Aprendizado Profundo (Deep Learning)

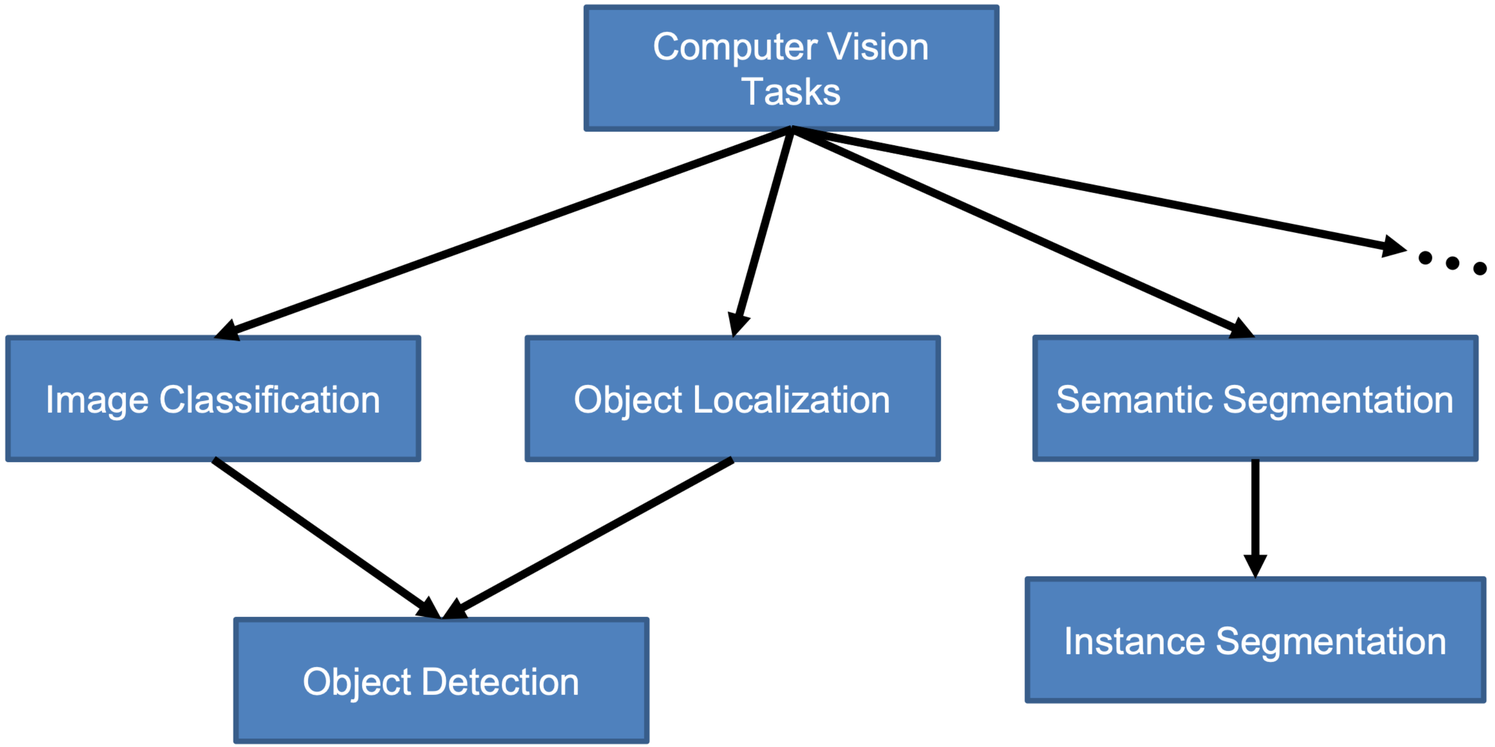
Object Detection and Location

Dario Oliveira (dario.oliveira@fgv.br)

Content

- Computer Vision tasks
- Datasets for Object Detection
- Performance Metrics
- YOLO
- Mask R-CNN

Computer Vision tasks



Computer Vision tasks

CAT

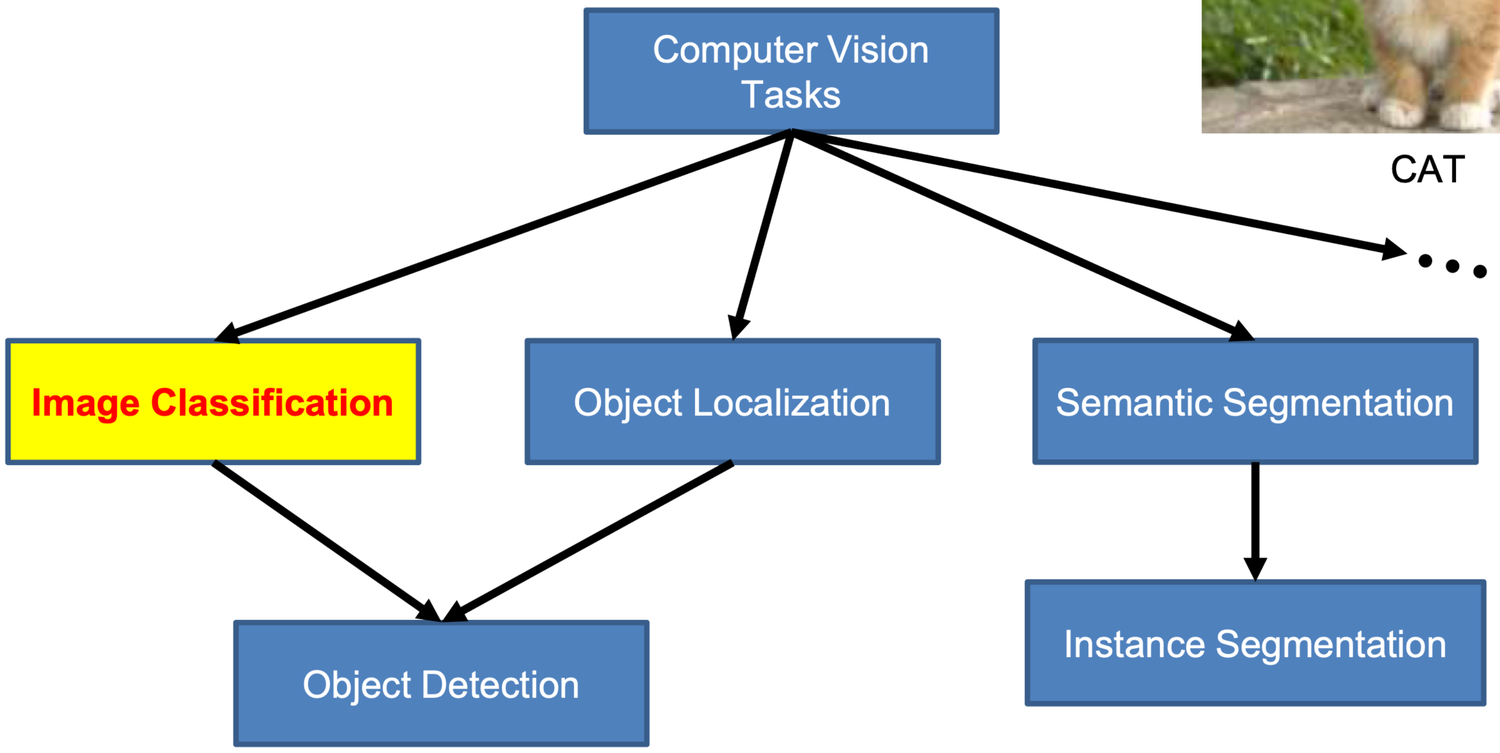
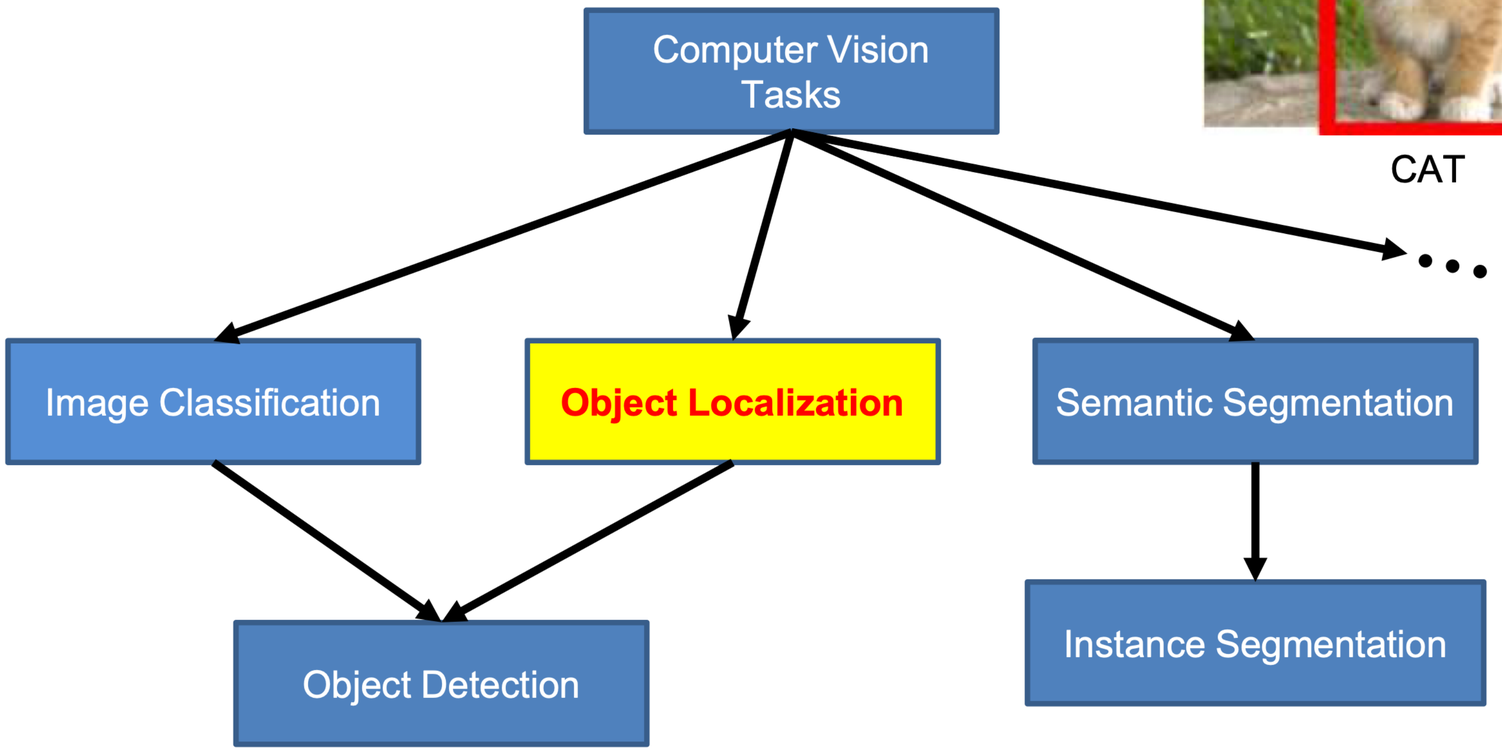


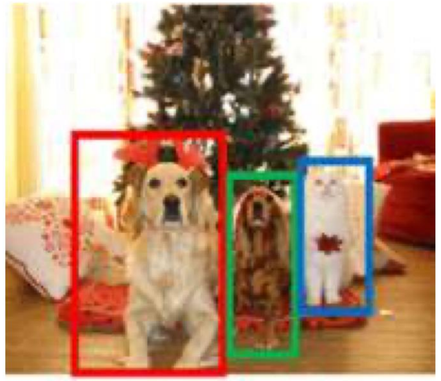
Image Classification: assigns a label to an image.

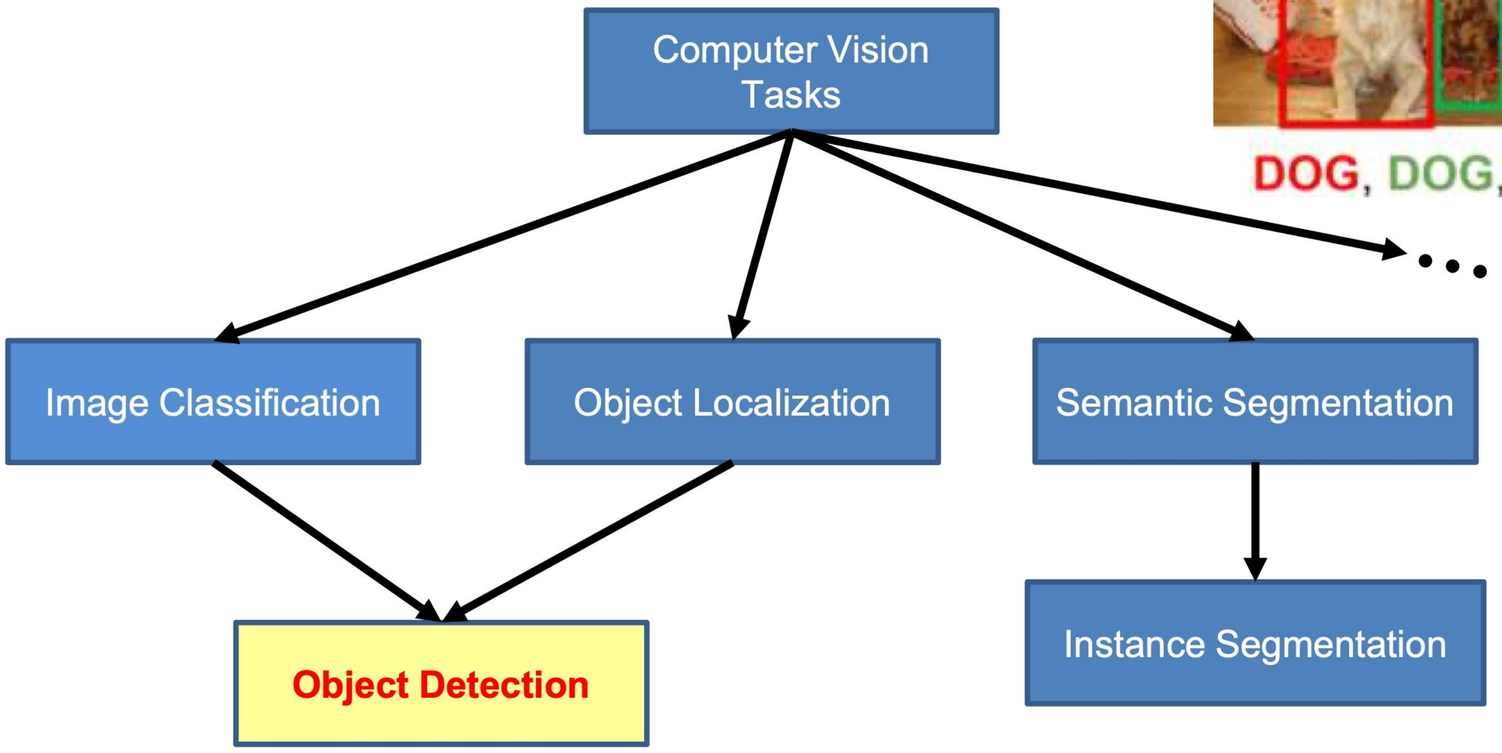
Computer Vision tasks



Object Localization: locates the single object in the image and indicates its location with a bounding box.

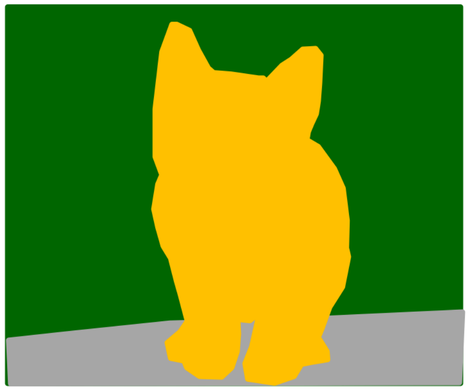
Computer Vision tasks

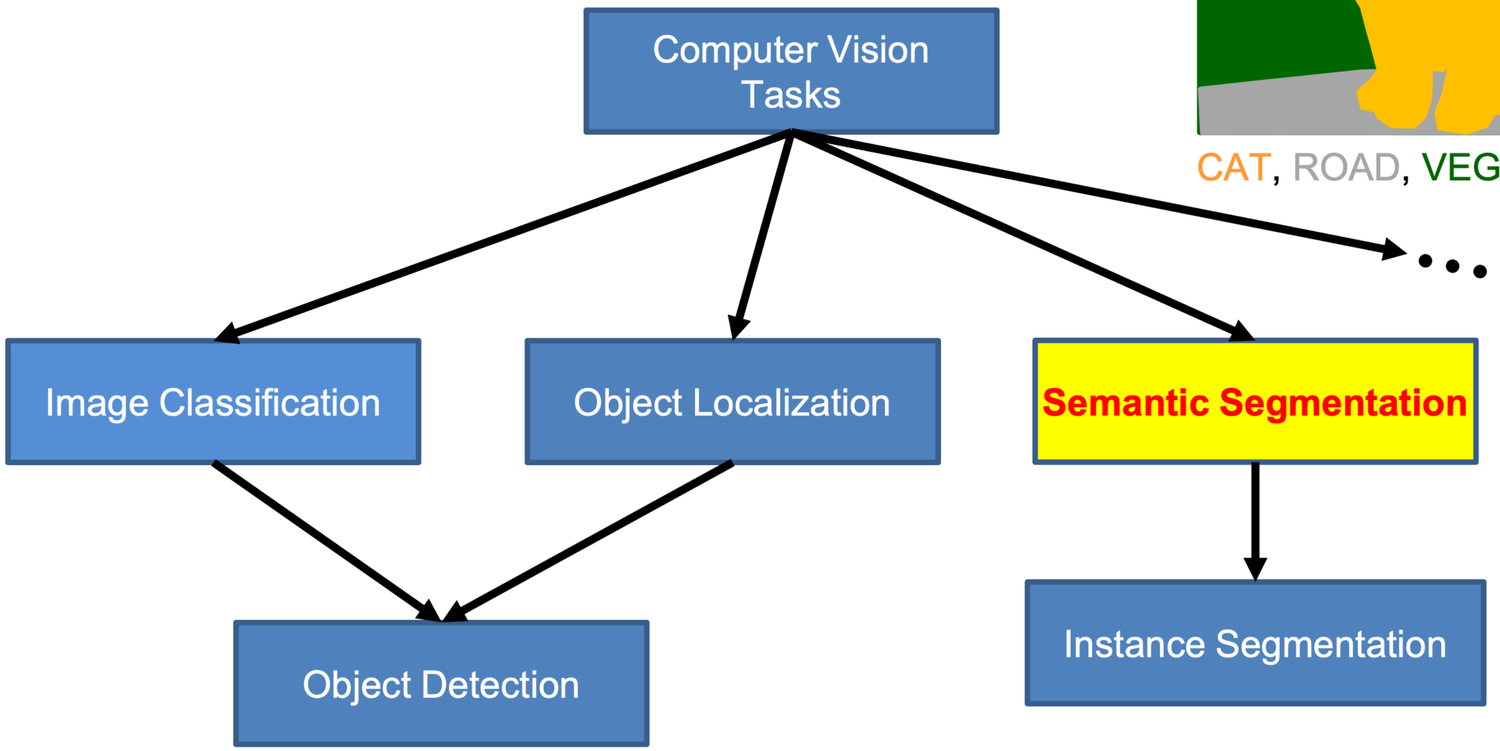




Object Detection: locates multiple (if any) instances of object classes with bounding boxes.

