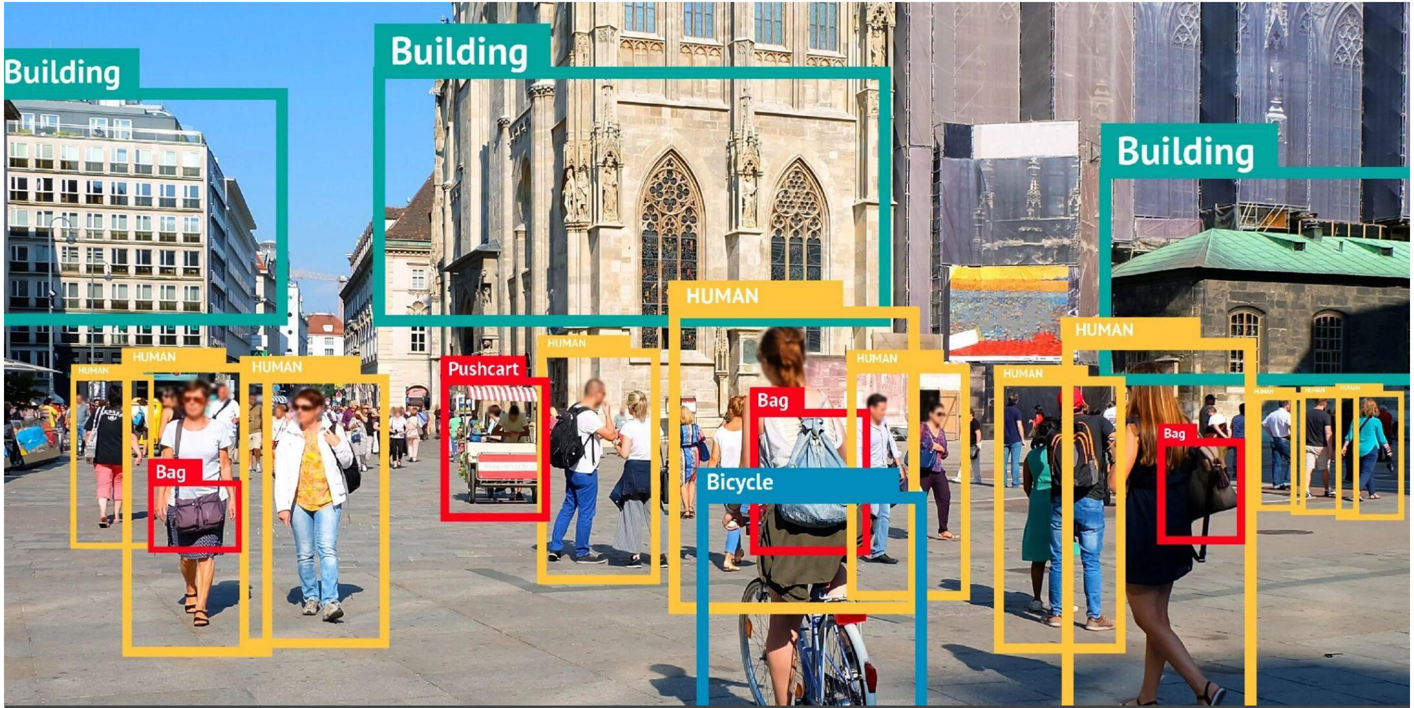
Object detection

Valeriya Strizhkova
05.10.21

Overview

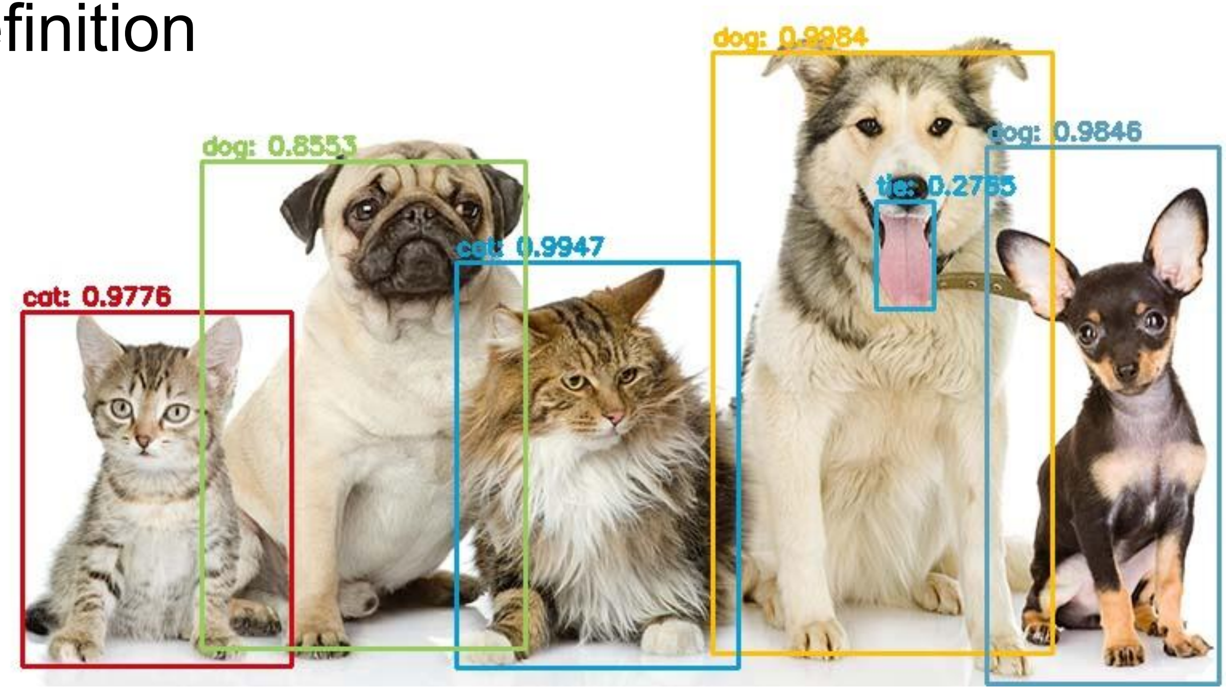
- Introduction
- Performance evaluation
- Object detectors
    - R-CNN
    - Fast R-CNN
    - Faster R-CNN

Object detection



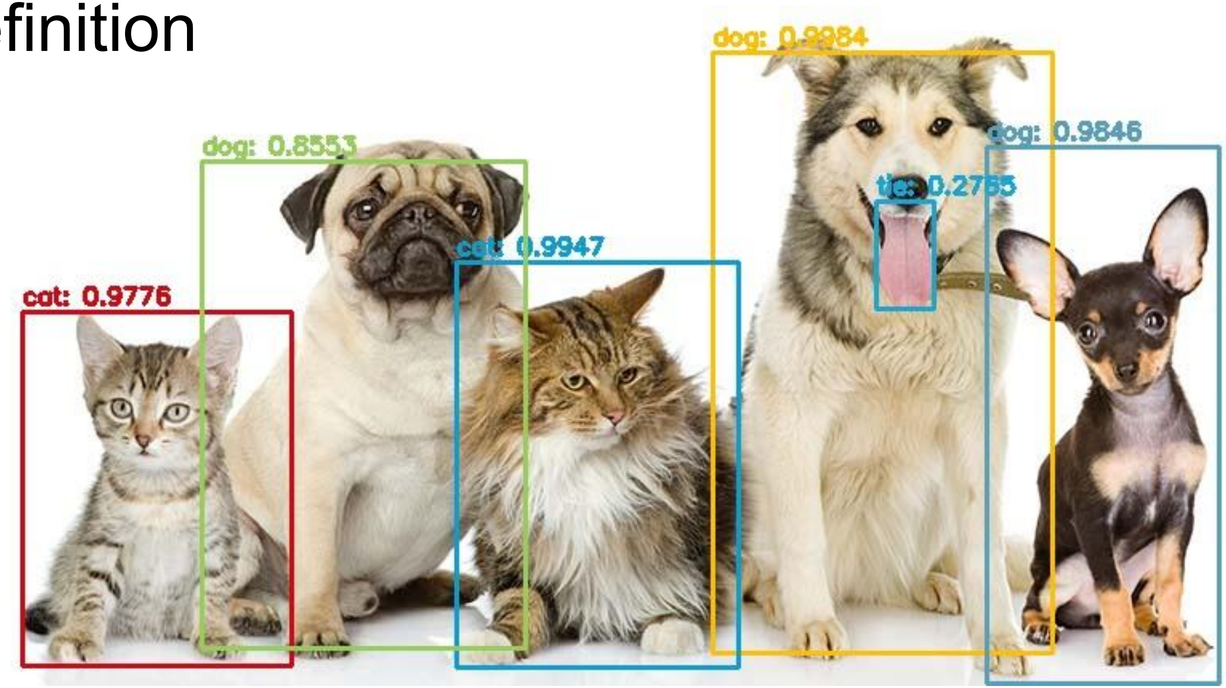
Goal: Localize (Bounding Box) and classify all objects in the image

Task definition



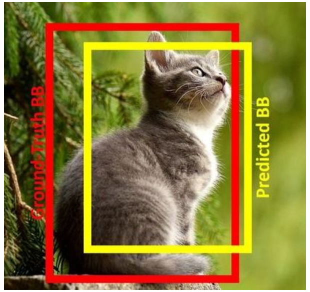
- **Input**: RGB image
- **Output**: Set of bounding boxes with category label and confidence

Task definition

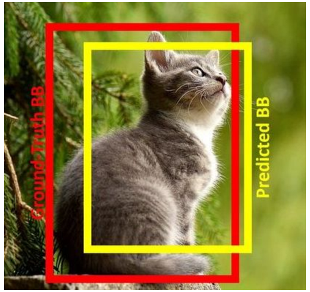


- **Input**: RGB image
- **Output**: Set of bounding boxes with category label and confidence
- The number of objects is not known

Performance evaluation



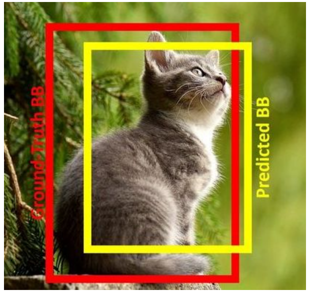
Performance evaluation



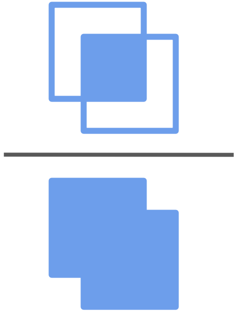
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} =$$

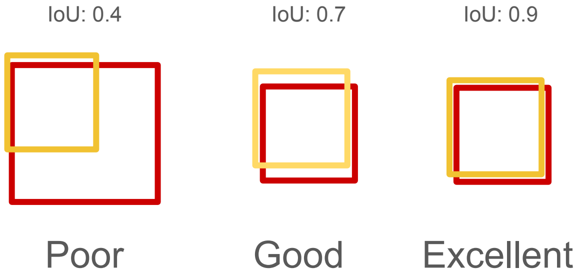
predicted bbox vs true bbox

Performance evaluation



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





predicted bbox vs true bbox

Performance evaluation

		Real Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

Performance evaluation

		Real Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

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

Performance evaluation

Real Label

Predicted Label		Positive	Negative
	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

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

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

Performance evaluation

		Real Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

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

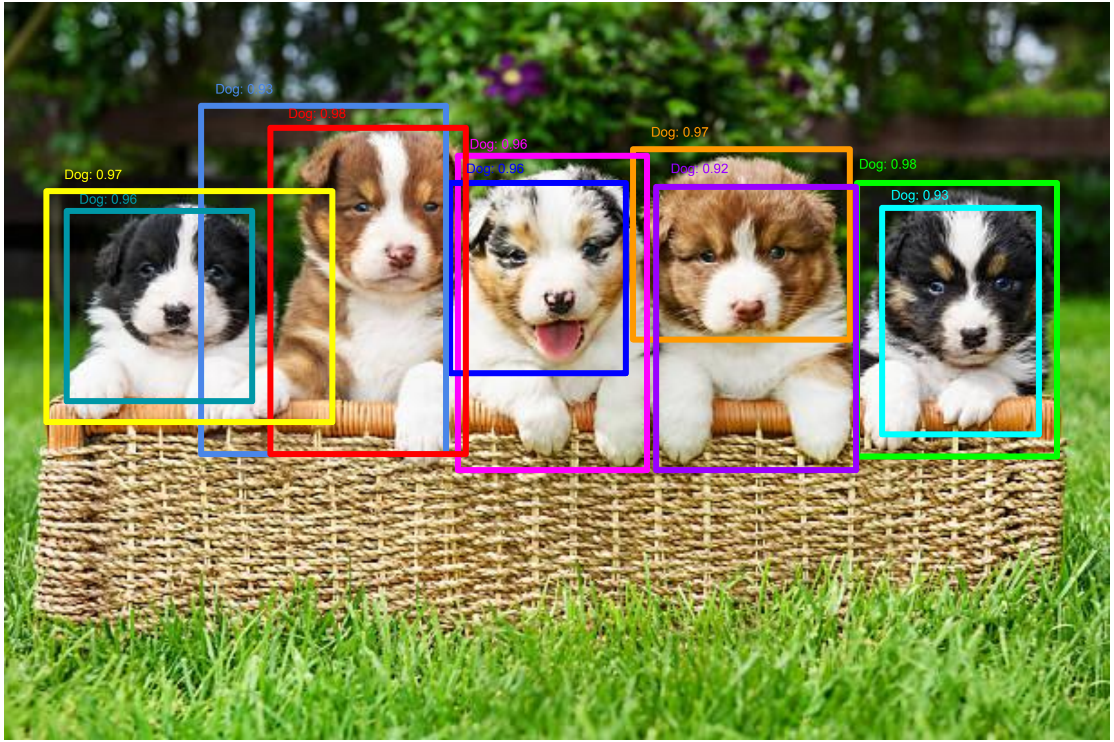
$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{total positive results}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{total disease cases}}$$

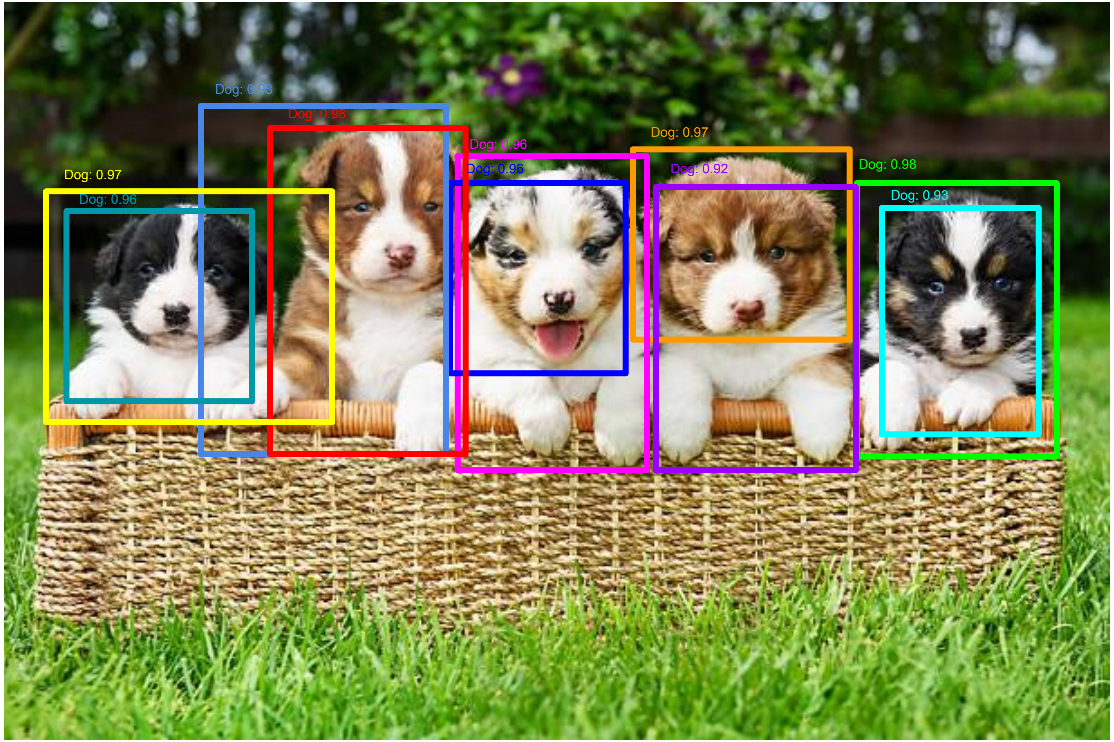
Performance evaluation: Average Precision

Performance evaluation: Average Precision



Performance evaluation: Average Precision

Rank	Correct?
1	True
2	True
3	False
4	False
5	False
6	True
7	True
8	False
9	False
10	True



Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} = \frac{2}{3} = 0.67$$

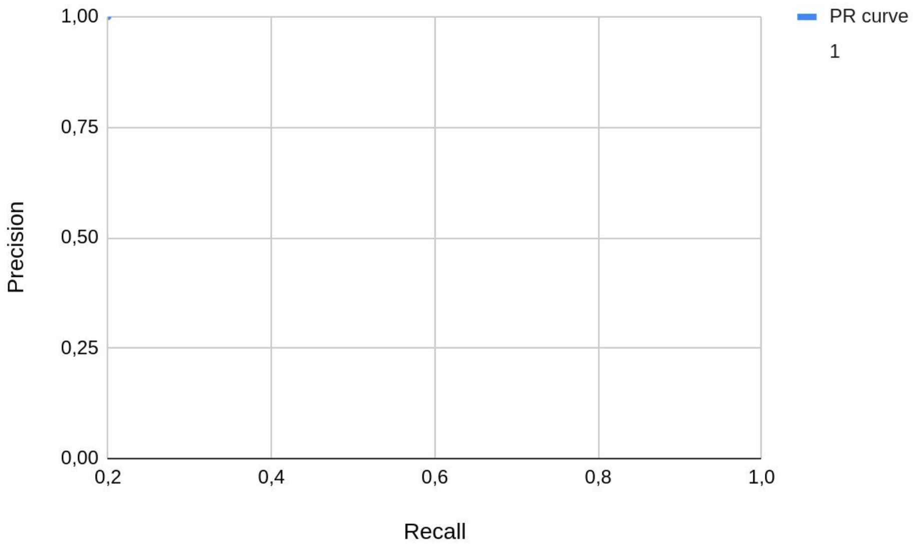
$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} = \frac{2}{5} = 0.4$$

Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑

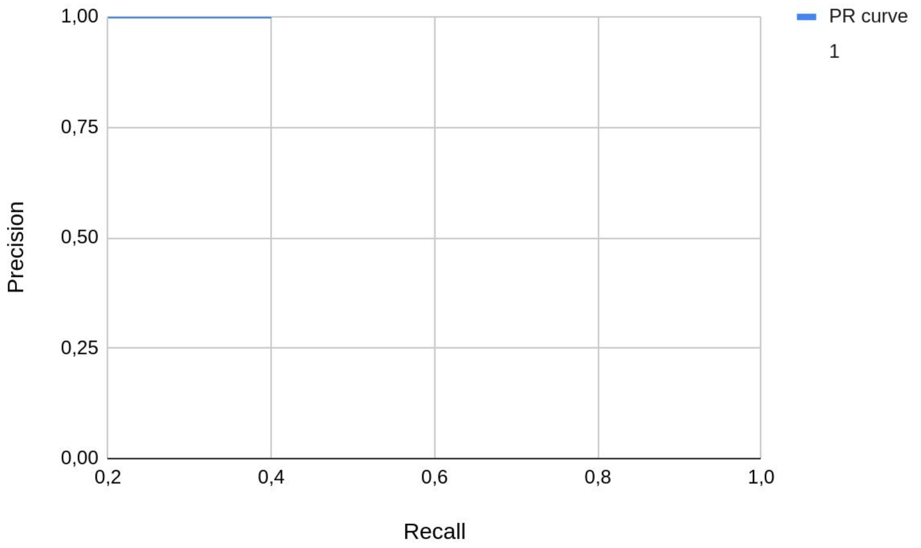
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	**True**	**1.0**	**0.2**
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



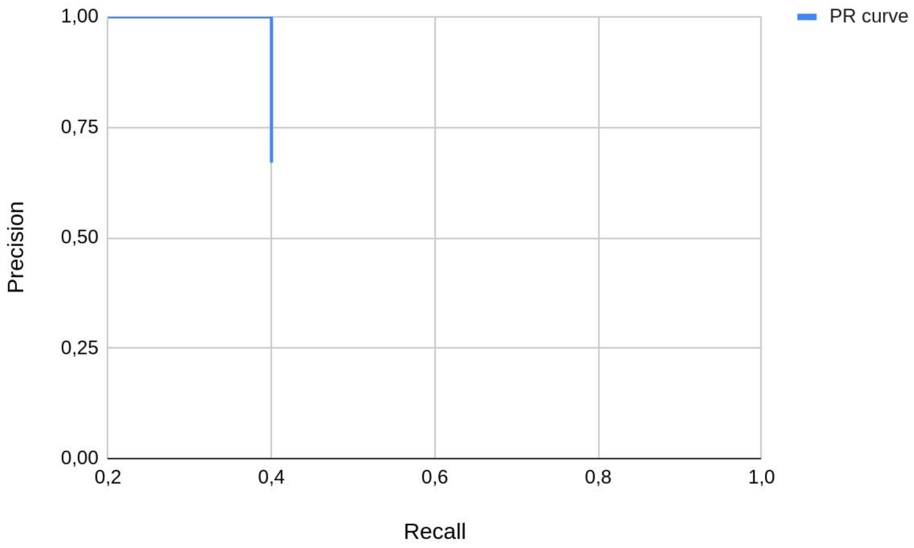
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	**True**	**1.0** —	**0.4** ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



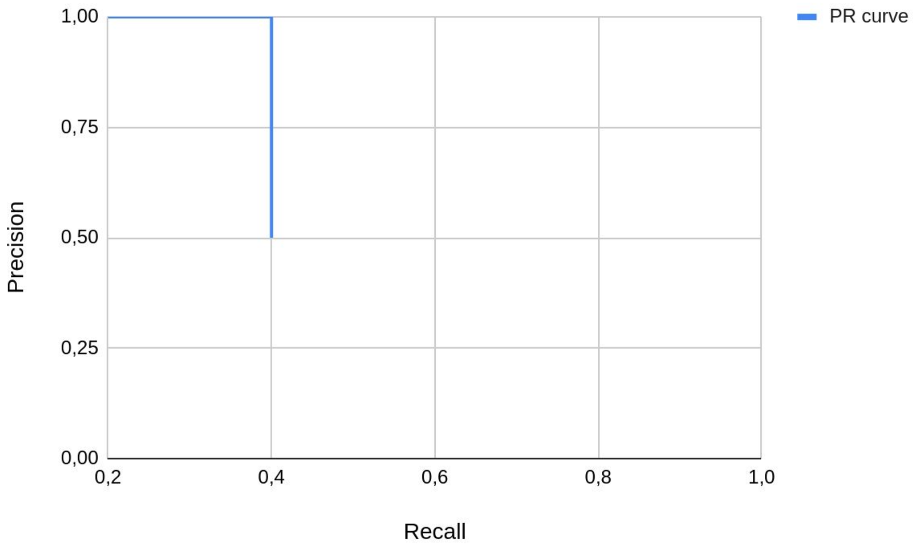
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	**False**	**0.67** ↓	**0.4** —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



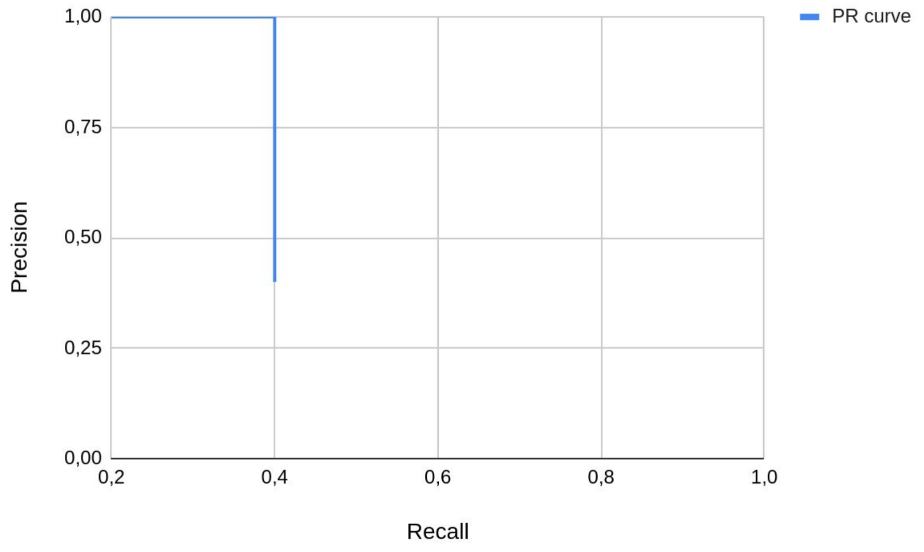
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	**False**	**0.5** ↓	**0.4** —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



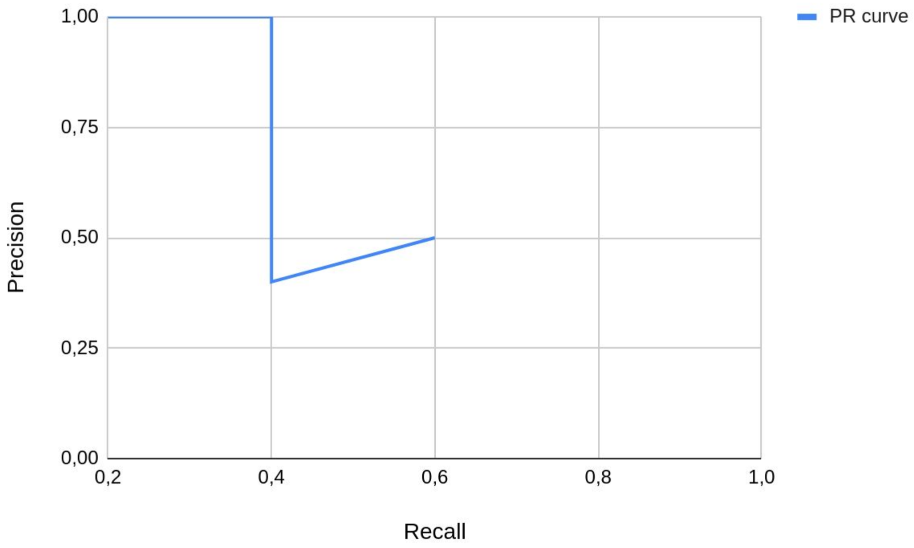
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	**False**	**0.4** ↓	**0.4** —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



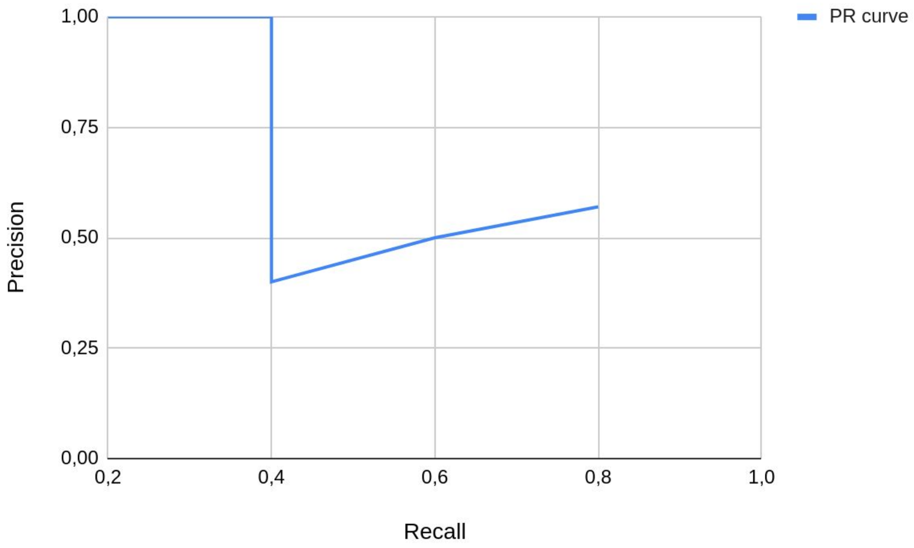
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	**True**	**0.5** ↑	**0.6** ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



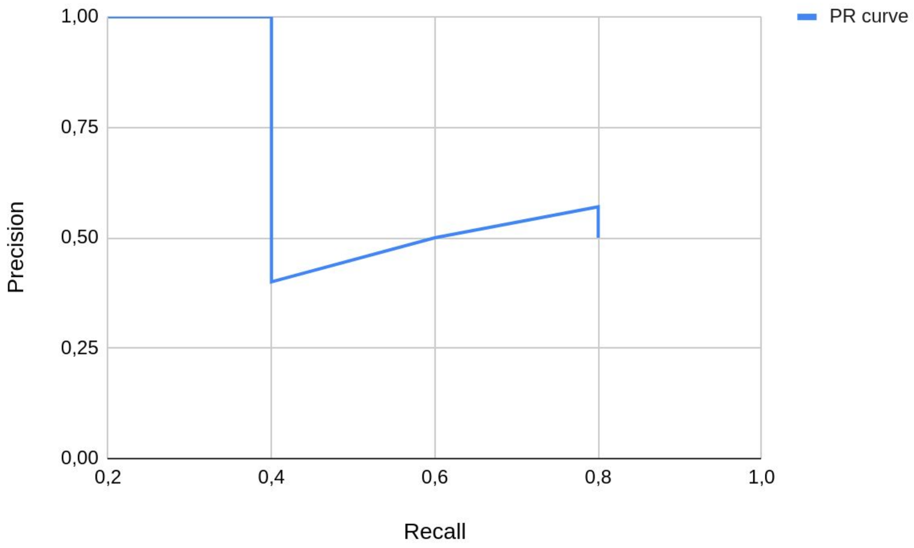
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	**True**	**0.57** ↑	**0.8** ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



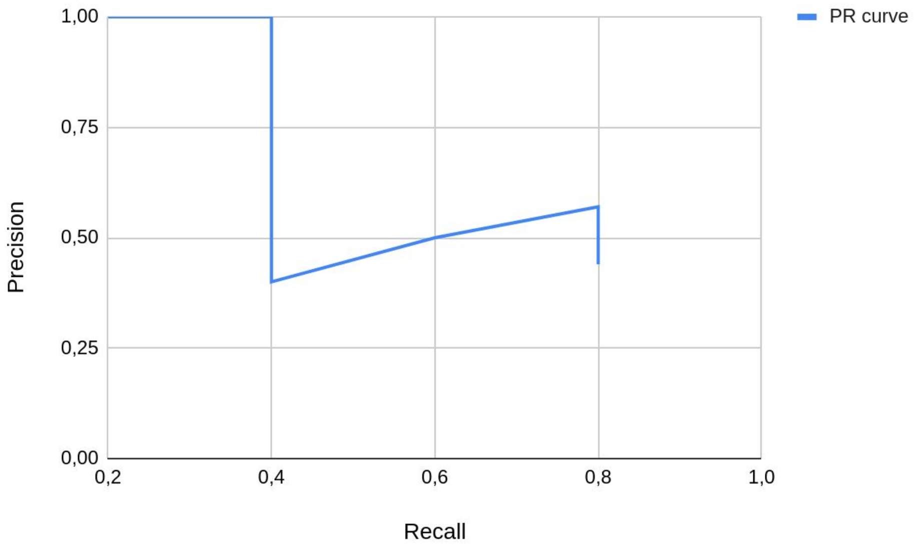
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	**False**	**0.5** ↓	**0.8** —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



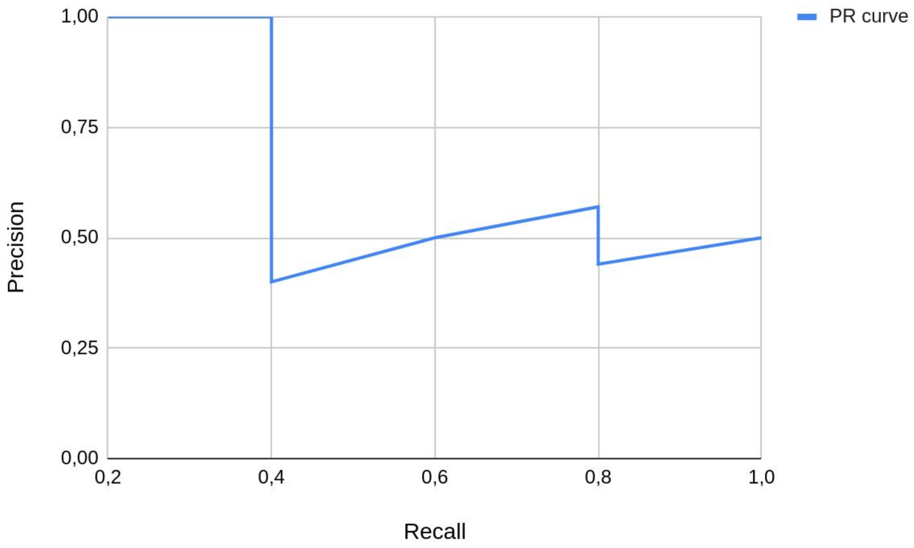
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	**False**	**0.44** ↓	**0.8** —
10	True	0.5 ↑	1.0 ↑



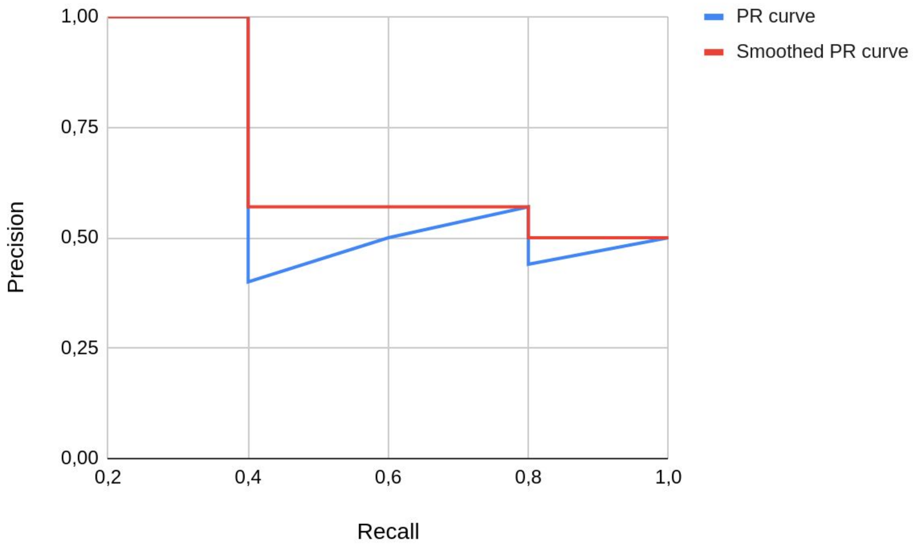
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	**True**	**0.5** ↑	**1.0** ↑



