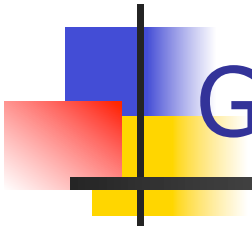


Generic Object-Face detection

Jana Kosecka

Many slides adapted from P. Viola, K. Grauman, S. Lazebnik and many others

Today

- Window-based generic object detection
  - basic pipeline
  - boosting classifiers
  - face detection as case study

Generic category recognition: basic framework

- Build/train object model
  - Choose a representation
  - Learn or fit parameters of model / classifier
- Generate candidates in new image
- Score the candidates

Window-based models
Building an object model

Given the representation, train a binary classifier

Car/non-car Classifier

Yes, car.

No, not a car.

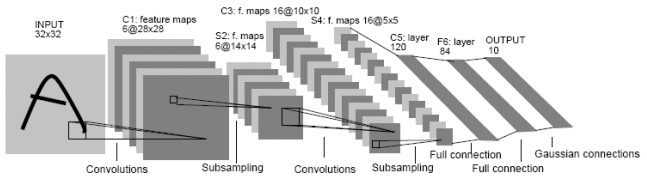
Discriminative classifier construction

Nearest neighbor

10^6 examples

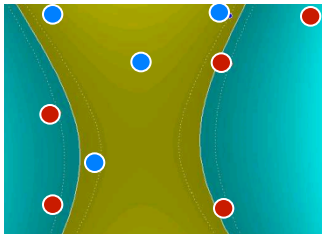
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



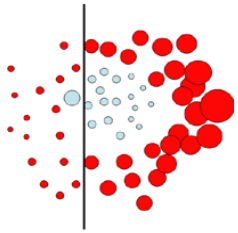
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



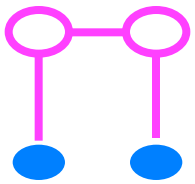
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006,…

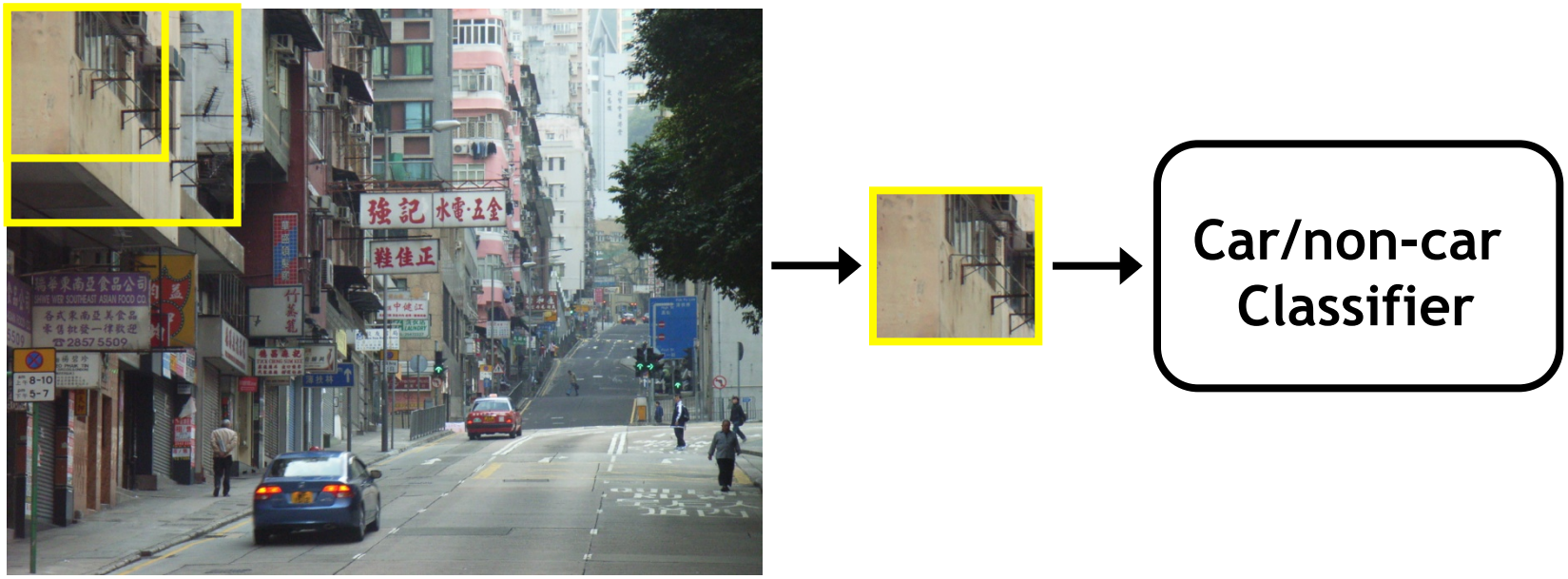
Conditional Random Fields



McCallum, Freitag, Pereira 2000; Kumar, Hebert 2003
…

Slide adapted from Antonio Torralba

Window-based models
Generating and scoring candidates



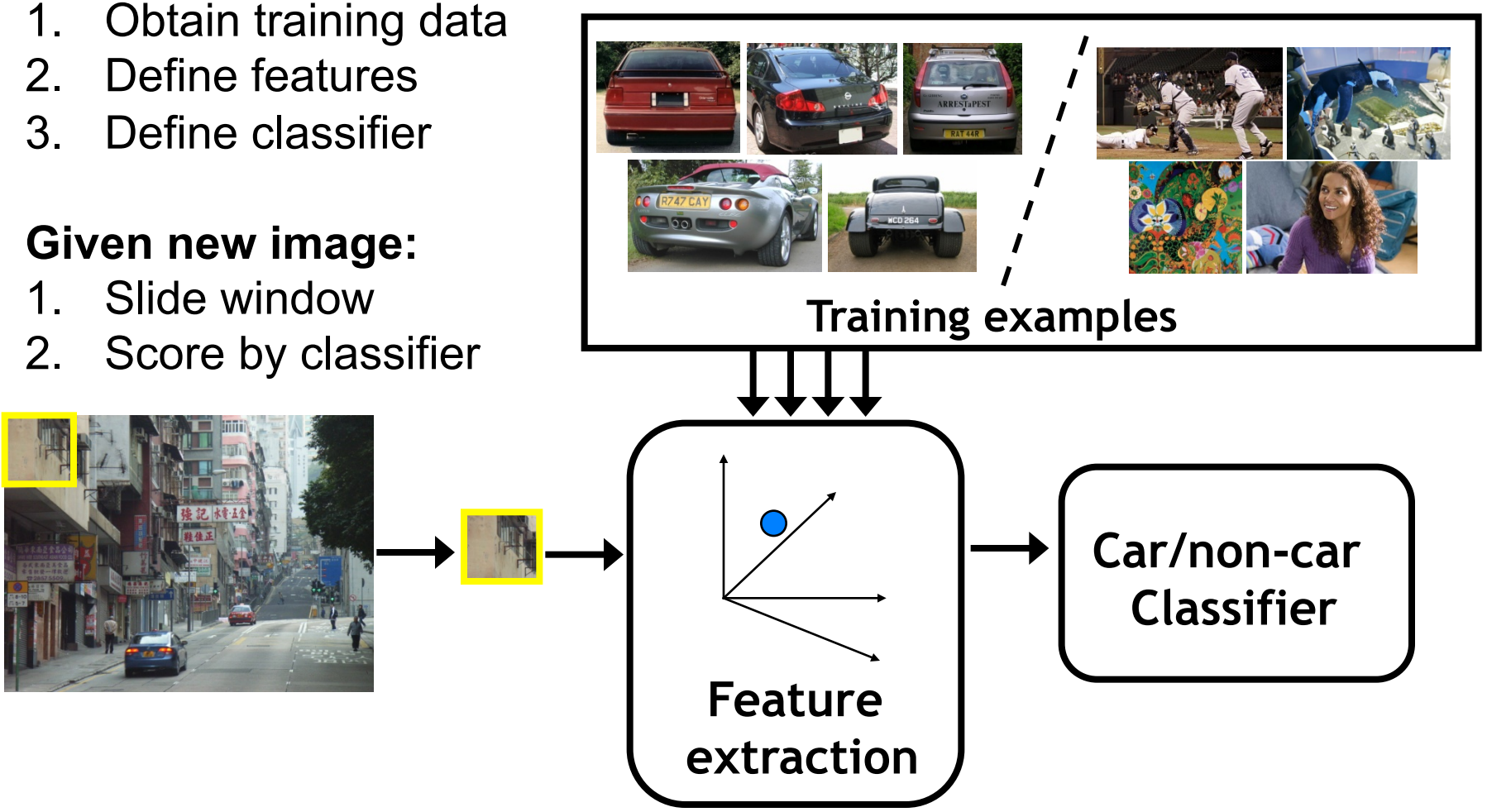
Window-based object detection: recap

Training:

1. Obtain training data
2. Define features
3. Define classifier

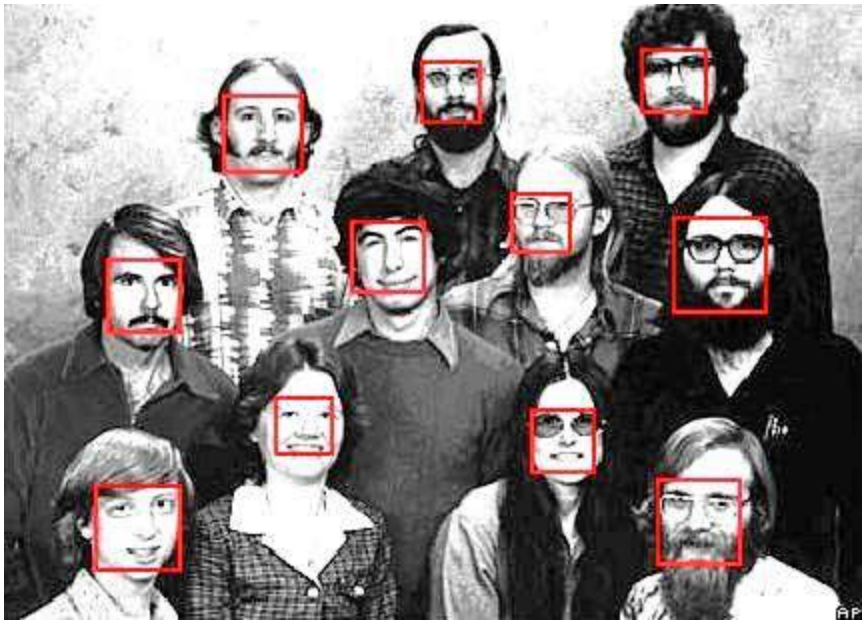
Given new image:

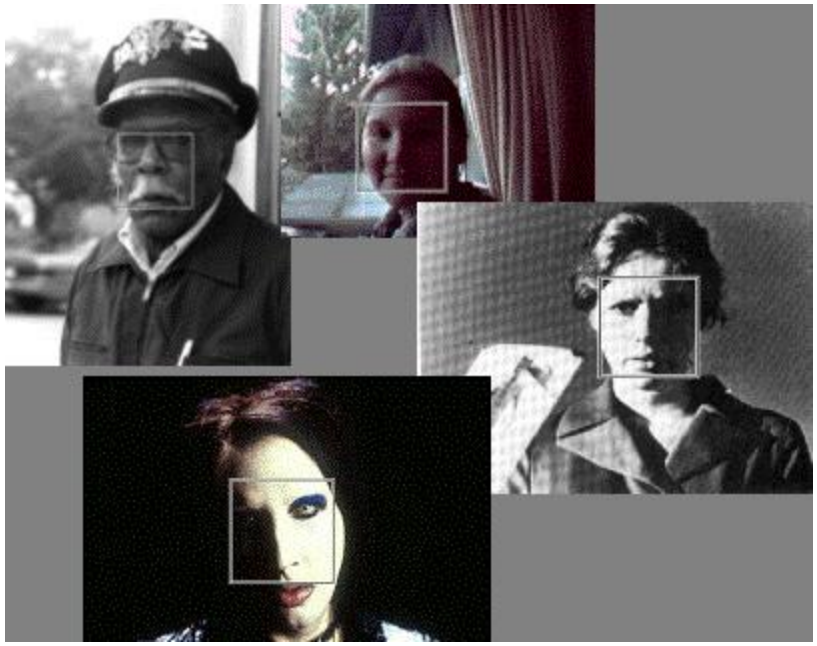
1. Slide window
2. Score by classifier



Kristen Grauman

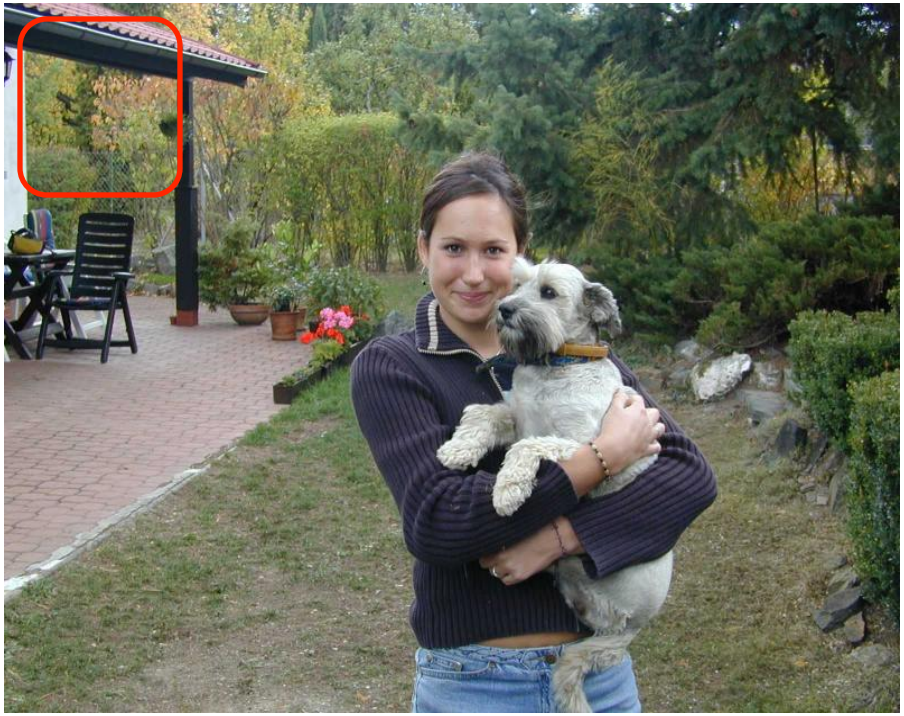
Face detection



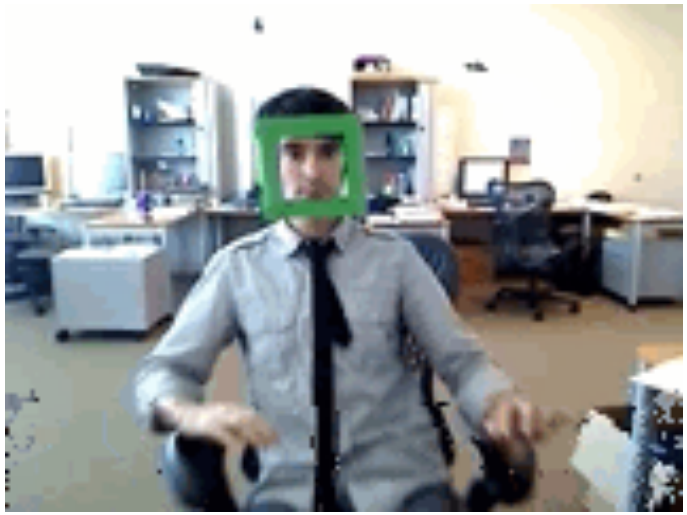


Face detection

- Basic idea: slide a window across image and evaluate a face model at every location