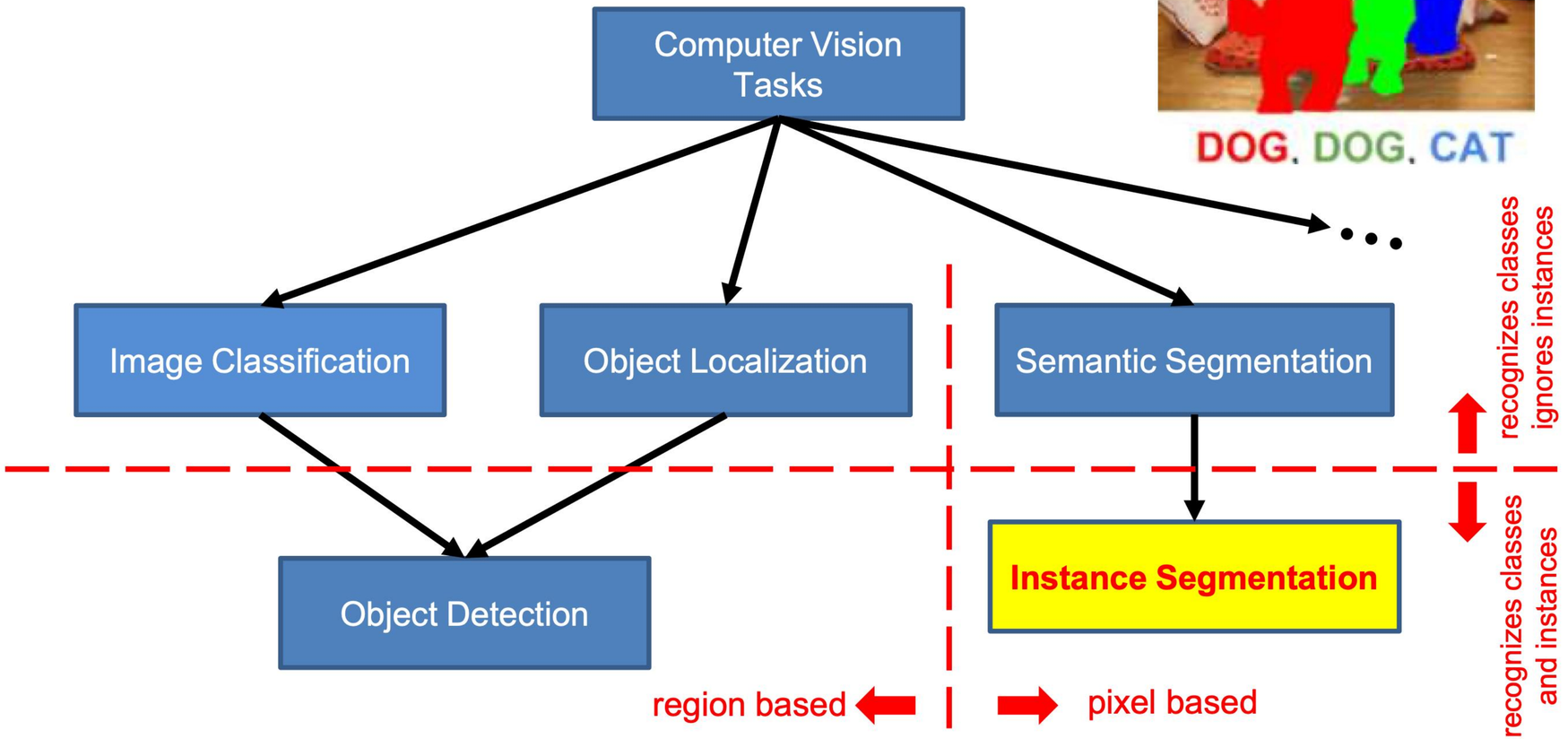
Computer Vision tasks





Semantic Segmentation: links a class label to individual pixels with no concern to separate instances.

Computer Vision tasks



Instance Segmentation: identifies each instance of each object class at the pixel level.

Content

- Computer Vision tasks
- Datasets for Object Detection
- Performance Metrics
- YOLO
- Mask R-CNN

Datasets

- PASCAL Visual Object Classification (PASCAL VOC).
- 8 different challenges from 2005 and 2012.
- Around 10k images of 20 categories.

Source: Everingham et al., 2012 Visual Object Classes Challenge 2012 Dataset

Datasets

- Common Objects in COntext (COCO).
- Multiple challenges.
- Around 200k images of 80 categories.
- 1.5 million object instances.

Source: Arthur Ouaknine, Review of Deep Learning Algorithms for Object Detection (2018)

Content

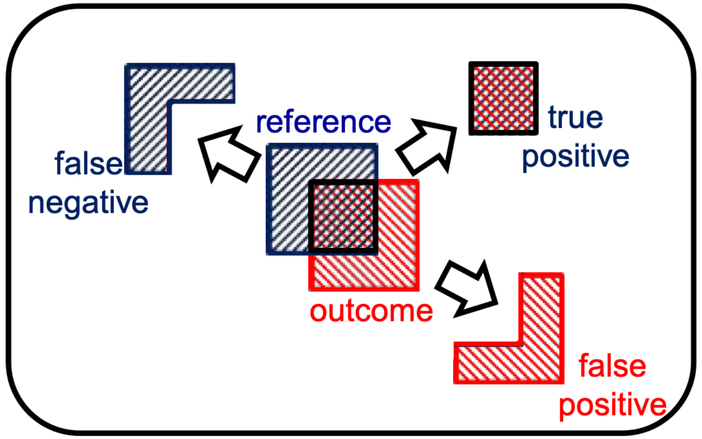
- Computer Vision tasks
- Datasets for Object Detection
- Performance Metrics
- YOLO
- Mask R-CNN

Performance Metrics

$$\textbf{Intersection over Union } (IoU) = \frac{\blacksquare}{\blacksquare}$$

$$\textbf{Precision} = \frac{true\ positive}{true\ positive + false\ positive} = \frac{\blacksquare}{\blacksquare}$$

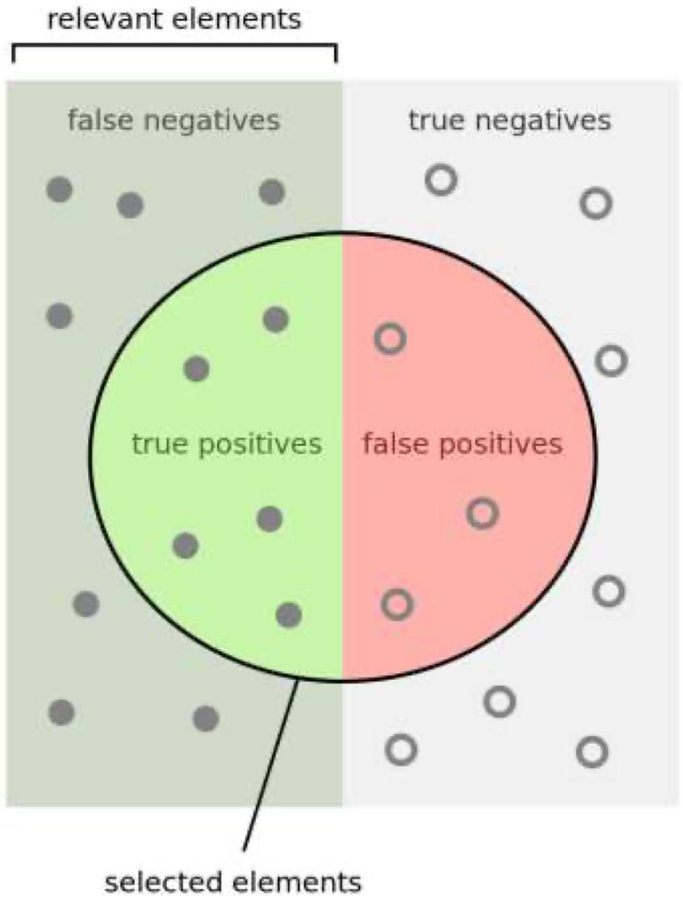
$$\textbf{Recall} = \frac{true\ positive}{true\ positive + false\ negative} = \frac{\blacksquare}{\blacksquare}$$

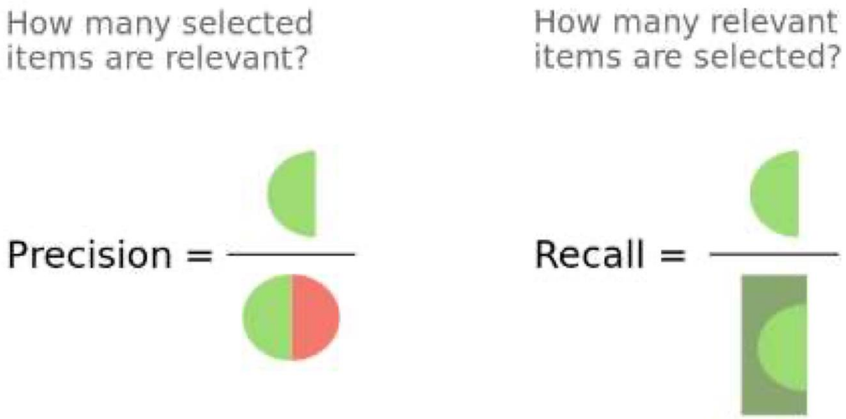


$$\textbf{Average Precision } (AP) = precision\ over\ all\ classes$$

Performance Metrics

Precision and Recall:



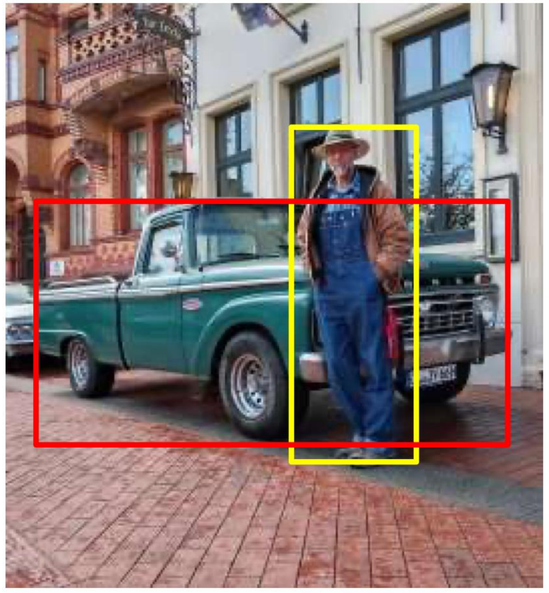


$$\textbf{Precision} = \frac{true\ positive}{true\ positive + false\ positive}$$

$$\textbf{Recall} = \frac{true\ positive}{true\ positive + false\ negatives}$$

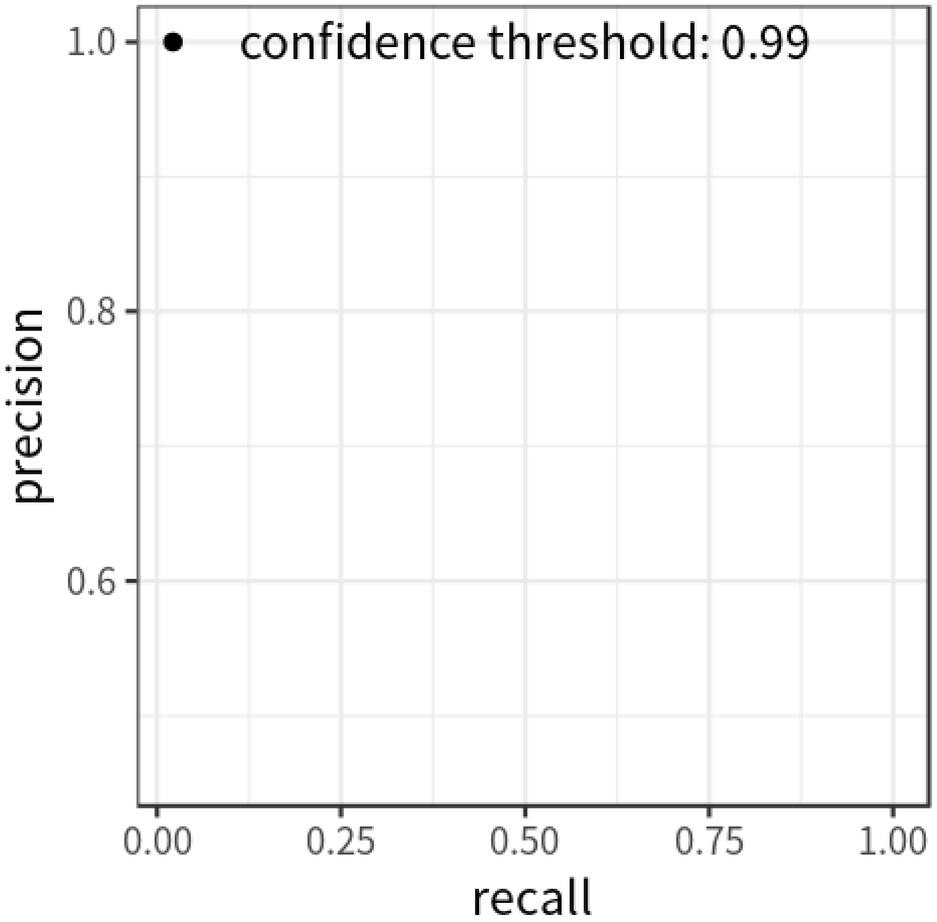
Performance Metrics

- Most competitions use **mean average precision** (mAP) as principal metric to evaluate object detector, though there are variations in definitions and implementations.
- mAP is derived from precision vs. recall values, by varying a **confidence score**.
- The confidence score refers to the probability that an **anchor box** contains an object.



Performance Metrics

Precision-recall curve



As the threshold for the confidence score decreases:

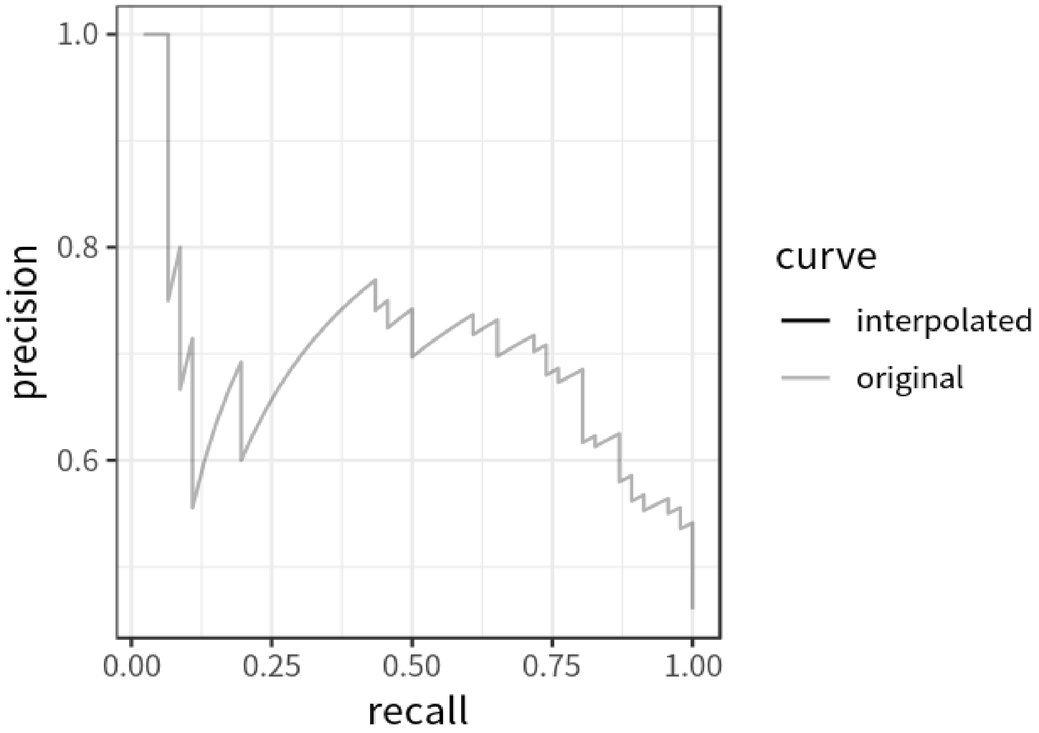
- **recall** increases monotonically;
- **precision** can go up and down, but the general tendency is to decrease.

$$\textbf{\textit{Precision}} = \frac{\text{▮}}{\text{▮}} \qquad \textbf{\textit{Recall}} = \frac{\text{▮}}{\text{▮}}$$

Source: Nickzeng (2018)

Performance Metrics

Precision-recall curve: average precision



Average precision (AP) is the precision averaged across all unique recall levels.

It is based on the interpolated precision-recall curve.

