# Category-level localization

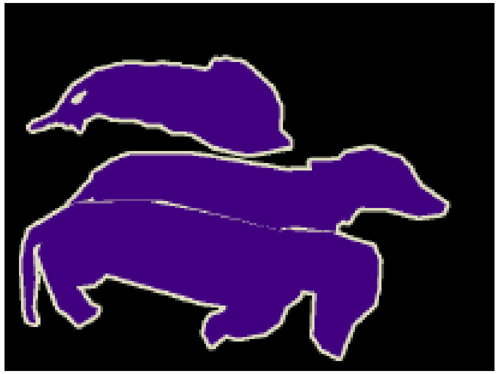
Cordelia Schmid

# Recognition

- Classification
  - Object present/absent in an image
  - Often presence of a significant amount of background clutter

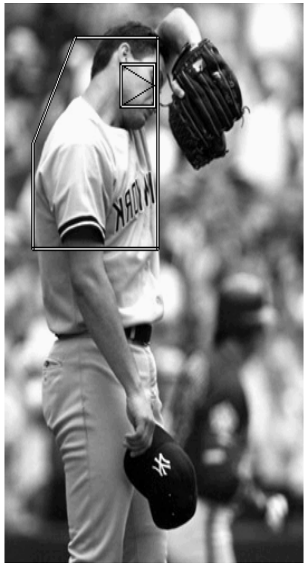
- Localization / Detection
  - Localize object within the frame
  - Bounding box or pixel-level segmentation

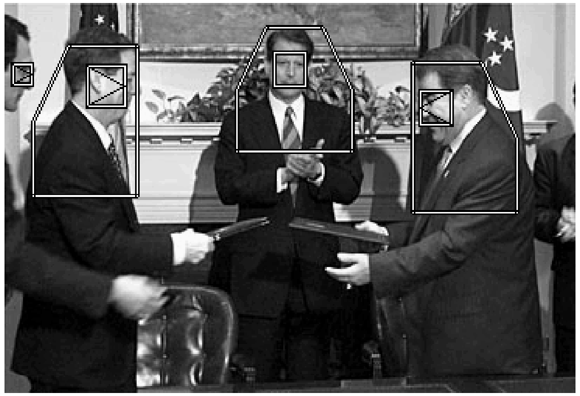
# Pixel-level object classification

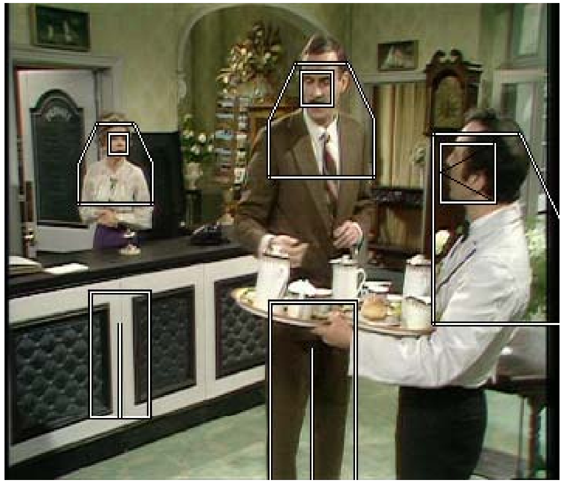



# Difficulties

- Intra-class variations







- Scale and viewpoint change
- Multiple aspects of categories

# Approaches

- Intra-class variation

  => Modeling of the variations, mainly by learning from a large dataset, for example by SVMs

- Scale + limited viewpoints changes

  => multi-scale approach

- Multiple aspects of categories

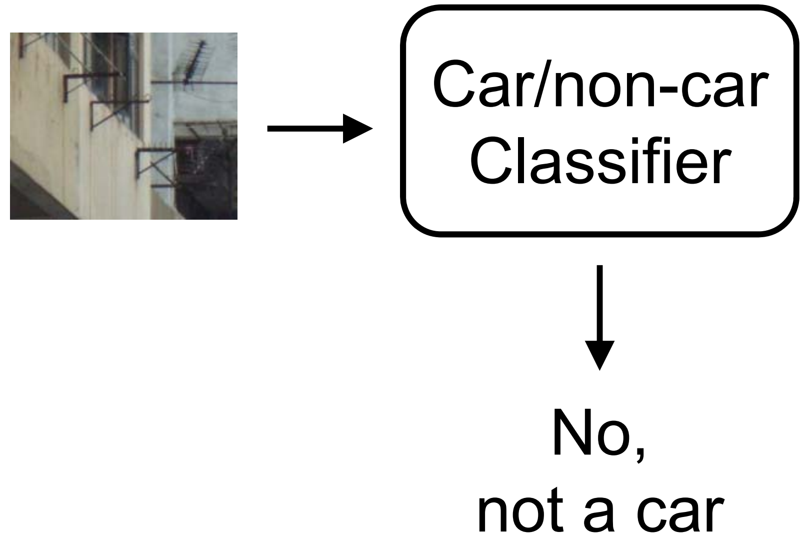
  => separate detectors for each aspect, front/profile face, build an approximate 3D “category” model

# Outline

1. *Sliding window detectors*
2. Features and adding spatial information
3. Histogram of Oriented Gradients (HOG)
4. State of the art algorithms and PASCAL VOC

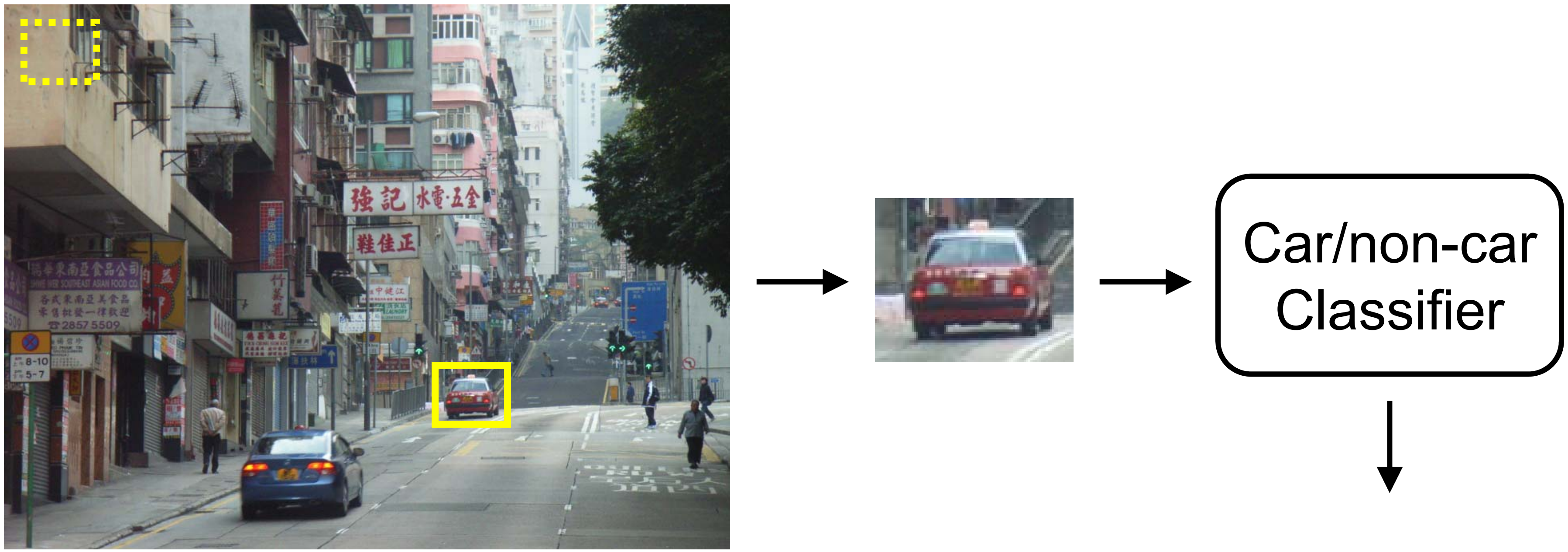
# Sliding window detector

- Basic component: binary classifier



# Sliding window detector

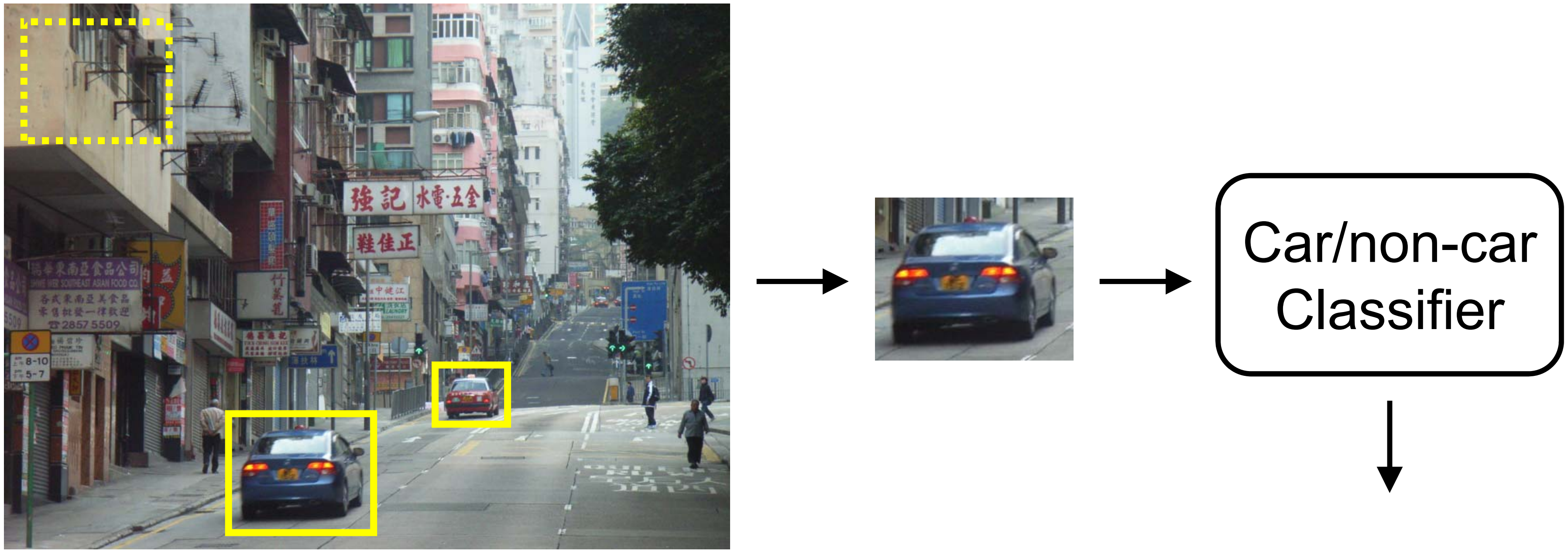
- Detect objects in clutter by **search**




- **Sliding window**: exhaustive search over position and scale

# Sliding window detector

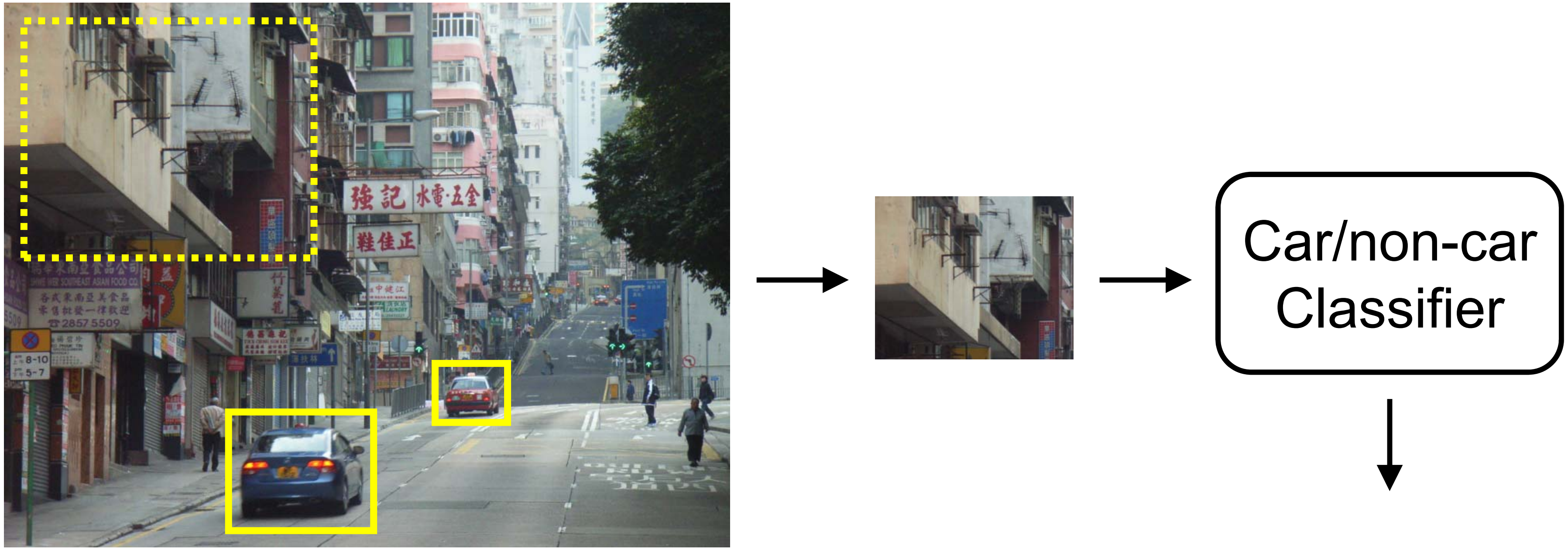
- Detect objects in clutter by **search**




- **Sliding window**: exhaustive search over position and scale

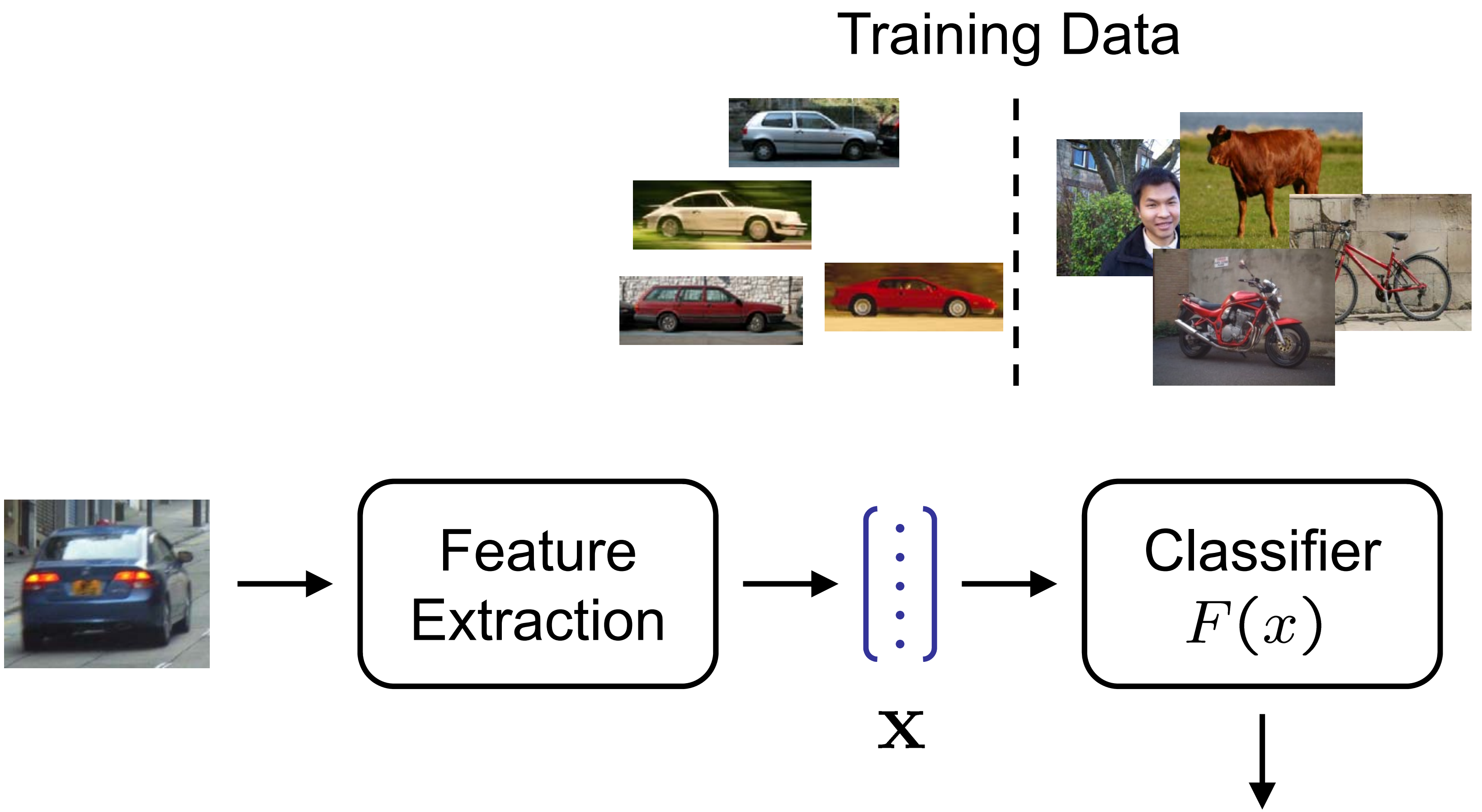
# Detection by Classification

• Detect objects in clutter by **search**




• **Sliding window**: exhaustive search over position and scale
(can use same size window over a spatial pyramid of images)

# Window (Image) Classification



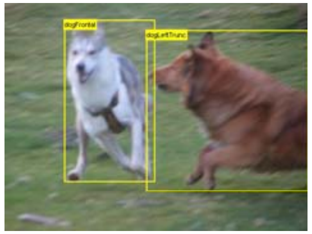

Car/Non-car

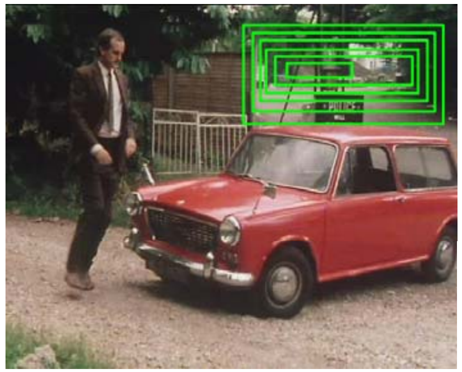
$$P(c|\mathbf{x}) \propto F(\mathbf{x})$$