Face detection



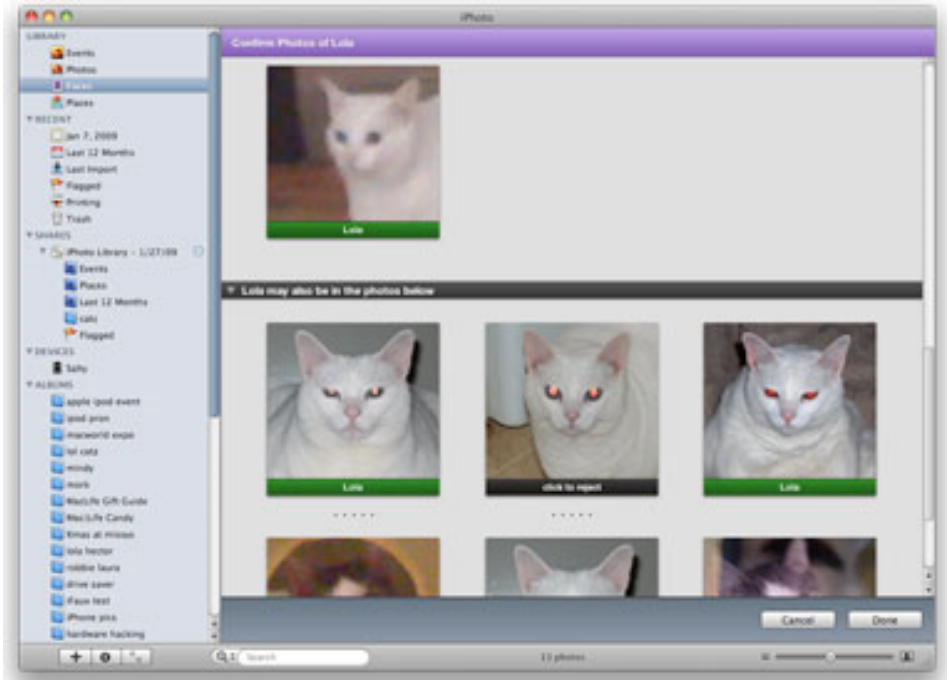
Behold a state-of-the-art face detector!
(Courtesy Boris Babenko)

Consumer application: Apple iPhoto

http://www.apple.com/ilife/iphoto/

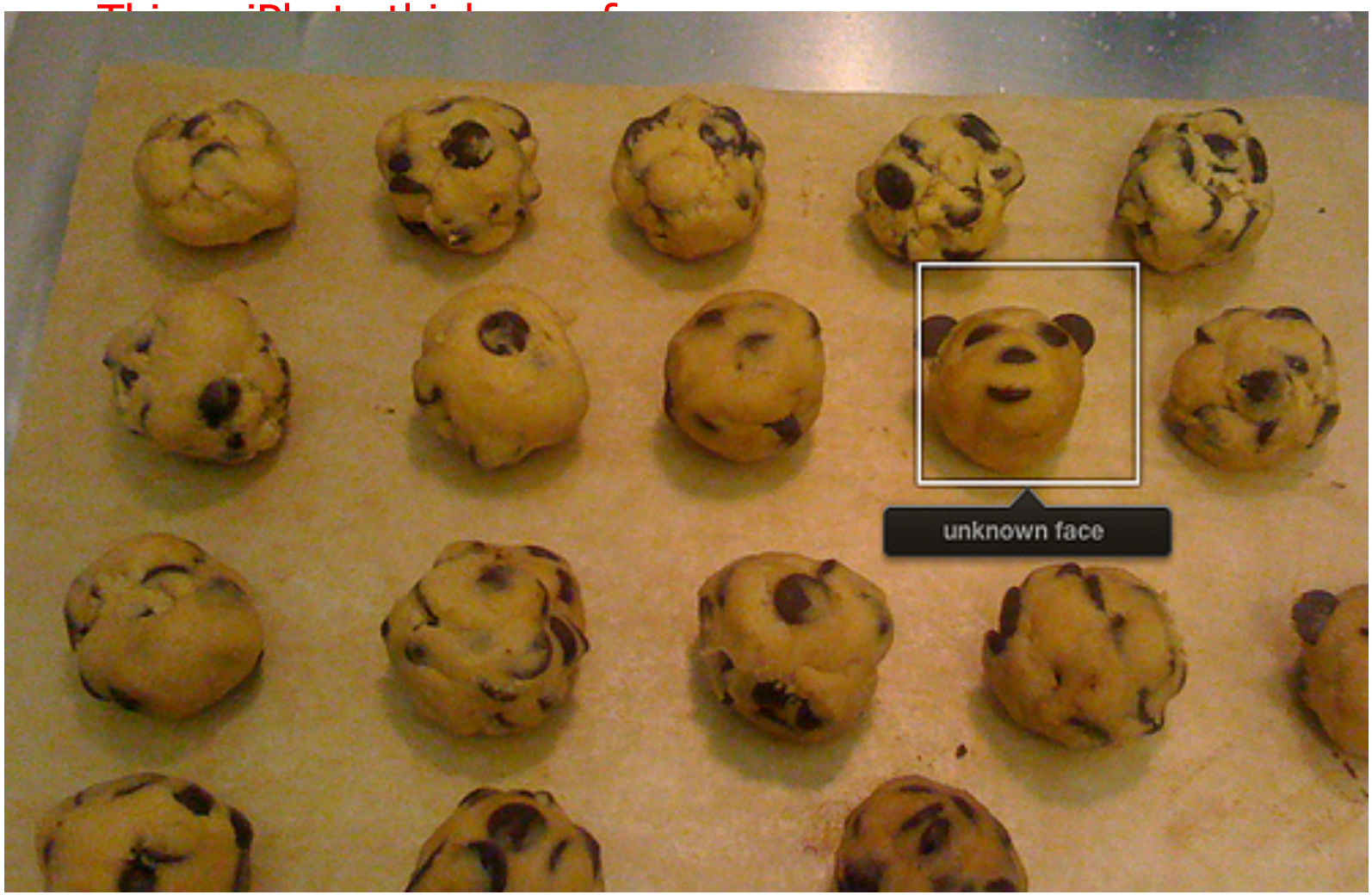
Consumer application: Apple iPhoto

- Can be trained to recognize pets!



http://www.maclife.com/article/news/iphotos_faces_recognizes_cats

Consumer application: Apple iPhoto



Funny Nikon ads

"The Nikon S60 detects up to 12 faces."

Funny Nikon ads

"The Nikon S60 detects up to 12 faces."

Challenges of face detection

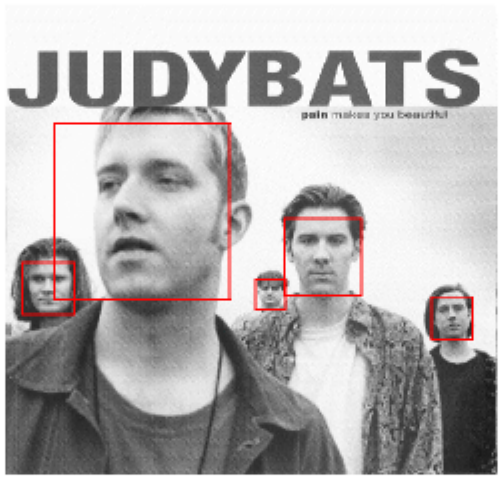
- Sliding window detector must evaluate tens of thousands of location/ scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate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

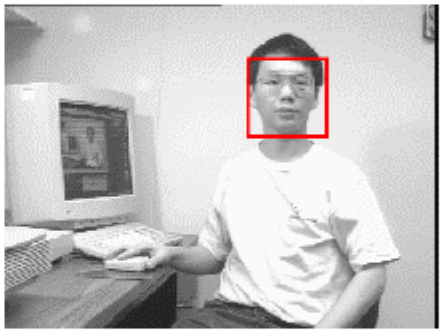
The Viola/Jones Face Detector

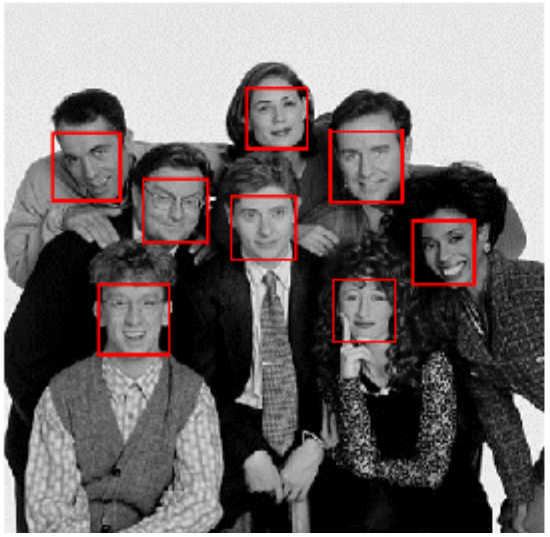
- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

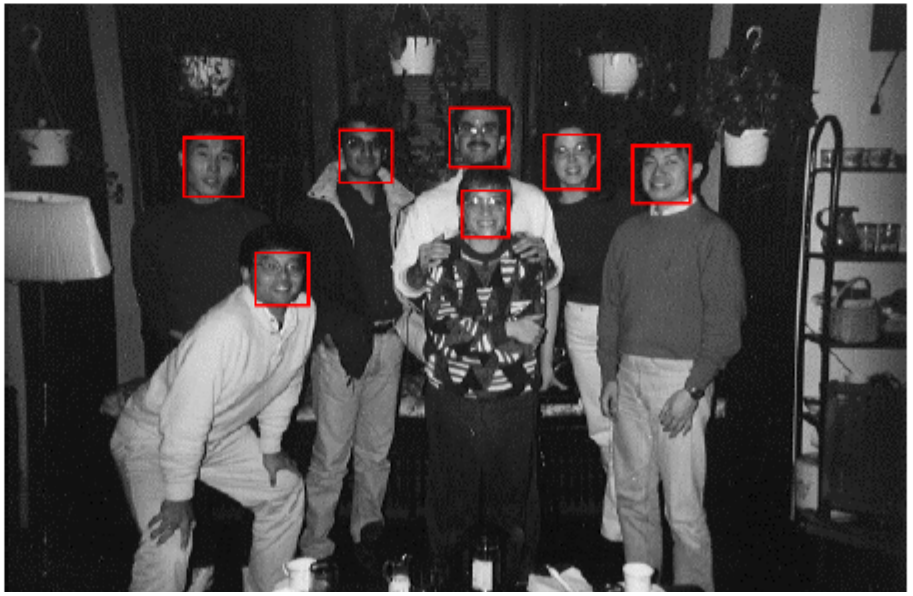
P. Viola and M. Jones.
Rapid object detection using a boosted cascade of simple features. CVPR 2001.
P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Viola-Jones Face Detector: Results

