

Computer Vision

CS308

Feng Zheng

SUSTech CS Vision Intelligence and Perception

Week 10



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY



Content

- Brief Review
- Classical Methods of Object Detection
 - The Viola-Jones Face Detector
 - ✓ Harr-like features - Integral images
 - ✓ Feature selection - Adaboosting
 - ✓ Cascading
 - Deformable Part Models (DPM)
 - ✓ Histogram of gradients
 - ✓ Pictorial Structure- a star-structured part-based model
 - ✓ Latent (Structured) SVM

Brief Review



Logistic Regression

- **Data:** Inputs are continuous vectors of length K . Outputs are discrete $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ where $\mathbf{x} \in \mathbb{R}^K$ and $y \in \{0, 1\}$

- **Model:** Logistic function applied to dot product of parameters with input vector

$$p_{\theta}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

➤ **Prediction:** Output is the **most probable** class.

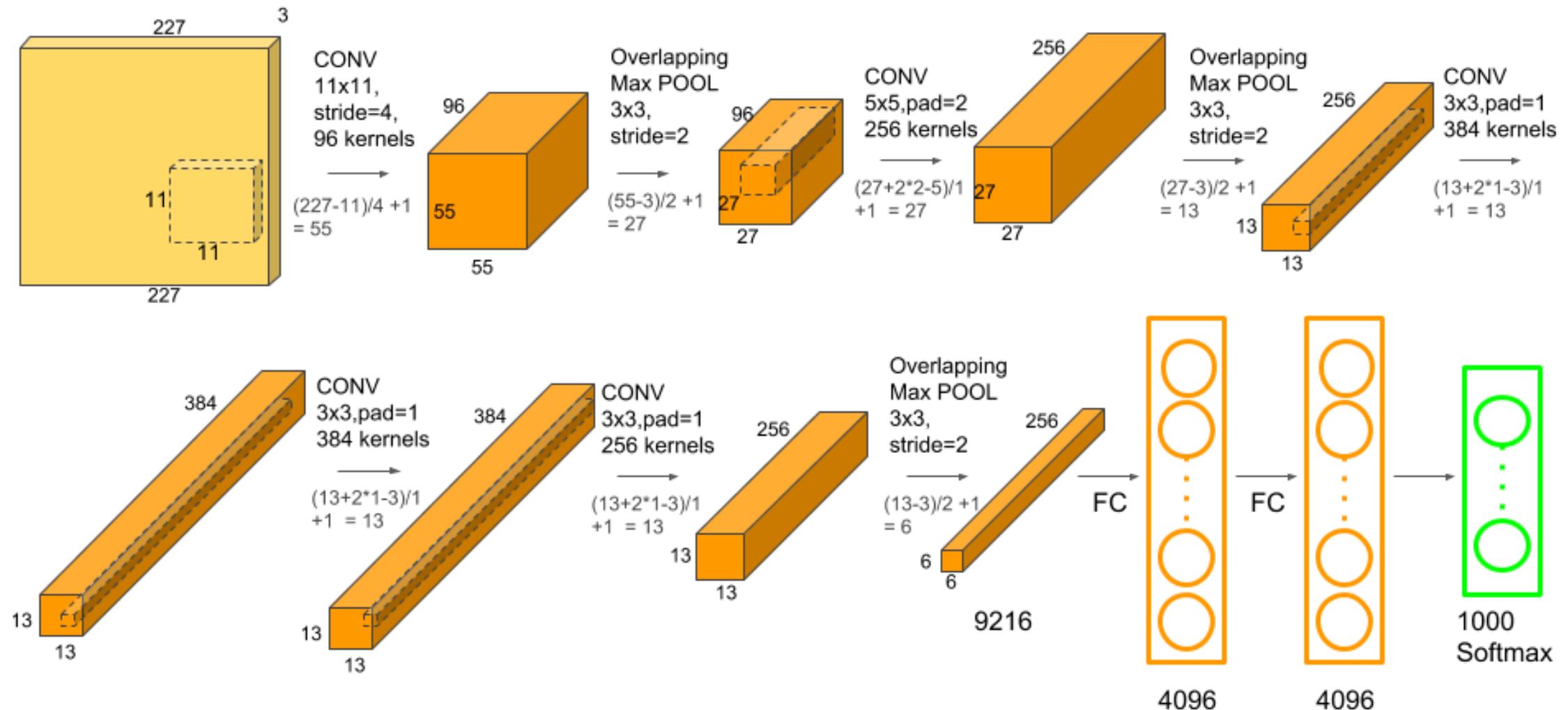
$$\hat{y} = \operatorname{argmax}_{y \in \{0,1\}} p_{\theta}(y|\mathbf{x})$$

- **Learning:** Finds the parameters that **minimize some objective function**.

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



Deep Convolutional Neural Networks





Training: Backpropagation

Simple Example: The goal is to compute $J = \cos(\sin(x^2) + 3x^2)$ on the forward pass and the derivative $\frac{dJ}{dx}$ on the backward pass.

Forward

$$J = \cos(u)$$

$$u = u_1 + u_2$$

$$u_1 = \sin(t)$$

$$u_2 = 3t$$

$$t = x^2$$

Backward

$$\frac{dJ}{du} += -\sin(u)$$

$$\frac{dJ}{du_1} += \frac{dJ}{du} \frac{du}{du_1}, \quad \frac{du}{du_1} = 1$$

$$\frac{dJ}{dt} += \frac{dJ}{du_1} \frac{du_1}{dt}, \quad \frac{du_1}{dt} = \cos(t)$$

$$\frac{dJ}{dt} += \frac{dJ}{du_2} \frac{du_2}{dt}, \quad \frac{du_2}{dt} = 3$$

$$\frac{dJ}{dx} += \frac{dJ}{dt} \frac{dt}{dx}, \quad \frac{dt}{dx} = 2x$$

Object Detection



Image Classification versus Object Detection

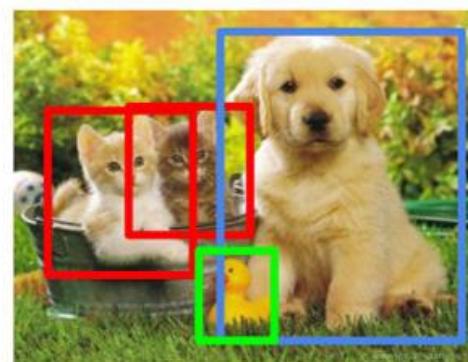
- If you want to classify an image into a certain **category**, you use image classification
- If you aim to identify the **location** of objects in an image, and, for example, count the number of instances of an object, you can use object detection

Classification



CAT

Object Detection



CAT, DOG, DUCK



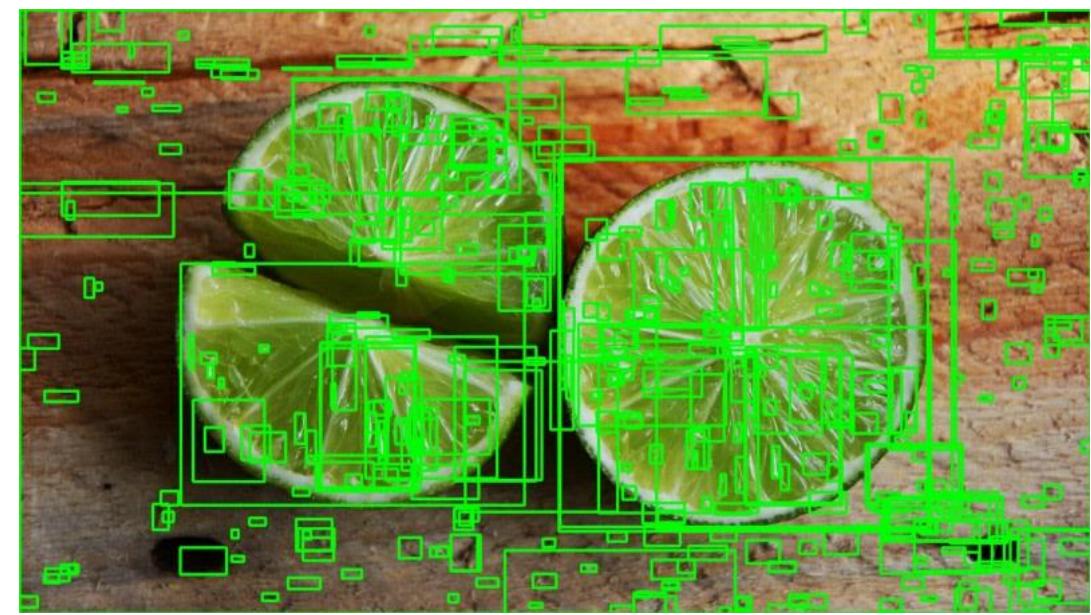
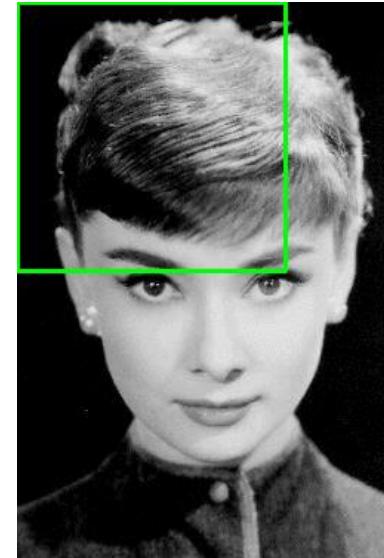
General Object Detection Framework-Sliding Window

- First: **region proposal**
 - A **model or algorithm** is used to generate regions of interest or region proposals
 - They **are a large set of bounding boxes** spanning the full image
- In the second step: **feature extraction** and recognition
 - Visual features are **extracted** for each of the bounding boxes
 - They are **evaluated** and it is determined whether and which objects are present in the proposals based on visual features
- In the final post-processing step: **box clustering**
 - **Overlapping** boxes are combined into a **single** bounding box



Region Proposals

- Several methods
 - Cascade + sliding window
 - Selective search
 - Use more complex visual features extracted from the image to generate regions
- An important trade-off that is made with region proposal generation is the **number of regions** vs. the **computational complexity**





Evaluation Metric

- Determining whether an object **exists** in the image (classification)
 - There is the need to associate a "**confidence score**" or **model score** with each bounding box detected and to assess the model at various level of confidence

$$Precision = \frac{TP}{TP + FP}$$

TP = True positive

$$Recall = \frac{TP}{TP + FN}$$

TN = True negative

FP = False positive

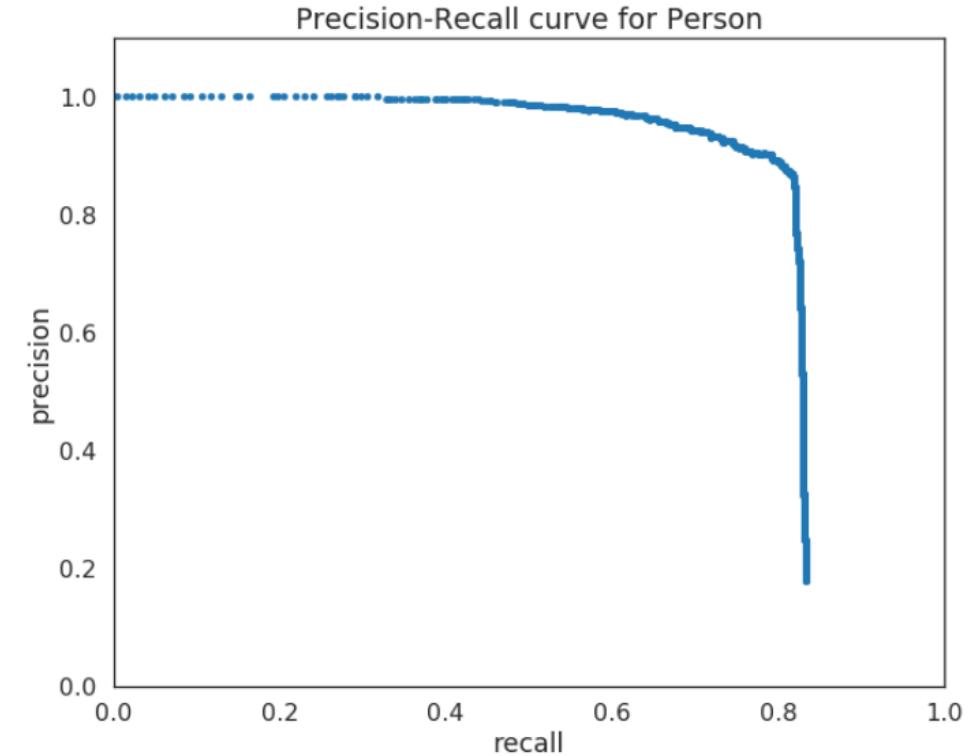
FN = False negative

- Determining the **location** of the object (localization, a regression task)



mean Average Precision (mAP)

- The precision-recall curve is computed from the model's detection output, by **varying the model score threshold** that determines what is counted as a model-predicted positive detection of the class
- The final step to calculating the **AP score** is to take the average value of the precision across all recall values

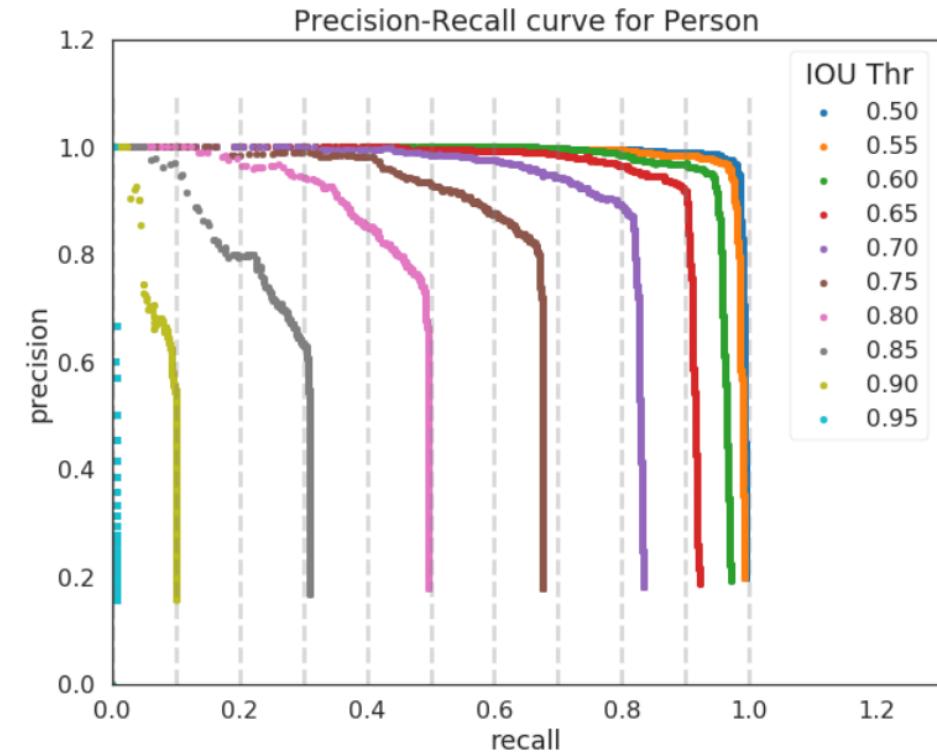
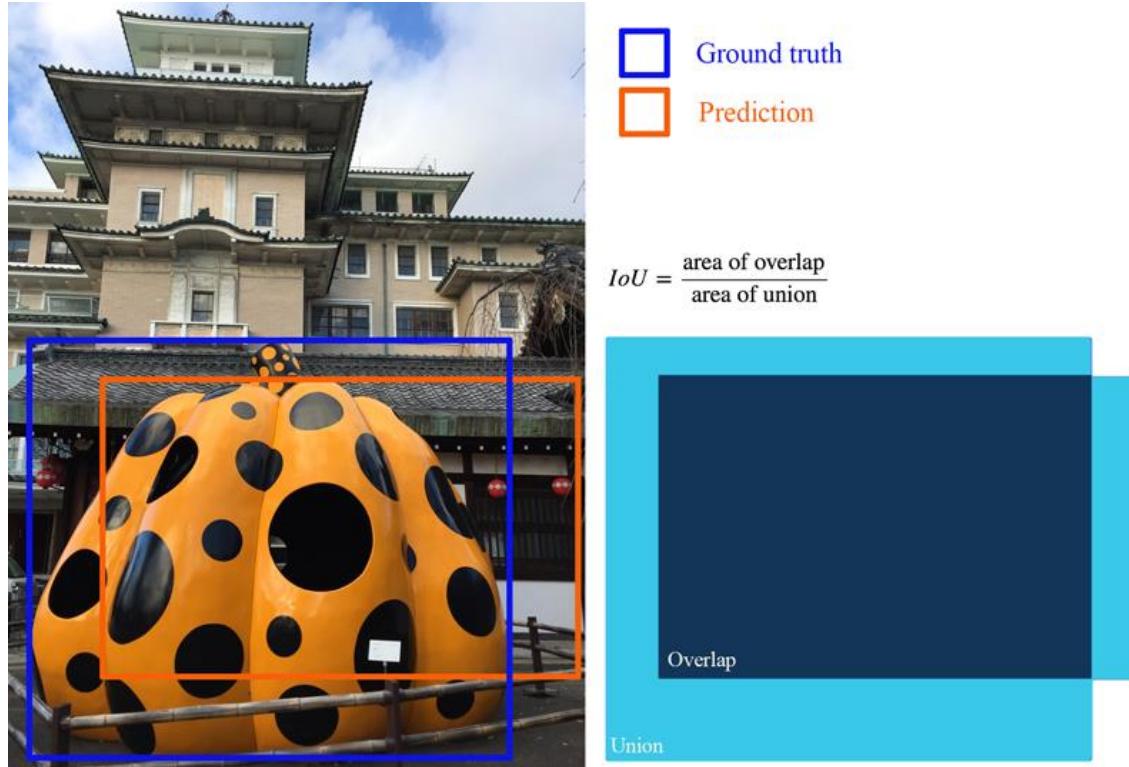


