Object Detection

Single Shot Detectors, Faster R-CNN

Table of Contents

Object Detction: Task

- Find **bounding boxes**, which encompass objects within an image
- Dataset defines the kind/type of object and the corresponding box

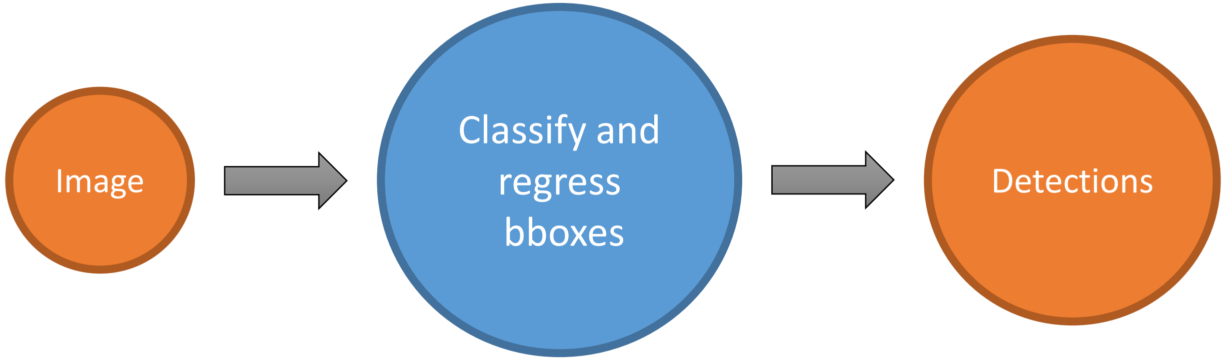
Main approaches in object detection

- One-Stage methods
  - Single Shot Multibox Detector (SSD)
  - You Only Look Once (YOLO)
  - CenterNet
- Two-Stage methods
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN

One-Stage Methods

Single Shot Detector, YOLO

One-Stage Methods



- Classification and localization in one step
- Singe Shot methods are generally very fast
- But they suffer from worse accuracy
- Examples
  - SSD: Single Shot MultiBox-Detektor
  - YOLO: You Only Look Once

Object detection: task

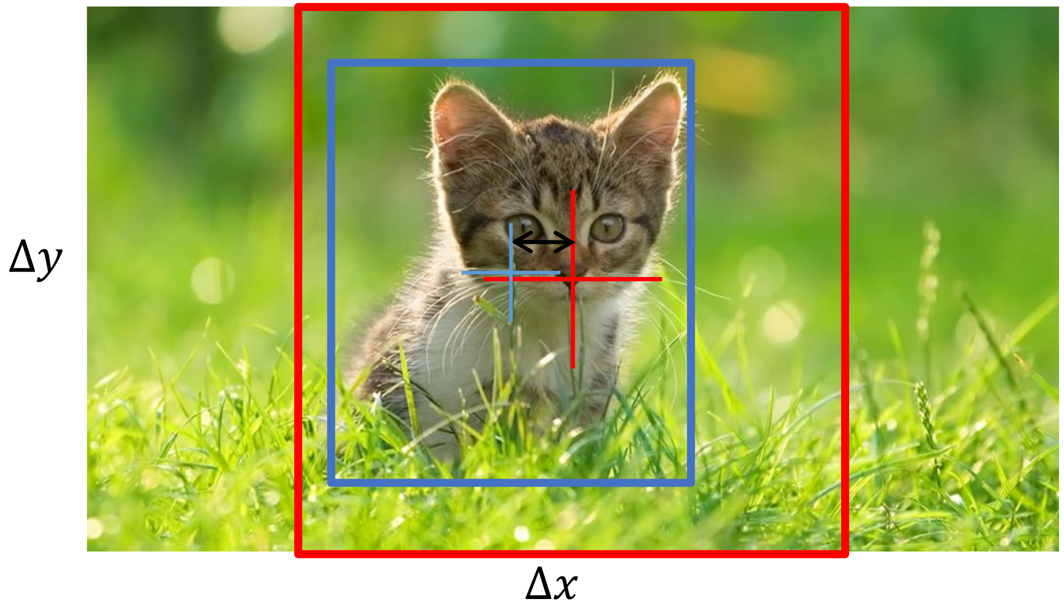
How can a network predict both the class as well as a bounding box?

- Class: Softmax
- Bounding Box? Defined by 4 values (coordinates/height/width)

Example: Only one object

Reference BB

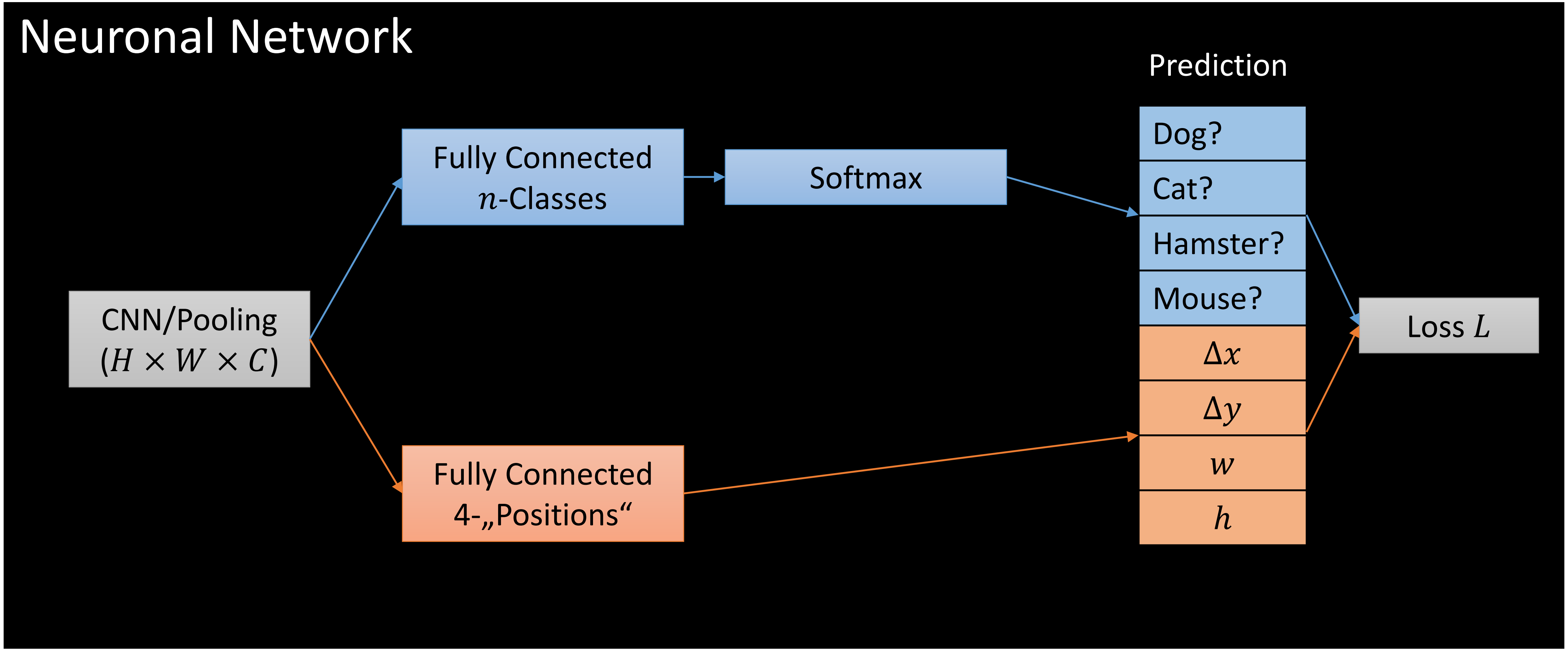
- Strictly defined
- e.g. always in the center, $h_{reference} = w_{reference}$



Class	Ground Truth
Dog?	0
Cat?	100%
Hamster?	0
Mouse?	0
Δx	-0.2
Δy	-0.05
w	0.3
h	0.2

$$w = \log_2 \frac{w_{target}}{w_{reference}}, \qquad h = \log_2 \frac{h_{target}}{h_{reference}}$$

Example: Only one object - Loss



Example: Only one object - Loss

Loss:

Ratio
(Hyperparameter)

$$L = \frac{1}{N}\left(L_{conf} + \alpha L_{loc}\right)$$

Confidence of class

Confidence of
localization

Ground Truth	Prediction
0	20%
100%	20%
0	80%
0	0%
-0.2	0.2
-0.05	-0.04
0.3	0.6
0.2	-0.2

Example: Only one object - Loss

L_{conf} (the usual Cross-Entropy):

- Confidence c (Output of a Softmax)
- Correct label p

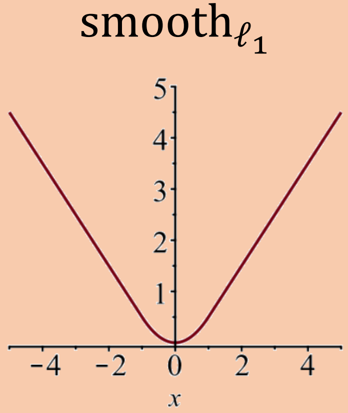
$$L_{conf} = -\sum_{n}^{N} \log c_n^p$$

Ground Truth c	Prediction p
0	20%
100%	20%
0	80%
0	0%
-0.2	0.2
-0.05	-0.04
0.3	0.6
0.2	-0.2

Example: Only one object - Loss

L_{loc} (Distance of vectors):

$$L_{loc} = +\sum_{n}^{N} \mathrm{smooth}_{\ell_1}\left(\vec{l}_n - \vec{g}_n\right)$$



Ground Truth $\vec{g}_n$	Prediction $\vec{l}_n$
0	20%
100%	20%
0	80%
0	0%
-0.2	0.2
-0.05	-0.04
0.3	0.6
0.2	-0.2

Multi Object Detection

Design a NN that can find

- several objects
- of varying shapes
- with multiple classes
- in arbitrary/varying scales
- (optimally in real time)

Multi Object Detection (SSD)

Problems

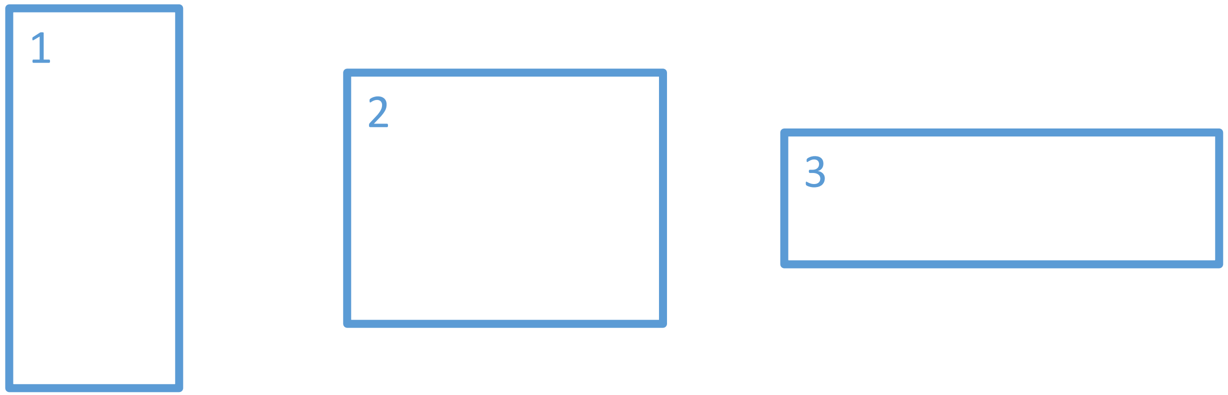
- Several objects
- Of varying shapes
- Many classes
- In arbitrary/varying scales
- Real time (60 FPS)