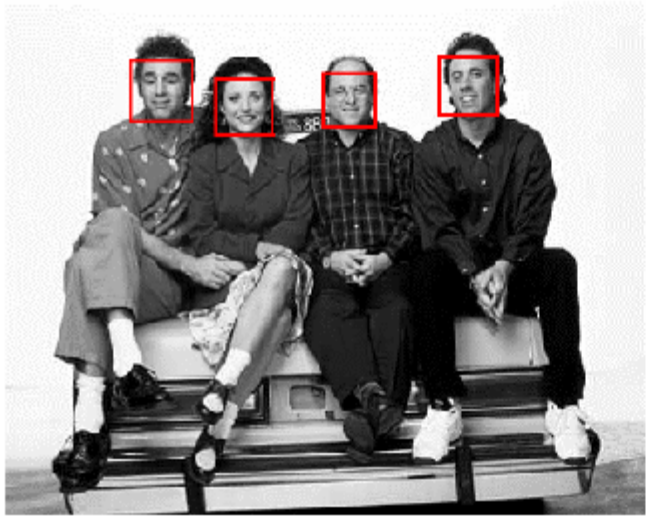


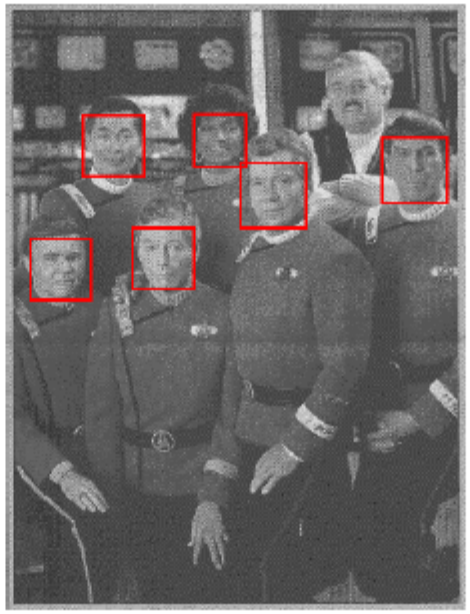


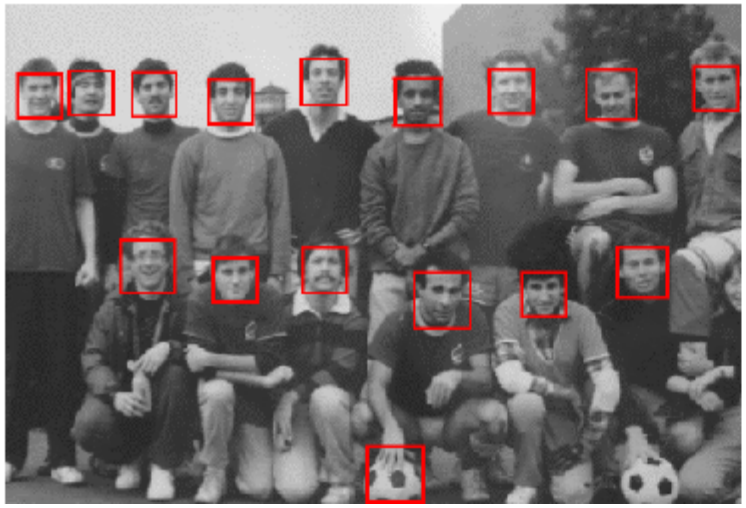


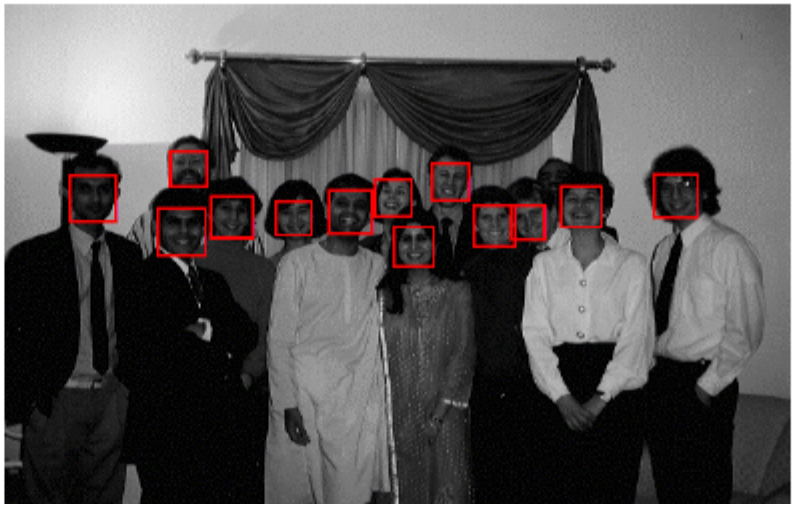


Viola-Jones Face Detector: Results





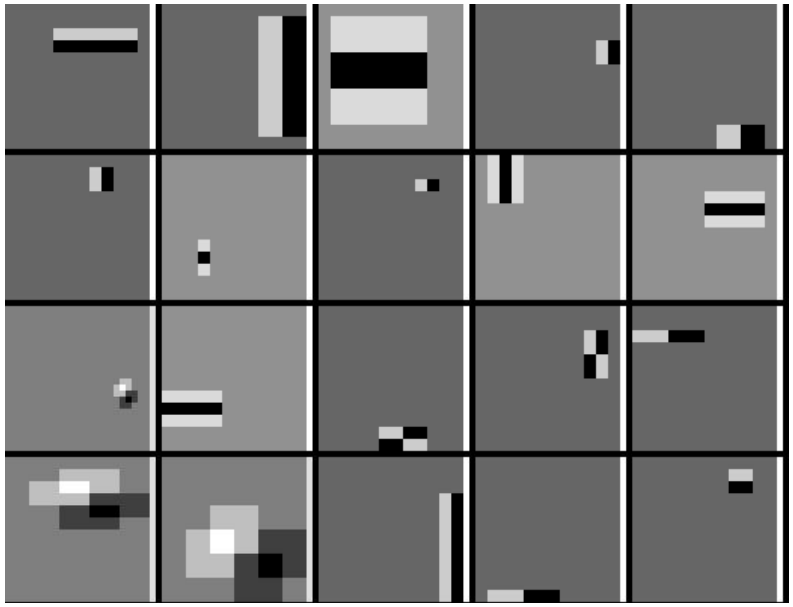




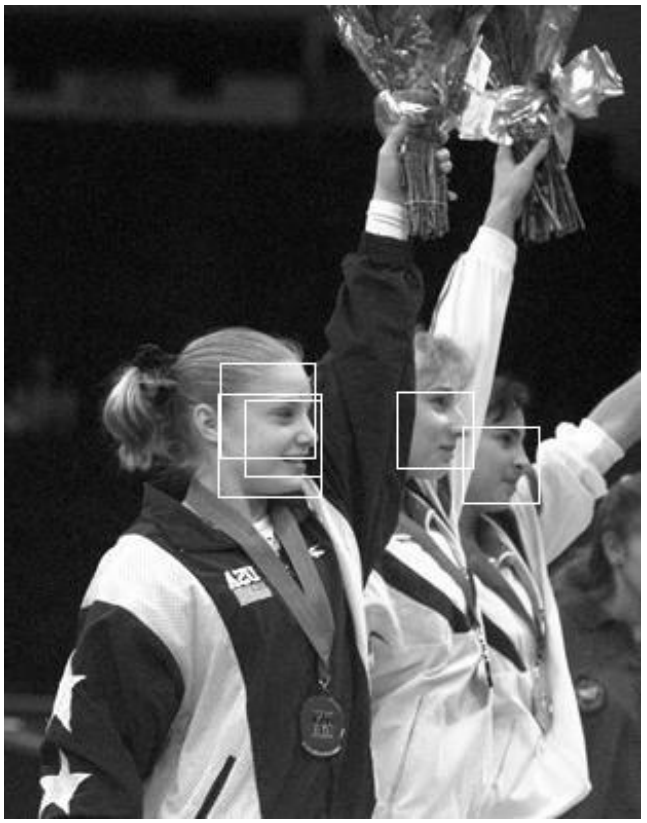
Viola-Jones Face Detector: Results

Detecting profile faces?

Can we use the same detector?

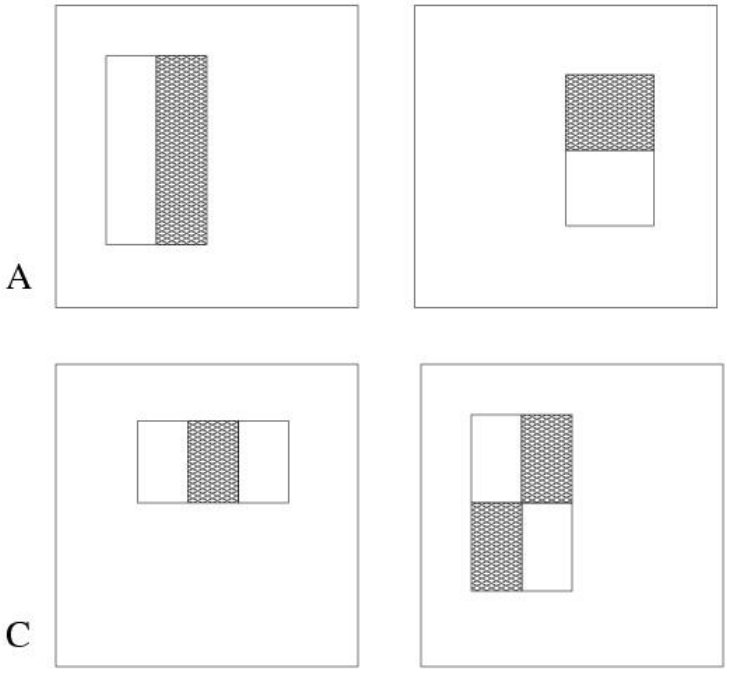


Viola-Jones Face Detector: Results



Idea: Using Many Simple Features

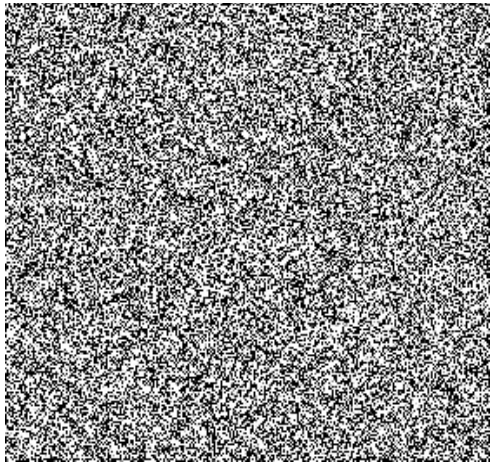
- Viola Jones / Haar Features



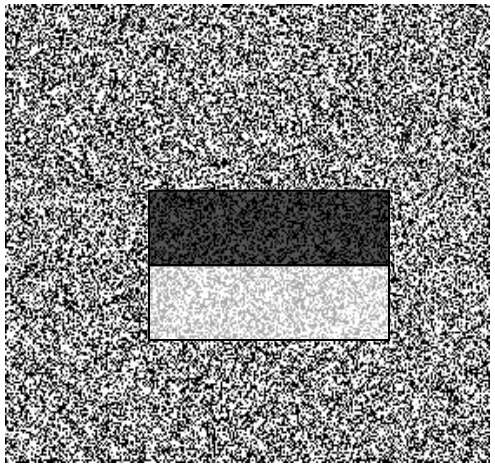
(Generalized) Haar Features:

- rectangular blocks, white or black
- 3 types of features:
  - two rectangles: horizontal/vertical
  - three rectangles
  - four rectangles
- in 24x24 window: 180,000 possible features

Example

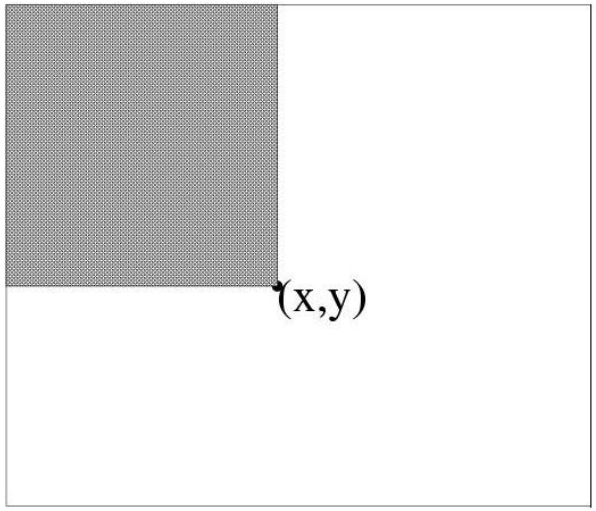


Source



Result

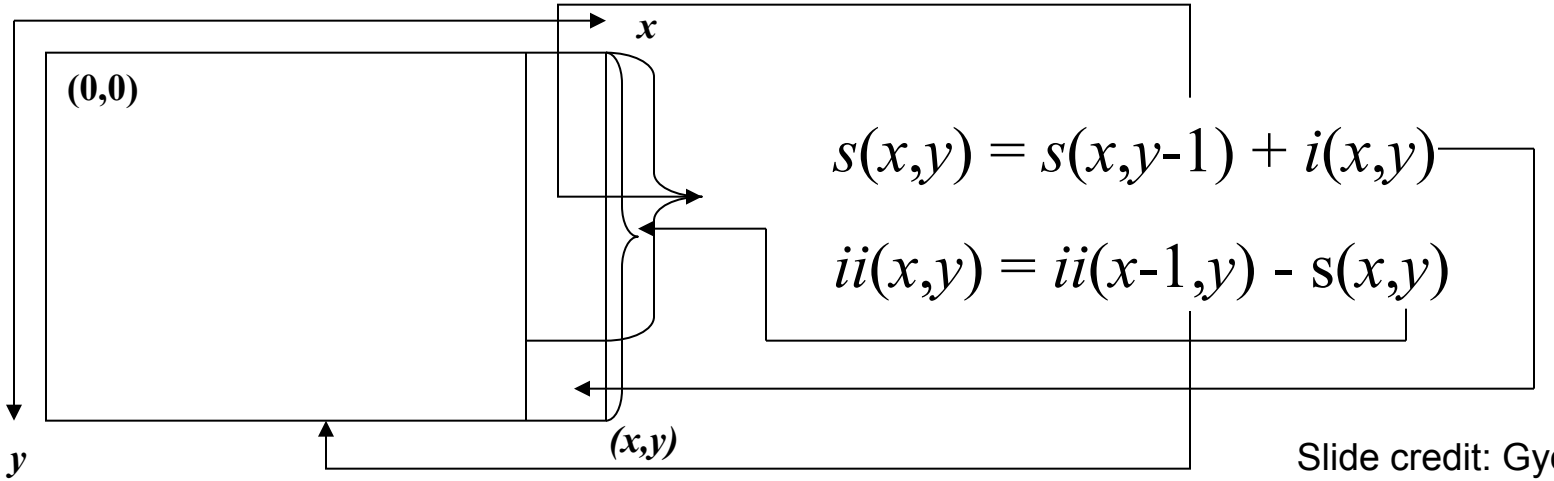
Integral Image



Def: The *integral image* at location (*x*,*y*), is the sum of the pixel values above and to the left of (*x*,*y*), inclusive. We can calculate the integral image representation of the image in a single pass.

