Computer Vision:
Object detection

Siheng Chen 陈思衡

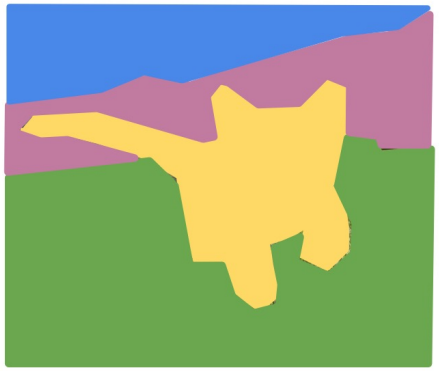
Common computer vision tasks

Classification

CAT

No spatial extent

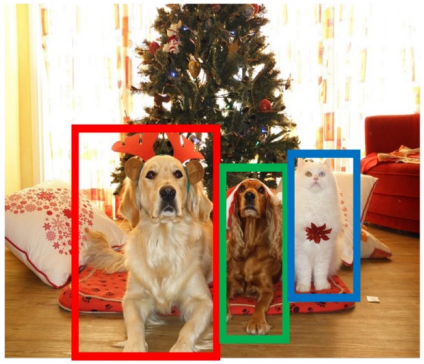
Semantic Segmentation



GRASS, CAT, TREE, SKY

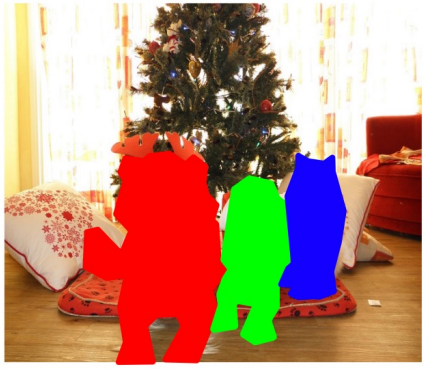
No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation

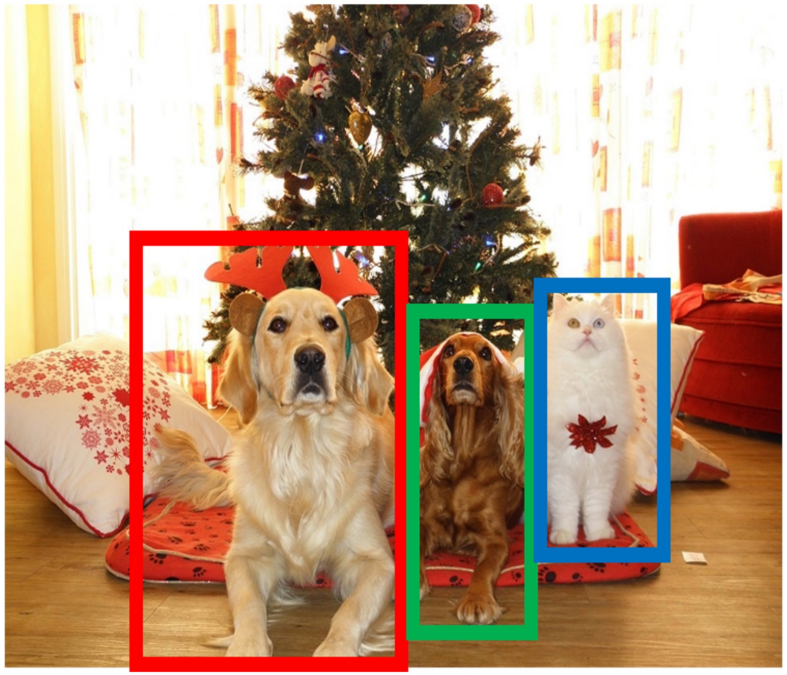


DOG, DOG, CAT

Multiple Objects

Object detection

Task: Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos



Input: Single RGB Image
Output: A set of detected objects;

For each object:

1. Category label
   (from fixed, known set of categories)
2. Bounding box
   (four numbers: x, y, width, height)

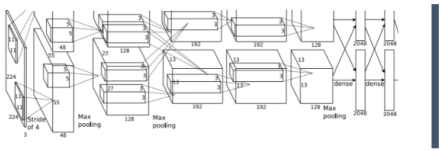
Object detection

Detecting a single object

what

Backbone:
pretrained on ImageNet

This image is CC0 public domain



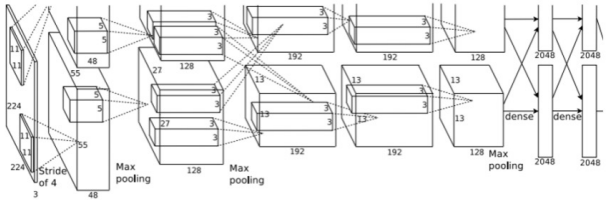
Vector:
4096

Treat localization as a regression problem!

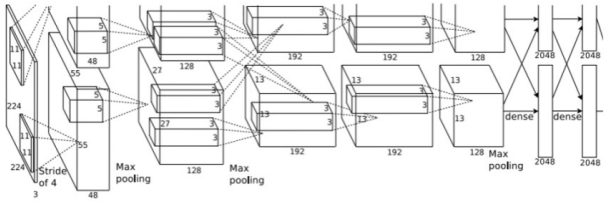
where

Object detection

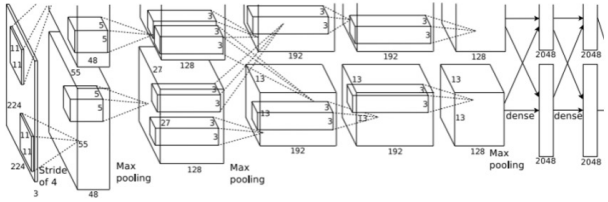
Detecting multiple objects



CAT: (x, y, w, h)



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

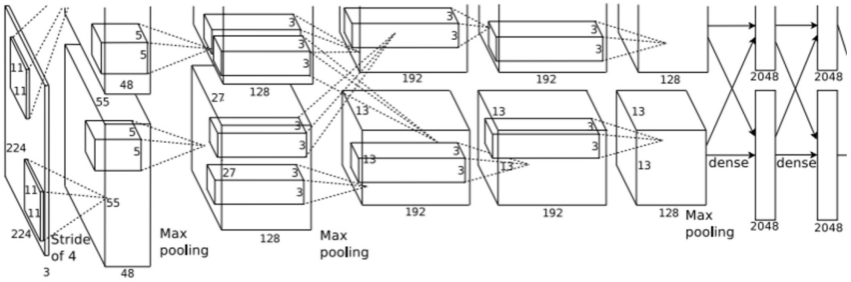
Need different numbers of outputs per image

Object detection

Detecting multiple objects

Sliding Window

Apply a CNN to many different crops of the image,
CNN classifies each crop as object or background



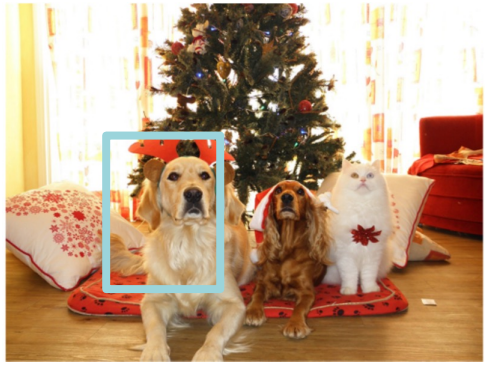
Dog? NO
Cat? YES
Background? NO

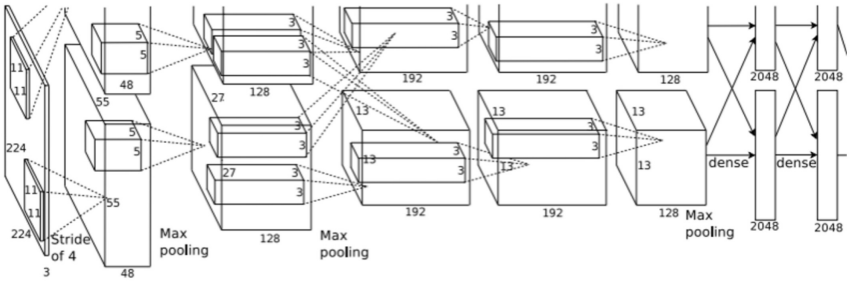
Object detection

Detecting multiple objects

Sliding Window

Apply a CNN to many different crops of the image,
CNN classifies each crop as object or background





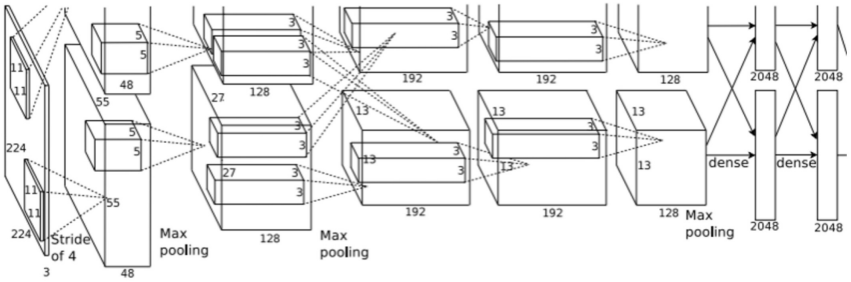
Dog? NO
Cat? YES
Background? NO

Object detection

Detecting multiple objects

Sliding Window

Apply a CNN to many different crops of the image,
CNN classifies each crop as object or background



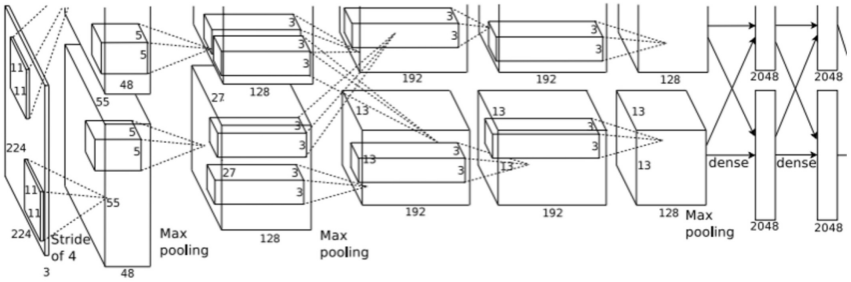
Dog? NO
Cat? YES
Background? NO

Object detection

Detecting multiple objects

Sliding Window

Apply a CNN to many different crops of the image,
CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Object detection

Detecting multiple objects

Sliding Window

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions: (W – w + 1) * (H – h + 1)

Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

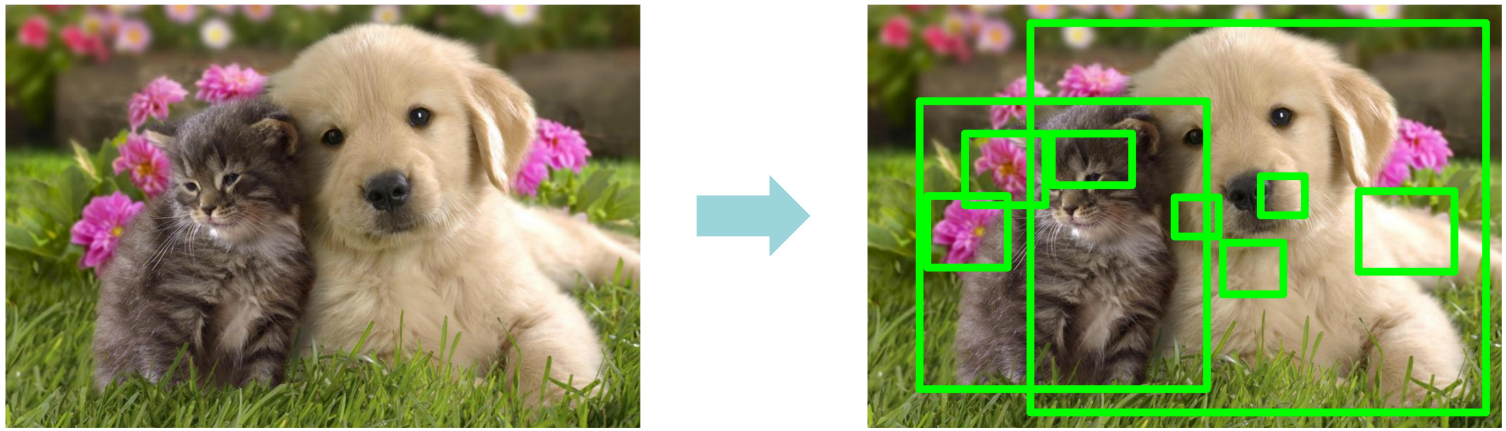
$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

800 x 600 image has ~58M boxes!
No way we can evaluate them all

Object detection

Detecting multiple objects

- Region proposals
  - Find a small set of boxes that are likely to cover all objects
  - Often based on heuristics: e.g. look for “blob-like” image regions
  - Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds



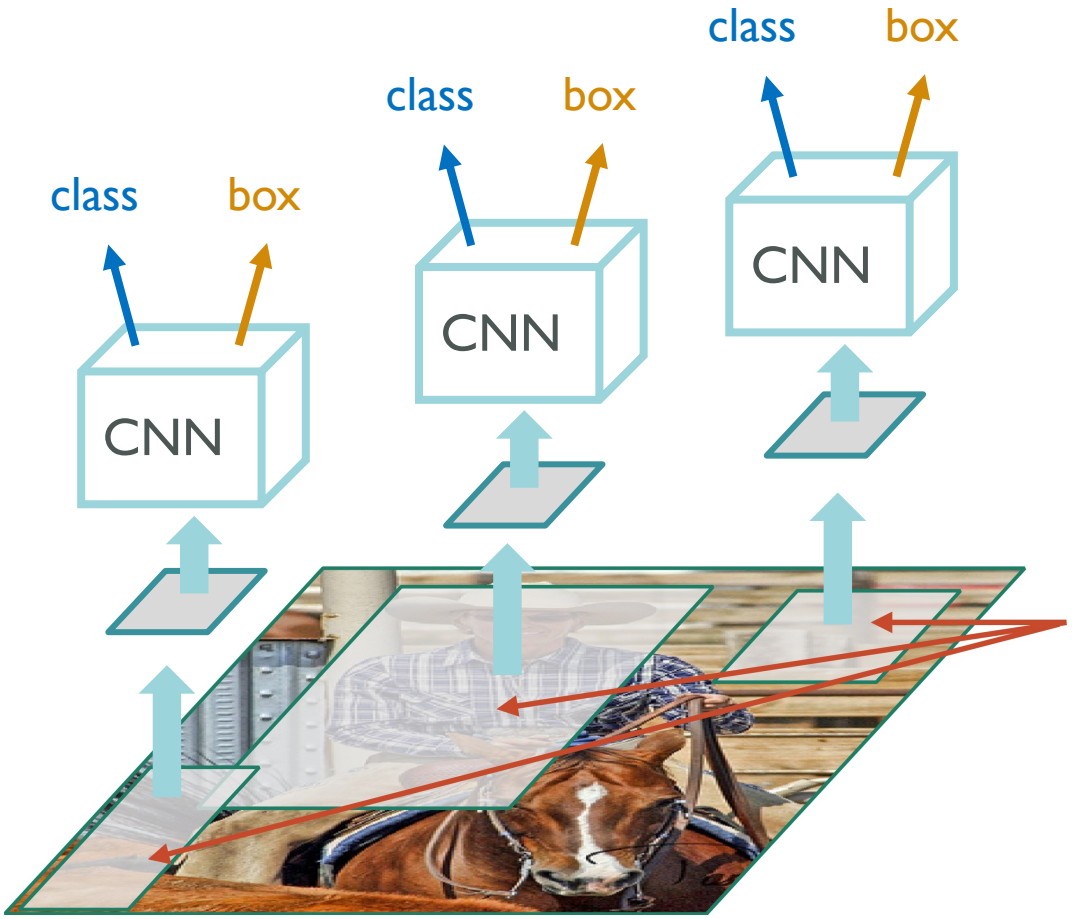
Object detection

R-CNN: Region-Based CNN

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

Object detection

R-CNN: Region-Based CNN



Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through CNN

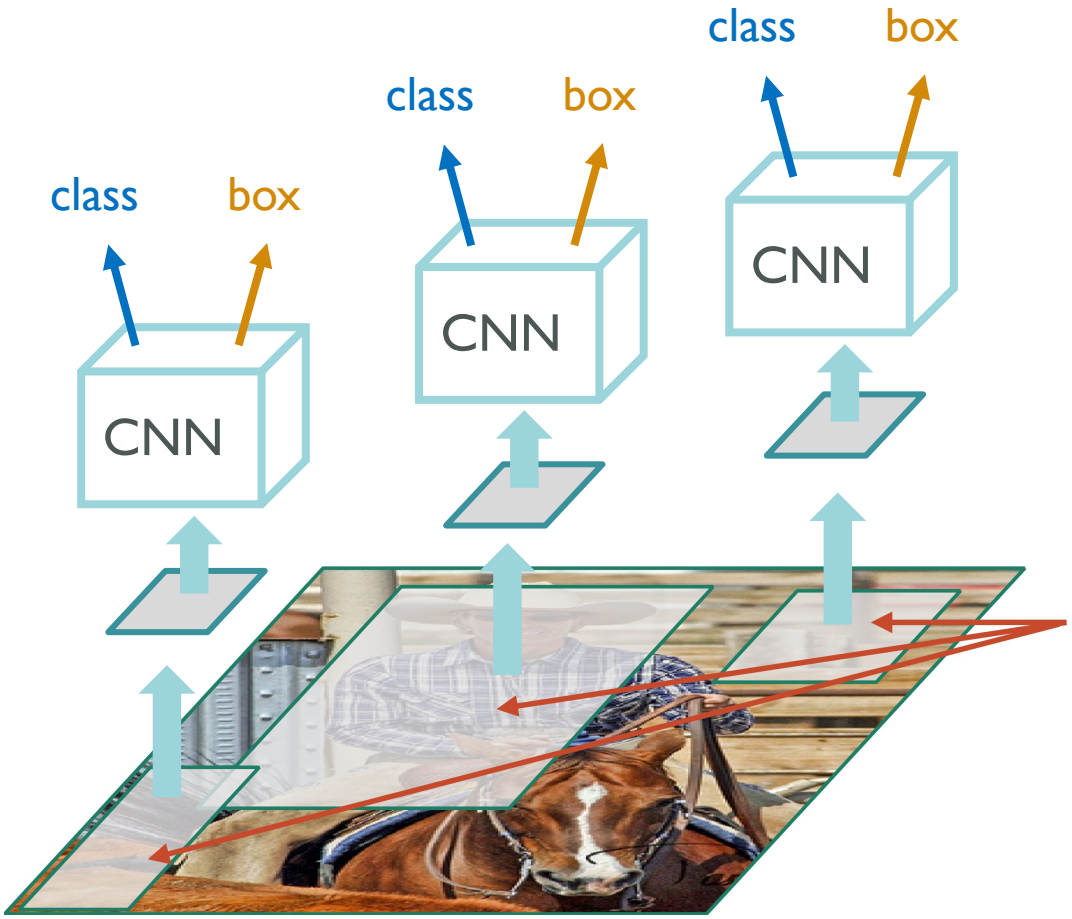
Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

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

Object detection

R-CNN: Region-Based CNN



Region proposal: (p_x, p_y, p_h, p_w)

Transform: (t_x, t_y, t_h, t_w)

Output box: (b_x, b_y, b_h, b_w)

Translate relative to box size:

$b_x = p_x + p_w t_x$; $b_y = p_y + p_h t_y$

Log-space scale transform:

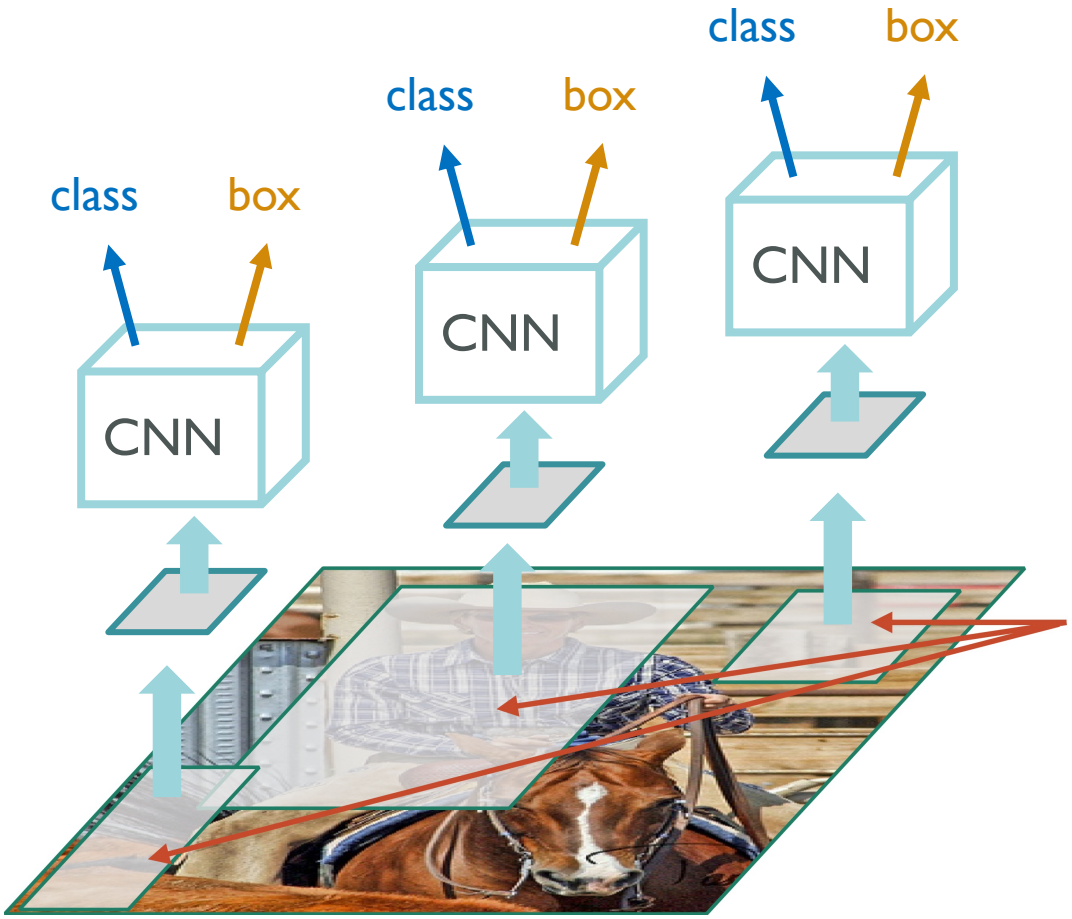
$b_w = p_w \exp(t_w)$ $b_h = p_h \exp(t_h)$

Regions of Interest (RoI)
from a proposal method (~2k)

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

Object detection

R-CNN: Region-Based CNN



Input: Single RGB image

1. Run region proposal method to compute ~2000 region proposals

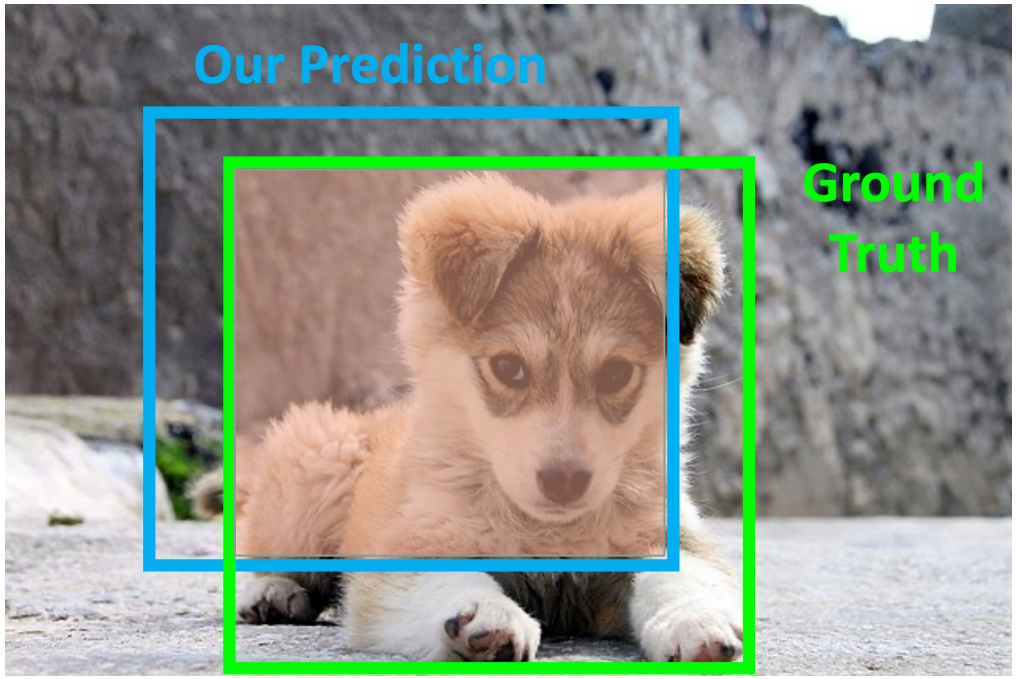
2. Resize each region to 224x224 and run independently through CNN to predict class scores and bbox transform

3. Use scores to select a subset of region proposals to output (Many choices here: threshold on background, or per-category? Or take top K proposals per image?)

4. Compare with ground-truth boxes

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

Object detection

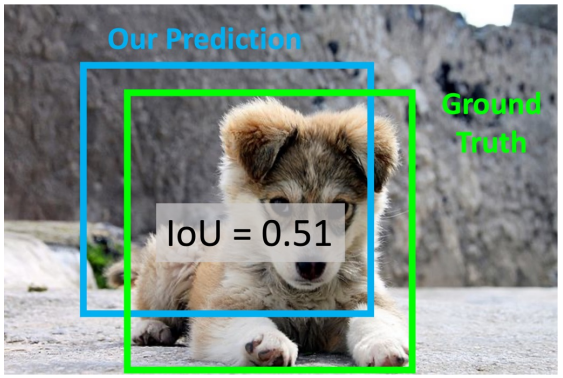


How can we compare our prediction to the ground-truth box?

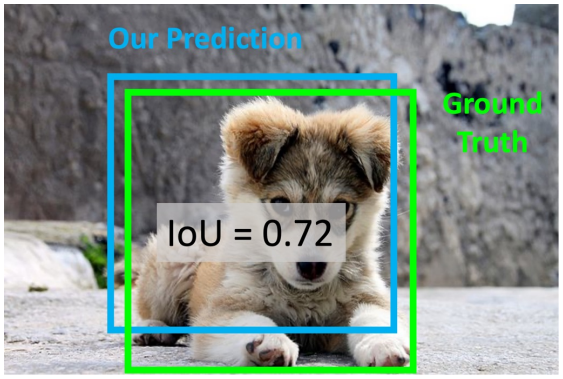
Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

$$\text{IoU} = \frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

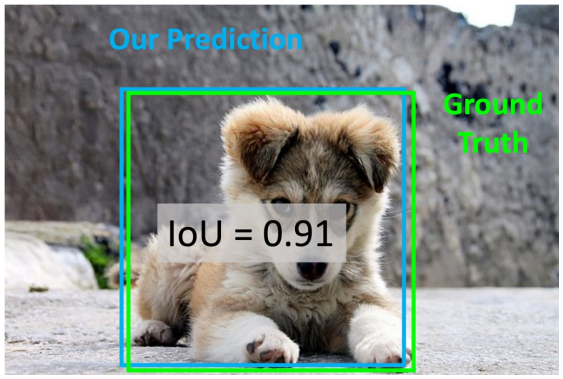
Object detection



IoU > 0.5 is decent



IoU > 0.7 is pretty good



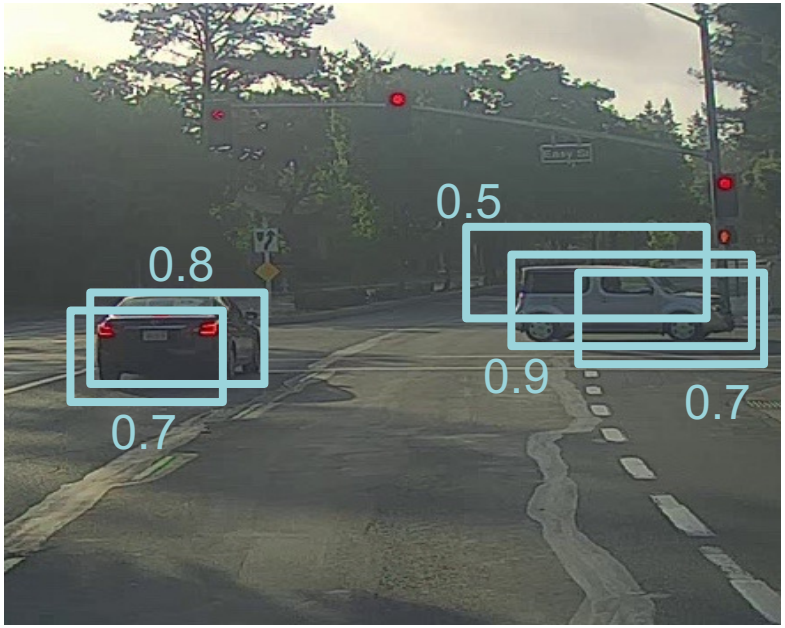
IoU > 0.9 is almost perfect

$$\text{IoU} = \frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

Object detection

How to produce final box

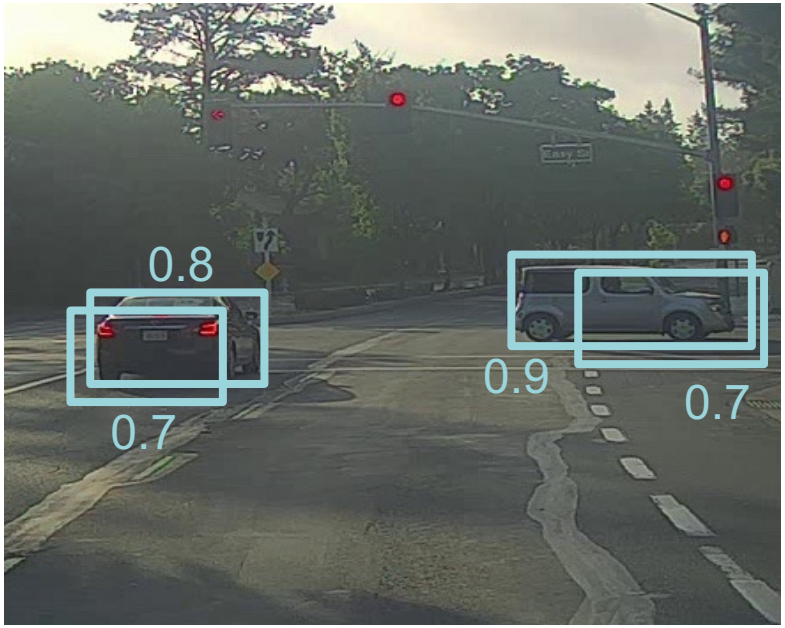
1. Get the classification score & bounding boxes



