Lecture 08 – Image Detection and Segmentation

Prof. André Gustavo Hochuli

gustavo.hochuli@pucpr.br
aghochuli@ppgia.pucpr.br

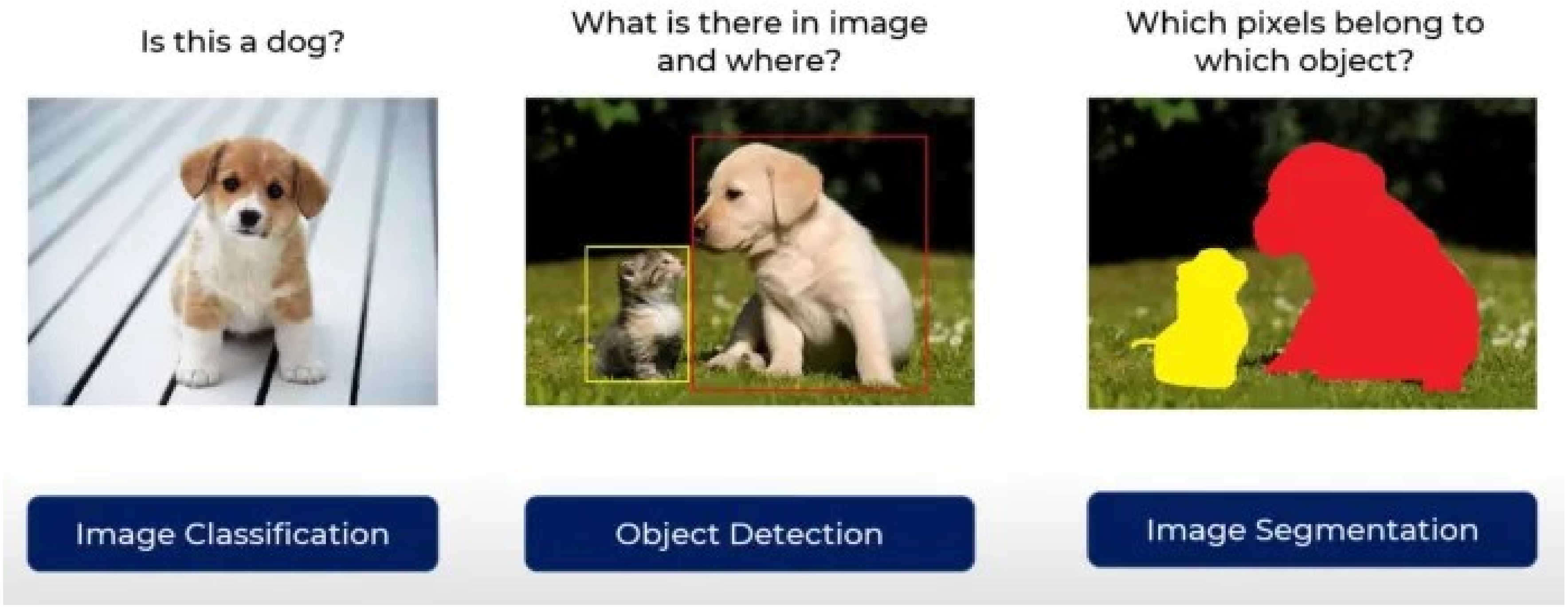
Topics

- Review of Lecture 10 – CNN Applications and Tricks

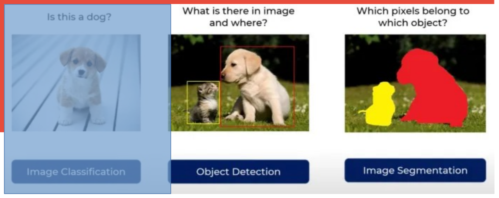
- Classification vs Segmentation
    - Classification
    - Object Detection
    - Segmentation
- Practice

Introduction
Filtering / Morphology
Finding Components
Feat. Extraction
Classification / ML
Deep Learning (Classf)
PBJL-1
AVAL-1
Deep Learning (Detect)
Deep Learning (GANs)
Advanced Topics
PBJL-2
AVAL-2

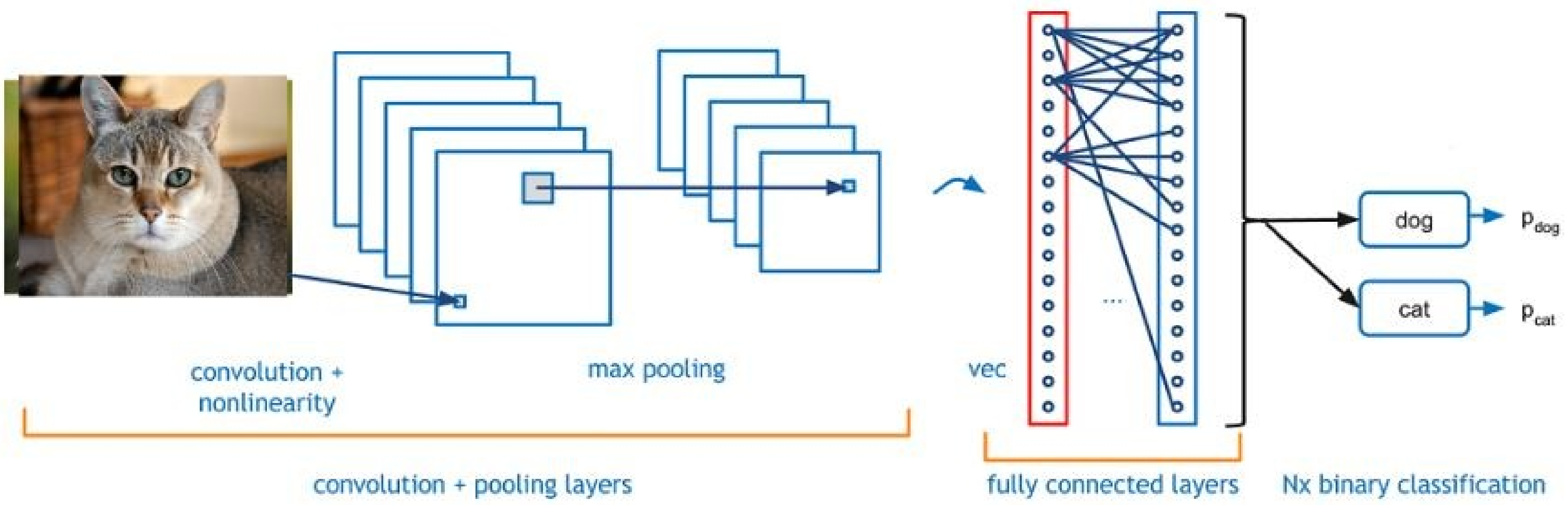
Classification vs Segmentation


Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

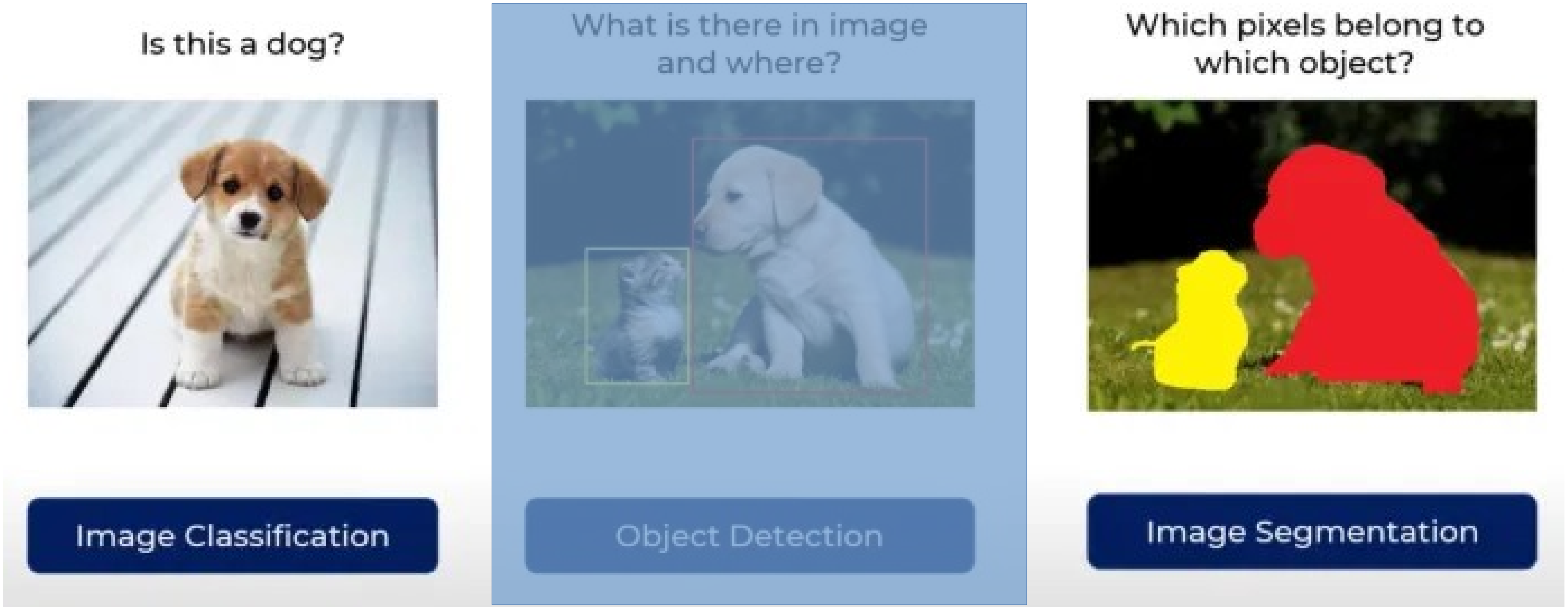
Classification



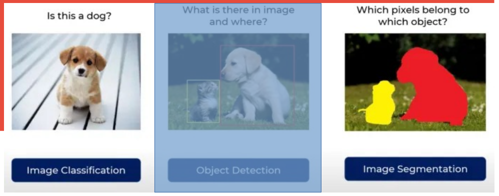
Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

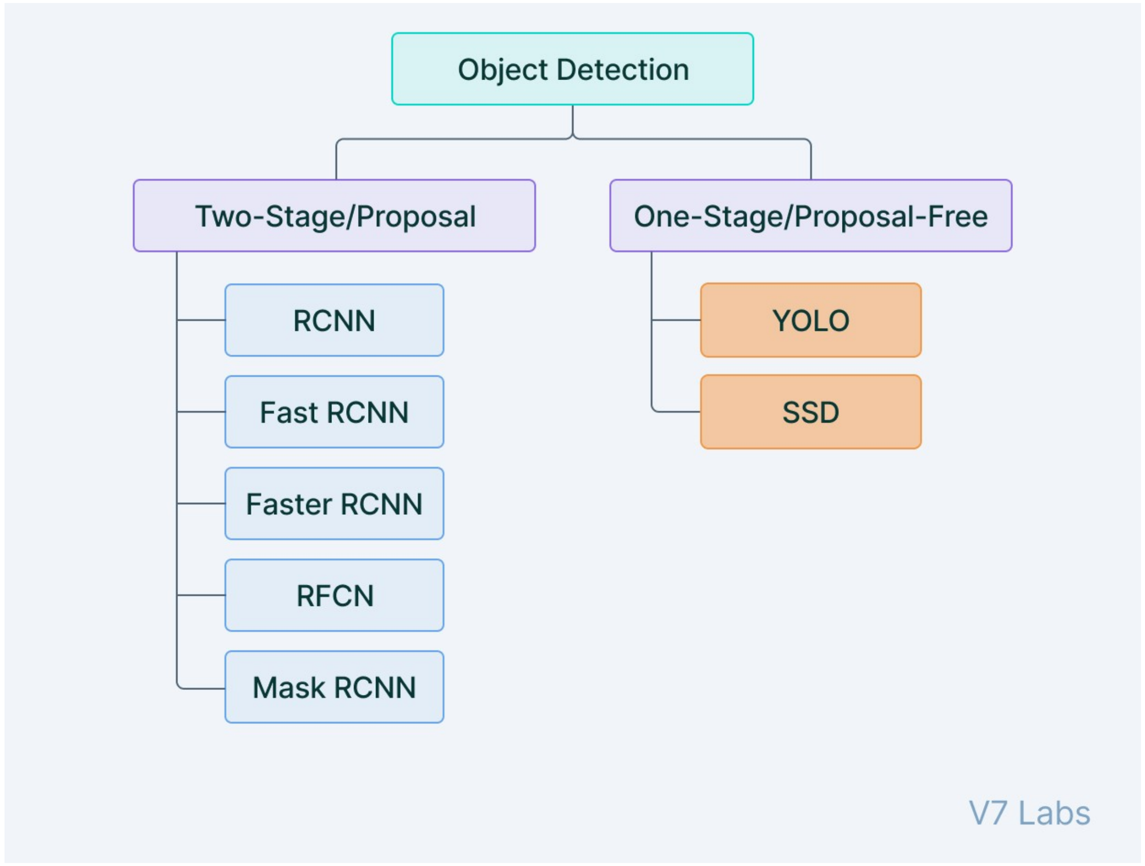


Object Detection

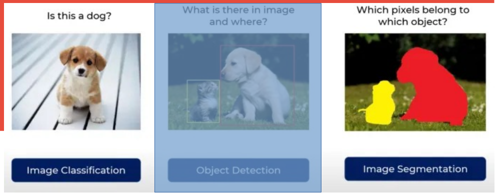

Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

Object Detection

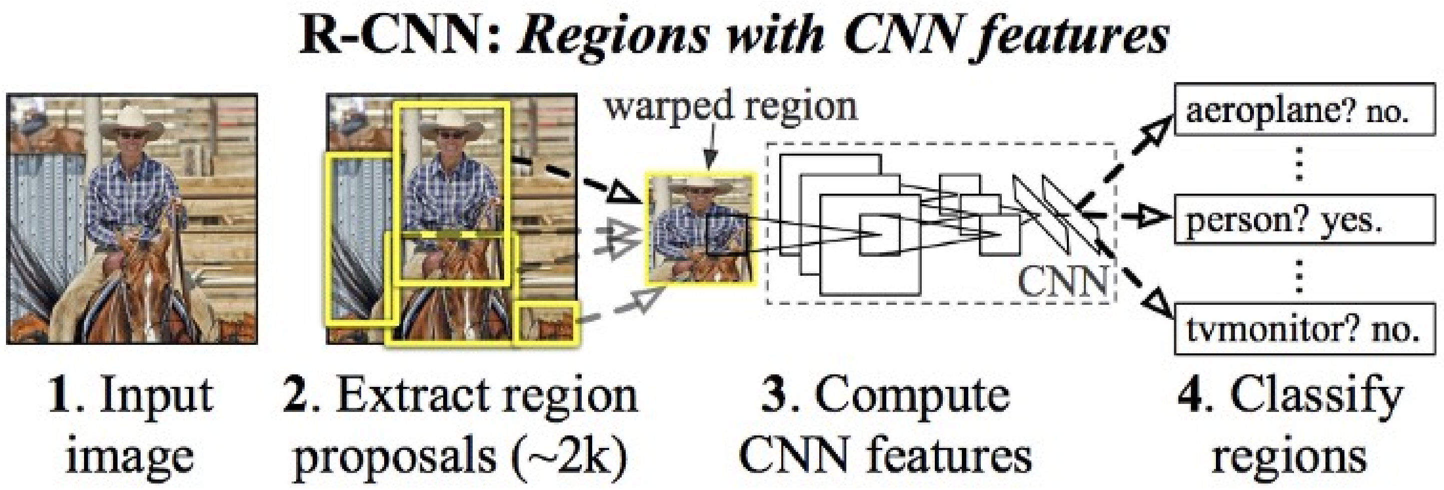

Is this a dog?
Image Classification
What is there in image and where?
Object Detection
Which pixels belong to which object?
Image Segmentation



