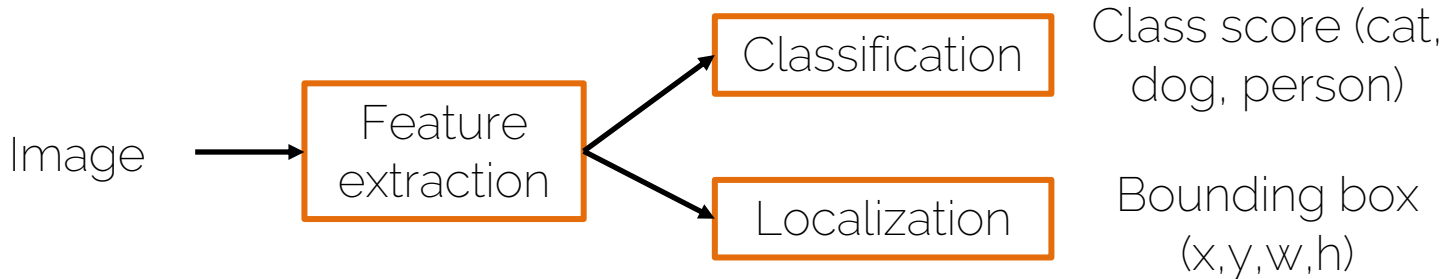


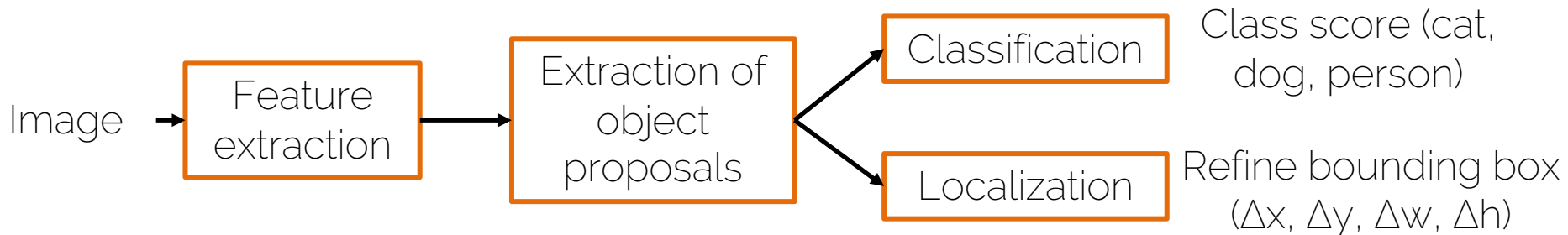
Object detection (part 1 recap)

Types of object detectors

- One-stage detectors



- Two-stage detectors

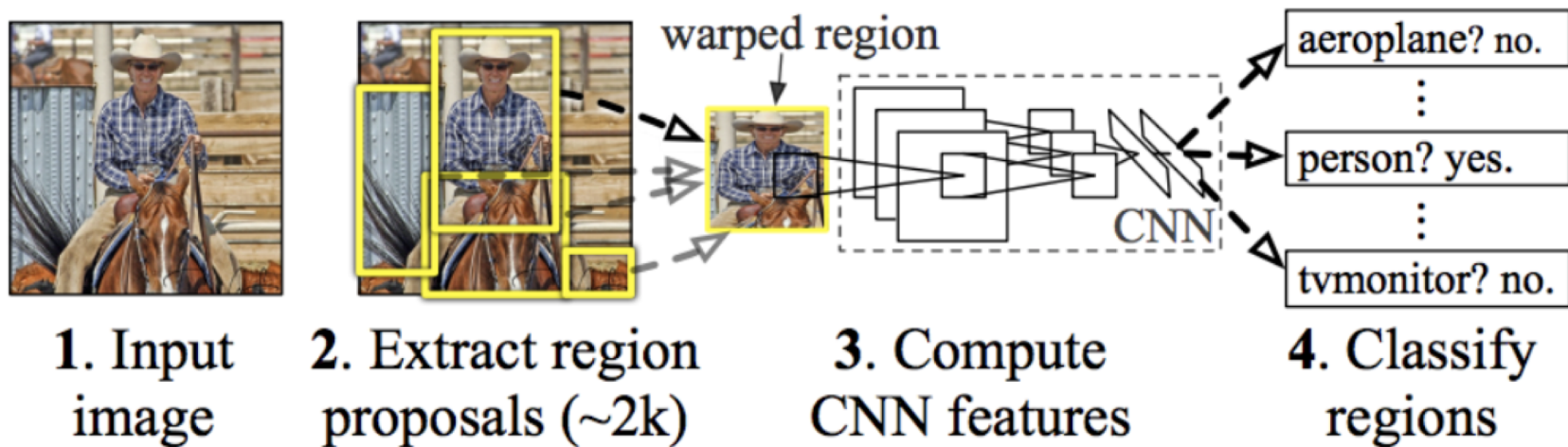


Types of object detectors

- One-stage detectors
 - YOLO, SSD, RetinaNet
 - CenterNet, CornerNet, ExtremeNet
- Two-stage detectors
 - R-CNN, Fast R-CNN, Faster R-CNN ←
 - SPP-Net, R-FCN, FPN

