What can('t) we do we ConvNets?

Pose regression + Object detection

Karel Zimmermann
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Outline

- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks

OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf

input

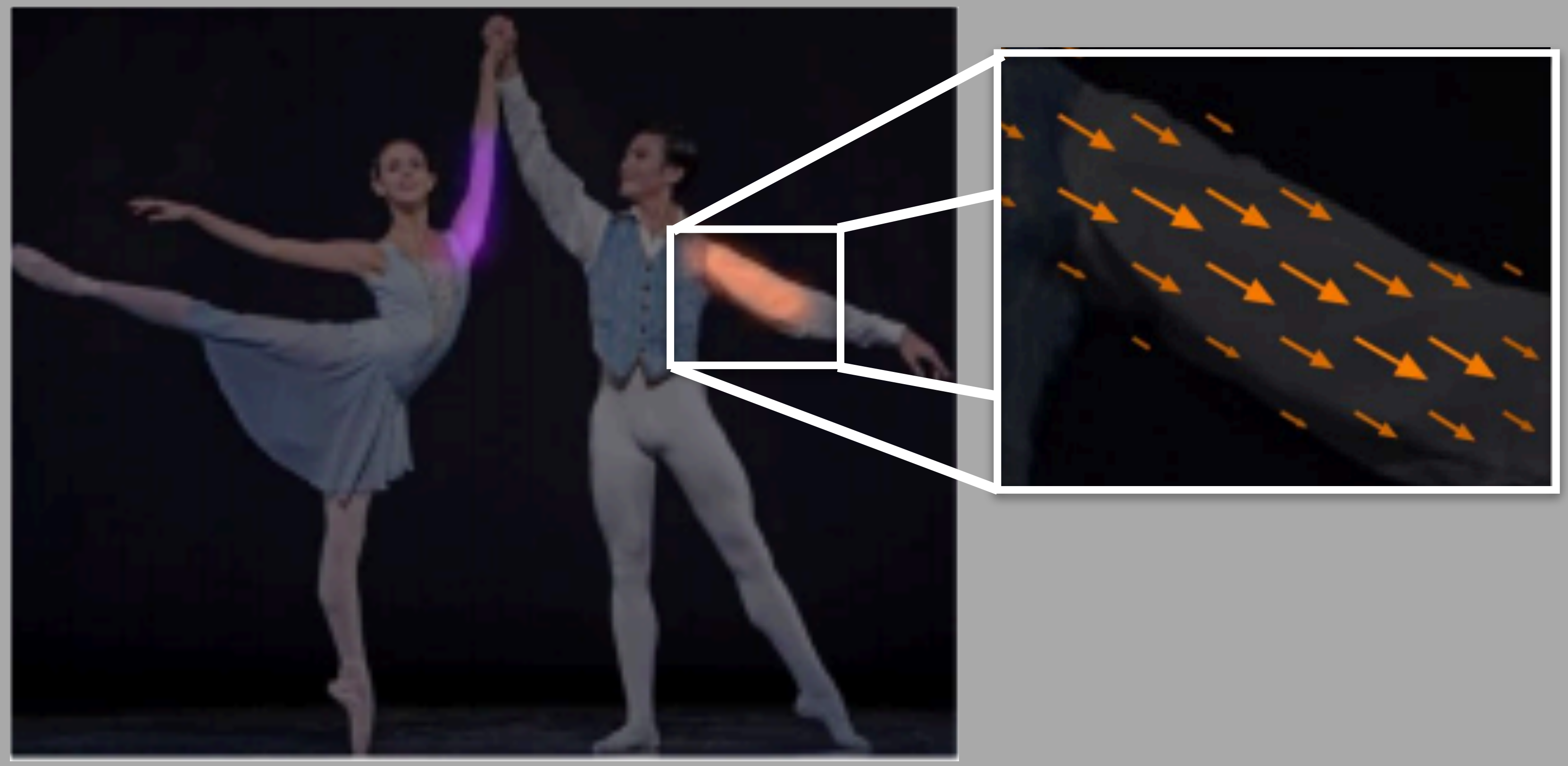
VGG-19

$\mathbf{F}$

"Complicated stuff inside":

(1) detect joints

(2) estimate limbs directions



output

PoseTrack challenge (ICCV 2017/ECCV 2018)
https://posetrack.net

Pose regression references

- PoseTrack benchmark a datasets
  https://posetrack.net

- Guler et al. (Facebook Research), DensePose
  https://arxiv.org/abs/1802.00434
  https://github.com/facebookresearch/Densepose
  https://www.youtube.com/watch?v=EMjPqgLX14A&feature=youtu.be

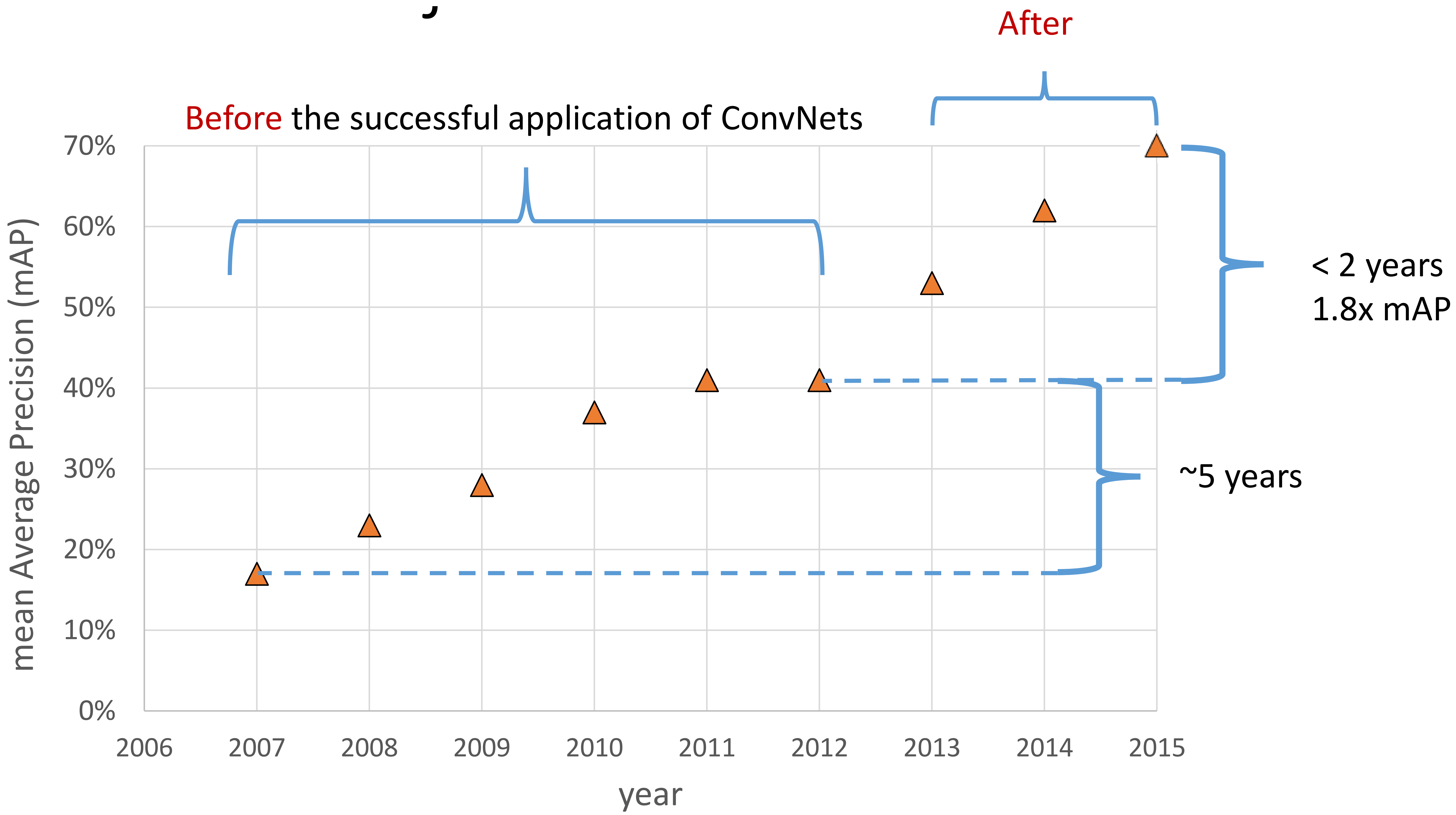
- Realtime Multi-Person 2D Human Pose Estimation using Part Affinity Fields, CVPR 2017 Oral
  https://www.youtube.com/watch?v=pW6nZXeWlGM

- Integral Human Pose Regression [Sun ECCV 2018]
  Microsoft Research
  https://arxiv.org/abs/1711.08229
  https://github.com/JimmySuen/integral-human-pose

Outline

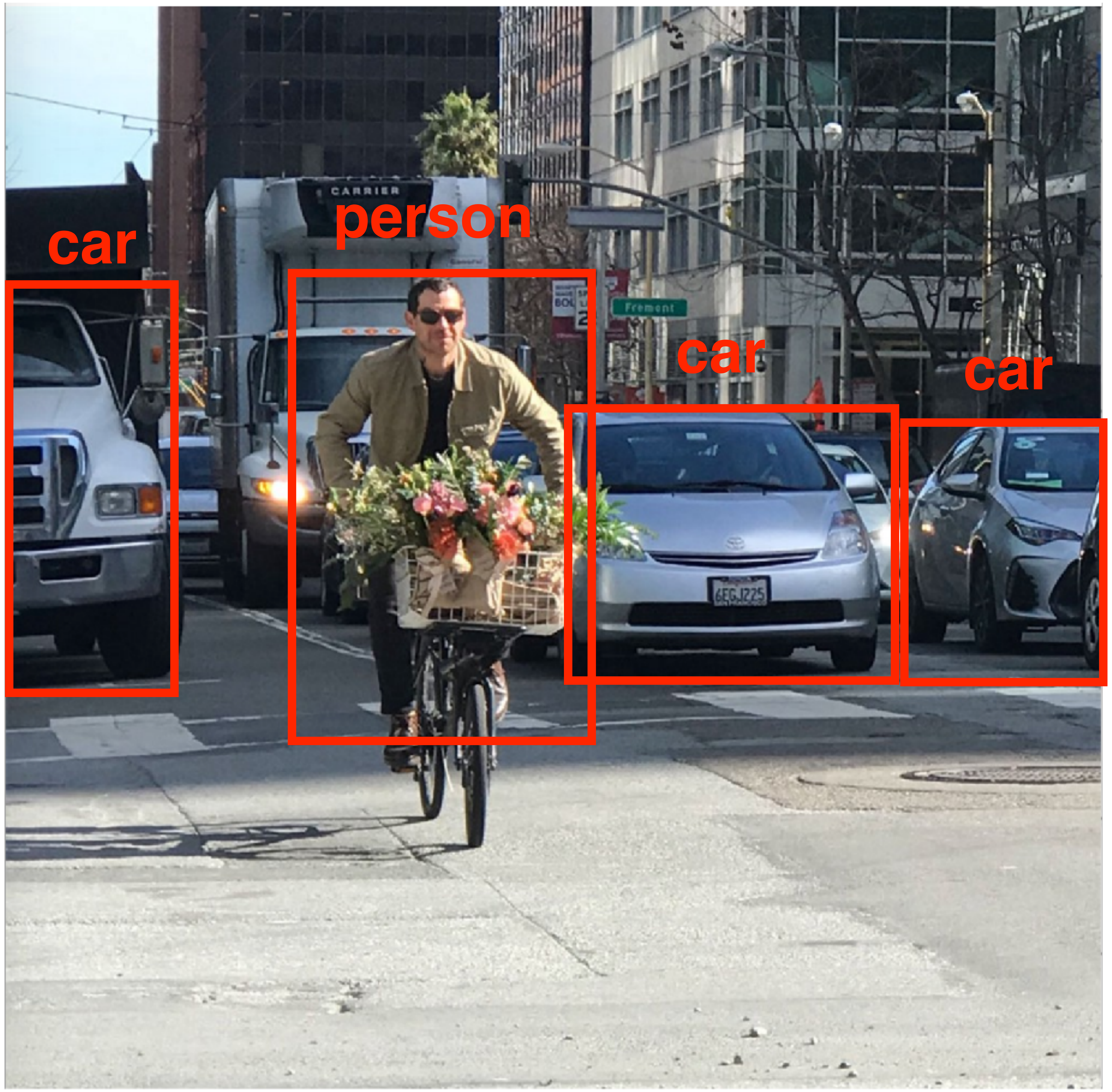
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of feature matching networks