Object detection

How to produce final box

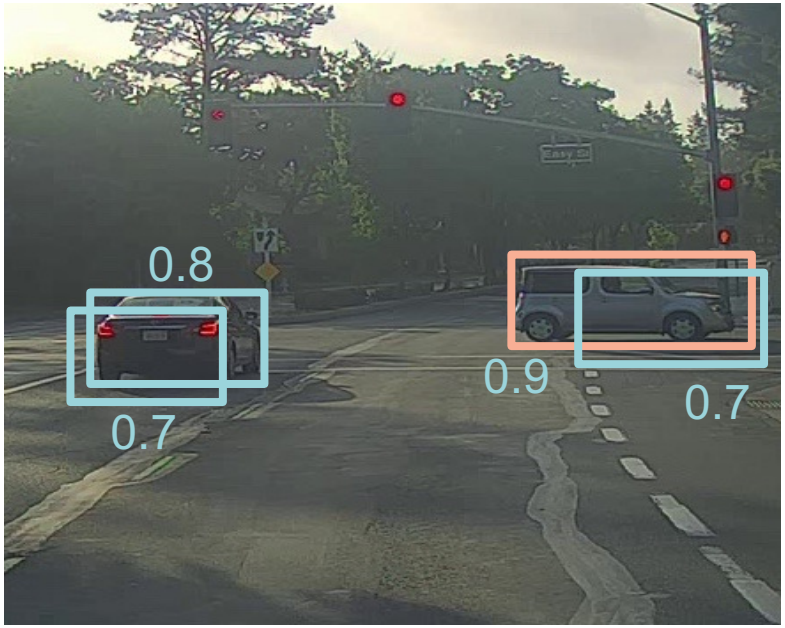
1. Get the classification score & bounding boxes
2. Discard boxes with low classification score (<0.6)



Object detection

How to produce final box

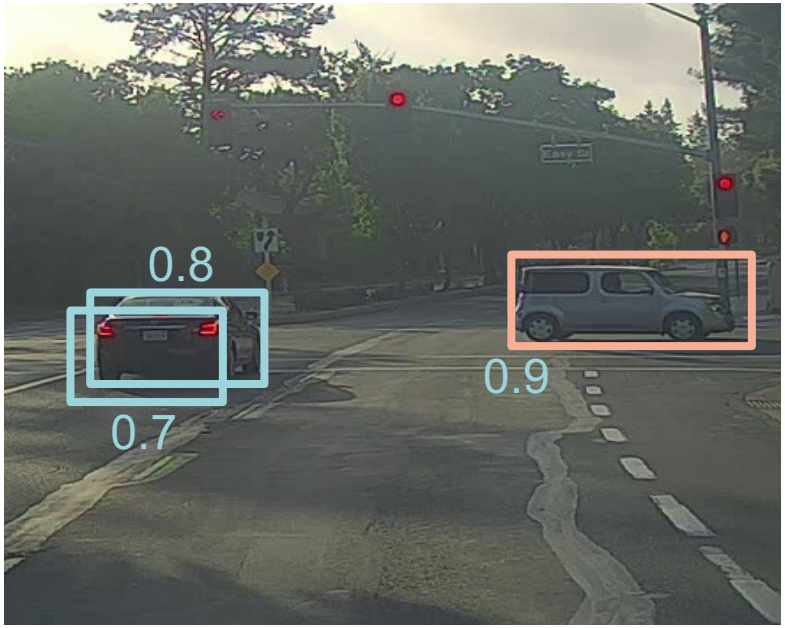
1. Get the classification score & bounding boxes
2. Discard boxes with low classification score (<0.6)
3. Pick the box with the largest classification score and output that as a prediction



Object detection

How to produce final box

1. Get the classification score & bounding boxes
2. Discard boxes with low classification score (<0.6)
3. Pick the box with the largest classification score and output that as a prediction
4. Discard any remaining box with IoU ≥0.5 with the box output in the previous step

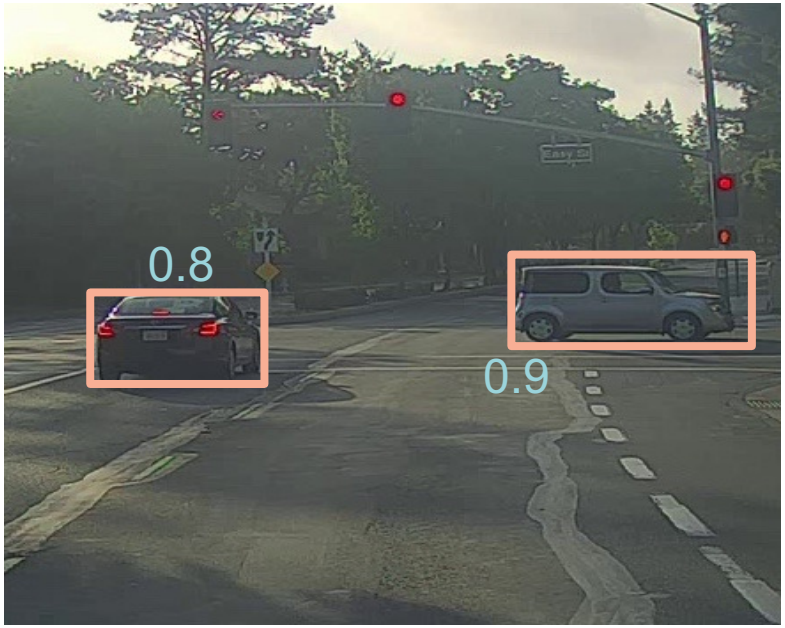


Object detection

How to produce final box

1. Get the classification score & bounding boxes
2. Discard boxes with low classification score (<0.6)
3. Pick the box with the largest classification score and output that as a prediction
4. Discard any remaining box with IoU ≥0.5 with the box output in the previous step

Non-max suppression (NMS)

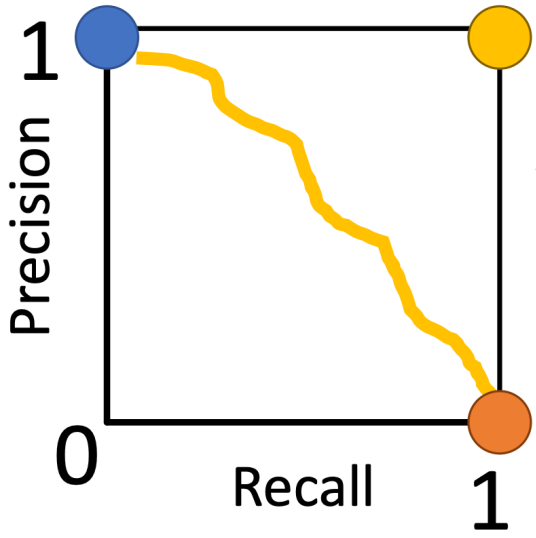


Object detection

Evaluation metric

- True detection: high intersection over union
- Precision: #true detections / #detections
- Recall: #true detections / #true positives

Reject everything: no mistakes

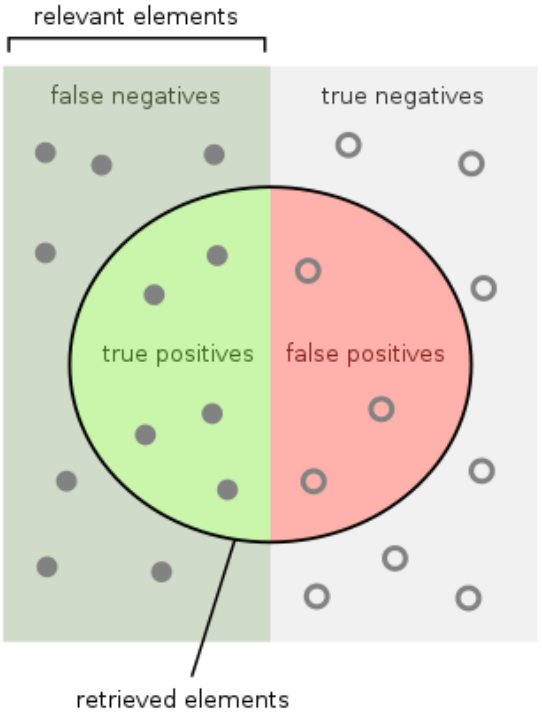


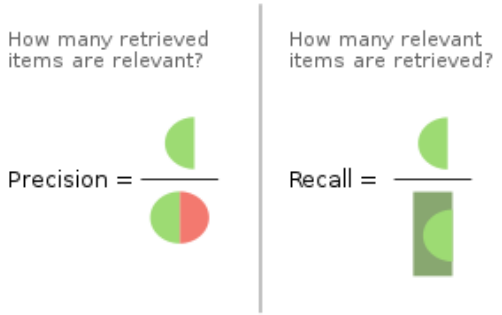
Ideal!

Average precision@***IoU

Summarize by area under curve

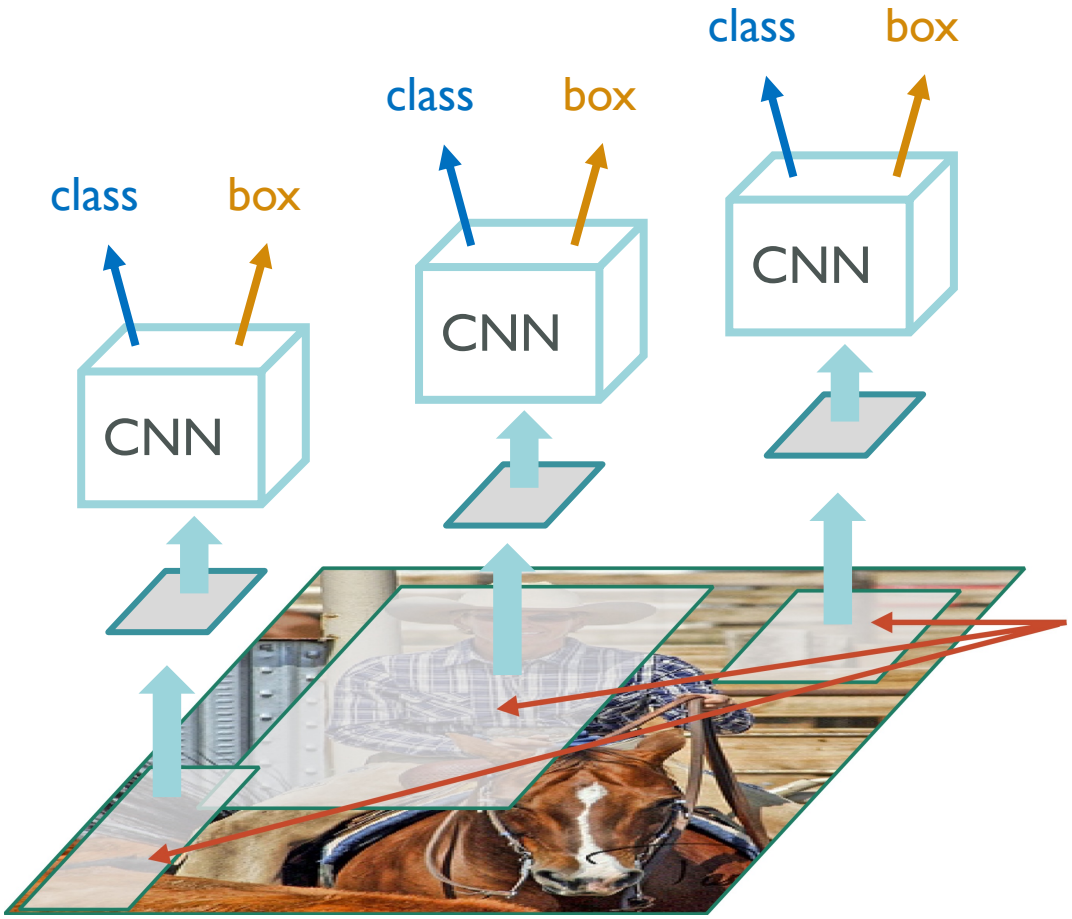
Accept everything: Miss nothing





Object detection

R-CNN: Region-Based CNN



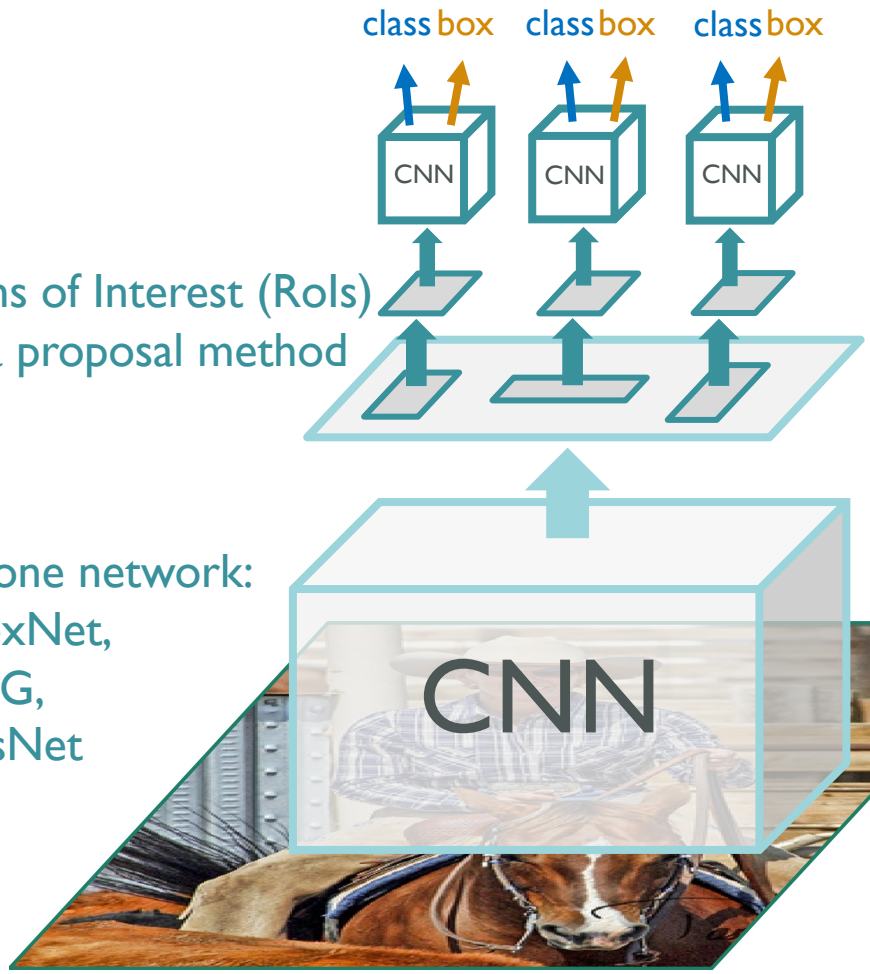
Problem: Very slow! Need to do ~2k forward passes for each image!