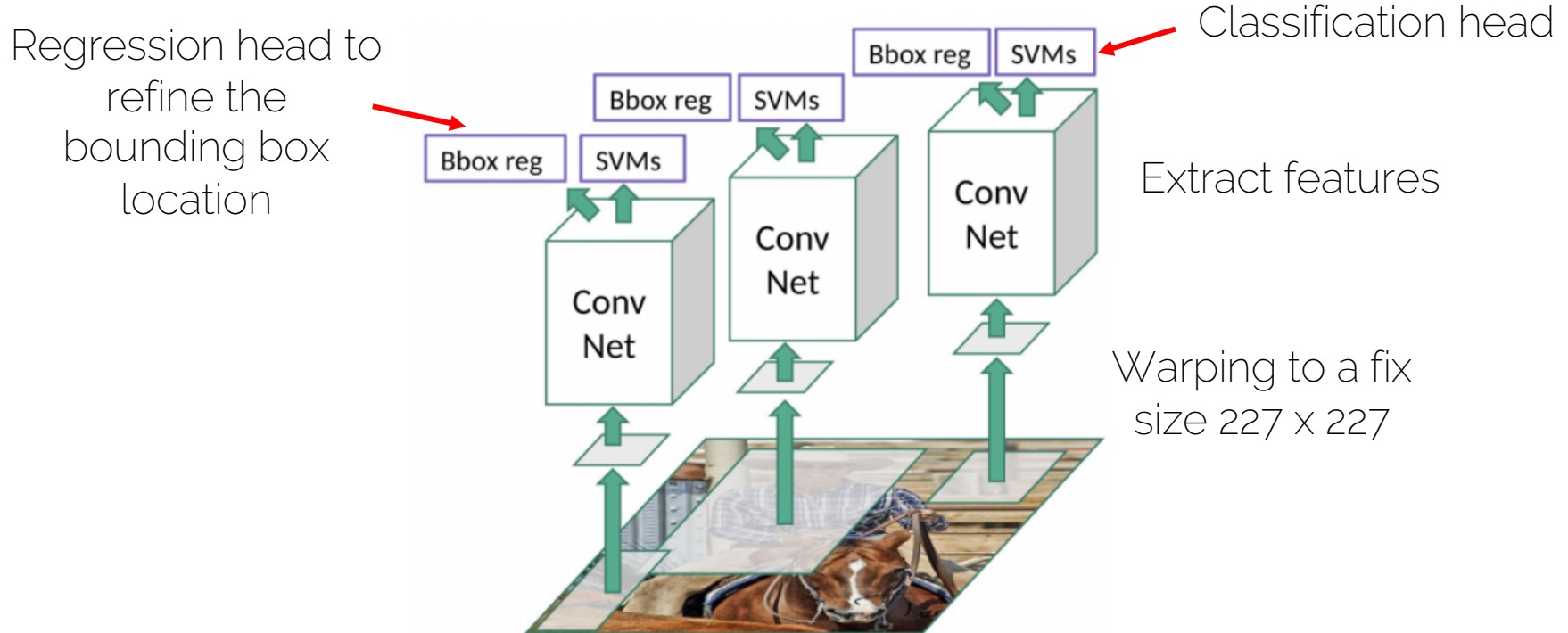
Two-stage detectors

R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

R-CNN

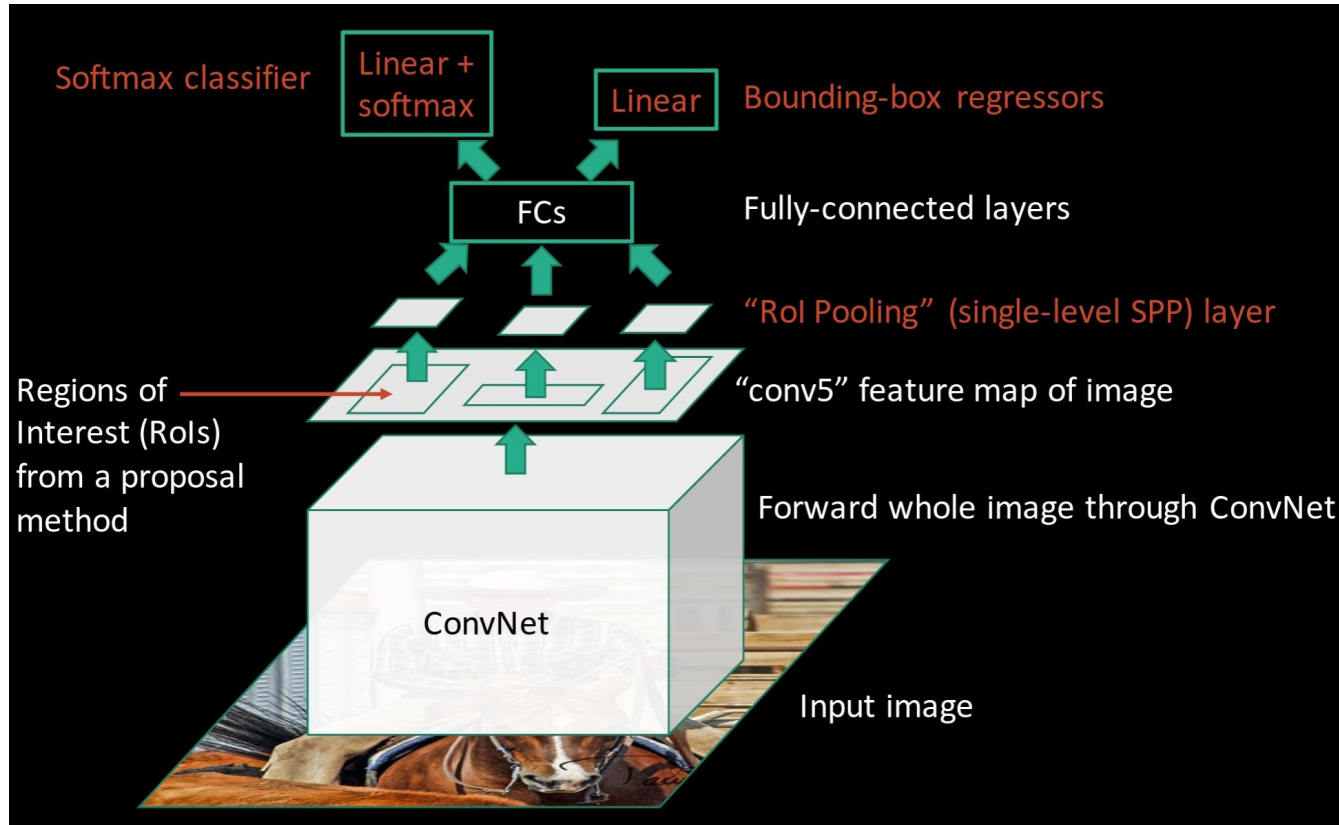


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

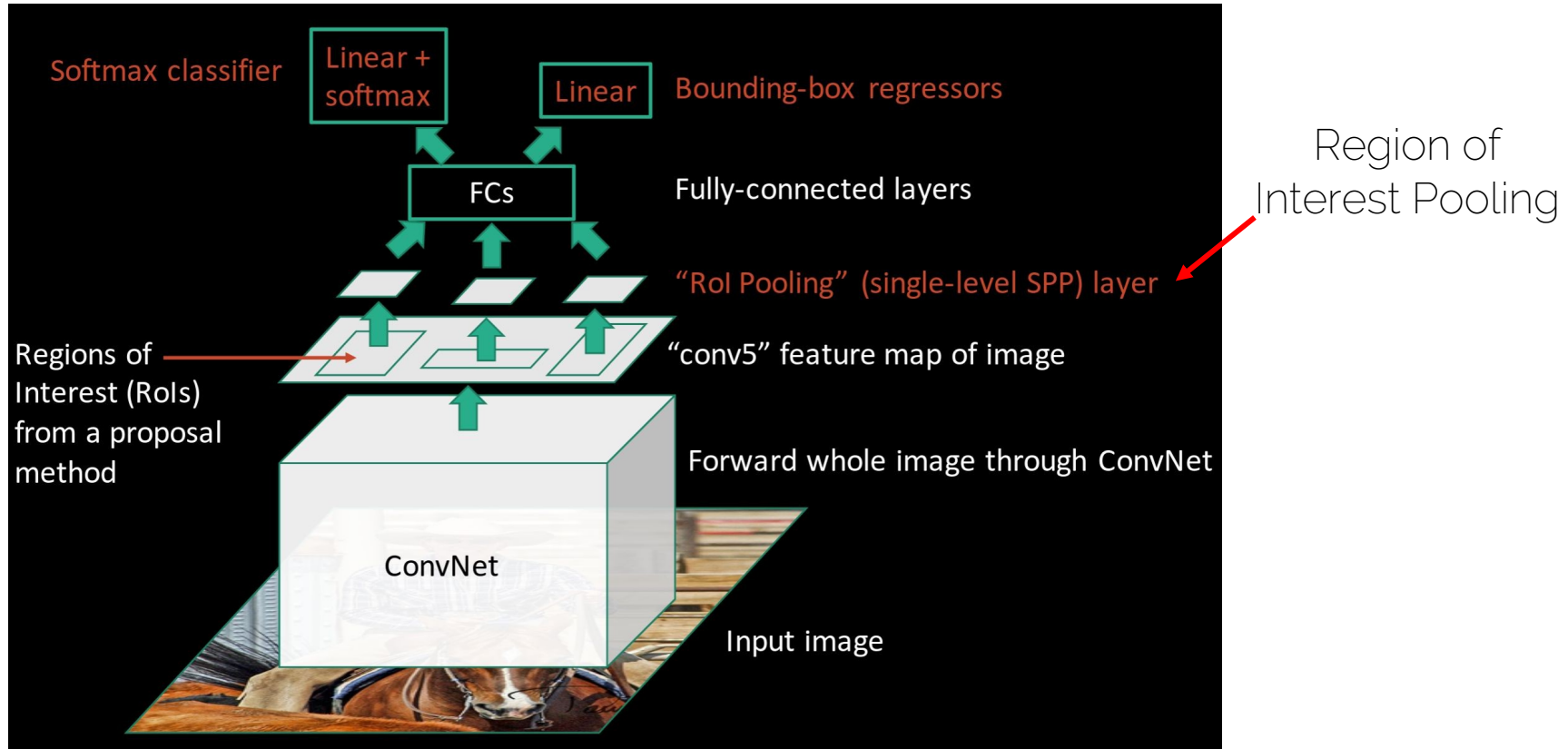
R-CNN

- Training scheme:
 - 1. Pre-train the CNN on ImageNet
 - 2. Finetune the CNN on the number of classes the detector is aiming to classify (softmax loss)
 - 3. Train a linear Support Vector Machine classifier to classify image regions. One SVM per class! (hinge loss)
 - 4. Train the bounding box regressor (L2 loss)

Fast R-CNN



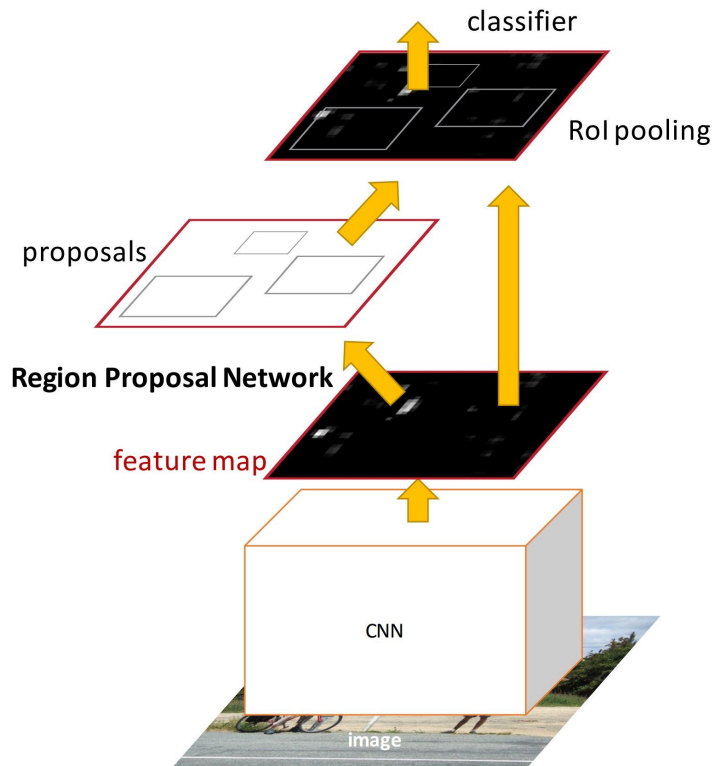
Fast R-CNN



Fast R-CNN Results

- Problem to be solved:
 - Proposal generation is still computed in a separate step, which makes it time consuming

Faster R-CNN:



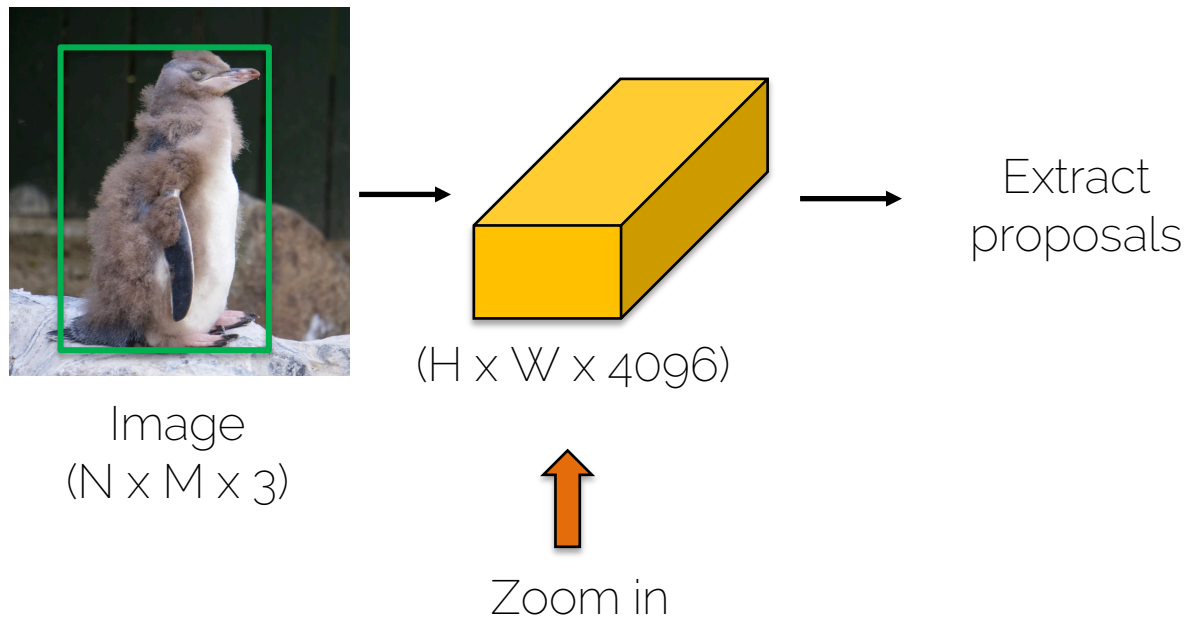
- Solution: Have the proposal generation integrated with the rest of the pipeline
- Region Proposal Network (RPN) trained to produce region proposals directly.
- After RPN, everything is like Fast R-CNN

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

Object detection (part 2)

Region proposal network

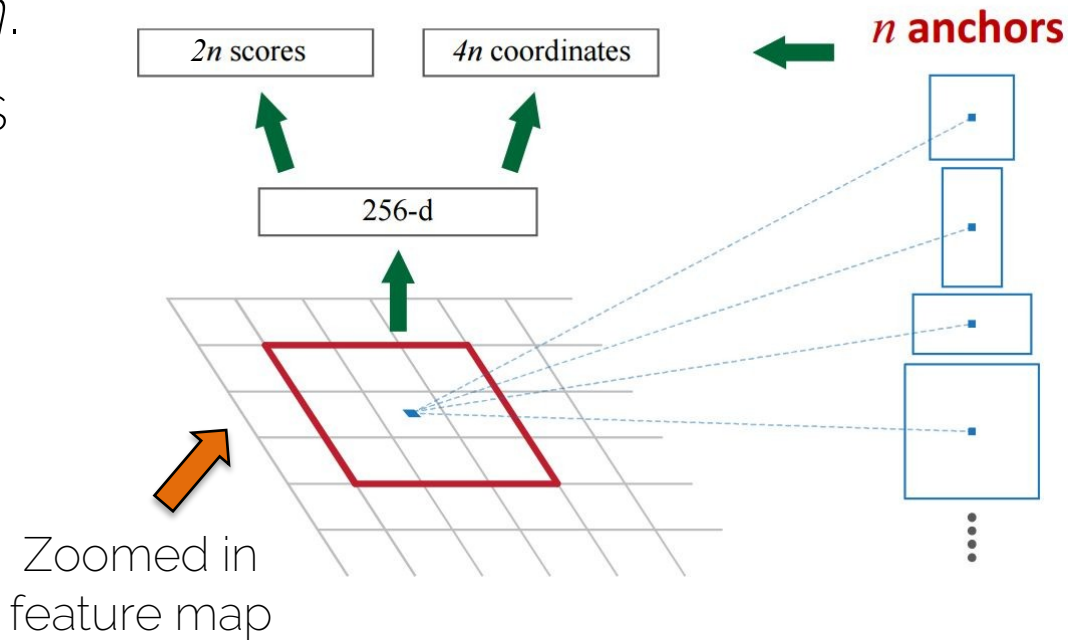
- How to extract proposals



- How many proposals?
- ✓ We need to decide a fixed number
- Where are they placed?
- ✓ Densely