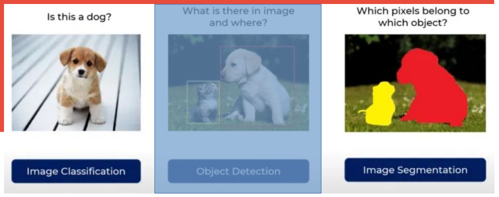
Object Detection - RCNN


Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

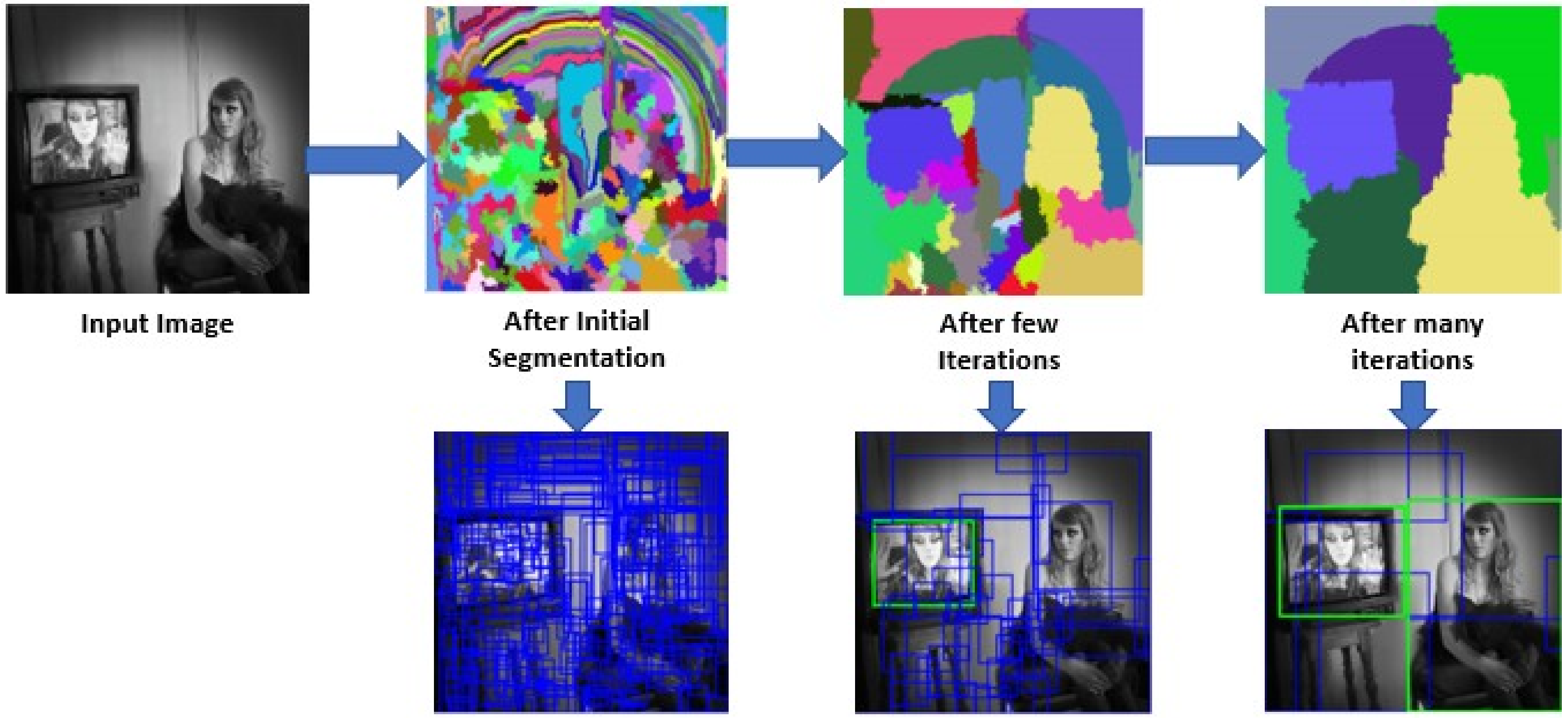
- Region Based Convolutional Neural Network (2014) - Ross Girshick
- Selective Search Algorithm (Region Proposal)
- CNN (Classification)



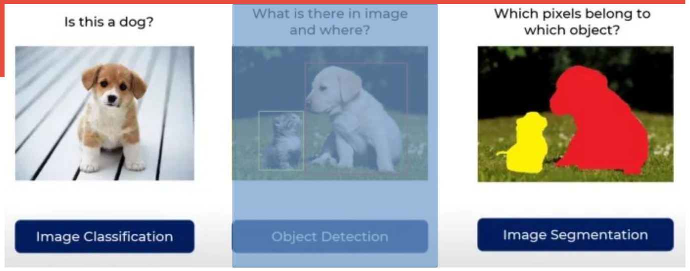
Object Detection - RCNN


Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

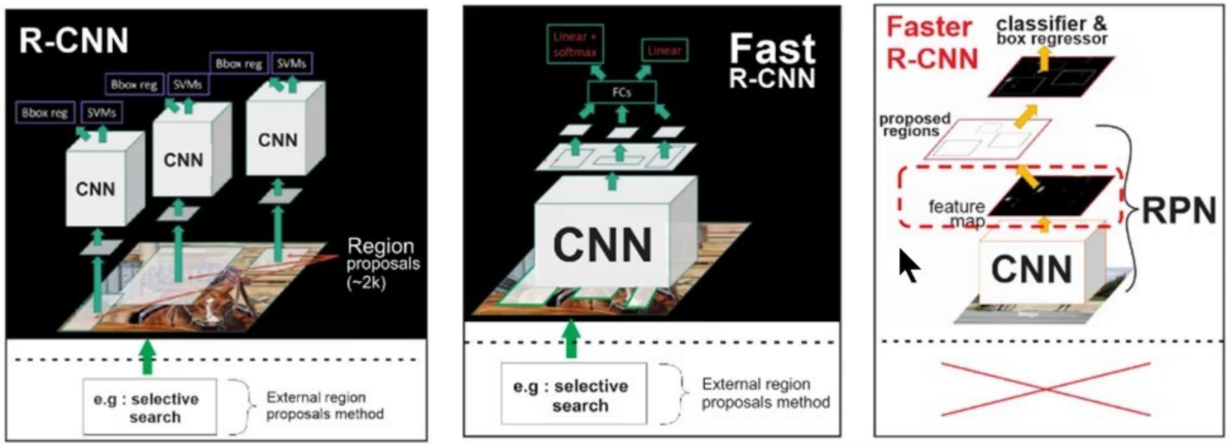
- Selective Search Algorithm (Region Proposal)

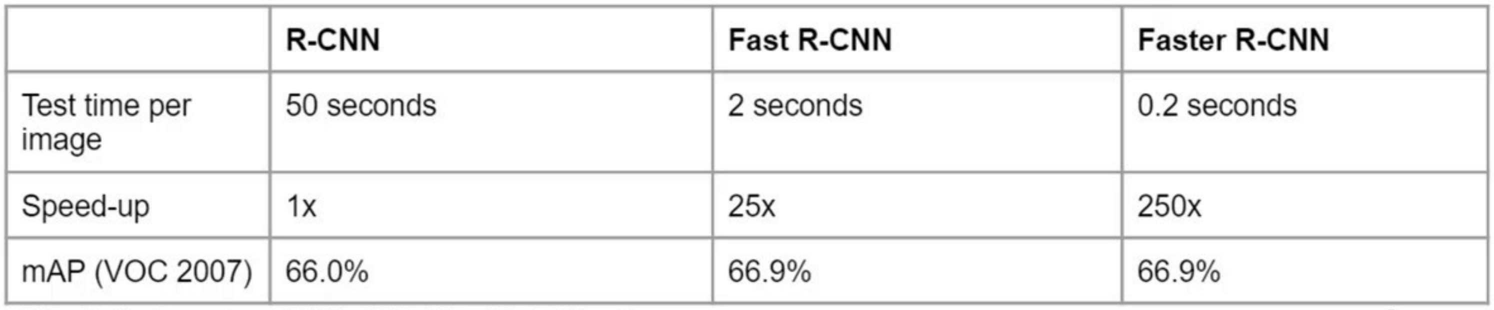


Object Detection - RCNN


Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

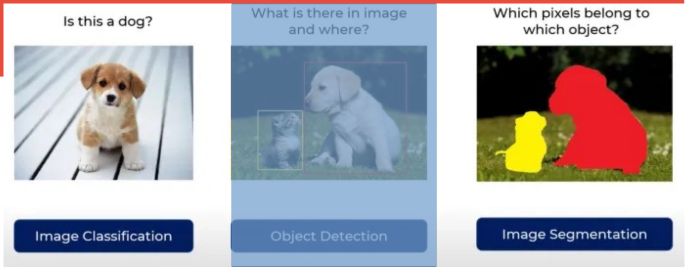
- R-CNN: Selective Search->CNN
- Fast: End-to-end (Sel. Search->ROI Pooling→FC)
- Faster: Region Proposal Network (RPN)



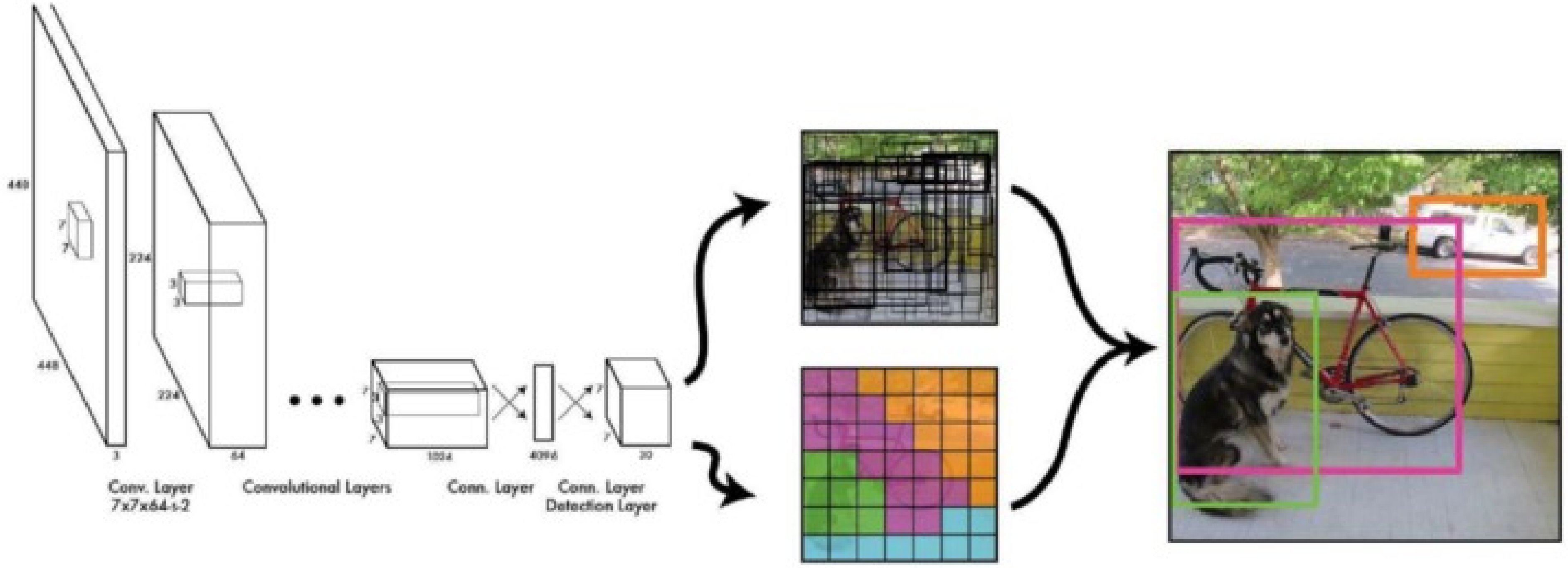


	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

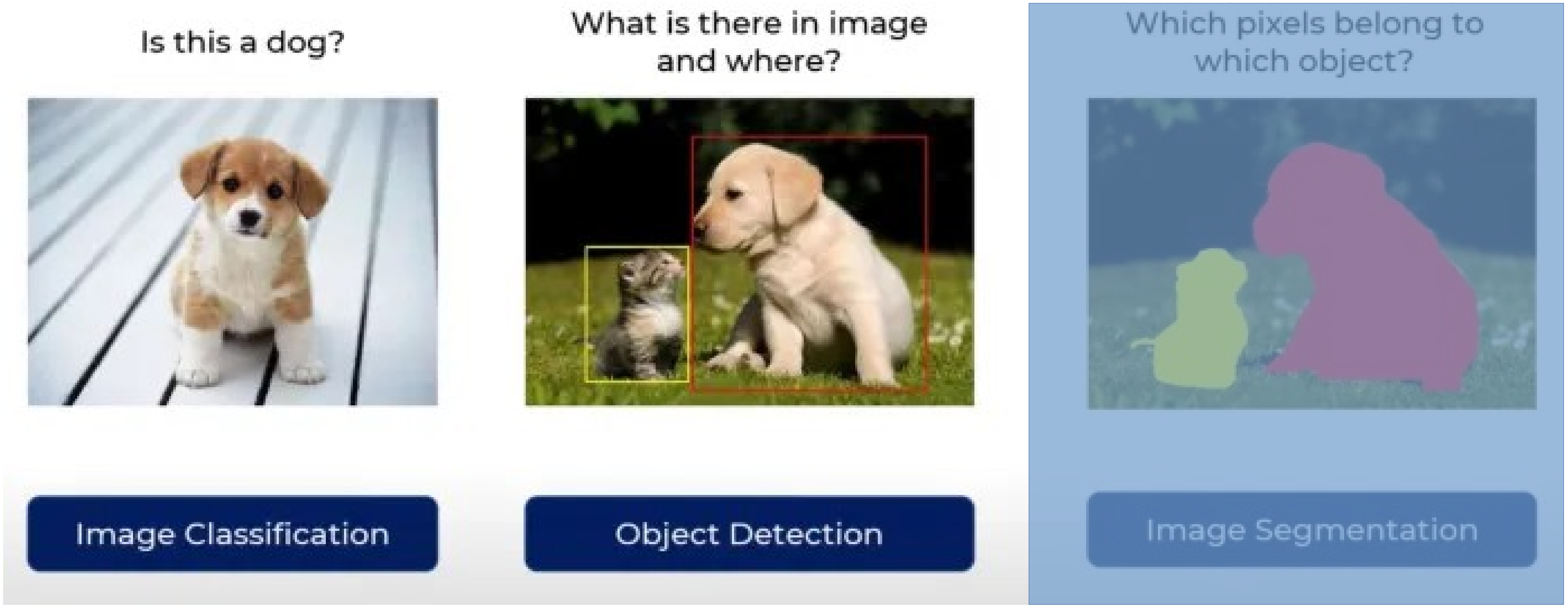
Object Detection - Yolo


Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

- You Look Once (YoLo - 2015 - now)
  - Joseph Redmon / Ross Girshick
- Fast End-to-End Architecture

