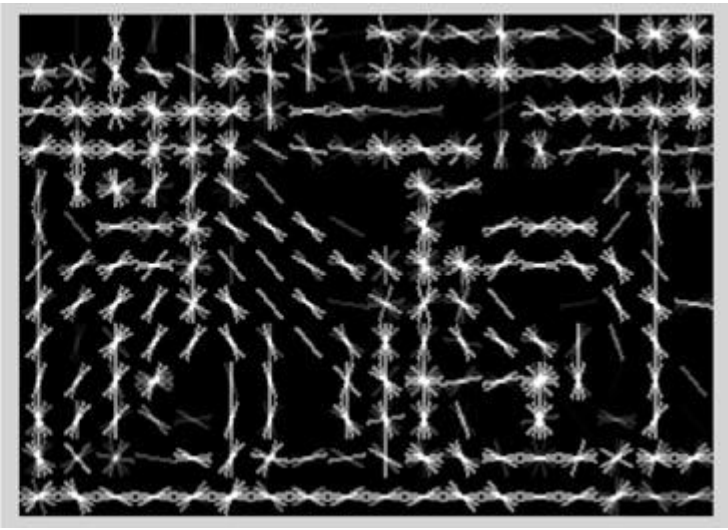
Object Detection with Statistical Template



Computer Vision

Jia-Bin Huang, Virginia Tech

Many slides from D. Hoiem, J. Hays

Administrative stuffs

- HW 5 is out
  - Due 11:59pm on Wed, November 16
  - Scene categorization
  - Please start early
- Final project proposal
  - Feedback via emails

Today's class

- Review/finish supervised learning
- Overview of object category detection
- Statistical template matching
  - Dalal-Triggs pedestrian detector (basic concept)
  - Viola-Jones detector (cascades, integral images)
  - R-CNN detector (object proposals/CNN)

Image Categorization

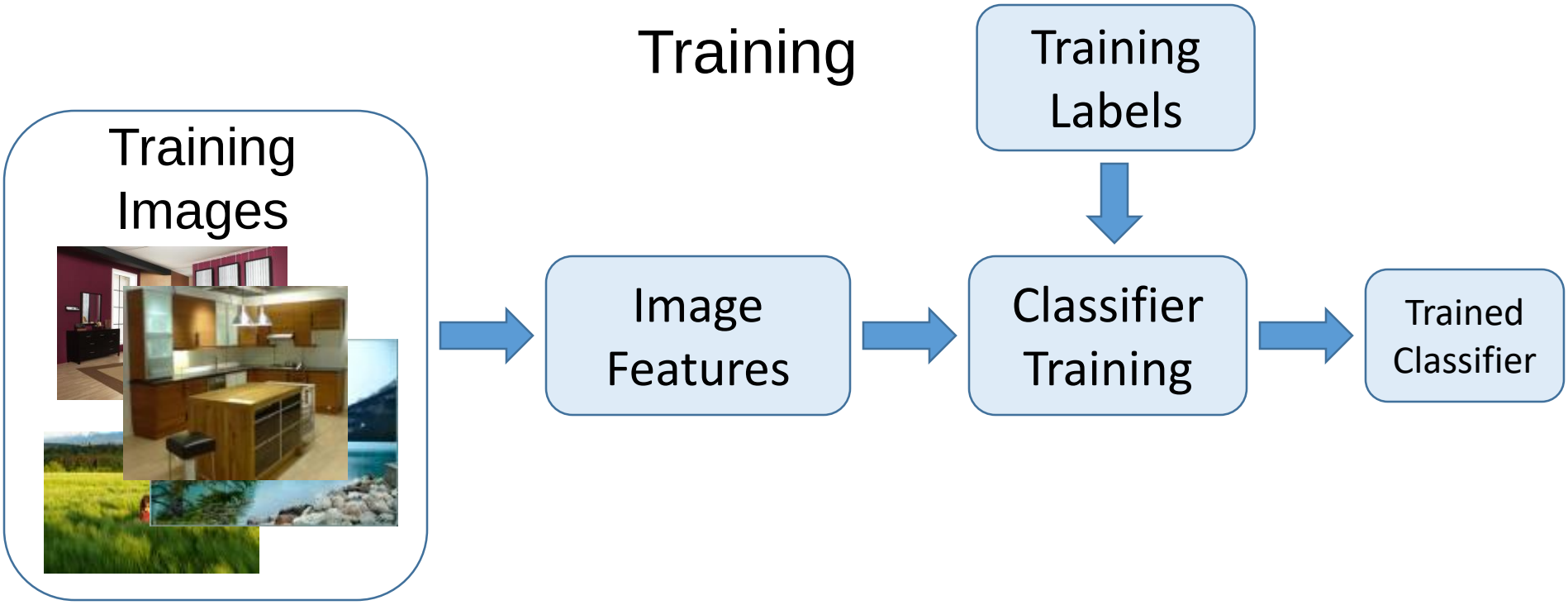


Image Categorization

Training

Training Images

Training Labels

Image Features

Classifier Training

Trained Classifier

Testing

Test Image

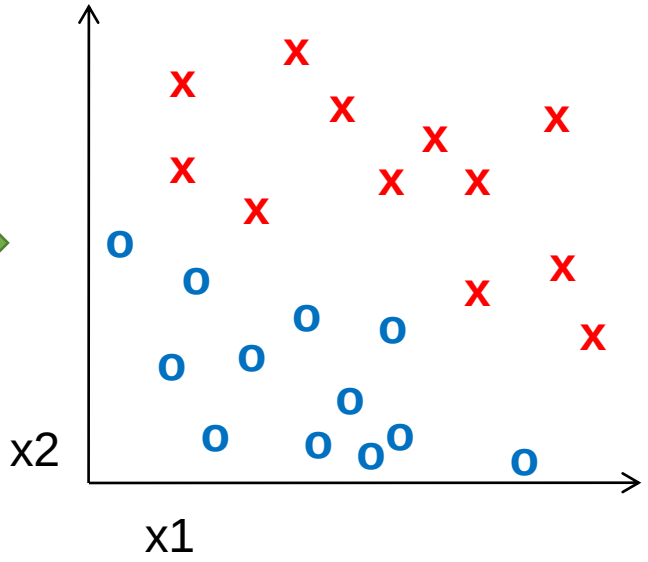
Image Features

Trained Classifier

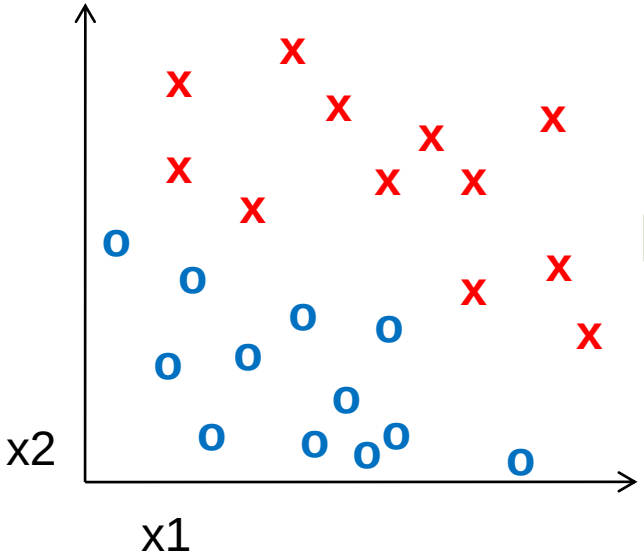
Prediction

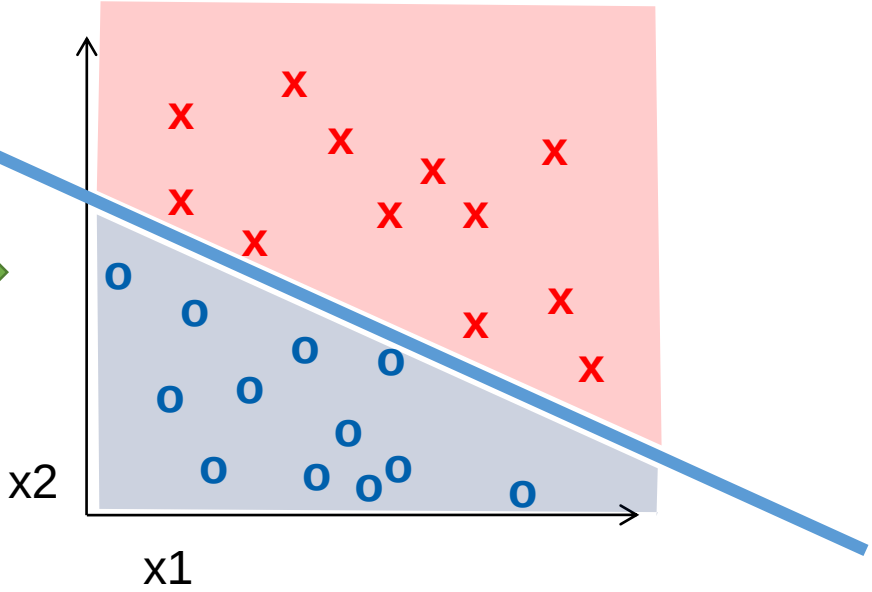
Outdoor

- **Image features**: map images to feature space



- **Classifiers**: map feature space to label space





Different types of classification

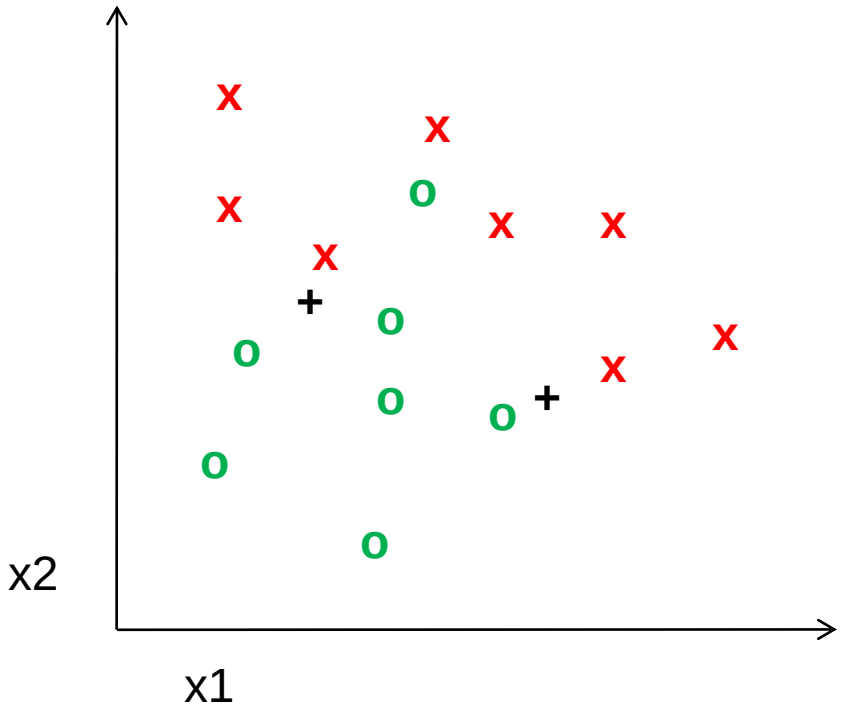
- **Exemplar-based**: transfer category labels from examples with most similar features
  - What similarity function? What parameters?
- **Linear classifier**: confidence in positive label is a weighted sum of features
  - What are the weights?
- **Non-linear classifier**: predictions based on more complex function of features
  - What form does the classifier take? Parameters?
- **Generative classifier**: assign to the label that best explains the features (makes features most likely)
  - What is the probability function and its parameters?

Note: You can always fully design the classifier by hand, but usually this is too difficult. Typical solution: learn from training examples.

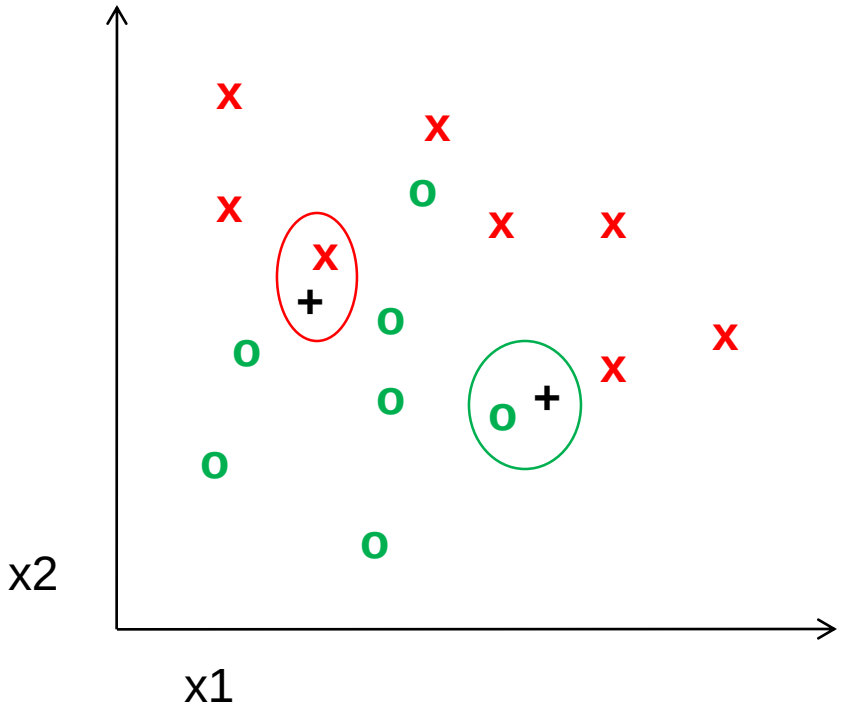
Exemplar-based Models

- Transfer the label(s) of the most similar training examples

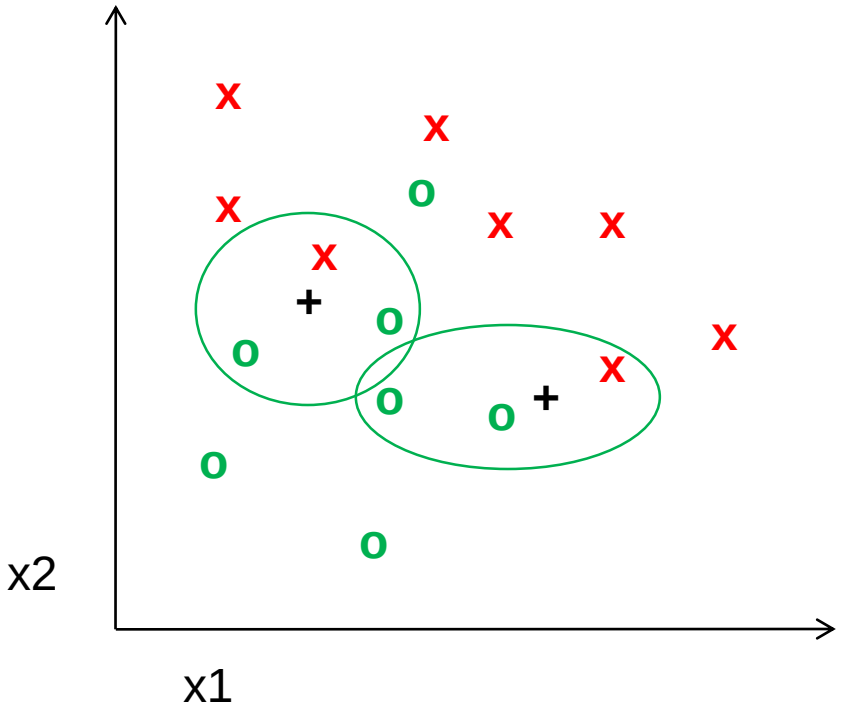
K-nearest neighbor classifier



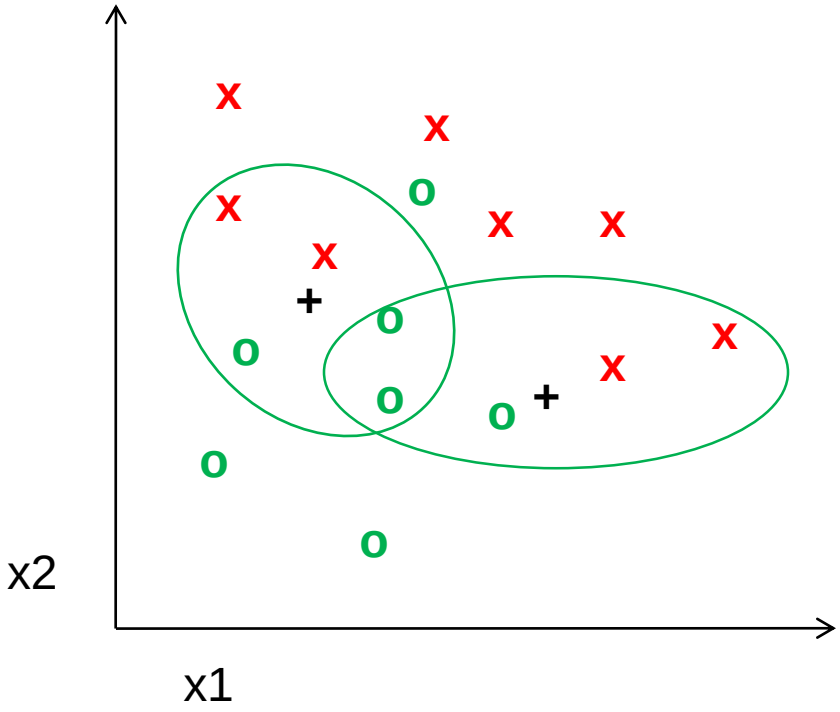
1-nearest neighbor



3-nearest neighbor



5-nearest neighbor



Using K-NN

- Simple, a good one to try first
- Higher K gives smoother functions
- No training time (unless you want to learn a distance function)
- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error

Discriminative classifiers

Learn a simple function of the input features that confidently predicts the true labels on the training set

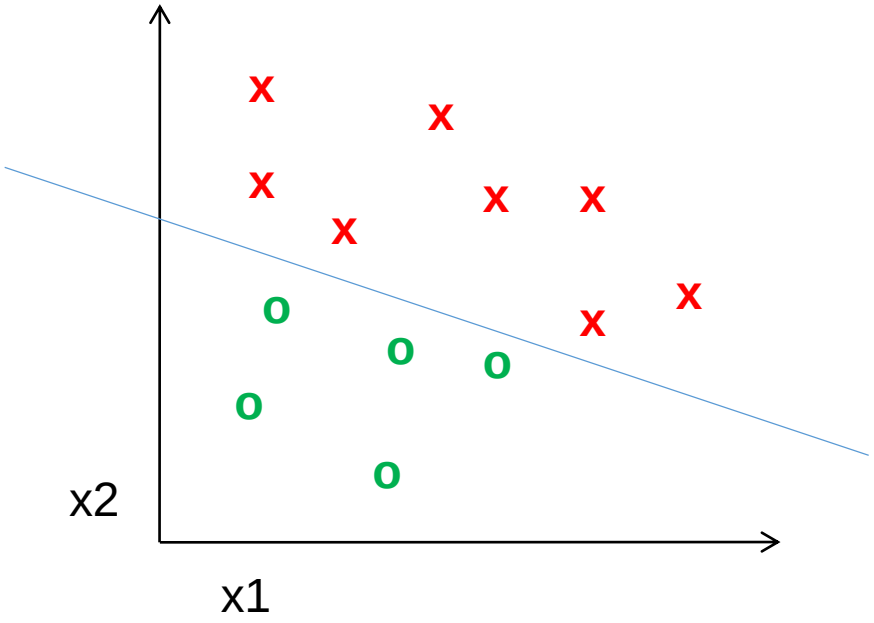
$$y = f(x)$$

Training Goals

1. Accurate classification of training data
2. Correct classifications are confident
3. Classification function is simple