The interpolated precision p_{interp} at a certain recall level r is defined as the highest precision found for any higher recall level:

$$p_{interp}(r) = \max_{r' \geq r}(r')$$

Source: Nickzeng (2018)

Performance Metrics

Mean average precision (mAP)

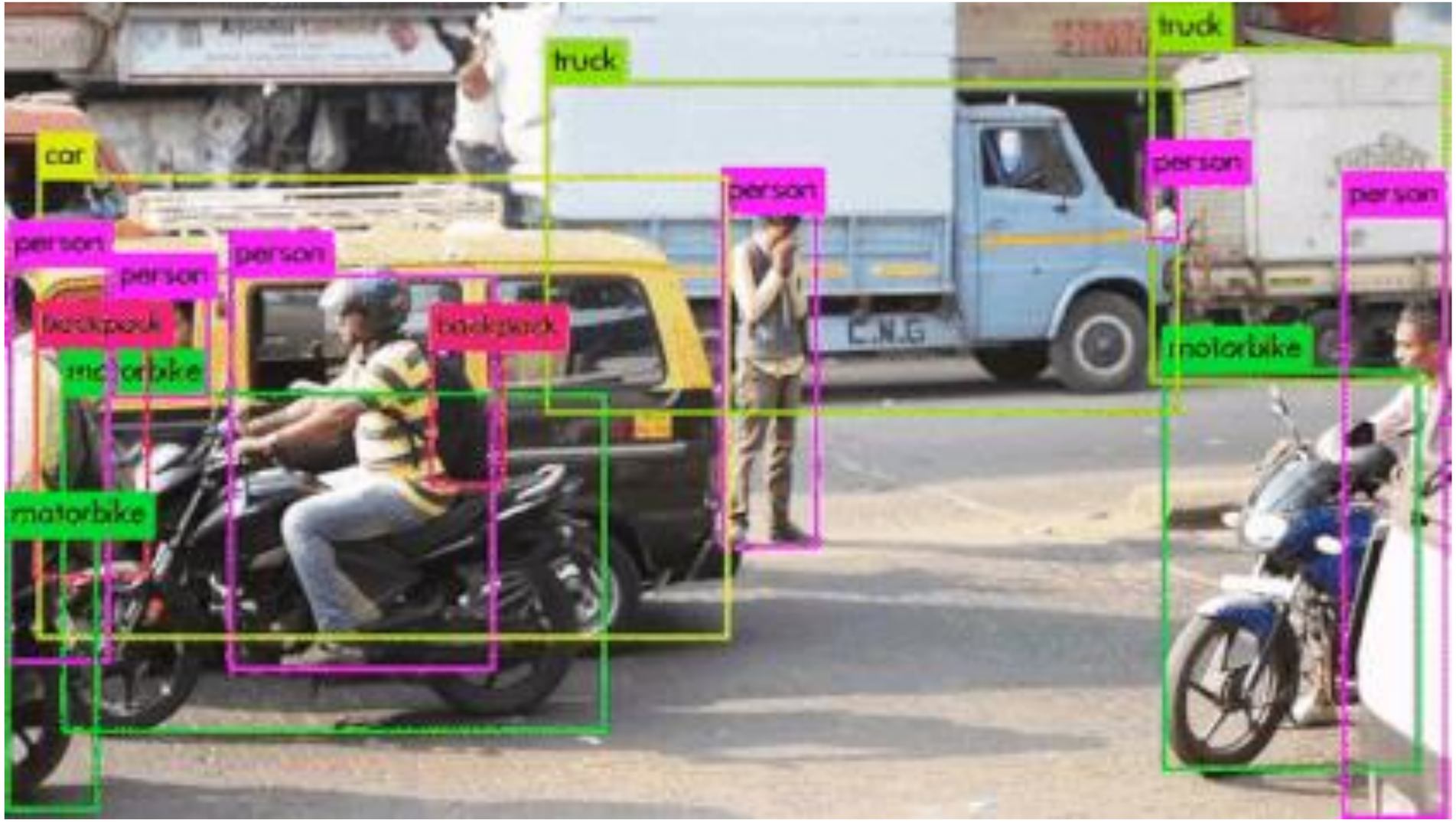
- The calculation of AP involves one class.
- However, in object detection, there are usually $K > 1$ classes.
- Mean average precision (mAP) is defined as the mean of AP across all K classes:

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K}$$

Content

- Computer Vision tasks
- Datasets for Object Detection
- Performance Metrics
- YOLO
- Mask R-CNN

Yolo v3



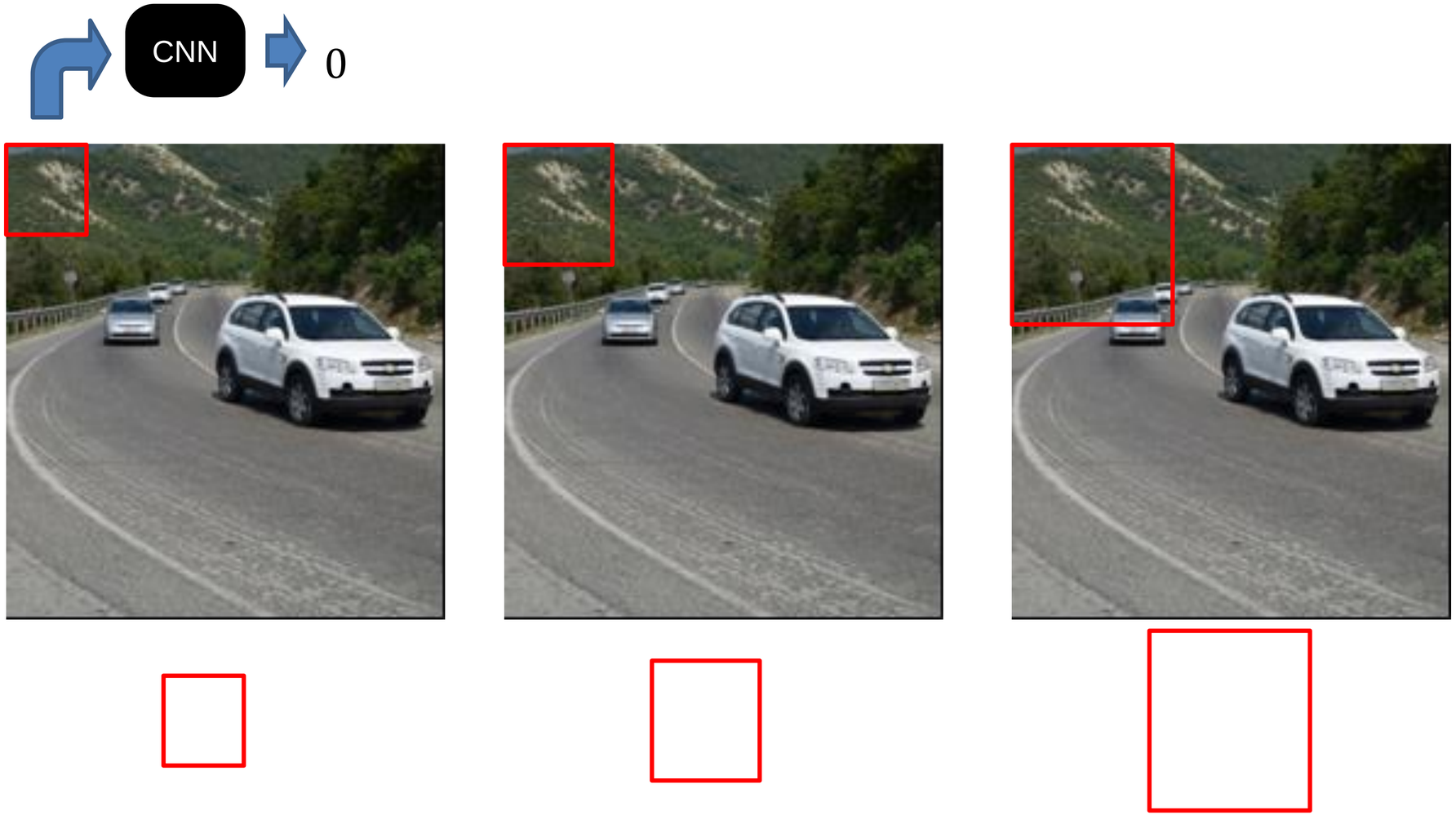
Source: Sik-Ho Tsang Review: YOLOv3 – Yo Only Look Once (Object Detection)

Example of Object Detection

x	y
	1
	1
	1
	0
	0

Training set

Sliding Window Detection



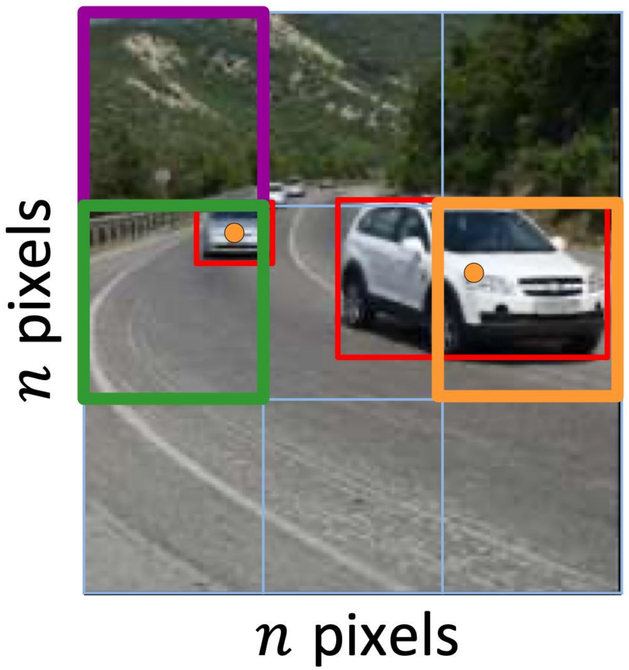
Problem: High computational cost!

Sliding window detection

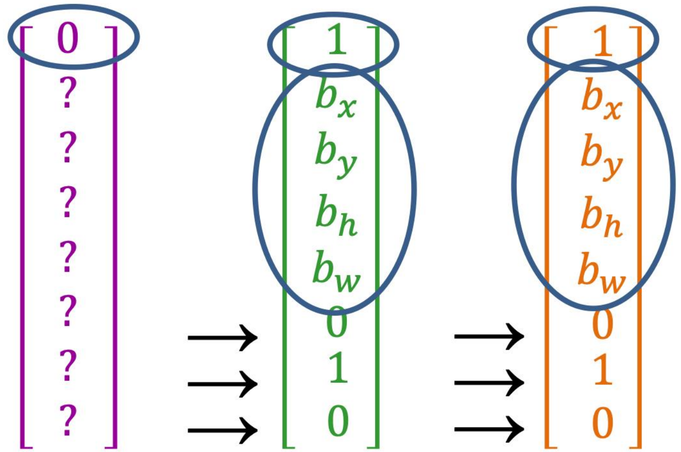
- A small stride would be better: visit nearly all pixels in the input image.
- To decrease computational cost: use a larger stride, at the cost of accuracy.
- Pre-deep learning methods adopted such sliding window procedure with simple classifiers (e.g., linear): acceptable computational effort.
- Using the sliding window procedure having a CNN as the basic classifier is unfeasible.

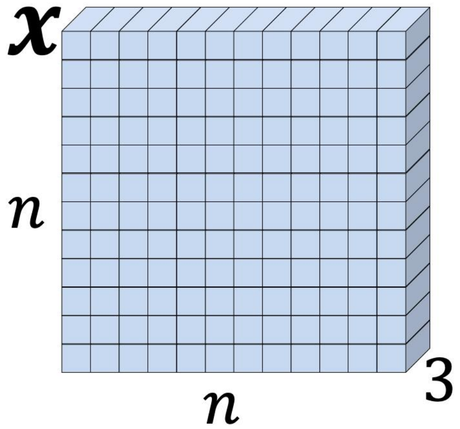
Accurate Bounding Boxes

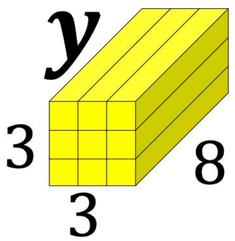
YOLO basics



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





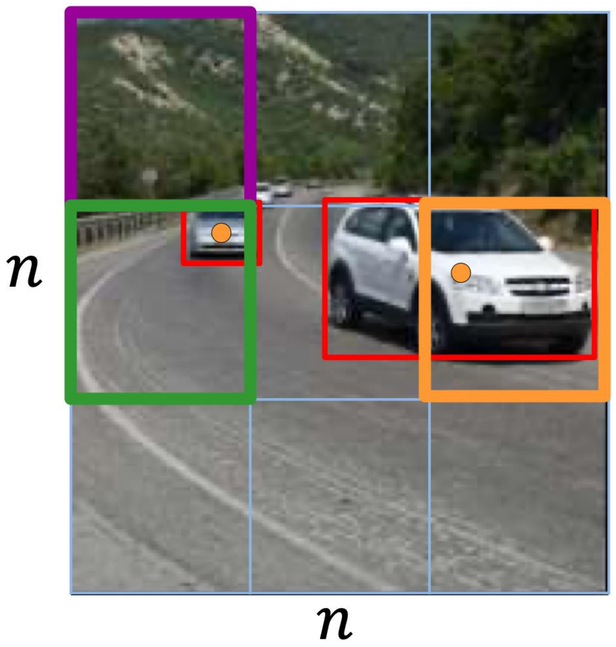


Training labels:

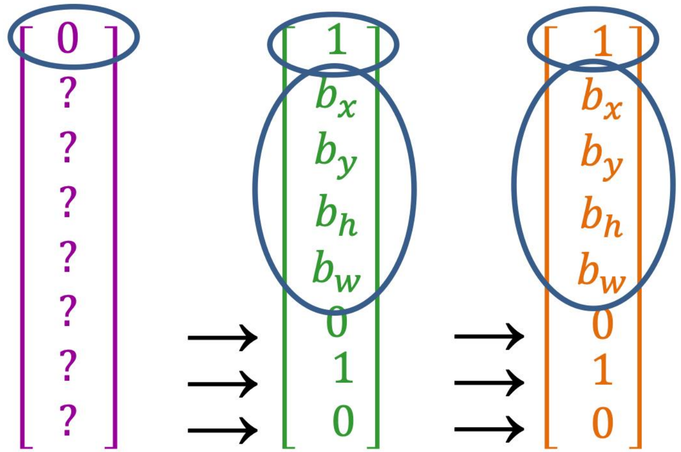
- Contains the bounding box coordinates for each grid cell.
- Indicates if there is an object in each grid cell, and from which class.
- The object is assigned to just one grid cell: the one that contains its midpoint.

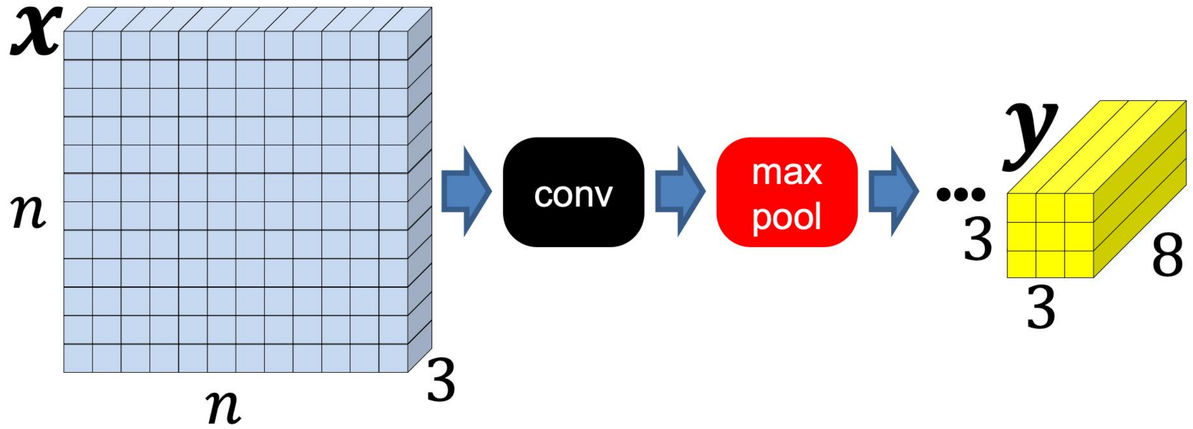
Redmon, et al., 2016, You Only Look Once: Unified, Real-Time Object Detection

YOLO basics



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

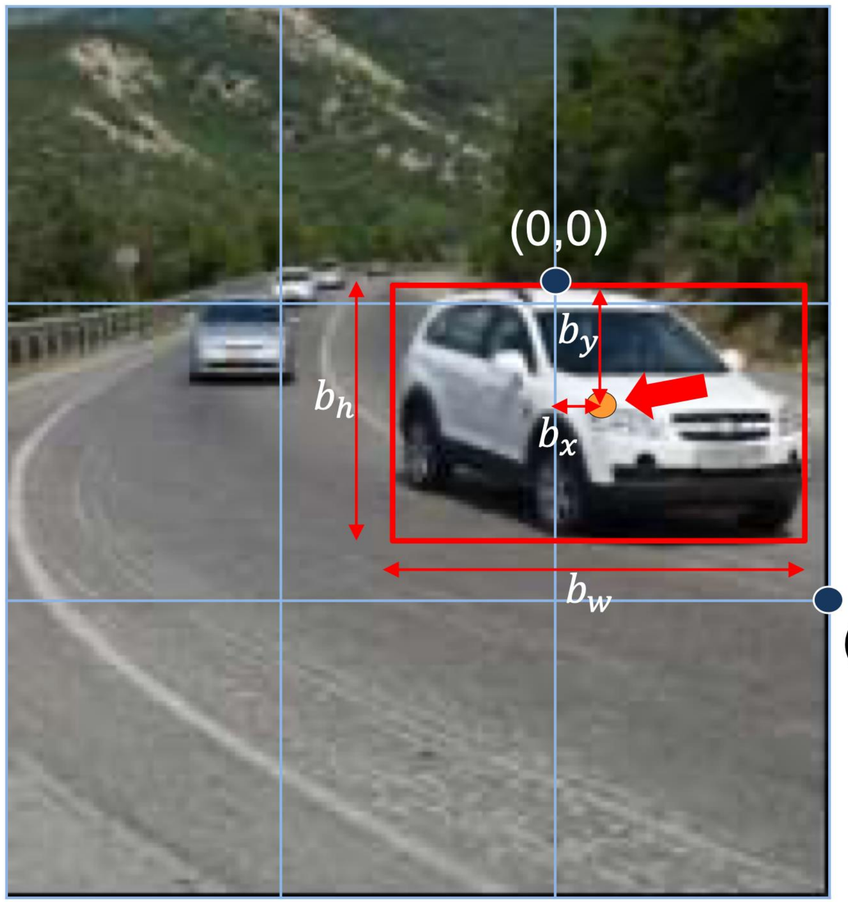




- With finer grids you reduce the chance of multiple objects in the same grid cell.
- A single CNN accounts for all objects in the image: efficient!