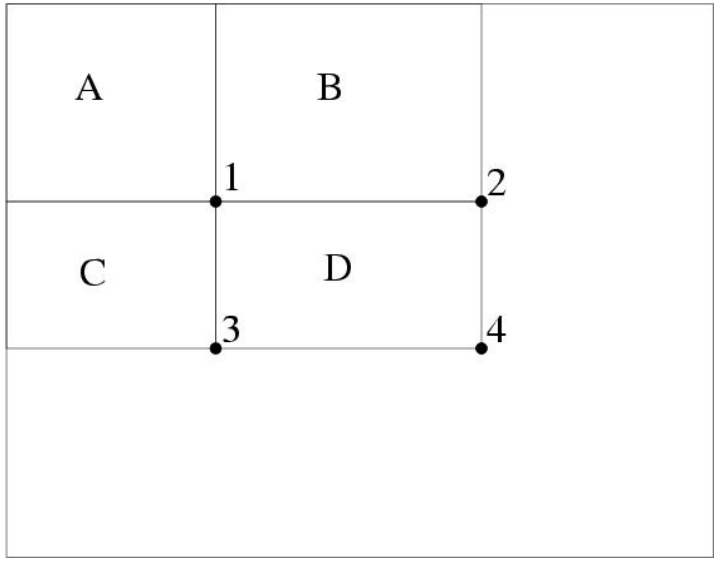
ii(x,y) – value of the integral image – sum of all pixels above and left of (x,y)

s(x,y) – cummulative row sum



Slide credit: Gyozo Gidofalvi

Efficient Computation of Rectangle Value



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

Example: Rectangle D

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

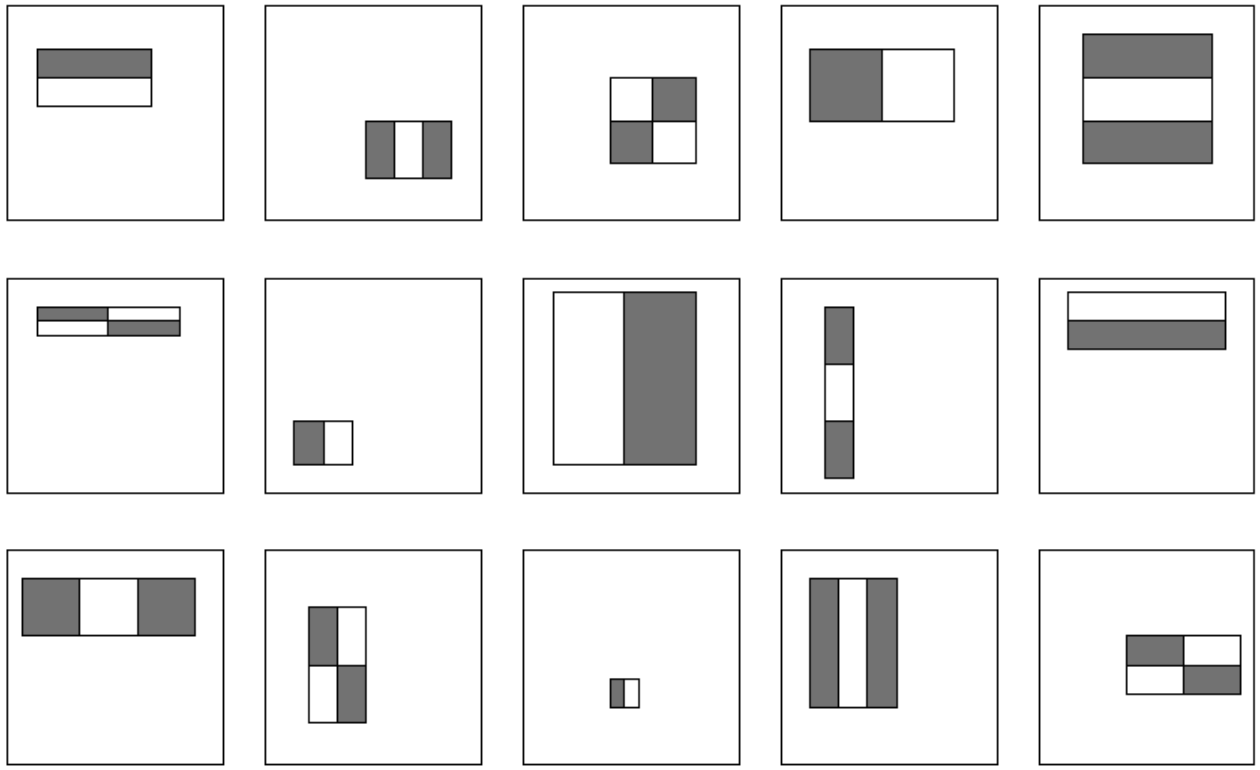
As a result two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively.

Idea: Compute lot of simple features – outputs of convolution with the box like filters

Object detection: classification problem

Feature selection

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



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

Boosting for face detection

- Define weak learners based on rectangle features

value of rectangle feature

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) > p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

window

parity

threshold

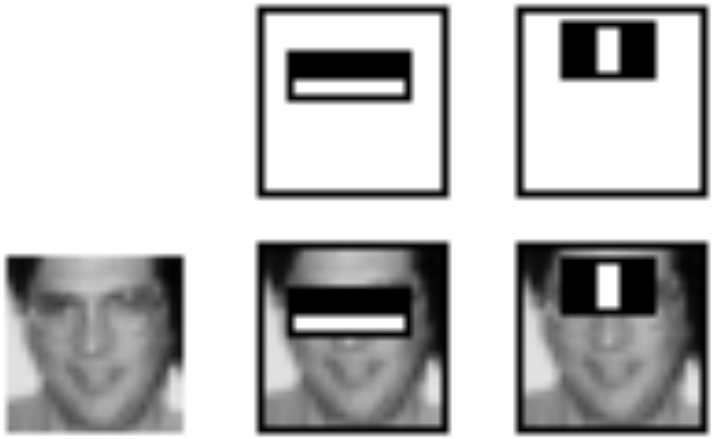
Boosting

- 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

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

AdaBoost Idea (in Viola/Jones):

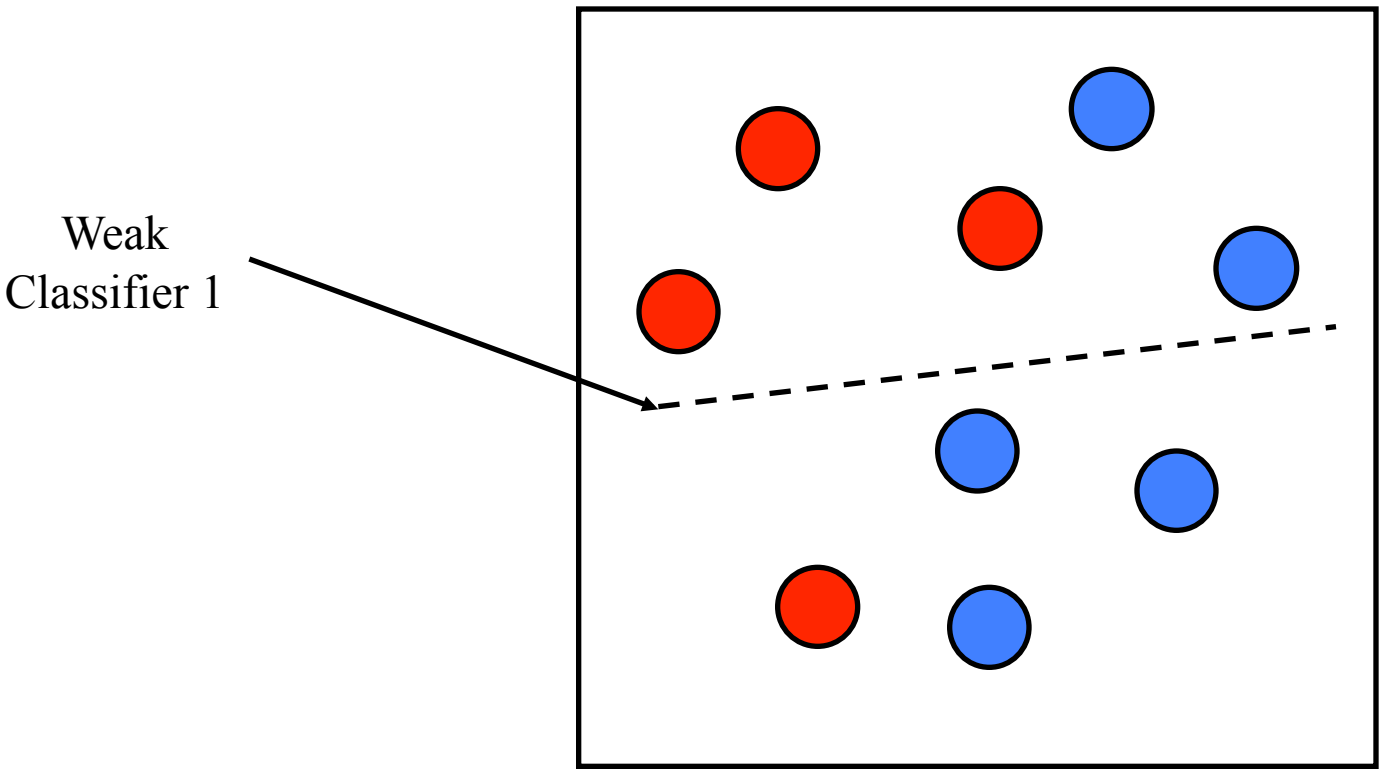
- Given set of "weak" classifiers:
  - Pick best one
  - Reweight training examples, so that misclassified images have larger weight
  - Reiterate; then linearly combine resulting classifiers