Region proposal network

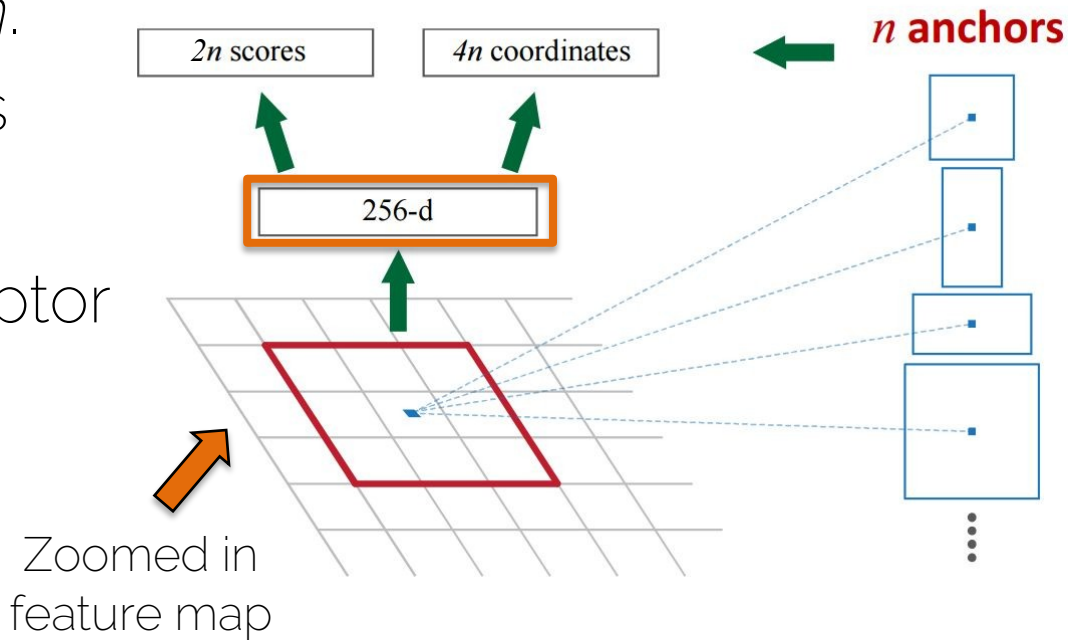
- We fix the number of proposals by using a set of $n=9$ anchors *per location*.
- 9 anchors = 3 scales and 3 aspect ratios



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

Region proposal network

- We fix the number of proposals by using a set of $n=9$ anchors *per location*.
- 9 anchors = 3 scales and 3 aspect ratios
- We extract a descriptor *per location*



Region proposal network

- How to extract proposals

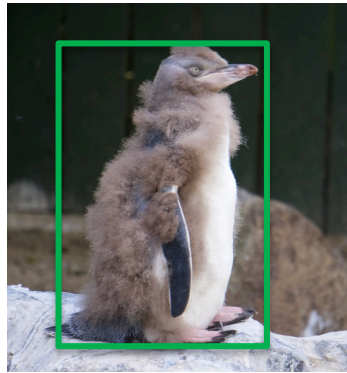
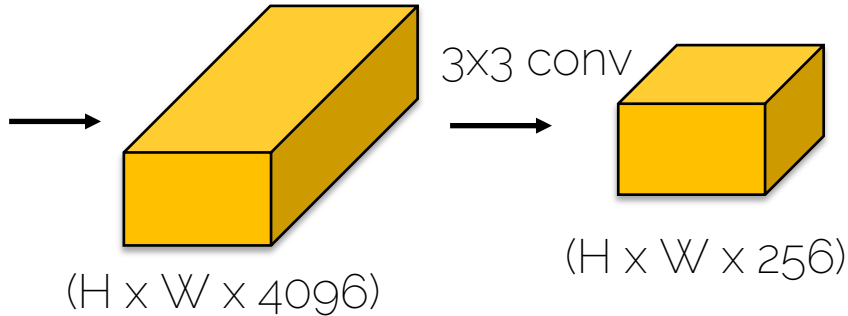


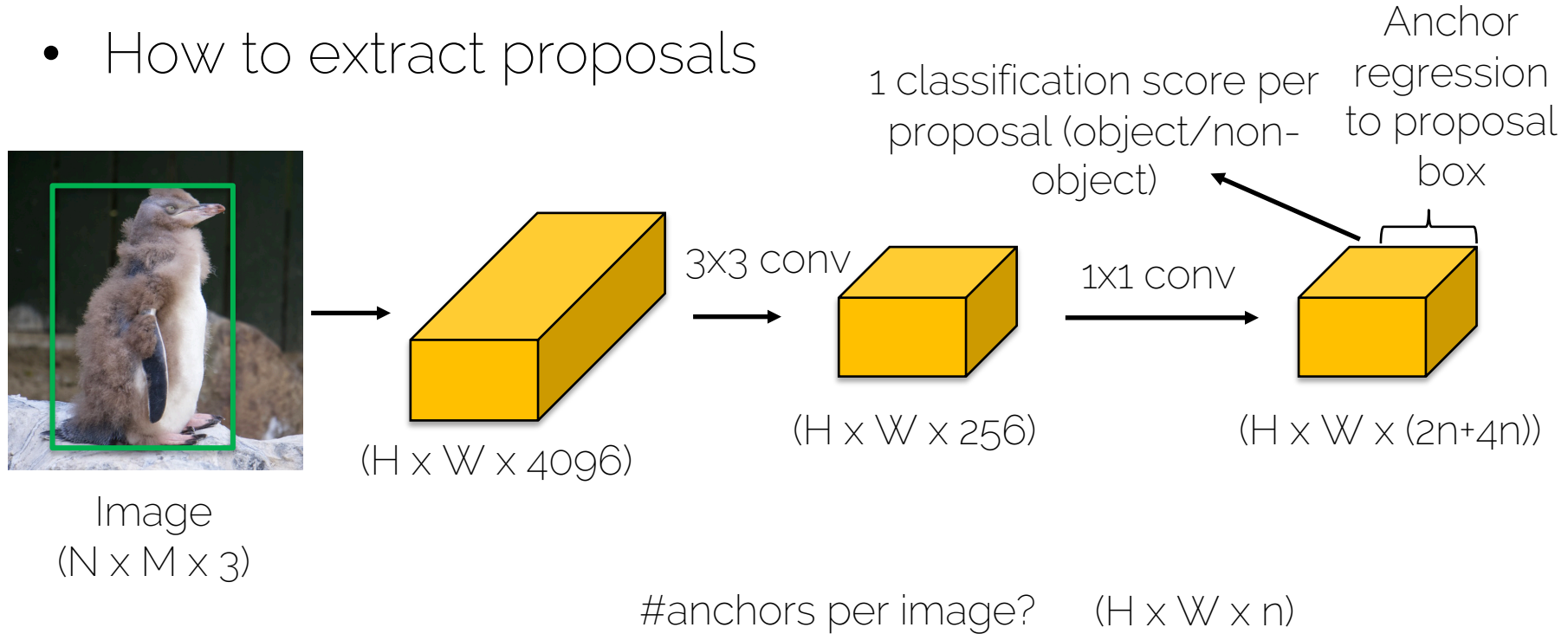
Image
($N \times M \times 3$)



#anchors per image? ($H \times W \times n$)

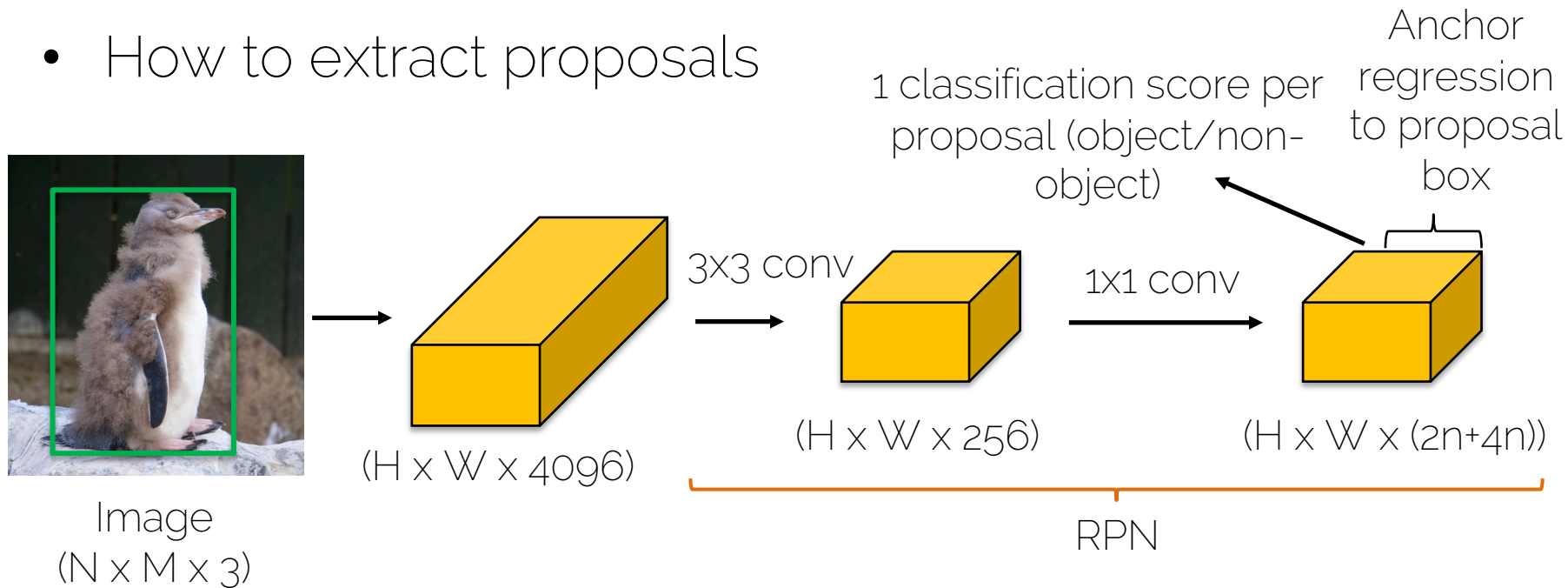
Region proposal network

- How to extract proposals



Region proposal network

- How to extract proposals



Per feature map location, I get a set of anchor correction and classification into object/non-object

RPN: training and losses

- Classification ground truth: We compute p^* which indicates how much an anchor overlaps with the ground truth bounding boxes

$$p^* = 1 \quad \text{if} \quad \text{IoU} > 0.7$$

$$p^* = 0 \quad \text{if} \quad \text{IoU} < 0.3$$

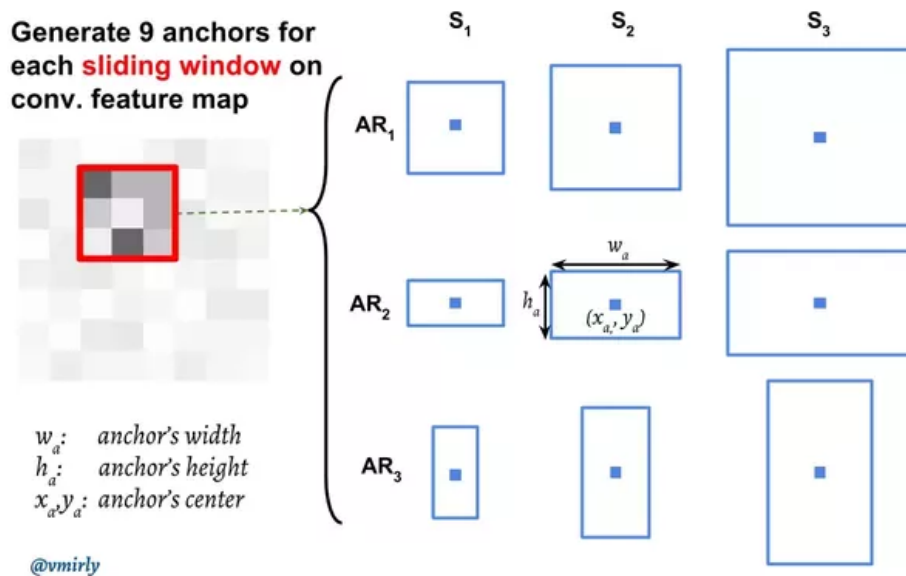
- 1 indicates the anchor represent an object (foreground) and 0 indicates background object. The rest do not contribute to the training.

