

# Object Detectors overview

Alexander Kozlov

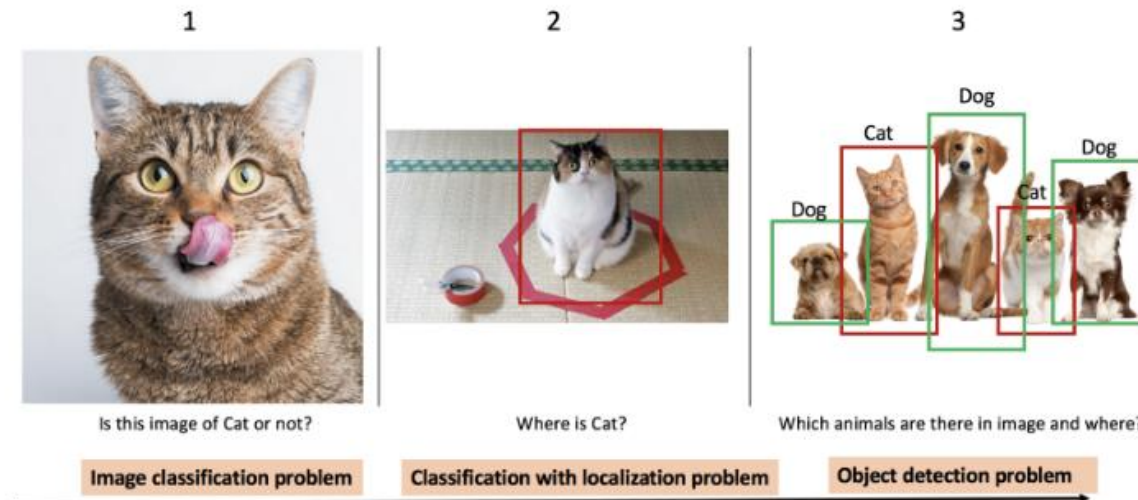
Internet of Things Group

# Agenda

- Problem definition
- Classical approaches
- Deep Learning approaches

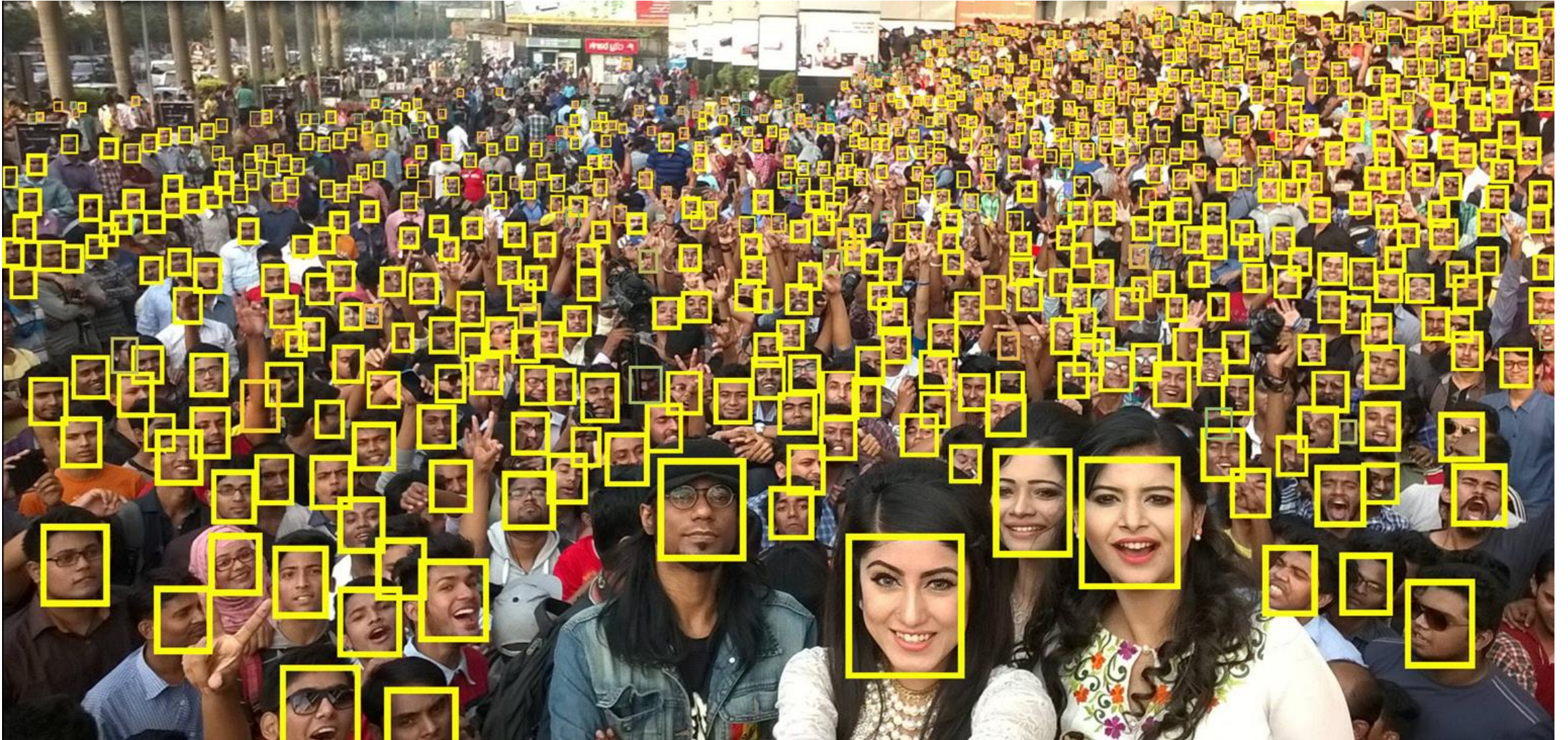
# Problem Definition

- Find objects on the image
  - Coordinates of bounding boxes
  - Class probabilities
- Basically used in conjunction w/ other algorithms





# Example: face detection (one class)



Images credit: P. Hu and D. Ramanan "Finding Tiny Faces". World's largest selfie



## Example: object detection on the road (several classes)



Images credit: A. Kozlov et al. "Development of Real-time ADAS Object Detector for Deployment on CPU"

# Example: object detection in 3D space



Images credit: X. Chen et al. "Multi-View 3D Object Detection Network for Autonomous Driving"

# Metric

- Intersection over Union (IoU) of two bounding boxes can be computed as:

`(a & b).area() / (a.area() + b.area() - (a & b).area())`, not `(a & b).area() / (a | b).area()`

- Average precision

*TP* = True positive

*TN* = True negative

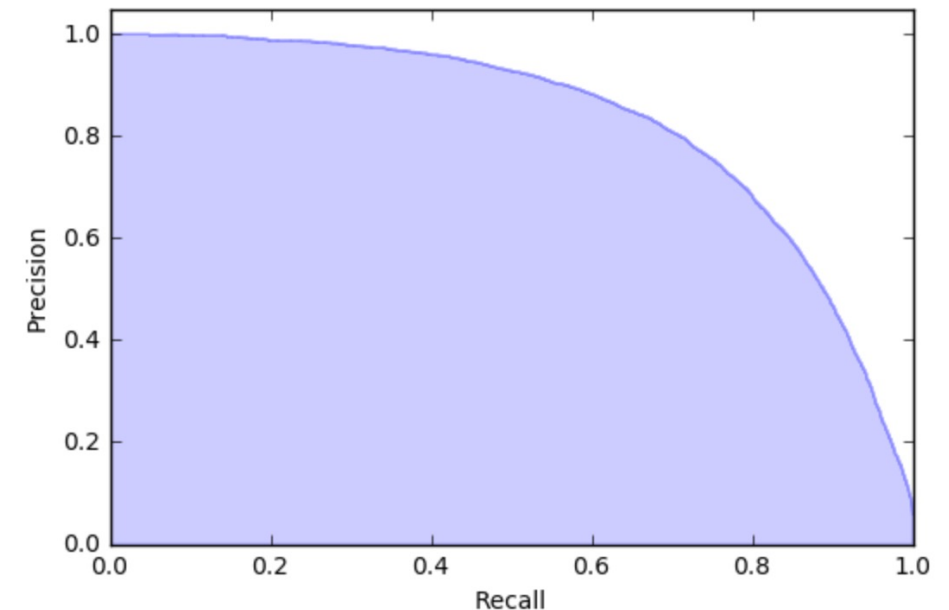
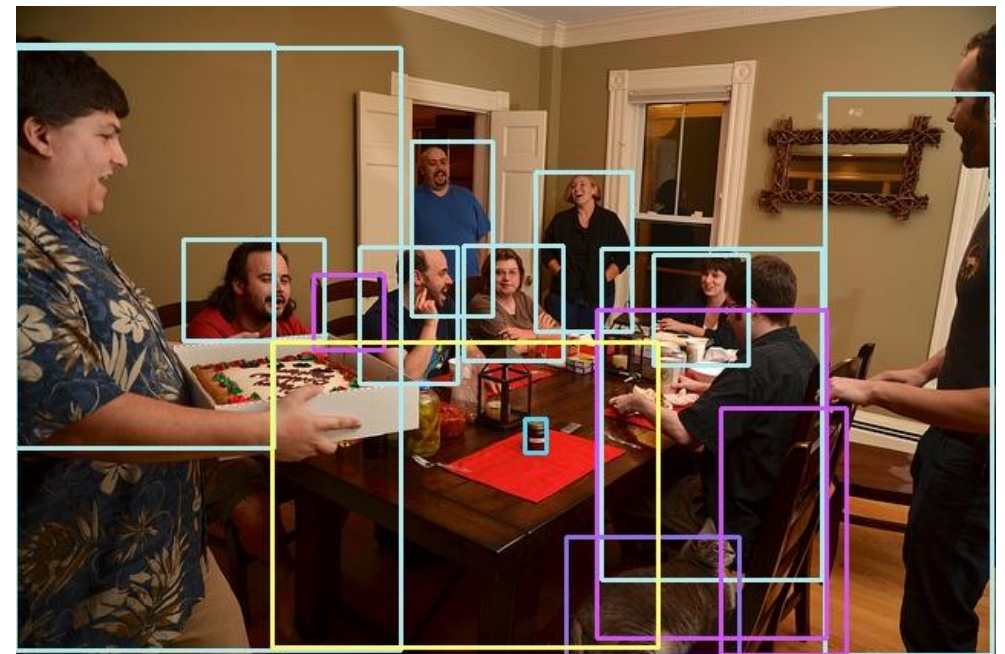
*FP* = False positive