Possible Solutions

- Instead of one prediction, i.e. one bounding box per pixel, new class „Background“
- e.g. 4 or more reference bounding boxes instead of only one
- Each box (similar to segmentation of $n+1$ classes), use a pretrained network
- Output for „every pooling layer“
- GPU, network of Conv/Pooling layers

Single Shot Detector (SSD)

- Define d „possible“ default bounding boxes of different shapes:

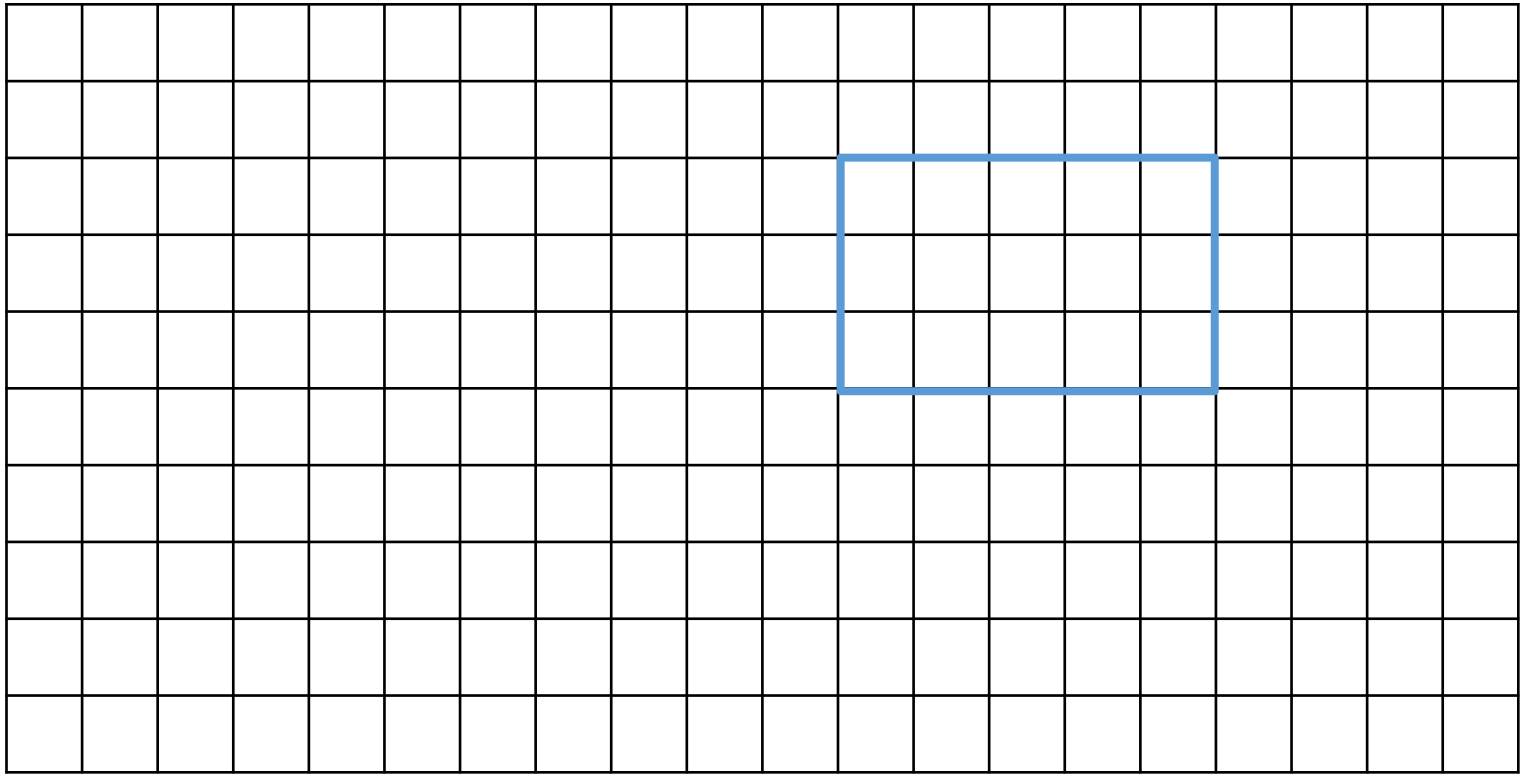


e.g. for people, … e.g. for cars, … e.g. license plates, …

- Parameters: $d^{cx}, d^{cy}, d^{w}, d^{h}$ (Center pixel, size in pixels)

Single Shot Detector (SSD)

Some image

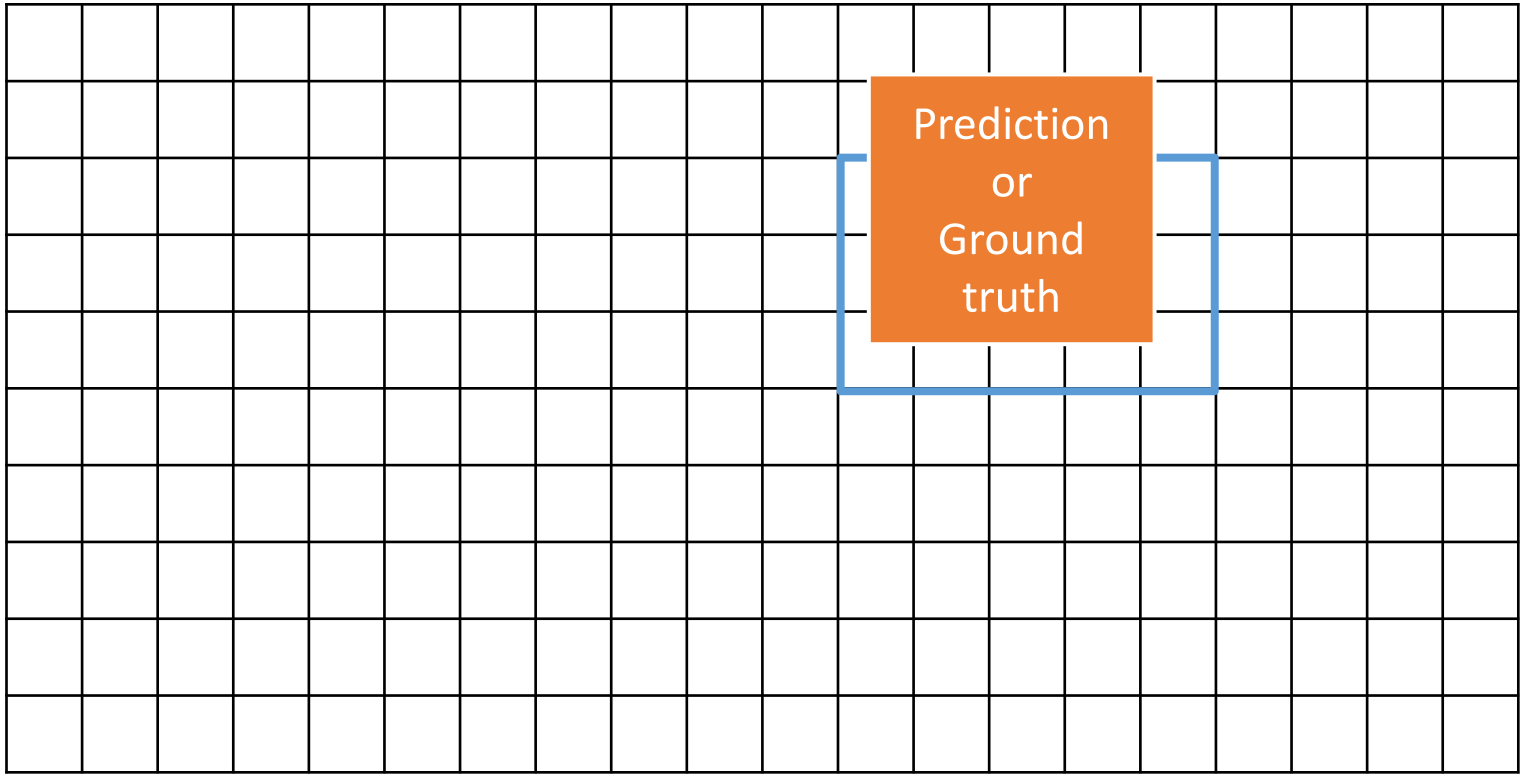


Parameters

$$d^{cx} = 13$$
$$d^{cy} = 3$$
$$d^{w} = 5$$
$$d^{h} = 3$$

Single Shot Detector (SSD)