- Features usually engineered
- Classifier learnt from data

# Problems with sliding windows …

- aspect ratio
- granularity (finite grid)
- partial occlusion
- multiple responses



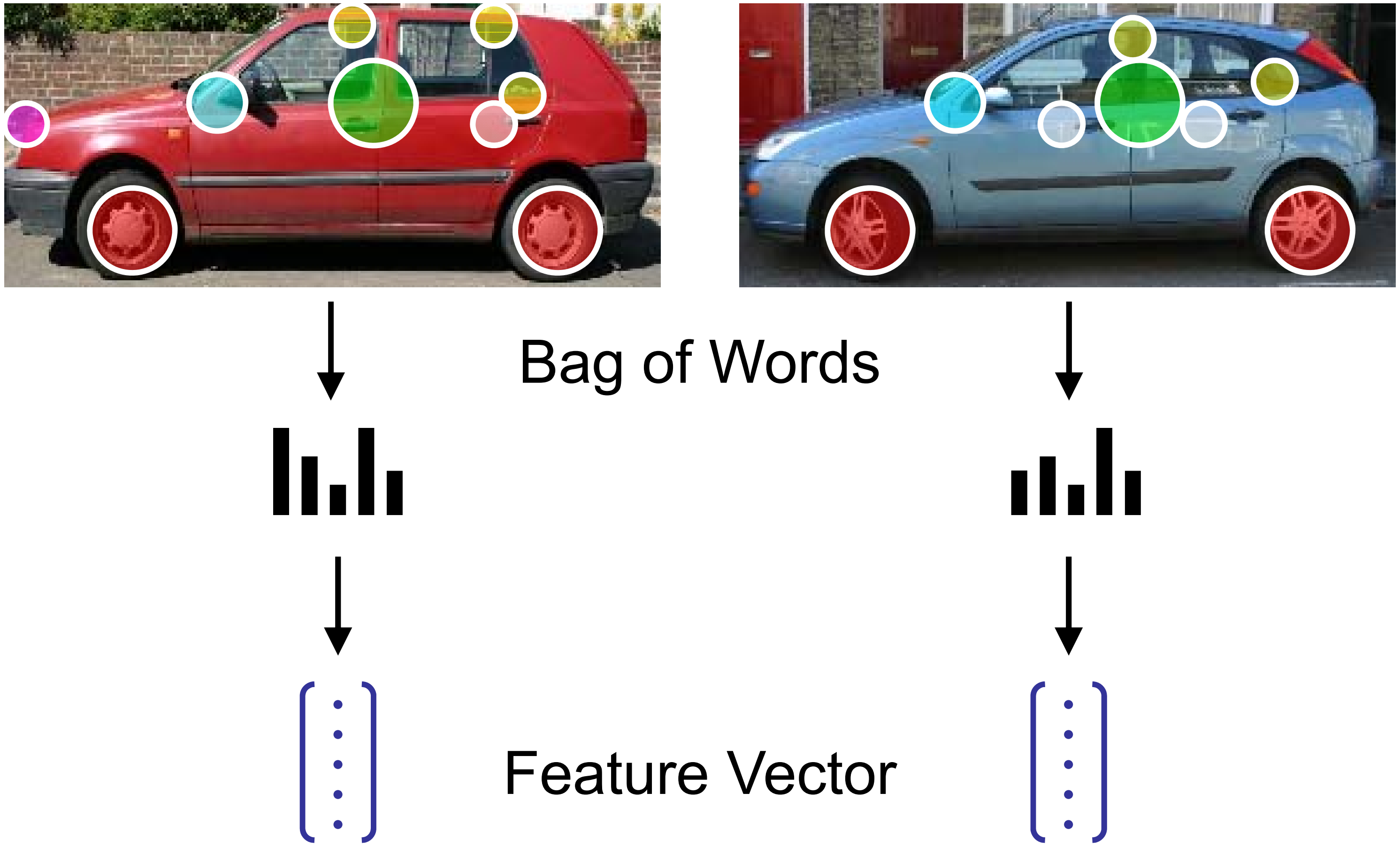
# Outline

1. Sliding window detectors
2. *Features and adding spatial information*
3. Histogram of Oriented Gradients (HOG)
4. State of the art algorithms and PASCAL VOC

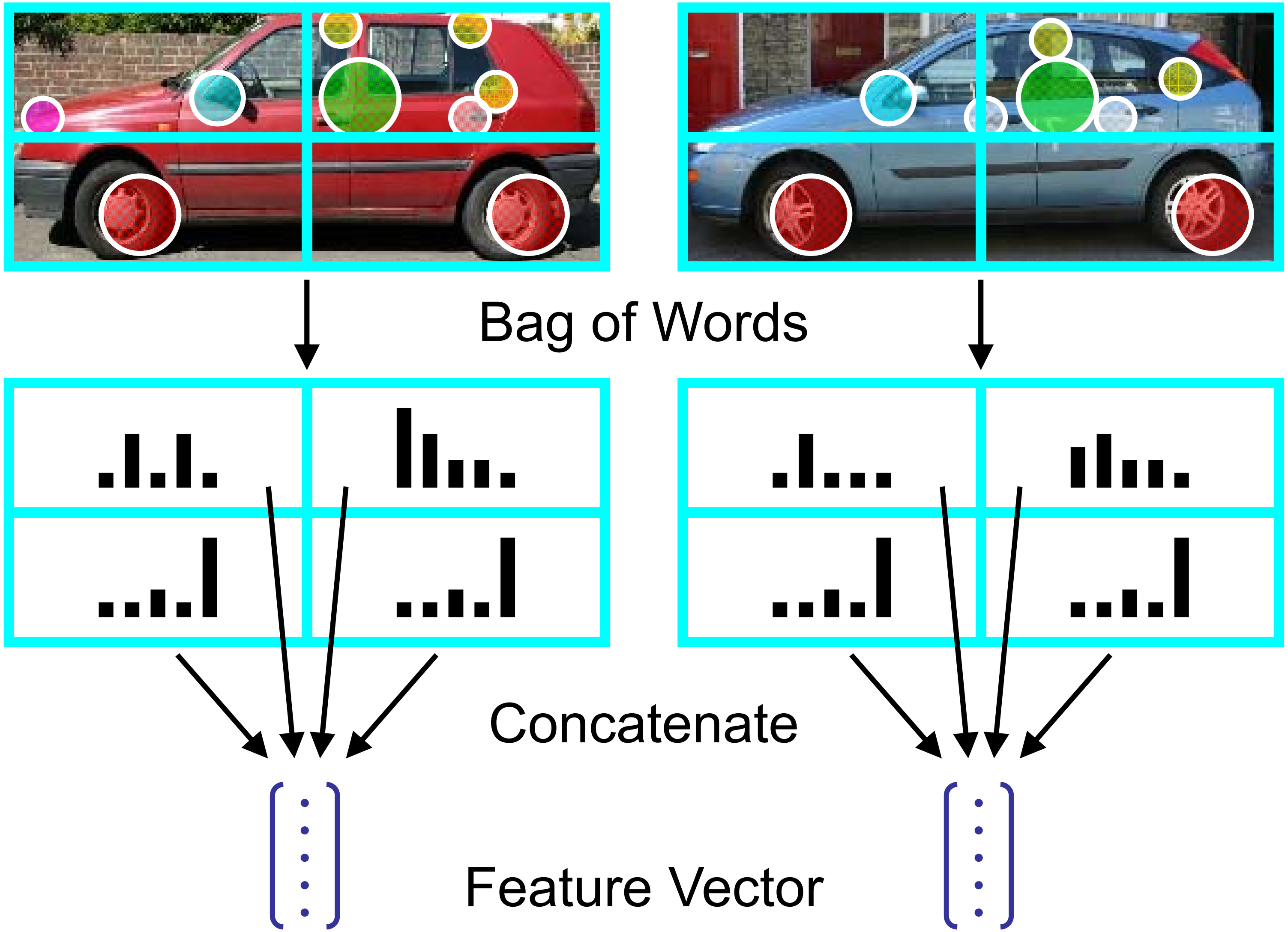
# BOW + Spatial pyramids

Start from BoW for region of interest (ROI)

- no spatial information recorded
- sliding window detector

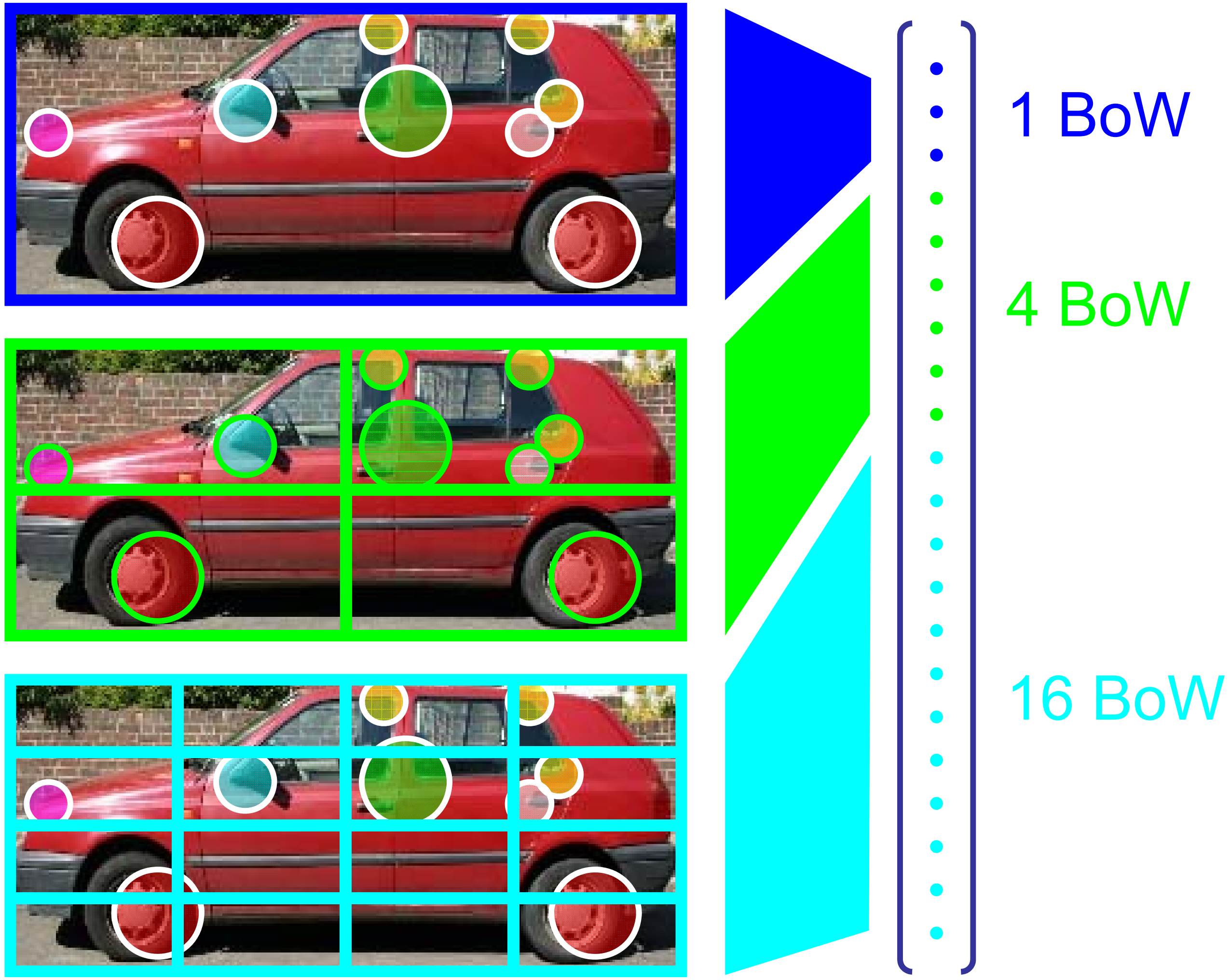


# Adding Spatial Information to Bag of Words




Keeps fixed length feature vector for a window

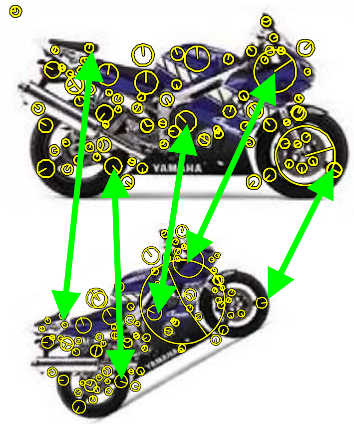
# Spatial Pyramid – represent correspondence



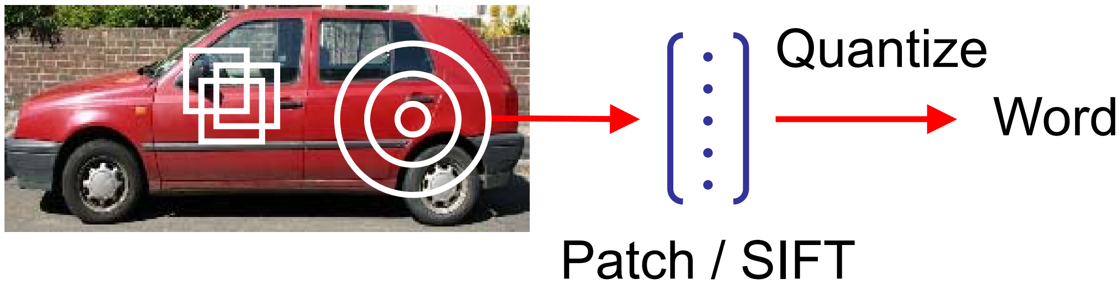
# Dense Visual Words

- Why extract only **sparse** image fragments?

- Good where lots of invariance is needed, but not relevant to sliding window detection?



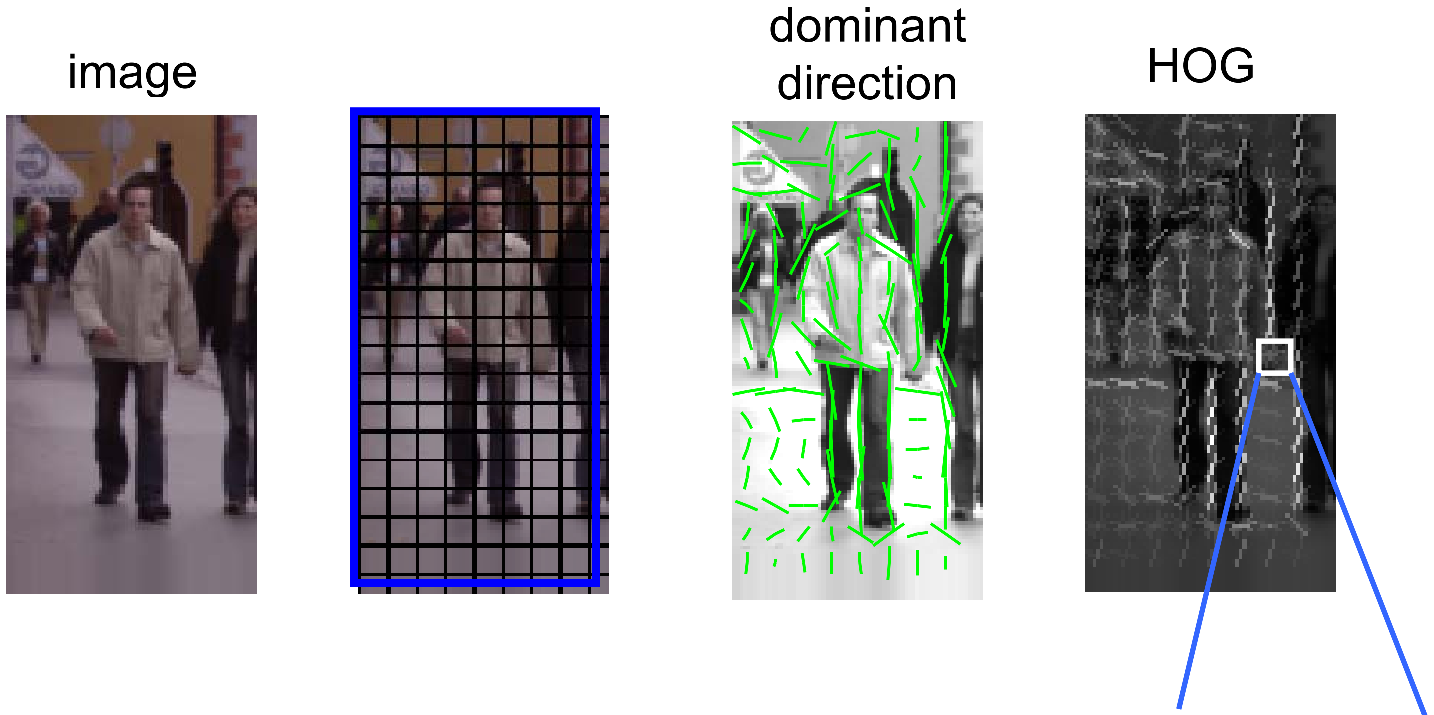
- Extract **dense** visual words on an overlapping grid

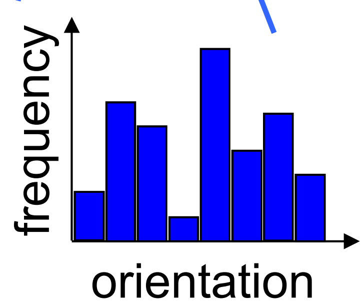


# Outline

1. Sliding window detectors
2. Features and adding spatial information
3. *Histogram of Oriented Gradients + linear SVM classifier*
4. State of the art algorithms and PASCAL VOC

