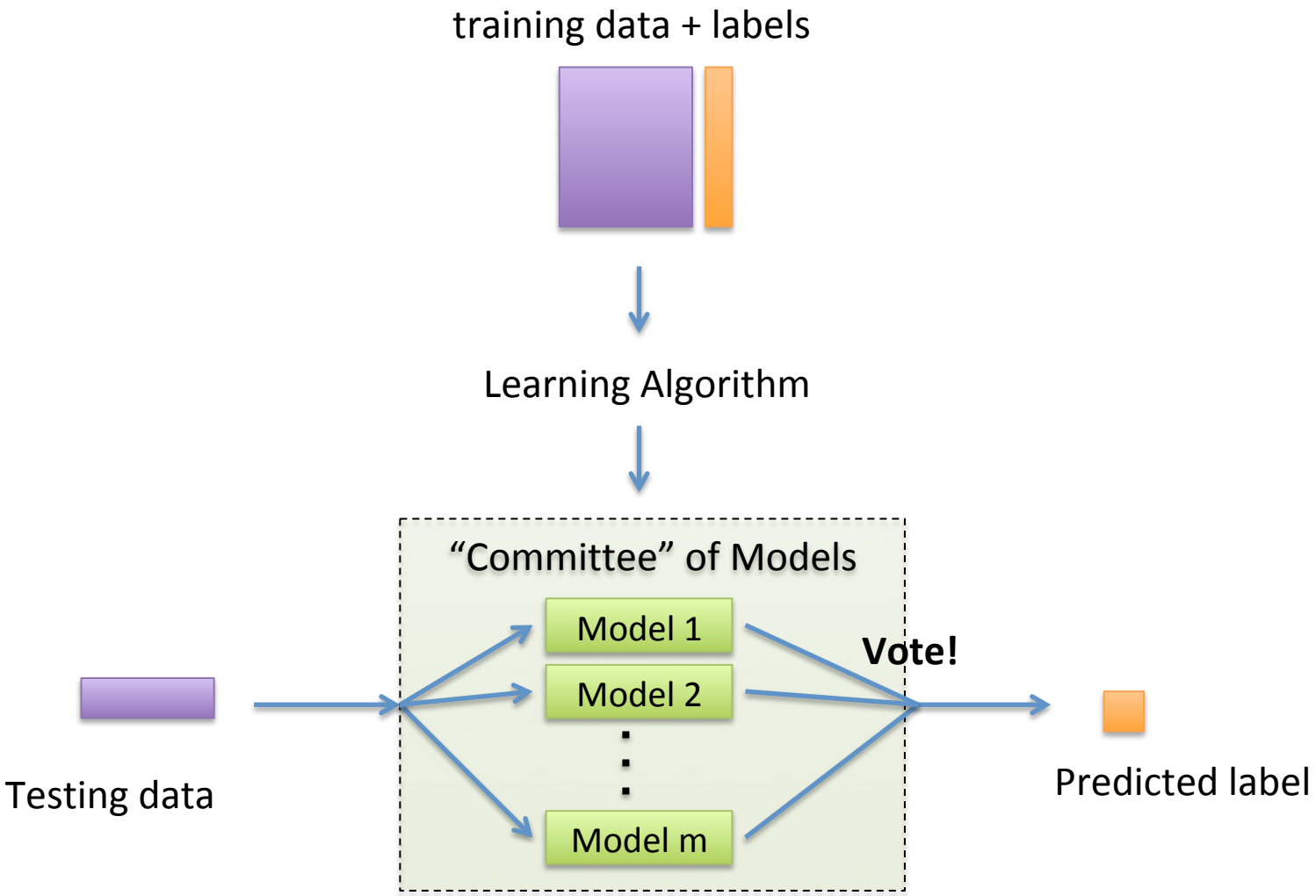


The Supervised Learning Pipeline (the two-class case)



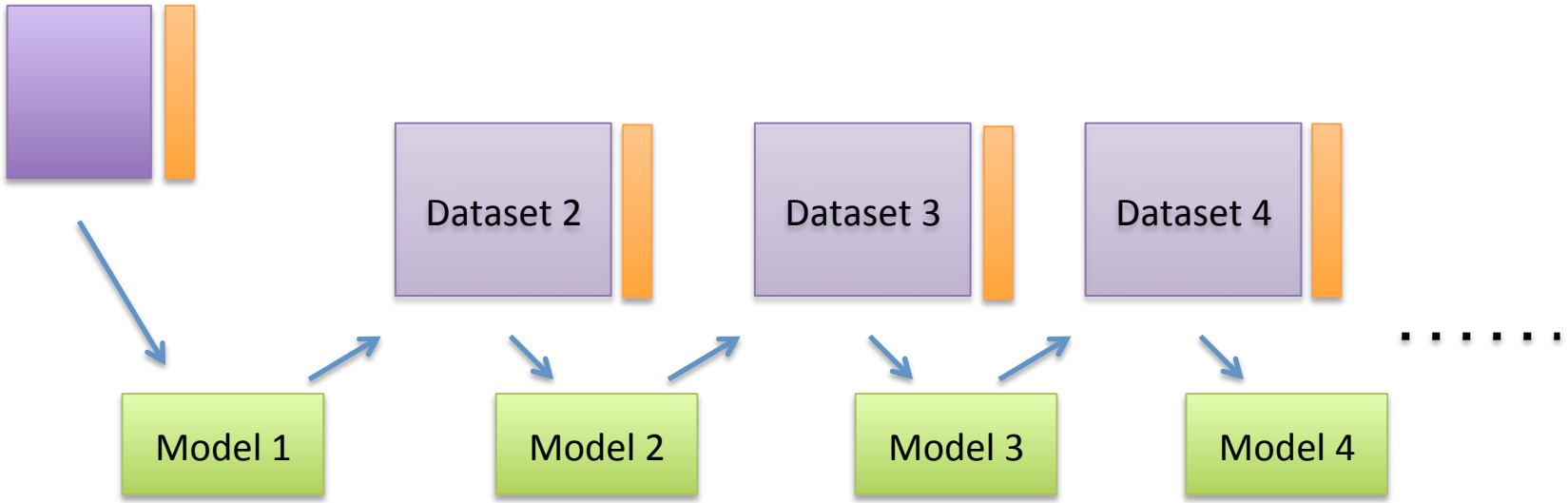
Courtesy: Gavin Brown

The *Ensemble* Approach



Courtesy: Gavin Brown

Boosting



"Boosting" algorithms build an ensemble, sequentially.

