AA 2022/2023

Machine Learning for Modelling: *Supervised Learning*

Simone Bianco

1

A road map of object detection

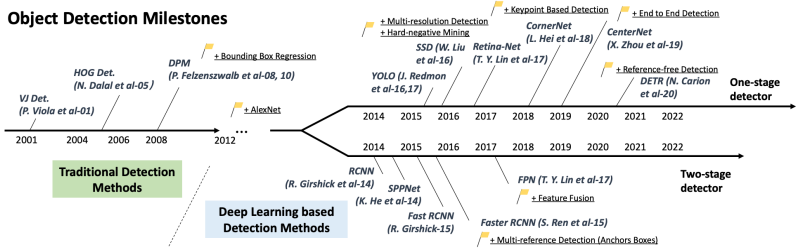


Fig. 2: A road map of object detection. Milestone detectors in this figure: VJ Det. [10, 11], HOG Det. [12], DPM [13–15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20–22], SSD [23], FPN [24], Retina-Net [25], CornerNet [26], CenterNet [27], DETR [28].

2

A road map of object detection

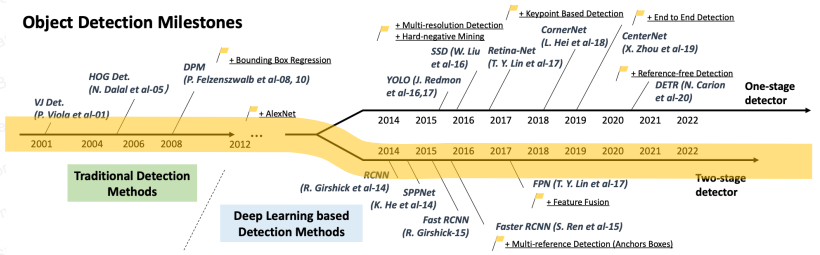


Fig. 2: A road map of object detection. Milestone detectors in this figure: VJ Det. [10, 11], HOG Det. [12], DPM [13–15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20–22], SSD [23], FPN [24], Retina-Net [25], CornerNet [26], CenterNet [27], DETR [28].

3

A road map of object detection

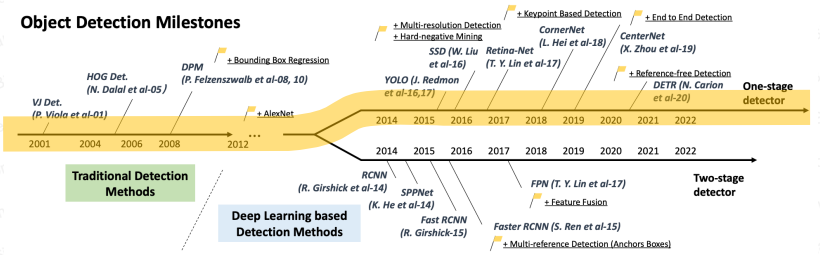


Fig. 2: A road map of object detection. Milestone detectors in this figure: VJ Det. [10, 11], HOG Det. [12], DPM [13–15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20–22], SSD [23], FPN [24], Retina-Net [25], CornerNet [26], CenterNet [27], DETR [28].

4

Two-stage vs one-stage

Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network:

- **Two-shot object detection** uses two passes of the input image to make predictions about the presence and location of objects.
- The first pass generates a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than single-shot object detection but is also more computationally expensive.

5

Two-stage vs one-stage

Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network:

- **Single-shot object detection** uses a single pass of the input image to make predictions about the presence and location of objects in the image.
- They process an entire image in a single pass, making them computationally efficient.
- However, single-shot object detection is generally less accurate than other methods, and it's less effective in detecting small objects.
- Such algorithms can be used to detect objects in real time in resource-constrained environments.

6

Two-stage vs one-stage

- Overall, the choice between single-shot and two-shot object detection depends on the specific requirements and constraints of the application.
- Generally, single-shot object detection is better suited for real-time applications, while two-shot object detection is better for applications where accuracy is more important.

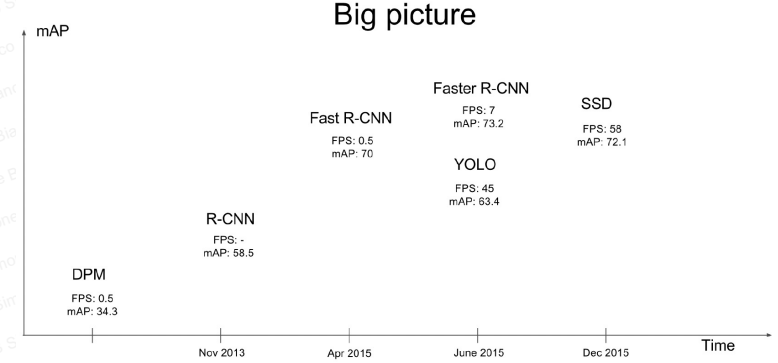
7

AA 2022/2023

You Only Look Once (YOLO)

8

YOLO



9

YOLO: main concepts

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

- Object detection:
  - cast as a regression problem (as already seen in R-CNNs)
- YOLO: only one feedforward;
  - global context
- Unified (real-time detection):
  - YOLO: 45 fps
  - Fast YOLO: 155 FPS
- General representation:
  - Robust on various backgrounds
  - Other domains

[v1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE CVPR (pp. 779-788).