$$AP = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}_i)$$



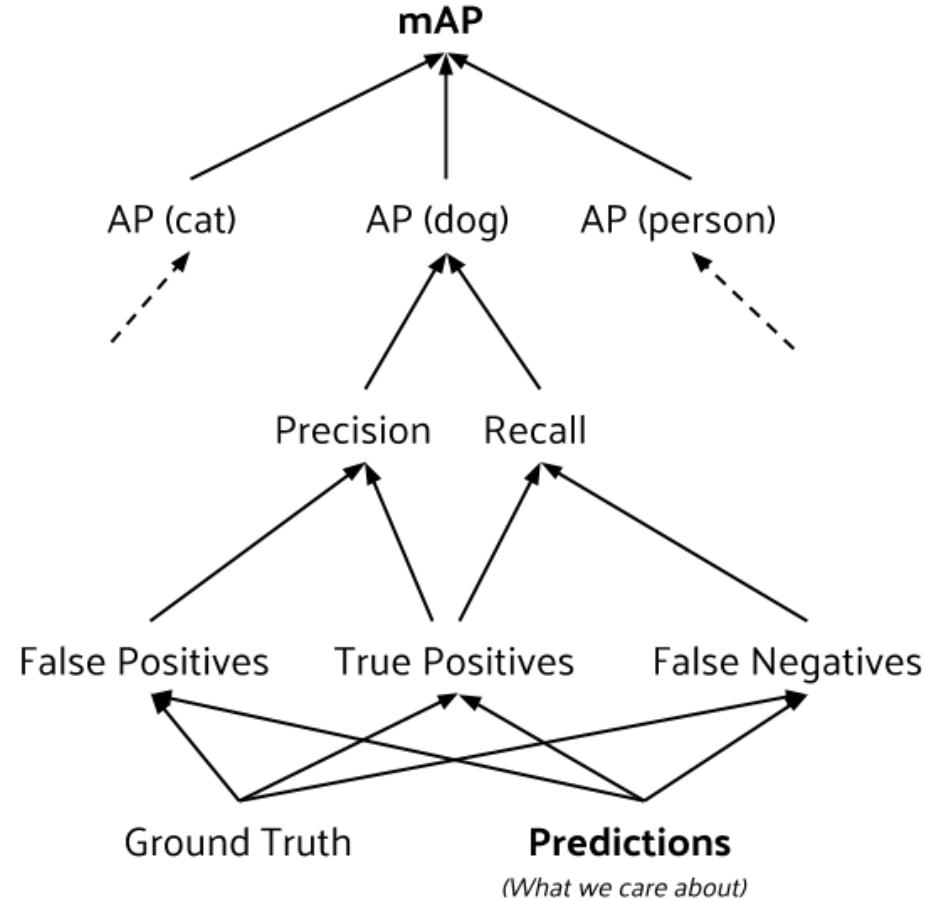
Localization and Intersection over Union

- IoU measures the **overlap** between 2 boundaries





mean Average Precision (mAP)

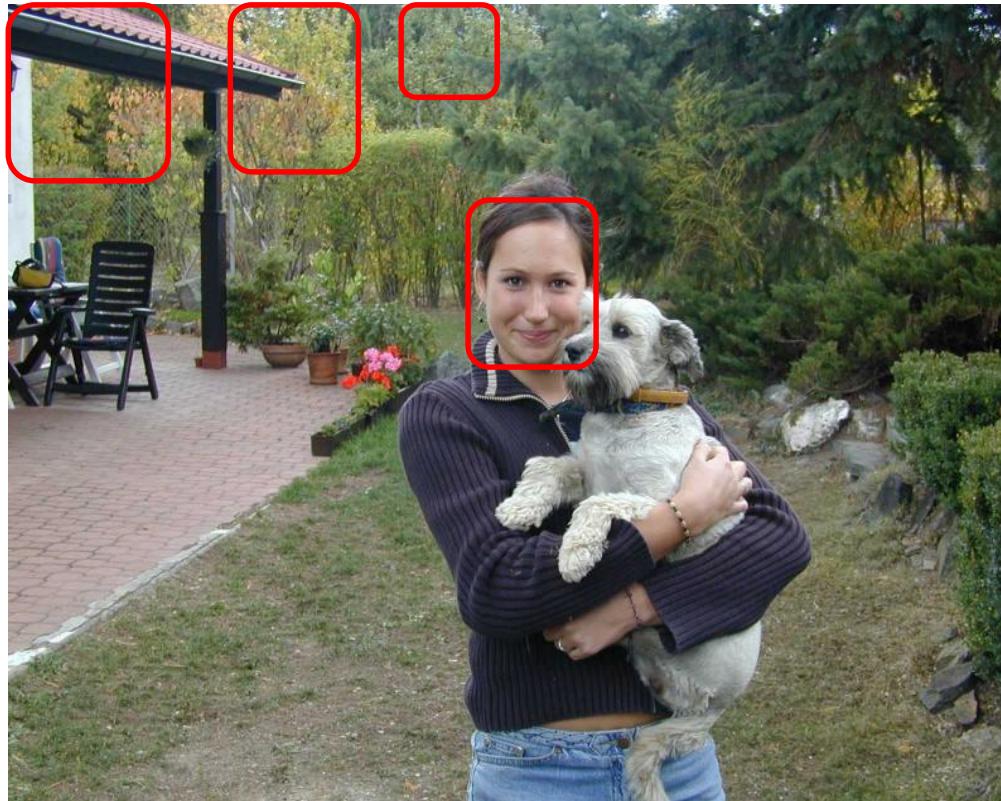


The Viola-Jones Face Detector



Face Detection

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





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 $\sim 10^6$ pixels and a comparable number of candidate face locations
 - To avoid having a false positive in every 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

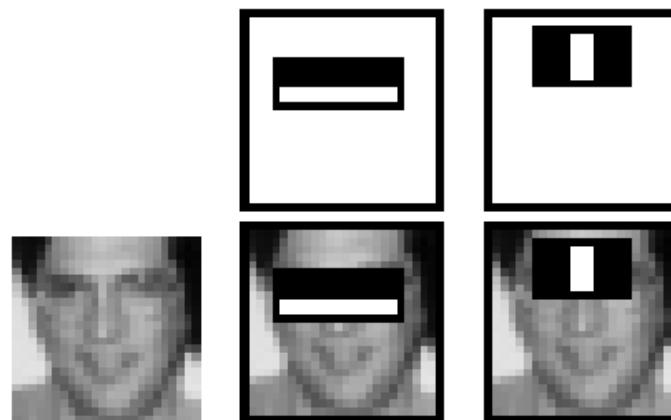
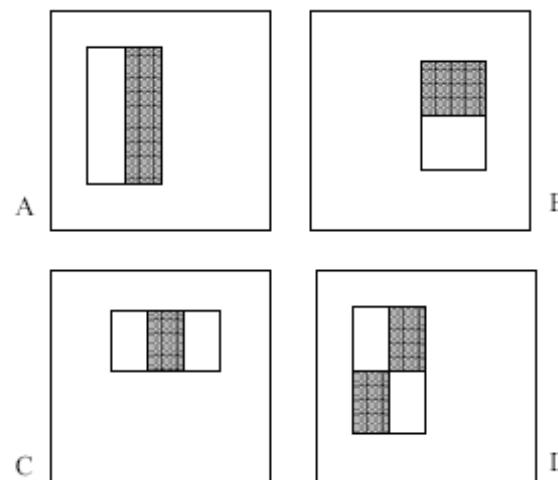
P. Viola and M. Jones. [Rapid object detection using a boosted cascade of simple features.](#)
CVPR 2001.

P. Viola and M. Jones. [Robust real-time face detection.](#) IJCV 57(2), 2004.



Image Features

- “Rectangle filters”: $\text{Value} = \sum(\text{pixels in white area}) - \sum(\text{pixels in black area})$
 - They are **easy** 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

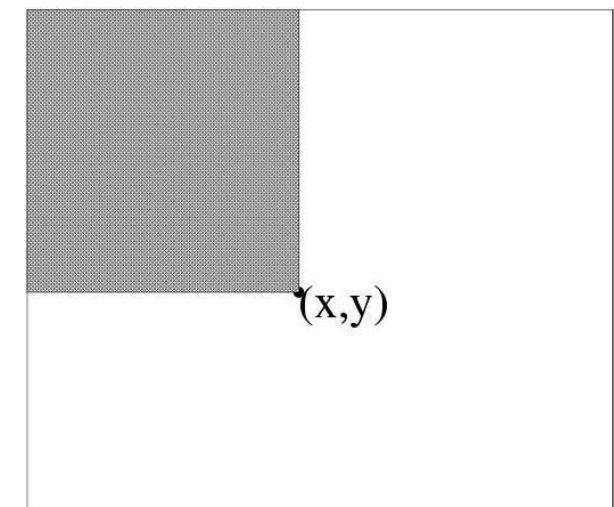
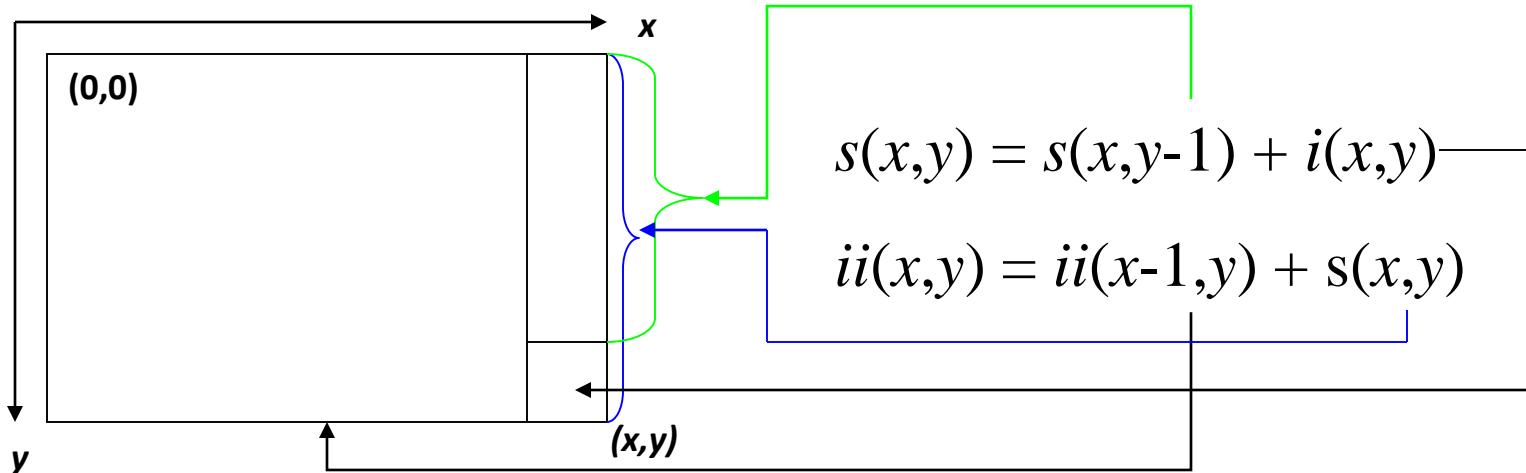


- *two-rectangle feature type*
- *three-rectangle feature type*
- *four-rectangle feature type*



Integral Images

- Def: The *integral image* $ii(x,y)$ at location (x,y) , is the sum of the pixel values above and to the left of (x,y) , inclusive
 - Using the following two recurrences, where $i(x,y)$ is the pixel value of original image at the given location and $s(x,y)$ is the cumulative column sum, we can calculate the integral image representation of the image in a single pass



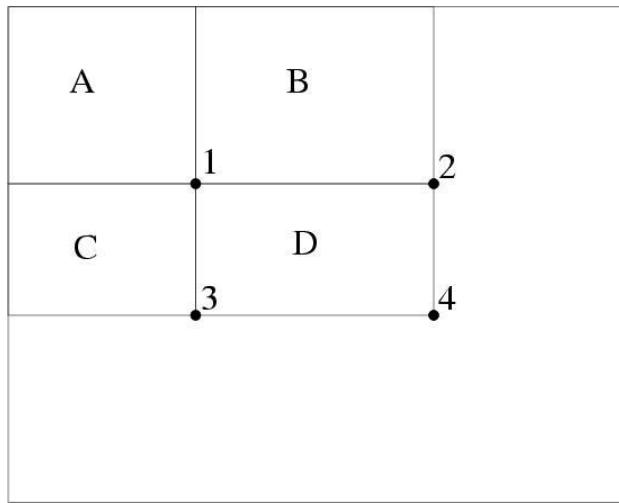


Rapid Evaluation of Rectangular Features

- Using the integral image representation one can compute the value of **any rectangular sum** in constant time.
 - The integral sum inside rectangle D we can compute as:

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

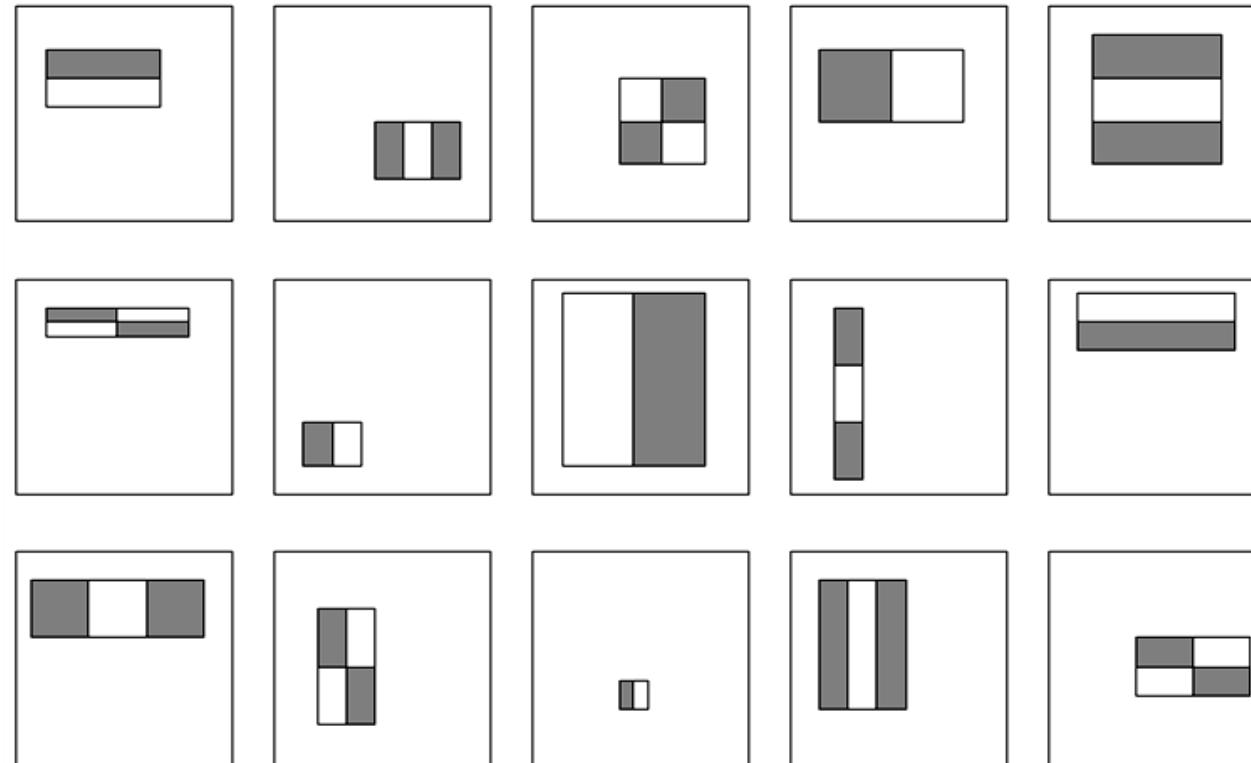
- As a result two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively





Feature Selection

- For a 24×24 detection **region**, the number of possible rectangle features is $\sim 160,000$!