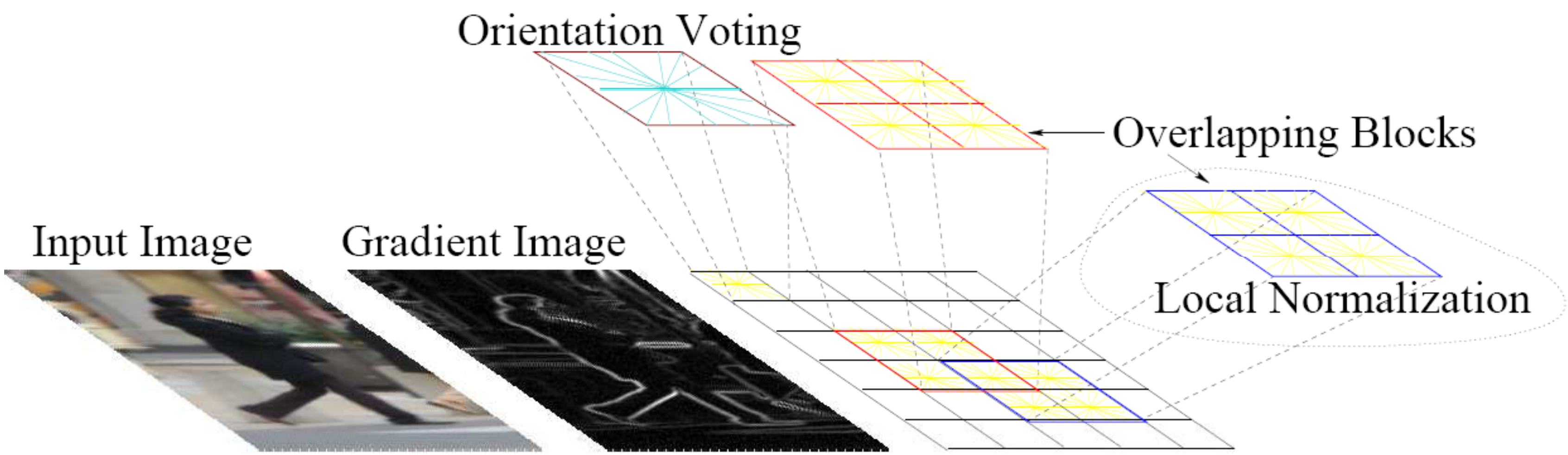
# Feature: Histogram of Oriented Gradients (HOG)



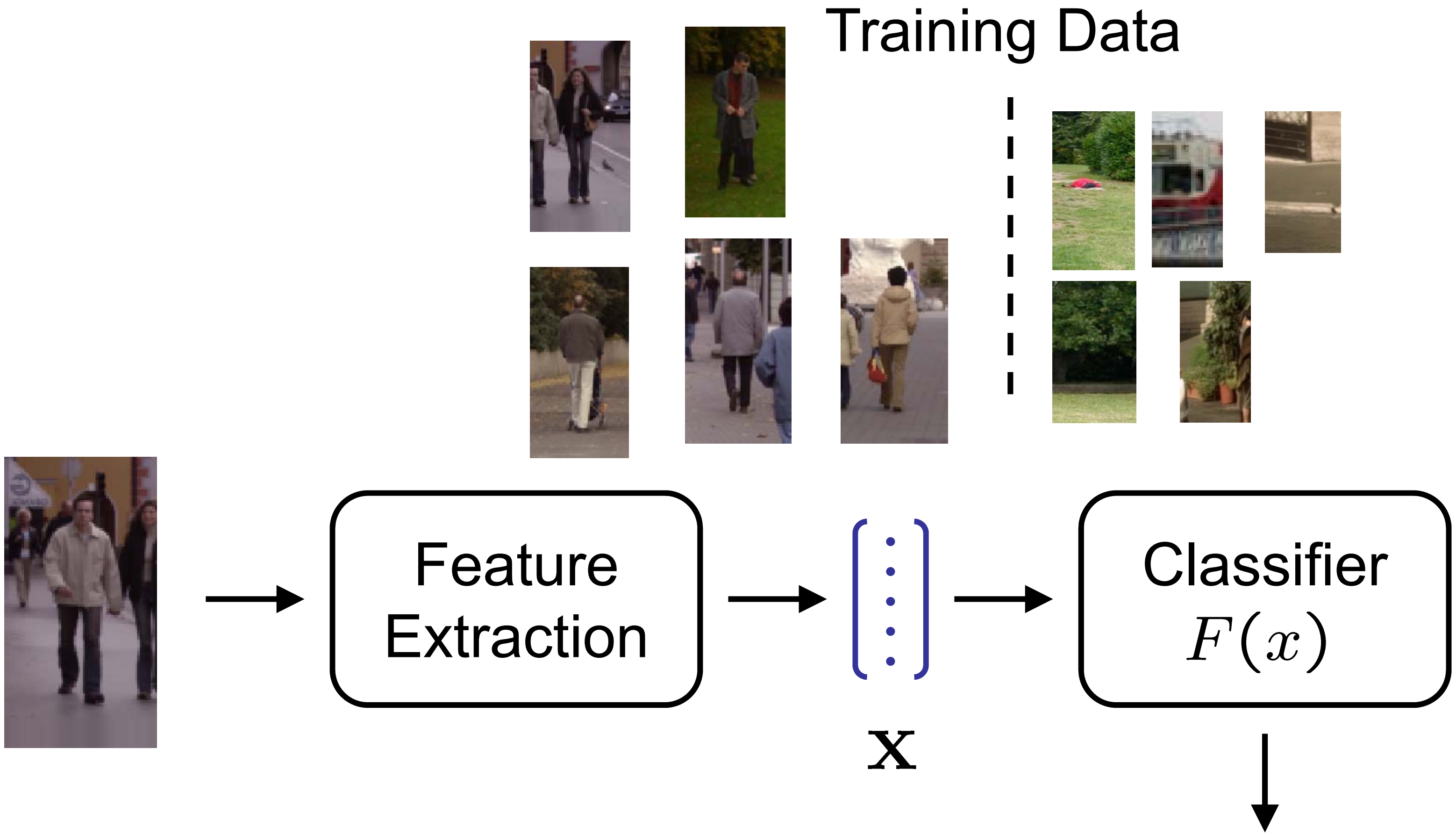

- tile 64 x 128 pixel window into 8 x 8 pixel cells
- each cell represented by histogram over 8 orientation bins (i.e. angles in range 0-180 degrees)

# Histogram of Oriented Gradients (HOG) continued




- Adds a second level of overlapping spatial bins re-normalizing orientation histograms over a larger spatial area
- Feature vector dimension (approx) = 16 x 8 (for tiling) x 8 (orientations) x 4 (for blocks) = 4096

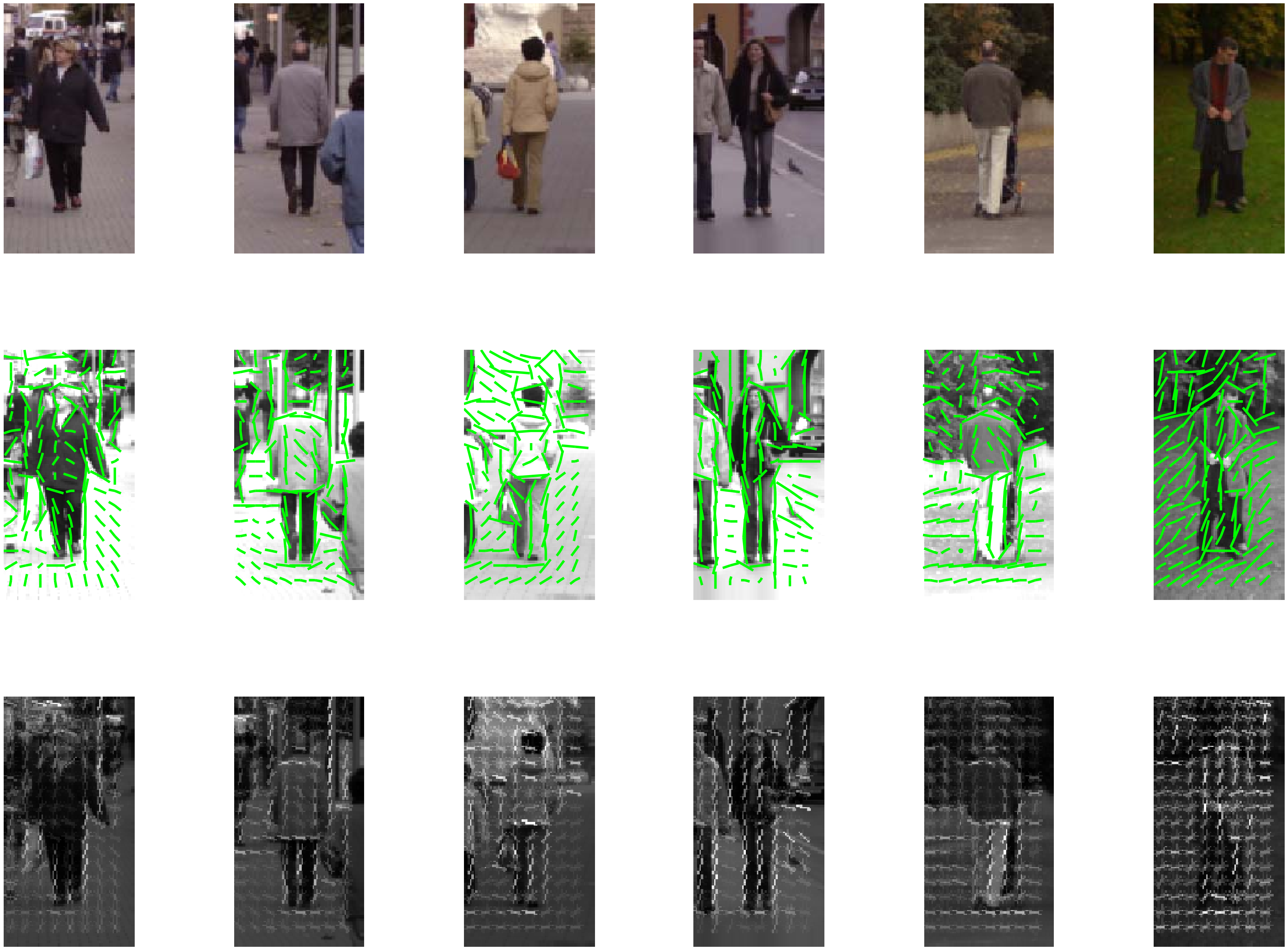
# Window (Image) Classification



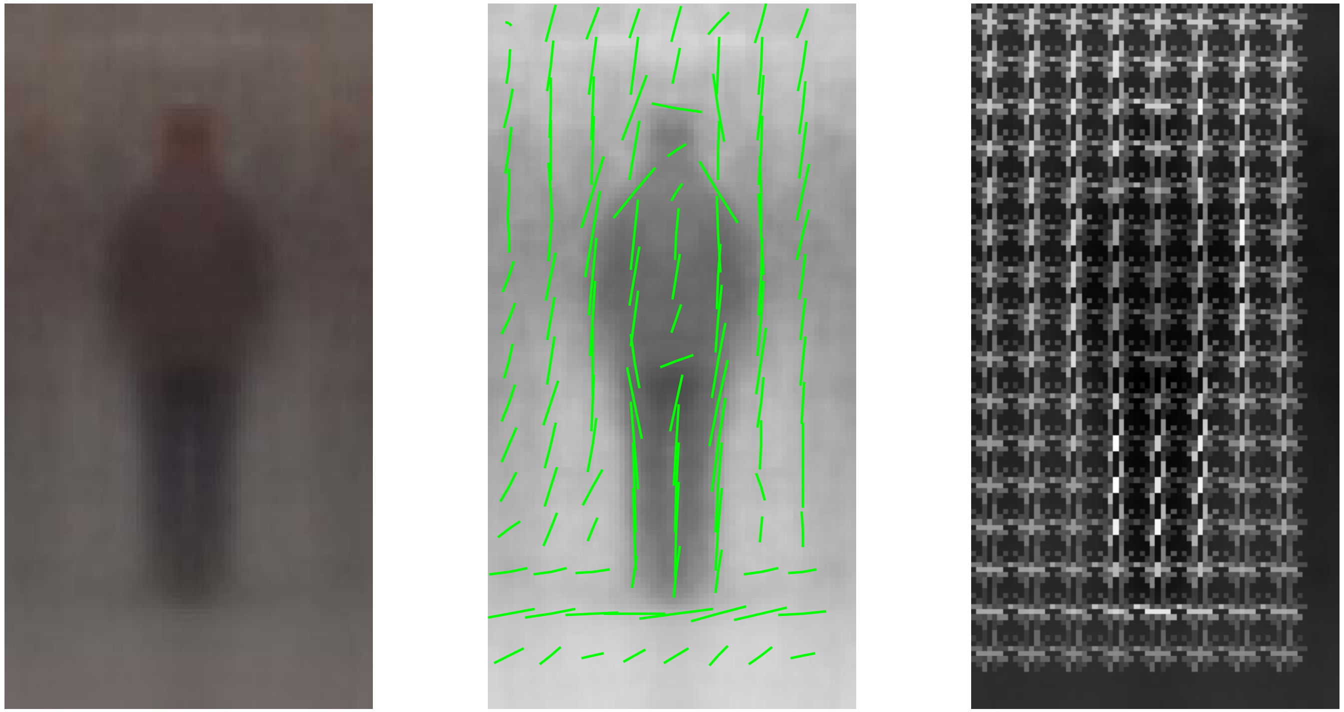

pedestrian/Non-pedestrian

$$P(c|\mathbf{x}) \propto F(\mathbf{x})$$

- HOG Features
- Linear SVM classifier



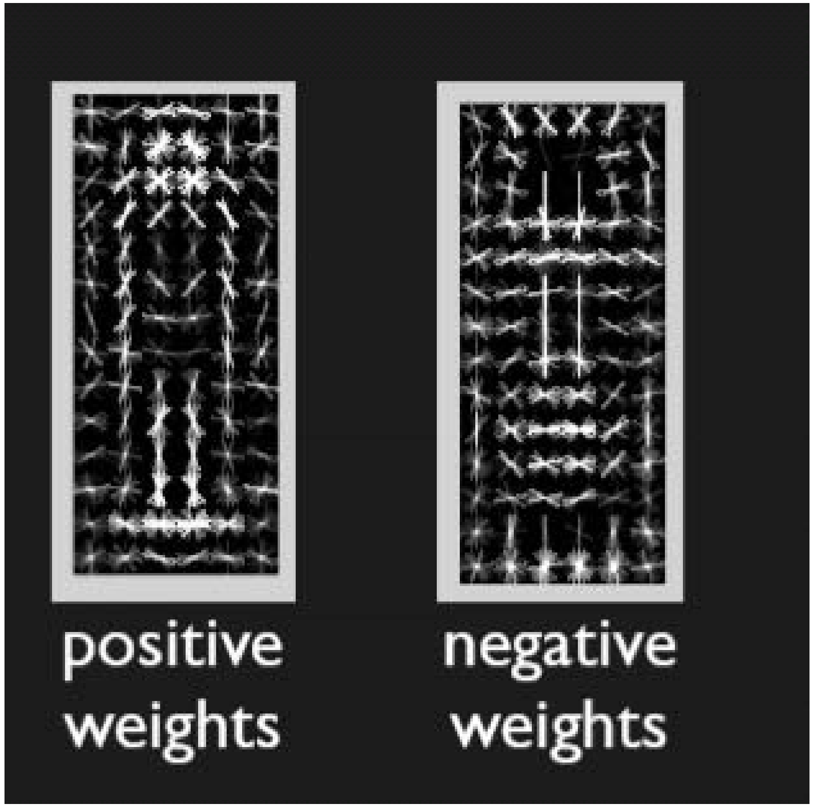
# Averaged examples

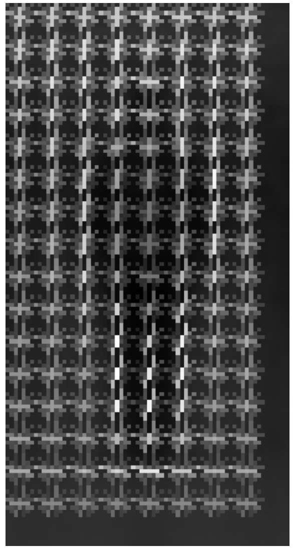



Dalal and Triggs, CVPR 2005

# Learned model

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$



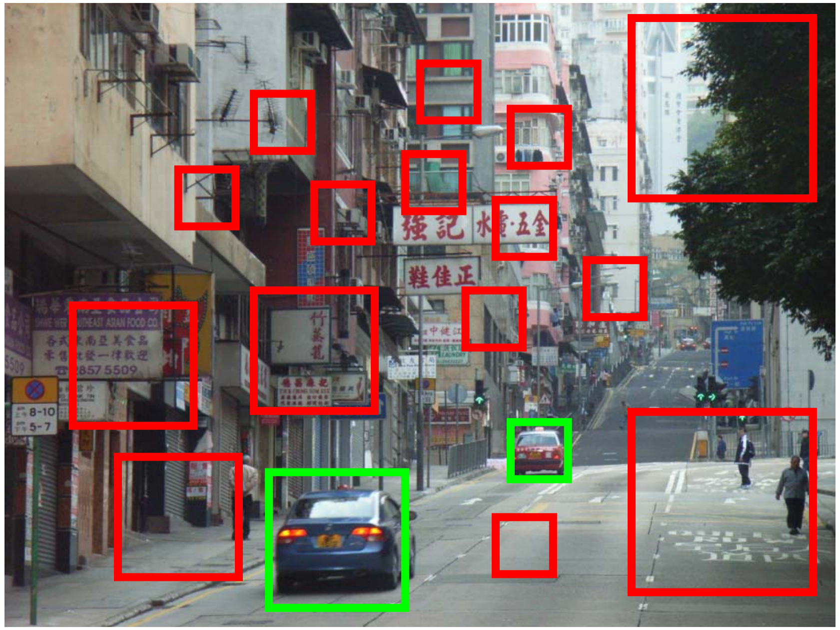
average over
positive training data

# Training a sliding window detector

- Unlike training an image classifier, there are a (virtually) infinite number of possible negative windows
- Training (learning) generally proceeds in three distinct stages:
  1. Bootstrapping: learn an initial window classifier from positives and random negatives
  2. Hard negatives: use the initial window classifier for detection on the training images (inference) and identify false positives with a high score
  3. Retraining: use the hard negatives as additional training data