10

YOLO: main concepts

- YOLO: single regression problem
- Image → bounding box coordinates and class probability
- Extremely fast
- Global reasoning
- Generalizable representation

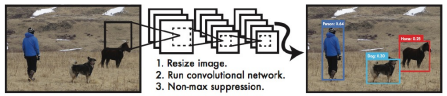


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

11

YOLO: unified detection

- All BBox, All classes

1. Image → divided into S x S grids
2. For each grid cell:
   → B: displacement of the BBox (for precise localization) and Confidence score
   → C: class probabilities wrt # classes

- On Pascal VOC (20 classes):
  - S=7
  - B=2
  - C=20

→ prediction is a 7x7x30 tensor

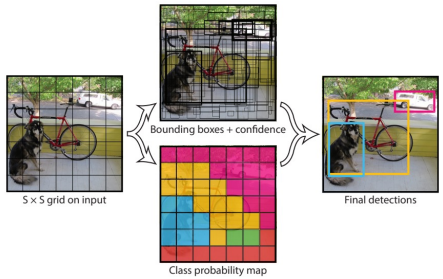
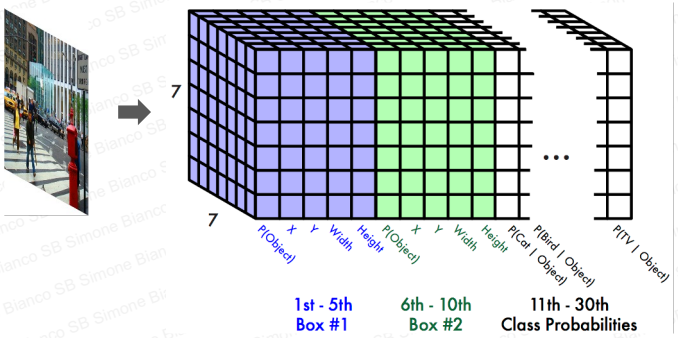


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

12

YOLO: unified detection



13

YOLO: unified detection

- YOLO divides an input image into an S × S grid.
- If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.
- Each grid cell predicts B bounding boxes and confidence scores for those boxes.
- These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

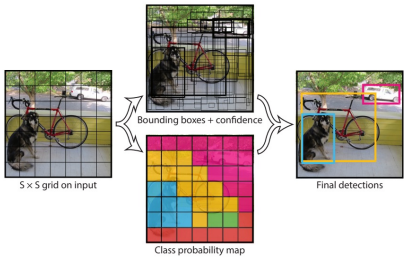


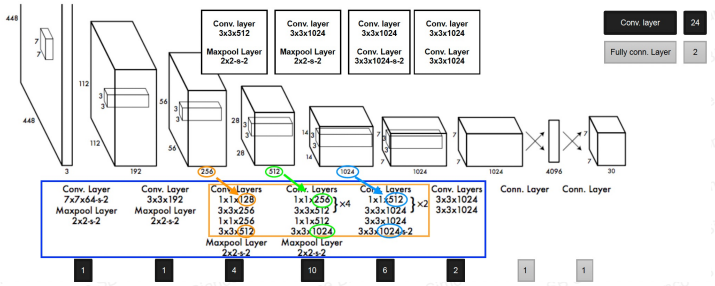
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

14

YOLO: network design

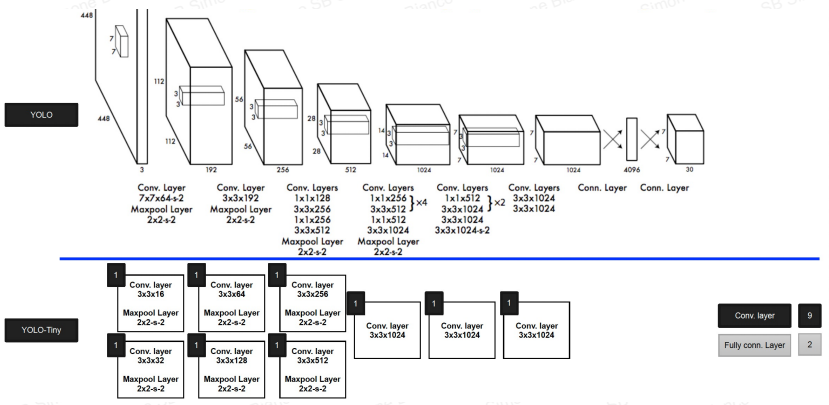
- Modified GoogLeNet
- 1x1 reduction layer

Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1×1 reduction layers followed by 3×3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.



