Summer Camp 2021

Object Detectors overview

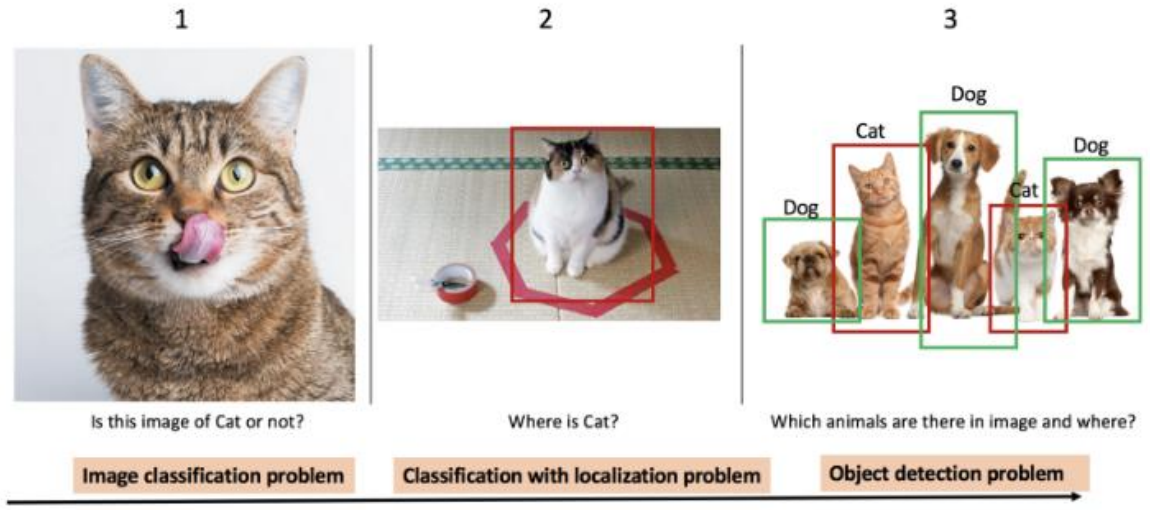
Davidyuk Igor

Agenda

- Problem statement
- Classical approaches
- Deep Learning approaches

Problem Definition

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



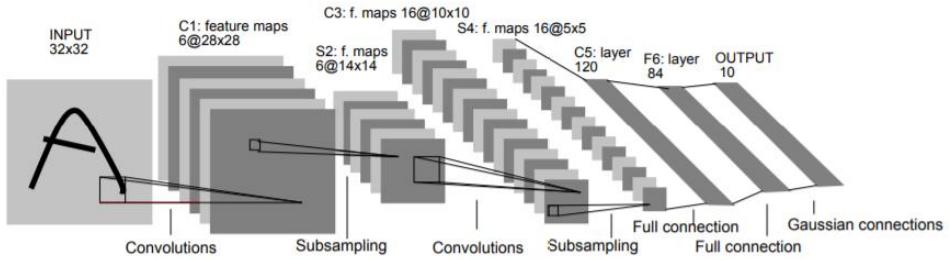
Object detection

Localization

- Generate hypothesis about object location

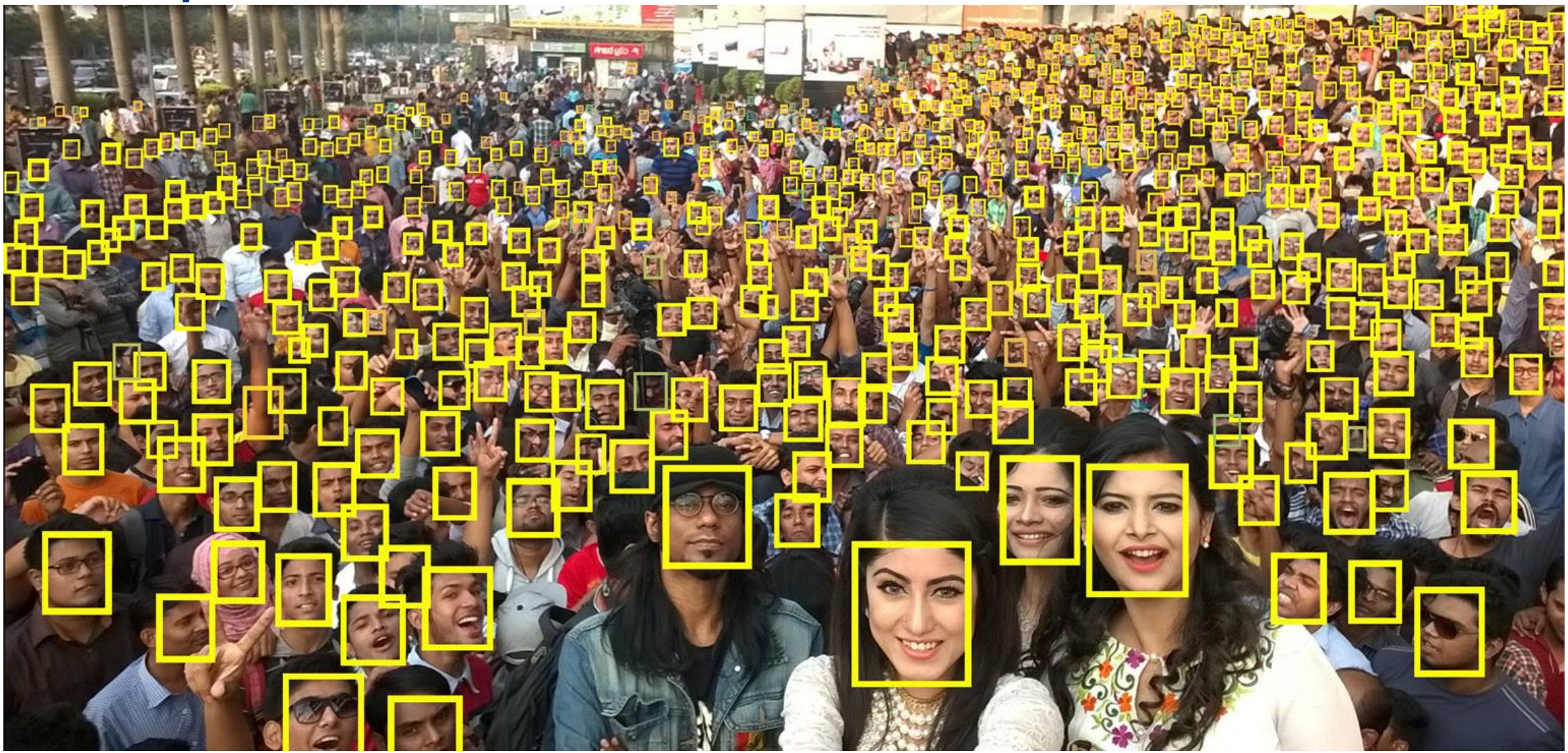
Classification

- Hypothesis verification



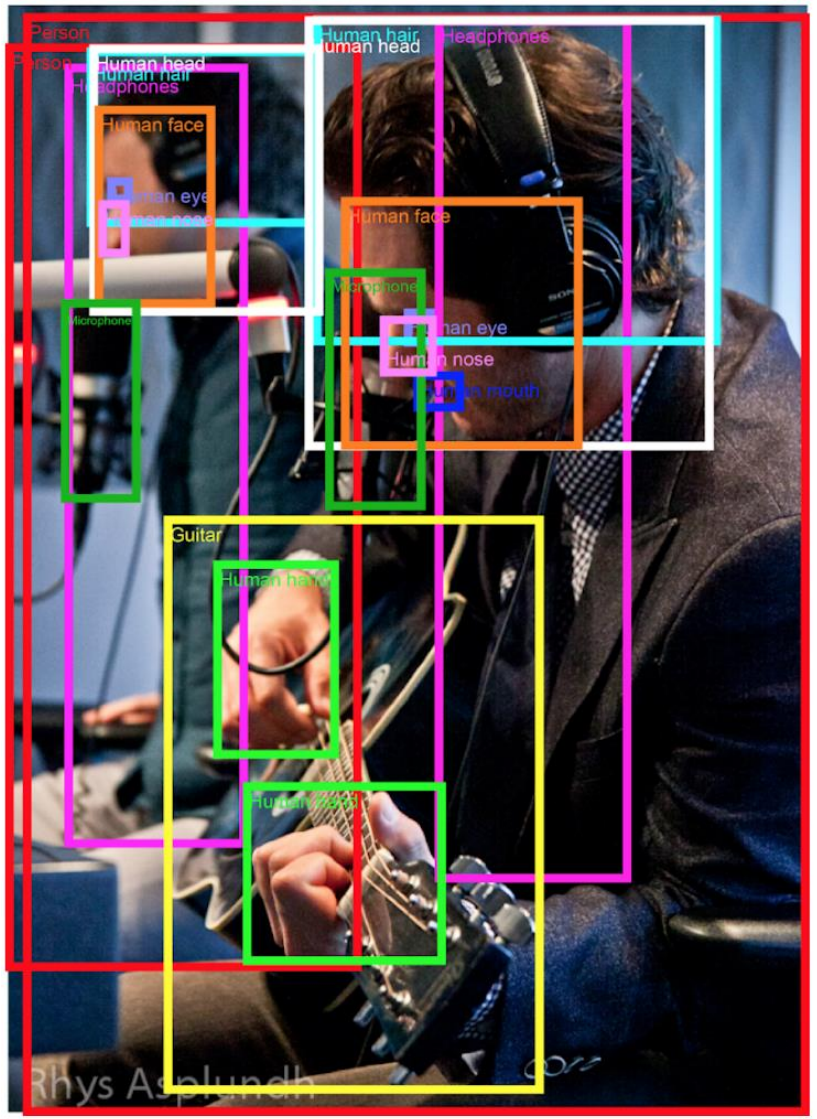
Images credit: J.R.R. Uijlings et al. “Selective Search for Object Recognition”, Y. LeCun “Gradient-based learning applied to document recognition”

Example: face detection (one class)



Images credit: P. Hu and D. Ramanan “Finding Tiny Faces”. World’s largest selfie

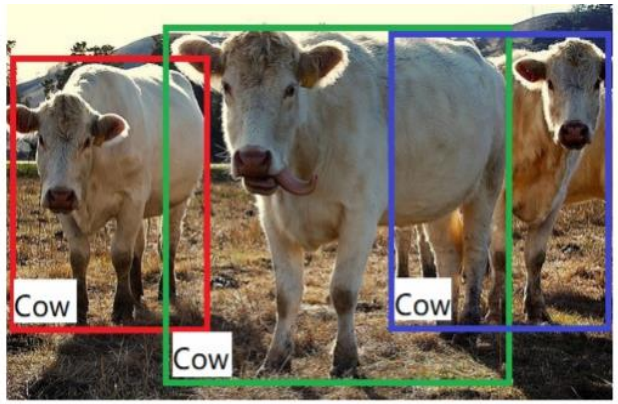
What objects are where?



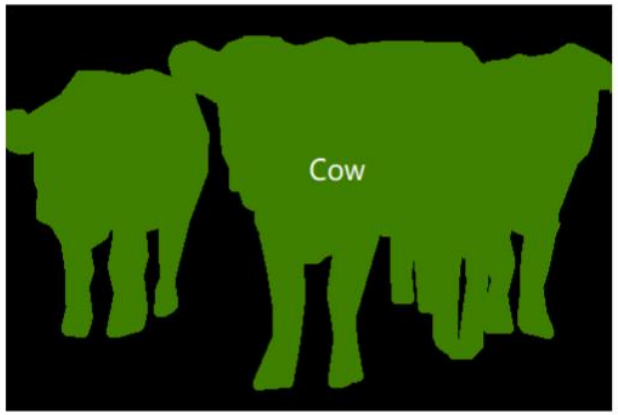
Images credit: Open Images dataset

Connected problems

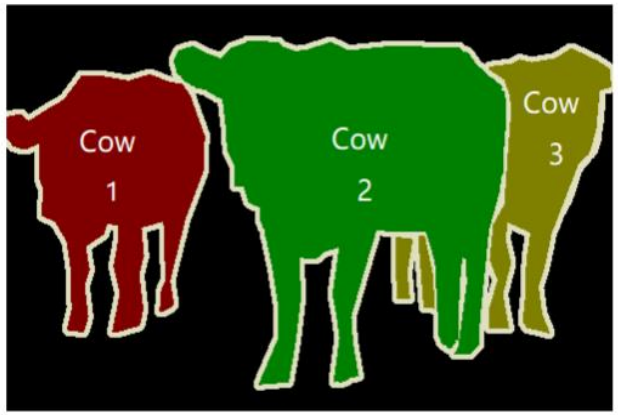
(a) Image Classification



(b) Object Detection



(c) Semantic Segmentation



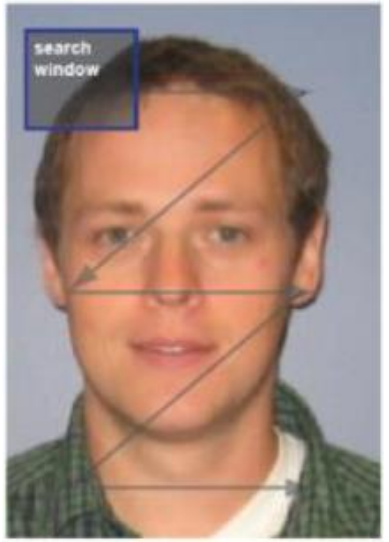
(d) Instance Segmentation

Agenda

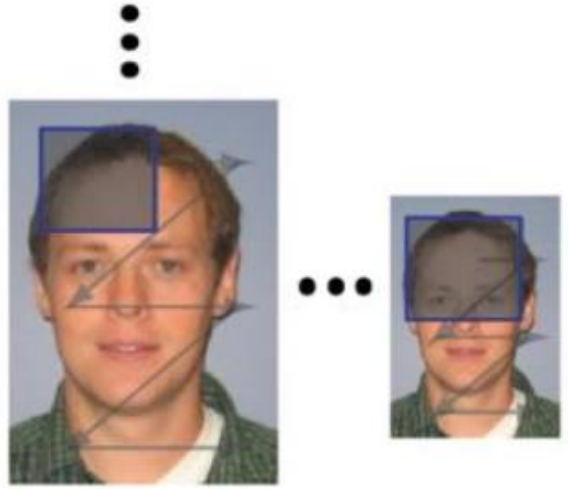
- Problem statement
- **Classical approaches**
- Deep Learning approaches

Localization: sliding window

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



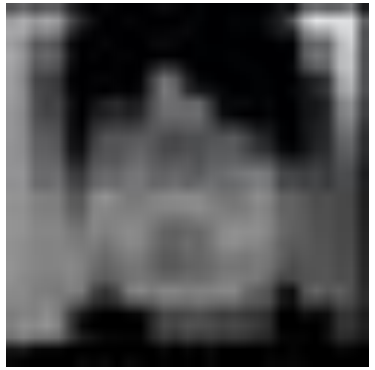
To detect objects of different sizes, a *pyramid of images* is used.