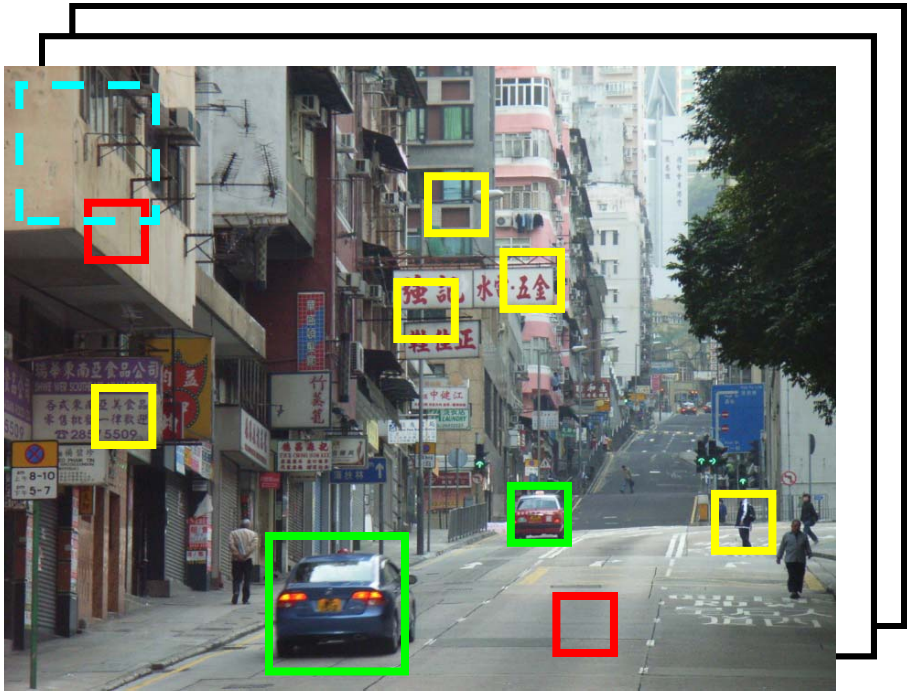
# Training a sliding window detector

- Object **detection** is inherently asymmetric: much more "non-object" than "object" data



- Classifier needs to have very low false positive rate
- Non-object category is very complex – need lots of data

# Bootstrapping



1. Pick negative training set at random
2. Train classifier
3. Run on training data
4. Add false positives to training set
5. Repeat from 2

- Collect a finite but diverse set of non-object windows
- Force classifier to concentrate on **hard negative** examples
- For some classifiers can ensure equivalence to training on entire data set

# Example: train an upper body detector

- Training data – used for training and validation sets
  - 33 Hollywood2 training movies
  - 1122 frames with upper bodies marked

- First stage training (bootstrapping)
  - 1607 upper body annotations jittered to 32k positive samples
  - 55k negatives sampled from the same set of frames

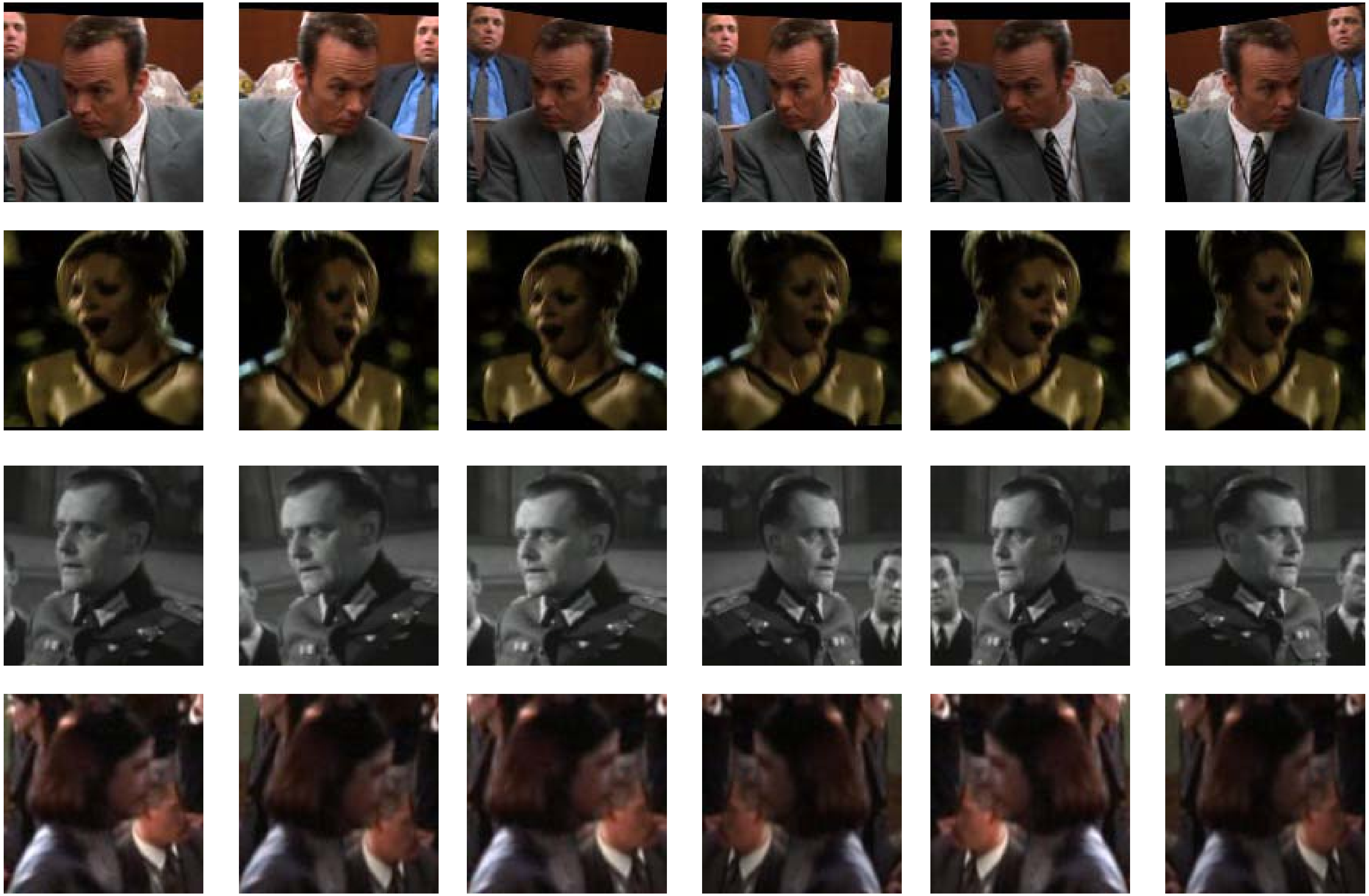
- Second stage training (retraining)
  - 150k hard negatives found in the training data

# Training data – positive annotations

# Positive windows

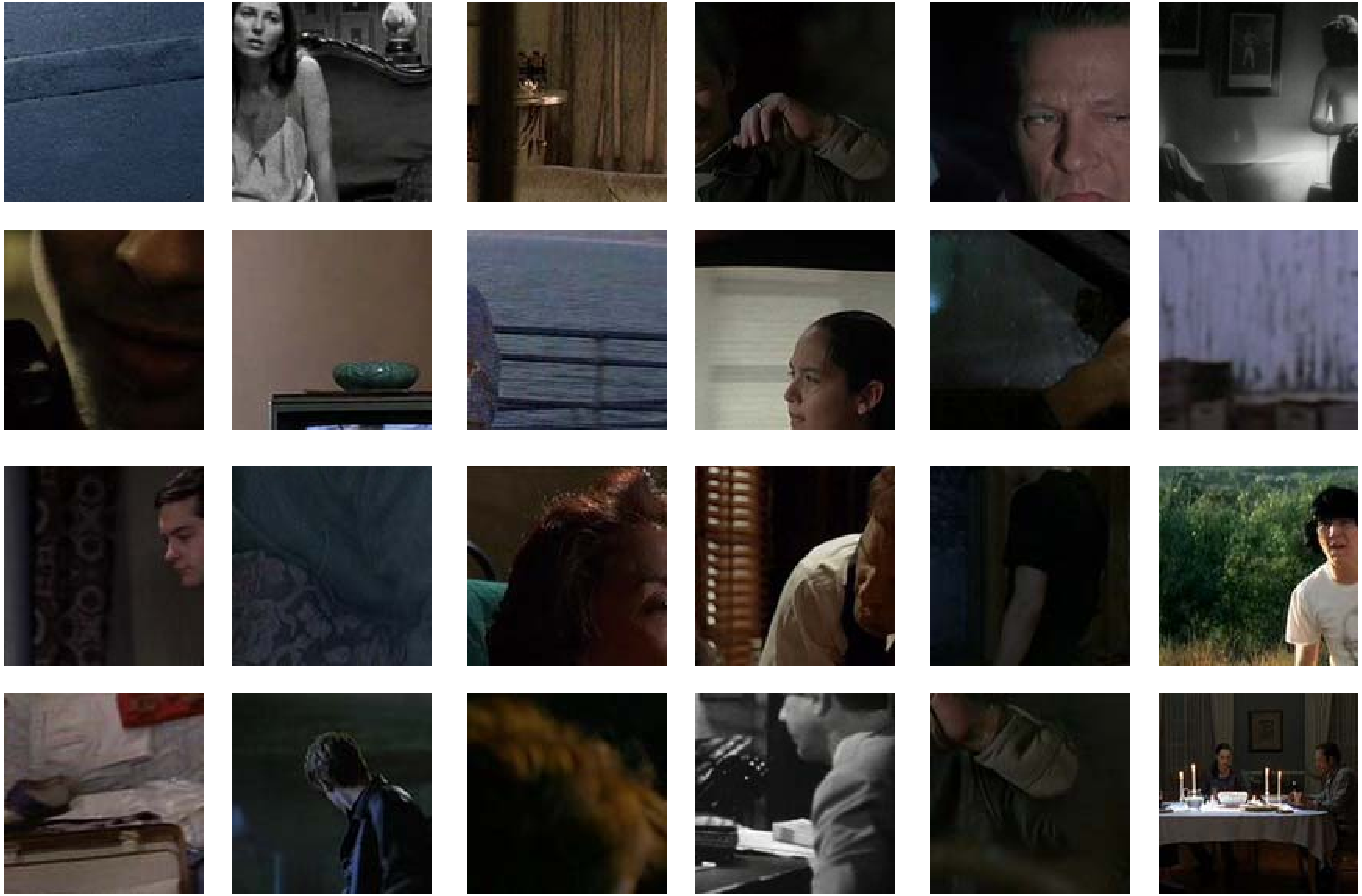

Note: common size and alignment

# Jittered positives



# Jittered positives

# Random negatives



# Random negatives

# Window (Image) first stage classification

Jittered positives

random negatives

HOG Feature Extraction

**x**

Linear SVM Classifier

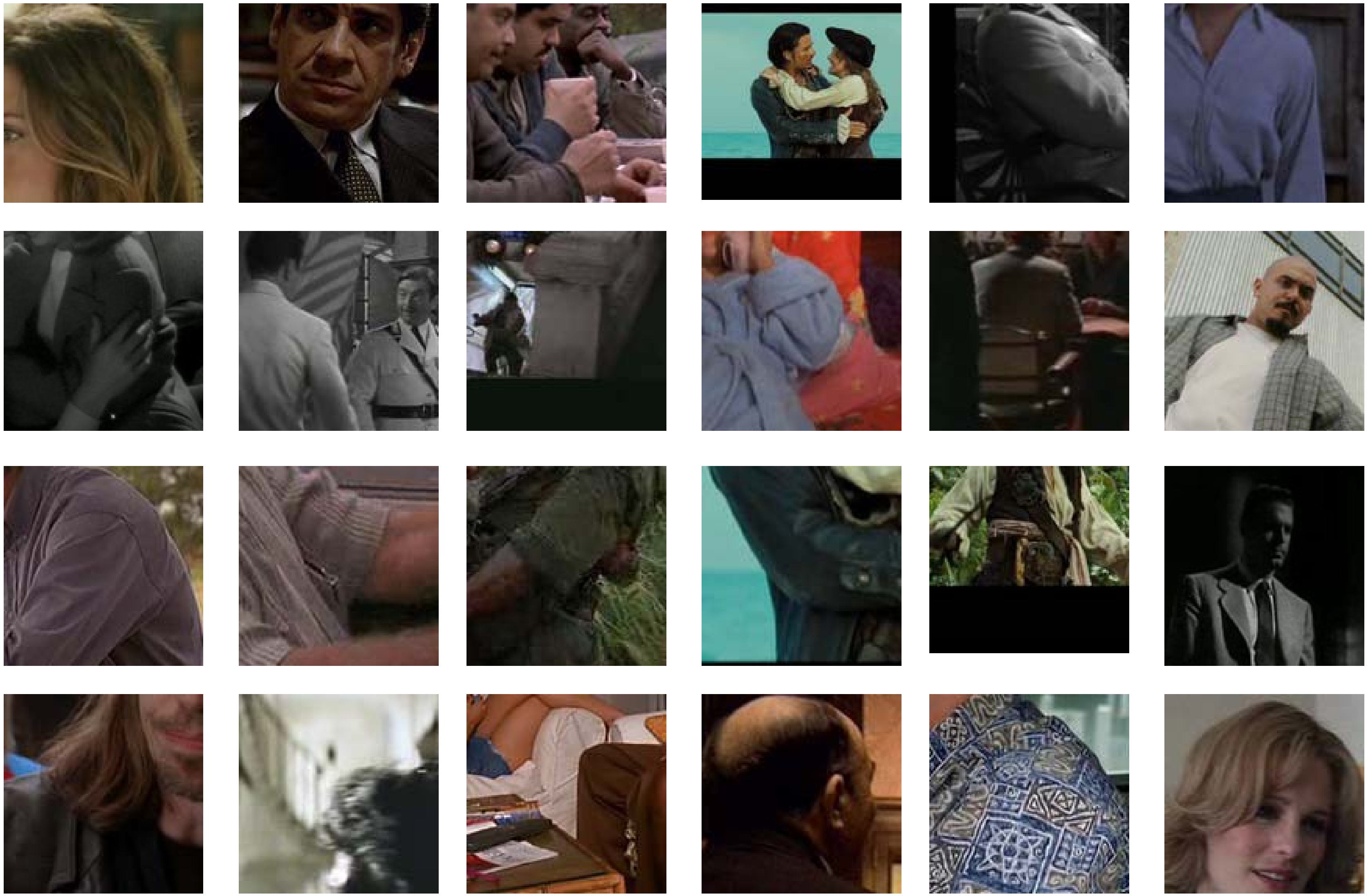
$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- find high scoring false positives detections

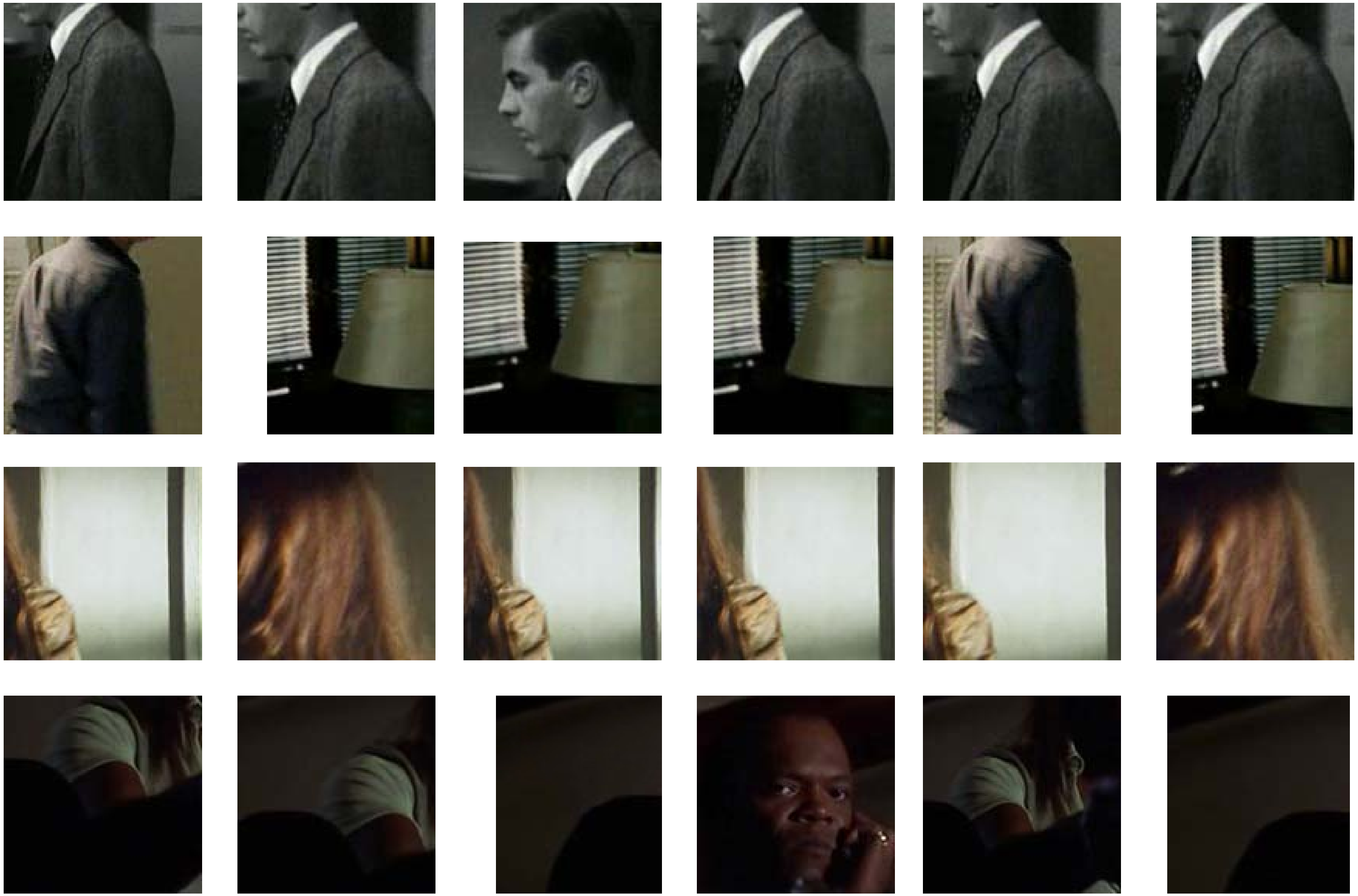
- these are the hard negatives for the next round of training