15

YOLO vs Fast YOLO: network design

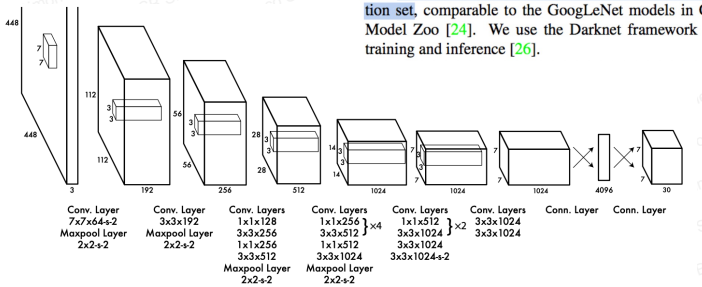


16

YOLO: training

- Pre-train with ImageNet 1000-class ILSVRC challenge dataset

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [30]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe's Model Zoo [24]. We use the Darknet framework for all training and inference [26].

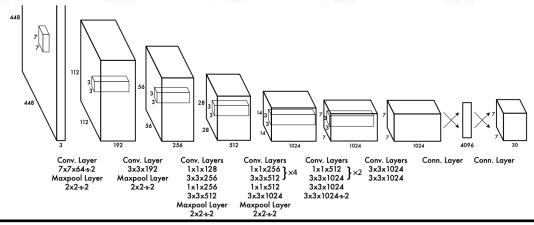


17

YOLO: training

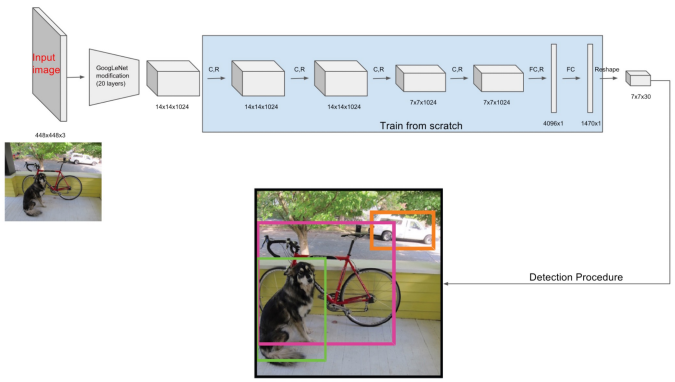
- Convert the model to perform detection
- Add both convolutional and fully connected layers
- Increase input resolution: 224x224 → 448x448

We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [29]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224×224 to 448×448.