Pascal VOC object detection challenge



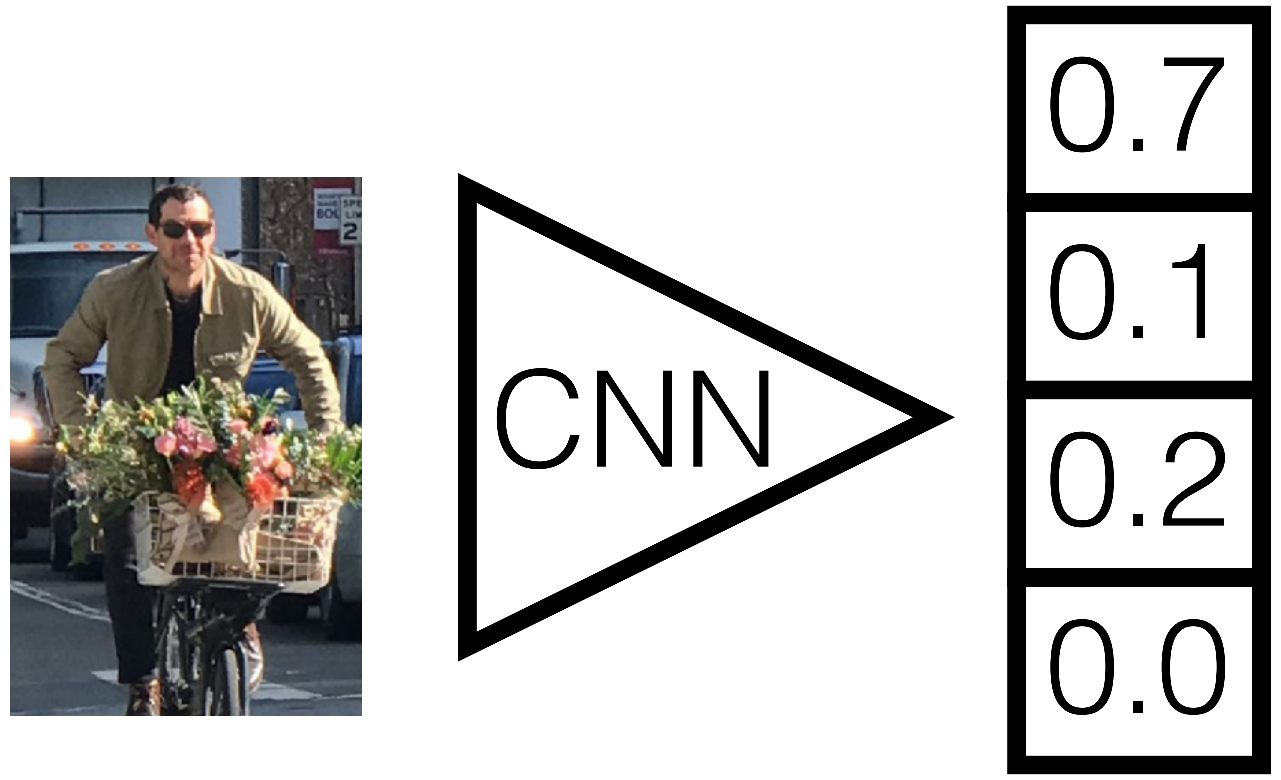
Object detection

Object detection



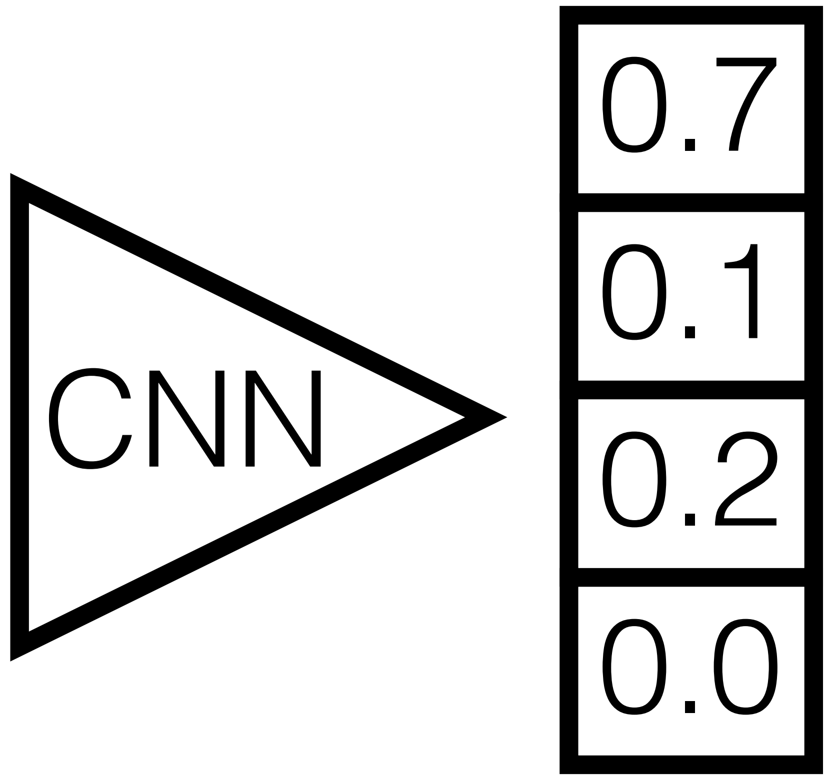
Object detection

Object detection

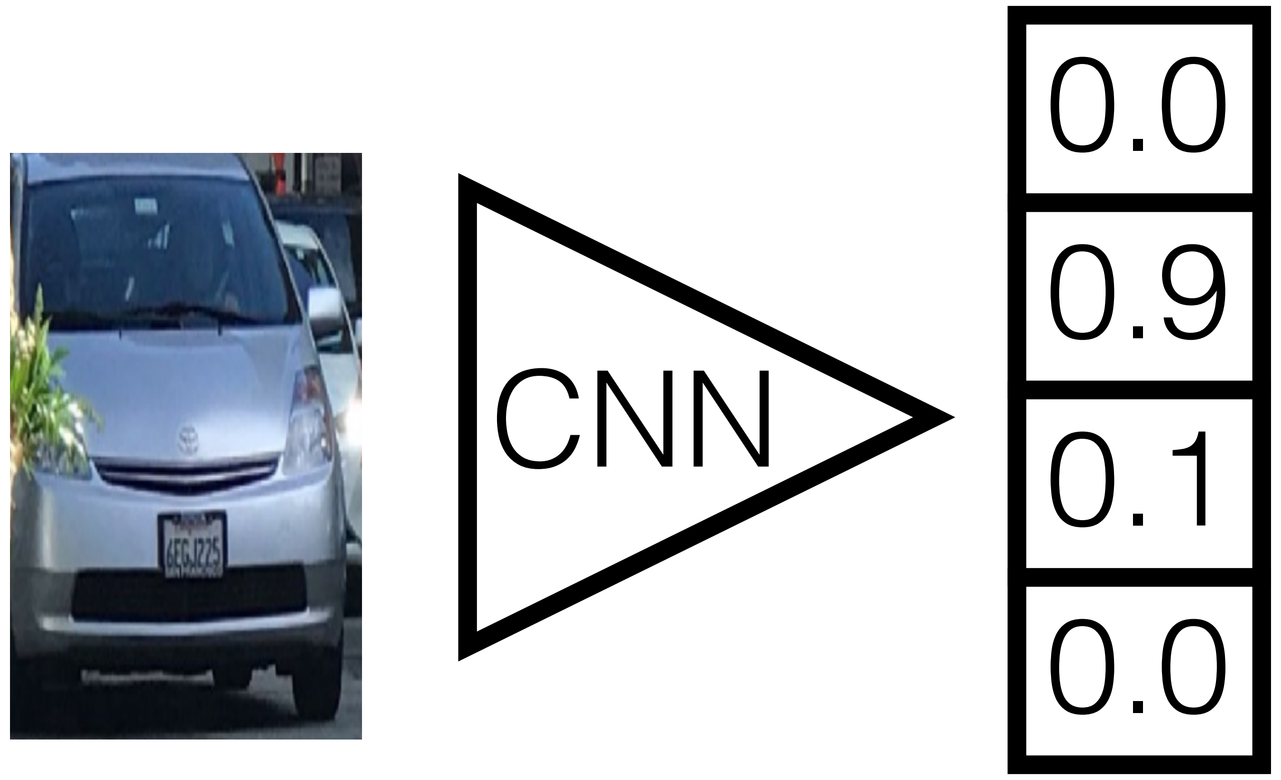


class: person

Object detection



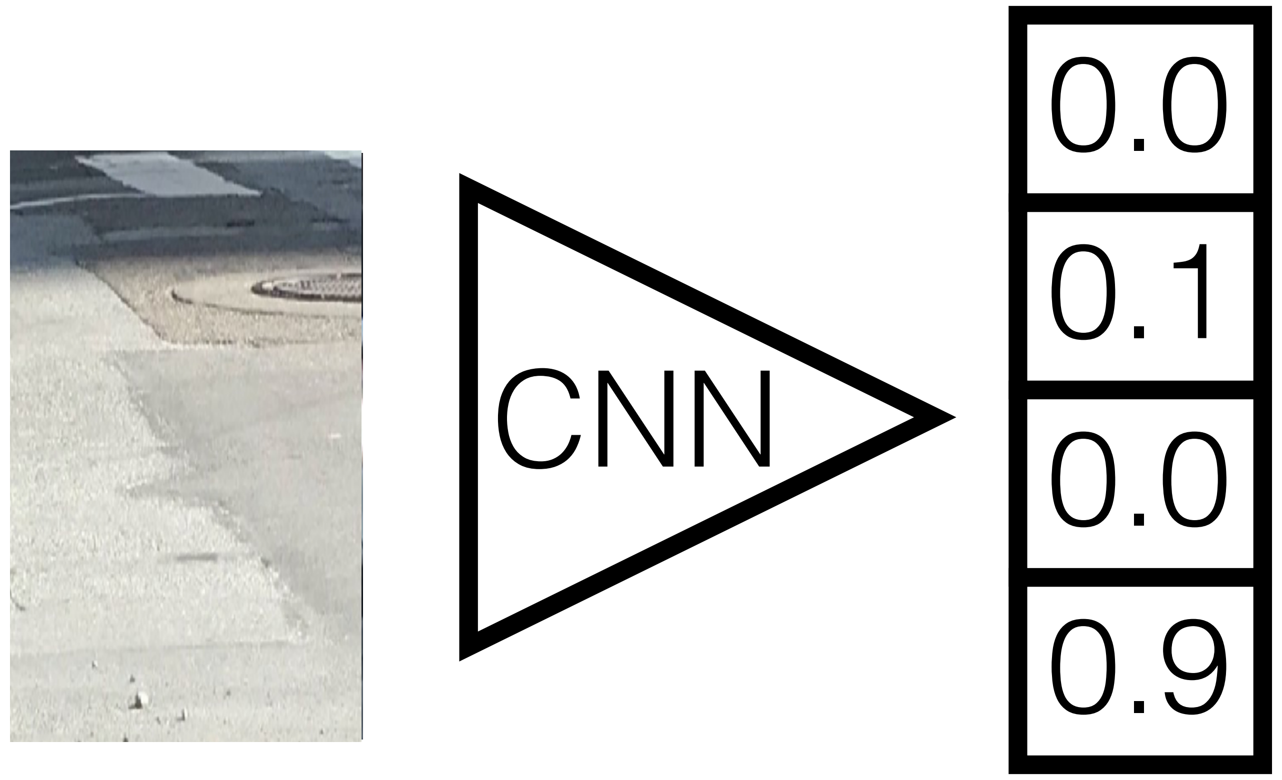
Object detection



class: car

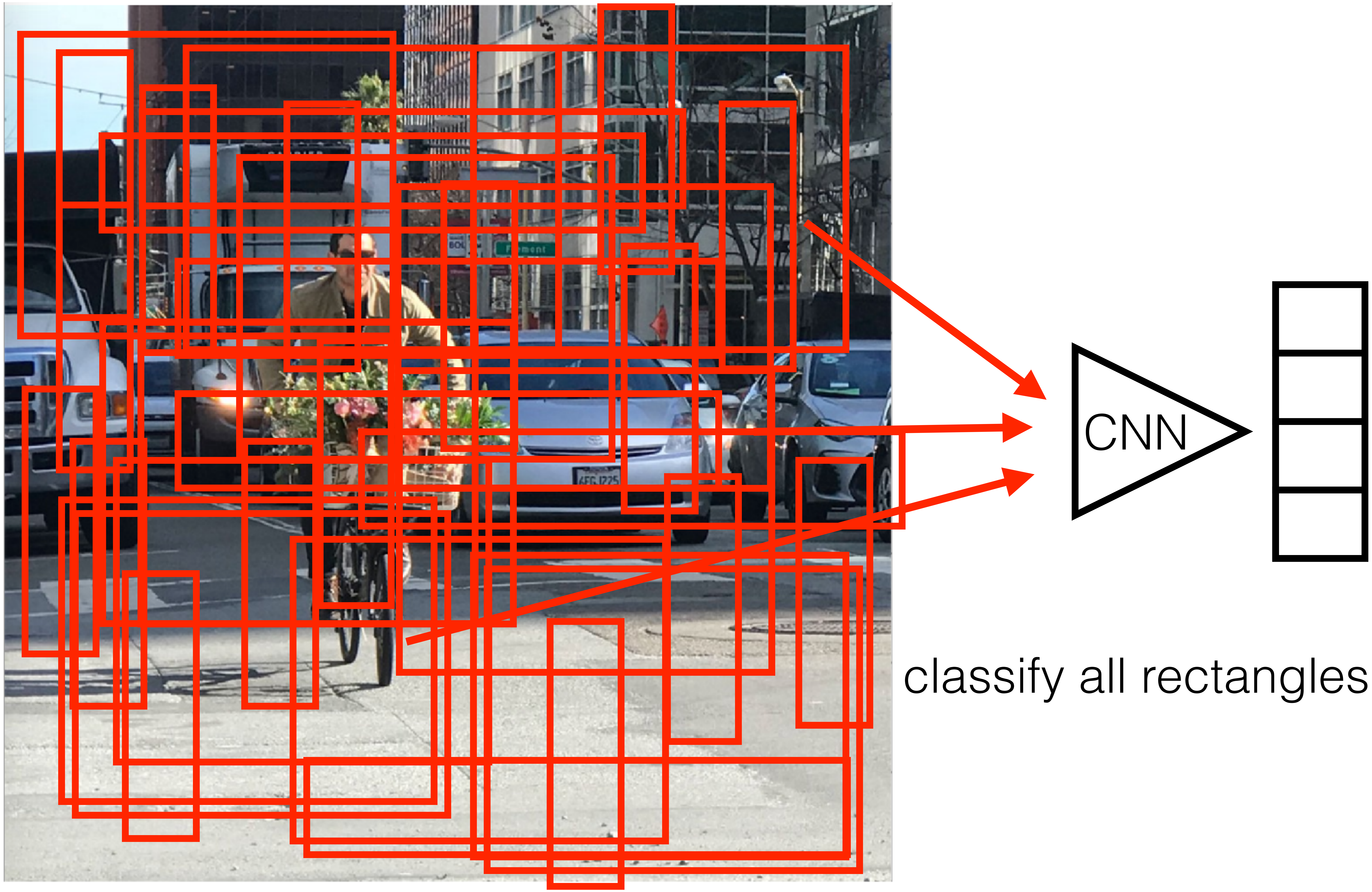
Object detection

Object detection



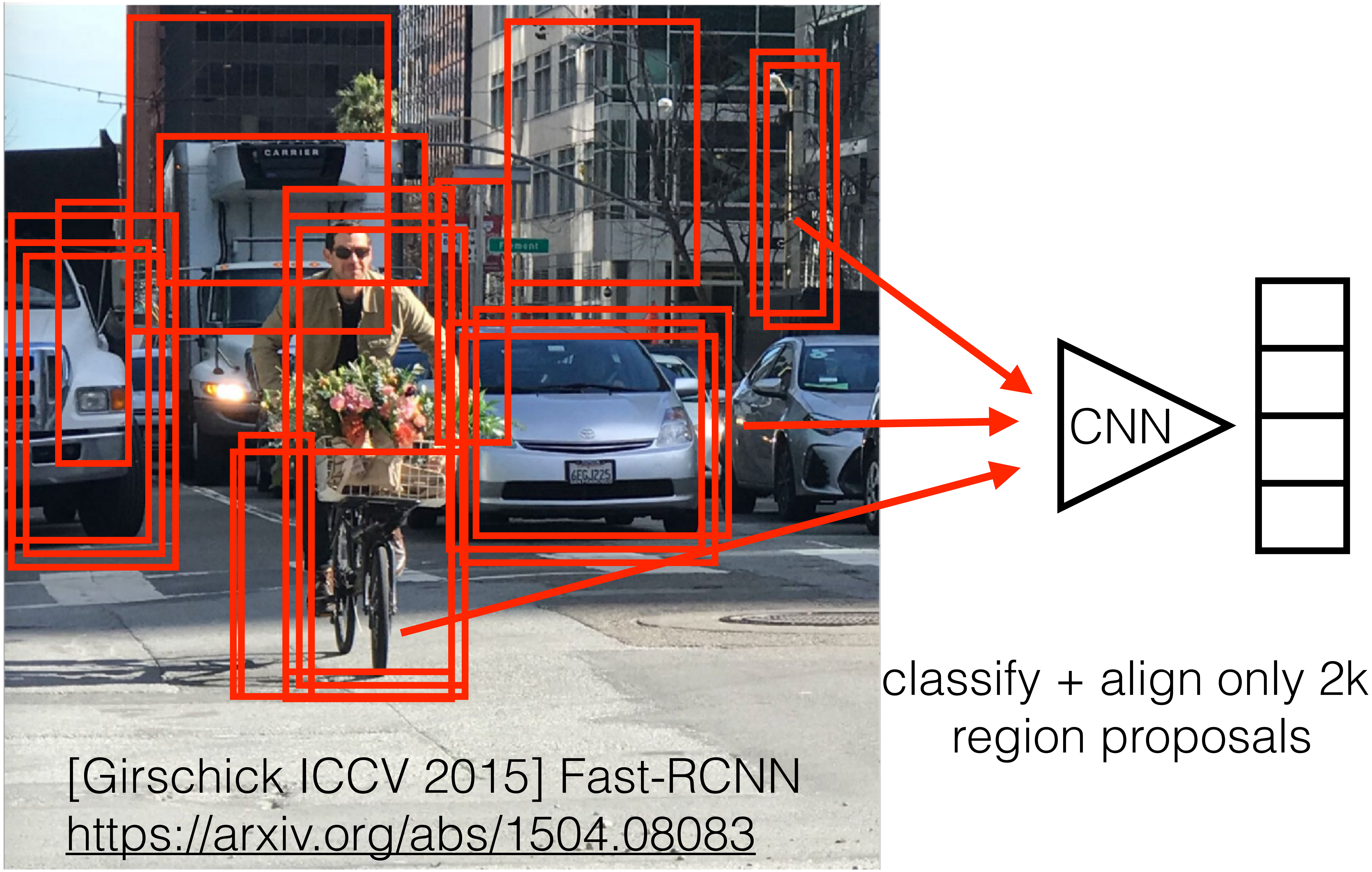
class: background

Object detection



- H x W x Aspect_Ratio x Scales x 0.001 sec = **months**

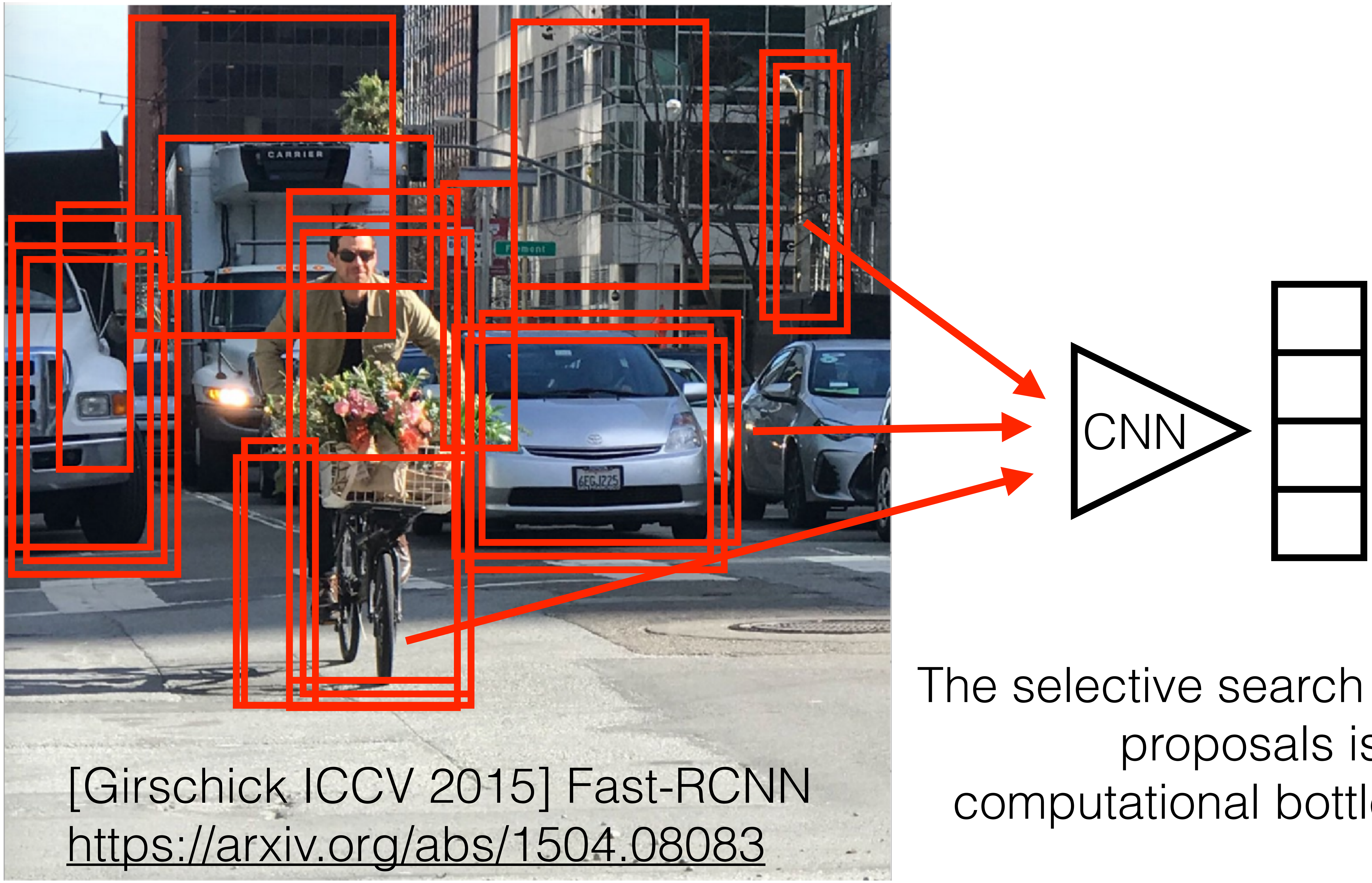
Object detection



[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083

classify + align only 2k region proposals

