VISUAL OBJECT DETECTION

Hakan Çevikalp

Eskisehir Osmangazi University

Machine Learning and Computer Vision Laboratory

Slide Credits:

Bill Triggs

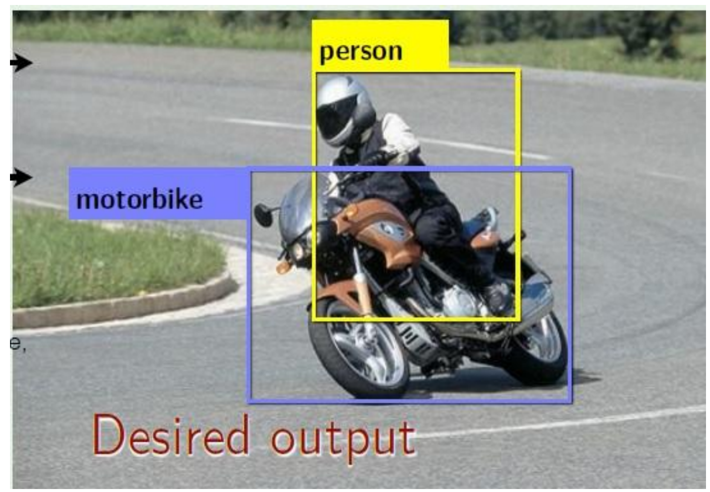
Ross Girshick

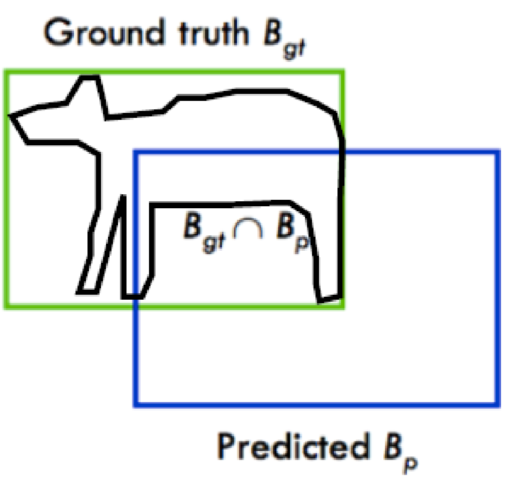
Andrej Karpathy

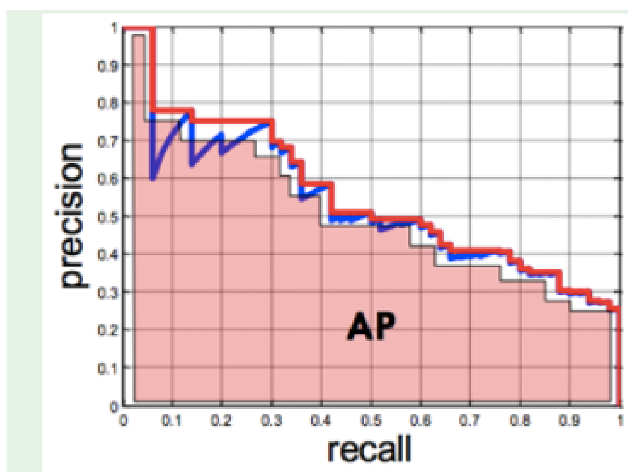
Kaiming He

Míriam Bellver

Object Detection Task







Performance Summary

Average Precision (AP)

0 is the worst, 1 is perfect

Machine Learning Based Object Detectors

- Overcomplete feature set+normalization+rectification

-- templates, Gabor filters, wavelets, edge detectors...

- Learning method can be naive bayes, SVM, perceptron, etc.
- Train on a large set of hand-marked data

-- positives and random negatives

-- bootstrap by adding failures

- To use, scan image at multiple positions, and scales.

Image Scanning Detectors

Detection Phase

Scan image(s) at all scales and locations

↓

Extract features over windows

↓

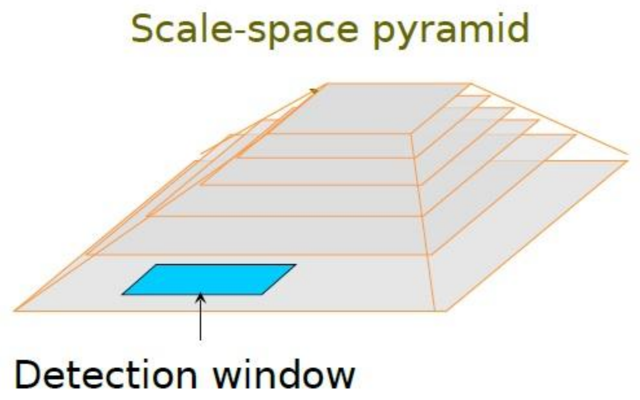
Run window classifier at all locations

↓

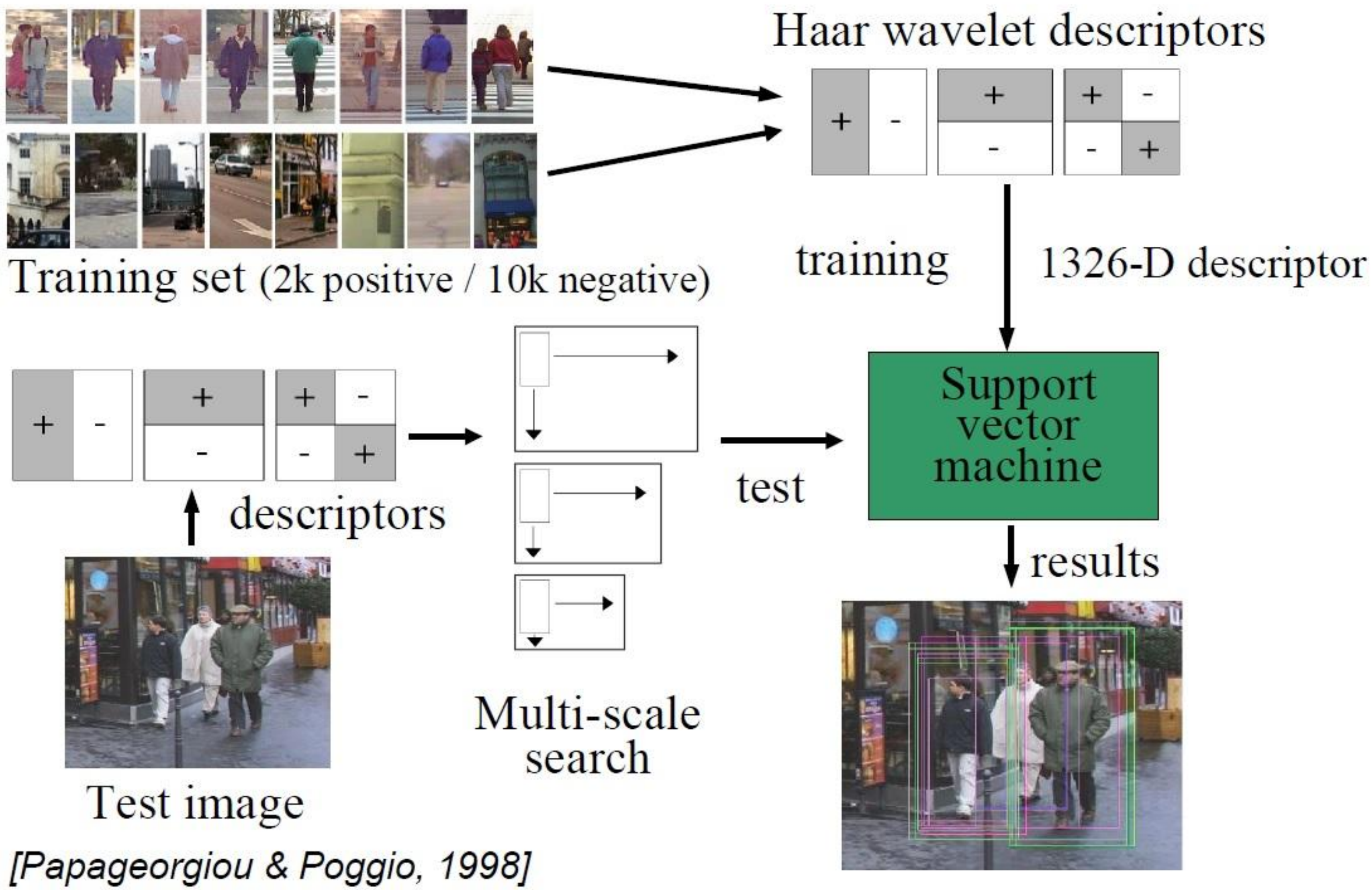
Fuse multiple detections in 3-D position & scale space

↓

Object detections with bounding boxes



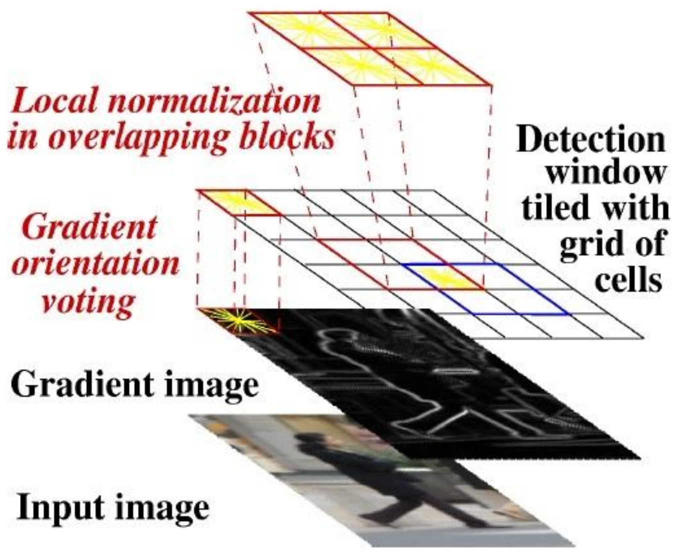
Haar/Wavelet SVM Human Detector



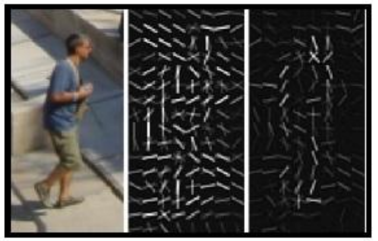
[Papageorgiou & Poggio, 1998]

Histogram of Oriented Gradient Human Detector

- Descriptors are a grid of local Histograms of Oriented Gradients (HOG)
- Linear SVM for runtime efficiency
- Tolerates different poses, clothing, lighting and background
- Assumes upright fully visible people

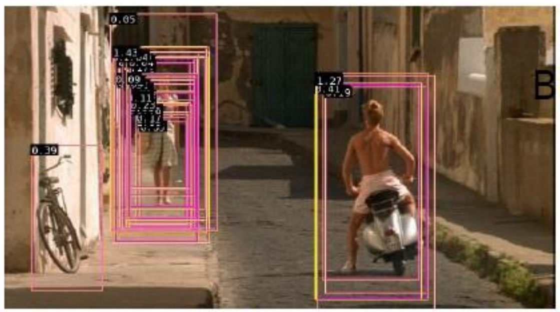


Importance weighted responses



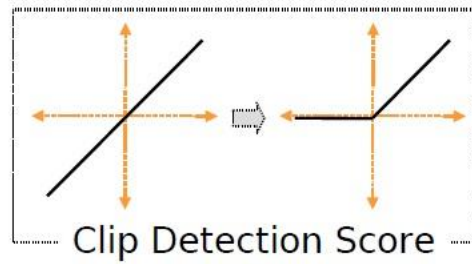
[Dalal & Triggs, CVPR 2005)]

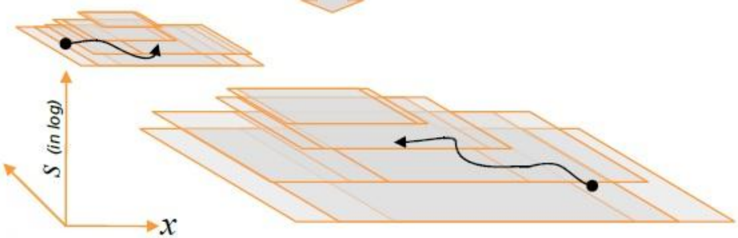
Multi-Scale Object Localization



Multi-scale dense scan of detection window

Bias



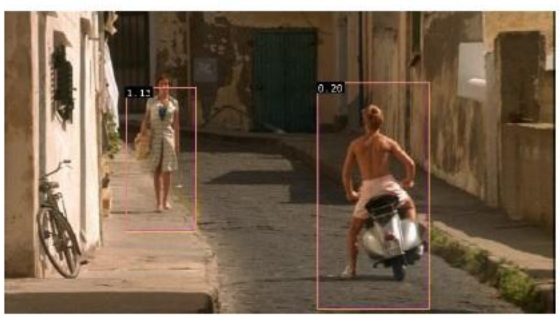


Threshol
d

$$H_i = [\exp(s_i)\,\sigma_x, \exp(s_i)\,\sigma_y, \sigma_s]$$

$$f(x) = \sum_i^n w_i \exp\left(-\|(x - x_i)/H_i^{-1}\|^2/2\right)$$

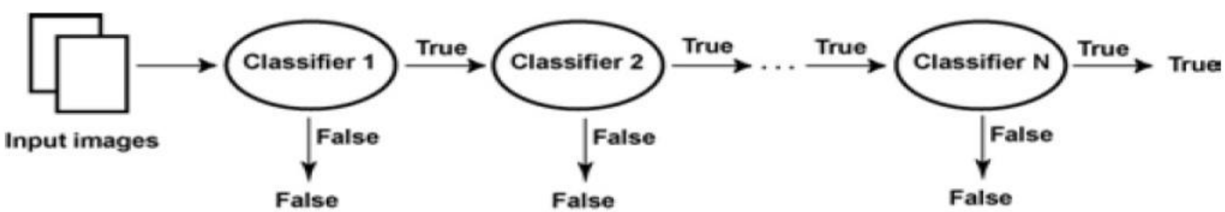
Apply robust mode detection, like mean shift

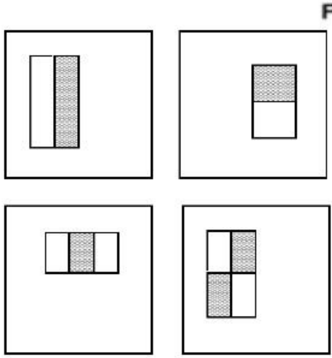


Final detections

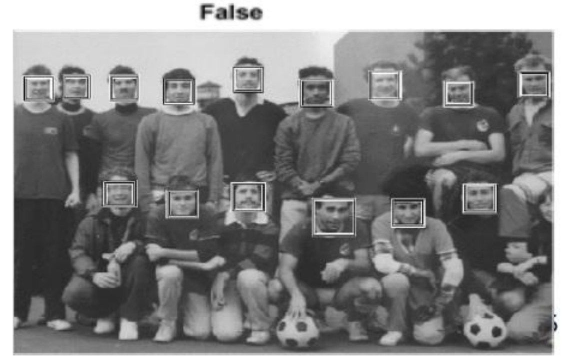
Detection By Using Cascade Classifiers (Viola-Jones AdaBoost Cascade Face Detector)

- A computationally efficient architecture that rapidly rejects unpromising windows
  - A chain of classifiers that each reject some fraction of the negative training samples while keeping almost all positive ones
- Each classifier is an AdaBoost ensemble of rectangular Haar-like features sampled from a large pool