Some image



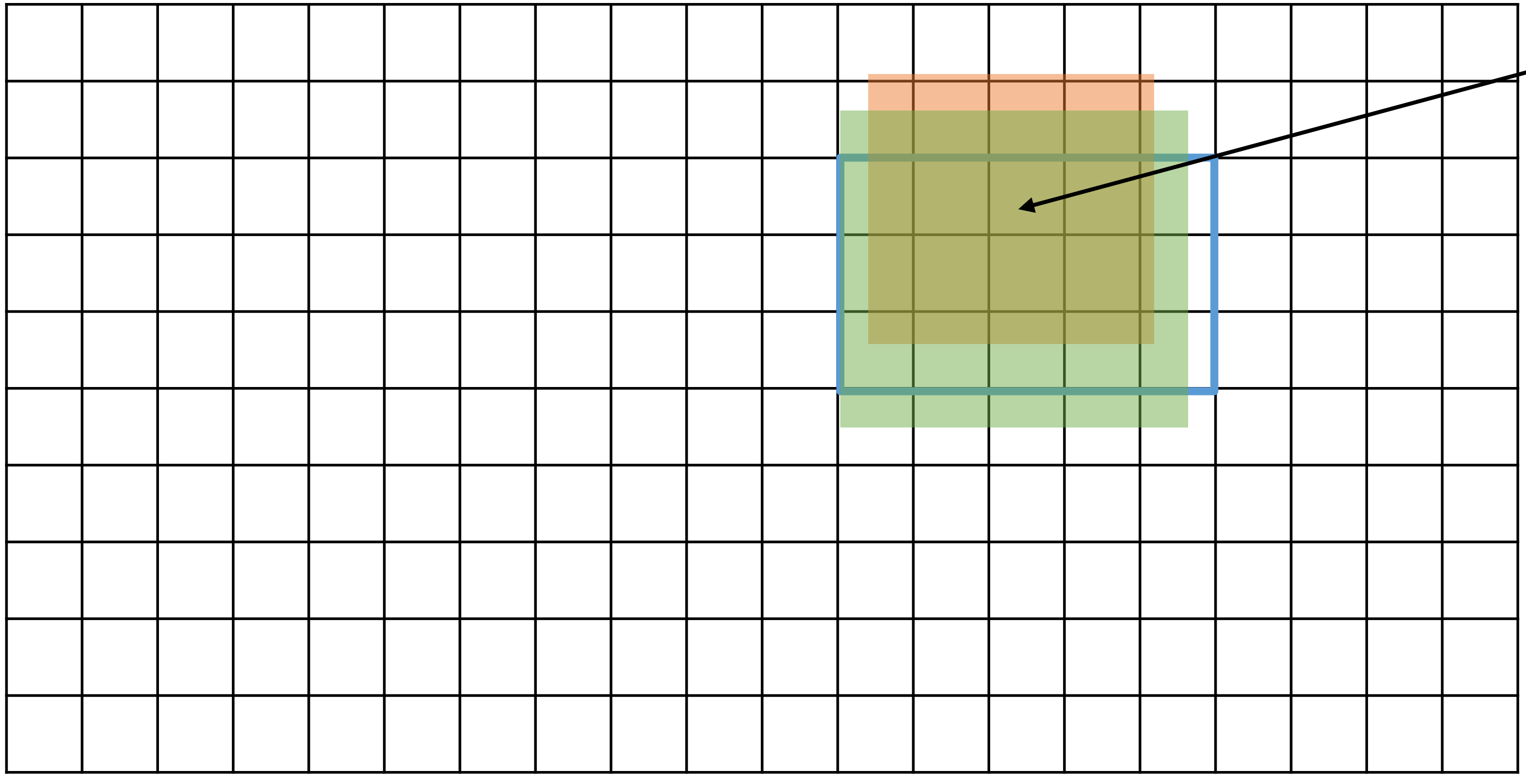
Parameters

- $d^{cx} = 13$
- $d^{cy} = 3$
- $d^{w} = 5$
- $d^{h} = 3$

Parameters (relative)

- $\hat{g}^{cx} = \frac{12.5-13}{5}$
- $\hat{g}^{cy} = \frac{2.3-3}{3}$
- $\hat{g}^{w} = \log\frac{3.3}{5}$
- $\hat{g}^{h} = \log\frac{3.2}{3}$

Single Shot Detector (SSD)



Count as correct, when IoU (Intersection over union) of Ground Truth and Prediction > 0.5

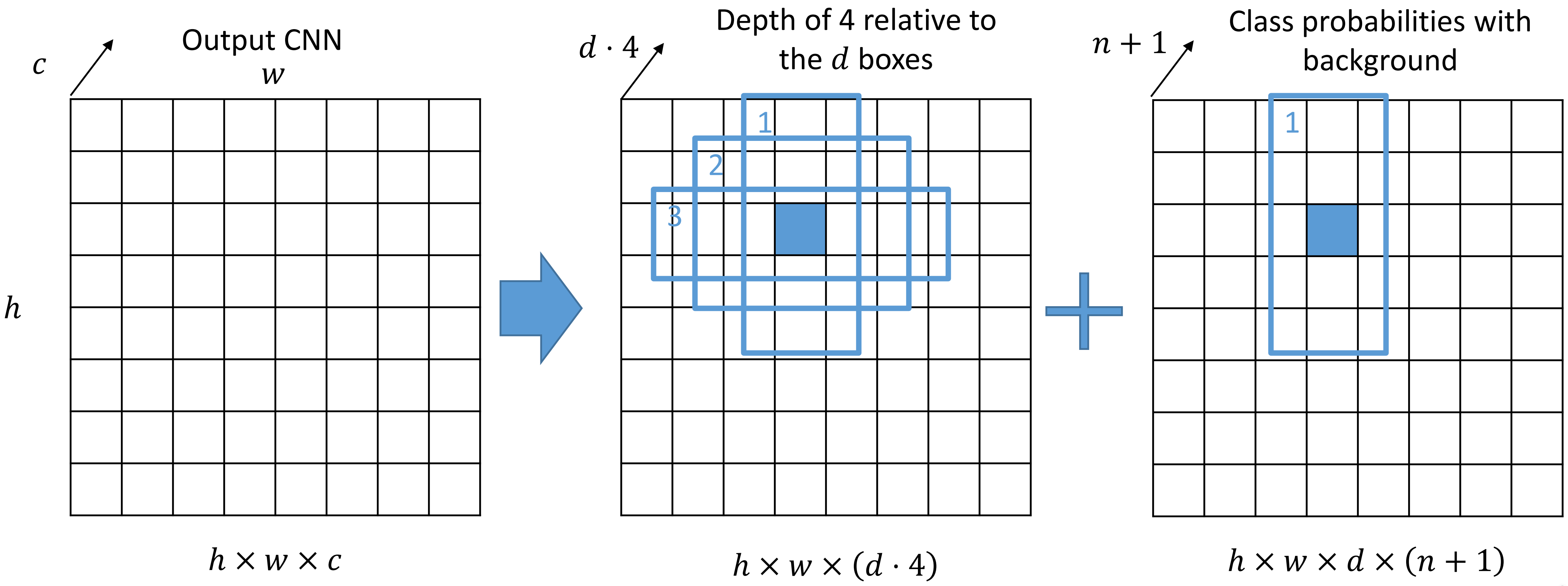
$$IoU = \frac{A \cap B}{A \cup B}$$

Single Shot Detector (SSD)

Summary so far:

- For every pixel d „default bounding boxes“ with 4 parameters each
- For every bounding box $n + 1$ classes (background class for „no object present“)
- Ground truth and prediction in relative coordinates (4 parameters) corresponding to the d default boxes

SSD: Dimensions



SSD: Dimensions

Output layer (for one scale) has:

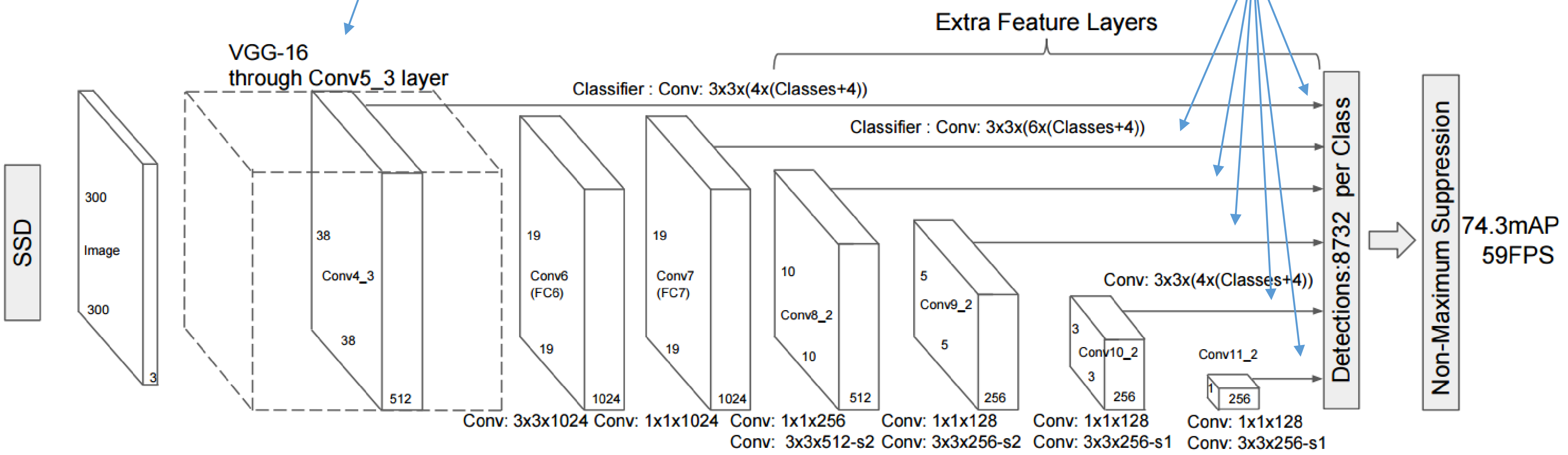
$$h \cdot w \cdot (d \cdot 4) + h \cdot w \cdot d \cdot (n + 1)$$
$$= h \times w \times d \times (4 + n + 1)$$

entries.

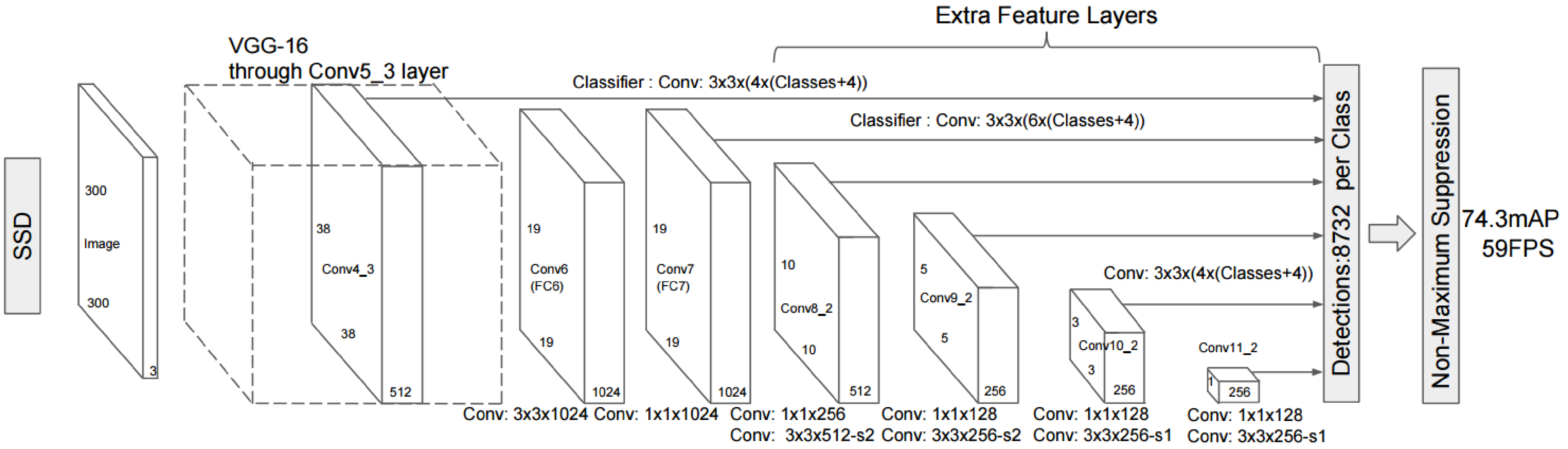
SSD: Architecture

Pretrained network for object detection (final layers removed)

Output for several scales (Concat)



SSD: Architecture



$h \times w$	38×38	19×19	10×10	5×5	3×3	1×1	
d	4	6	6	6	4	4	
Total per C.	5776	2166	600	150	36	4	8.732

SSD: Matching strategy

- Main question: When does a bounding box count as a positive match and when as a negative one?
- SSD easy approach in two steps
  1. First match every GT box to the prediction boxes (called default boxes in the paper)
     → Box with maximum overlap is positive match (one default box per GT box)
  2. Then match each remaining prediction box to the ground truth
     → every overlap > 0.5 counts as a positive match