Each model corrects the mistakes of its predecessors.

Courtesy: Gavin Brown

Boosting

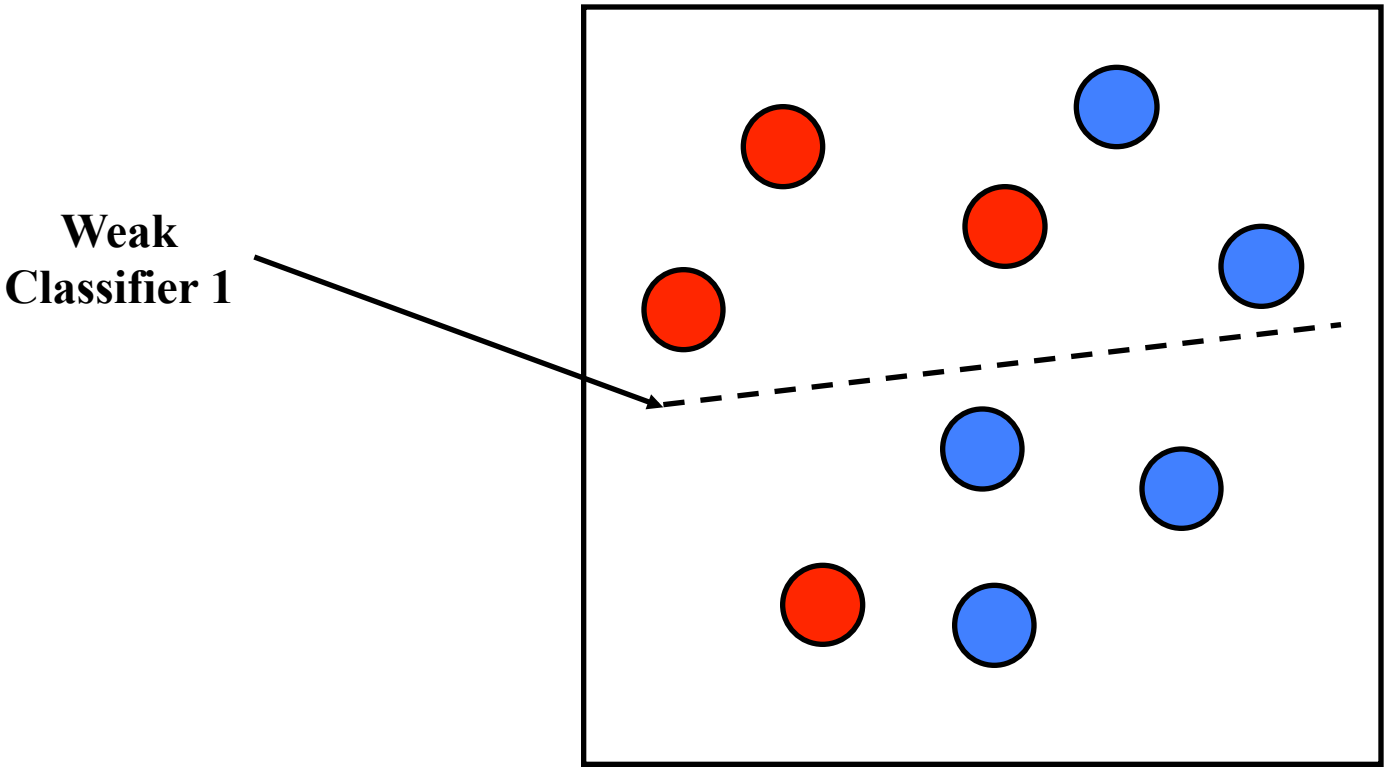
- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
  - A weak learner need only do better than chance