RPN: training and losses

- For an image, we randomly sample 256 anchors to form a mini-batch (balanced objects vs. non-objects)
- We calculate the classification loss (binary cross-entropy).
- Those anchors that do contain an object are used to compute the regression loss

RPN: training and losses

- Each anchor is described by the center position, width and height x_a, y_a, w_a, h_a



RPN: training and losses

- Each anchor is described by the center position, width and height x_a, y_a, w_a, h_a
- What the network actually predicts are t_x, t_y, t_w, t_h

Normalized x $t_x = (x - x_a)/w_a,$ $t_y = (y - y_a)/h_a,$ Normalized y

Normalized width $t_w = \log(w/w_a),$ $t_h = \log(h/h_a),$ Normalized height

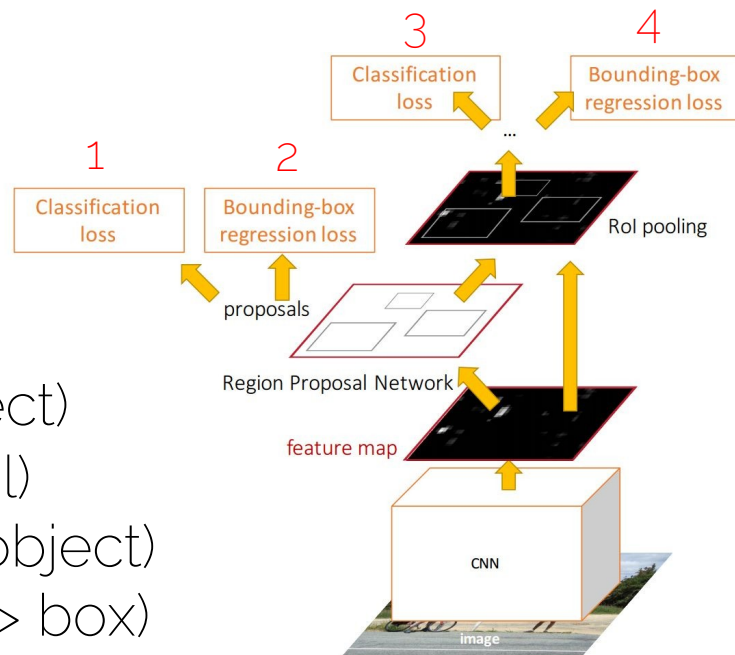
- Smooth L1 loss on regression targets

Faster R-CNN: Training

- First implementation, training of RPN separate from the rest.
- Now we can train jointly!

- Four losses:

1. RPN classification (object/non-object)
2. RPN regression (anchor $\rightarrow$ proposal)
3. Fast R-CNN classification (type of object)
4. Fast R-CNN regression (proposal $\rightarrow$ box)



Faster R-CNN

- 10x faster at test time wrt Fast R-CNN
- Trained end-to-end including feature extraction, region proposals, classifier and regressor
- More accurate, since proposals are learned. RPN is fully convolutional

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

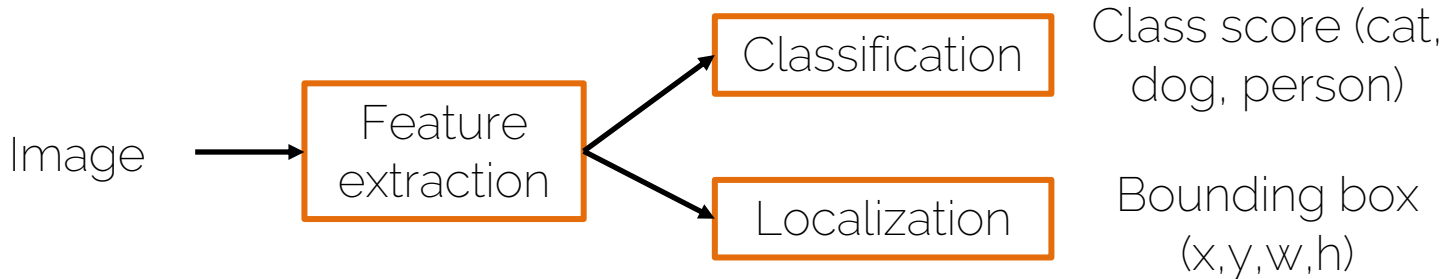
Related works

- Shrivastava, Gupta, Girshick. "Training region-based object detectors with online hard example mining". CVPR 2016.
- Dai, Li, He and Sun. "R-FCN: Object detection via region-based fully convolutional networks". 2016.
- Dai, Qi, Xiong, Li, Zhang, Hu and Wei. "Deformable convolutional networks". ICCV 2017.
- Lin, Dollar, Girshick, He, Hariharan and Belongie. "Feature Pyramid Networks for object detection". CVPR 2017.

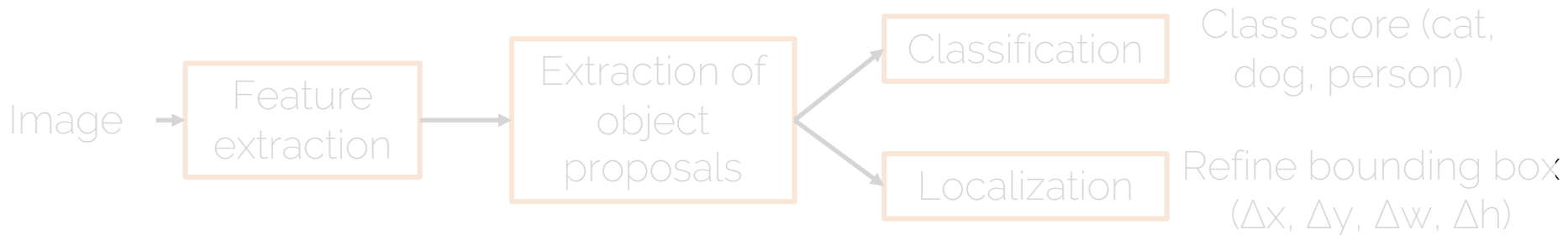
One-stage detectors

Types of object detectors

- One-stage detectors

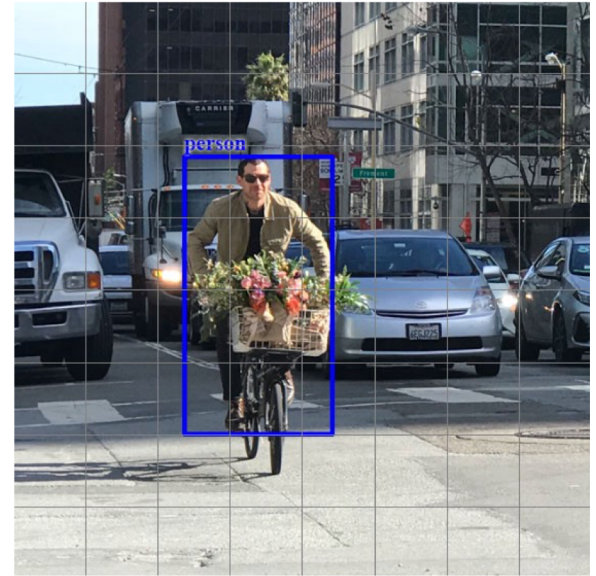


- Two-stage detectors



YOLO: You Only Look Once

- Recall sliding window object detection
- To make it efficient, we will “slide our window” only on certain locations of the image
- We divide our image in a grid.



Redmon et al, "You only look once: Unified real-time object detection", CVPR 2016.

YOLO: You Only Look Once

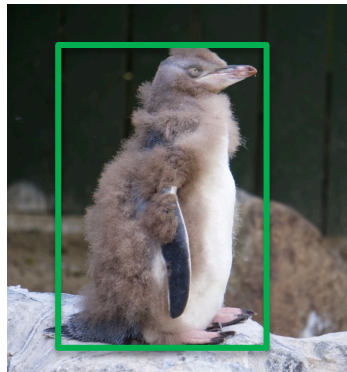
- We will place a box at the center of each cell in the grid, and this will be our initial box guess for that object



Redmon et al, "You only look once: Unified real-time object detection", CVPR 2016.

YOLO: You Only Look Once

- Direct regression from image to box coordinates



Convolutional neural network

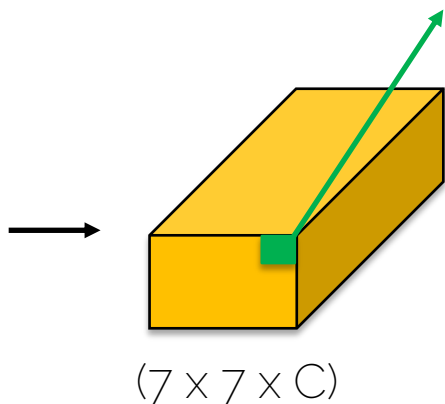
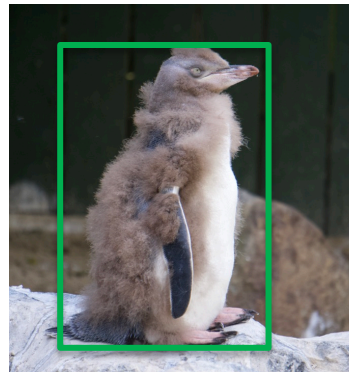


Detections

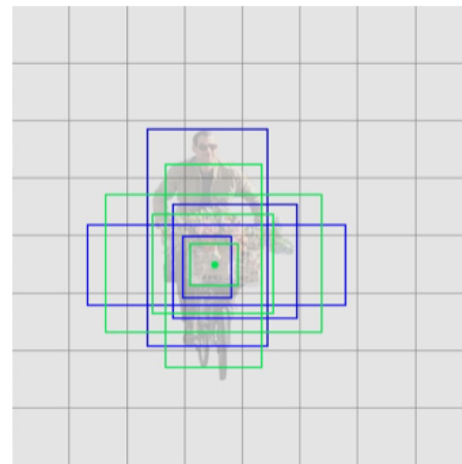
Redmon et al, "You only look once: Unified real-time object detection", CVPR 2016.

YOLO: You Only Look Once

- In YOLOv2 we use anchor boxes



For each grid location we predict n boxes

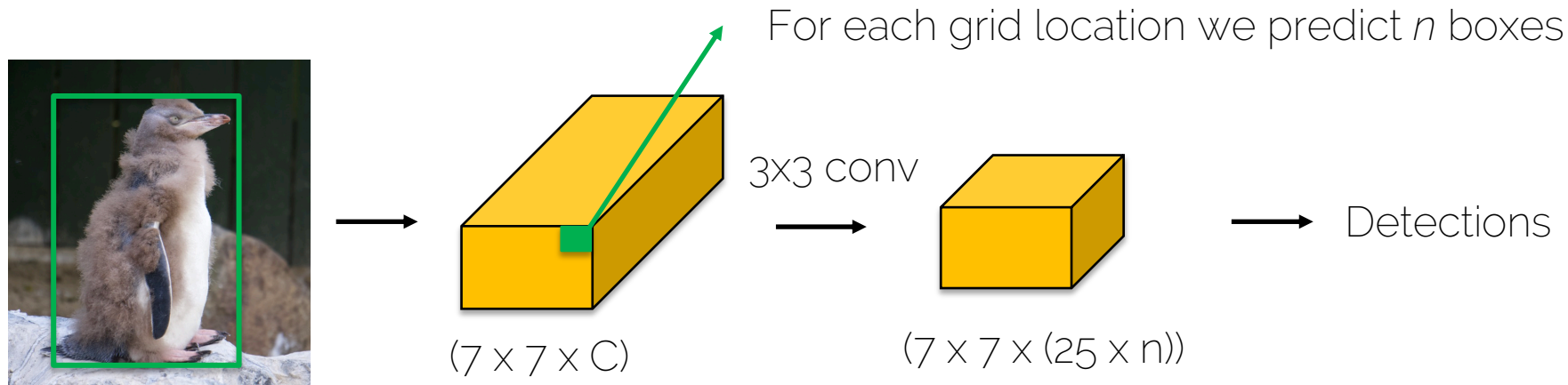


Green = anchors
Blue = predictions

Redmon and Farhadi, "YOLOg000: Better, faster, stronger", CVPR 2017.