- cost = # training images x inference on each image

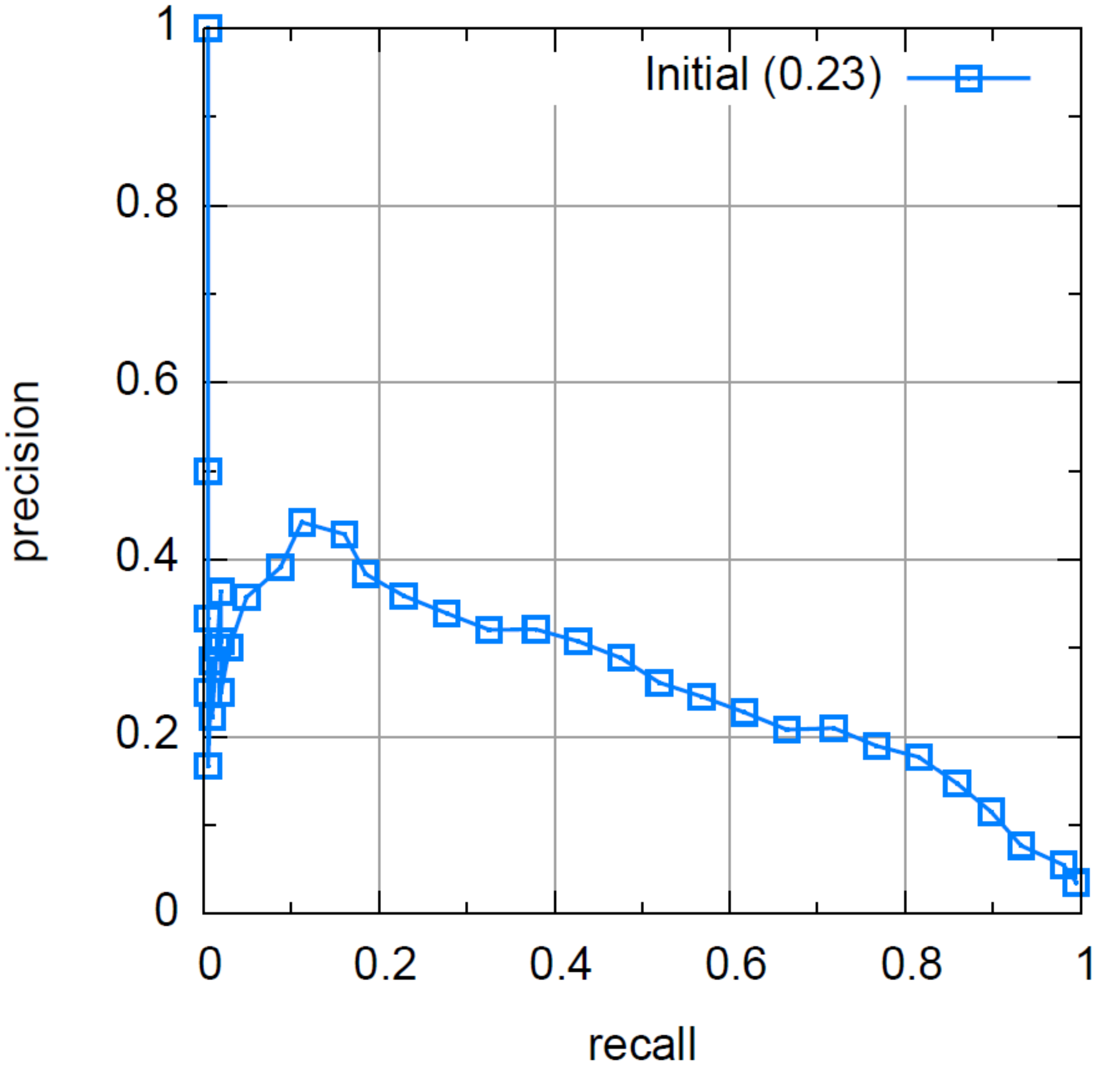
# Hard negatives



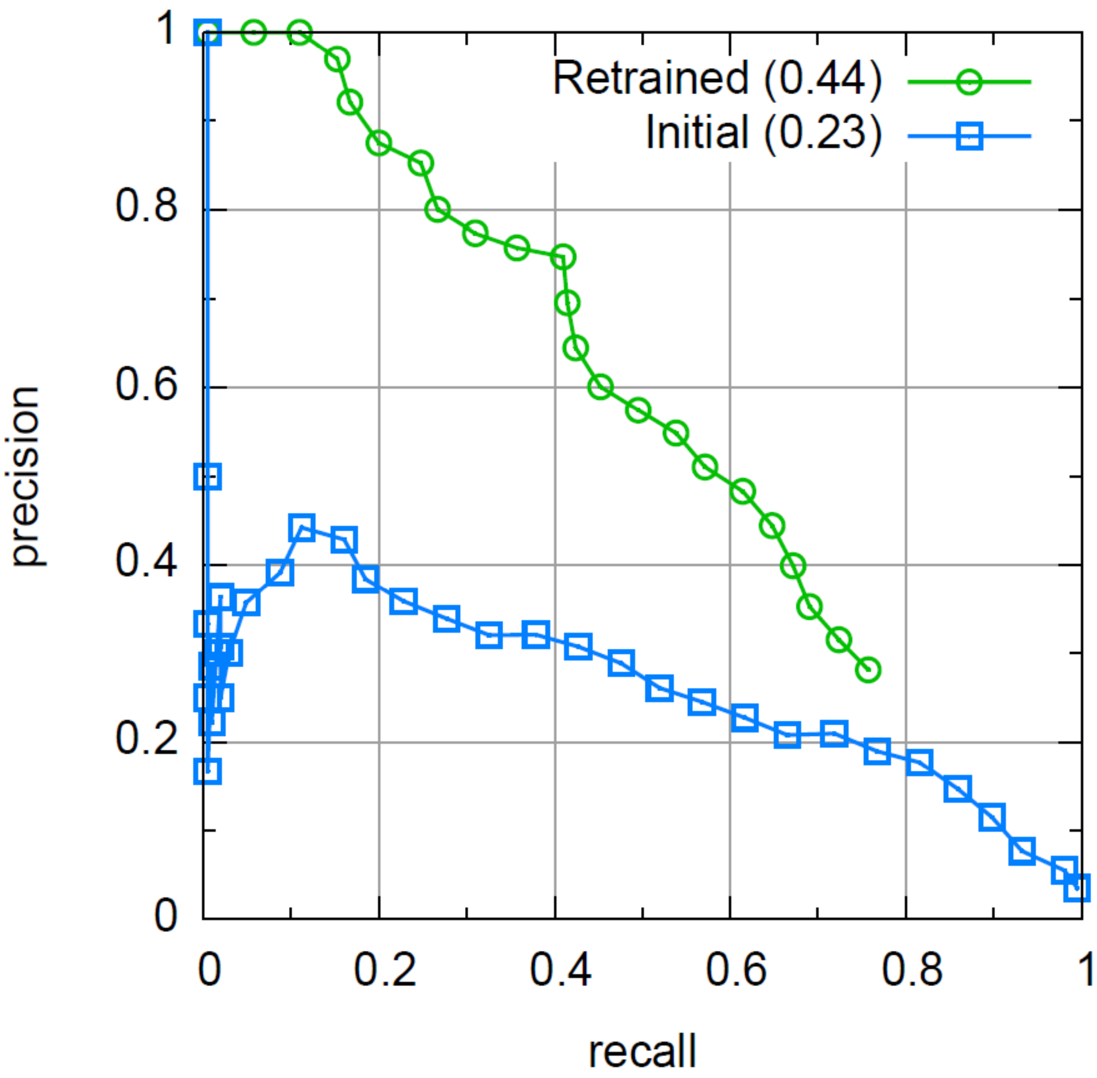
# Hard negatives



# First stage performance on validation set

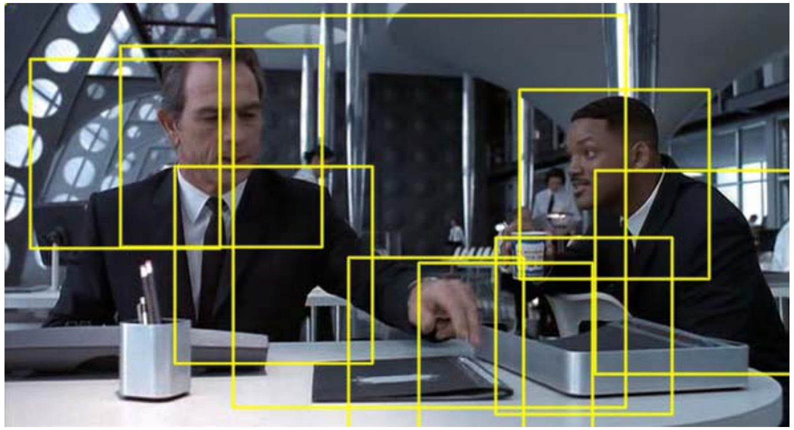


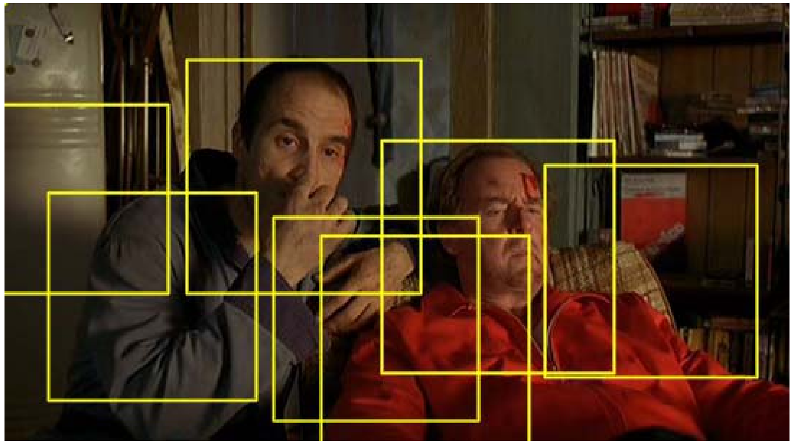
# Effects of retraining



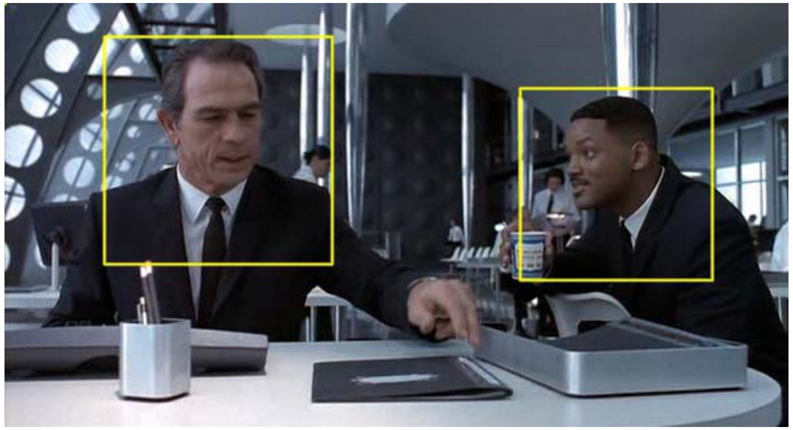
# Side by side

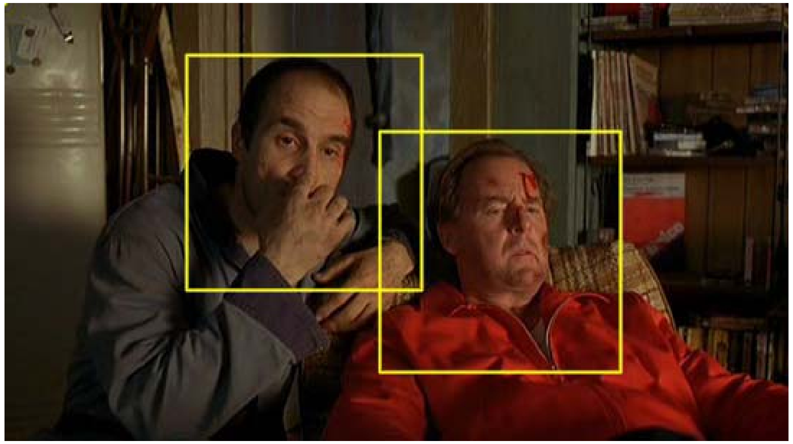
before retraining





after retraining

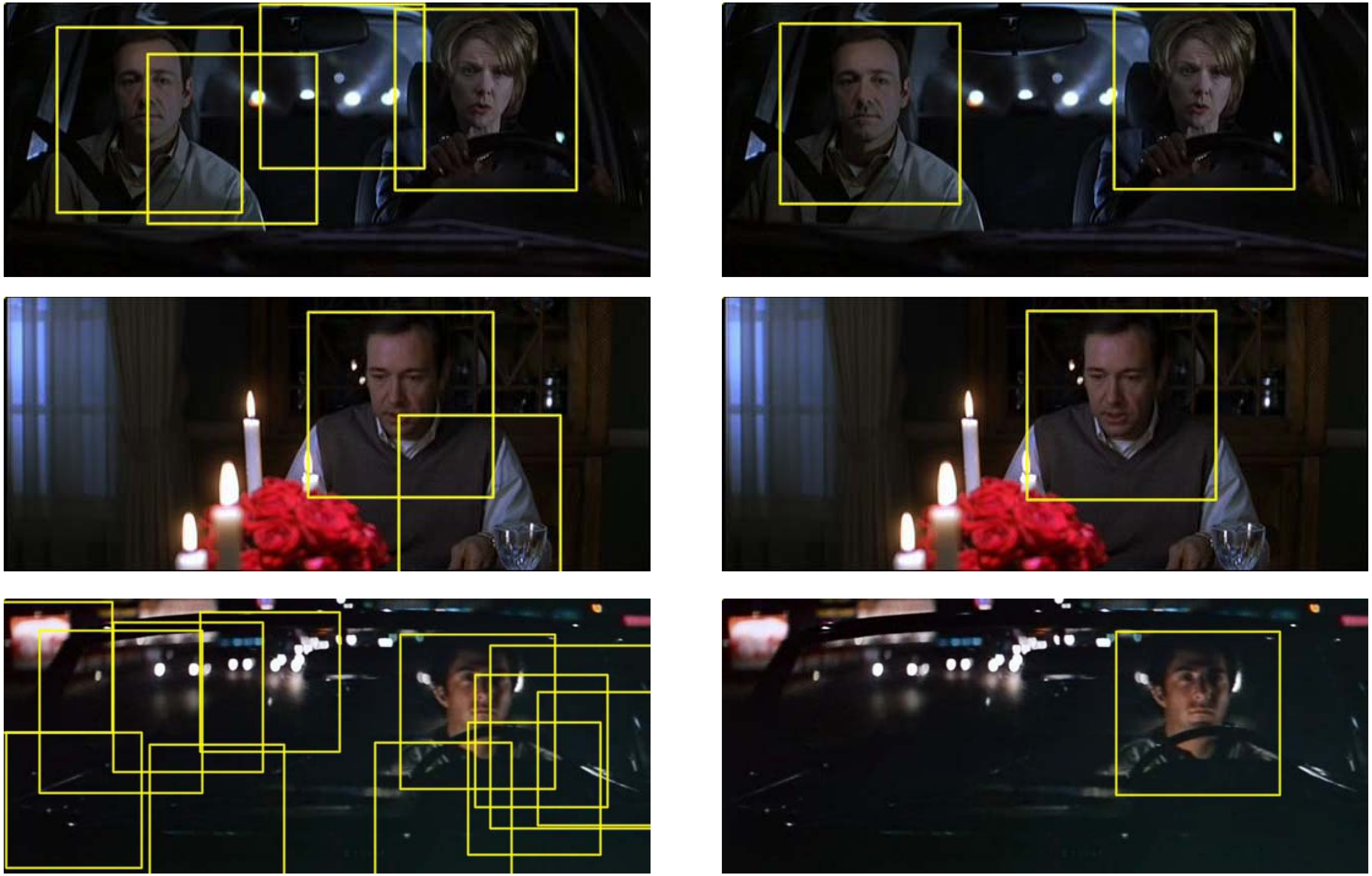




# Side by side

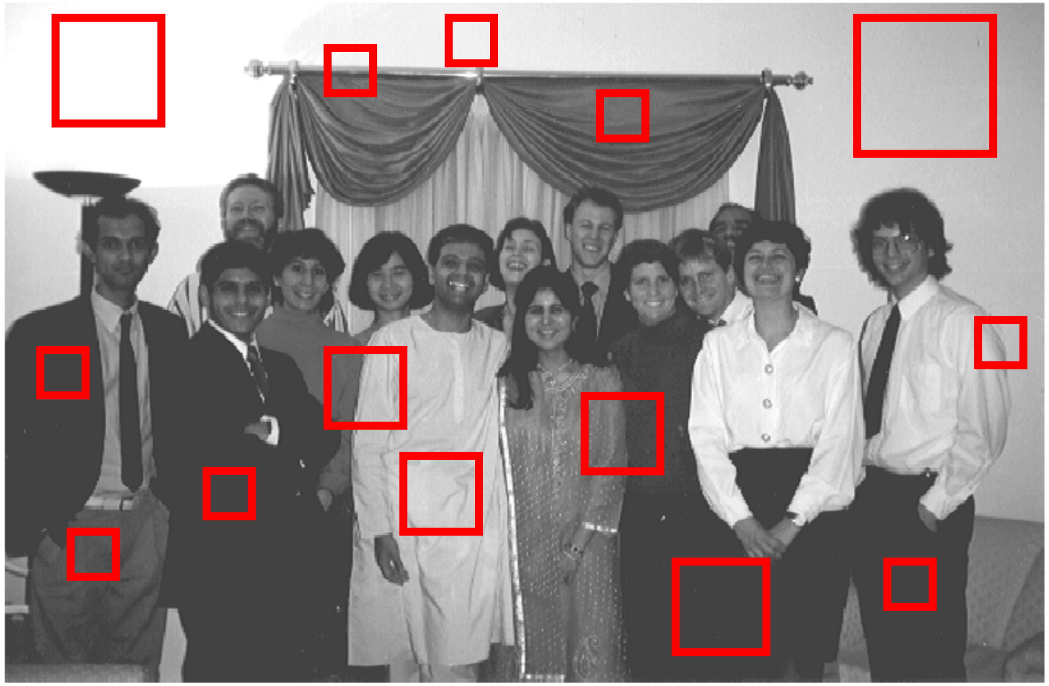
before retraining

after retraining



# Accelerating Sliding Window Search

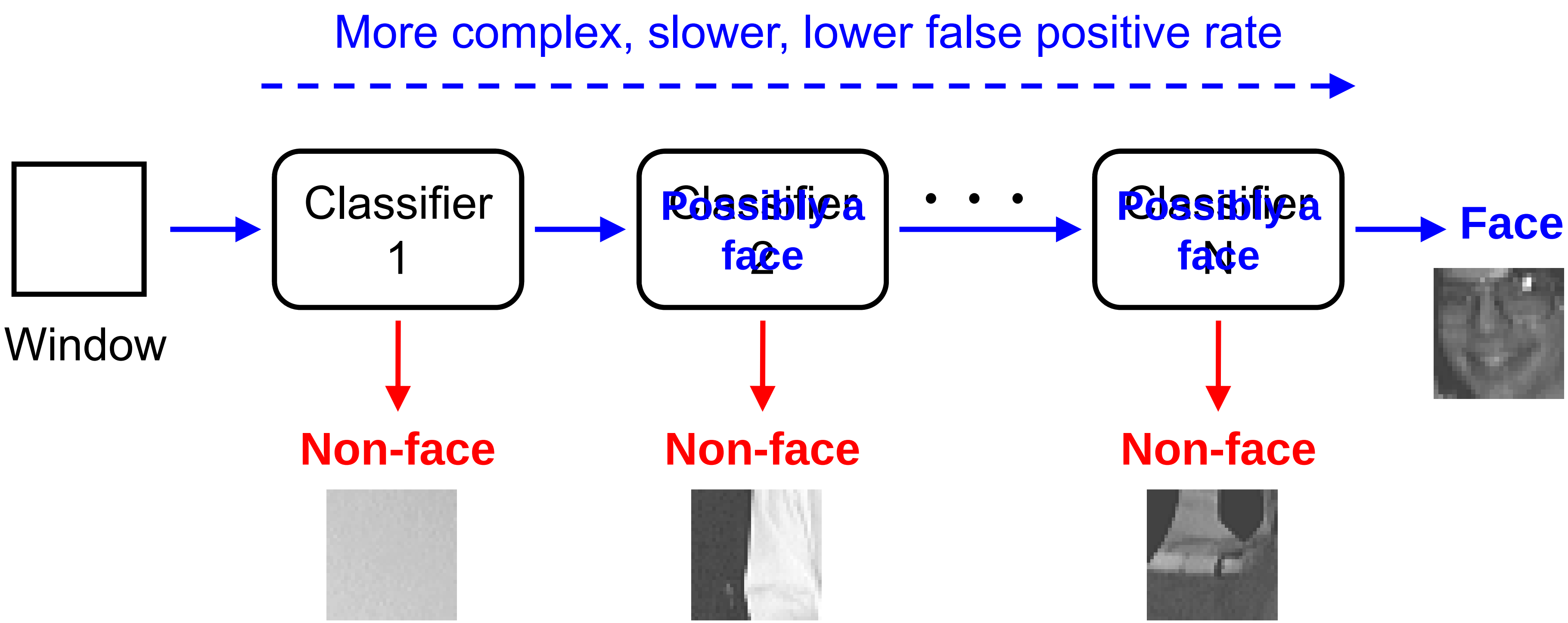
- Sliding window search is slow because so many windows are needed e.g. x × y × scale ≈ 100,000 for a 320×240 image



- Most windows are clearly not the object class of interest
- Can we speed up the search?

# Cascaded Classification

• Build a sequence of classifiers with increasing complexity




• Reject easy non-objects using simpler and faster classifiers

# Cascaded Classification

- Slow expensive classifiers only applied to a few windows ➔ significant speed-up

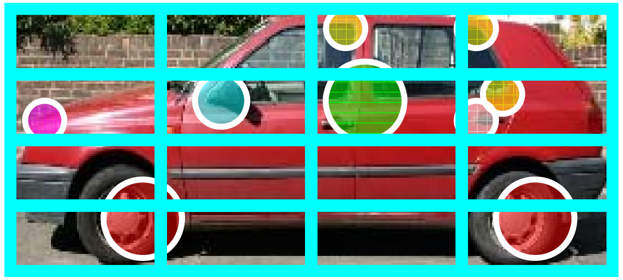
- Controlling classifier complexity/speed:
  - Number of support vectors [Romdhani et al, 2001]
  - Number of features [Viola & Jones, 2001]
  - Two-layer approach [Harzallah et al, 2009]

# Summary: Sliding Window Detection

- Can convert any image classifier into an object detector by sliding window. Efficient search methods available.

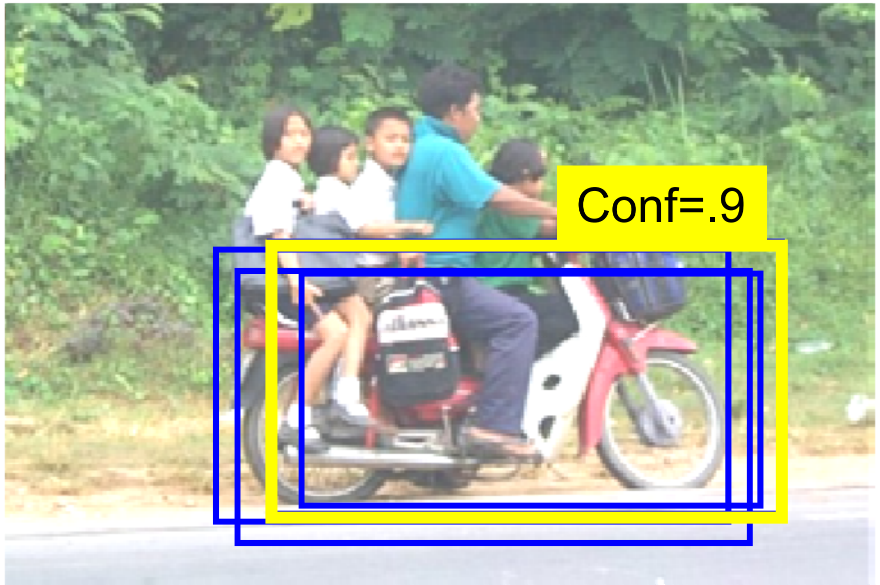
- Requirements for invariance are reduced by searching over e.g. translation and scale

- Spatial correspondence can be "engineered in" by spatial tiling



# Test: Non-maximum suppression (NMS)

- Scanning-window detectors typically result in multiple responses for the same object