Solution: Run CNN *before* cropping!

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

Object detection

Fast R-CNN



Per-region network relatively lightweight

Crop + Resize features

Regions of Interest (RoIs) from a proposal method

Image features

Run the whole image through CNN

Heavy computation

Backbone network:

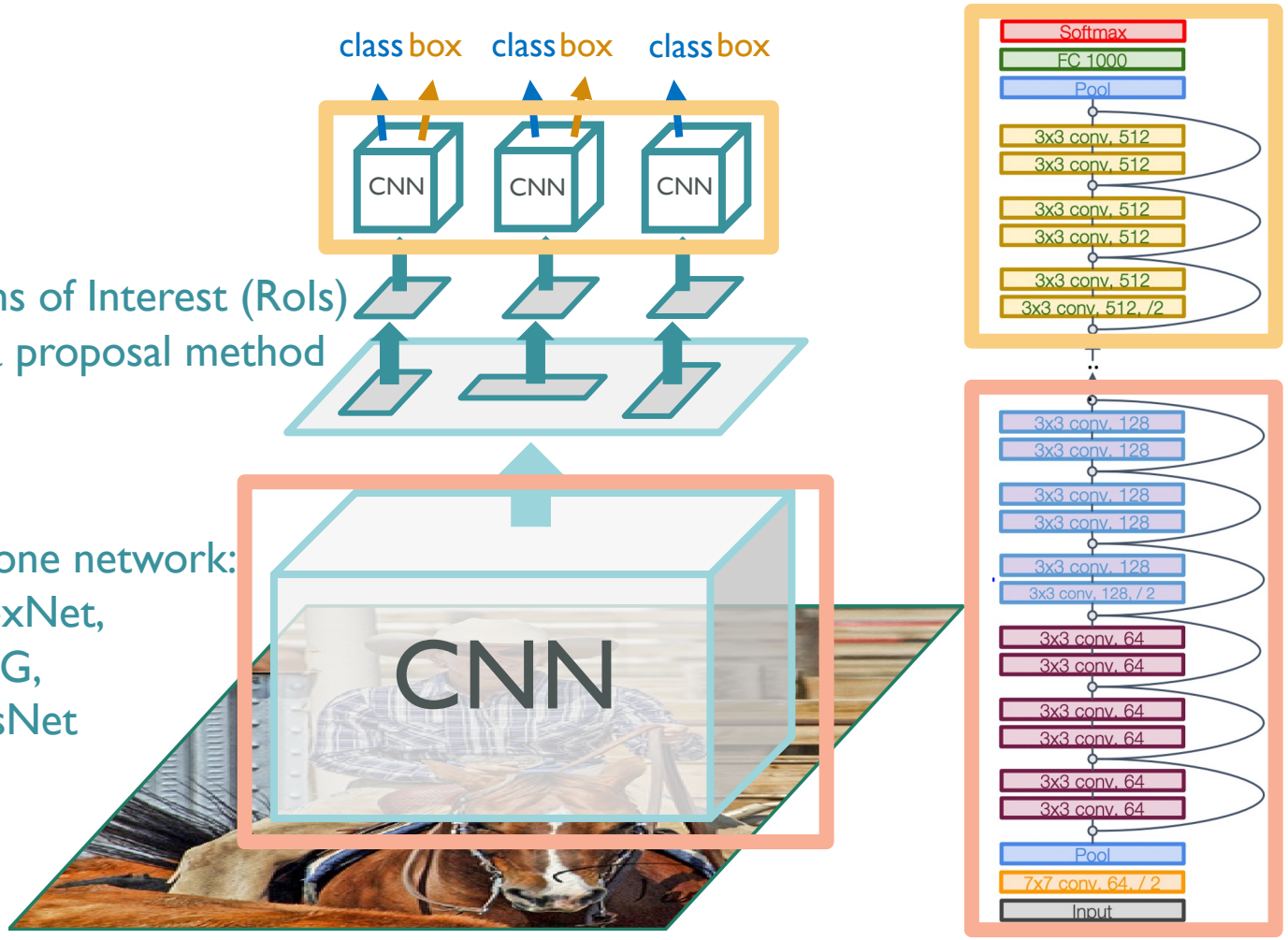
- AlexNet,
- VGG,
- ResNet

Input image

Girshick, "Fast R-CNN", ICCV 2015.

Object detection

Fast R-CNN



For ResNet,
last stage is used as per-region network;
the rest of the network is used as backbone

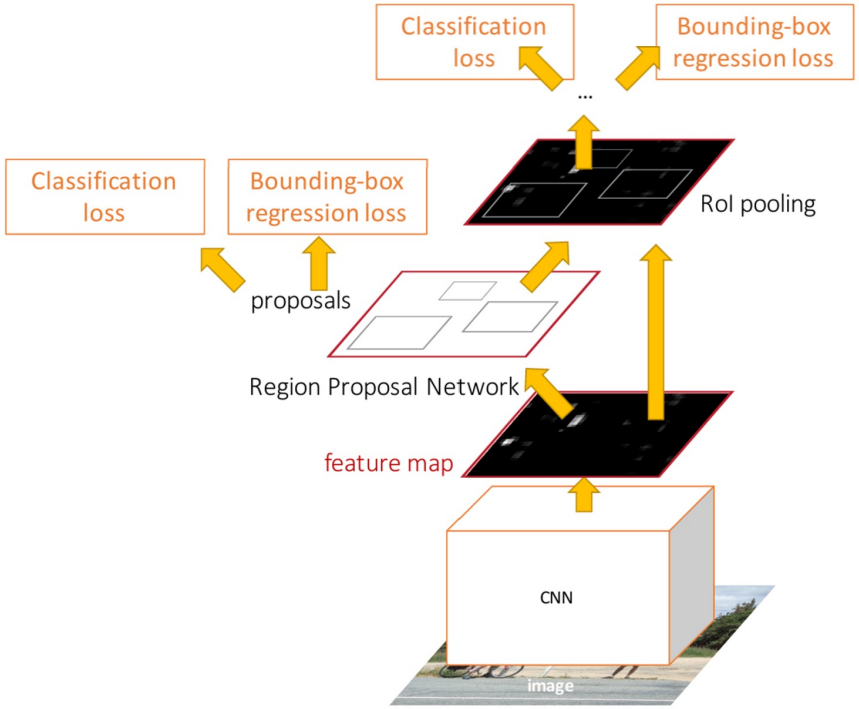
Girshick, "Fast R-CNN", ICCV 2015.

Object detection

Faster R-CNN

Insert region proposal network (RPN) to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



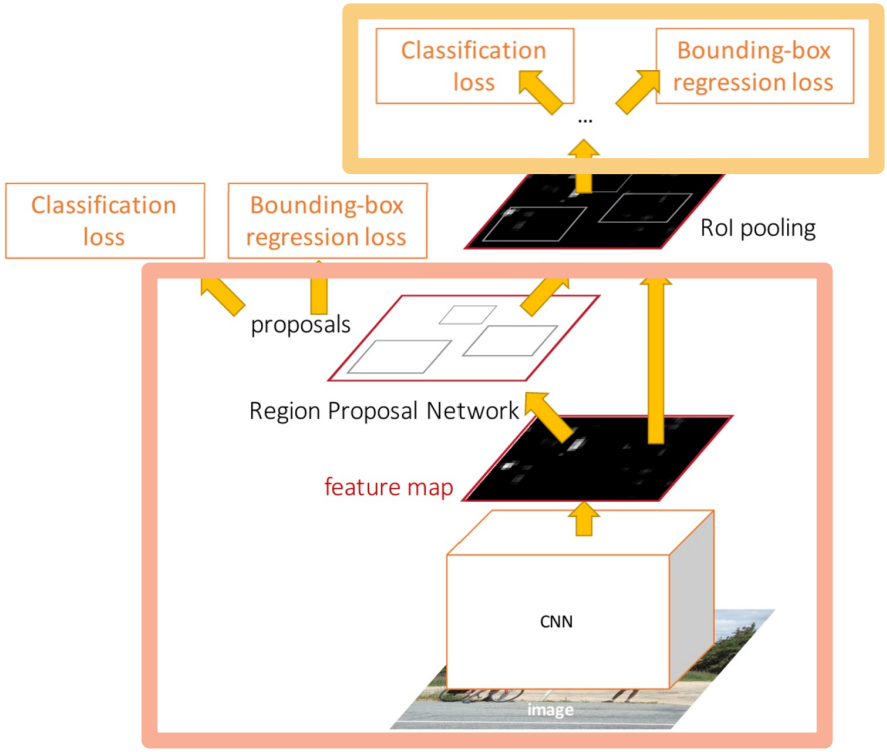
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Object detection

Faster R-CNN is a two-stage object detector

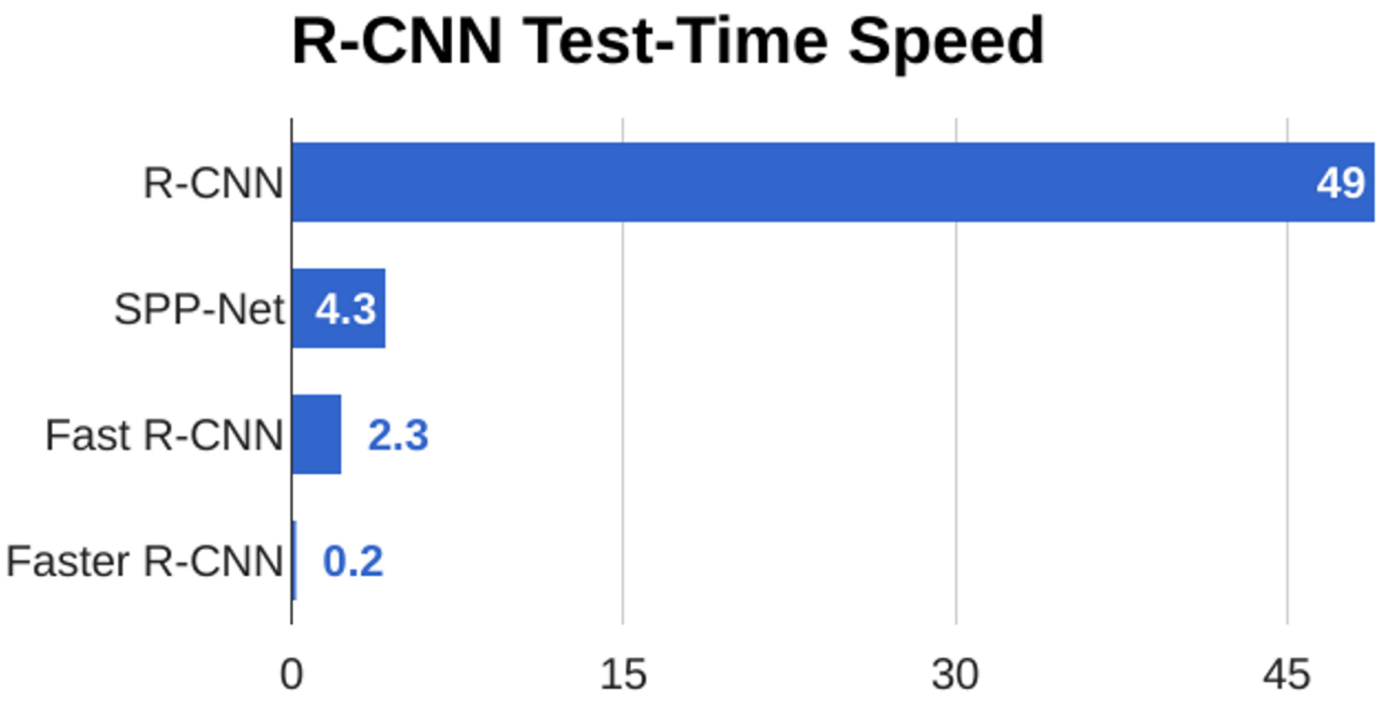
First stage: Run once per **image**
- Backbone network
- Region proposal network

Second stage: Run once per **region**
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



