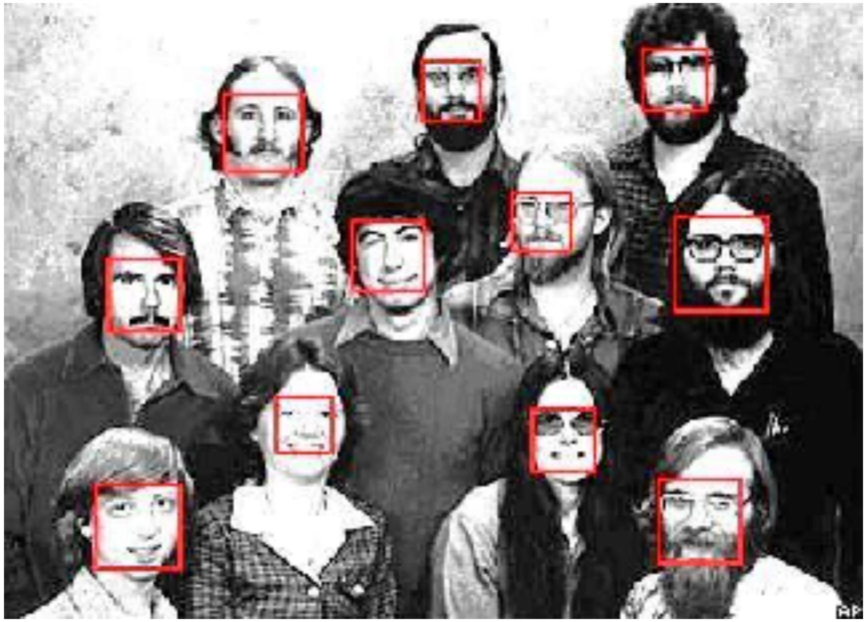
ICCS413 - Lecture 18

Ensemble Learning and Face Detection

Sunsern Cheamanunkul

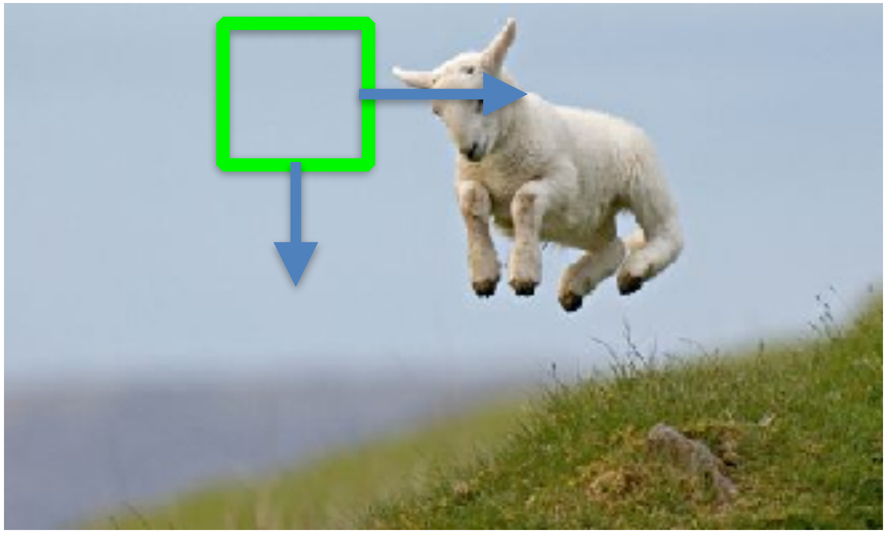
Mahidol University International College

Face Detection



Basic Concepts

- Slide a window across the image.
- Test each window for a face



Challenges

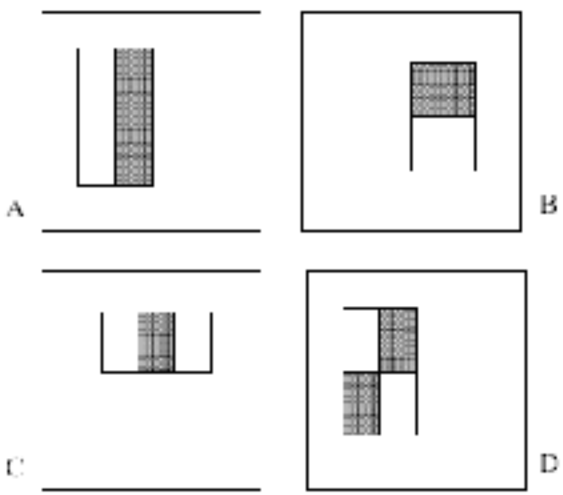
- Sliding window detector must evaluate >= 10k location/scale combinations
- Faces are rare: 0-10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A hi-res image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having false positives in every image, the false positive rate has to be very small (less than 10^{-6})

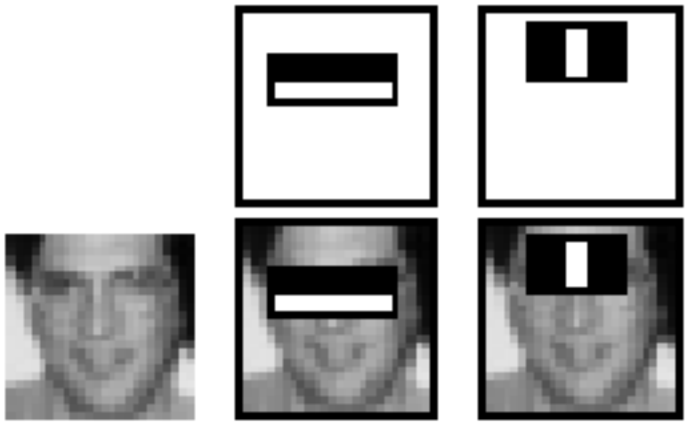
Viola-Jones Face Detector

- Robust - High true positive rate while maintaining low false positive rate
- Real-time - For practical applications at least 2 frames per second must be processed.
- Face Detection - not recognition. The goal is to distinguish faces from non-faces (face detection is the first step in the identification process)