Redmon, et al., 2016, You Only Look Once: Unified, Real-Time Object Detection

Parametrization of box coordinates



$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \\ \rightarrow 0.2 \\ \rightarrow 0.5 \\ \rightarrow 0.9 \\ \rightarrow 1.5 \\ \\ \\ \end{matrix}$$

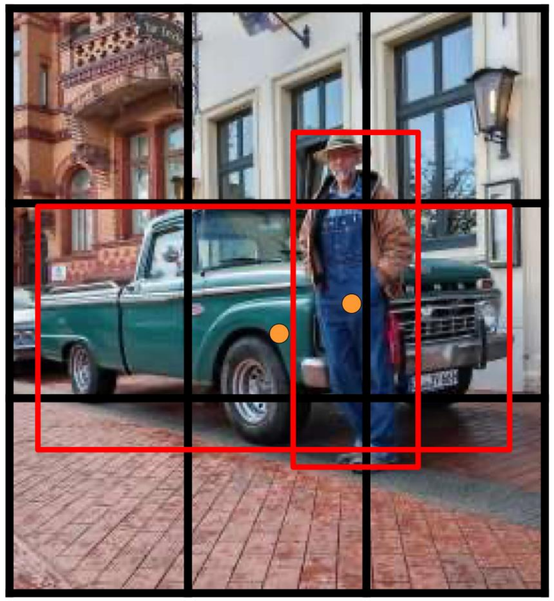
between 0 and 1

can be > 1

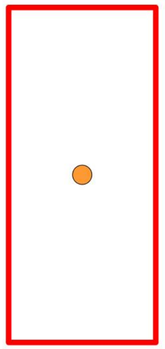
There are many other parametrizations.

Redmon, et al., 2016, You Only Look Once: Unified, Real-Time Object Detection

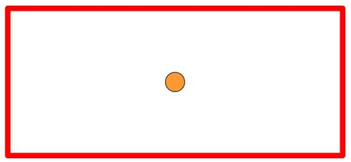
Overlapping objects



anchor box 1



anchor box 2



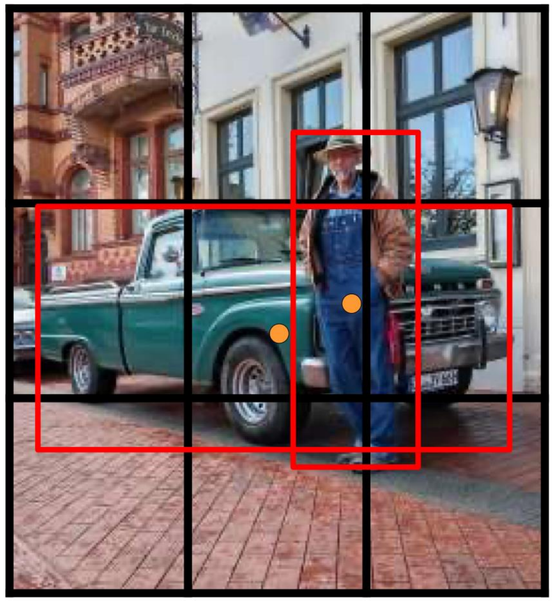
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1

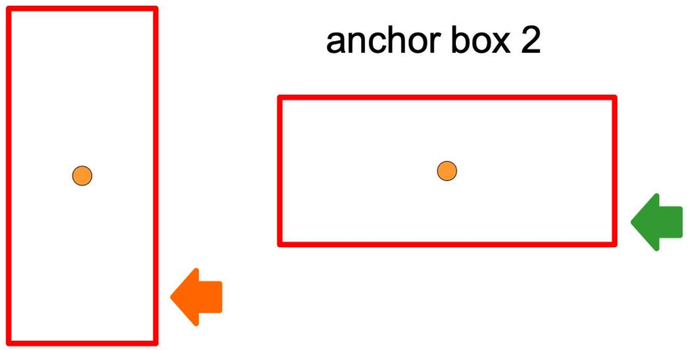
Anchor box 2

Source: Andrew Ng. https://www.youtube.com/watch?v=RTlwl2bv0Tg

Dealing with overlapping objects



anchor box 1



anchor box 2

car only?

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{anchor} \\ \text{box 1} \\ \\ \\ \\ \\ \\ \\ \\ \text{anchor} \\ \text{box 2} \\ \\ \\ \\ \\ \\ \end{matrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Source: Andrew Ng. https://www.youtube.com/watch?v=RTlwl2bv0Tg

Anchor box algorithm

- Previously:

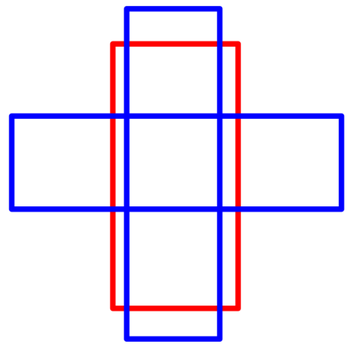
  Each object in training image was assigned to the grid cell that contains that object's midpoint.

- With two anchor boxes:

  Each object in training image is assigned to the grid cell that contains object's midpoint *and* to the anchor box for the grid cell with highest IoU.

Previously $\boldsymbol{y}$: 3×3×8

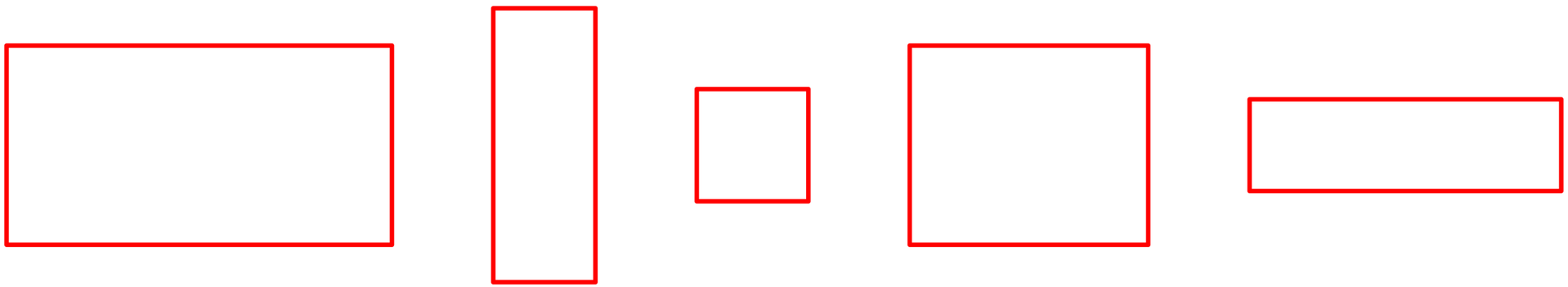
Two anchor boxes $\boldsymbol{y}$: 3×3×16



Source: Andrew Ng. https://www.youtube.com/watch?v=RTlwl2bv0Tg

Selecting anchor box priors (shapes and sizes)

- If you have in your application tall thin objects, or let's say round objects, you choose anchor boxes accordingly.
- Common is to select 5 to 10 anchor boxes this way.
- A more advanced way to do so is to use some clustering algorithm to find out a limited number of anchor boxes that represent the shapes that often occurs in your application.



Source: Redmon, J. & Farhadi, A., 2016 YOLO9000: Better, Faster Stronger

Selecting anchor box priors (shapes and sizes)

- Use a clustering algorithm to find representative shapes.

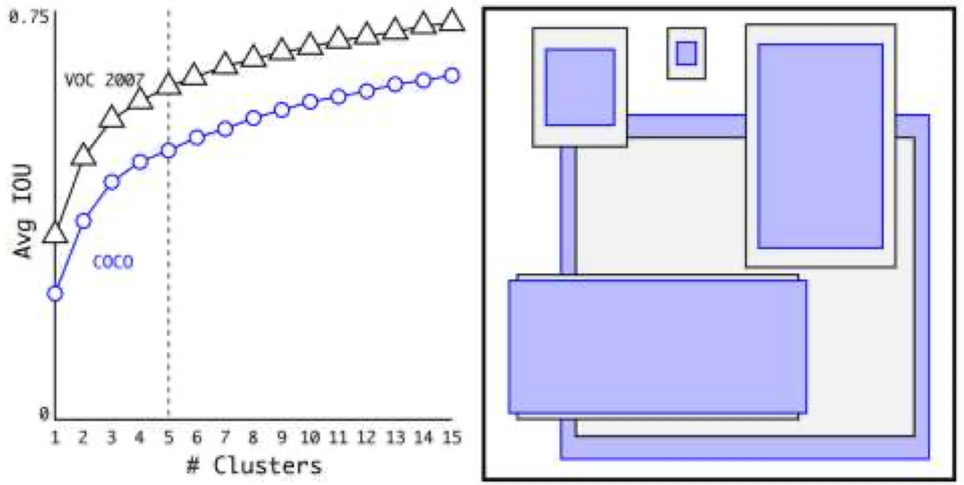
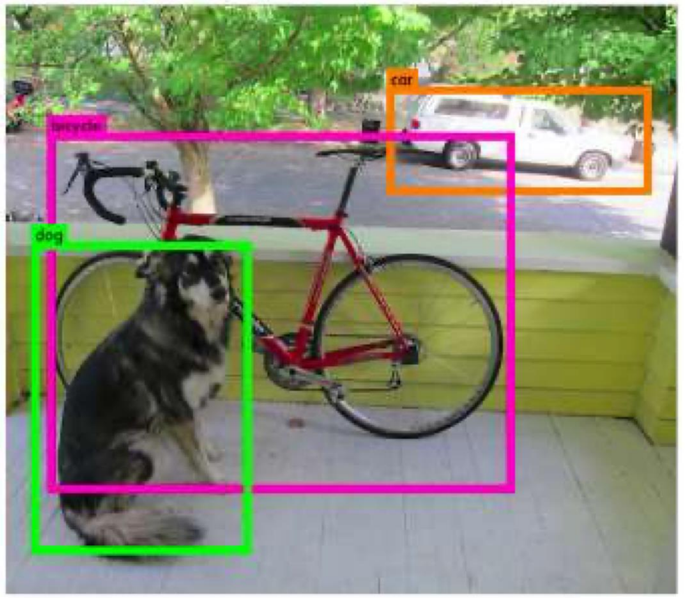


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that k = 5 gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

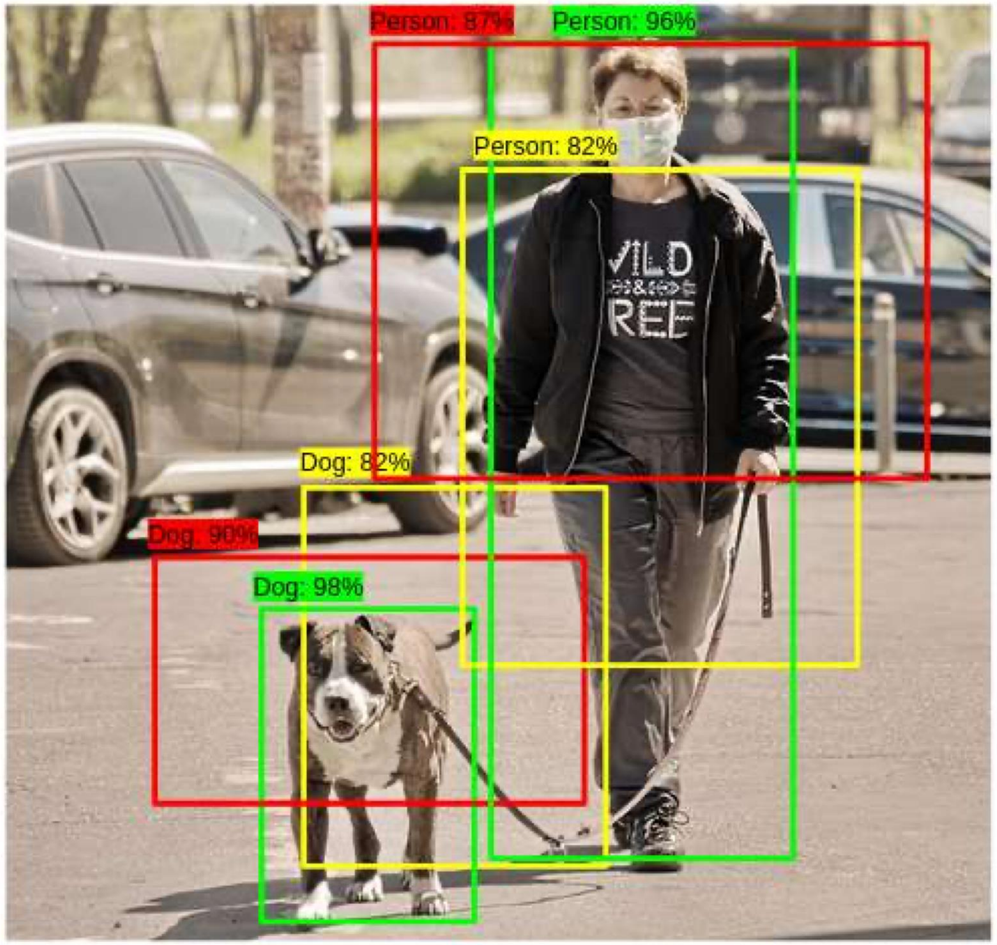
Source: Redmon, J. & Farhadi, A., 2016 YOLO9000: Better, Faster Stronger

Overlapping boxes

- In practice a lot of boxes will be generated: as many as possible.
- After the network inference, most boxes need to be discarded.
- The basic idea is: if two boxes for the same object class overlap two much, select the one with the largest objecness.



Overlapping boxes: non-maximum suppression

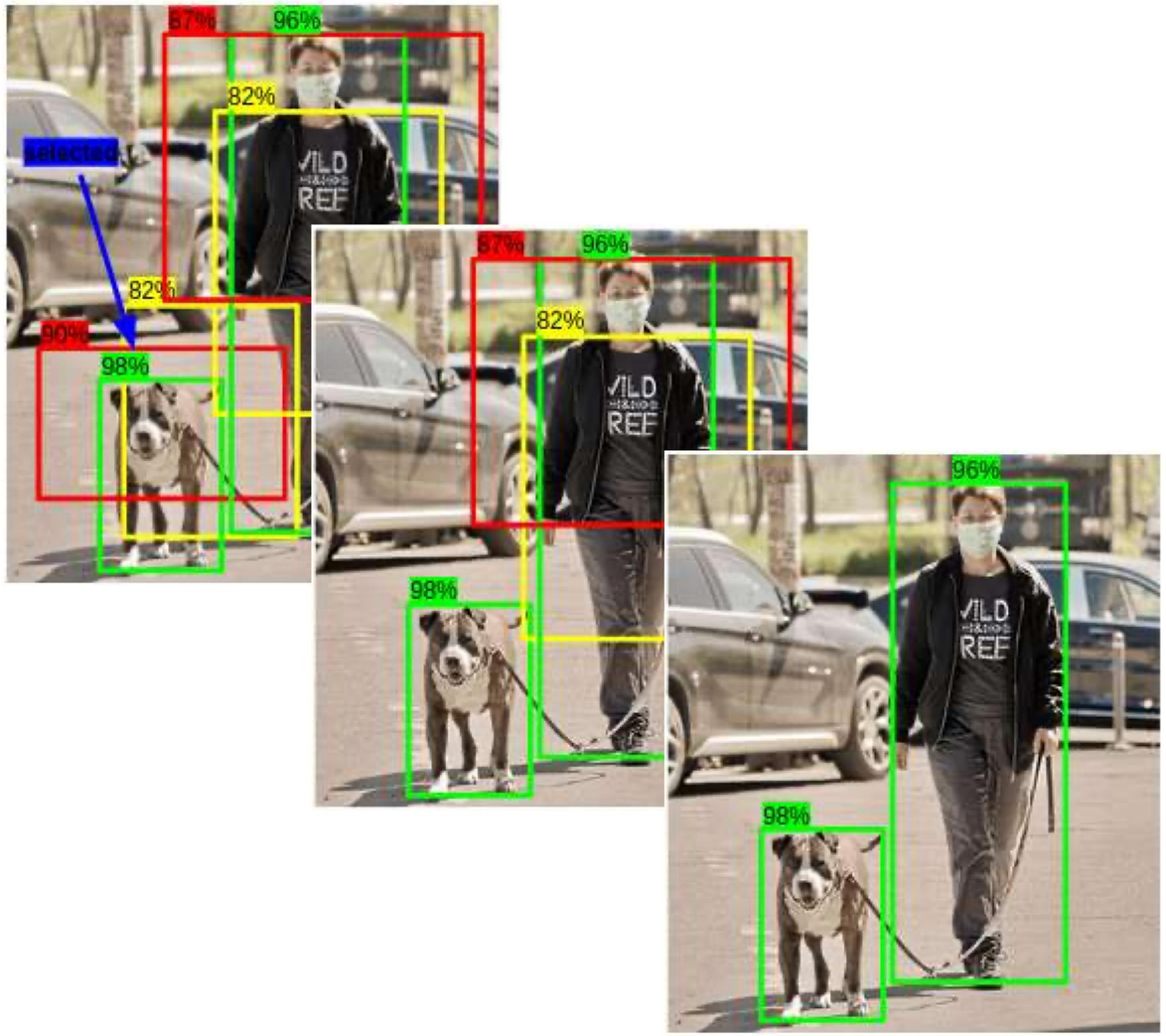


Often generated boxes overlap!

For each class:

- Discard all boxes with $p_c \leq 0.6$
- Pick the box with the largest p_c and output it as a prediction.
- Discard other boxes with $IoU \geq 0.5$ with the box output of the previous step, and repeat for the remaining box with the next largest p_c.

Overlapping boxes: non-maximum suppression



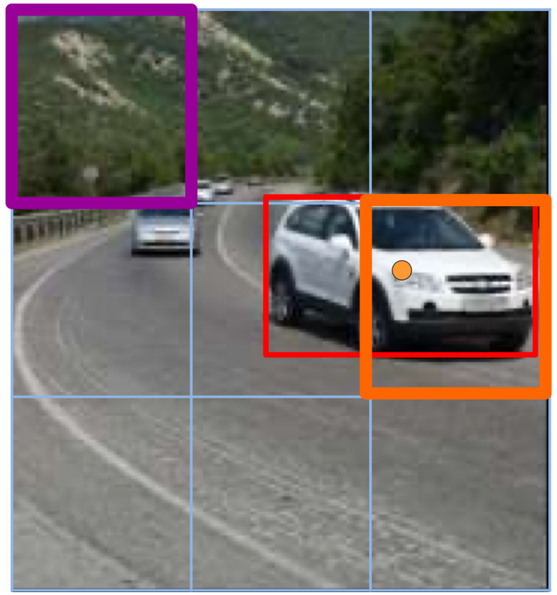
Often generated boxes overlap!