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

Object detection

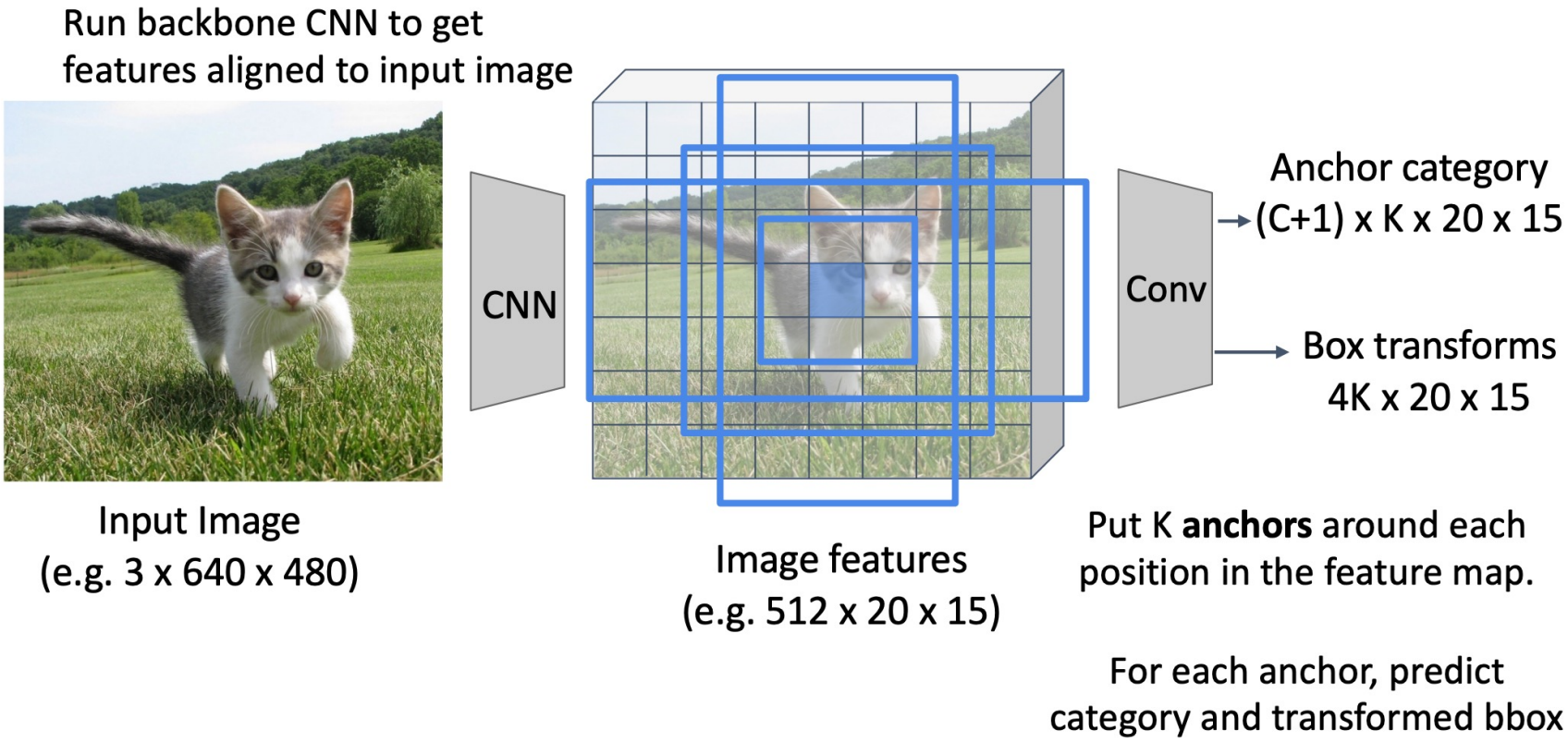


Object detection

https://www.youtube.com/watch?v=1zFeQmbJBaI

Object detection

Single-stage object detection



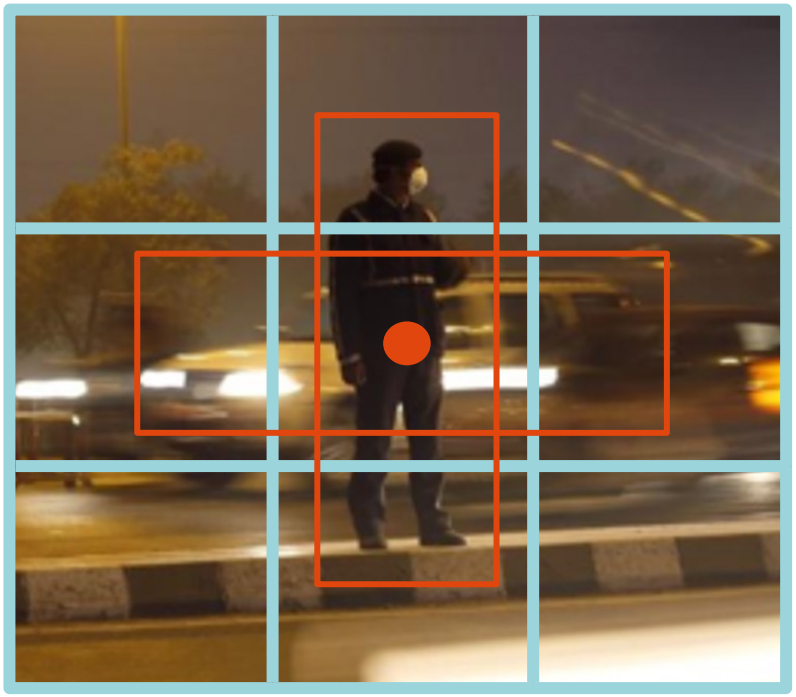
Object detection

Anchor box

Redmon et al., 2015, You Only Look Once: Unified real-time object detection

Object detection

Anchor box: Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn



YOLO's loss function compares each object in the ground truth with one anchor.

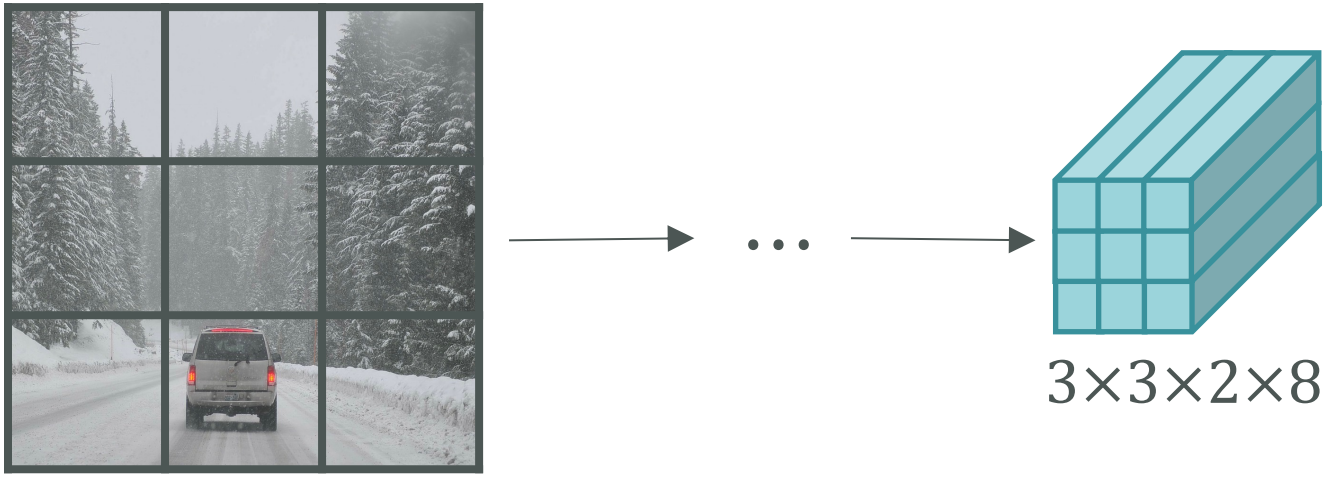
It picks the anchor (before any offsets) with highest IoU compared to the ground truth.

Then the predictions are added as offsets to the anchor.

Redmon et al., 2015, You Only Look Once: Unified real-time object detection

Object detection

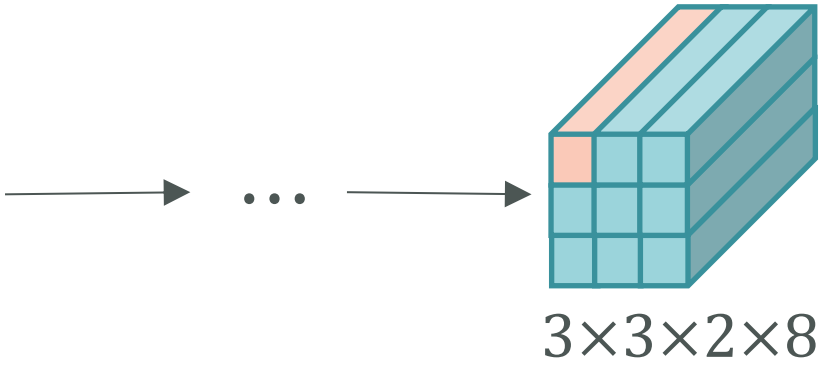
YOLO



Redmon et al., 2015, You Only Look Once: Unified real-time object detection

Object detection

YOLO



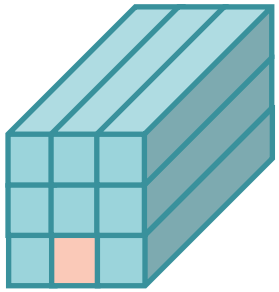
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Redmon et al., 2015, You Only Look Once: Unified real-time object detection

Object detection

YOLO

$$\rightarrow \cdots \rightarrow$$



$3\times3\times2\times8$

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Redmon et al., 2015, You Only Look Once: Unified real-time object detection

Object detection

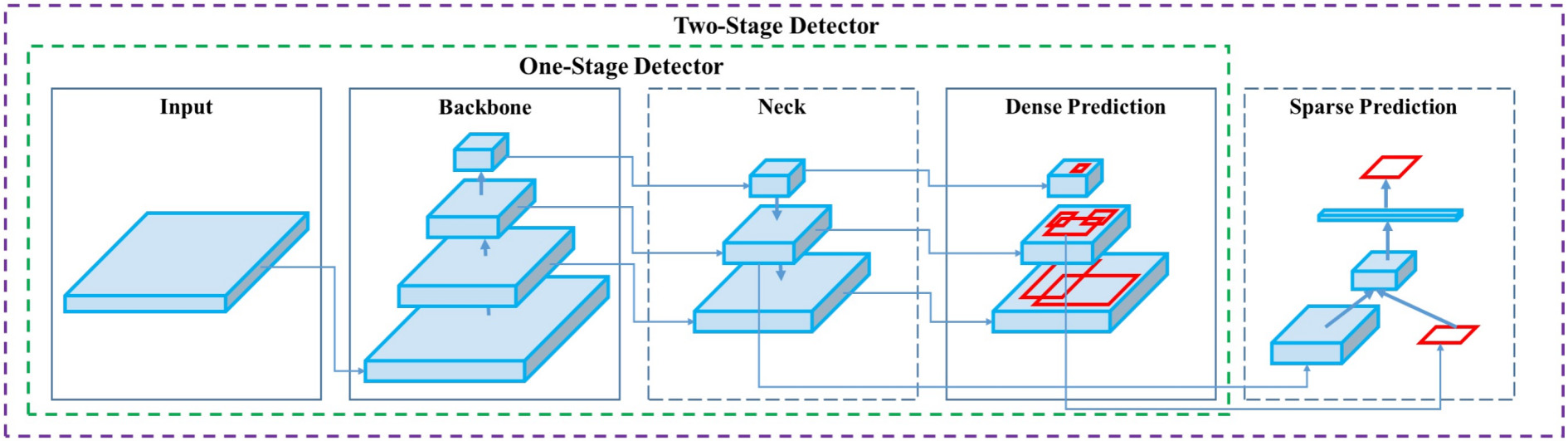
YOLO

How to detect by YOLO

1. For each grid call, get 2 predicted bounding boxes.
2. Get rid of low probability predictions.
3. For each class (pedestrian, car, motorcycle) use non-max suppression (NMS) to generate final predictions.

Redmon et al., 2015, You Only Look Once: Unified real-time object detection

Object detection



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

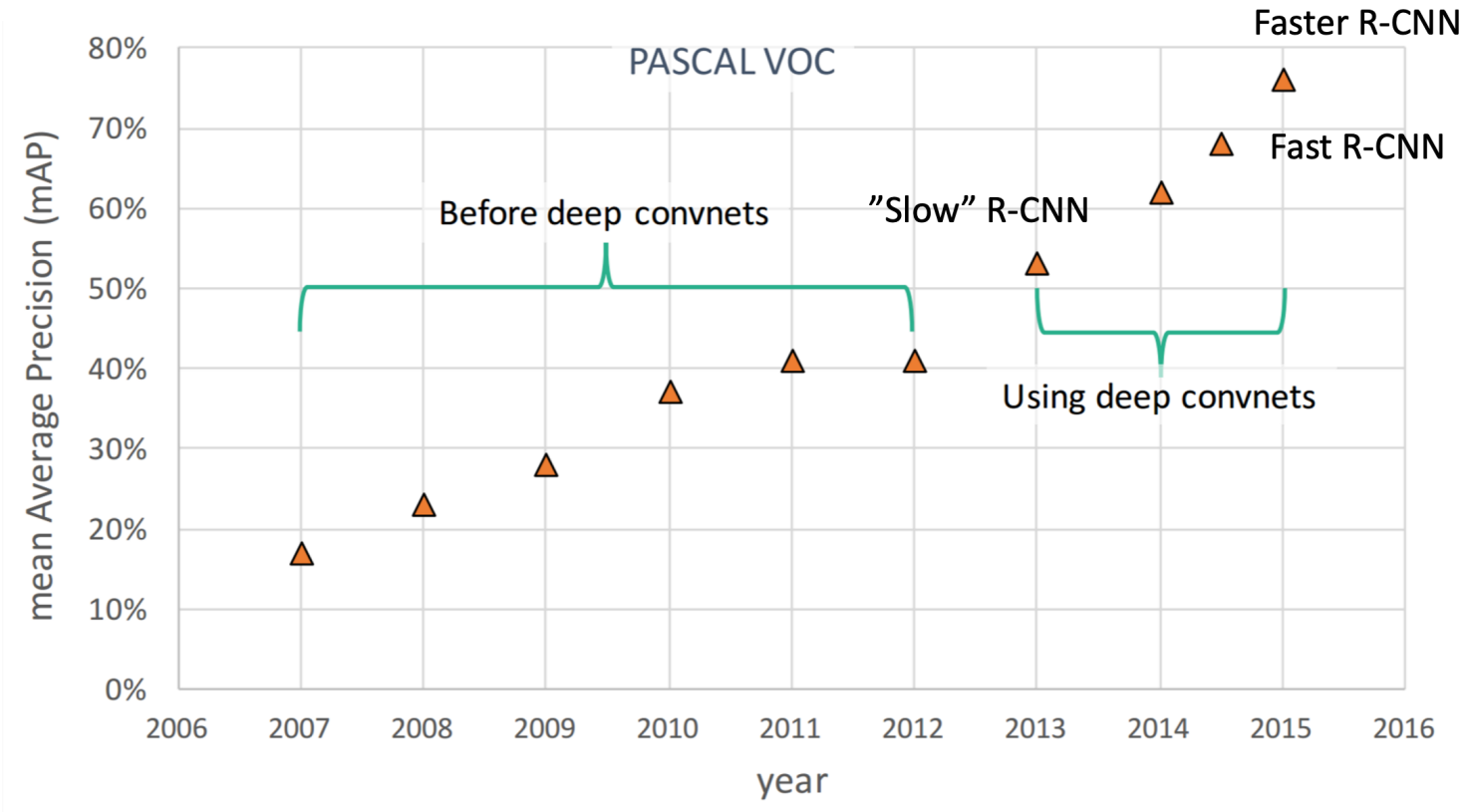
Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