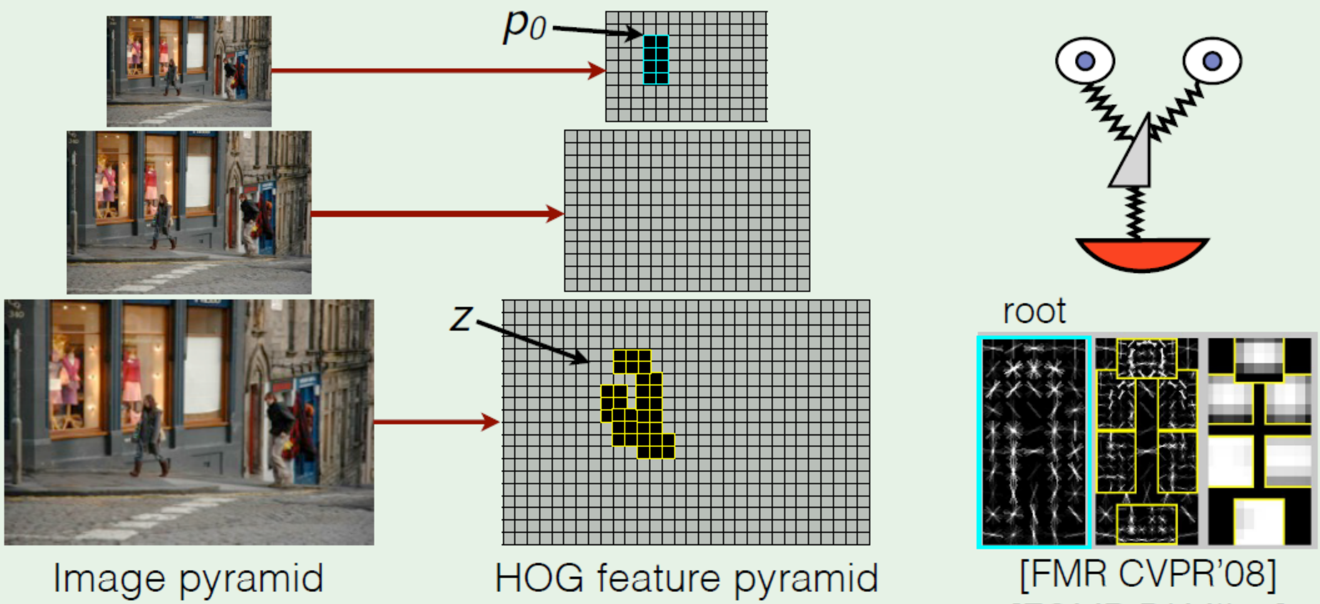


Rectangular Haar features and the first two features chosen by AdaBoost



P. Viola, M. Jones, "Robust Real Time Face Detection," IJCV, 2004.

Deformable Part Based Detectors



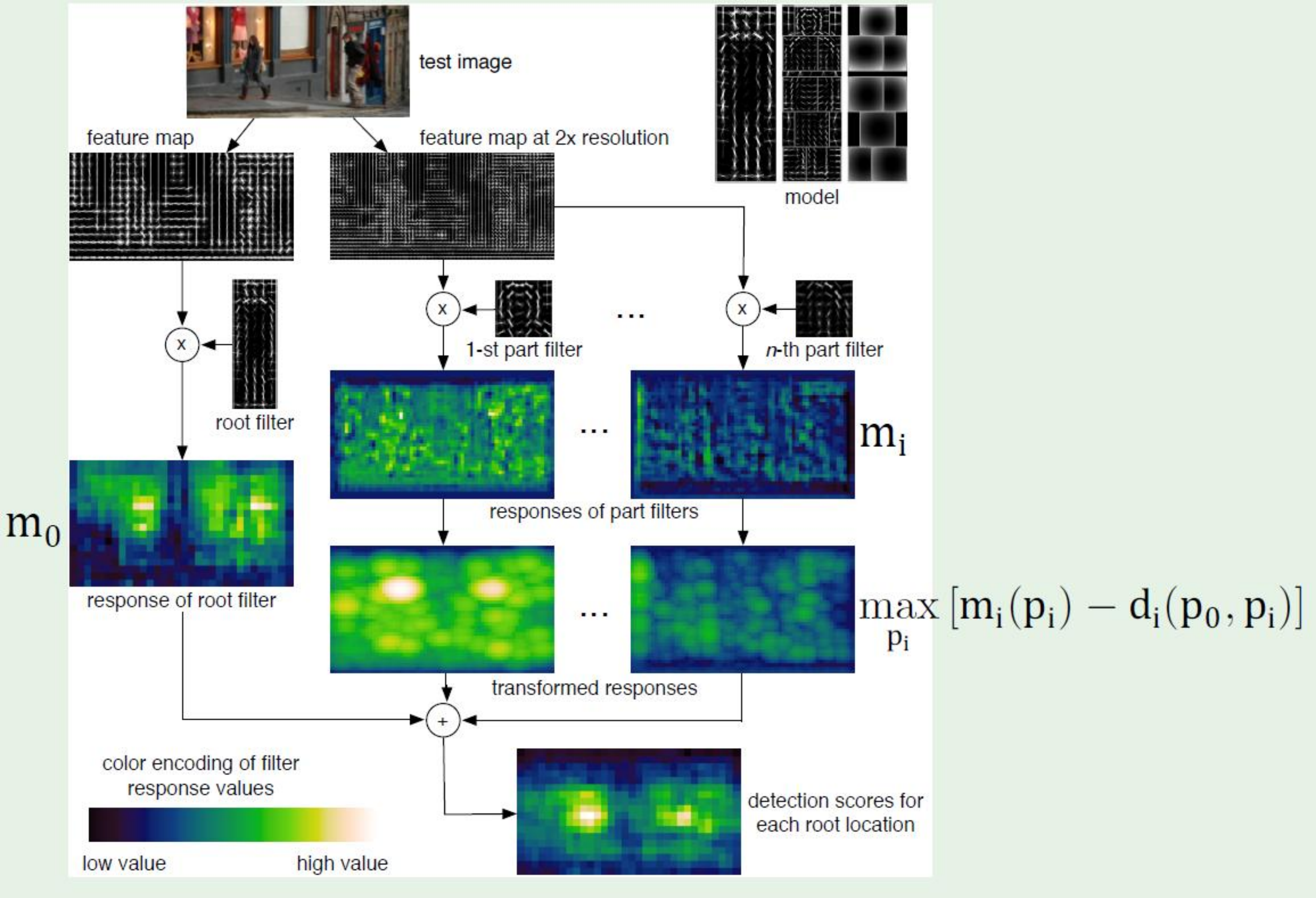
[FMR CVPR'08]
[FGMR PAMI'10]

- Add parts to the Dalal & Triggs detector
  - HOG features
  - Linear filters / sliding-window detector
  - Discriminative training

$z = (p_1, \ldots, p_n)$

$$\text{score}(I, p_0) = \max_{p_1,\ldots,p_n} \underbrace{\sum_{i=0}^{n} m_i(I, p_i)}_{\text{Filter scores}} - \underbrace{\sum_{i=1}^{n} d_i(p_0, p_i)}_{\text{Spring costs}}$$

Detection


test image
feature map
feature map at 2x resolution
model
root filter
1-st part filter
n-th part filter
...
m_i
m_0
responses of part filters
response of root filter
$\max_{p_i} [m_i(p_i) - d_i(p_0, p_i)]$
transformed responses
color encoding of filter response values
low value
high value
detection scores for each root location

Computing Final Scores

$$z = (p_1, \ldots, p_n)$$

$$\text{score}(I, p_0) = \max_{p_1,\ldots,p_n} \underbrace{\sum_{i=0}^{n} m_i(I, p_i)}_{\text{Filter scores}} - \underbrace{\sum_{i=1}^{n} d_i(p_0, p_i)}_{\text{Spring costs}}$$

Filter Scores $m_i(I, p_i) = w_i \bullet \Phi(I, p_i)$

Spring Costs $d_i(p_0, p_i) = d_i \bullet (dx^2, dy^2, dx, dy)$

$$score(I, p_0) = \max_{z} \quad w \bullet \Phi(I, (p_0, z))$$

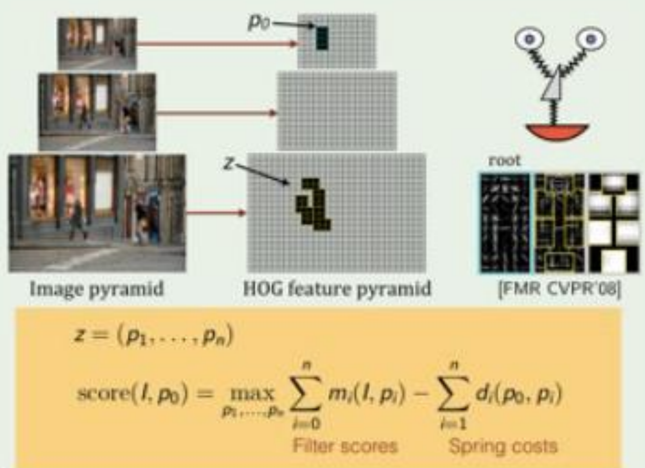
Latent Training

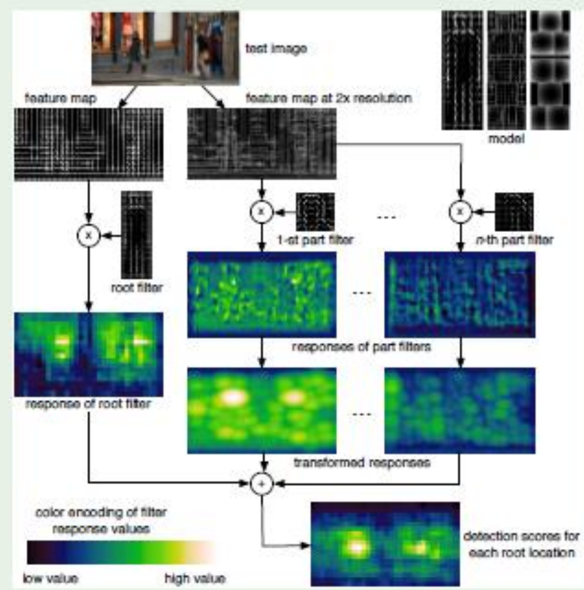
- ‘Latent SVM’ methodology allows use to estimate the unknown part position variables and appearance-class labels during both training and testing.
- This also provides a fine-tuning of the labeled instance positions during training, thus greatly sharpening the resulting models and allowing training from less precise annotations.

Training – Step 1

$$Z_{Pi} = \underset{z \in Z(x_i)}{\operatorname{argmax}} \, \mathbf{w}_{(t)} \cdot \Phi(x_i, z) \quad \forall i \in P$$

This is just detection:





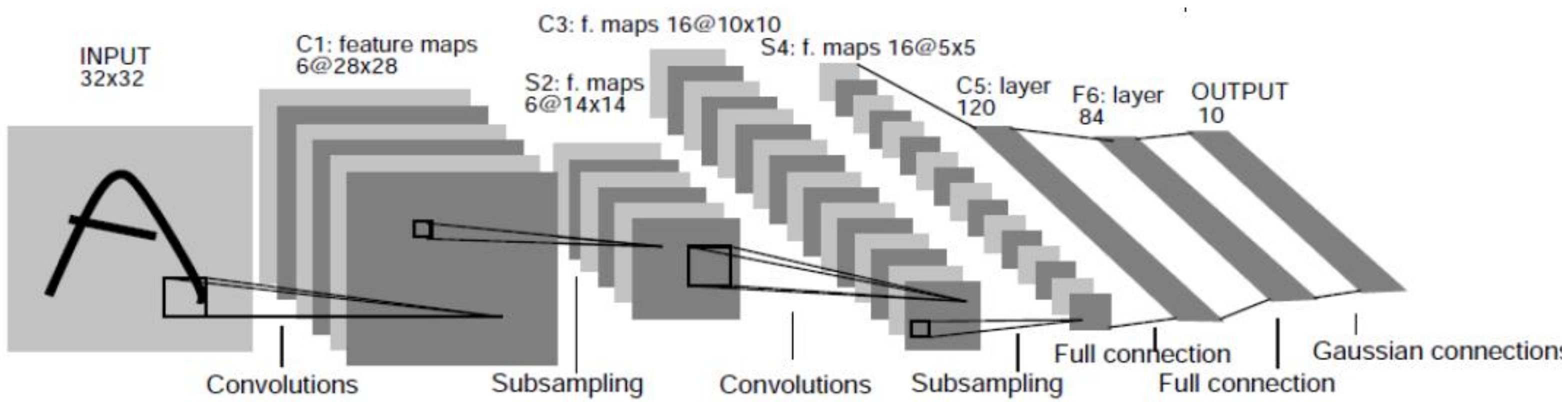
Training – Step 2

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in P}\max\{0, 1-\mathbf{w}\cdot\Phi(x_i, Z_{Pi})\}$$
$$+ C\sum_{i\in N}\max\{0, 1+\max_{z\in Z(x)}\mathbf{w}\cdot\Phi(x_i, z)\}$$

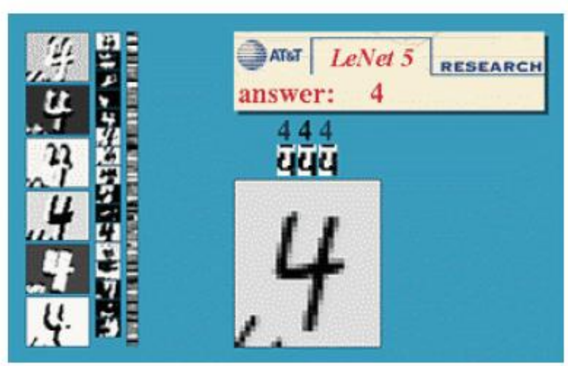
- Convex
- Similar to a structural SVM
- But, recall 500 million to 1 billion negative examples!
- Can be solved by a working set method
- – "bootstrapping"
- – "data mining"
- – "constraint generation"
- – requires a bit of engineering to make this fast.

Deep Neural Network Based Detectors

- A series of banks of convolution filters that alternately analyse the output images of the previous bank ("simple cells") and spatially pool the resulting rectified responses ("complex cells").
- Trained by gradient descent on large training sets.