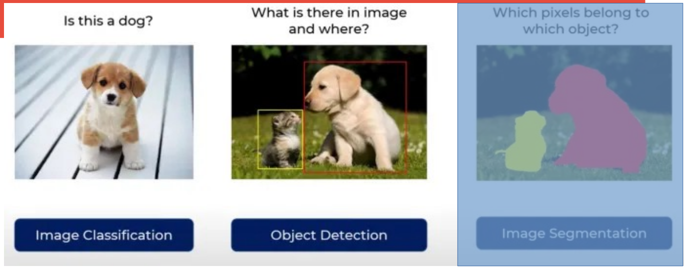


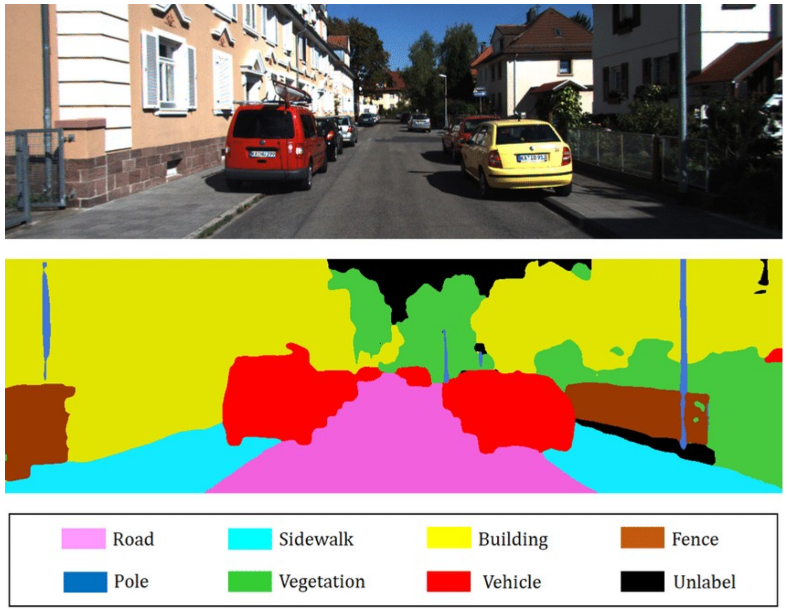
Segmentation

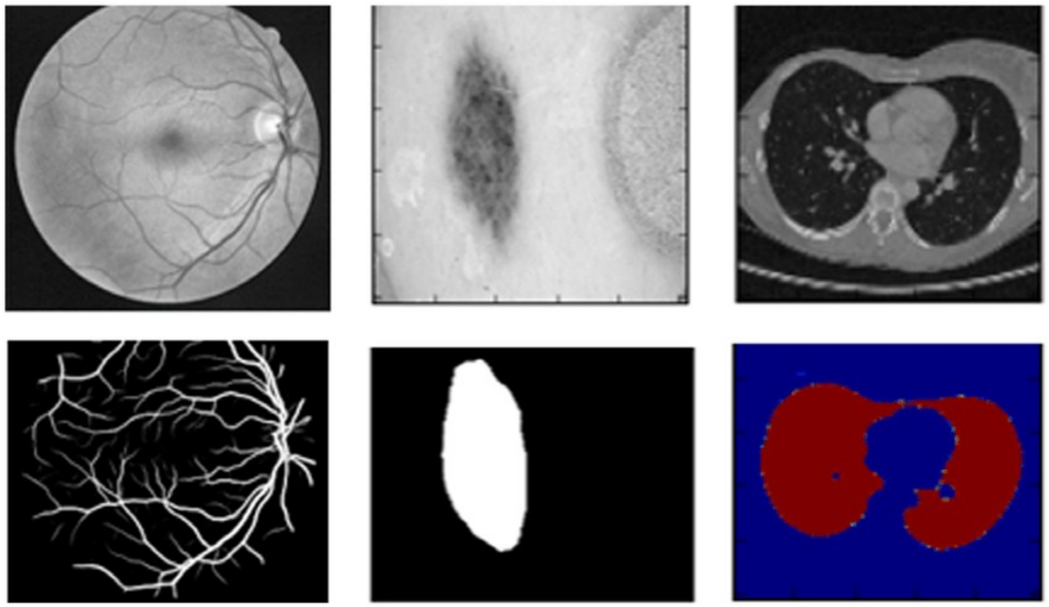

Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

Segmentation

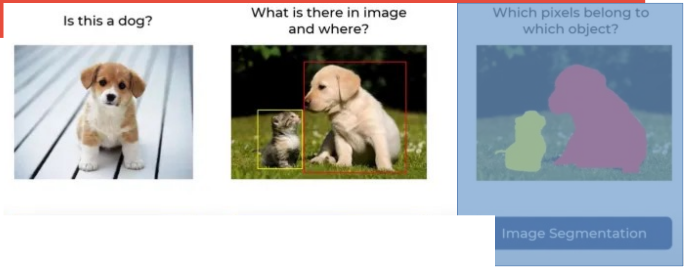
- Classification at pixel level


Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

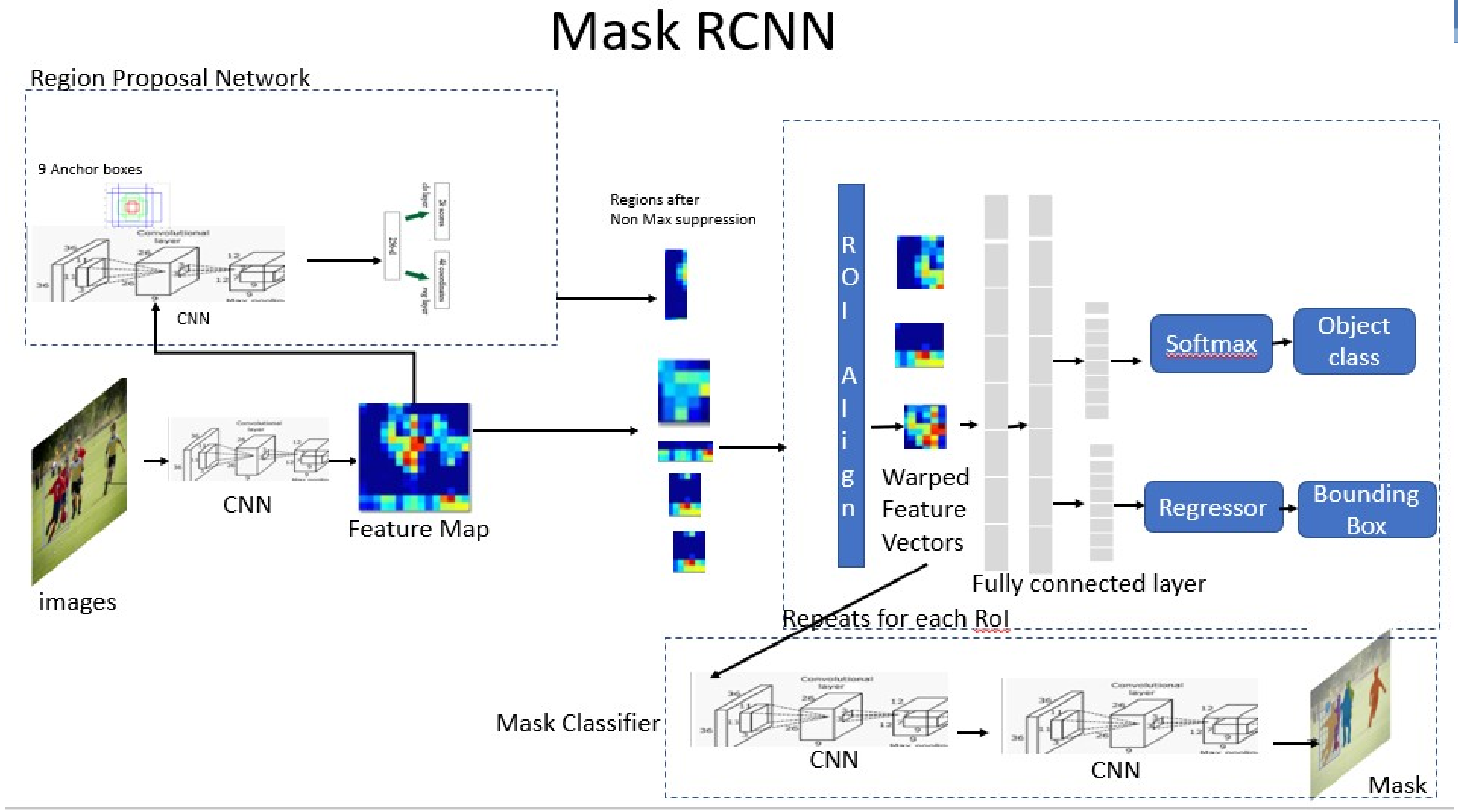




Segmentation – Mask RCNN

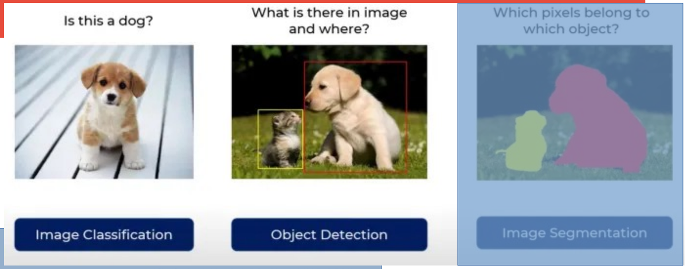

Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Segmentation

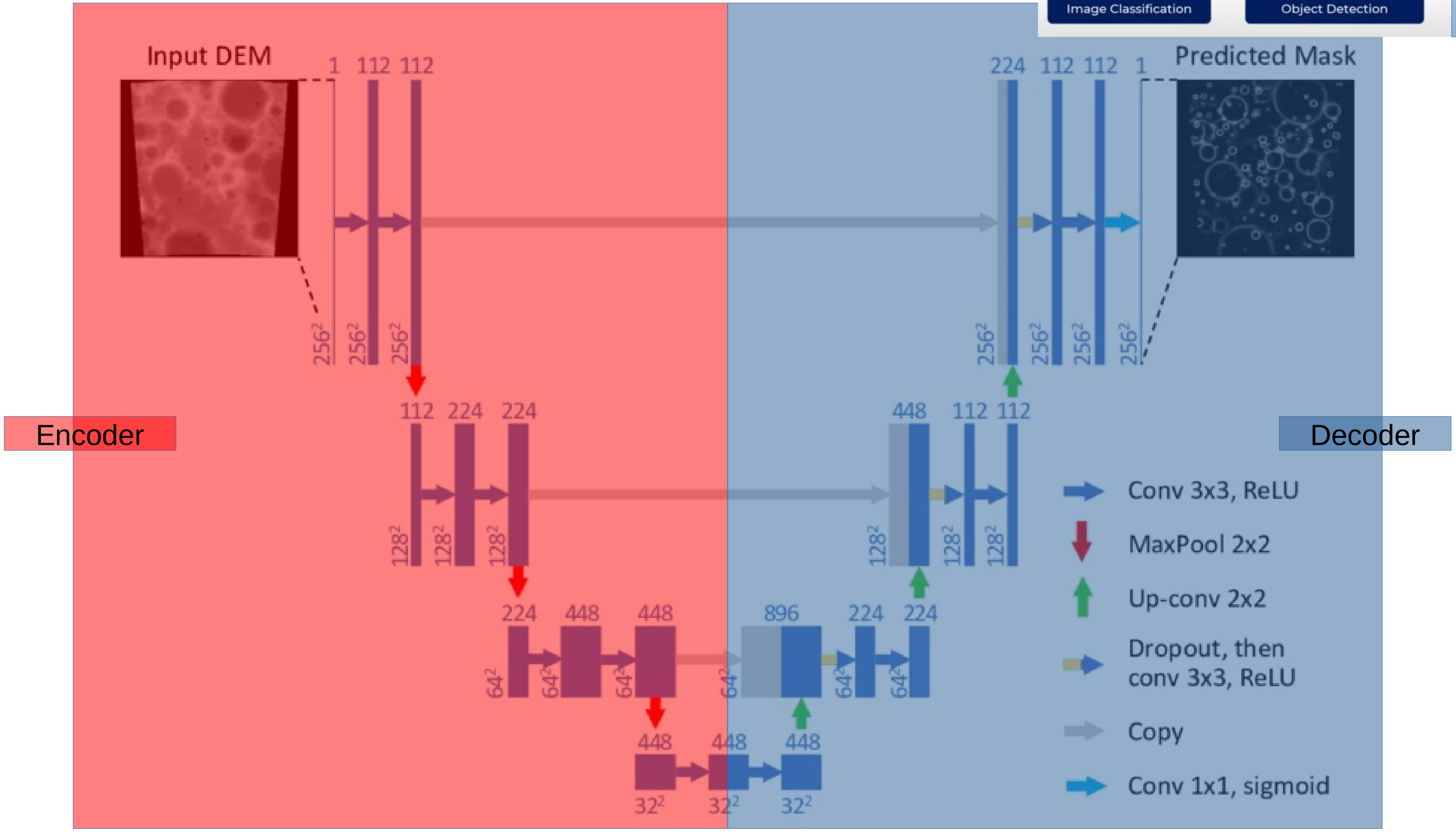
- Faster R-CNN with Binary Mask (2017)



Segmentation - UNET

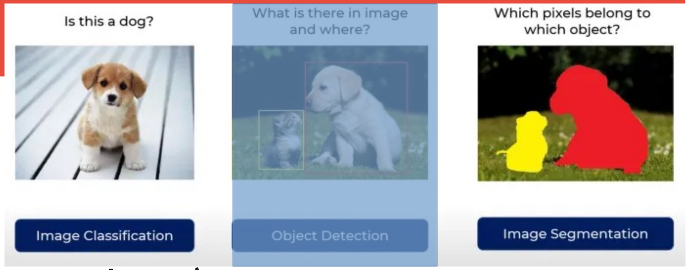
- U-Net (Encoder and Decoder)


Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

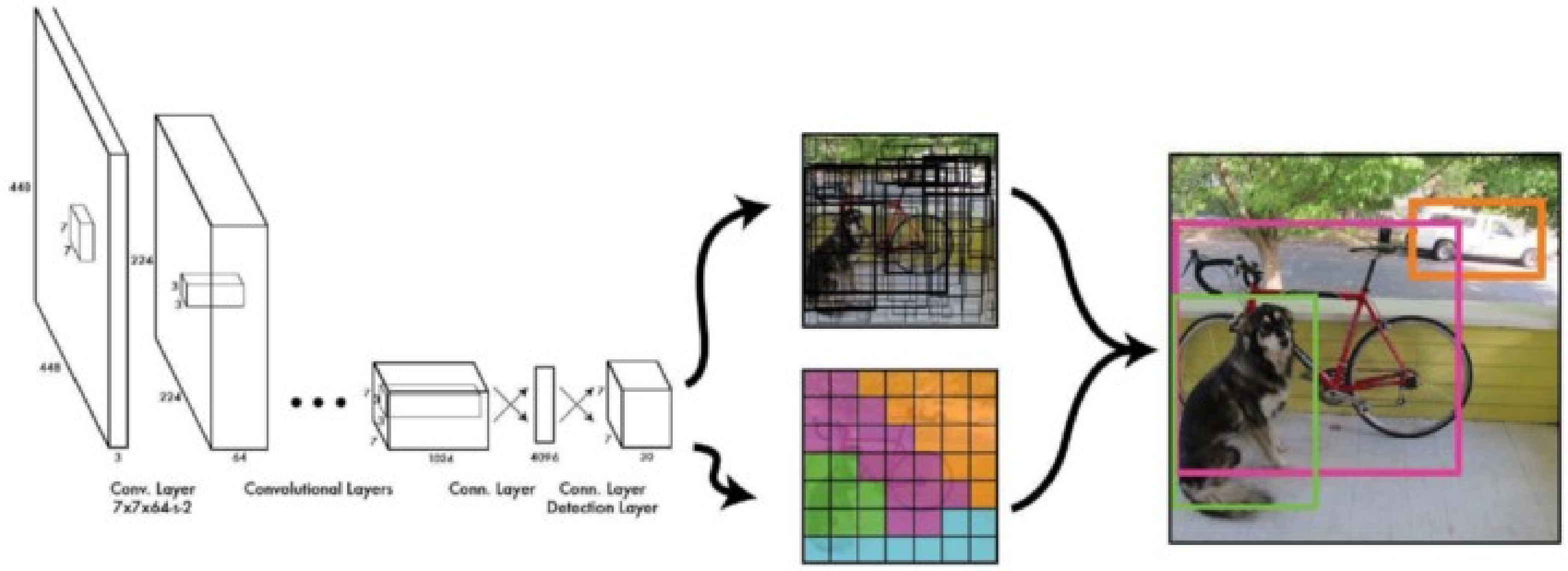


YoLo – You Look Once

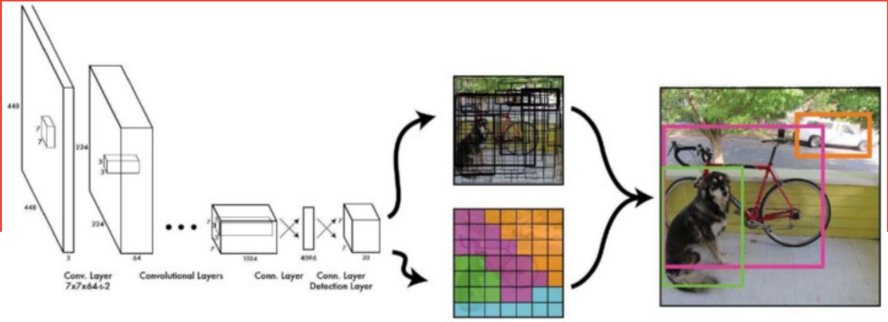
Object Detection - Yolo


Is this a dog?
Image Classification
What is there in image and where?
Object Detection
Which pixels belong to which object?
Image Segmentation

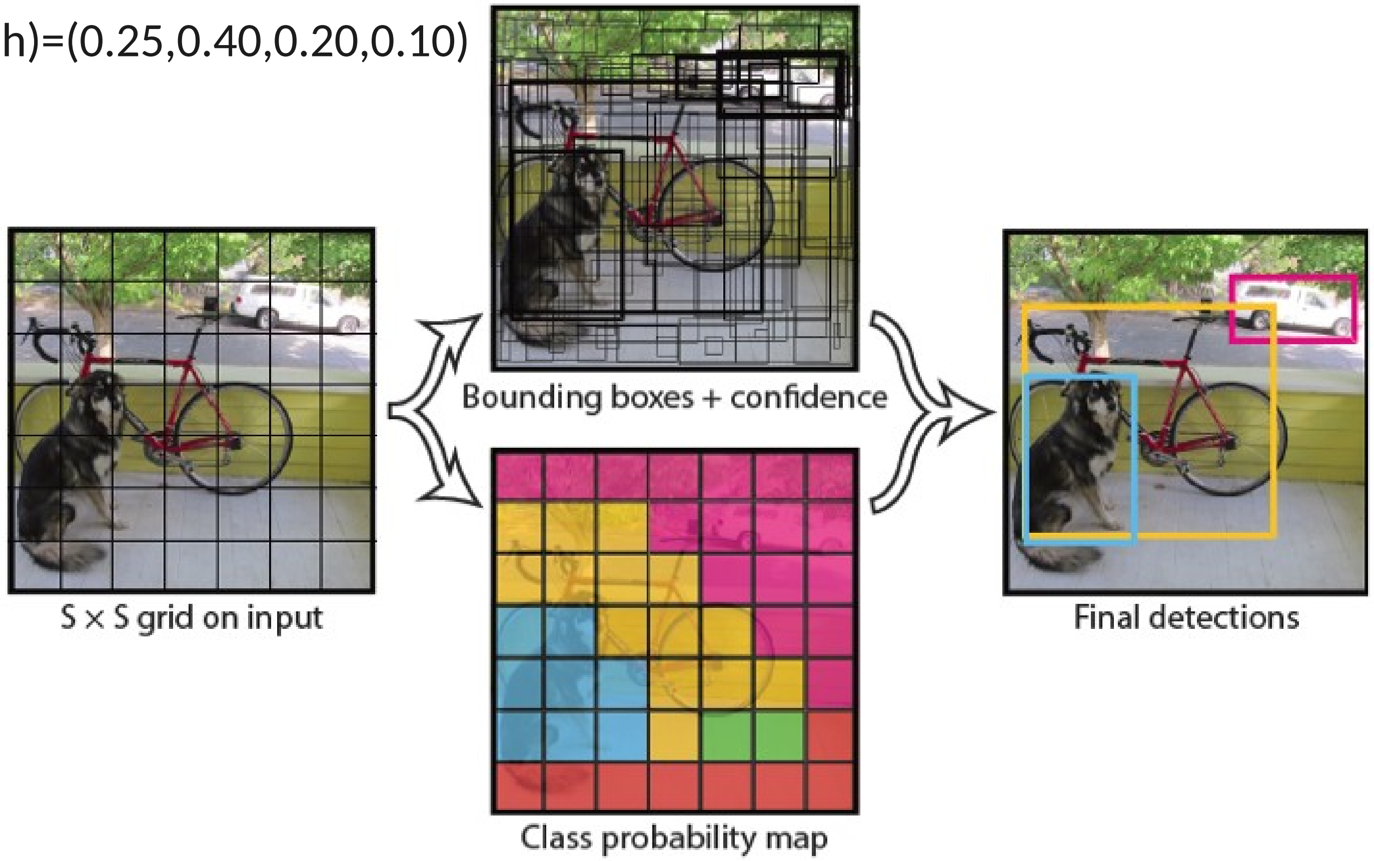
- You Look Once (YoLo - 2015 - now)
  - The input image passes through convolutional backbone (e.g., Darknet)
  - The output is a feature map of lower spatial resolution (e.g for instance, 80×80, 20x20)
  - The split is applied on the latent (feature map), not the raw input. Each feature cell corresponds to a specific spatial region (receptive field) of the input image.



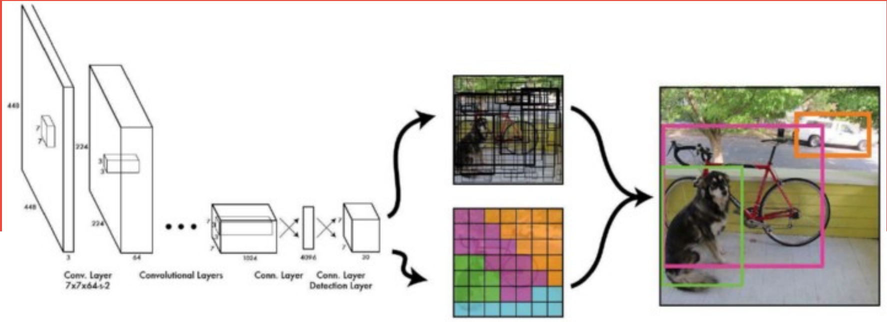
Object Detection - Yolo



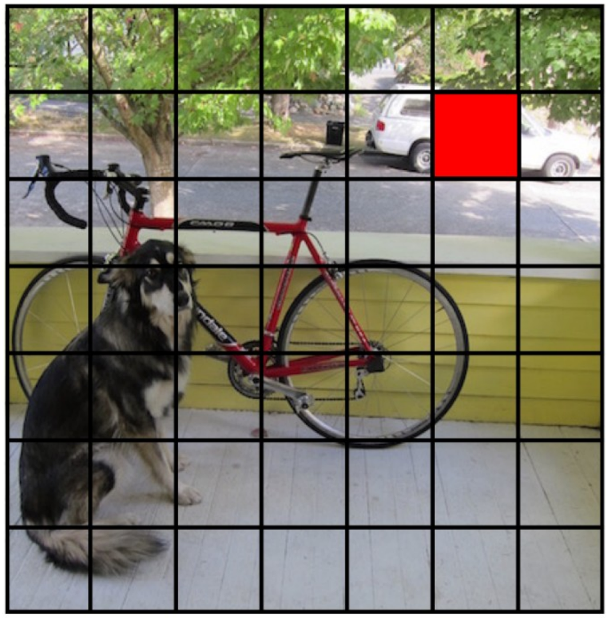
- Regression is the key!
  - Bounding boxes are treated as continuous variablesin normalized image coordinates.
    - (x,y,w,h)=(0.25,0.40,0.20,0.10)

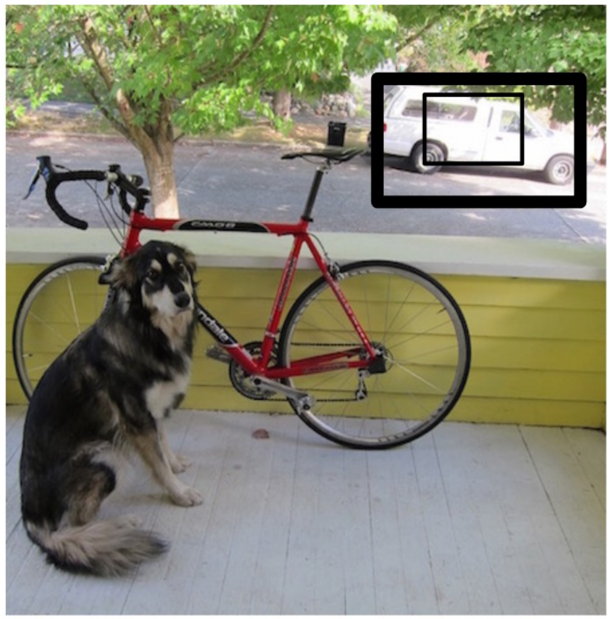


Object Detection - Yolo

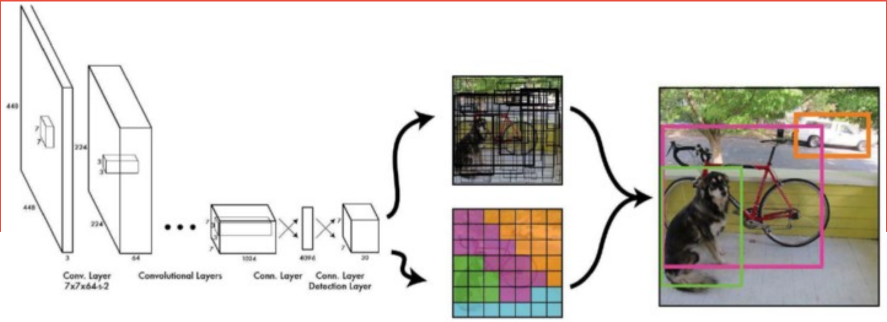


- Each cell predicts bounding boxes and confidences
    - P(object): [0,1] quantifies the confidence that any object occupies this box (not background)

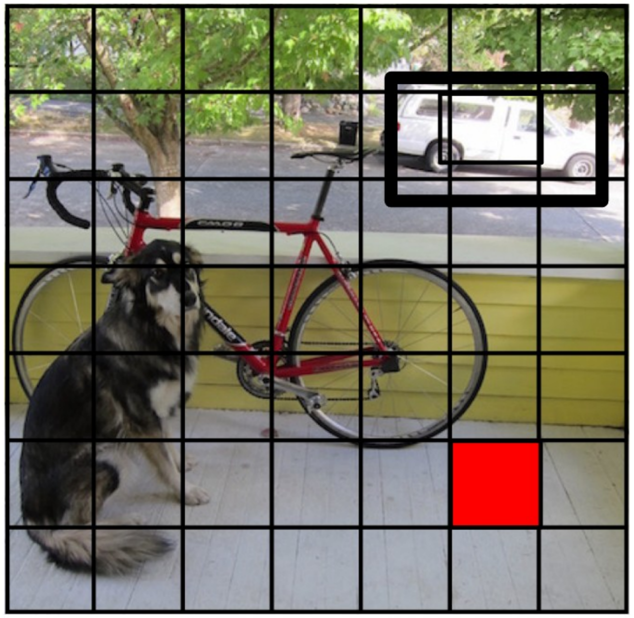


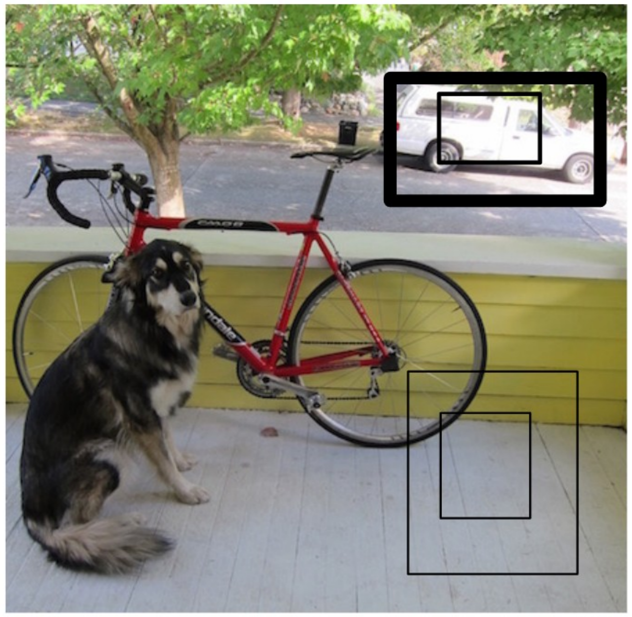


Object Detection - Yolo

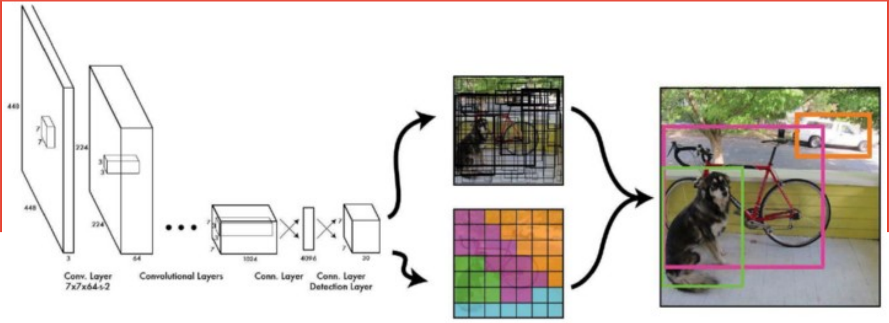


- Each cell predicts bounding boxes and confidences
  - P(object): [0,1] quantifies the confidence that any object occupies this box (not background)