Features

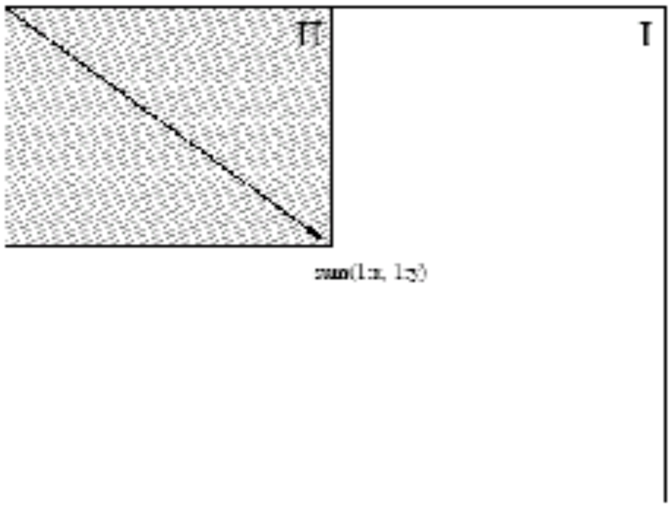
- Four basic types
  - Simple to calculate
  - The white areas are subtracted from the black ones.
  - A special representation of the sample called the integral image makes feature extraction faster.





Integral Images

- Summed area tables



- A representation that means any rectangle's values can be calculated in four accesses of the integral image.

Fast Pixel Summation

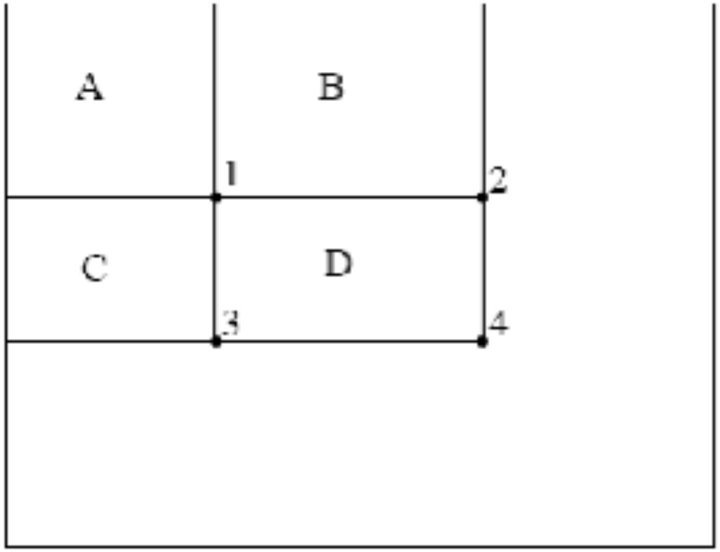
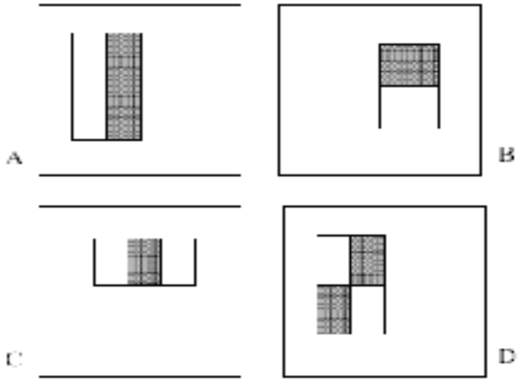


Figure 3: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A+B$, at location 3 is $A+C$, and at location 4 is $A+B+C+D$. The sum within D can be computed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

Feature extraction

- Features are extracted from each sub-window of the image.
  - The base size for a sub-window is 24 by 24 pixels.
  - Each of the four feature types are scaled and shifted across all possible combinations
- In a 24 pixel by 24 pixel sub window there are ~160,000 possible features to be calculated.