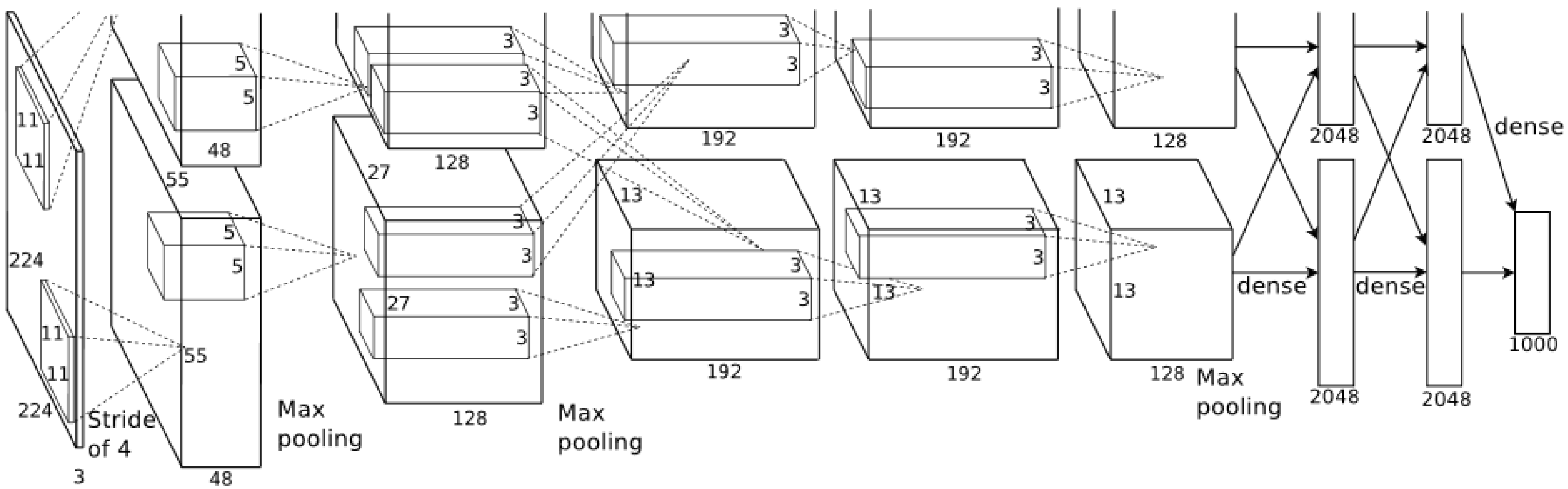


AT&T system – reads ~10% of U.S. cheques



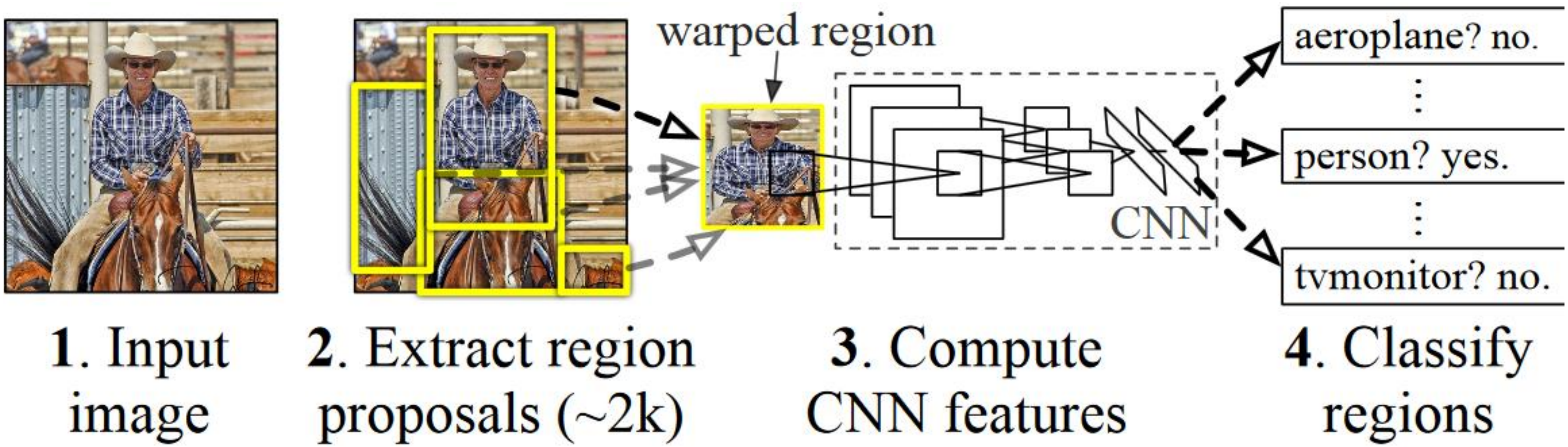
[Lecun 1992-8]

ImageNet ILSVRC'12 winner



Krizhevsky, Sutskever, and Hinton.
ImageNet Classification with Deep Convolutional Neural Networks.
NIPS 2012.

R-CNN: "Regions with CNN features"



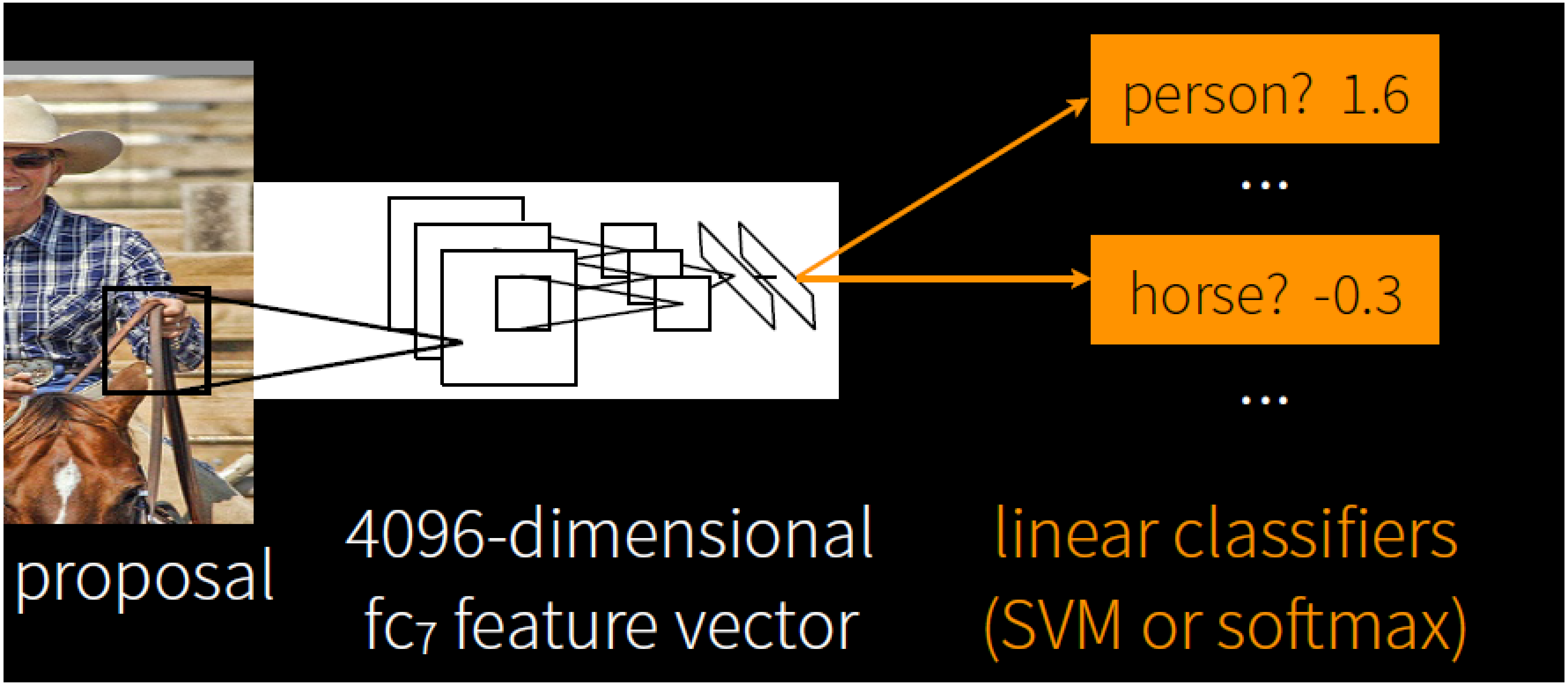
- Proposal-method agnostic, many choices
  - Selective Search [van de Sande, Uijlings et al.] (Used in this work)
  - Objectness [Alexe et al.]
  - Category independent object proposals [Endres & Hoiem]
  - CPMC [Carreira & Sminchisescu]
  - Edge Boxes [Zitnick and Dollar]
  - DeepBox [Kuo et al.]
  - Deep MultiBox [Erhan et al.]

Steps

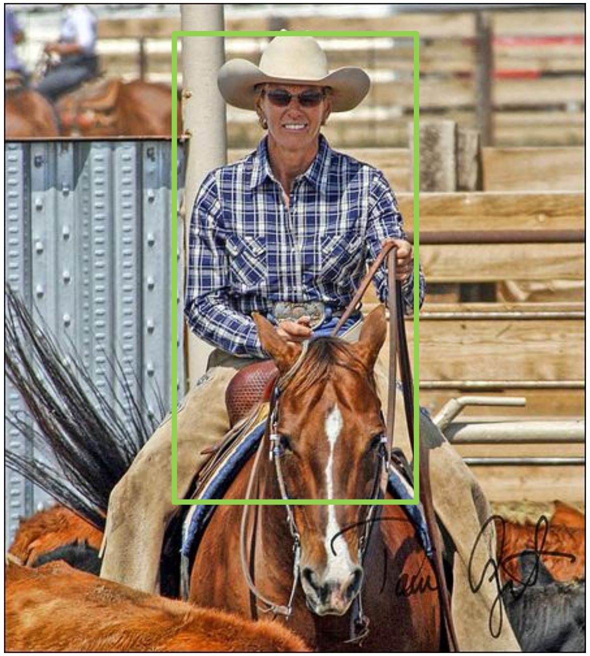
Crop

Scale to 227x227

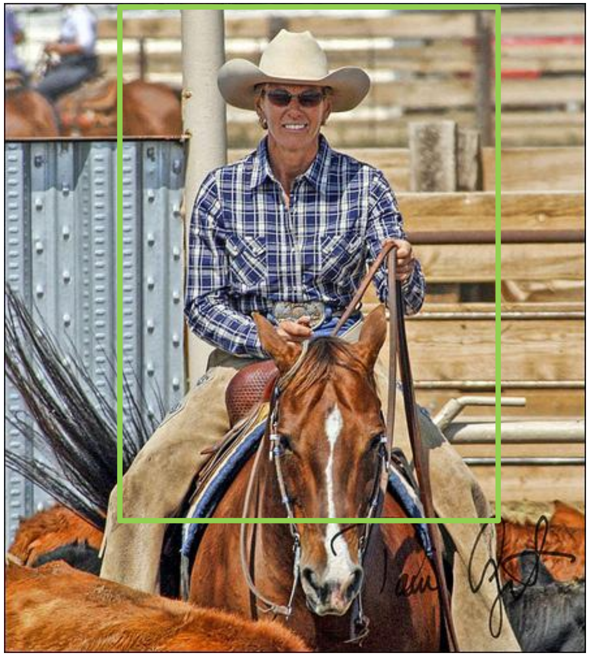
Steps


person? 1.6
...
horse? -0.3
...
proposal
4096-dimensional
fc$_7$ feature vector
linear classifiers
(SVM or softmax)

Steps



Linear Regression on CNN features



Original proposal

Predicted object bounding box

Bounding-box regression

OverFeat: Classification + Localization

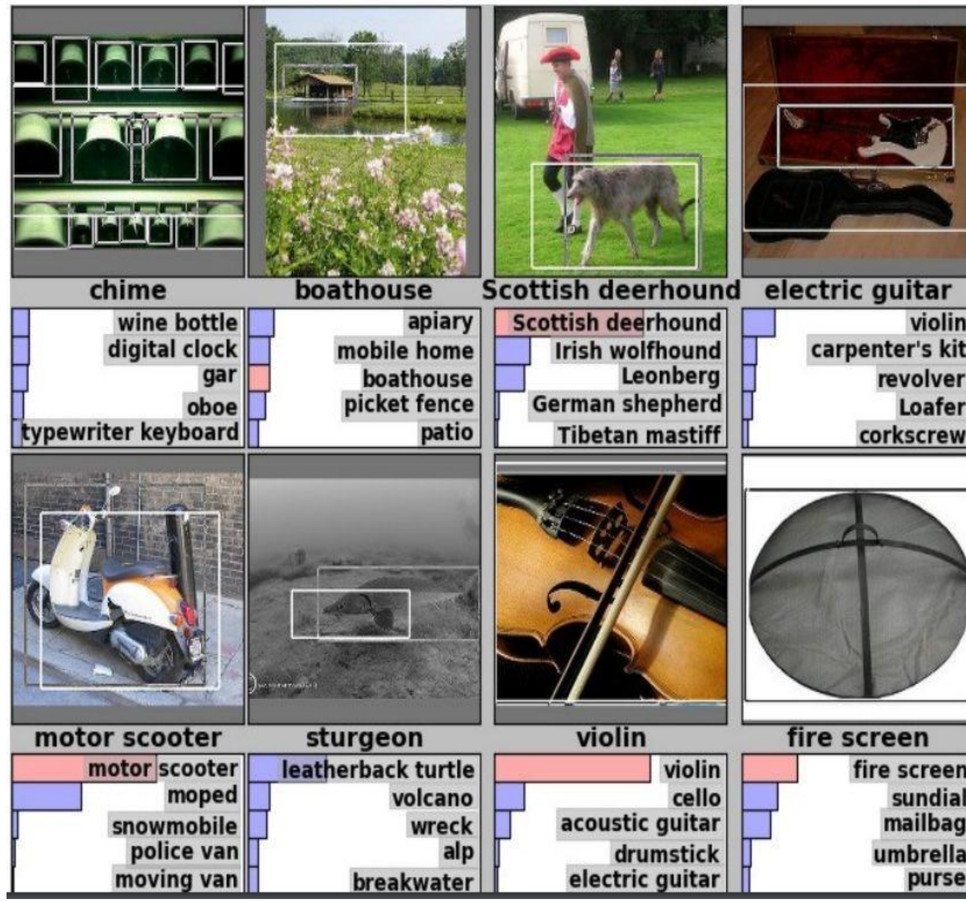
1000 classes (same as classification)

Each image has 1 class, at least one bounding box

~800 training images per class

Algorithm produces 5 (class, box) guesses

Example is correct if at least one one guess has correct class AND bounding box at least 0.5 intersection over union (IoU)



Sermanet et al., "OverFeat: Integrated Recognition, Localization and Detection using colvolutional networks," ICLR, 2014.