SSD: Loss function

$$L = \frac{1}{N}\left(L_{conf} + \alpha L_{loc}\right)$$

Match of jth GT box with ith default box (0 or 1)

$$L_{conf}(x,c) = -\sum_{i \in Pos} x_{ij}^{p} \log c_i^{p} - \sum_{i \in Neg} \log c_i^{BG}$$

All currently positive matches (regardless if label is correct or not): Learn classes

All currently negative matches: learn background

SSD: Loss function

$$L = \frac{1}{N}\left(L_{conf} + \alpha L_{loc}\right)$$

Relative position/size of jth bounding box at position i

Prediciton

Ground Truth

$$L_{loc}(x, \hat{l}, g) = -\sum_{i \in Pos} x_{ij}^{p}\, \text{smooth}_{\ell_1}\left[\hat{l}_i - \hat{g}_{ji}\right]$$

All currently positive matches: learn offset

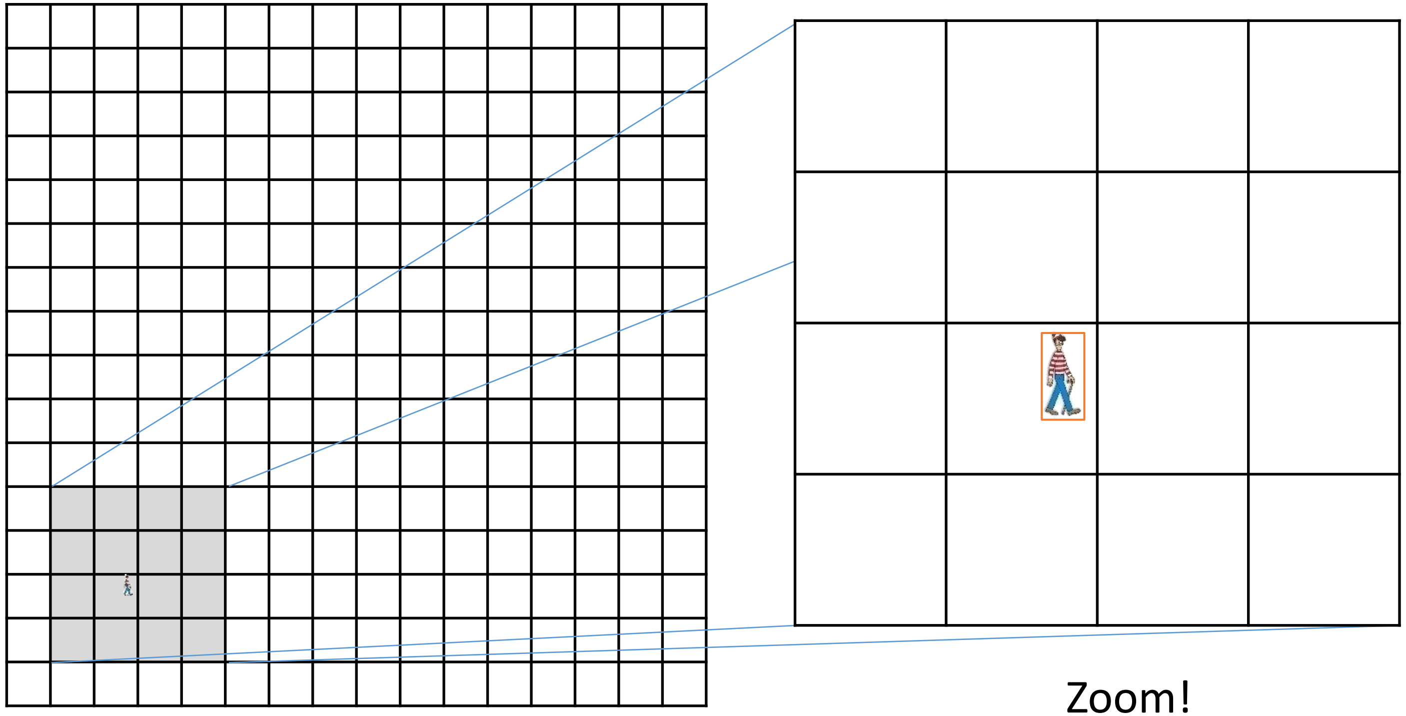
$$\hat{l}_i = \begin{pmatrix} \Delta l_{ij}^{cx} \\ \Delta l_{ij}^{cy} \\ \Delta l_{ij}^{w} \\ \Delta l_{ij}^{h} \end{pmatrix} \quad \hat{g}_{ji} = \begin{pmatrix} \Delta g_{ij}^{cx} \\ \Delta g_{ij}^{cy} \\ \Delta g_{ij}^{w} \\ \Delta g_{ij}^{h} \end{pmatrix} = \begin{pmatrix} \dfrac{g_j^{cx} - d_i^{cx}}{d_i^{w}} \\ \dfrac{g_j^{cy} - d_i^{cy}}{d_i^{h}} \\ \log \dfrac{g_j^{w}}{d_i^{w}} \\ \log \dfrac{g_j^{h}}{d_i^{h}} \end{pmatrix}$$

Default Box Shapes

Hypothetical input

16×16 Feature Map

Match with very small objects

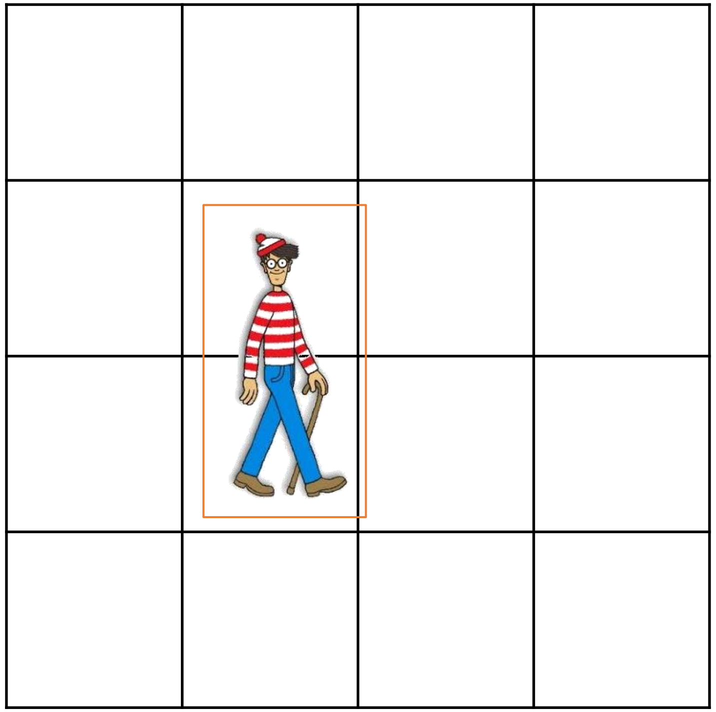


Zoom!

Default Box Shapes

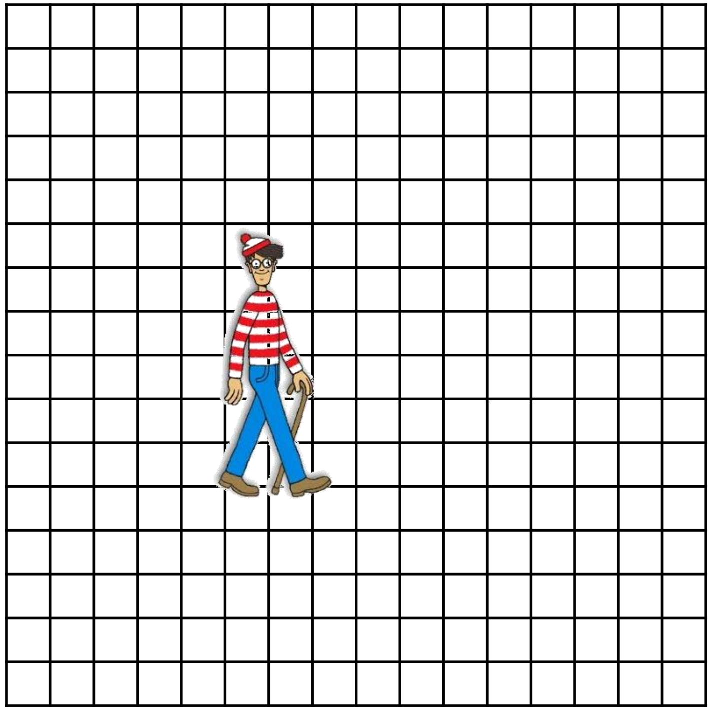
Hypothetical input

Match with very large objects



4×4 Feature Map

No Match with very large objects



16×16 Feature Map

Default Box Shapes

- Default box position corresponds to pixel position (center) in the CNN feature maps

  $\left(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|}\right)$, where $|f_k|$ is the size of the kth FM

- Choose m feature maps from the entire CNN and define scaling factor s_k (corresponds to the default box size):
  - last FM scaling factor (size) of $s_{max} = 0.9$
  - First FM factor of $s_{min} = 0.2$
  - Linear interpolation in between

Default Box Shapes

- Choose the aspect ratios

$$a_r \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{1}{1}, \frac{2}{1}, \frac{3}{1}\right\}$$

- Determine height and width:
  - $h = s_k / \sqrt{a_r}$
  - $w = s_k \sqrt{a_r}$

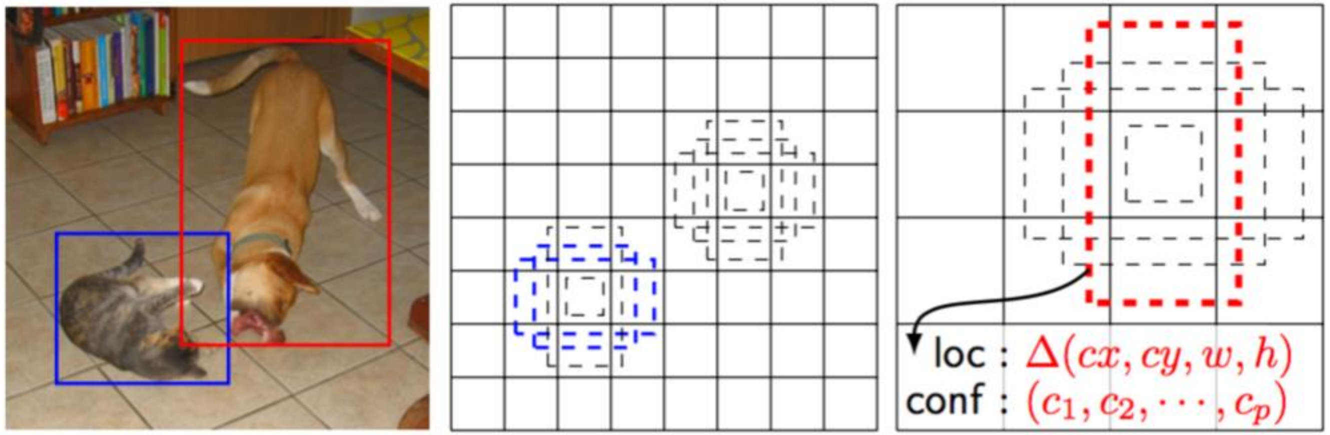
- Additional 6th BB with $a_r = 1$ and $s'_k = \sqrt{s_k s_{k+1}}$ (middle between the different scalings)
- With 4 BBs 1:3 and 3:1 are omitted