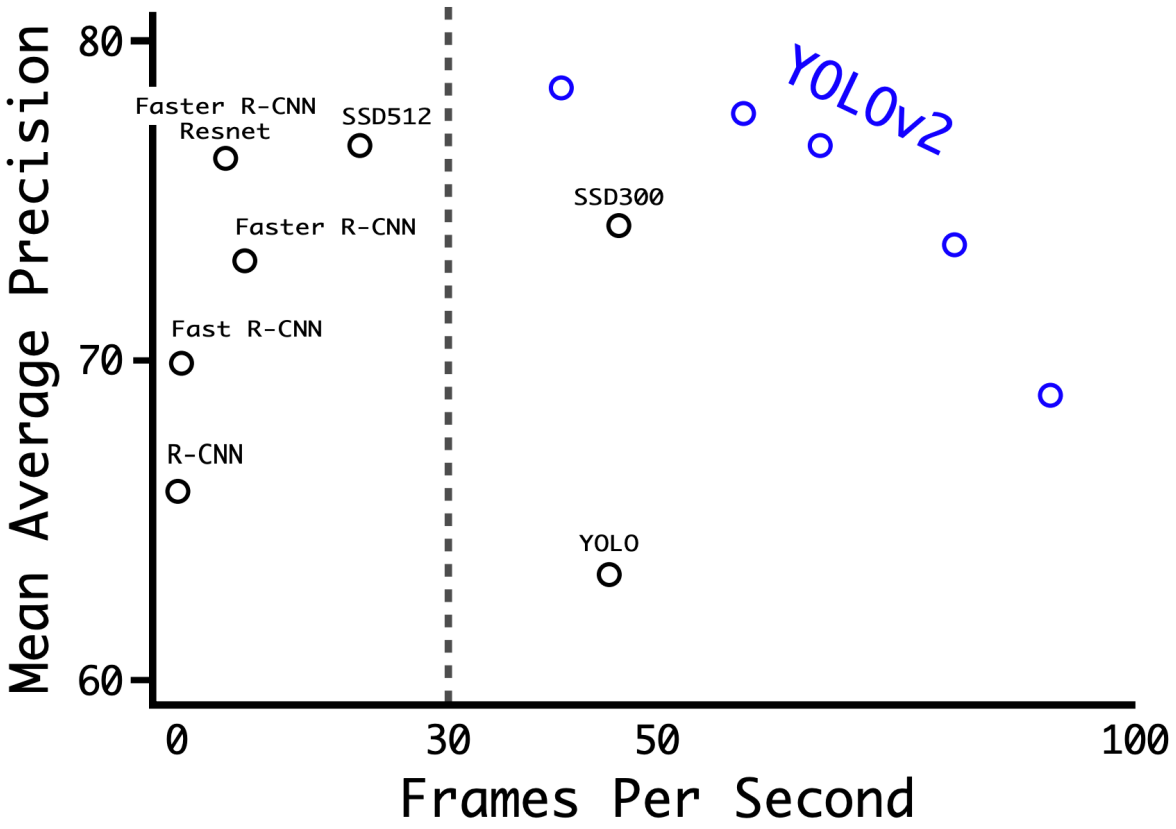
Bochkovskiy et al, YOLOv4: Optimal Speed and Accuracy of Object Detection