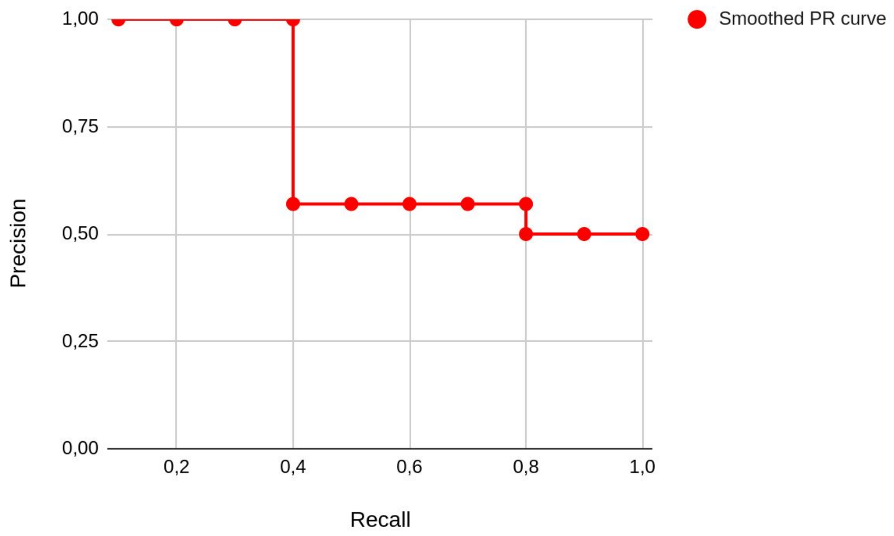
Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



Performance evaluation: Average Precision

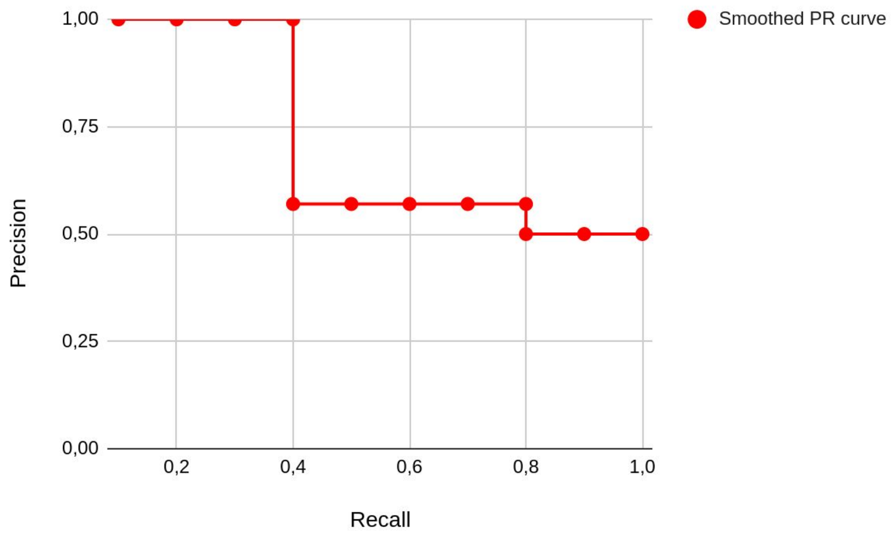
Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑



$$AP = \frac{1}{11} \times \sum_{r \in 0.0, 0.1, \ldots, 1.0} p(r) = \frac{1}{11} \times (p(0) + p(0.1) + \ldots + p(1.0))$$

Performance evaluation: Average Precision

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0 —	0.4 ↑
3	False	0.67 ↓	0.4 —
4	False	0.5 ↓	0.4 —
5	False	0.4 ↓	0.4 —
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑
8	False	0.5 ↓	0.8 —
9	False	0.44 ↓	0.8 —
10	True	0.5 ↑	1.0 ↑

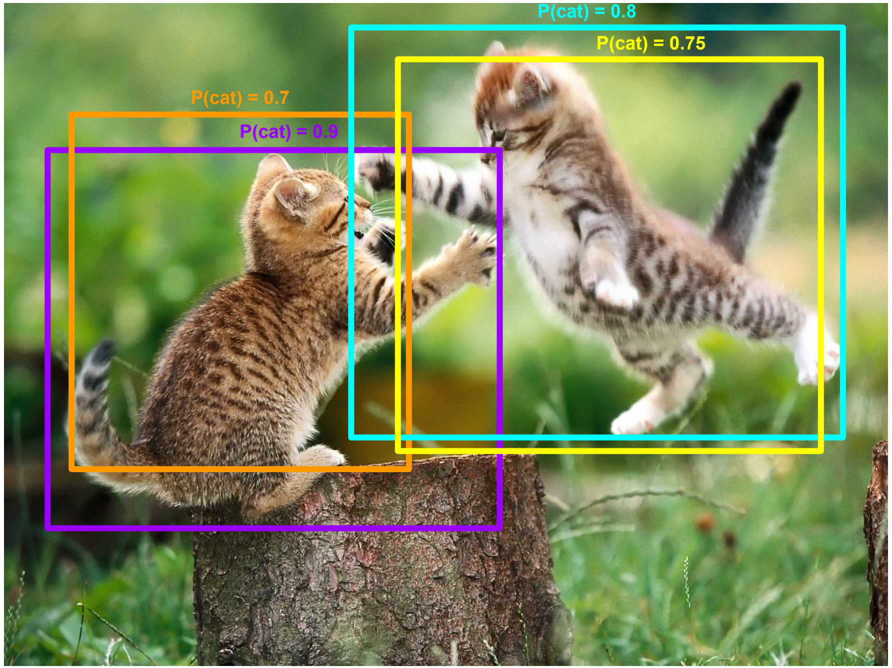


$$AP = \frac{1}{11} \times \sum_{r \in 0.0, 0.1, \dots, 1.0} p(r) = \frac{1}{11} \times (p(0) + p(0.1) + \dots + p(1.0))$$

$$AP = \frac{1}{11} \times (5 \times 1.0 + 4 \times 0.57 + 2 \times 0.5) = 8.28$$

Overlapping Boxes: Non-Max Suppression (NMS)

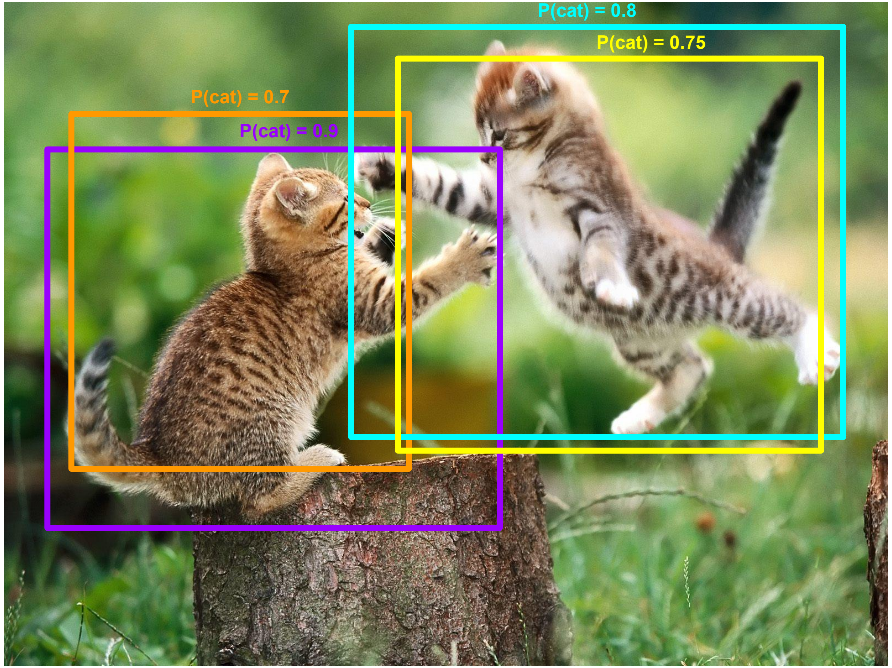
Problem: Object detectors often output many overlapping detections.



Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections.

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

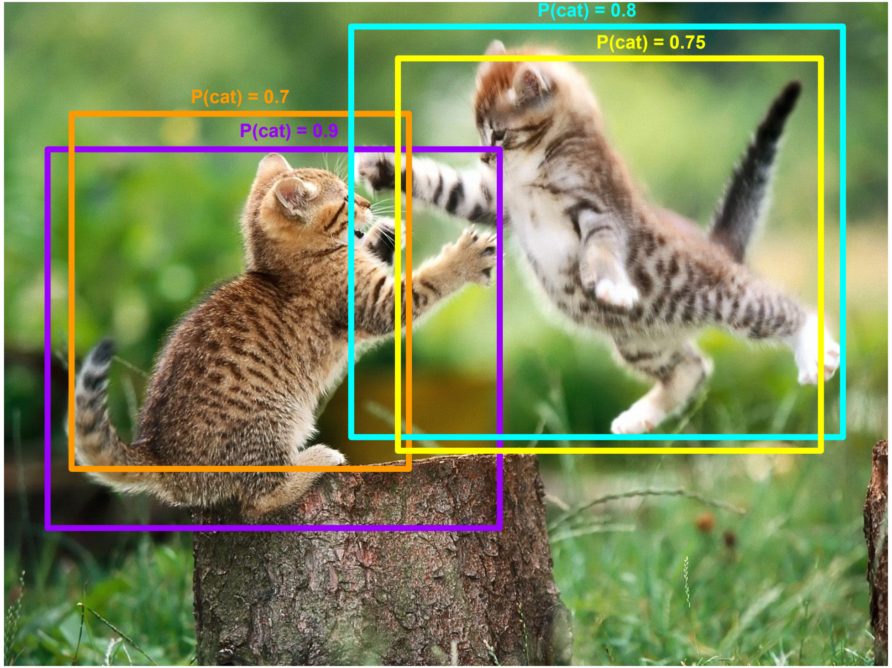


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections.

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



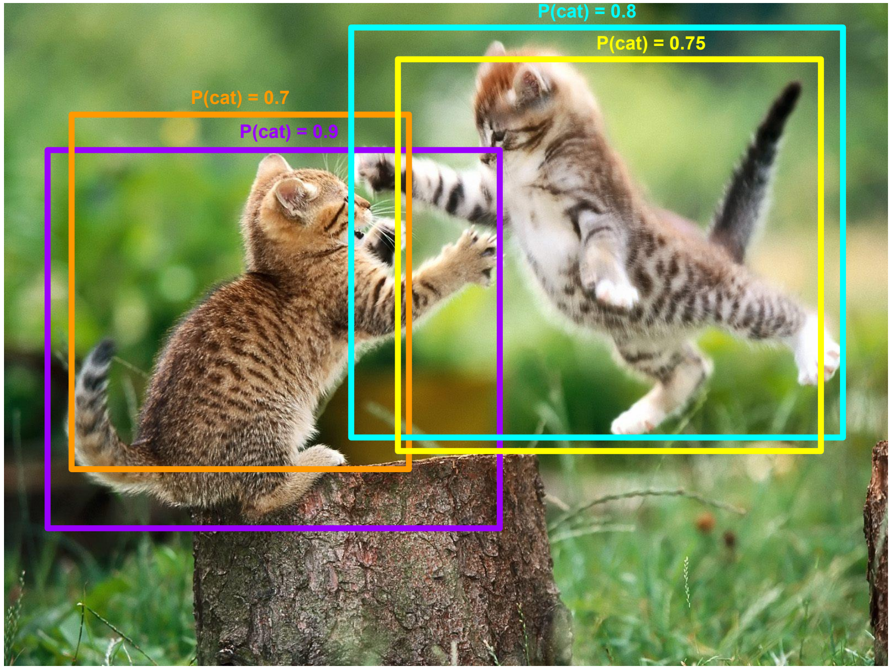
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections.

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.8**
IoU(■, ■) = 0.08
IoU(■, ■) = 0.01



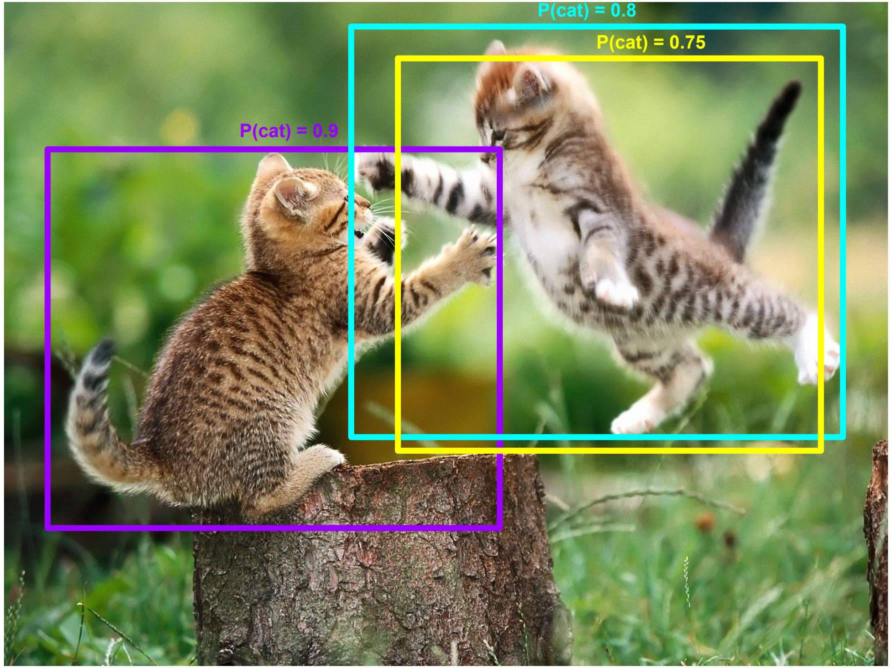
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections.

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(,) = **0.95**



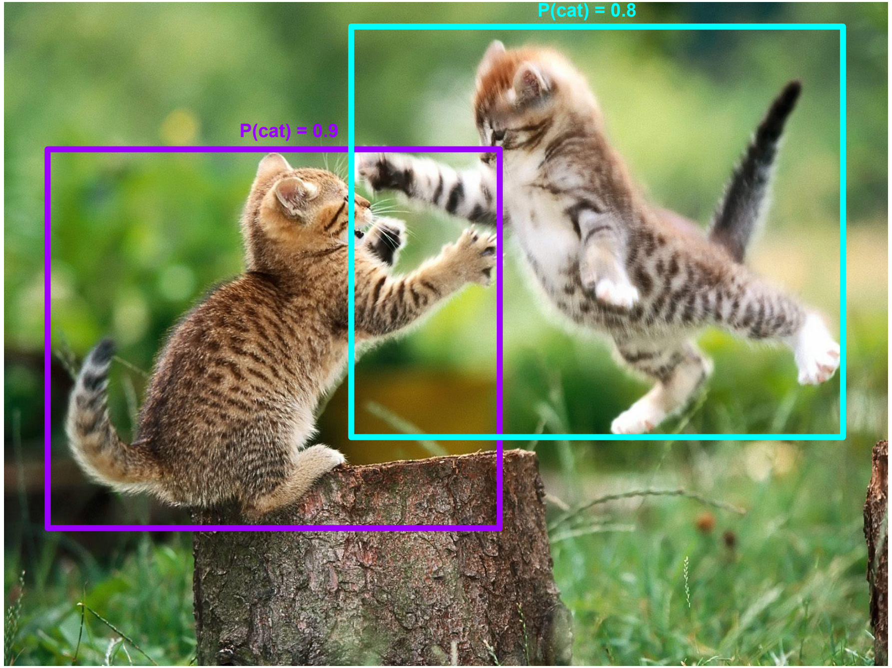
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections.