For each class:

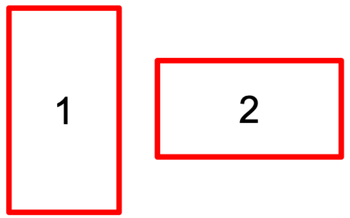
- Discard all boxes with $p_c \leq 0.6$
- Pick the box with the largest p_c and output it as a prediction.
- Discard other boxes with $IoU \geq 0.5$ with the box output of the previous step, and repeat for the remaining box with the next largest p_c.

https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

YOLO training



1. Pedestrian
2. Car
3. Motorcycle

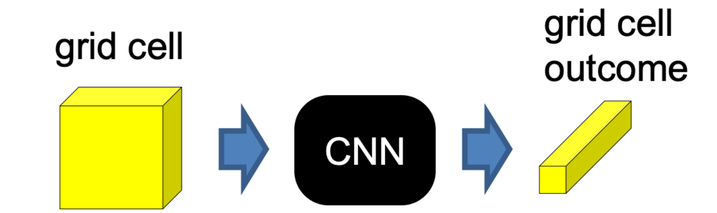


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{anchor} \\ \text{box 1} \\ \\ \\ \\ \\ \\ \\ \\ \text{anchor} \\ \text{box 2} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \\ \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix}$$

In our example

y: 3×3×2×8

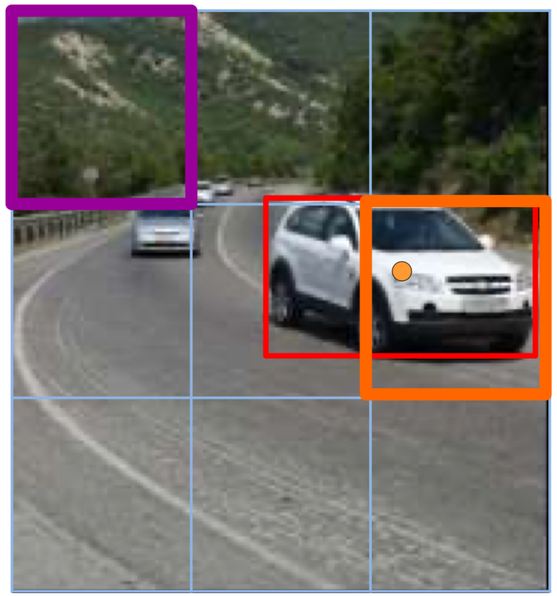
- 5+ #classes
- # anchors
- grid cells



$m \times m \times 3$ $\quad$ $1 \times 1 \times 16$

Source: Andrew Ng. https://www.youtube.com/watch?v=RTlwl2bv0Tg

YOLO prediction



CNN

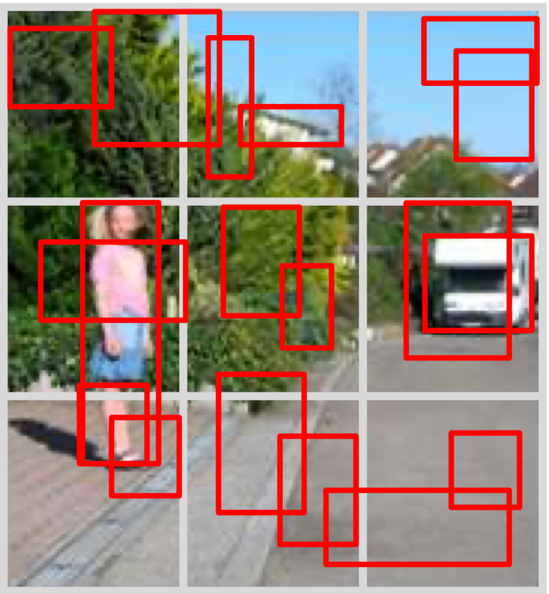
$1 \times 1 \times 16$

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{anchor} \\ \text{box 1} \end{matrix} \begin{bmatrix} \mathbf{0} \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{anchor} \\ \text{box 2} \end{matrix}$$

Source: Andrew Ng. https://www.youtube.com/watch?v=RTlwl2bv0Tg

YOLO prediction

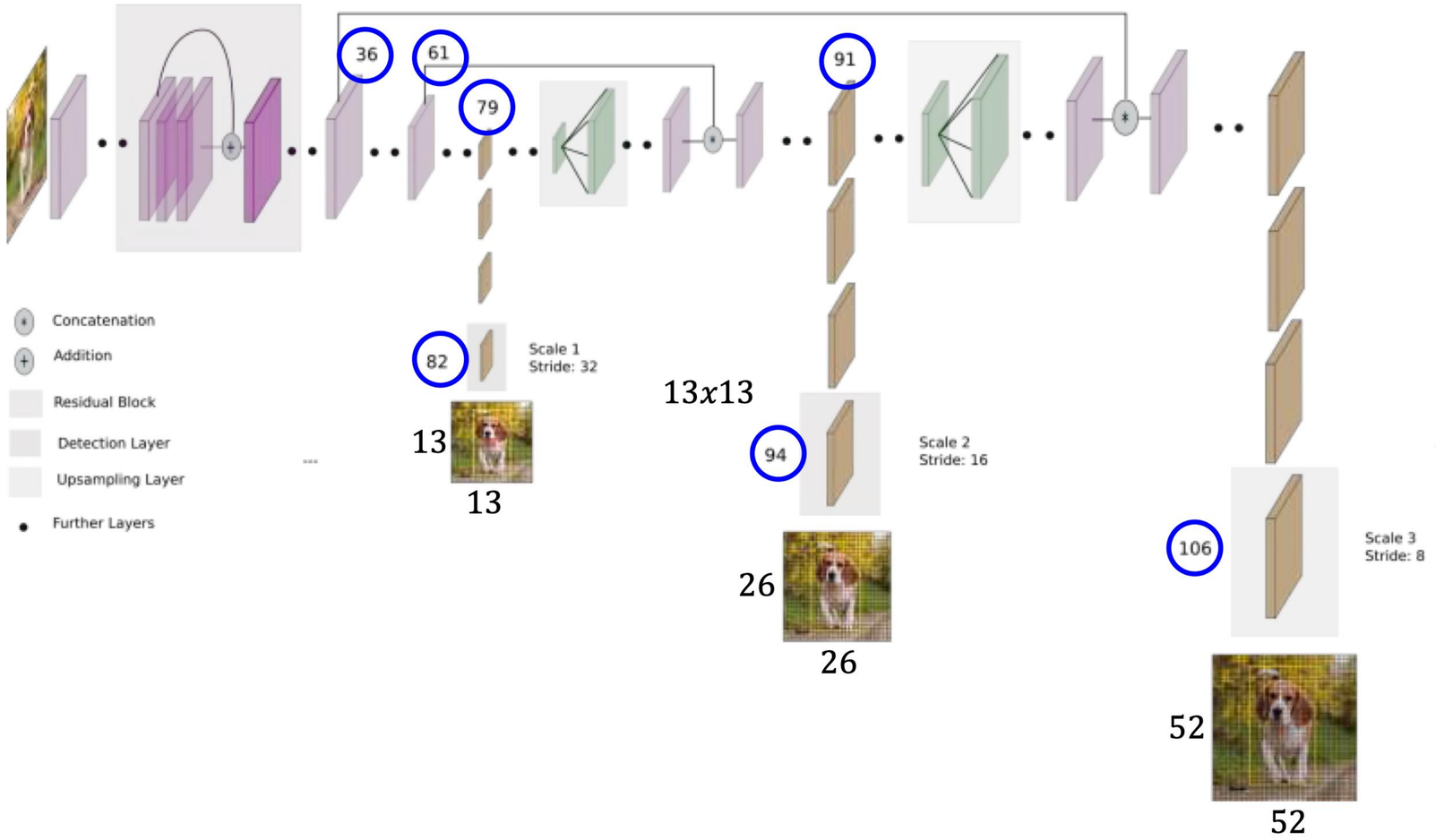
- For each grid cell, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, motorcycle, car) use non-max suppression to generate final predictions.



Source: https://www.youtube.com/watch?v=9s_FpMpdYW8&t=70s

YOLOv3 network architecture

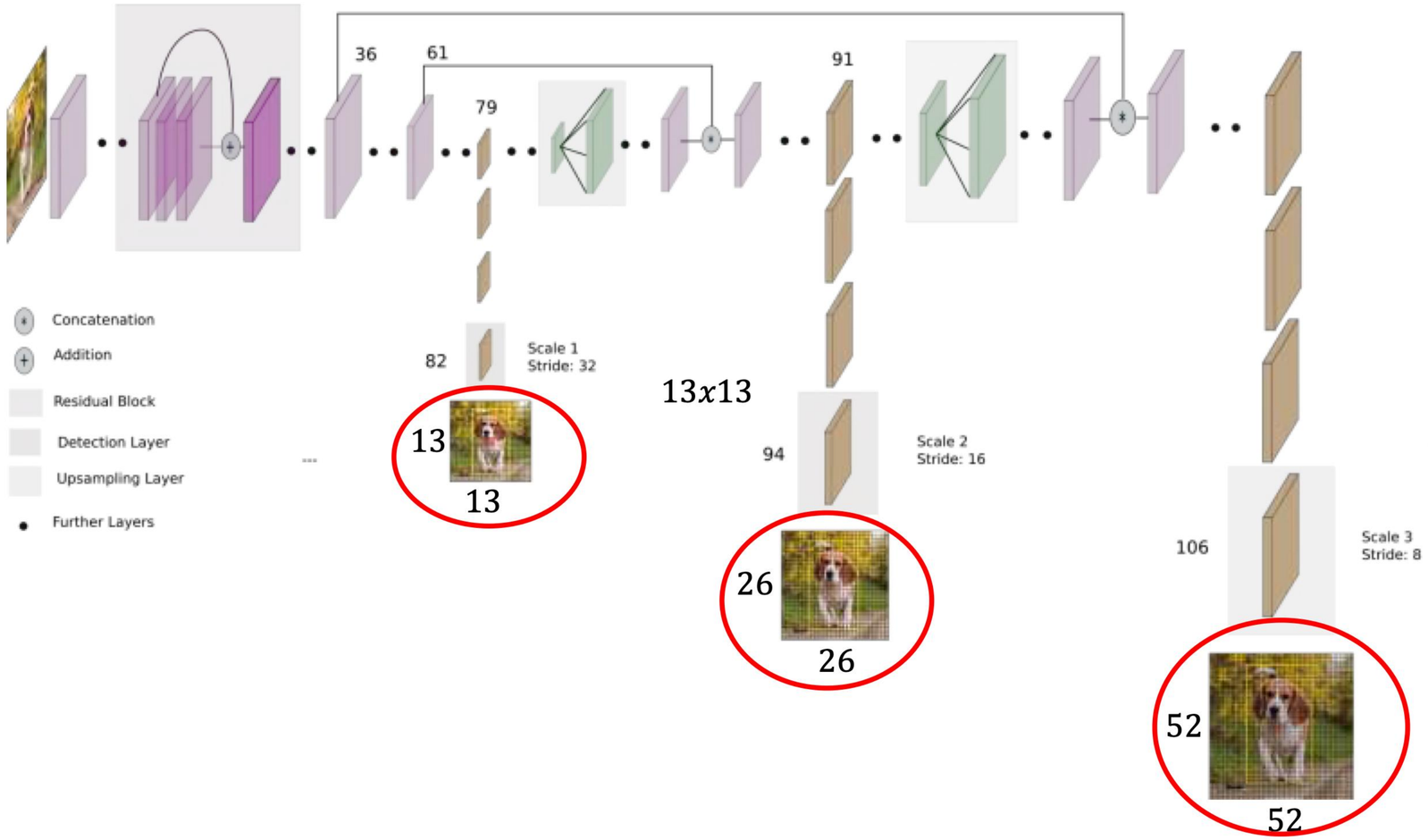
- Deeper than YOLO 9000: from 53 to 106 layers



Source: Ayoosh Kathuria, What's new in YOLO V3?

YOLOv3 network architecture

- Detection at three scales: better at detecting smaller objects
- Input image:
  320, 416 or 608
- Skip connections



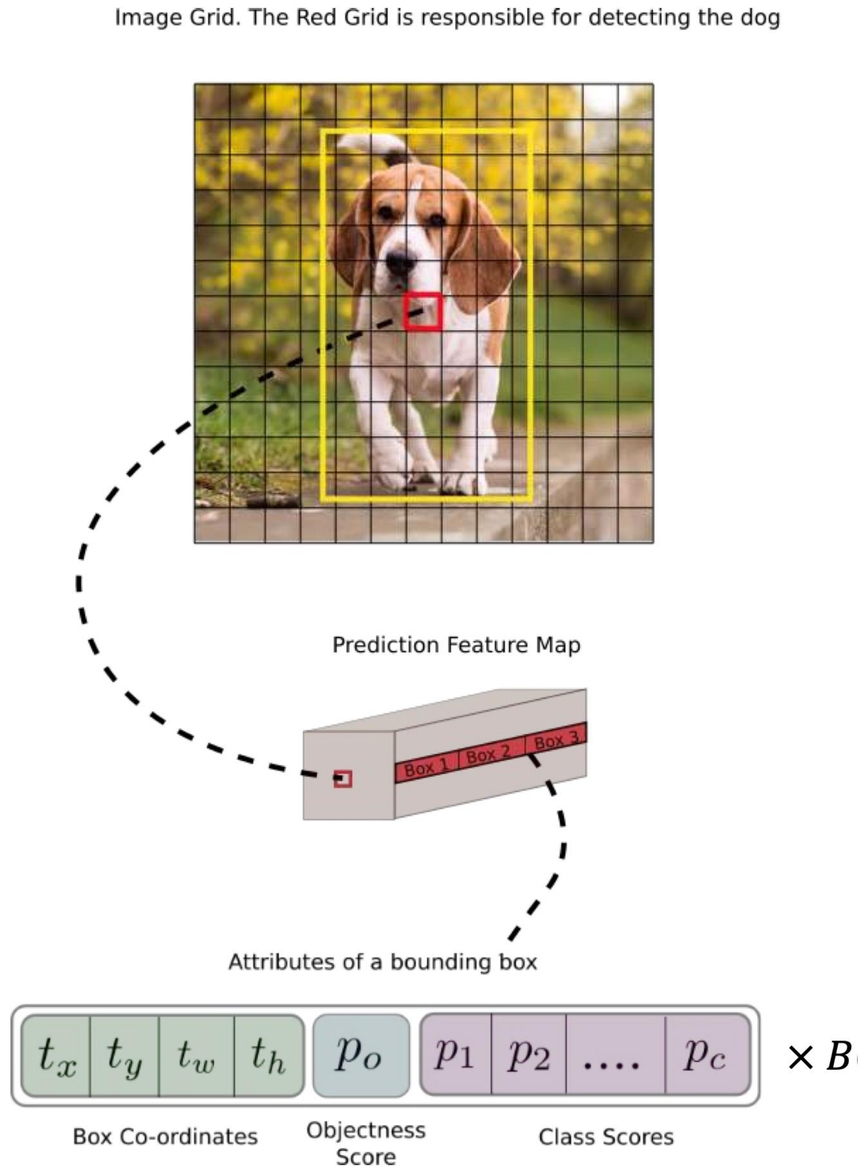
Source: Ayoosh Kathuria, What's new in YOLO V3?

YOLOv3 anchors

- For each cell there are B anchors.
- For each anchor the prediction is a vector of dimension $5 + C$.
- The number of anchor predictions is:

$$(13 \times 13 + 26 \times 26 + 52 \times 52) = \mathbf{10,647}$$

- This is 10 times YOLO9000!
- For COCO, $B = 3,\ C = 80$ anchors/predictions per grid cell.



YOLOv3 loss function

$$\mathcal{L}_{YOLO} =$$

$$y = \begin{bmatrix} t_0 \\ t_x \\ t_y \\ t_h \\ t_w \\ s_1 \\ \vdots \\ s_C \end{bmatrix}$$

for empty anchors

$$\lambda_{nobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{nobj} \left[-log\left(1 - \sigma(t_0)\right) \right]$$

for non-empty anchors

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} \left[(t_x - \hat{t}_x)^2 + \left(t_y - \hat{t}_y\right)^2 + (t_w - \hat{t}_w)^2 + (t_h - \hat{t}_h)^2 \right]$$

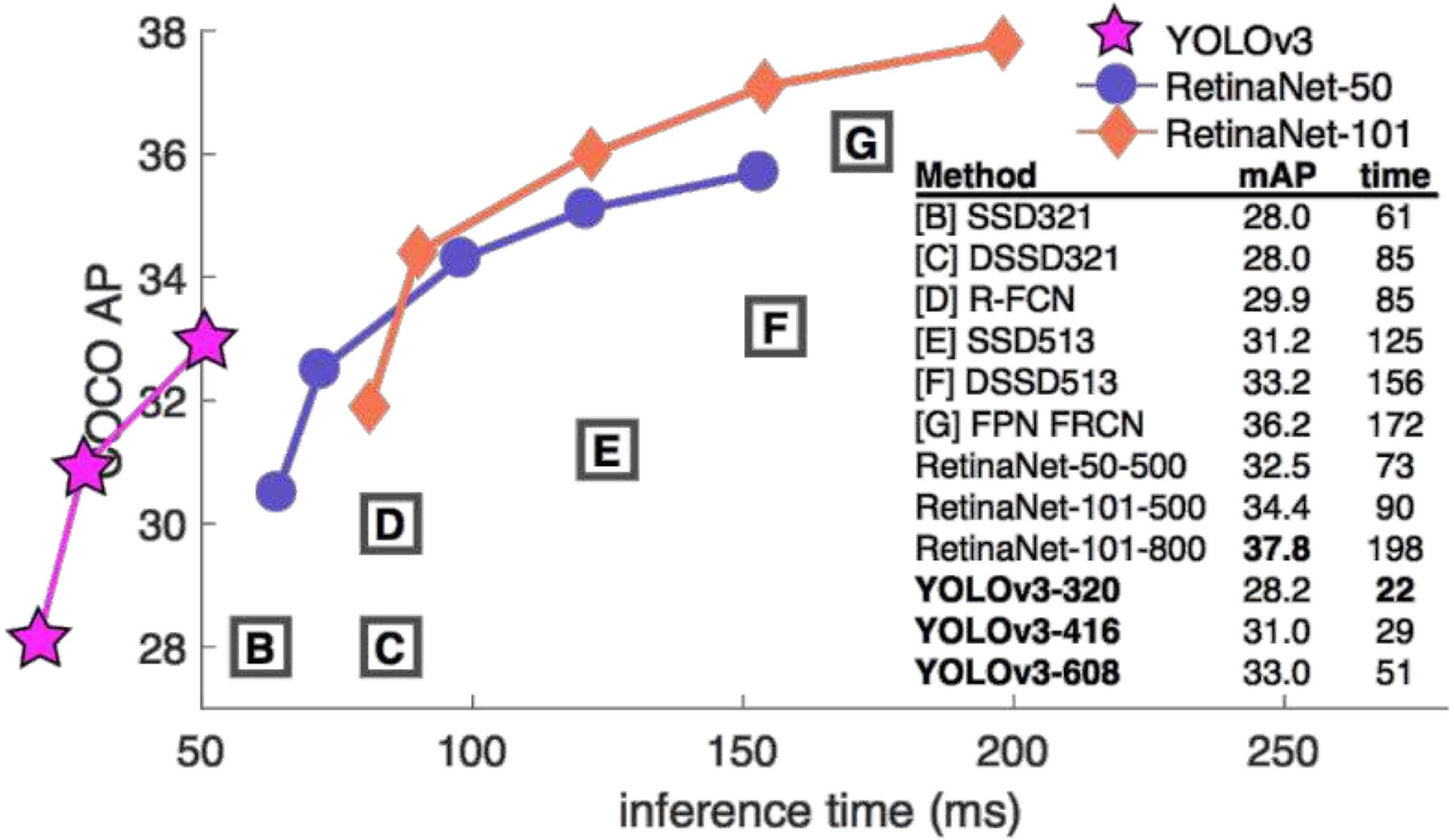
for non-empty anchors

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} \left[-log\left(\sigma(t_0)\right) + \sum_{k=1}^{C} BCE\left(\hat{y}_x, \sigma(s_k)\right) \right]$$

where $1_{i,j}^{obj}$ $1_{i,j}^{nobj}$ are indicator functions; λ_{nobj} and λ_{coord} are weights;

$y/\hat{y}$ are ground truth/prediction; σ is the sigmoid; and BCE is the binary cross entropy.