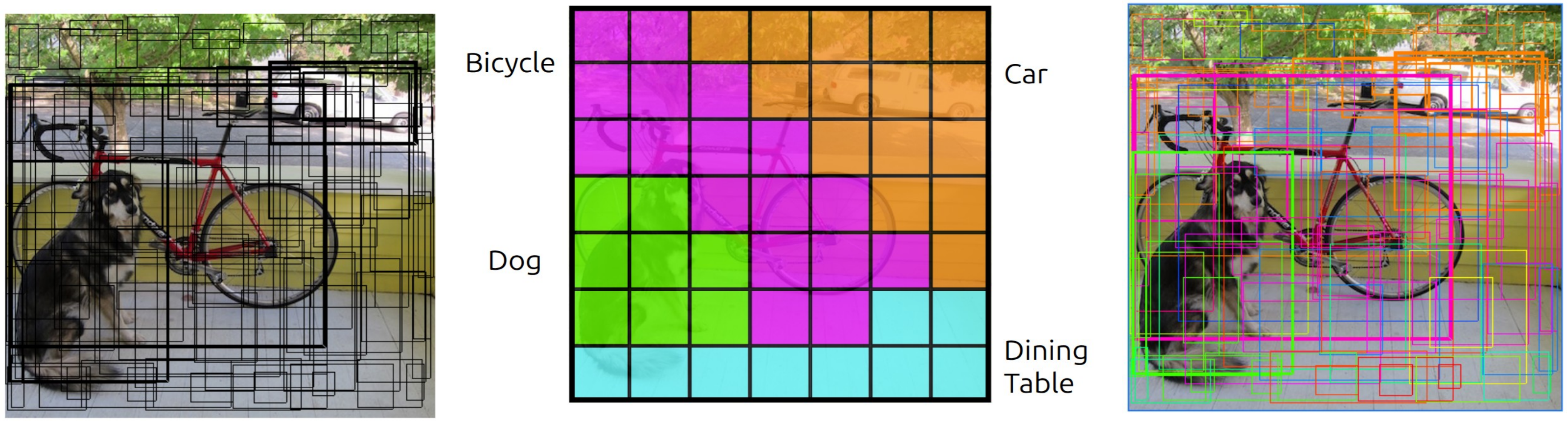




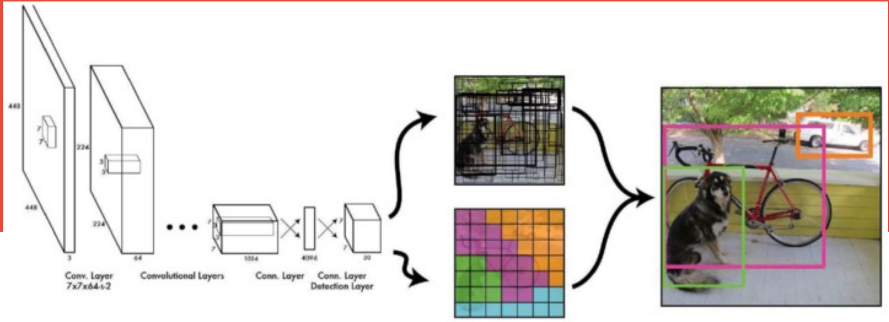
Object Detection - Yolo



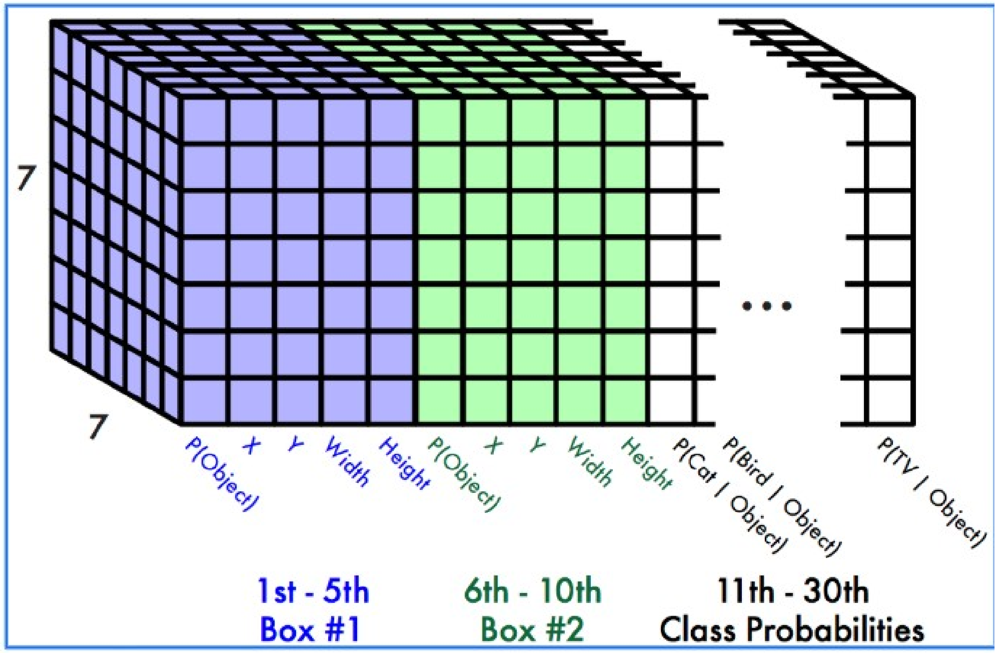
- Class Prob = P(class) => P(car) = 0.8
- Conditionated Prob: e.g P(class | object) => P(car) = 0.9
- Confidence:
  - P(Object) * P (Car | Object)   = 0.72



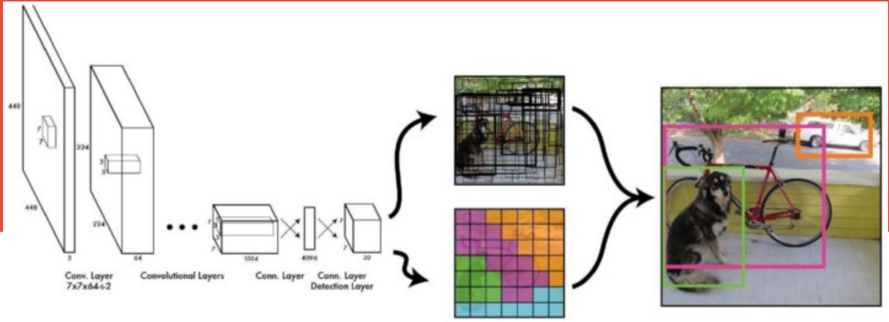
Object Detection - Yolo



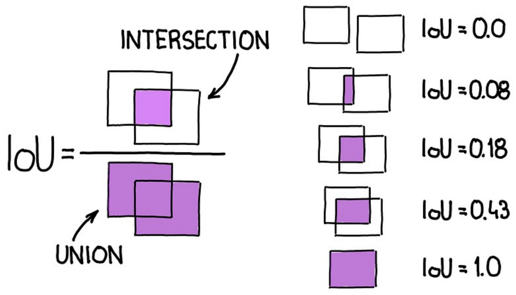
- Each cell predicts:
  - Bounding Boxes:
    - 4 Coordinates (X,Y,W,H)
    - 1 Confidence
  - I.E PASCAL VOC
    - 7x7 Grid
    - 2 Bounding Box / Cell
    - 20 Classes
    - 7 * 7 * (2 * 5 + 20) = 7x7*30 tensors per cell => 1470 predictions per image

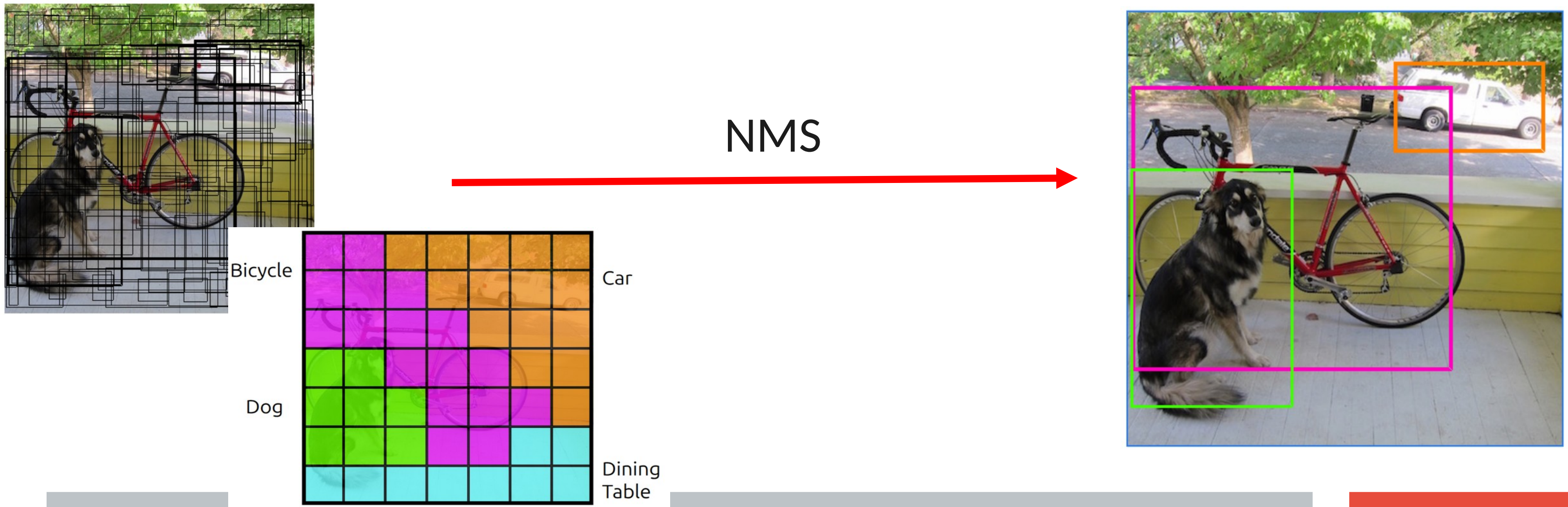


Object Detection - Yolo

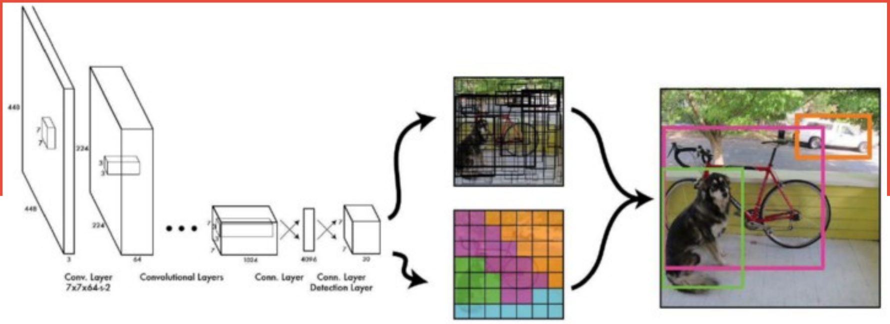


- Non-Maximum Supression (NMS)
  - Sort all boxes by confidence score (P(object)×P(class).
  - Pick the box with the highest score → keep it as the best detection.
    - Compute IoU between this box and all others
    - Remove all boxes with IoU above the suppression threshold (e.g., 0.5)
    - Repeat until all boxes are processed

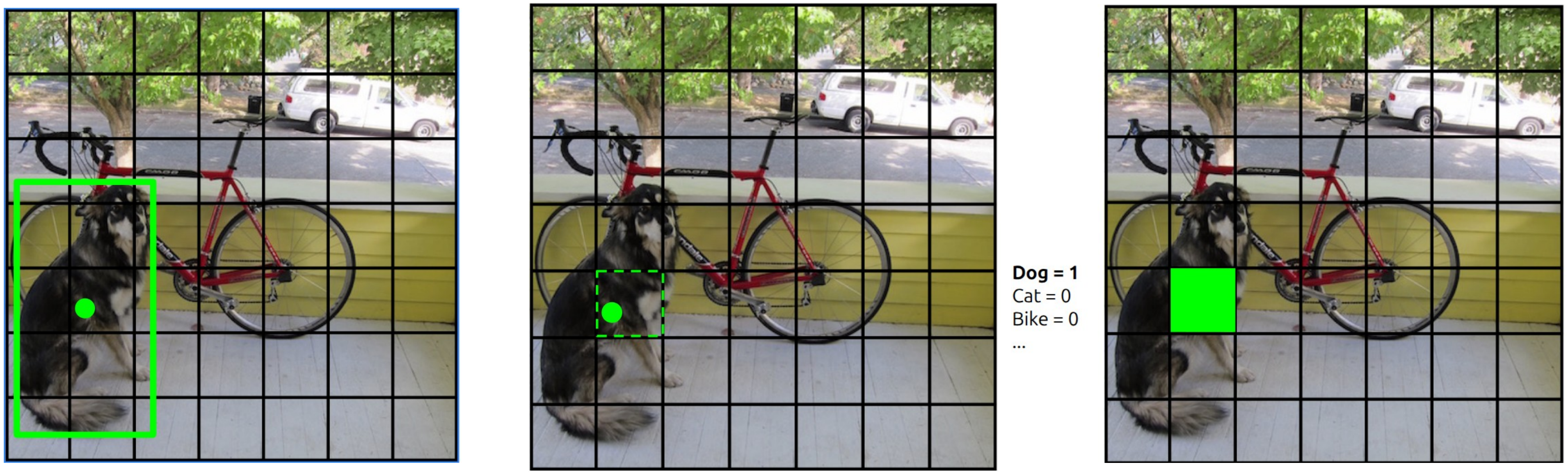




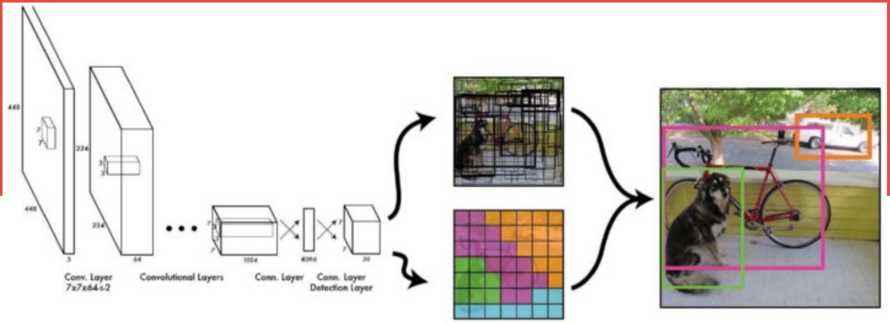
Object Detection - Yolo



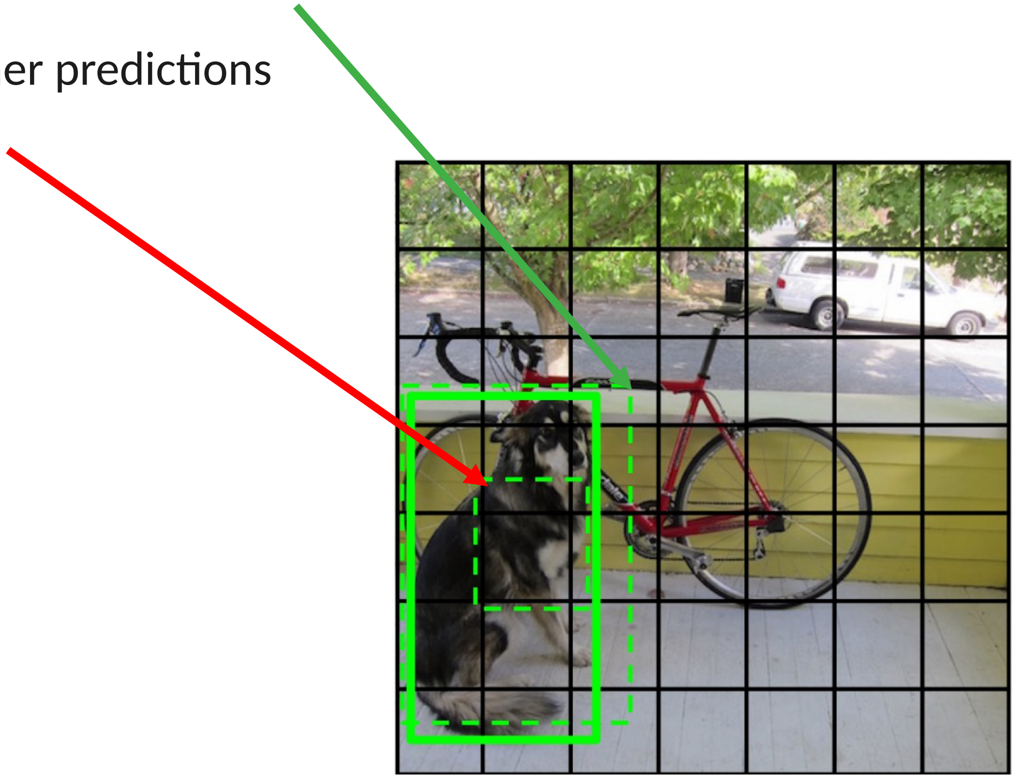
- Training
  - Match example to the right cell (ground-truth)