Default Box Scaling

Default BB positions in different feature maps

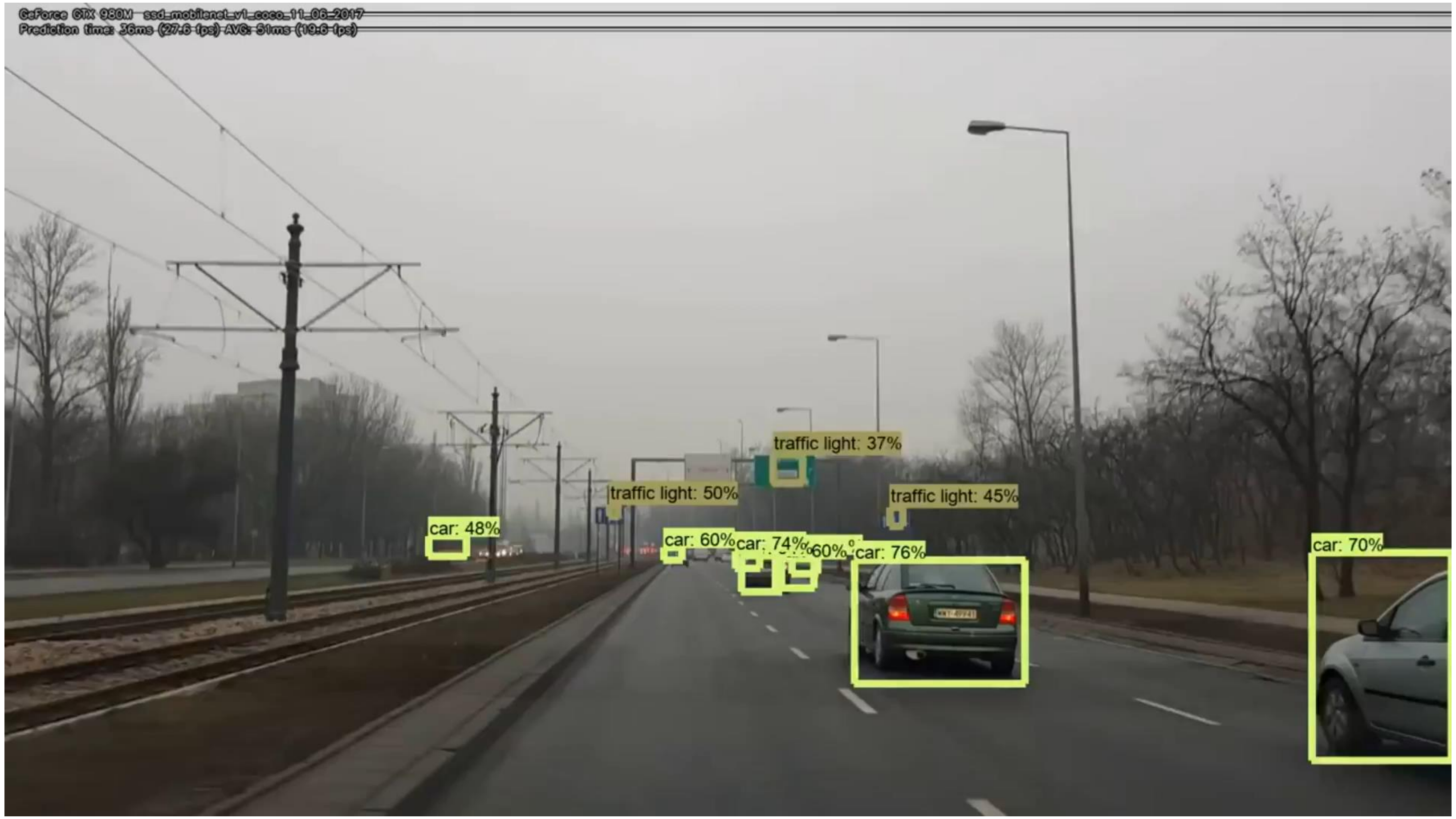
Match (2 BBs) with cat

Match (1 BB) with dog



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

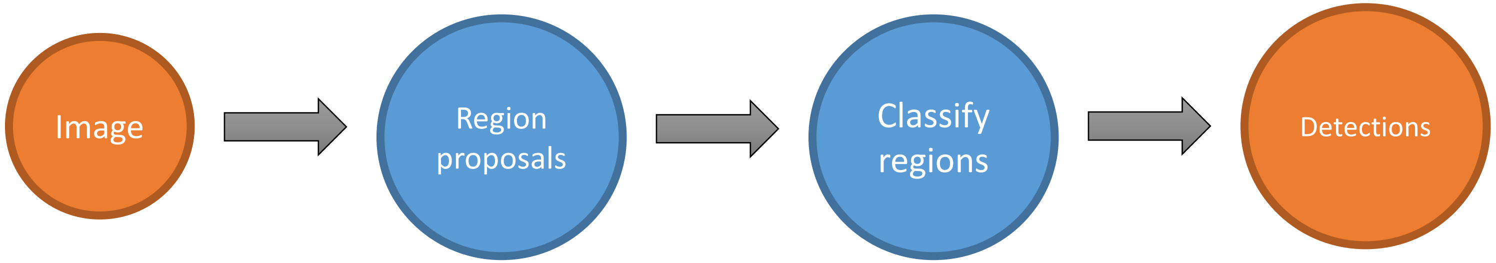
Application SSD: Traffic



Two-Stage Detectors

Faster R-CNN

Two-Stage Methods



- Generate region proposals (instead of sliding window)
- Classify every proposed region
- Two step methods are more accurate
- But require more computation power, so inference time is longer
- Examples
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN

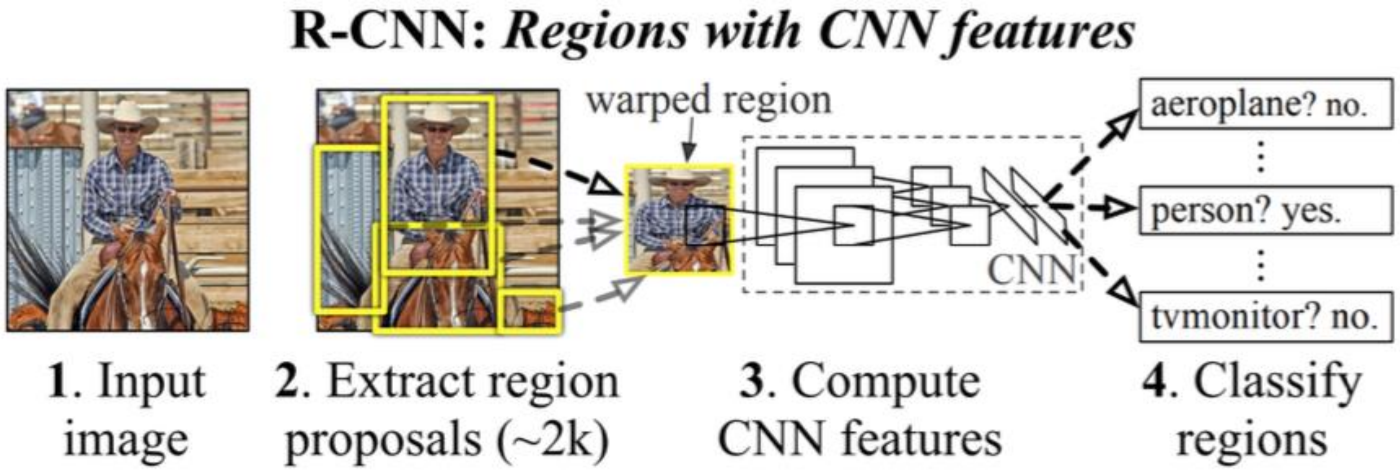
„Historic“ Development

1. Girshick et al., 2014: R-CNN
   - (slow) network, classifying region proposals
2. Girshick, 2015: Fast R-CNN
   - Faster network by improving the classification process
3. Ren et al., 2016: Faster R-CNN
   - Even faster by integrating region proposals into the architecture

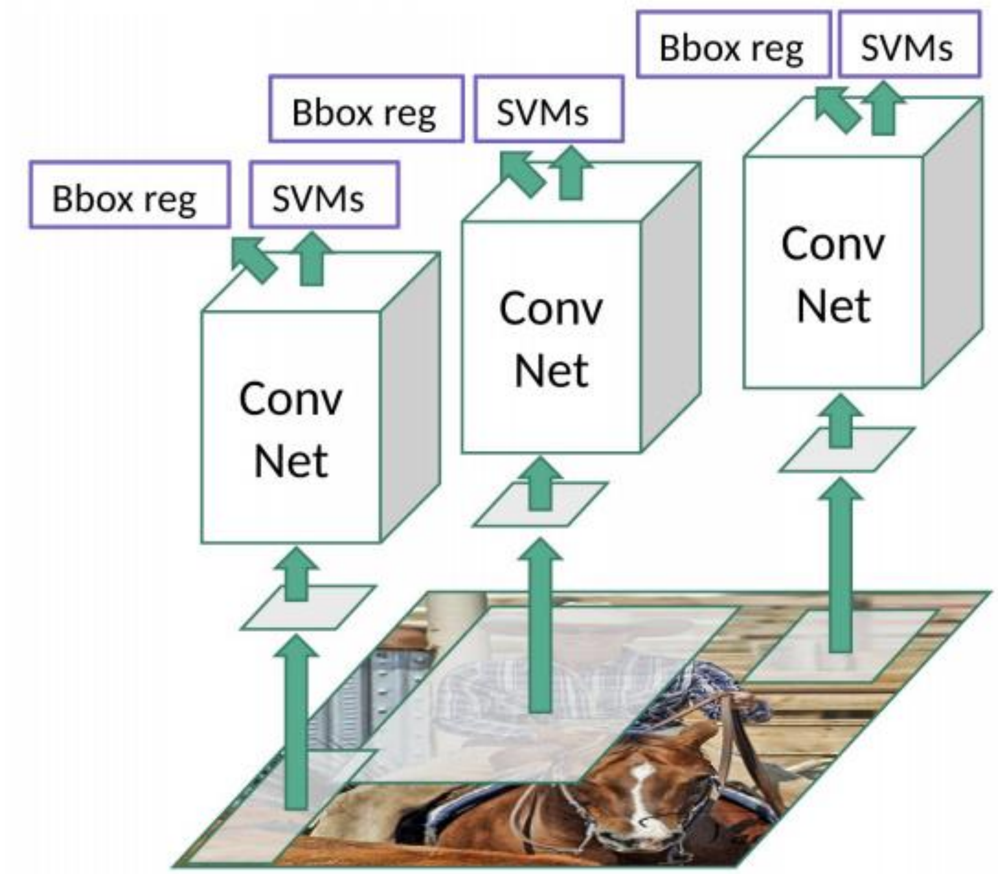
R-CNN [Girshick et al. (2014)]