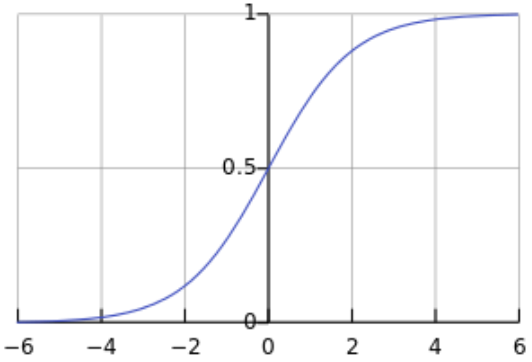
Classifiers: Logistic Regression

- Objective
- Parameterization
- Regularization
- Training
- Inference



$$\log \frac{P(x_1, x_2 \mid y=1)}{P(x_1, x_2 \mid y=-1)} = \mathbf{w}^T \mathbf{x}$$

$$P(y=1 \mid x_1, x_2) = 1/\left(1+\exp\left(-\mathbf{w}^T \mathbf{x}\right)\right)$$

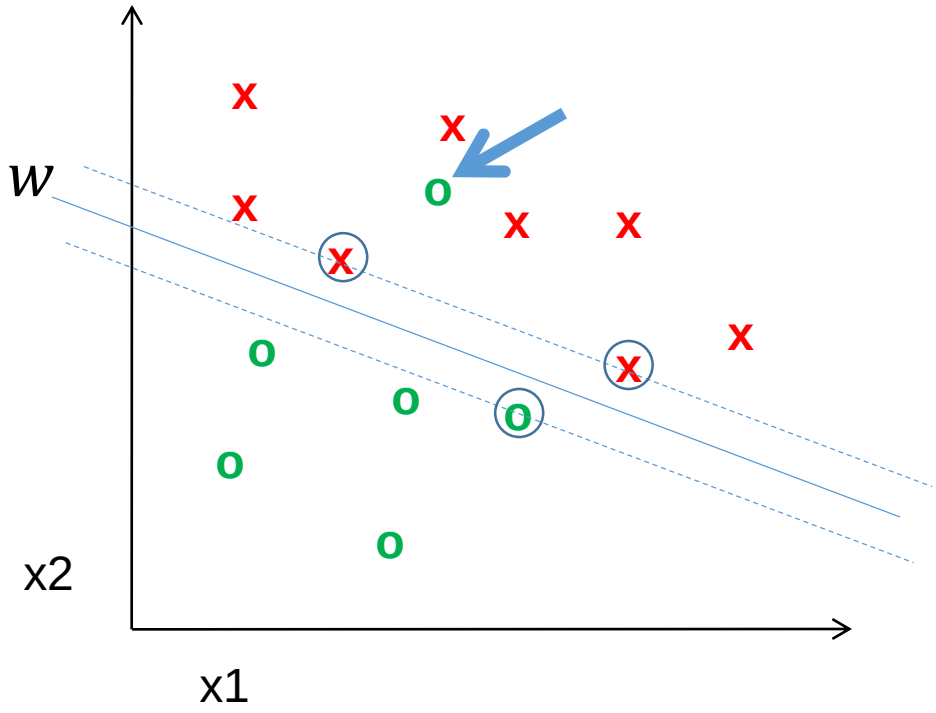


The **objective function** of most discriminative classifiers includes a **loss term** and a **regularization term**.

Using Logistic Regression

- Quick, simple classifier (good one to try first)
- Use L2 or L1 regularization
  - L1 does feature selection and is robust to irrelevant features but slower to train

Classifiers: Linear SVM



$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i + \lambda\|w\|^2$$

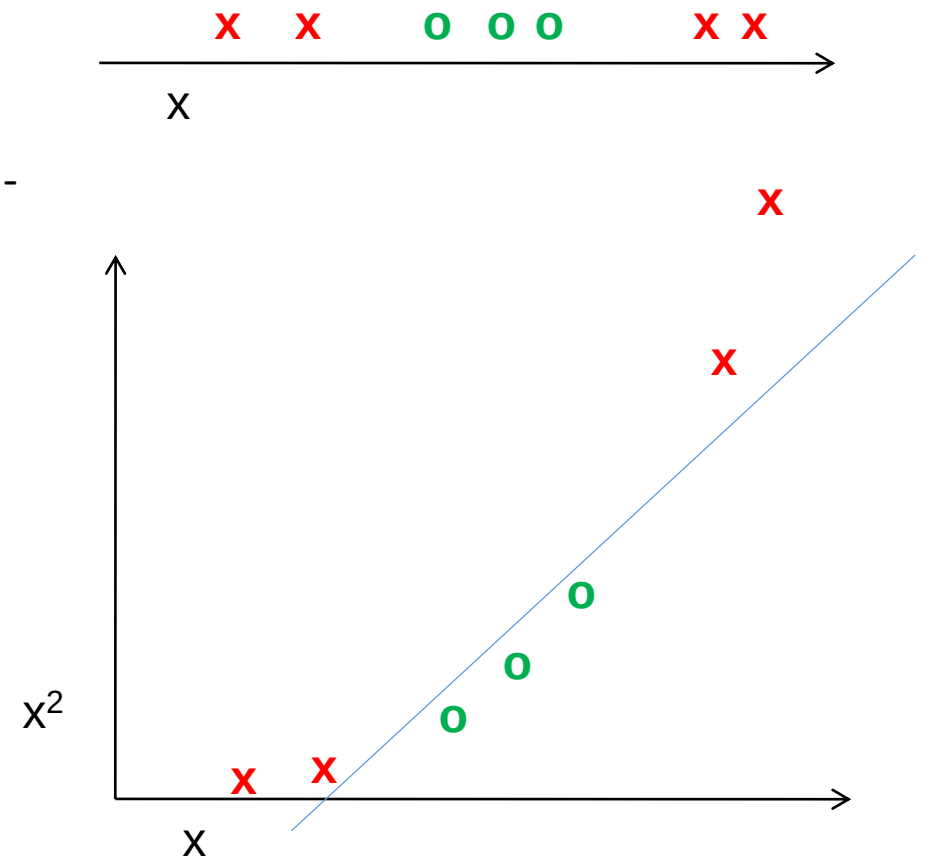
$$\text{subject to } y_i\left(x_i \cdot w + b\right) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i.$$

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x}_i$$

Classifiers: Kernelized SVM

Kernel trick:

- implicitly mapping their inputs into high-dimensional feature spaces



$$\vec{w} = \sum_{i=1}^{n} c_i y_i \varphi(\vec{x}_i)$$

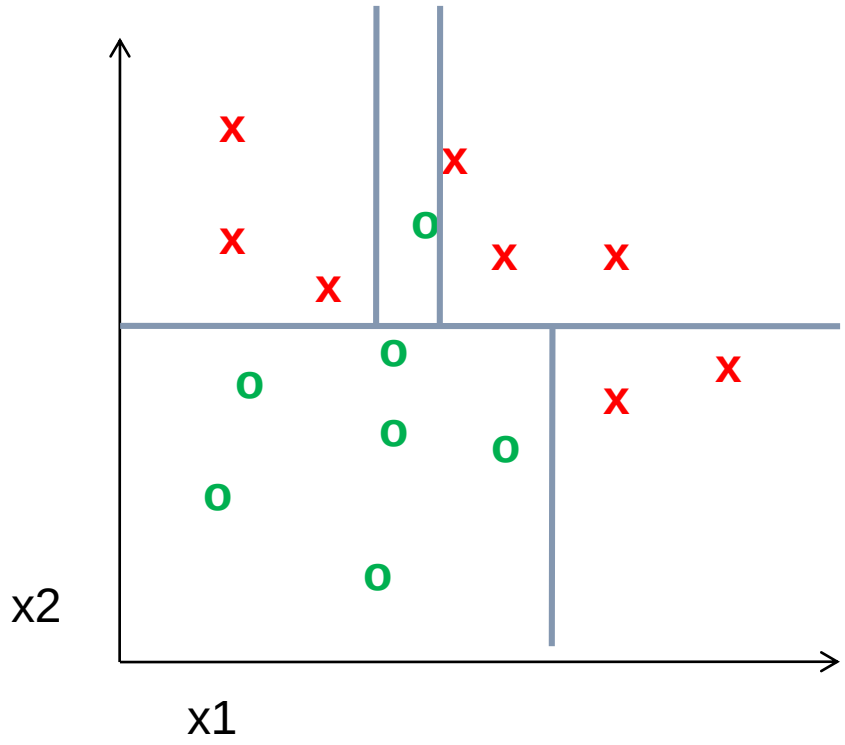
$$\vec{z} \mapsto \operatorname{sgn}(\vec{w} \cdot \varphi(\vec{z}) + b) = \operatorname{sgn}\left(\left[\sum_{i=1}^{n} c_i y_i k(\vec{x}_i, \vec{z})\right] + b\right)$$

Using SVMs

- Good general purpose classifier
  - Generalization depends on margin, so works well with many weak features
  - No feature selection
  - Usually requires some parameter tuning

- Choosing kernel
  - Linear: fast training/testing – start here
  - RBF: related to neural networks, nearest neighbor
  - Chi-squared, histogram intersection: good for histograms (but slower, esp. chi-squared)
  - Can learn a kernel function

Classifiers: Decision Trees



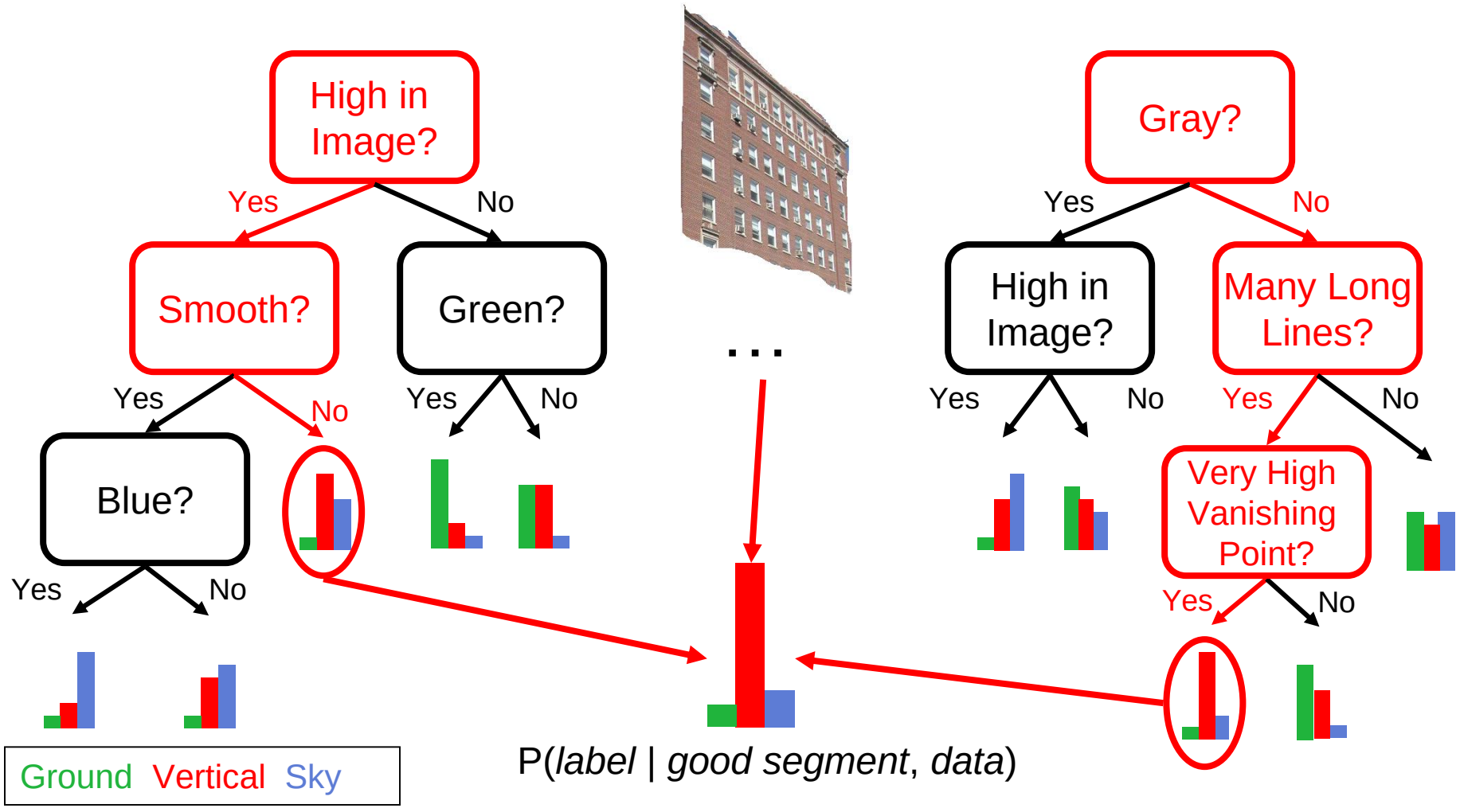
Ensemble Methods: Boosting

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 $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

figure from Friedman et al. 2000

Boosted Decision Trees



[Collins et al. 2002]

Using Boosted Decision Trees

- Flexible: can deal with both continuous and categorical variables
- How to control bias/variance trade-off
  - Size of trees
  - Number of trees
- Boosting trees often works best with a small number of well-designed features
- Boosting “stubs” can give a fast classifier

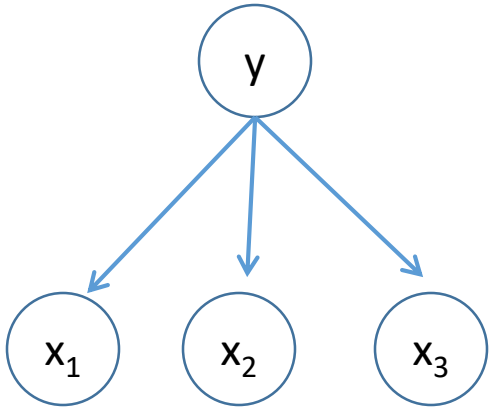
Generative classifiers

- Model the joint probability of the features and the labels
  - Allows direct control of independence assumptions
  - Can incorporate priors
  - Often simple to train (depending on the model)

- Examples
  - Naïve Bayes
  - Mixture of Gaussians for each class

Naïve Bayes

- Objective
- Parameterization
- Regularization
- Training
- Inference



Conditional independence

$$p(x_i | x_{i+1}, \ldots, x_n, C_k) = p(x_i | C_k)$$

Inference

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i | C_k)$$

Using Naïve Bayes

- Simple thing to try for categorical data
- Very fast to train/test

Web-based demo

- SVM
- Neural Network
- Random Forest

Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Deep networks
- Etc.

Which is the best one?

No Free Lunch Theorem

Generalization Theory

- It’s not enough to do well on the training set: we want to also make good predictions for new examples

Bias-Variance Trade-off

$$E(MSE) = noise^2 + bias^2 + variance$$

Unavoidable error

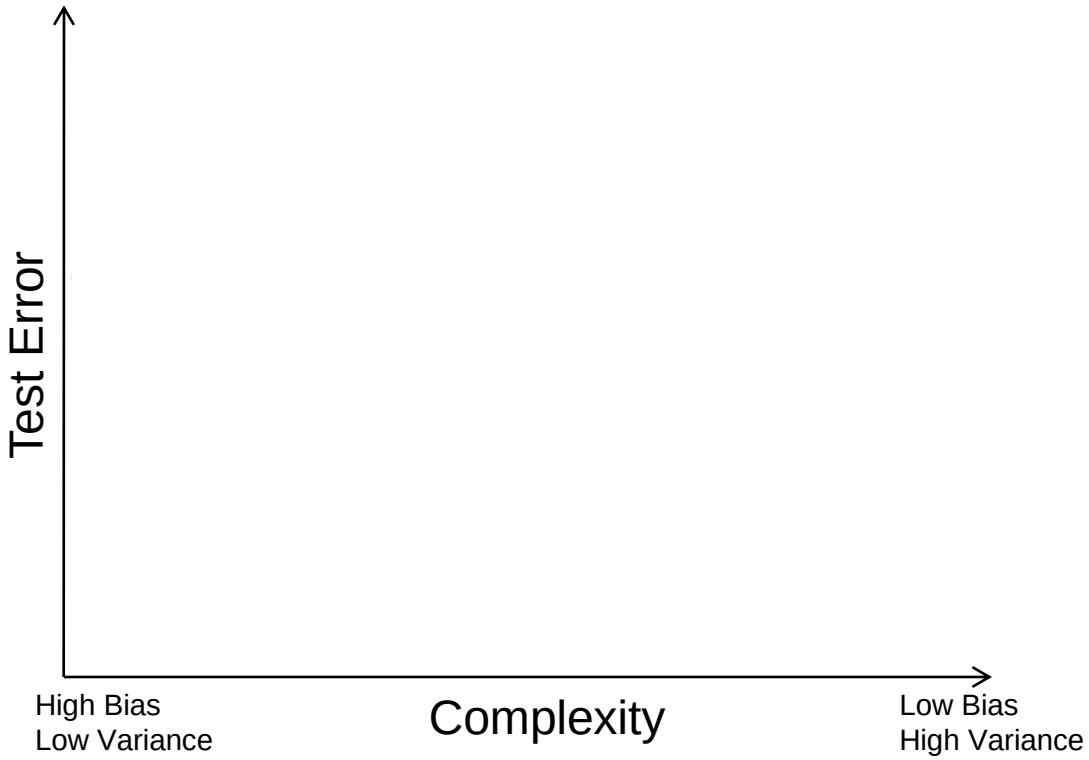
Error due to incorrect assumptions

Error due to variance parameter estimates from training samples

See the following for explanation of bias-variance (also Bishop's "Neural Networks" book):
http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