- 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

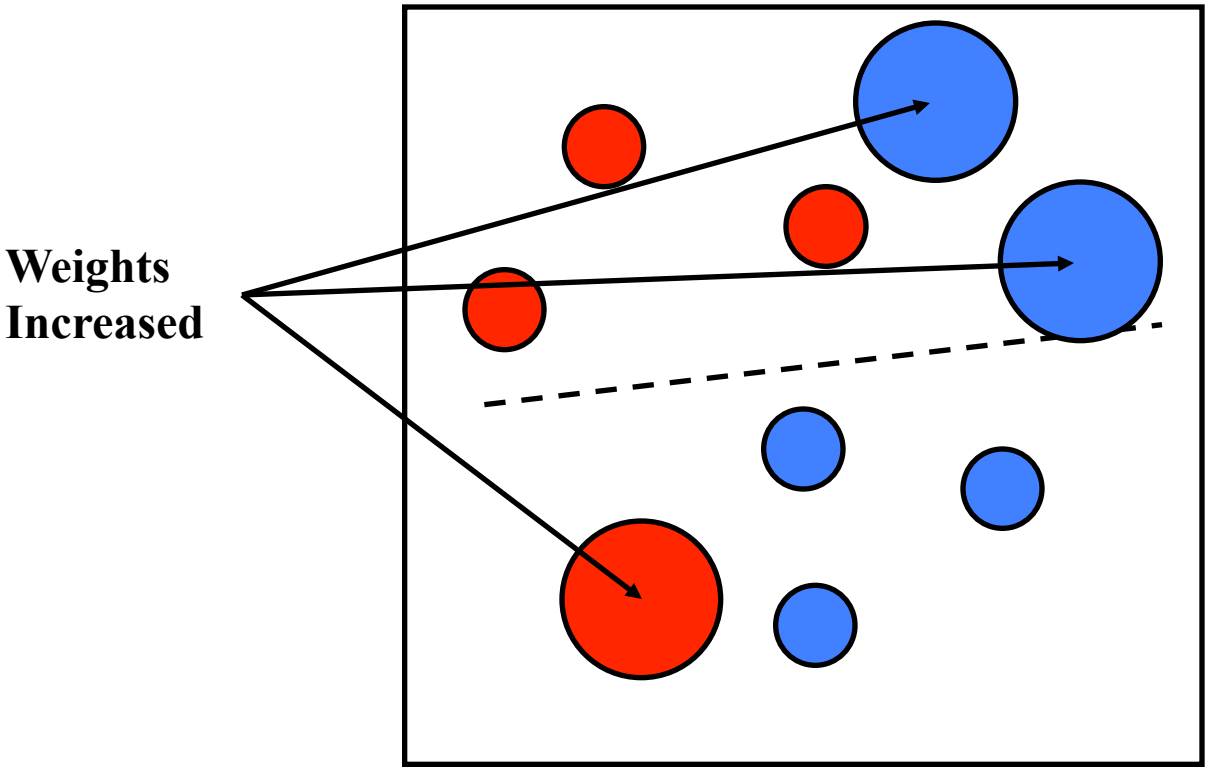
Boosting

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest *weighted* training error
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
- Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