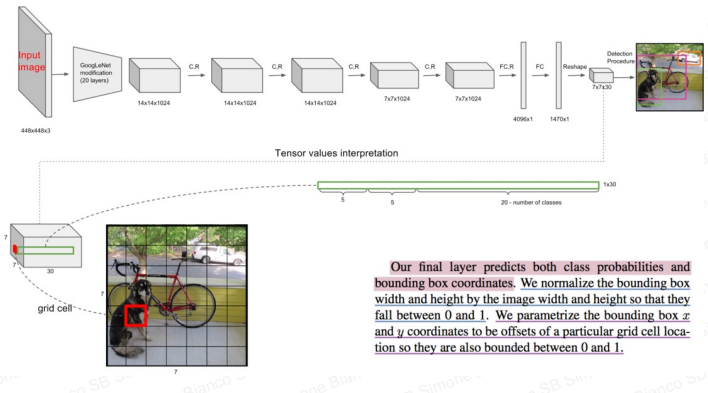
18

YOLO: training



19

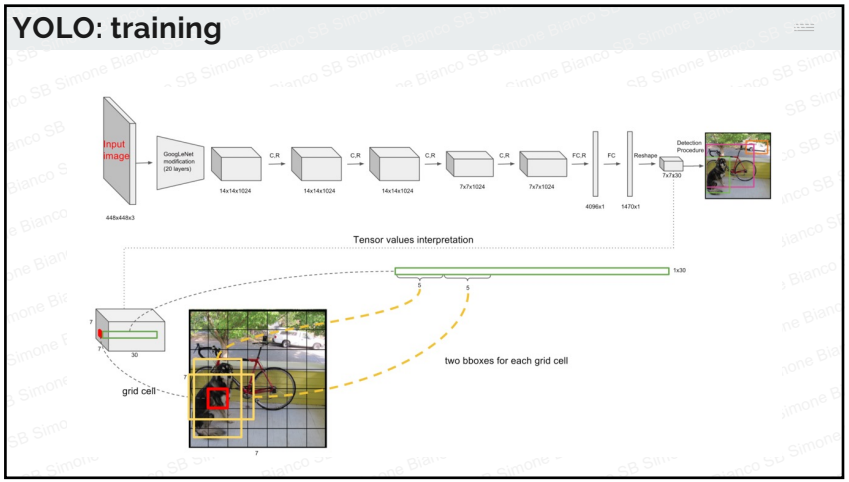
YOLO: training



Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

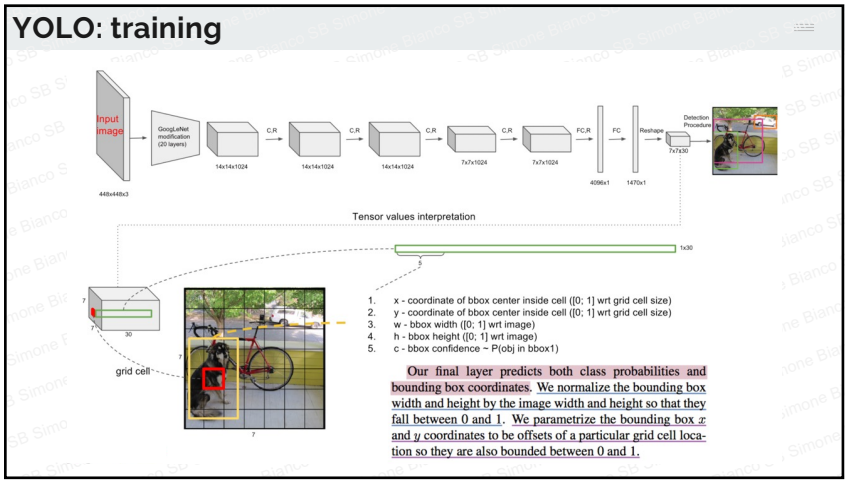
20

YOLO: training



21

YOLO: training



Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

22

YOLO: loss function

Loss function (sum-squared error)

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

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

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

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

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

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 $\lambda_{\text{noobj}} = .5$.

23

YOLO: loss function

Loss function (sum-squared error)

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

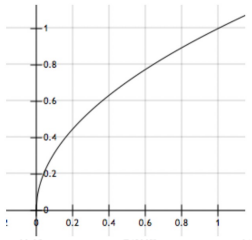
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

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

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

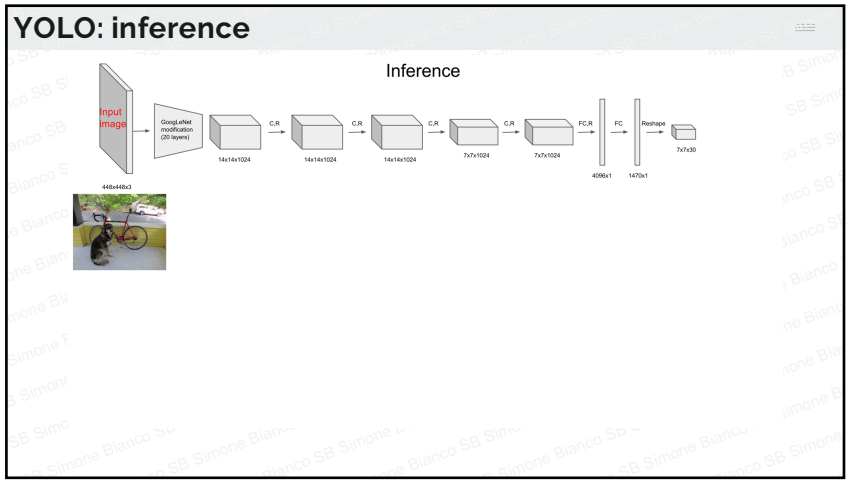
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)$$

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.