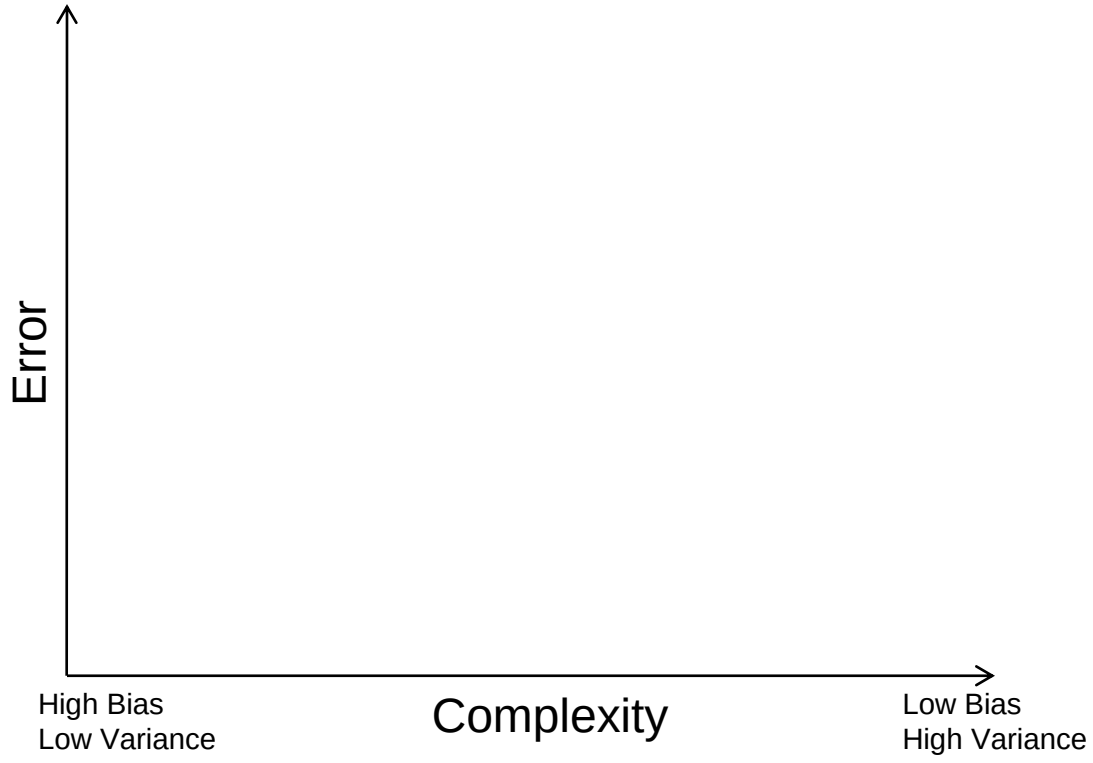
Bias and Variance

$$\text{Error} = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

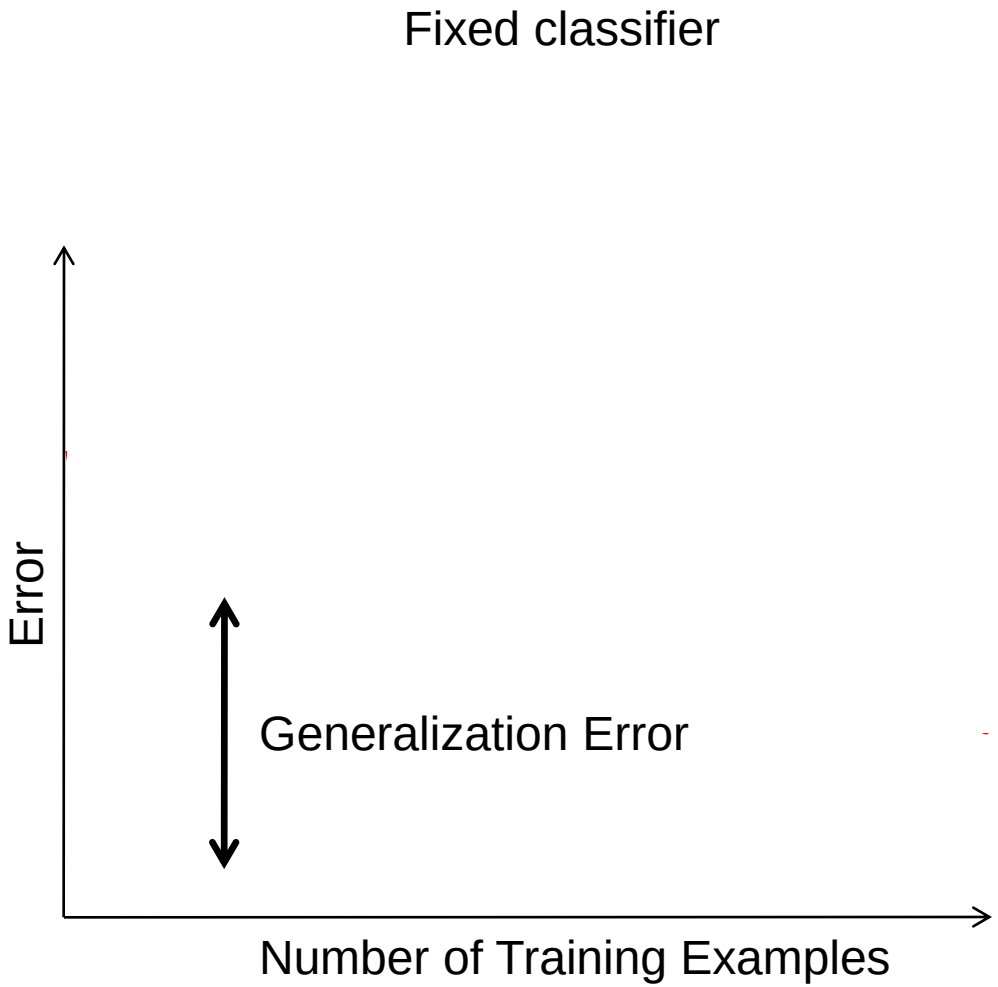


Choosing the trade-off

- Need validation set
- Validation set is separate from the test set



Effect of Training Size



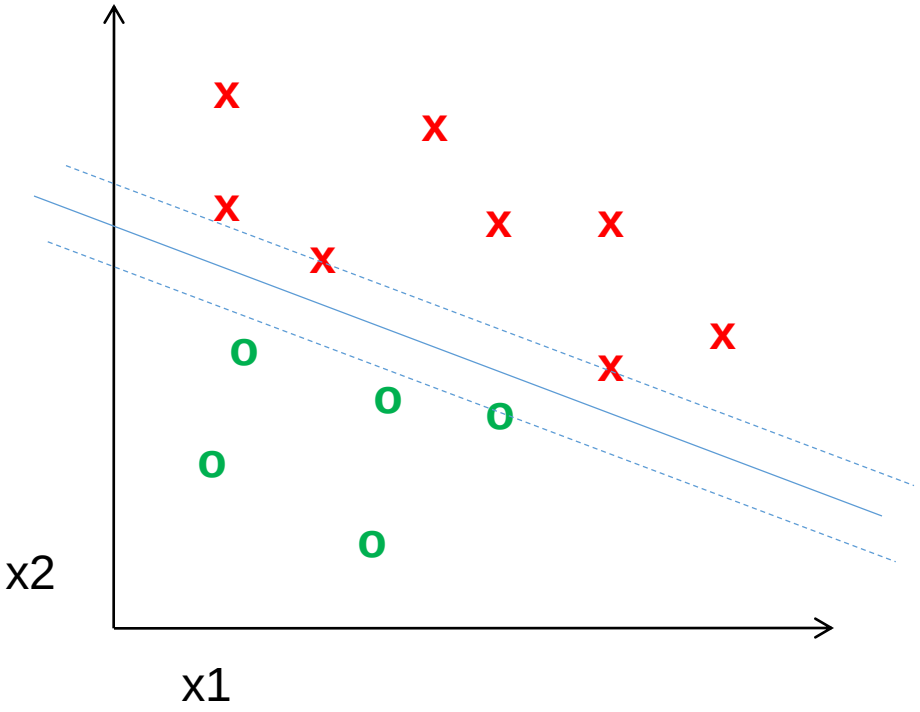
How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Use fewer features
- Get more training data

Which of these could actually lead to greater error?

Reducing Risk of Error

- Margins



The perfect classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: makes assumptions that fit the problem
- Regularization: right level of regularization for amount of training data
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for objective function in evaluation

Comparison

assuming x in {0 1}

	Learning Objective	Training	Inference
Naïve Bayes	$\text{maximize} \sum_i \left[\sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$	$\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$	$\theta_1^T \mathbf{x} + \theta_0^T (1 - \mathbf{x}) > 0$ where $\theta_{1j} = \log \frac{P(x_j = 1 \mid y = 1)}{P(x_j = 1 \mid y = 0)}$, $\theta_{0j} = \log \frac{P(x_j = 0 \mid y = 1)}{P(x_j = 0 \mid y = 0)}$
Logistic Regression	$\text{maximize} \sum_i \log(P(y_i \mid \mathbf{x}, \boldsymbol{\theta})) + \lambda \lVert \boldsymbol{\theta} \rVert$ where $P(y_i \mid \mathbf{x}, \boldsymbol{\theta}) = 1/(1 + \exp(-y_i \boldsymbol{\theta}^T \mathbf{x}))$	Gradient ascent	$\boldsymbol{\theta}^T \mathbf{x} > t$
Linear SVM	$\text{minimize } \lambda \sum_i \xi_i + \frac{1}{2} \lVert \boldsymbol{\theta} \rVert$ such that $y_i \boldsymbol{\theta}^T \mathbf{x} \geq 1 - \xi_i \ \ \forall i, \ \xi_i \geq 0$	Quadratic programming or subgradient opt.	$\boldsymbol{\theta}^T \mathbf{x} > t$
Kernelized SVM	complicated to write	Quadratic programming	$\sum_i y_i \alpha_i K(\hat{\mathbf{x}}_i, \mathbf{x}) > 0$
Nearest Neighbor	most similar features → same label	Record data	y_i where $i = \underset{i}{\text{argmin}} \ K(\hat{\mathbf{x}}_i, \mathbf{x})$

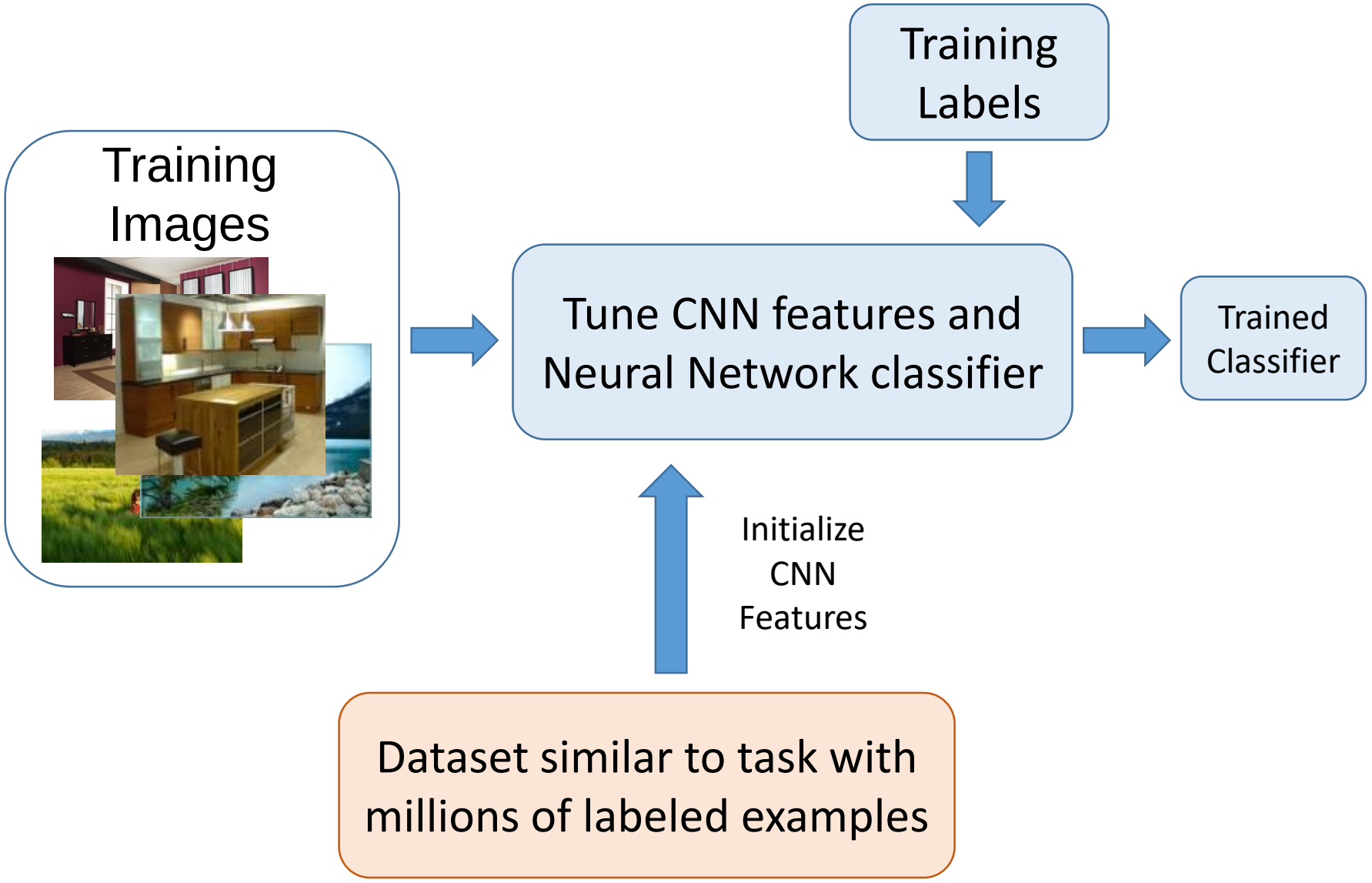
Characteristics of vision learning problems

- Lots of continuous features
  - E.g., HOG template may have 1000 features
  - Spatial pyramid may have ~15,000 features
- Imbalanced classes
  - often limited positive examples, practically infinite negative examples
- Difficult prediction tasks

When a massive training set is available

- Relatively new phenomenon
  - MNIST (handwritten letters) in 1990s, LabelMe in 2000s, ImageNet (object images) in 2009, …
- Want classifiers with low bias (high variance ok) and reasonably efficient training
- Very complex classifiers with simple features are often effective
  - Random forests
  - Deep convolutional networks

New training setup with moderate sized datasets



Practical tips

- Preparing features for linear classifiers
  - Often helps to make zero-mean, unit-dev
  - For non-ordinal features, convert to a set of binary features

- Selecting classifier meta-parameters (e.g., regularization weight)
  - Cross-validation: split data into subsets; train on all but one subset, test on remaining; repeat holding out each subset
    - Leave-one-out, 5-fold, etc.

- Most popular classifiers in vision
  - *SVM*: linear for when fast training/classification is needed; performs well with lots of weak features
  - *Logistic Regression*: outputs a probability; easy to train and apply
  - *Nearest neighbor*: hard to beat if there is tons of data (e.g., character recognition)
  - *Boosted stumps or decision trees*: applies to flexible features, incorporates feature selection, powerful classifiers
  - *Random forests*: outputs probability; good for simple features, tons of data
  - *Deep networks / CNNs*: flexible output; learns features; adapt existing network (which is trained with tons of data) or train new with tons of data

- Always try at least two types of classifiers

Making decisions about data

- 3 important design decisions:
  1) What data do I use?
  2) How do I represent my data (what feature)?
  3) What classifier / regressor / machine learning tool do I use?
- These are in decreasing order of importance
- Deep learning addresses 2 and 3 simultaneously (and blurs the boundary between them).
- You can take the representation from deep learning and use it with any classifier.

Things to remember

- No free lunch: machine learning algorithms are tools
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
  - Though with enough data, smart features can be learned
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Some Machine Learning References

- General
  - Tom Mitchell, *Machine Learning*, McGraw Hill, 1997
  - Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995

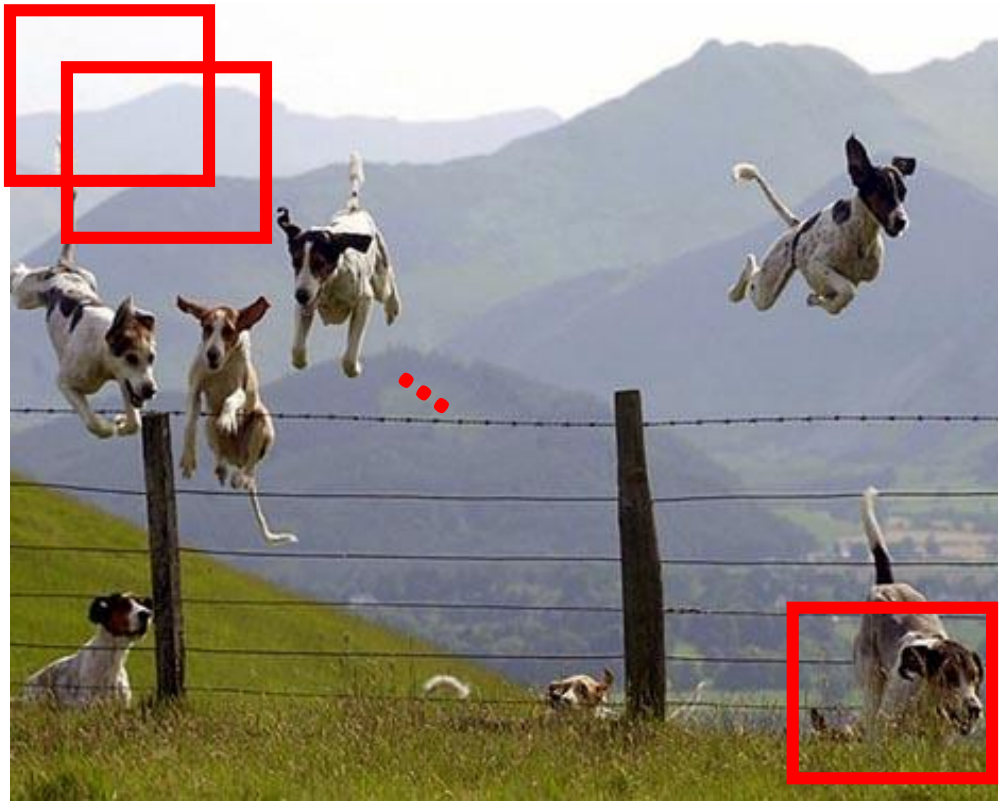
- Adaboost
  - Friedman, Hastie, and Tibshirani, “Additive logistic regression: a statistical view of boosting”, Annals of Statistics, 2000

- SVMs
  - http://www.support-vector.net/icml-tutorial.pdf

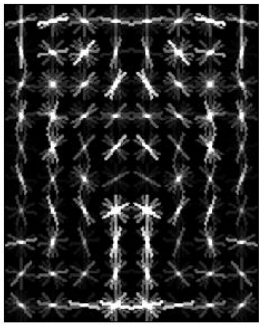
- Random forests
  - http://research.microsoft.com/pubs/155552/decisionForests_MSR_TR_2011_114.pdf

Object Category Detection

- Focus on object search: “Where is it?”
- Build templates that quickly differentiate object patch from background patch



Dog Model



Object or
Non-Object?

Challenges in modeling the object class

Illumination

Object pose

Clutter

Occlusions

Intra-class appearance

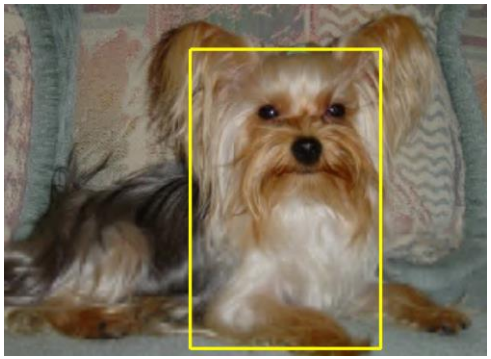
Viewpoint

Slide from K. Grauman, B. Leibe

Challenges in modeling the non-object class

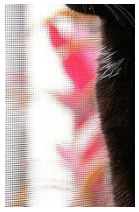
True Detections

Bad Localization



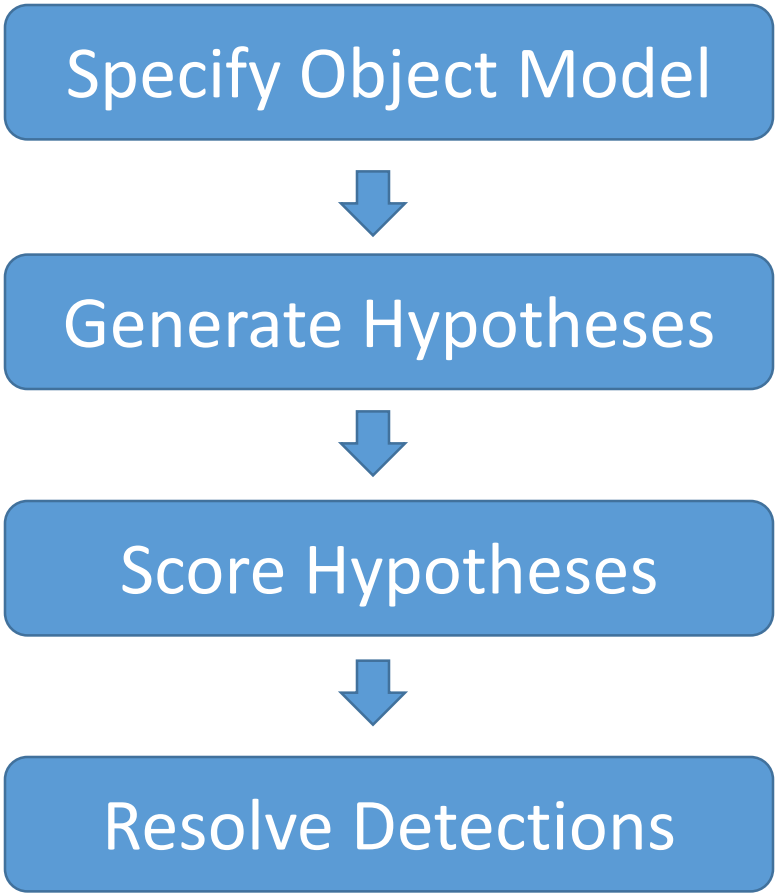
Confused with Similar Object

Misc. Background



Confused with Dissimilar Objects

General Process of Object Recognition

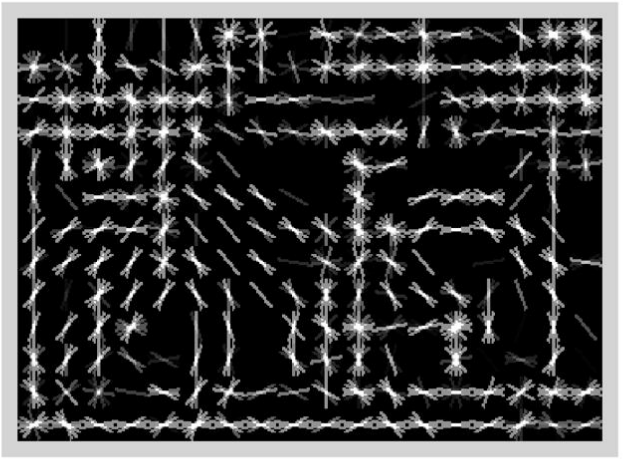


What are the object parameters?