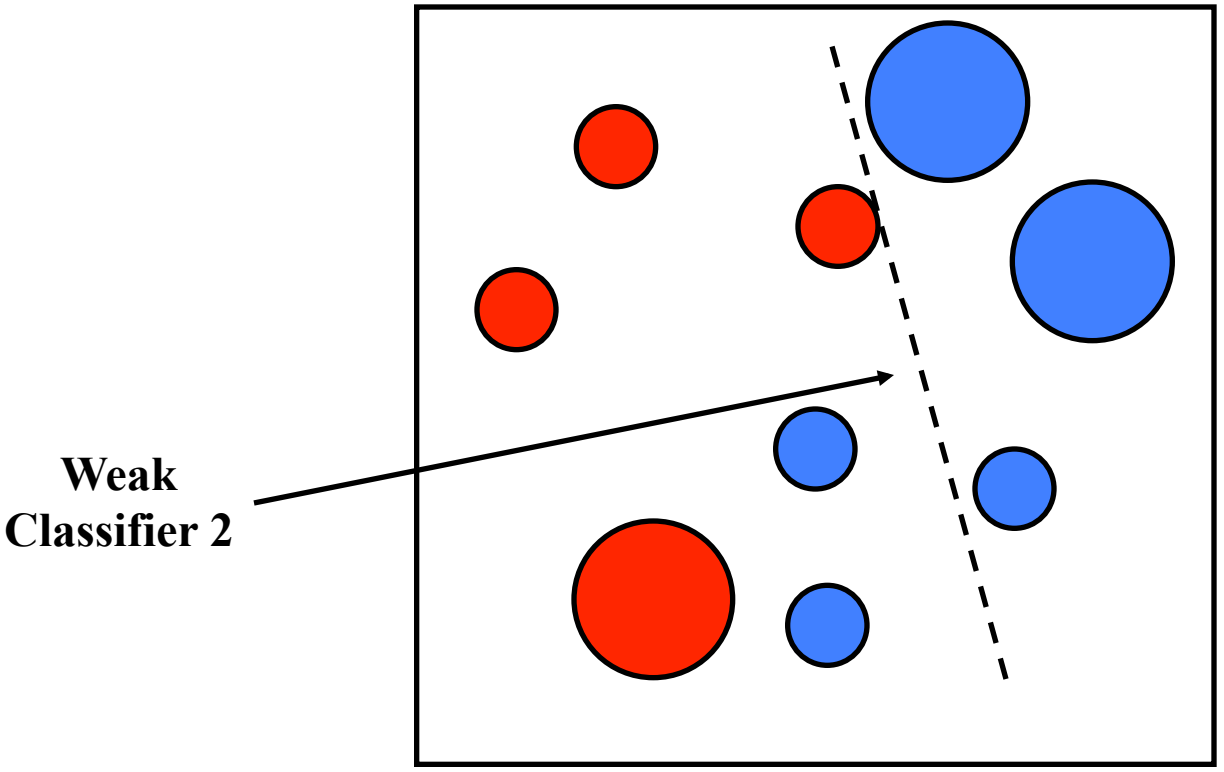
Boosting at work



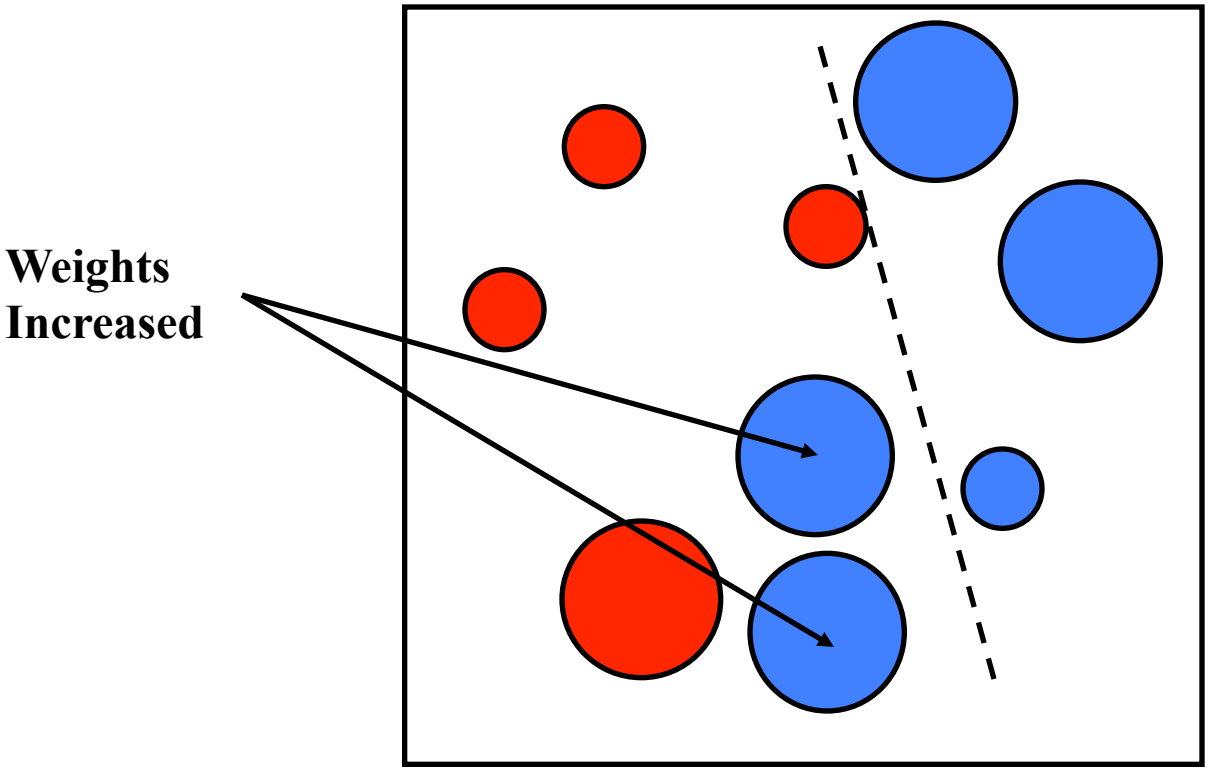
Boosting at work



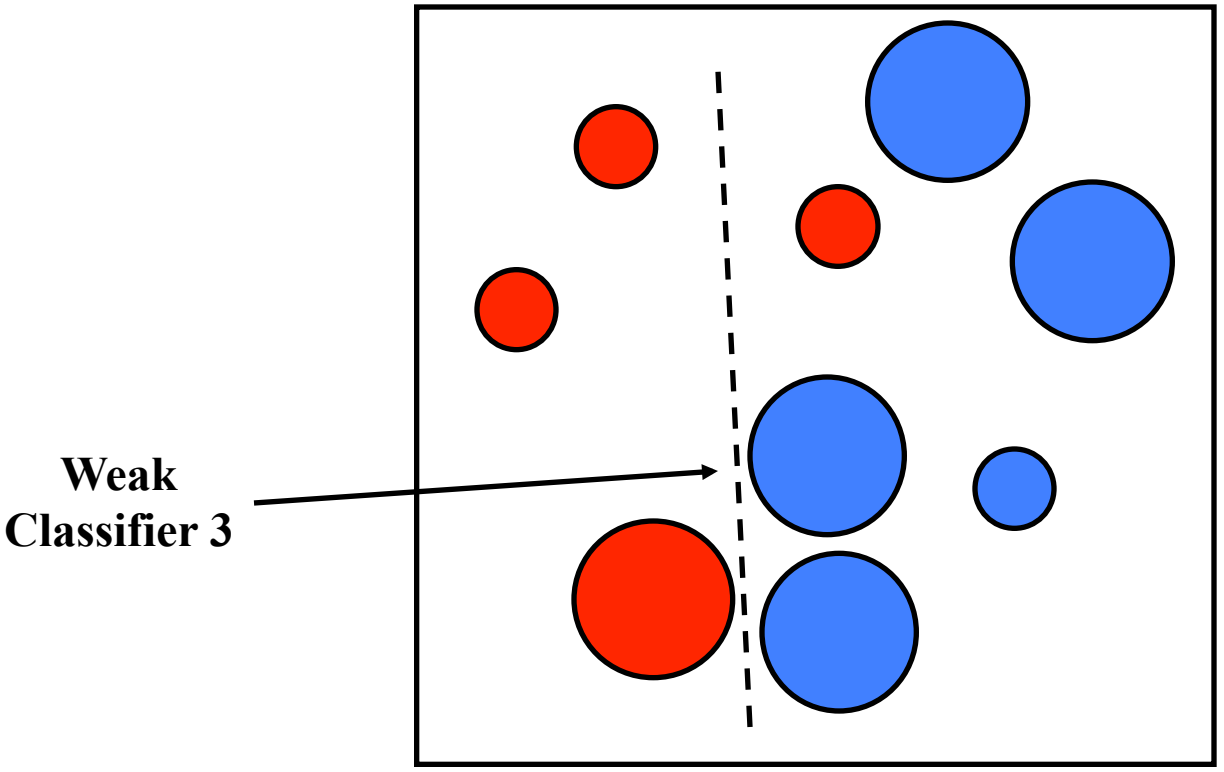
Boosting at work



Boosting at work

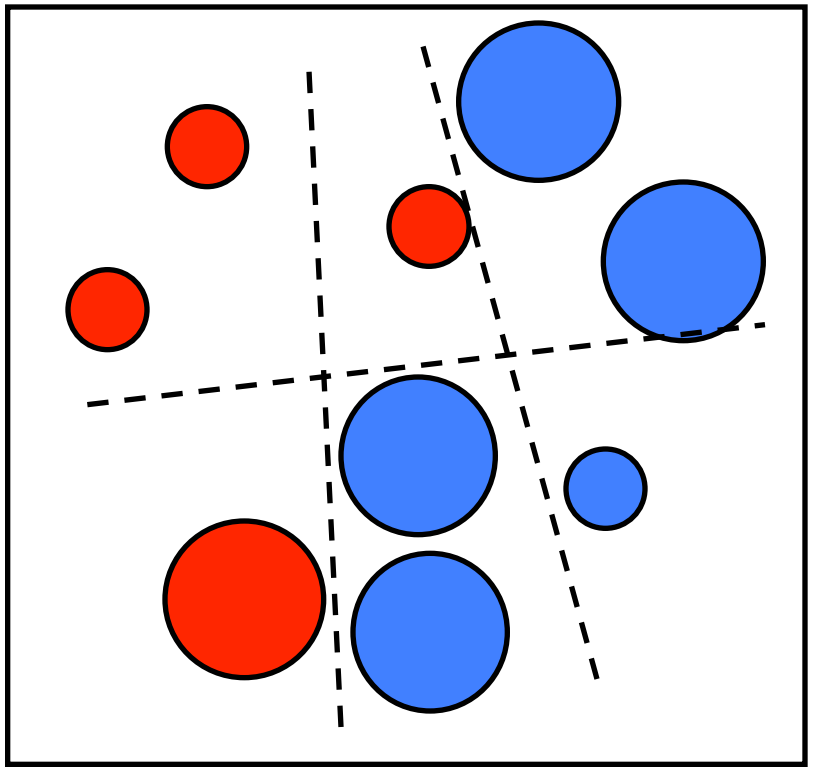


Boosting at work



Boosting at work

Final classifier is a combination of weak classifiers



Boosting for Face Detection

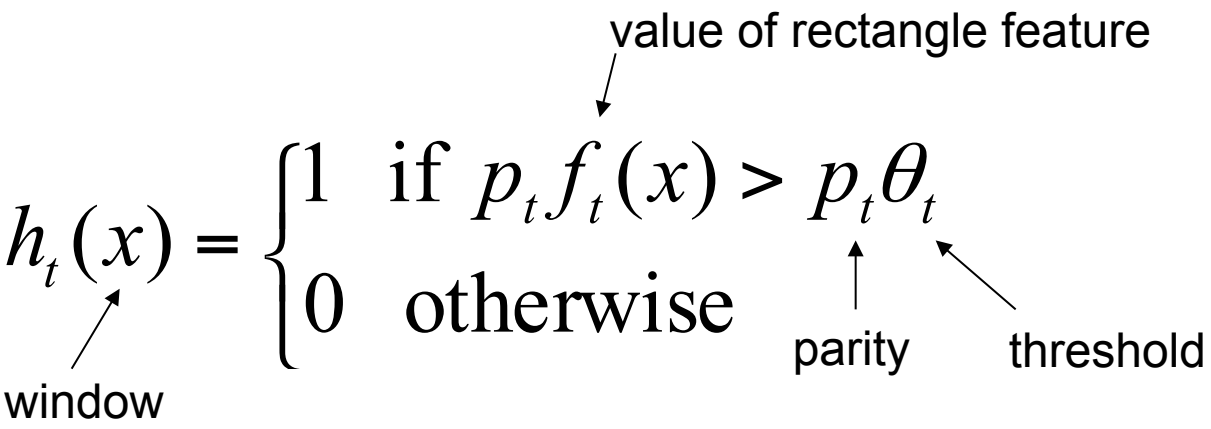
- Define weak learners based on rectangle features

- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples

- Computational complexity of learning: $O(MNK)$