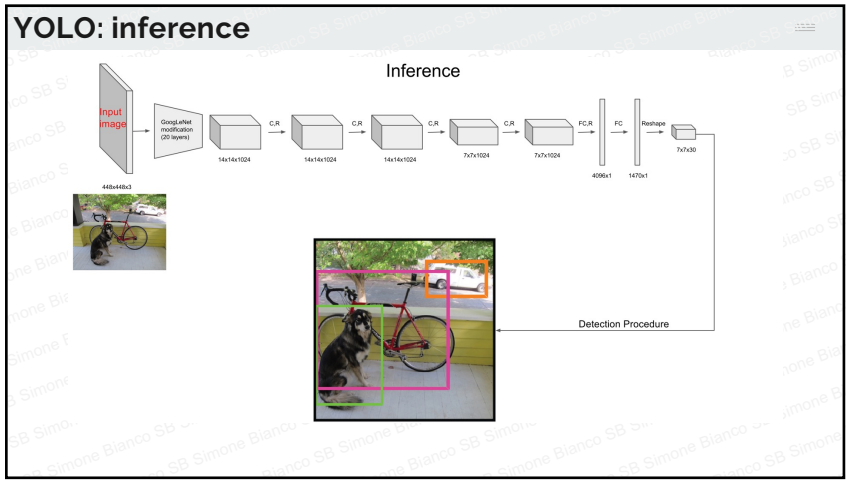
24

YOLO: inference



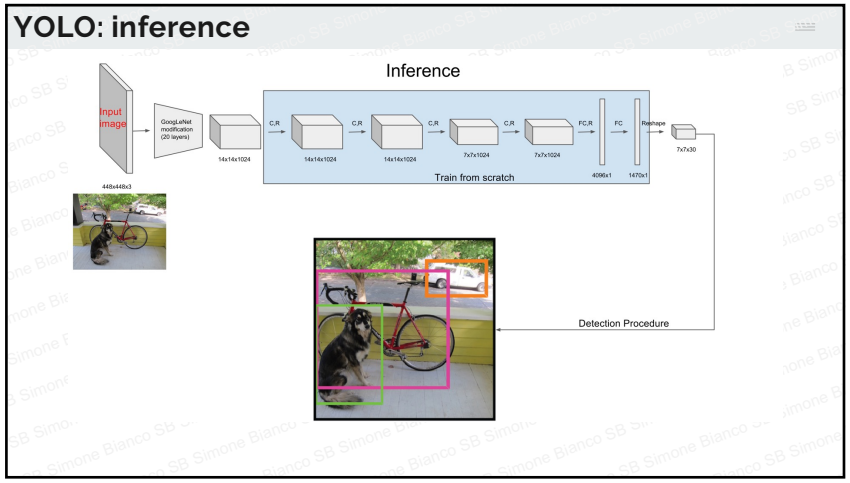
25

YOLO: inference



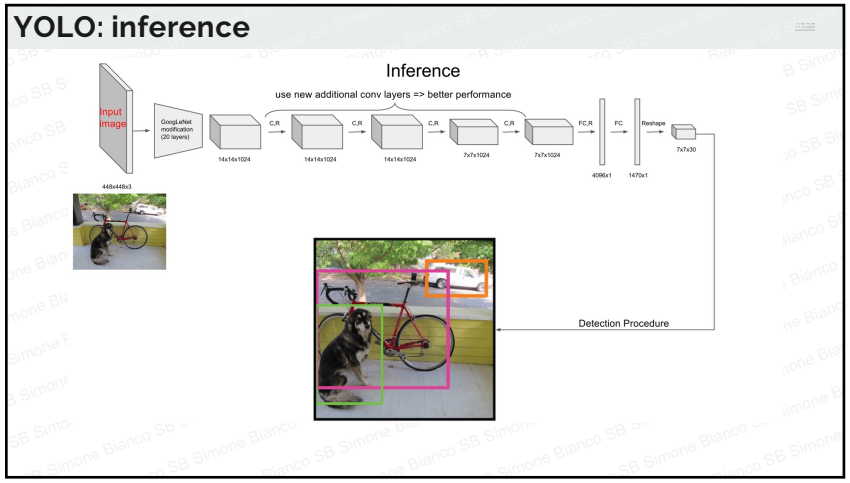
26

YOLO: inference



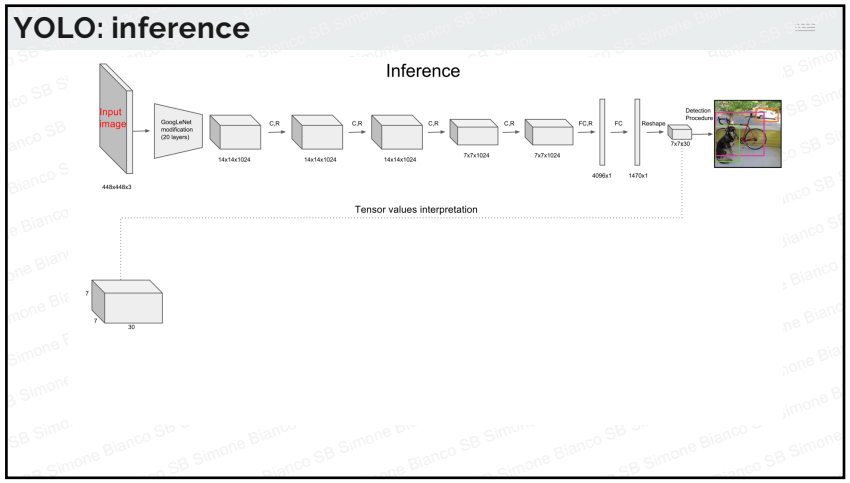
27