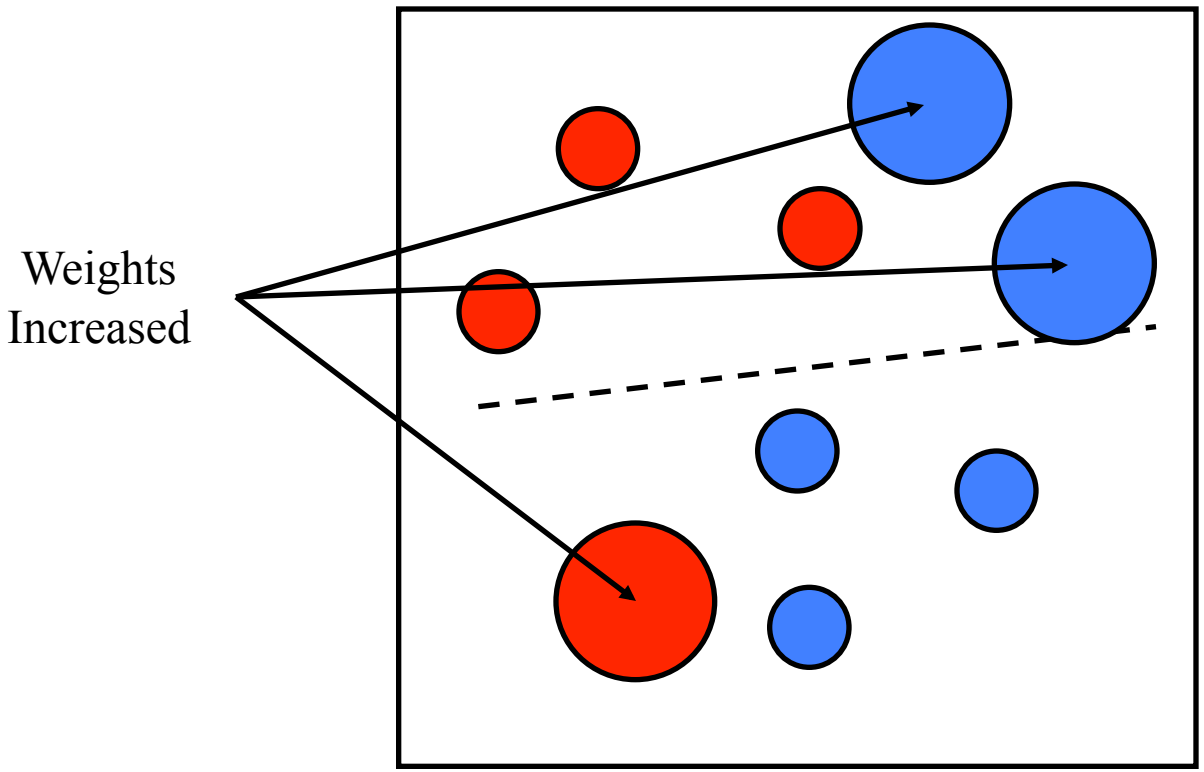
Weak classifiers: Haar features

Boosting illustration

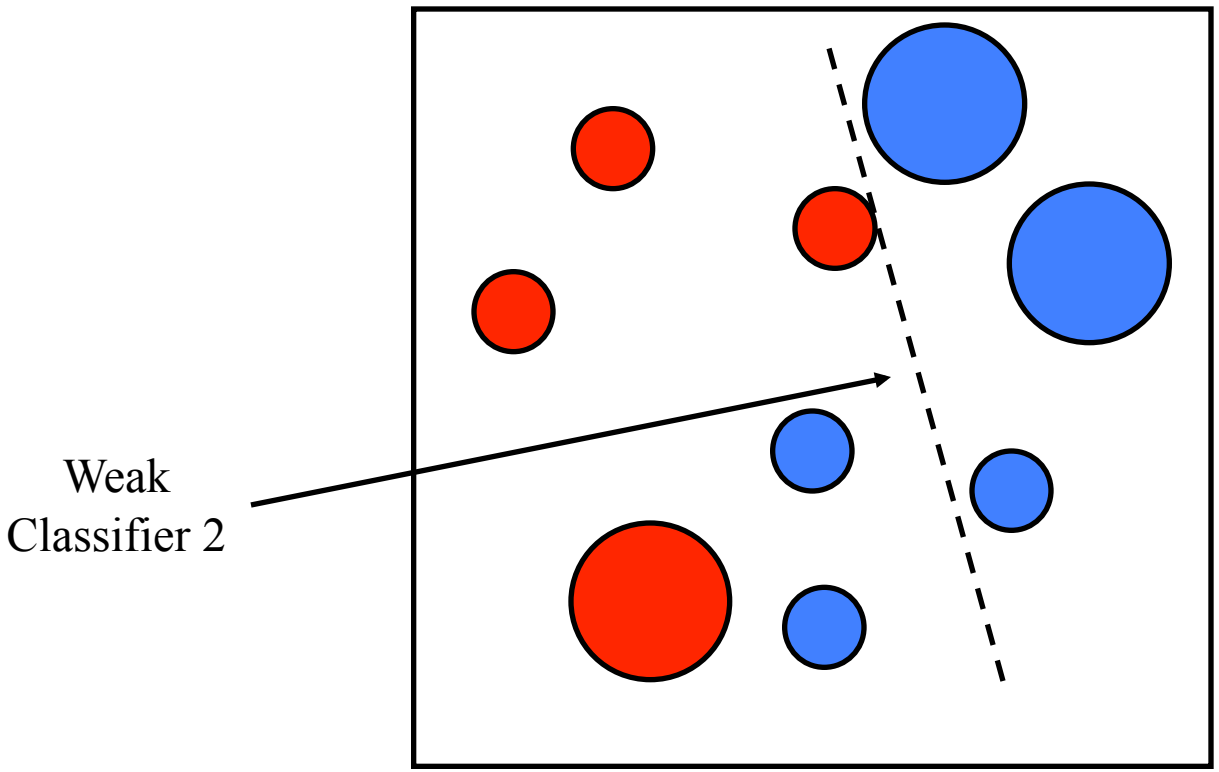
- Weak classifier is a hyperplane



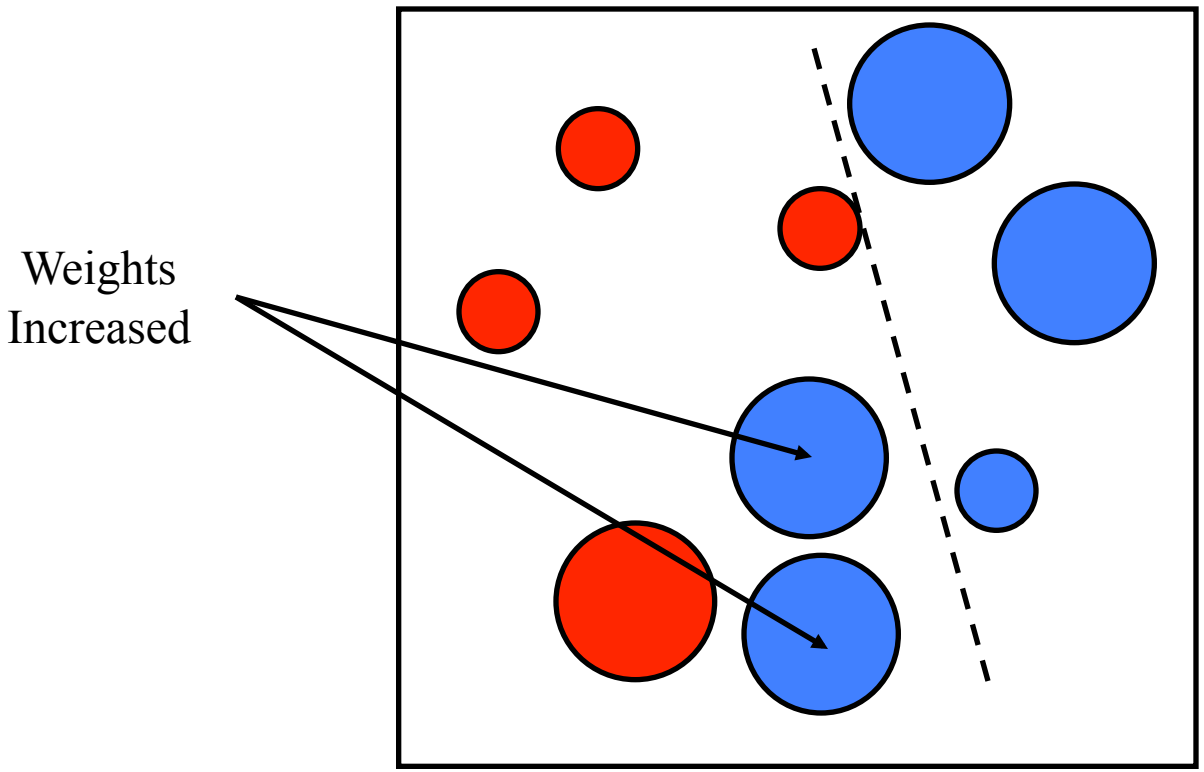
Boosting illustration



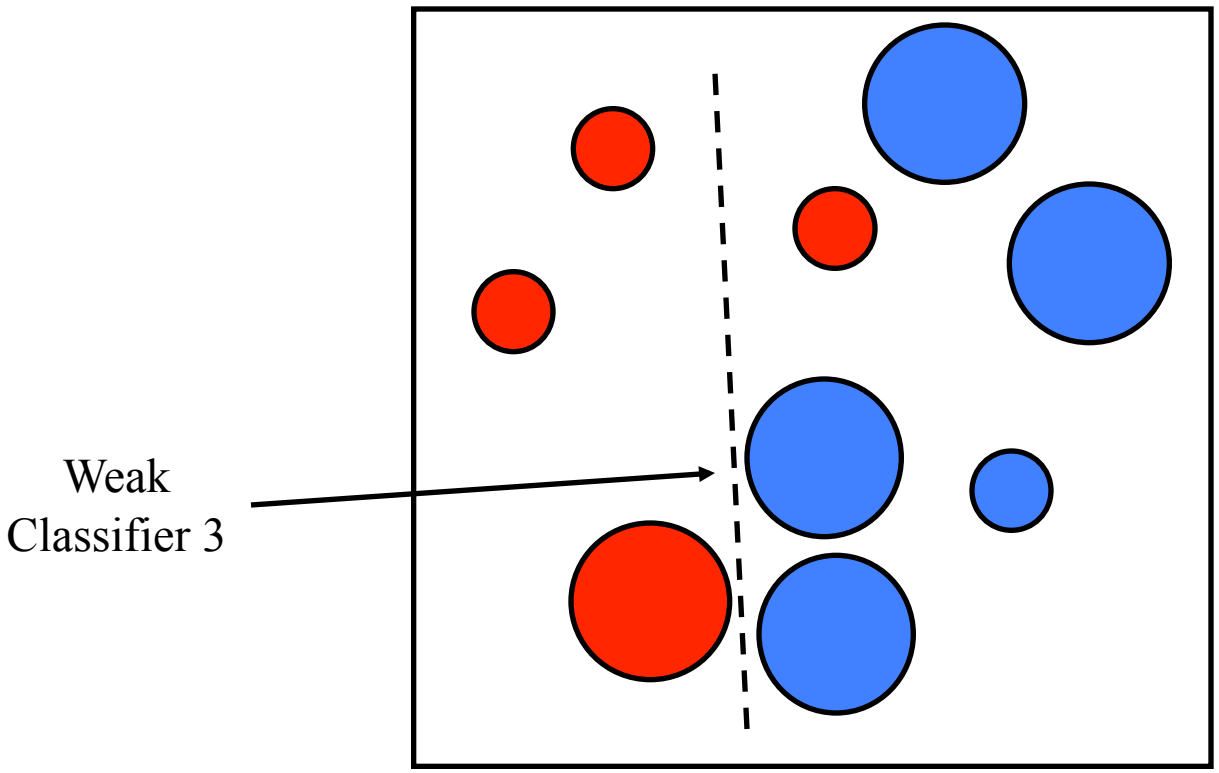
Boosting illustration



Boosting illustration

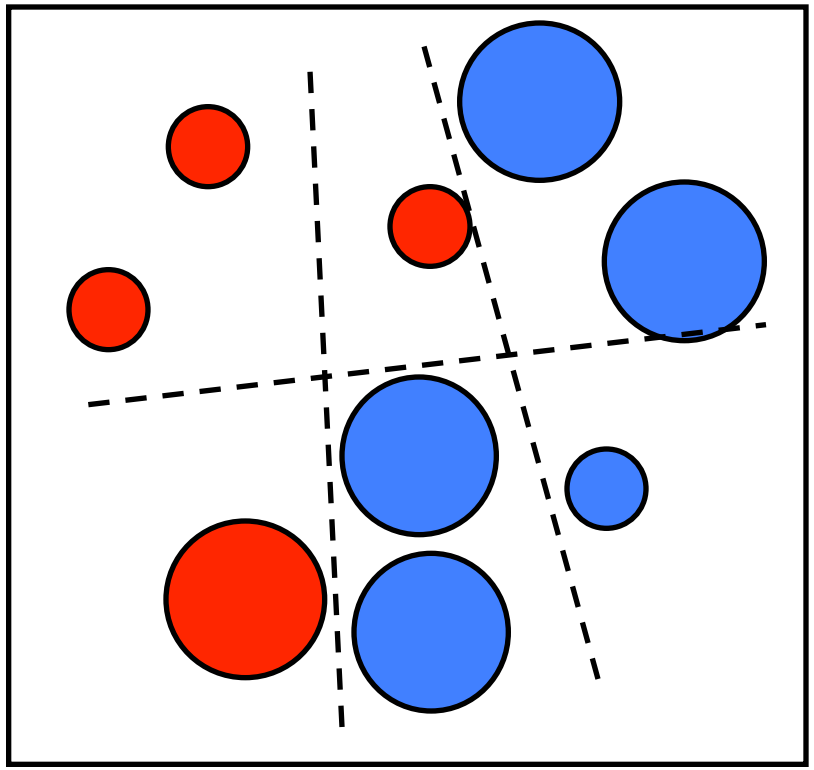


Boosting illustration



Boosting illustration

Final classifier is
a combination of weak
classifiers



Boosting vs. SVM

- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement
- Disadvantages
  - Needs many training examples
  - Often doesn't work as well as SVM (especially for many-class problems)

AdaBoost learning algorithm

Discrete AdaBoost(Freund & Schapire 1996*b*)

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

Boosting: training

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest *weighted* training error
  - Raise 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)

Slide credit: Lana Lazebnik

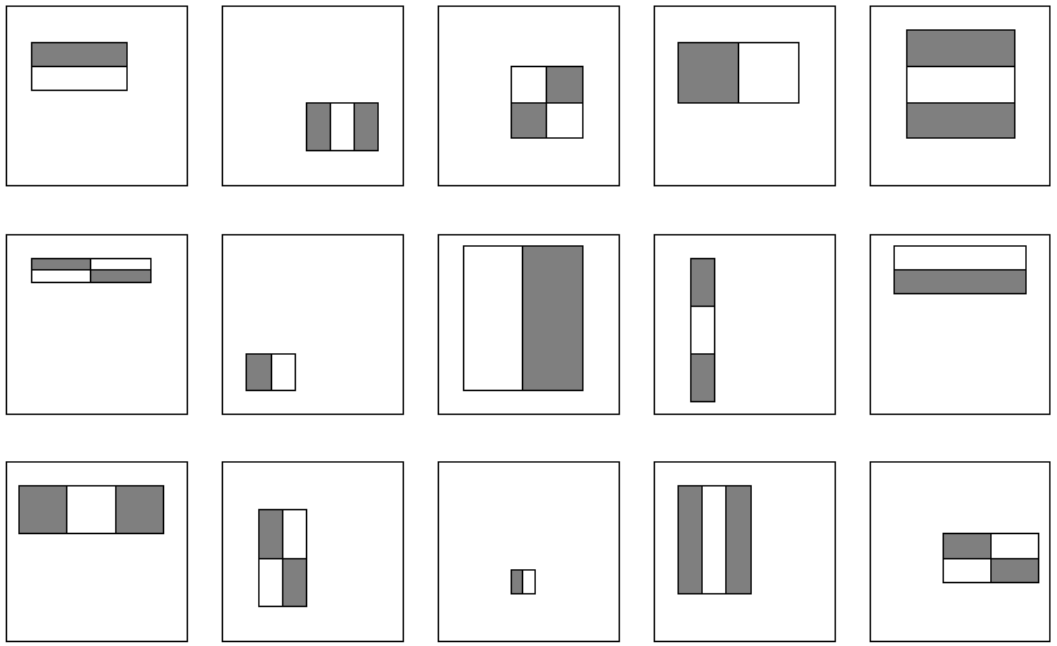
Boosting: pros and cons

- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement

- Disadvantages
  - Needs many training examples
  - Often found not to work as well as an alternative discriminative classifier, support vector machine (SVM)
    - especially for many-class problems

Slide credit: Lana Lazebnik

Viola-Jones detector: features



Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

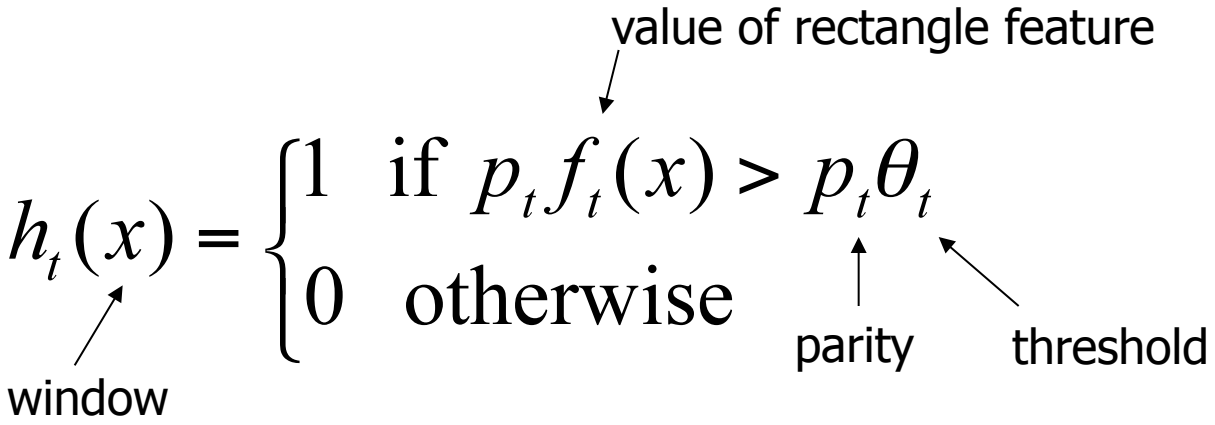
Which subset of these features should we use to determine if a window has a face?