Feature Selection

- For a 24×24 detection region, the number of possible rectangle features is $\sim 160,000$!
- 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?**



Problem

- A image: tens of thousands of location/scale rectangles
- A region of 24×24 : the number of possible rectangle features is $\sim 160,000$
- Faces are rare: 0-10 per image



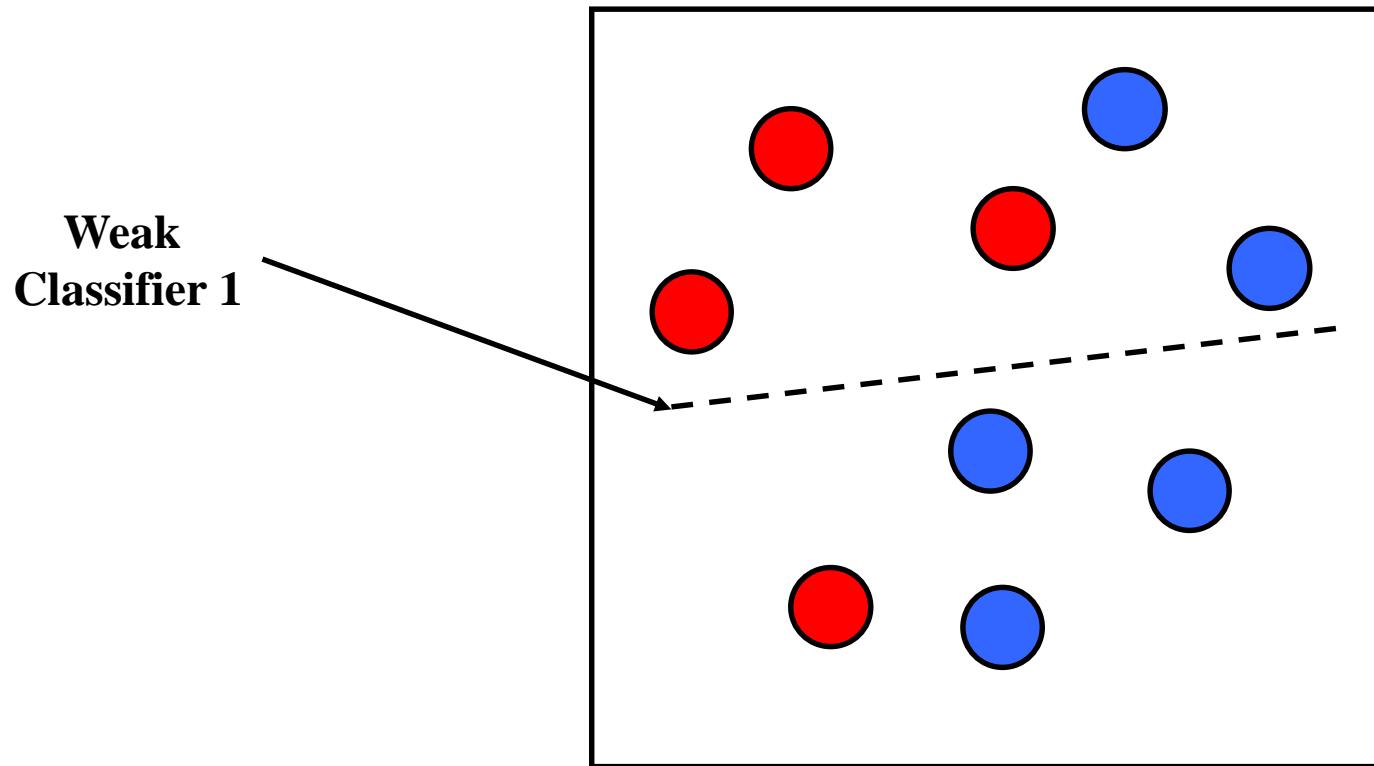
Training Procedure

- 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)

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



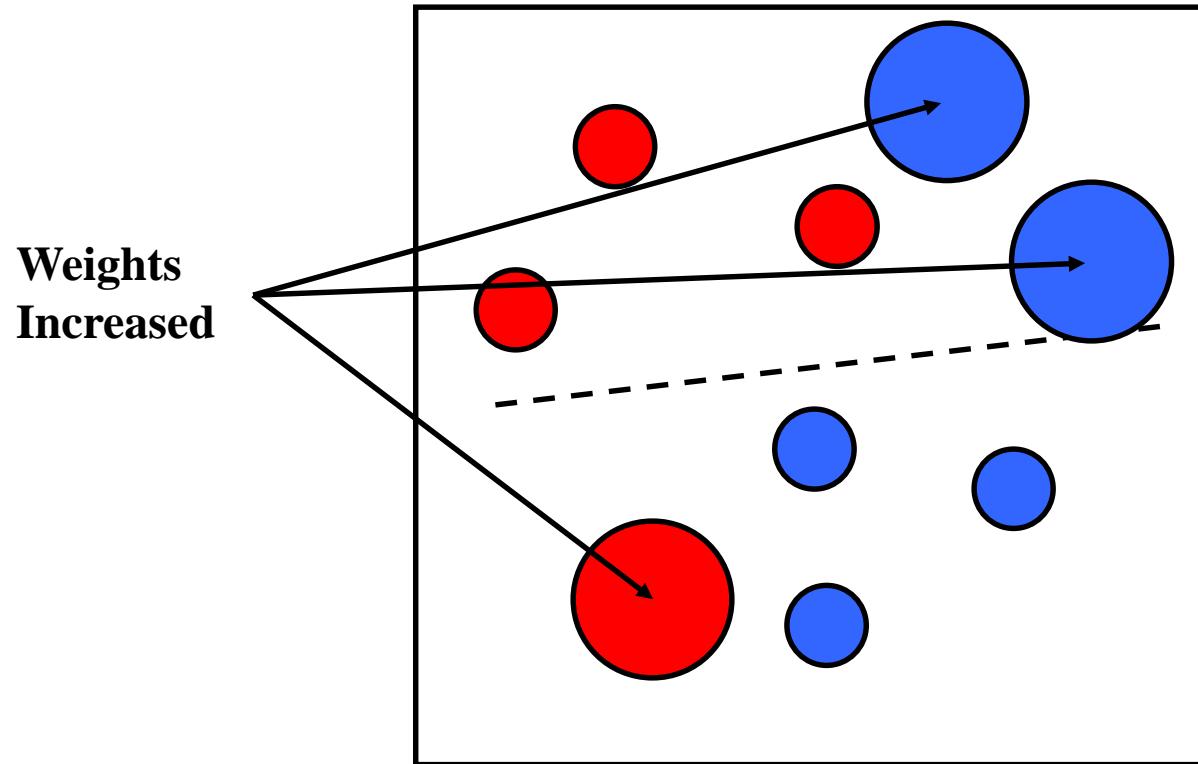
Boosting Illustration



What is the weak classifier?