Object detection

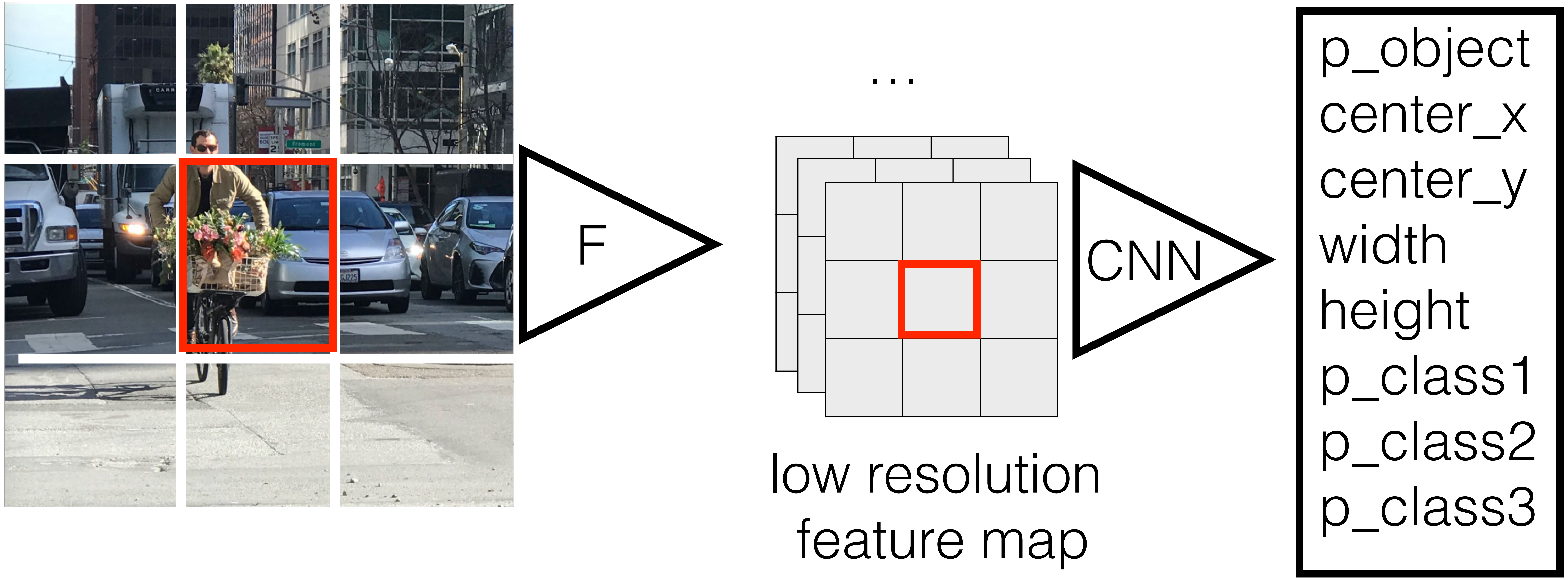


[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083

The selective search for region proposals is computational bottleneck !!!

- (find 2k cand.) + (2k cand. x 0.001 sec) = **47+2 sec = 49 sec**

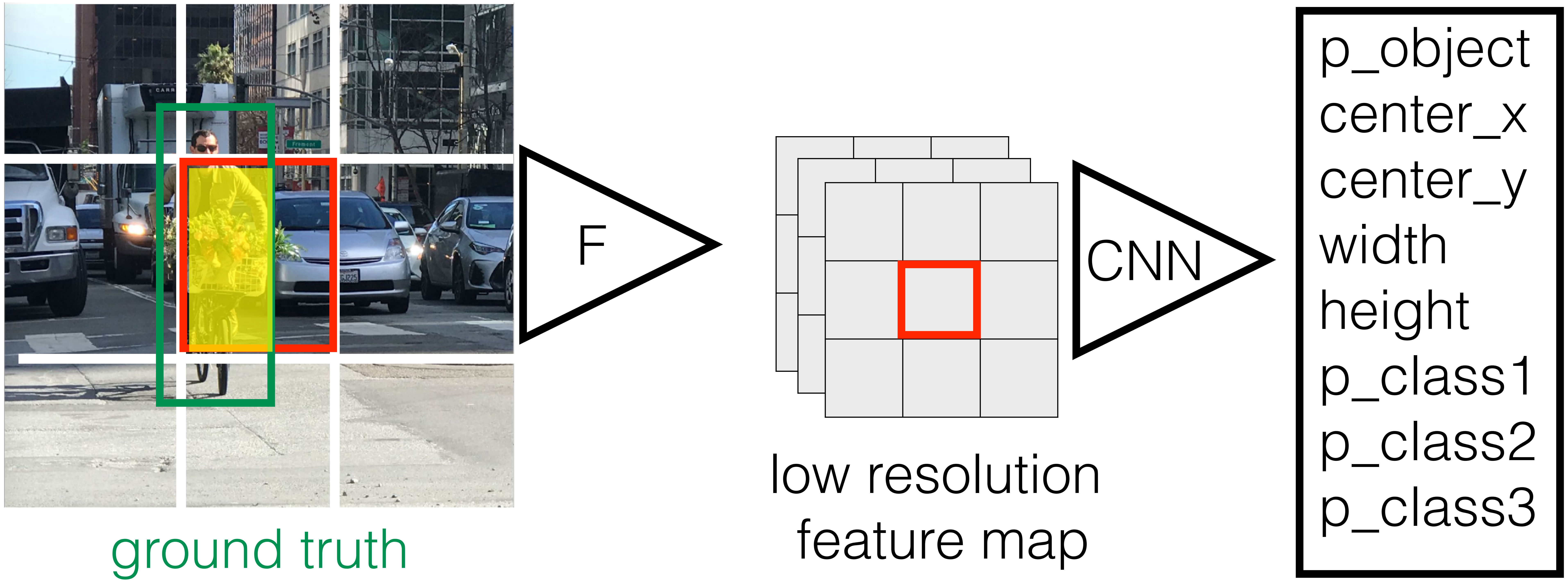
YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



- divide image into 3x3 sub images (corresponding to its receptive fields)
- predict relative position, objectness, class for each sub-im

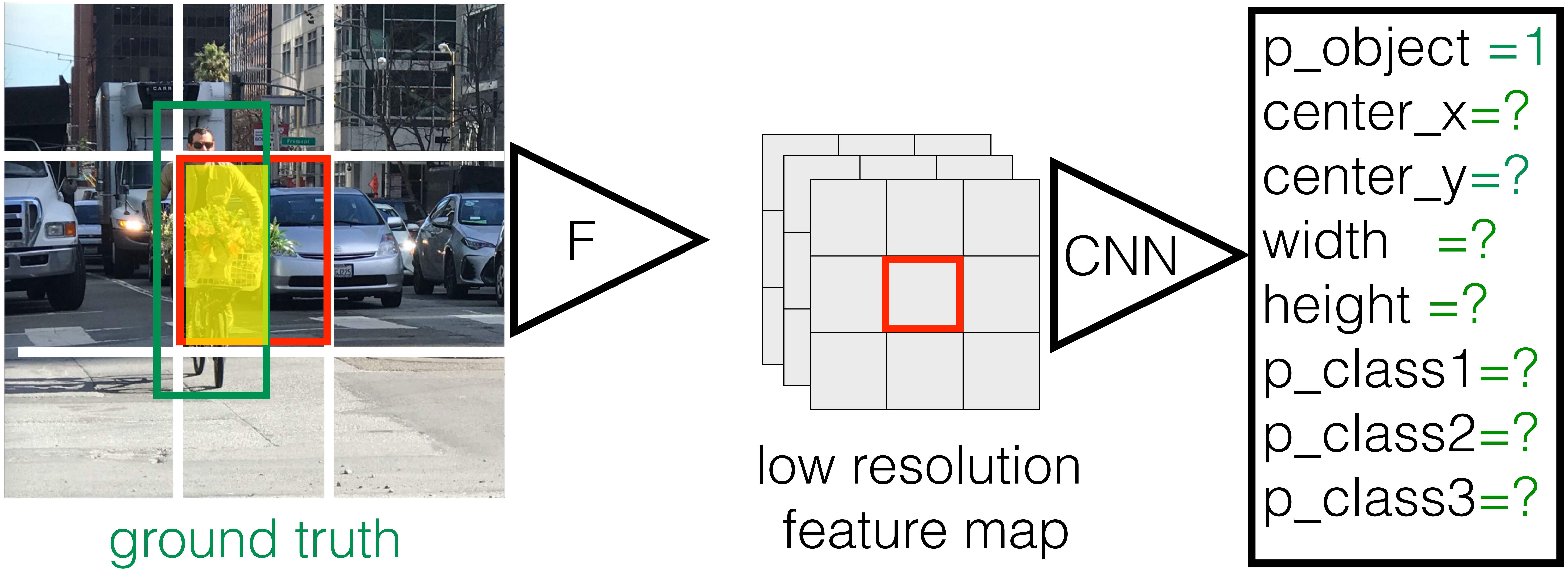
YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- learn from ground truth

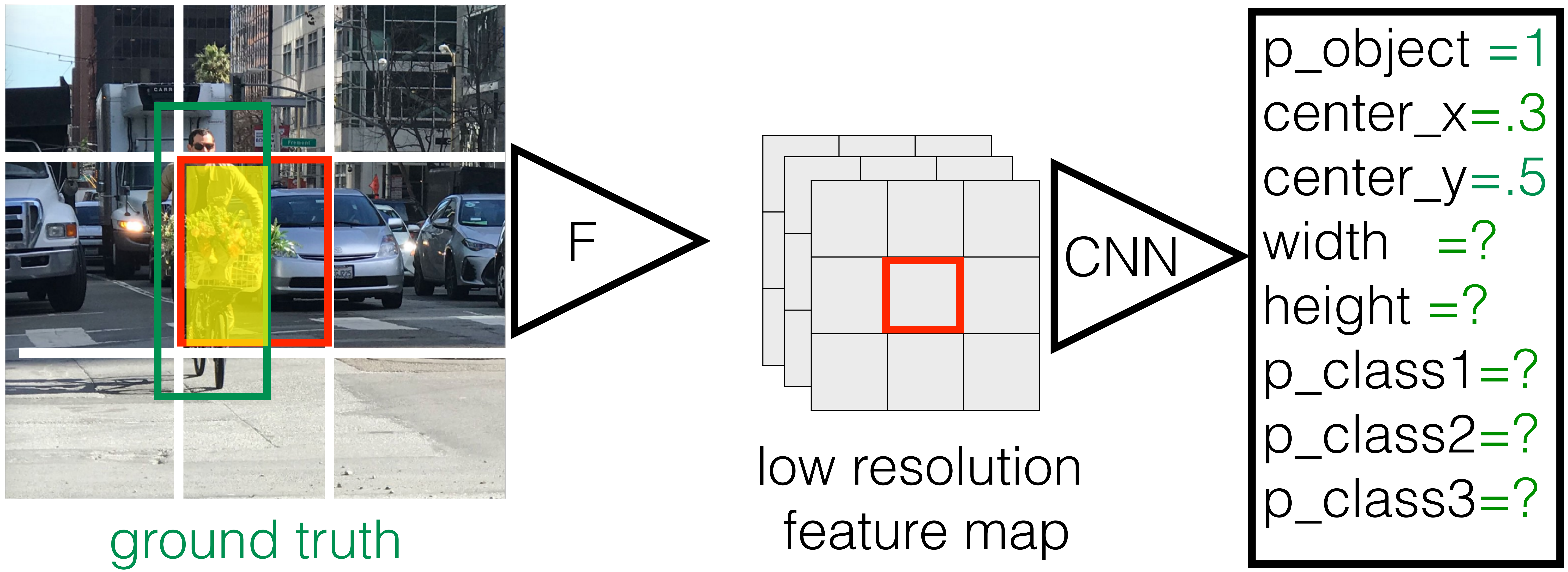
YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- learn from ground truth