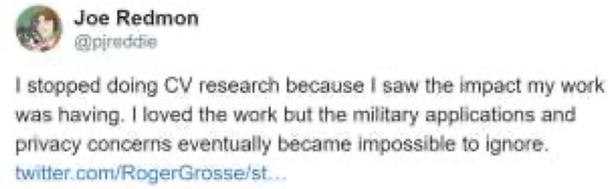
Performance over COCO



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

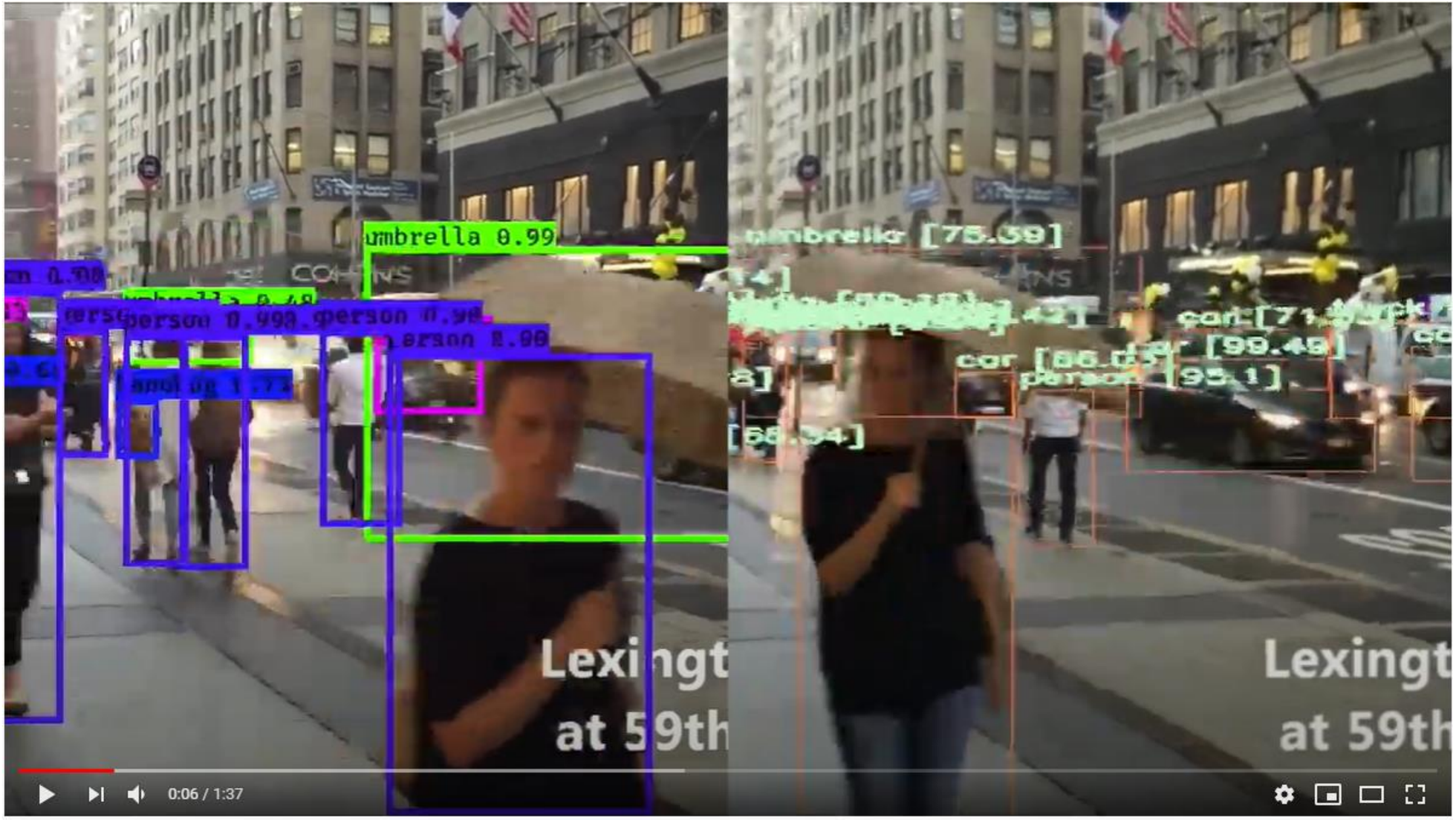
YOLO family

- YOLO: Redmon,J., Divvala, S., Girshick, R., Farhadi, A., (2016), *You Only Look Once: Unified, Real-Time Object Detection*
- YOLO9000: Redmon, J., Farhadi, A., (2016), *YOLO9000: Better, Faster, Stronger*.
  - Batch normalization & high-resolution input images
  - Introduction of anchor boxes
  - Rather than predicting position and size, offsets are predicted for moving and reshaping the pre-defined anchor boxes.
- YOLOv3: Redmon, J., Farhadi, A., (2018), *YOLOv3: An Incremental Improvement*.
  - Detection at 3 scales
  - Choice of anchor boxes
  - More anchor boxes per image (from 845 to 19,647)
  - Change in the loss function



- YOLOv4: Bochkovskiy, A., Wang, C.Y., Liao, H. Y. M., (2020) *YOLOv4: Optimal Speed and Accuracy of Object Detection*
  - *"We offer a state-of-the-art detector which is faster (FPS) and more accurate …than all available alternative detectors."*

YOLO v3 vs YOLO v4



#要轉載請註明出處好嗎
YOLOv3 VS YOLOv4

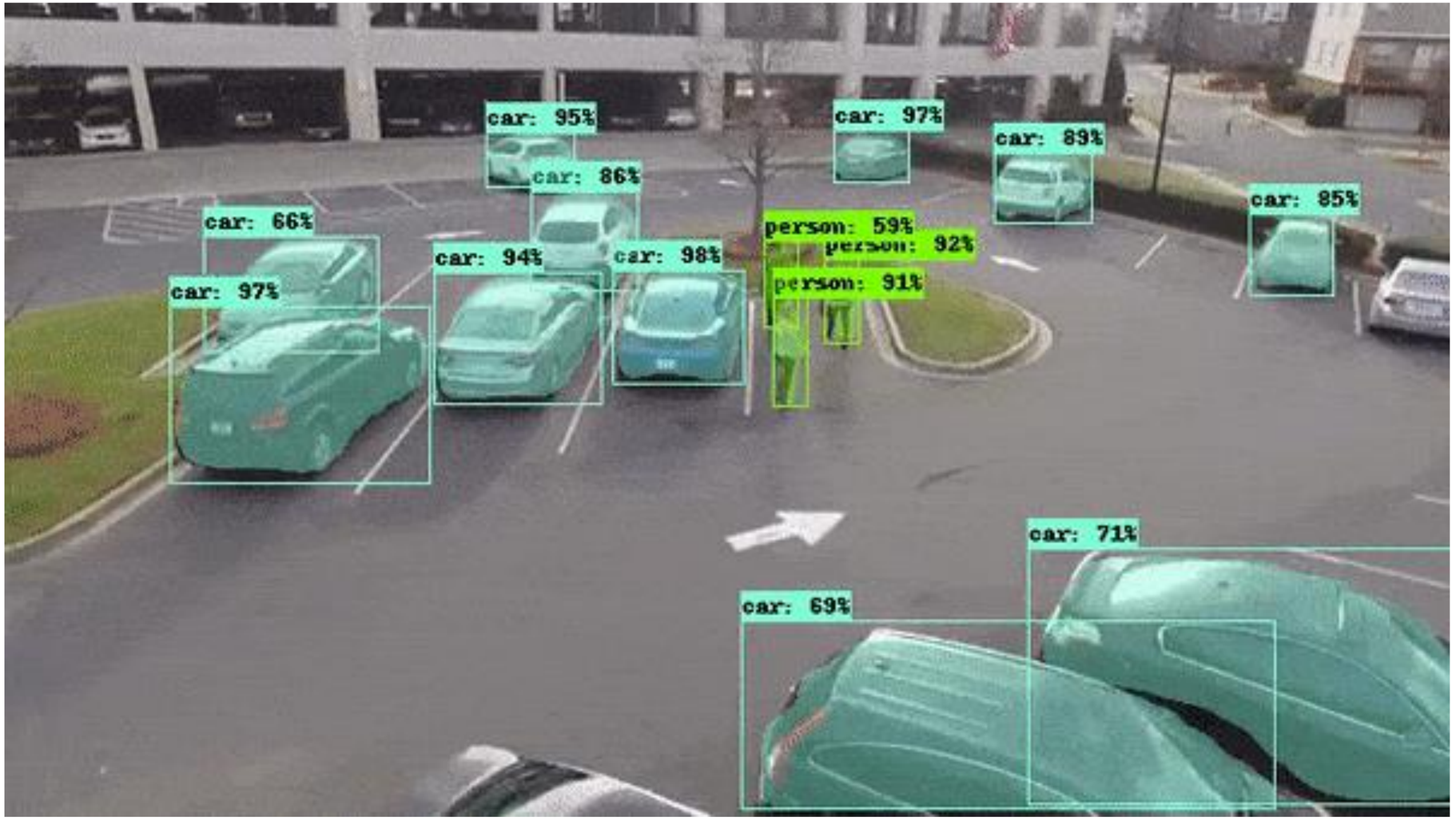
Other Object Detection approaches

- R-CNN Rich feature hierarchies for accurate object detection and semantic segmentation (2014)
- Fast R-CNN (2015)
- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (2016)
- R-FCN: Object Detection via Region-based Fully Convolutional Networks (2016)
- SSD: Single Shot MultiBox Detector (2016)
- FPN: Feature Pyramid Networks for Object Detection (2017)
- RetinaNet: Focal Loss for Dense Object Detection (2018)

Content

- Computer Vision tasks
- Datasets for Object Detection
- Performance Metrics
- YOLO
- Mask R-CNN

Mask-RCNN


car: 95%
car: 97%
car: 89%
car: 86%
car: 85%
car: 66%
person: 59%
person: 92%
car: 94%
car: 98%
person: 91%
car: 97%
car: 71%
car: 69%

Computer Vision tasks

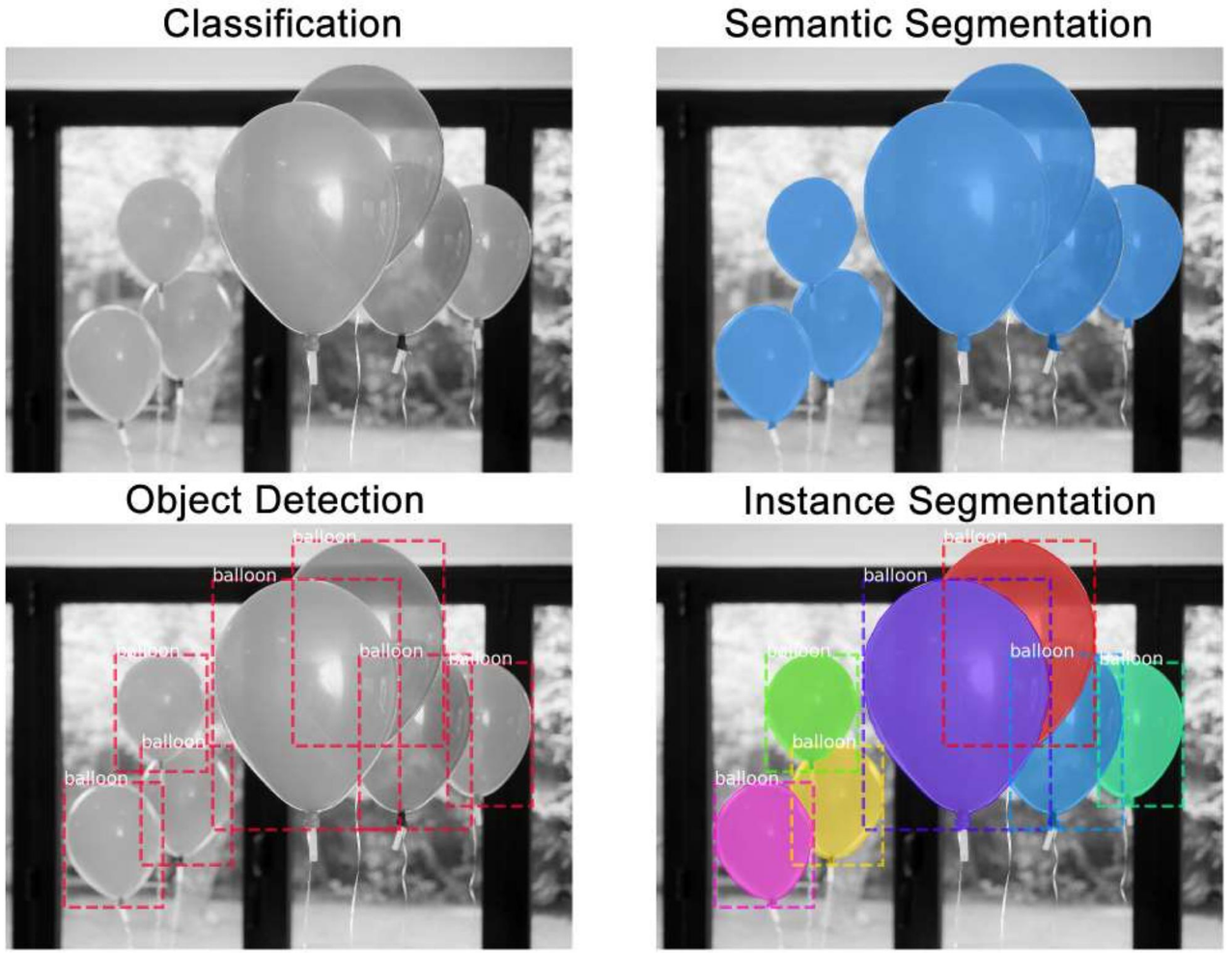


Figure source: [Tiba Razmi, (2019) Mask R-CNN]

Instance Segmentation

Combines object detection and semantic segmentation.

semantic segmentation

Mask R-CNN

Two-stage framework (meta algorithm):

a) generates proposals (areas likely to contain an object);

b) performs semantic segmentation of the proposal areas.