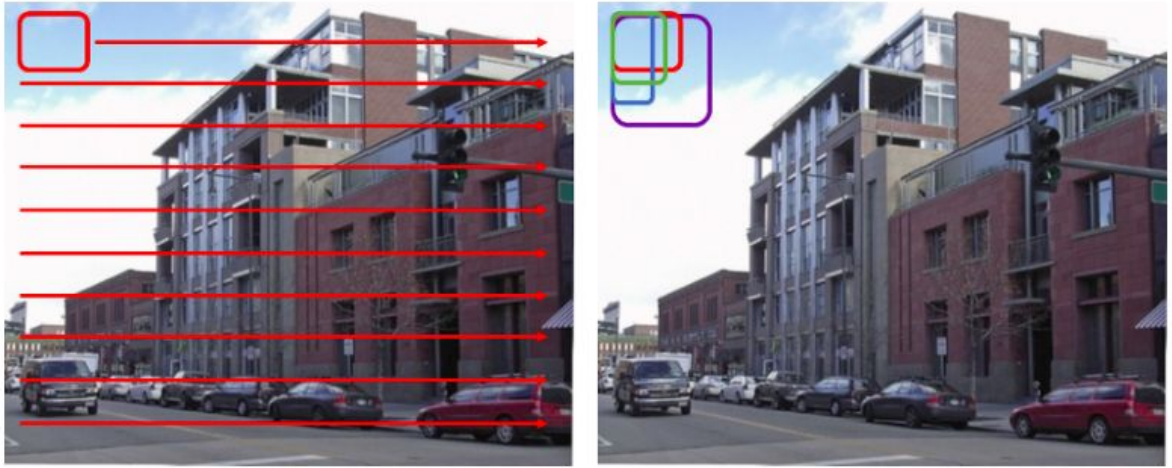
Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

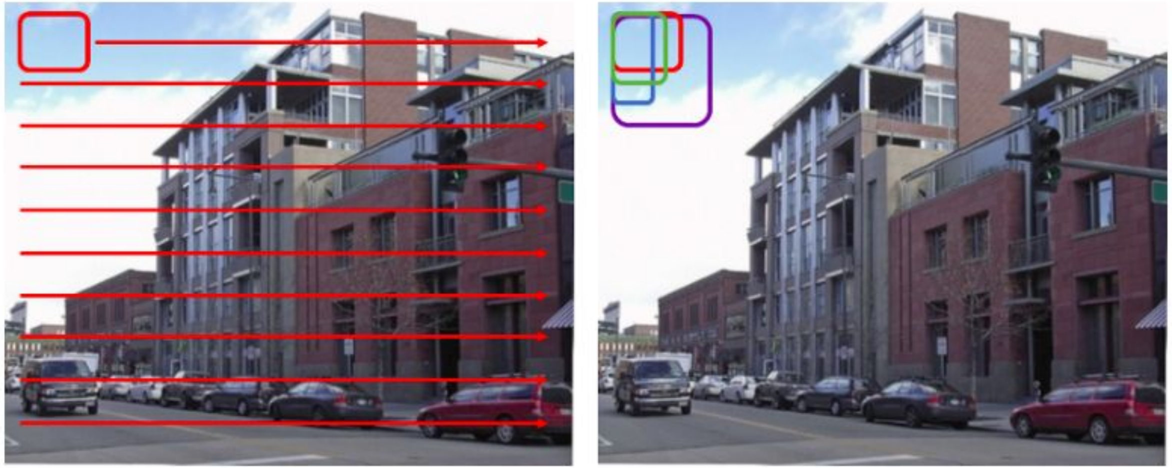
IoU(■, ■) = **0.95**



Sliding Window Object Detection

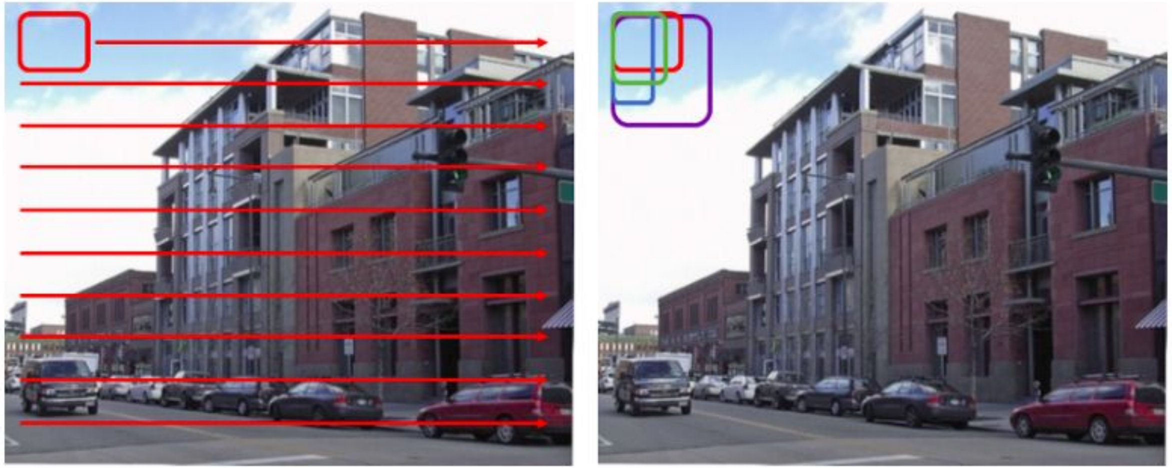


Sliding Window Object Detection



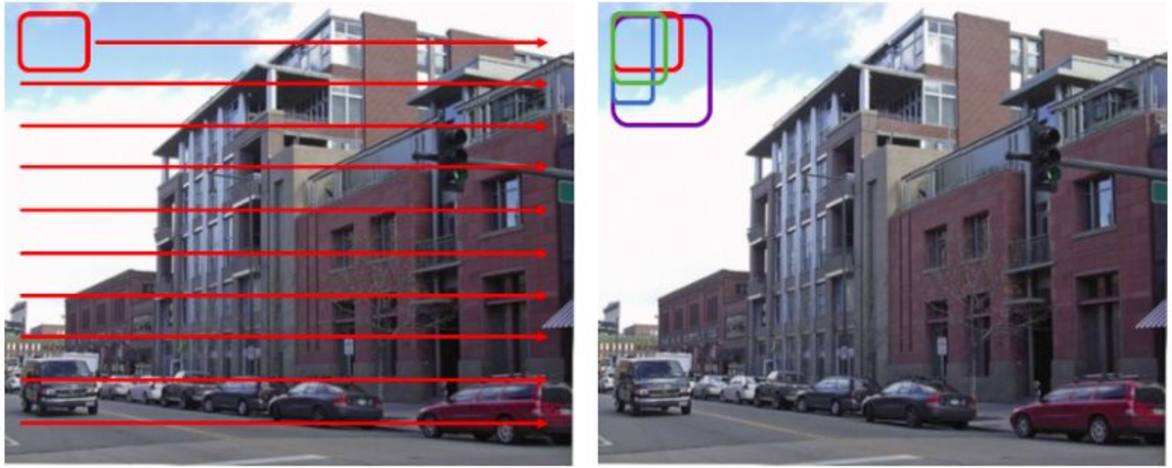
- Run **sliding window** of fixed size over the image and extract features for each image

Sliding Window Object Detection



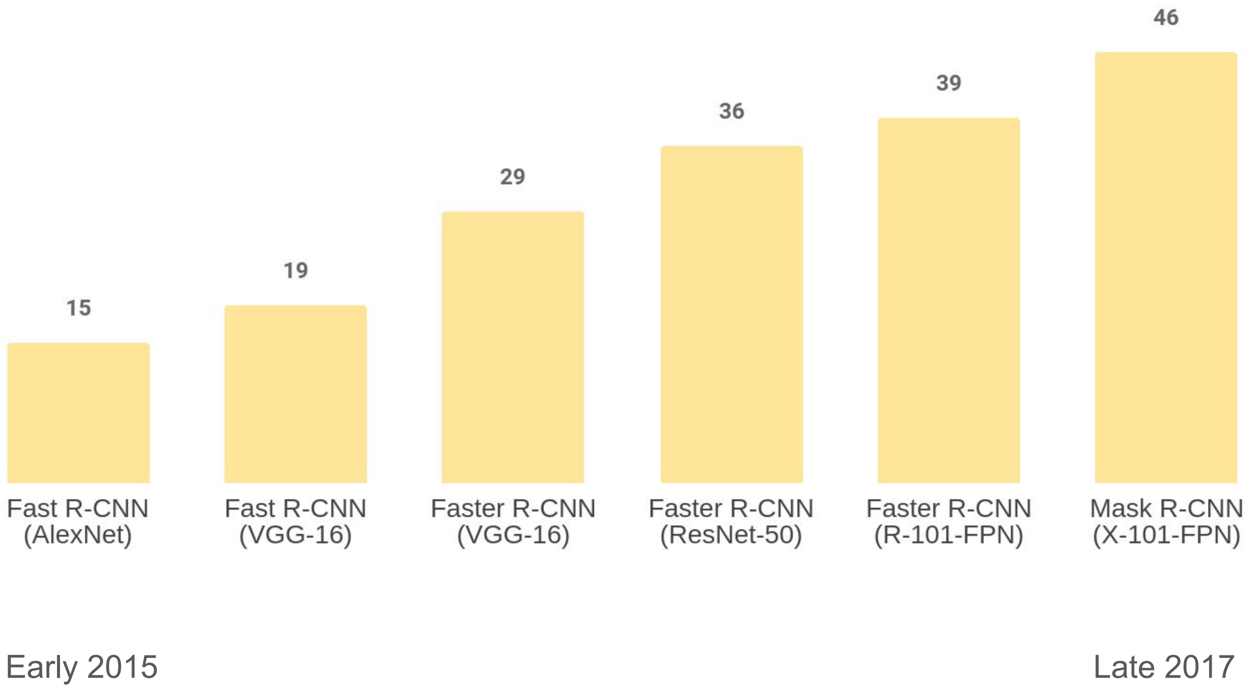
- Run **sliding window** of fixed size over the image and extract features for each image
- **Classify each crop**

Sliding Window Object Detection



- Run **sliding window** of fixed size over the image and extract features for each image
- **Classify each crop**
- Iterate over multiple aspect ratios and scales and perform **non-max suppression**

COCO Object Detection Average Precision (%)

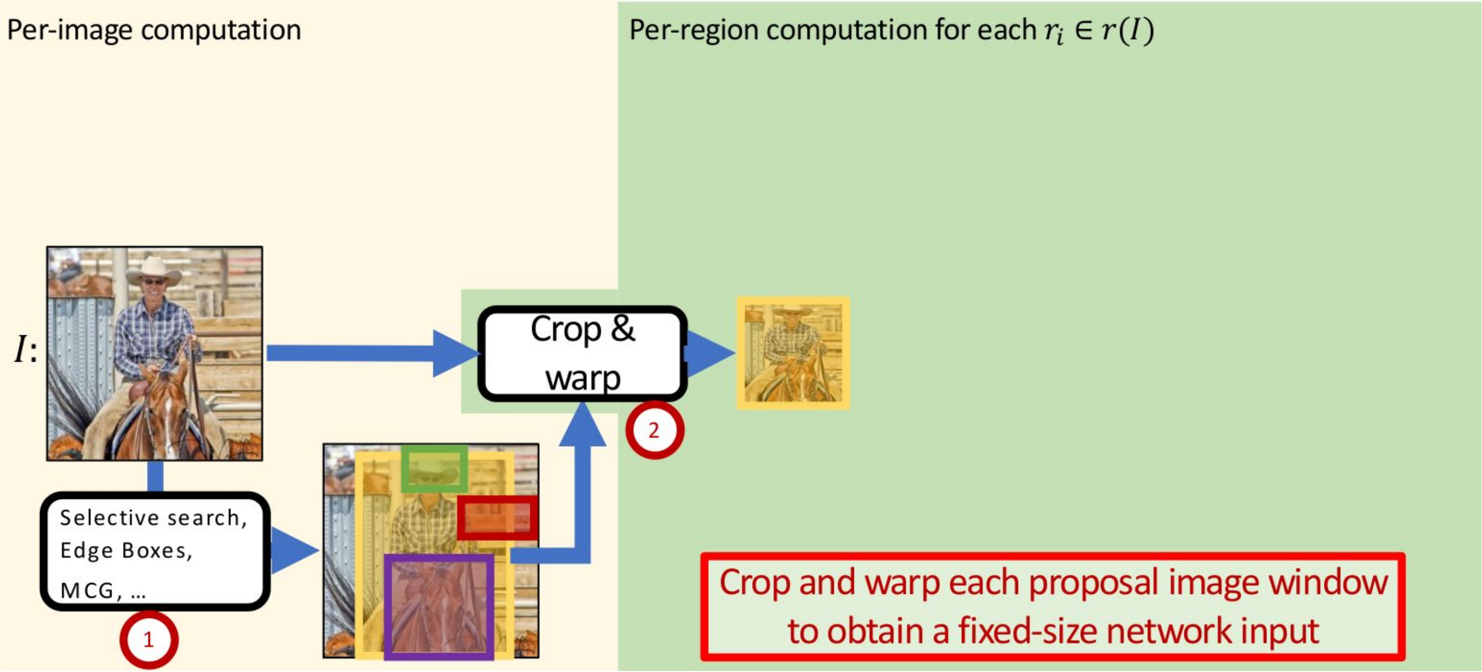


Object Detection with Deep Neural Networks
Slide Credits: Ross Girshick

R-CNN [1]: Region-based Convolutional Neural Network

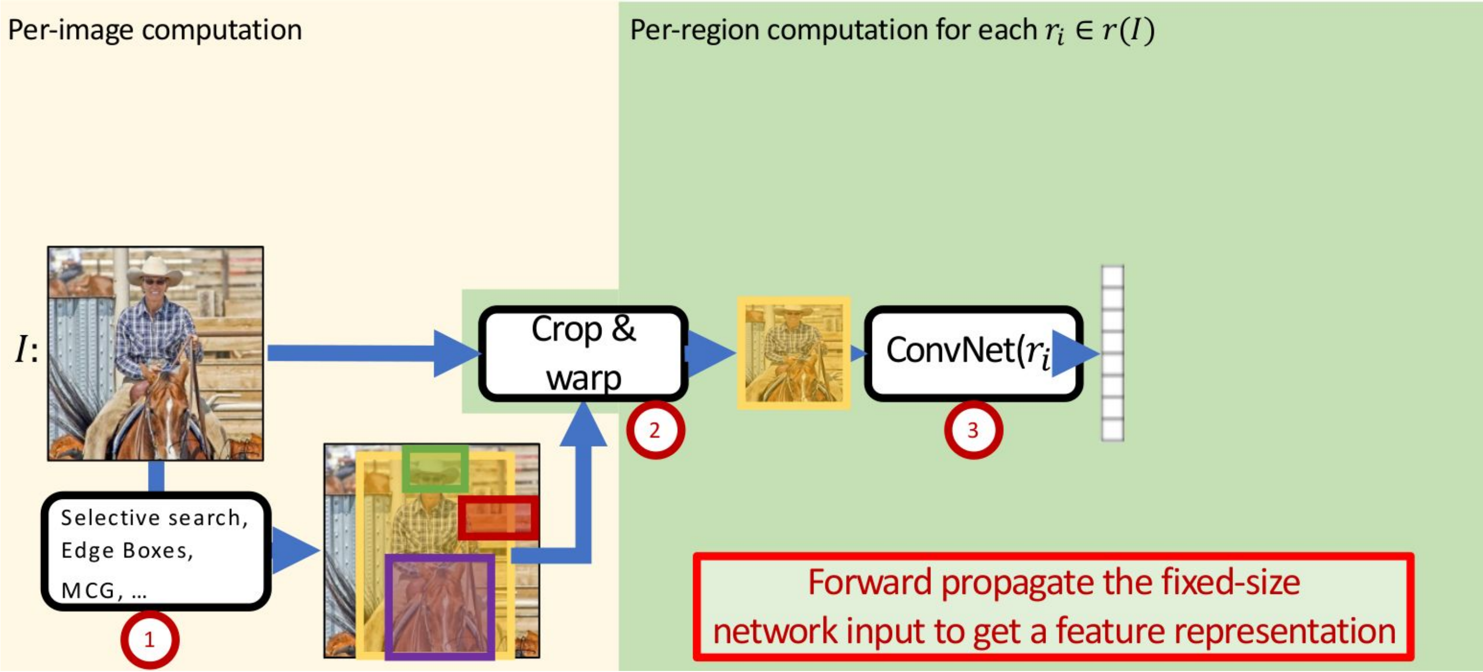
[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN [1]: Region-based Convolutional Neural Network