Use AdaBoost both to select the informative features and to form the classifier

Kristen Grauman

Boosting for face detection

- Define weak learners based on rectangle features



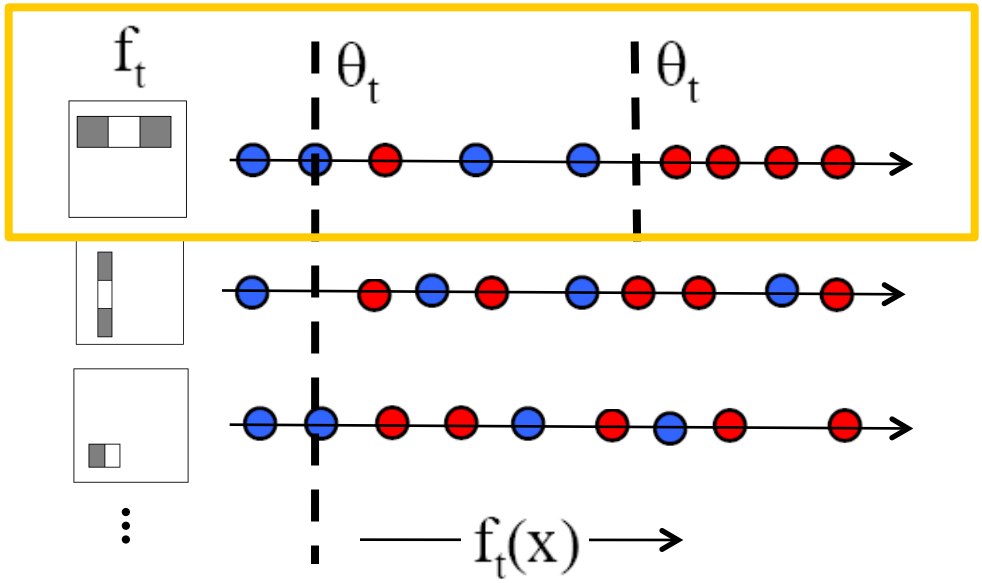
- For each round of boosting:
  Evaluate each rectangle filter on each example
  Select best filter/threshold combination based on weighted training error reweight examples

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

Viola-Jones detector: AdaBoost

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

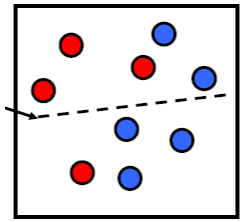
$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/threshold combo.

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

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

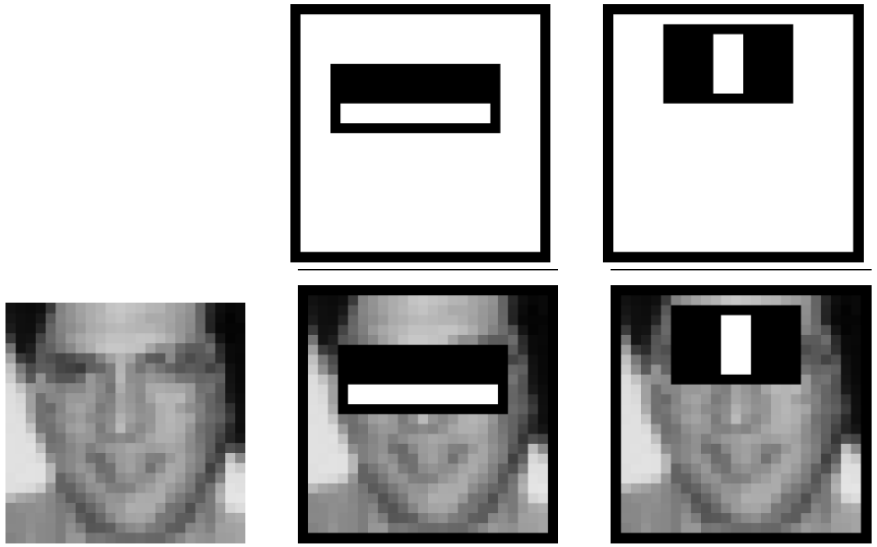
For T rounds

Evaluate weighted error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Viola-Jones Face Detector: Results



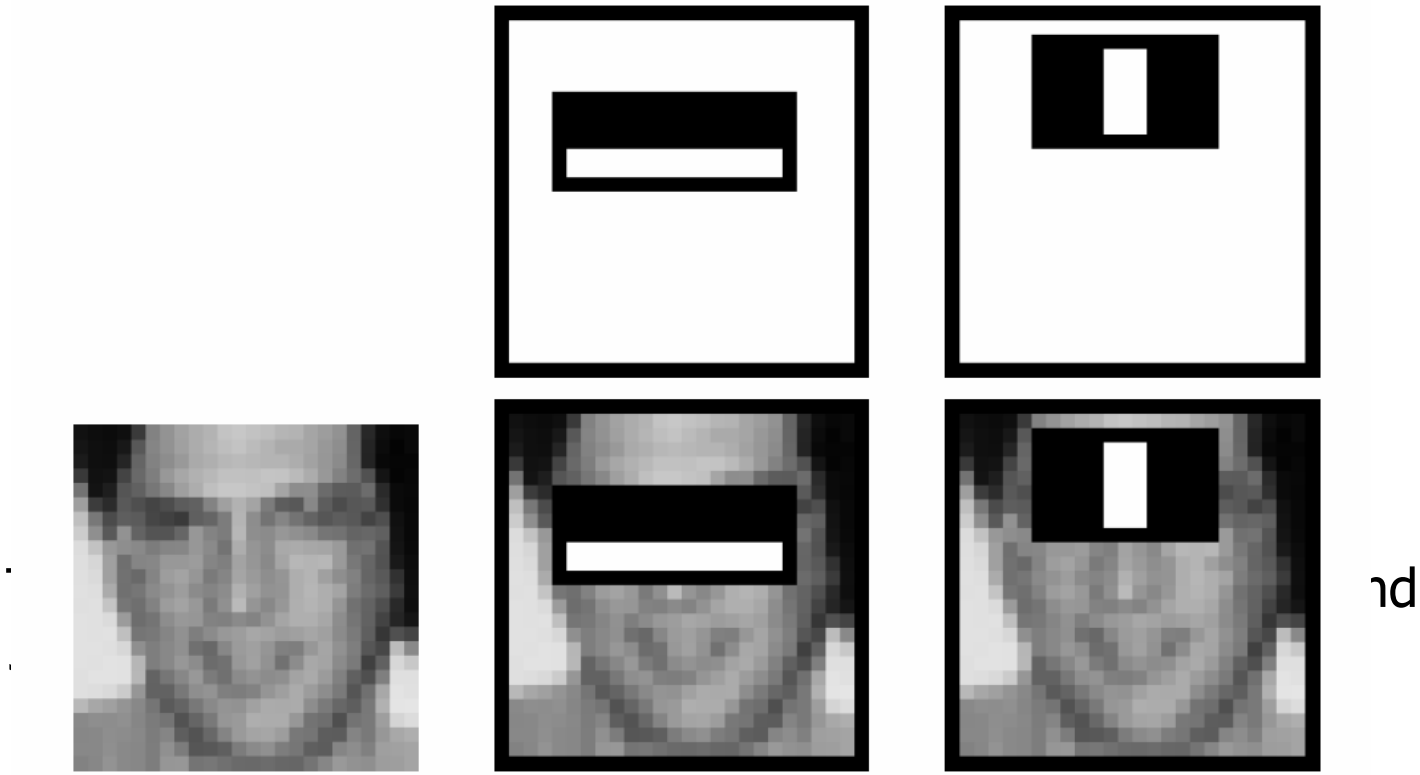
First two features selected

- Even if the filters are fast to compute, each new image has a lot of possible windows to search.

- How to make the detection more efficient?

Boosting for face detection

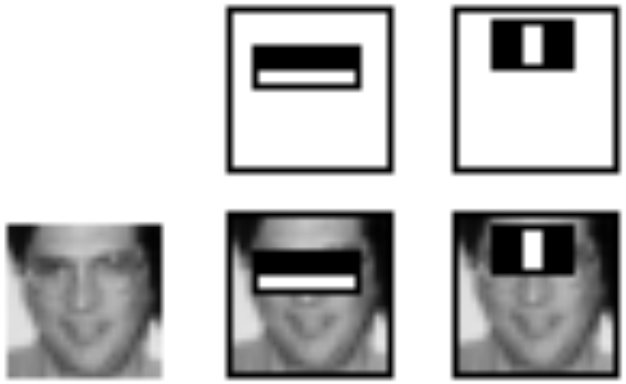
- First two features selected by boosting:

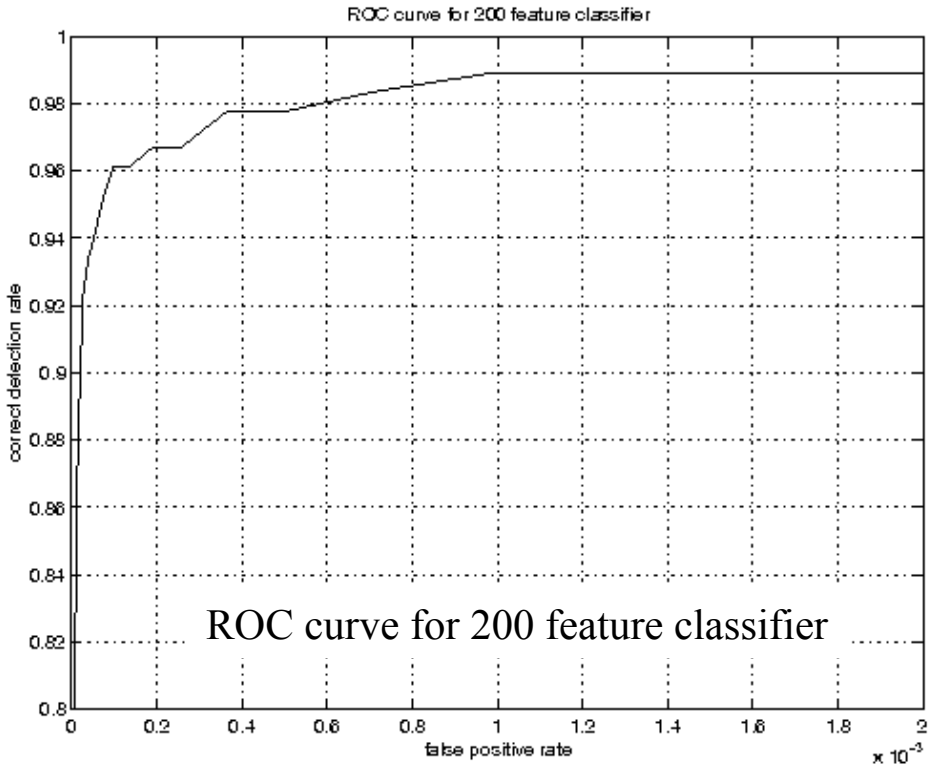


Example Classifier for Face Detection

A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084 false positives.

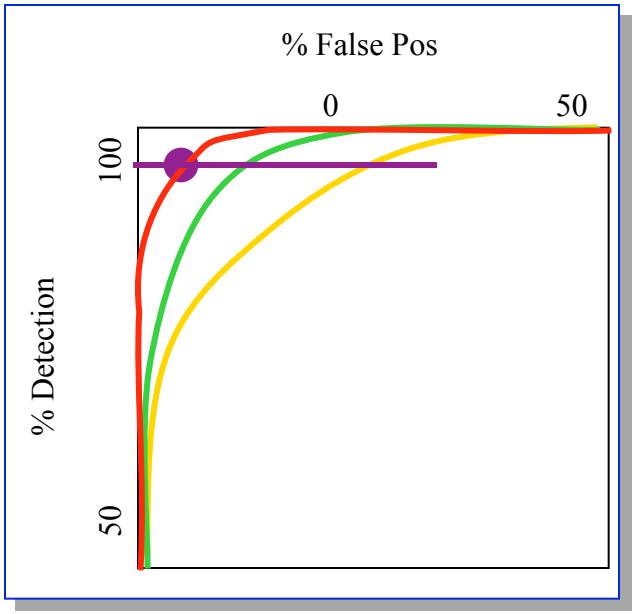


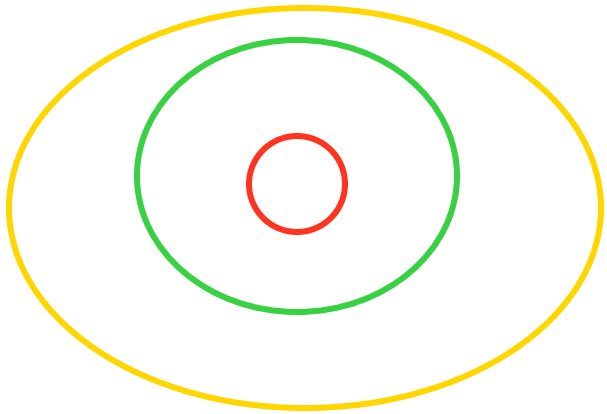


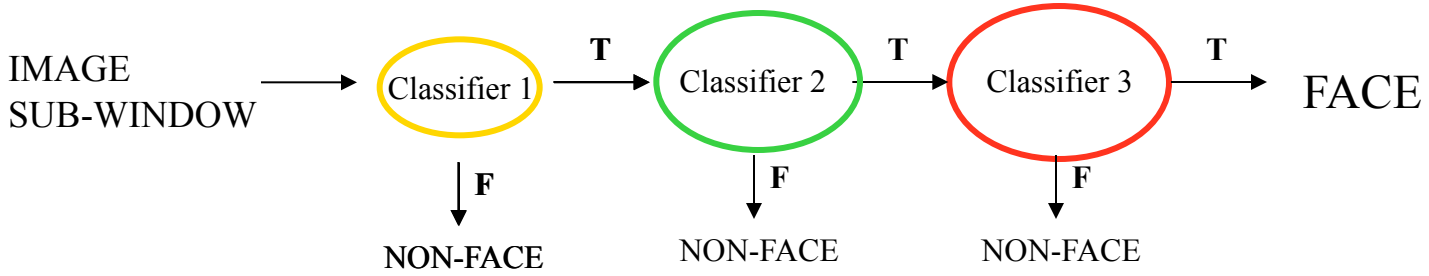
Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Classifier are Efficient