Idea #1: Localization as Regression

Input: image

Neural Net

Output:
Box coordinates
(4 numbers)

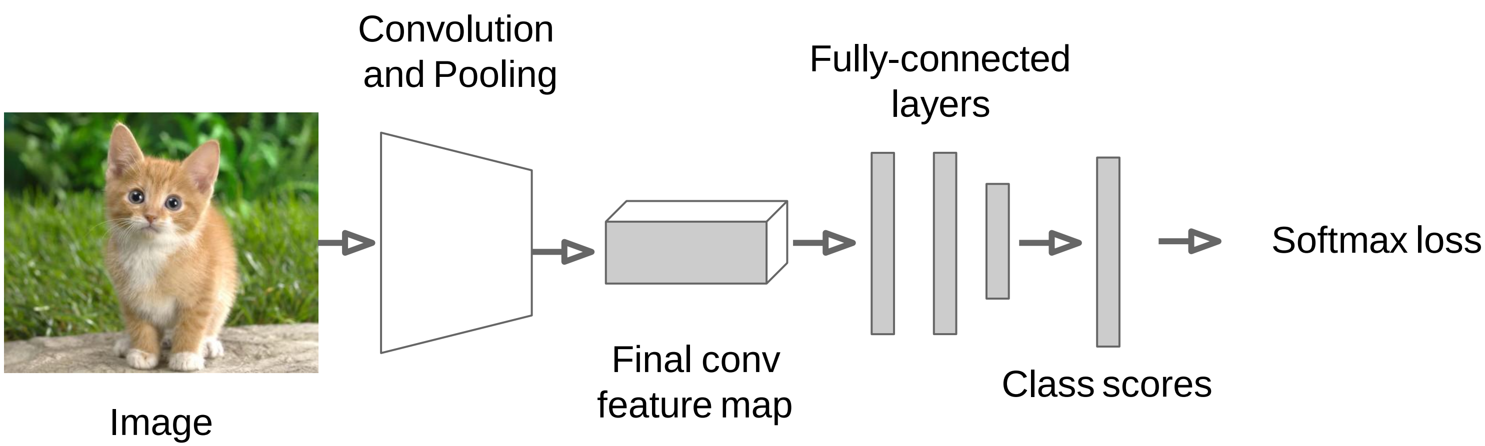
Correct output:
box coordinates
(4 numbers)

Loss:
L2 distance

Only one object,
simpler than detection

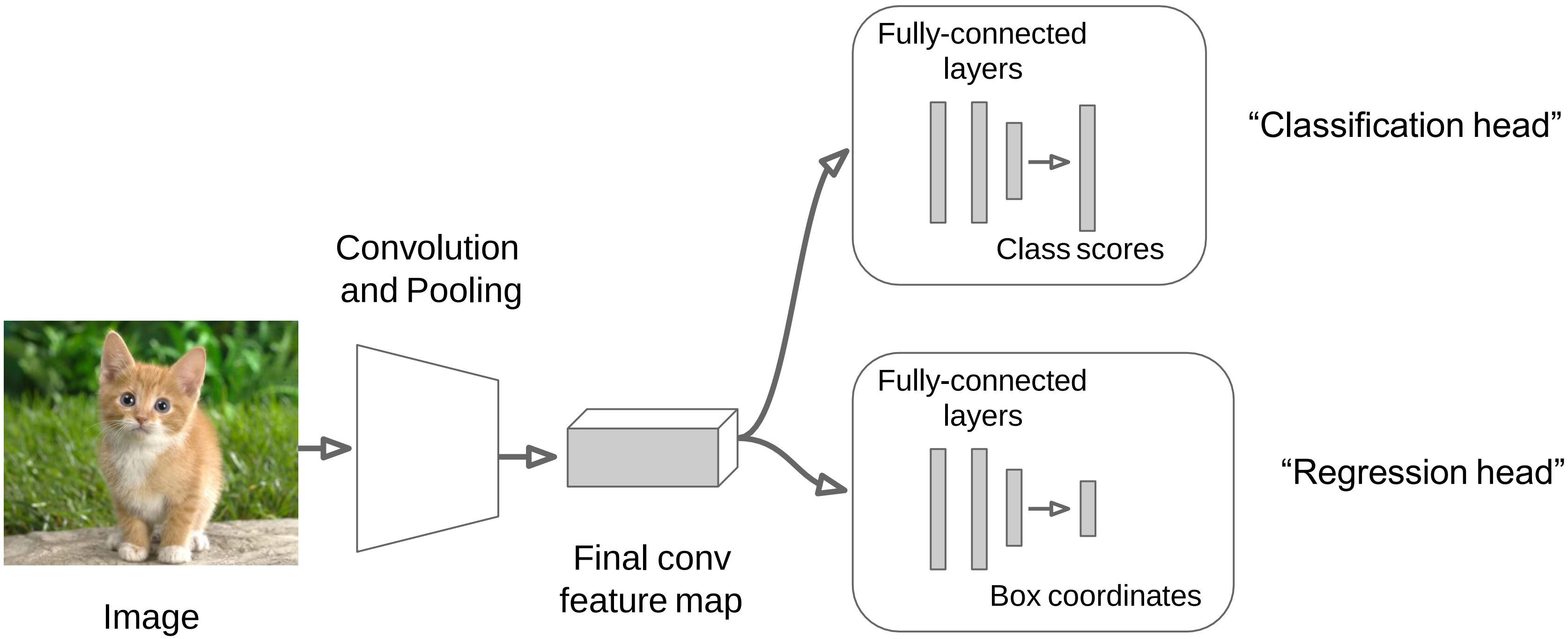
Simple Recipe for Classification + Localization

Step 1: Train (or download) a classification model (AlexNet, VGG, GoogLeNet)



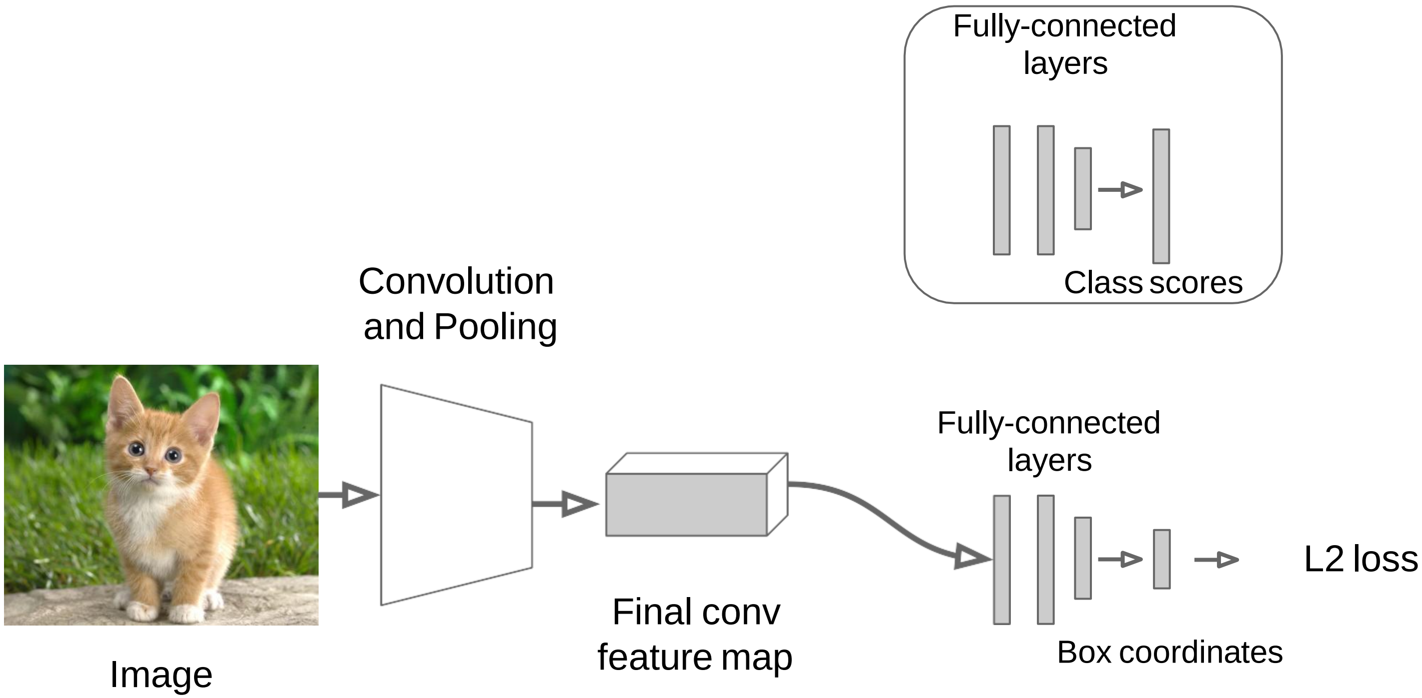
Simple Recipe for Classification + Localization

Step 2: Attach new fully-connected "regression head" to the network



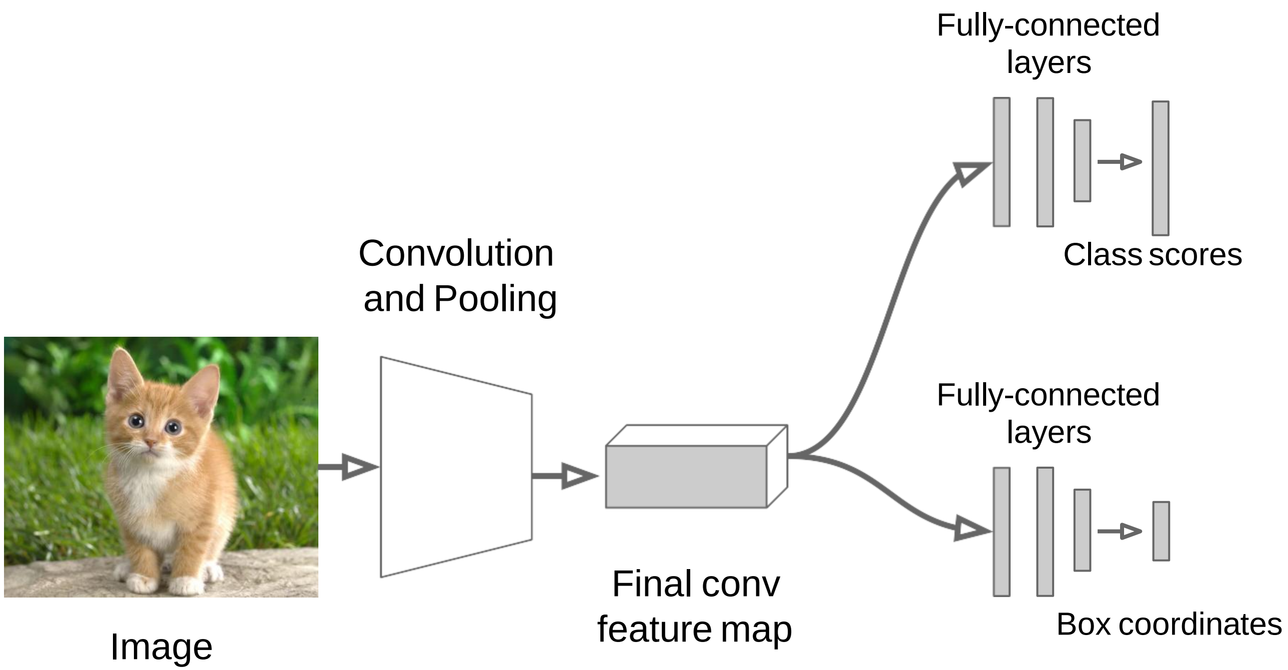
Simple Recipe for Classification + Localization

Step 3: Train the regression head only with SGD and L2 loss



Simple Recipe for Classification + Localization

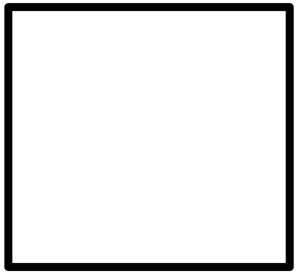
Step 4: At test time use both heads



Idea #2: Sliding Window

- Run classification + regression network at multiple locations on a high-resolution image
- Convert fully-connected layers into convolutional layers for efficient computation
- Combine classifier and regressor predictions across all scales for final prediction

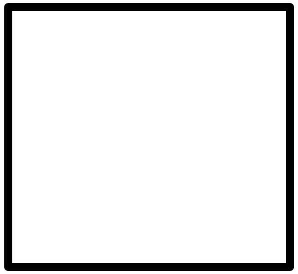
Sliding Window: Overfeat



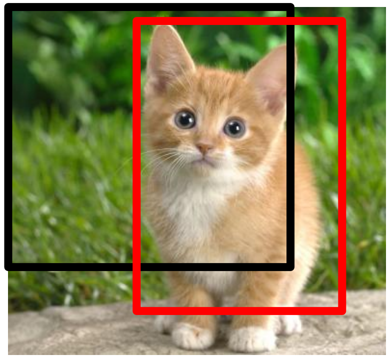
Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

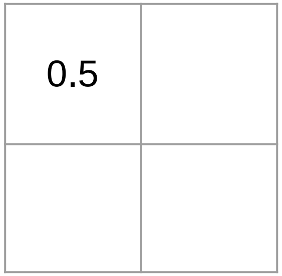
Sliding Window: Overfeat



Network input:
3 x 221 x 221

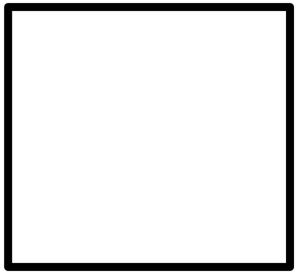


Larger image:
3 x 257 x 257

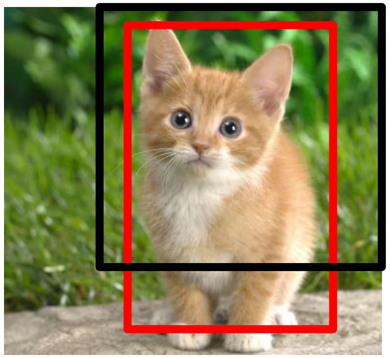


Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221

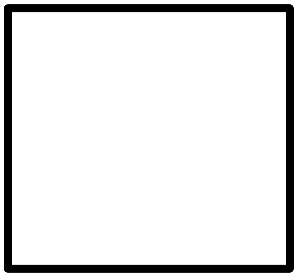


Larger image:
3 x 257 x 257

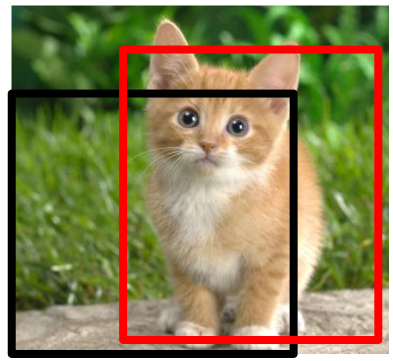
0.5	0.75

Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221

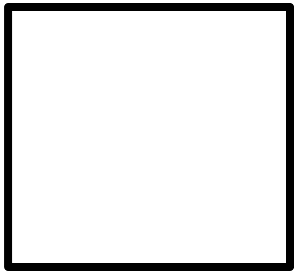


Larger image:
3 x 257 x 257

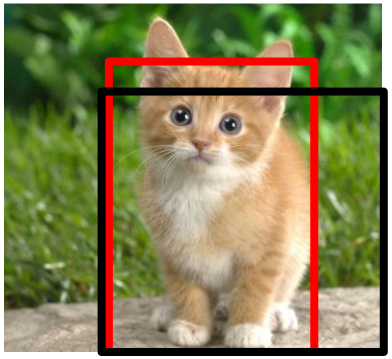
0.5	0.75
0.6	

Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221



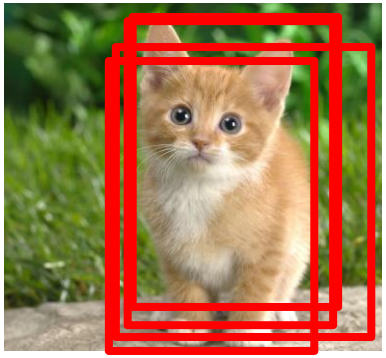
Larger image:
3 x 257 x 257

0.5	0.75
0.6	0.8

Classification scores:
P(cat)

Sliding Window: Overfeat

Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

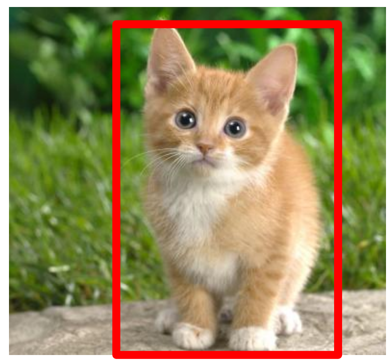
0.5	0.75
0.6	0.8

Classification scores:
P(cat)

Sliding Window: Overfeat

Greedily merge boxes and scores (details in paper)

Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

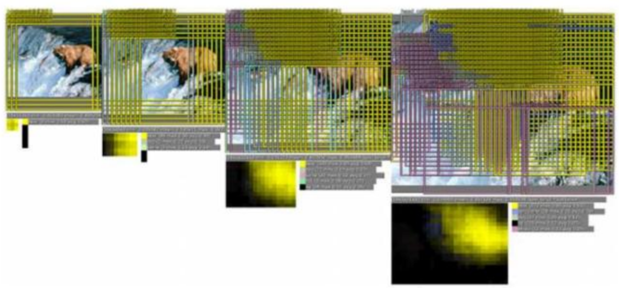
0.8

Classification score: P (cat)

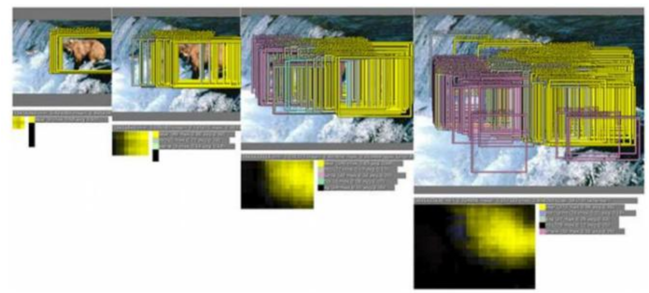
Sliding Window: Overfeat

In practice use many sliding window locations and multiple scales

Window positions + score maps



Box regression outputs



Final Predictions

The network is run at each location and at six different scales. This sliding window approach is computationally feasible for a ConvNet (as compared to other types of models) because computations for overlapping regions are shared.