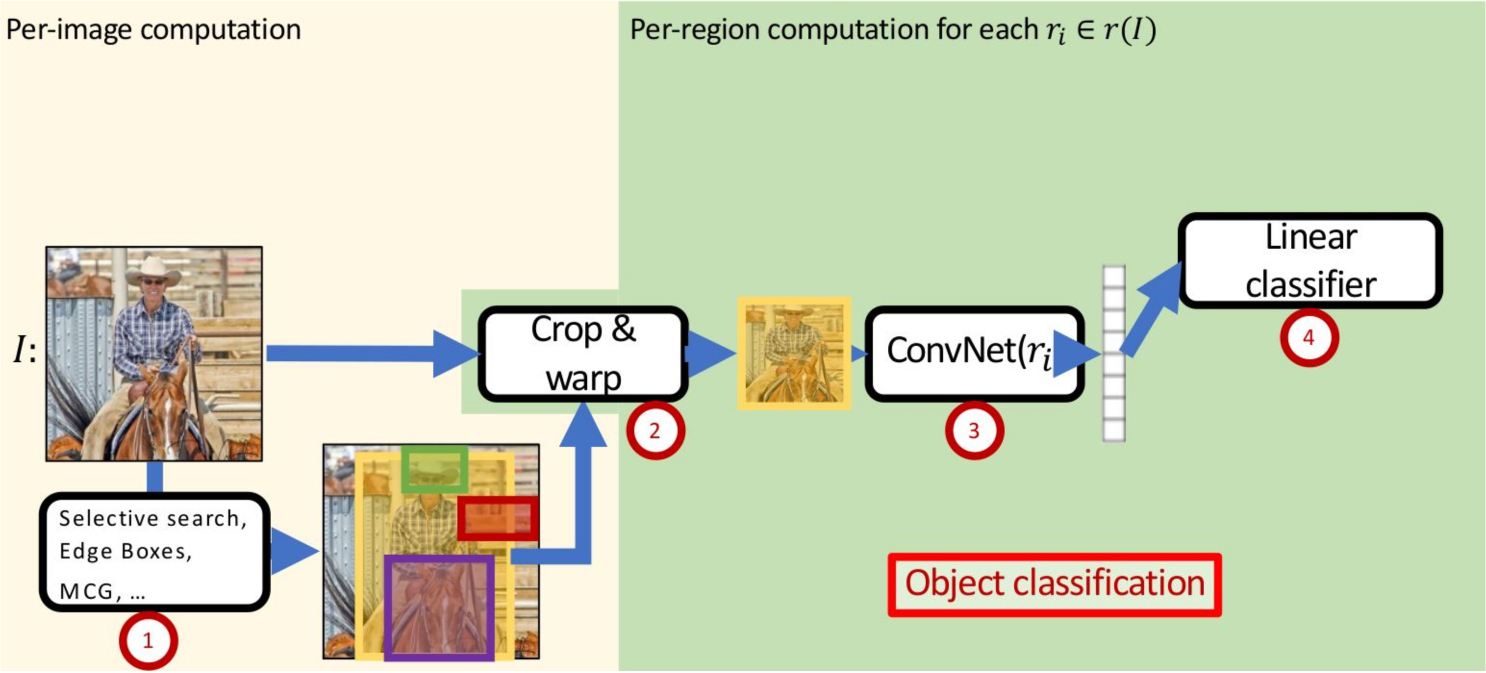
[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN [1]: Region-based Convolutional Neural Network



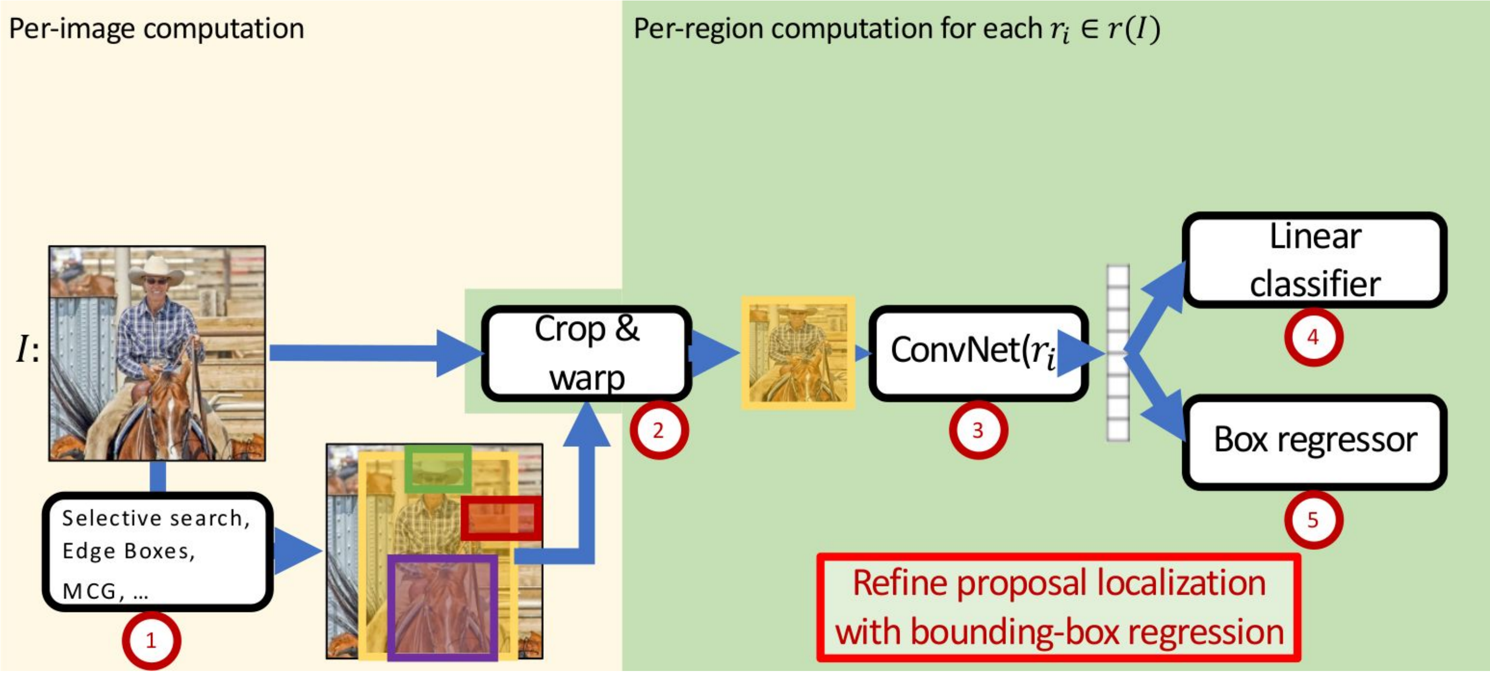
[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN [1]: Region-based Convolutional Neural Network



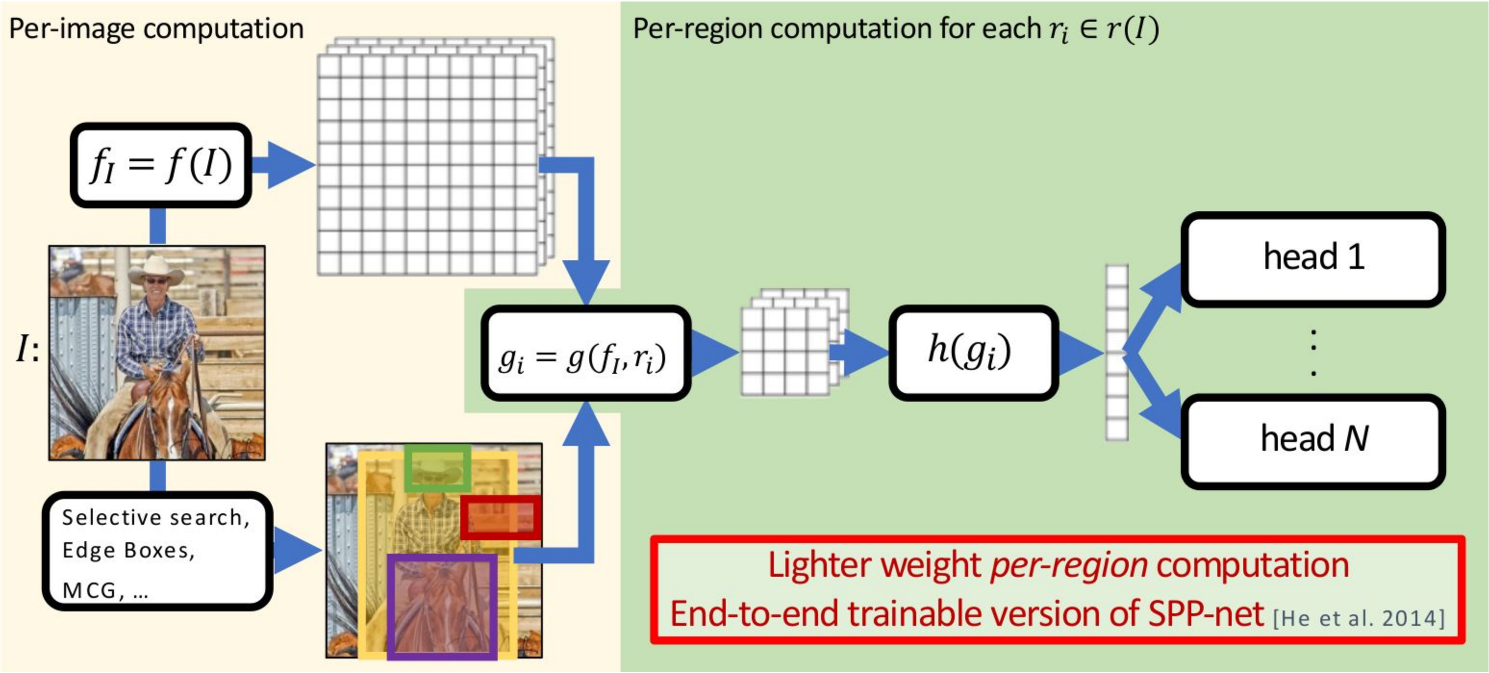
[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN [1]: Region-based Convolutional Neural Network

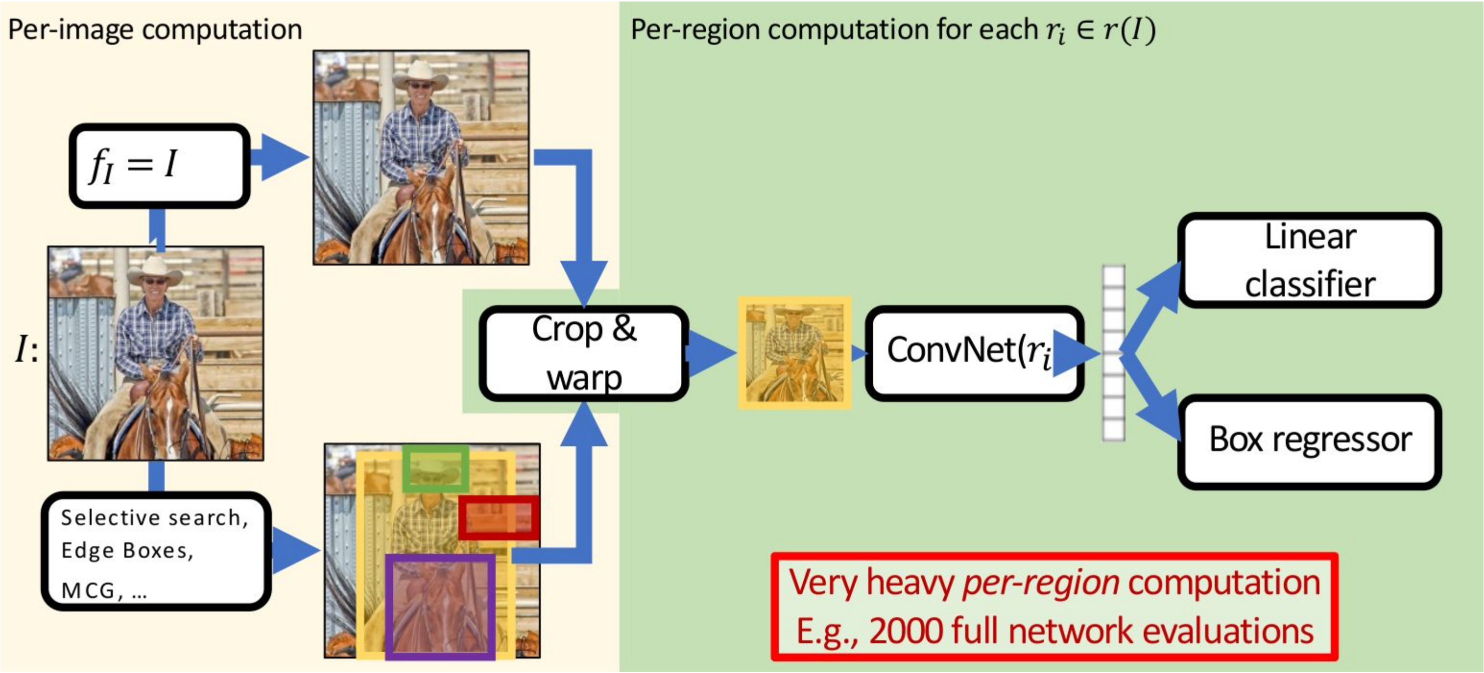


[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Generalized framework



R-CNN in the Generalized Framework



[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN: Problems

- Heavy per-region computation (2000 full network evaluations)

[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN: Problems

- Heavy per-region computation (2000 full network evaluations)
- No computation / feature sharing

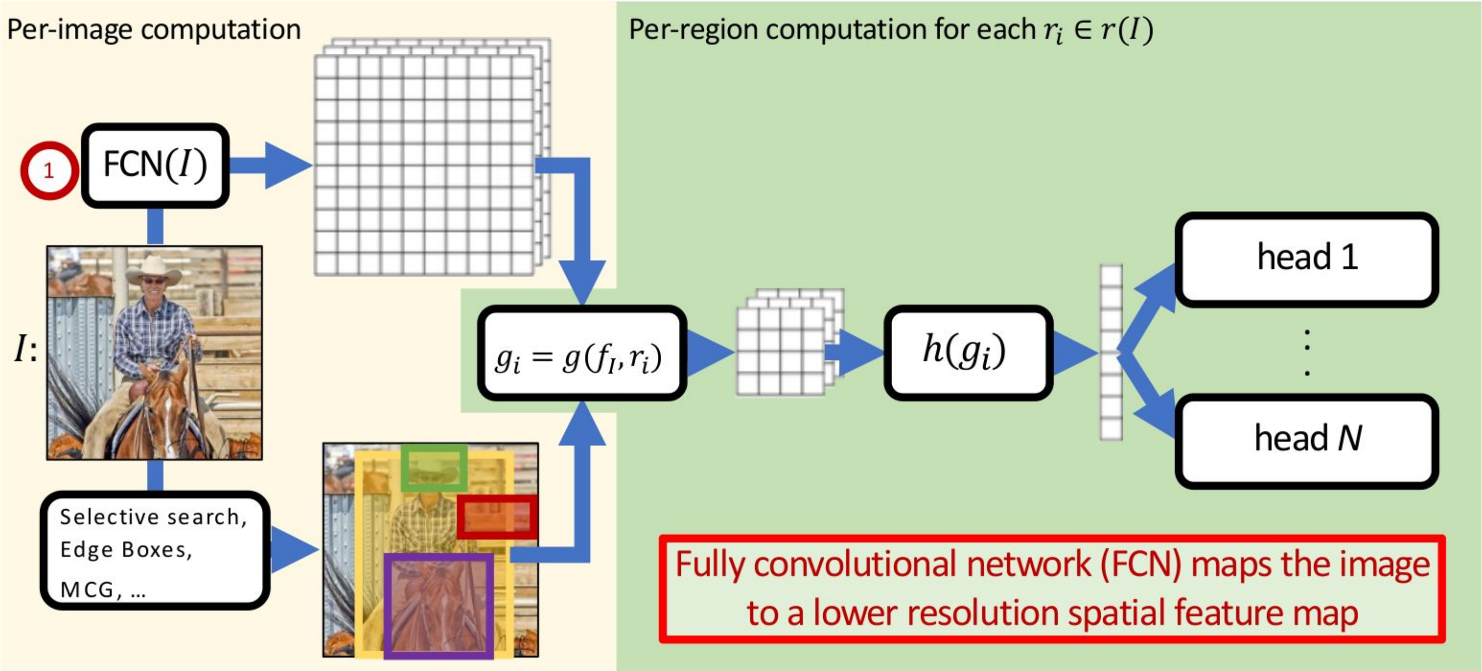
[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN: Problems

- Heavy per-region computation (2000 full network evaluations)
- No computation / feature sharing
- Slow region proposal method adds to runtime

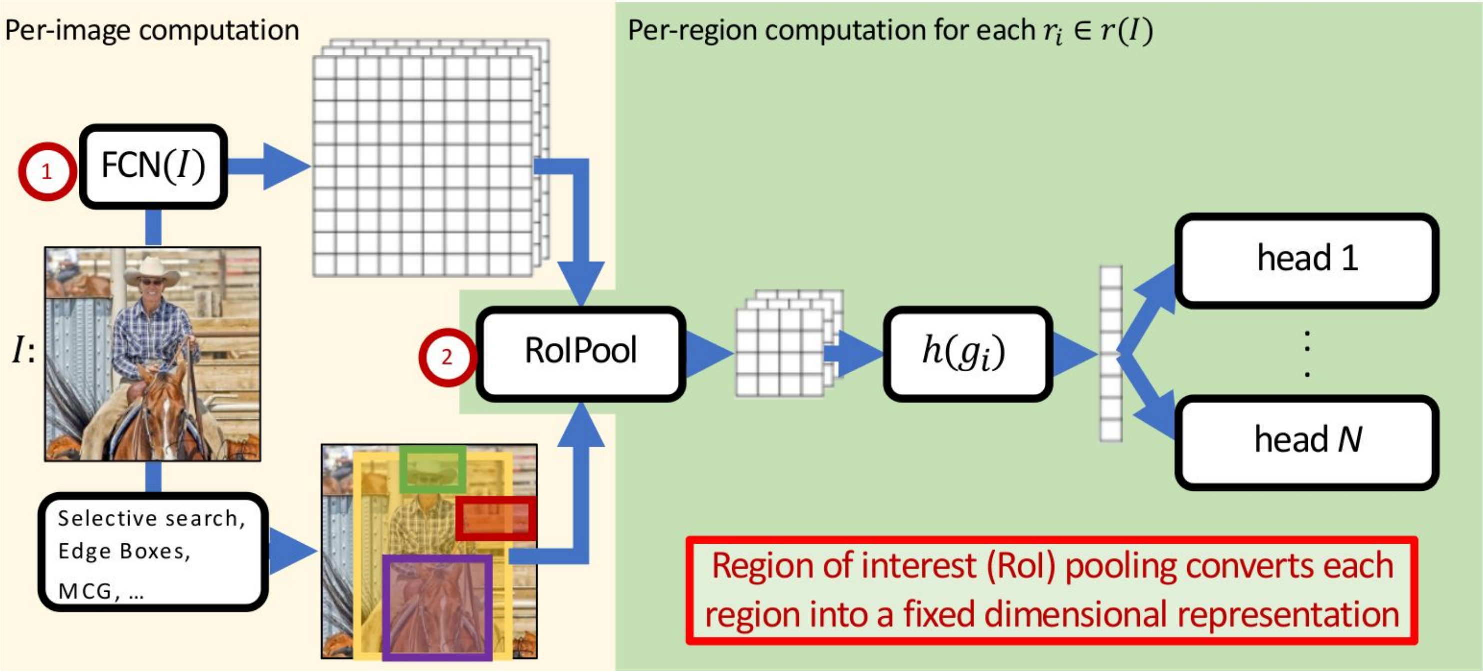
[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Fast R-CNN