  - M rounds, N examples, K features

Boosting for Face Detection

Define weak learners based on rectangle features



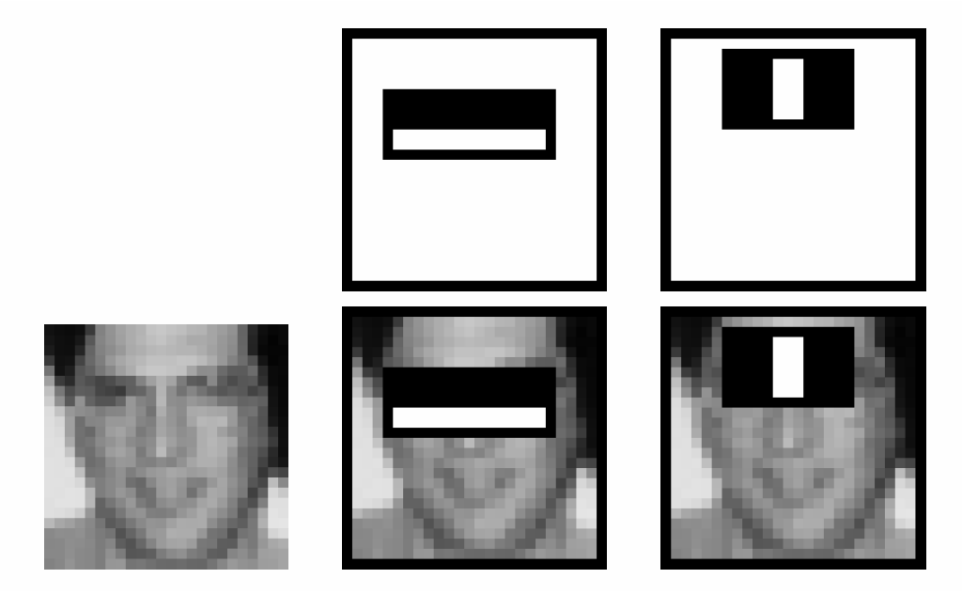
x is a 24x24 sub-window of an image

Boosting Algorithm

- 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,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error 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:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Boosting for Face Detection

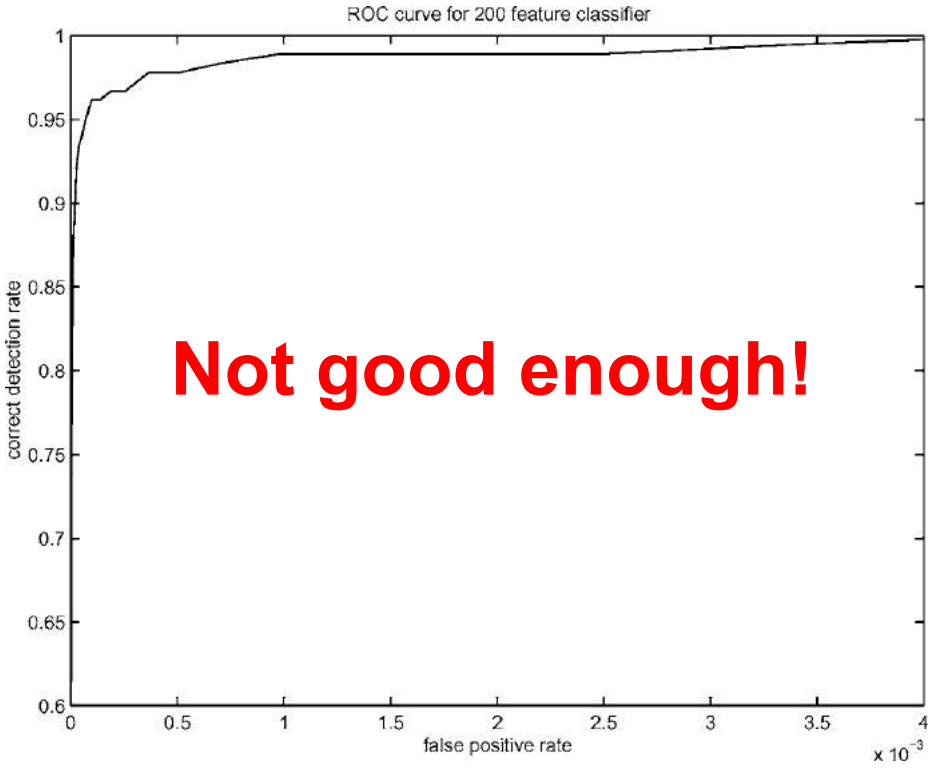
- First two features selected by boosting:



- This feature combination can yield 100% detection rate and 50% false positive rate

Boosting for face detection

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

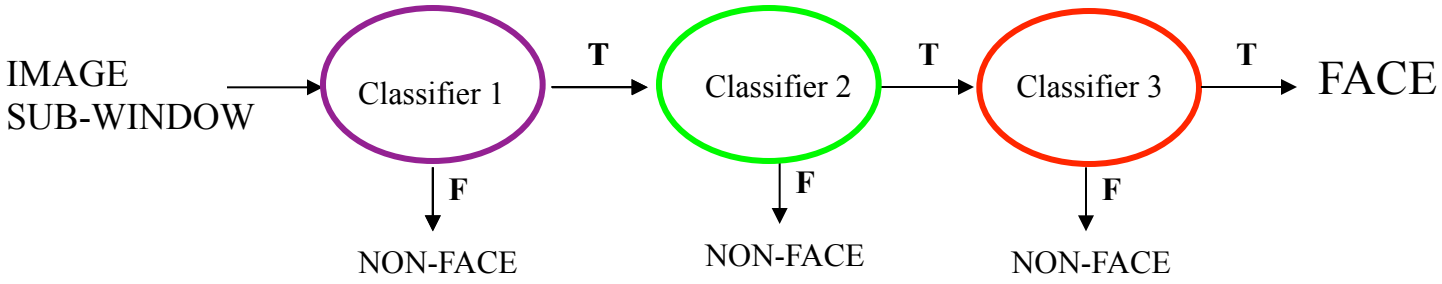


Receiver operating characteristic (ROC) curve

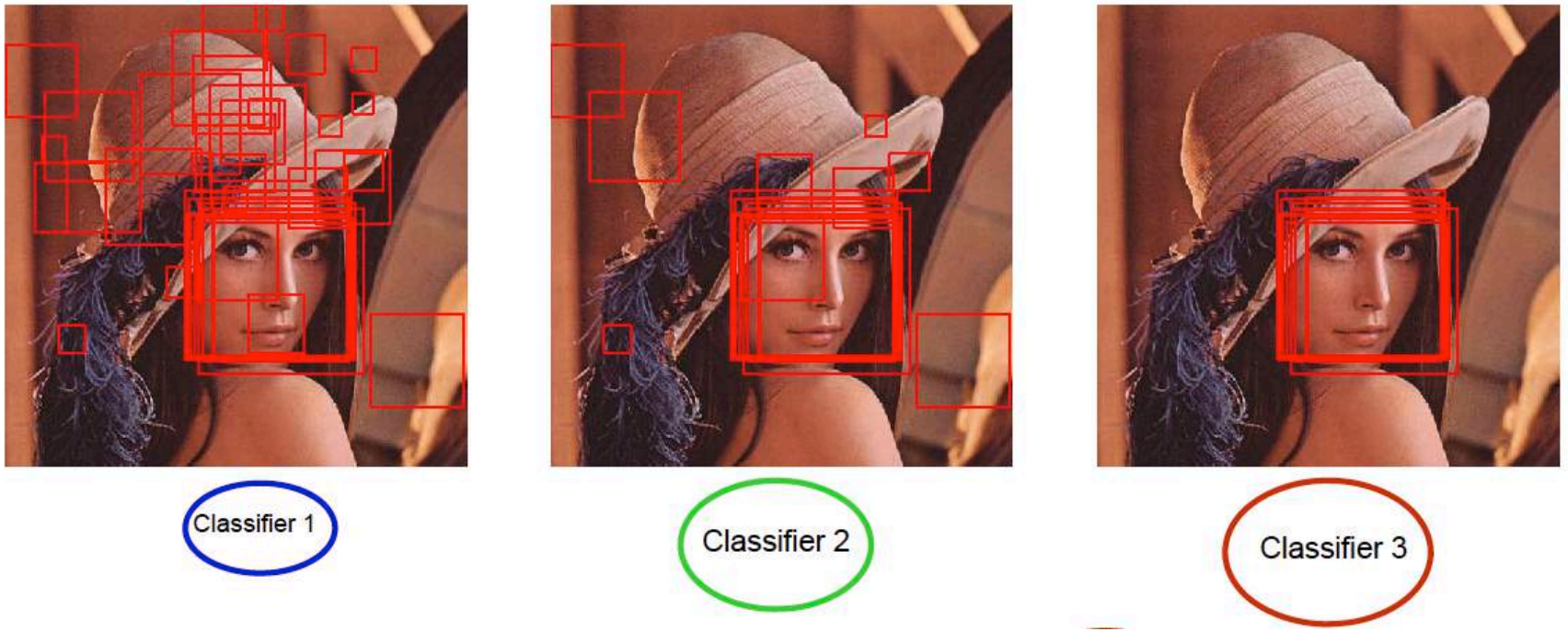
Unfortunately, the most straightforward technique for improving detection performance, adding features to the classifier, directly increases computation time.