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 $\in \{0,1\}$.



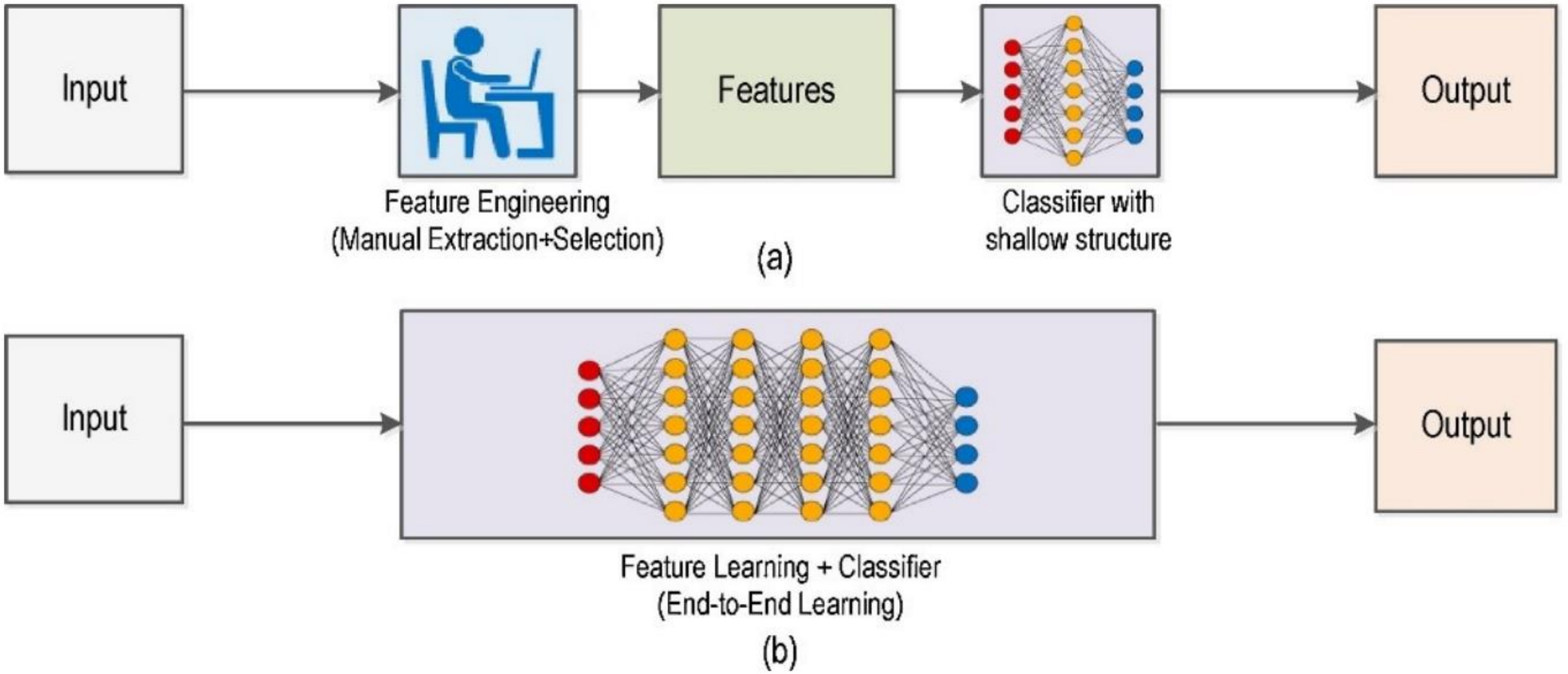
16x16 pixels

x_i = (128, 128, 60, …, 0, 0), features – values of pixels.

y_i = 1.

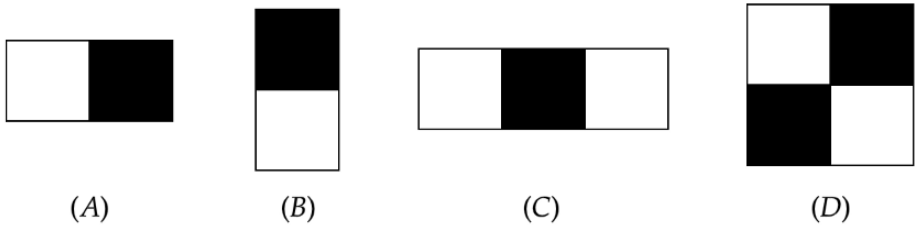
Feature-vector has the same size for any object

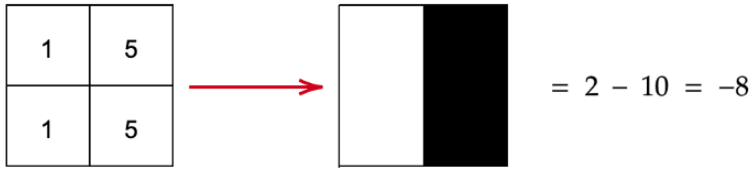
Generating features



Images credit: Niall O' Mahony et al. "Deep Learning vs. Traditional Computer Vision"

Viola Jones Face Detector: CVPR 2001





	x_0	x_1	x_2	x_3
y_0	1	12	45	10
y_1	6	5	11	4
y_2	3	7	10	8
y_3	5	9	4	7

Original Image (Grayscale)

$$ii(x, y) = \sum_{x' \leqslant x,\, y' \leqslant y} i(x', y')$$

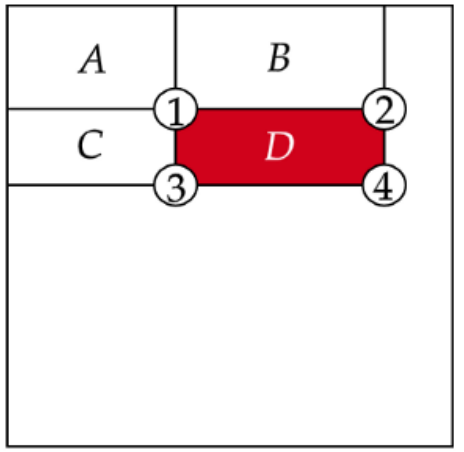
i – original image

ii – integral image

	x_0	x_1	x_2	x_3
y_0	1	13	58	68
y_1	7	24	80	94
y_2	10	34	100	122
y_3	15	48	118	147

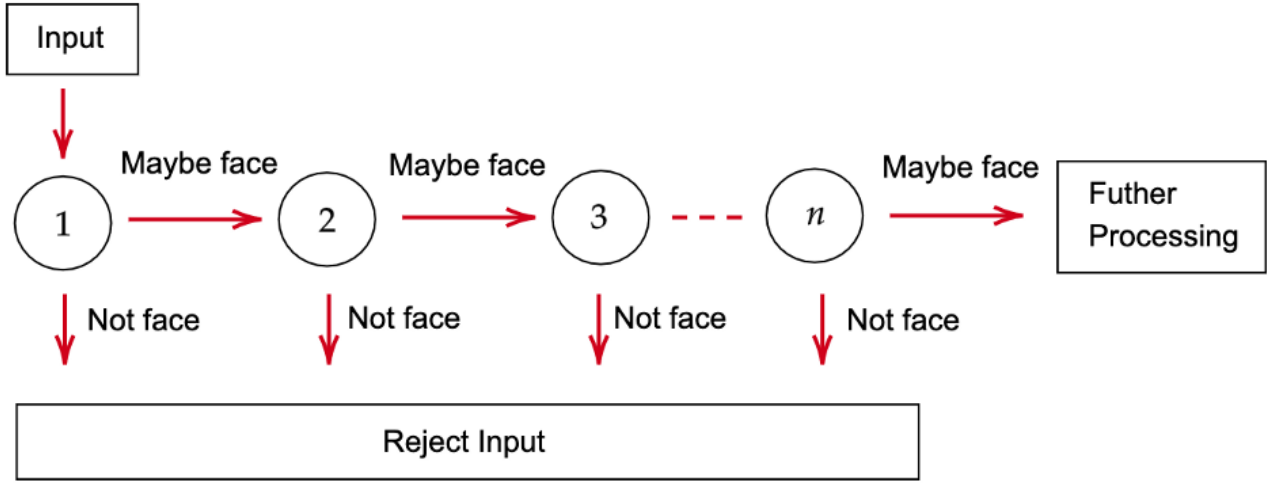
Integral Image

- Classifier input resolution is 24x24
- Over 170000 Haar-like features extracted
- Preparing integral images optimizes calculations



$$D = ④ - ② - ③ + ①$$
$$= ④ + ① - (② + ③)$$

Images credit: https://towardsdatascience.com/understanding-face-detection-with-the-viola-jones-object-detection-framework-c55cc2a9da14



Minimizing:

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

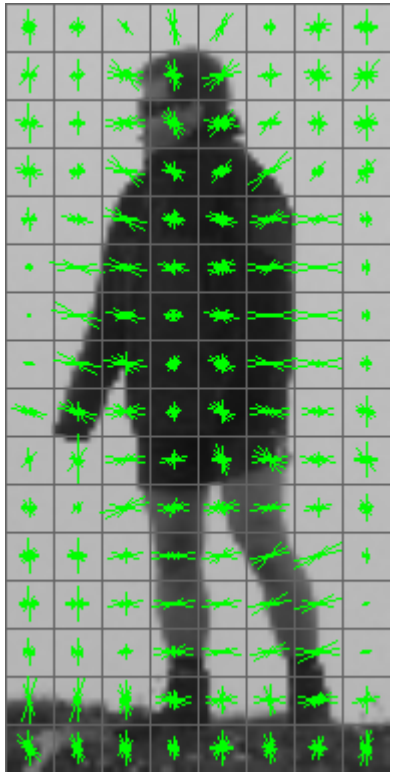
- AdaBoost algorithm used to produce strong classifiers (around 6000 features left)
- Cascade Classifier was constructed to process many negative examples . Original implementation has 38 stages
- Detector can process 15 images (384x288) per second running on a 700Mhz Pentium III CPU

Histogram of Oriented Gradients (HOG) - 2005

1. Filter images with kernels:

-1	0	1

-1
0
1



2. Calculate the magnitude and orientation of the gradient:

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

$$\theta = \arctan \frac{g_y}{g_x}$$

3. Divide the image into 8×8 cells

4. Calculate 9-bin histogram of gradients in these cells

5. Normalize histograms for each block

6. Concatenate histograms – feature vector is ready!

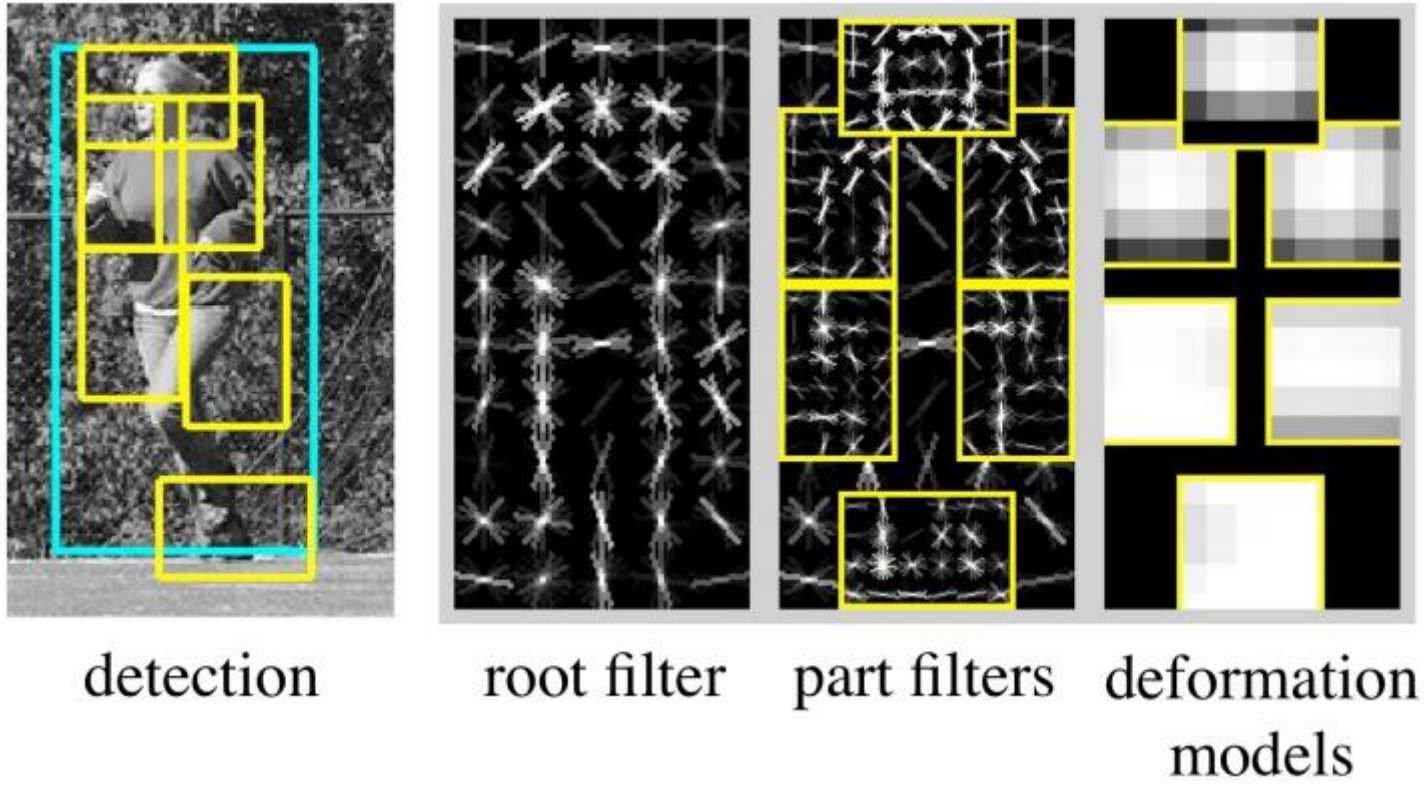
7. Train an SVM

Attempts to make features invariant to translation, scale, illumination

Images credit: https://learnopencv.com/image-recognition-and-object-detection-part1/

Deformable Part-based Models (2008)

- the peak of the traditional (pre-DL) object detection methods. Winner of VOC-09 detection challenge.
- star-structured part-based model defined by a "root" filter + a set of parts filters + associated deformation models
- sliding window + feature pyramid
- representation of the class of models by a mixture of star models (different viewpoints, occlusions etc.)



- Learn model structure, filters and deformation cost.
- R. Girshick further formulated important techniques such as “hard negative mining” and “bounding box regression”.