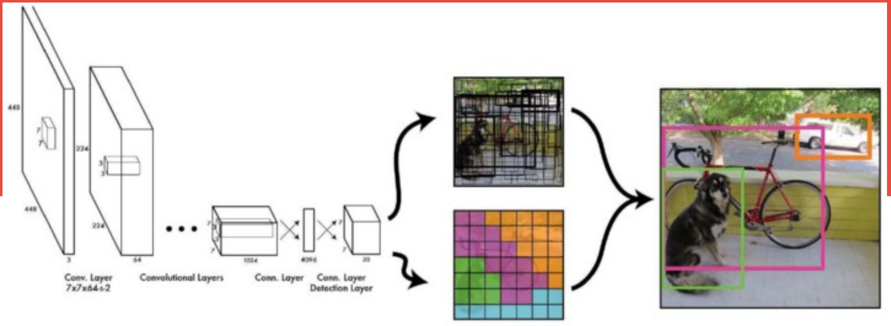
Object Detection - Yolo



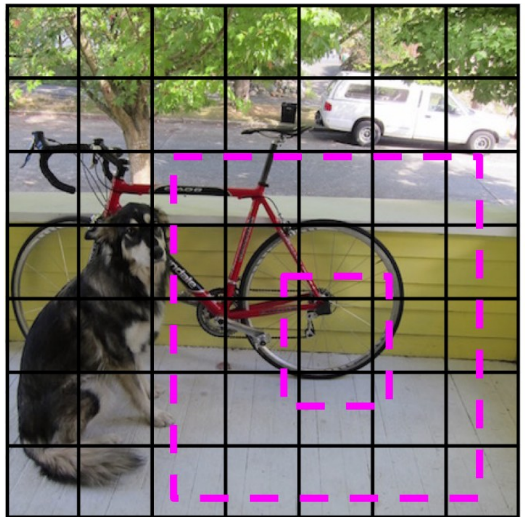
- Training
  - Predict Bounding-Boxes
    - Selects the best fit and increases its confidence
    - Penalizes all other predictions

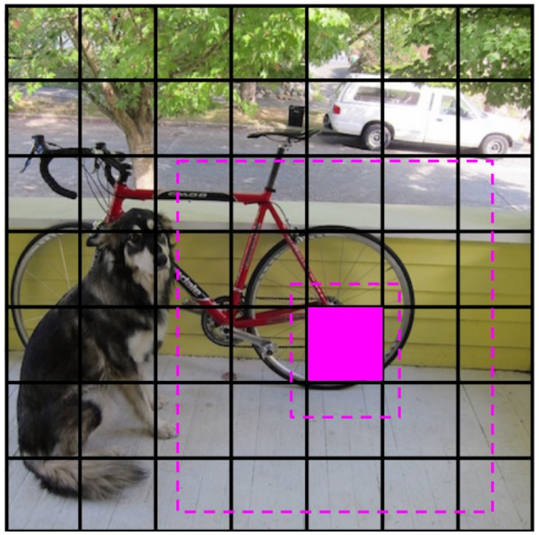


Object Detection - Yolo

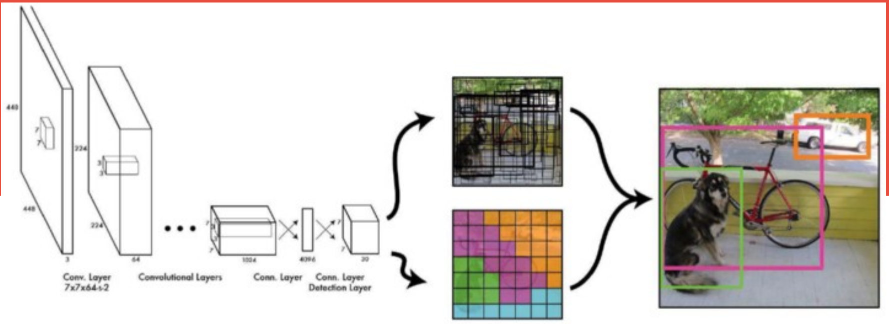


- Training
  - Penalizes when the prediction does not match any class (i.e., background).



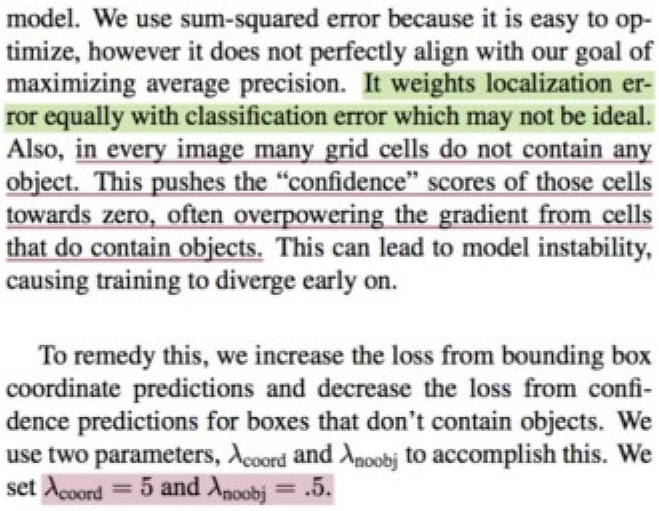


Object Detection - Yolo



loss function:

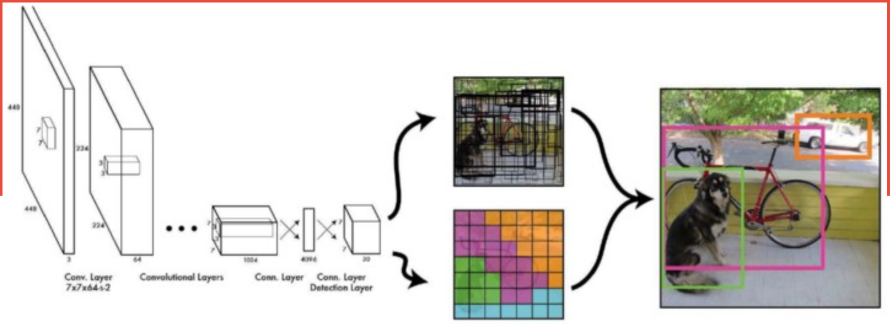
$$
\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]
$$
$$
+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]
$$
$$
+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2
$$
$$
+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2
$$
$$
+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
$$


model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. We use two parameters, λ_{coord} and λ_{noobj} to accomplish this. We set $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = .5$.

$$\lambda_{\text{coord}}=5, \ \lambda_{\text{noobj}}=0.5$$

Object Detection - Yolo



loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\textrm{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{noobj}}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \boxed{\mathbb{1}_{i}^{\text{obj}}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

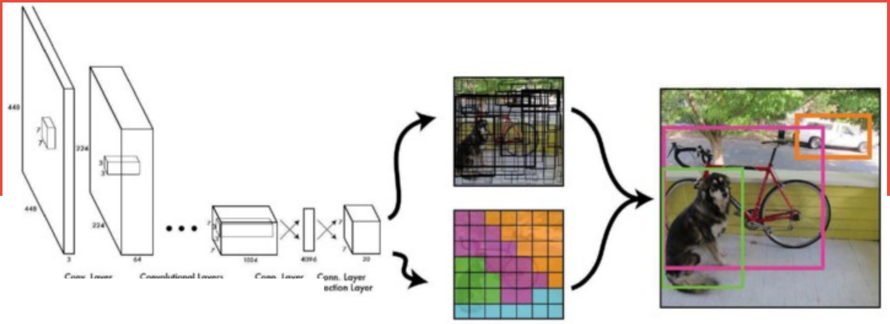
$\boxed{\mathbb{1}_{ij}^{obj}}$ **The *j*th bbox predictor** in ***cell i*** is "responsible" for that prediction

$\boxed{\mathbb{1}_{ij}^{noobj}}$

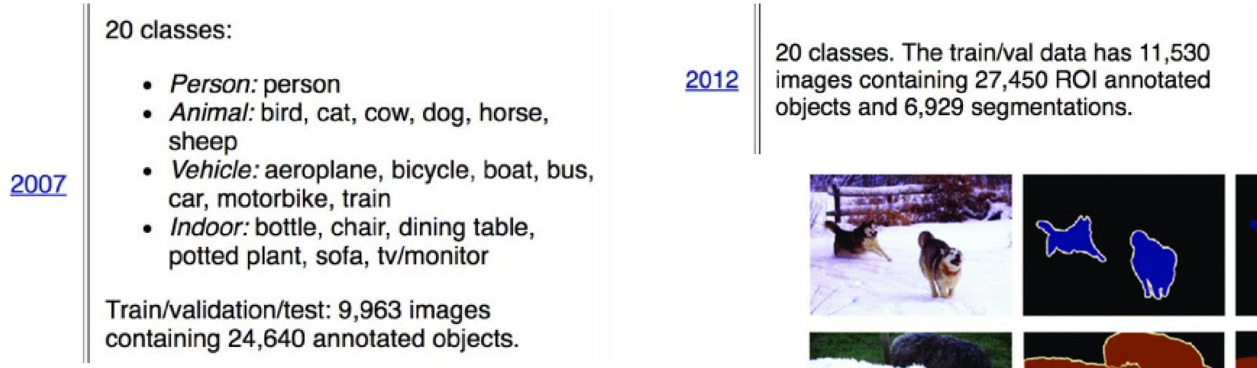
$\boxed{\mathbb{1}_{i}^{obj}}$ If object appears in ***cell i***

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

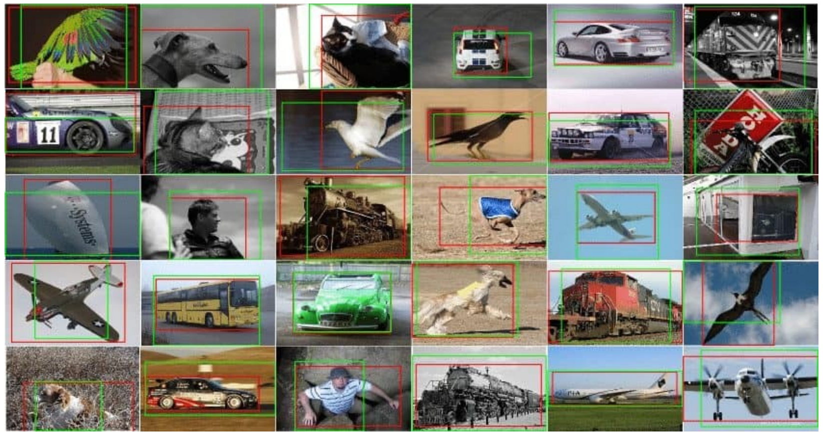
Object Detection - Yolo

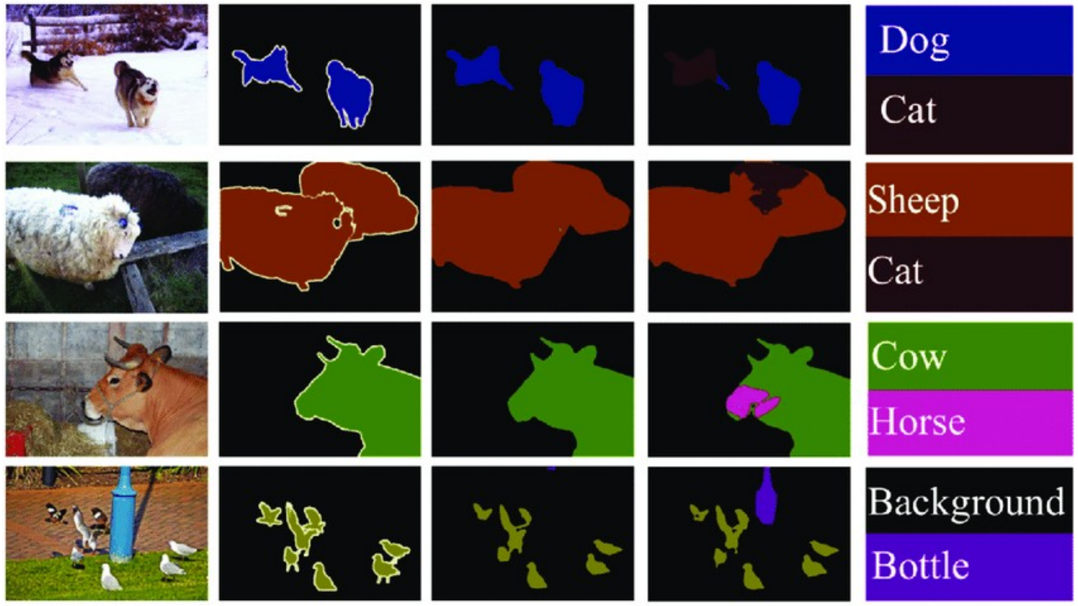


- Datasets
- PASCAL VOC 2007 & VOC 2012

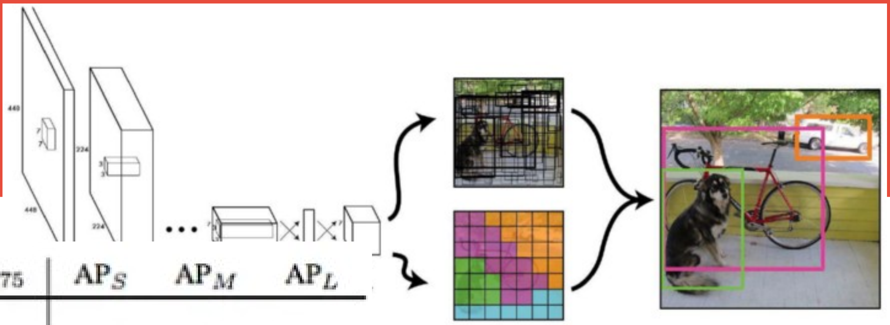


2007	20 classes: • *Person:* person • *Animal:* bird, cat, cow, dog, horse, sheep • *Vehicle:* aeroplane, bicycle, boat, bus, car, motorbike, train • *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor Train/validation/test: 9,963 images containing 24,640 annotated objects.
2012	20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.