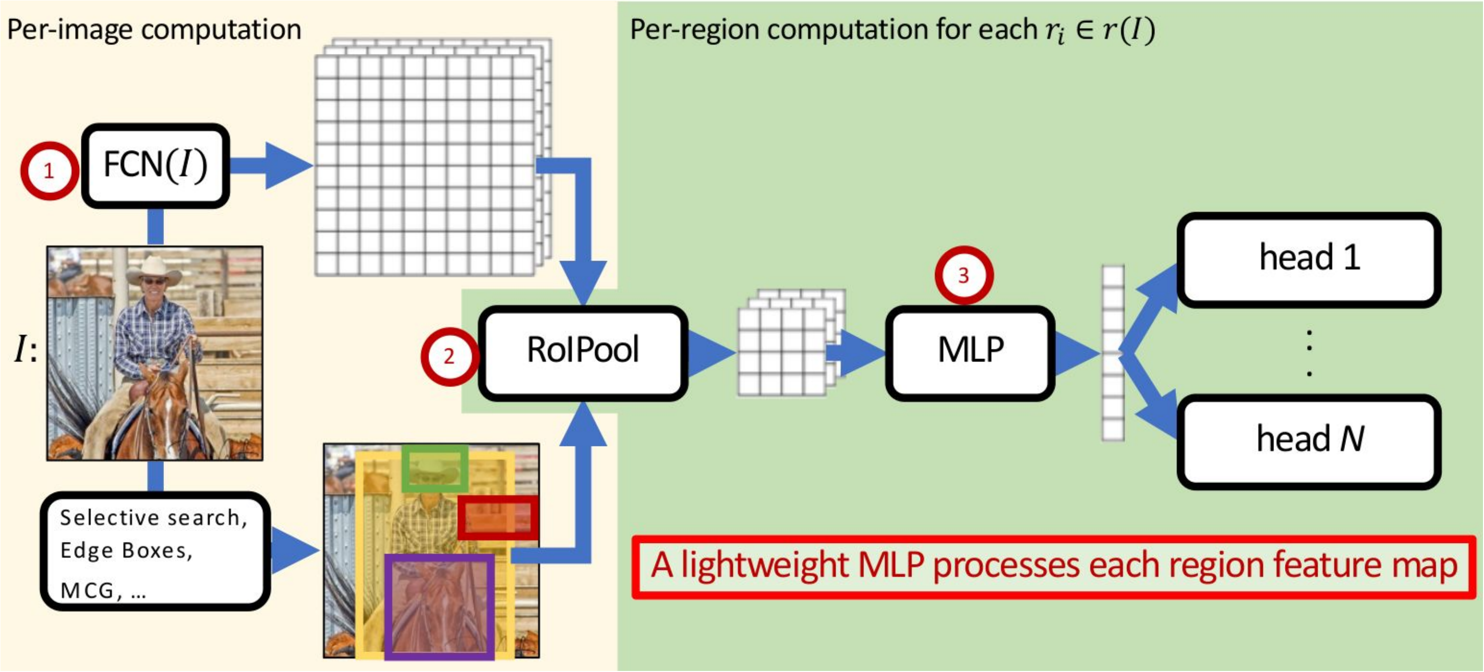
[1] Girshick. Fast R-CNN. ICCV, 2015.

Fast R-CNN



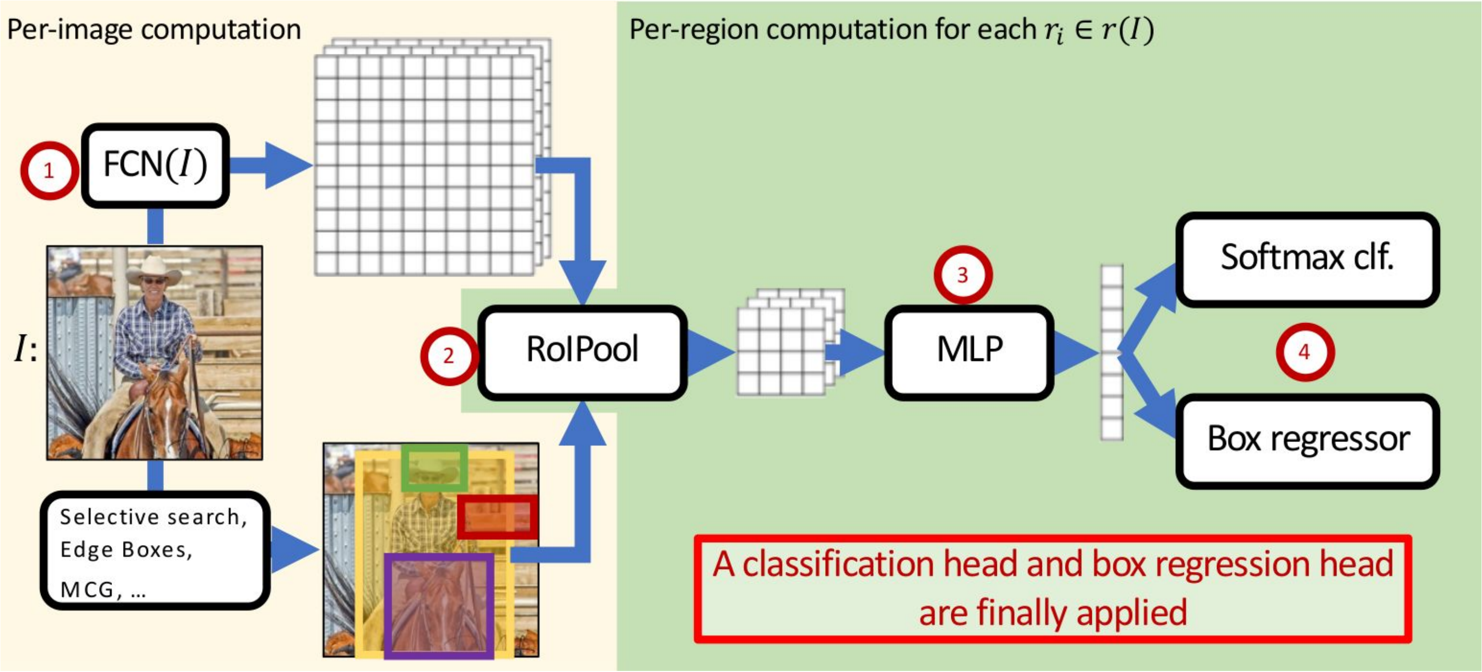
[1] Girshick. Fast R-CNN. ICCV, 2015.

Fast R-CNN



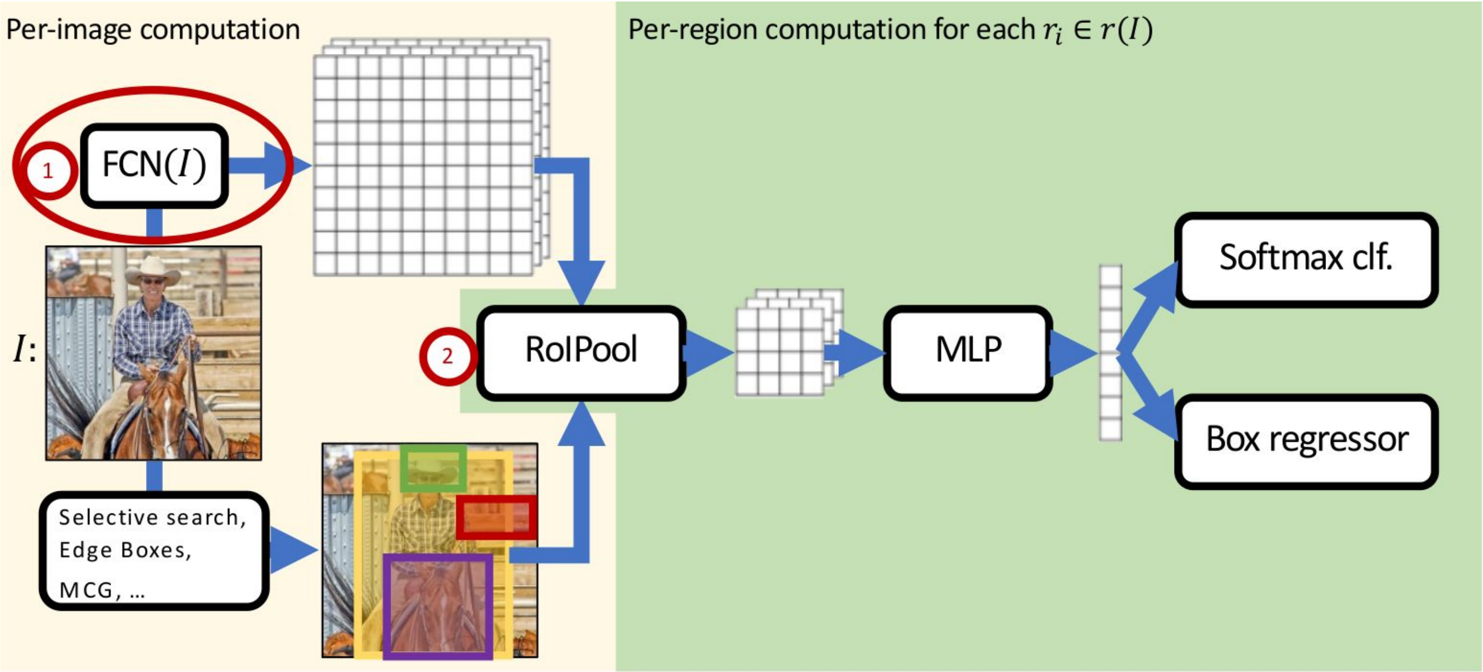
[1] Girshick. Fast R-CNN. ICCV, 2015.

Fast R-CNN



[1] Girshick. Fast R-CNN. ICCV, 2015.

Fast R-CNN



[1] Girshick. Fast R-CNN. ICCV, 2015.

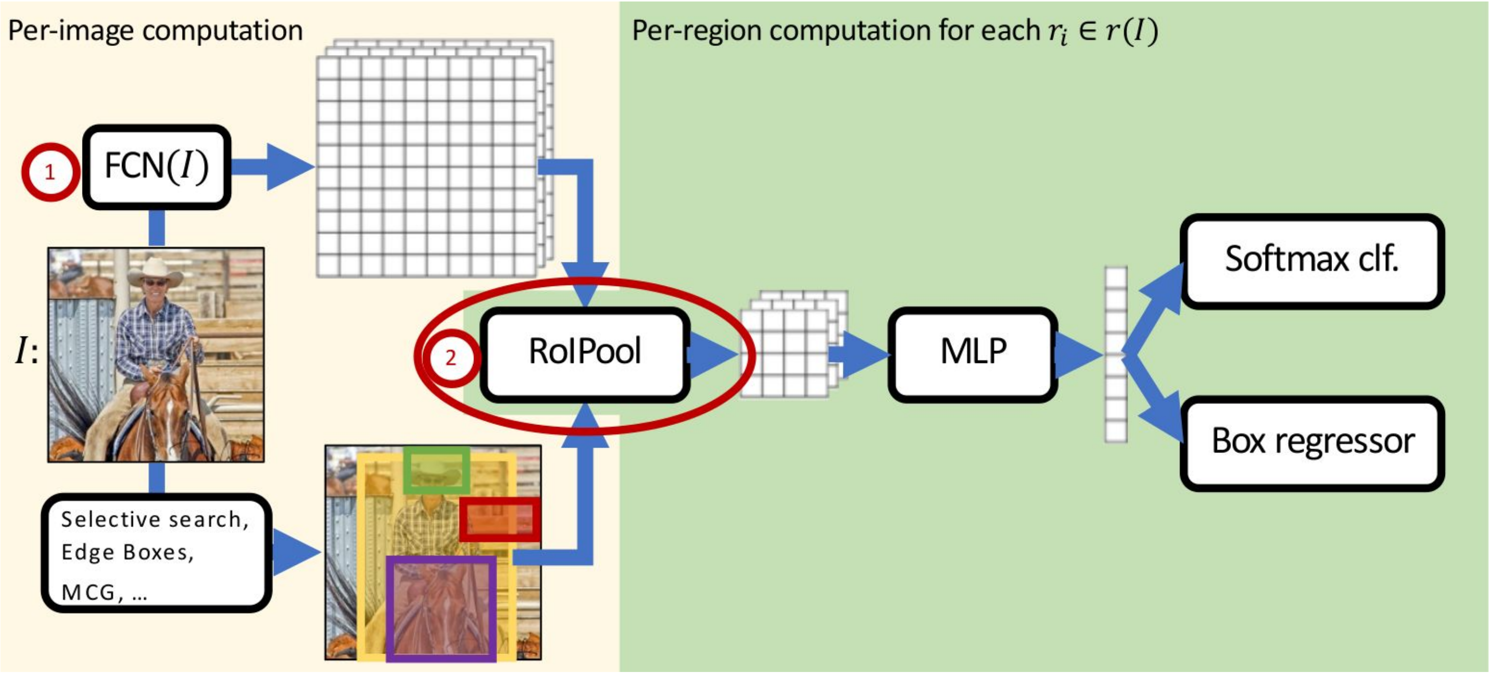
Fast R-CNN

Backbone:

- Use any standard convolutional network as “backbone” architecture, e.g. AlexNet, VGG, ResNet, Inception, ResNeXt, DenseNet, …
- Remove global pooling => output spatial dims proportional to input spatial dims
- A good network exploits the strongest recognition backbone (features matter!)

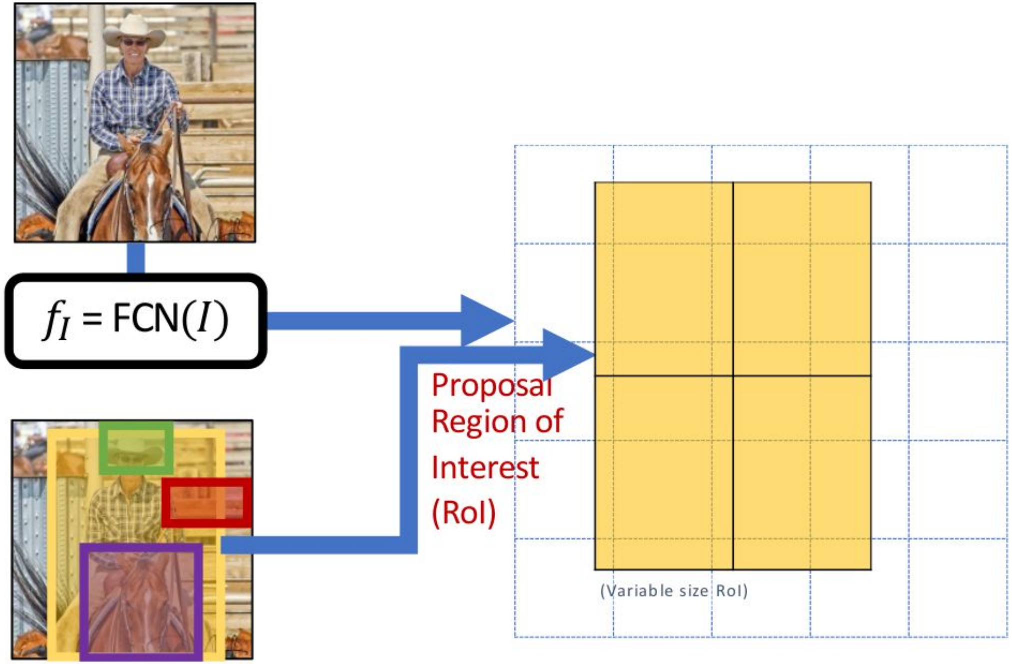
[1] Girshick. Fast R-CNN. ICCV, 2015.

Fast R-CNN



[1] Girshick. Fast R-CNN. ICCV, 2015.