Specifying an object model

1. Statistical Template in Bounding Box
   - Object is some (x,y,w,h) in image
   - Features defined wrt bounding box coordinates

Image

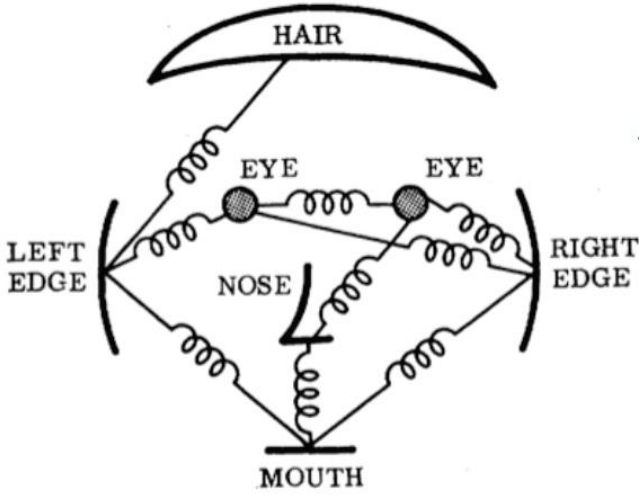


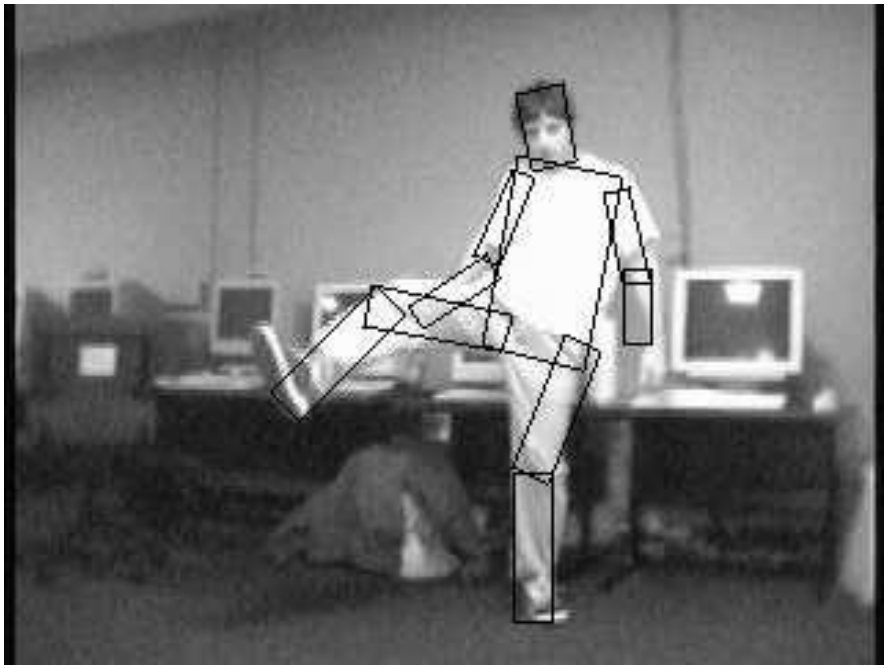
Template Visualization

Images from Felzenszwalb

Specifying an object model

2. Articulated parts model
   - Object is configuration of parts
   - Each part is detectable



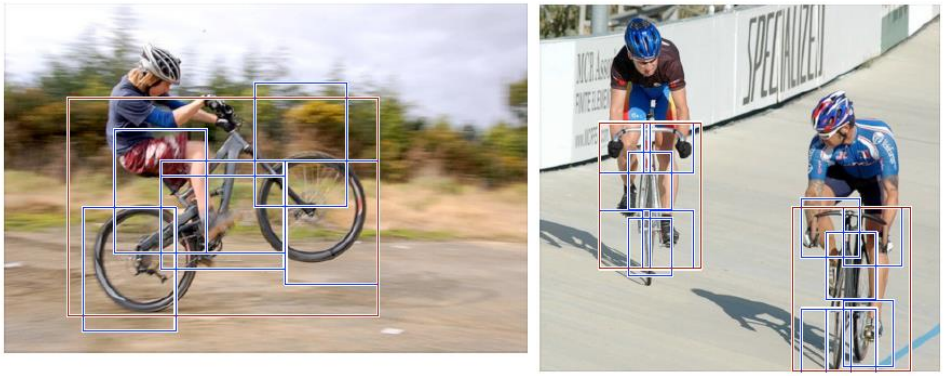


Images from Felzenszwalb

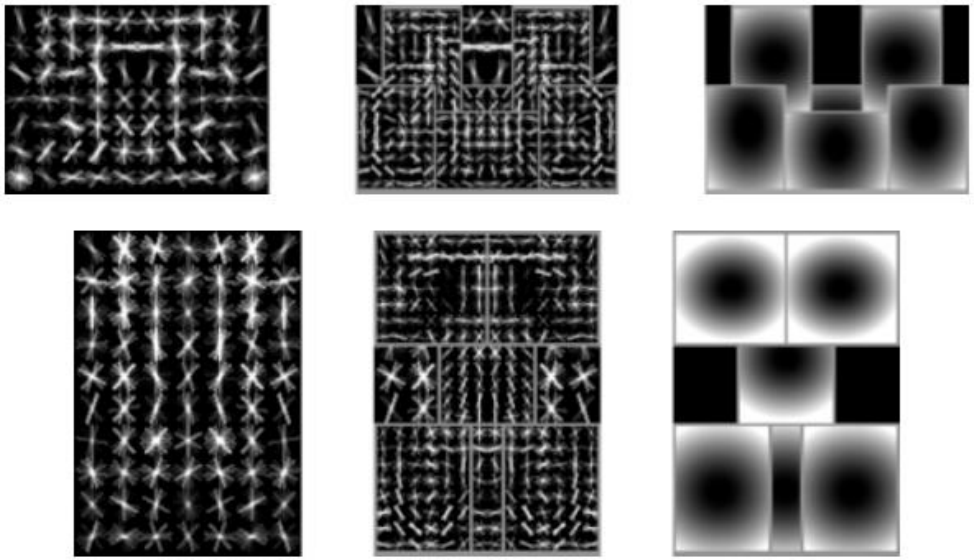
Specifying an object model

3. Hybrid template/parts model

Detections



Template Visualization



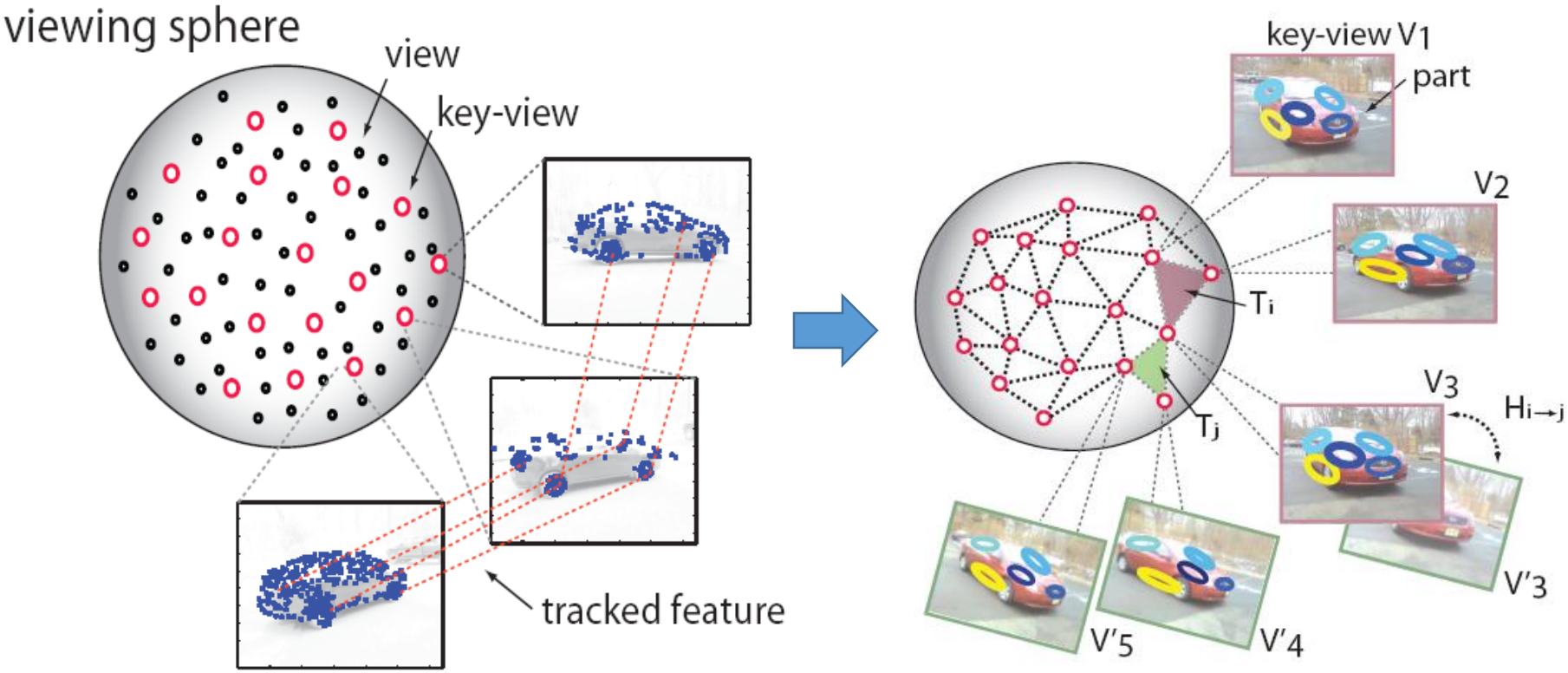
root filters coarse resolution | part filters finer resolution | deformation models

Felzenszwalb et al. 2008

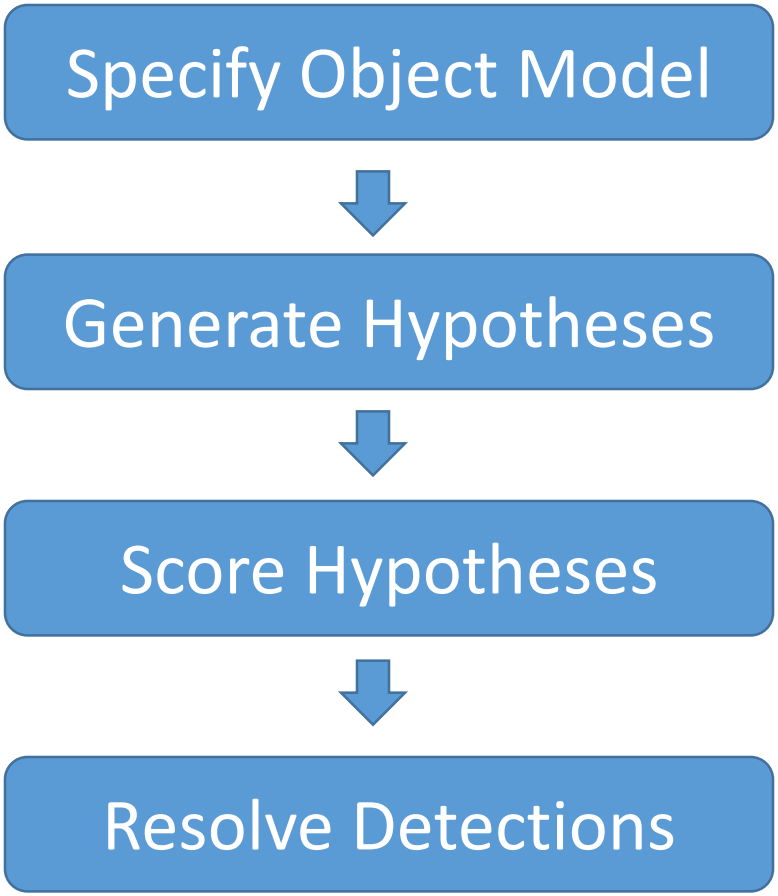
Specifying an object model

4. 3D-ish model

- Object is collection of 3D planar patches under affine transformation



General Process of Object Recognition



Propose an alignment of the model to the image

Generating hypotheses

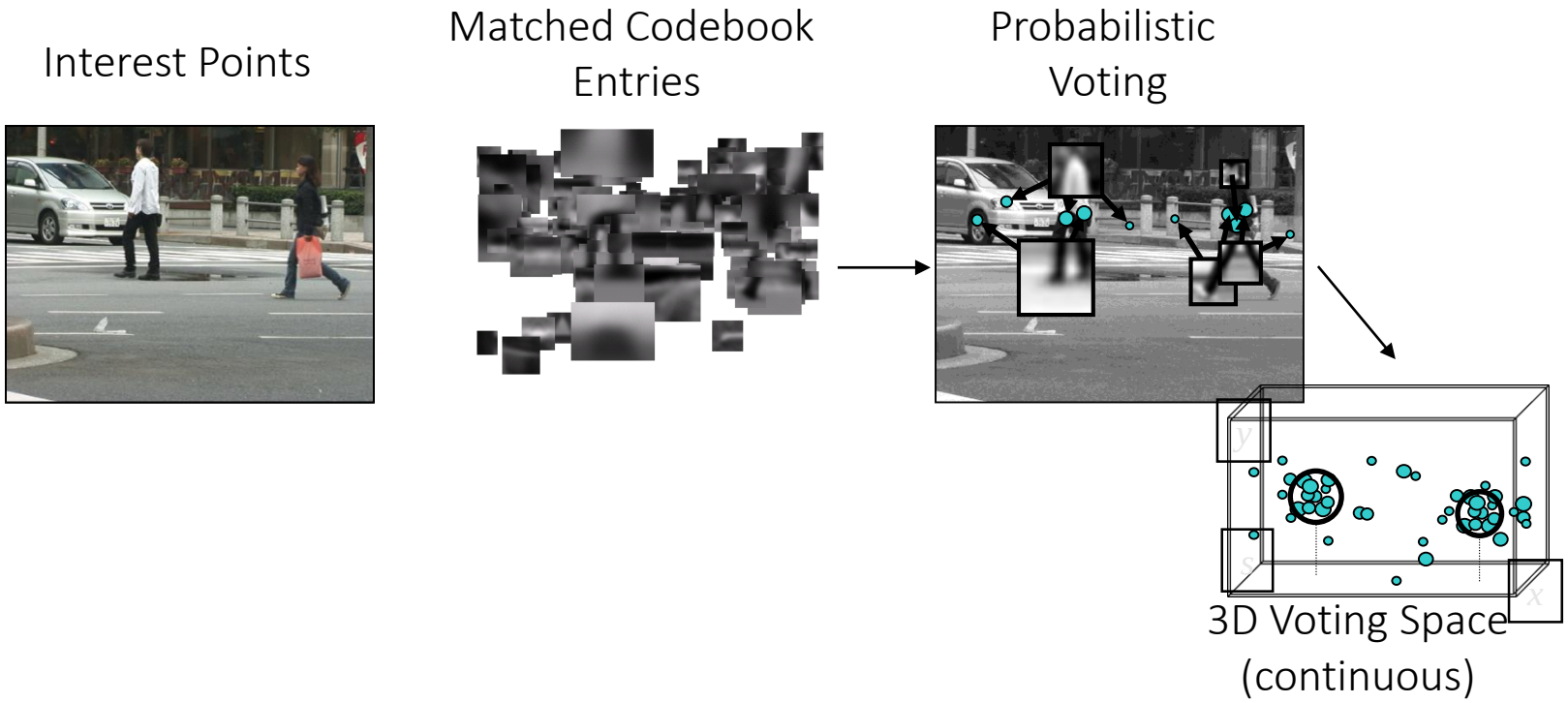
1. Sliding window
   - Test patch at each location and scale

Generating hypotheses

1. Sliding window
   - Test patch at each location and scale

Generating hypotheses

2. Voting from patches/keypoints



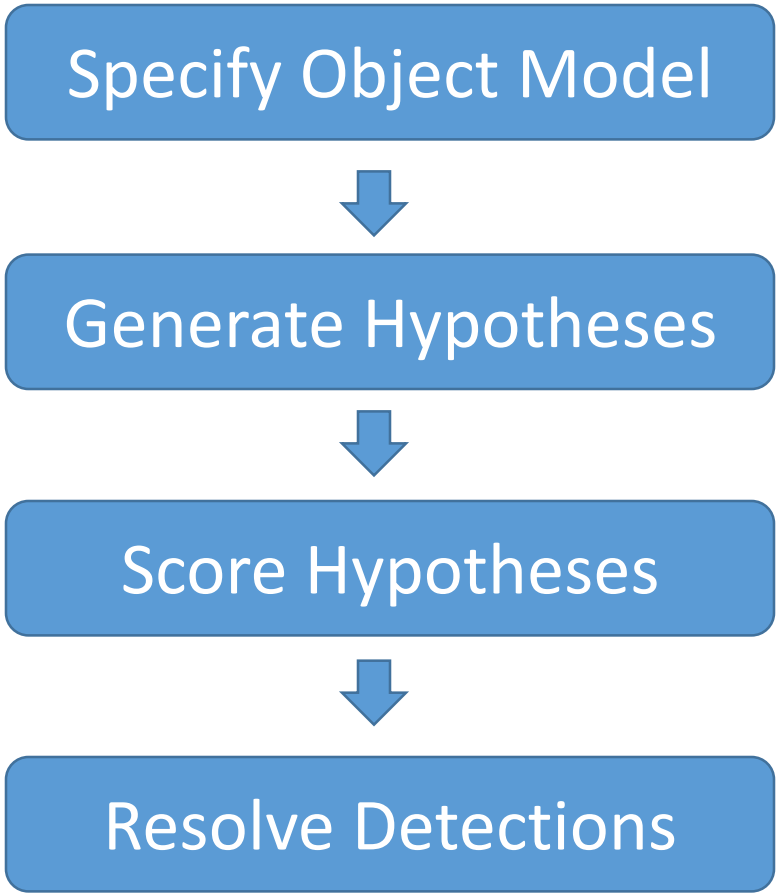
ISM model by Leibe et al.