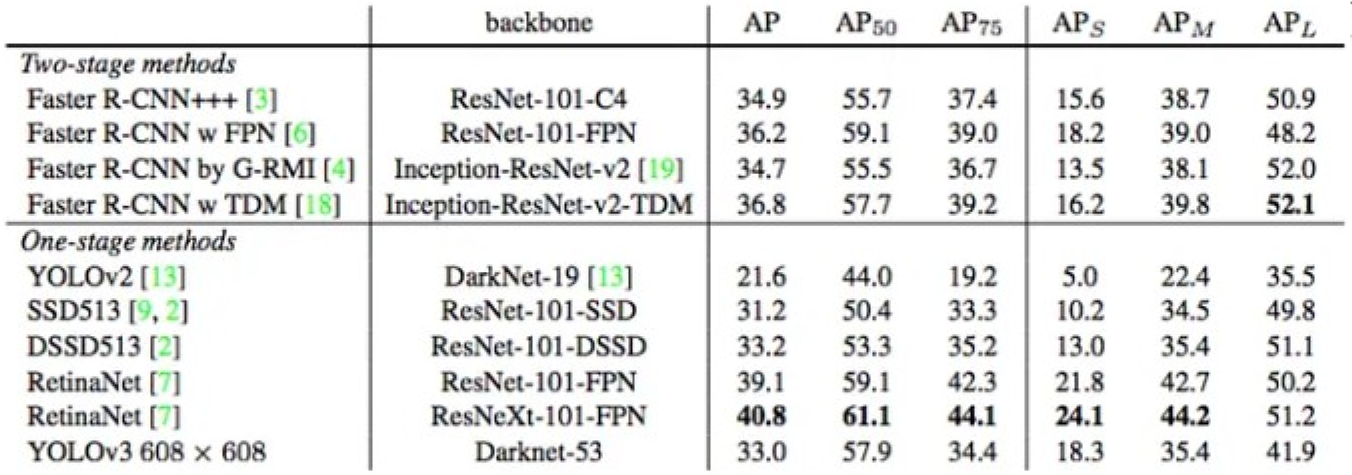




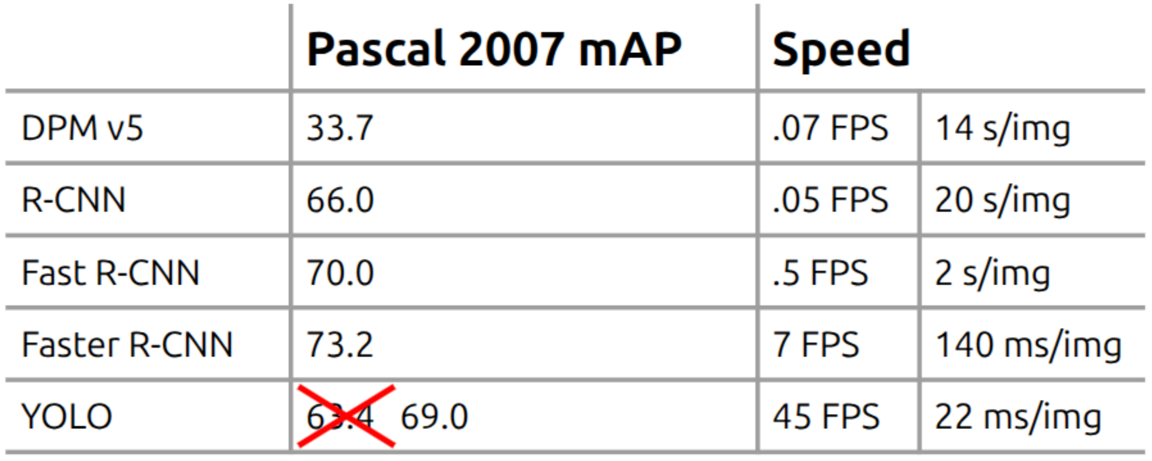
Object Detection - Yolo



•

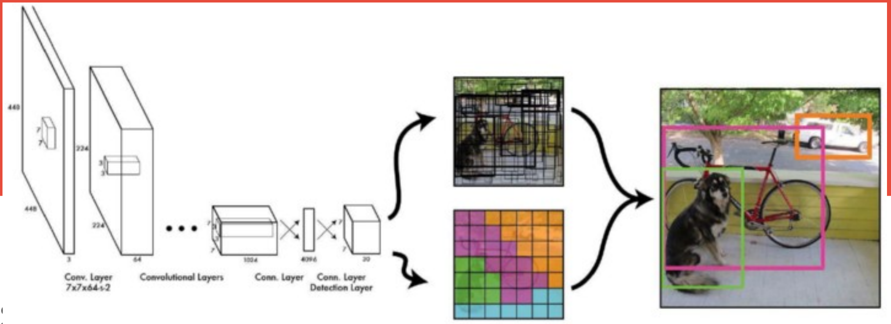


	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9



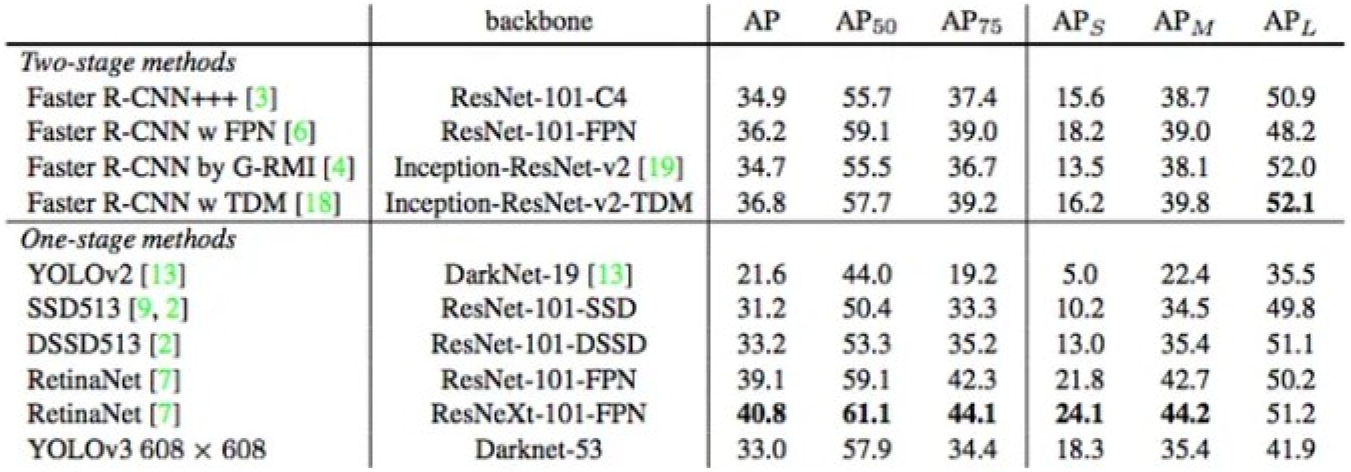
	Pascal 2007 mAP	**Speed**	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	~~63.4~~ 69.0	45 FPS	22 ms/img

Object Detection - Yolo



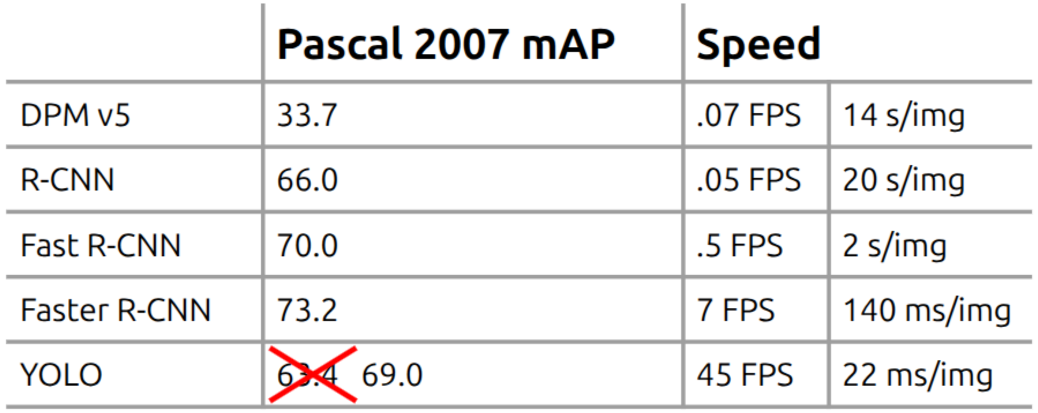
- mAP measures a detector’s average precision acros

-



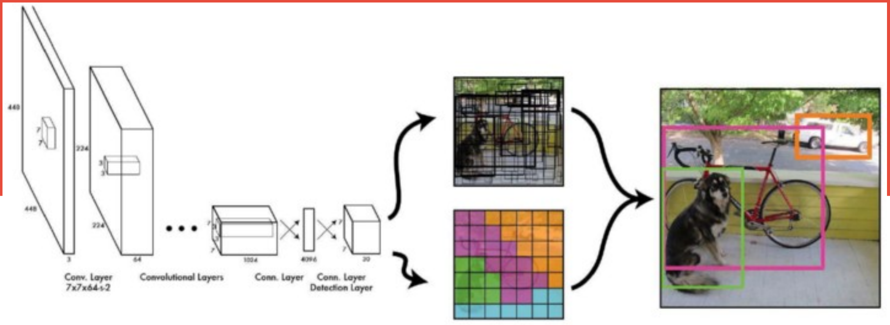
	backbone	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

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



	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	~~63.4~~ 69.0	45 FPS	22 ms/img

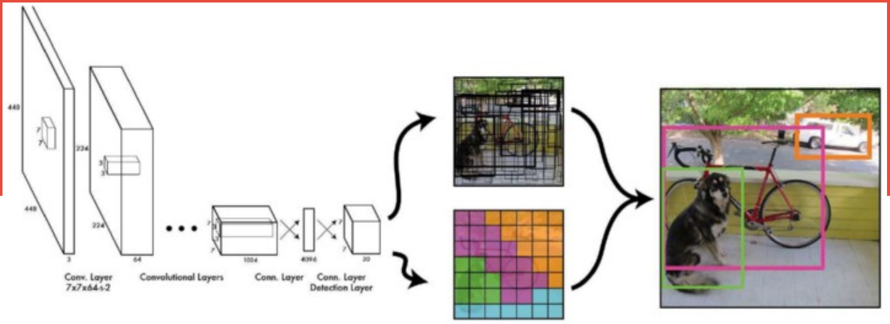
Object Detection - Yolo



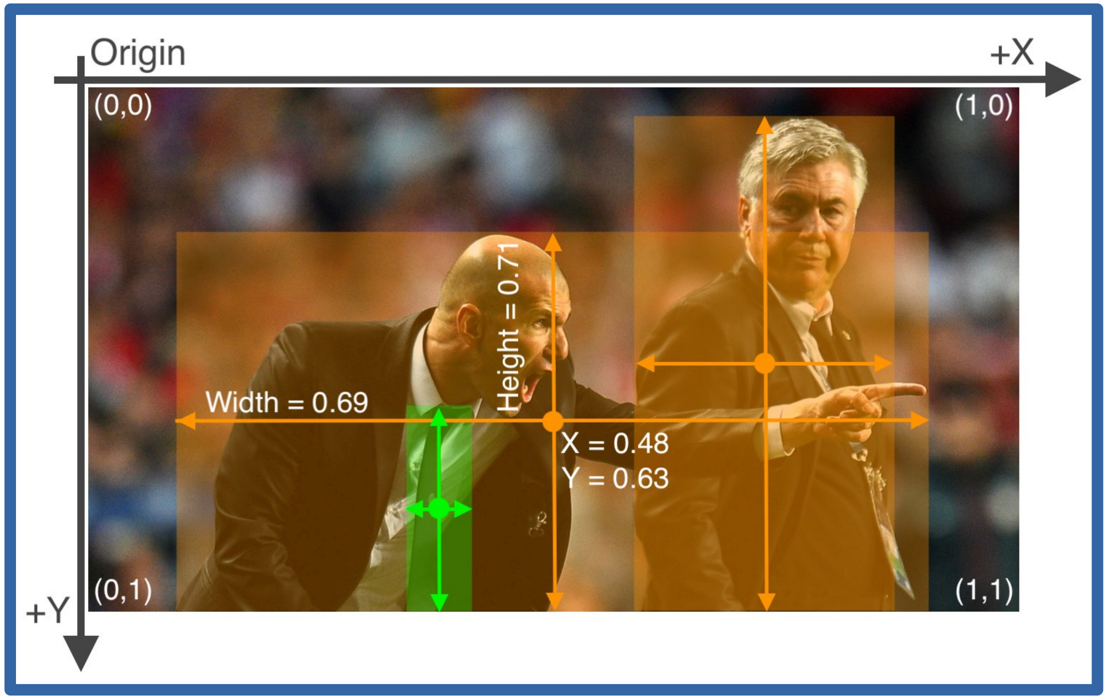
- Let’s Code!
  - This exercise will utilize Ultralytics (www.ultralytics.com) as the framework.
    - Single Image
    - Frame by Frame (Video / Camera)
  - Check out the GitHub repository, specifically the yolo-ultralytics folder.