*FN* = False negative

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

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

$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0))$$





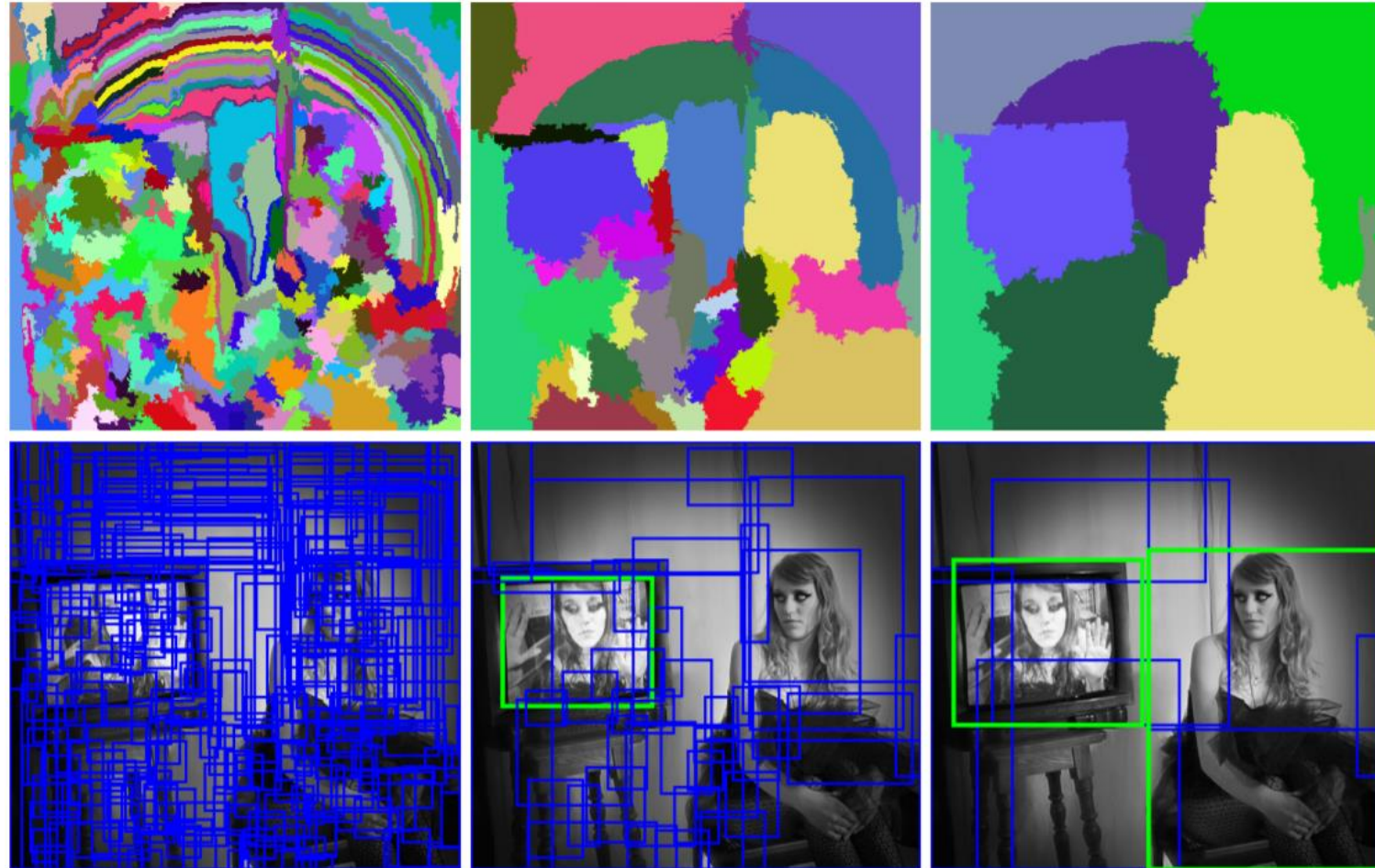
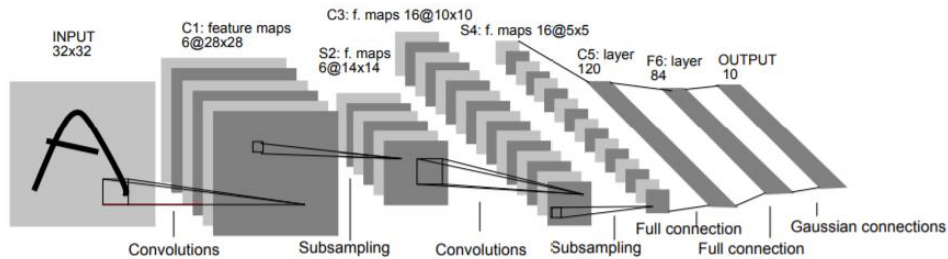
# Object detection

## Localization

- Generate hypothesis about object location

## Classification

- Hypothesis verification



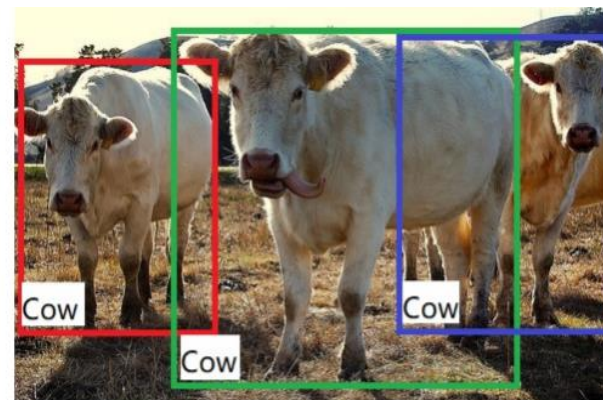
Images credit: J.R.R. Uijlings et al. "Selective Search for Object Recognition", Y. LeCun "Gradient-based learning applied to document recognition"



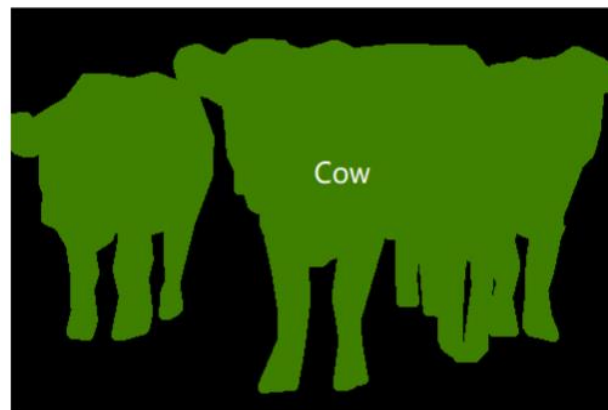
# Connected problems



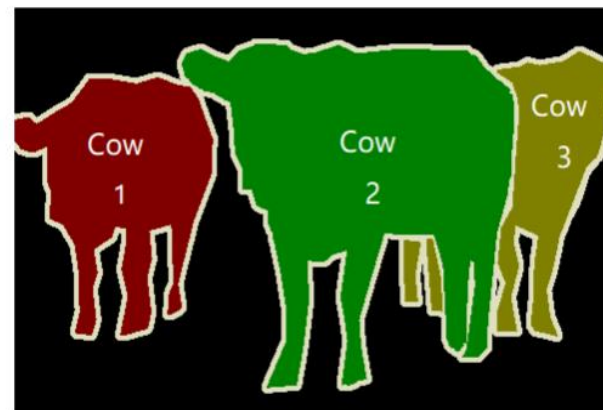
(a) Image Classification



(b) Object Detection



(c) Semantic Segmentation

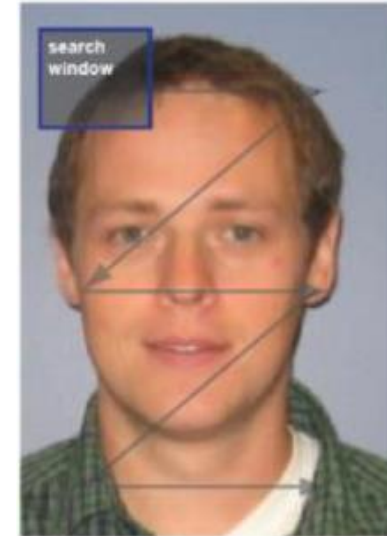


(d) Instance Segmentation

# Localization: sliding window

To find the coordinates of the object, a “*sliding window*” is used: the classifier is applied to every possible window arrangement

To detect objects of different sizes, a *pyramid of images* is used.