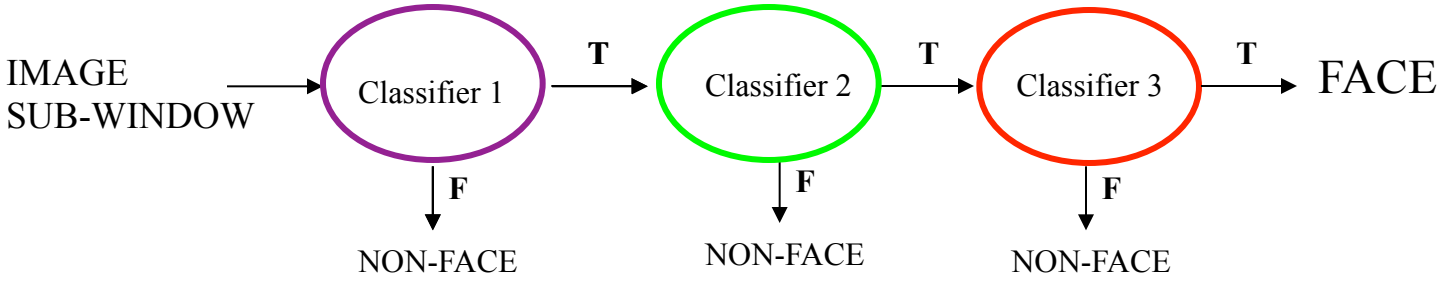
Attentional Cascade

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



Attentional Cascade

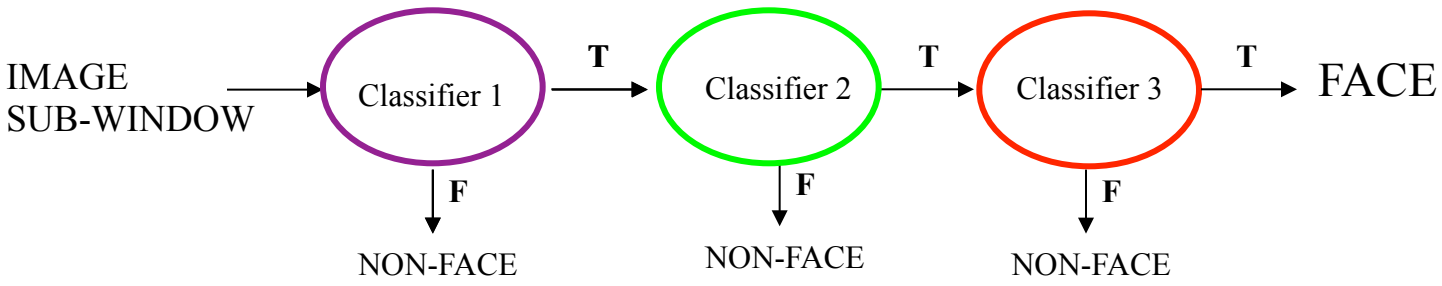




Attentional Cascade

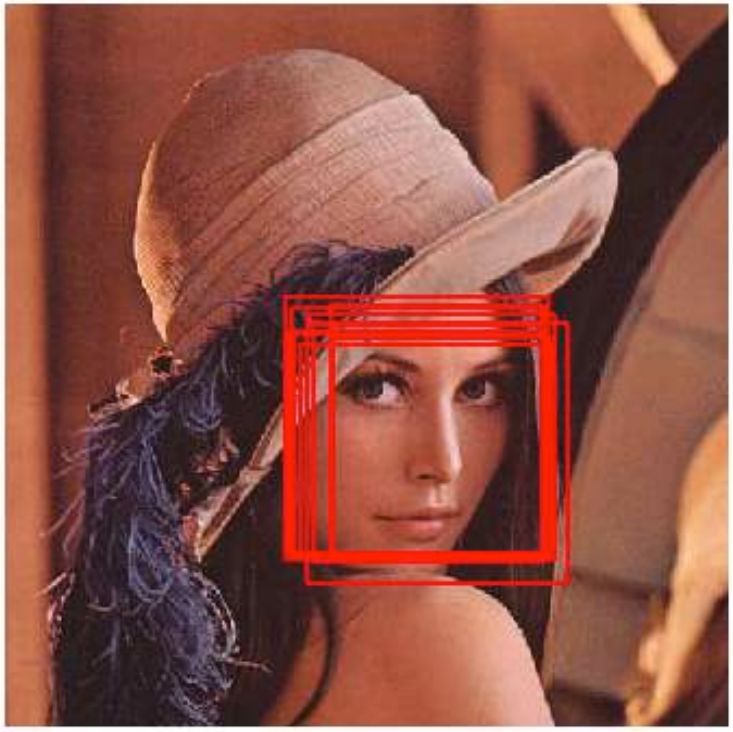
- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages

- A detection rate of 0.9 and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$)