Generating hypotheses

3. Region-based proposal

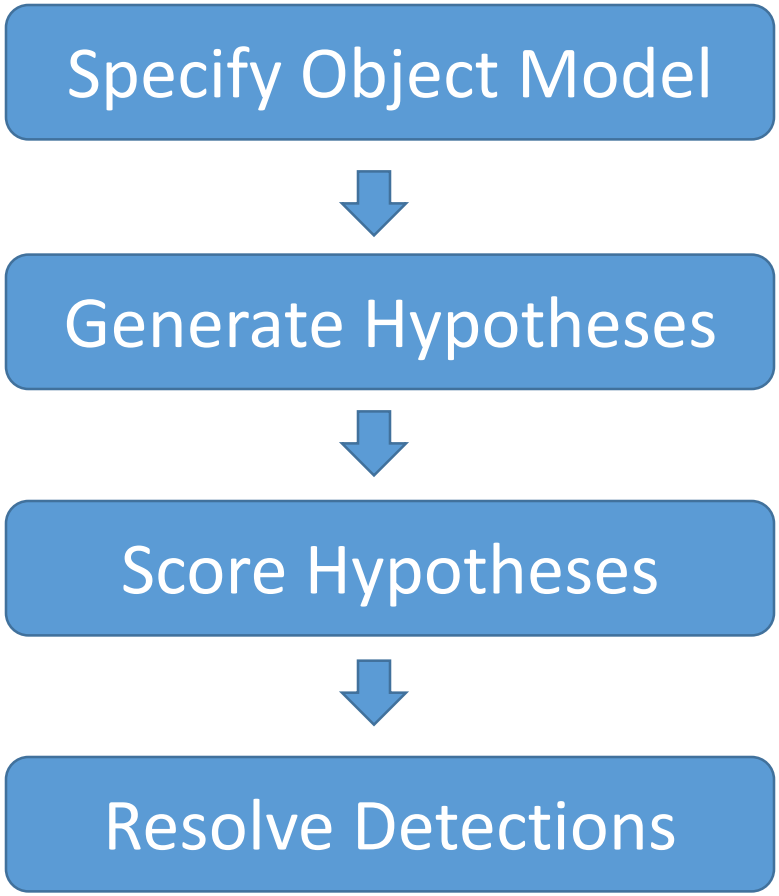
Endres Hoiem 2010

General Process of Object Recognition



Mainly-gradient based or CNN features, usually based on summary representation, many classifiers

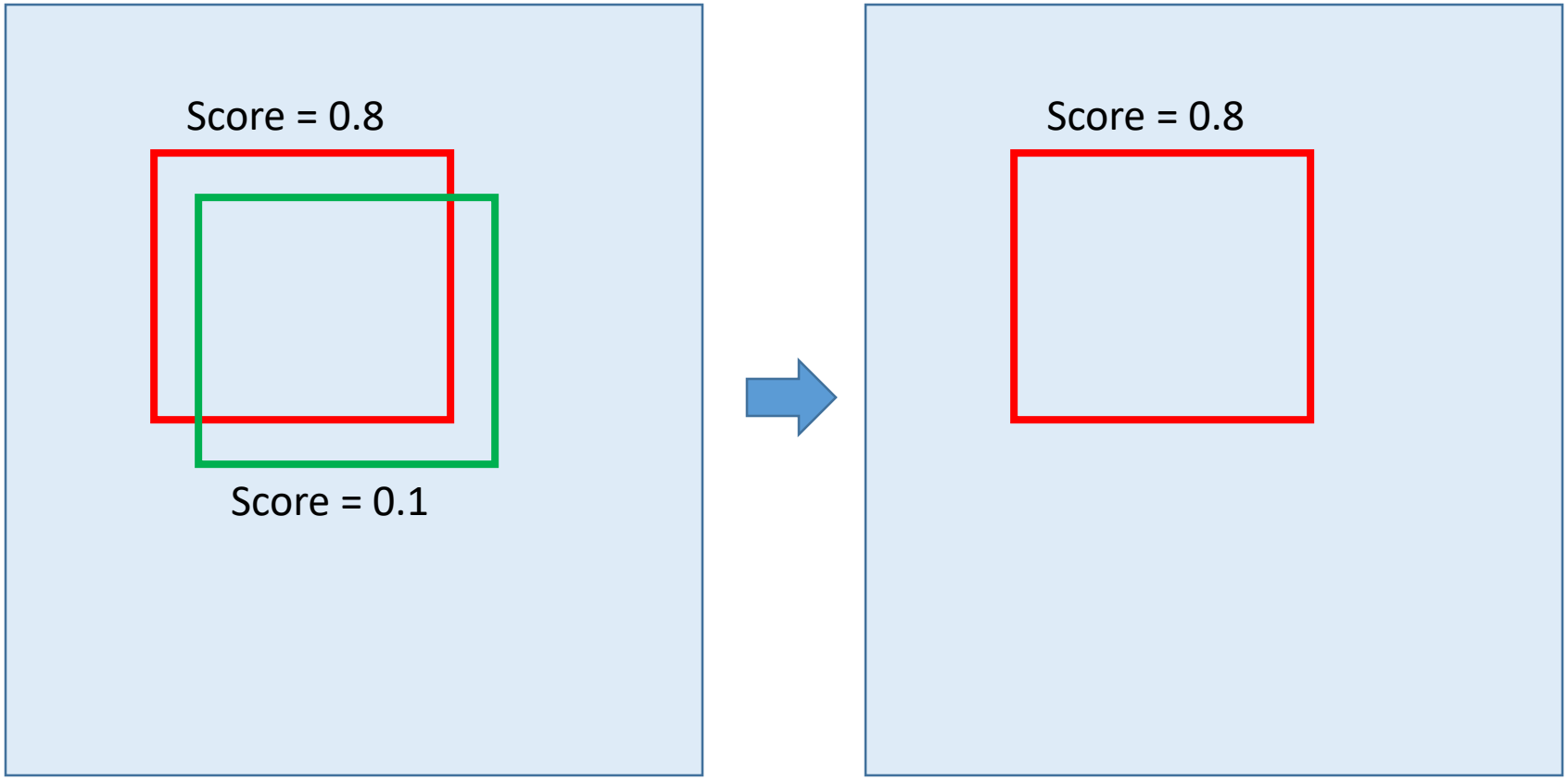
General Process of Object Recognition



Rescore each proposed object based on whole set

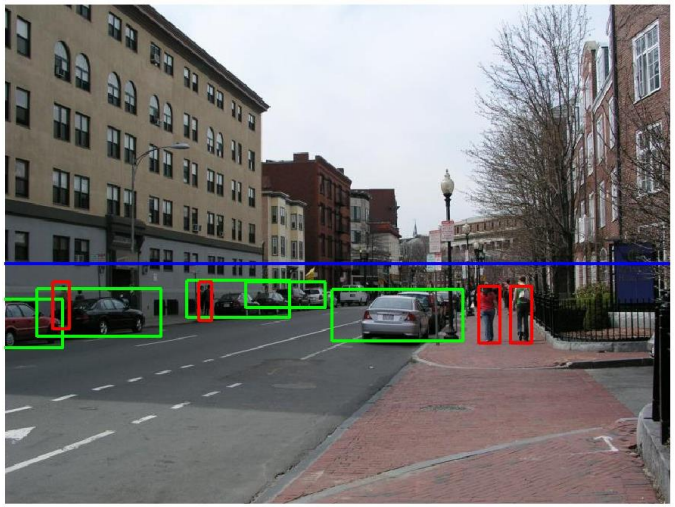
Resolving detection scores

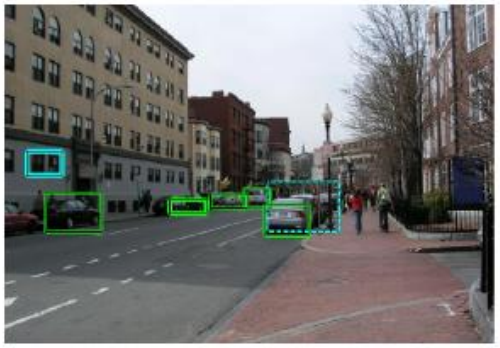
1. Non-max suppression



Resolving detection scores

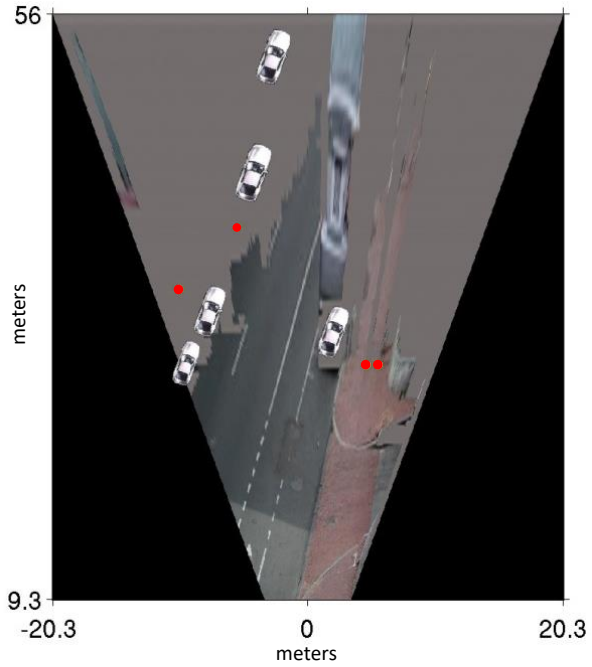
2. Context/reasoning





(g) Car Detections: Local

(h) Ped Detections: Local



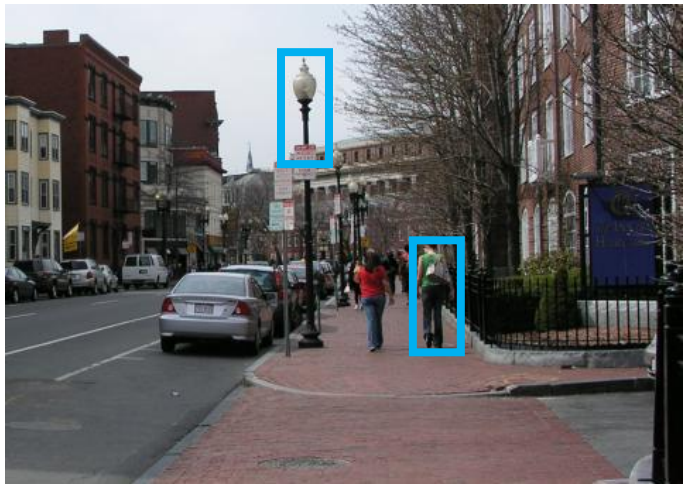
Hoiem et al. 2006

Object category detection in computer vision

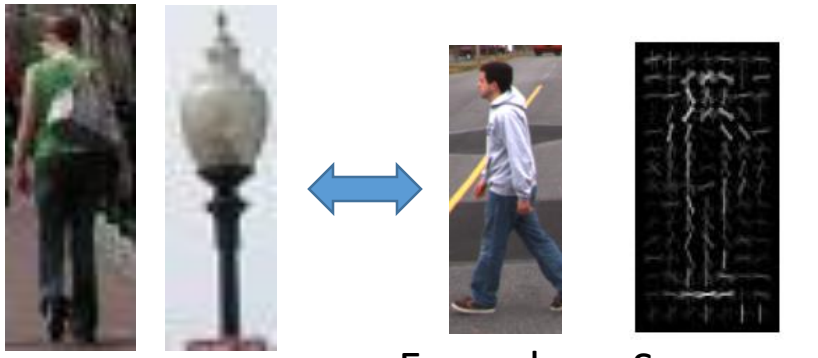
Goal: detect all pedestrians, cars, monkeys, etc in image

Basic Steps of Category Detection

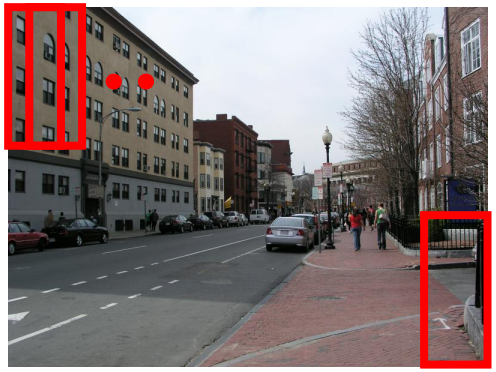
1. Align
   - E.g., choose position, scale orientation
   - How to make this tractable?



2. Compare
   - Compute similarity to an example object or to a summary representation
   - Which differences in appearance are important?



Sliding window: a simple alignment solution



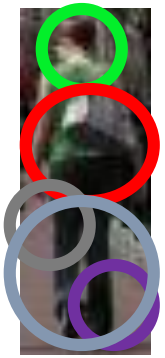
Each window is separately classified

Statistical Template

- Object model = sum of scores of features at fixed positions

+3 +2 -2 -1 -2.5 = -0.5 ? > 7.5

Non-object



+4 +1 +0.5 +3 +0.5 = 10.5 ? > 7.5

Object

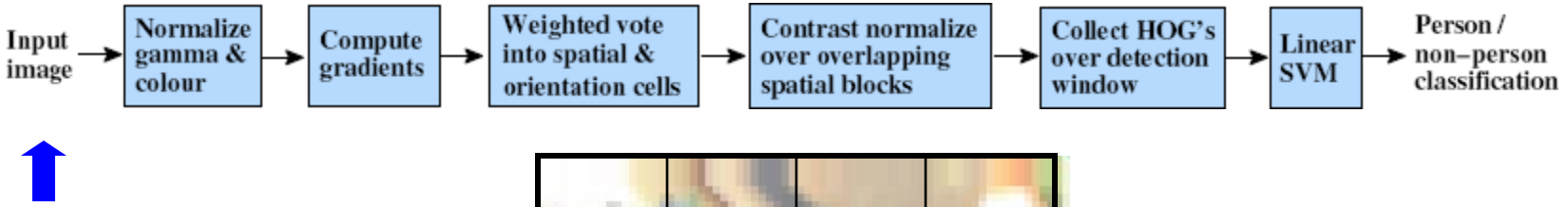
Design challenges

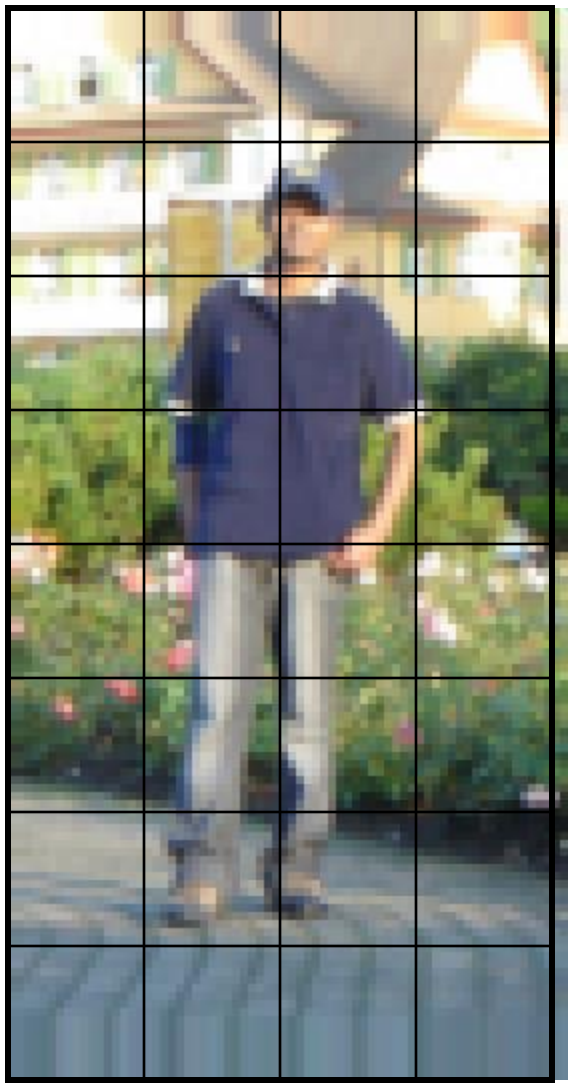
- How to efficiently search for likely objects
  - Even simple models require searching hundreds of thousands of positions and scales
- Feature design and scoring
  - How should appearance be modeled? What features correspond to the object?
- How to deal with different viewpoints?
  - Often train different models for a few different viewpoints
- Implementation details
  - Window size
  - Aspect ratio
  - Translation/scale step size
  - Non-maxima suppression

Example: Dalal-Triggs pedestrian detector

1. Extract fixed-sized (64x128 pixel) window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

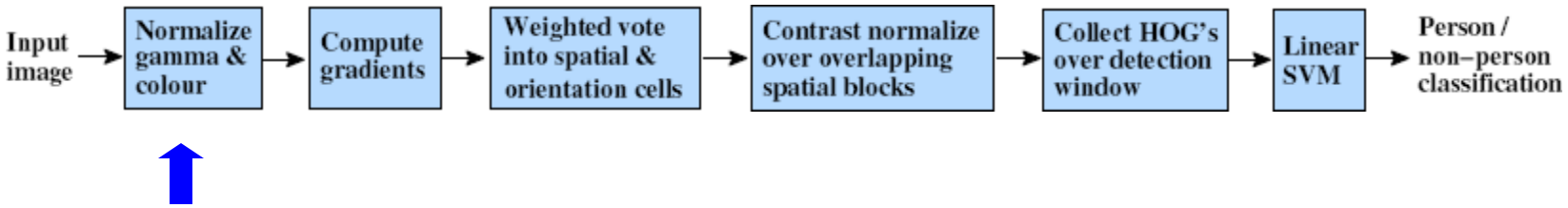
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





Slides by Pete Barnum

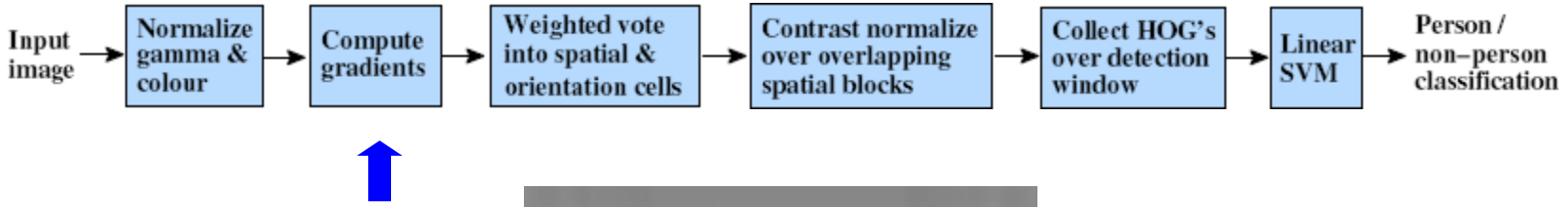
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



- Tested with
  - RGB
  - LAB
  - Grayscale

  } Slightly better performance vs. grayscale
- Gamma Normalization and Compression
  - Square root
  - Log

  } Very slightly better performance vs. no adjustment



Outperforms

-1	0	1

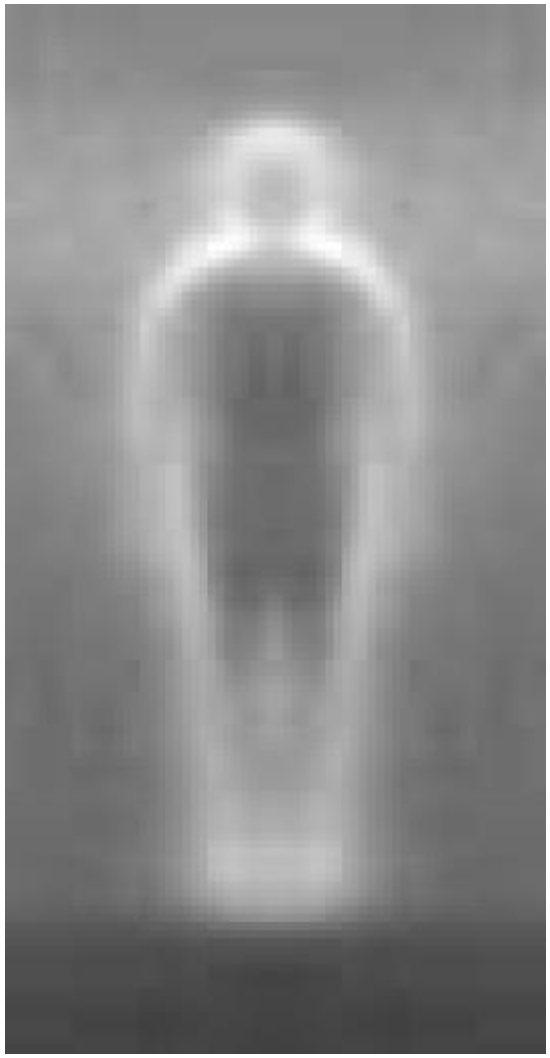
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