⋮



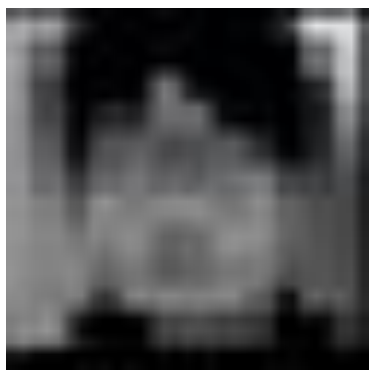
...



# Binary classification: features

Training dataset: set of object-label pairs :  $(x_i, y_i), i=1..N$ .

$x_i$  – feature-vector  $\in R^n$ ,  $y_i$  – object class  $\in \{0, 1\}$ .



16x16 pixels

$x_i = (128, 128, 60, \dots, 0, 0)$ , features – values of pixels.

$y_i = 1$ .

Feature-vector has the same size for any object



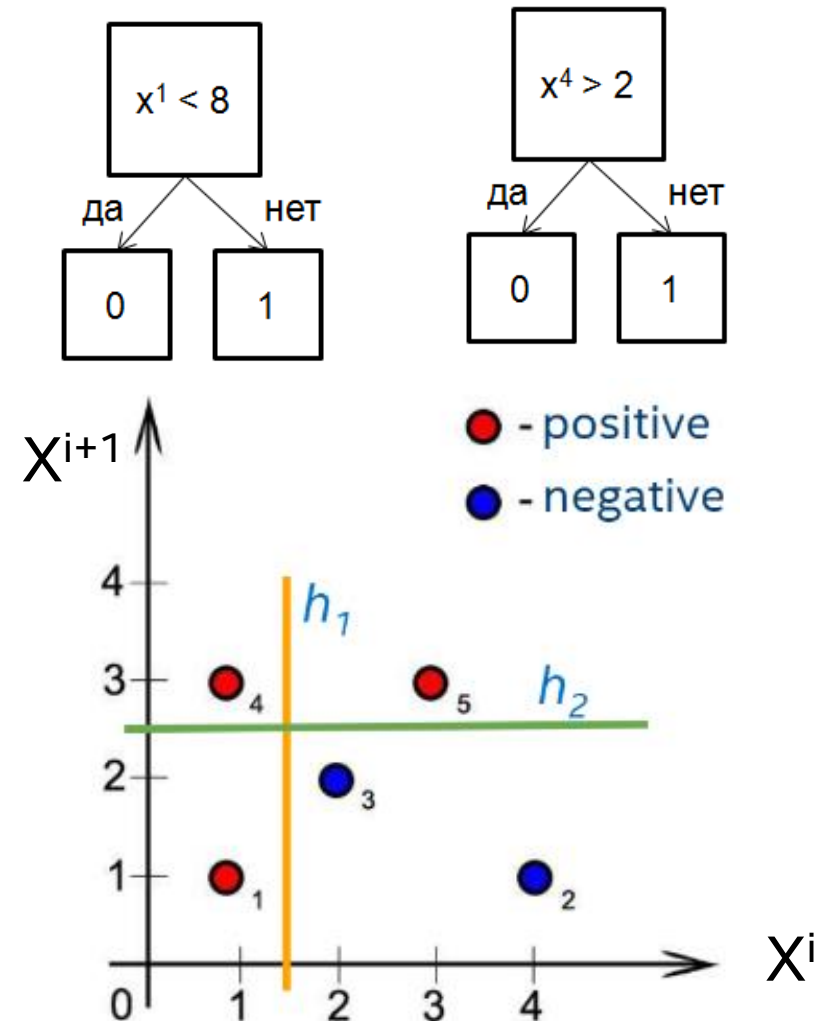
# Binary classification: decision tree

Splits the feature space by one (multiple) coordinates.

Parameters of a split (sign and threshold) are selected to minimize the classification error:

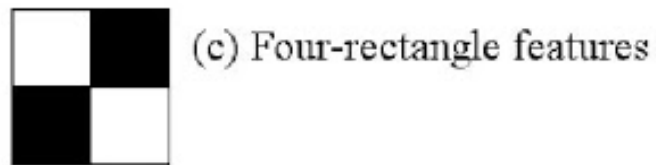
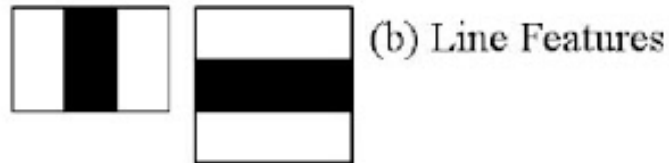
$$\sum_{i=1}^N |label_{gt} - label_{predicted}|$$

Several “weak” classifiers are combined into one strong (AdaBoost).



# Haar features (2004)

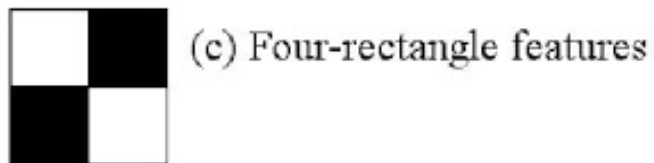
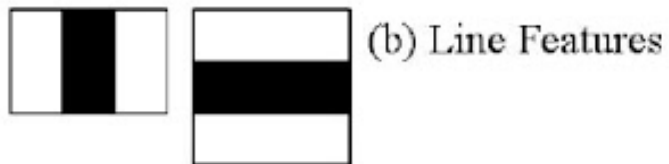
Feature-vector is formed from the values of the convolution with one of the pre-defined kernels. Kernels (black = -1, white = 1):



Images credit: <https://www.youtube.com/watch?v=zLBAJ93-AEQ>

# Haar features (2004)

Feature-vector is formed from the values of the convolution with one of the pre-defined kernels calculated in each position on the image  
Kernels (black = -1, white = 1):





# What is the difference between features?

Good features are invariant:

- To light conditions
- To scale
- To rotation

Popular features:

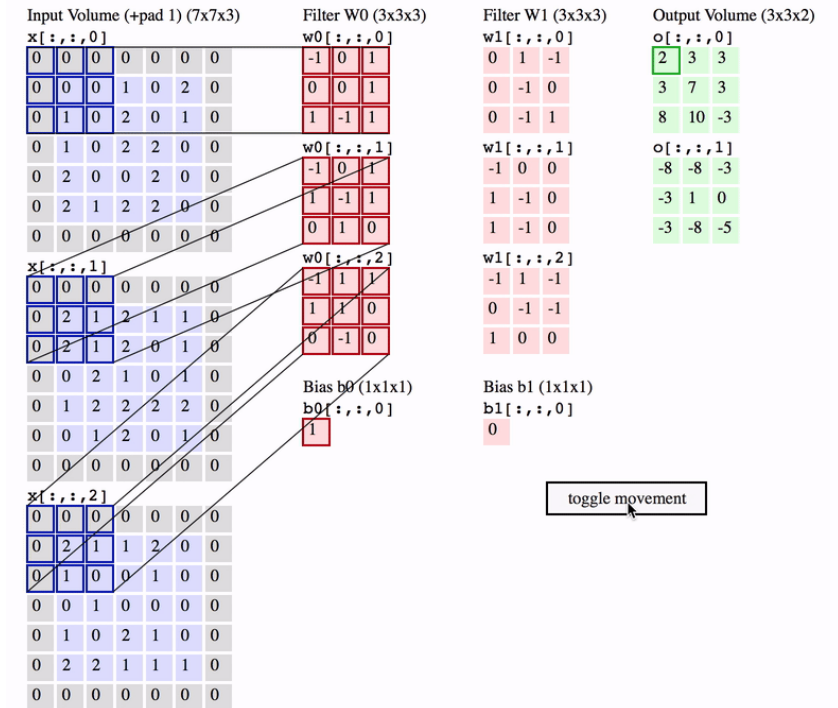
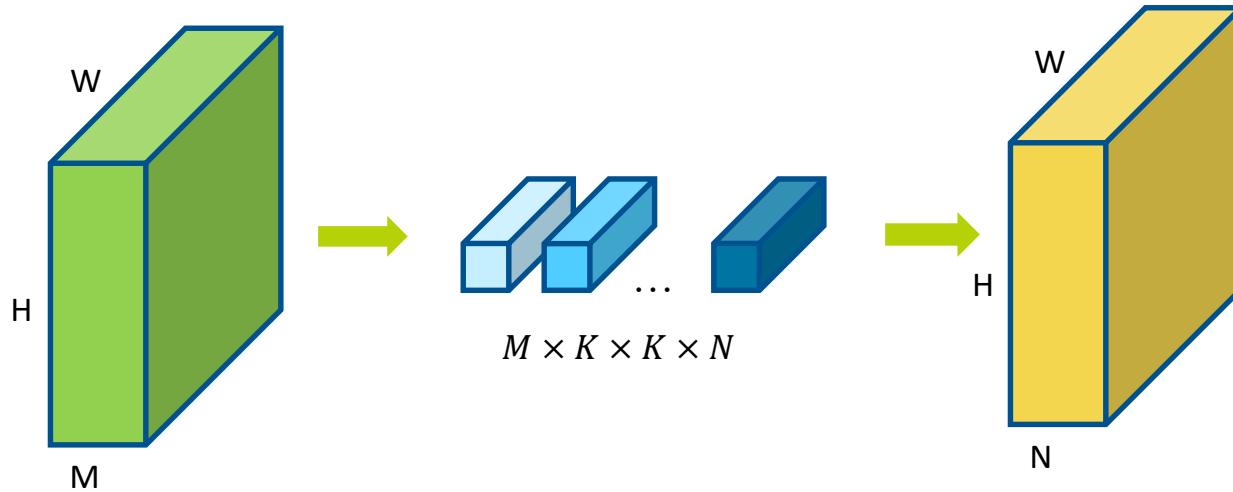
**LBP:** Local Binary Patterns (1994) - T. Ojala et al. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions".