Object detection



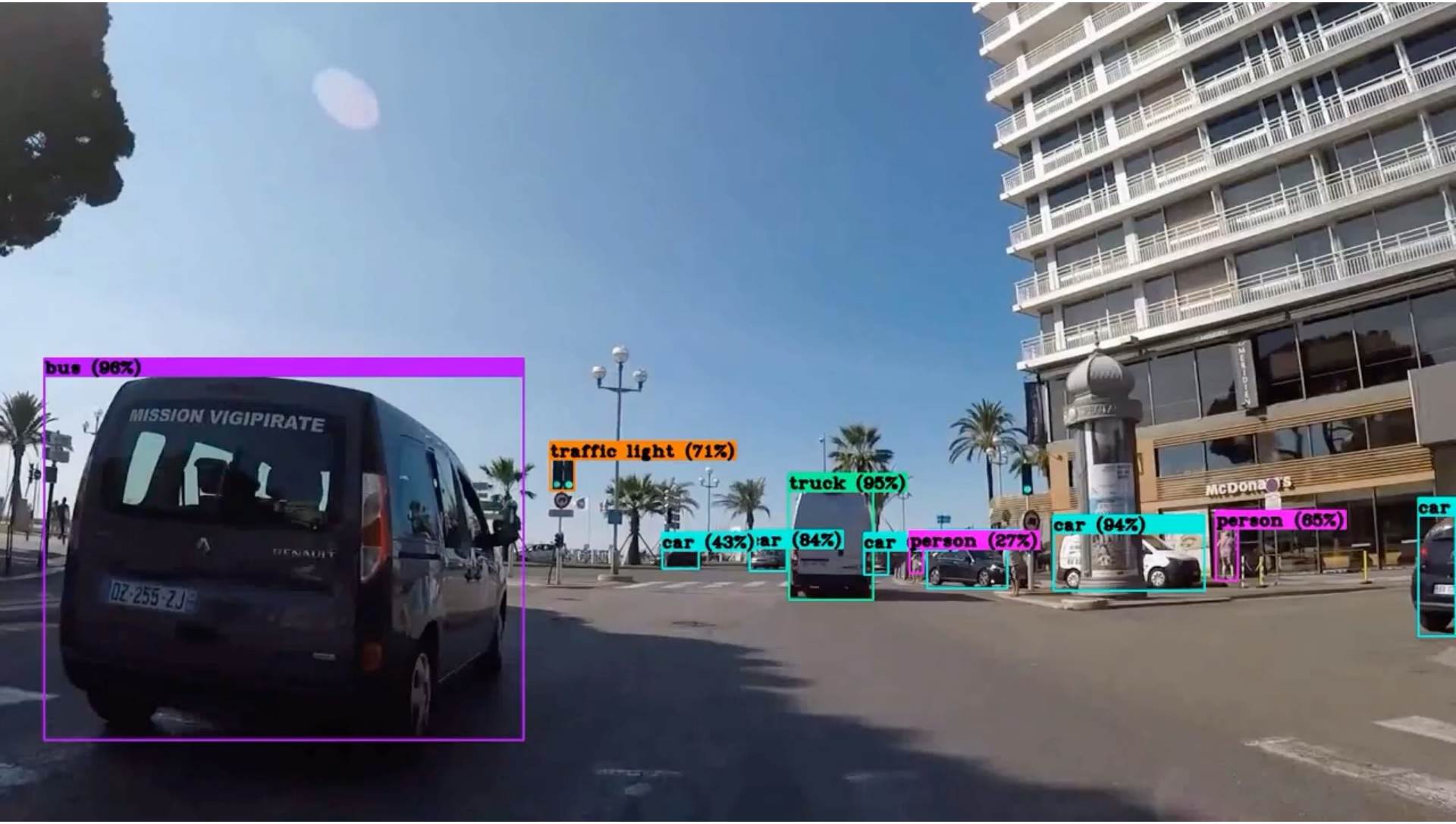
Object detection

YOLO v2



Joseph Redmon, Ali Farhadi, YOLO9000: Better, Faster, Stronger, 2016

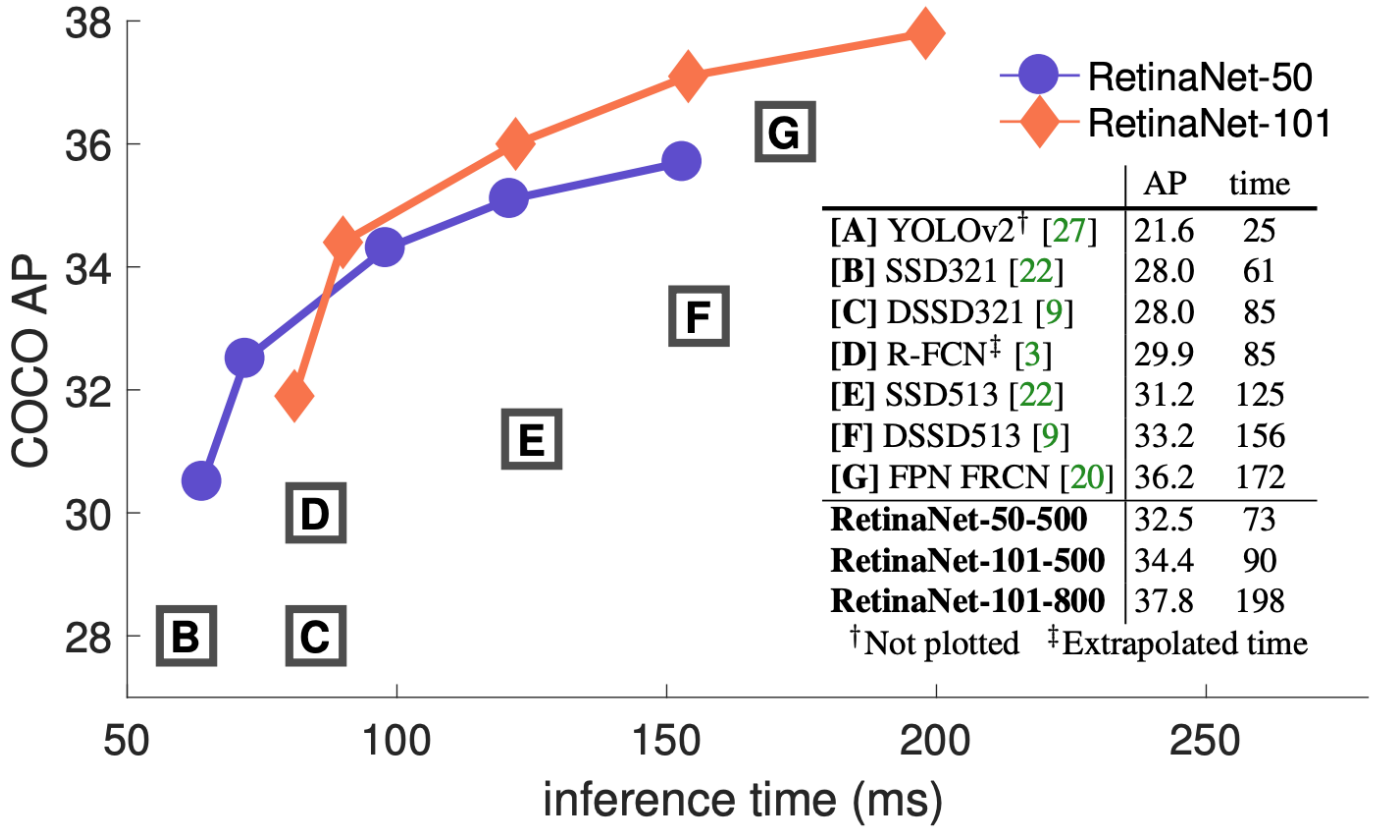
Object detection



https://www.youtube.com/watch?v=h08N0HX16l8

Object detection

RetinaNet

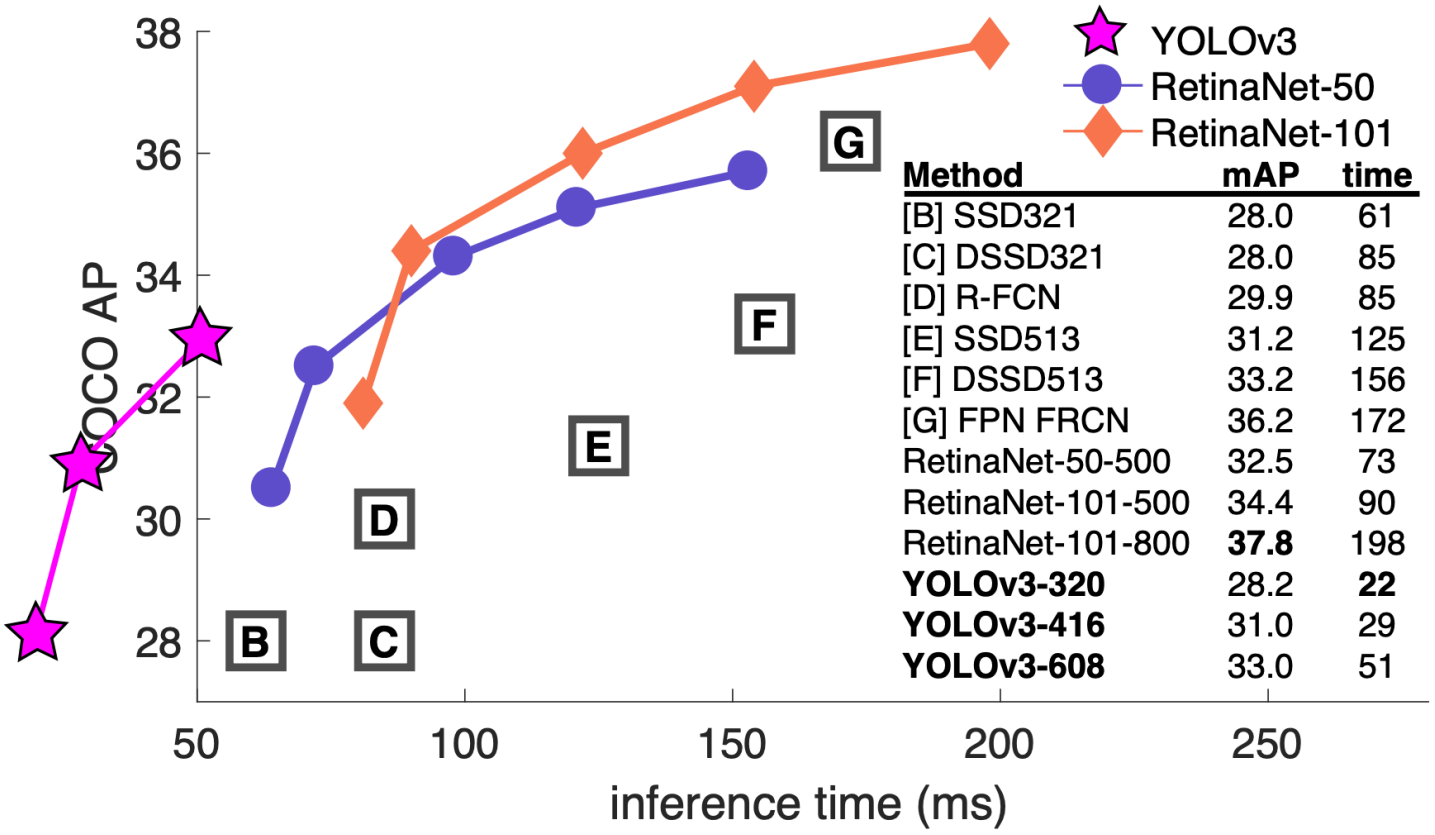


	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

Object detection

YOLO v3

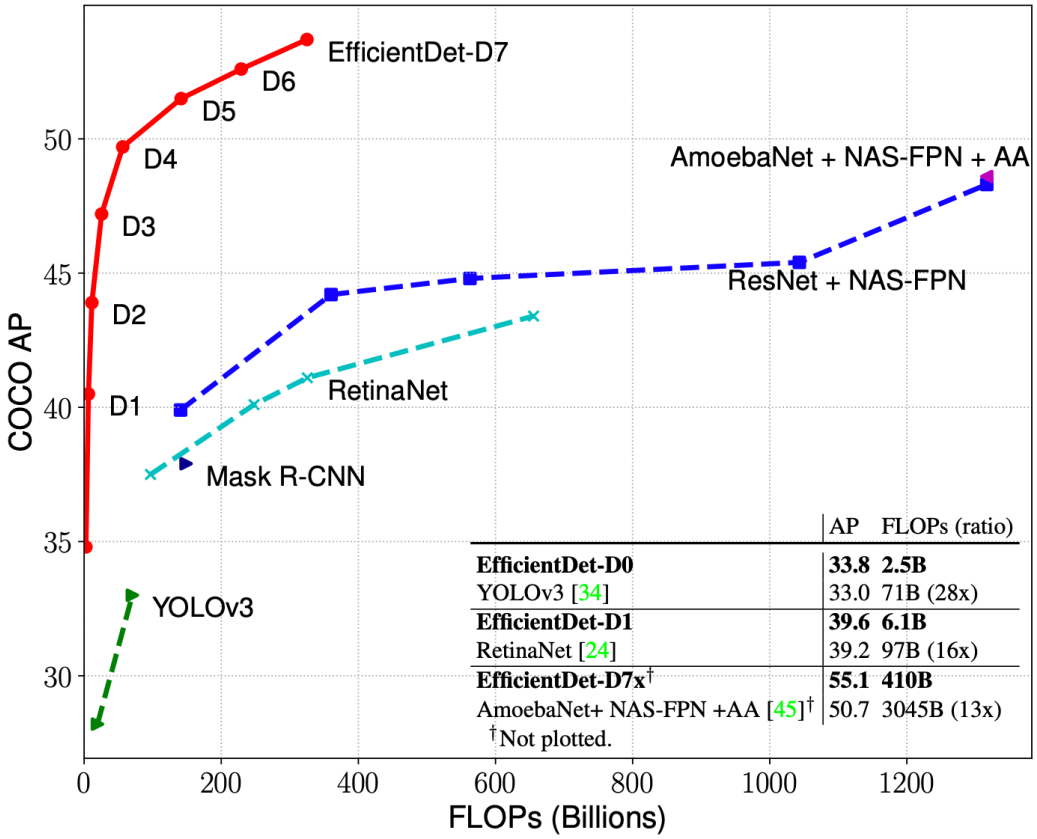


Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Joseph Redmon, Ali Farhadi, YOLOv3: An Incremental Improvement, 2018

Object detection

EfficientDet



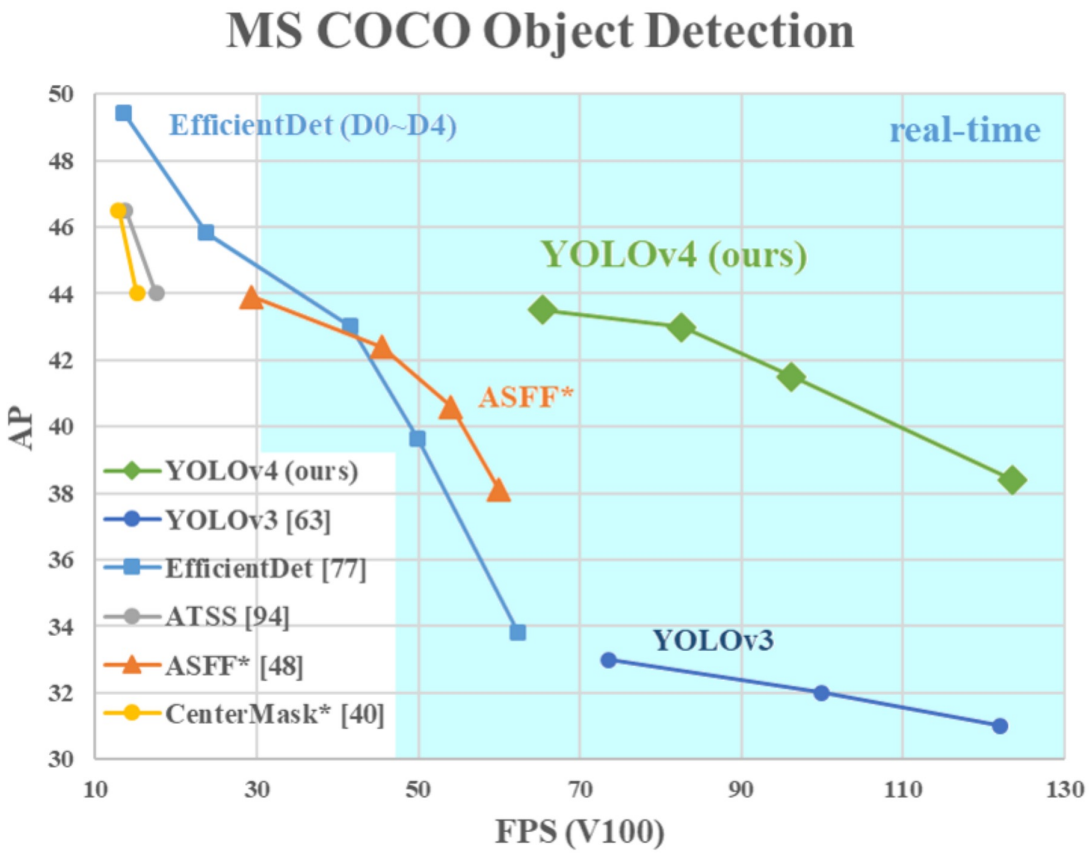
	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [34]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [24]	39.2	97B (16x)
EfficientDet-D7x†	**55.1**	**410B**
AmoebaNet+ NAS-FPN +AA [45]†	50.7	3045B (13x)

†Not plotted.

Tan et al, EfficientDet: Scalable and Efficient Object Detection, 2020

Object detection

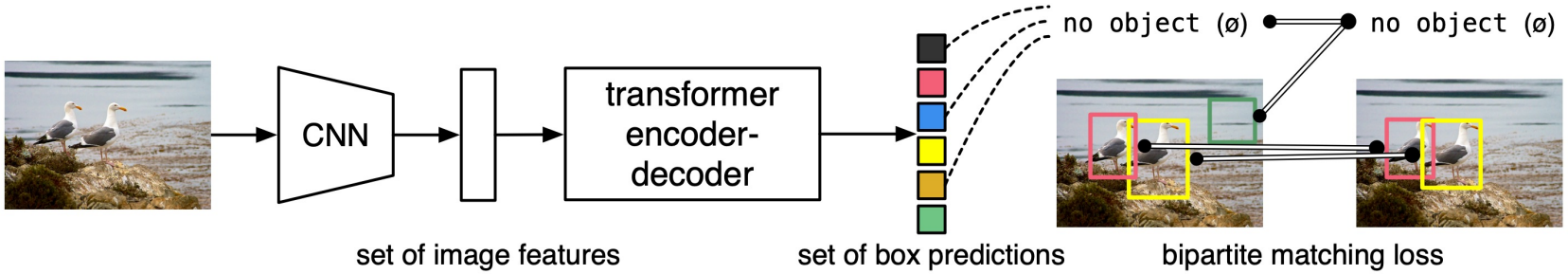
YOLO v4



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

Object detection

DETR: Recognize object detection as a set prediction task



1. First ever end-to-end detection framework (no proxy task, no anchor no proposal, no NMS)
2. Object detection set prediction loss
3. Leverage Transformer, to extract global information, enabling the removal of redundant boxes

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko, End-to-End Object Detection with Transformers, ECCV 2020

Object detection

DETR: Recognize object detection as a set prediction task

- Object detection set prediction loss

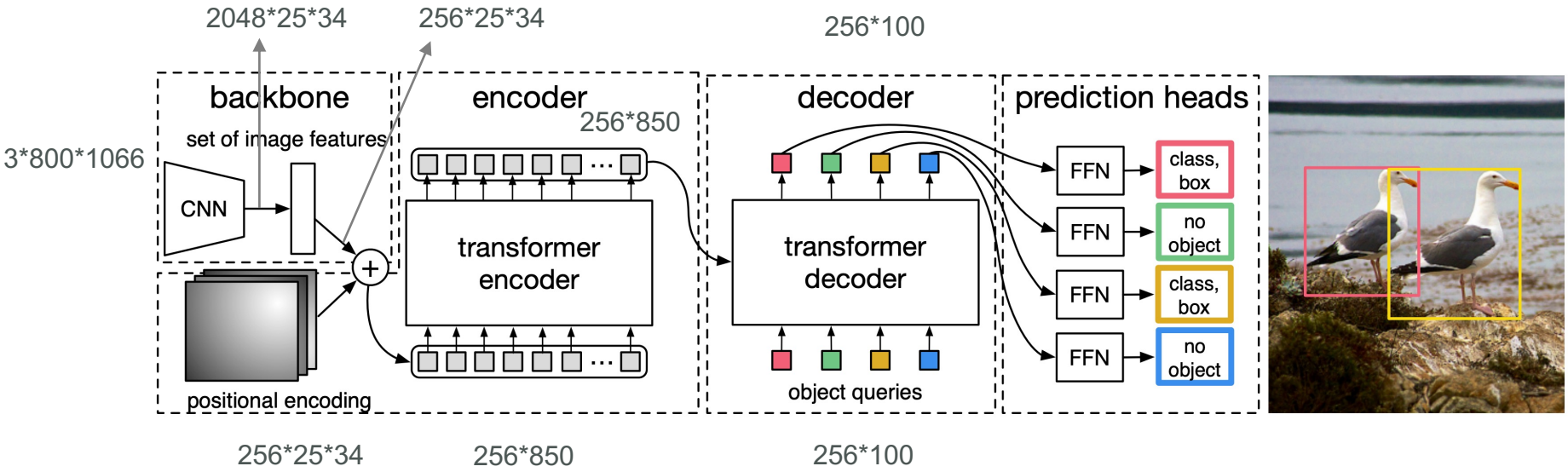
$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$$

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

Object detection

DETR: Recognize object detection as a set prediction task

- Transformer-based architecture



Object detection

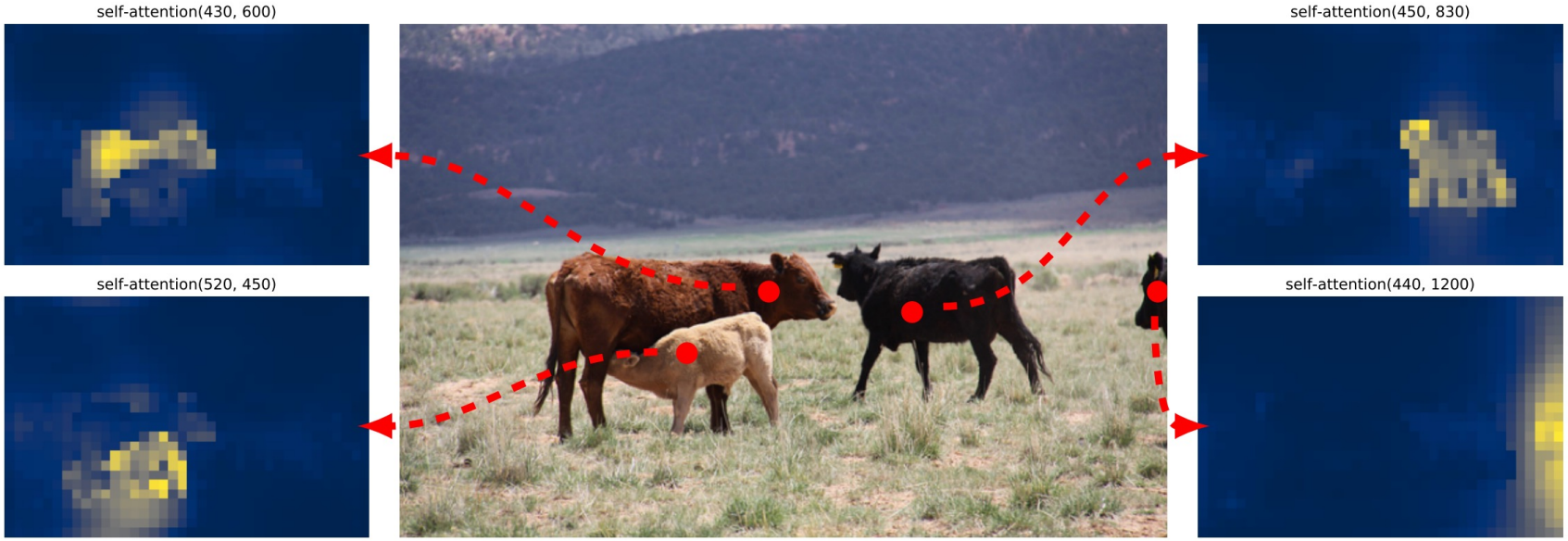
DETR: Recognize object detection as a set prediction task

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

Comparable with Faster R-CNN

Object detection

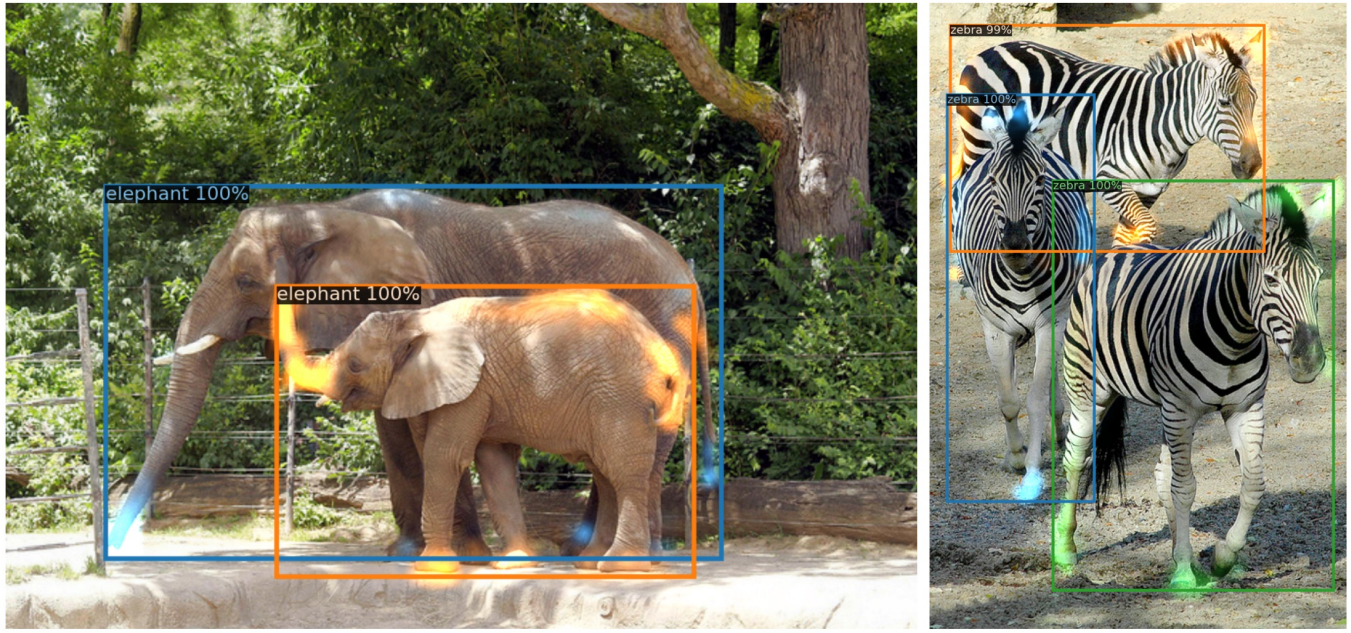
DETR: Recognize object detection as a set prediction task



Encoder self-attention for a set of reference points.
The encoder is able to separate individual instances.