3 Modules:

1. Extracting proposals via selective search [Uijlings et al.(2013)].
2. CNN for feature extraction
3. Classification using a Support Vector Machine (SVM)



R-CNN [Girshick et al. (2014)]



- Bbox module adjusts BB coordinates
- Selective Search extracts around 2000 proposals

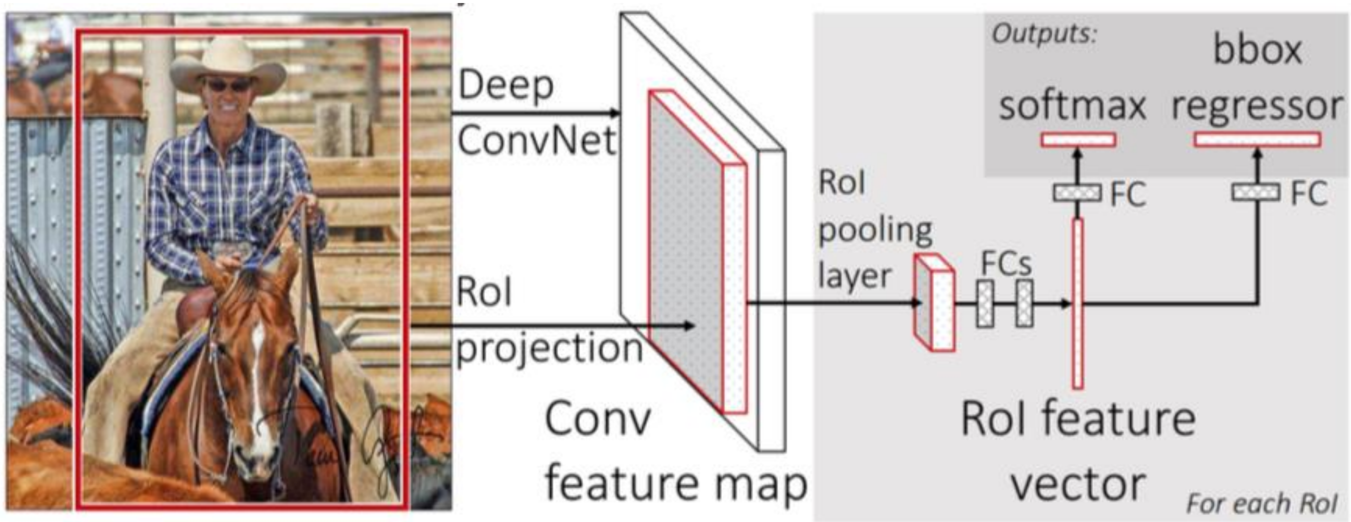
→2000 CNN computations

→Around 50 seconds per image

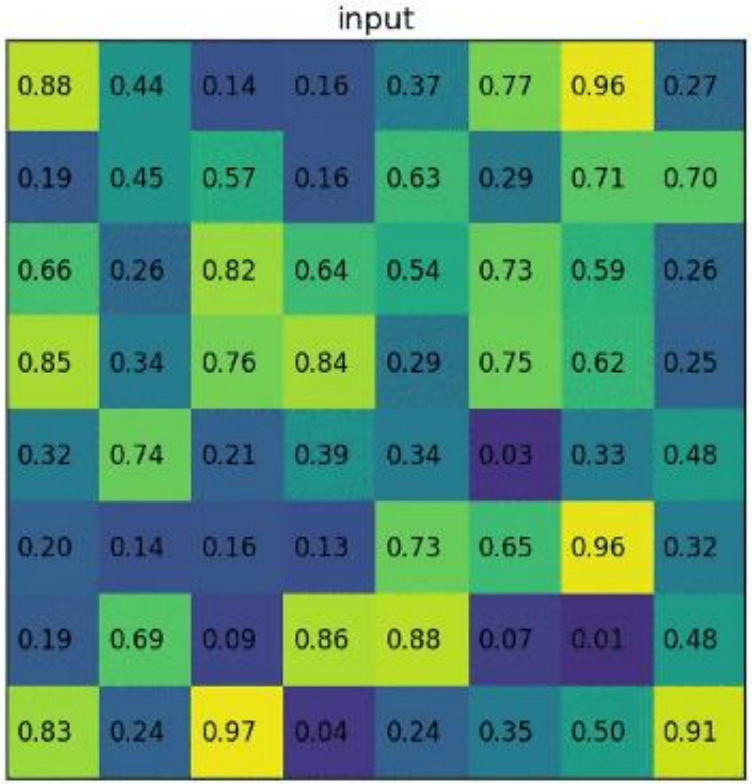
→Selective Search does not learn

Fast R-CNN [Girshick (2015)]

- Innovation: RoI (Region of Interest) pooling layer
  - Input: CNN Feature Maps + RoI coordinates as matrix
  - Output: vector of fixed length
- Fully Connected Layer used for both classification and regression (localization)



RoI Pooling: Principle



- Example:
  - An 8x8 Feature Map
  - A RoI of size 5x7, Position (0,0)
- Reduced to strictly defined dimensions (here 2x2)
- Then further processing using FC layers

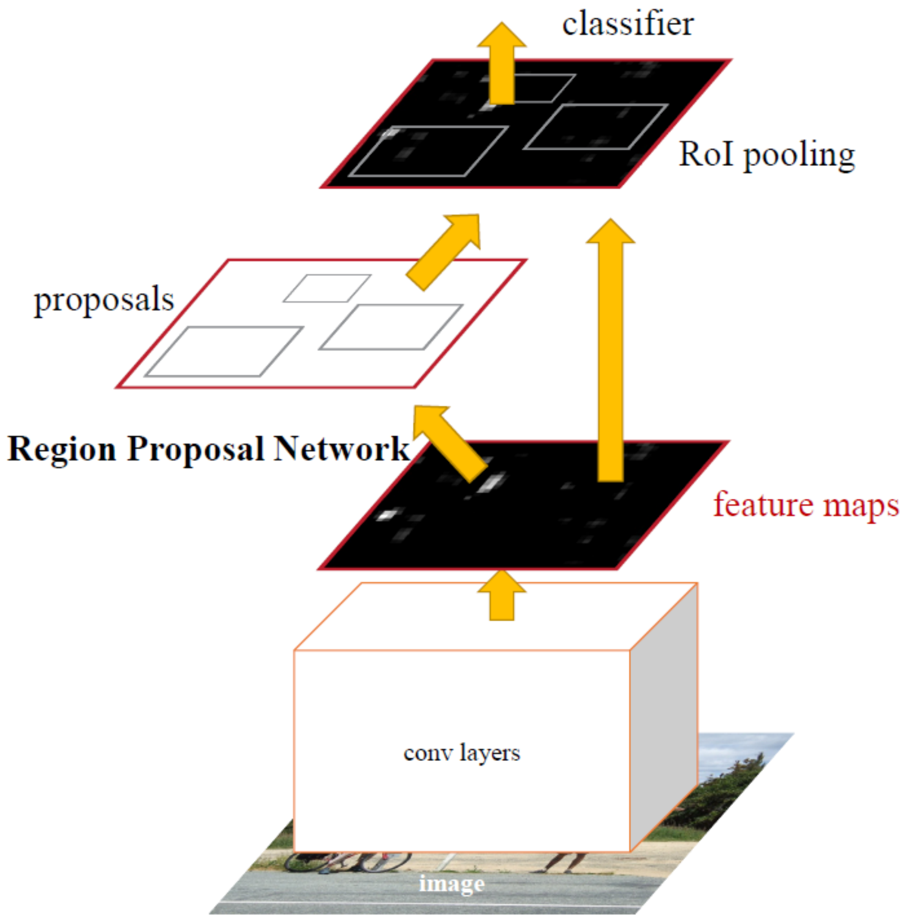
Fast R-CNN [Girshick (2015)]

- Significant acceleration of training and inference (+ better performance)
- But: Selective Search is still a bottleneck
  - No training parameters
  - Too many proposals

	Fast R-CNN			R-CNN		
	S	**M**	**L**	**S**	**M**	**L**
train time (h)	**1.2**	2.0	9.5	22	28	84
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0
▷ with SVD	**0.06**	0.08	0.22	-	-	-
test speedup	98×	80×	146×	1×	1×	1×
▷ with SVD	169×	150×	**213**×	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0
▷ with SVD	56.5	58.7	66.6	-	-	-

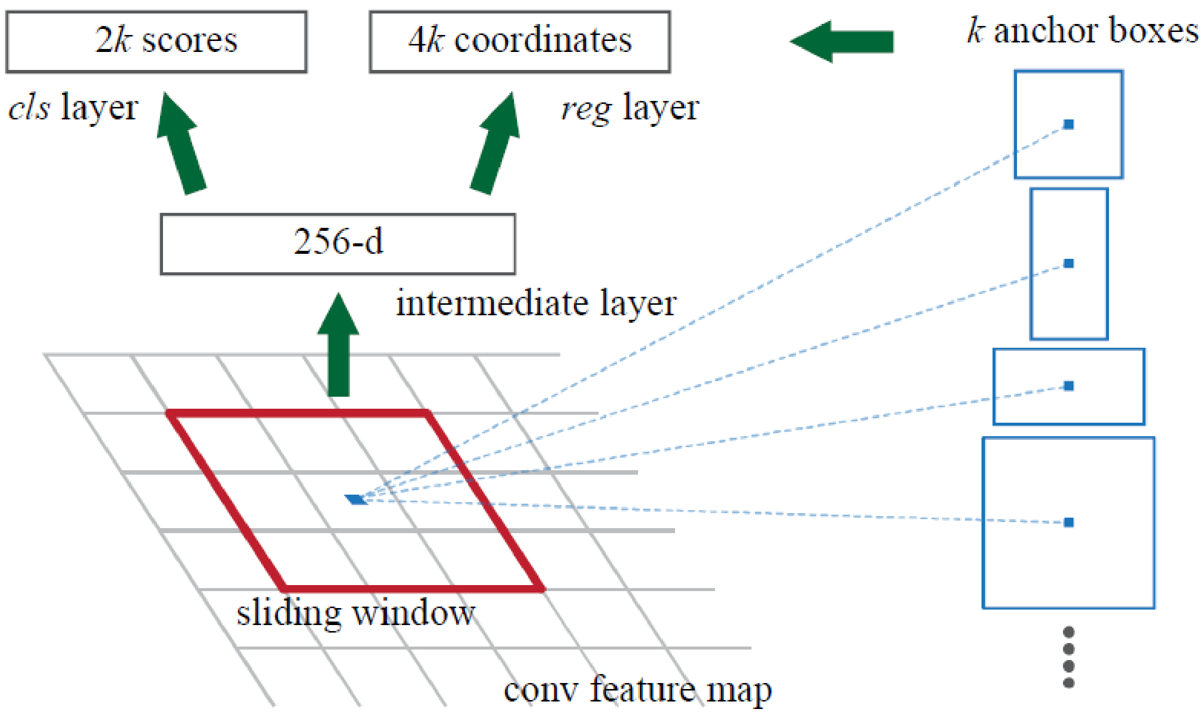
Faster R-CNN [Ren et al. (2015)]