**HOG:** Histogram of Oriented Gradients (2005) - N. Dalal et al. "Histograms of Oriented Gradients for Human Detection".

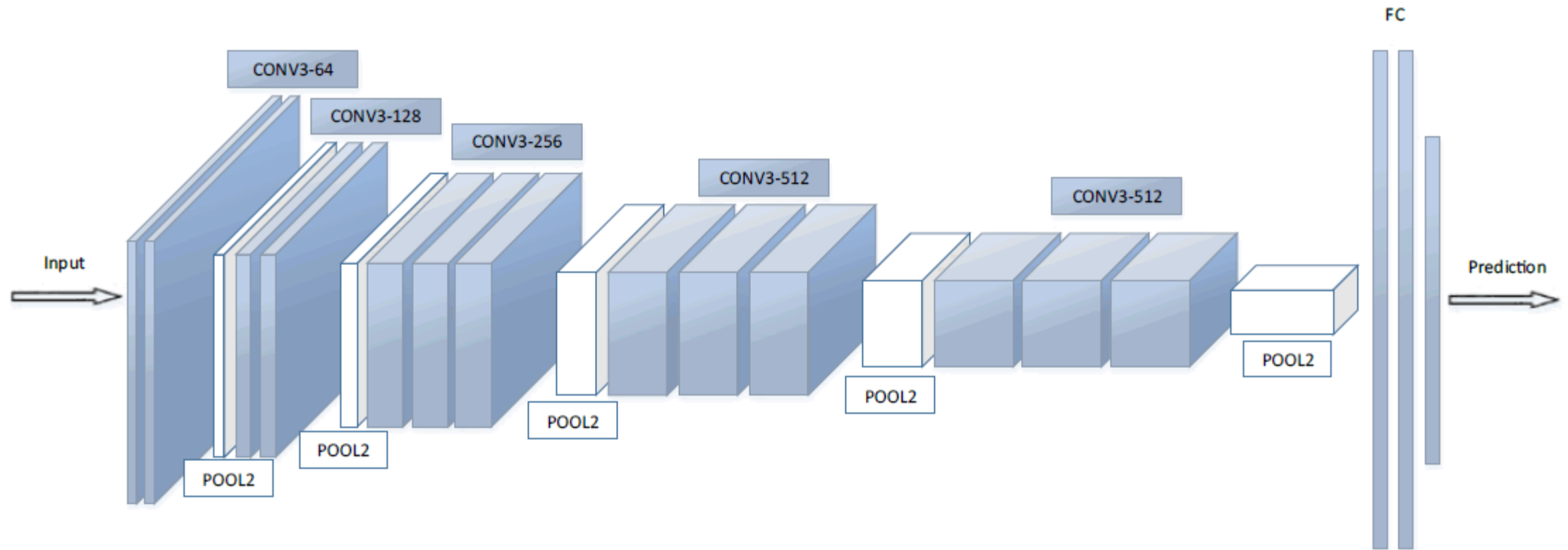
**ICF:** Integral Channel Features (2009) - P. Dollár et al. "Integral Channel Features".

**FCF:** Filtered Channel Features (2015) - S. Zhang et al. "Filtered Channel Features for Pedestrian Detection".

# Convolution



# VGG CNN





# Non-Maximum Suppression (NMS)

- Detector outputs multiple detections for the same object
- NMS discards all except one with best features, e.g. with highest score
- Can be learnable



# R(egion based)-CNN (CVPR, 2014)

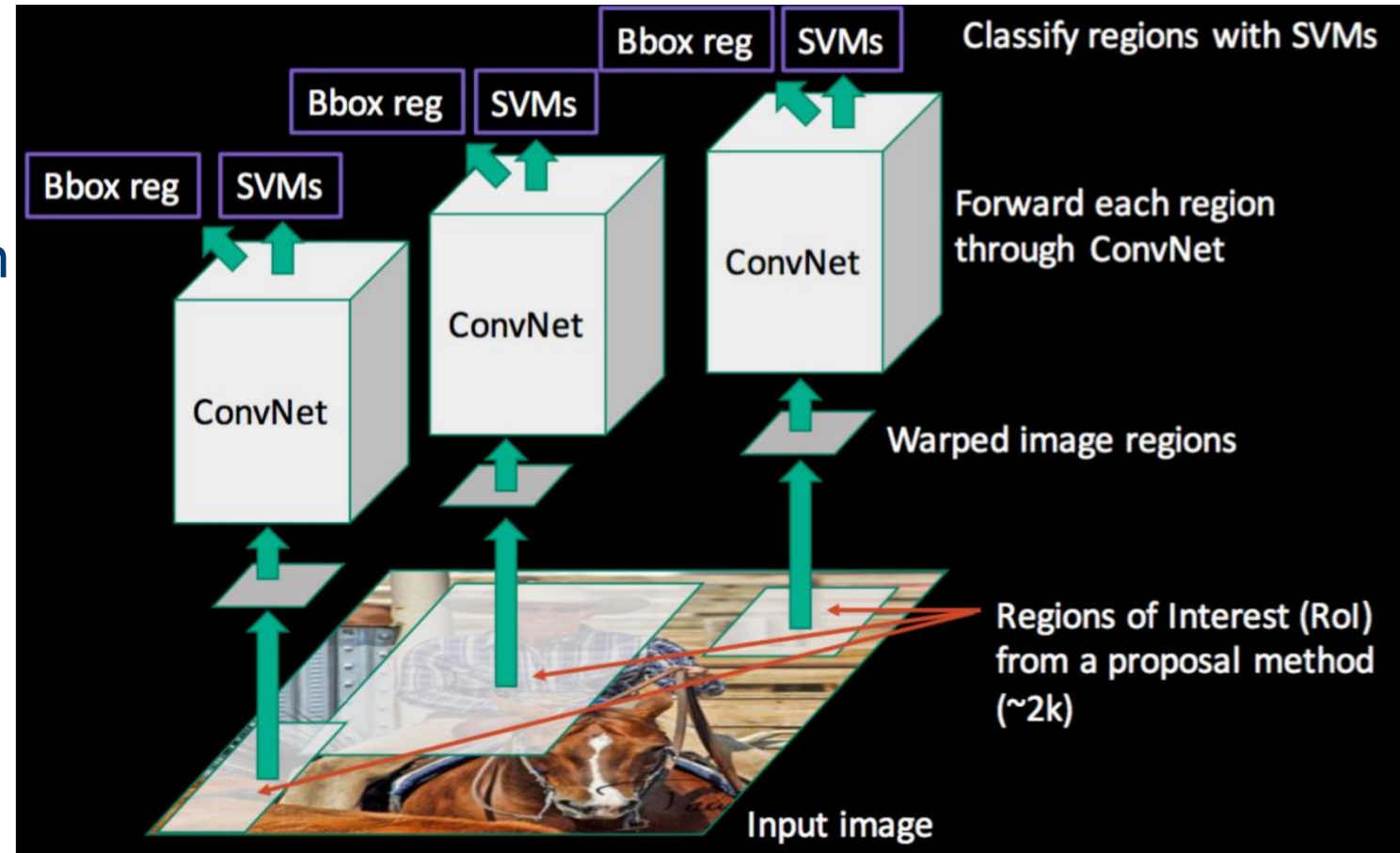
## Motivation:

- Deep learning-based classifiers are known to be powerful but quite slow
  - Operating in a sliding window fashion seems to be very slow
  - Good proposal generation stage can potentially resolve the issue



# R(egion based)-CNN (CVPR, 2014)

- Region proposal algorithm (Selective Search) is used to get RoIs
- Off-the-shelf image classification net (AlexNet) is used to extract features for every RoI
- SVM classifier is used to classify RoIs as objects or background
- Linear regression is applied to localize bounding boxes inside RoIs



# Fast R-CNN (2015)

## Drawbacks of R-CNN:

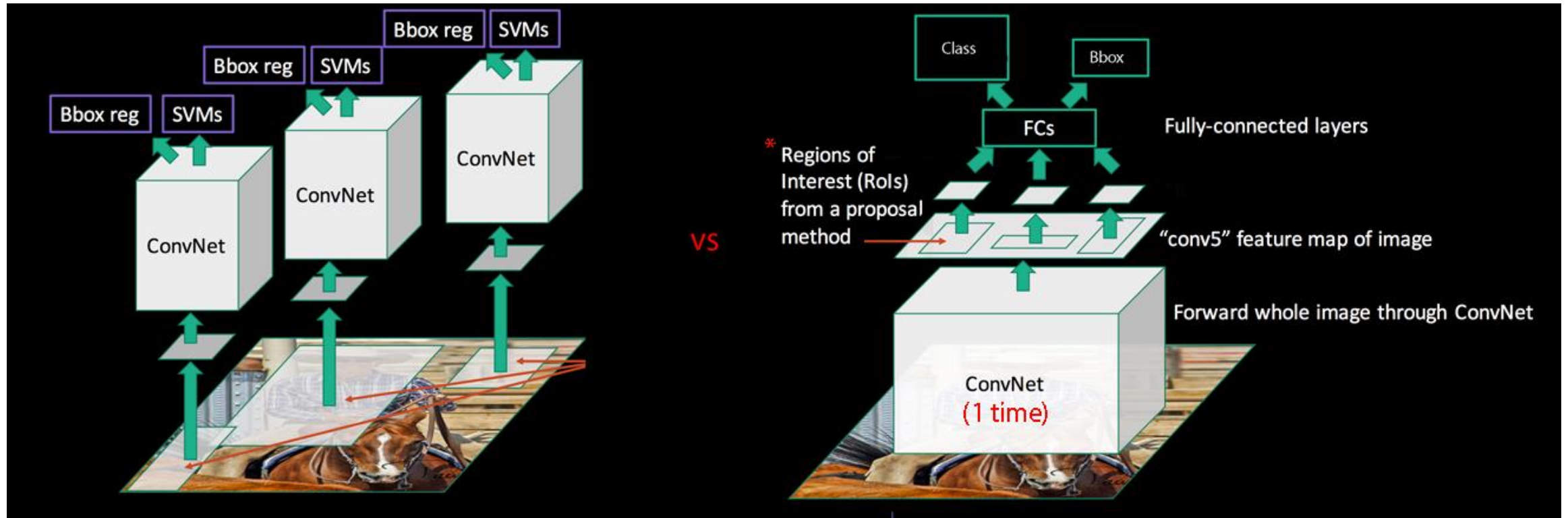
- Complicated multi-stage model is hard to train
- Despite the use of Selective Search is slow at test time