Images credit: http://www.cse.psu.edu/~rtc12/CSE586/lectures/dpmlecture.pdf

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”.

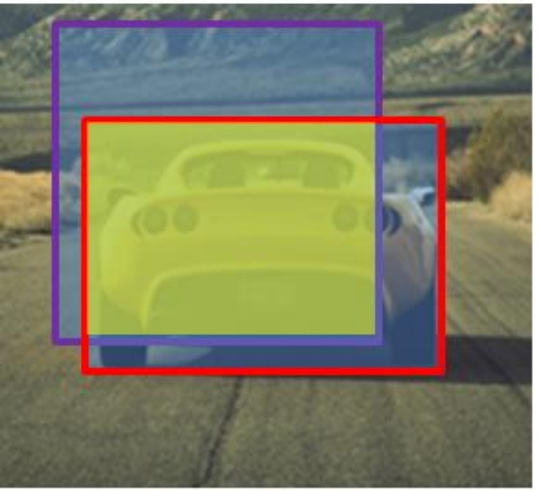
ICF: Integral Channel Features (2009) - P. Dollár et al. “Integral Channel Features”.

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

Measure of localization accuracy: IoU

Ground-truth bounding boxes are hand labeled

Predictions are generated by the model



Intersection over union (IoU)

$$= \frac{\text{size of } \square}{\text{size of } \square}$$

"Correct" if IoU $\geq 0{,}5$

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Thresholding IoU gives us a measure of TP, FP and FN.

Images credit: https://github.com/Cartucho/mAP

Precision and recall

TP = True positive
TN = True negative
FP = False positive
FN = False negative

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

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

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

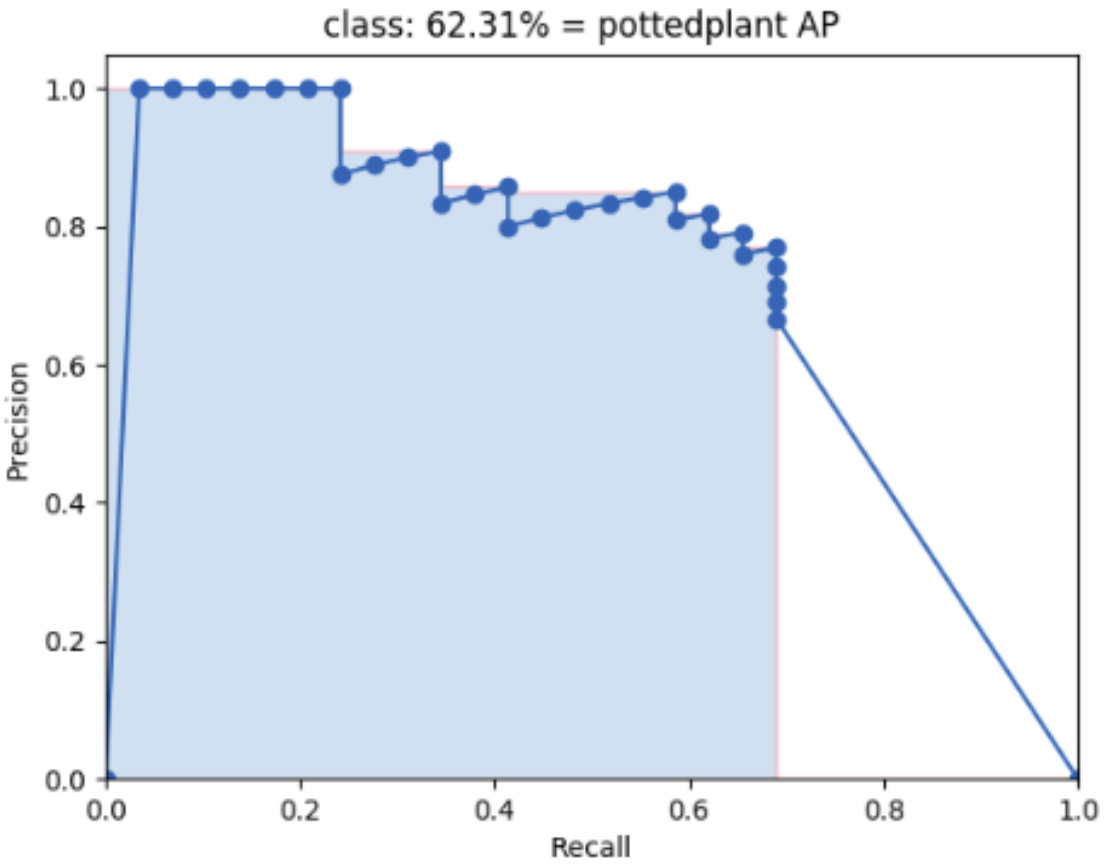
True positive only counted if class prediction matches label

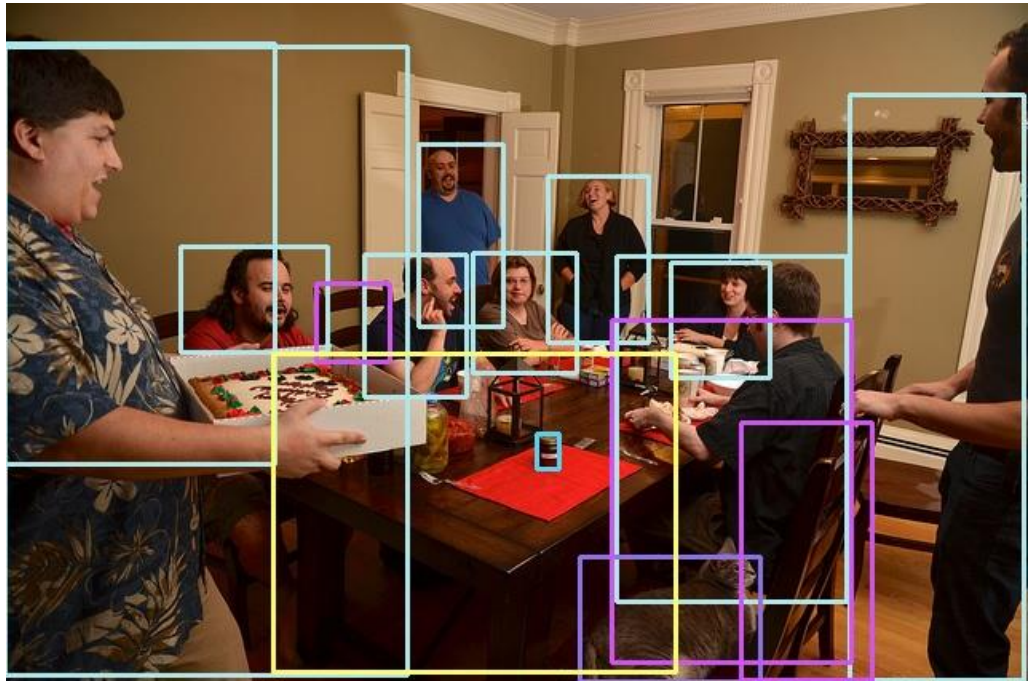
		Predicted Label: Positive	Predicted Label: Negative
Actual Label	Positive	TRUE POSITIVE TP	FALSE NEGATIVE FN
	Negative	FALSE POSITIVE FP	TRUE NEGATIVE TN

Metric to measure localization + classification

Average precision for each class

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





$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$AP_k =$ *the AP of class k*

$n =$ *the number of classes*

Images credit: https://github.com/Cartucho/mAP

Popular object detection datasets

Dataset	train		validation		trainval		test		classes
	images	objects	images	objects	images	objects	images	objects	
VOC-2007	2,501	6,301	2,510	6,307	5,011	12,608	4,952	14,976	20
VOC-2012	5,717	13,609	5,823	13,841	11,540	27,450	10,991	-	
ILSVRC-2014	456,567	478,807	20,121	55,502	476,688	534,309	40,152	-	
ILSVRC-2017	456,567	478,807	20,121	55,502	476,688	534,309	65,500	-	
MS-COCO-2015	82,783	604,907	40,504	291,875	123,287	896,782	81,434	-	80
MS-COCO-2018	118,287	860,001	5,000	36,781	123,287	896,782	40,670	-	
OID-2018	1,743,042	14,610,229	41,620	204,621	1,784,662	14,814,850	125,436	625,282	600

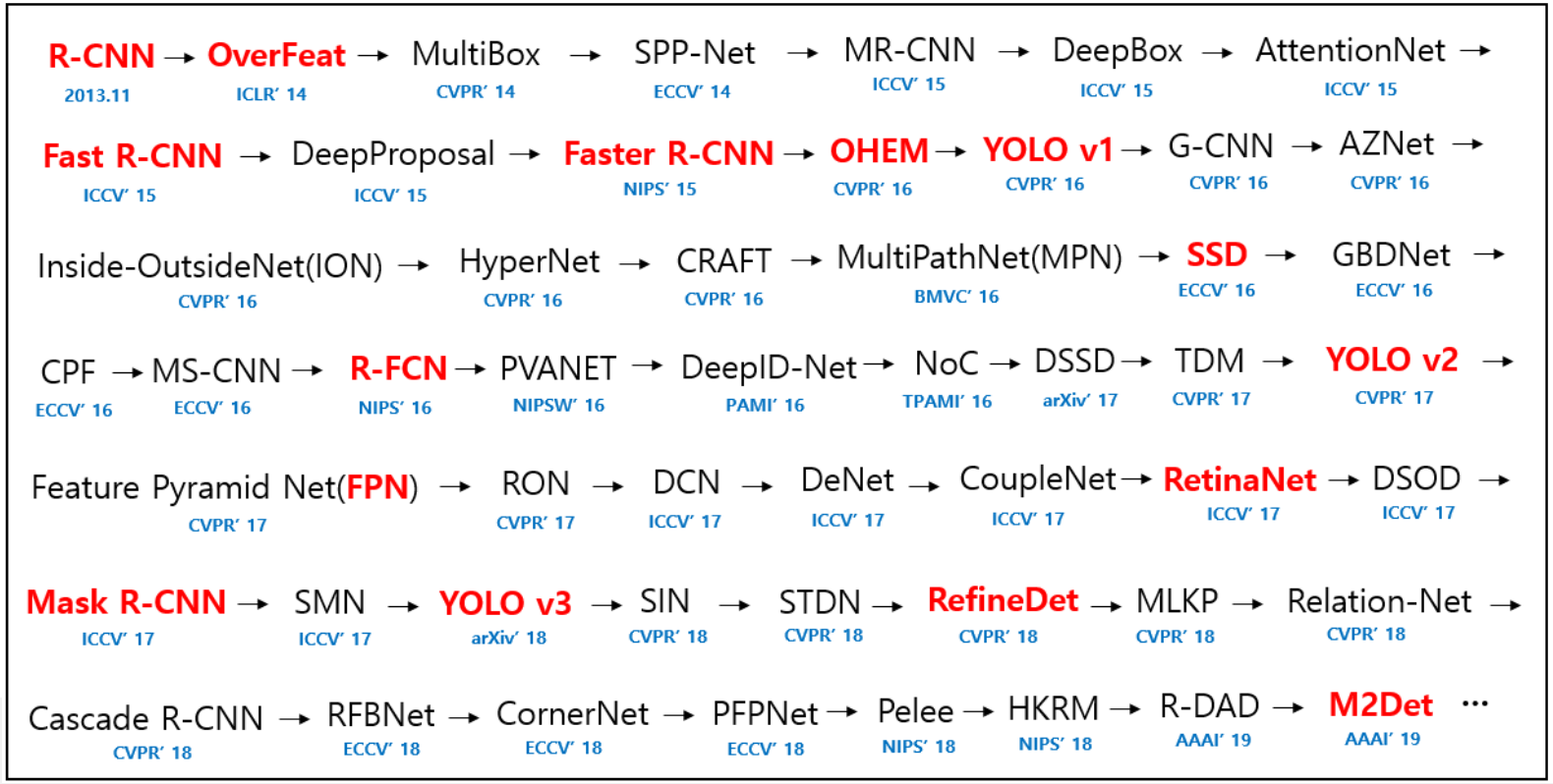
+ numerous specialized datasets

Table credit: Zhengxia Zou et al. Object Detection in 20 Years: A Survey

Agenda

- Problem statement
- Classical approaches
- Deep Learning approaches

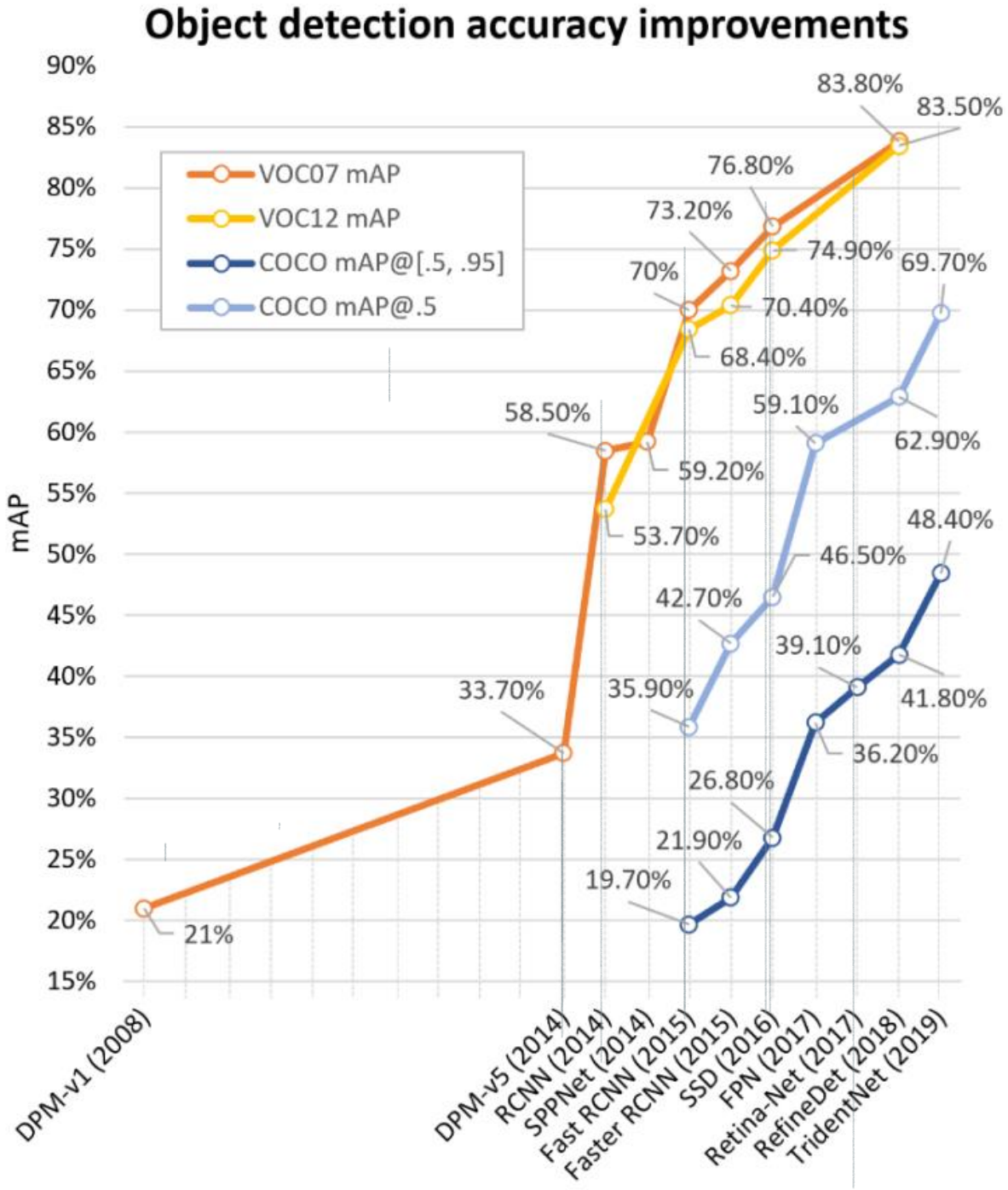
Table credit: https://github.com/hoya012/deep_learning_object_detection