- Extension: Region proposal network (RPN)
1. Region proposals are generated from CNN-output
2. Object detection same as Fast R-CNN (RoI pooling)
- Uses the same CNN for both steps (weight sharing)



Region Proposal Network

- Define von k anchor boxes (analogous to default boxes in SSD)
  - Defined by scale and aspect ratio
  - By default: $k = 9$ Boxes
- Sliding window with fixed size $n \times n$
  - Mapping to lower-dimensional representation (with $n \times n$ convolution layer)
  - Then *reg* and *cls* with 1×1 convolution layers



Proposal matching strategy

Positive Label (Object present)

- Anchor with highest IoU to GT-Box
- Anker with Overlap > 0.6

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Negative Label (no Object)

- IoU < 0.3 for all GT-Boxes

Neutral Label

- When no other criterium is met
- Does not go into loss

Loss-Function: RPN

Binary Cross Entropy

Weighting

$$\sum_{j=x,y,w,h} \mathrm{smooth}_{L_1}(t_j - t_j^*)$$

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Normalization
(optional)

ith
anchor

Object label
*: Ground Truth

Only non-
background

Coordinates
*: Ground Truth

Loss-Function: Fast R-CNN

Multi-Class Cross Entropy

Weighting

$$\sum_{j=x,y,w,h} \text{smooth}_{L_1}(t_j - t_j^*)$$

$$L(\{p\},\{t\}) = L_{cls}(p, p^*) + \lambda[p^* \geq 1]L_{reg}(t, t^*)$$

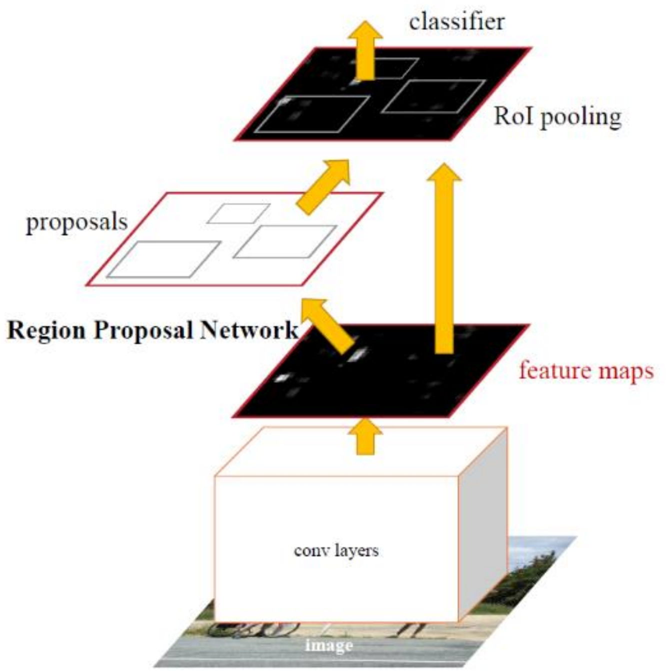
Object label

0 fpr p^*= 0
1 else

Coordinates

Training: Options

1. Alternating Training (4-Step)
   1. RPN trained separately
   2. Fast R-CNN trained with proposals
   3. RPN with Fast R-CNN initialized,
      only unique layers trained
   4. Same with Fast R-CNN → Conv-Layers shared
2. Approximate Joint Training
   - Combine RPN and Fast R-CNN Loss without Backpropagation at RoI pooling
3. Non-approximate Joint Training
   - Full compination of RPN and Fast R-CNN Loss with „RoI warping“ layer



After Training

Post-Processing and Evaluation

Recap: Dimensions

Output layer (for one scale) has

$$h \cdot w \cdot (d \cdot 4) + h \cdot w \cdot d \cdot (n+1)$$
$$= h \times w \times d \times (4 + n + 1)$$

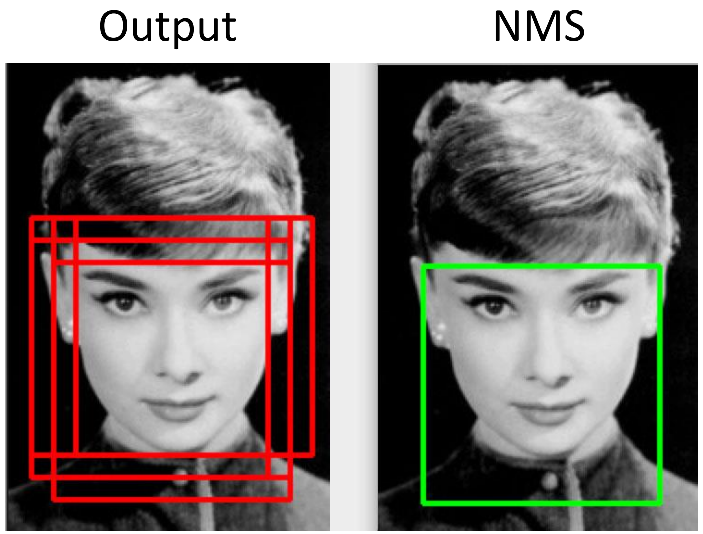
entries.

Faster R-CNN with similar dimensions

Prediction

Huge number of bounding boxes, but which should we keep?

- Confidence-threshold of 0.5 (removes most boxes)
- Non-maximum suppression (NMS)



Non-Maximum Suppression

1. Find BB with highest confidence
2. Remove all BBs with overlap higher than a threshold (e.g. 45%)
3. Repeat step 1 until no BBs are removed anymore

Example

Domain specific filtering

- Integration of domain specific knowledge for general architectures
- e.g. remove boxes of cars in images inside buildings
- e.g. remove certain illnesses depending on the organ that is pictured

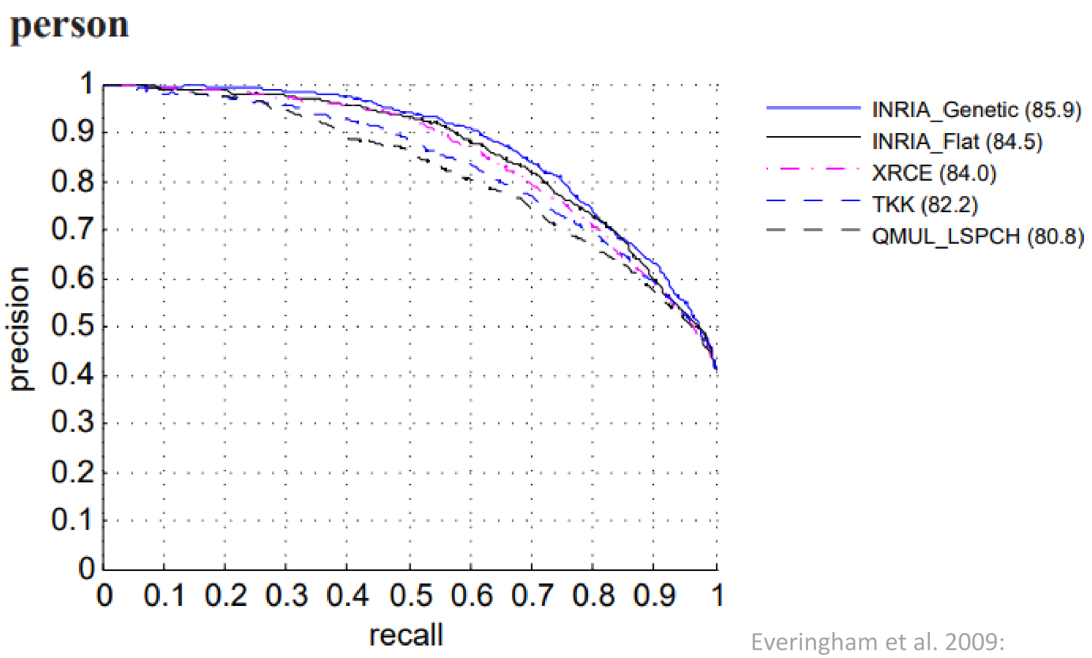
Evaluation: Mean Average Precision

- Standard metric from Pascal VOC and COCO
- Evaluates both classification and localization
- First compute precision and recall
  - Define a IoU threshold (e.g. 50%)
  - Determine all possible matches with this IoU
  - Define a confidence threshold (e.g. 50%)
  - Determine for every class above this threshold TP, FP, TN, FN
  - Compute Precision and Recall with these values

→ Confidence of multiple models can vary in accuracy

Evaluation: Mean Average Precision

- Variation of threshold values („rank“)
  - Change the threshold and compute Precision and Recall for every variation
  - Choose 11 equally distributed recall levels: [0, 0.1, 0.2, …, 1] = Rank
  - Average Precision (AP) is defined as the mean of precision at this recall/rank *r*
  - Use the maximum possible precision at rank *r*
  - mAP as mean over all classes
- COCO adds variation of IoU
  - Multiple mAP values for IoUs of e.g. 25%, 50%, 75%