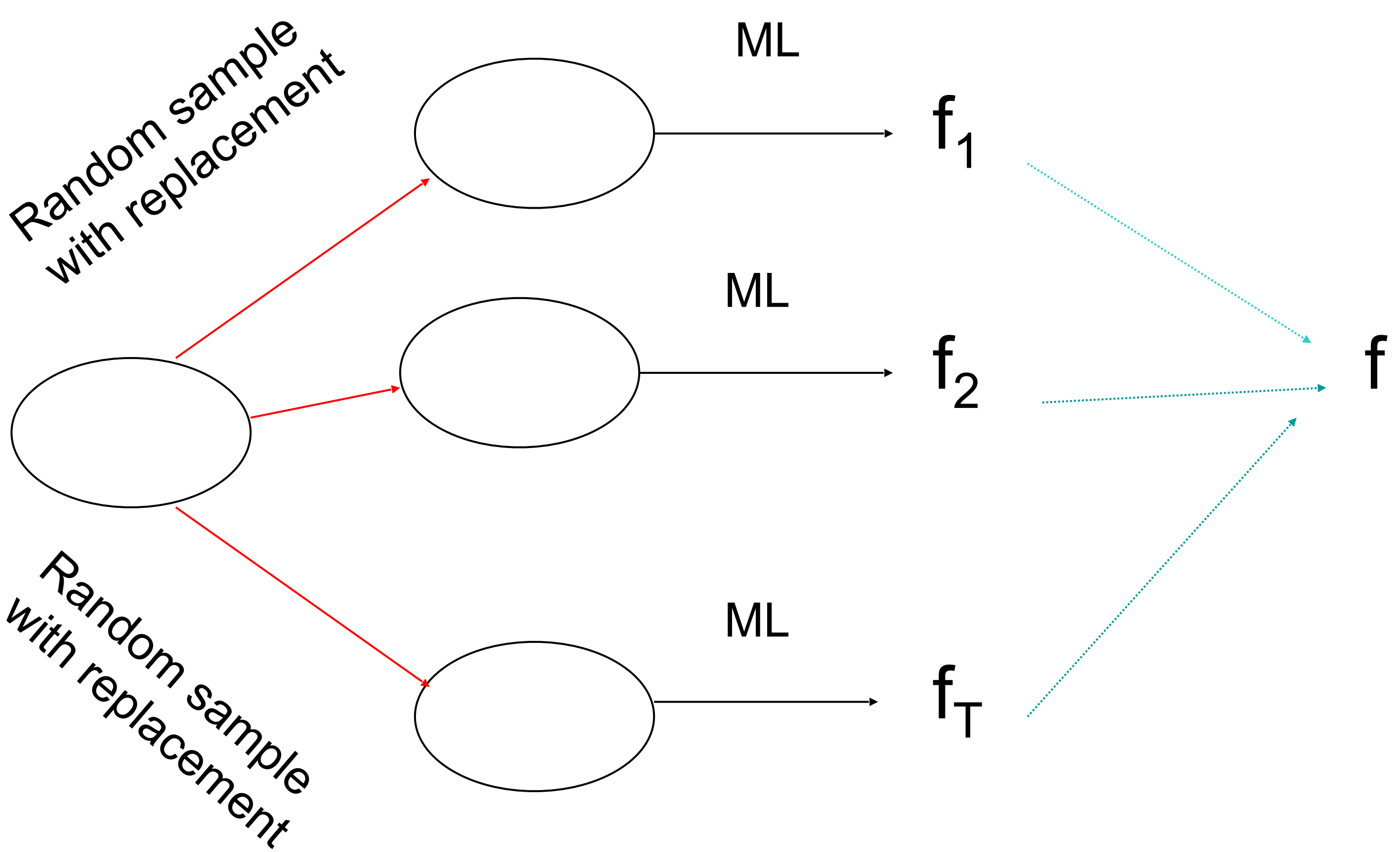
Challenges of the Learning Task

- High number of features (~160,000)
- Only a few hundreds training examples
- Which features to use?
- How to avoid overfitting?
- Boosting!
    - Combine many weak classifiers into a strong classifier

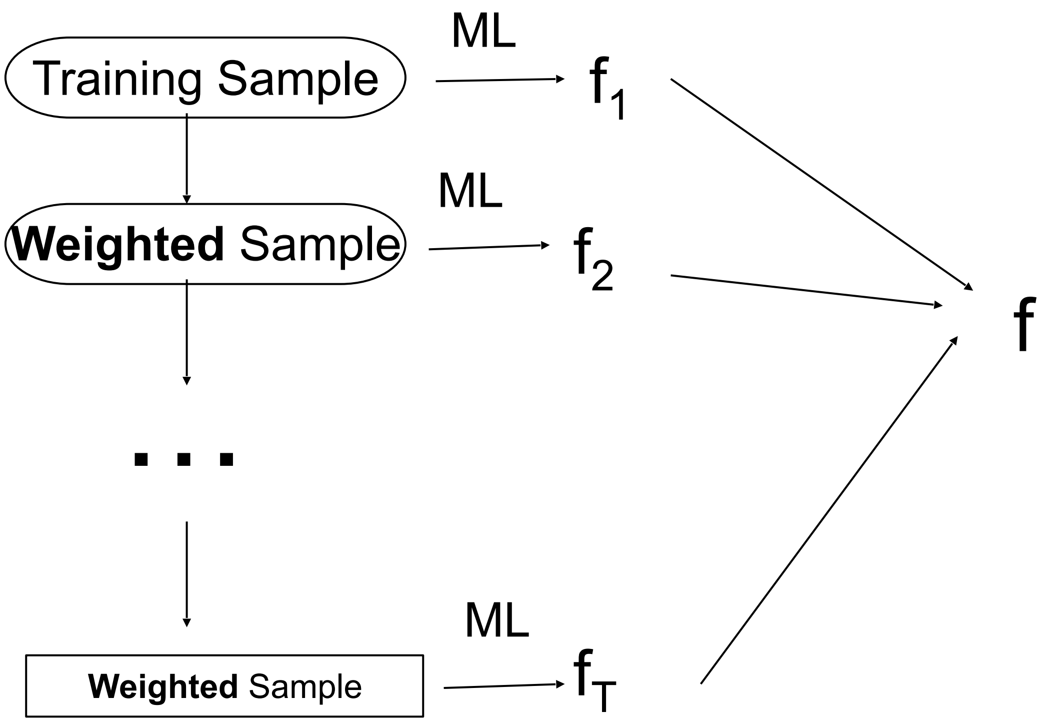
Ensemble Learning

- a process by which **multiple models**, such as classifiers or experts, are strategically **generated and combined** to solve a particular computational intelligence problem such as classification, prediction, function approximation.
- Examples:
  - Bagging
  - Boosting

Bagging


Random sample
with replacement
ML
f1
ML
f2
f
ML
fT
Random sample
with replacement

Boosting


Training Sample
ML
f_1
Weighted Sample
ML
f_2
...
Weighted Sample
ML
f_T
f

Overview of boosting

- Introduced by Schapire and Freund in 1990s.
- “Boosting”: convert a weak learning algorithm into a strong one.
- Main idea: Combine many “weak” classifiers to produce a “strong” classifier.
- Algorithms:
  - **AdaBoost**: adaptive boosting
  - Gentle Boost
  - BrownBoost
  - Logit Boost
  - etc.

Intuition

- Train a set of weak hypotheses: h_1, …., h_T.
- The combined hypothesis H is a **weighted** majority vote of the T weak hypotheses.
  ➔Each hypothesis h_t has a weight α_t.

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

- During the training, focus on the examples that are misclassified.
  ➔ At round t, example x_i has the weight $D_t(i)$.