Revolution in object detection

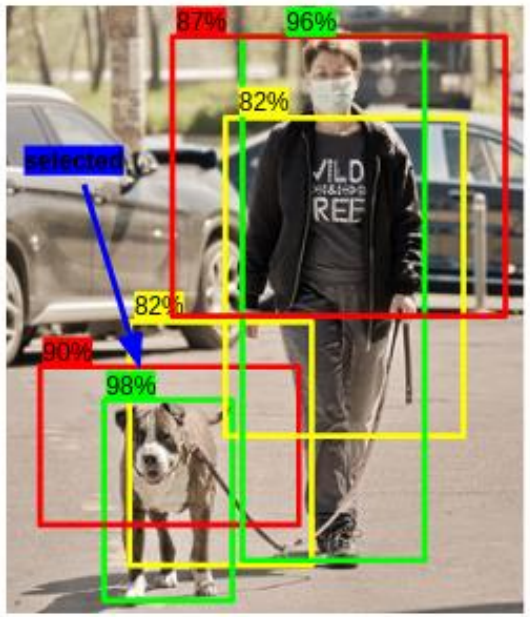
Handcrafting features is hard, yet accuracy stays low

Could we somehow find good features automatically?

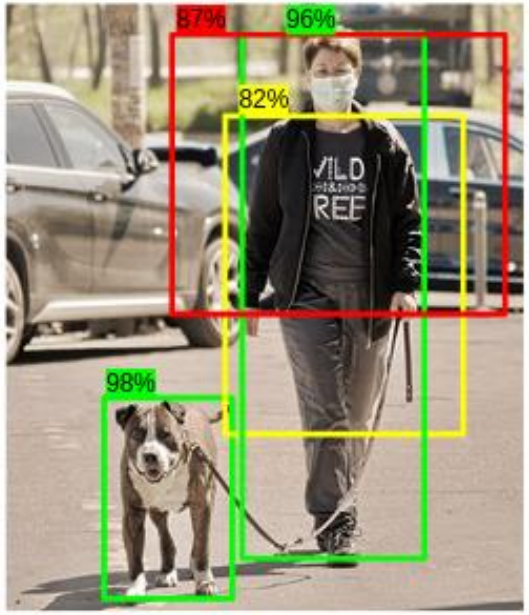


Non-Maximum Suppression (NMS)

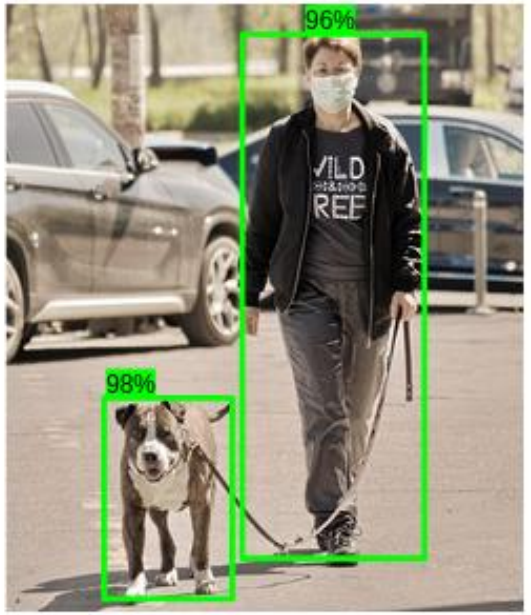
- Detector outputs multiple detections for the same object
- NMS discards all except one with the best features, e.g. with the highest score
- Can be learnable



Step 1: Selecting Bounding box with highest score

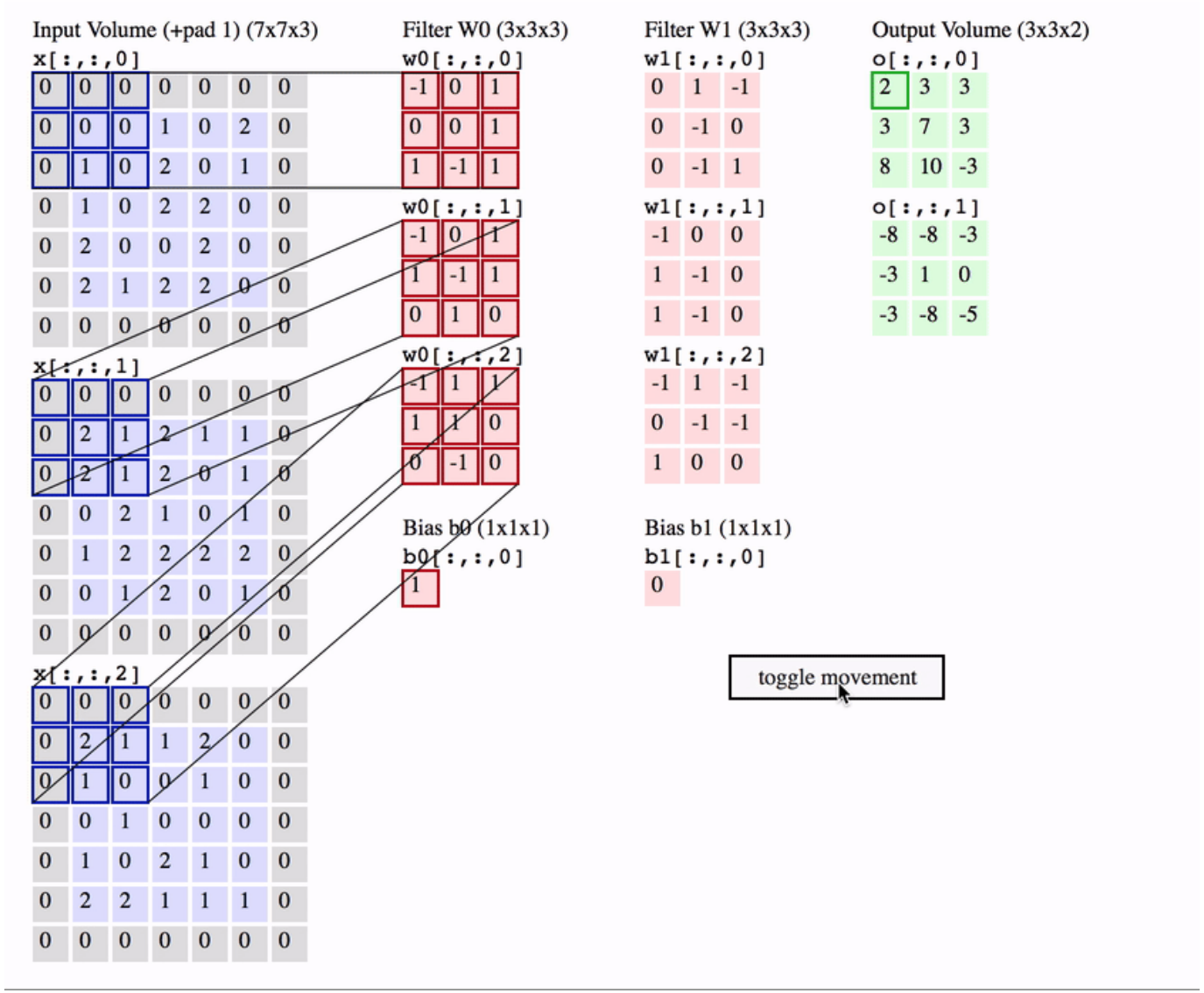


Step 3: Delete Bounding box with high overlap

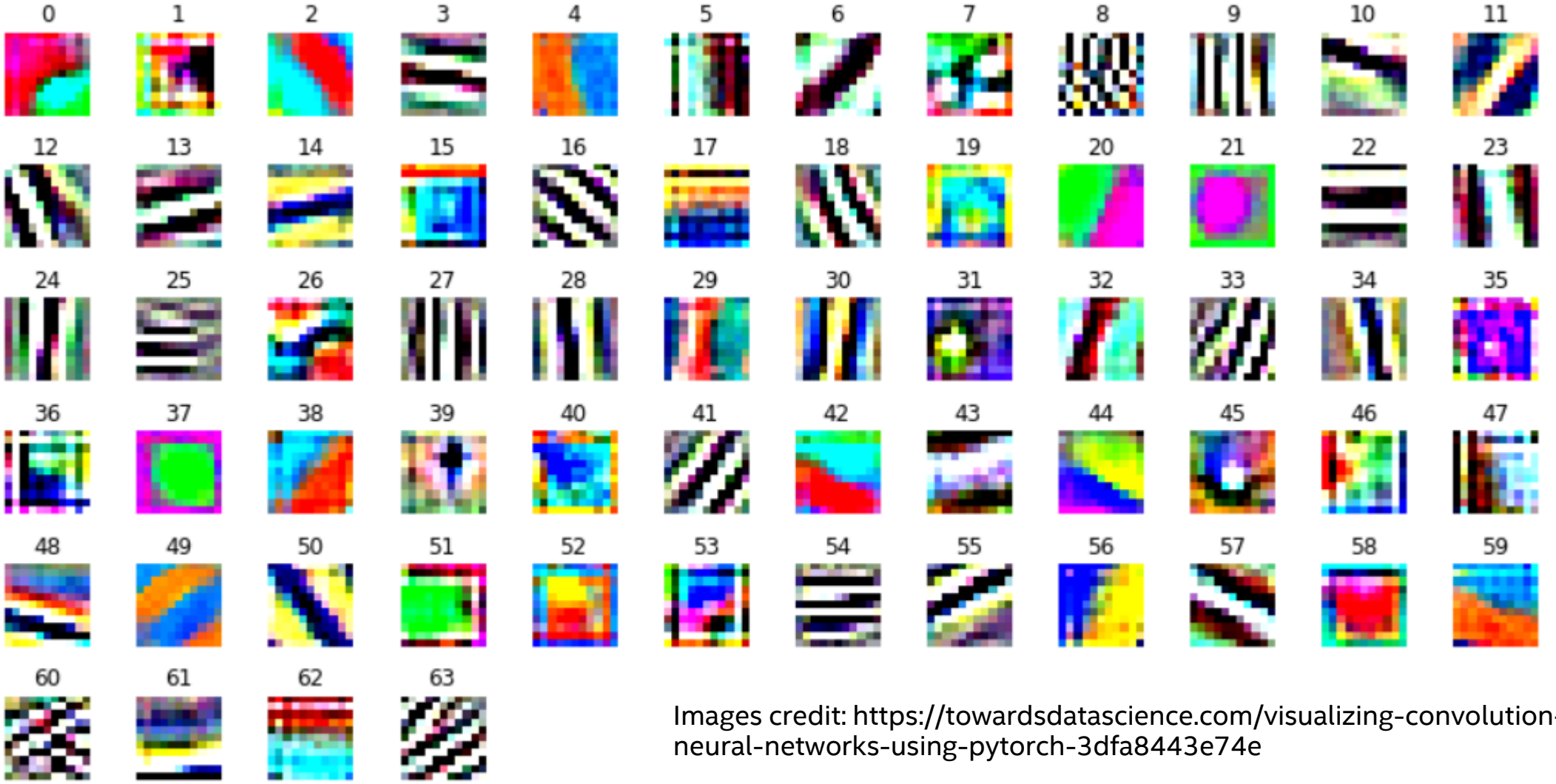


Step 5: Final Output

Convolution

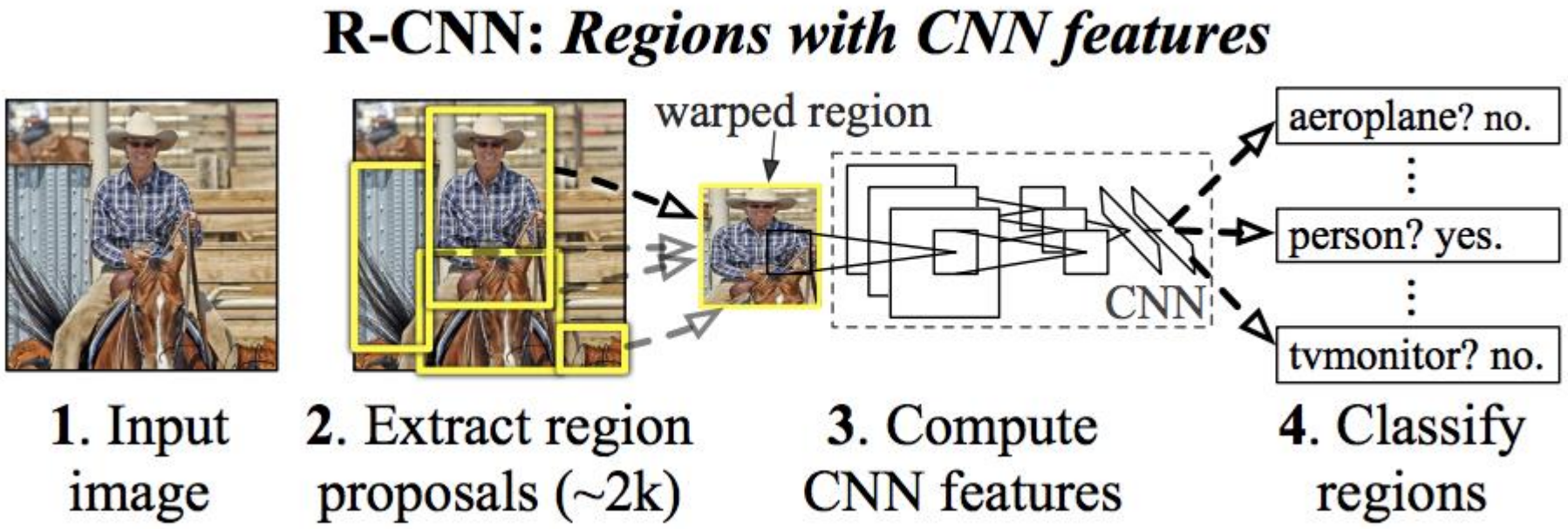


Filters from first convolution layer in AlexNet



Images credit: https://towardsdatascience.com/visualizing-convolution-neural-networks-using-pytorch-3dfa8443e74e