YOLO: You Only Look Once

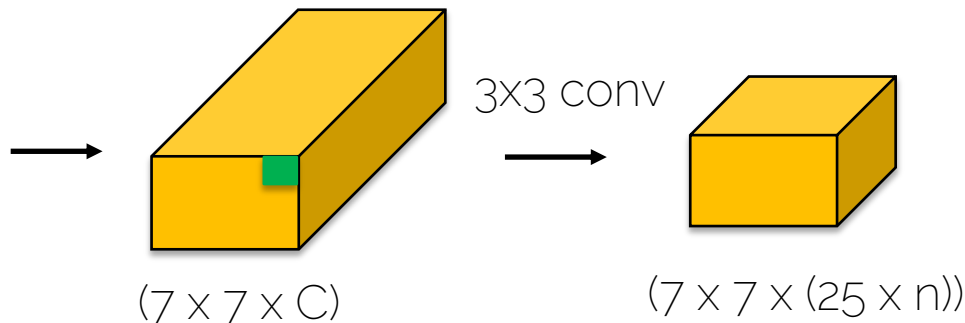
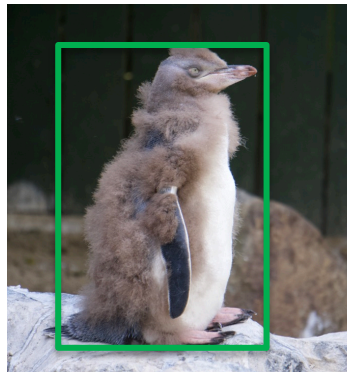
- In YOLOv2 we use anchor boxes



Similar to Faster R-CNN region proposal network

YOLO: You Only Look Once

- In YOLOv2 we use anchor boxes

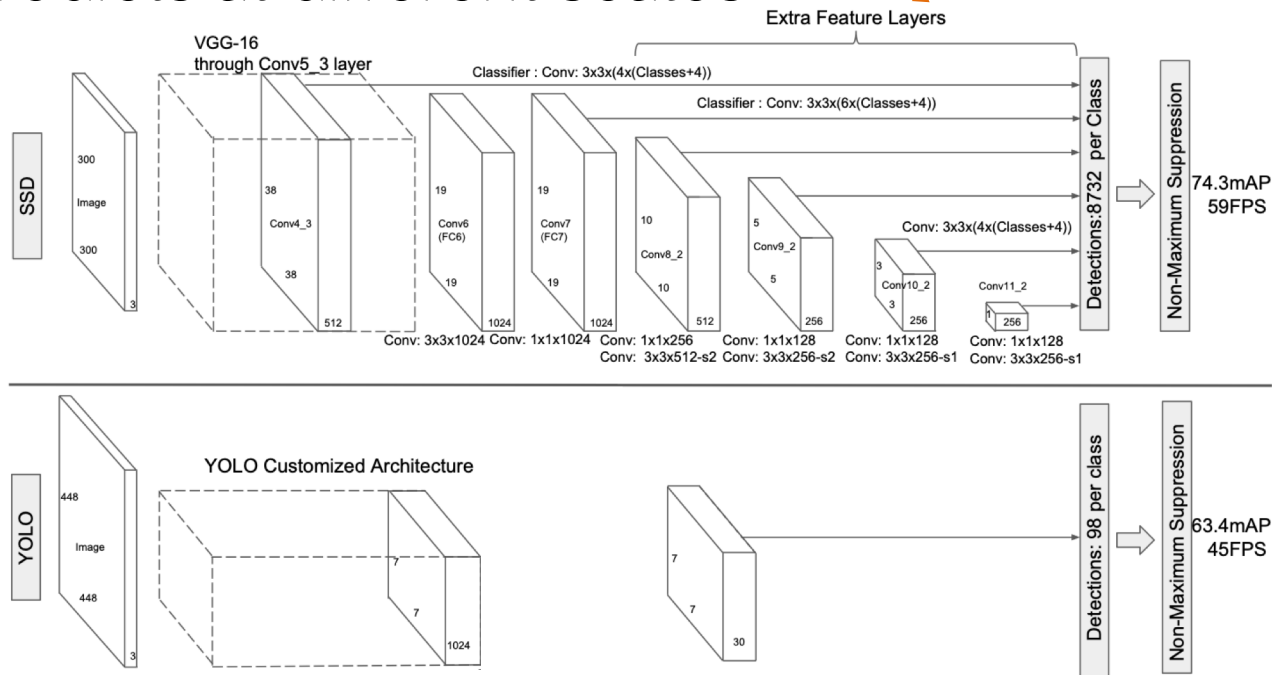


- 1 class prediction (20)
- Anchor relative box regression (4)
- Object/non-object (1)

Redmon and Farhadi, "YOLOg000: Better, faster, stronger", CVPR 2017.

SSD: Single Shot multibox Detector

- SSD predicts at different scales



Liu et al. "SSD: Single shot multibox detector". ECCV 2016

YOLO and SSD

- PROS:
 - Very fast
 - End-to-end trainable and fully convolutional
 - SSD detects more objects than YOLO
- CONS:
 - Performance is not as good as two-stage detectors
 - Difficulty with small objects

Problem with one-stage detectors?

- Two-stage detectors:
 - Classification only work on “interesting” foreground regions (proposals, ~1-2k). Most background examples are already filtered out.
 - Class balance between foreground and background objects is manageable.
 - Classifier can concentrate on analyzing proposals with rich information content

Problem with one-stage detectors?

- One-stage detectors:
 - Many locations need to be analyzed (100k) densely covering the image → foreground-background imbalance
 - Hard negative mining is useful, but not sufficient

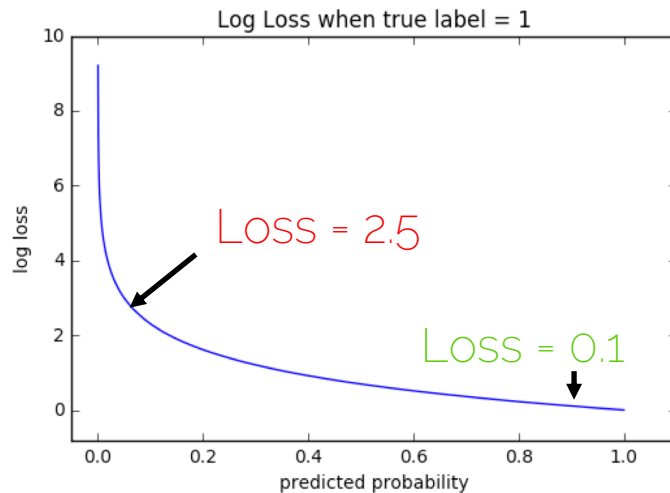


Many negative examples, no useful signal

Few positive examples, rich information for learning

RetinaNet

- Solution: change the loss function!
- Recall cross-entropy loss



Very hard
example, we
get loss 2.5

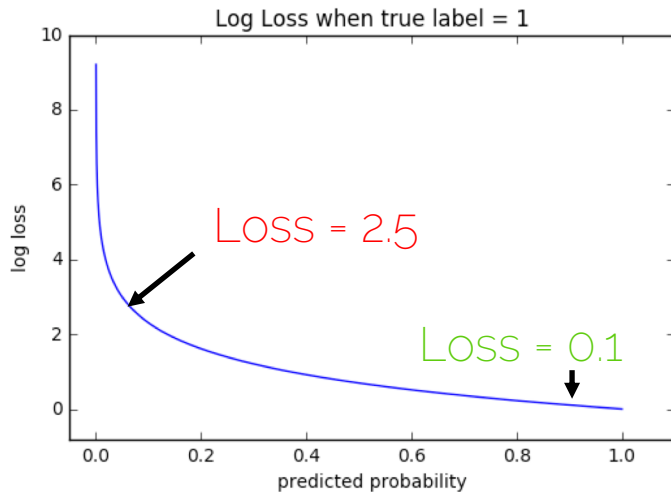
Well classified
(easy)
examples, we
get loss 0.1

RetinaNet

- Solution: change the loss function!
- Recall cross-entropy loss

100 hard
examples * 2.5
= 250

100000 easy
examples * 0.1
= 10000



Very hard
example, we
get loss 2.5

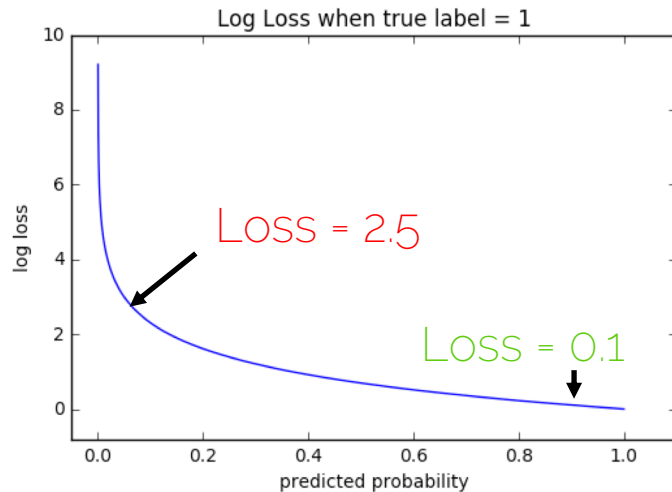
Well classified
(easy)
examples, we
get loss 0.1