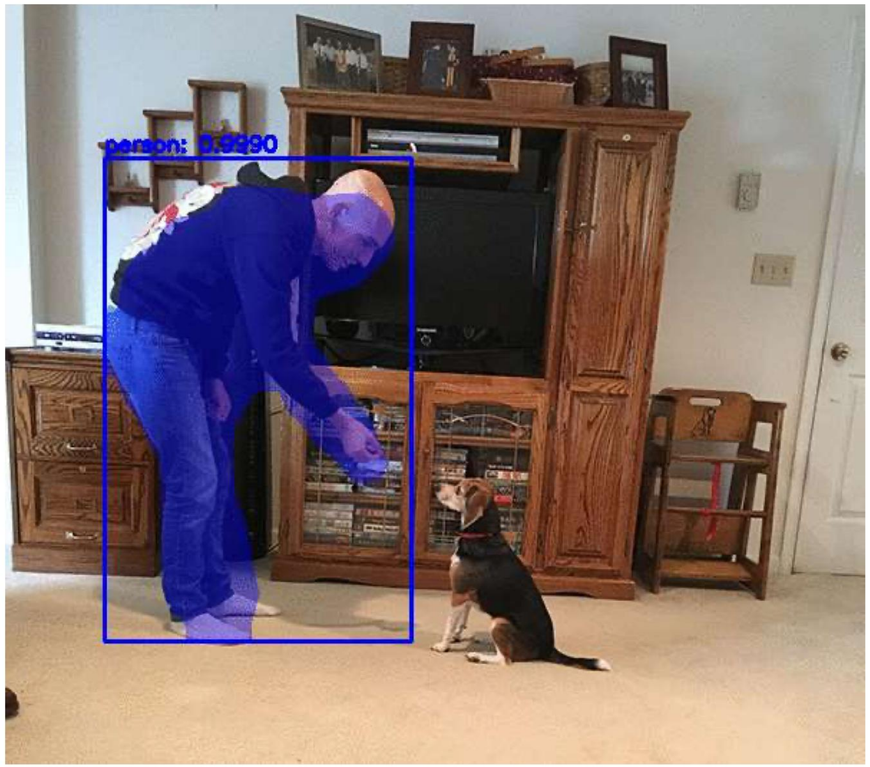


Figure source: https://www.pyimagesearch.com/2018/11/19/mask-r-cnn-with-opencv/

Mask R-CNN

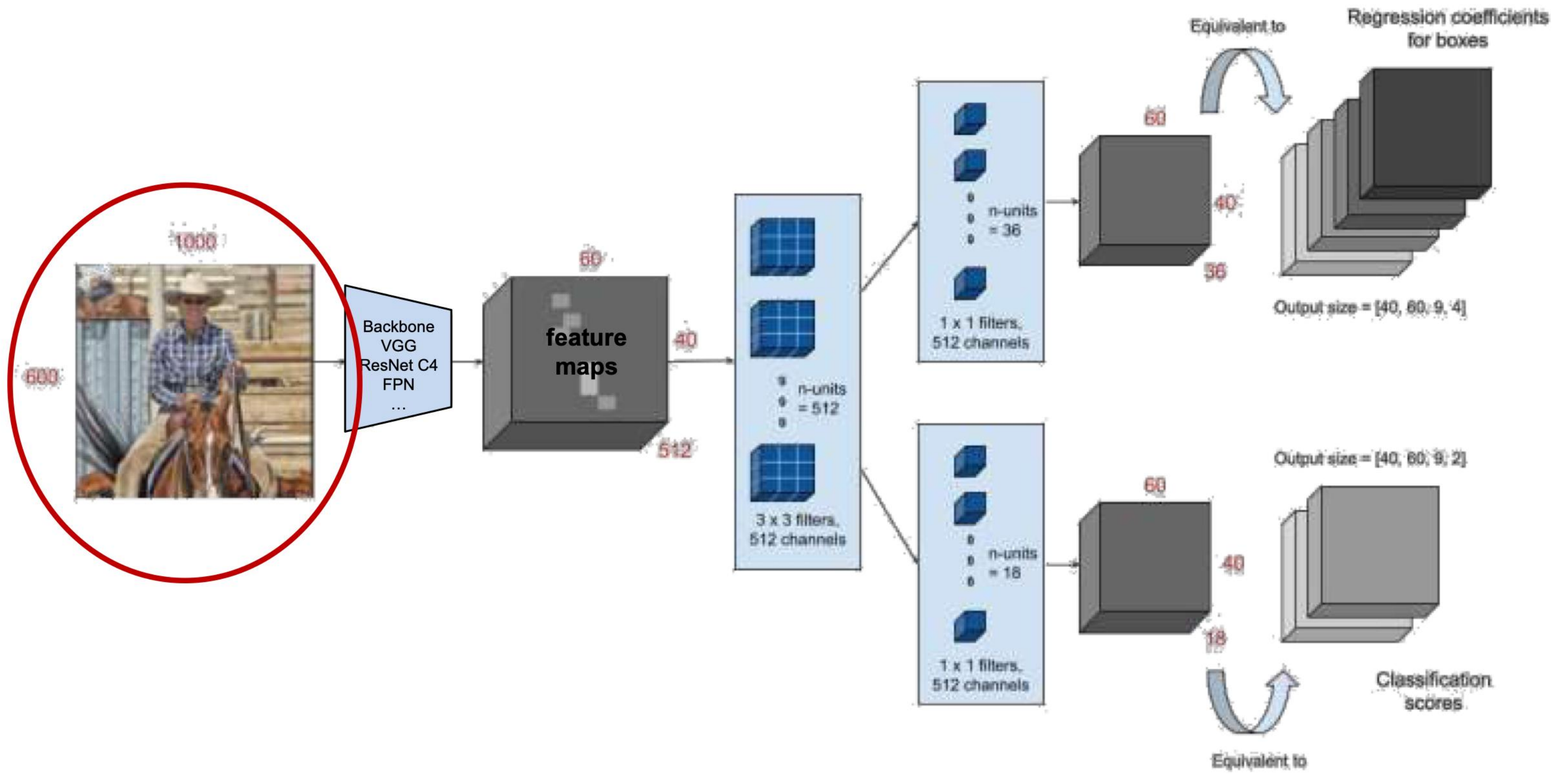
- Proposed by Kaiming He and co-authors in 2018.
- Built upon Faster R-CNN.
- Involves three main steps:

a) **Region Proposal Network (RPN)** generates region proposals;
b) **Region of Interest Alignment (ROIAlign)** reshapes the region proposal to a standard shape/size; and
c) **Mask generation** segments the image inside of object bounding boxes.

Figure source: https://www.pyimagesearch.com/2018/11/19/mask-r-cnn-with-opencv/

Region Proposal Network (RPN)

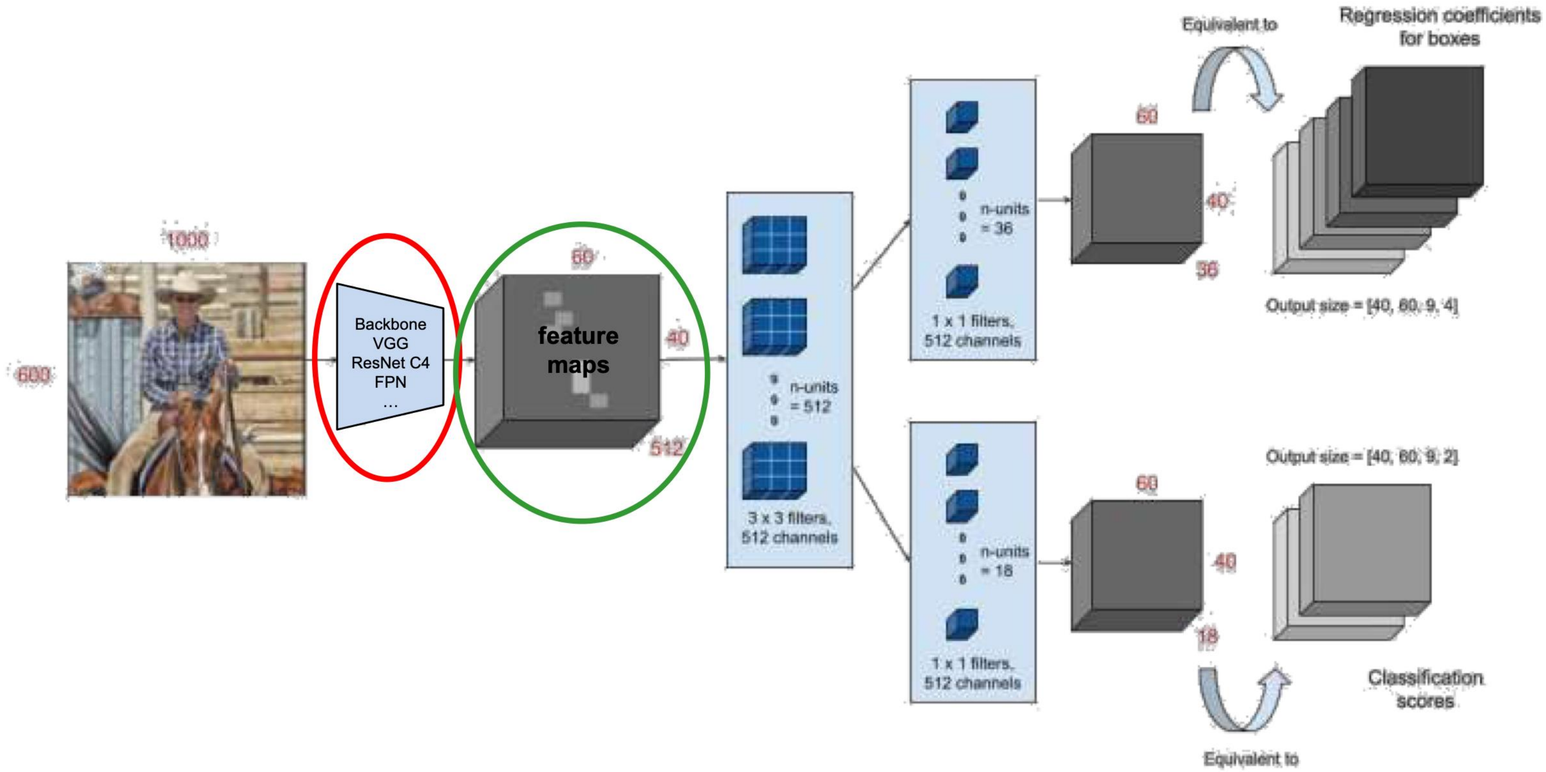
Input image is resized to fit the backbone:



Source: [Shipla Ananth, (2019) Faster R-CNN for object detection]

Region Proposal Network (RPN)

Backbone network produces feature maps:



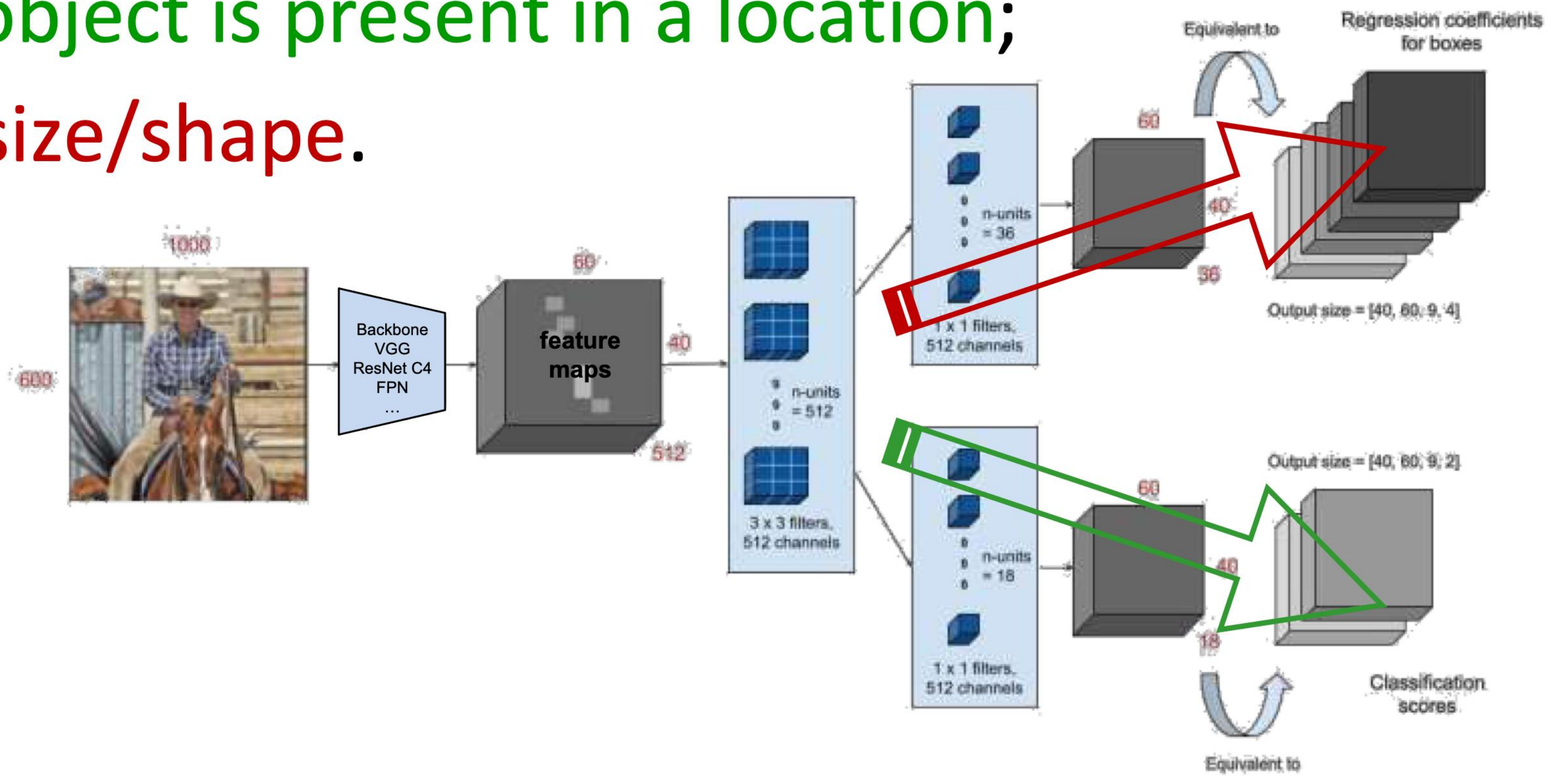
Source: [Shipla Ananth, (2019) Faster R-CNN for object detection]

Region Proposal Network (RPN)

The network has to learn:

a) whether an object is present in a location;

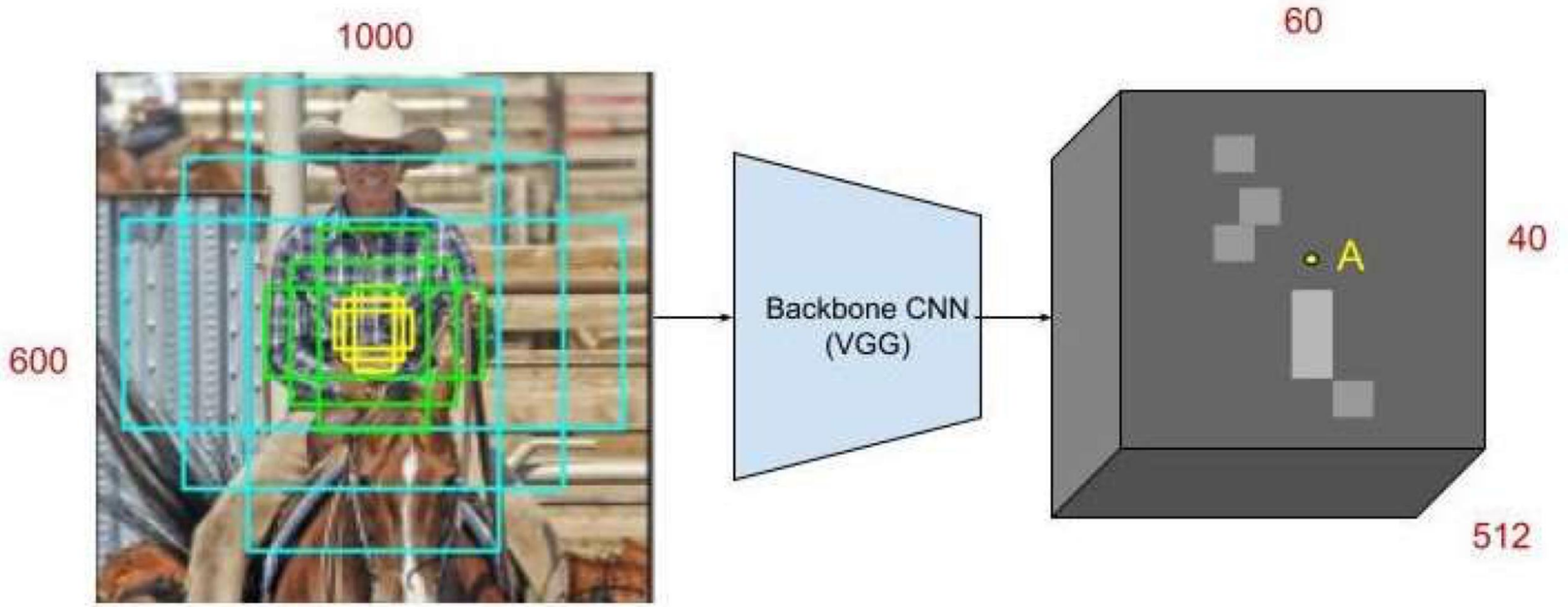
b) estimate its size/shape.



Source: [Shipla Ananth, (2019) Faster R-CNN for object detection]

Region Proposal Network (RPN)

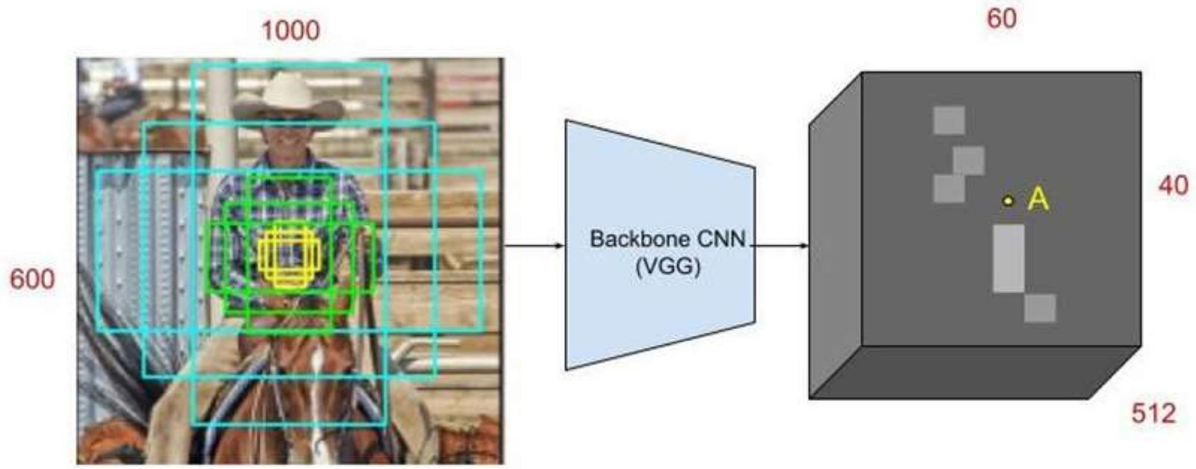
- Done by placing a set of *k* anchors on the input image for each location on the output feature map.
- Example of 9 anchors per image location:



Source: Shipla Ananth, (2019) Faster R-CNN for object detection

Region Proposal Network (RPN)

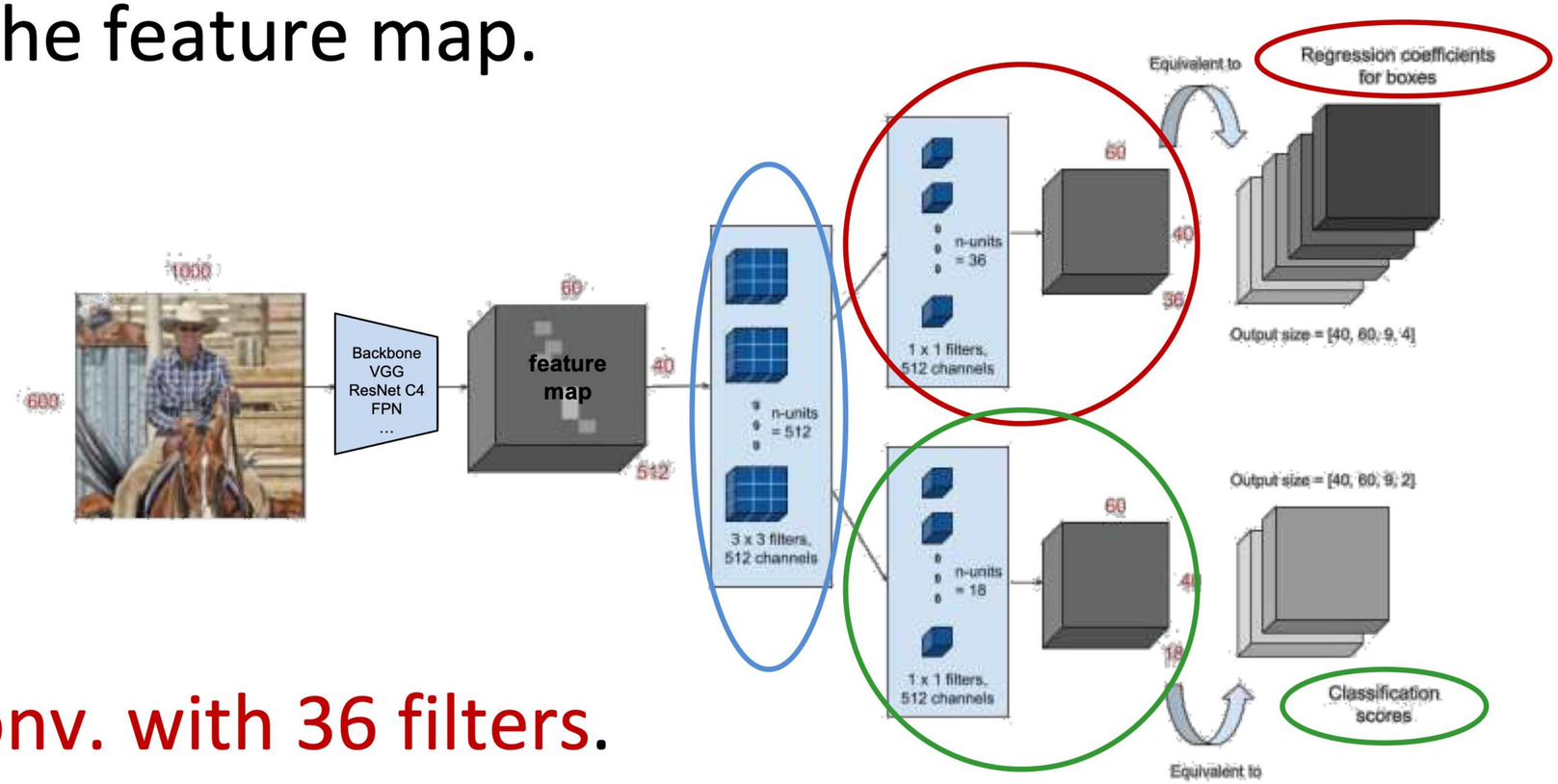
- As the network moves through each pixel in the feature map, it has to check whether these *k* anchors actually contain objects.
- Then it refines these anchors' corresponding bounding boxes to finally deliver them as **object proposals**.