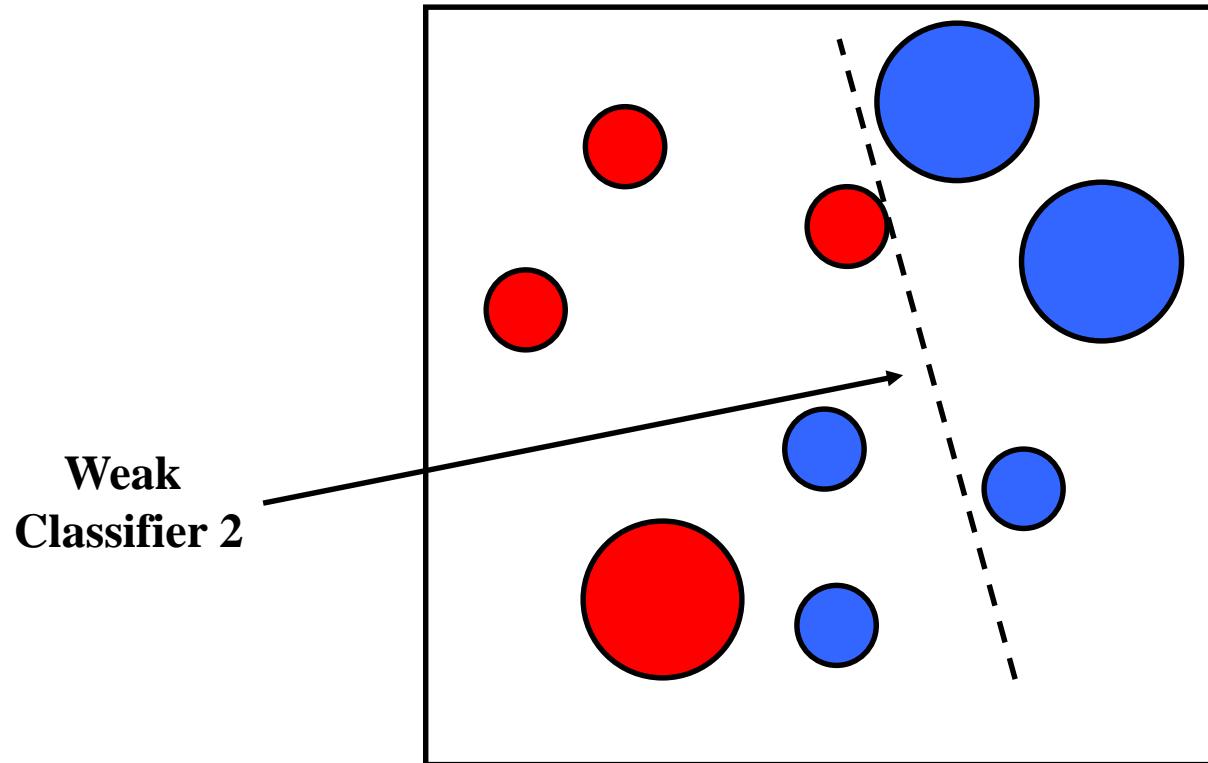


Boosting Illustration



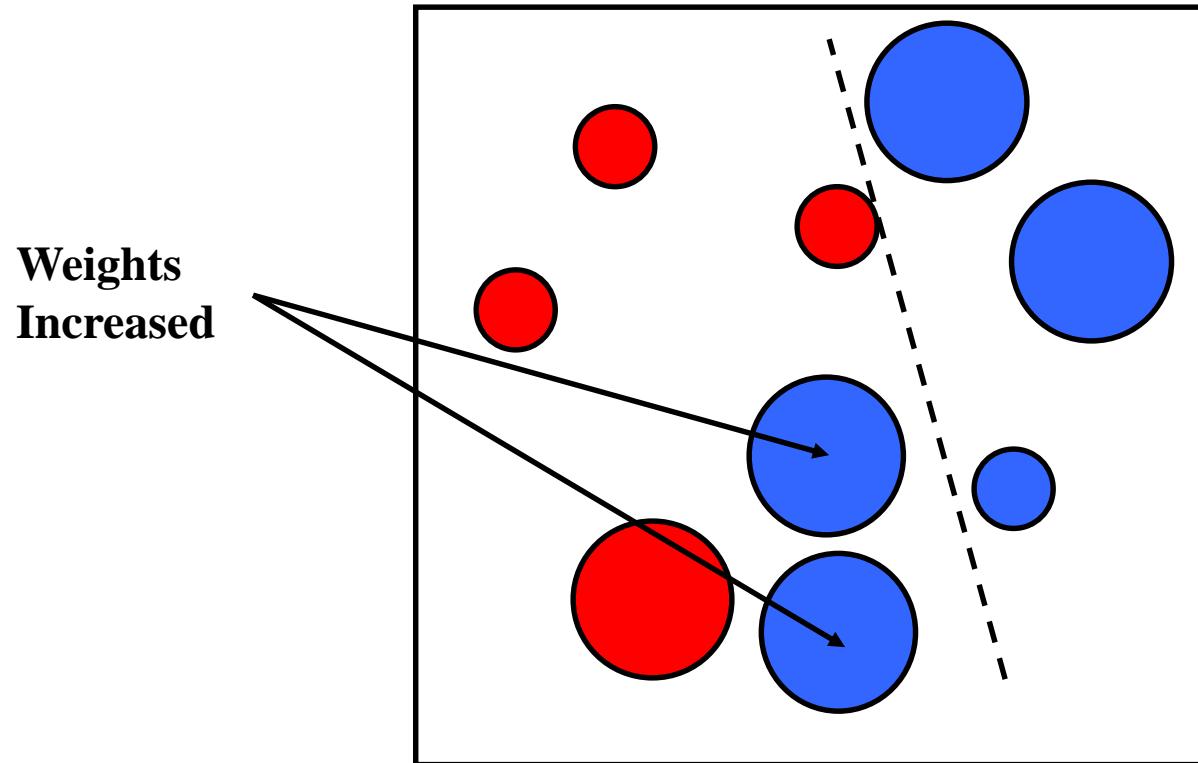


Boosting Illustration



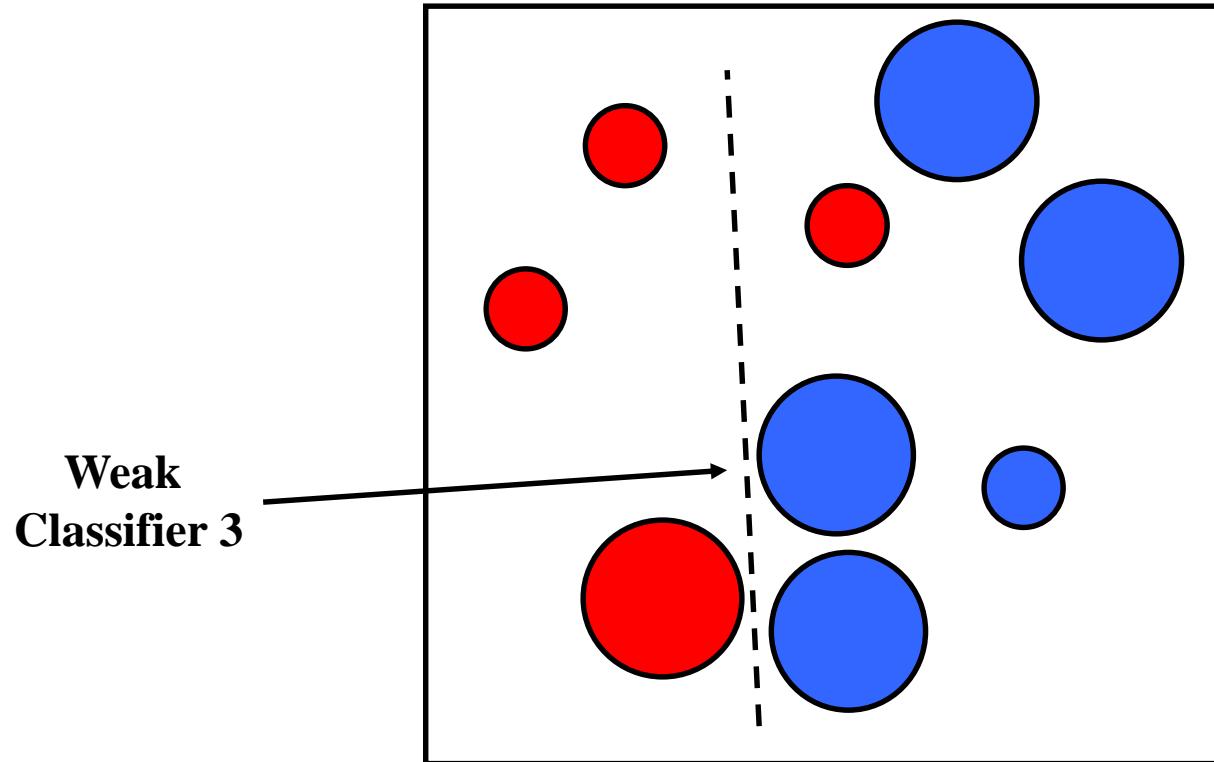


Boosting Illustration





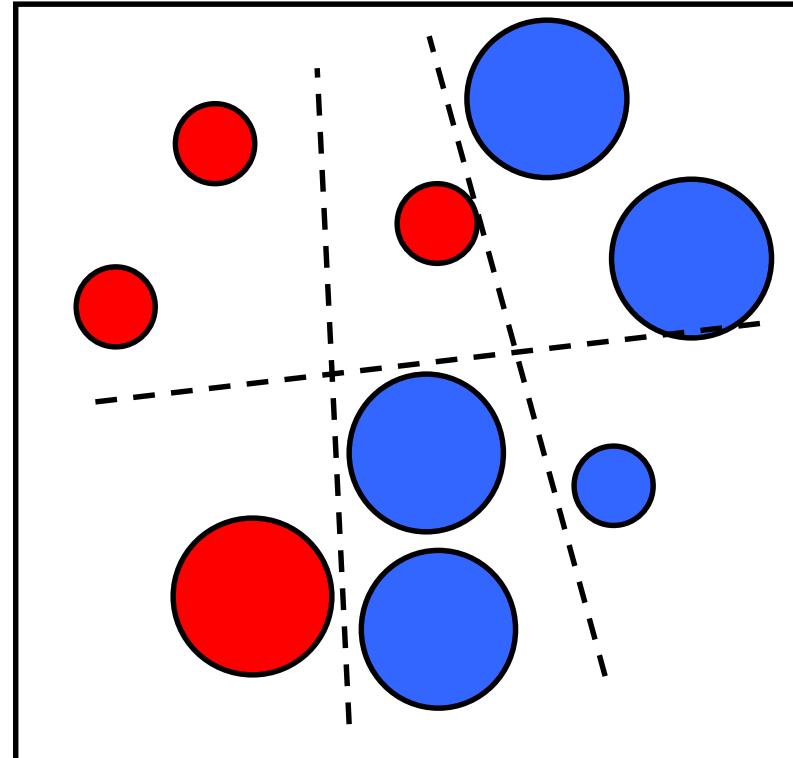
Boosting Illustration





Boosting Illustration

Final classifier is
a linear
combination of
weak classifiers





Boosting vs. SVM

- Advantages of boosting

- Integrates classification with **feature selection** (support vector?)
- Complexity of training is **linear** instead of quadratic in the number of training examples (depending on inner loop)
- Flexibility in the choice of weak learners, boosting scheme
- Testing is fast (no kernel)
- Easy to implement

- Disadvantages

- Needs many training examples
- Often doesn't work as well as SVM (especially for many-class problems)



Boosting for Face Detection

- Define weak learners based on rectangle features

Parity: indicating
the direction of the
inequality sign

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

value of rectangle feature

threshold

window



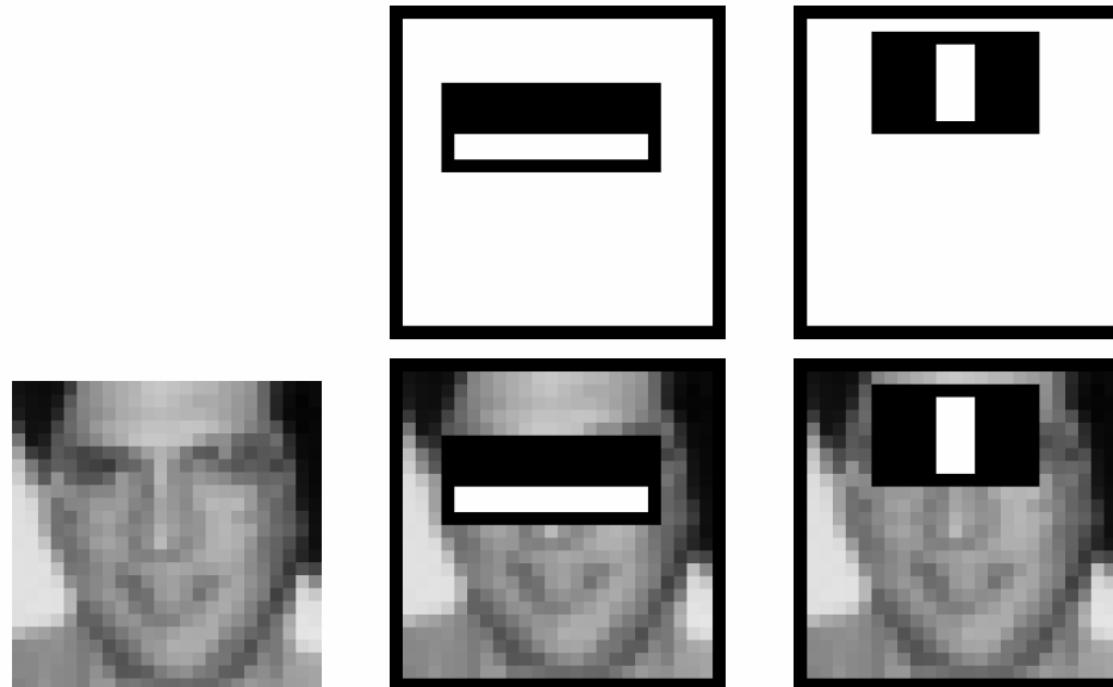
Boosting for Face Detection

- 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

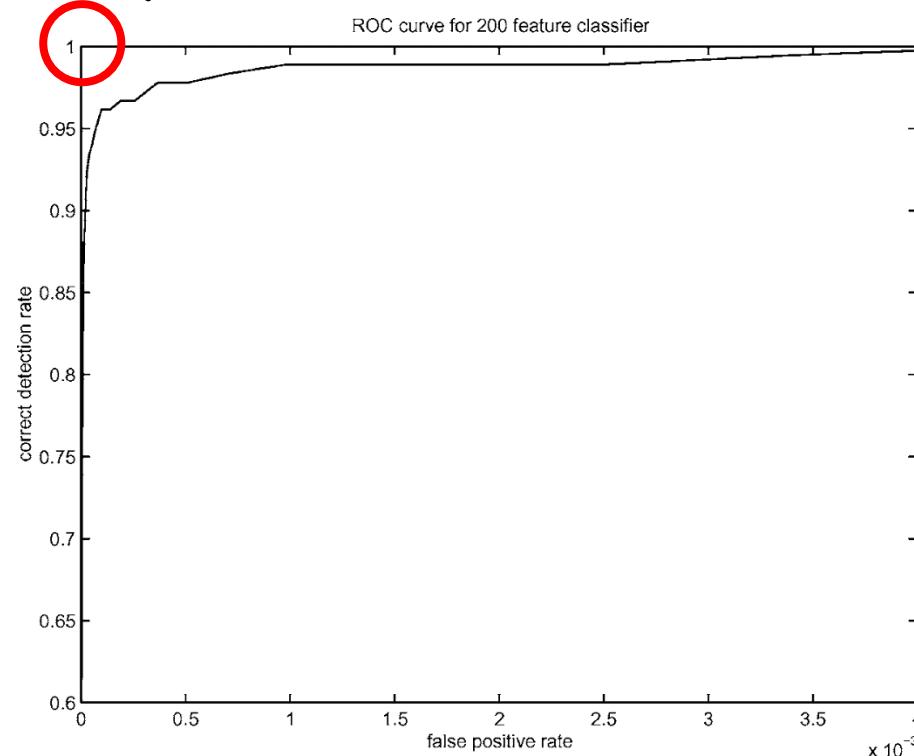
- First **two features** selected by boosting:
 - This feature combination can yield 100% detection rate and 50% **false positive rate** (what does it mean?)





Boosting for face detection

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

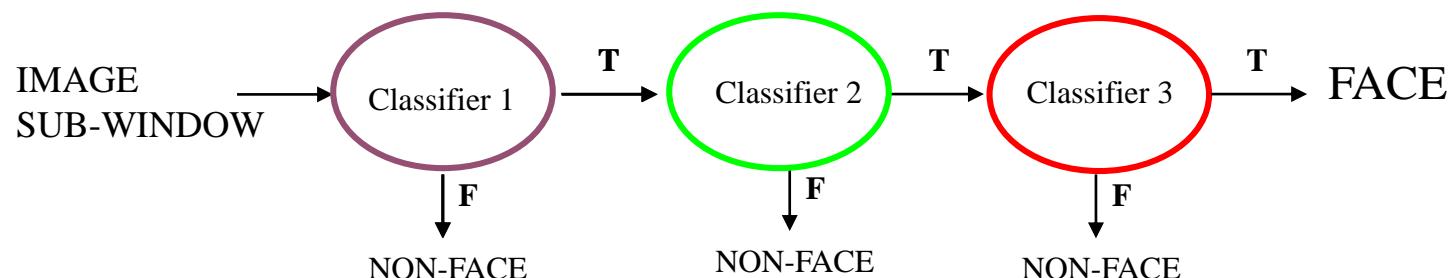


Receiver operating characteristic (ROC) curve



Attentional Cascade

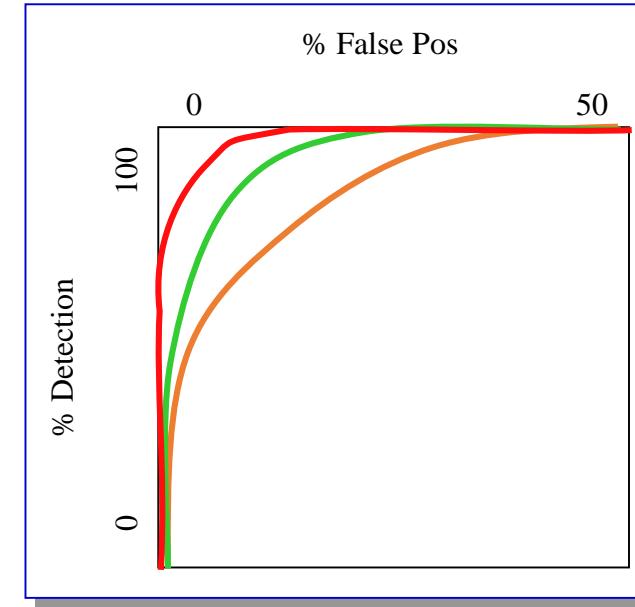
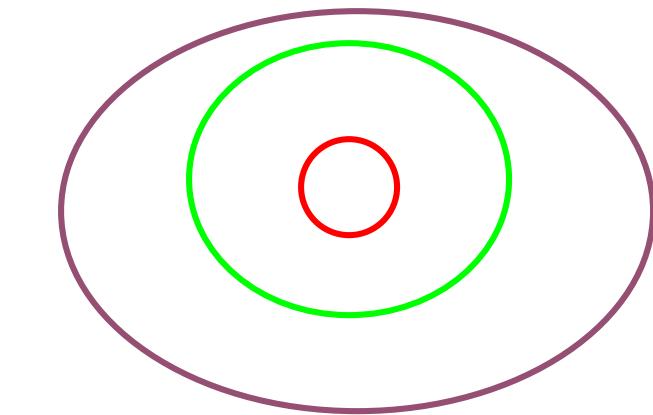
- We start with simple classifiers which **reject many of the negative** sub-windows while detecting **almost all positive** sub-windows (e.g. the first two features)
- **Positive response** from the first classifier **triggers** the evaluation of a **second** (more complex) classifier
- A **negative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



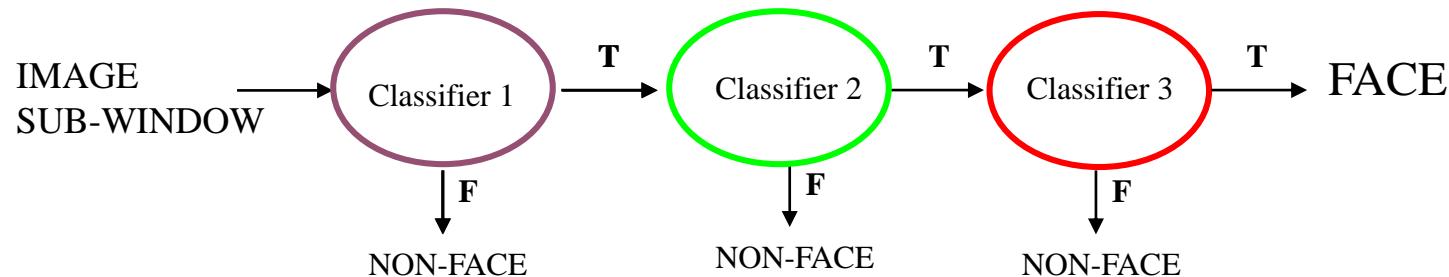


Attentional Cascade

- Chain classifiers that are progressively **more complex** and have **lower false positive rates**:



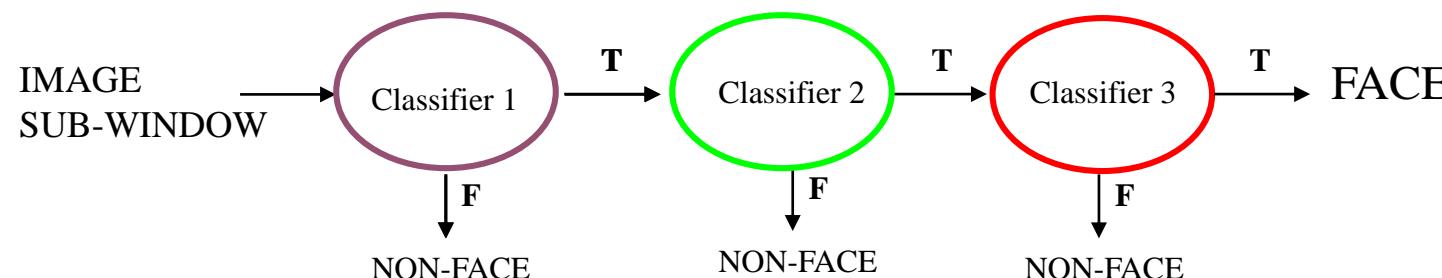
Receiver
operating
characteristic





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}
 - A 10-stage cascade
 - Each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$)
 - Each stage has a false positive rate of 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$)



What does each classifier look like?
Every stage rejects a little bit



Training the Cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
 - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
 - Test on a validation set
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage



The Implemented System

- Training data
 - 5000 faces
 - ✓ All frontal, rescaled to 24x24 pixels
 - 300M sub-windows of non-faces
 - ✓ 9500 non-face images
 - Faces are normalized
 - ✓ Scale, translation
- Many variations
 - Across individuals
 - Illumination
 - Pose