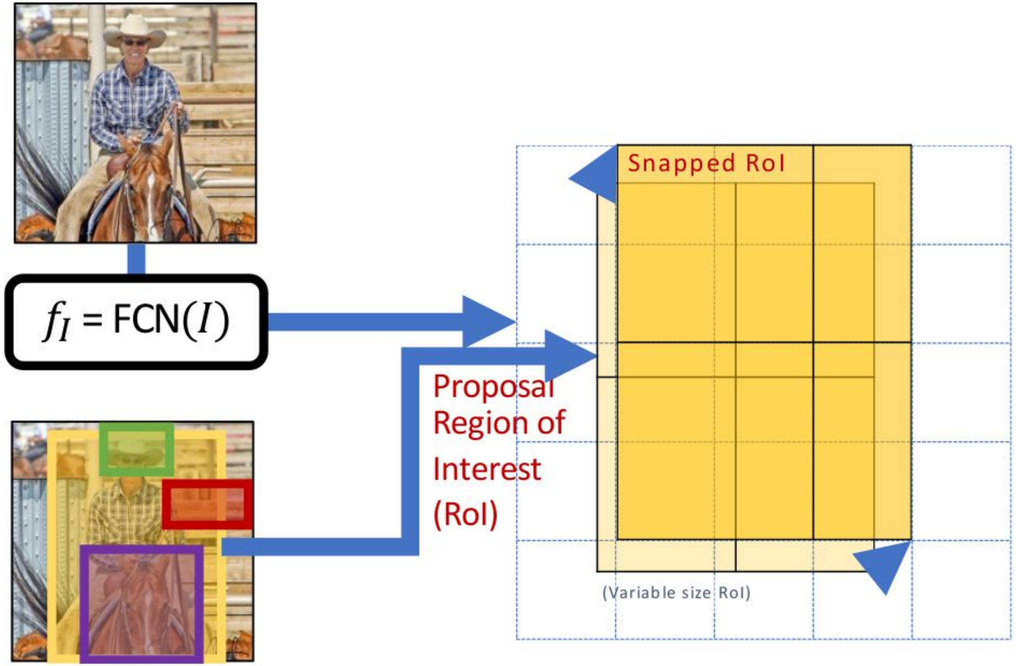
Fast R-CNN



Key innovation in SPP-net
[He et al. 2014]

[1] Girshick. Fast R-CNN. ICCV, 2015.

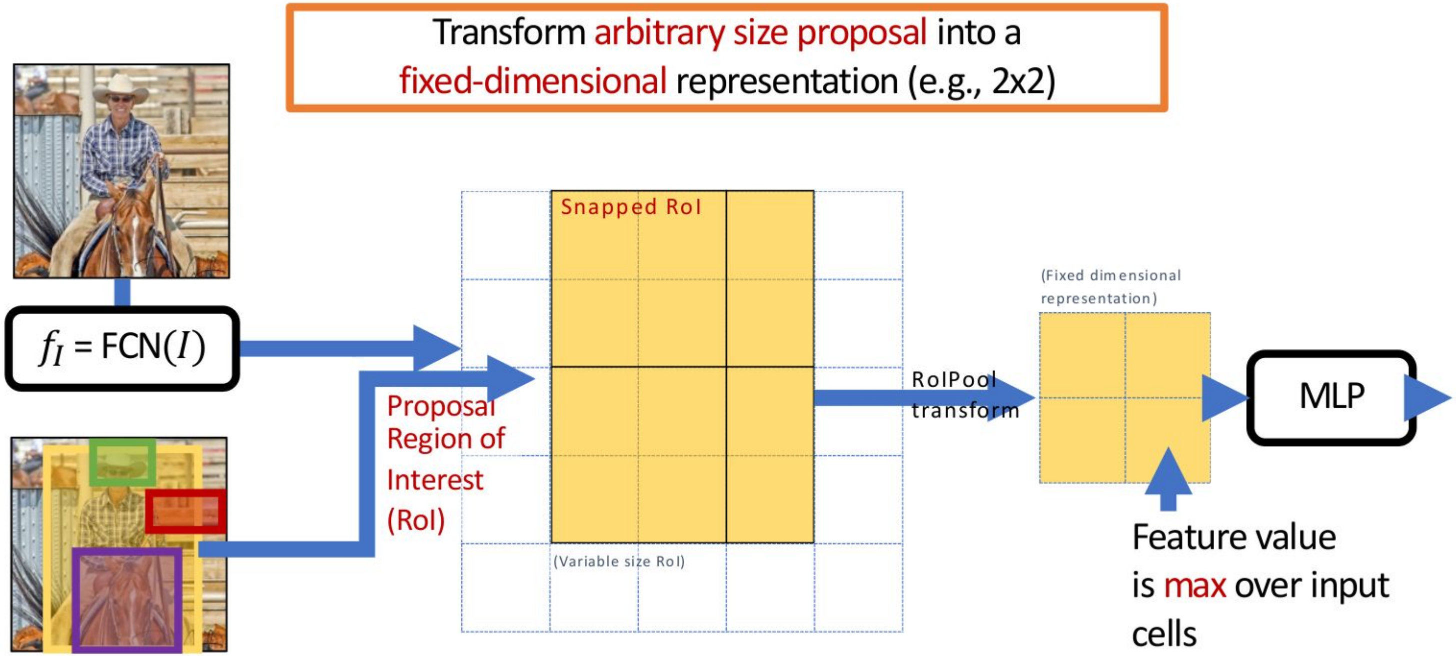
Fast R-CNN



Key innovation in SPP-net
[He et al. 2014]

[1] Girshick. Fast R-CNN. ICCV, 2015.

Fast R-CNN



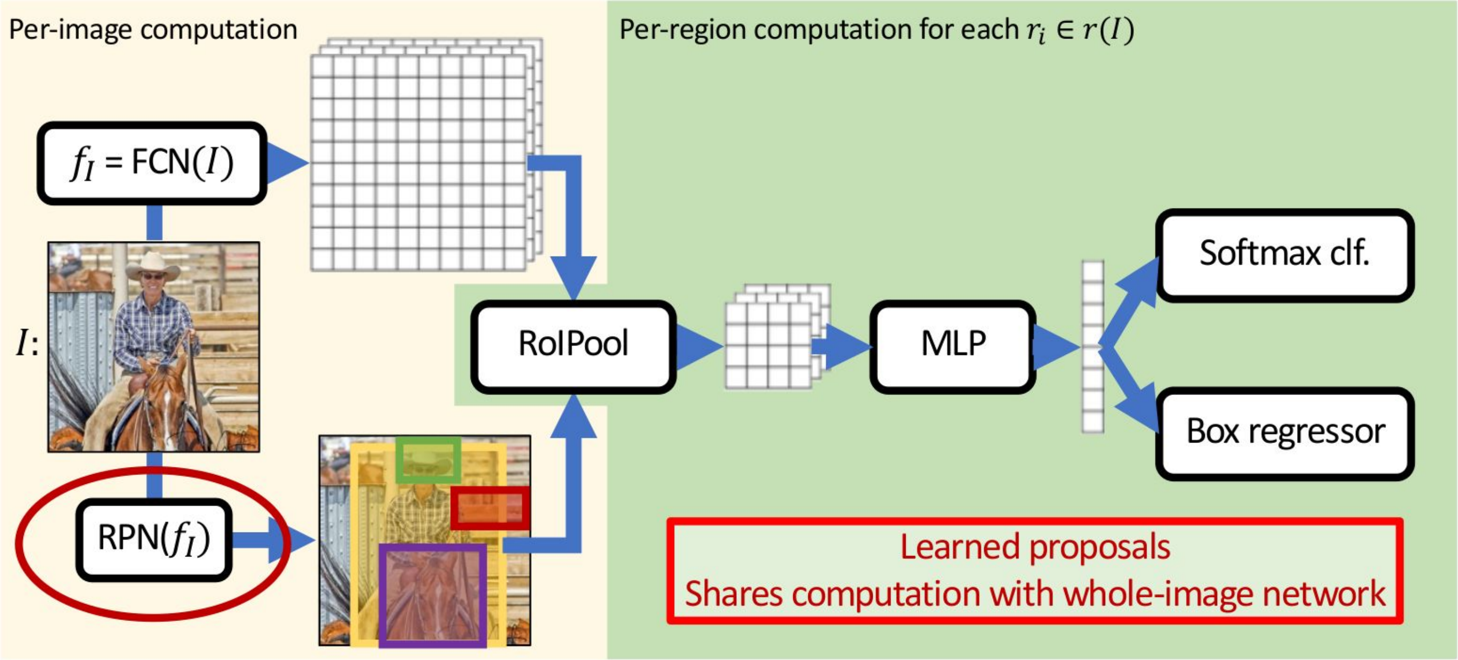
[1] Girshick. Fast R-CNN. ICCV, 2015.

Fast R-CNN problems

- ~~Heavy per-region computation (2000 full network evaluations)~~
- ~~No computation / feature sharing~~
- Slow region proposal method adds to runtime
- Generic proposal method has low recall

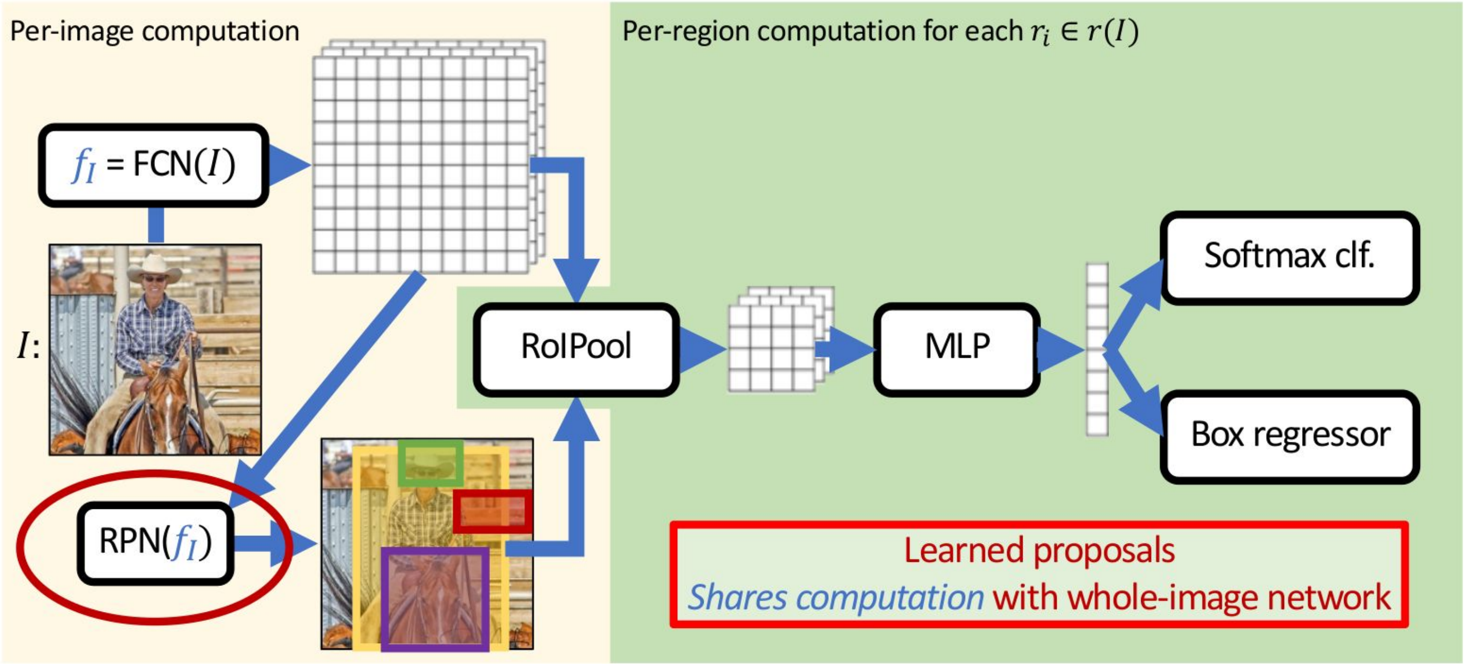
[1] Girshick, Donahue, Darrell and Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Faster R-CNN



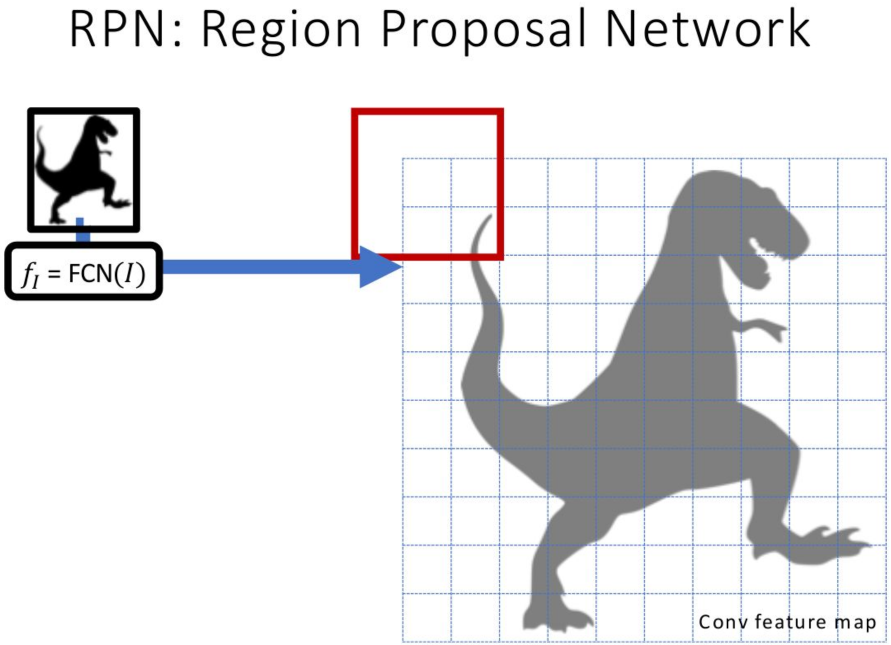
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN



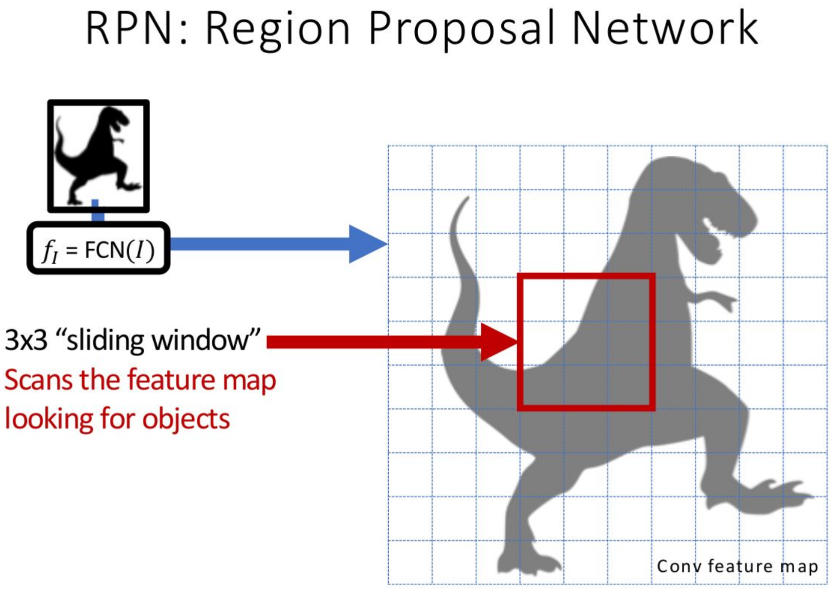
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



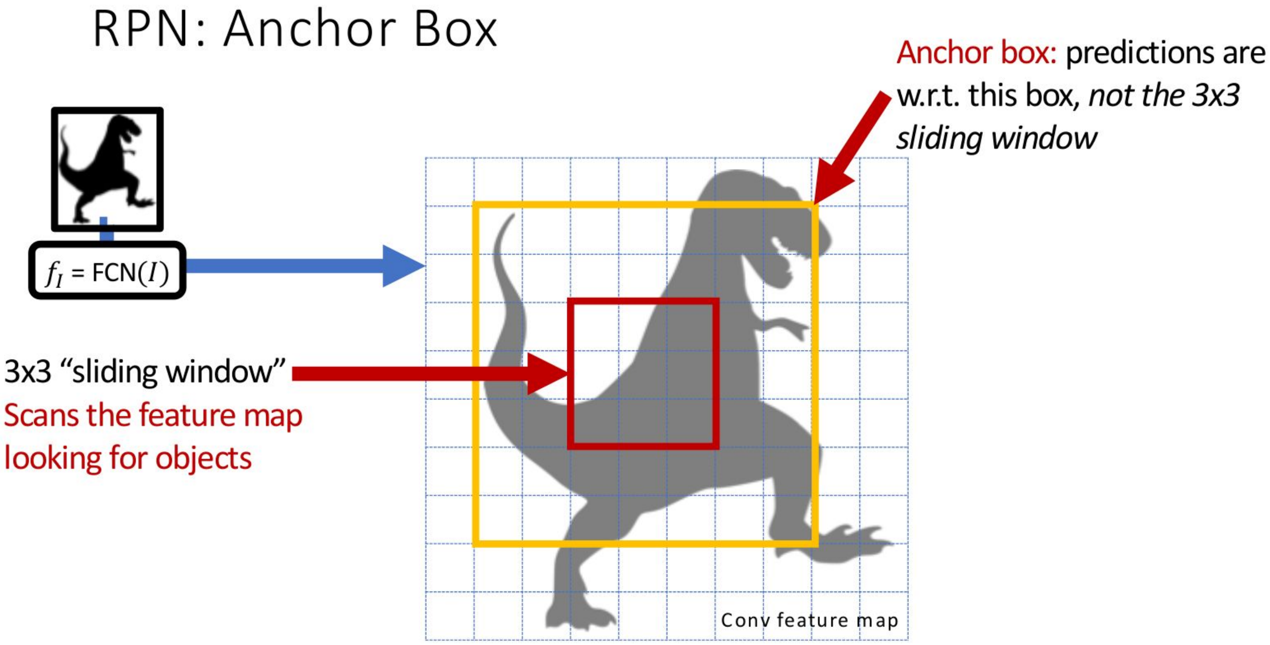
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