- To remove multiple responses, a simple greedy procedure called "Non-maximum suppression" is applied:

NMS:
1. Sort all detections by detector confidence
2. Choose most confident detection $d_i$; remove all $d_j$ s.t. $overlap(d_i,d_j)>T$
3. Repeat Step 2. until convergence

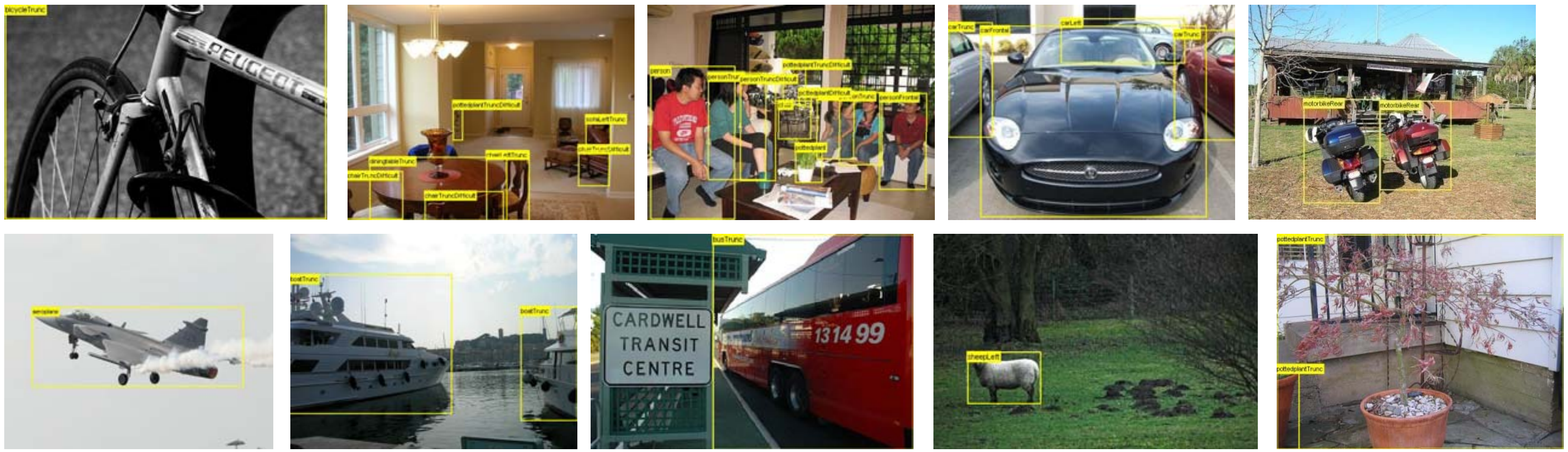
# Outline

1. Sliding window detectors
2. Features and adding spatial information
3. HOG + linear SVM classifier
4. *PASCAL VOC and state of the art algorithms*

# PASCAL VOC dataset - Content

- 20 classes: aeroplane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, train, TV

- Real images downloaded from flickr, not filtered for “quality”



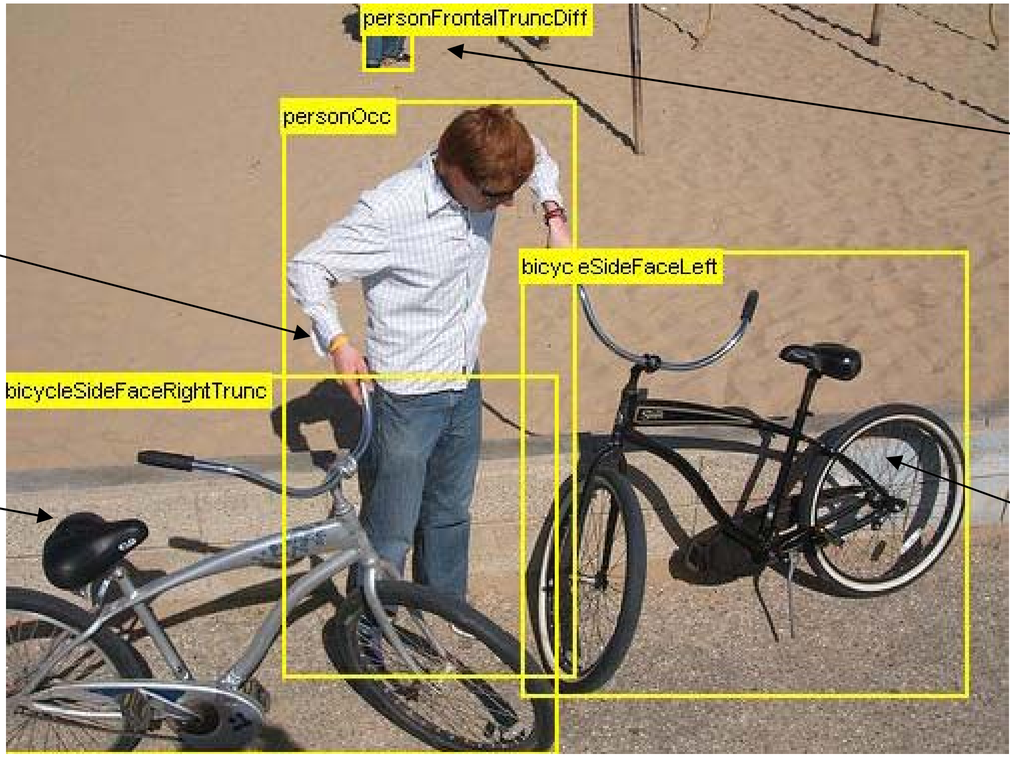

- Complex scenes, scale, pose, lighting, occlusion, ...

# Annotation

- Complete annotation of all objects

**Occluded**
Object is significantly occluded within BB

**Truncated**
Object extends beyond BB




**Difficult**
Not scored in evaluation

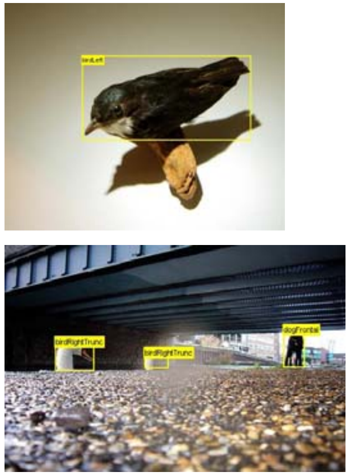
**Pose**
Facing left

# Examples

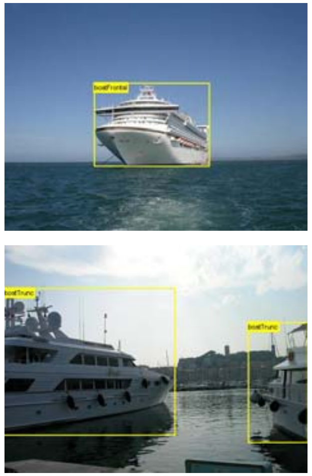
Aeroplane

Bicycle


Bird



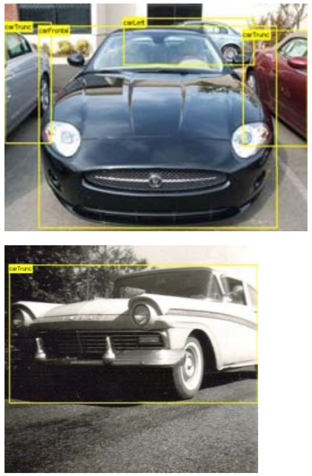
Boat



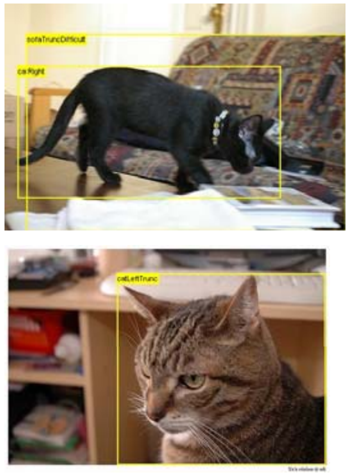
Bottle

Bus

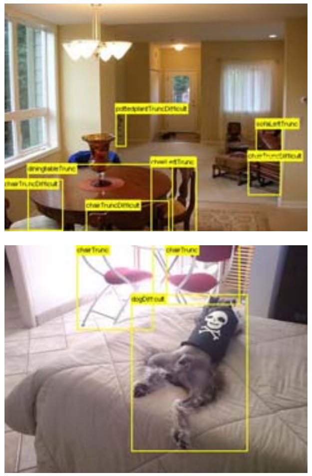

Car



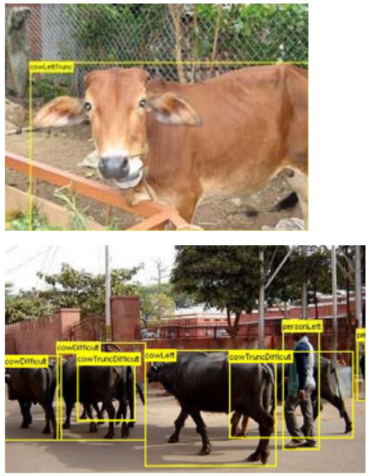
Cat



Chair



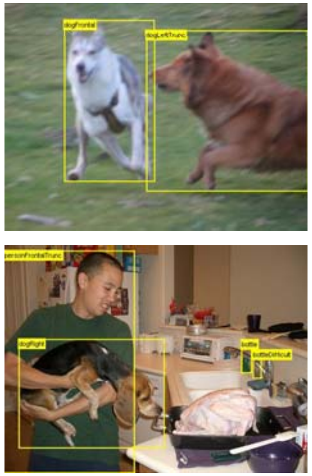
Cow



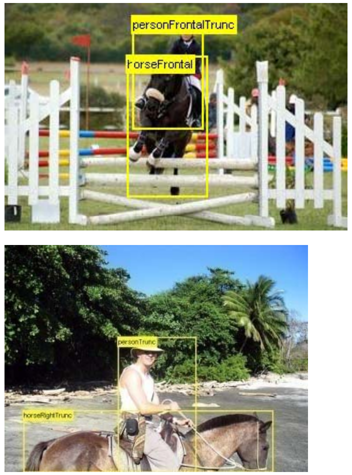
# Examples

Dining Table

Dog

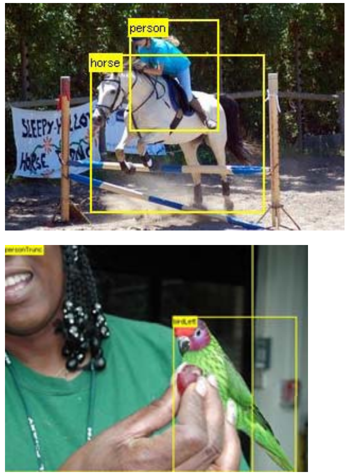


Horse



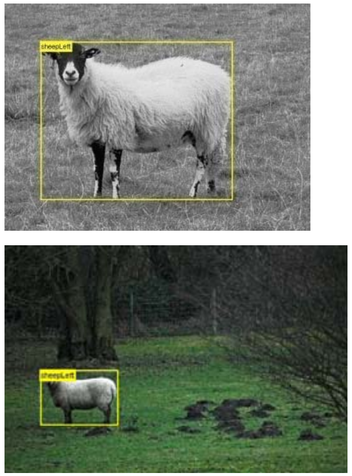

Motorbike

Person

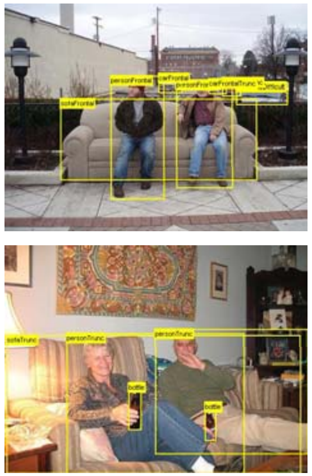



Potted Plant

Sheep



Sofa



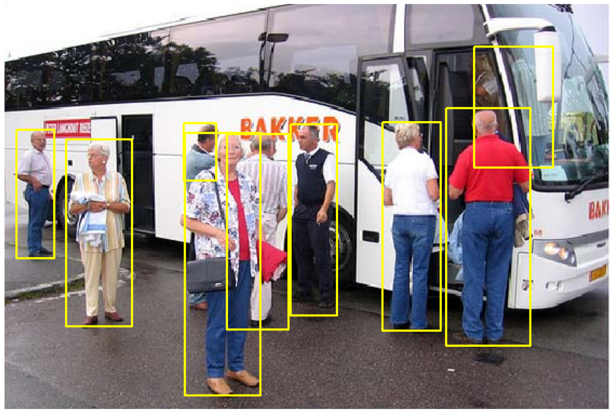
Train

TV/Monitor

# Main Challenge Tasks

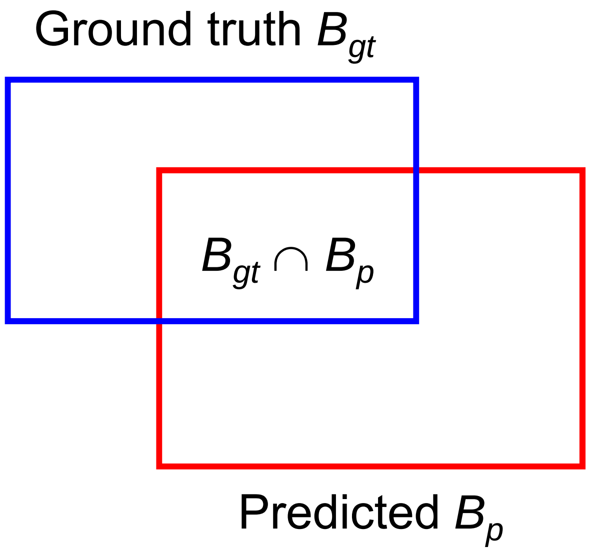
- **Classification**
  - Is there a dog in this image?
  - Evaluation by precision/recall