## Solution – Fast R-CNN:

- Merge classifier and regressor to the convnet itself to train it end-to-end
- Apply convnet to the whole image and crop RoIs on high-level feature map to make detection faster



# Fast R-CNN (2015)



Deep features compute once per image, not per proposal



# Faster R-CNN

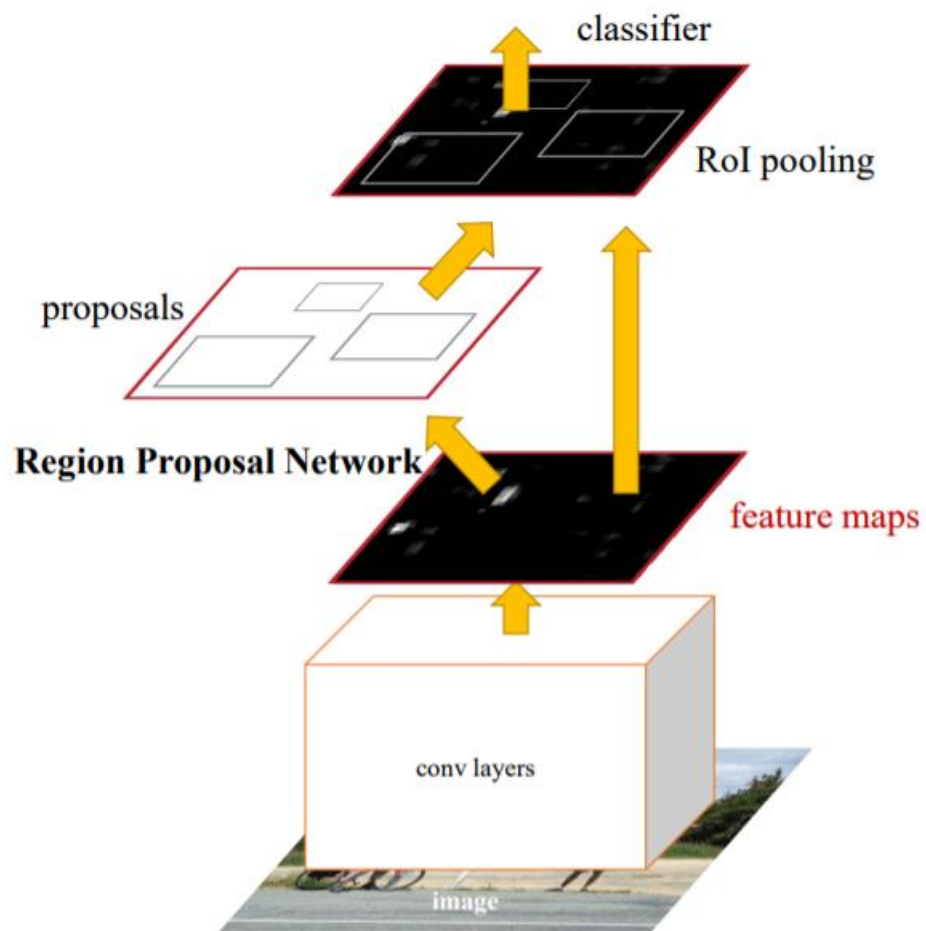
## Drawbacks of Fast R-CNN:

- 25x faster than R-CNN, but still too slow. Mostly because of the Selective Search now (~2s per VGA image)

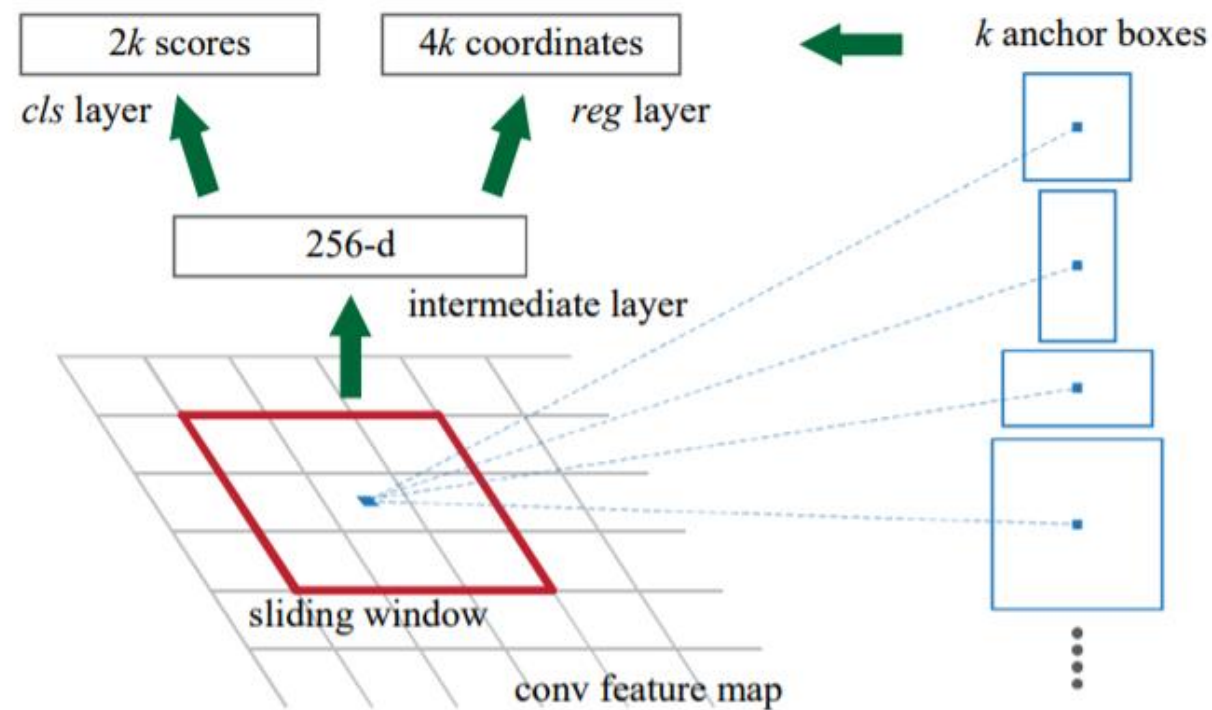
## Solution – Faster R-CNN:

- Merge region proposal stage into the net as well

# Faster R-CNN (2015)

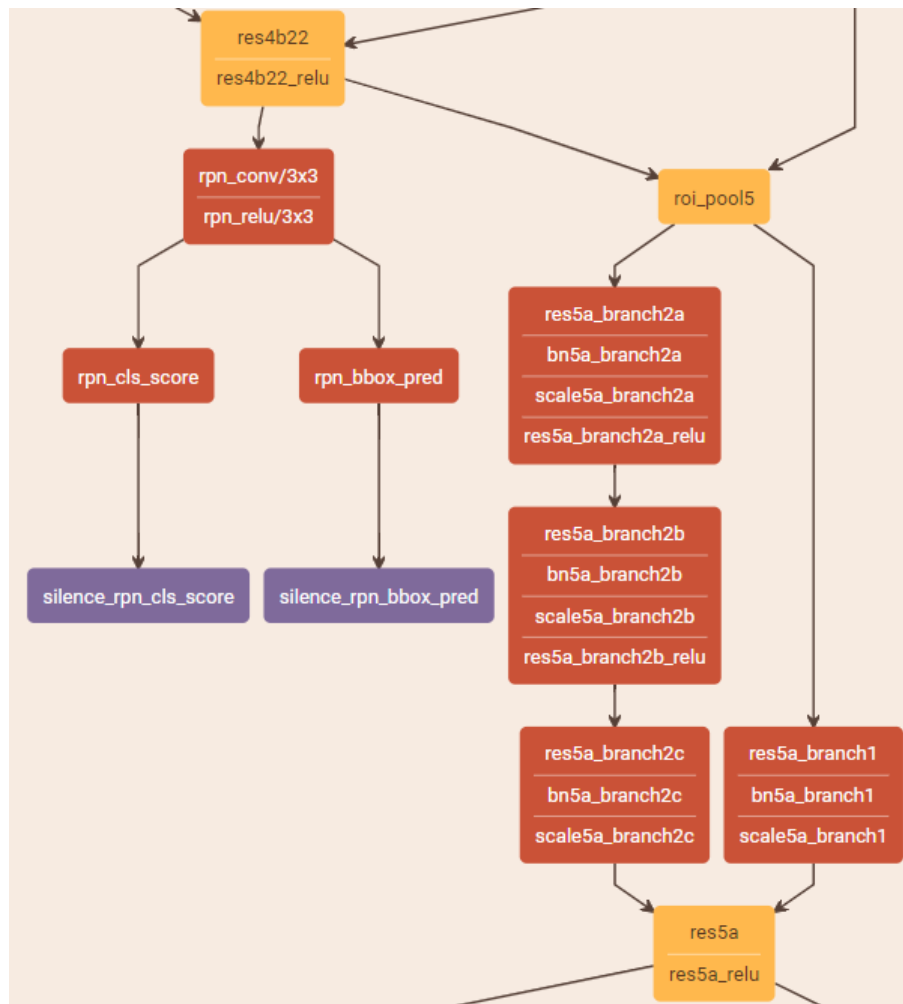


Faster R-CNN pipeline

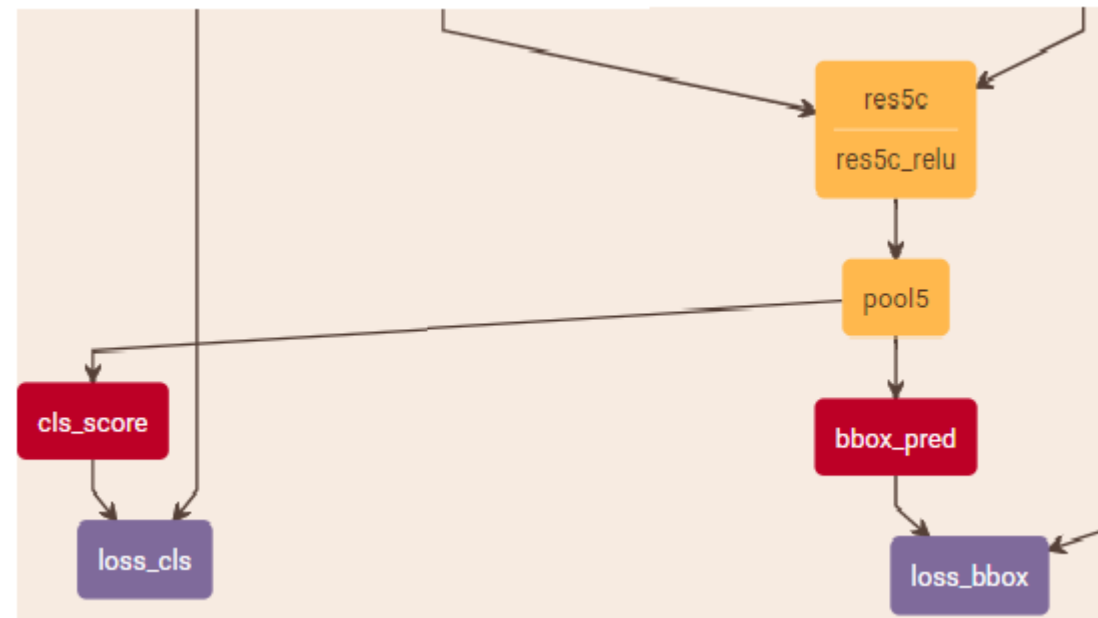


Region Proposals Network

# Faster R-CNN With Resnet 101 Example



...



# R-FCN

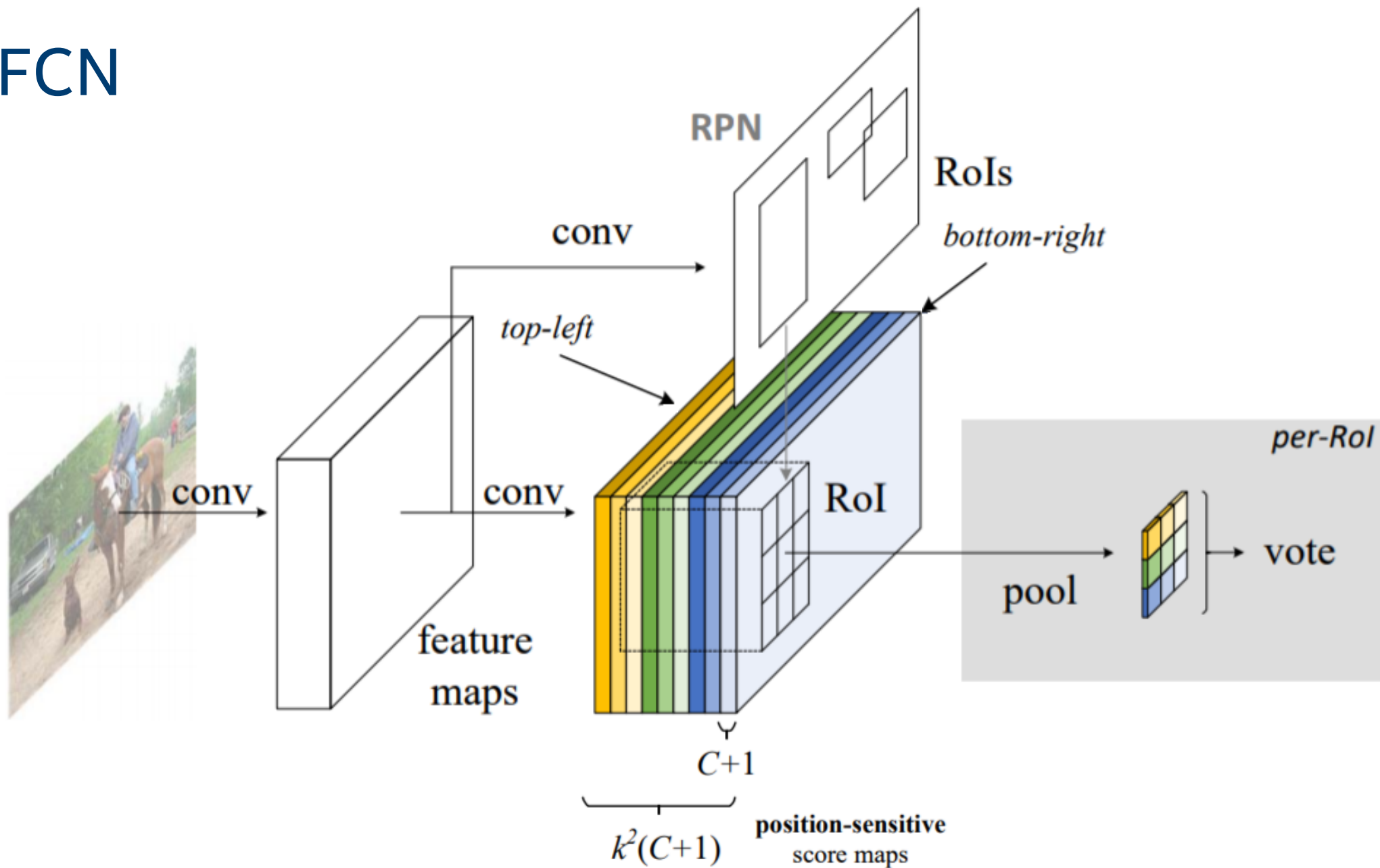
## Motivation:

- Made object detection network with 100% shared computations
- Allow translation variance to localize object position

Methodologies of *region-based* detectors using **ResNet-101**

	R-CNN [7]	Faster R-CNN [19, 9]	R-FCN [ours]
depth of shared convolutional subnetwork	0	91	101
depth of RoI-wise subnetwork	101	10	<b>0</b>