Basic Setting

- Binary classification problem
- Training data:

$$(x_1, y_1), \ldots, (x_m, y_m), where \;\; x_i \in X, y_i \in Y = \{-1, 1\}$$

- $D_t(i)$: the weight of x_i at round t. $D_1(i)=1/m$.

- A learner L that finds a weak hypothesis h_t: X ➔ Y given the training set and D_t

- The error of a weak hypothesis h_t:

$$\epsilon_t \equiv \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] \equiv \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

AdaBoost algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Strengths of AdaBoost

- It has no parameters to tune (except for the number of rounds)
- It is fast, simple and easy to program.
- It comes with a set of theoretical guarantee (e.g., training error, test error)
- Instead of trying to design a learning algorithm that is accurate over the entire space, we can focus on finding base learning algorithms that only need to be better than random.
- It can identify outliners: i.e. examples that are either mislabeled or that are inherently ambiguous and hard to categorize.

Weakness of AdaBoost

- The actual performance of boosting depends on the data and the base learner.
- AdaBoost is sensitive to noise.
- When the number of outliners is very large, the emphasis placed on the hard examples can hurt the performance.

  → "Gentle AdaBoost", "BrownBoost"

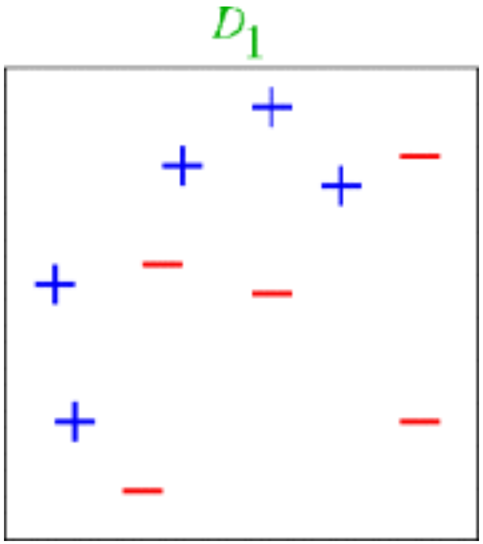
Bagging vs. Boosting (Freund and Schapire 1996)

- Bagging always uses resampling rather than reweighting.
- Bagging does not modify the distribution over examples or mislabels, but instead always uses the uniform distribution
- In forming the final hypothesis, bagging gives equal weight to each of the weak hypotheses

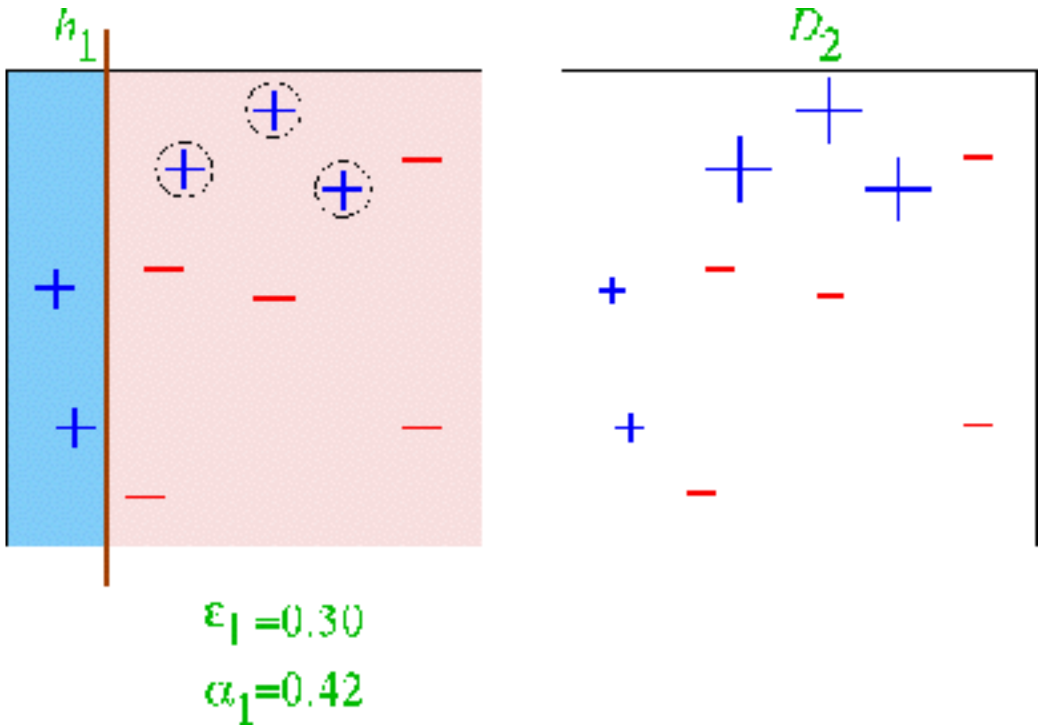
Weak Classifier

- Decision stumps = decision tree with only a single root node
  - Certainly a very weak learner!
  - Say the attributes are real-valued
  - Decision stump algorithm looks at all possible thresholds for each attribute
  - Selects the one with the max information gain
  - Resulting classifier is a simple threshold on a single feature
    - Outputs a +1 if the attribute is above a certain threshold
    - Outputs a -1 if the attribute is below the threshold