RetinaNet

- Solution: change the loss function!
- Recall cross-entropy loss

100 hard
examples * 2.5
= 250

100000 easy
examples * 0.1
= 10000

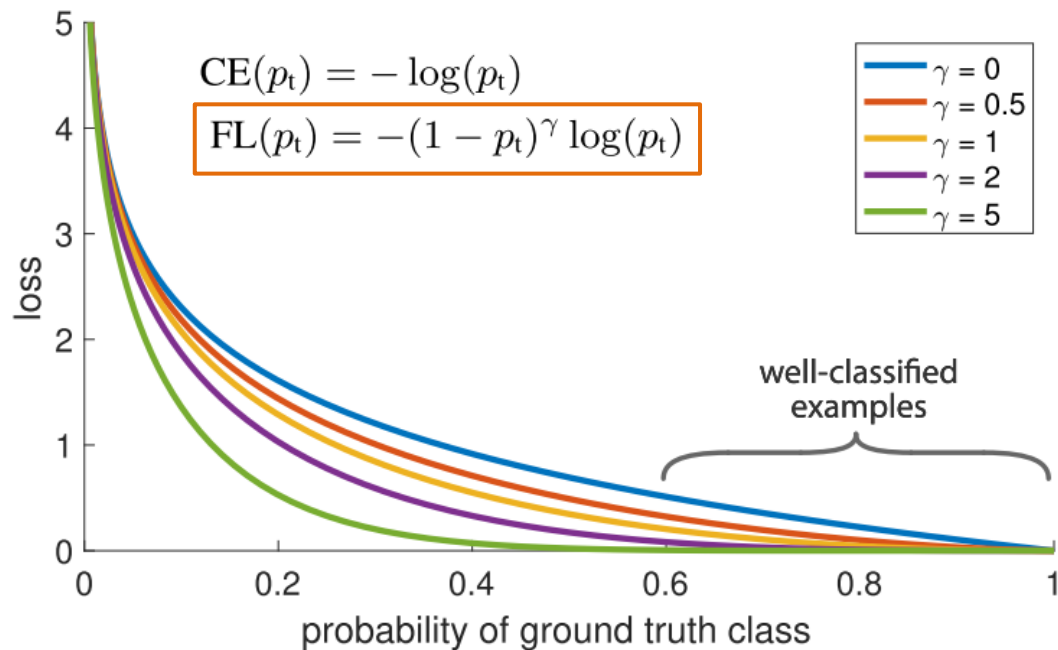


Very hard
example, we
get loss 2.5

Well classified
(easy)
examples, we
get loss 0.1

RetinaNet

- Proposed: Focal loss

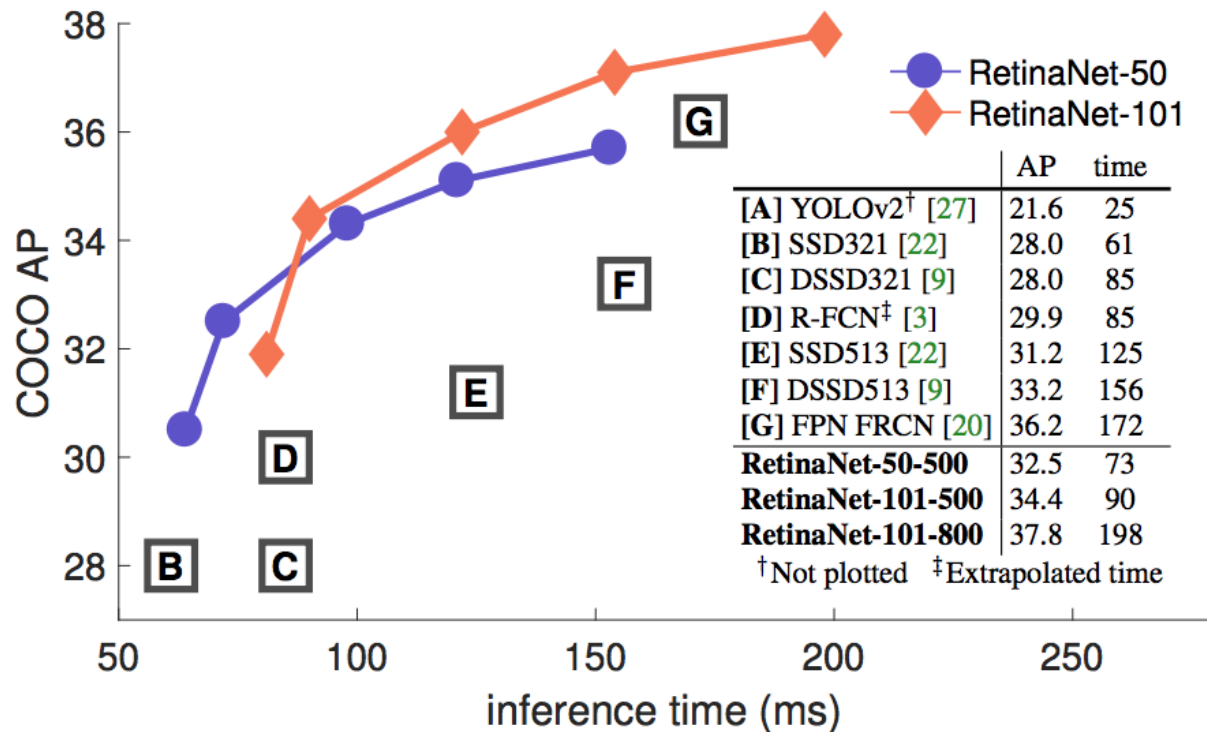


- When $\gamma = 0$ it is equivalent to the cross-entropy loss
- As γ goes towards 1, the easy examples are down-weighted.
- Example: $\gamma = 2$, if $p_t = 0.9$, FL is 100 lower than CE.

RetinaNet

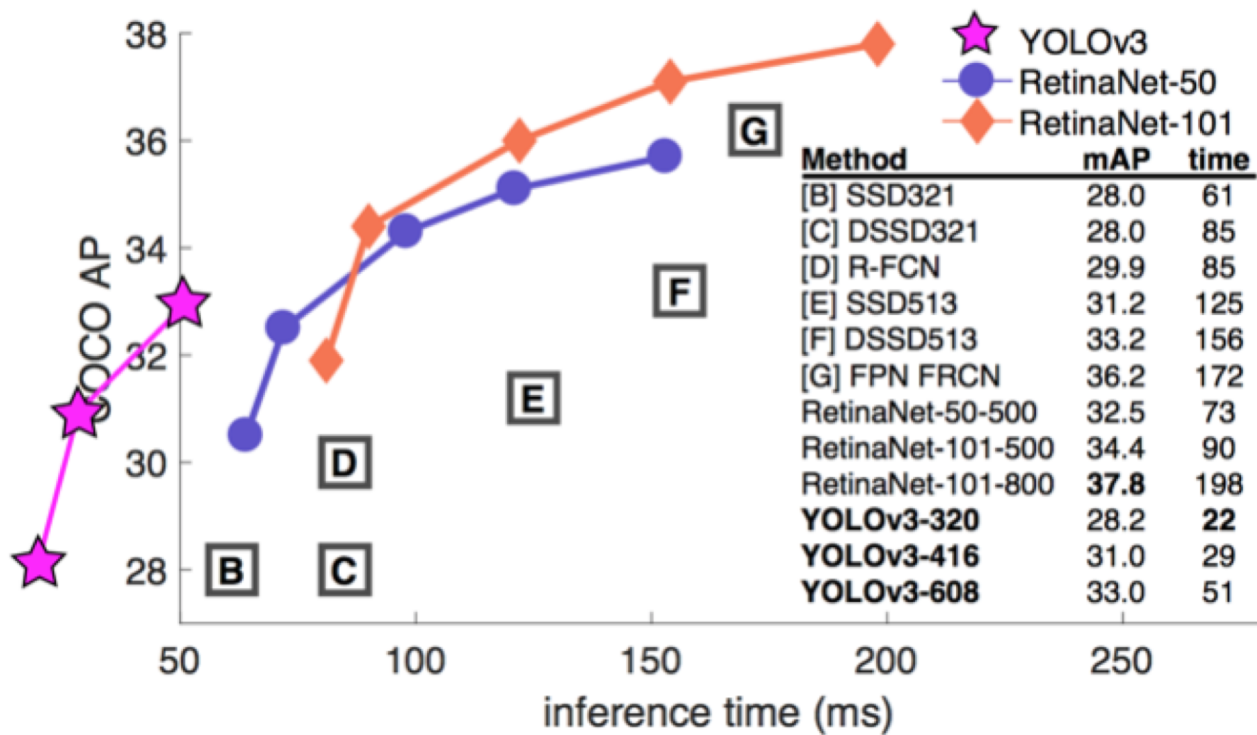
- Proposed: Focal loss
- Powerful feature extraction: ResNet
- Multi-scale prediction
- 9 anchors per level, each one with a classification and regression target

RetinaNet



TY Lin et al. "Focal Loss for Dense Object Detection ". ICCV 2017

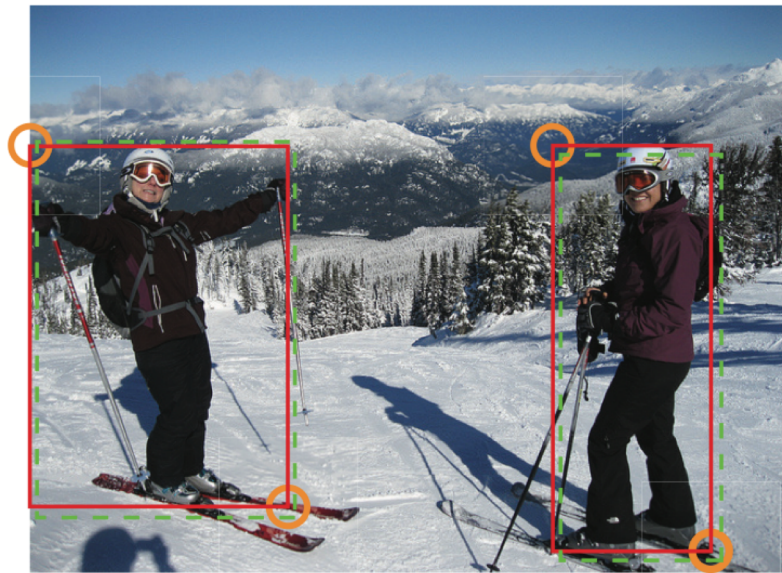
RetinaNet



One-stage (point-based) detectors

Getting rid of anchors?

- CornerNet: express bounding boxes with 2 points, the top-left and bottom-right corners.

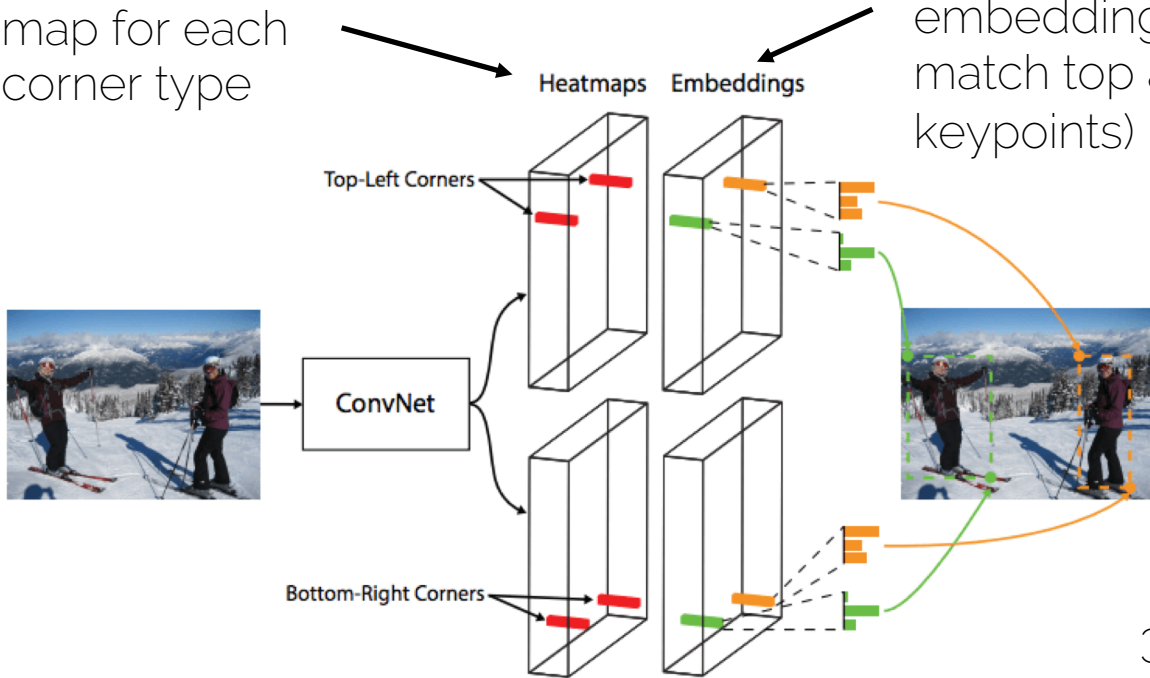


H. Law and J. Deng. „CornerNet: Detecting Objects as Paired Keypoints“. ECCV 2018

CornerNet

1. Probability map for each corner type

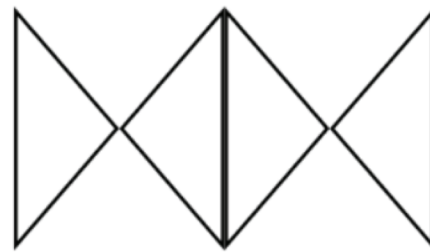
2. Box identification with an embedding (needed to match top and bottom keypoints)



H. Law and J. Deng. „CornerNet: Detecting Objects as Paired Keypoints“. ECCV 2018

CornerNet

- Hourglass network, originally published in:
 - A. Newell et al. "Stacked hourglass networks for human pose estimation". 2016