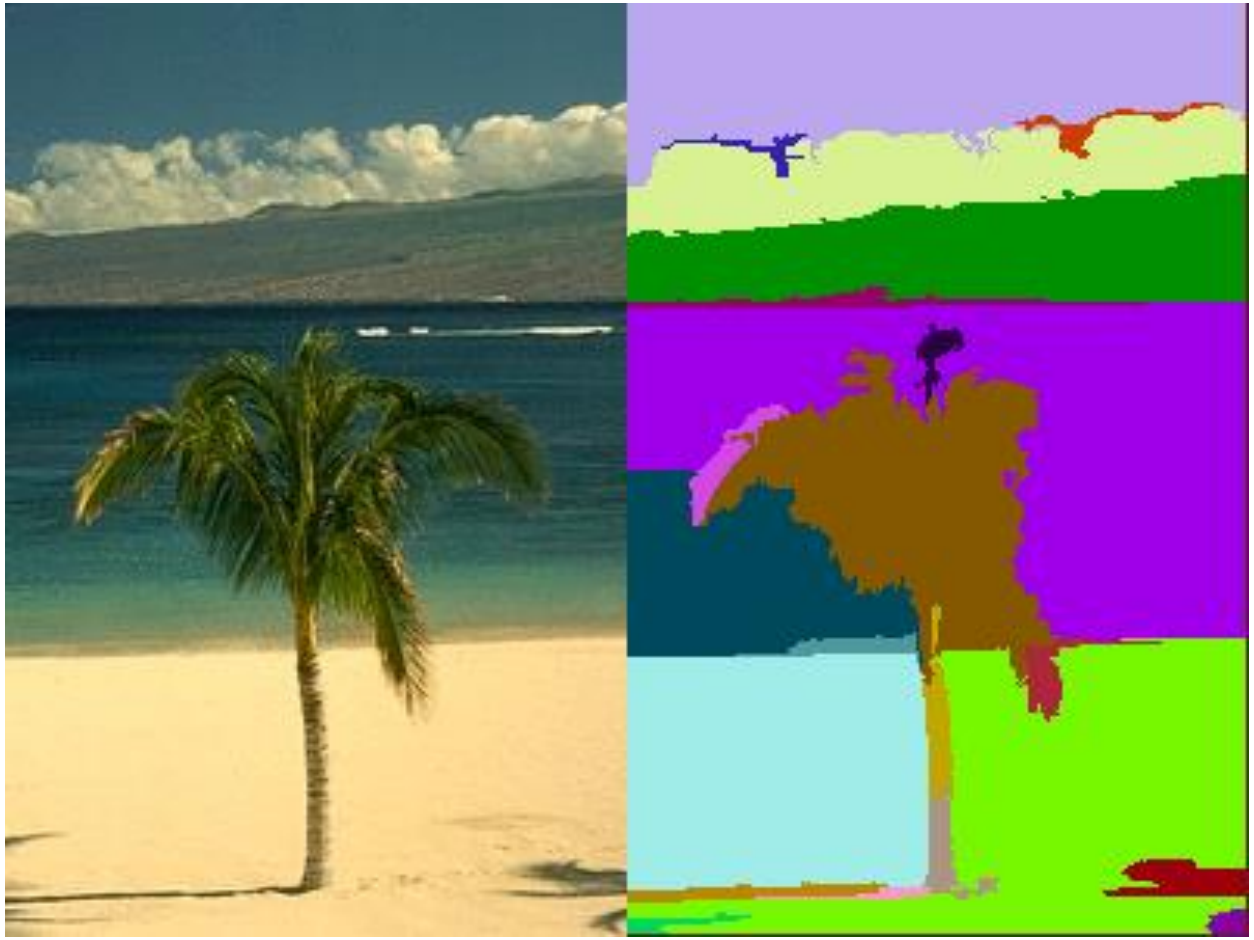
R(egion based)-CNN (CVPR, 2014)



- 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

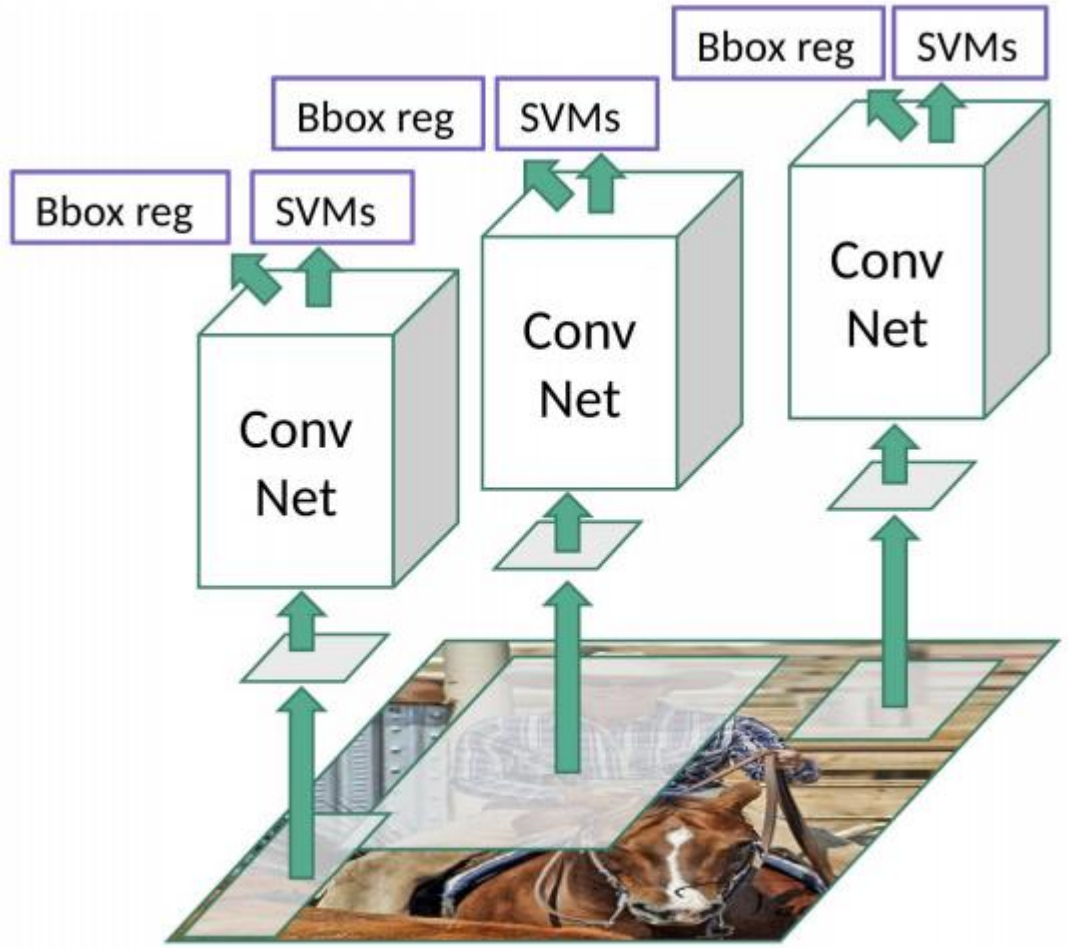
Selective Search for Object Detection

- Color similarity
- Texture similarity
- Size similarity
- Shape similarity
- A final meta-similarity, which is a linear combination of the above similarity measures



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

- Region proposal algorithm - Selective Search is used to generate Regions of Interest (RoI)
- 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



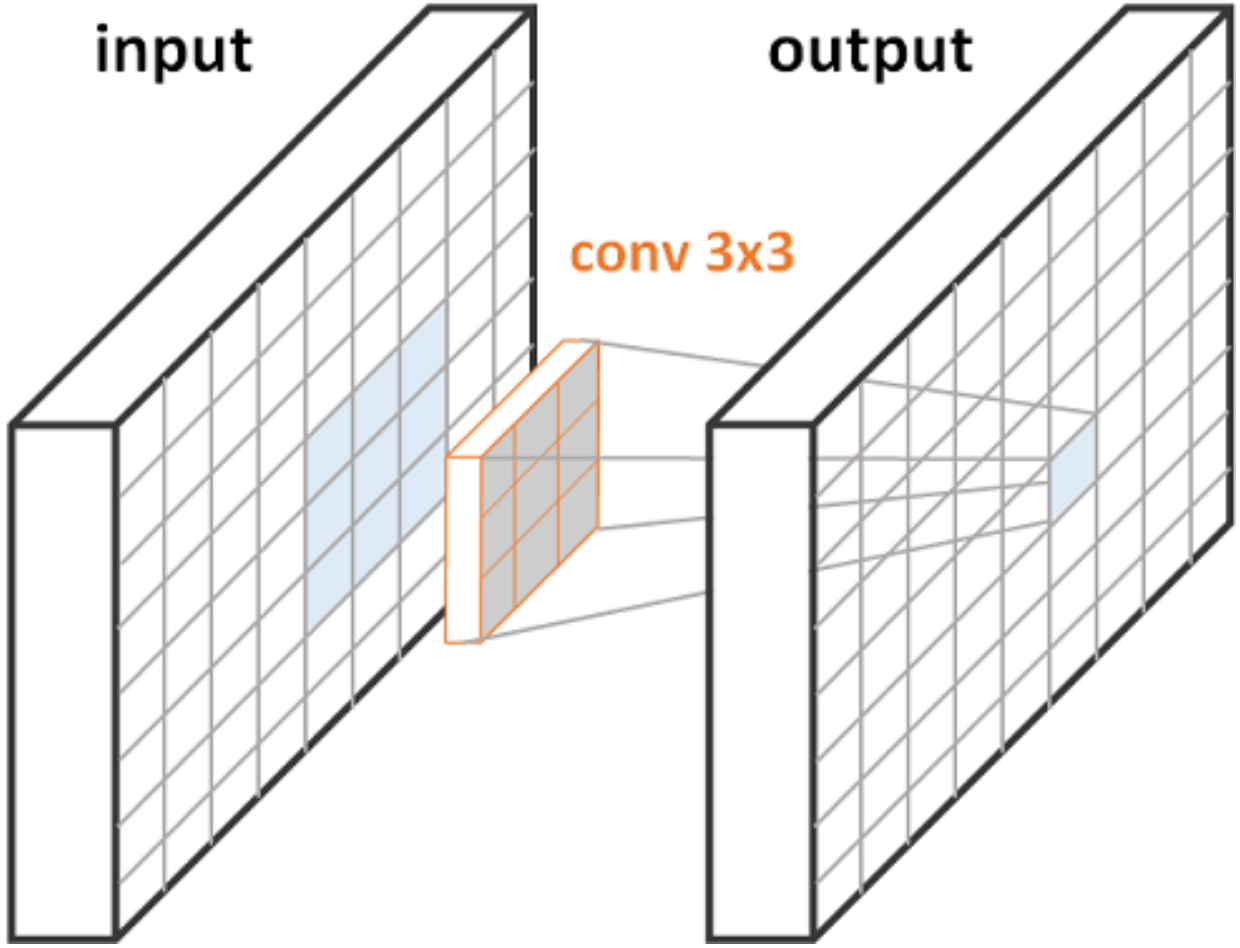
R(egion based)-CNN (CVPR, 2014)

- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.
- SVM is trained separately

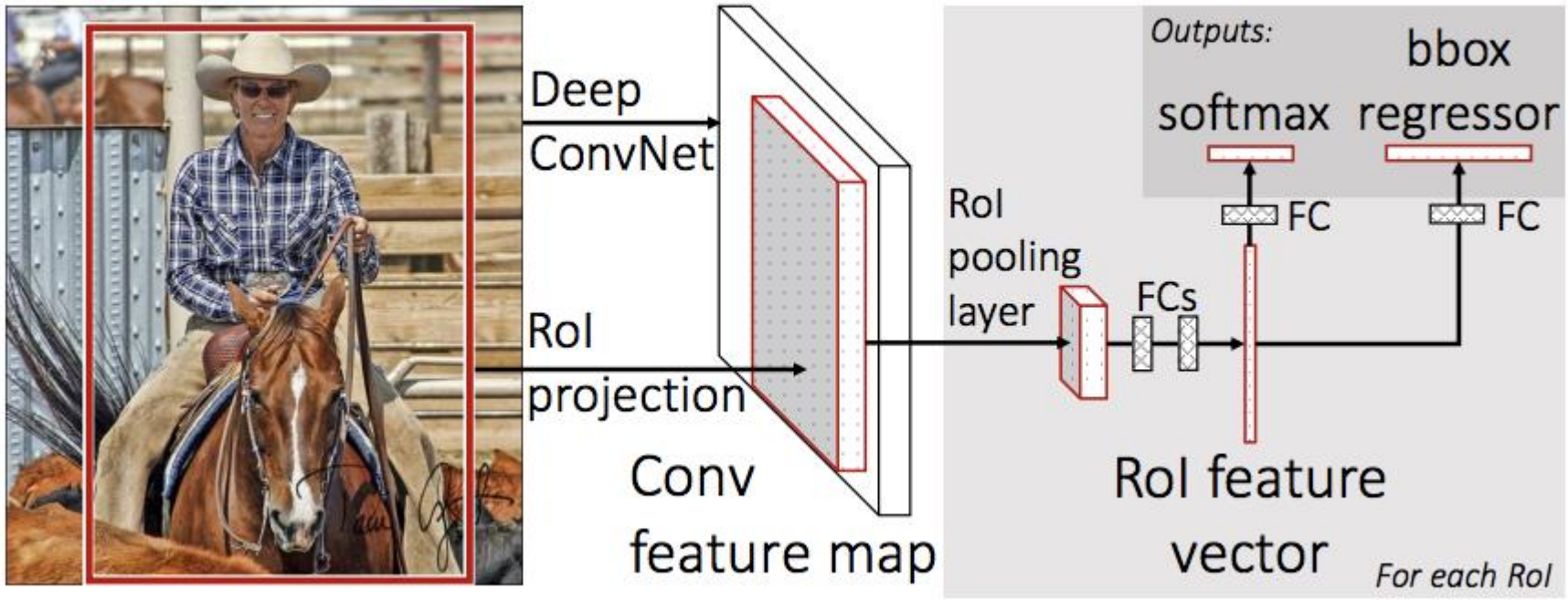
Fast R-CNN (2015)