- **Detection**
  - Localize all the people (if any) in this image
  - Evaluation by precision/recall based on bounding box overlap



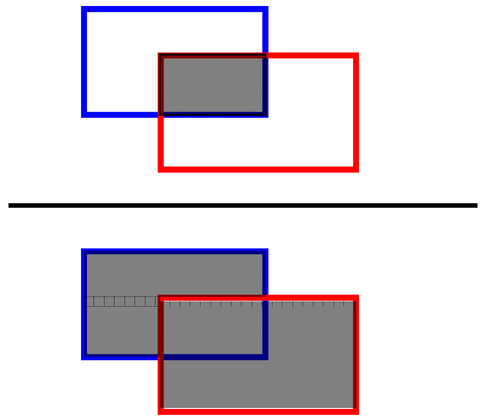
# Detection: Evaluation of Bounding Boxes

- Area of Overlap (AO) Measure




$$AO(B_{gt}, B_p) = \frac{|B_{gt} \bigcap B_p|}{|B_{gt} \bigcup B_p|}$$

Detection if

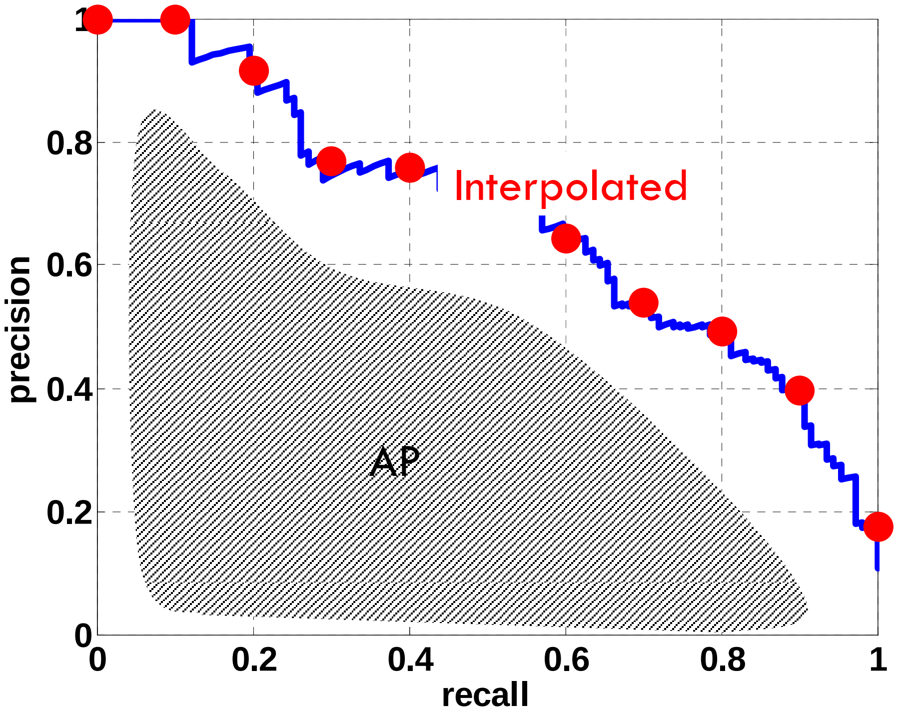


> Threshold

50%

# Classification/Detection Evaluation

- Average Precision [TREC] averages precision over the entire range of recall




- A good score requires both high recall and high precision
- Application-independent
- Penalizes methods giving high precision but low recall

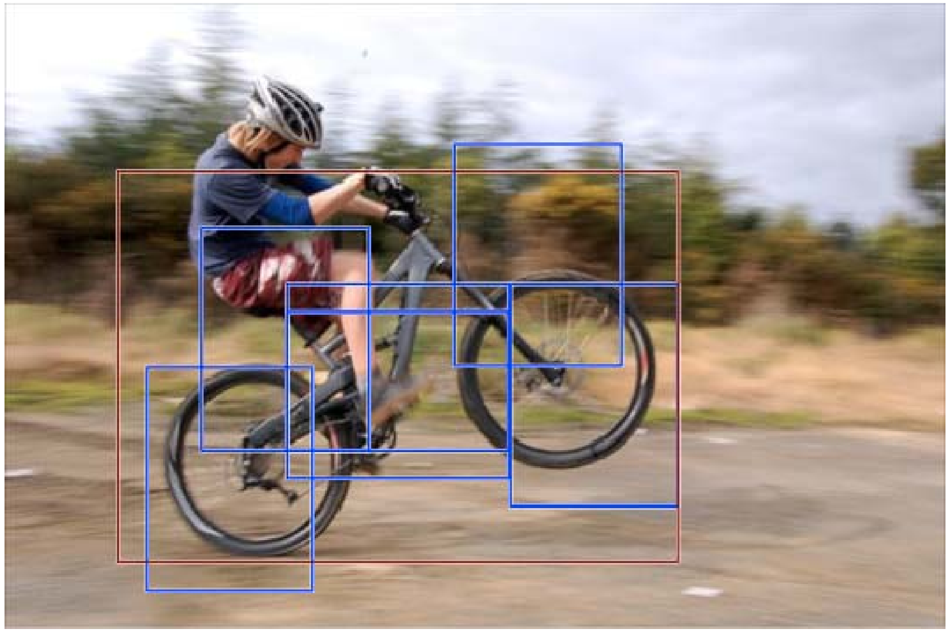
# Object Detection with Discriminatively Trained Part Based Models

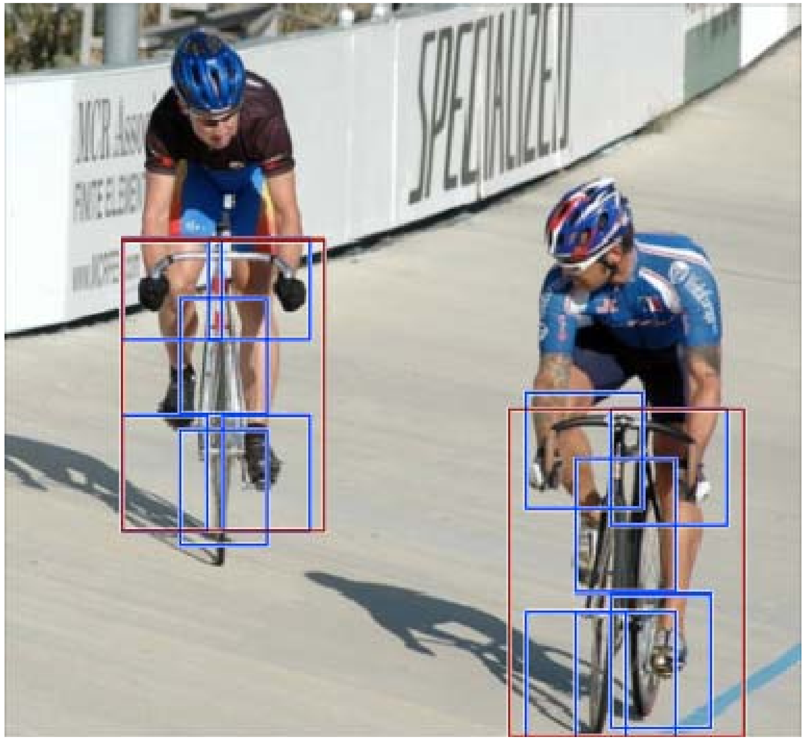
Pedro F. Felzenszwalb, David Mcallester,
Deva Ramanan, Ross Girshick

PAMI 2010

Matlab code available online:
http://www.cs.brown.edu/~pff/latent/

# Approach

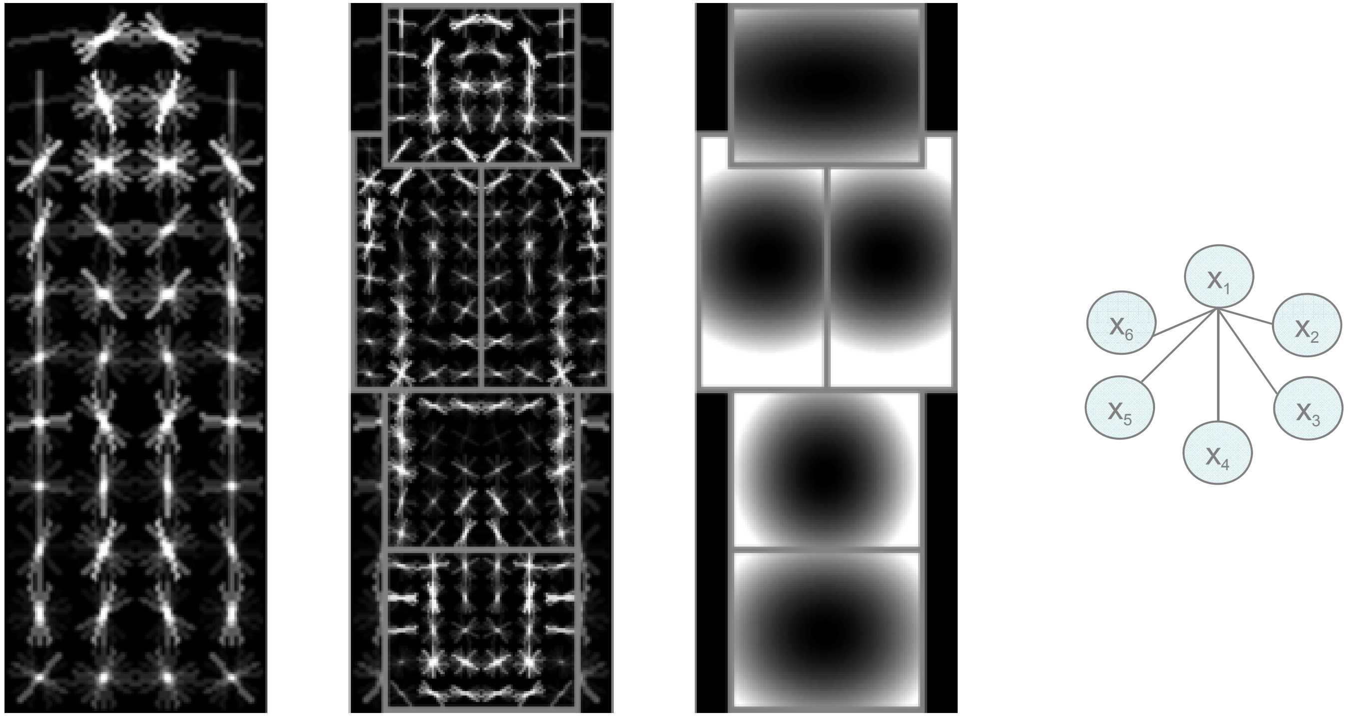





- Mixture of deformable part-based models
  - One component per “aspect” e.g. front/side view
- Each component has global template + deformable parts
- Discriminative training from bounding boxes alone

# Example Model

- One component of person model



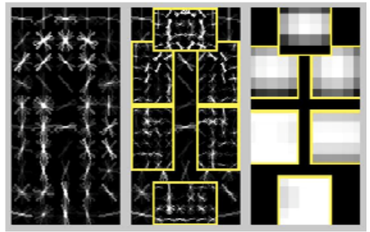

root filters
coarse resolution

part filters
finer resolution

deformation
models

# Object Hypothesis

- Position of root + each part
- Each part: HOG filter (at higher resolution)



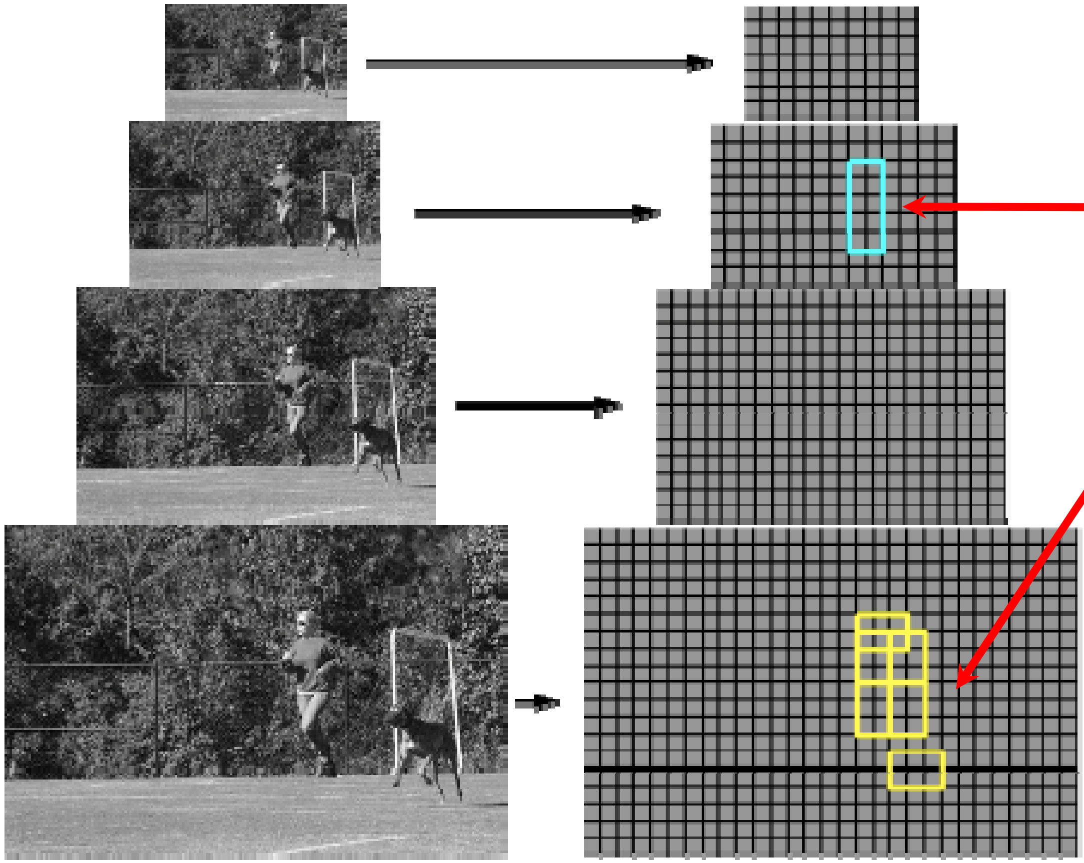


Image pyramid

HOG feature pyramid

$z = (p_0, \ldots, p_n)$

$p_0$ : location of root

$p_1, \ldots, p_n$ : location of parts

Score is sum of filter scores minus deformation costs

# Score of a Hypothesis

Appearance term

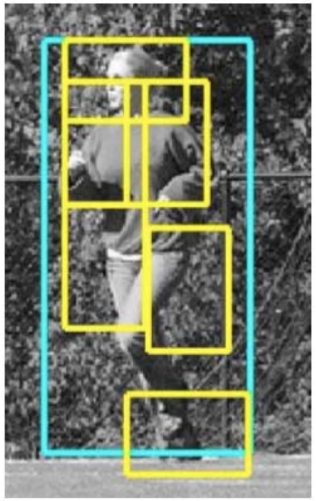
Spatial prior

$$\mathrm{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)$$

filters

displacements

deformation parameters



$$\mathrm{score}(z) = \beta \cdot \Psi(H, z)$$

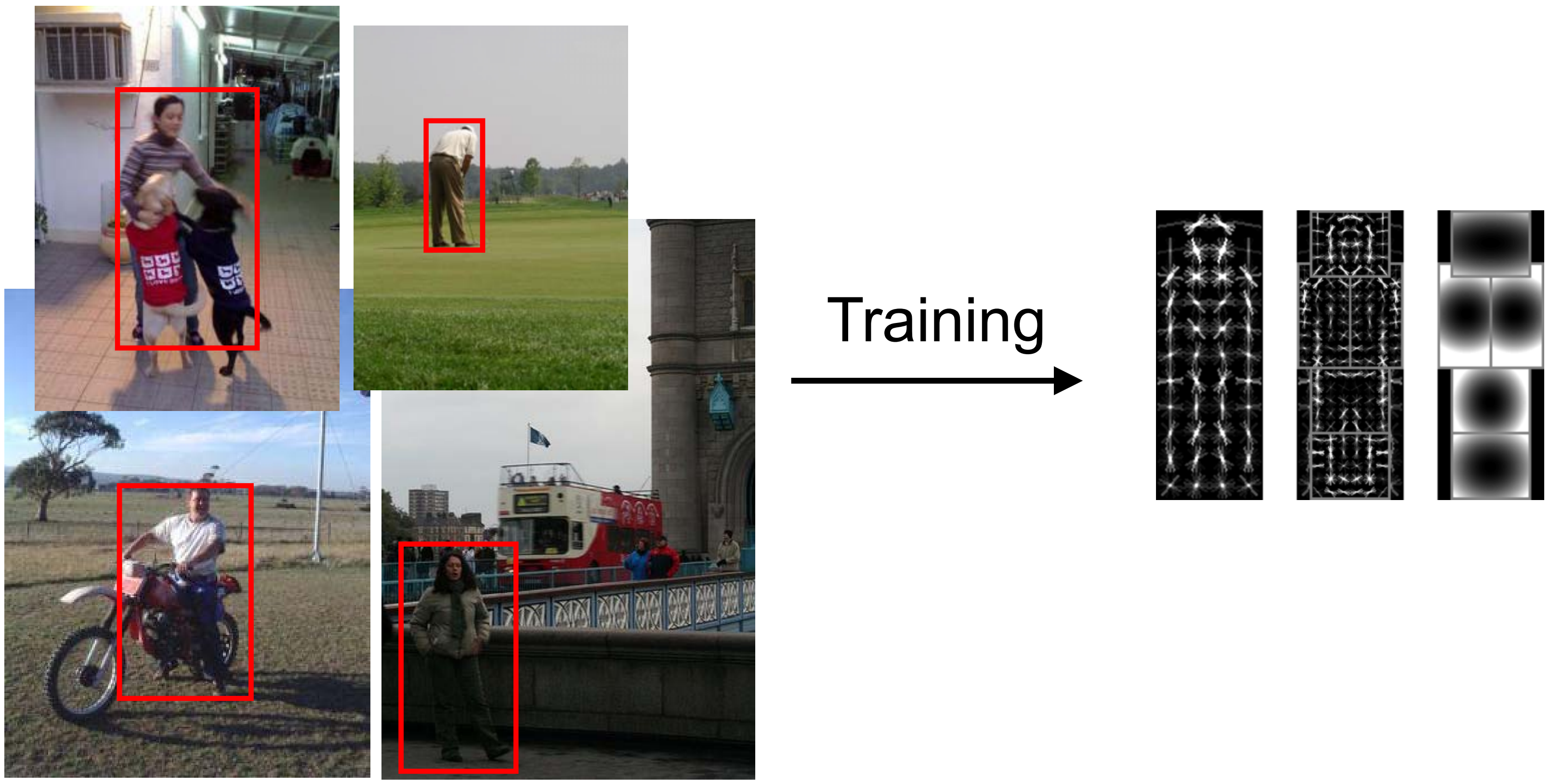
concatenation of filters and deformation parameters