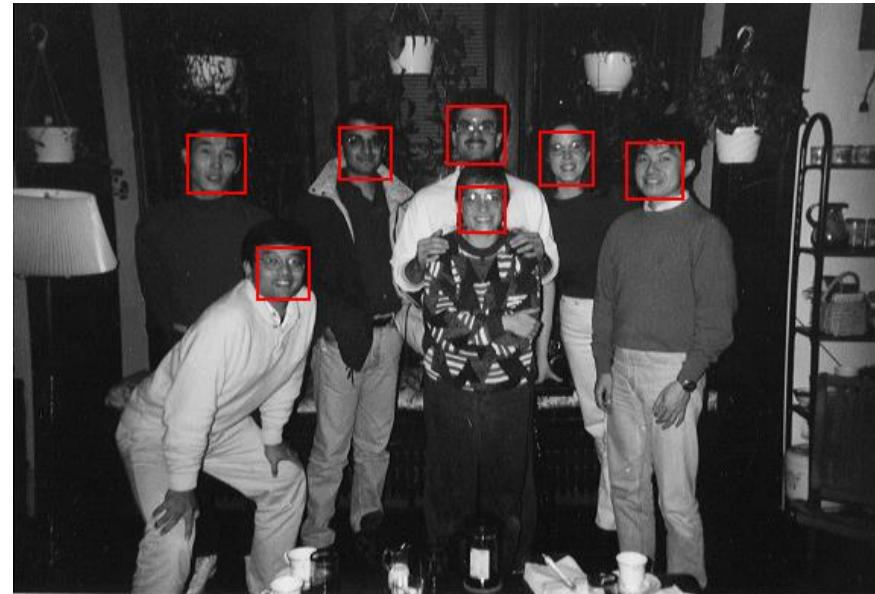
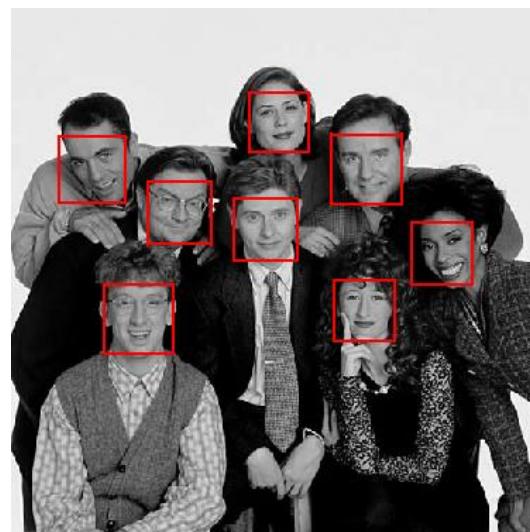
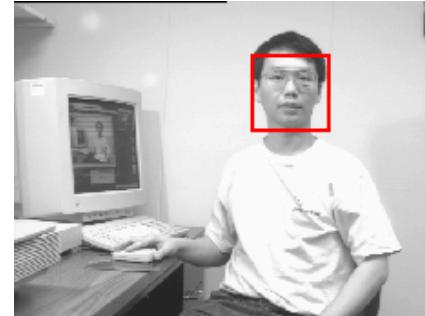
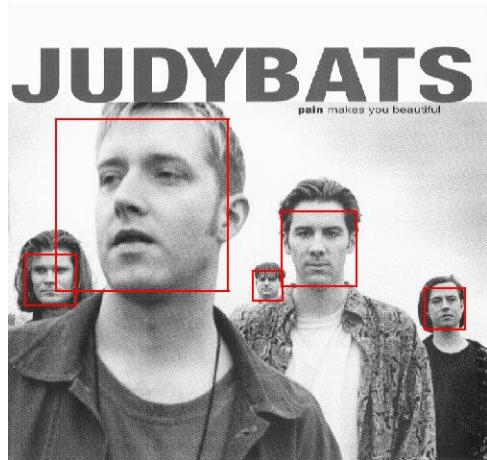


System Performance

- 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)



Output of Face Detector on Test Images

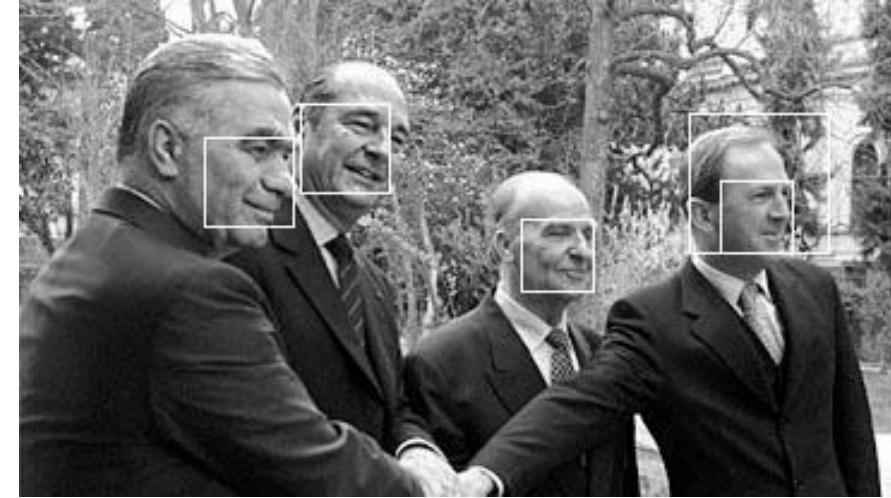




Other detection tasks

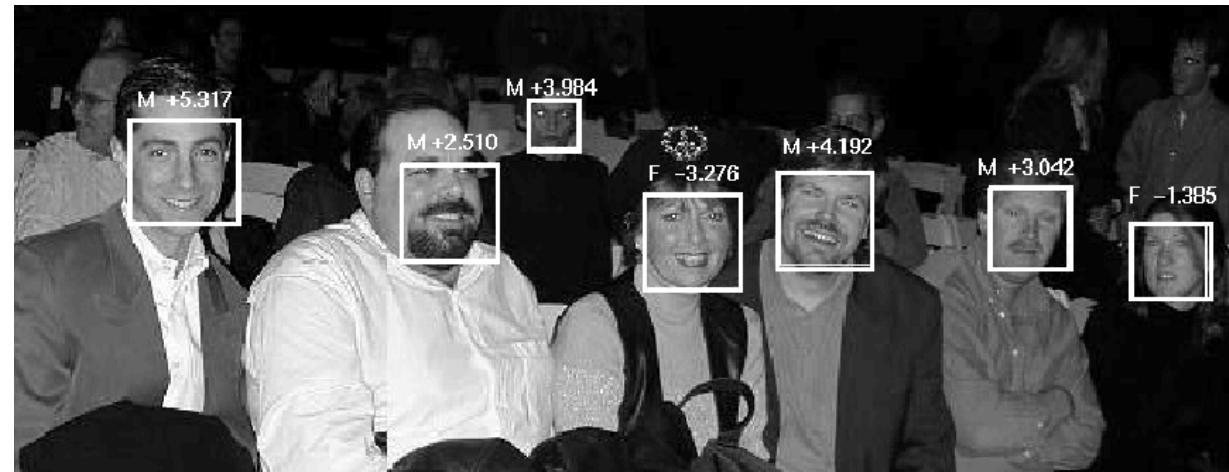


Facial Feature Localization



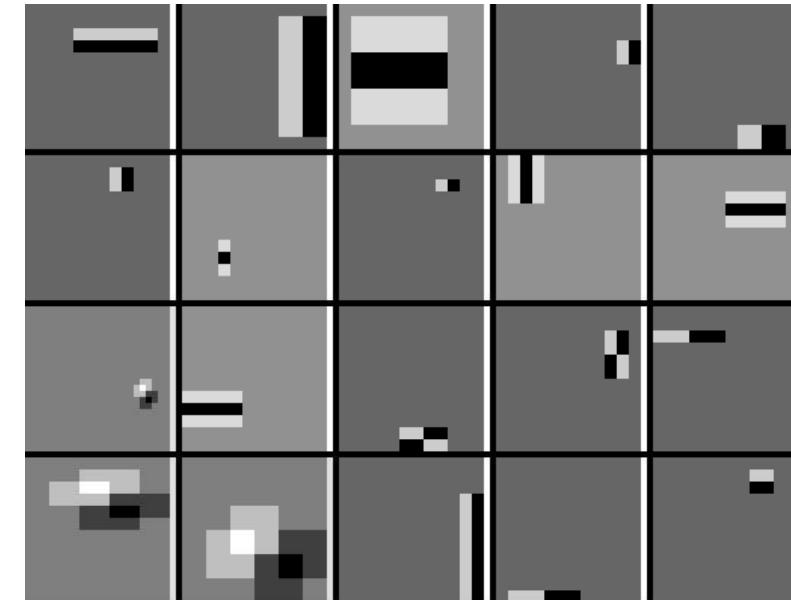
Profile Detection

Male vs.
female





Profile Features





Thinking

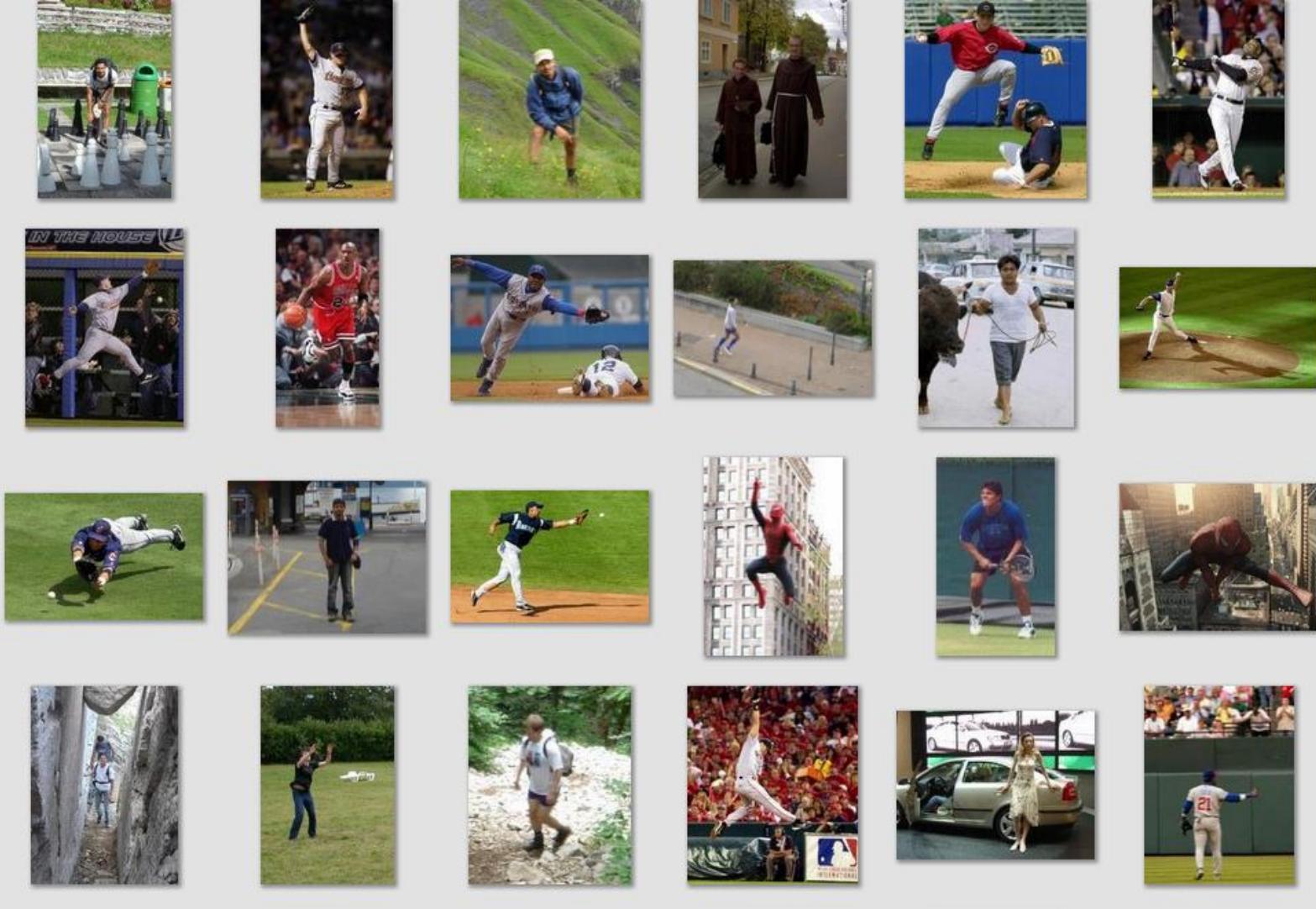
- How to detect a pedestrian?
- What is the difference between the two tasks?