- 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 Networks and train it end-to-end (removing SVM)
  - Apply feature extractor to the whole image and crop RoIs on high-level feature map to make detection faster

Mapping an image to a feature map with convolution



Fast R-CNN (2015)



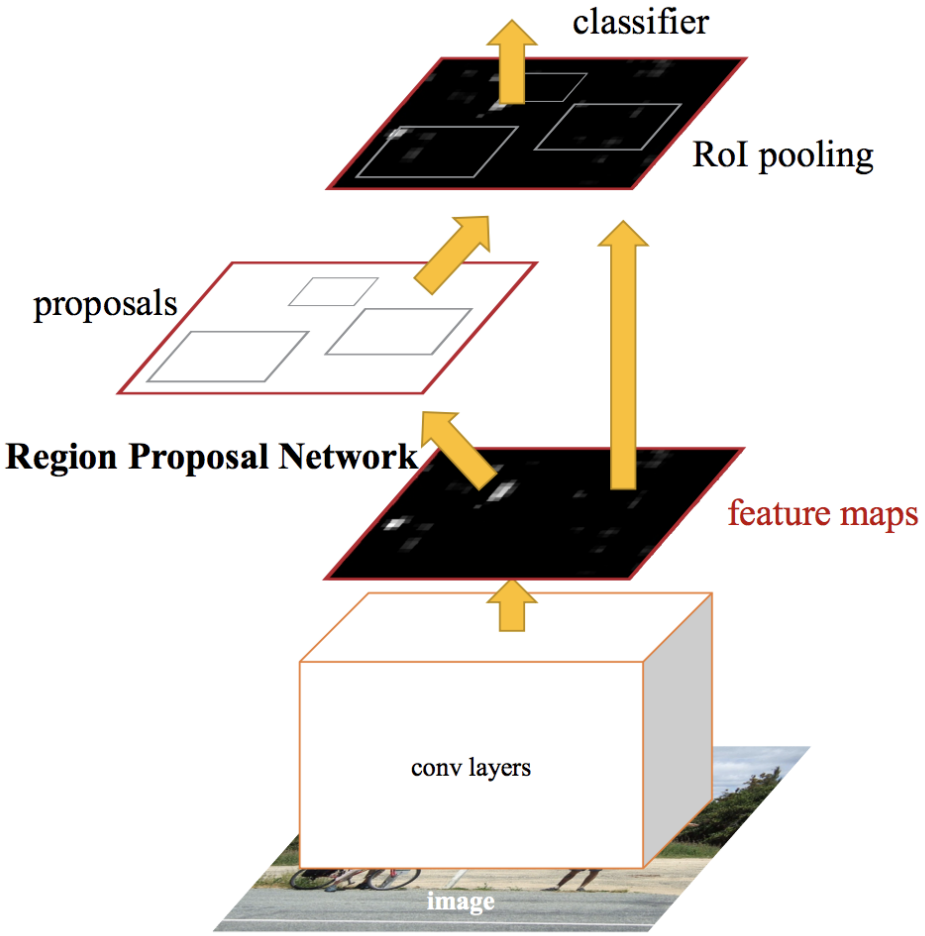
Deep features are calculated once per image, not per proposal.

RoI pooling outputs vectors of the same length for any proposal.

Faster R-CNN (2015)

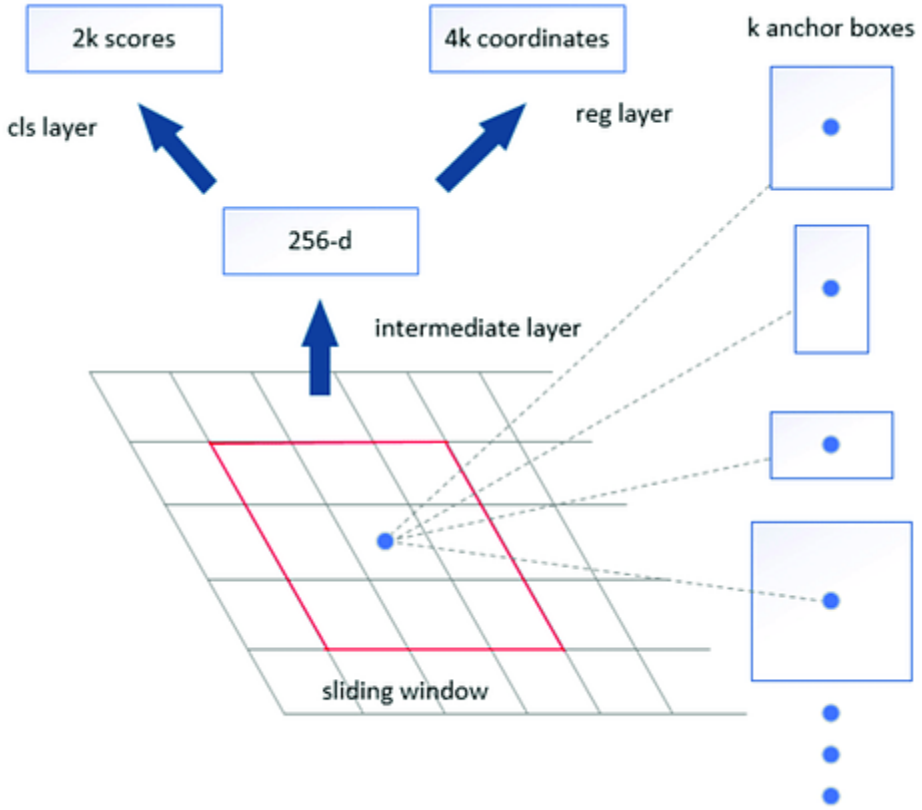
- Drawbacks of Fast R-CNN:
  - 25x faster then R-CNN, but still too slow. Mostly because of the Selective Search now (~2s per VGA image)
- Solution – Faster R-CNN:
  - Merge region proposal stage into the net as well
- Result:
  - The first end-to-end, and the first near-realtime deep learning detector

Faster R-CNN (2015)



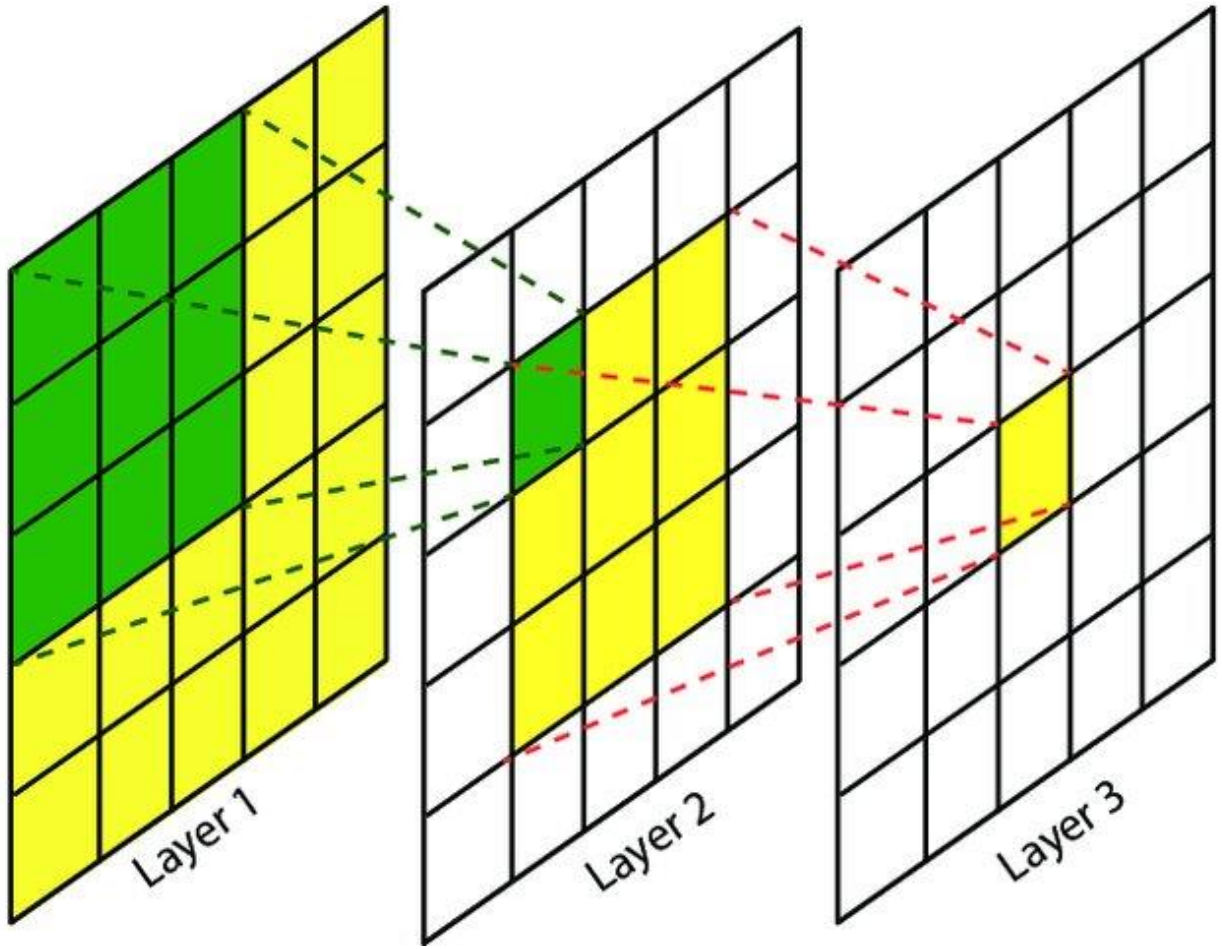
Faster R-CNN pipeline

Objectness threshold and NMS are applied to proposals



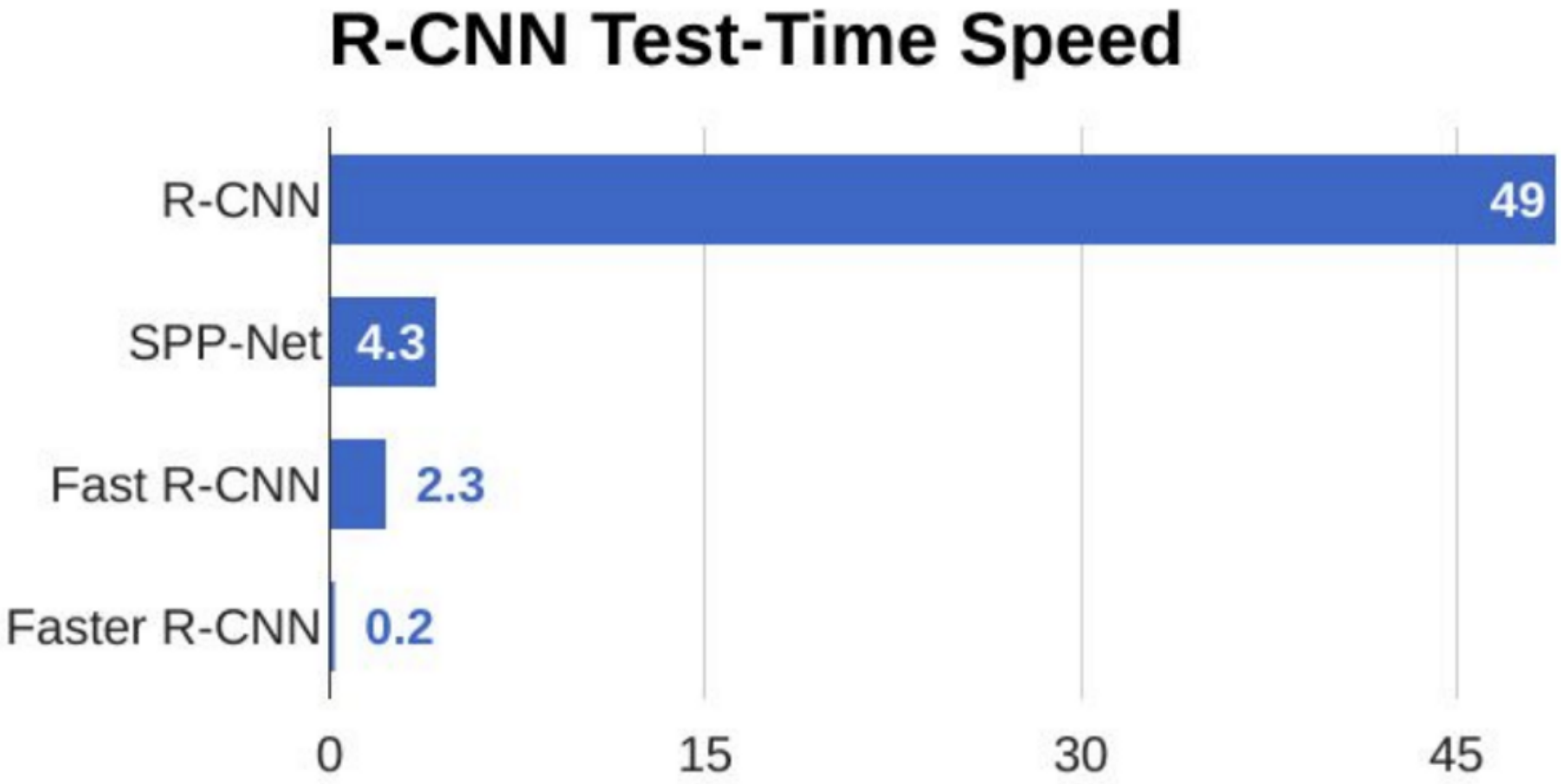
Region Proposals Network

Receptive field



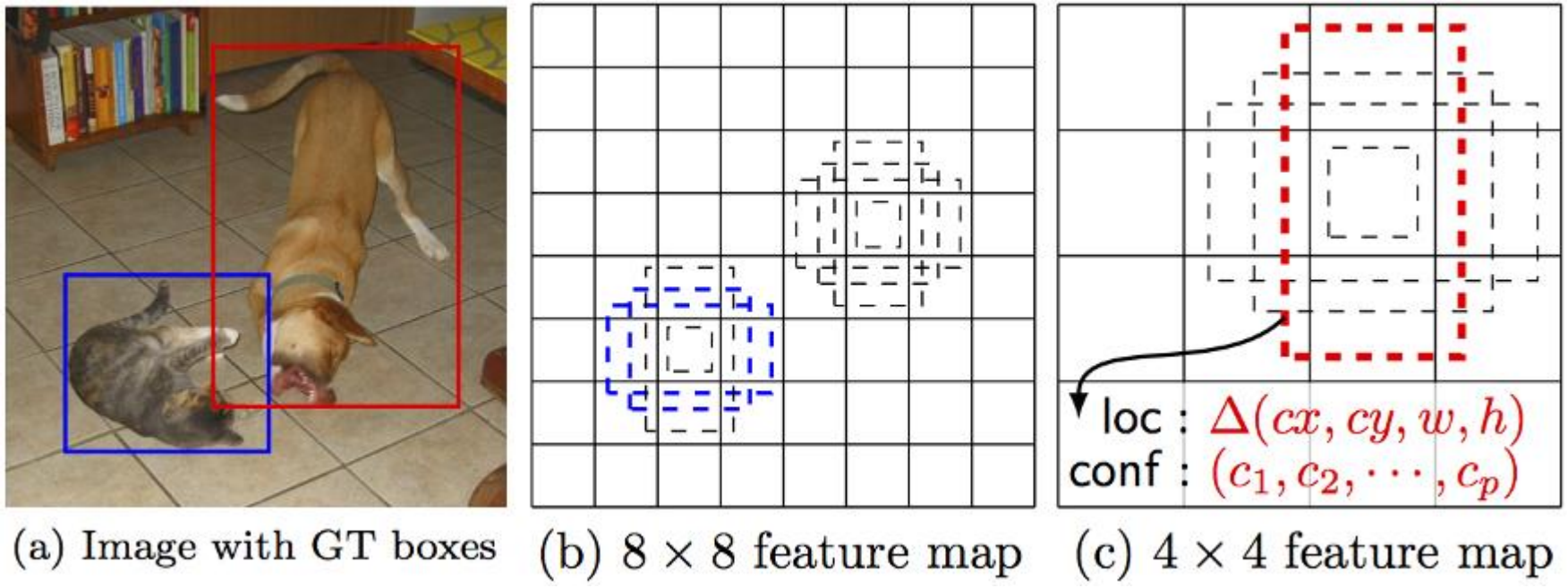
*The **receptive field** is defined as the region in the input space that a particular CNN's feature is affected by.*