0	1
-1	0

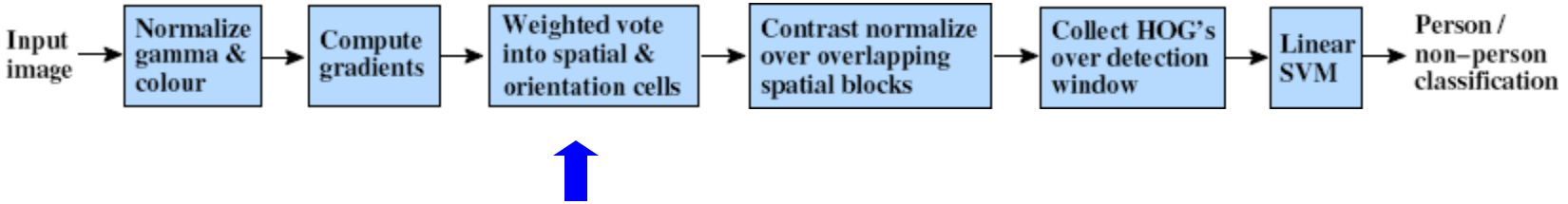
diagonal

-1	0	1
-2	0	2
-1	0	1

Sobel

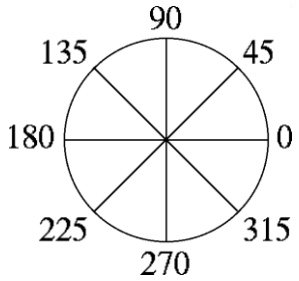
Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

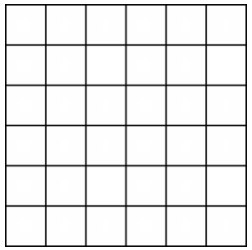


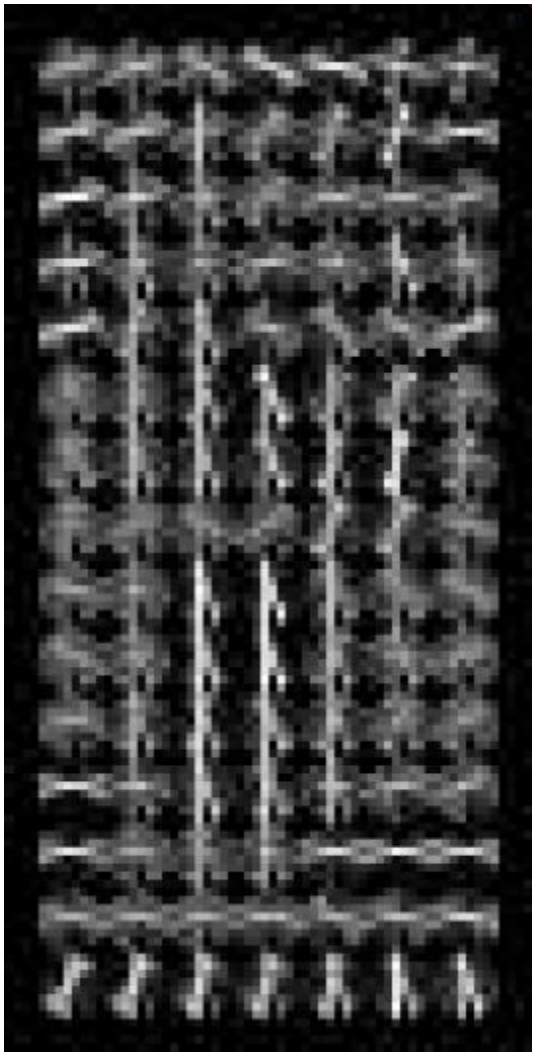
- Histogram of gradient orientations

Orientation: 9 bins (for unsigned angles)



Histograms in 8x8 pixel cells

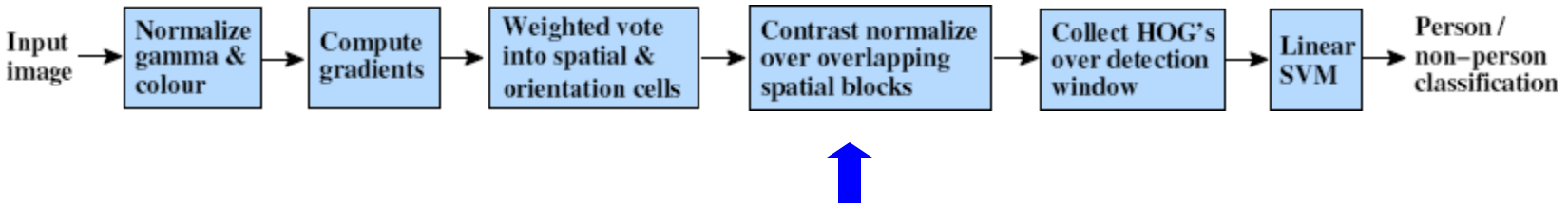




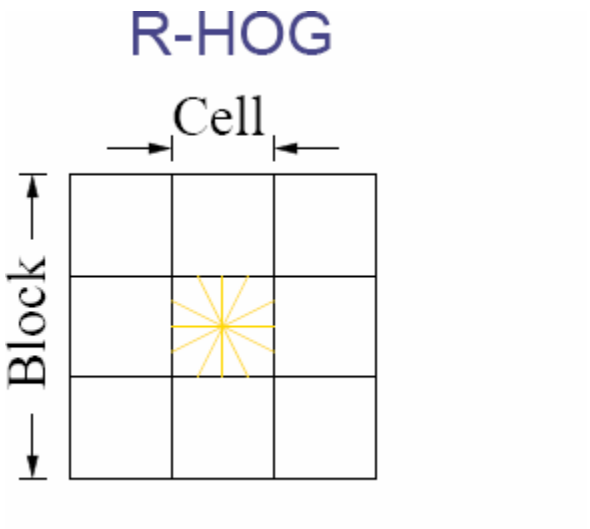
- Votes weighted by magnitude
- Bilinear interpolation between cells

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



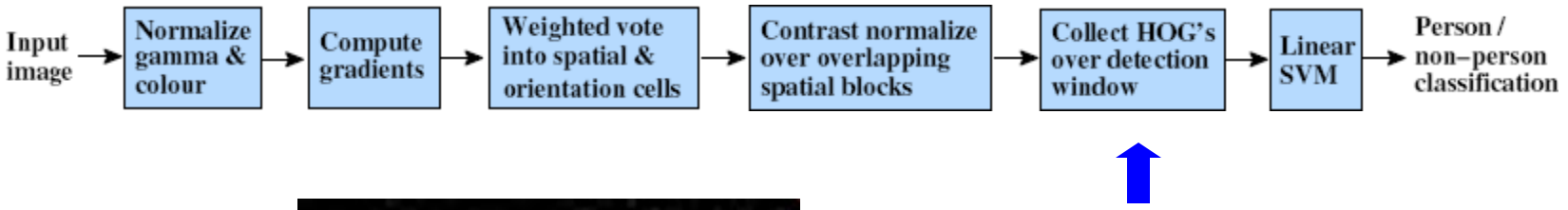
Normalize with respect to surrounding cells



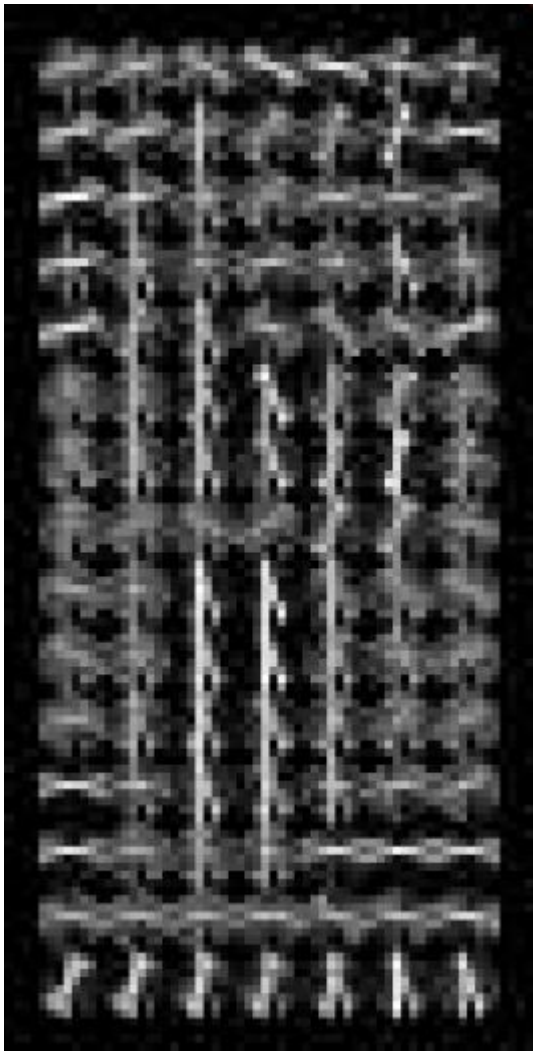
$$L2-norm: v \longrightarrow v/\sqrt{||v||_2^2+\epsilon^2}$$

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



X=



orientations

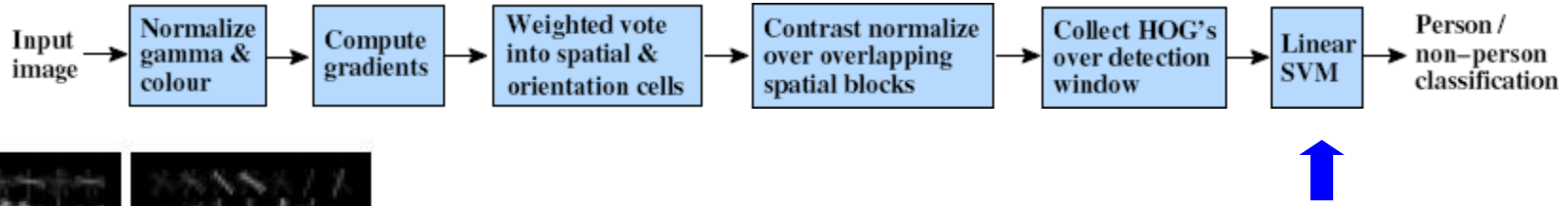
features = 15 x 7 x 9 x 4 = 3780

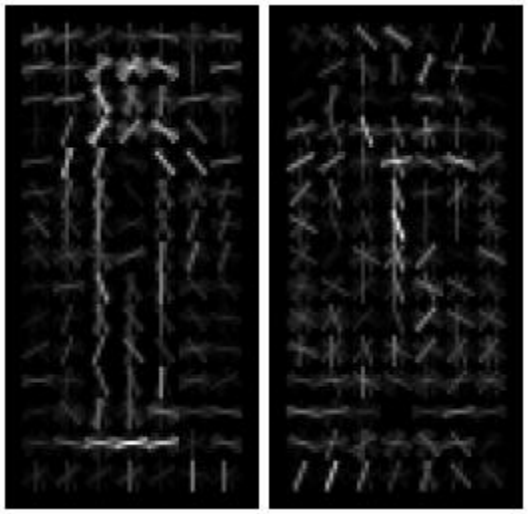
cells

normalizations by neighboring cells

Slides by Pete Barnum

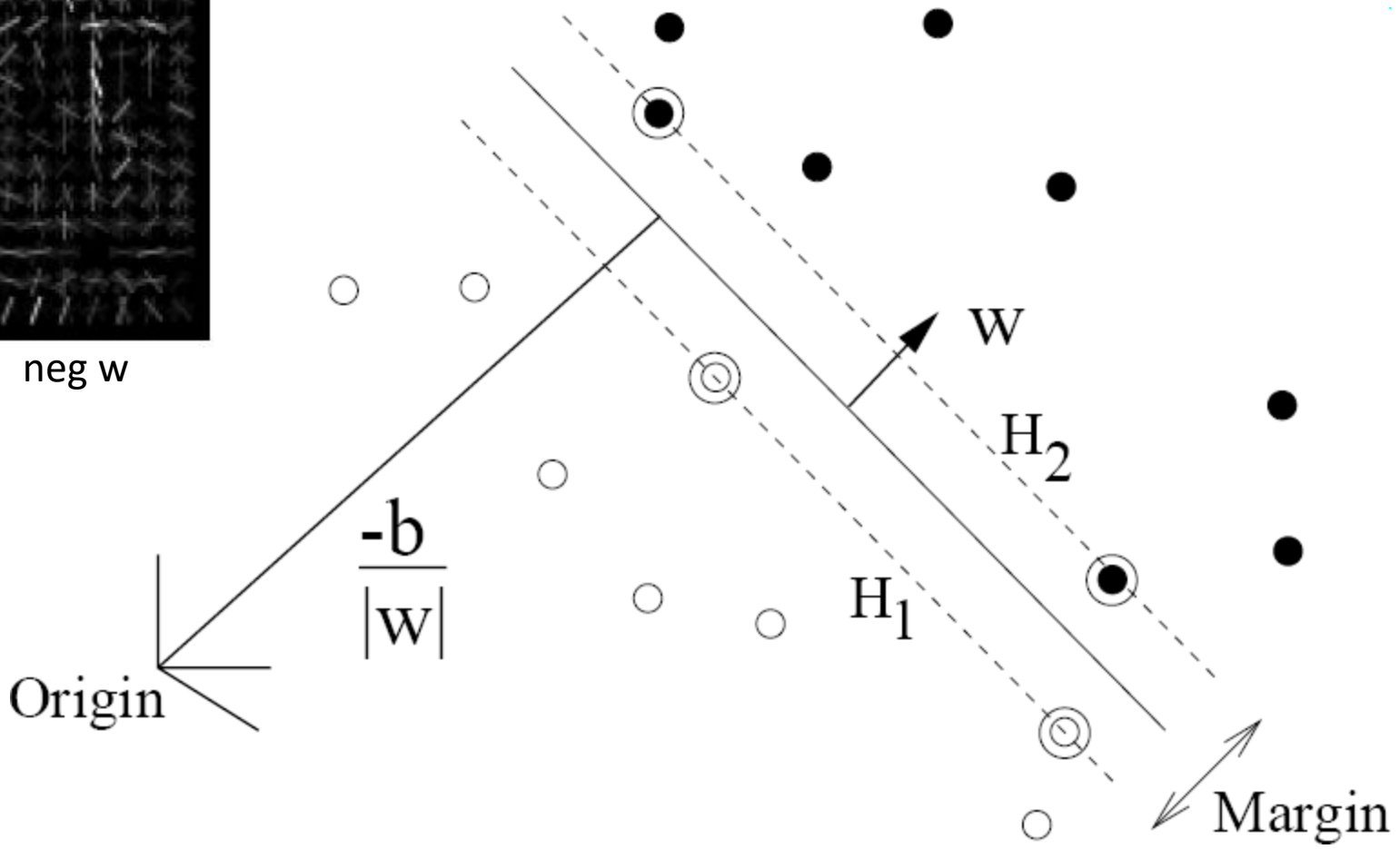
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





pos w

neg w



Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

$$0.16 = w^T x - b$$

$$sign(0.16) = 1$$

$$=>$$ pedestrian

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Detection examples

Something to think about…

- Sliding window detectors work
  - *very well* for faces
  - *fairly well* for cars and pedestrians
  - *badly* for cats and dogs

- Why are some classes easier than others?

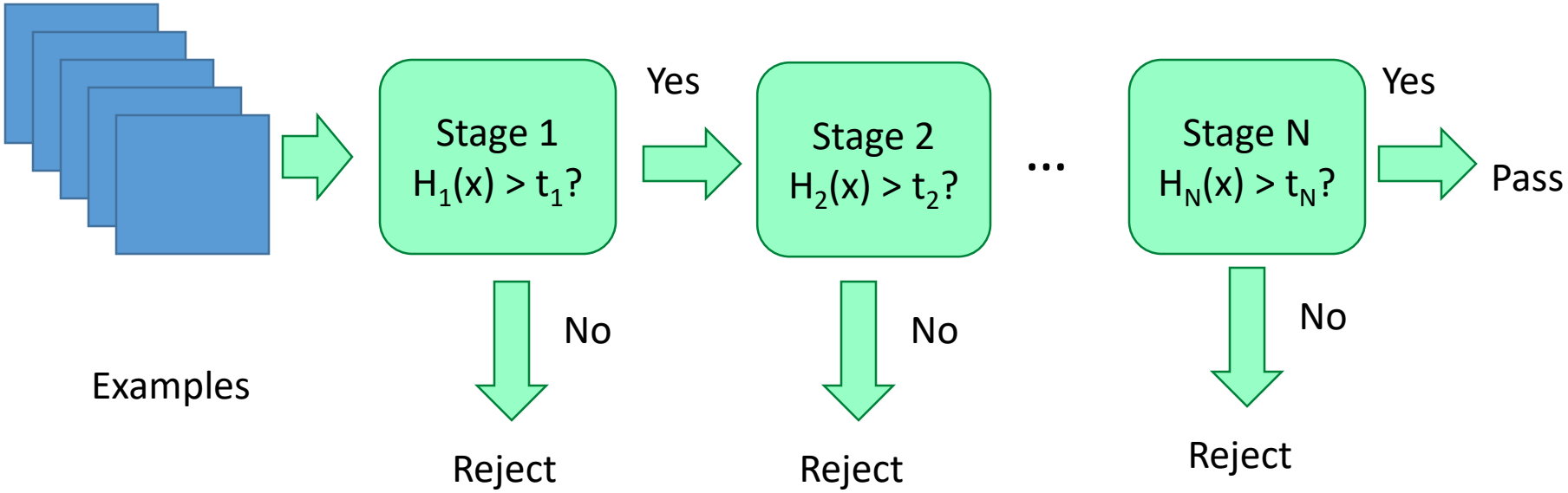
Viola-Jones sliding window detector

Fast detection through two mechanisms

- Quickly eliminate unlikely windows
- Use features that are fast to compute

Viola and Jones. Rapid Object Detection using a Boosted Cascade of Simple Features (2001).

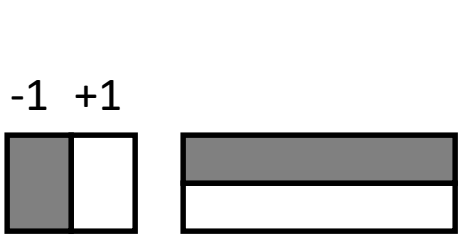
Cascade for Fast Detection



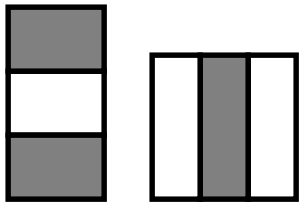
- Choose threshold for low false negative rate
- Fast classifiers early in cascade
- Slow classifiers later, but most examples don't get there

Features that are fast to compute

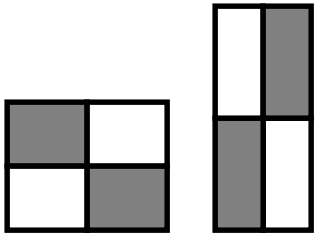
- "Haar-like features"
  - Differences of sums of intensity
  - Thousands, computed at various positions and scales within detection window



Two-rectangle features



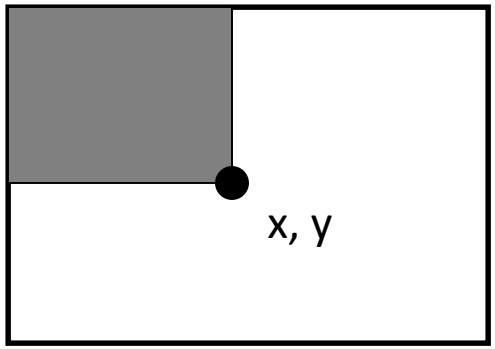
Three-rectangle features



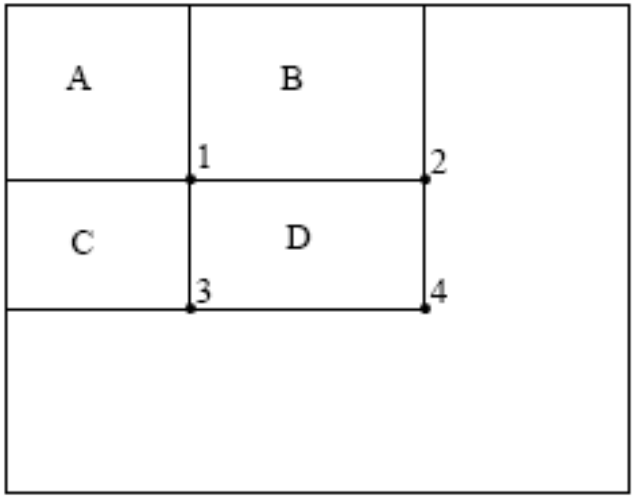
Etc.