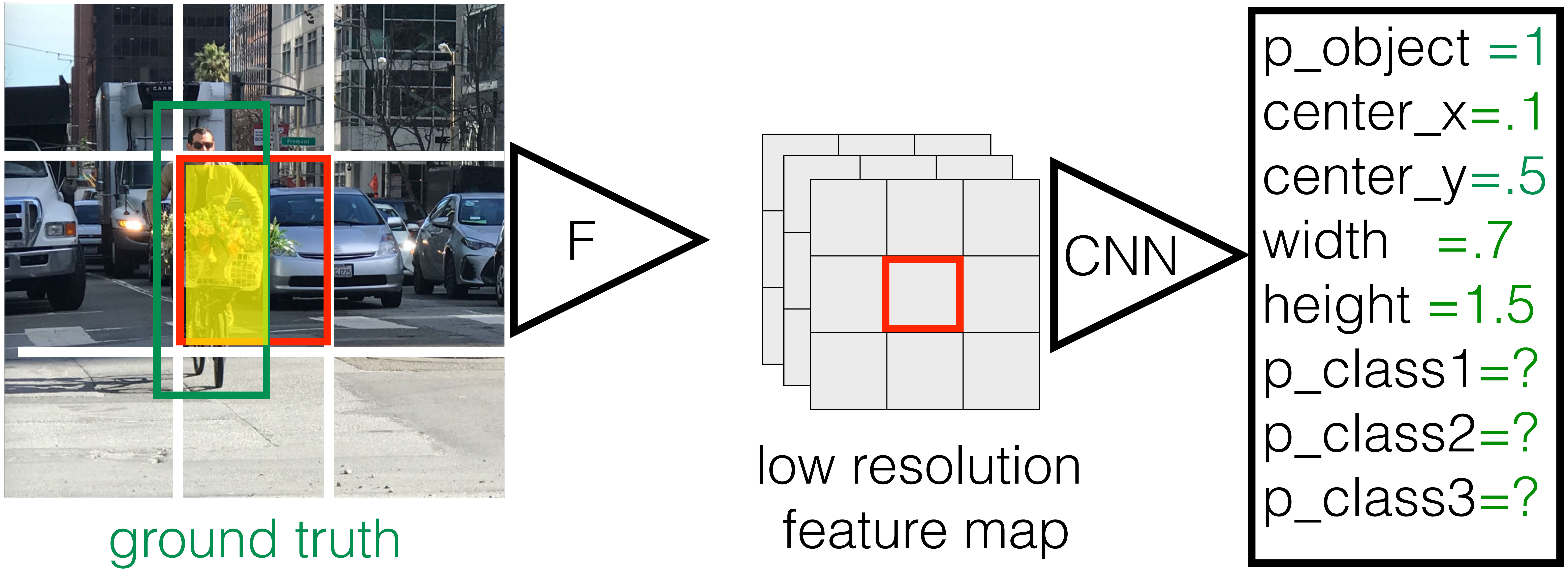
YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- learn from ground truth

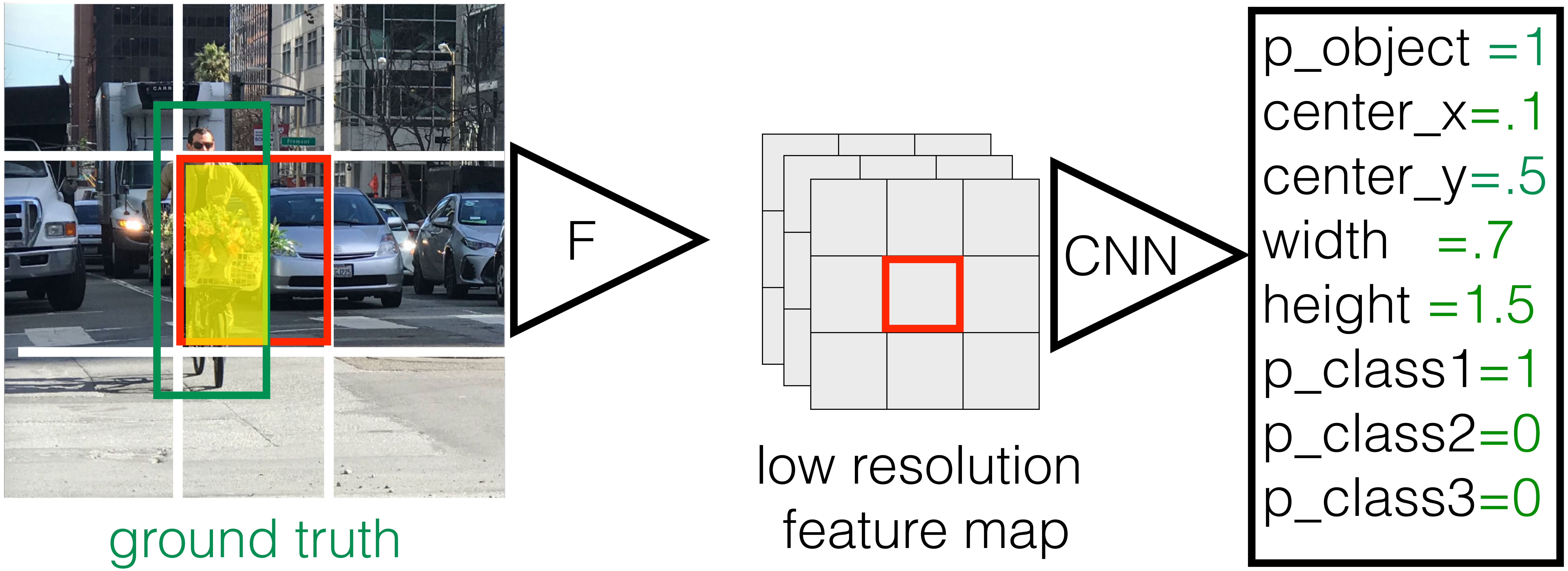
YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- learn from ground truth

YOLO and Faster RCNN architectures

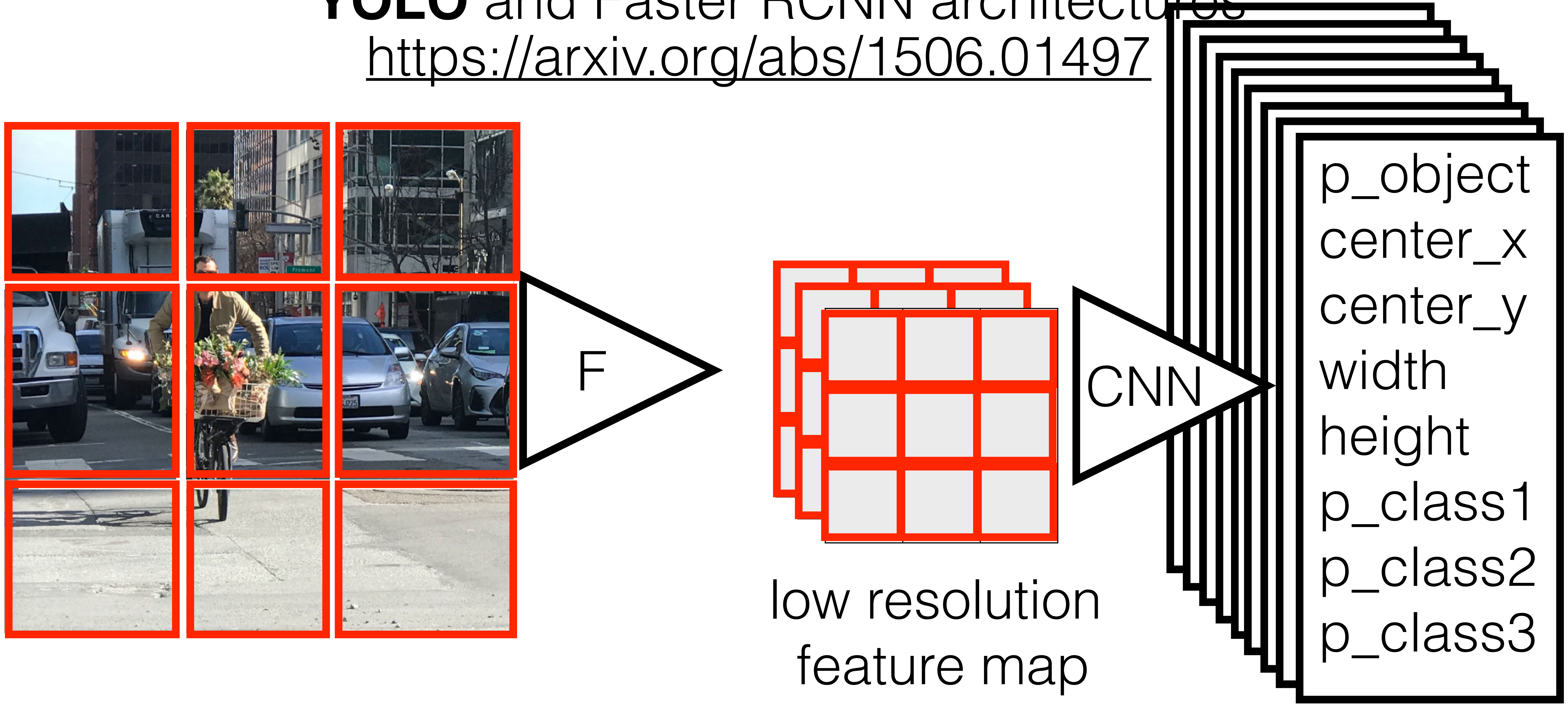
https://arxiv.org/abs/1506.01497



- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- learn from ground truth

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



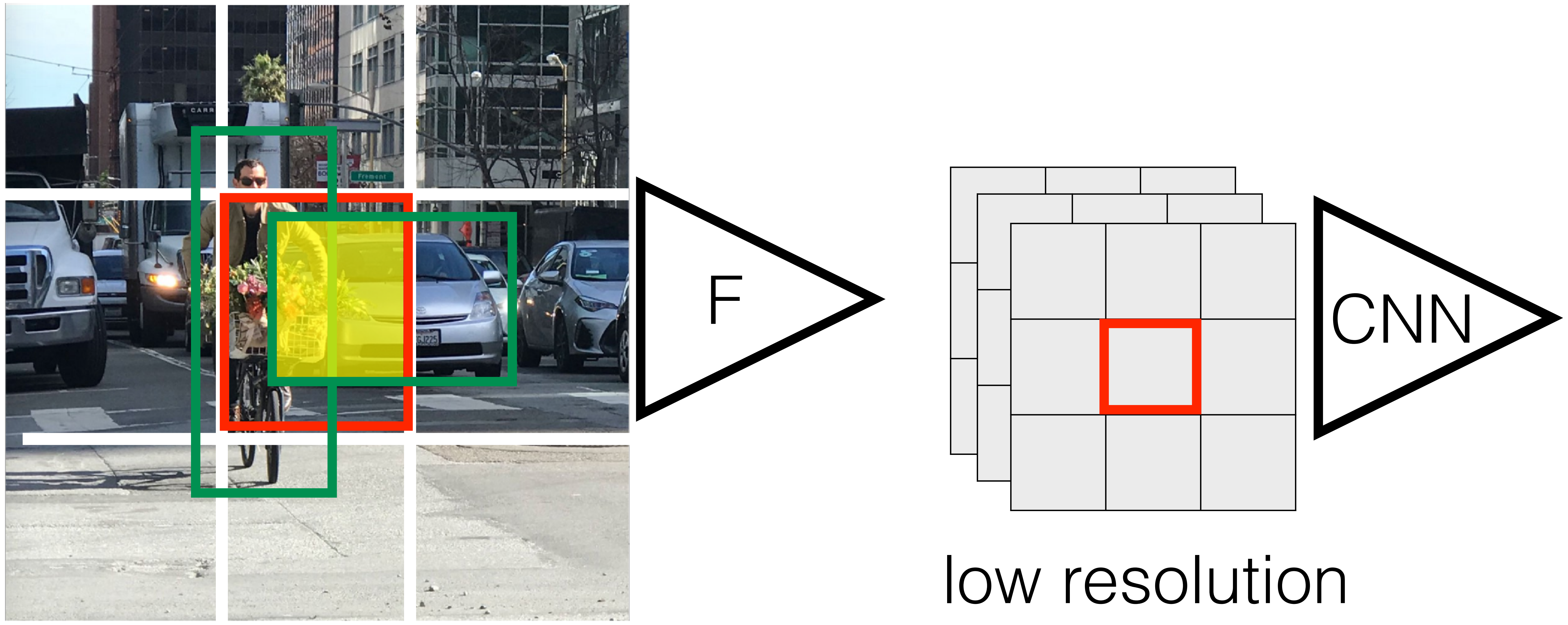
- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- each sub-image has its own output

Do you see any problem?

=> more obj in one sub-im

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



ground truth

low resolution feature map

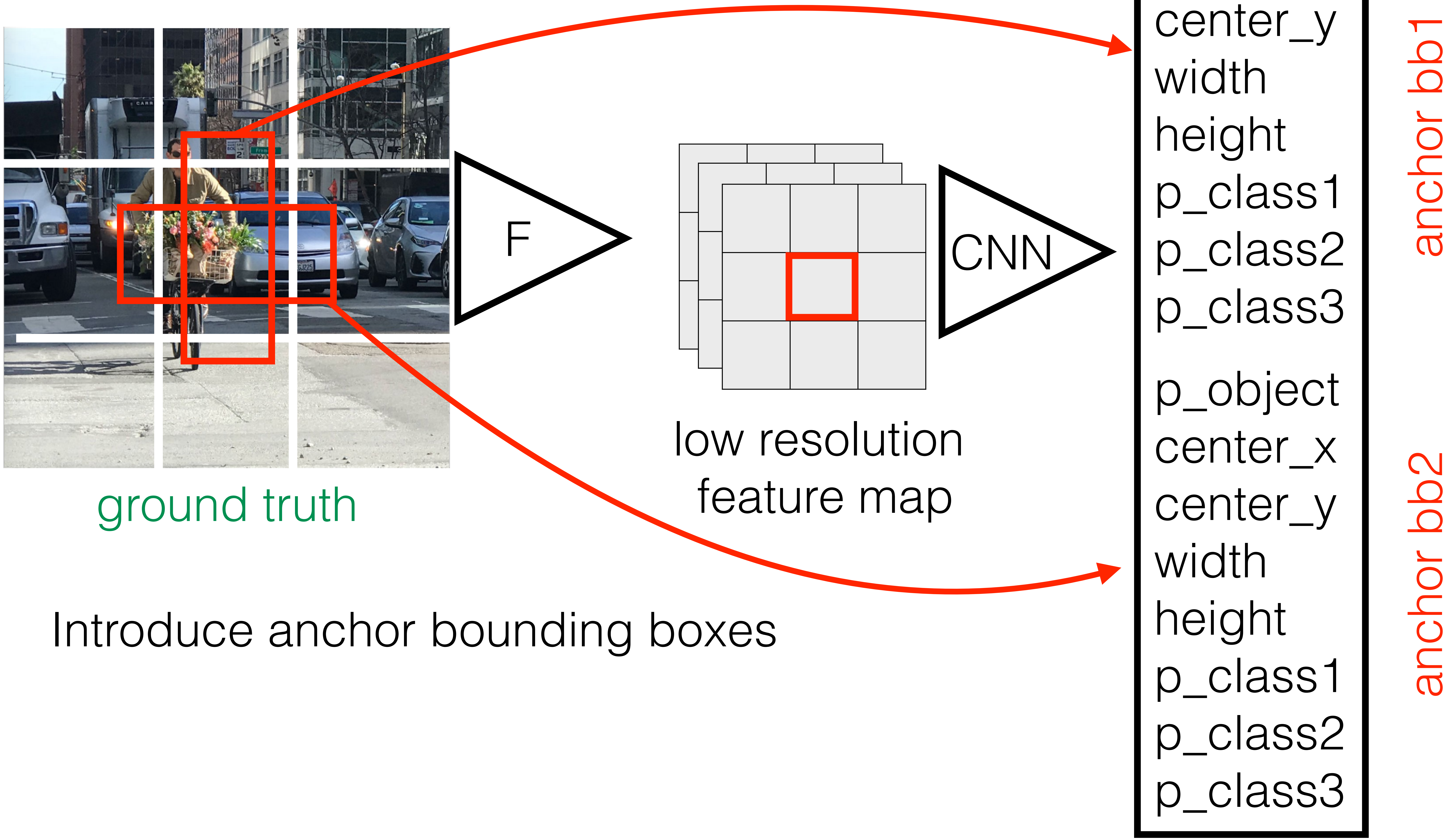
p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- each sub-image has its own output

Do you see any problem?