Inference time comparison



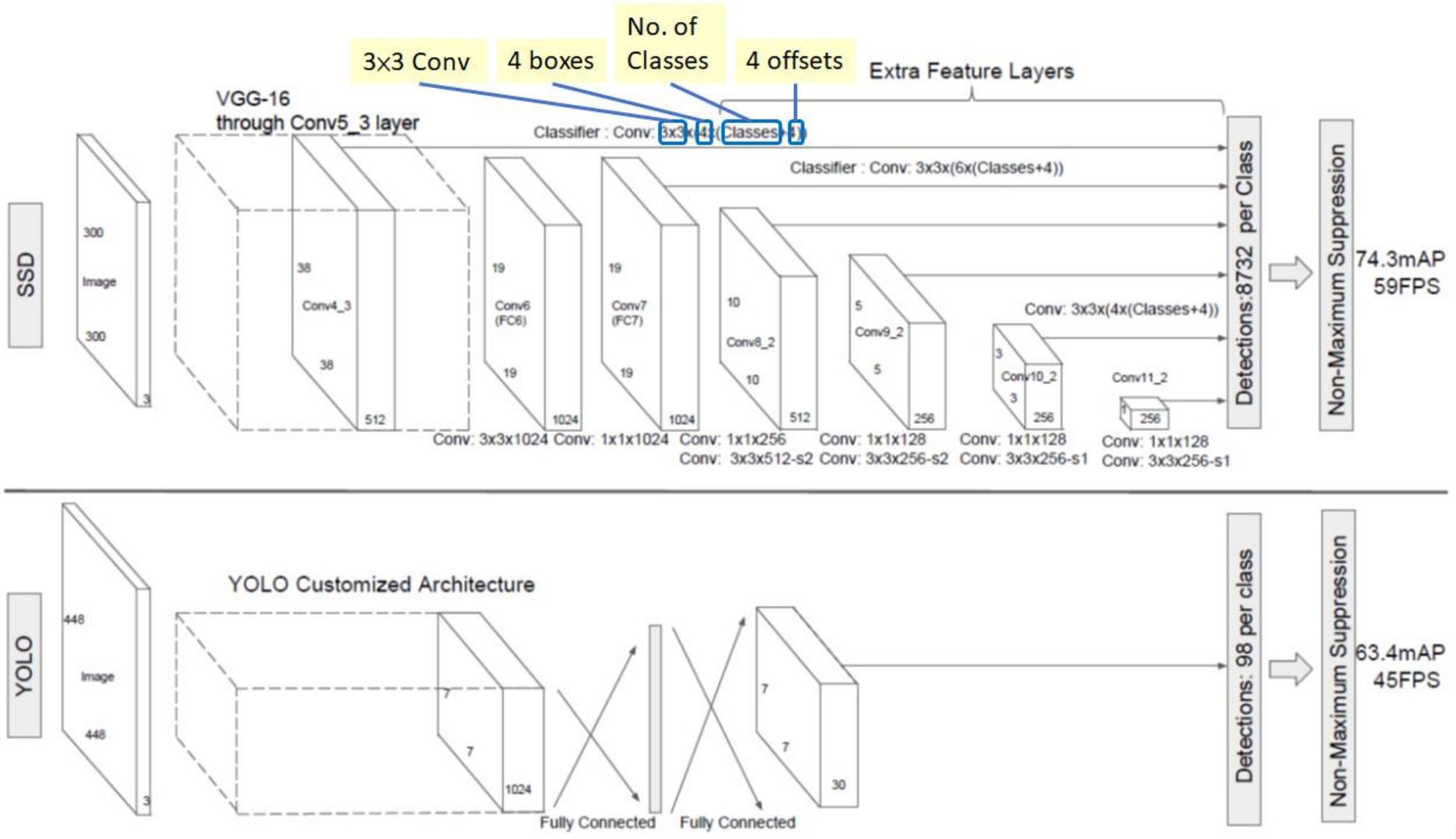
Single Shot Multibox Detector

- Motivation:
  - Region proposal net can be eliminated at all – introduce anchor boxes
  - Make multi-scale detection efficient



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Single Shot Multibox Detector

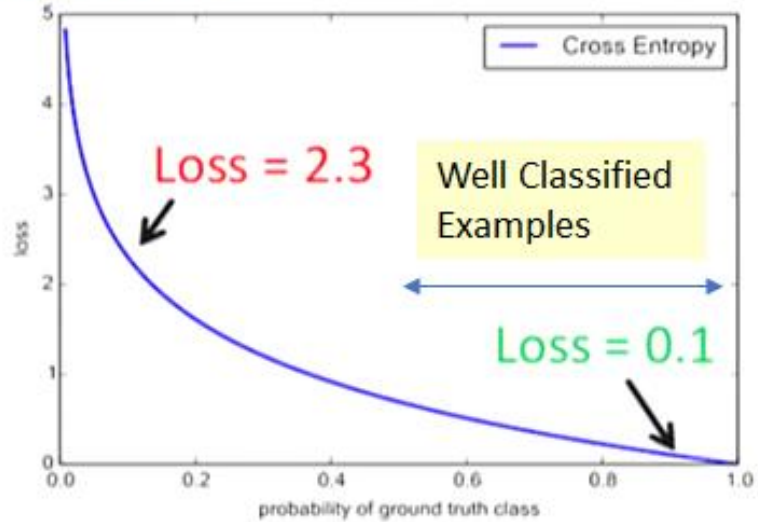


RetinaNet

- 100000 easy : 100 hard examples
- **40x bigger loss** from easy examples

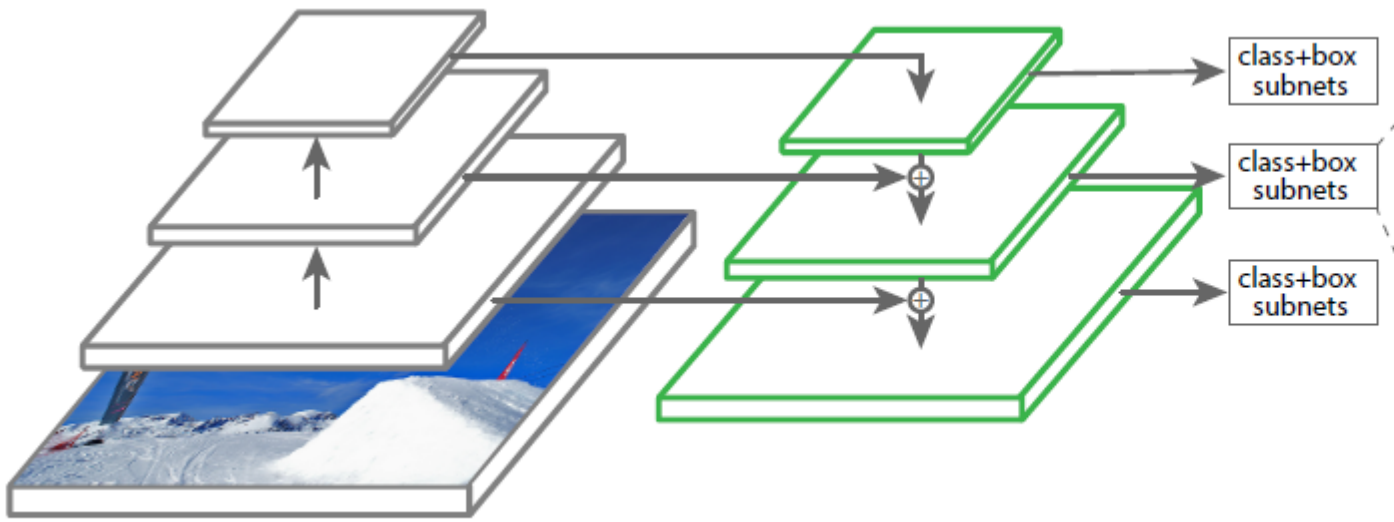
$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$

$$\mathrm{CE}(p, y) = \mathrm{CE}(p_t) = -\log(p_t).$$

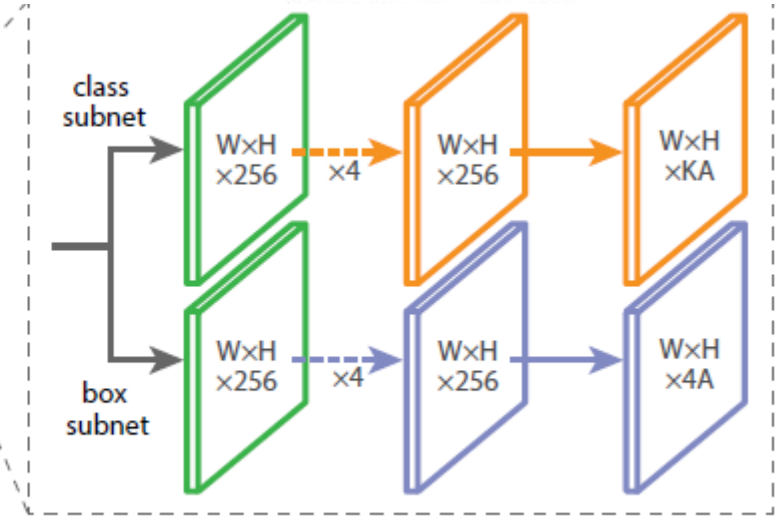


Suppressing easy samples with Focal Loss

$$\mathrm{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t).$$

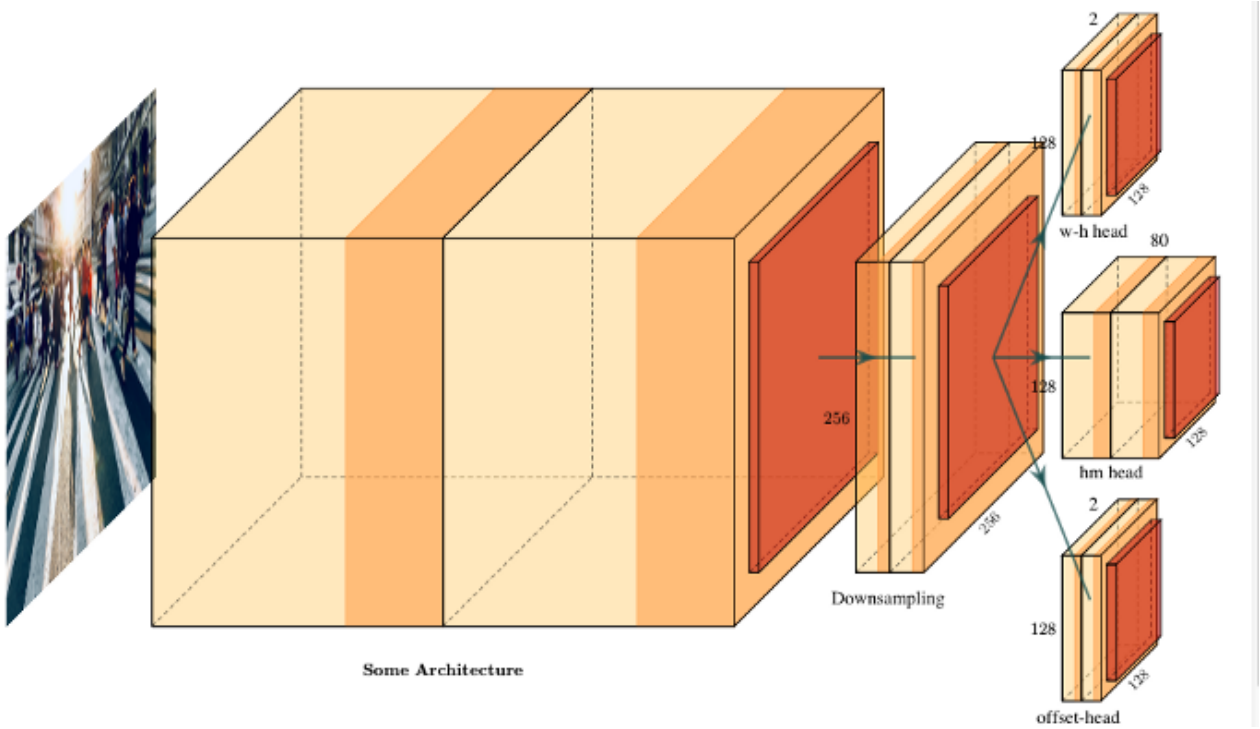


(a) ResNet (b) feature pyramid net



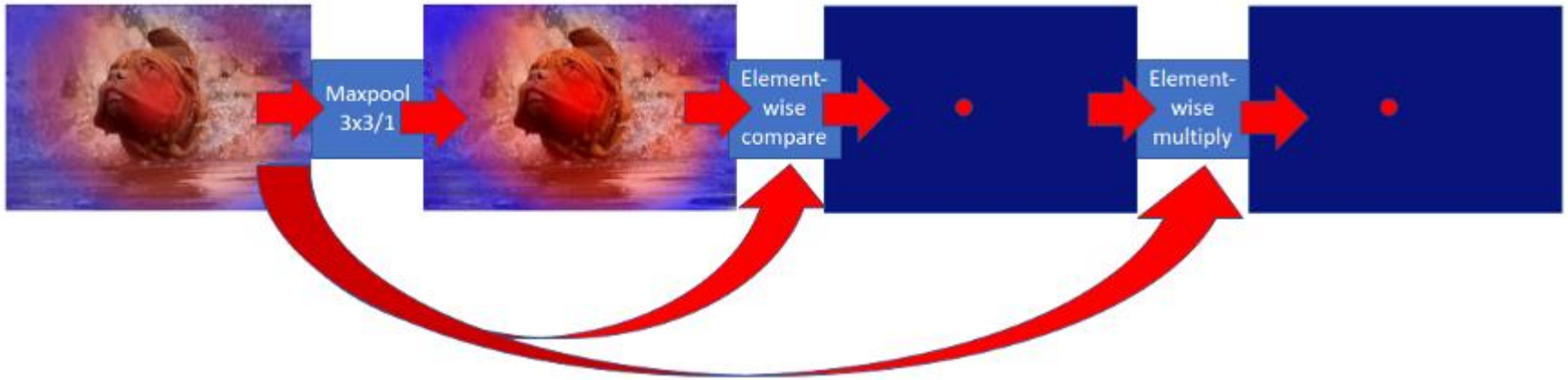
(c) class subnet (top) (d) box subnet (bottom)