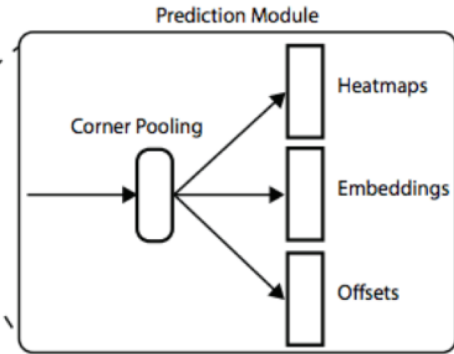
Hourglass Network

Top-left Corners

Prediction Module

Prediction Module

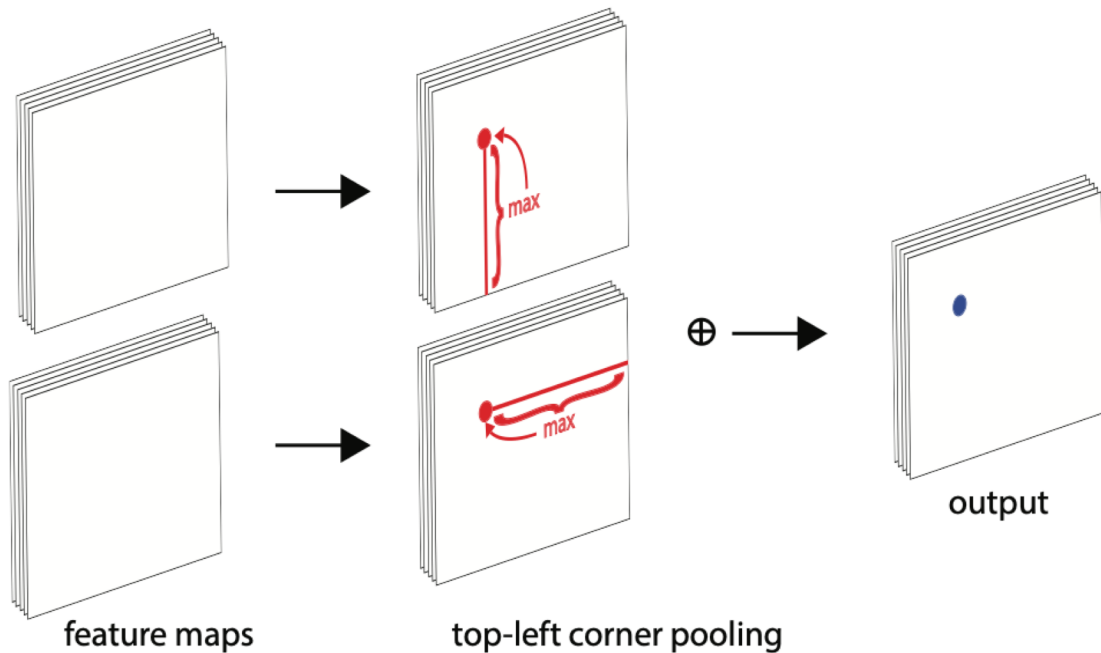
Bottom-right corners



We predict corners at a lower resolution and then regress an offset (bounding box correction as we have seen for all methods)

CornerNet

- Corner pooling



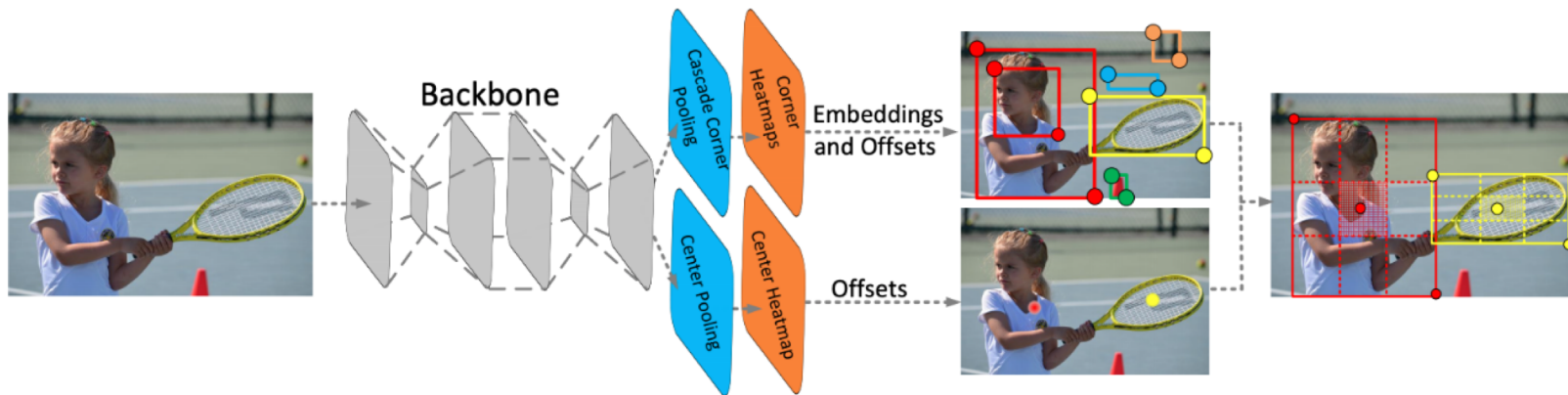
H. Law and J. Deng. „CornerNet: Detecting Objects as Paired Keypoints“. ECCV 2018

CornetNet

- What is the problem with CornetNet?
- Many incorrect bounding boxes (especially small) → too many False Positives
- Hypothesis: It is hard to infer the class of the box if the network is focused on the boundaries

CenterNet

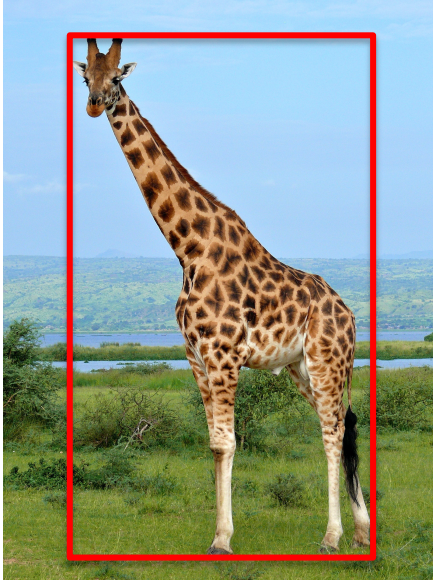
- Idea: focus on the *center* of the object to infer its class
- Use the corners as proposals, and the center to verify the class of the object and filter out outliers



K. Duan et al. „CenterNet: Keypoint Triplets for Object Detection“. arXiv 2018

ExtremeNet

- Bounding box corner representation is not ideal, why?



X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019

ExtremeNet

- Bounding box corner representation is not ideal, why?



● Corner lies on the object

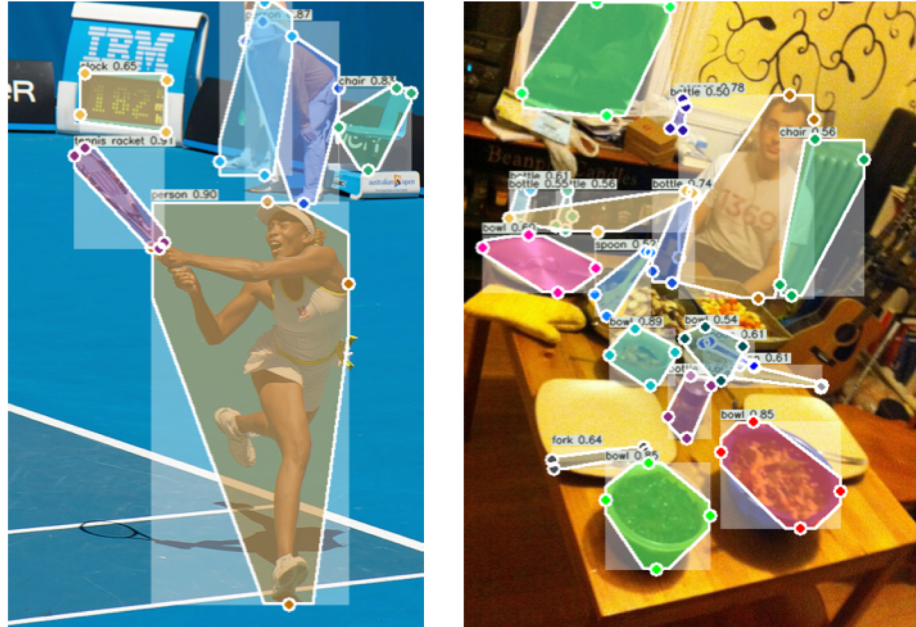
● Corner does not lie on the object

Hard for a CNN to
predict ● as a corner

X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019

ExtremeNet

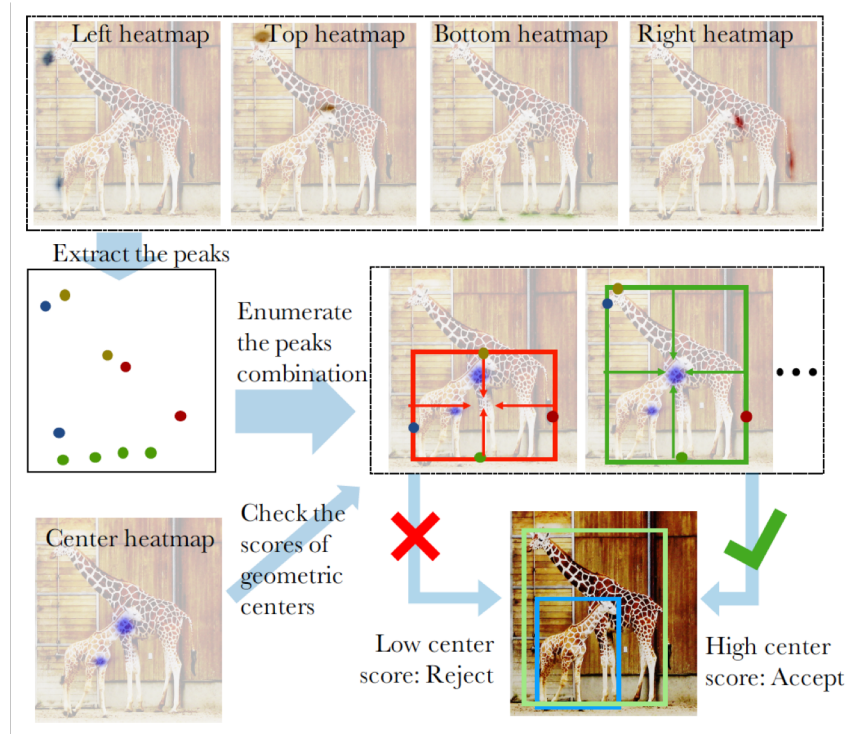
- Represent objects by their extreme points



X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019

ExtremeNet

- No need to predict embeddings for the box computation



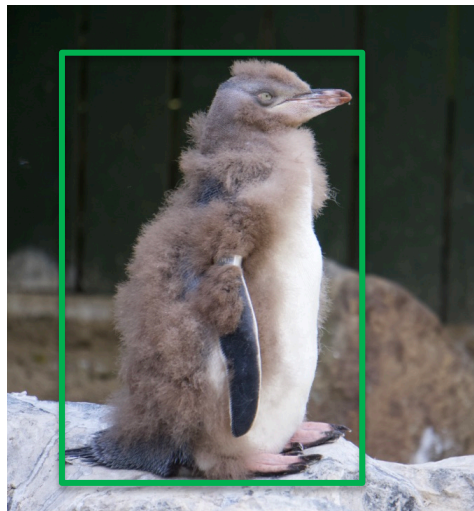
X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019

Extreme points

- Extreme points are used commonly for annotation
 - D. P. Papadopoulos et al. „Extreme clicking for efficient object annotation“. ICCV 2017
 - K. Maninis et al. „Deep extreme cut: From extreme points to object segmentation“. CVPR 2018

Detection evaluation

Detection evaluation



TP = True positives. FP = False positives. FN = False negatives

Detection evaluation

- Precision: how accurate your predictions are.