- Given a nested set of classifier hypothesis classes

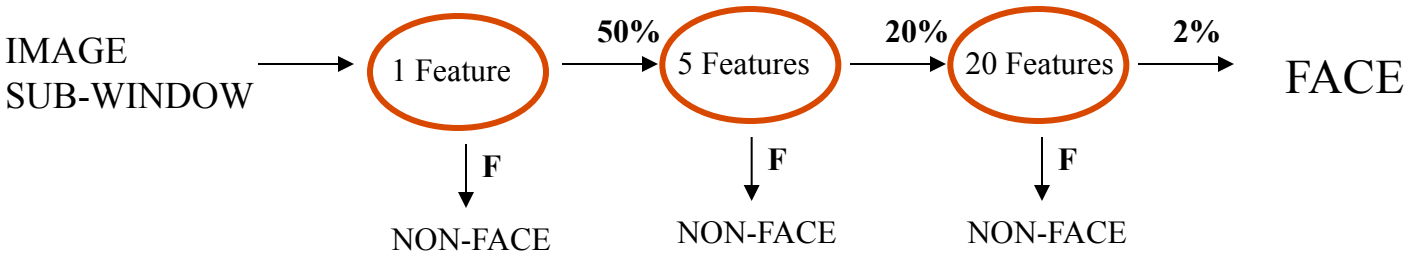






Slide credit: Frank Dellaert, Paul Viola, Forsyth&Ponce

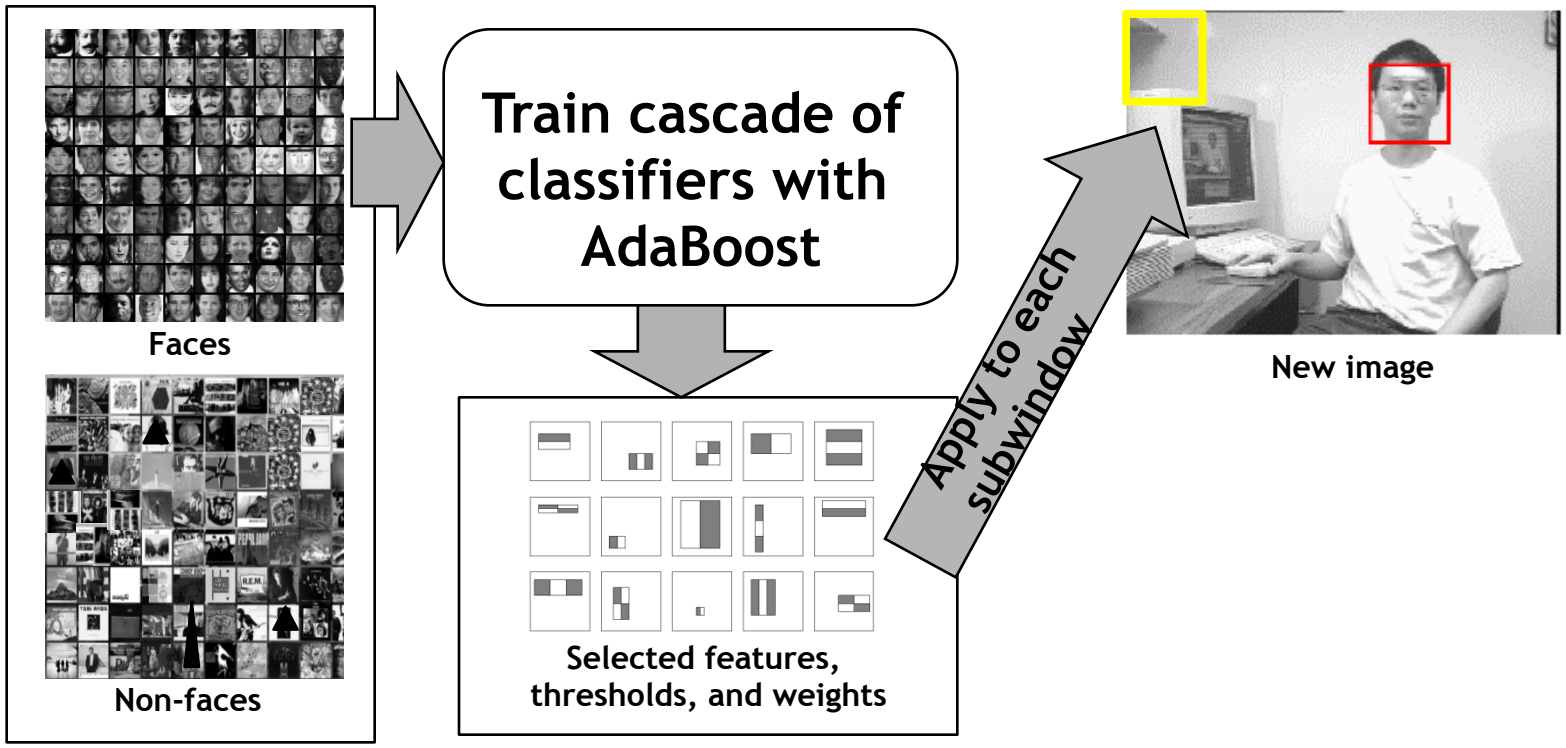
Cascaded Classifier



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

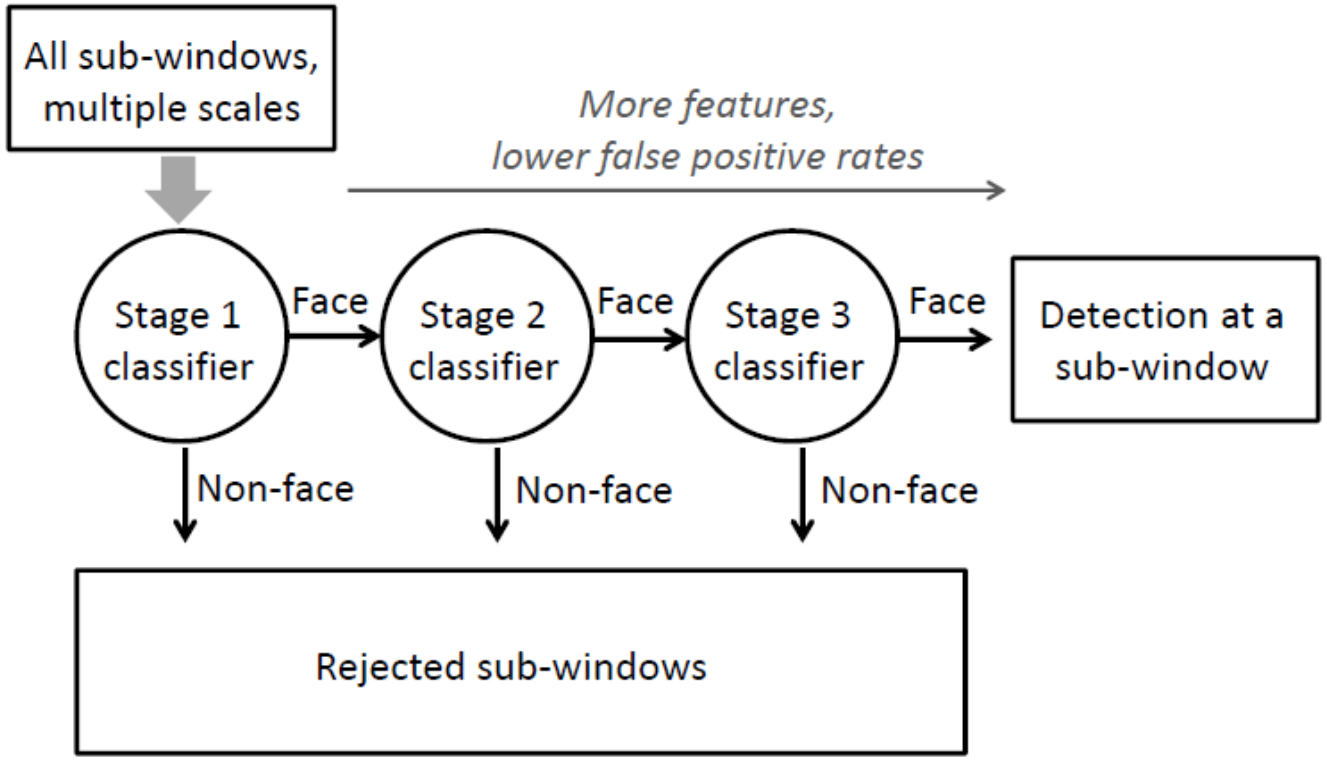
Viola-Jones detector: summary



Train with 5K positives, 350M negatives
Real-time detector using 38 layer cascade
6061 features in all layers

[Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

Cascading classifiers for detection

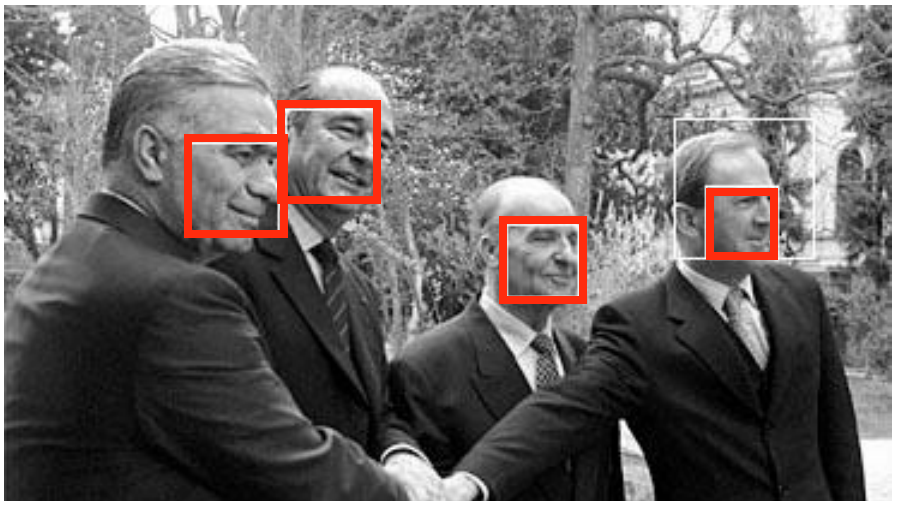


- Form a *cascade* with low false negative rates early on
- Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative

Kristen Grauman

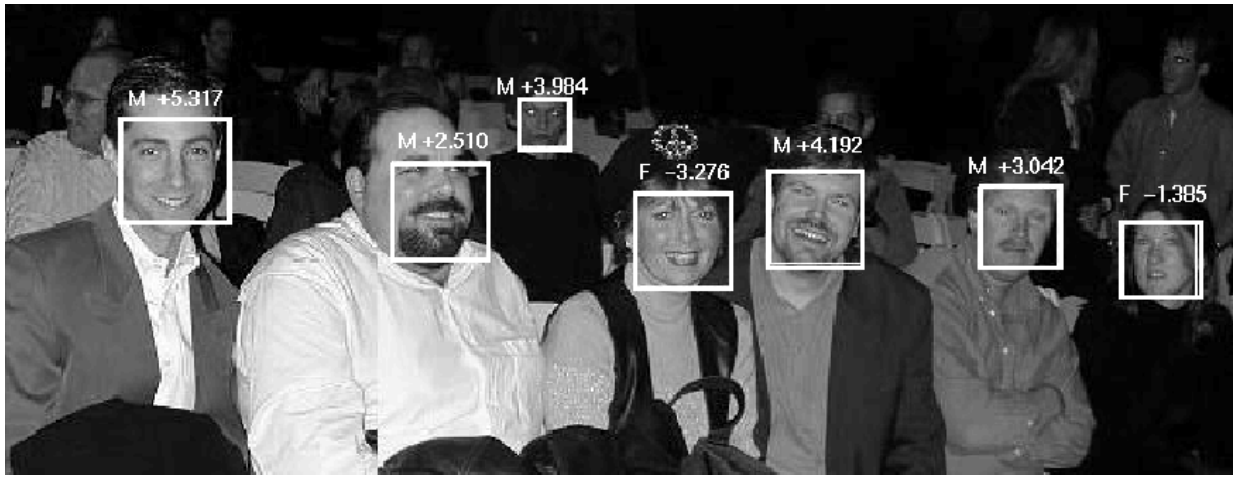
Solving other "Face" Tasks

Facial Feature Localization



Profile Detection

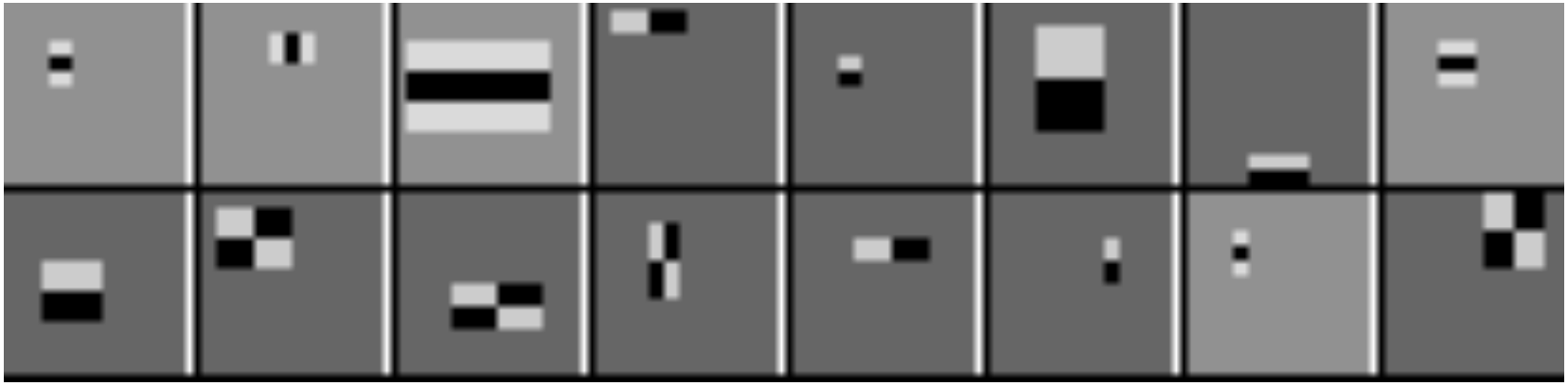
Demographic Analysis



Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Face Localization Features

- Learned features reflect the task

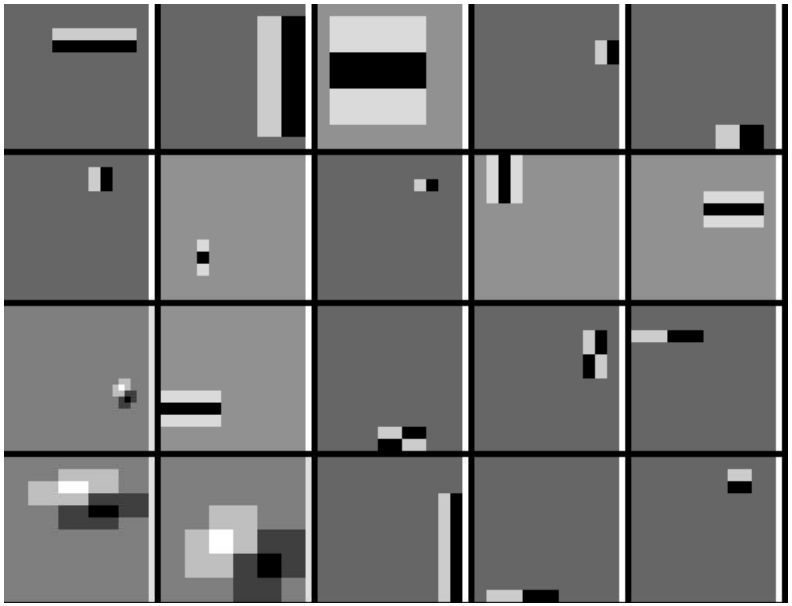


Slide credit: Frank Dellaert, Paul Viola, Forsyth&Ponce

Face Profile Detection

Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

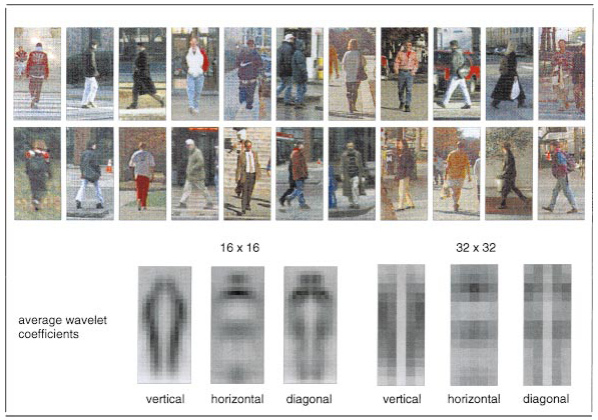
Face Profile Features



- What other categories are amenable to *window-based representation*?