Fast-CNN (Girshick, ICCV 2015)

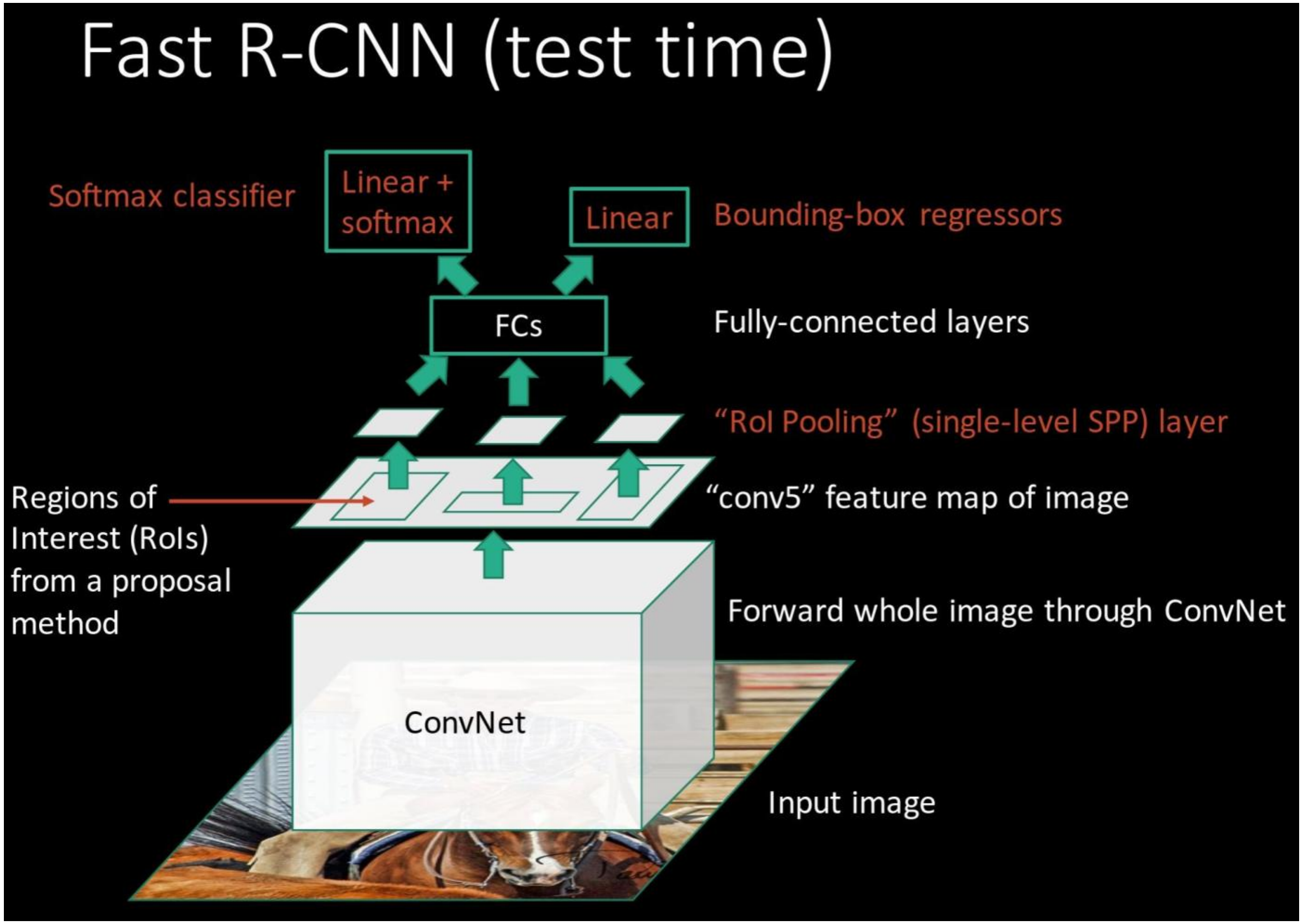
R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal

2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors

3. Complex multistage training pipeline

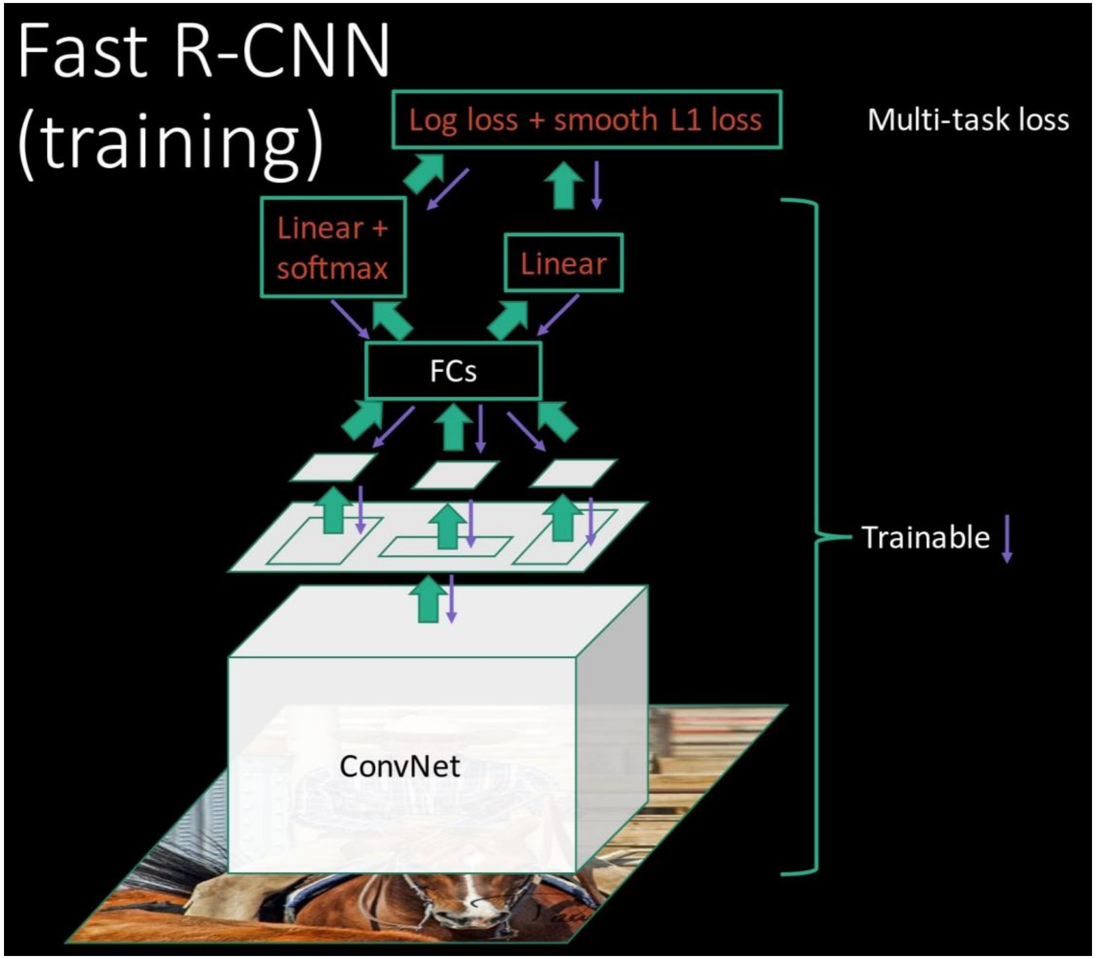
Fast CNN



R-CNN Problem #1: Slow at test-time due to independent forward passes of the CNN

Solution: Share computation of convolutional layers between proposals for an image

Fast CNN



R-CNN Problem #2:
Post-hoc training: CNN not updated in response to final classifiers and regressors

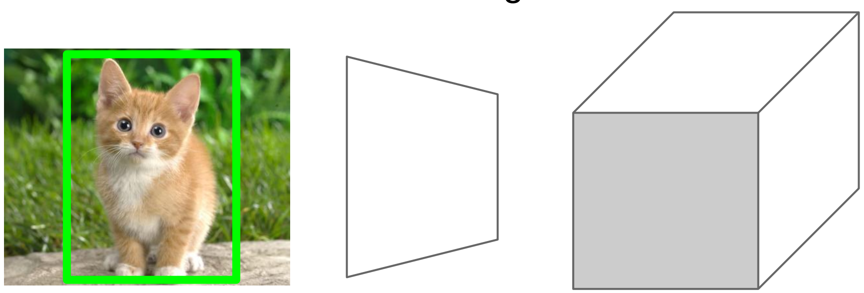
R-CNN Problem #3:
Complex training pipeline

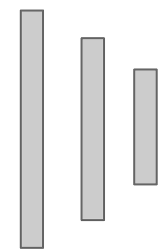
Solution:
Just train the whole system end-to-end all at once!

Fast R-CNN: Region of Interest Pooling

Convolution and Pooling

Fully-connected layers



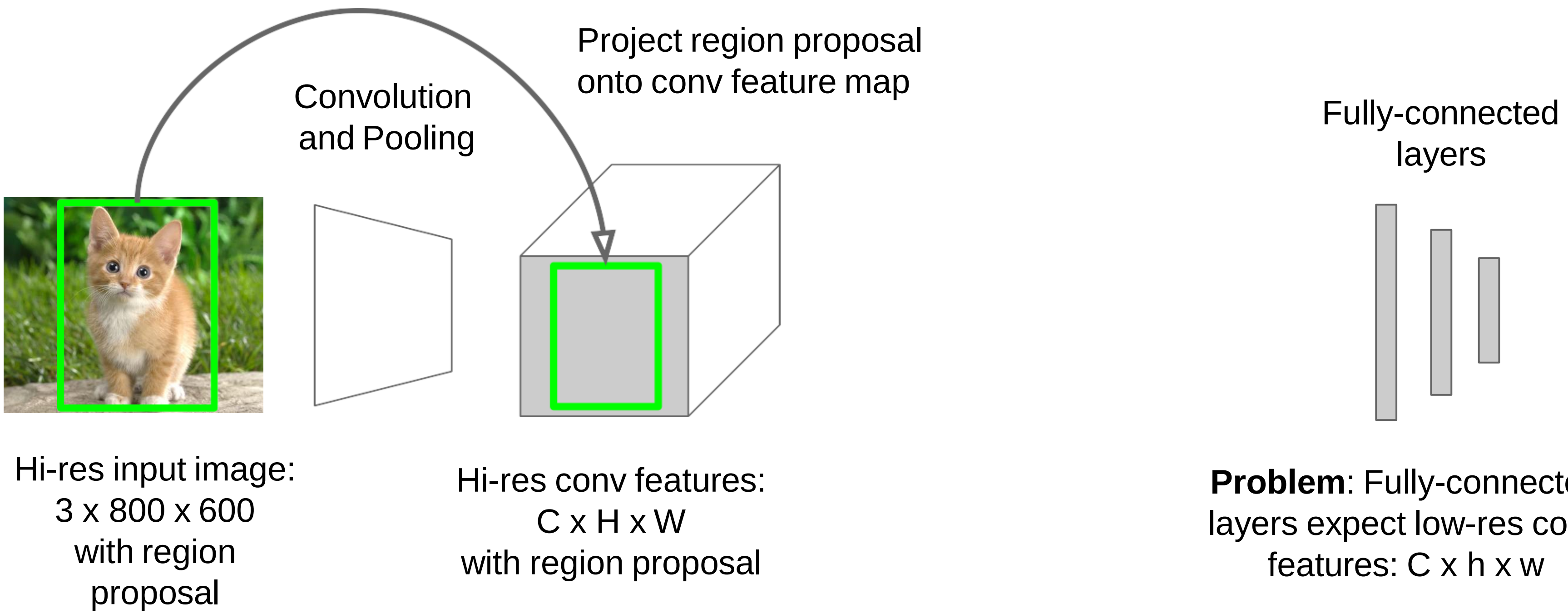


Hi-res input image:
3 x 800 x 600
with region
proposal

Hi-res conv features:
C x H x W
with region proposal

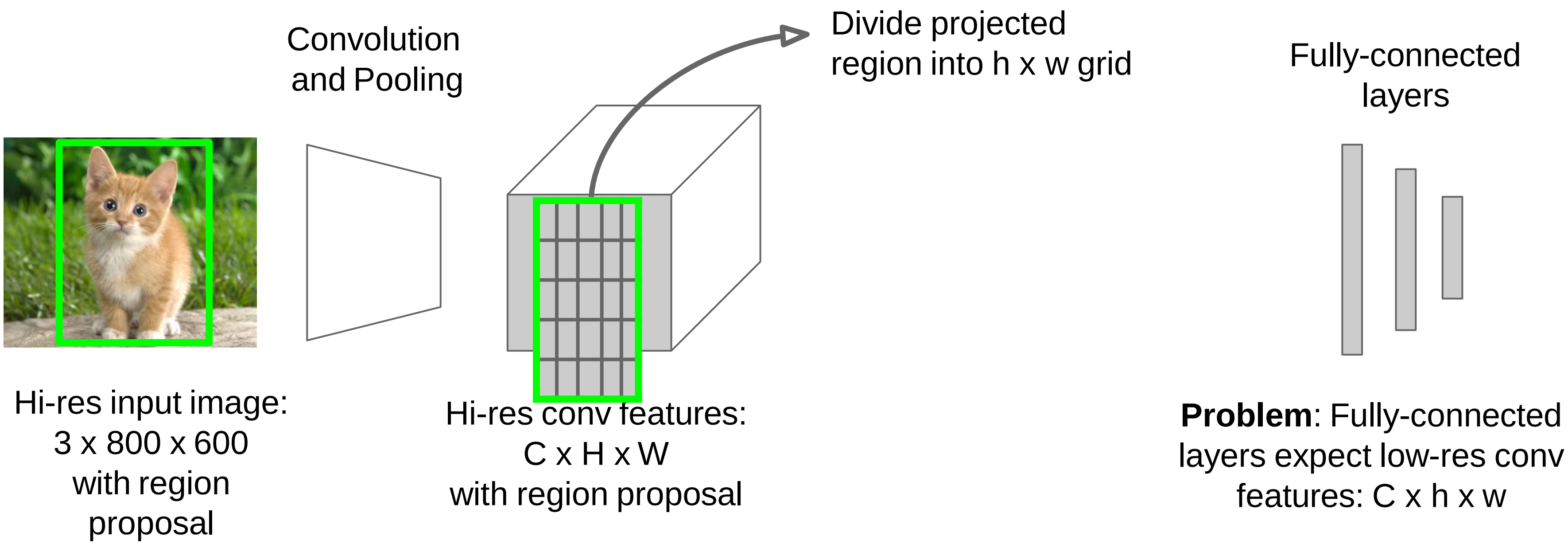
Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling

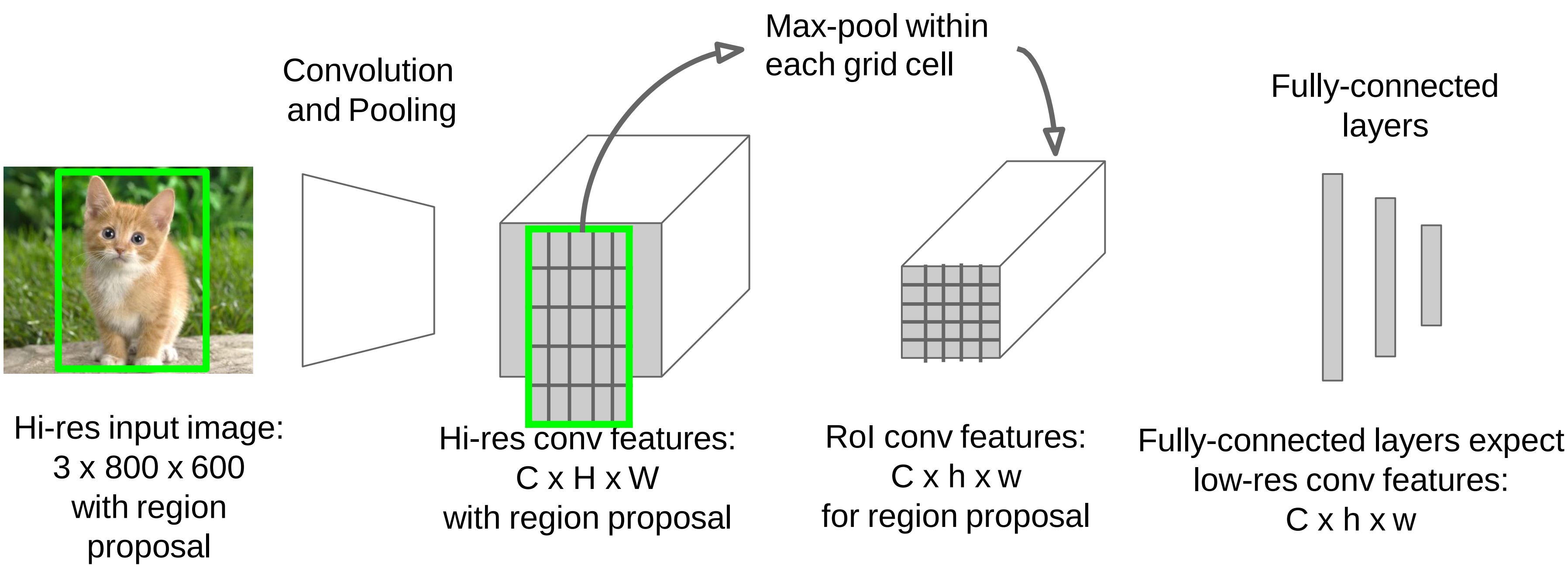


Problem: Fully-connected layers expect low-res conv features: C x h x w

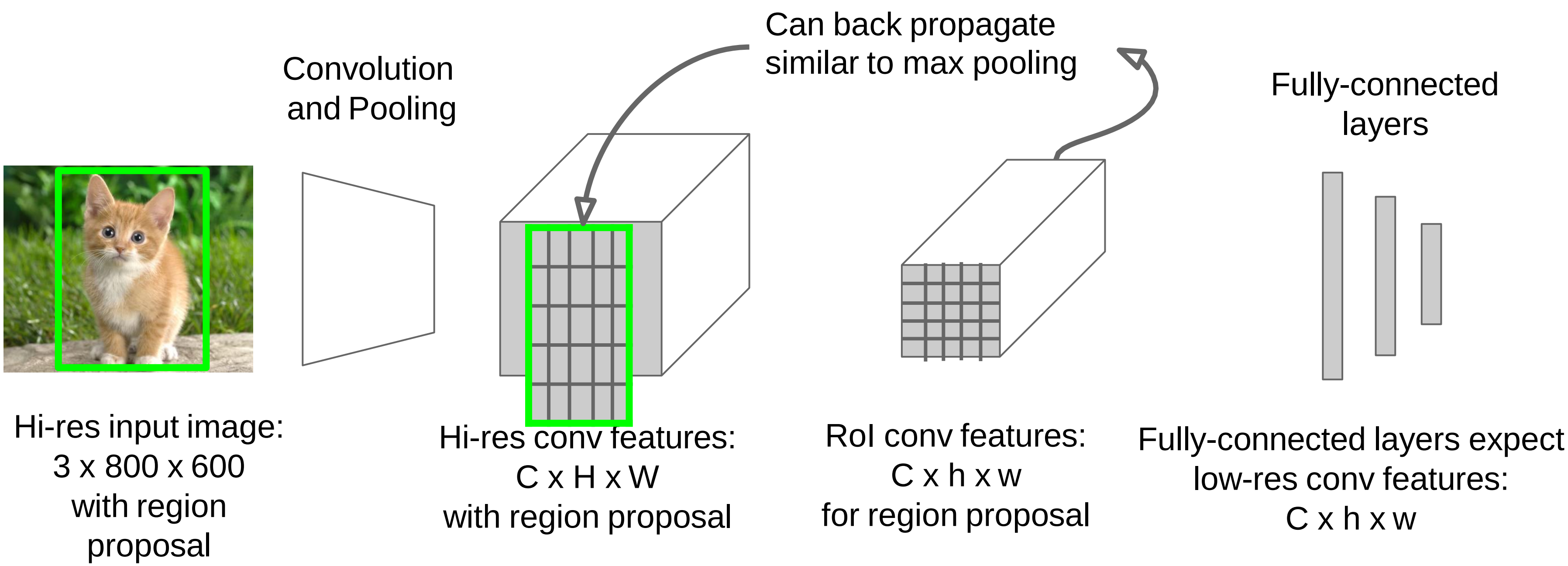
Fast R-CNN: Region of Interest Pooling



Fast R-CNN: Region of Interest Pooling



Fast R-CNN: Region of Interest Pooling



Fast R-CNN Results

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	**9.5 hours**
	(Speedup)	1x	**8.8x**
FASTER!	Test time per image	47 seconds	**0.32 seconds**
	(Speedup)	1x	**146x**

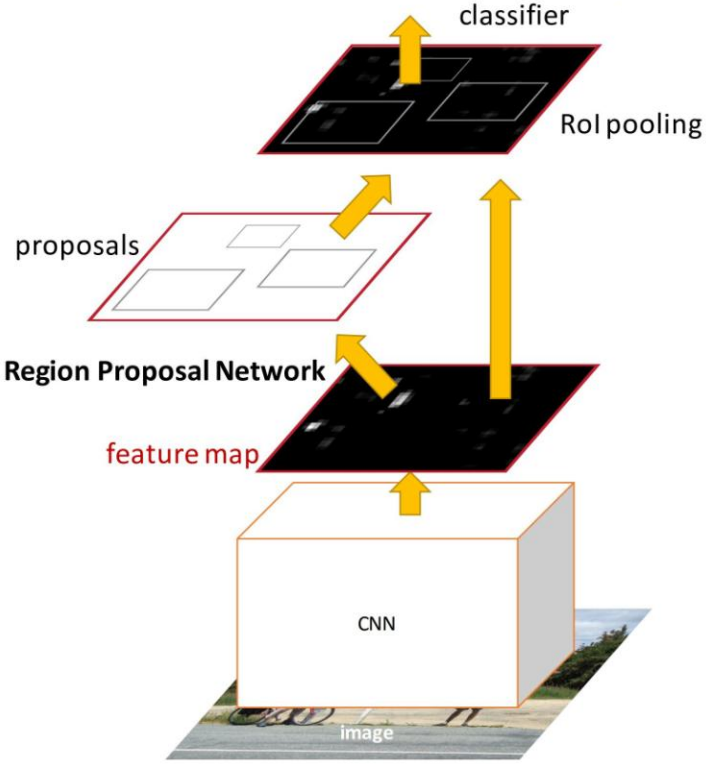
Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Problem:

Test-time speeds don't include region proposals

	R-CNN	**Fast R-CNN**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with Selective Search	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

Faster R-CNN (Ren et al., PAMI 2017)



Insert a **Region Proposal Network (RPN)** after the last convolutional layer

RPN trained to produce region proposals directly; no need for external region proposals!

After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

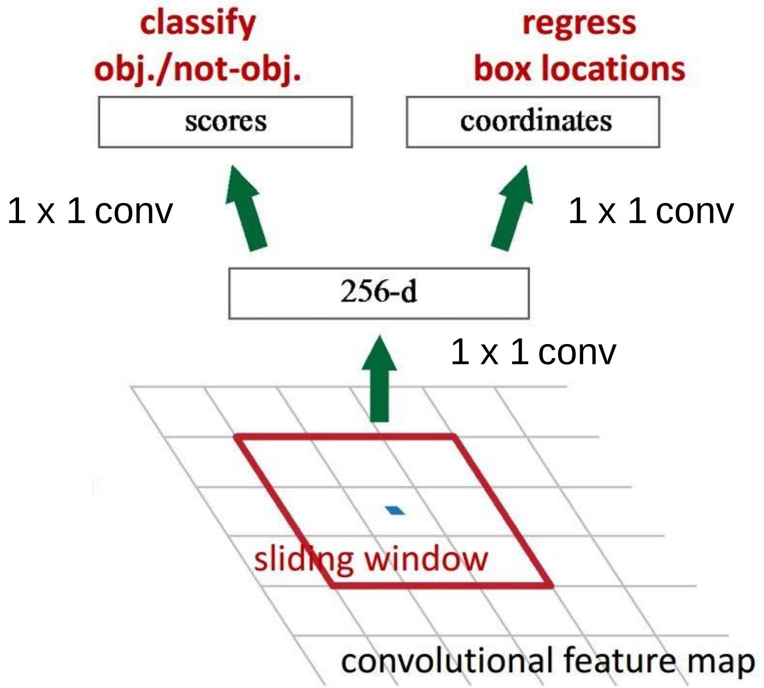
Faster R-CNN: Region Proposal Network

Slide a small window on the feature map

Build a small network for:
• classifying object or not-object, and
• regressing bbox locations

Position of the sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



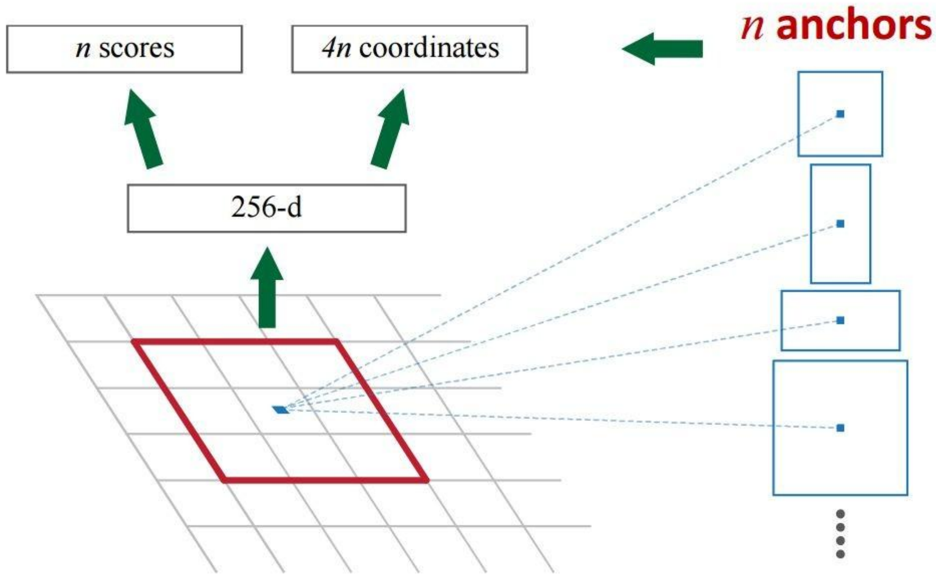
Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

Regression gives offsets from anchor boxes

Classification gives the probability that each (regressed) anchor shows an object



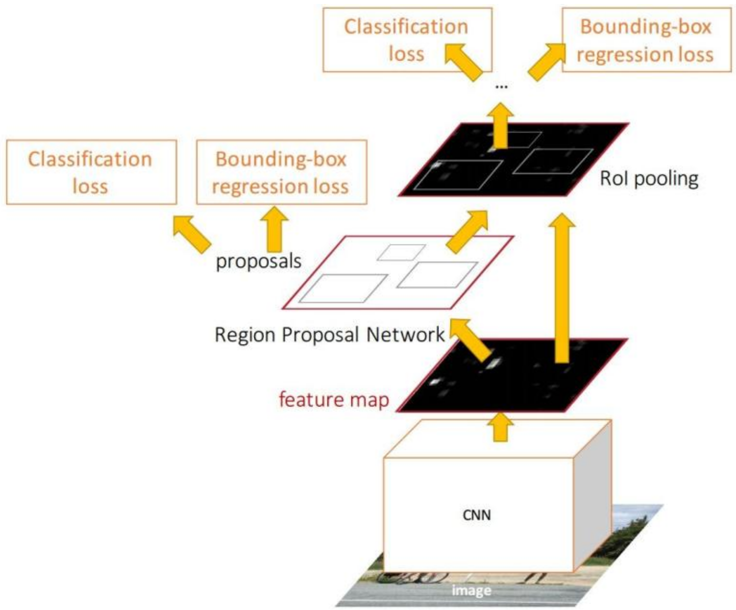
Faster R-CNN: Training

In the the paper: Ugly pipeline

- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!
One network, four losses

- RPN classification (anchor good / bad)
- RPN regression (anchor -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)



Faster R-CNN: Results

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

YOLO (My favorite)