=> more obj in one sub-im

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



Introduce anchor bounding boxes

- Perform region proposal by CNN => **0.05-0.2 sec**

Object detection

- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**
- Instead we can use elementary signal processing method to extract only 2k viable candidates: [Girschick ICCV 2015], Fast-RCNN https://arxiv.org/abs/1504.08083
  (find 2k cand.) + (2k cand. x 0.001 sec) = **47+2 sec = 49 sec**
- Perform region proposal by CNN => **0.05-0.2 sec**

[Faster RCNN 2017] https://arxiv.org/abs/1506.01497 (slower, works for smaller objs)

[Redmont CVPR 2018], https://arxiv.org/abs/1804.02767 (faster, small obj. problems)

How to report classifier quality?

Binary classifier testing presence of potentially dangerous case:

Positive class

GT CARS

Negative class

GT BKGD:

Binary classifier testing presence of potentially dangerous case:

Positive class

GT
CARS

CLS
CARS

Negative class

GT
BKGD:

CLS
BGGD:

Binary classifier testing presence of potentially dangerous case:

Positive class

GT
CARS

CLS
CARS

Negative class

GT
BKGD:

CLS
BGGD:

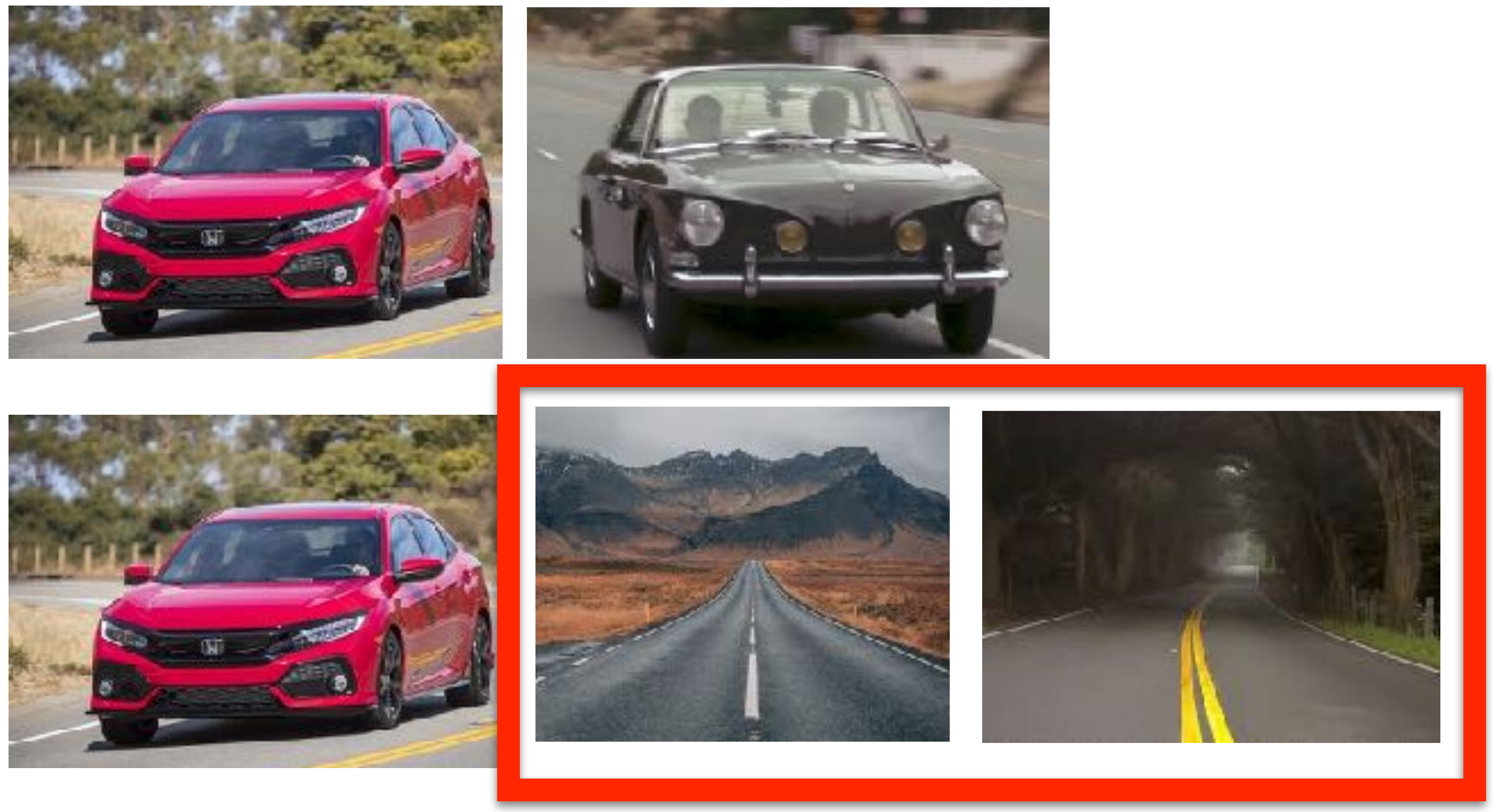
false negative (FN)… classifier **falsely** indicates positive class (e.g. car) as a **negative** class => missed danger

Binary classifier testing presence of potentially dangerous case:

Positive class

GT
CARS

CLS
CARS



Negative class

GT
BKGD:

CLS
BGGD:

false negative (FN) … classifier **falsely** indicates positive class (e.g. car)
as a **negative** class => missed danger

false positive (FP) … classifier **falsely** indicates negative class (e.g. background)
as a **positive** class => false alarm

Binary classifier testing presence of potentially dangerous case:

Positive class

GT CARS

CLS CARS

Negative class

GT BKGD:

CLS BGGD:

false negative (FN) … classifier **falsely** indicates positive class (e.g. car) as a **negative** class => missed danger

false positive (FP) … classifier **falsely** indicates negative class (e.g. background) as a **positive** class => false alarm

true positive (TP) … classifier correctly indicate ground **truth** positive class (e.g. car) as a **positive** class => correctly found danger

Binary classifier testing presence of potentially dangerous case:

Positive class

GT CARS


CLS CARS


Negative class

GT BKGD:


CLS BGGD:


false negative (FN)… classifier **falsely** indicates positive class (e.g. car) as a **negative** class => missed danger

false positive (FP) … classifier **falsely** indicates negative class (e.g. background) as a **positive** class => false alarm

true positive (TP) … classifier correctly indicate ground **truth** positive class (e.g. car) as a **positive** class => correctly found danger

true negative (TN) … classifier correctly indicate ground **truth** negative class (e.g. bckg) as a **negative** class => correctly found safety

Binary classifier testing presence of potentially dangerous case:

Positive class | Negative class

GT CARS

GT BKGD:

CLS CARS

CLS BGGD:

false negative (FN) =1

false positive (FP) =2

true positive (TP) =1

true negative (TN) =2

“1/3 of of samples classified as CARS are actually CARS”

$$\text{Precision (P)} = \frac{TP}{TP + FP} = \frac{1}{1 + 2} = 1/3$$

What is their meaning?

$$\text{Recall (R)} = \frac{TP}{TP + FN} = \frac{1}{1 + 1} = 1/2$$

“1/2 of all CARS is discovered”

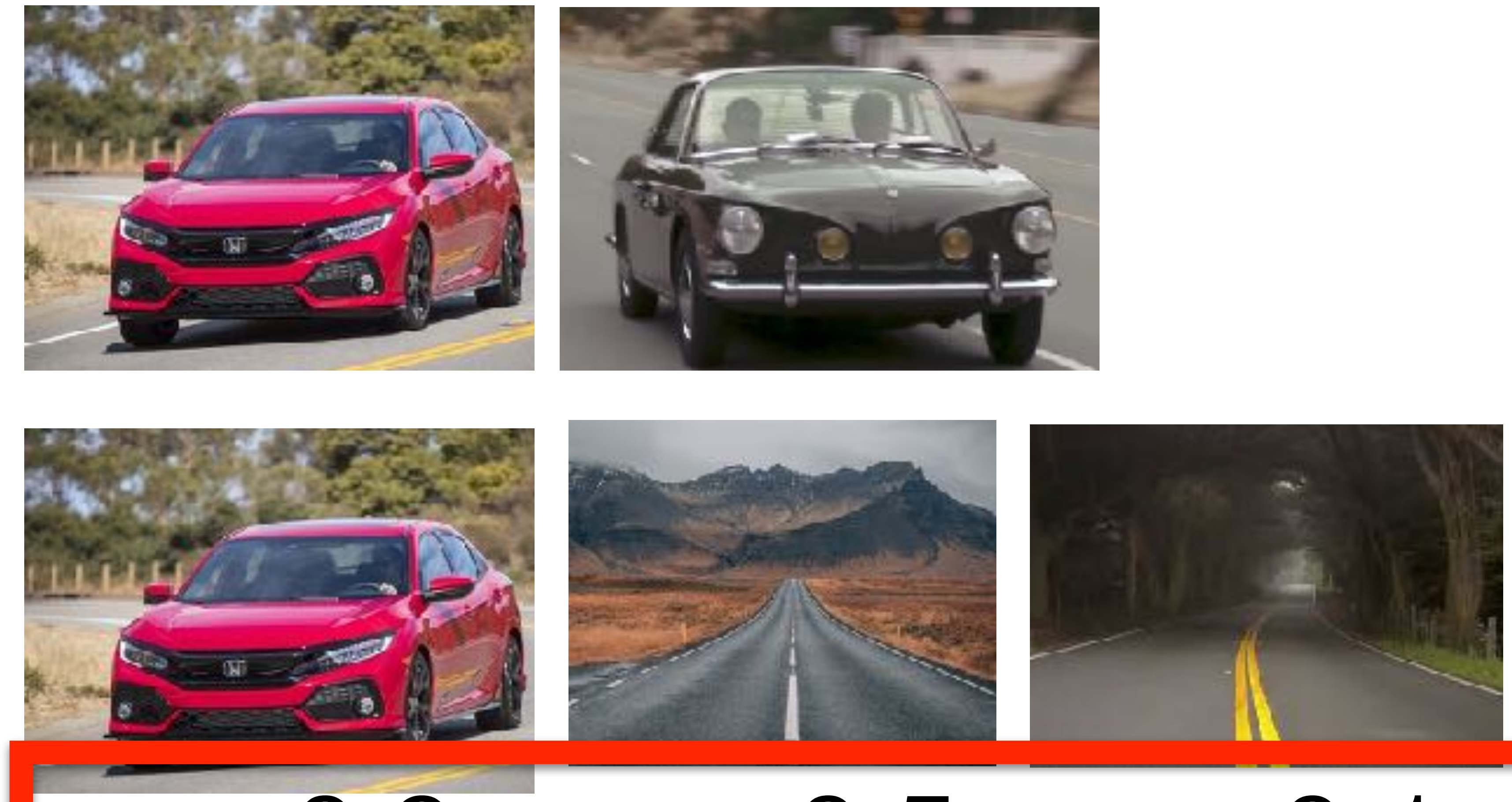
What is best classifier? Oracle: Precision = Recall = 1

Binary classifier testing presence of potentially dangerous case:

Positive class

GT CARS

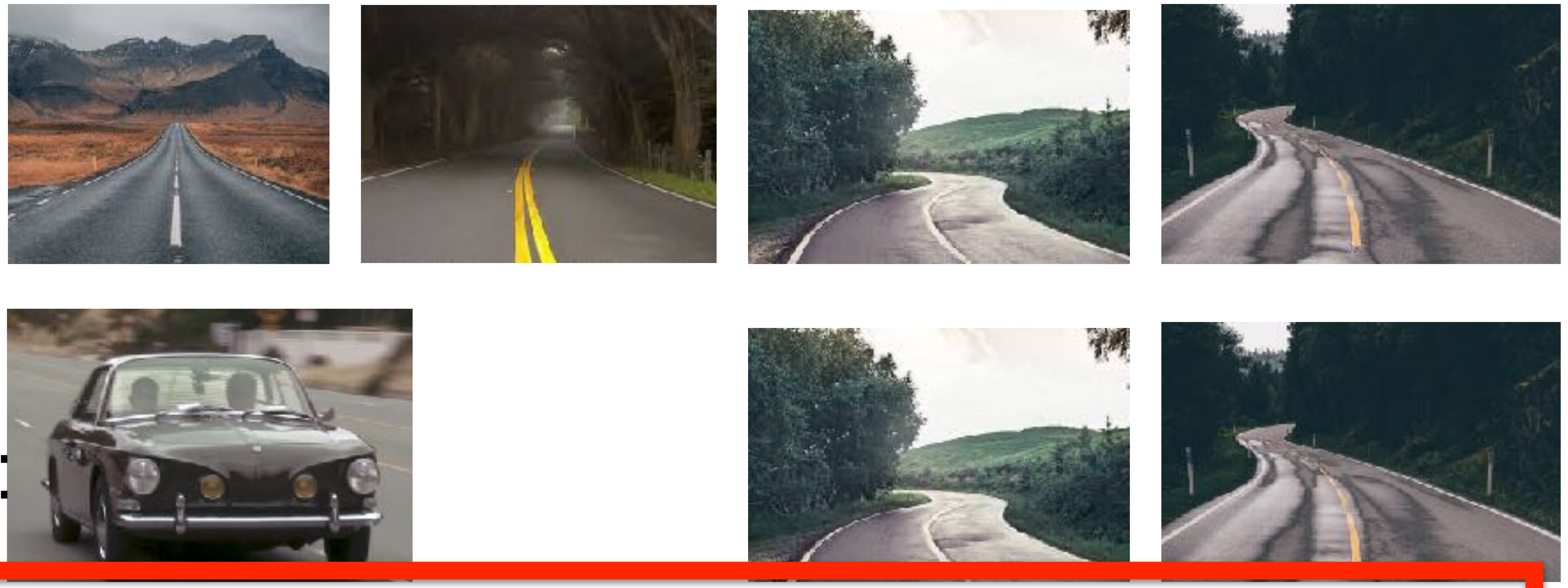
CLS CARS



Negative class

GT BKGD:

CLS BGGD:



0.9 0.5 0.1 -0.1 -0.4 -0.6

false negative (FN) =1

false positive (FP) =2

true positive (TP) =1

true negative (TN) =2

$$\text{Precision (P)} = \frac{TP}{TP + FP} = \frac{1}{1 + 2} = 1/3$$

$$\text{Recall (R)} = \frac{TP}{TP + FN} = \frac{1}{1 + 1} = 1/2$$

Oracle: Precision = Recall = 1

Binary classifier testing presence of potentially dangerous case:

Positive class

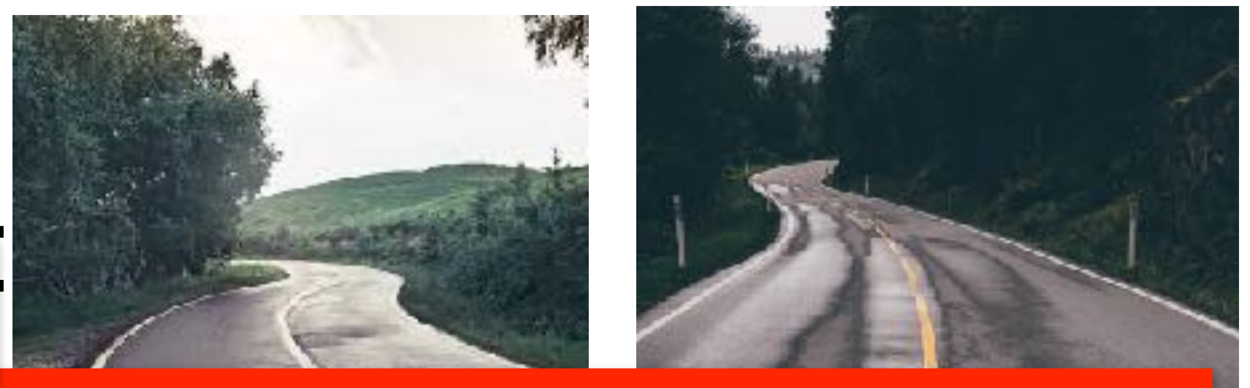
Negative class

GT CARS

GT BKGD:

CLS CARS

CLS BGGD:



0.9 0.5 0.1 -0.1 -0.4 -0.6

false negative (FN) =0

false positive (FP) =2

true positive (TP) =2

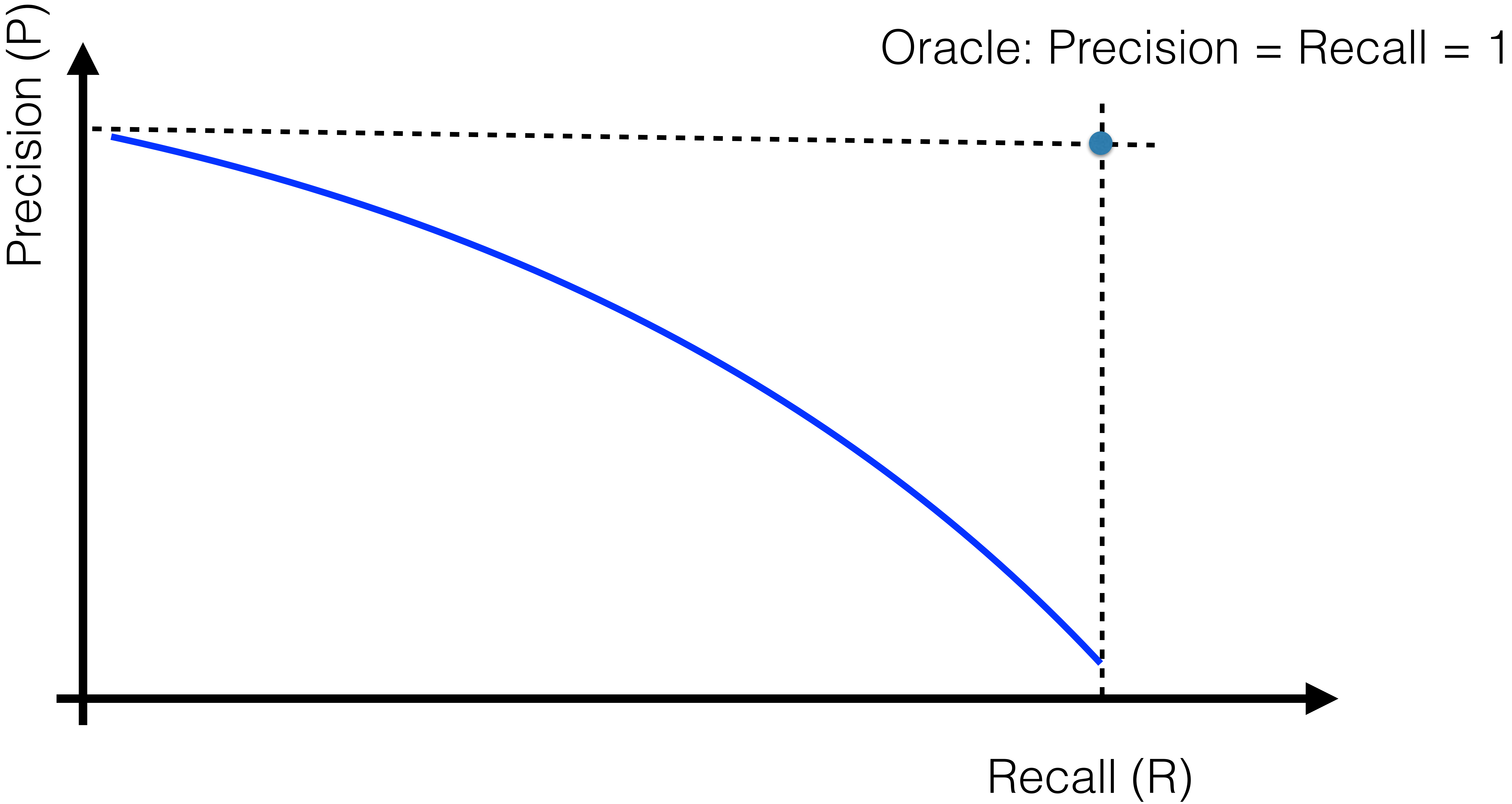
true negative (TN) =2

$$\text{Precision (P)} = \frac{TP}{TP + FP} = \frac{2}{2 + 2} = 1/2$$

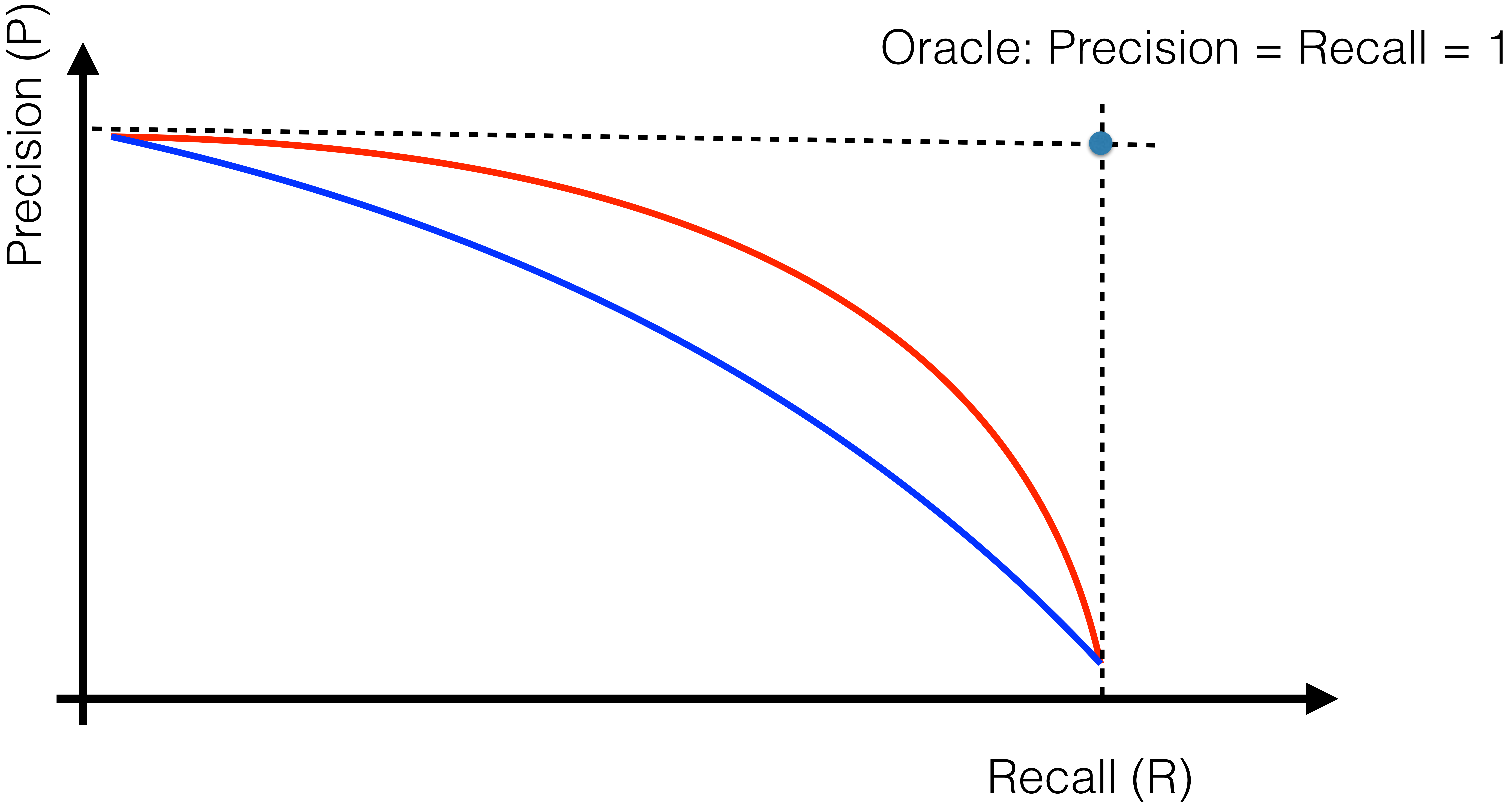
$$\text{Recall (R)} = \frac{TP}{TP + FN} = \frac{2}{2 + 0} = 1$$

Oracle: Precision = Recall = 1

Smoothed Precision-Recall curve



Smoothed Precision-Recall curve



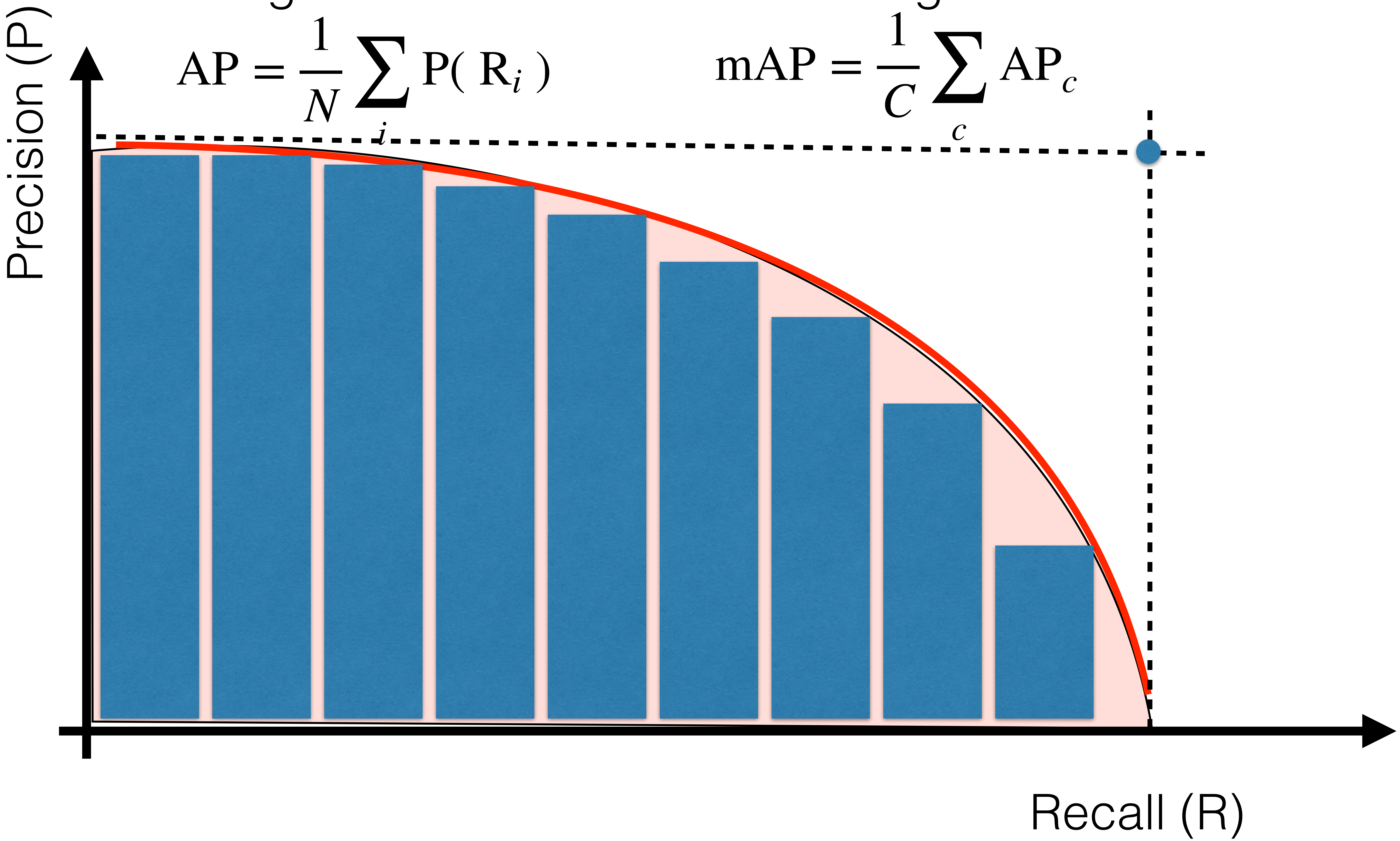
Smoothed Precision-Recall curve

Average Precission

$$\mathrm{AP} = \frac{1}{N}\sum_{i} \mathrm{P}(\,\mathrm{R}_i\,)$$

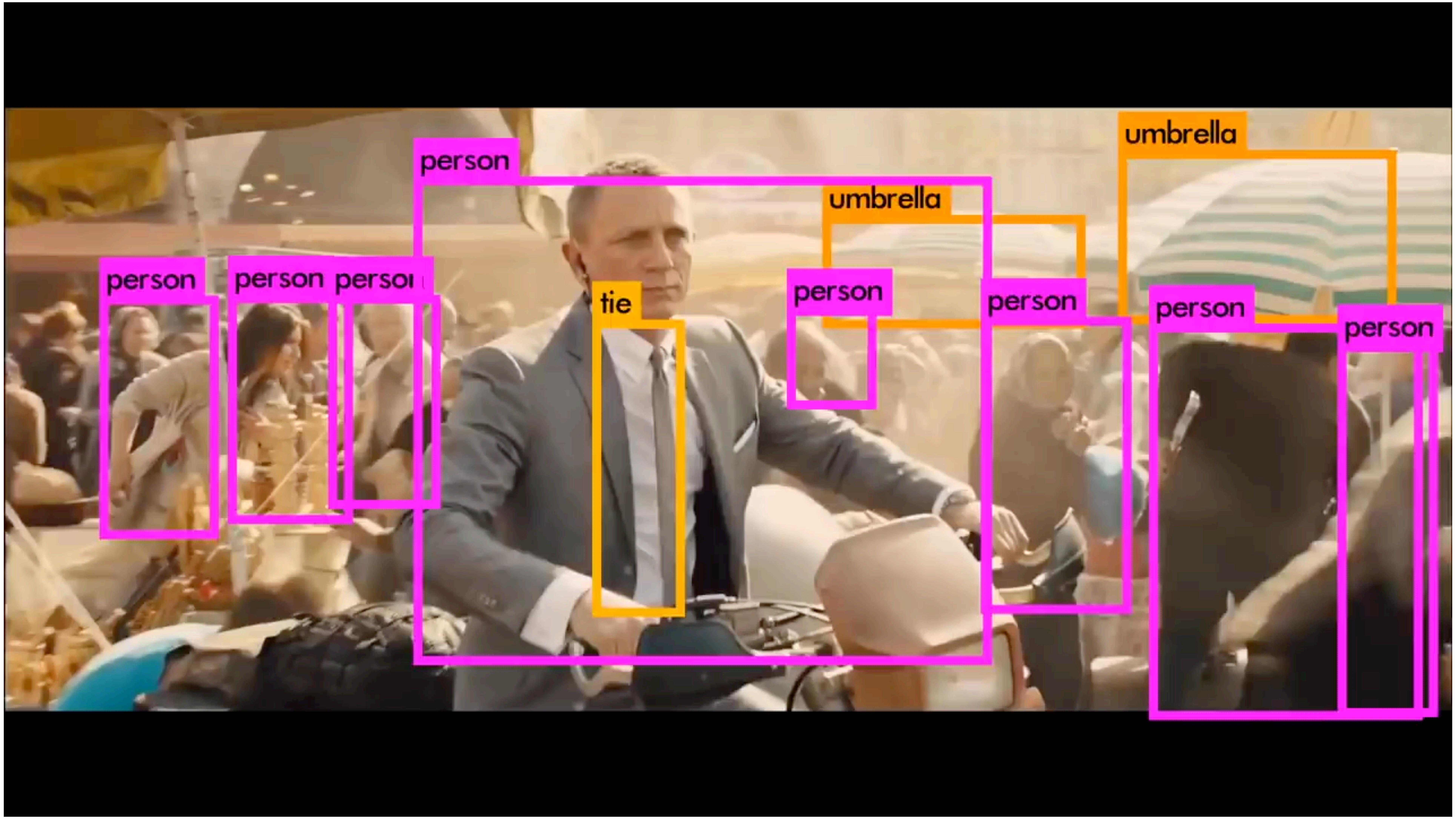
mean Average Precission

$$\mathrm{mAP} = \frac{1}{C}\sum_{c} \mathrm{AP}_c$$




YOLOv3
RetinaNet-50
RetinaNet-101
COCO mAP-50
inference time (ms)
58
56
54
52
50
48
50
100
150
200
250
B
C
D
E
F
G
Method mAP-50 time
[B] SSD321 45.4 61
[C] DSSD321 46.1 85
[D] R-FCN 51.9 85
[E] SSD513 50.4 125
[F] DSSD513 53.3 156
[G] FPN FRCN 59.1 172
RetinaNet-50-500 50.9 73
RetinaNet-101-500 53.1 90
RetinaNet-101-800 57.5 198
YOLOv3-320 51.5 22
YOLOv3-416 55.3 29
YOLOv3-608 57.9 51