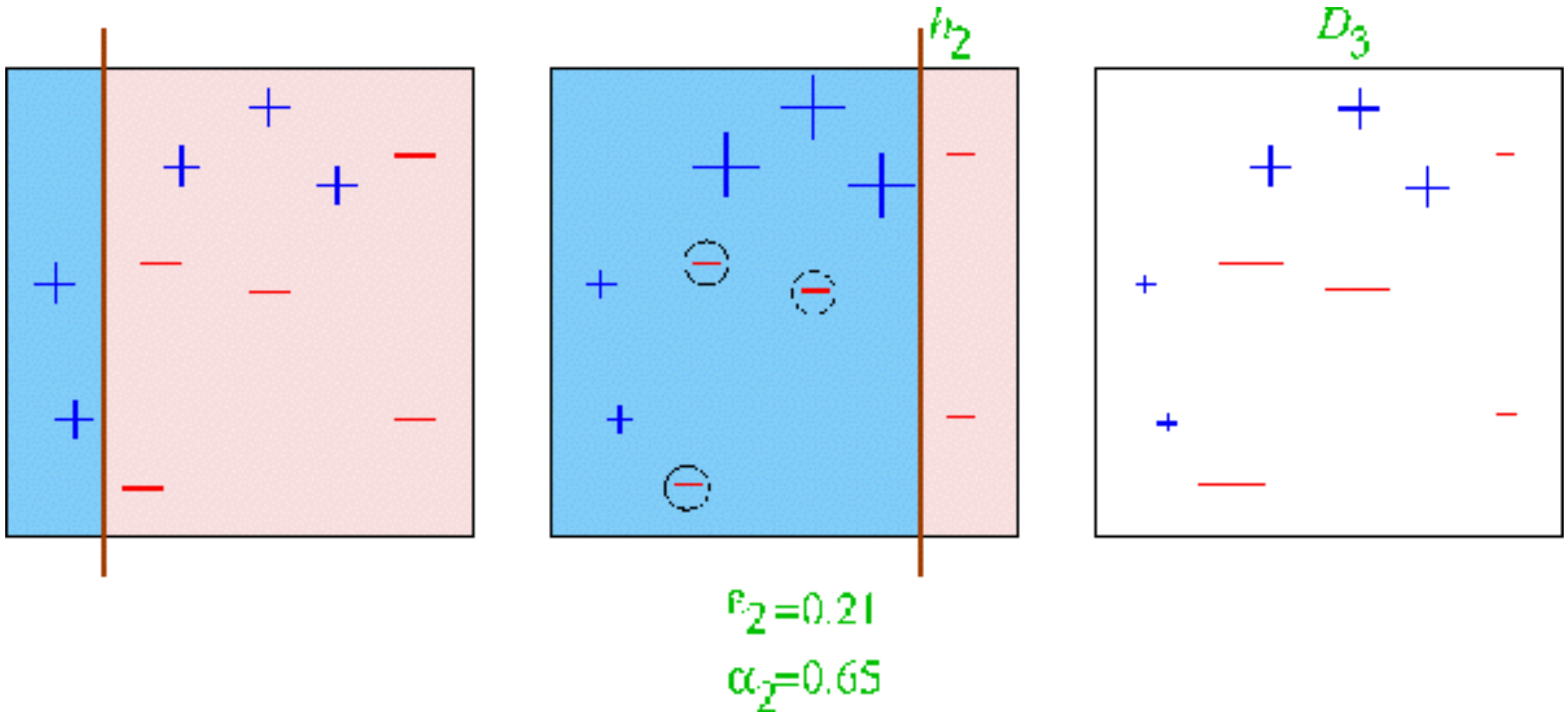
Boosting Example



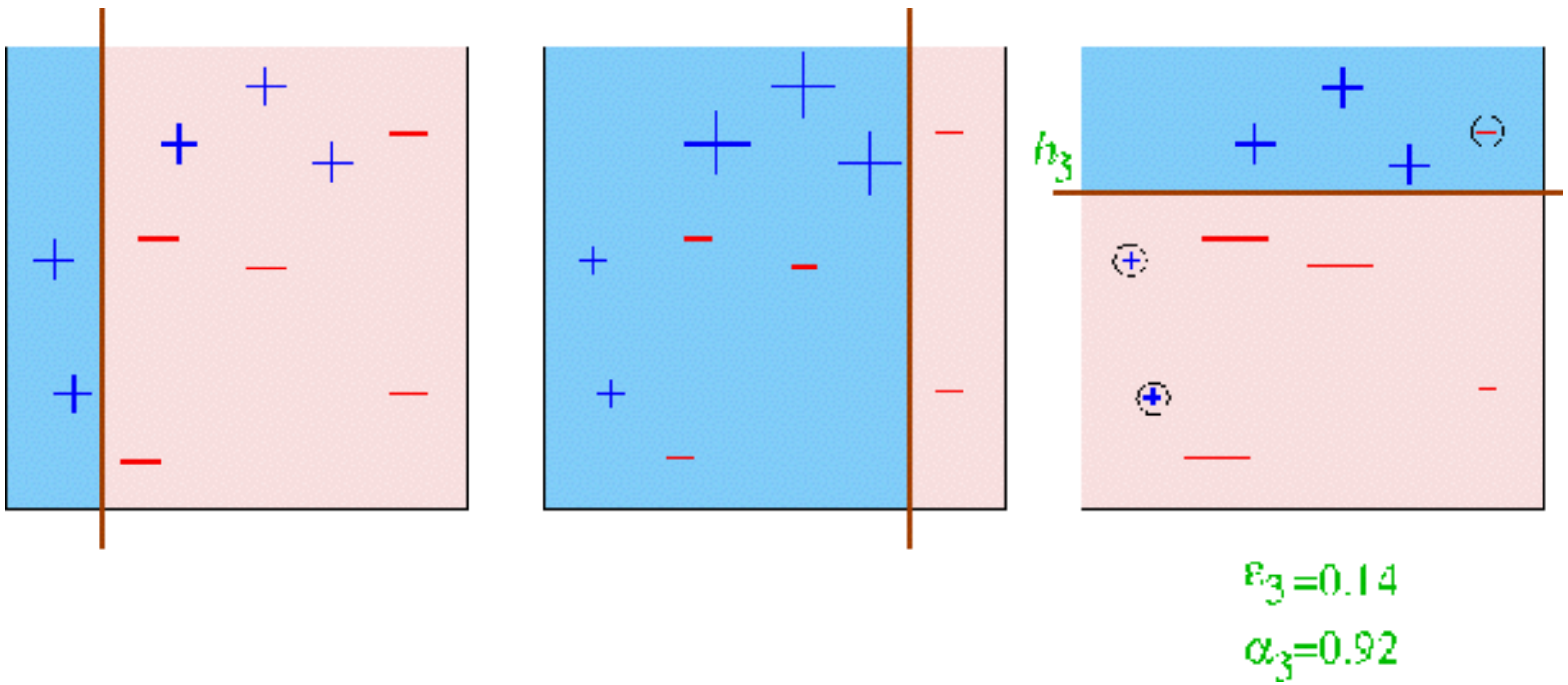
After 1st Rule



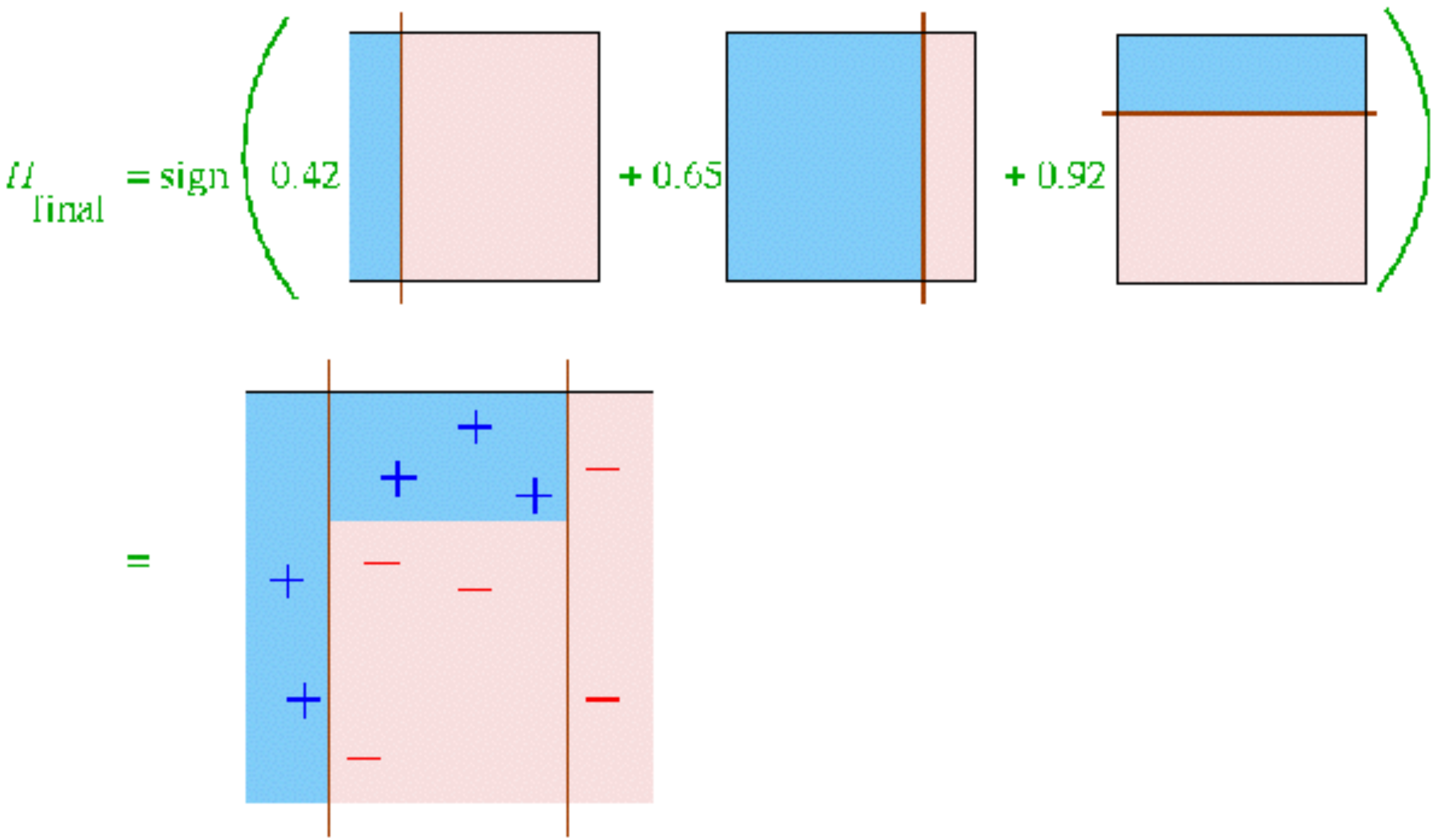
After 2nd Rule



After 3rd Rule



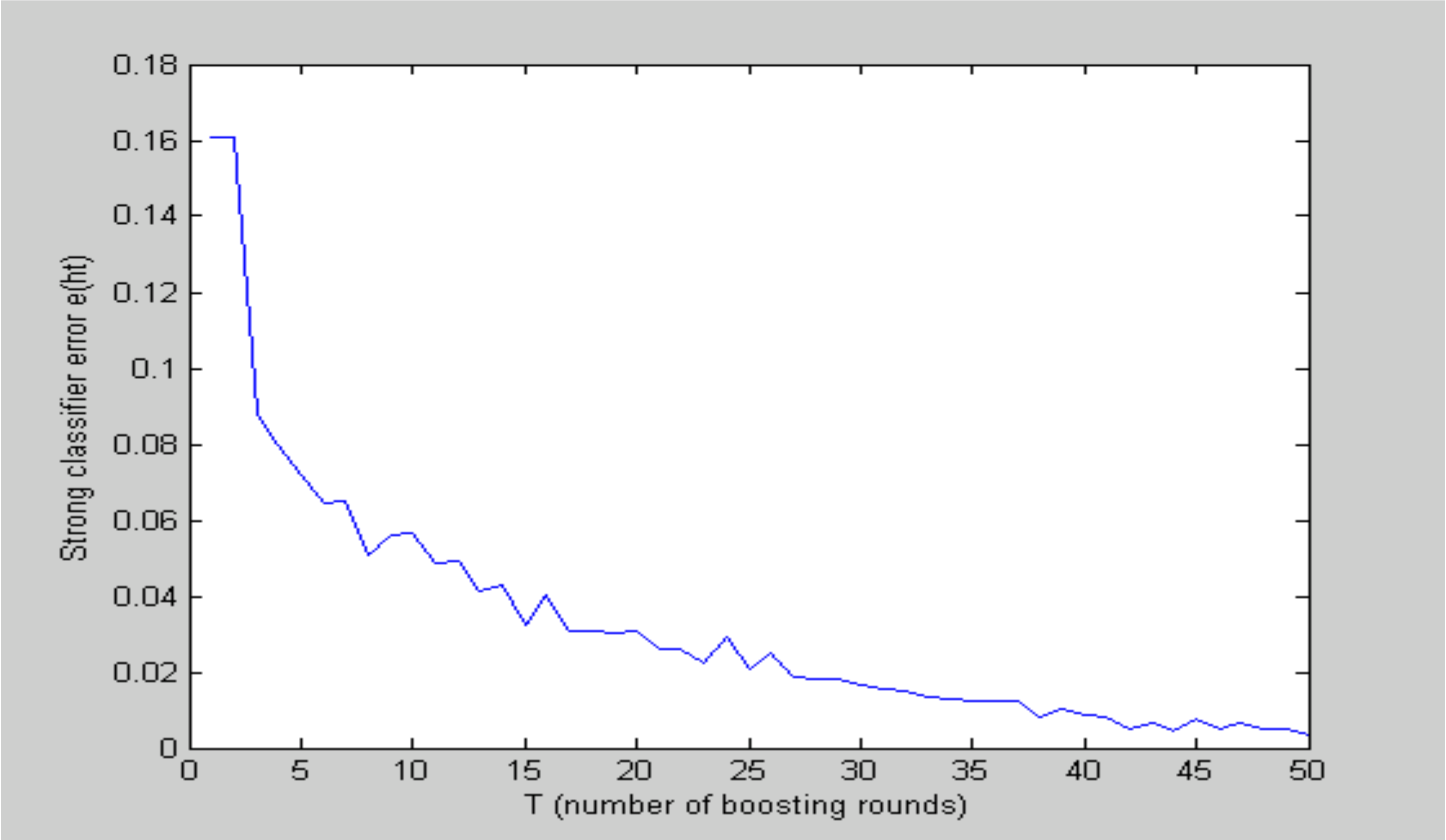
Final Classifier



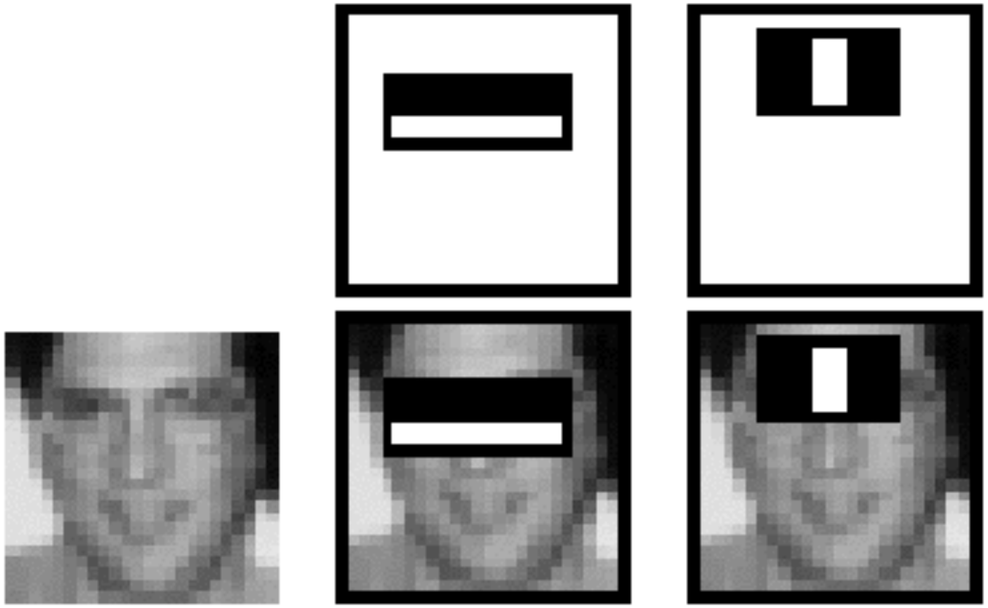
Boosting with Decision Stumps

- Viola-Jones algorithm
  - With K attributes (e.g., K = 160,000) we have 160,000 different decision stumps to choose from
  - At each stage of boosting
    - given reweighted data from previous stage
    - Train all K (160,000) single-feature stumps
    - Select the single best classifier at this stage
    - Combine it with the other previously selected classifiers
    - Reweight the data
    - Learn all K classifiers again, select the best, combine, reweight
    - Repeat until you have T classifiers selected
  - Very computationally intensive
    - Learning K decision stumps T times
    - E.g., K = 160,000 and T = 1000