Deformable Part Models

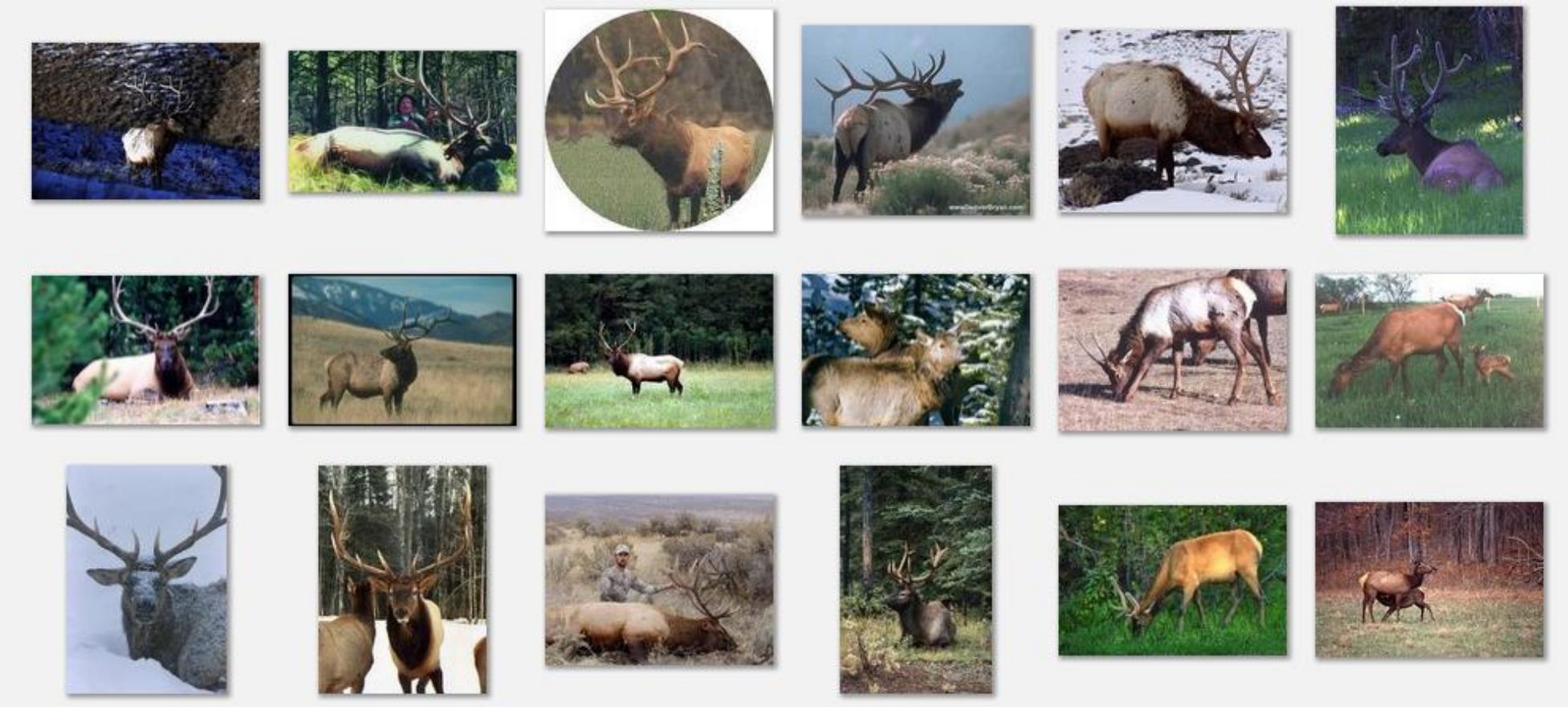


Deformable Objects





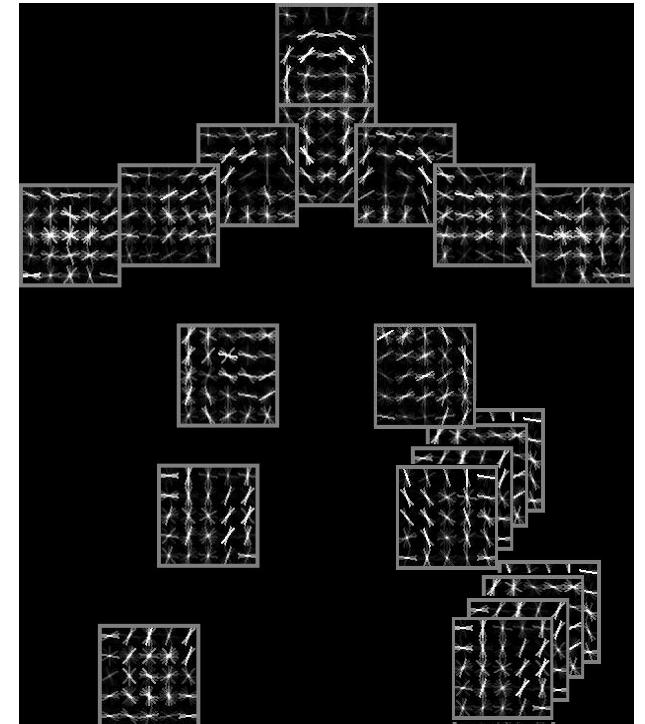
Non-rigid Objects





Challenges

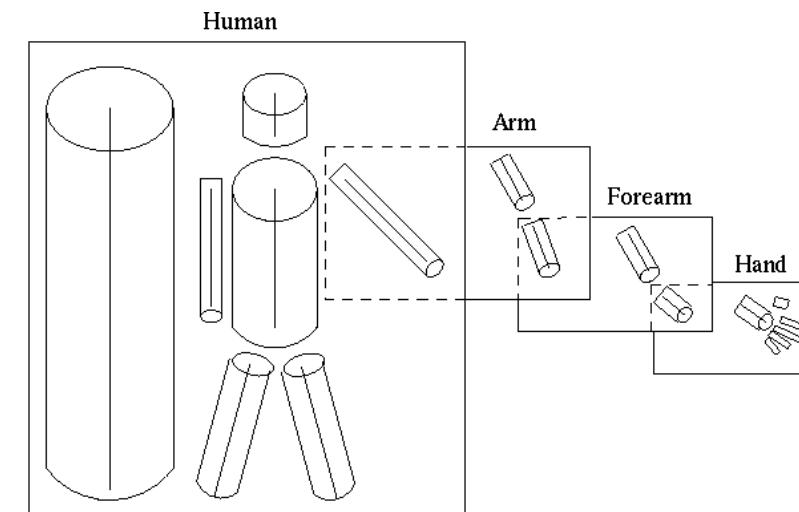
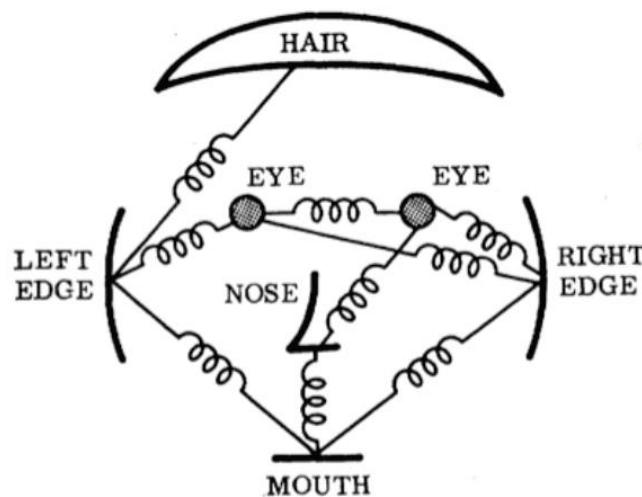
- High intra-class variations
 - Change in view point, deformation, and scale
 - Four small images of a human hand in different poses: raised, open-palmed, bent, and closed, illustrating the concept of deformation.
 - **Deformable**
- Therefore...
 - **Part-based model** might be a better choice !
 - Think: how to proceed the part-based model?





Part-based Representation

- Objects are decomposed into **parts** and **spatial relations** among parts
- E.g. Face model by Fischler and Elschlager '73
- E.g. David Marr at MIT(1978): proposing a **bottom-up** approach to scene understanding





Part-based Representation

- *K*-fans model (D.Crandall, et.all, 2005)

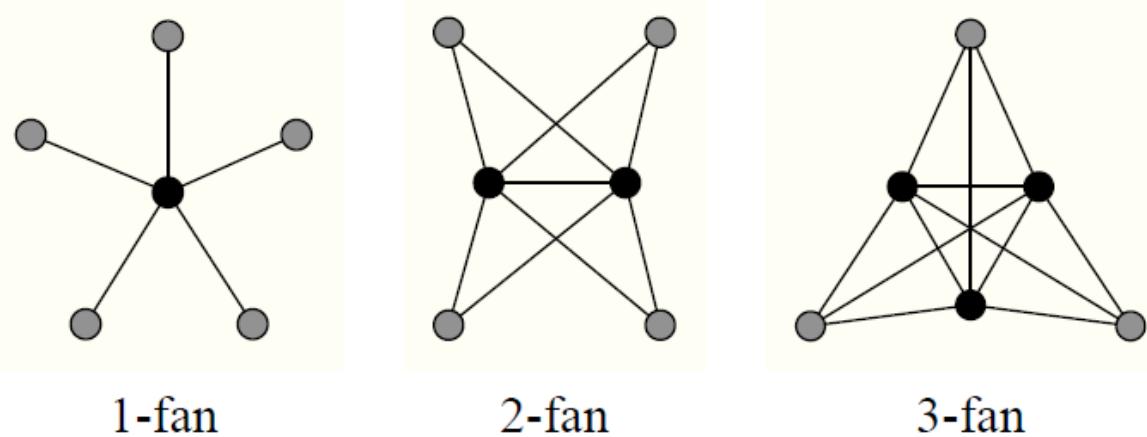
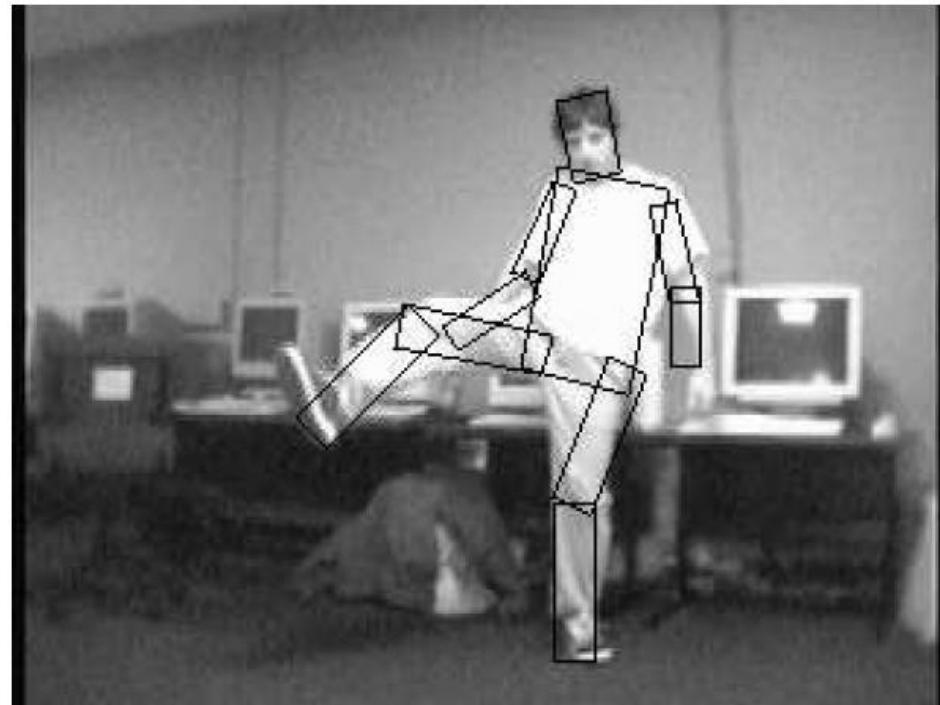
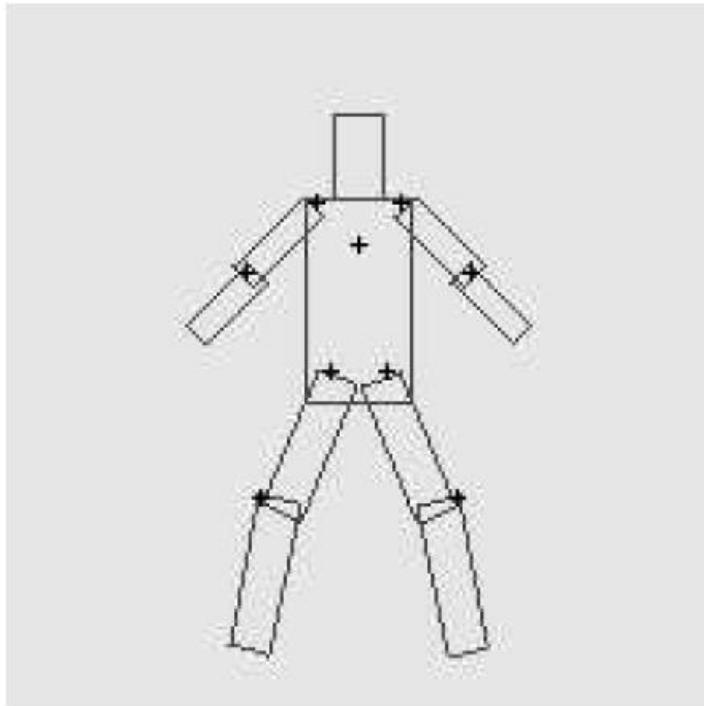


Figure 1. Some k -fans on 6 nodes. The reference nodes are shown in black while the regular nodes are shown in gray.



Part-based Representation

- Tree model → Efficient inference by dynamic programming





Pictorial Structure

- Matching = Local part evidence + Global constraint

$$L^* = \arg \min_L \left(\sum_{i=1}^n m_i(l_i) + \sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j) \right)$$

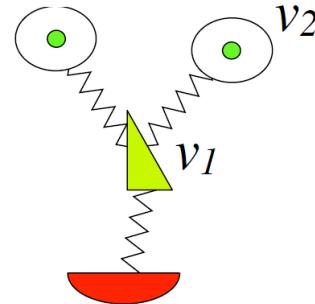
It is not the Euclidean distance

- $m_i(l_i)$: matching cost for part i
- $d_{ij}(l_i, l_j)$: deformable cost for connected pairs of parts
- (v_i, v_j) : connection between part i and j



Matching on Tree Structure

$$E(L) = \sum_{i=1}^n m_i(l_i) + \sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j)$$



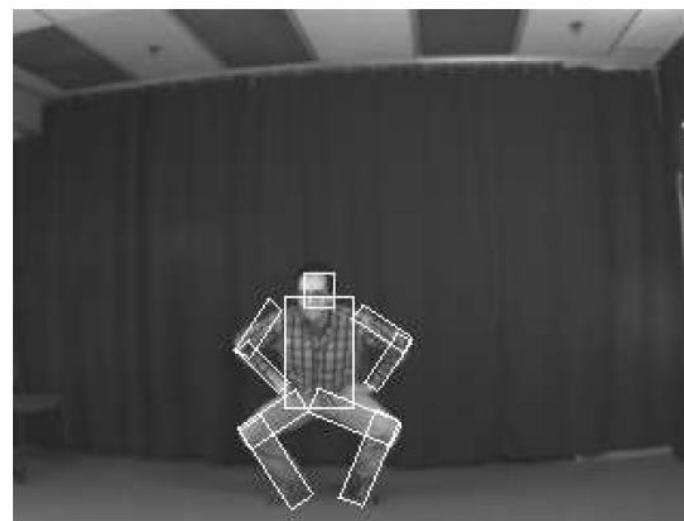
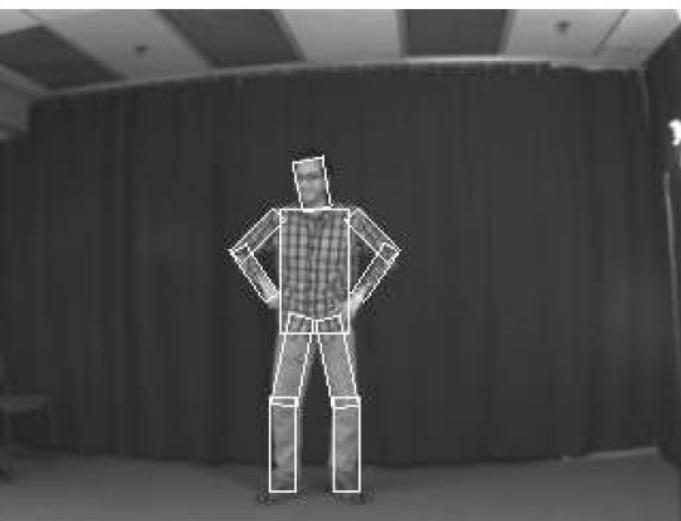
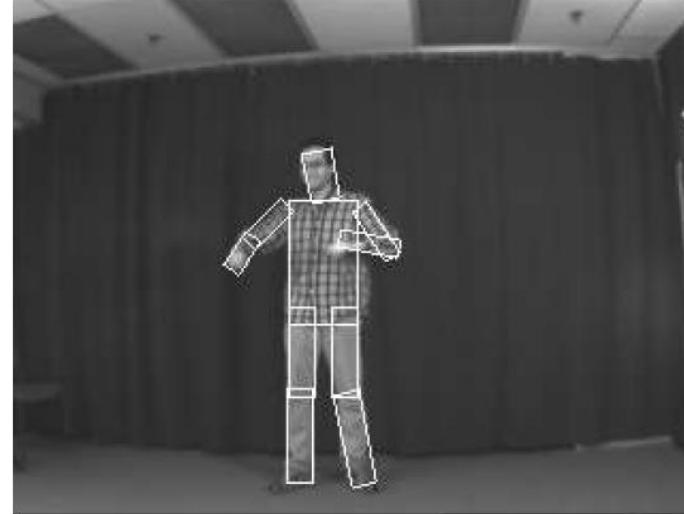
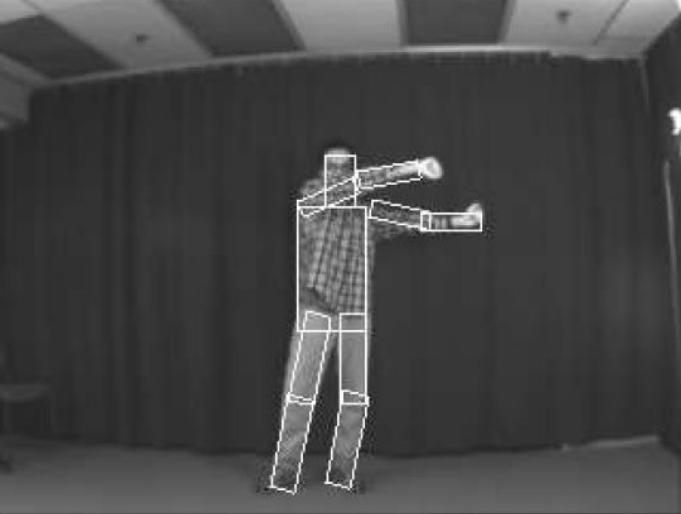
- For each l_1 , find best l_2 :

$$\text{Best}_2(l_1) = \min_{l_2} [m_2(l_2) + d_{12}(l_1, l_2)]$$

- Remove v_2 , and repeat with smaller tree, until only a single part
- Complexity: $O(nk^2)$: n parts, k locations per part