YOLO: inference



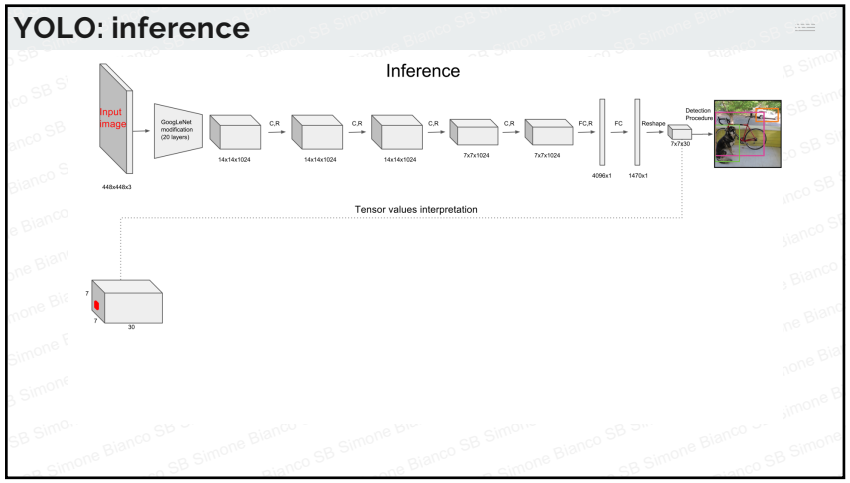
28

YOLO: inference



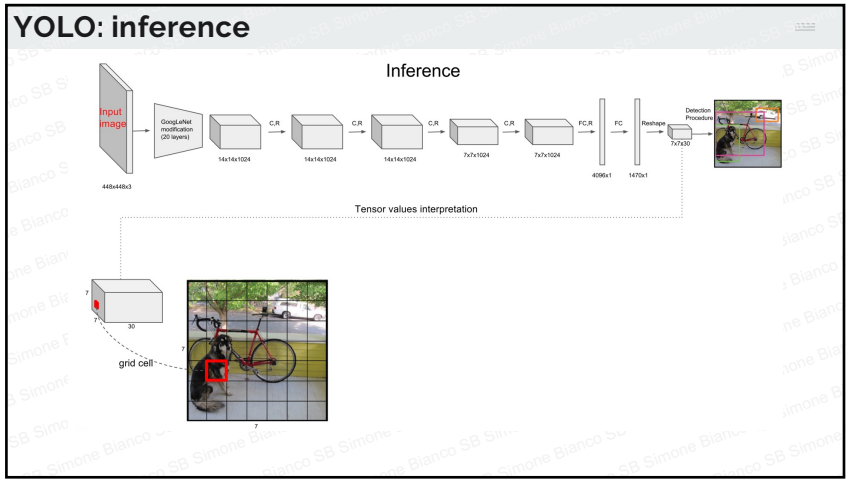
29

YOLO: inference



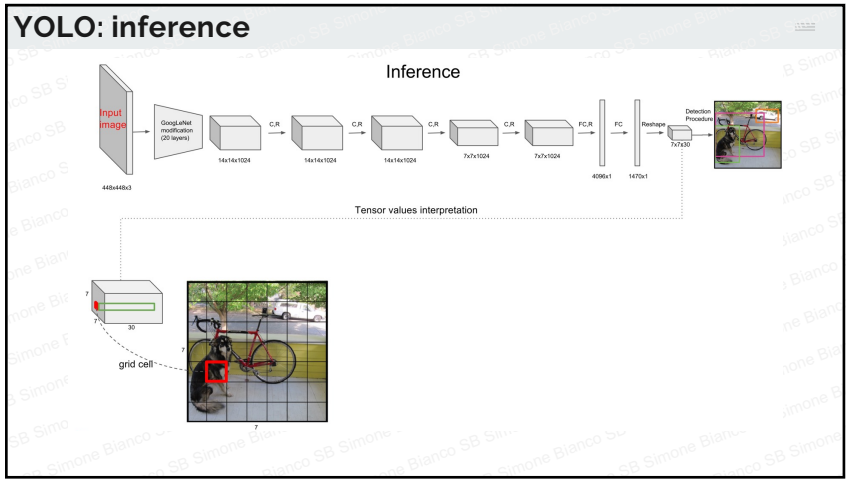
30

YOLO: inference



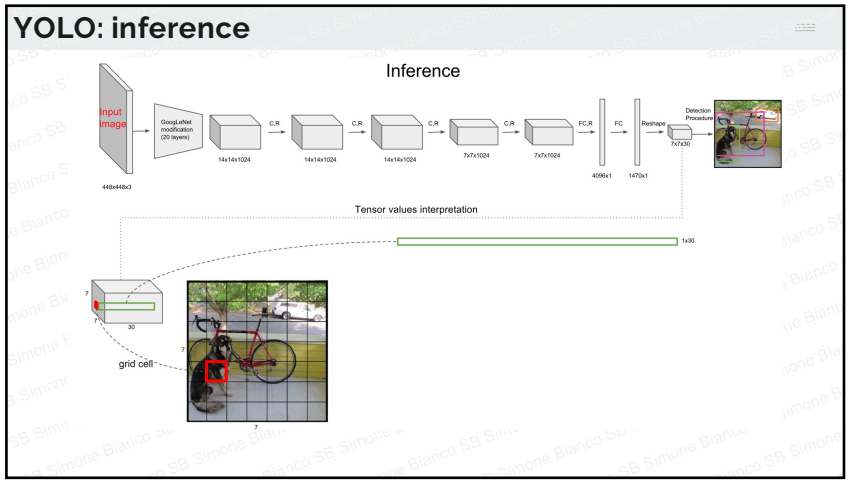