YoLo – Training and Annotation

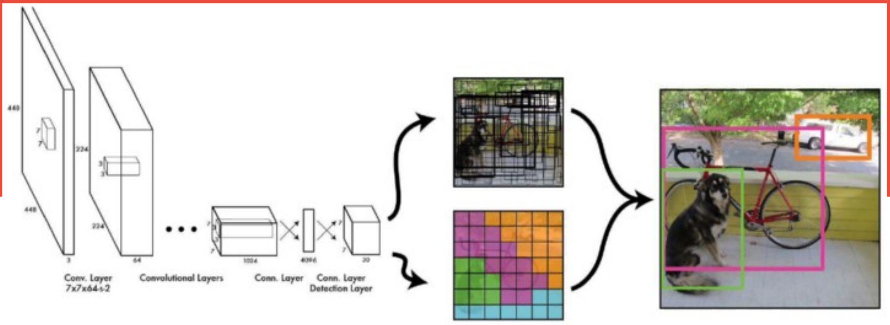
Yolo Training

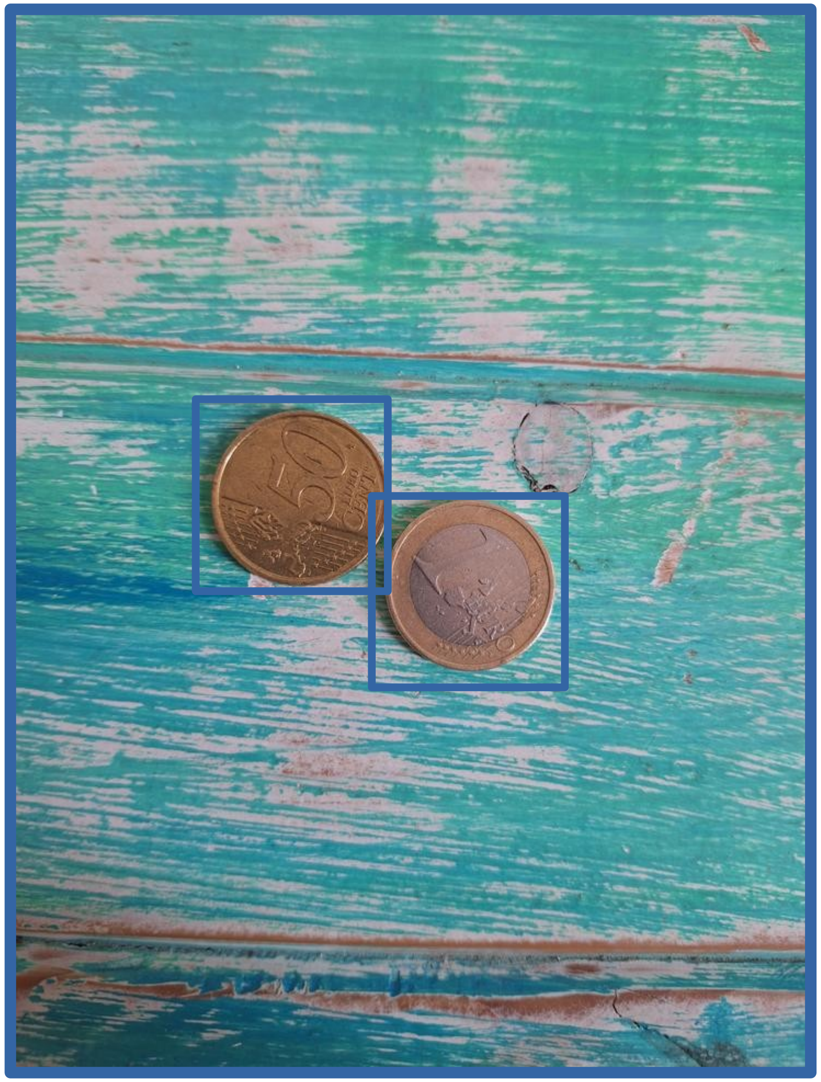


- Each image must include:
  - Bounding box coordinates (4 points) and the corresponding class label
  - Bounding boxes should be relative (normalized), not absolute pixel coordinates
- Avaliable annotation tools:
  - LabelImg (simple and lightweight)
  - LabelMe (browser-based)
  - CVAT (advanced, collaborative)
  - Roboflow Annotate

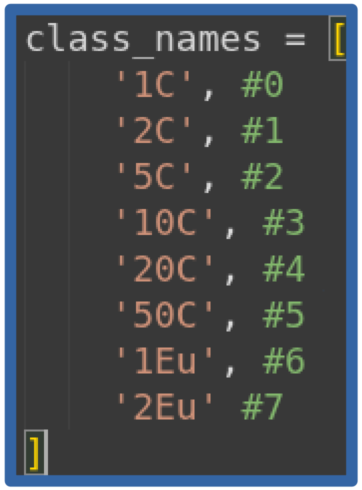


Yolo Training



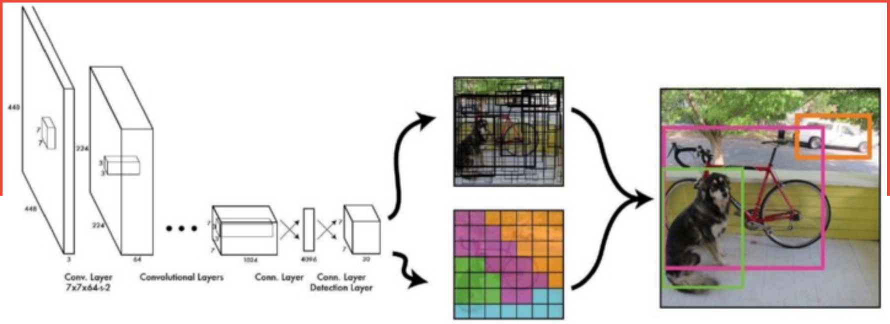


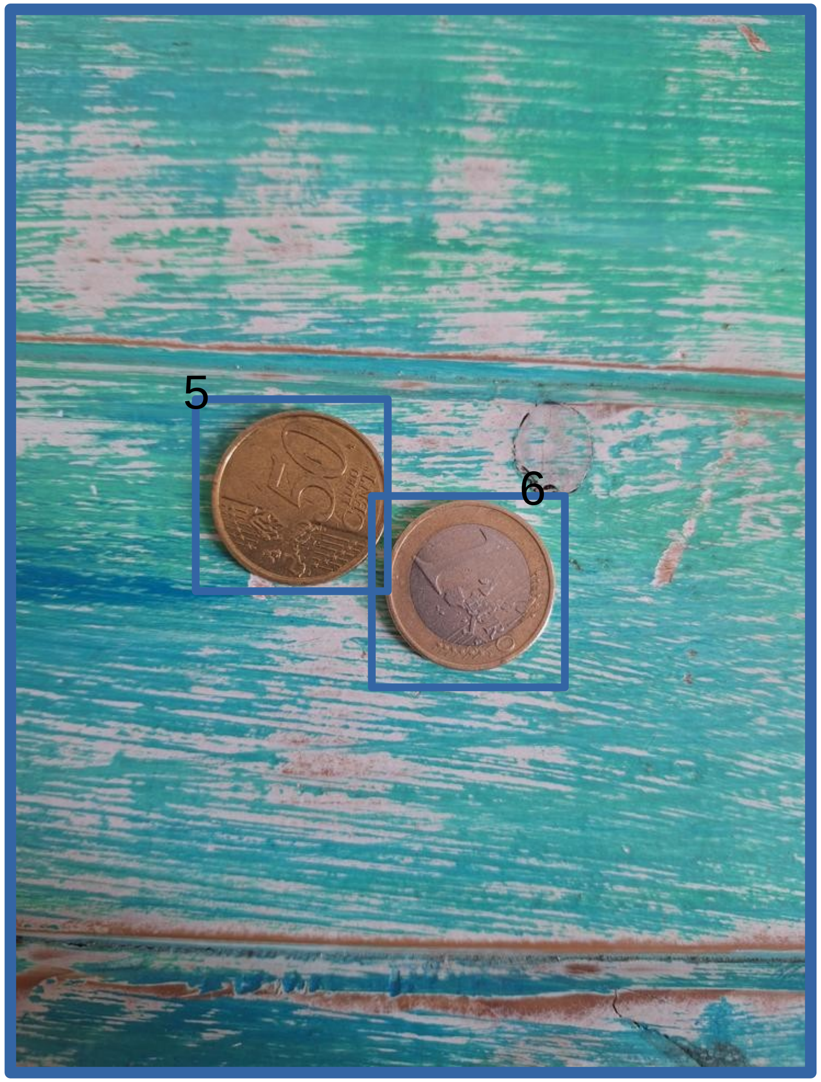
5 0.360417 0.459375 0.229167 0.165625
6 0.575000 0.543750 0.216667 0.162500



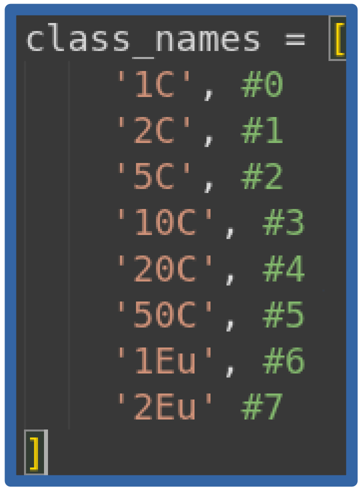
```
class_names = [
        '1C', #0
        '2C', #1
        '5C', #2
        '10C', #3
        '20C', #4
        '50C', #5
        '1Eu', #6
        '2Eu' #7
]
```

Yolo Training



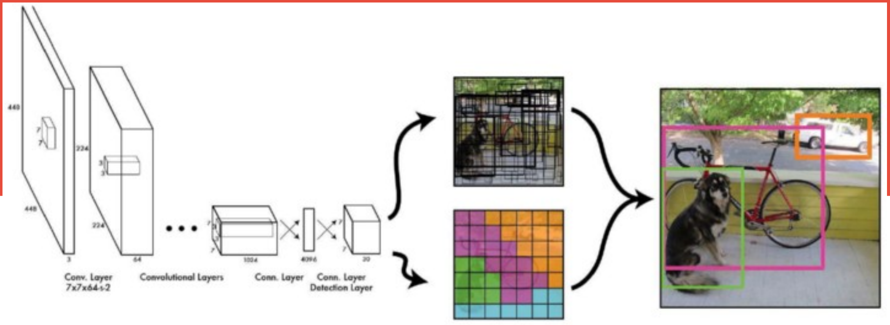


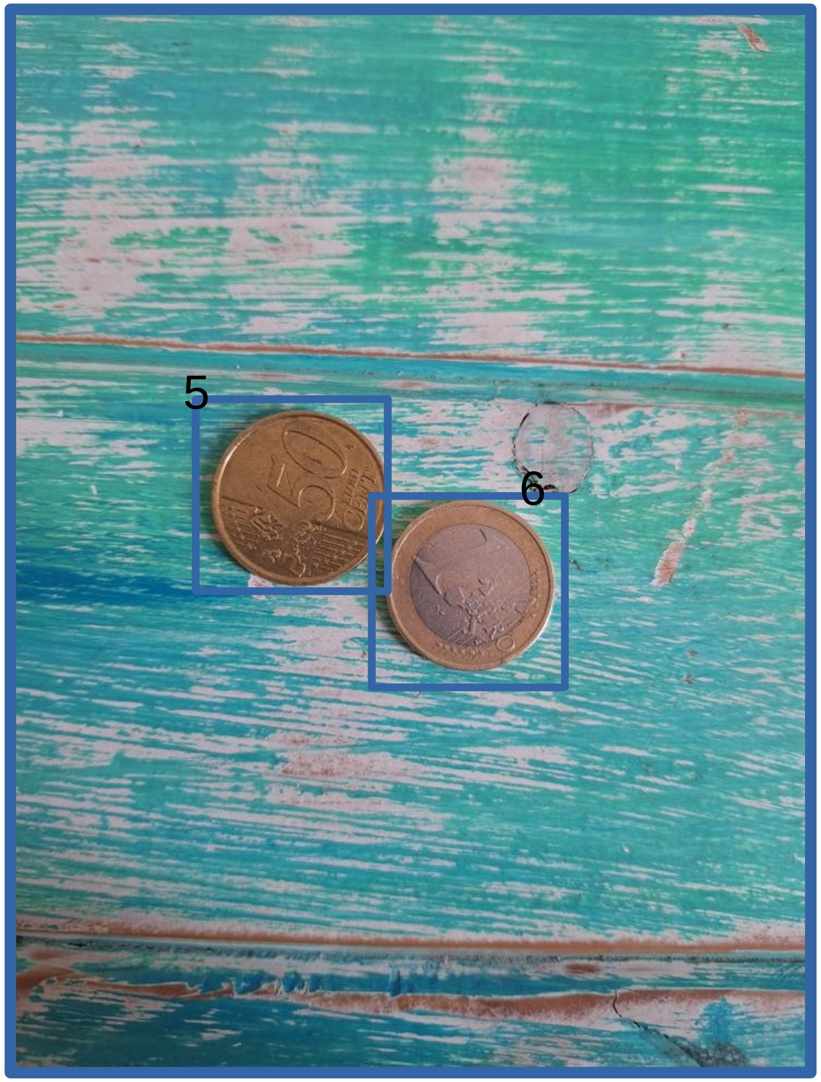
5 0.360417 0.459375 0.229167 0.165625
6 0.575000 0.543750 0.216667 0.162500

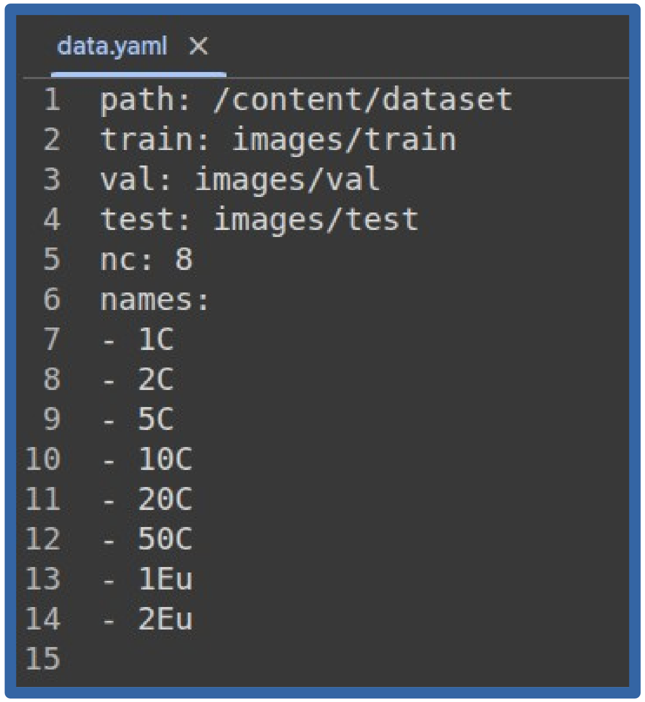


```
class_names = [
        '1C', #0
        '2C', #1
        '5C', #2
        '10C', #3
        '20C', #4
        '50C', #5
        '1Eu', #6
        '2Eu' #7
]
```

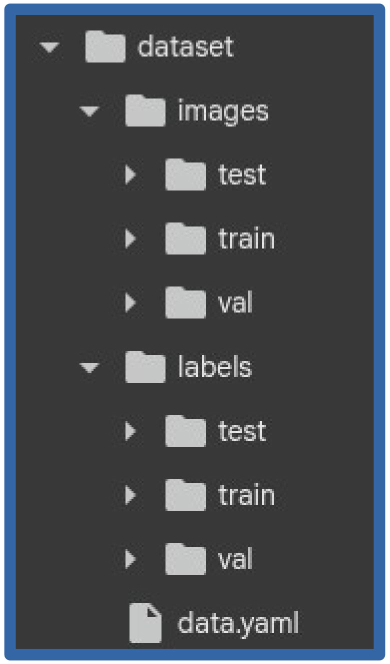
Yolo Training







```
path: /content/dataset
train: images/train
val: images/val
test: images/test
nc: 8
names:
- 1C
- 2C
- 5C
- 10C
- 20C
- 50C
- 1Eu
- 2Eu
```



```
dataset
  images
    test
    train
    val
  labels
    test
    train
    val
  data.yaml
```

Let's Code!