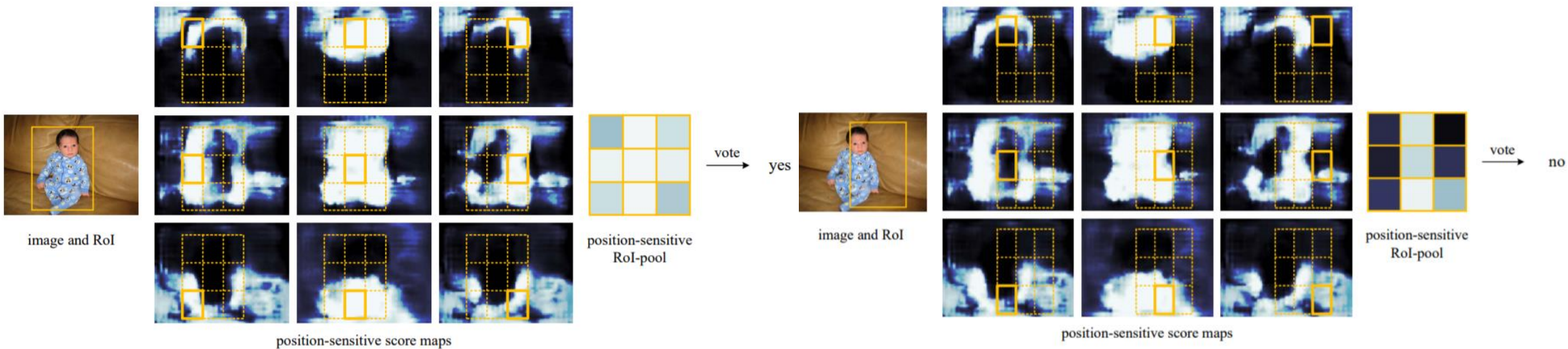
# R-FCN





# R-FCN

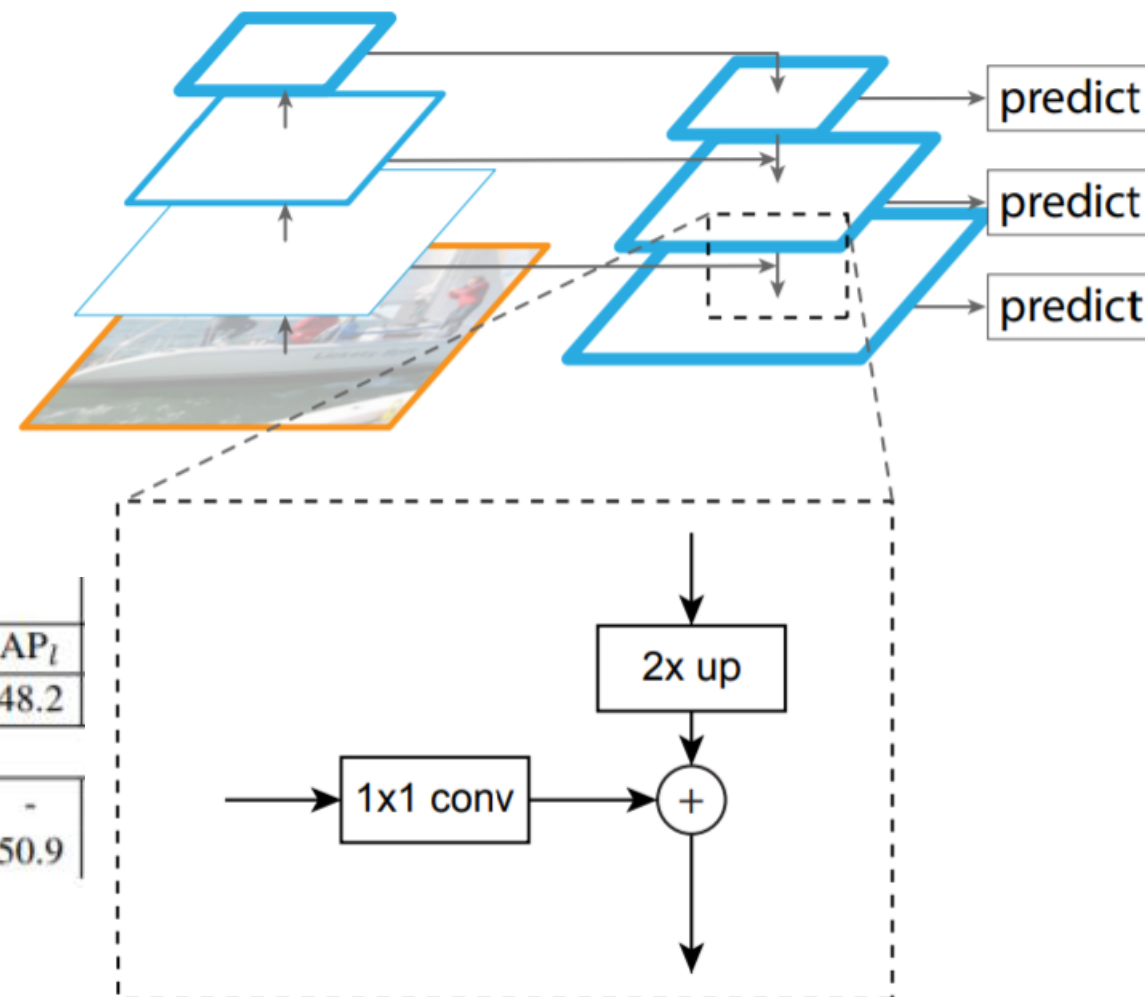
## Position-sensitive score maps



# Feature Pyramid Network (FPN)

## Motivation:

- Leverage the pyramidal shape of a ConvNet's feature hierarchy
- Provide strong semantics at all scales

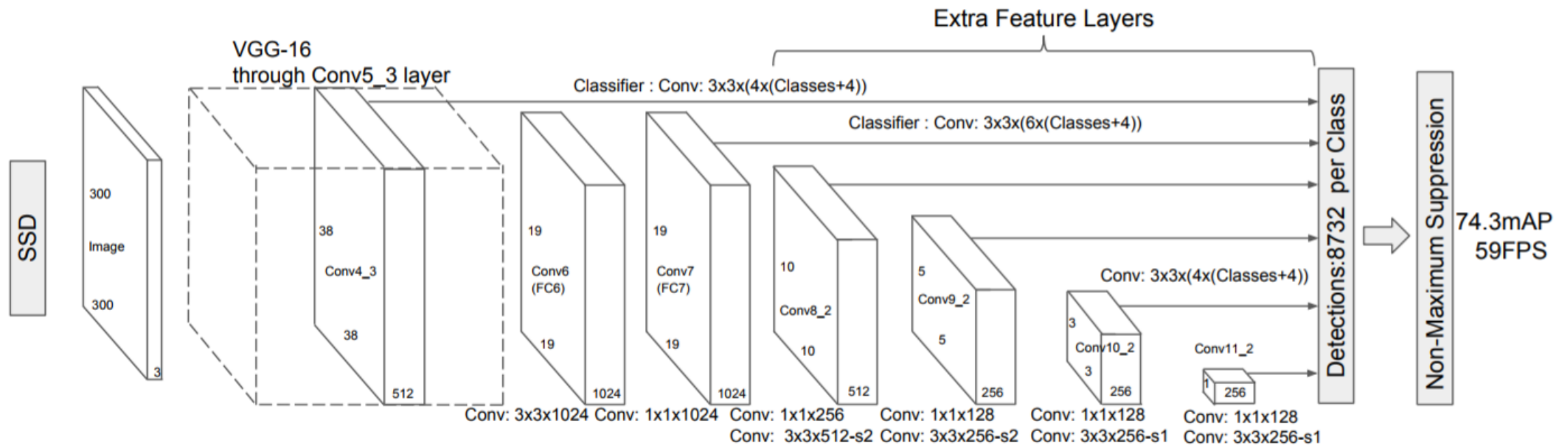


method	backbone	competition	test-dev			
			AP	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
ours, Faster R-CNN on <b>FPN</b>	ResNet-101	-	<b>36.2</b>	<b>18.2</b>	<b>39.0</b>	48.2
<i>Competition-winning single-model results follow:</i>						
G-RMI <sup>†</sup>	Inception-ResNet	2016	34.7	-	-	-
Faster R-CNN +++	ResNet-101	2015	34.9	15.6	38.7	50.9

# Single Shot Multibox Detector

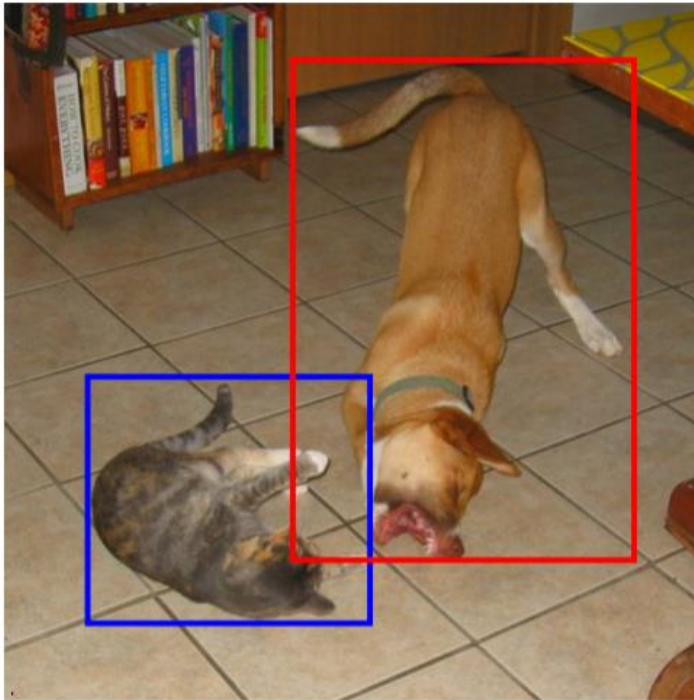
## Motivation:

- Region proposal net can be eliminated at all
- Make multi-scale detection efficient