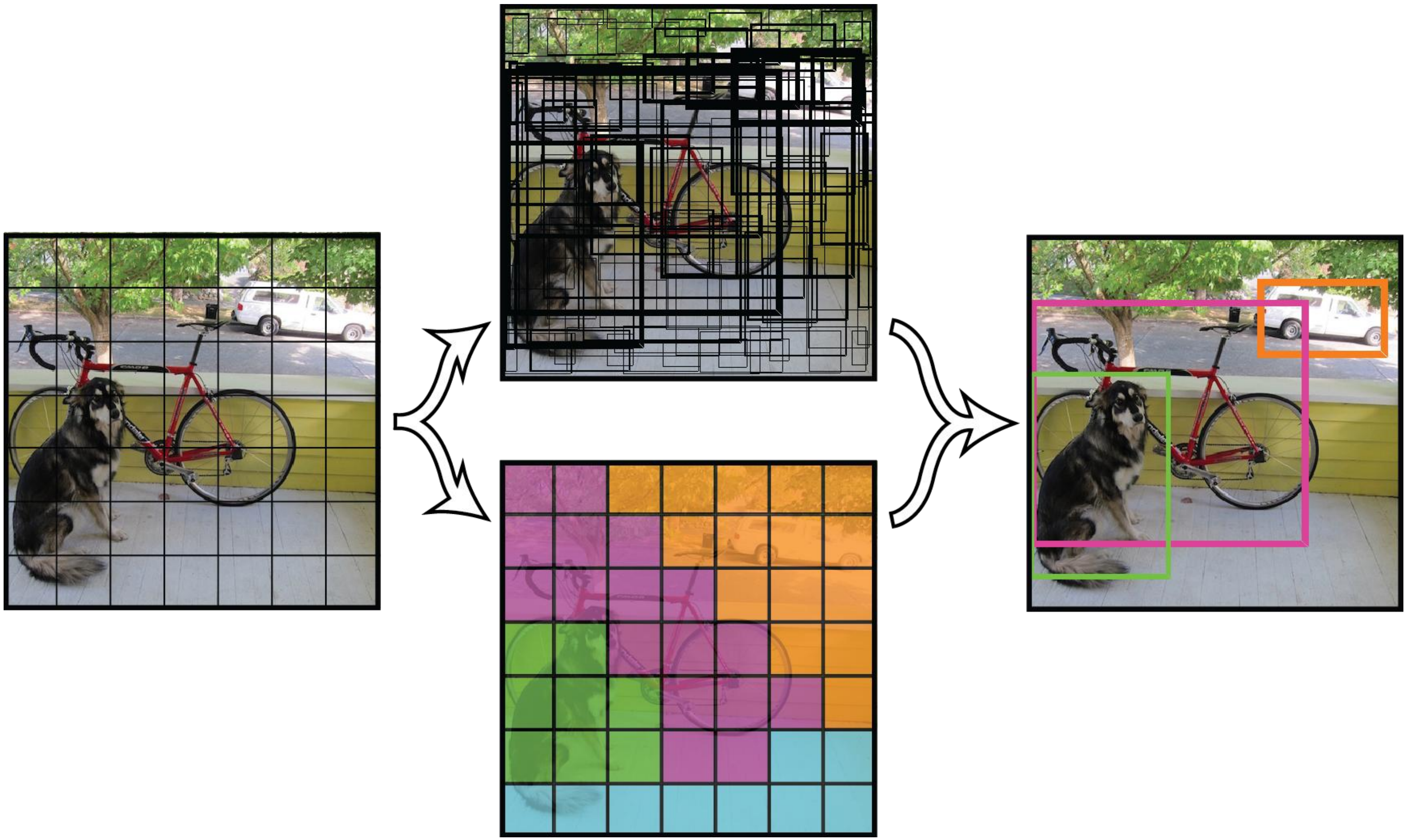
Detection as Single Regression Problem

Developed as Single Convolutional Network

Reason Globally on the Entire Image

Learns Generalizable Representations

YOLO



YOLO - Steps

- Divide the image into a SxS grid.

 If the center of an object fall into a grid cell, it will be the responsible for the object.

- Each grid cell predicts

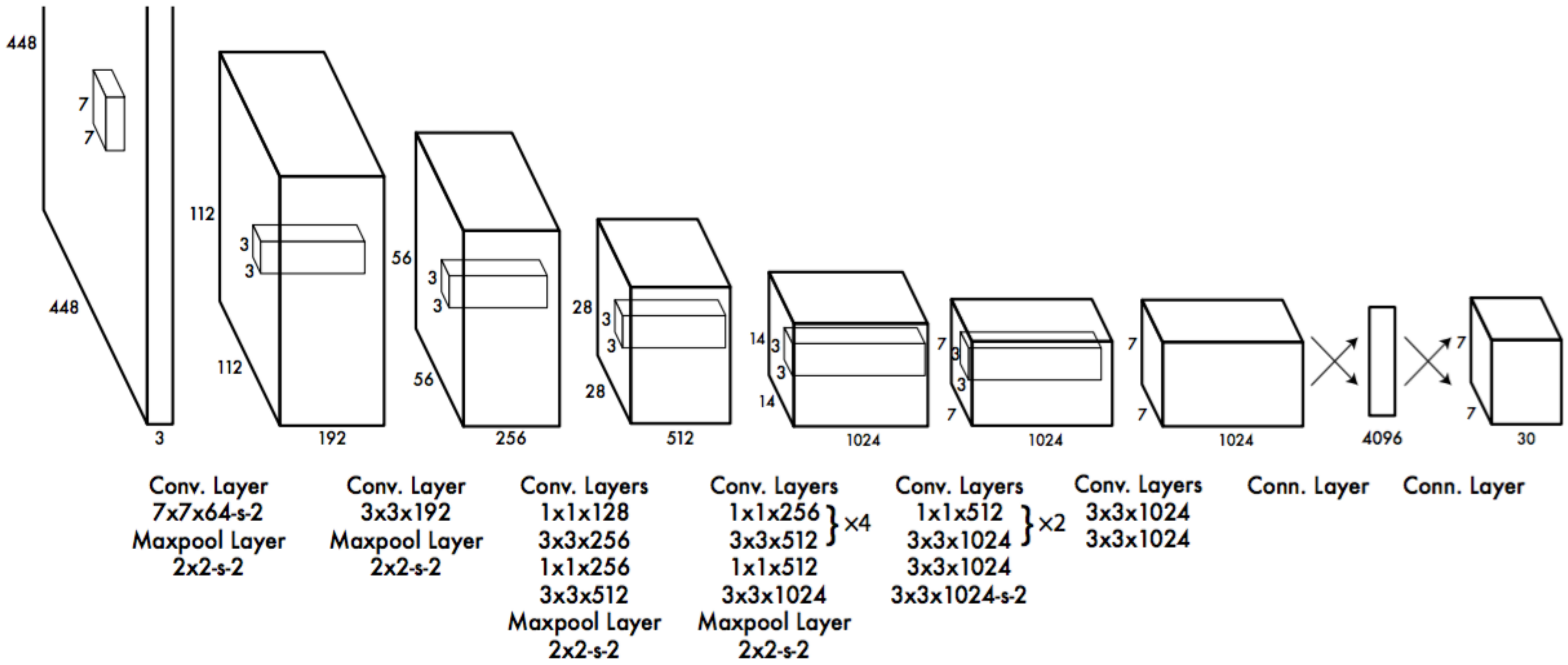
-- B bounding boxes;

-- B confidence scores as **C=Pr(Obj)*IOU;**

-- C conditional class probabilities P=Pr(class_i|Object);

- Confidence Prediction is obtained as IOU of predicted box and any ground truth box.

Design



448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer

Loss function

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

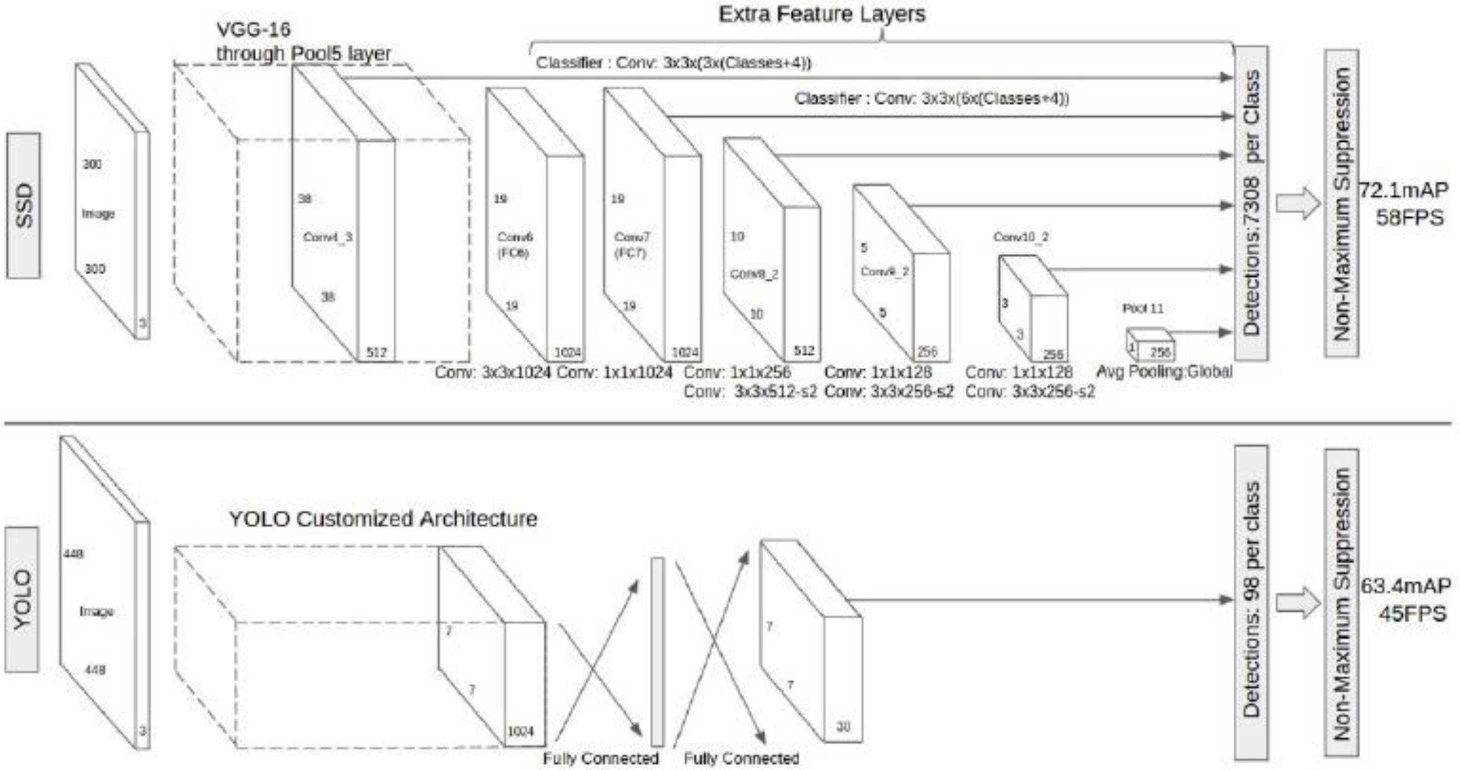
SSD: Single Shot MultiBox Detector

(Liu et al., ECCV 2016)

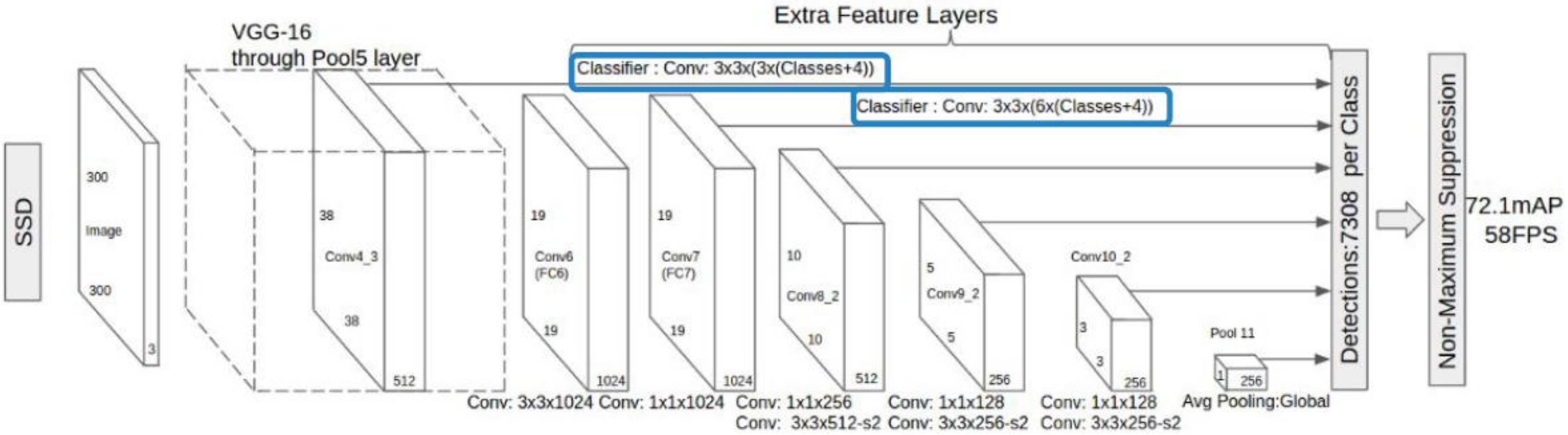
- A single-shot detector for multiple categories that is faster than state of the art single shot detectors (YOLO) and as accurate as Faster R-CNN,
- Predicts category scores and boxes offset for a fixed set of default BBs **using small convolutional filters applied to feature maps,**
- Predictions of different scales from feature maps of different scales, and separate predictions by aspect ratio,
- End-to-end training and high accuracy, **improving speed vs accuracy trade-off.**

MultiBox

Comparison to YOLO



SSD

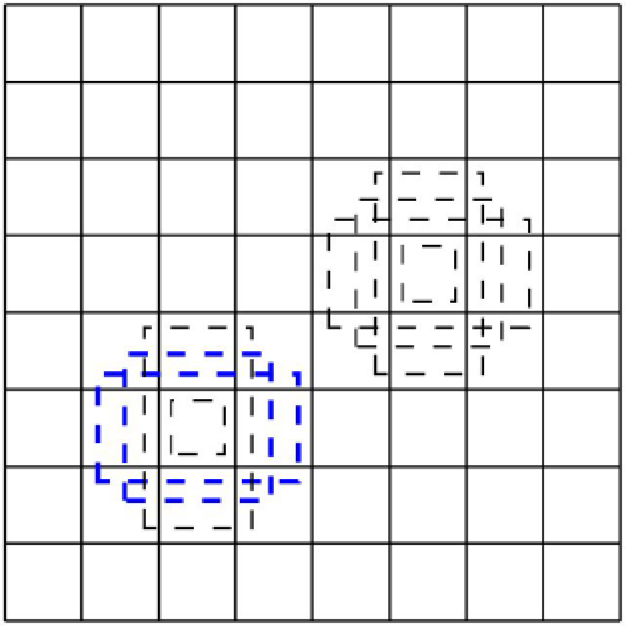


- **Convolutional predictors for detection:** On top of each conv feature map, there are a set of filters that predict detections for different aspect ratios and class categories