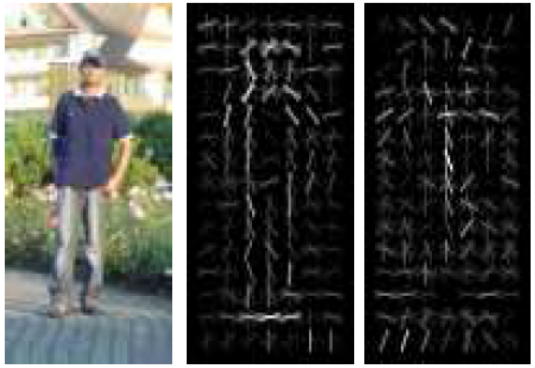
Pedestrian detection

- Detecting upright, walking humans also possible using sliding window's appearance/texture; e.g.,



SVM with Haar wavelets [Papageorgiou & Poggio, IJCV 2000]

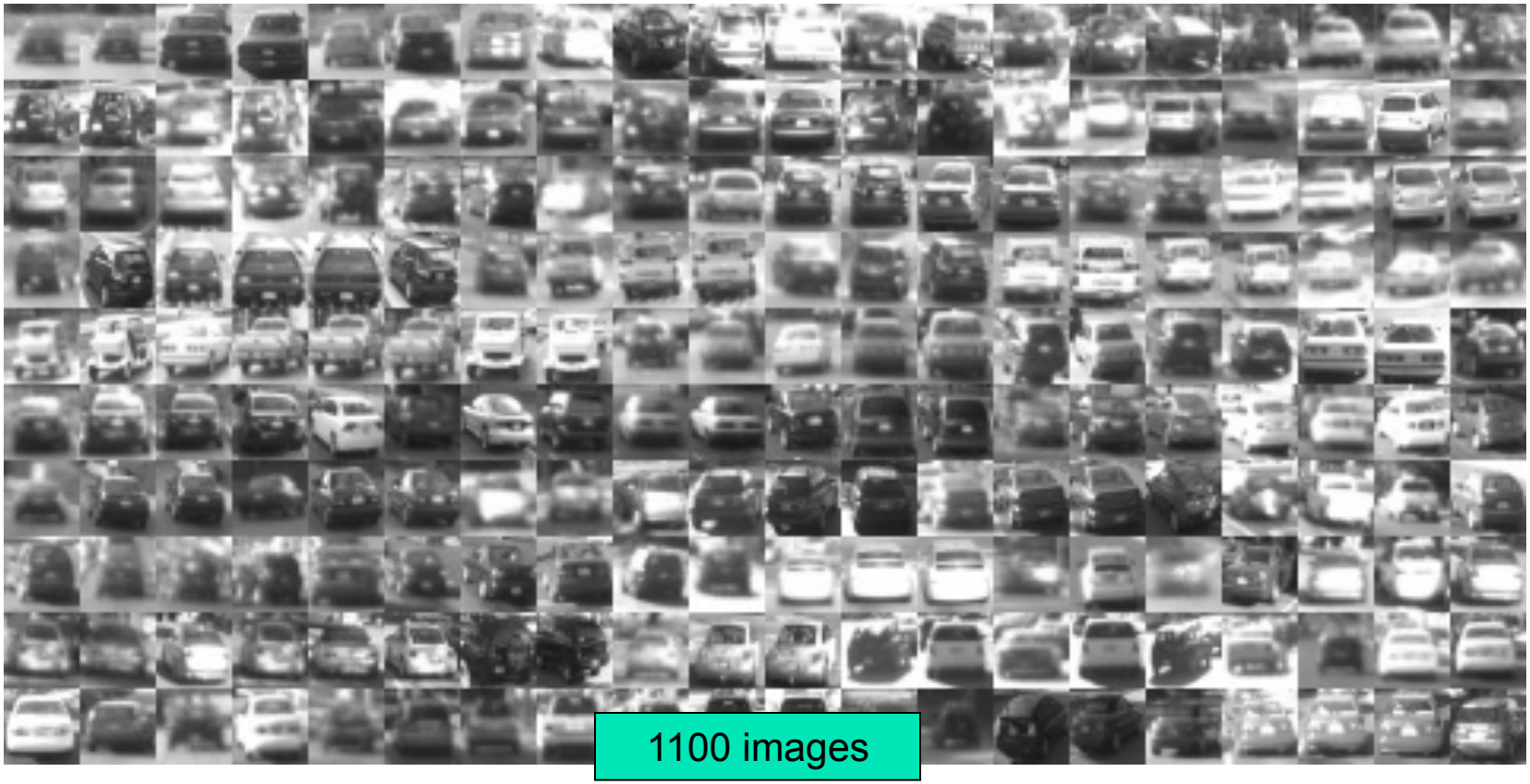
Space-time rectangle features [Viola, Jones & Snow, ICCV 2003]



SVM with HoGs [Dalal & Triggs, CVPR 2005]

Finding Cars (DARPA Urban Challenge)

- Hand-labeled images of generic car rear-ends
- Training time: ~5 hours, offline



Credit: Hendrik Dahlkamp

Generating even more examples

- Generic classifier finds all cars in recorded video.
- Compute offline and store in database

Credit: Hendrik Dahlkamp

Window-based detection: strengths

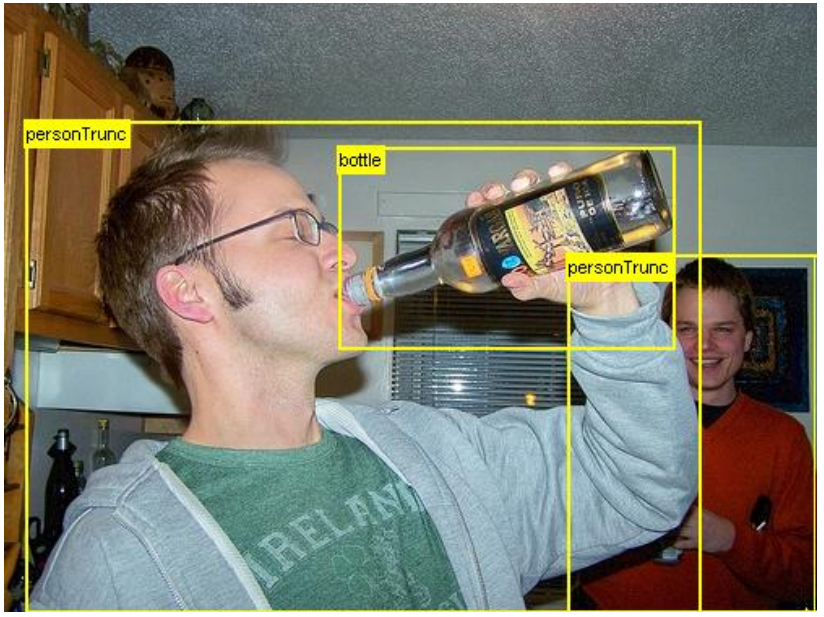
- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

Window-based detection: Limitations

- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
- With so many windows, false positive rate better be low

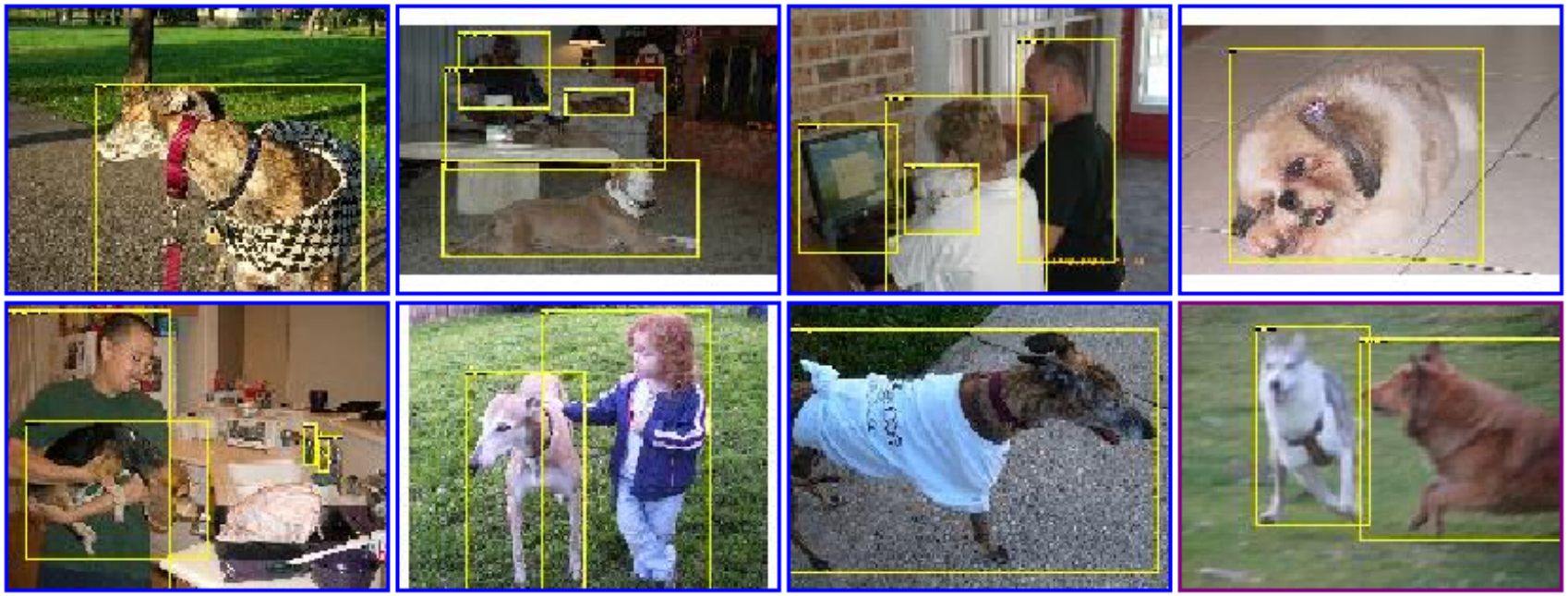
Limitations (continued)

- Not all objects are “box” shaped



Limitations (continued)

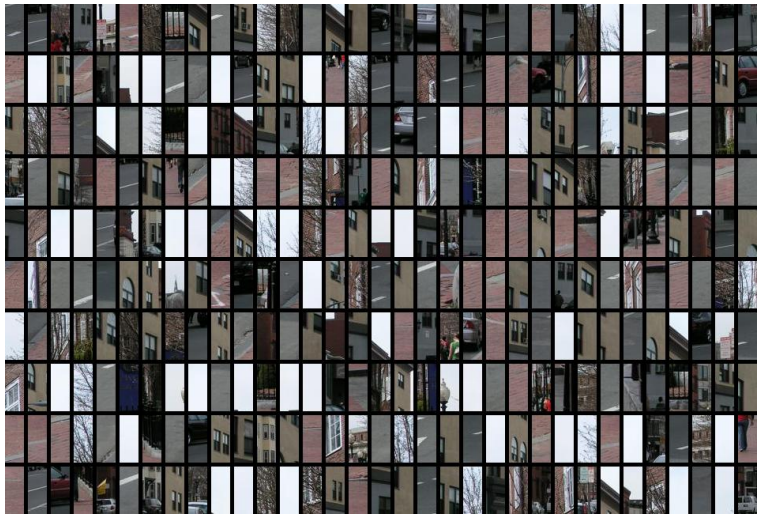
- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions



Limitations (continued)

- If considering windows in isolation, context is lost

Sliding window



Detector's view

Figure credit: Derek Hoiem

Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

Image credit: Adam, Rivlin, & Shimshoni

Summary

- Basic pipeline for window-based detection
  - Model/representation/classifier choice
  - Sliding window and classifier scoring
- Boosting classifiers: general idea
- Viola-Jones face detector
  - Exemplar of basic paradigm
  - Plus key ideas: rectangular features, Adaboost for feature selection, cascade
- Pros and cons of window-based detection

Summary Viola-Jones

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows

- Many simple features
  - Generalized Haar features (multi-rectangles)
  - Easy and efficient to compute
- Discriminative Learning:
  - finds a small subset for object recognition
  - Uses AdaBoost
- Result: Feature Cascade
  - 15fps on 700Mhz Laptop (=fast!)

- Applications, Face detection, Car detection, Many others