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

TP → Good boxes detected
TP + FP → All boxes that you detected

- Recall: how good you are at finding all positives

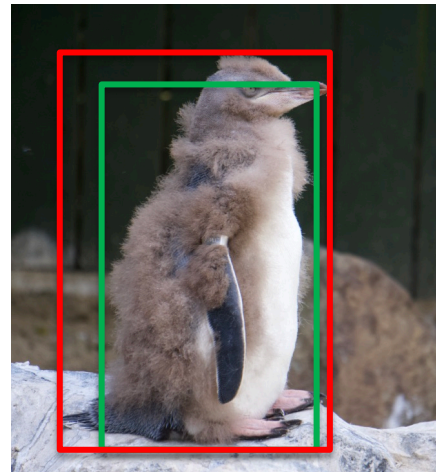
$$Recall = \frac{TP}{TP + FN}$$

TP → Good boxes detected
TP + FN → All ground truth boxes

TP = True positives. FP = False positives. FN = False negatives

Detection evaluation

- What is a positive?
- We use the Intersection over Union (IoU) or Jaccard Index
- If $\text{IoU} > 0.5 \rightarrow$ positive match
- Depending on the dataset it is 0.5 or 0.7



TP = True positives. FP = False positives. FN = False negatives

Detection evaluation: AP

- Mean Average Precision (mAP)
 - For each image and class independently, rank the predicted boxes by confidence score. Assign the boxes to the corresponding ground truth if $\text{IoU} > 0.5$.
 - Each ground truth box can only be matched to one predicted box.
 - For each class independently, compute Average Precision
 - Mean over all classes to obtain mAP



Average Precision

- Rank predictions by confidence

Rank	True/false	Precision	Recall
1	T		
2	T		
3	F		
4	T		
5	F		
6	F		
7	T		
8	F		
9	T		
10	T		
11	F		
12	F		

My method predicts 12 boxes, 6 of them are true boxes present in the image

Average Precision

- Rank predictions by confidence

Rank	True/false	Precision	Recall
1	T	1.0	0.17
2	T		
3	F		
4	T		
5	F		
6	F		
7	T		
8	F		
9	T		
10	T		
11	F		
12	F		

Average Precision

- Rank predictions by confidence

Rank	True/false	Precision	Recall
1	T	1.0	0.17
2	T	1.0	0.33
3	F		
4	T		
5	F		
6	F		
7	T		
8	F		
9	T		
10	T		
11	F		
12	F		

Average Precision

- Rank predictions by confidence

Rank	True/false	Precision	Recall
1	T	1.0	0.17
2	T	1.0	0.33
3	F	0.67	0.33
4	T		
5	F		
6	F		
7	T		
8	F		
9	T		
10	T		
11	F		
12	F		

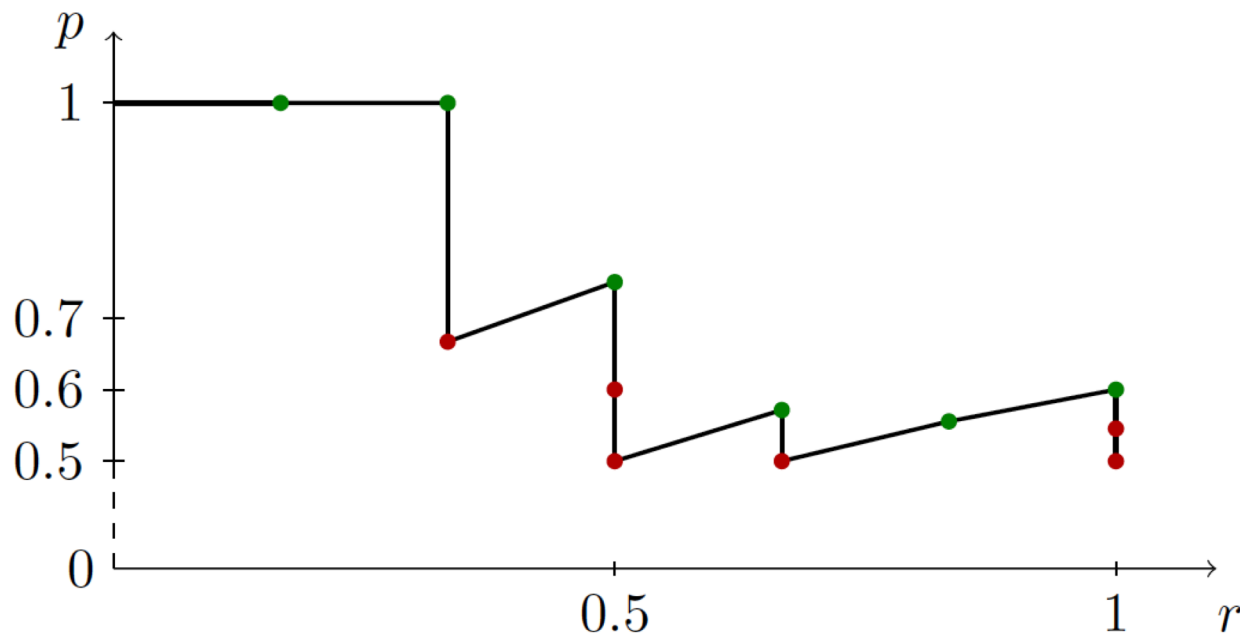
Average Precision

- Rank predictions by confidence

Rank	True/false	Precision	Recall
1	T	1.0	0.17
2	T	1.0	0.33
3	F	0.67	0.33
4	T	0.75	0.5
5	F	0.6	0.5
6	F	0.5	0.5
7	T	0.57	0.67
8	F	0.5	0.67
9	T	0.56	0.83
10	T	0.6	1.0
11	F	0.55	1.0
12	F	0.5	1.0

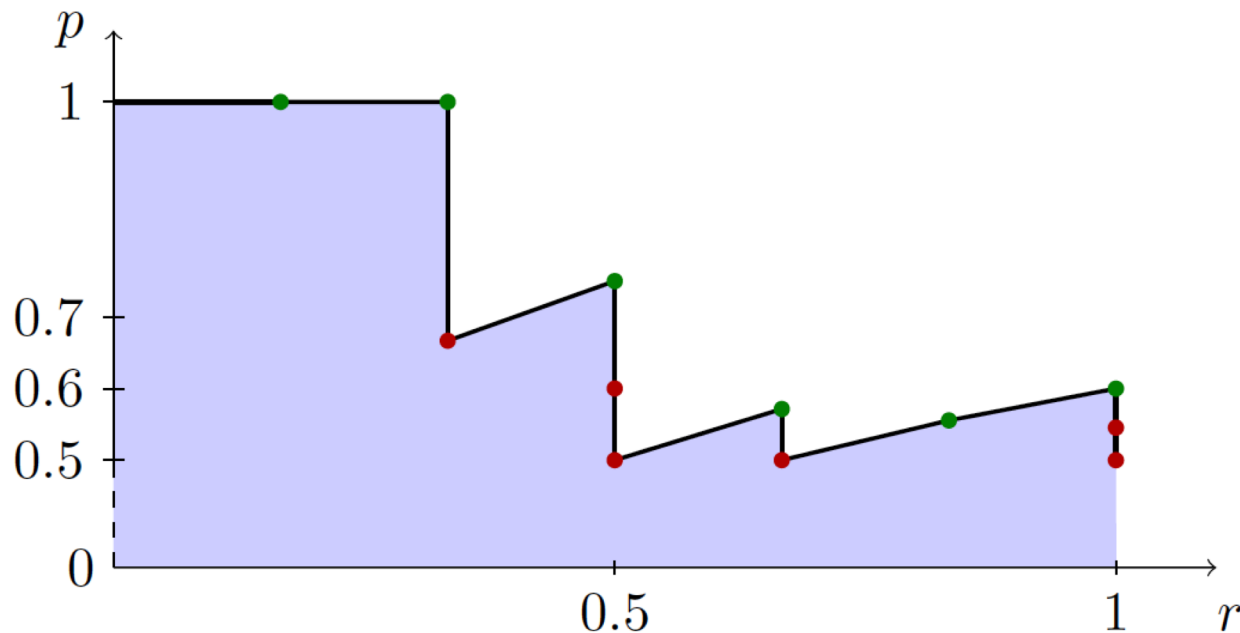
Average Precision

- Plot Precision vs Recall



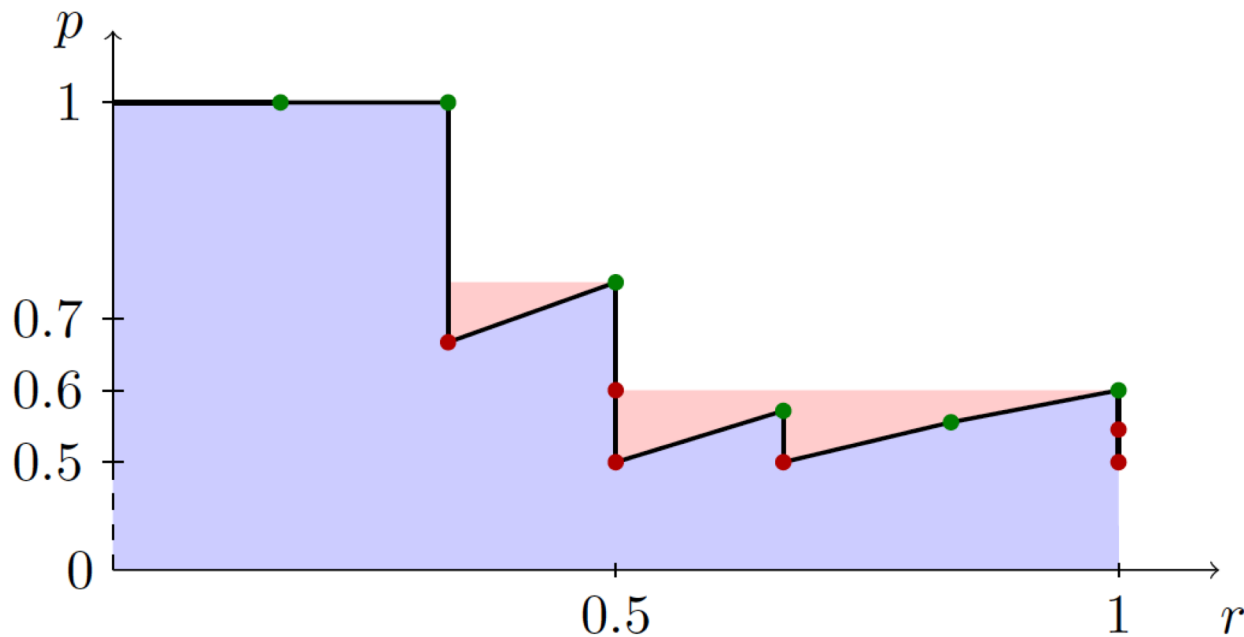
Average Precision

- Average precision = area under curve



Average Precision

- Average precision = area under curve (filled)



Interesting read

- Overview paper that analyzes backbone architecture selection, number of proposals used, etc. to see which has an effect on object detection accuracy.
- Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors". CVPR 2017

CV3DST competition

- Dates:
 - 26.11.19: Competition starts! Starting code and training set are released.
 - Before 25.12.19: Test set is released.
 - 02.02.20 (midnight): Competition closes
 - 03.02.20 (midnight): Abstract and code submission deadline
 - 04.02.20: Presenters are announced
 - 07.02.20: Presentation of selected methods

CV3DST competition

- Rules:
 - No off-the-shelf detectors or trackers can be used.
 - You can use off-the-shelf methods that compute additional features that you might use, e.g., optical flow. Nonetheless, this cannot be the only contribution of your method.
 - This is an individual effort, but sharing ideas between participants in the class is allowed, as long as you come up with different methods and implementations to solve the MOT problem.
 - We will not provide help to solve coding issues nor to decide what methods to use to improve your tracker. This is a real competition, so you are on your own!

CV3DST competition

- I get the bonus if....
 - You reach a minimum MOTA value (will be announced together with the test set).
 - You respect the rules of the challenge (this will be judged at our discretion. → make your own code, try to come up with ideas, show initiative)
- Presentations on February 7th:
 - Presence mandatory
 - GOAL: find out what has worked best/worst to solve the MOT problem
 - It is going to be an active discussion!

Next lectures

- 29.11: Detection 2
- 06.12: Single/Multiple object tracking
- 13.12: Multiple object tracking
- 20.12: Trajectory prediction
- Christmas Break
- 10.01: Video object segmentation
- 17.01: Semantic/Instance Segmentation
- 24.01: : Semantic/Instance Segmentation
- 31.01: Going towards 3D tracking and segmentation
- 07.02: Project Presentations