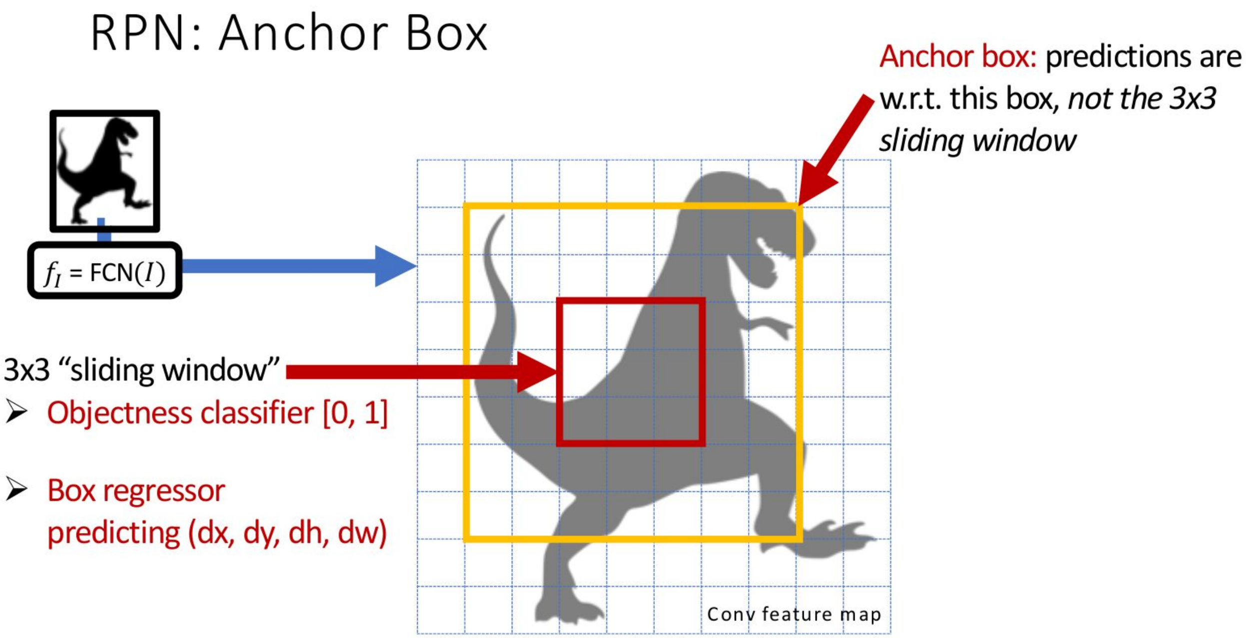
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



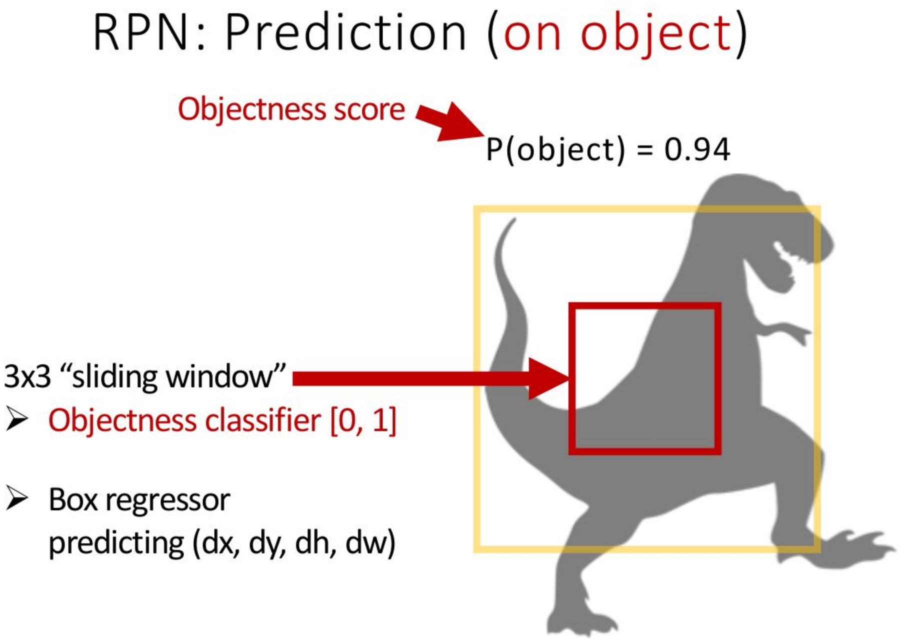
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



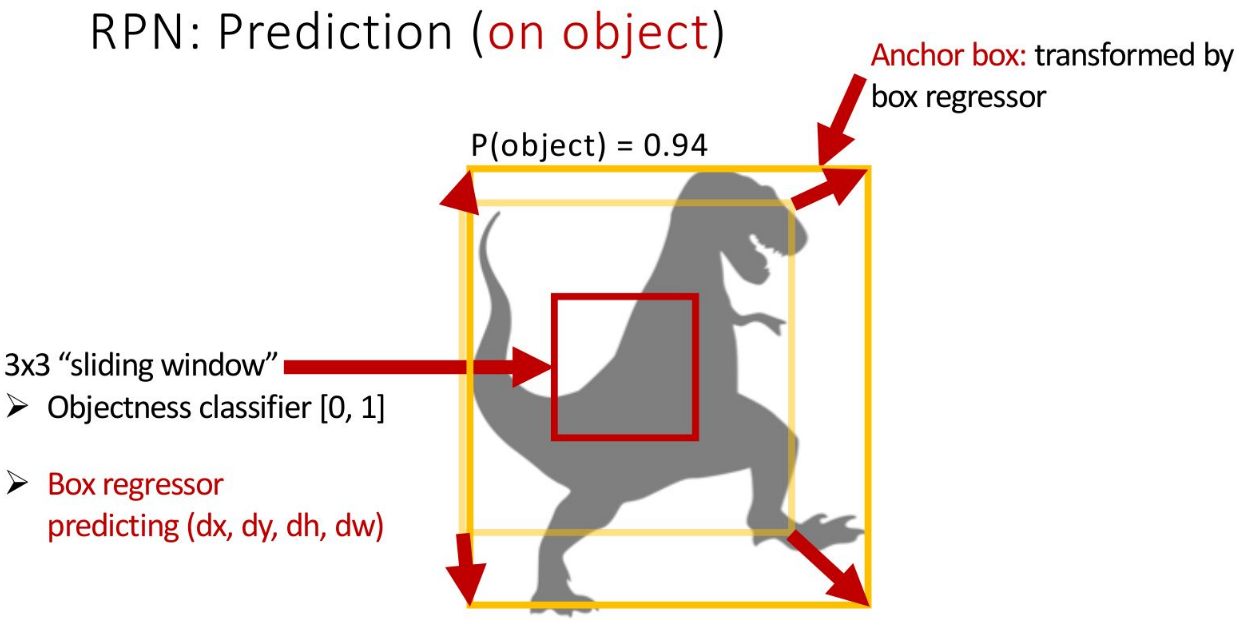
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



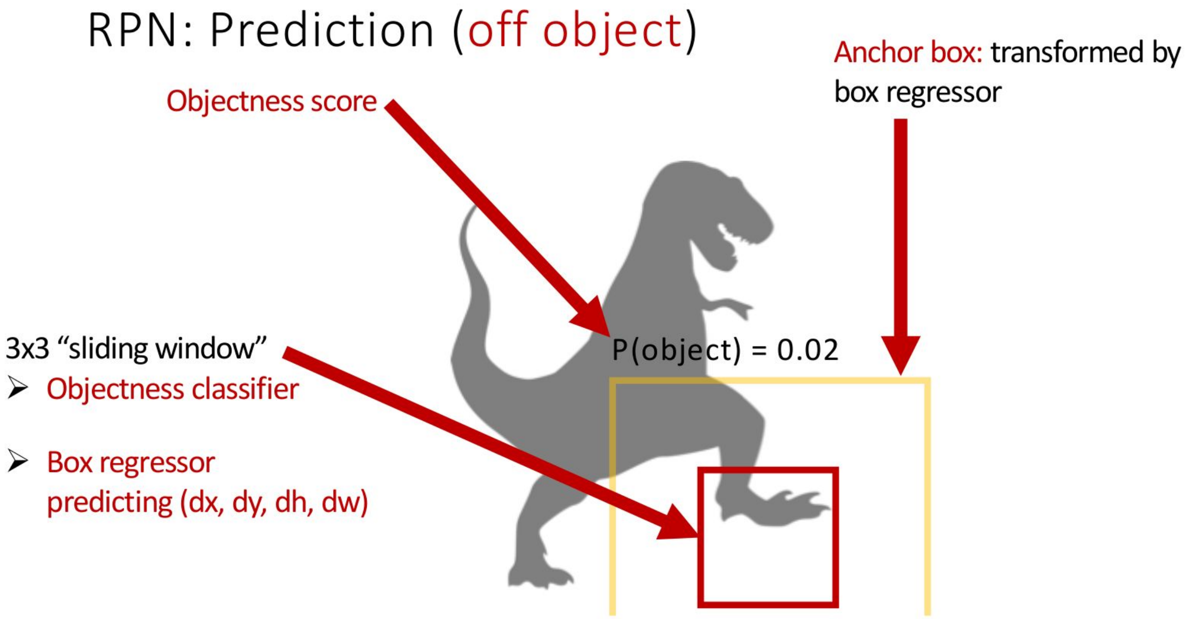
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



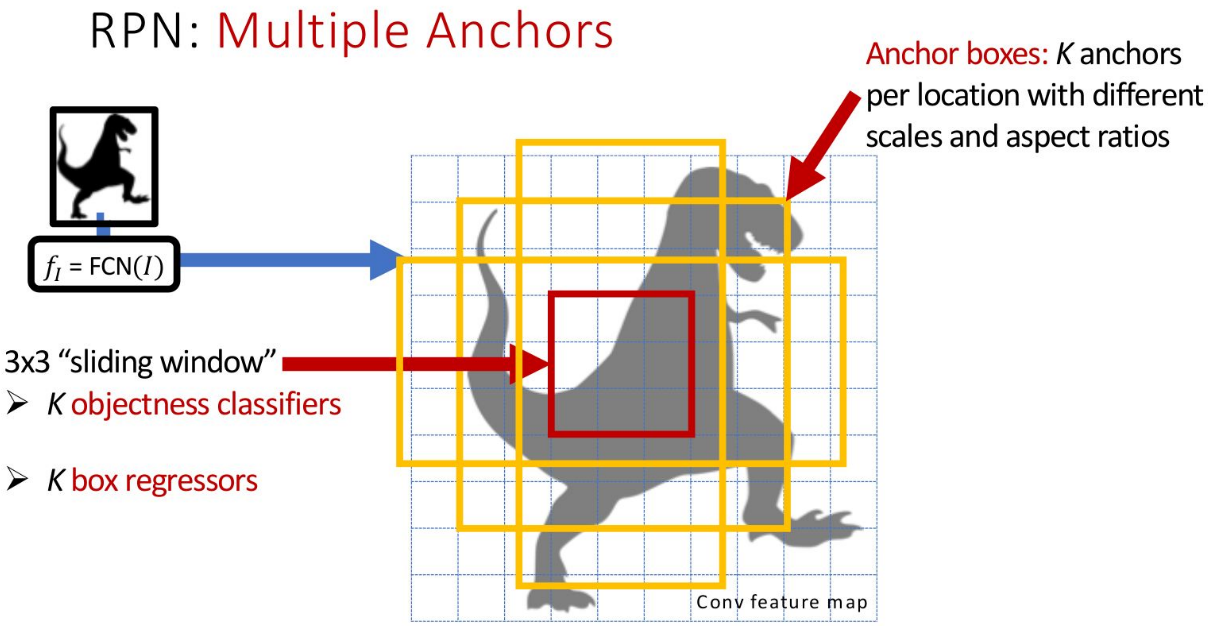
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



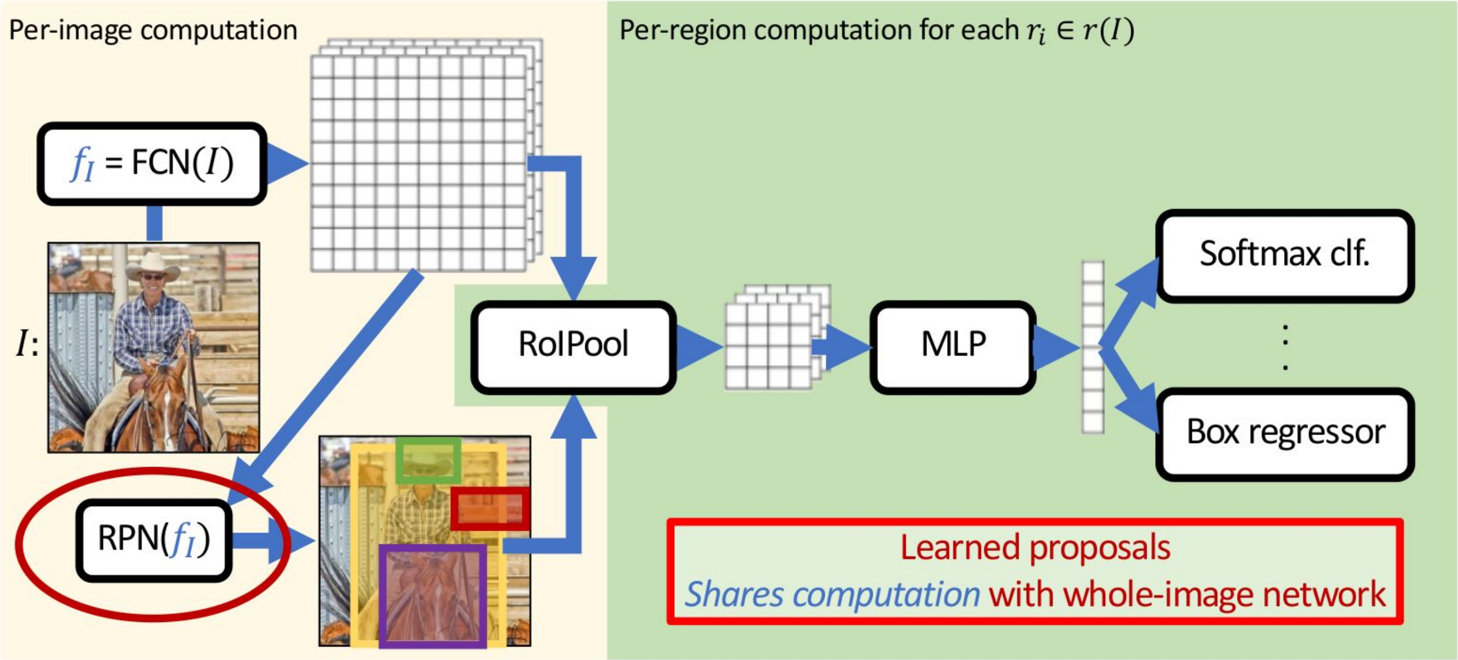
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)



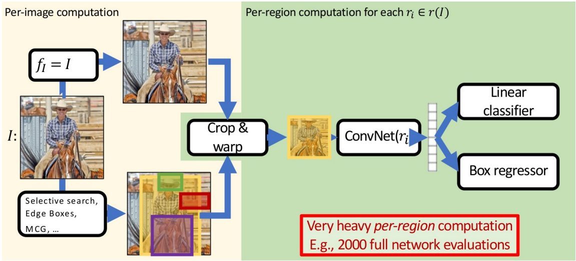
[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

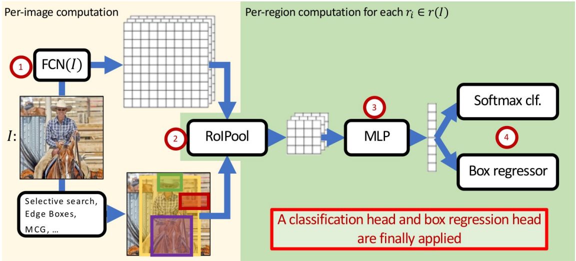
Faster R-CNN

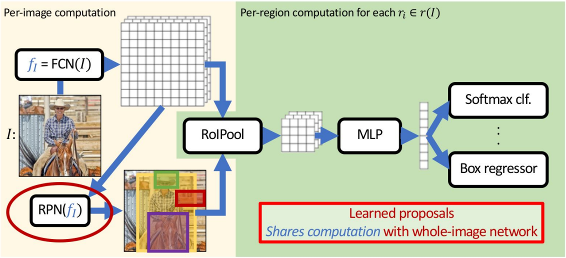


[1] Ren, He, Girshick and Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Summary







"Slow" R-CNN:

Run CNN independently for each region

Fast R-CNN:

Apply differentiable cropping to shared image features

Faster R-CNN:

Compute proposals with CNN

Q&A