concatenation of HOG features and part displacement features

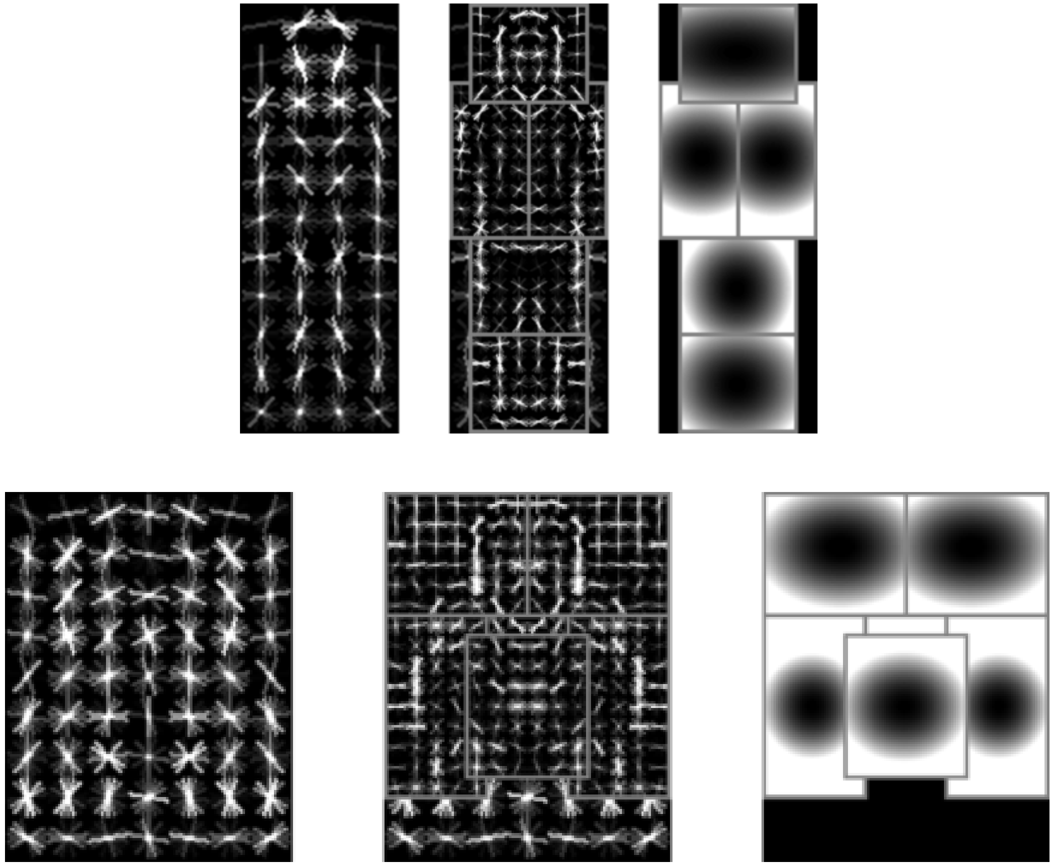
- Linear classifier applied to feature subset defined by hypothesis

# Training

- Training data = images + bounding boxes
- Need to learn: model structure, filters, deformation costs
- Latent SVM:
  - determine classifier and model parameters (location of the parts)



# Person Model

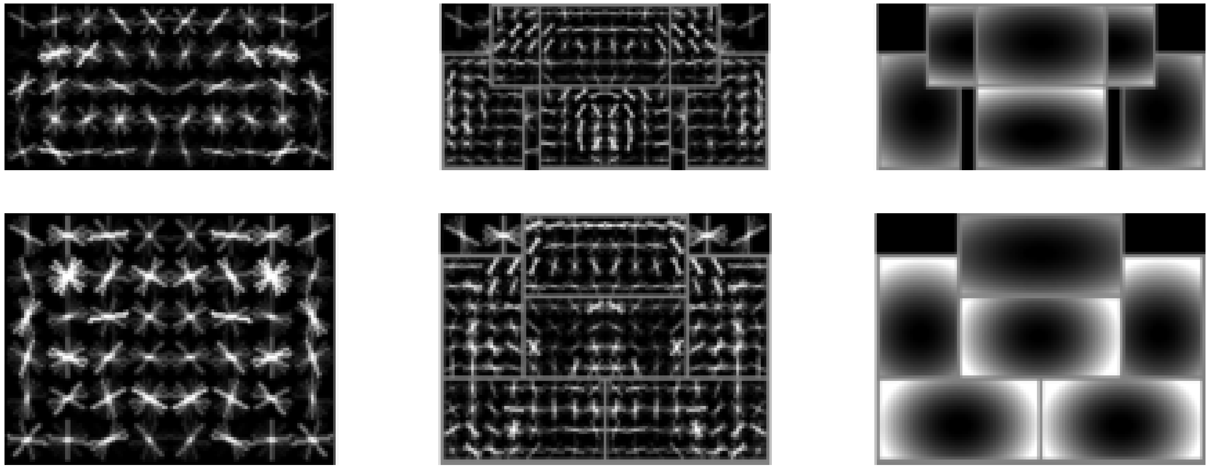


root filters
coarse resolution

part filters
finer resolution

deformation
models

# Car Model



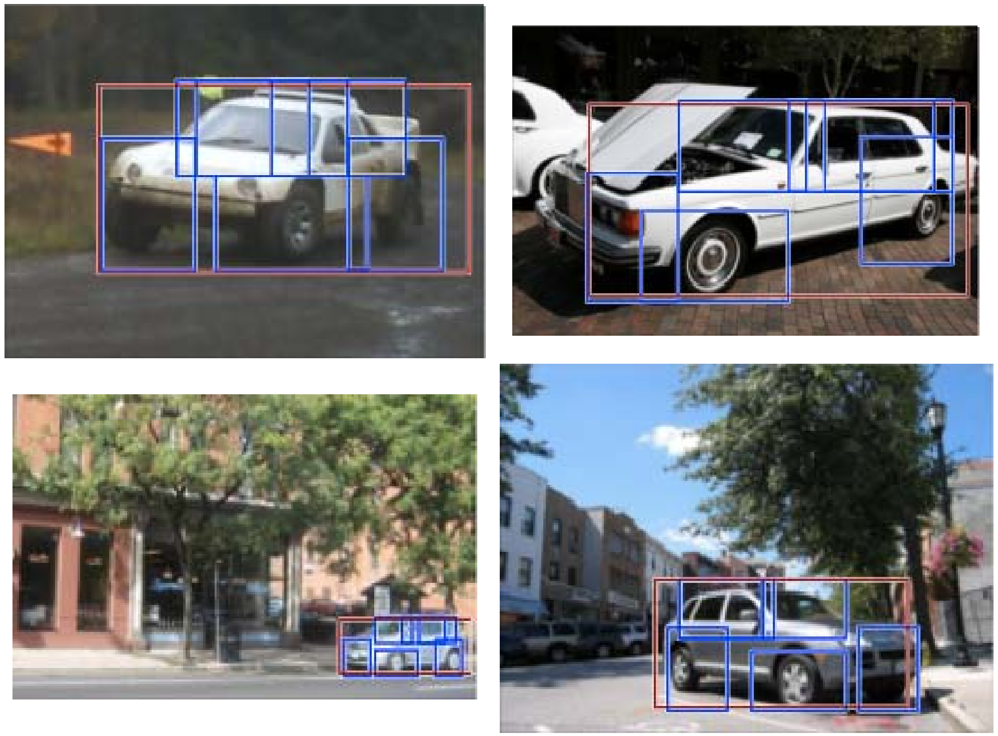
root filters
coarse resolution

part filters
finer resolution

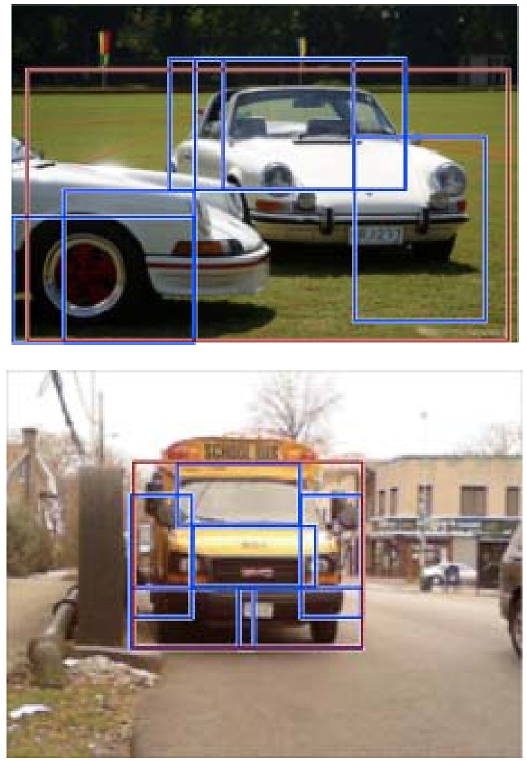
deformation
models

# Car Detections

high scoring true positives

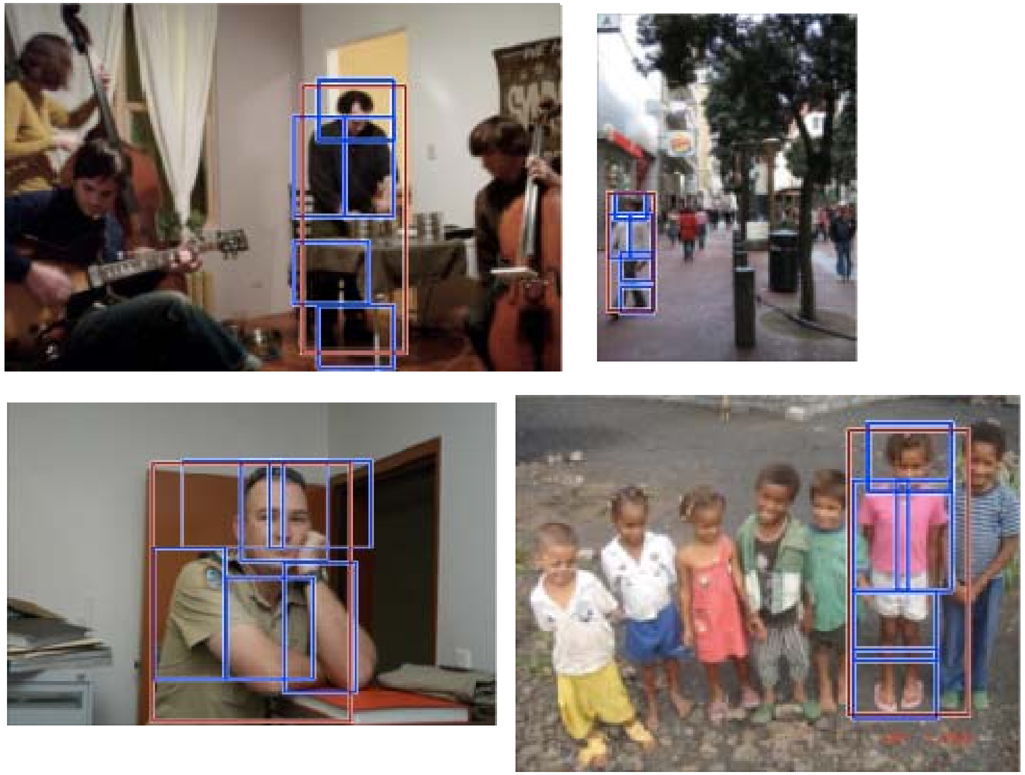


high scoring false positives



# Person Detections

high scoring true positives

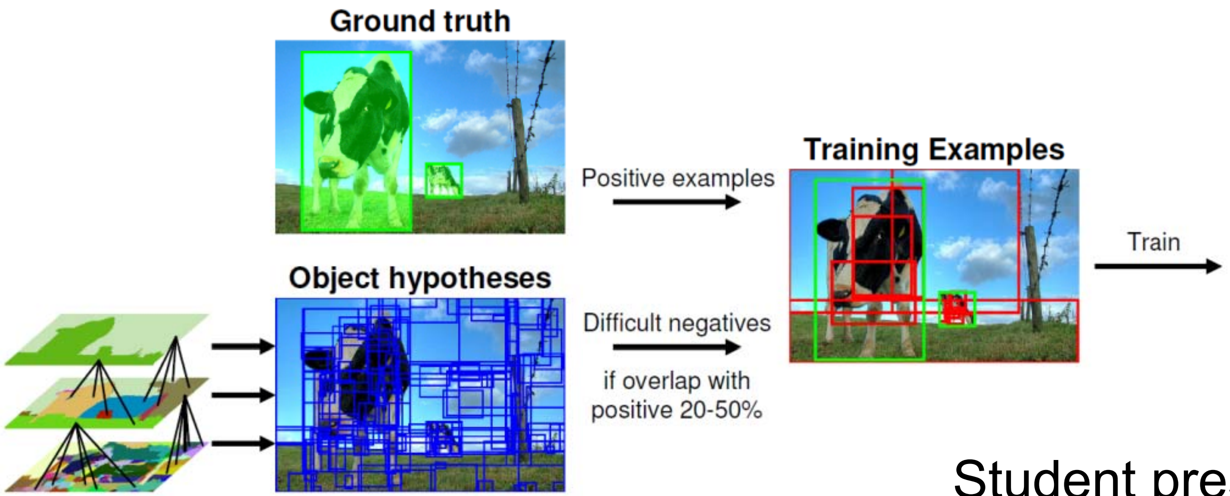


high scoring false positives
(not enough overlap)

# Selective search for object location [v.d.Sande et al. 11]

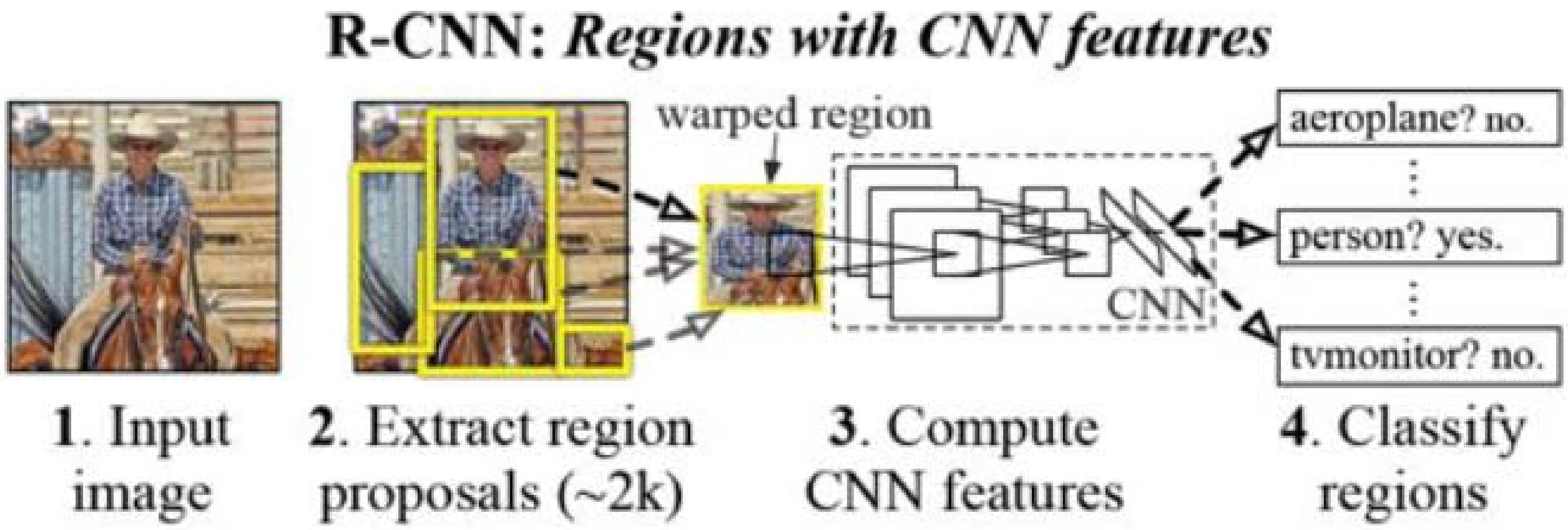
- Pre-select *class-independent* candidate image windows with segmentation

- Local features + bag-of-words
- SVM classifier with histogram intersection kernel + hard negative mining




Student presentation

# CNN features for detection




**Object detection system overview.** Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. **R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010**. For comparison, Uijlings et al. (2013) report 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. **The popular deformable part models perform at 33.4%.**

R. Girshick, J. Donahue, T. Darrell, and J. Malik,
Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR'14

Student presentation