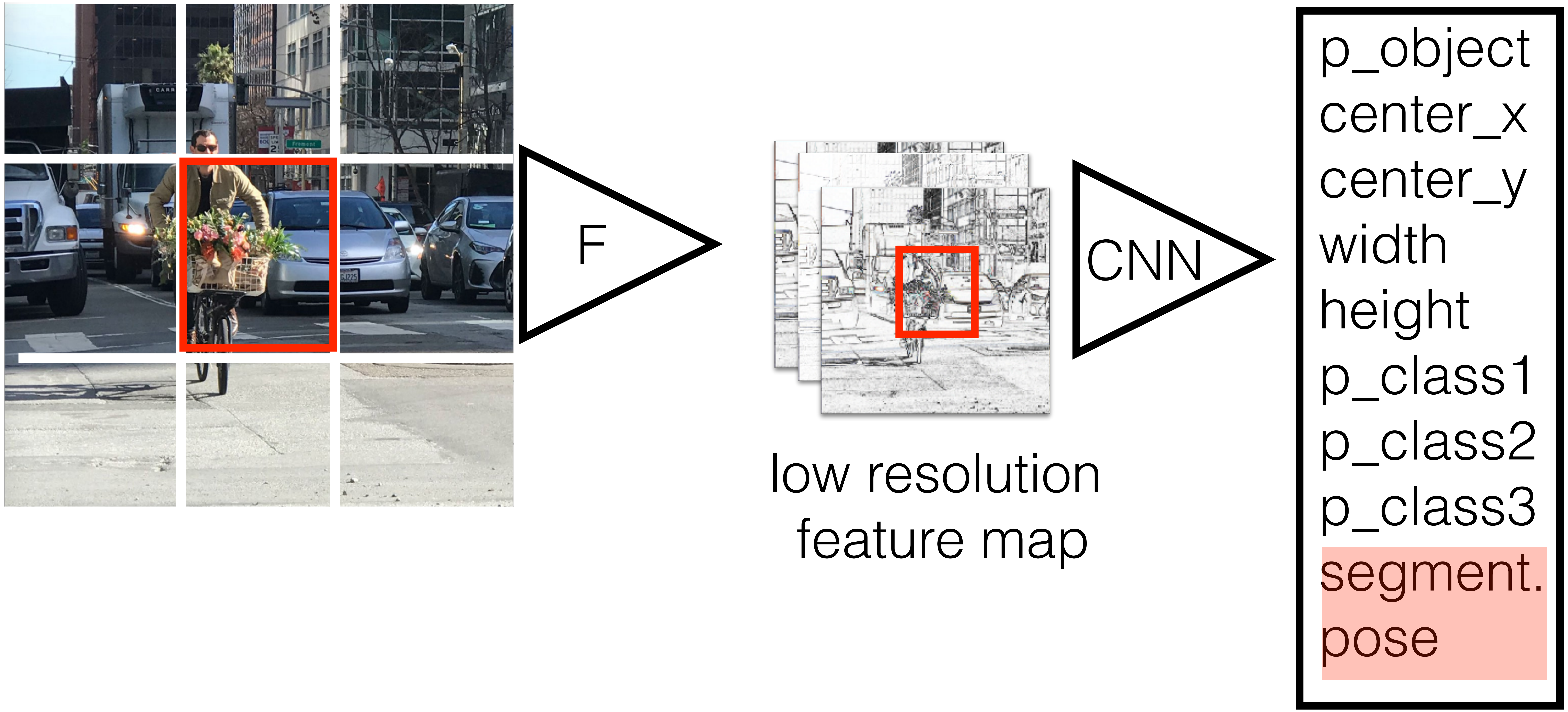
Deep convolutional - object detection



[Redmont CVPR 2018], https://arxiv.org/abs/1804.02767
code: https://pjreddie.com/darknet/yolo/

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



[He et al CVPR 2017] Mask-RCNN

https://arxiv.org/abs/1703.06870

Mask RCNN - results

[He et al CVPR 2017] Mask-RCNN
https://arxiv.org/abs/1703.06870

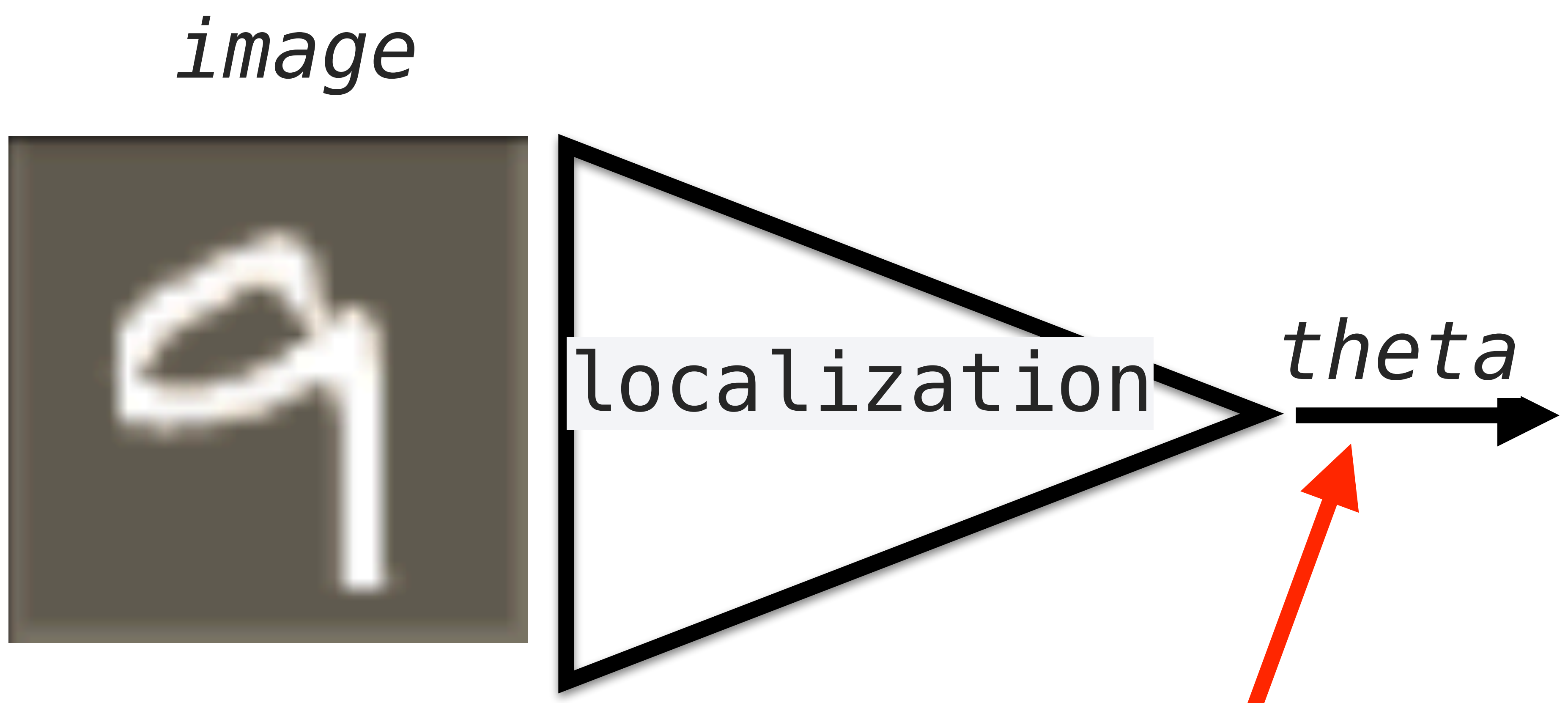
Mask RCNN - results

Outline

- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Spatial Transformer networks
- Architectures of feature matching networks

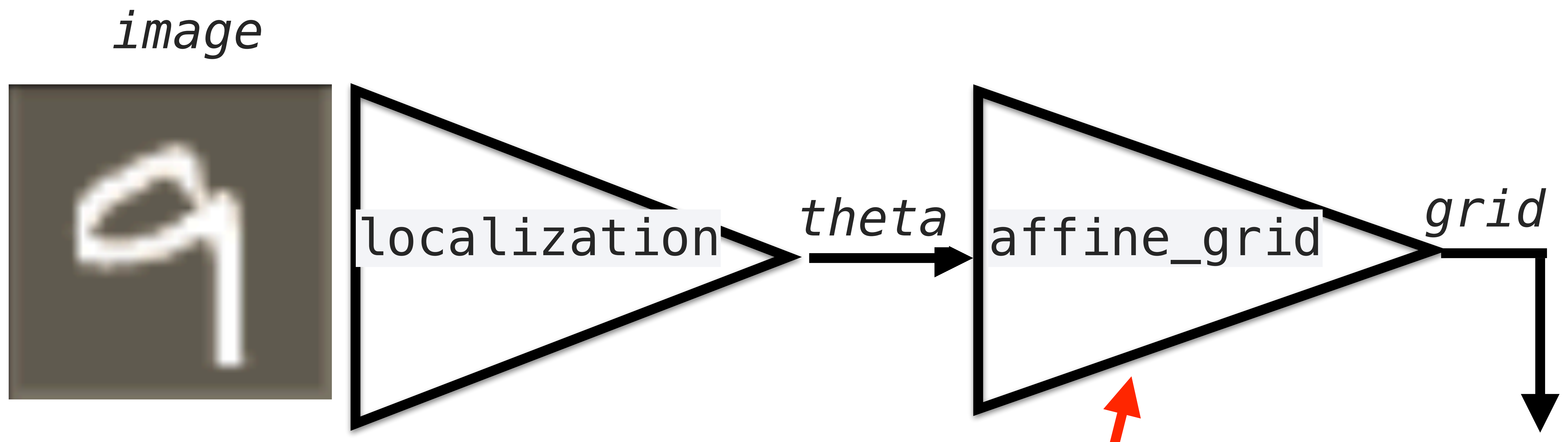
Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf



estimate parameters of 2D similarity transformation

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

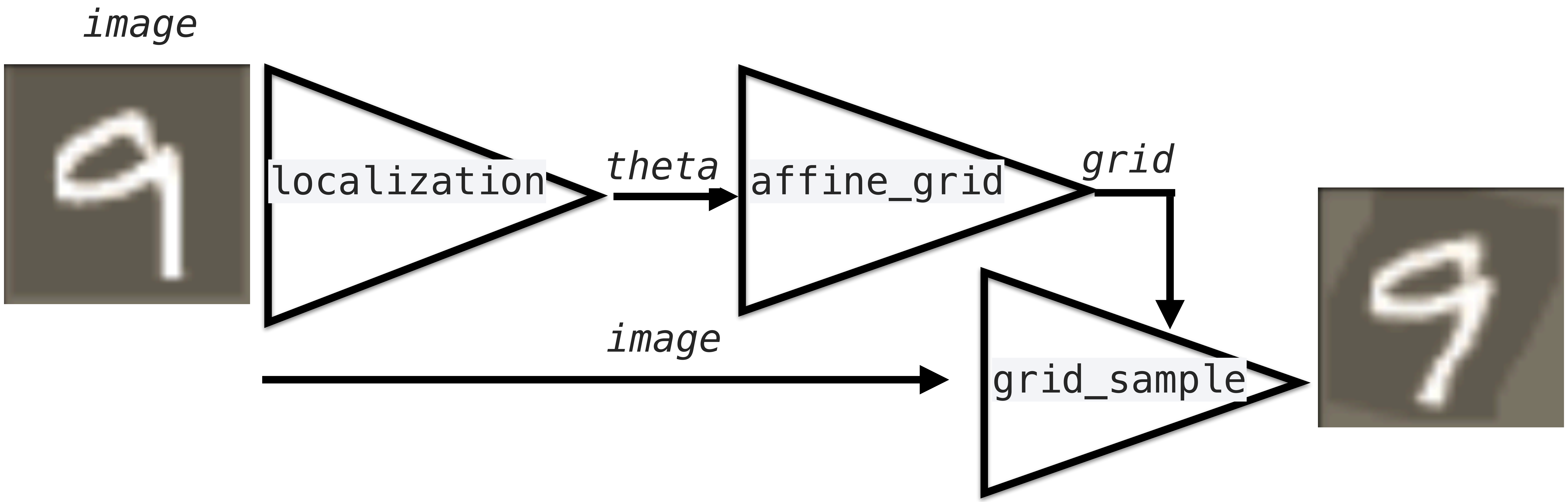


estimate pixel-wise correspondences of
the 2D similarity transformation

`torch.nn.functional.affine_grid(theta, size, align_corners=None)`

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

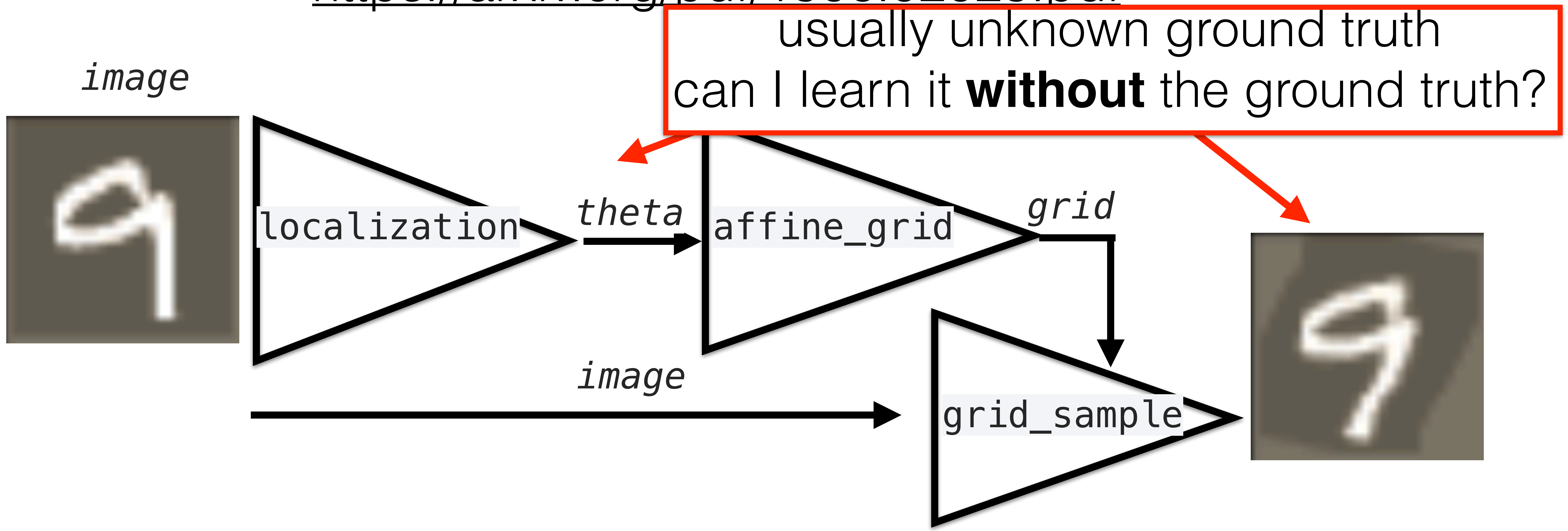



```
torch.nn.functional.affine_grid(theta, size, align_corners=None)

torch.nn.functional.grid_sample(input, grid, mode='bilinear',
                                padding_mode='zeros', align_corners=None)
```