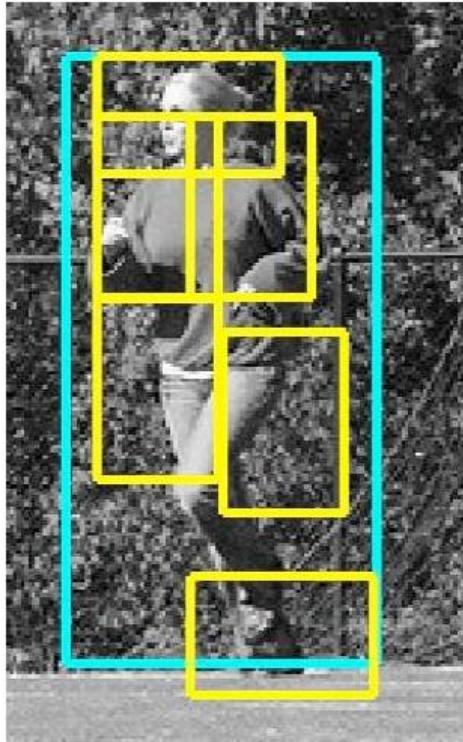


Sample Result on Matching Human

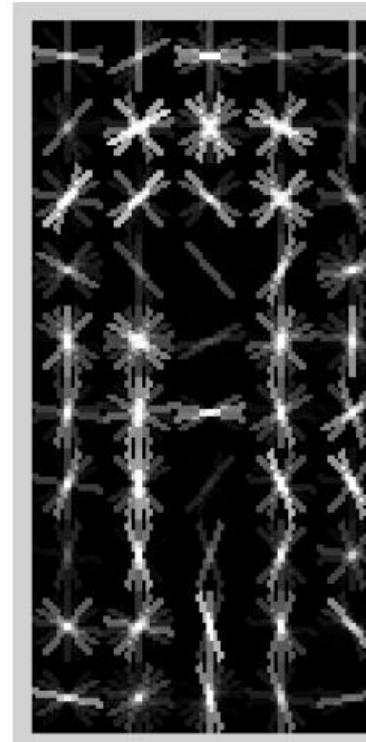




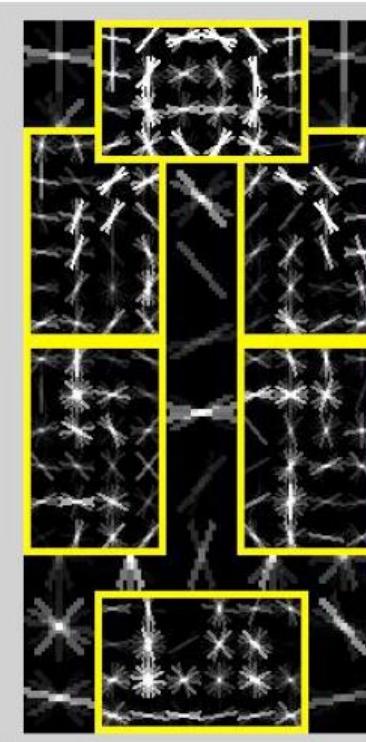
A Discriminatively Trained, Multiscale, Deformable Part Model



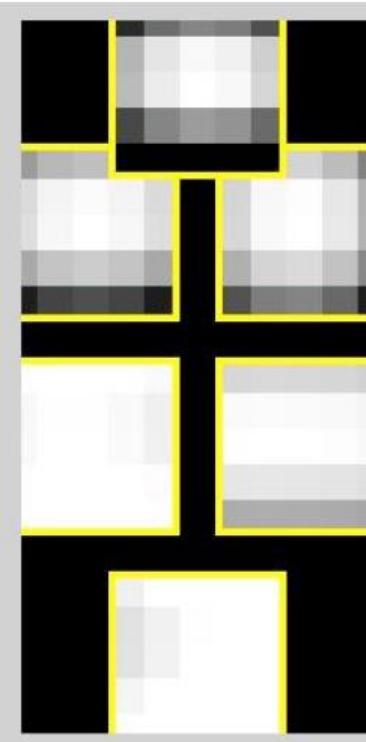
detection



root filter



part filters



deformation
models

Pedro Felzenszwalb, David McAllester and Deva Ramanan

A Discriminatively Trained, Multiscale, Deformable Part Model. IEEE TPAMI, 2010.



Thinking

- How to match each part?
- What is the deformation loss?



Histogram of Oriented Gradients

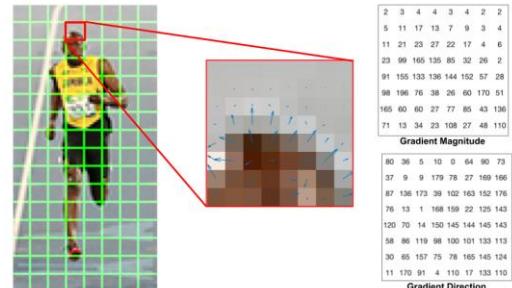
Step 1: Calculate the Gradient Images

-1	0	1
0		
1		

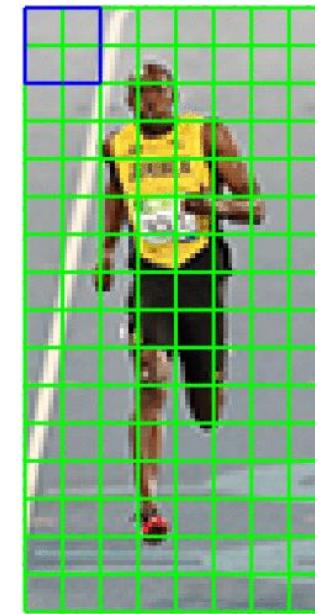
$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$



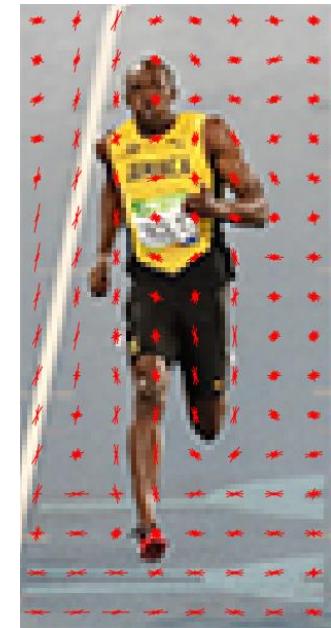
Step 2: Calculate Histogram of Gradients in 8×8 cells



Step 3: 16×16 Block Normalization



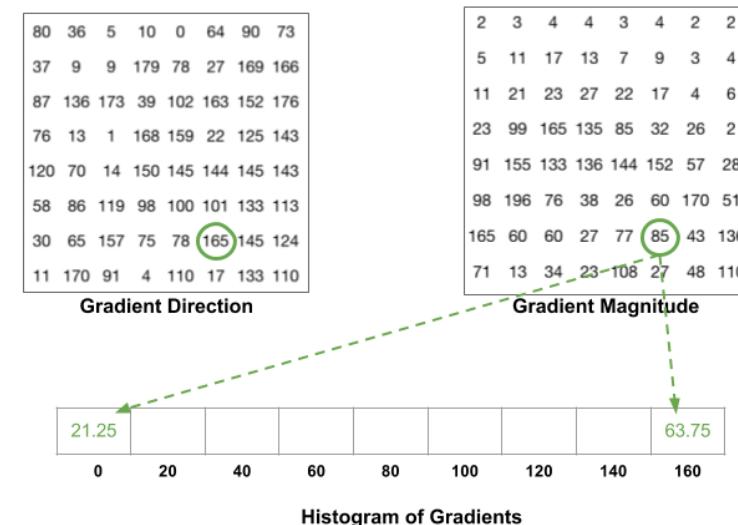
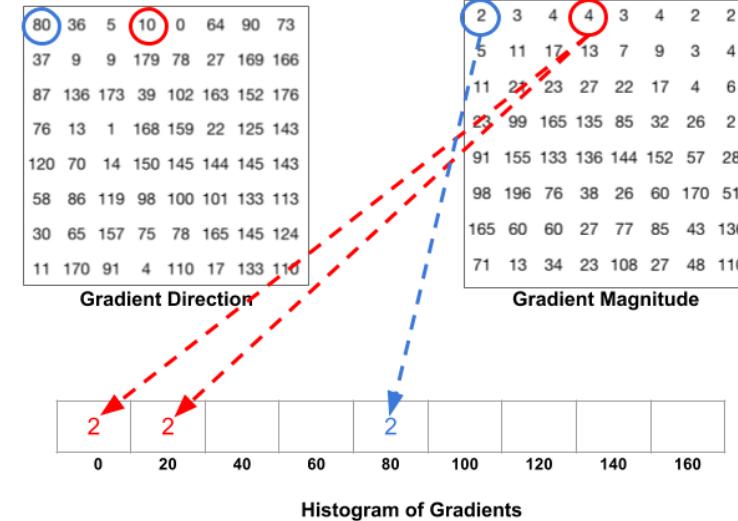
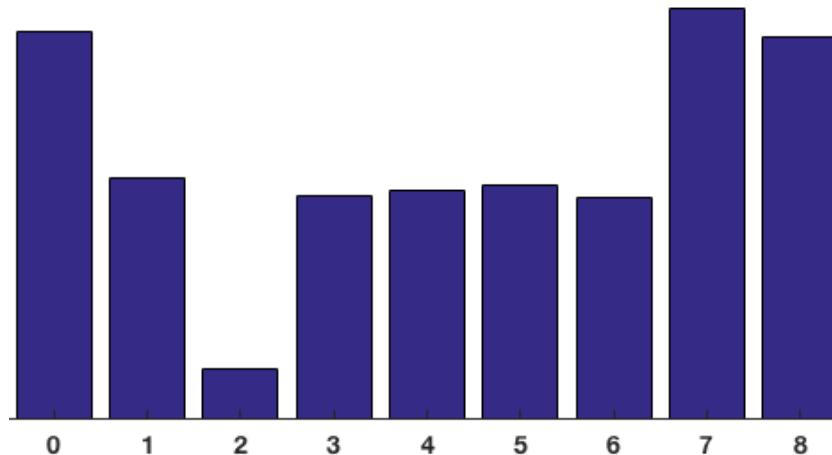
Step 4: Calculate the HOG feature vector





Histogram of Oriented Gradients

- Create a histogram of gradients in these 8×8 cells which contains 9 bins corresponding to angles 0, 20, 40 ... 160

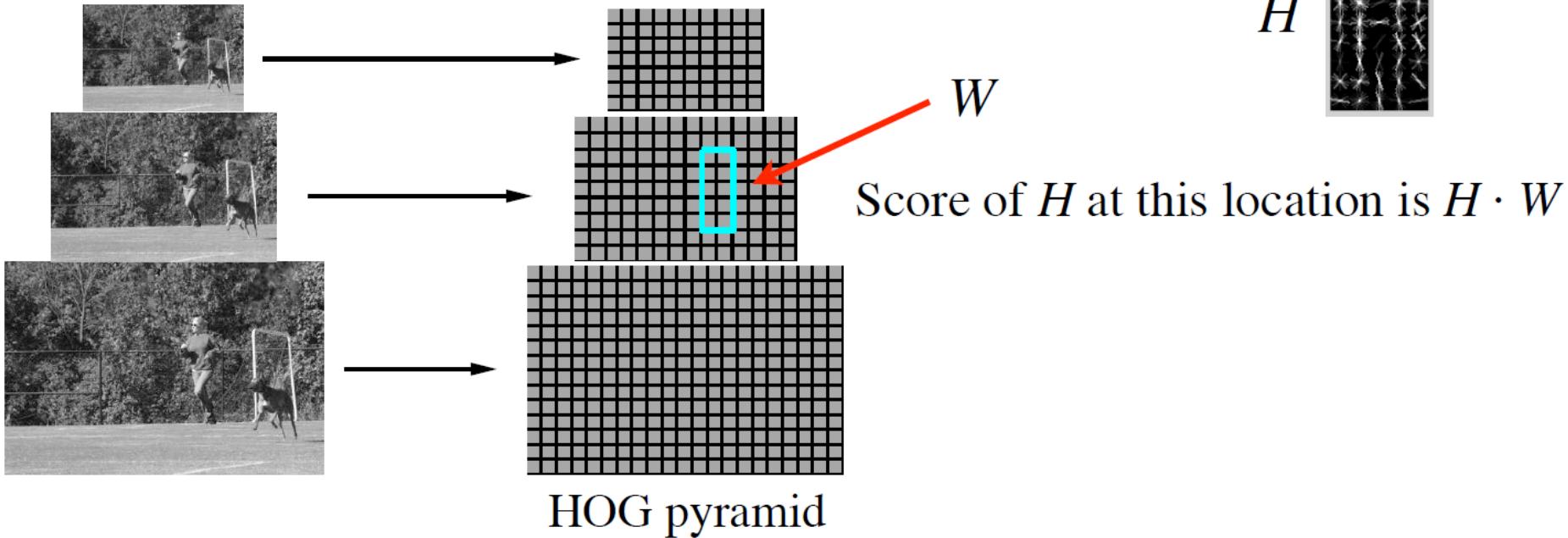




Filters = Learned Classifiers

- Filters are rectangular templates defining weights for features

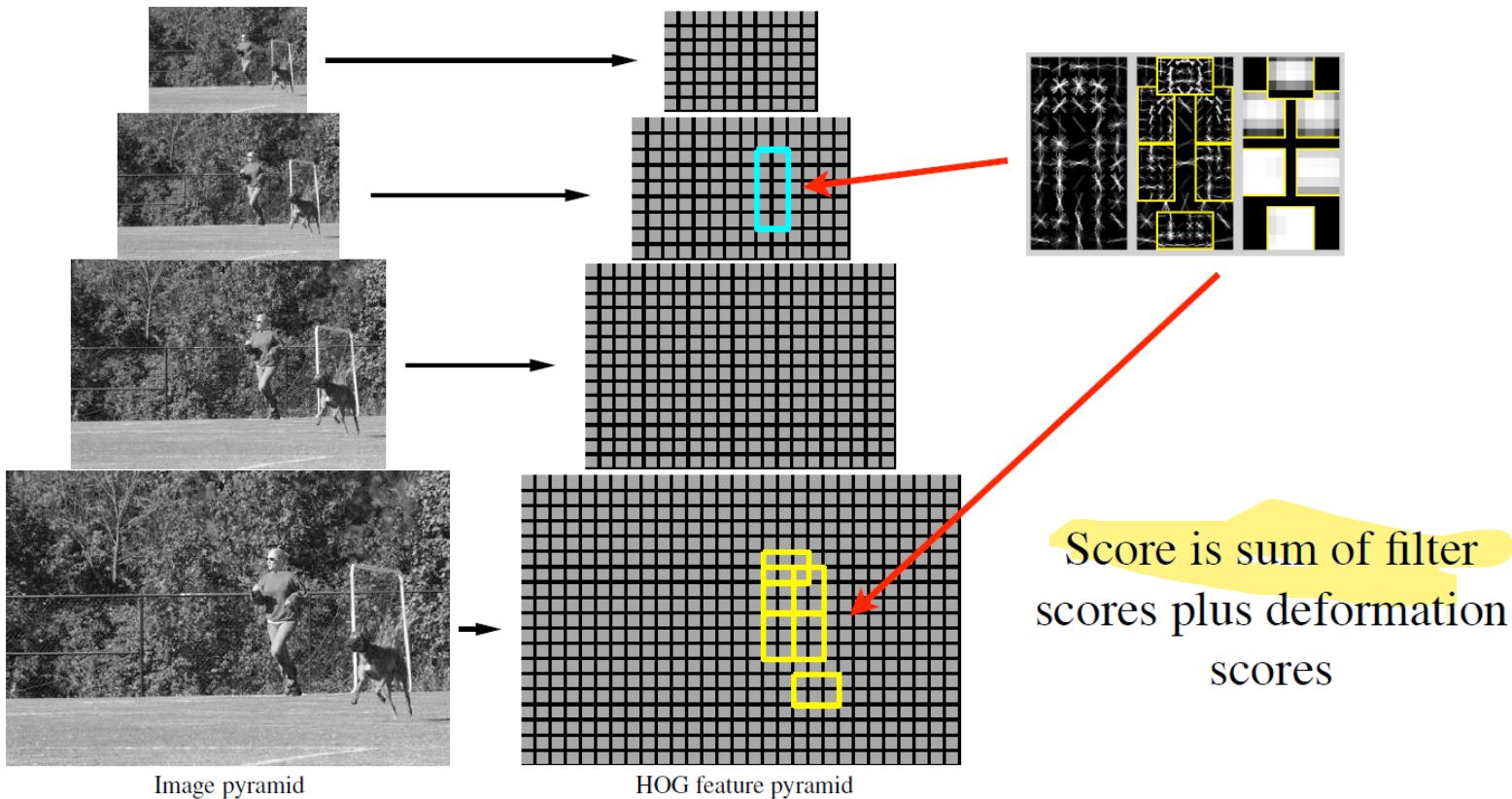
Think: How to obtain this filter?