31

YOLO: inference



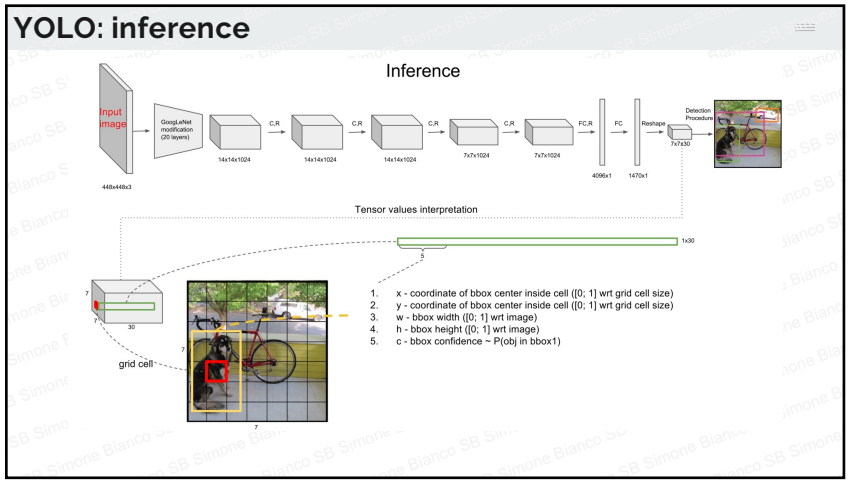
32

YOLO: inference



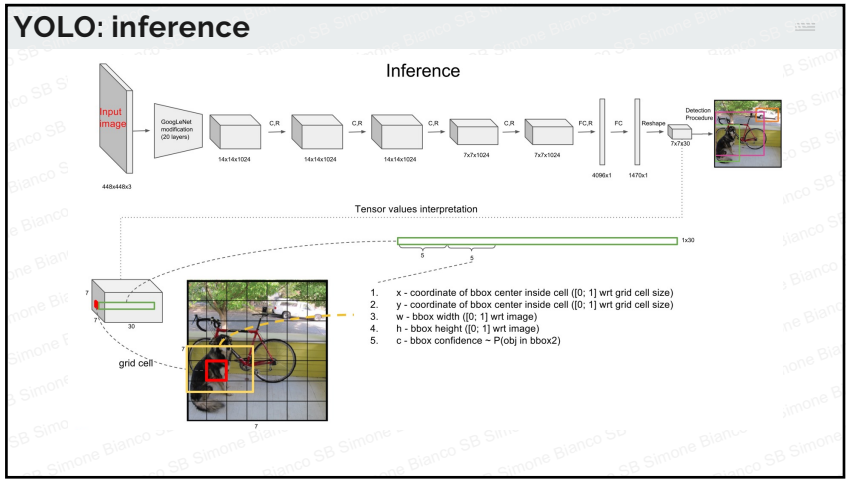
33

YOLO: inference



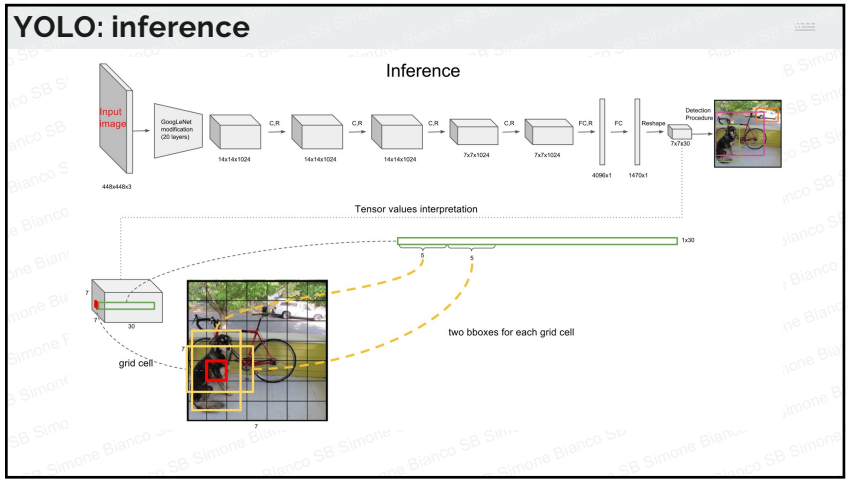
34

YOLO: inference