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

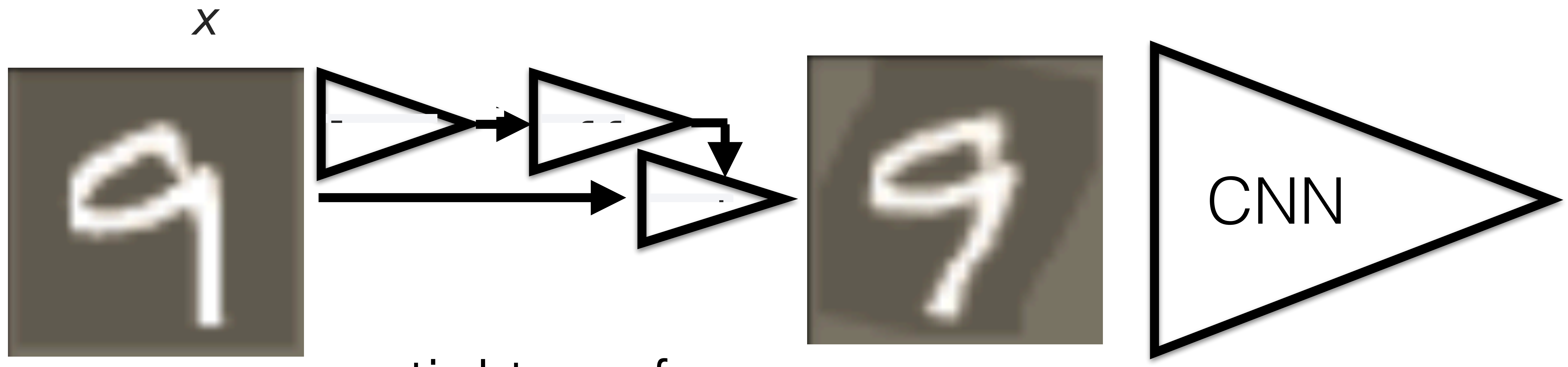



```
torch.nn.functional.affine_grid(theta, size, align_corners=None)

torch.nn.functional.grid_sample(input, grid, mode='bilinear',
                                padding_mode='zeros', align_corners=None)
```

Spatial Transformer networks [Jaderberg 2016]

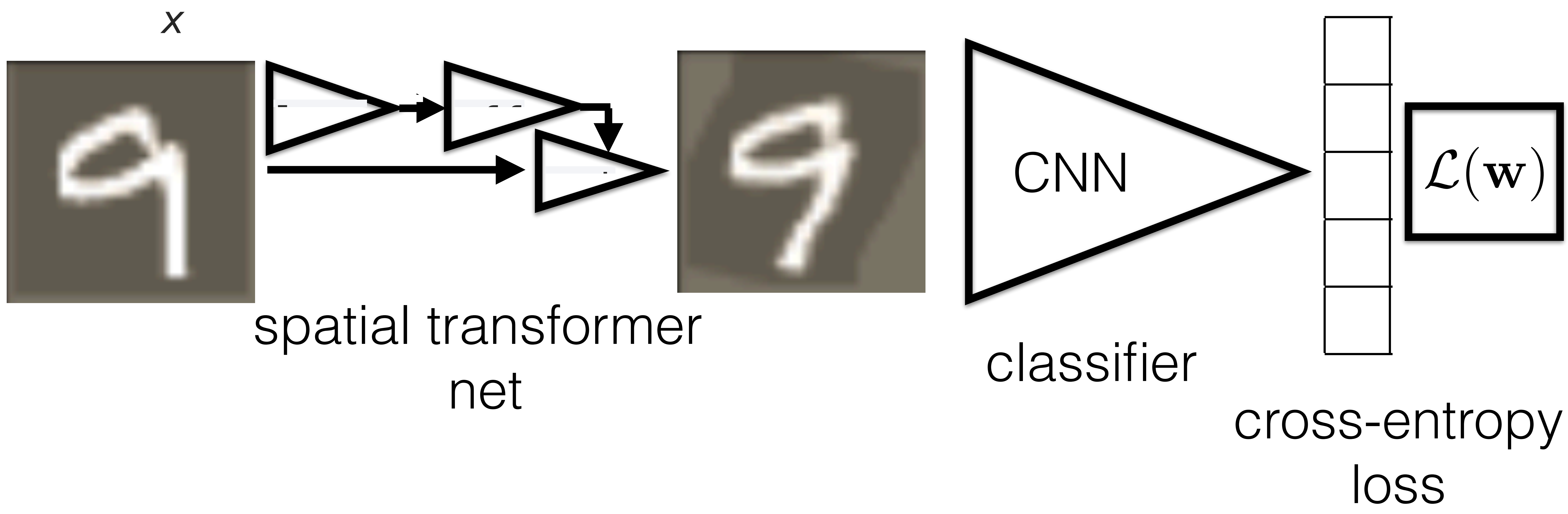
https://arxiv.org/pdf/1506.02025.pdf



spatial transformer net

classifier

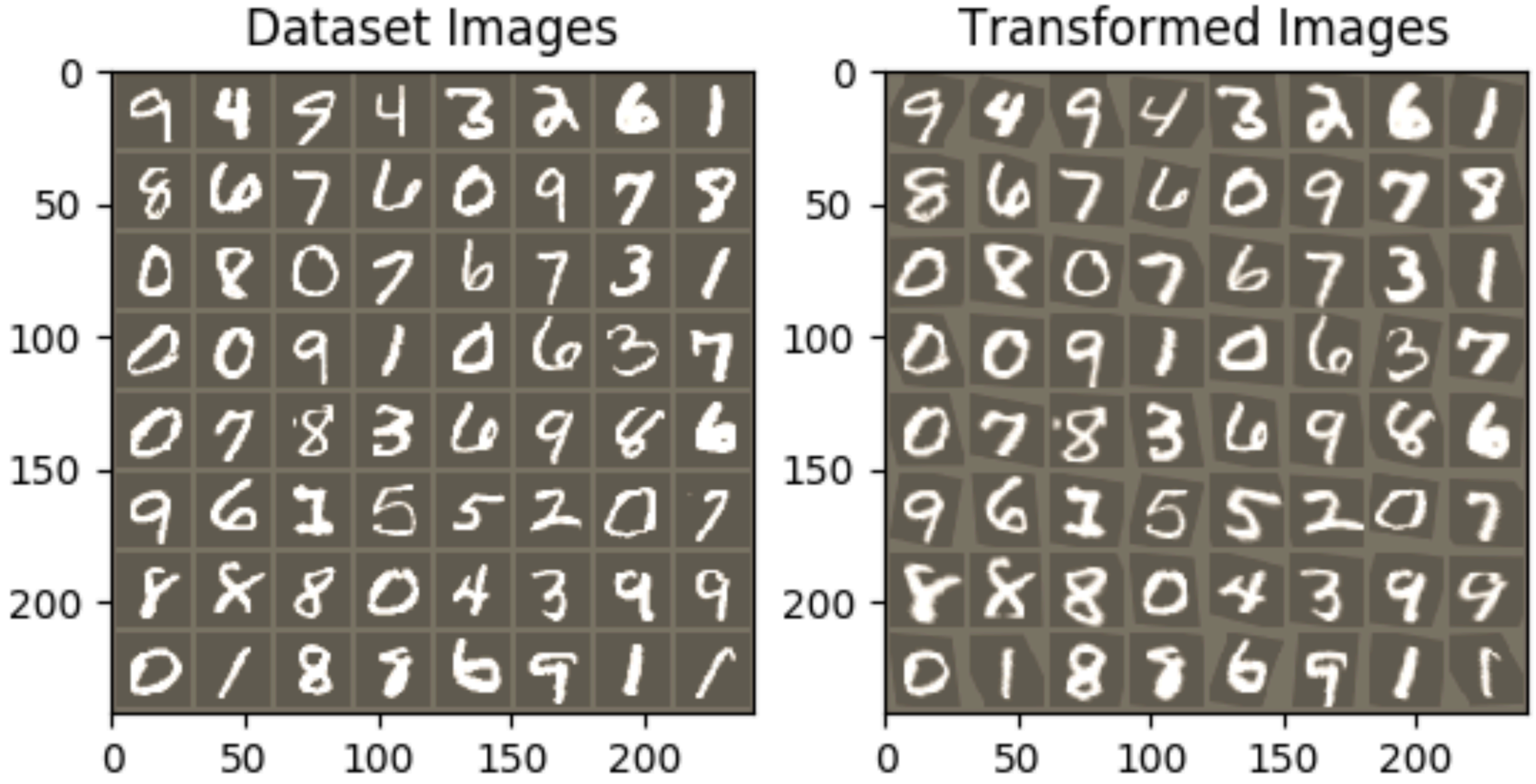
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



Backpropagation learns also STN weights, which perform the most suitable transformation for the classification task

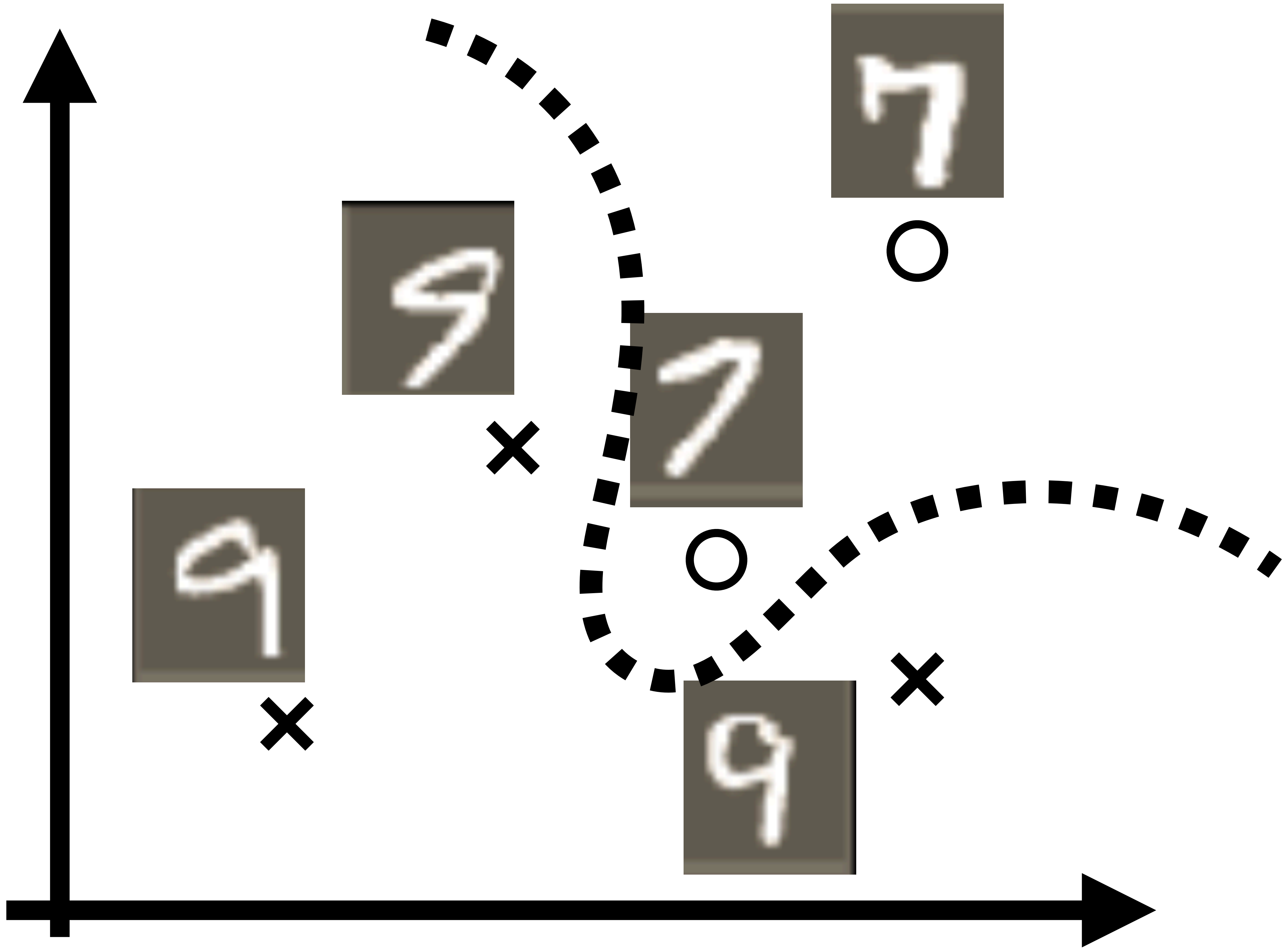
Spatial Transformer networks

https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html

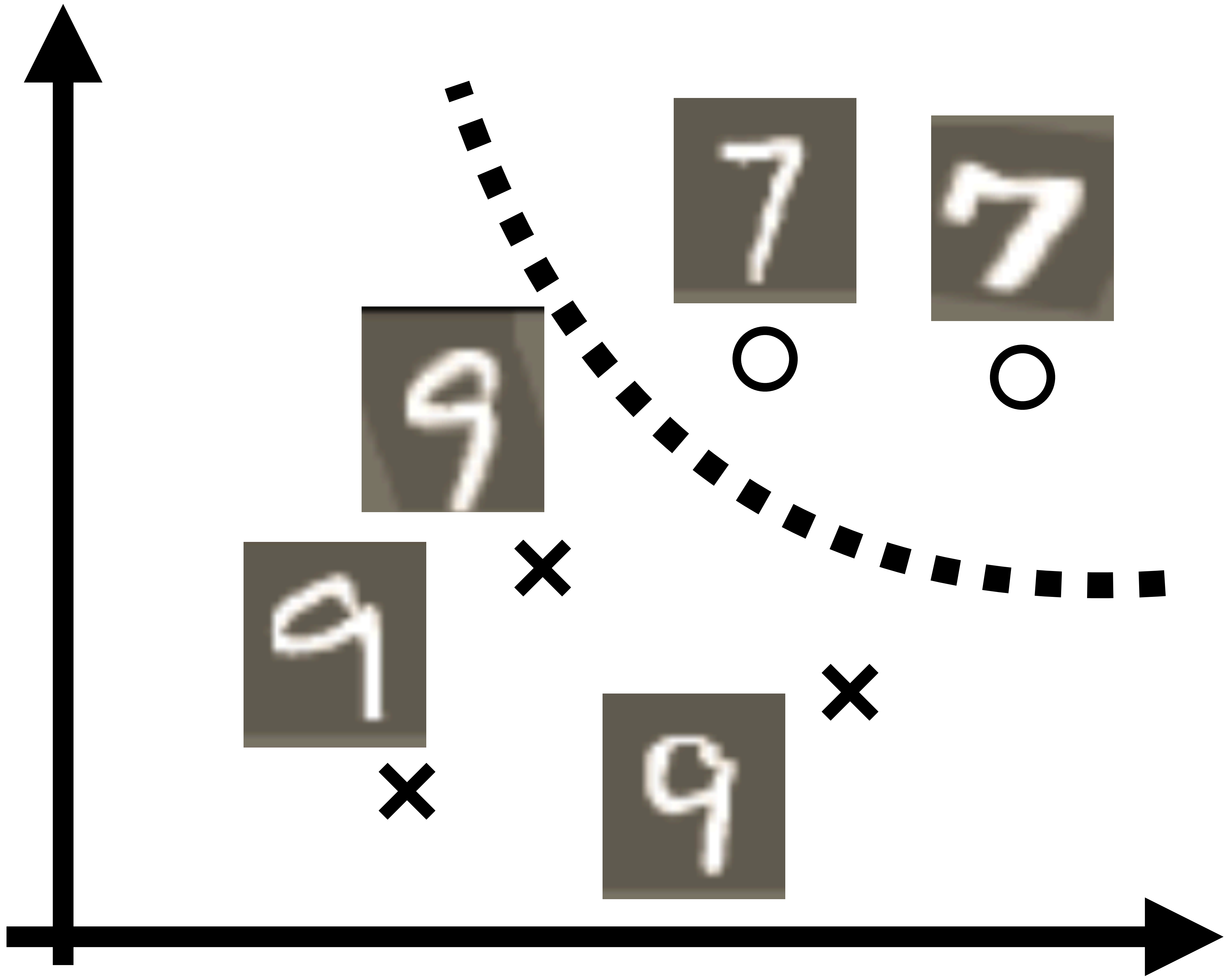


Spatial Transformer networks

I works better, because correctly-aligned numbers has smaller within-class scatter



“9” vs “7” - no compensation



“9” vs “7” compensated rot+transl
enforced strong prior about nature of the scatte

[Zimmermann et al TPAMI 2014]