Object detection

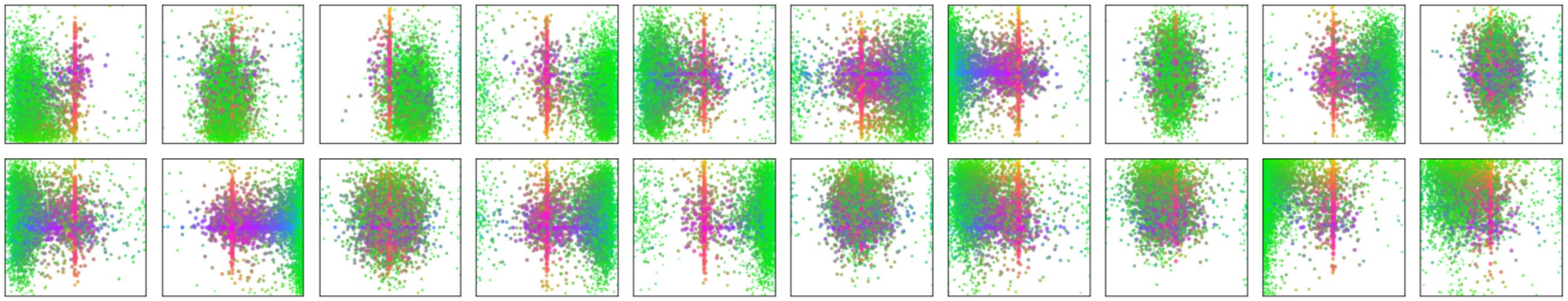
DETR: Recognize object detection as a set prediction task



Attention scores are coded with different colors for different objects.
Decoder typically attends to object extremities, such as legs and heads.

Object detection

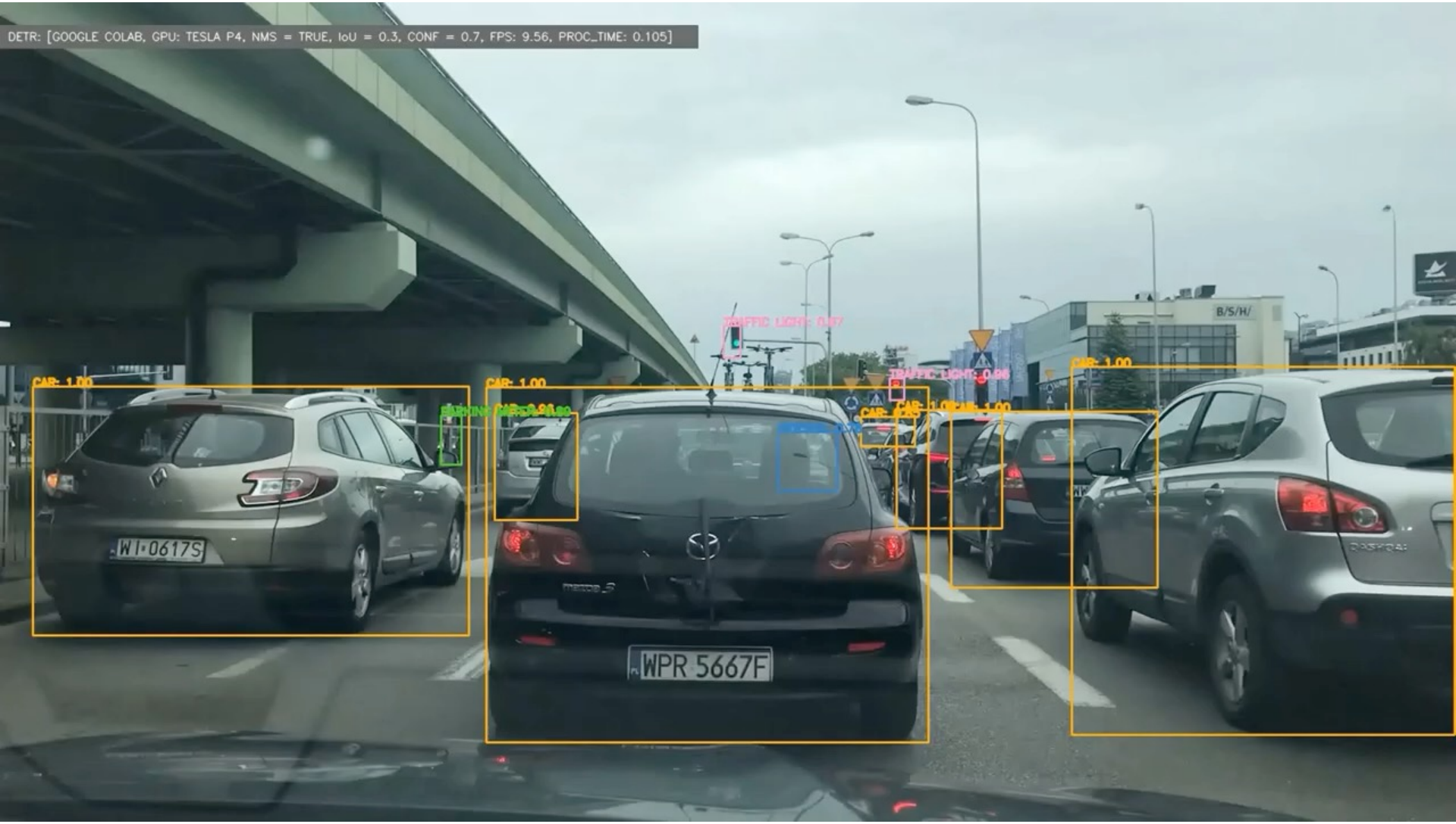
DETR: Recognize object detection as a set prediction task



Each slot learns to specialize on certain areas and box sizes with several operating modes

- Each box prediction is represented as a point with the coordinates of its center in the 1-by-1 square normalized by each image size.
- The points are color-coded so that green color corresponds to small boxes, red to large horizontal boxes and blue to large vertical boxes.

Object detection



https://www.youtube.com/watch?v=nsJ03rRX2K4

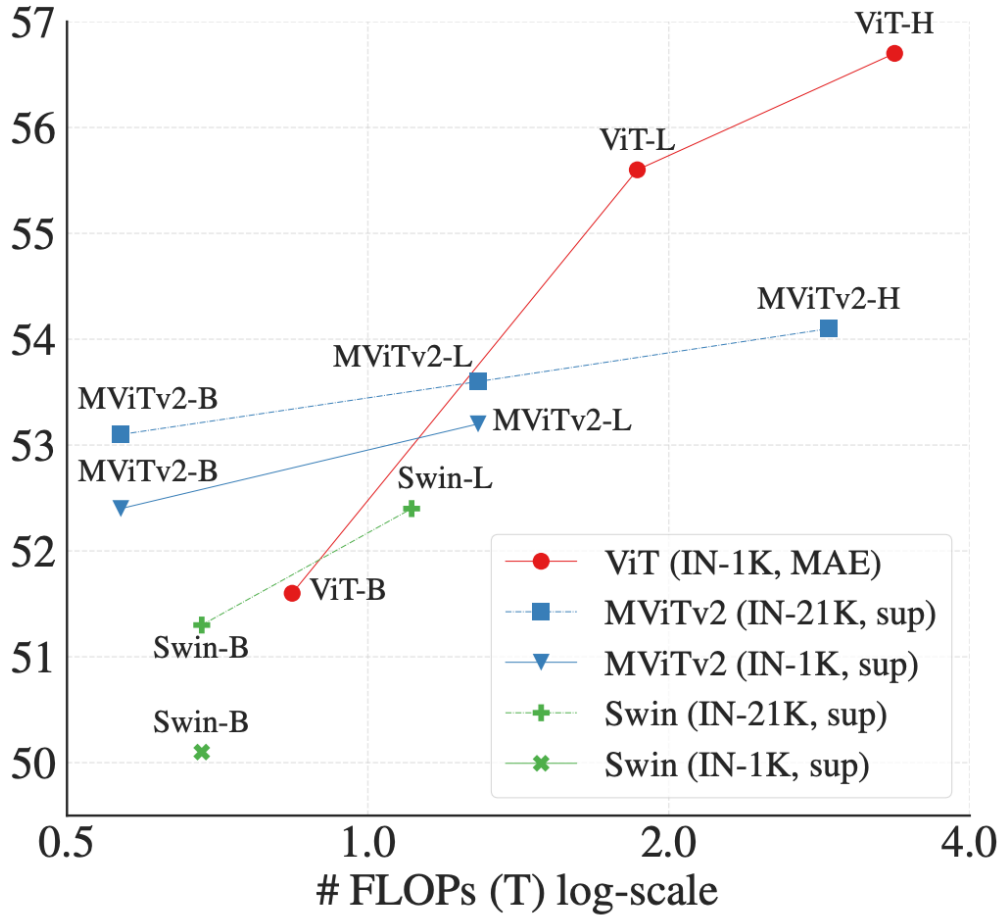
Object detection

Deformable DETR can achieve better performance than DETR (especially on small objects) with 10× less training epochs

Method	Epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	params	FLOPs	Training GPU hours	Inference FPS
Faster R-CNN + FPN	109	42.0	62.1	45.5	26.6	45.4	53.4	42M	180G	380	26
DETR	500	42.0	62.4	44.2	20.5	45.8	61.1	41M	86G	2000	28
DETR-DC5	500	43.3	63.1	45.9	22.5	47.3	61.1	41M	187G	7000	12
DETR-DC5	50	35.3	55.7	36.8	15.2	37.5	53.6	41M	187G	700	12
DETR-DC5+	50	36.2	57.0	37.4	16.3	39.2	53.9	41M	187G	700	12
Deformable DETR	50	43.8	62.6	47.7	26.4	47.1	58.0	40M	173G	325	19
+ iterative bounding box refinement	50	45.4	64.7	49.0	26.8	48.3	61.7	40M	173G	325	19
++ two-stage Deformable DETR	50	46.2	65.2	50.0	28.8	49.2	61.7	40M	173G	340	19

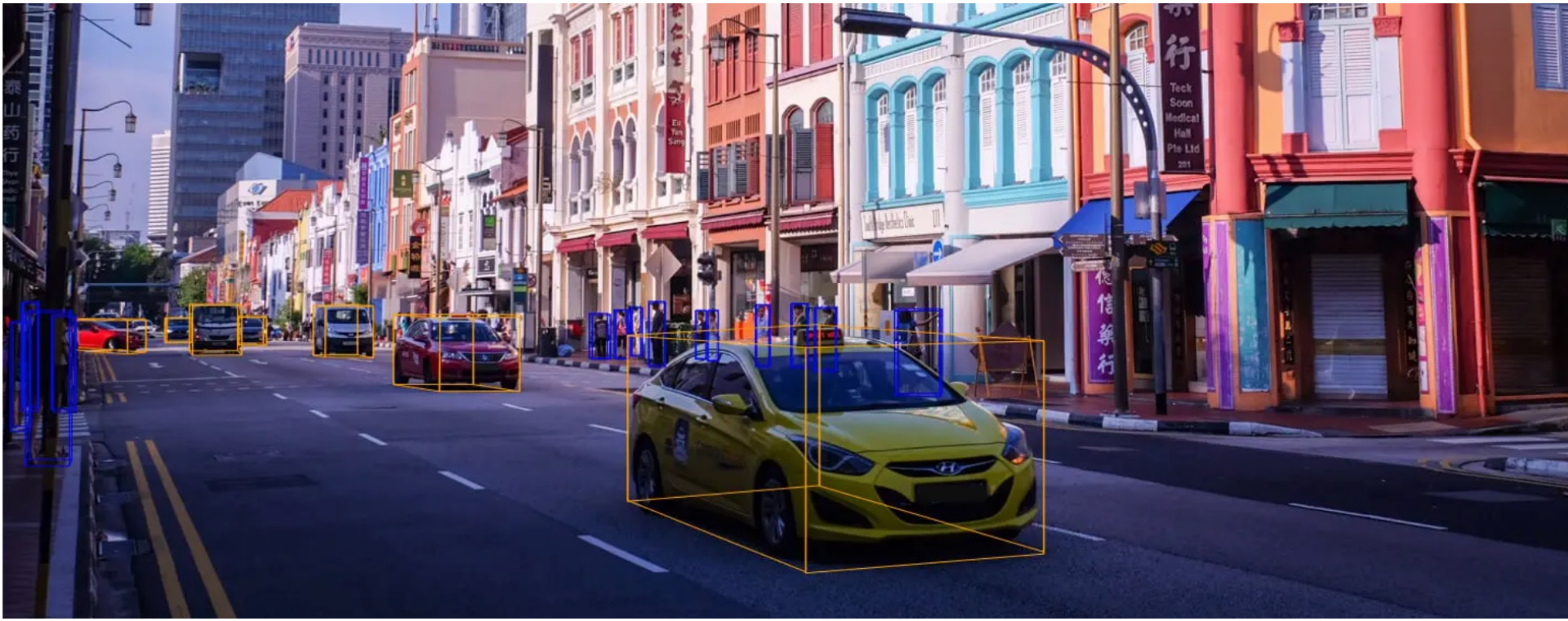
Object detection

ViTDet: Transformer



Yanghao Li, Hanzi Mao, Ross Girshick, Kaiming He, Exploring Plain Vision Transformer Backbones for Object Detection, 2022

Next lecture: 3D detection



Thank you very much!

sihengc@sjtu.edu.cn