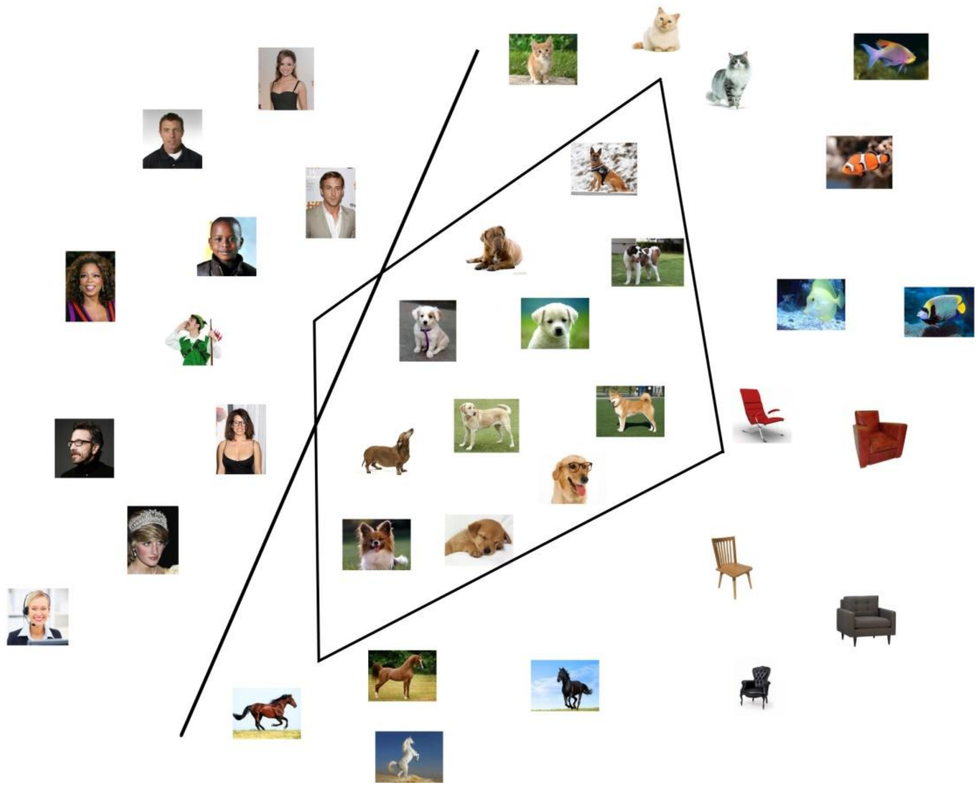
SSD

- **Default boxes and aspect ratios**

  Similar to the *anchors of Faster R-CNN, with the difference that SSD* applies them on several feature maps of different resolutions



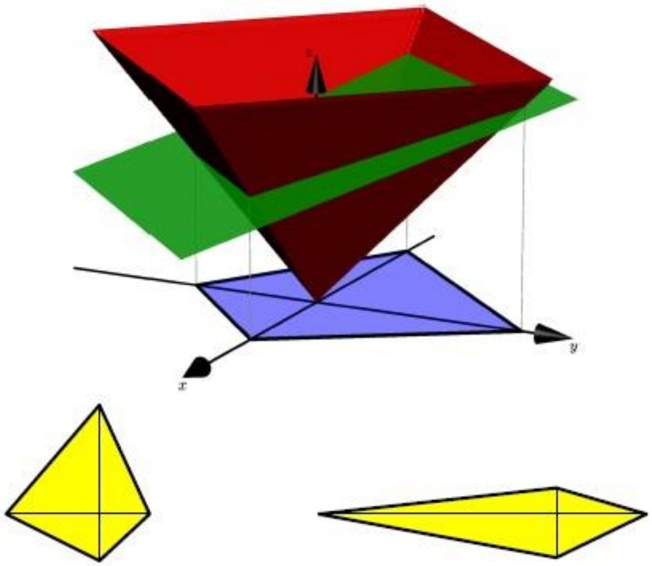
Part 2: (Bonus) – Polyhedral Convex Conic Functions for Classification



A decision hyperplane returned by an SVM successfully separates its training classes, dogs (positive) and people (negative). However it also assigns instances of novel classes such as cats, horses, fish and chairs to the dog class, sometimes with higher confidence scores than for dogs themselves. The problem is the over-large acceptance region – SVM only tries to separate dogs and people, not to bound the dog class. A lighter (e.g. Polyhedral or ellipsoidal) decision boundary improves this localization, reducing mis-classifications caused by unforeseen classes and outliers.

METHOD

- These classifiers use the polyhedral conic functions – essentially projections of hyperplane sections through L1 cones – to define their acceptance regions for positives.
- The Polyhedral Conic Functions and Extended Polyhedral Conic Functions respectively have the forms



$$f_{\mathbf{w},\gamma,\mathbf{s},b}(\mathbf{x}) = \mathbf{w}^T(\mathbf{x}-\mathbf{s}) + \gamma\|\mathbf{x}-\mathbf{s}\|_1 - b \qquad (PCF)$$

$$f_{\mathbf{w},\boldsymbol{\gamma},\mathbf{s},b}(\mathbf{x}) = \mathbf{w}^T(\mathbf{x}-\mathbf{s}) + \boldsymbol{\gamma}^T|\mathbf{x}-\mathbf{s}| - b \qquad (EPCF)$$

METHOD

Definition: A function $f(\mathbf{x}): R^d \to R$ is called polyhedral conic if its graph is a cone and all its level sets

$$S_\alpha = \left\{ \mathbf{x} \in R^d : f(\mathbf{x}) \le \alpha \right\}$$

for $\alpha \in R$, are polyhedrons.

Lemma: A graph of the PCF and EPCF functions is a polyhedral cone with a vertex at $(\mathbf{s}, -b)$.

The proposed polyhedral conic classifiers use PCF and EPCF, with decision regions $f(\mathbf{x}) \le 0$ for positives and $f(\mathbf{x}) > 0$ for negatives. Note that this is the opposite of the popular SVM decision rule.

METHOD – Binary Class formulation

- By defining $\tilde{\mathbf{x}} \equiv \begin{pmatrix} \mathbf{x}-\mathbf{s} \\ \|\mathbf{x}-\mathbf{s}\|_1 \end{pmatrix} \in R^{d+1}$, $\tilde{\mathbf{w}} \equiv \begin{pmatrix} -\mathbf{w} \\ -\gamma \end{pmatrix}$ and $\tilde{b} = b$ for PCF; $\tilde{\mathbf{x}} \equiv \begin{pmatrix} \mathbf{x}-\mathbf{s} \\ |\mathbf{x}-\mathbf{s}| \end{pmatrix} \in R^{2d}$, $\tilde{\mathbf{w}} \equiv \begin{pmatrix} -\mathbf{w} \\ -\boldsymbol{\gamma} \end{pmatrix}$, and $\tilde{b} = b$ for EPCF, we obtain the same decision functions as in SVM. So we can use SVM formulation to solve the optimization problem.

$$
\begin{aligned}
\underset{\tilde{\mathbf{w}},\tilde{b}}{\operatorname{arg\,min}} \quad & \frac{1}{2}\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} + C_+\sum_i \xi_i + C_-\sum_j \xi_j \\
\text{such that} \quad & \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i + \tilde{b} + \xi_i \geq 1, \qquad i \in I_+, \\
& \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_j + \tilde{b} + \xi_i \leq -1, \qquad j \in I_-, \\
& \xi_i, \xi_j \geq 0,
\end{aligned}
$$

Method

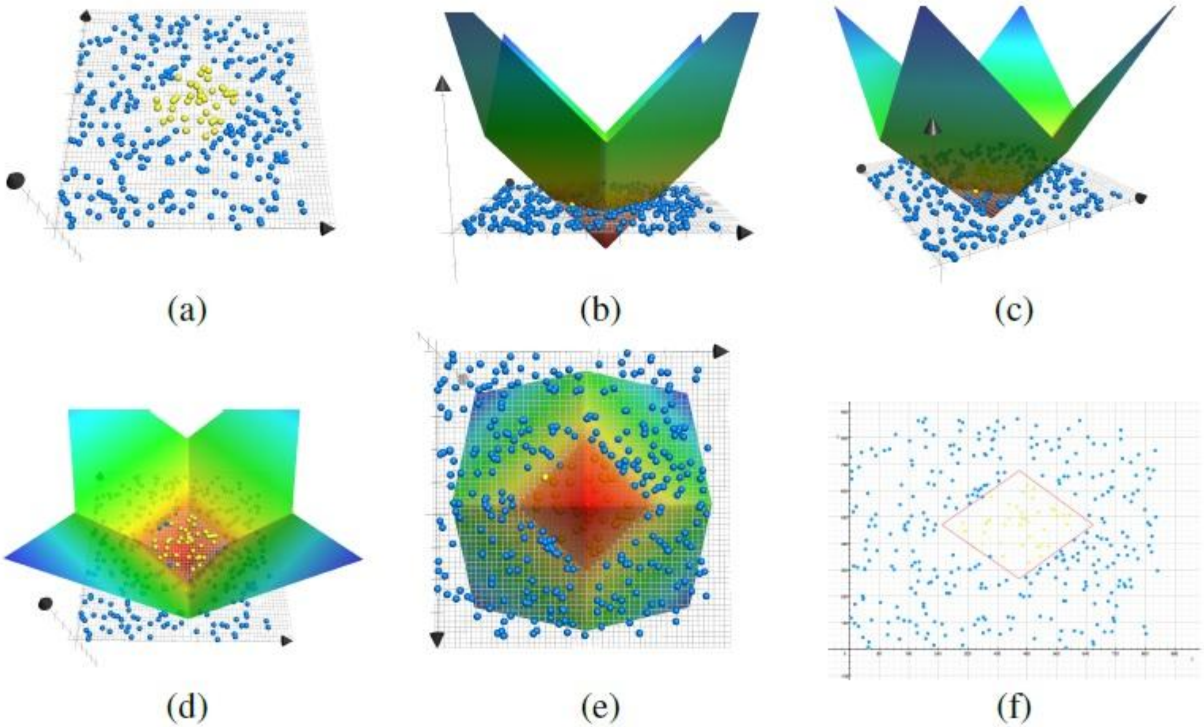


Figure 1: Visualization of PCC classifiers for 2D synthetic data: The positive acceptance regions are "kite-like"' octahedroids containing the points for which a linear hyperplane lies above an L_1 cone.(a): 2D positive (yellow) and negative (blue) samples, (b)-(e): views of positive-class acceptance regions from different angles in 3D, (f): Resulting "kite-like" acceptance region in 2D space.

One-Class PCC/EPCC Classifiers

- SVM formulation does not necessarily guarantee bounded acceptance regions.
- To return both convex and bounded polyhedral acceptacen regions, we need to ensure that slope weights to be less than gamma, i.e.,

$$\gamma > 0, \|\mathbf{w}\|_\infty < \gamma \quad \text{PCC,} \qquad \boldsymbol{\gamma} > 0, |w_i| < \gamma_i \ \text{ for } i = 1, \ldots, d \quad \text{EPCC}$$

The returned class region has a width which is roughly equal to O(b/γ), so we have to ensure that γ cannot shrink to zero for compact acceptance regions.

$$\begin{aligned}
\underset{\mathbf{w},\boldsymbol{\gamma}}{\arg\min} \quad & \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{n_+}\sum_i \xi_i + \frac{1}{n_-}\sum_j \xi_j - \boldsymbol{\kappa}^T\boldsymbol{\gamma} \\
\text{such that} \quad & \mathbf{w}^T(\mathbf{x}_i - \mathbf{s}) + \boldsymbol{\gamma}^T|\mathbf{x}_i - \mathbf{s}| - 1 < \xi_i, \qquad i \in I_+, \\
& \mathbf{w}^T(\mathbf{x}_j - \mathbf{s}) + \boldsymbol{\gamma}^T|\mathbf{x}_i - \mathbf{s}| - 1 \geq 1 - \xi_j, \qquad j \in I_-, \\
& \xi_i, \xi_j \geq 0,
\end{aligned} \qquad \text{OC-EPCC}$$

Comparison of PCC and EPCC Classifiers

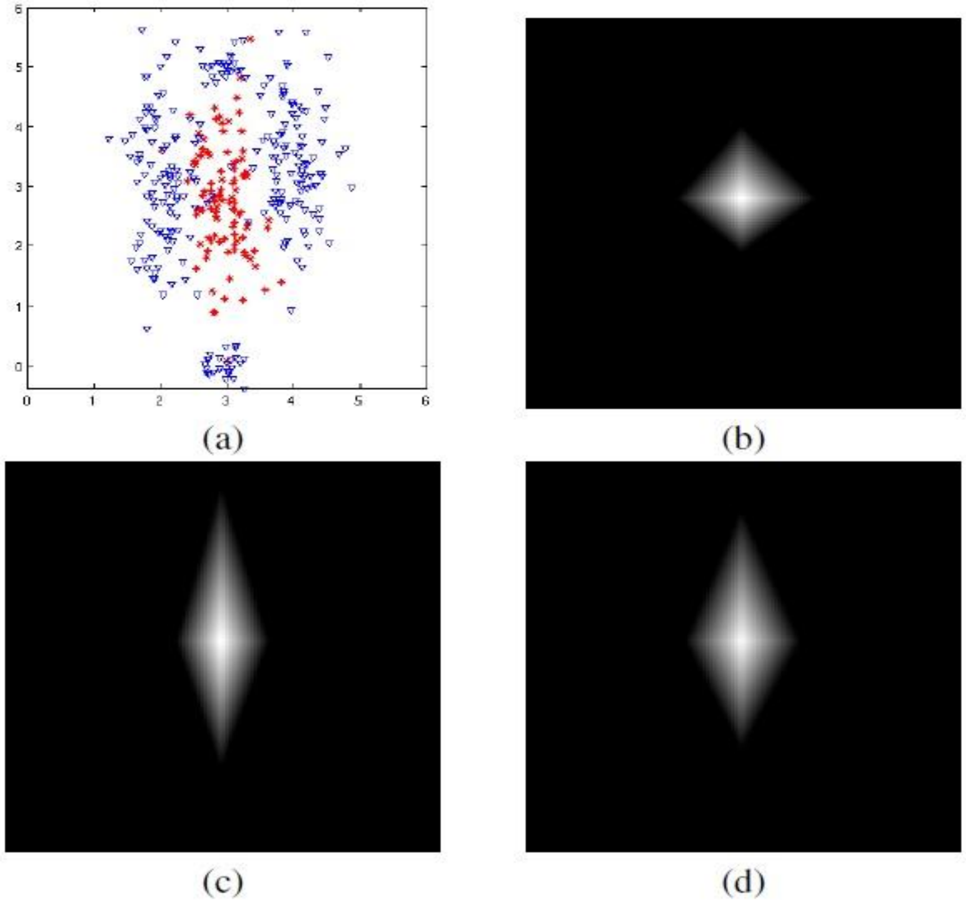


Figure 3. 2D synthetic data set (a) and the decision boundaries returned by (b) PCC, (c) EPCC, (d) OC-EPCC. Brighter pixels correspond to higher scores.

JOBS IN OUR LAB

- We are looking for 3 masters students and 1 Ph. D student to work on different Tubitak projects. Positions will start around September, 2017.

  One project involves aerial visual object tracking (mostly visual tracking on videos captured with drones) and we will hire 2 MS students and 1 Ph. D. student for this project. The other project is related to visual object detection and we will hire 1 MS student for this.
  For Ph. D position, the candidates satisfying the following conditions will be preferred:

  - Background in computer vision and pattern recognition,
  - C++ and Matlab programming experience under Linux,
  - Fluent spoken and written English.