Integral Images

- `ii = cumsum(cumsum(im, 1), 2)`



ii(x,y) = Sum of the values in the grey region



How to compute B-A?

How to compute A+D-B-C?

Feature selection with Adaboost

- Create a large pool of features (180K)
- Select features that are discriminative and work well together
  - "Weak learner" = feature + threshold + parity

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

  - Choose weak learner that minimizes error on the weighted training set
  - Reweight

Adaboost

- 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:

     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

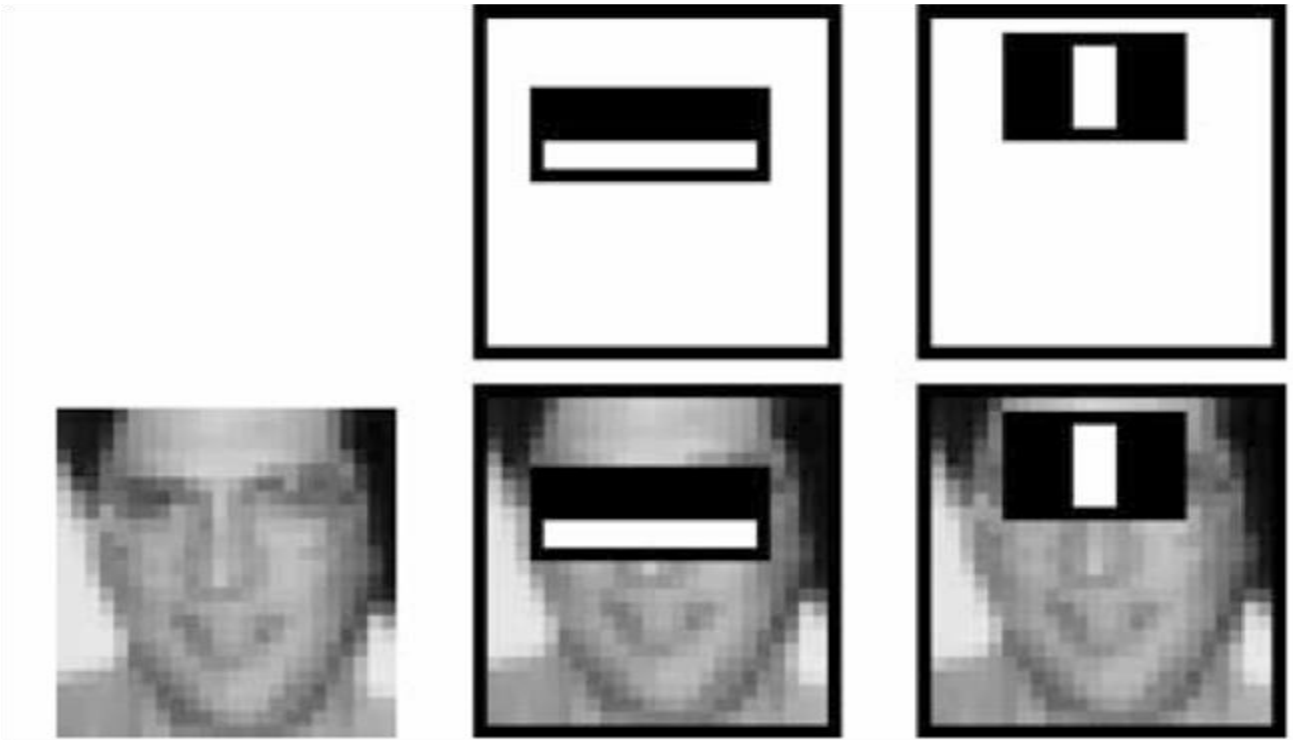
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

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

Top 2 selected features

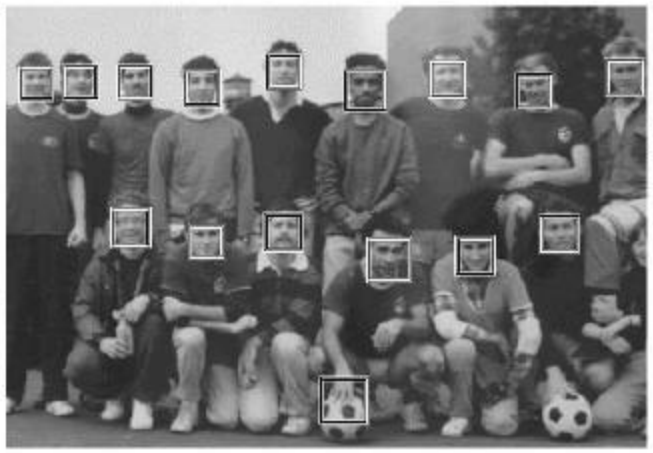


Viola-Jones details

- 38 stages with 1, 10, 25, 50 … features
  - 6061 total used out of 180K candidates
  - 10 features evaluated on average
- Training Examples
  - 4916 positive examples
  - 10000 negative examples collected after each stage
- Scanning
  - Scale detector rather than image
  - Scale steps = 1.25 (factor between two consecutive scales)
  - Translation 1*scale (# pixels between two consecutive windows)
- Non-max suppression: average coordinates of overlapping boxes
- Train 3 classifiers and take vote

Viola Jones Results

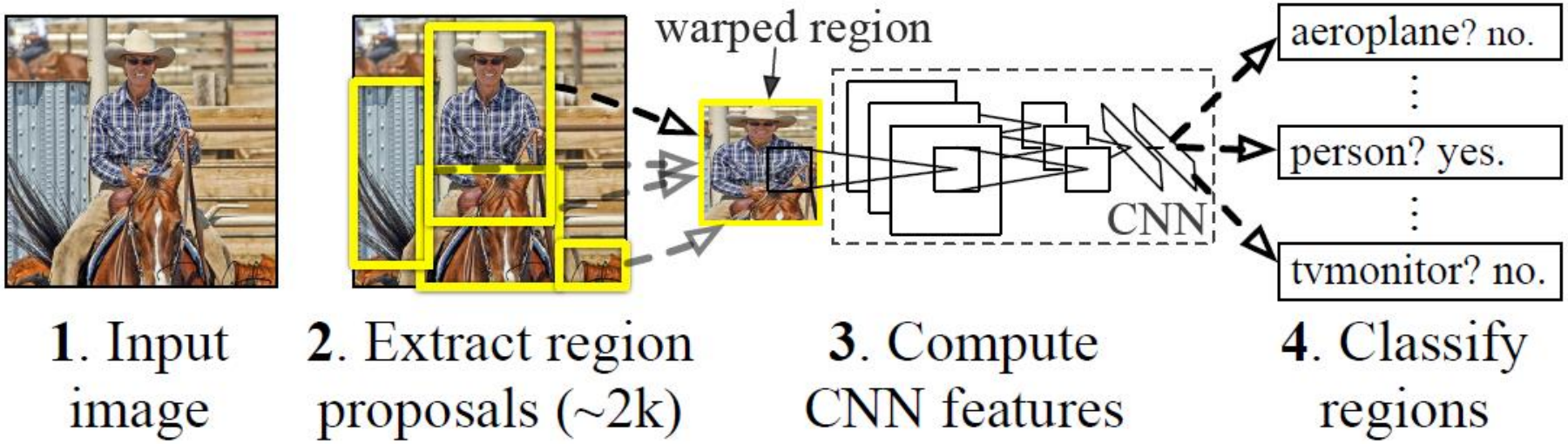
Speed = 15 FPS (in 2001)


Detector \ False detections	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2 %	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

MIT + CMU face dataset

R-CNN (Girshick et al. CVPR 2014)



- Replace sliding windows with "selective search" region proposals (Uijilings et al. IJCV 2013)
- Extract rectangles around regions and resize to 227x227
- Extract features with fine-tuned CNN (that was initialized with network trained on ImageNet before training)
- Classify last layer of network features with SVM

http://arxiv.org/pdf/1311.2524.pdf

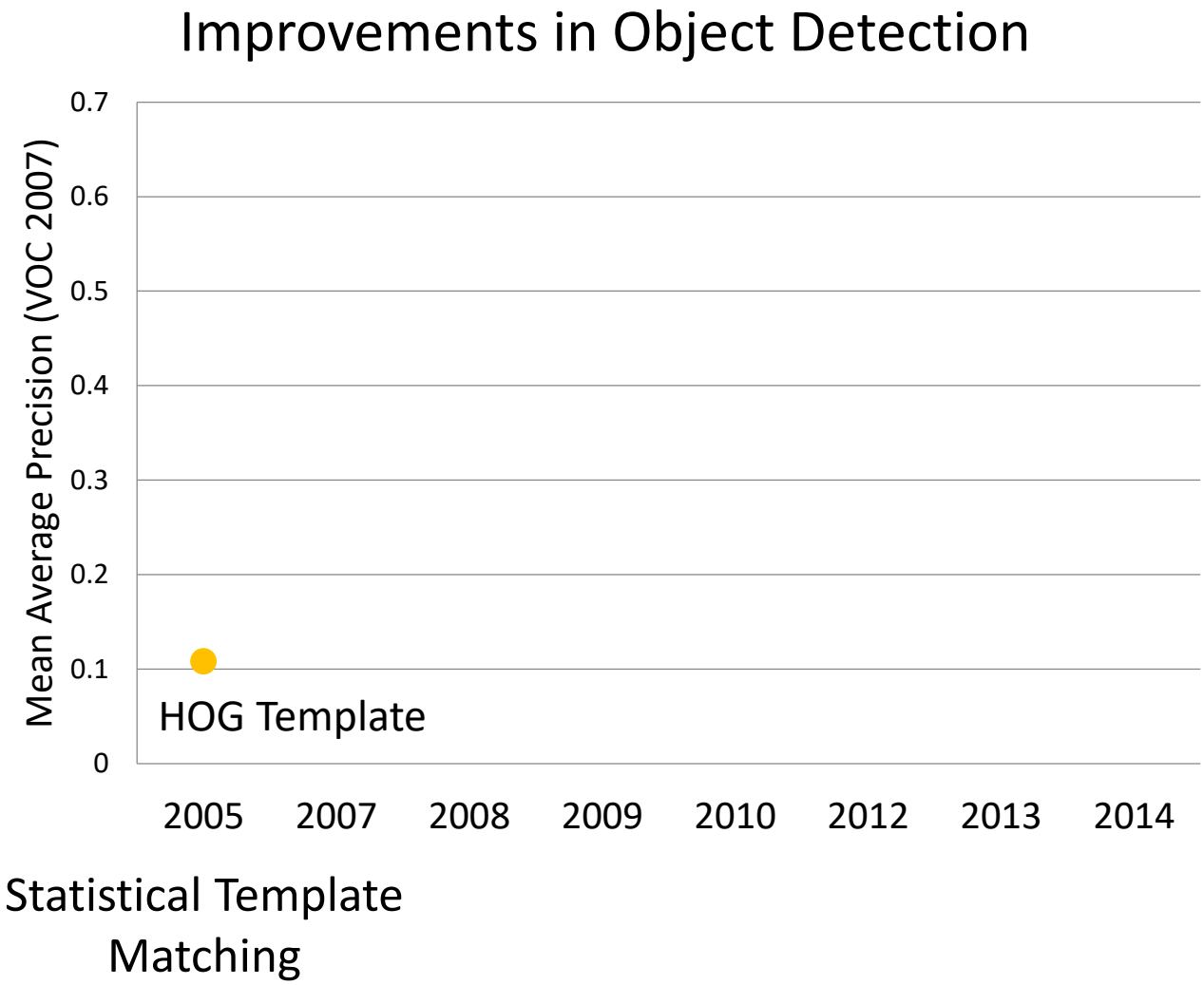
Sliding window vs. region proposals

Sliding window

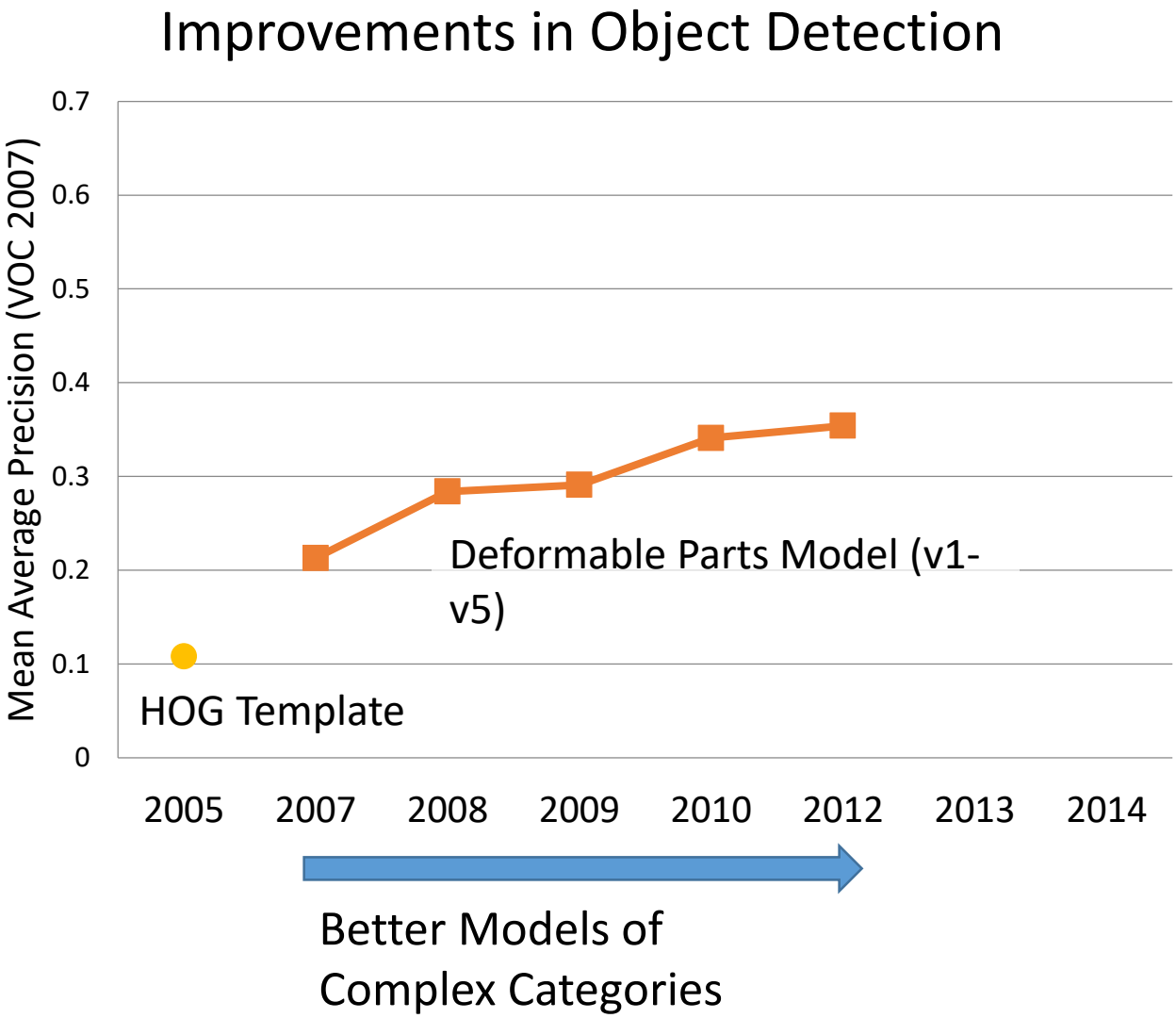
- Comprehensive search over position, scale (sometimes aspect, though expensive)
- Typically 100K candidates
- Simple
- Speed boost through convolution often possible
- Repeatable
- Even with many candidates, may not be a good fit to object

Region proposals

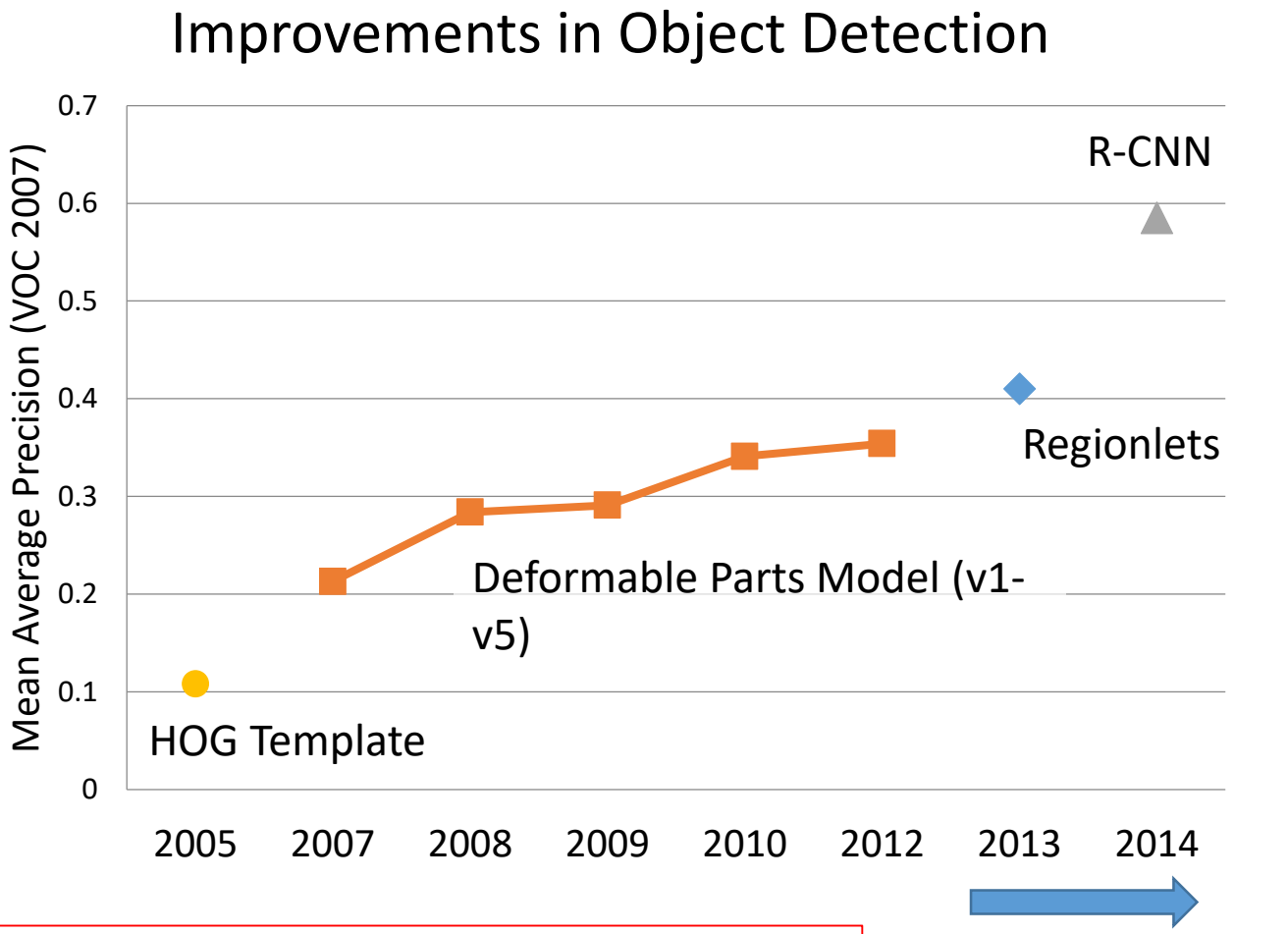
- Search over regions guided by image contours/patterns with varying aspect/size
- Typically 2-10K candidates
- Random (not repeatable)
- Requires a preprocess (currently 1-5s)
- Often requires resizing patch to fit fixed size
- More likely to provide candidates with very good object fit