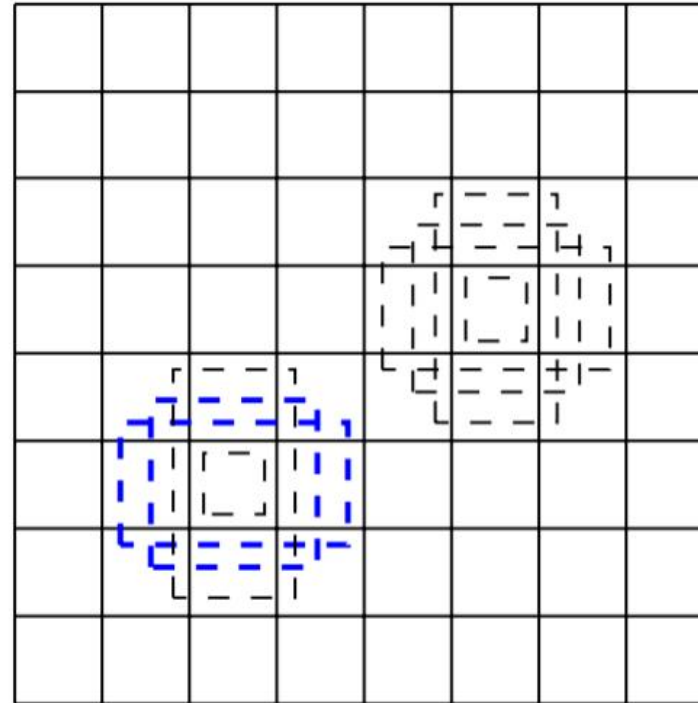


# Single Shot Multibox Detector

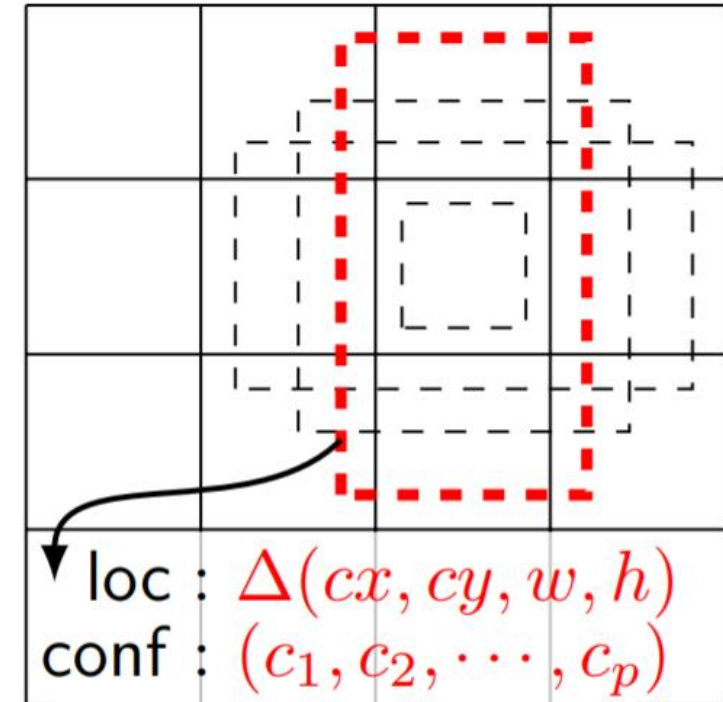
## Default boxes



(a) Image with GT boxes



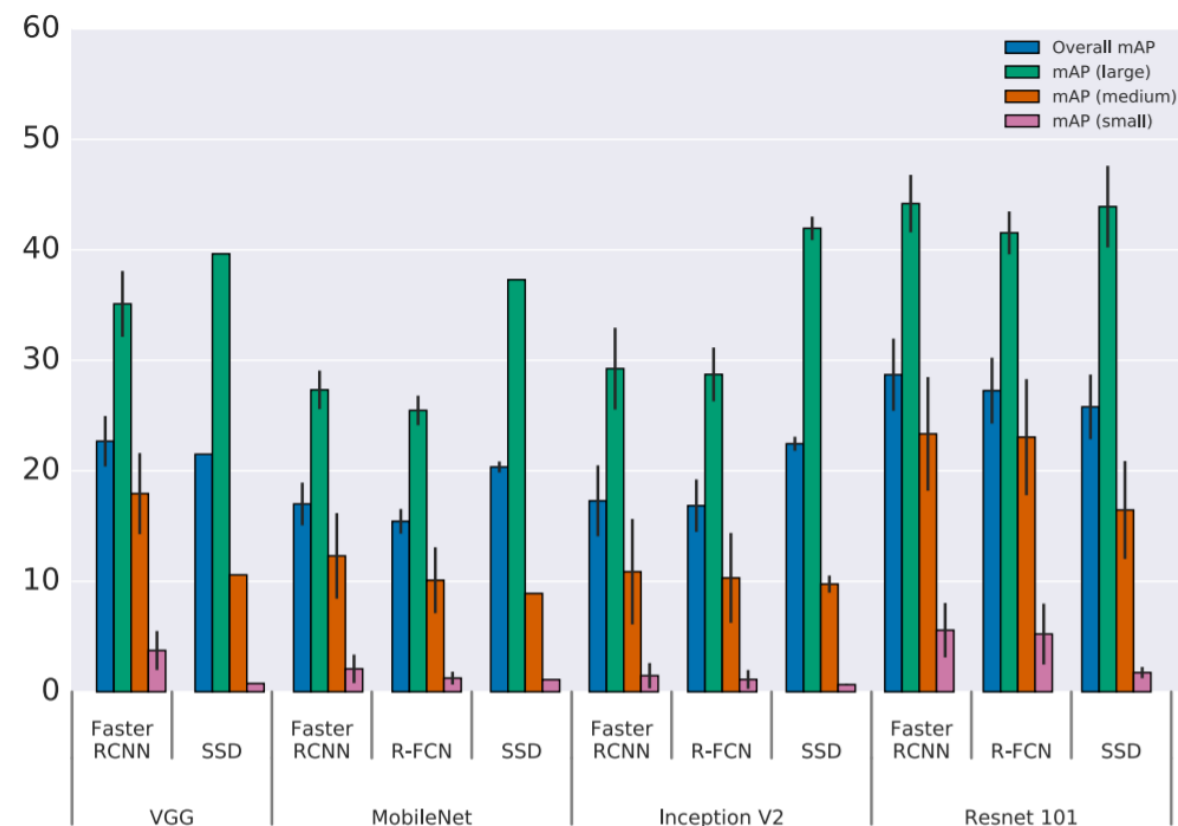
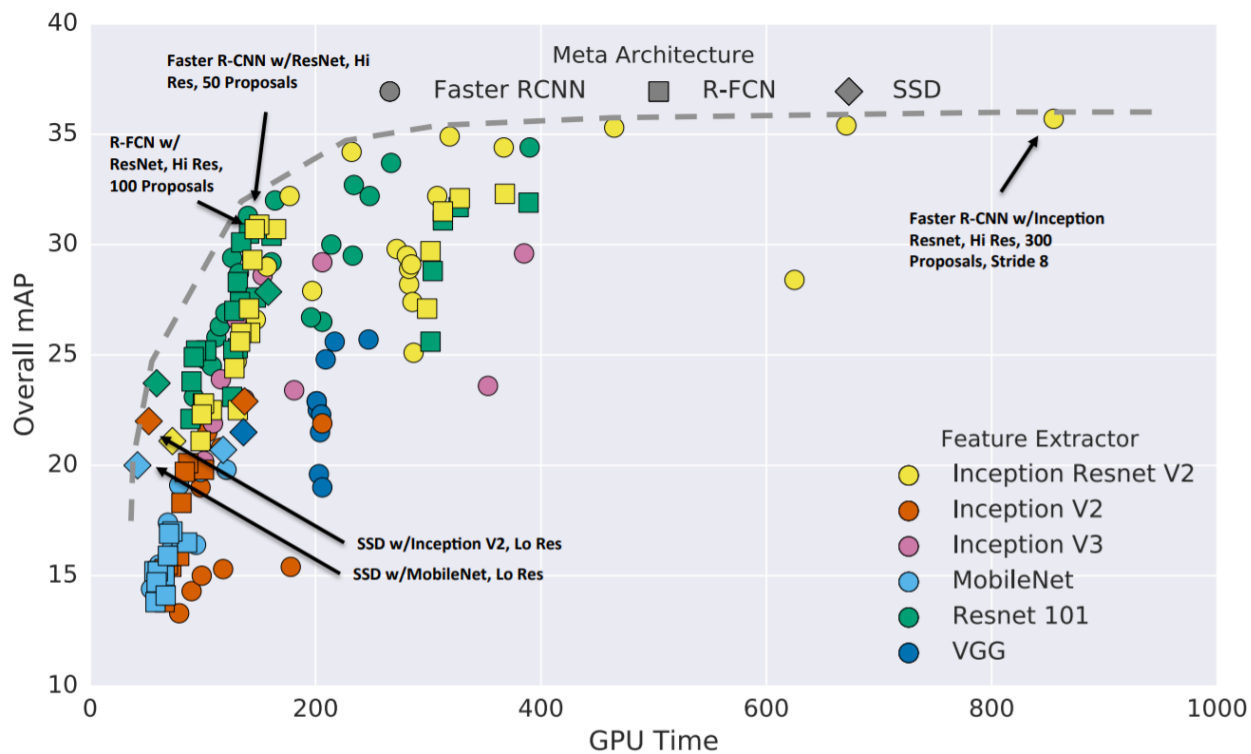
(b)  $8 \times 8$  feature map



(c)  $4 \times 4$  feature map

loc :  $\Delta(cx, cy, w, h)$   
conf :  $(c_1, c_2, \dots, c_p)$

# Detectors Speed-Accuracy trade-offs





# Transfer Learning

Very few people train an entire Convolutional Network from scratch (with random initialization)

It is common to pre-train a ConvNet on a very large dataset (e.g. ImageNet), and then use the ConvNet for the task of interest, major scenarios:

- ConvNet as fixed feature extractor
  - Take a ConvNet pre-trained on ImageNet, remove the classification layer (which outputs the 1000 class scores), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset
- Fine-tuning the ConvNet
  - Not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pre-trained network
- Pre-trained models
  - Use one of the uploaded pre-trained models for own task

# Recent state-of-the-art CNNs

- RetinaNet
- RefineDet
- Mask R-CNN (+instance segmentation)
- Cascade R-CNN
- Deformable Convolution Networks
- **Anchor-free methods:** CenterNet, CornerNet

# Popular OD repositories

- Tensorflow Object Detection API: [https://github.com/tensorflow/models/tree/master/research/object\\_detection](https://github.com/tensorflow/models/tree/master/research/object_detection)
- Detectron (Caffe2): <https://github.com/facebookresearch/Detectron>
- mmdetection (PyTorch): <https://github.com/open-mmlab/mmdetection>

# OpenVINO OD models

- OpenVINO: <https://github.com/openvinotoolkit/openvino>
- Open Model Zoo: [https://github.com/opencv/open\\_model\\_zoo](https://github.com/opencv/open_model_zoo)
  - Face detection (2 variants)
  - Person detection (2 variants)
  - Vehicle detection (3 variants)
  - Public models: SSD, Faster-RCNN, R-FCN, etc.

# Q&A