Object Hypothesis

- **Coarser level for the root filter (whole object) and higher level for part filters**





Deformable Parts

- A model consists of a **root filter** F_0 and **part model** (P_1, \dots, P_n) ,
 $P_i = (F_i, v_i, s_i, a_i, b_i)$:
 - Filter F_i , **location and size** of part (v_i, s_i) , and parameter to evaluate the **placement** of part (a_i, b_i)
- Score a placement

$$\sum_{i=0}^n F_i \cdot \phi(H, p_i) + \sum_{i=1}^n a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot (\tilde{x}_i^2, \tilde{y}_i^2), \quad (1)$$

where $(\tilde{x}_i, \tilde{y}_i) = ((x_i, y_i) - 2(x, y) + v_i)/s_i$ gives the location of the i -th part relative to the root location. Both \tilde{x}_i and \tilde{y}_i should be between -1 and 1 .

→ Using dynamic programming to find best placement



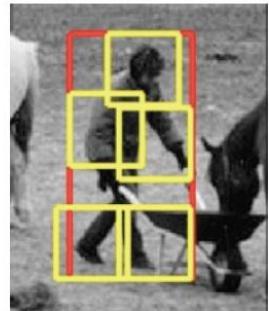
Learning

- Training data consists of images with labeled bounding boxes → Learn the model structure, filters and deformation costs





SVM-like Model



w is a model
 x is a detection window
 z are filter placements

$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

concatenation of filters and deformation parameters

concatenation of features and part displacements



Latent SVM

- Linear SVM (convex) when z is fixed

$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

- Solving by coordinate descent

1. Fixed w , find the latent variable z for the positive examples

$$z_i = \operatorname{argmax}_{z \in Z(x_i)} w \cdot \Phi(x, z).$$

2. Fixed z , solve the Linear SVM to find w



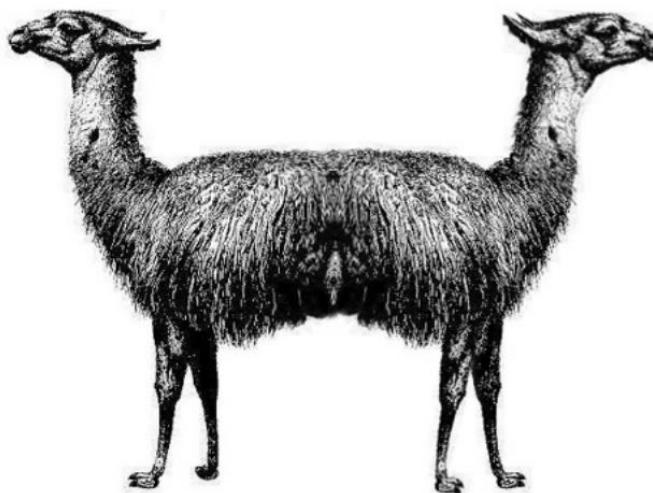
Implementation Details

- Select root filter window size
- Initialize root filter by training model without latent variables on un-occluded examples
- Root filter update: get new positives (best score and significant overlap with ground truth), add to positives and retrain
- Part initialization: sequentially choosing area a having high positive score and $6a = 80\%$ root area



Problem

$$(\text{horse} + \text{horse}) / 2 = \text{?}$$

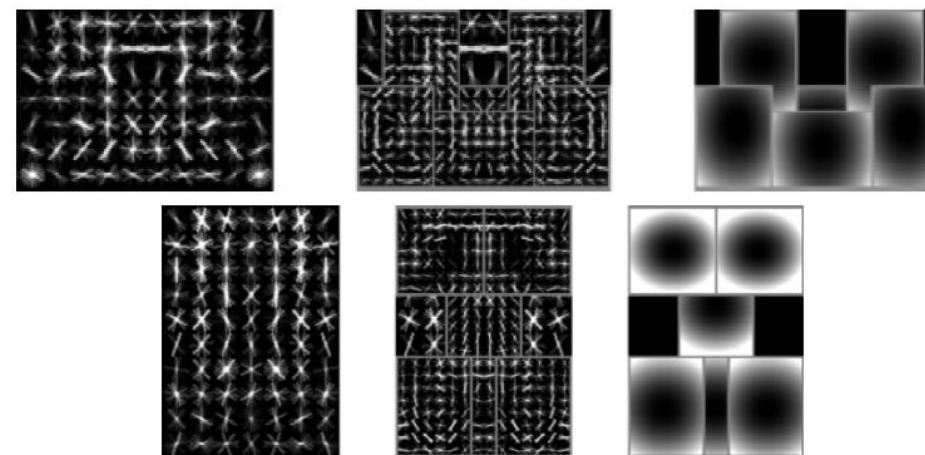
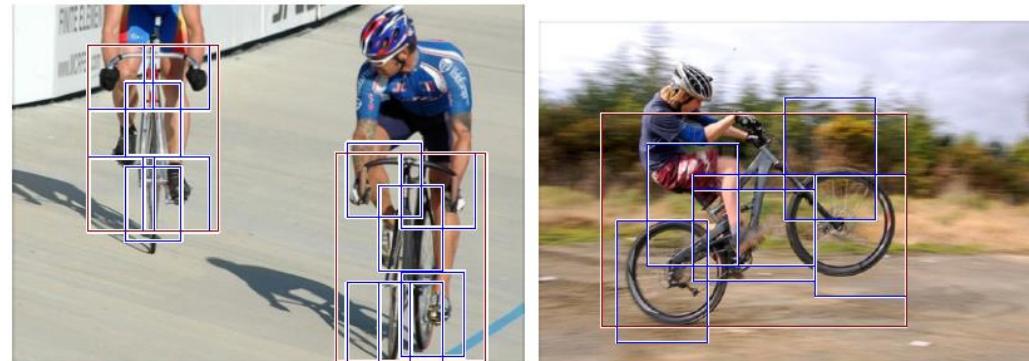


This was supposed to
detect horses



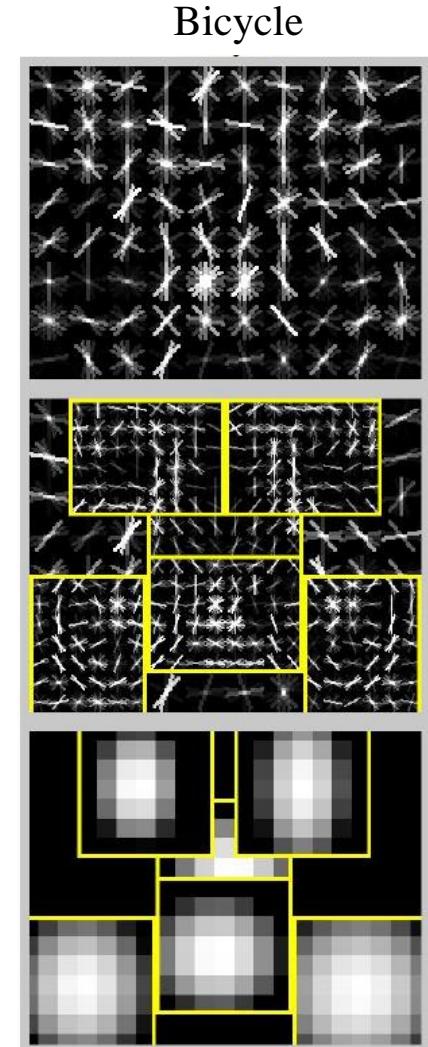
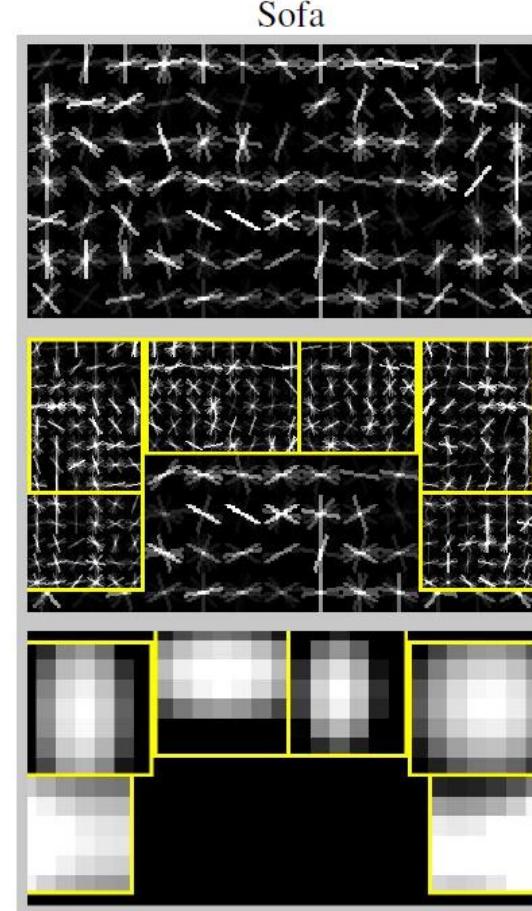
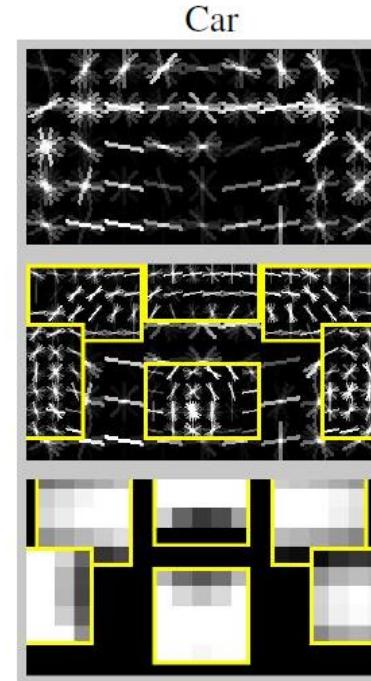
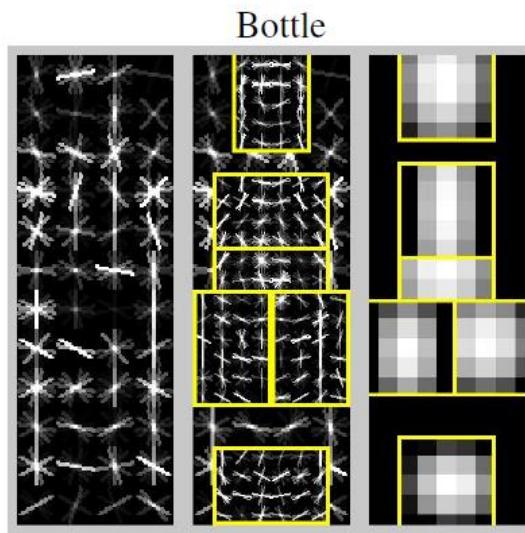
Deformations Mixture Models

- Detections obtained with a 2 component bicycle model



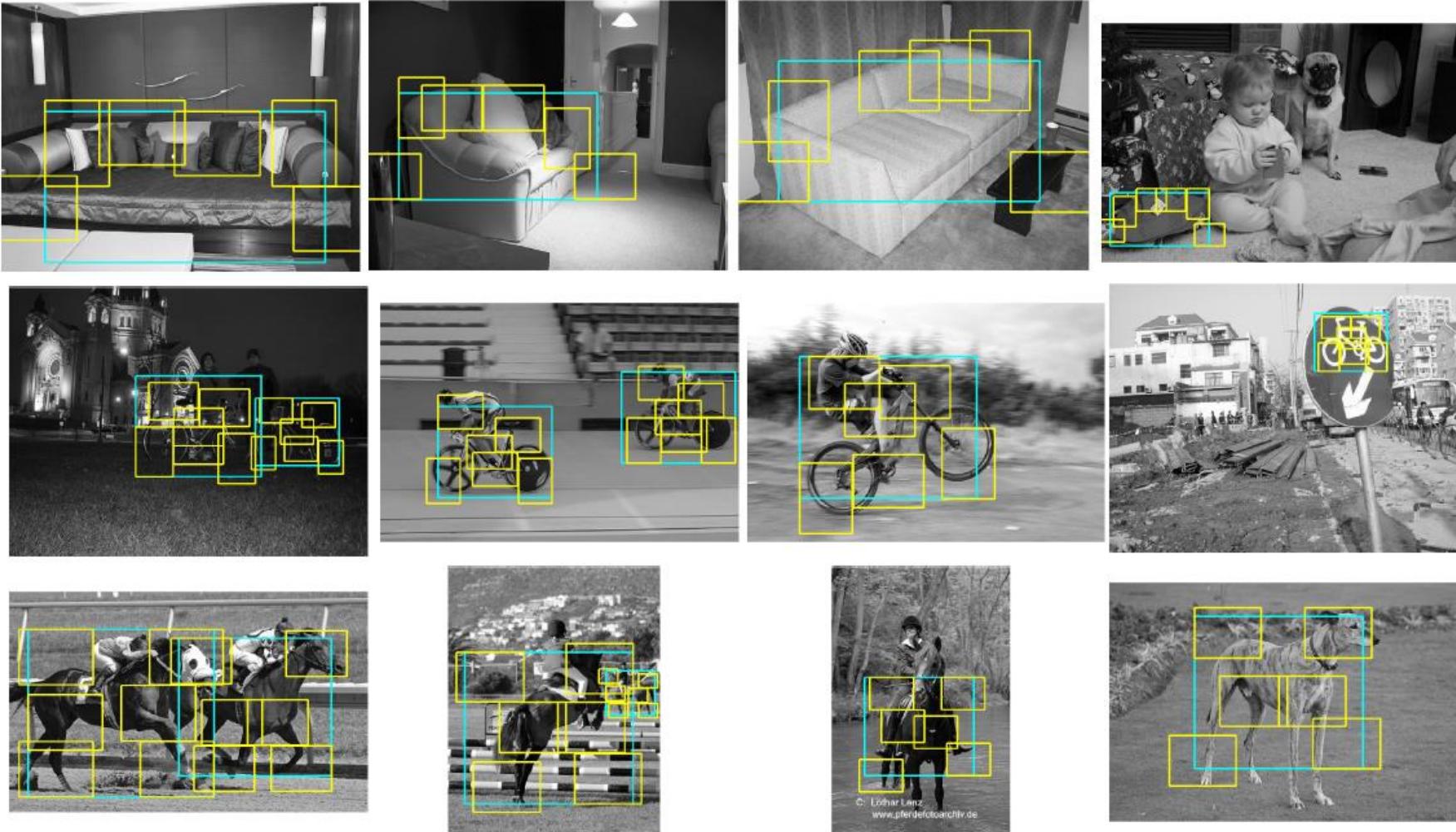


Learned Models





Sample Results



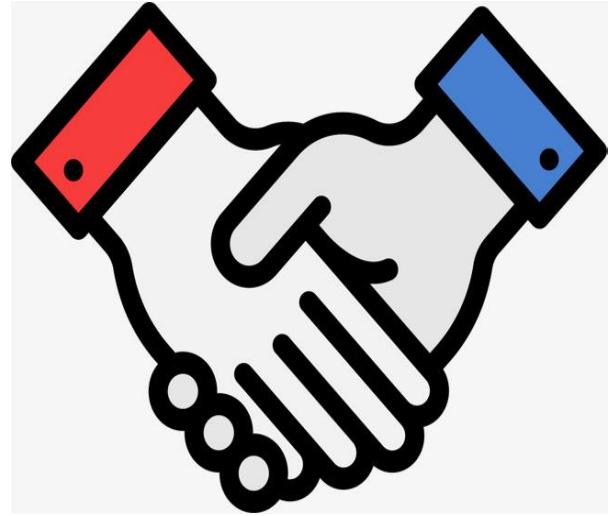
C: Lothar Lenz
www.pferdefotearchiv.de

Conclusions



Conclusion

- Summary: Viola/Jones detector
 - Rectangle features
 - Integral images for fast computation
 - Boosting for feature selection
 - Attentional cascade for fast rejection of negative windows
- Summary: Deformable Part Models (DPM)
 - Histogram of gradients
 - Partial models - Pictorial Structure
 - Latent (Structured) SVM



Thanks



zhengf@sustc.edu.cn