Source: Shipla Ananth, (2019) Faster R-CNN for object detection

Region Proposal Network (RPN)

- First a 3x3 convolution with 512 filters is applied to the feature map.



- Then a 1x1 conv. with 36 filters.
- And a 1x1 conv. with 18 filters.

Source: Shipla Ananth, (2019) Faster R-CNN for object detection

Non-maximum suppression

The regions delivered by RPN are validate as follows:

- Discard all boxes with objectness $p_i \leq threshold$.
- Pick the box with the largest p_i and output it as a prediction.
- Discard any remaining box with $IoU \geq 0.5$ with the box output of the previous step.
- Pick the remaining box with the largest p_i and repeat the previous step, until only one box is left.
- The algorithm must run separately for each class.

Mask R-CNN

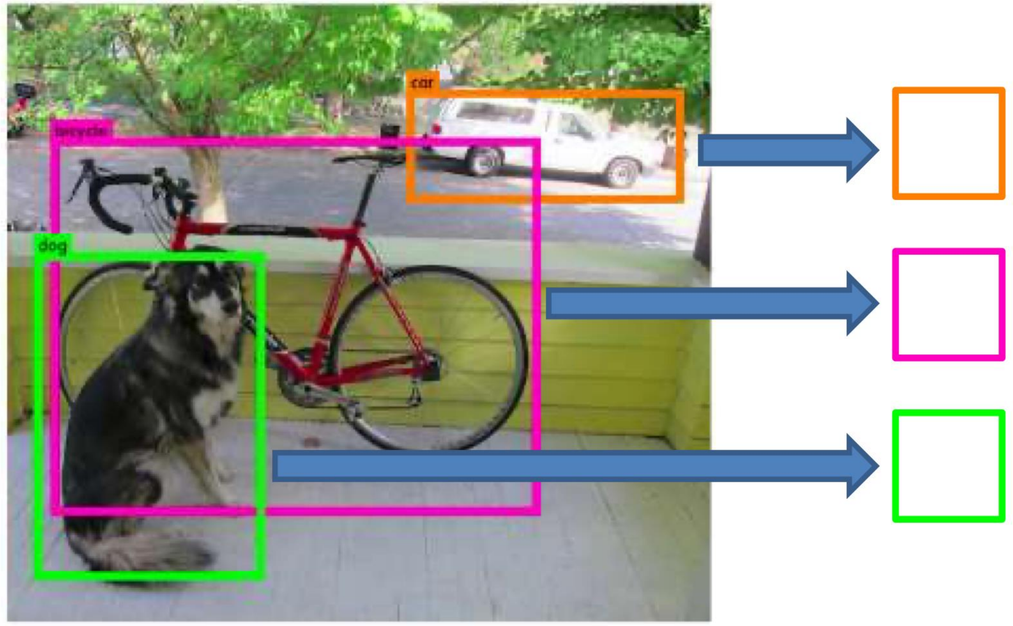
- Proposed by Kaiming He and co-authors in 2018.
- Built upon Faster R-CNN.
- Involves three main steps:

a) **Region Proposal Network (RPN)** generates region proposals;
b) **Region of Interest Alignment (ROIAlign)** reshapes the region proposal to a standard shape/size; and
c) **Mask generation** segments the image inside of object bounding boxes.

Figure source: https://www.pyimagesearch.com/2018/11/19/mask-r-cnn-with-opencv/

Region of Interest Alignment (ROIAlign)

- Problem: the Mask generator network has a fixed size input, but the boxes come with various shapes and sizes.
- Before forwarded to the mask generator, the region proposals (anchors) must be resized to a fixed shape/size.



ROI pooling

Before forwarded to the mask generator, the region proposals (anchors) must be resized to a fixed shape/size.

Example: anchor is 4×6 and the Mask Generator input is 2×2.

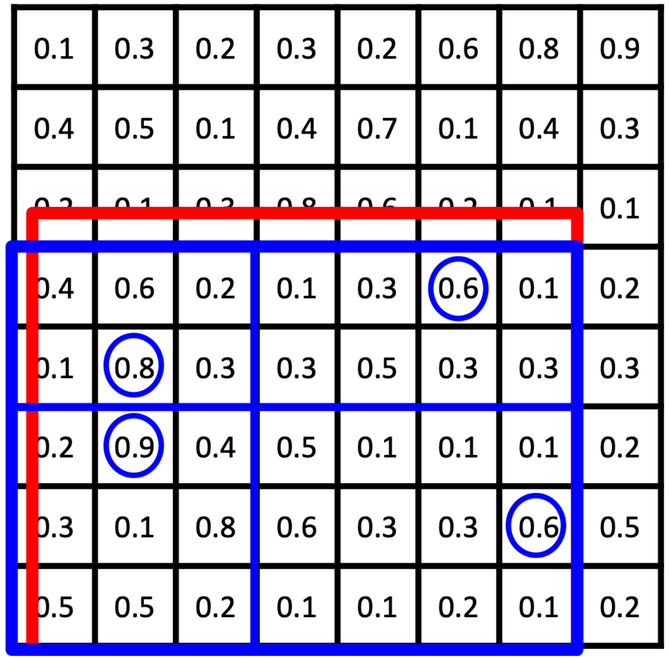
0.1	0.3	0.2	0.3	0.2	0.6	0.8	0.9
0.4	0.5	0.1	0.4	0.7	0.1	0.4	0.3
0.2	0.1	0.3	0.8	0.6	0.2	0.1	0.1
0.4	0.6	0.2	0.1	0.3	0.6	0.1	0.2
0.1	0.8	0.3	0.3	0.5	0.3	0.3	0.3
0.2	0.9	0.4	0.5	0.1	0.1	0.1	0.2
0.3	0.1	0.8	0.6	0.3	0.3	0.6	0.5
0.5	0.5	0.2	0.1	0.1	0.2	0.1	0.2

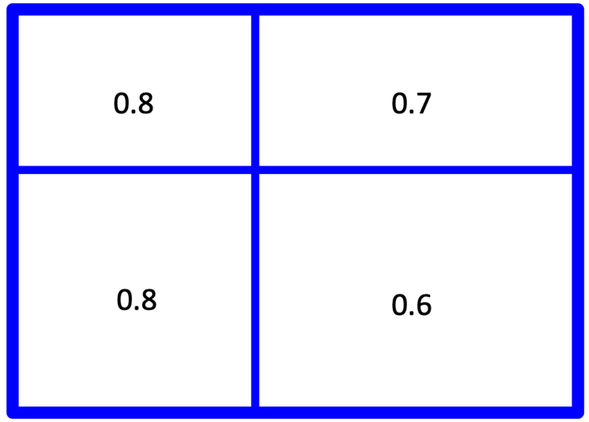
0.8	0.7
0.8	0.6

ROI pooling

Often, the anchor size is not an integer number of the FCN input cells/pixels.

Possible solution: round to the nearest integer.

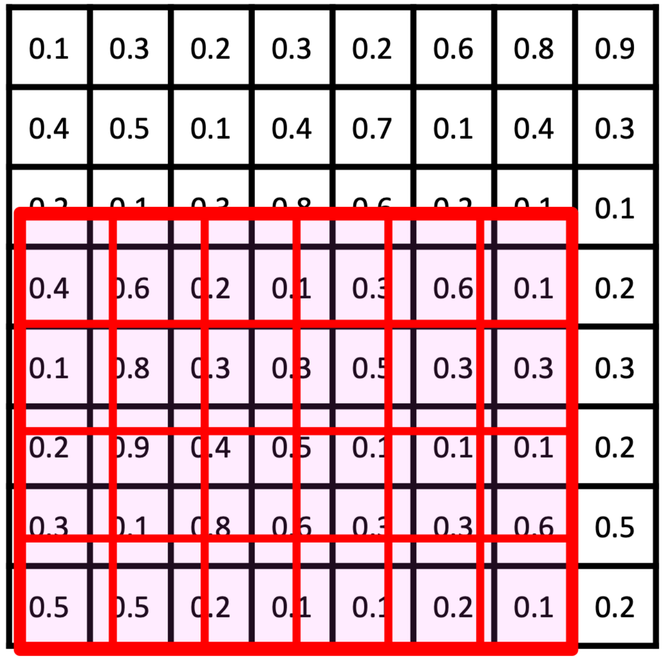


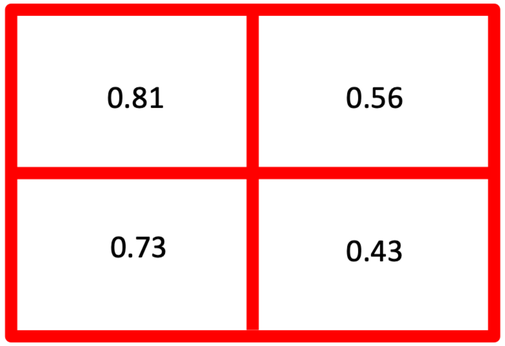


ROI alignment

Instead of rounding to the nearest integer, the values are bilinear interpolated.

ROIAlign “improves mask accuracy by 10% to 50%”.





Mask R-CNN

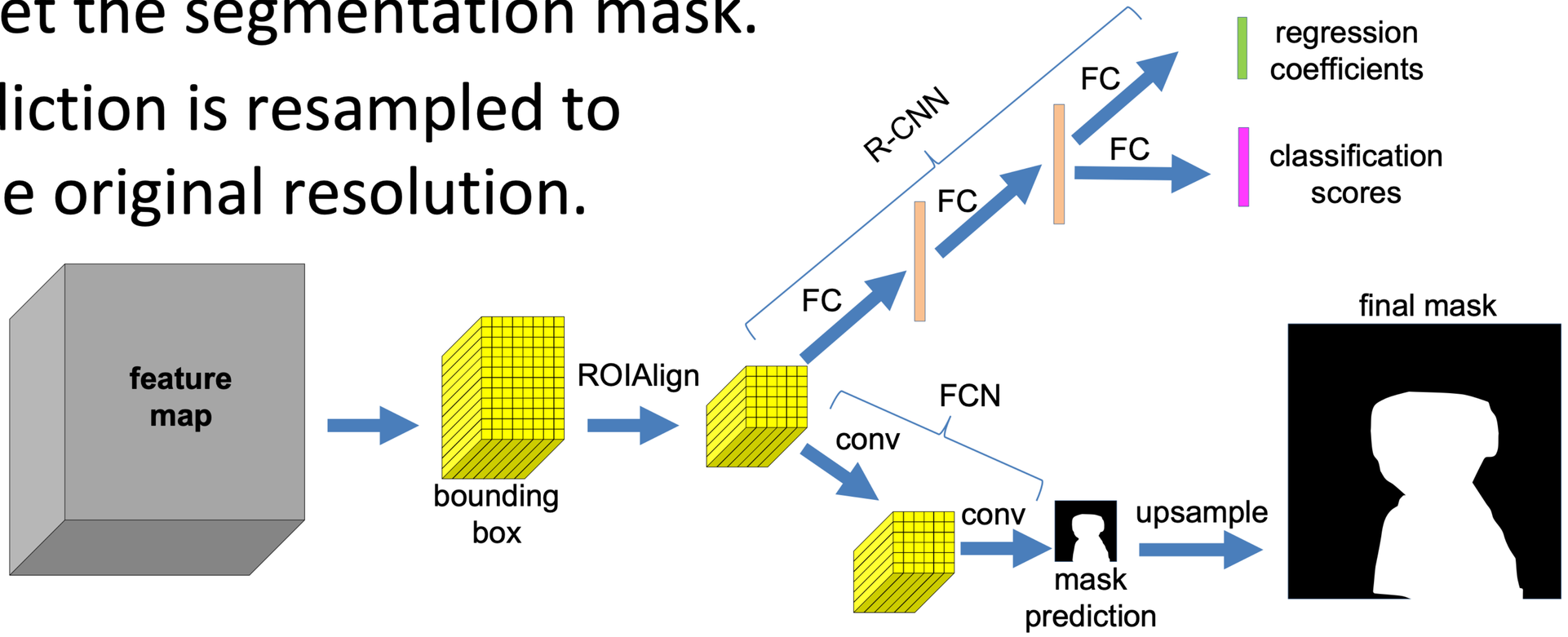
- Proposed by Kaiming He and co-authors in 2018.
- Built upon Faster R-CNN.
- Involves three main steps:

a) **Region Proposal Network (RPN)** generates region proposals;
b) **Region of Interest Alignment (ROIAlign)** reshapes the region proposal to a standard shape/size; and
c) **Mask generation** segments the image inside of object bounding boxes.

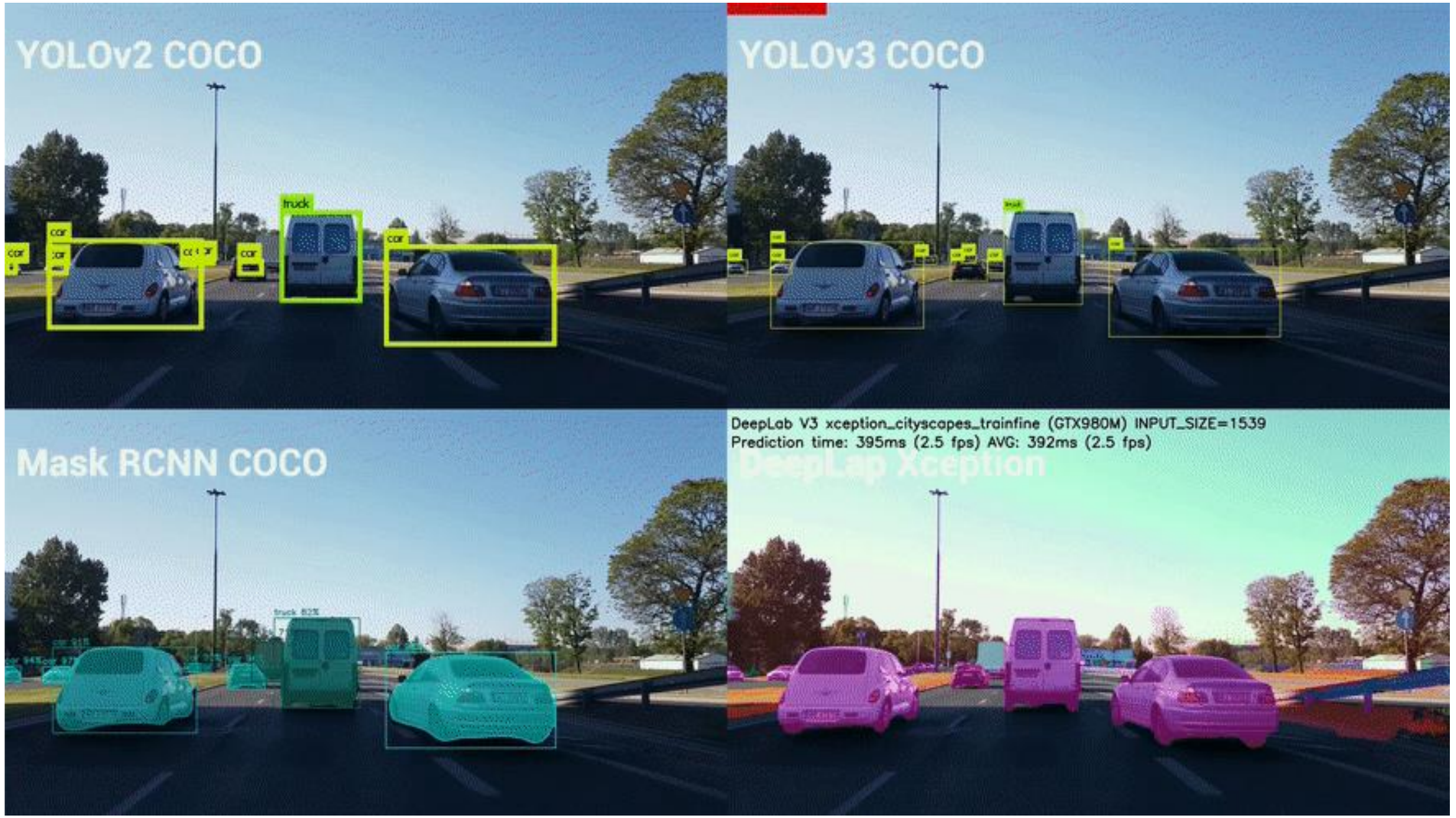
Figure source: https://www.pyimagesearch.com/2018/11/19/mask-r-cnn-with-opencv/

Steps of mask generation

- Fixed size feature map from ROI align goes into R-CNN resulting in bounding box predictions and class scores.
- Fixed size feature map from ROI align goes into a FCN to get the segmentation mask.
- Mask prediction is resampled to recover the original resolution.



OD vs IS vs SS



Click here to watch the demo.

Main Sources

- **Original paper:** He, K., Gkioxari, G., Dollár, P., Girshick, R., (2018), Mask R-CNN.
- **Presentation at ICCV17** by Kaiming He.

Next Lecture

Thursday
Lab Object Detection

See you next class!