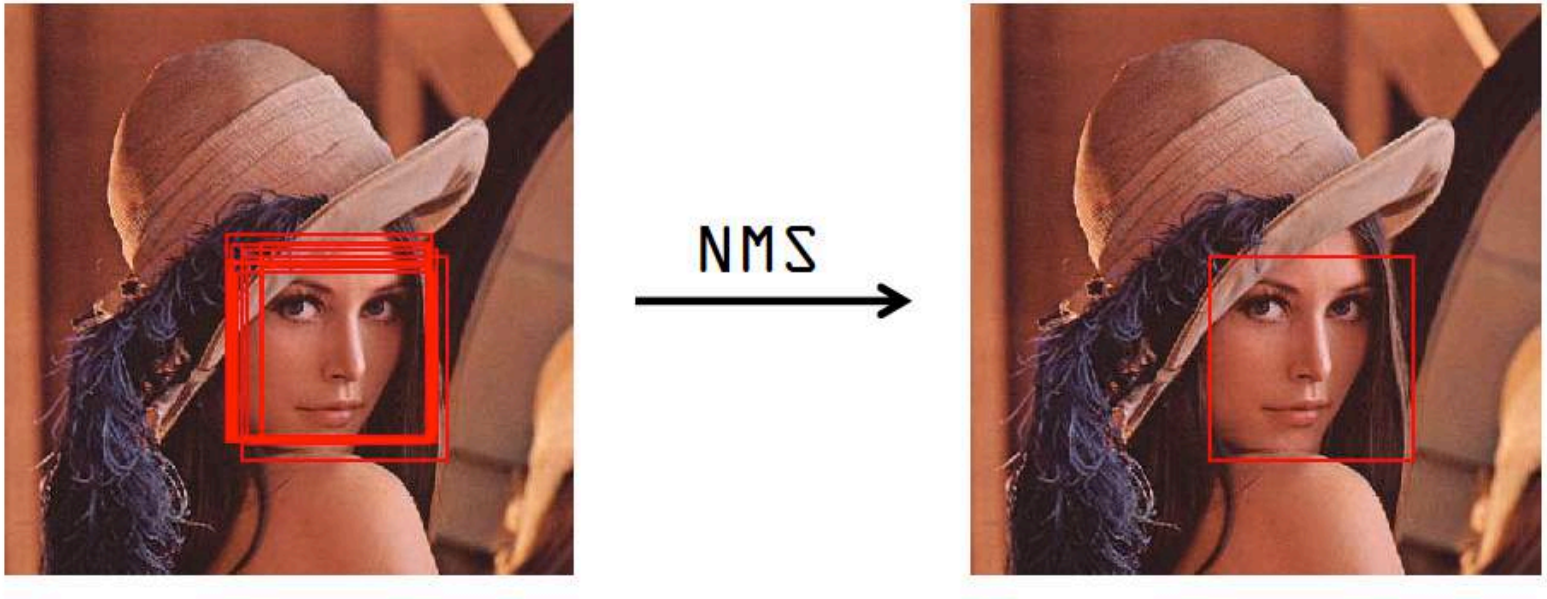
Multiple Detections

- There will typically be several detections for a single face
- This is true for all appearance-based methods discussed



Non-maximum Suppression

- The set of detections are first partitioned into disjoint subsets
- Two detections are in the same subset if their regions overlap
- Each partition yields a single final detection
- The corners of the final bounding region are the average of the corners of all detections in the set



The Implemented System

Training Data

- 4916 hand labeled faces
- 10000 non faces
- Faces are normalized
  - Scale, translation

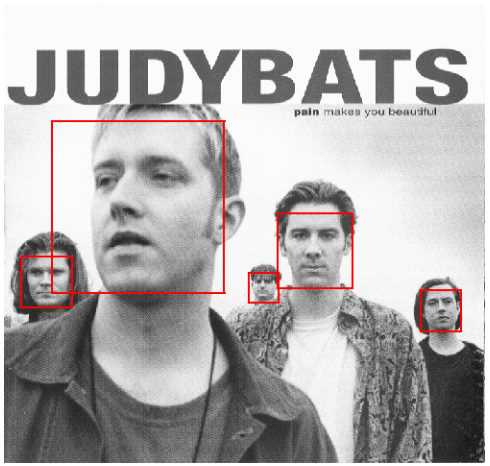
Many variations

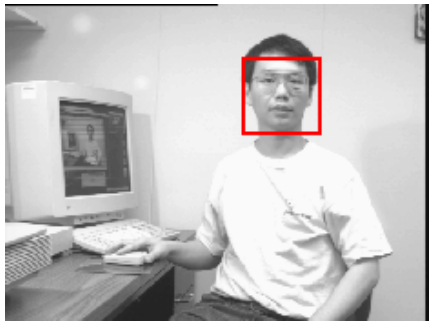
- Across individuals
- Illumination
- Pose (rotation both in plane and out)

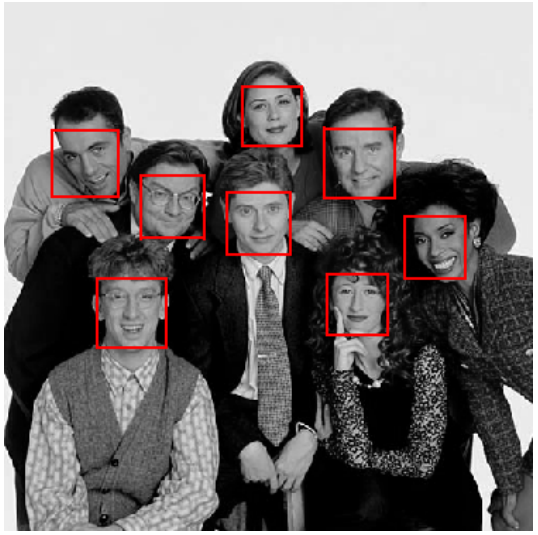
System Performance

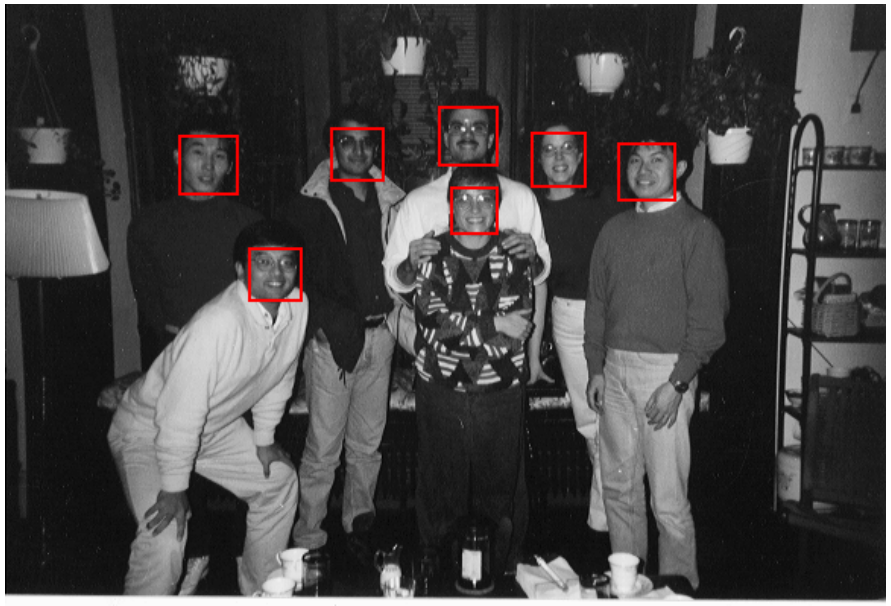
- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

Viola-Jones at Work









Viola-Jones at Work