Training error as a function of boosting iterations



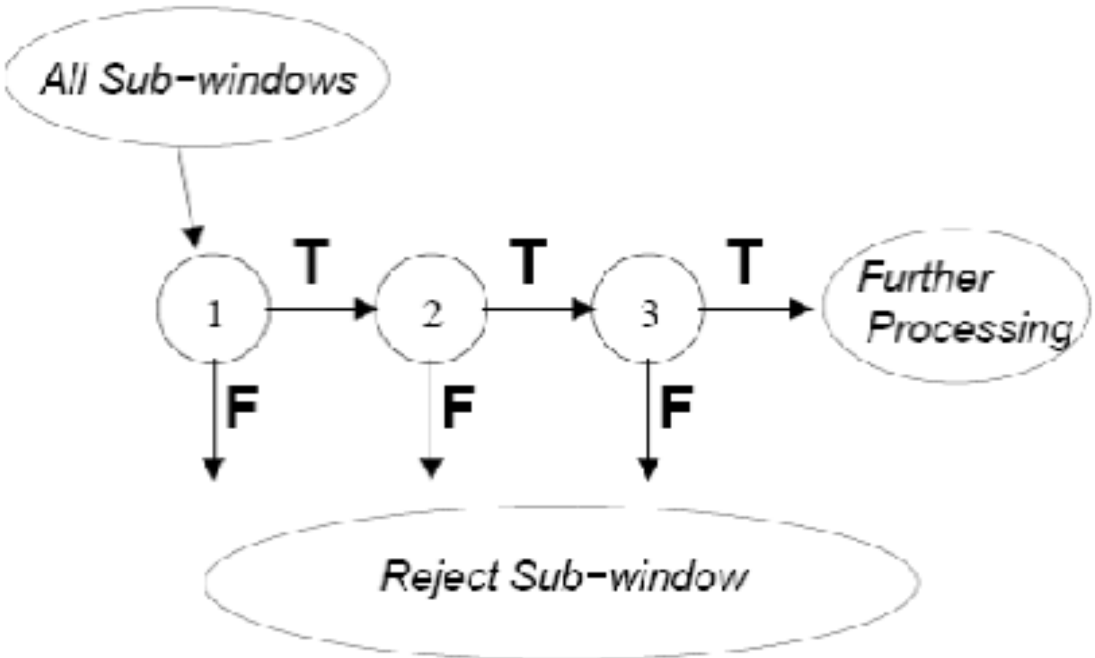
Top features identified by Boosting



Real-time Face Detection

- Basic classifier operates on 24 x 24 subwindows
- Scaling:
  - Scale the detector (rather than the images)
  - Features can easily be evaluated at any scale
  - Scale by factors of 1.25
- Location:
  - Move detector around the image (e.g., 1 pixel increments)
- Final Detections
  - A real face may result in multiple nearby detections
  - Post-process detected subwindows to combine overlapping detections into a single detection

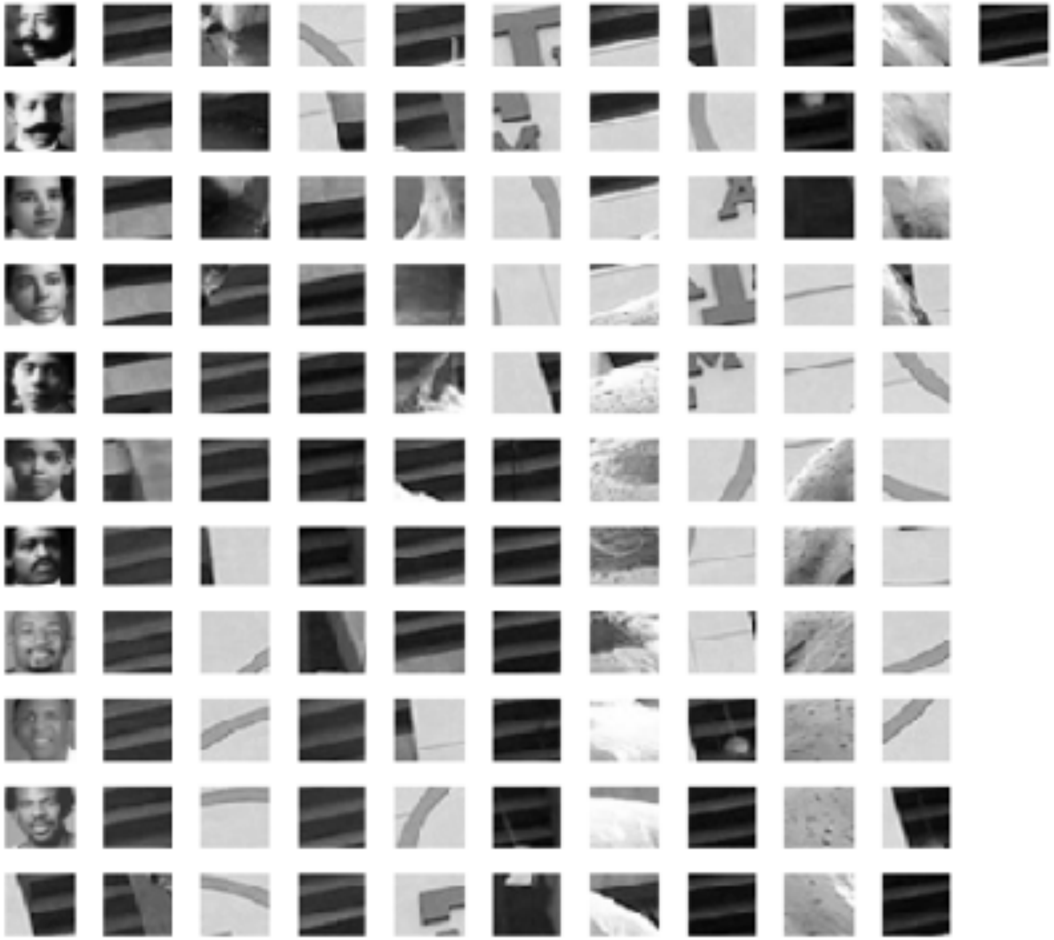
Cascading


All Sub-windows
1
2
3
T
T
T
F
F
F
Further Processing
Reject Sub-window

Training Images

- Examples of 24x24 images with faces

Face vs Nonface



Results