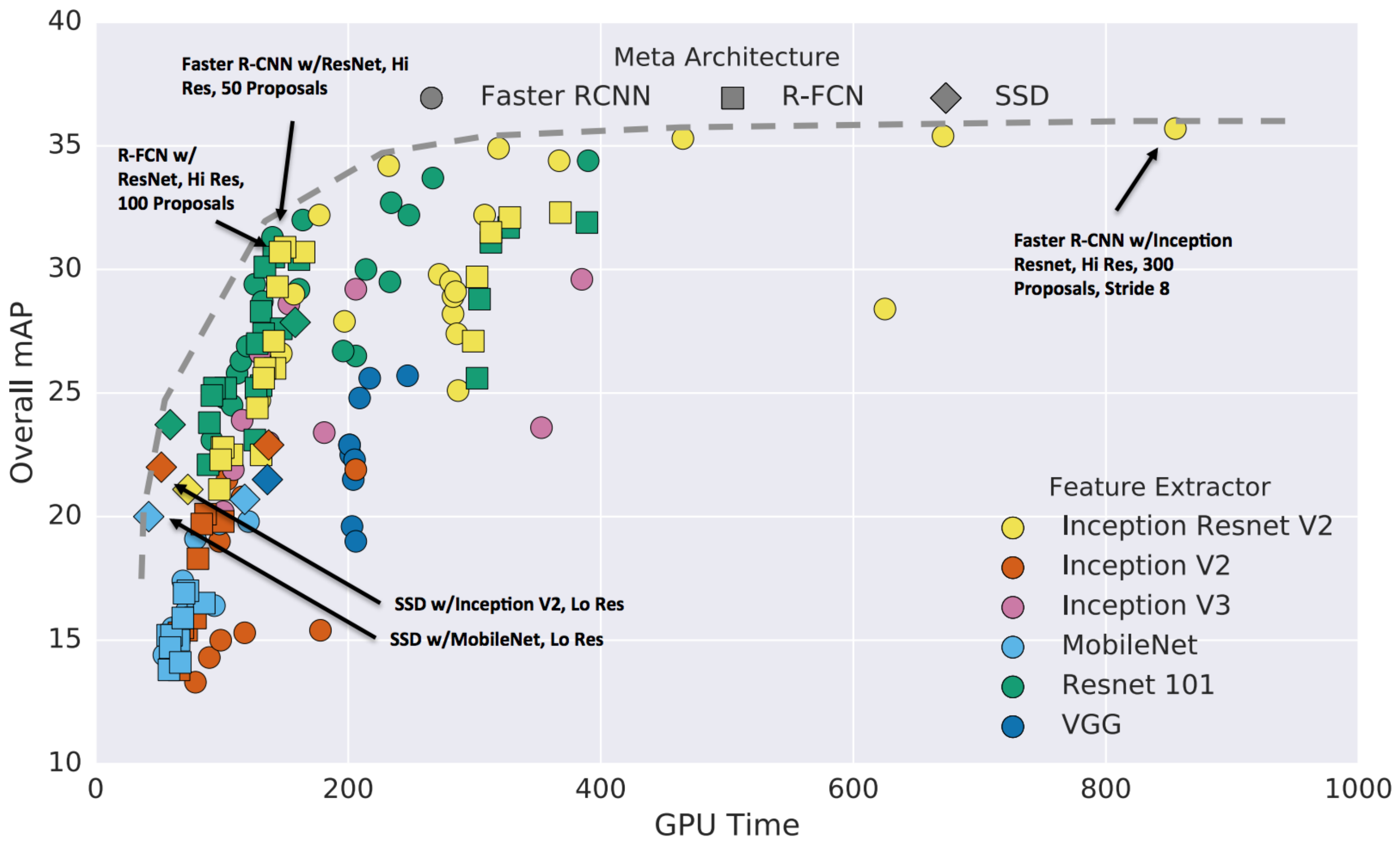
(b) top 5 by AP

Everingham et al. 2009: The PASCAL Visual Object Classes (VOC) Challenge

Example curve for AP computation

Trade-Off Real time detection

Example 1: Comparing different detectors and backbones

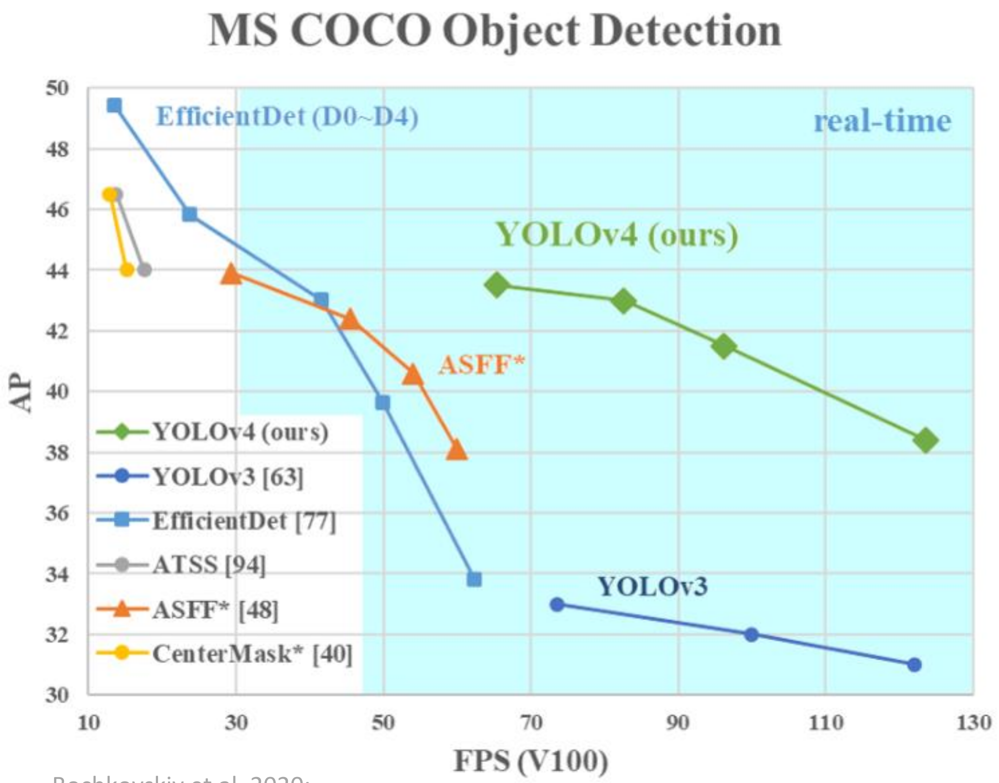


Huang et al. 2017: speed/accuracy trade-offs for modern convolutional object detectors

Trade-Off Real time detection

Example 2

- YOLOv4:
  - Comparison of different architectures
  - Trade-Off between accuracy (AP) and speed (FPS)
  - Computed on Tesla V100
- For all algorithms higher FPS equals to lower AP



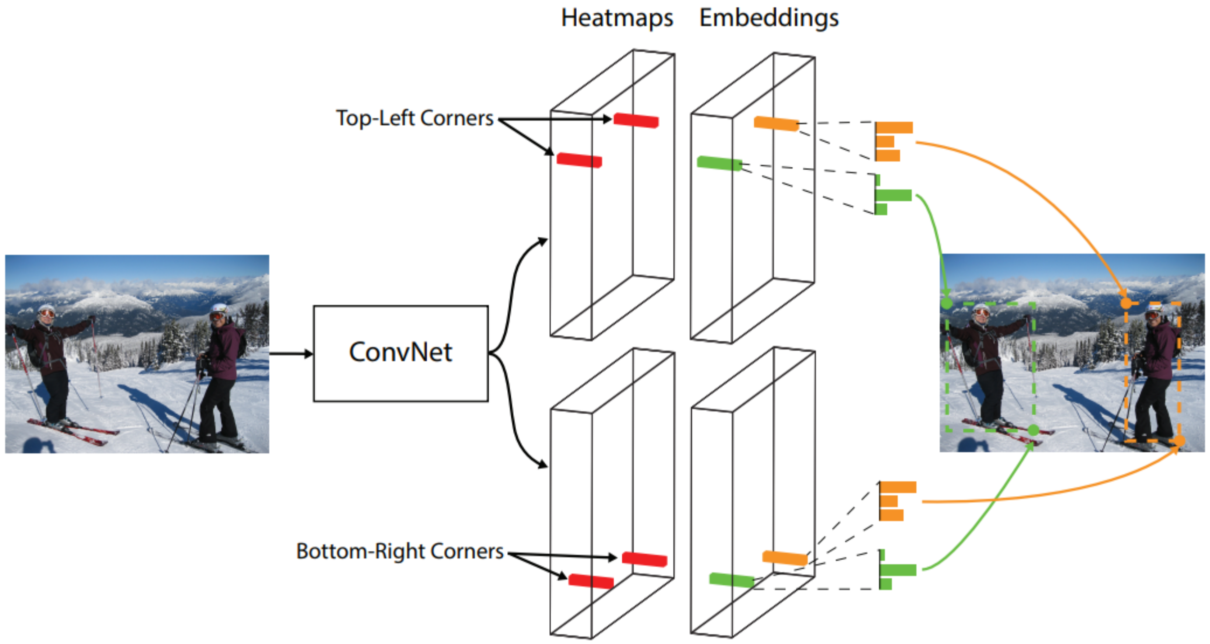
Bochkovskiy et al. 2020:
YOLOv4: Optimal Speed and Accuracy
of Object Detection

Outlook: Anchor free Methods

CornerNet, CenterNet

CornerNet [Law and Deng (2019)]

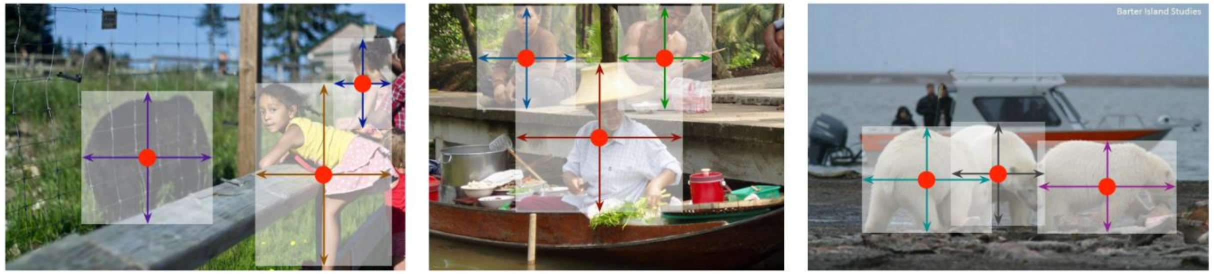
- One-Stage Detector
- Object defined as a pair of „Keypoints“ = corner pairs
- Trained to detect and assign corners
- New layer: Corner Pooling
- 42.2% mAP on COCO



Law and Deng et al. 2019:
CornerNet: Detecting Objects as
Paired Keypoints

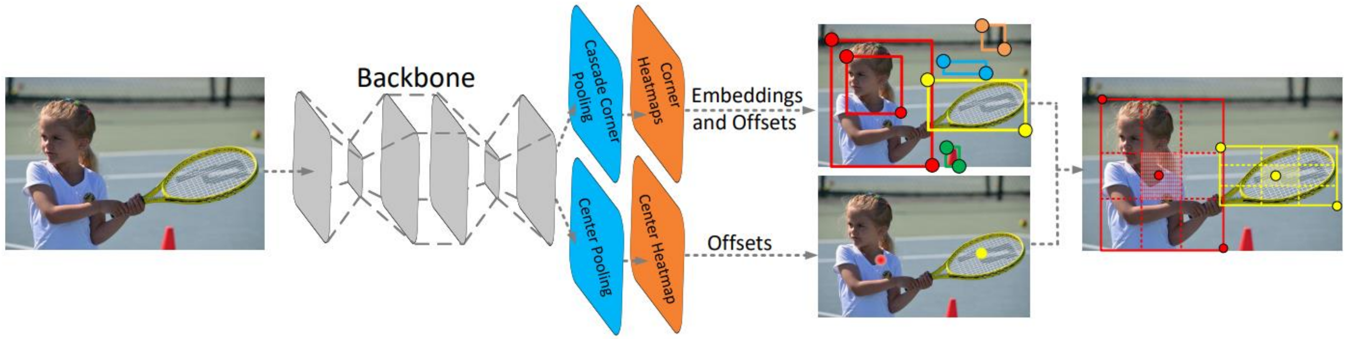
CenterNet(s): Objects as Points

- Predict center, regress object size (up to 45.1% mAP)



Zhou et al. 2019:
Objects as Points

- CornerNet extention to three points (47.0% mAP on COCO)



Duan et al. 2019:
CenterNet: Keypoint Triplets for
Object Detection

Outlook: Exercise and next lecture

- Exercise: SSD in detail
  - Inference with SSD
  - Change different parameters/structure of SSD
  - Application based questions
- Next lecture: Sequence2Sequence
  - Application of recurrent networks
  - Examples
  - CTC algorithm