CenterNet: Objects as Points

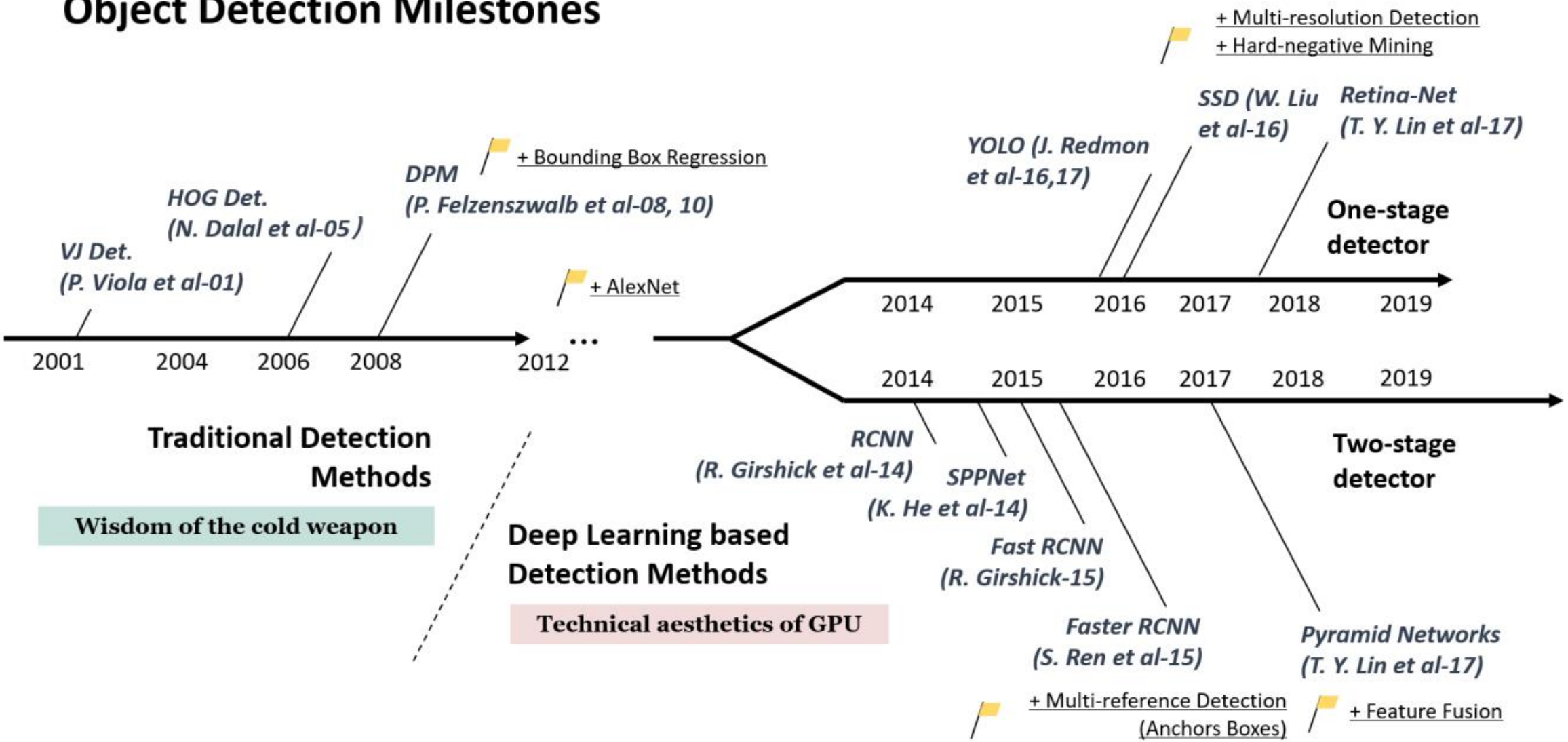


Some Architecture

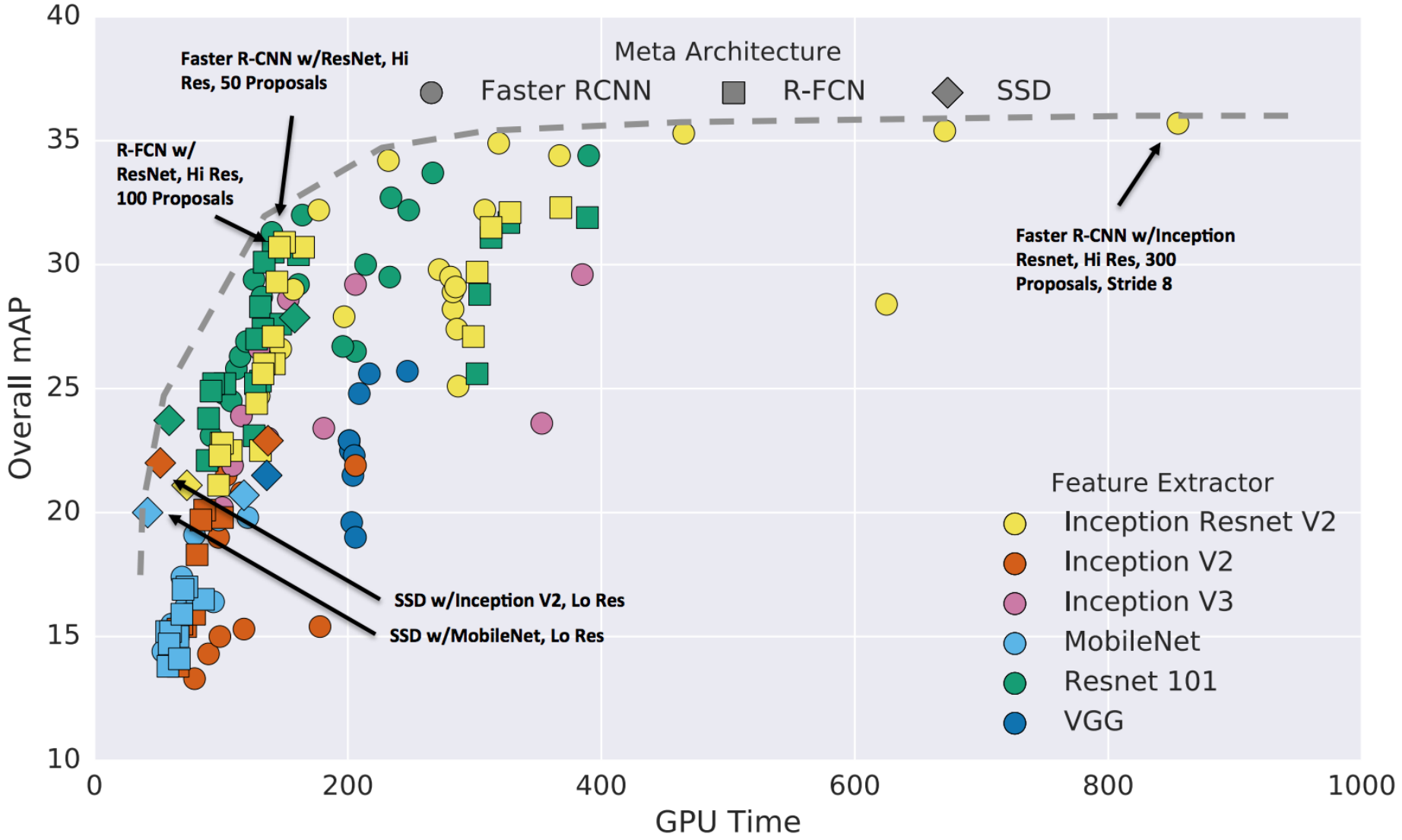
- No anchor boxes
- No NMS needed



Object Detection Milestones



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

Popular OD repositories

- Tensorflow Object Detection API: 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
  - Face detection (2 variants)
  - Person detection (2 variants)
  - Vehicle detection (3 variants)
  - Public models: SSD, Faster-RCNN, R-FCN, etc.

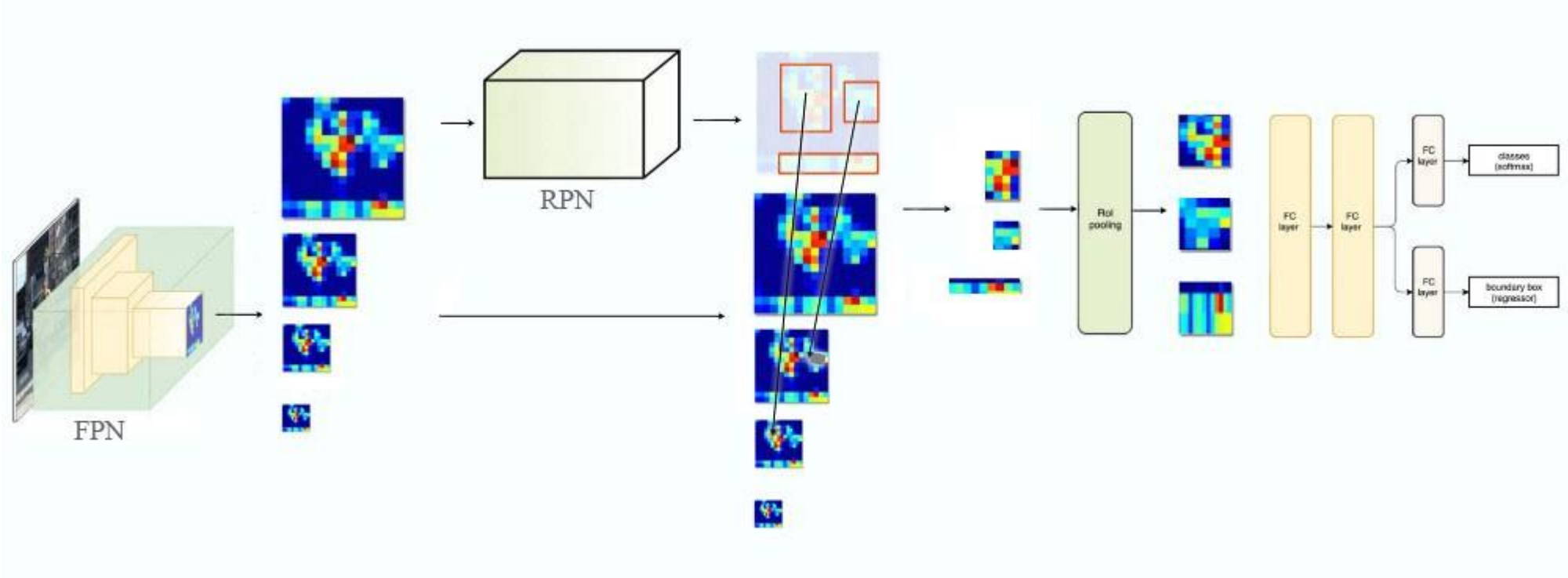
Q&A


intel

EfficientDet

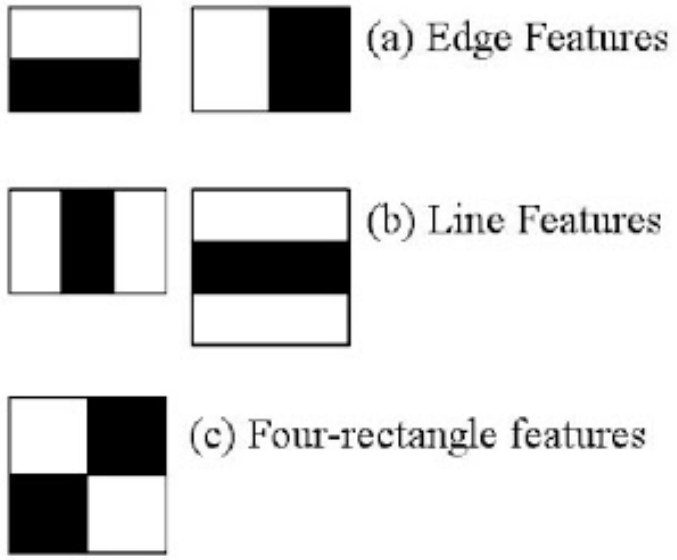
Feature Pyramid Network (FPN)

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



Haar features (2004)

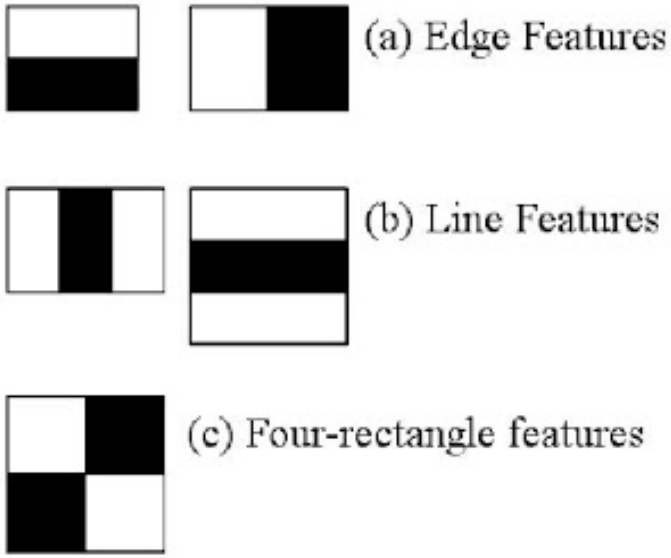
Feature-vector is formed from the values of the convolution with one of the pre-defined kernels. Kernels (black = -1, white = 1):



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



nite = 1):