HOG: Dalal-Triggs 2005



HOG: Dalal-Triggs 2005 DPM: Felzenszwalb et al. 2008-2012

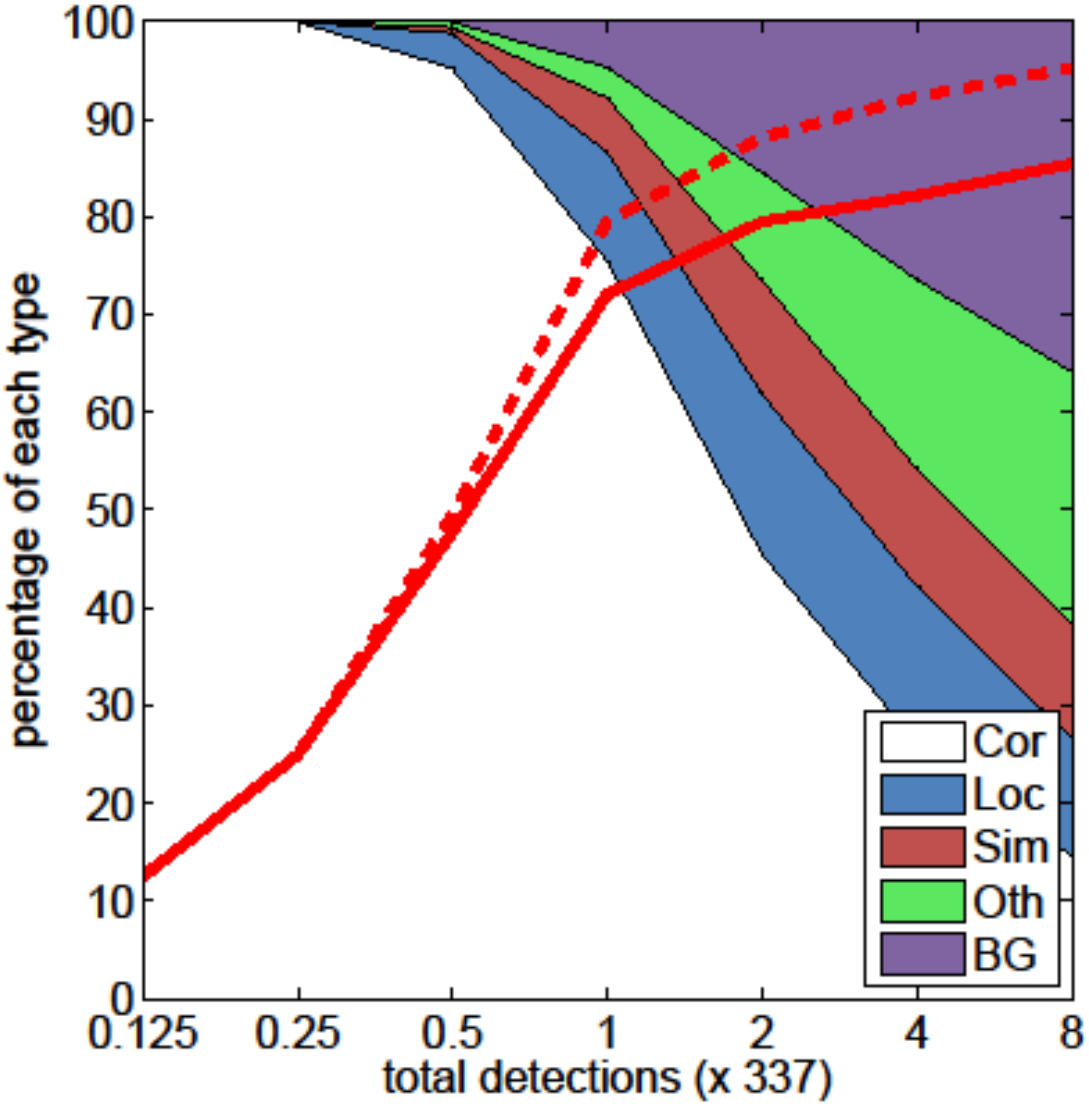


Key Advance: Learn effective features from massive amounts of labeled data *and* adapt to new tasks with less data

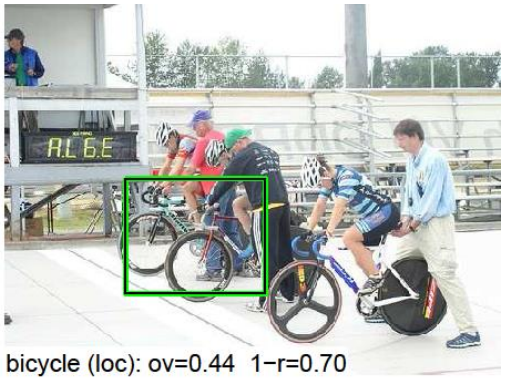
HOG: Dalal-Triggs 2005 DPM: Felzenszwalb et al. 2008-2012 Regionlets: Wang et al. 2013 R-CNN: Girshick et al. 2014

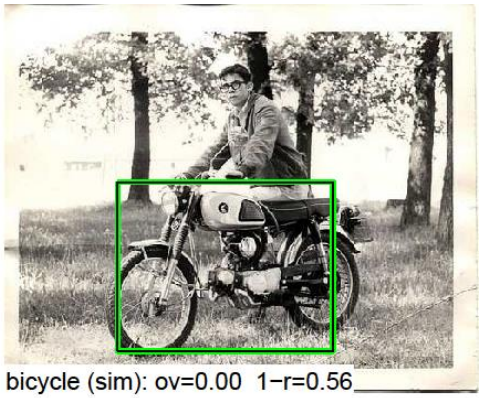
Mistakes are often reasonable

Bicycle: AP = 0.73



Confident Mistakes

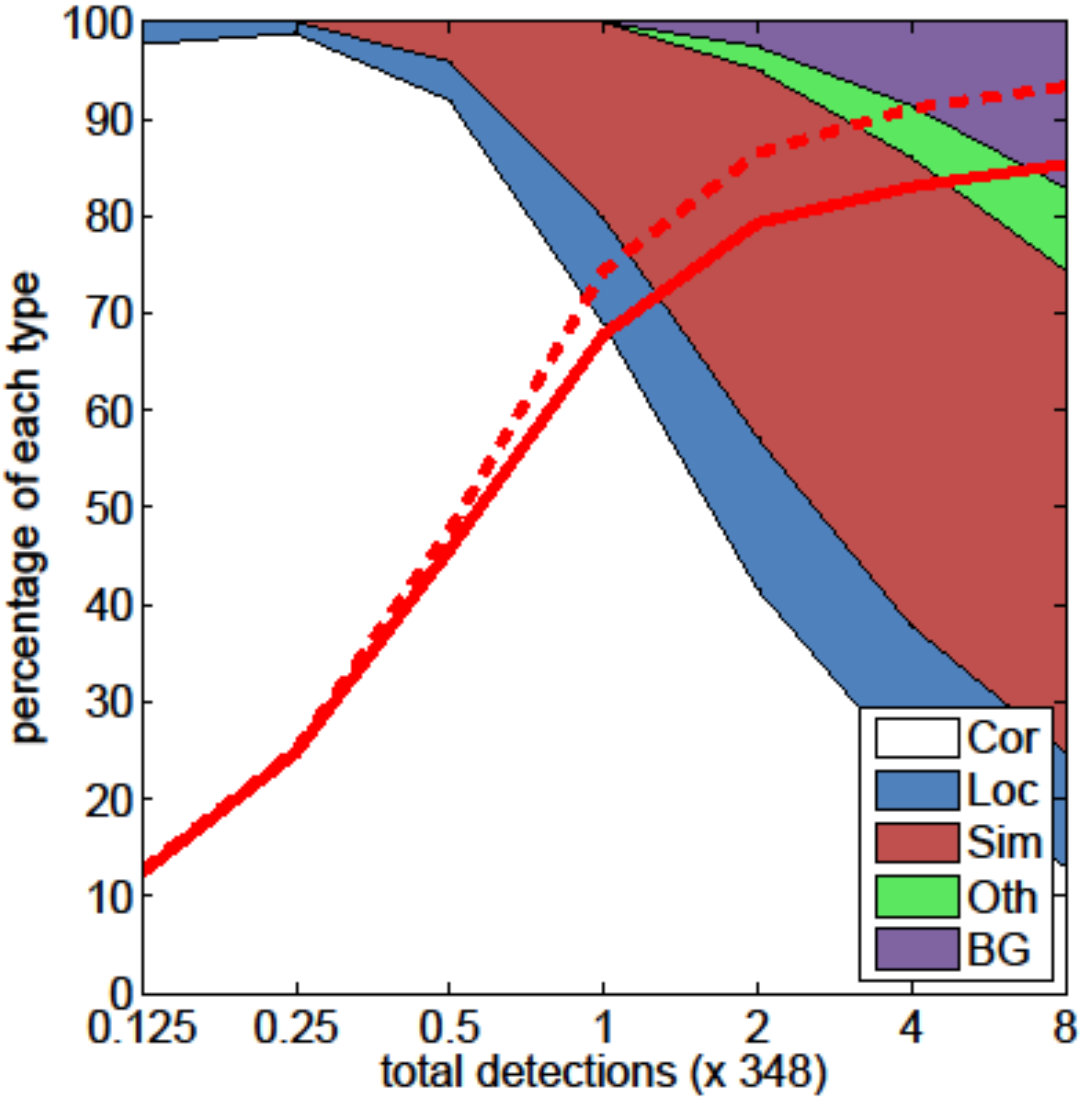




R-CNN results

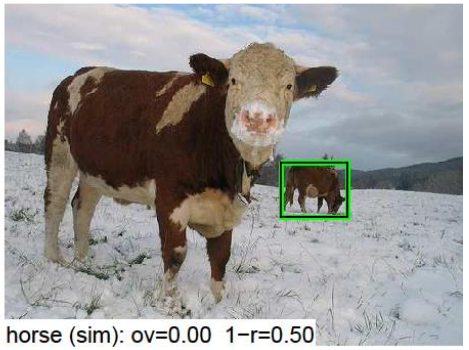
Mistakes are often reasonable

Horse: AP = 0.69

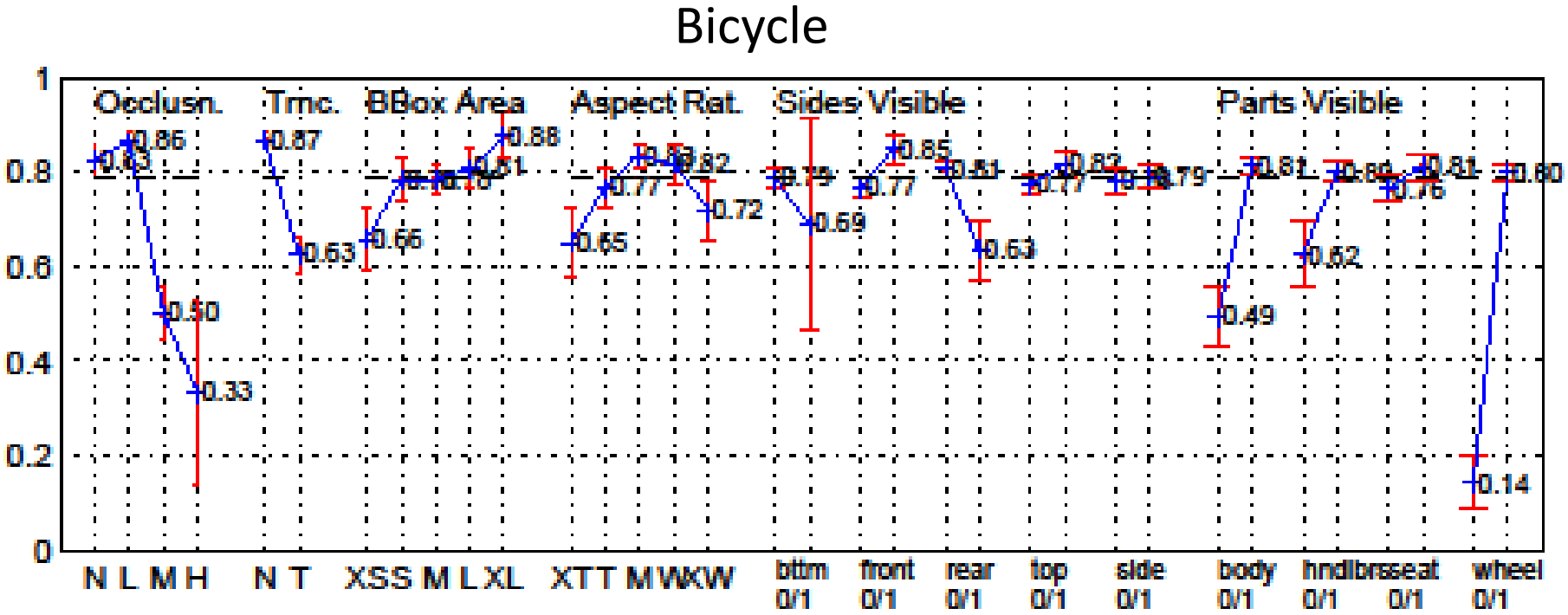


R-CNN results

Confident Mistakes



Misses are often predictable



Small objects, distinctive parts absent or occluded, unusual views

R-CNN results

Strengths and Weaknesses of Statistical Template Approach

Strengths

- Works very well for non-deformable objects: faces, cars, upright pedestrians
- Fast detection

Weaknesses

- Sliding window has difficulty with deformable objects (proposals works with flexible features works better)
- Not robust to occlusion
- Requires lots of training data

Tricks of the trade

- Details in feature computation really matter
  - E.g., normalization in Dalal-Triggs improves detection rate by 27% at fixed false positive rate
- Template size
  - Typical choice for sliding window is size of smallest detectable object
  - For CNNs, typically based on what pretrained features are available
- "Jittering" to create synthetic positive examples
  - Create slightly rotated, translated, scaled, mirrored versions as extra positive examples
- Bootstrapping to get hard negative examples
  1. Randomly sample negative examples
  2. Train detector
  3. Sample negative examples that score > -1
  4. Repeat until all high-scoring negative examples fit in memory

Influential Works in Detection

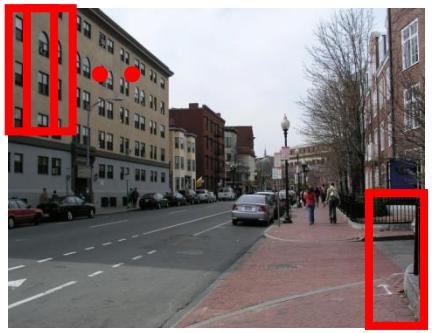
- Sung-Poggio (1994, 1998) : ~2100 citations
  - Basic idea of statistical template detection (I think), bootstrapping to get "face-like" negative examples, multiple whole-face prototypes (in 1994)
- Rowley-Baluja-Kanade (1996-1998) : ~4200
  - "Parts" at fixed position, non-maxima suppression, simple cascade, rotation, pretty good accuracy, fast
- Schneiderman-Kanade (1998-2000,2004) : ~2250
  - Careful feature/classifier engineering, excellent results, cascade
- Viola-Jones (2001, 2004) : ~20,000
  - Haar-like features, Adaboost as feature selection, hyper-cascade, very fast, easy to implement
- Dalal-Triggs (2005) : ~11000
  - Careful feature engineering, excellent results, HOG feature, online code
- Felzenszwalb-Huttenlocher (2000): ~1600
  - Efficient way to solve part-based detectors
- Felzenszwalb-McAllester-Ramanan (2008,2010)? ~4000
  - Excellent template/parts-based blend
- Girshick-Donahue-Darrell-Malik (2014) ~300
  - Region proposals + fine-tuned CNN features (marks significant advance in accuracy over hog-based methods)

Fails in commercial face detection

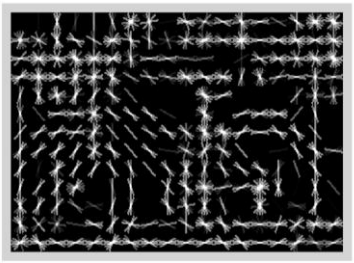
http://www.oddee.com/item_98248.aspx

Summary: statistical templates

Propose Window → Extract Features → Classify → Post-process



Sliding window: scan image pyramid



HOG

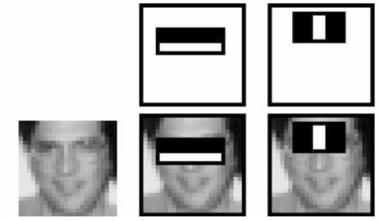
SVM

Boosted stubs

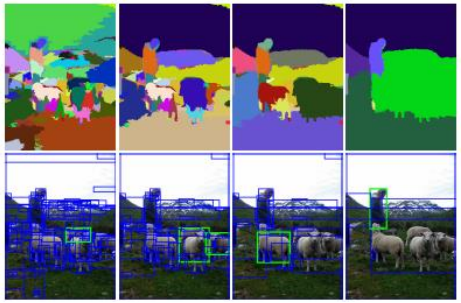
Neural network

Non-max suppression

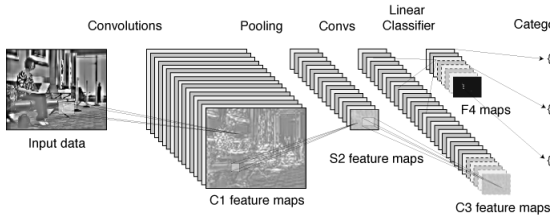
Segment or refine localization



Fast randomized features



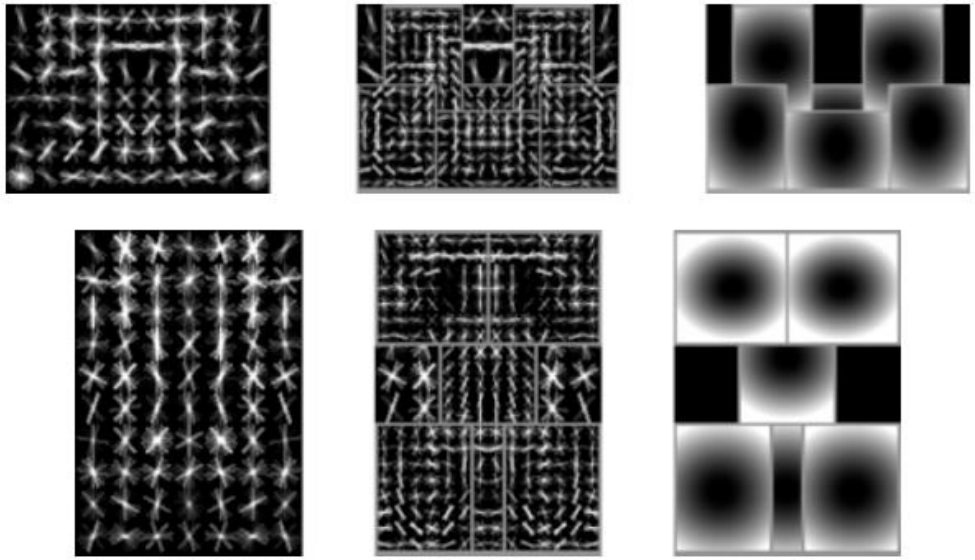
Region proposals: edge/region-based, resize to fixed window



CNN features

Next class

- Part-based models and pose estimation



root filters
coarse resolution

part filters
finer resolution

deformation
models