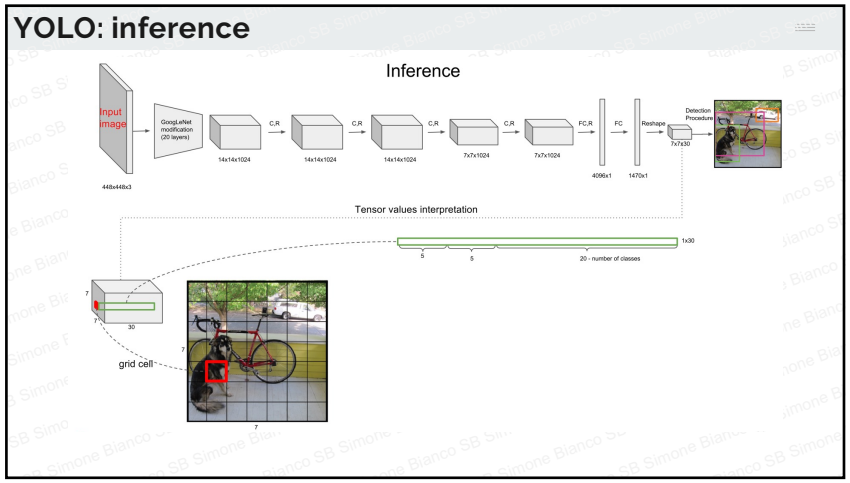
35

YOLO: inference



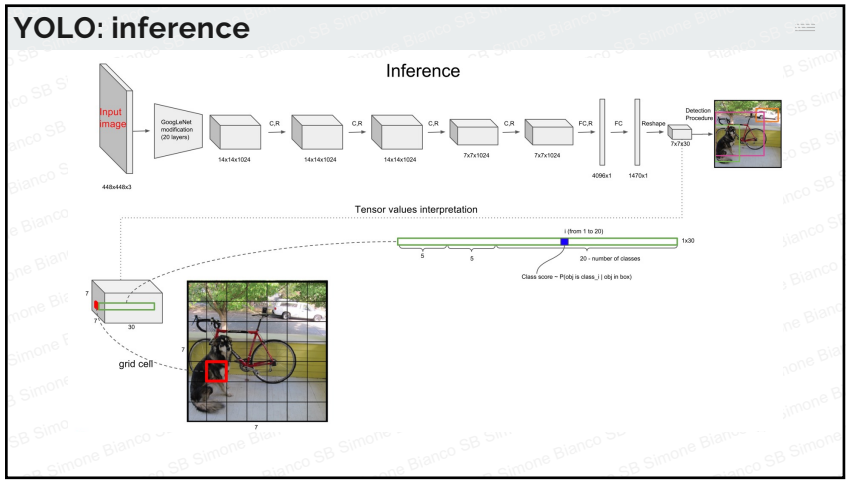
36

YOLO: inference



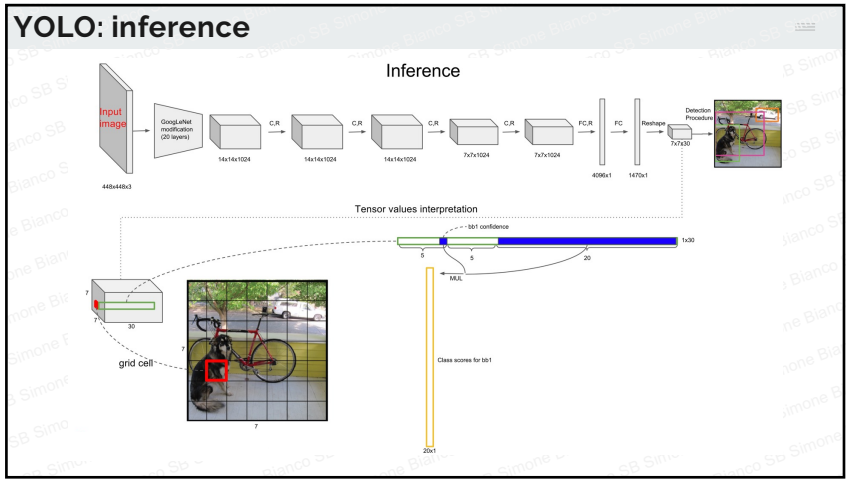
37

YOLO: inference



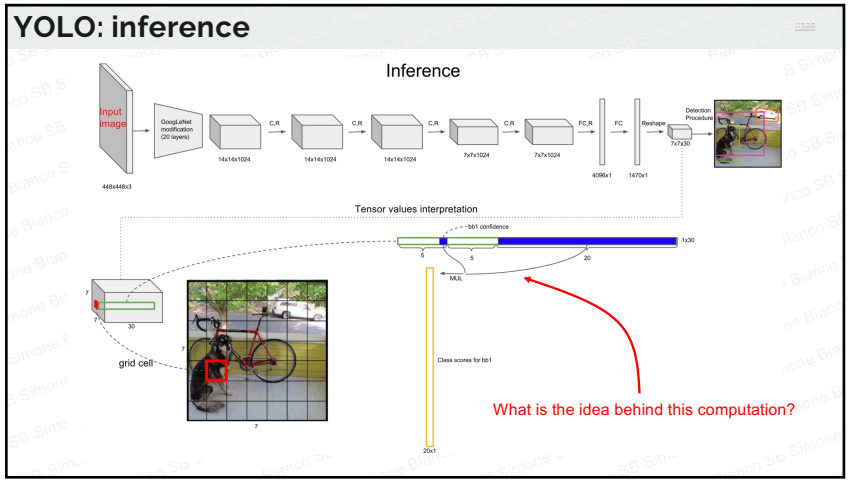
38

YOLO: inference



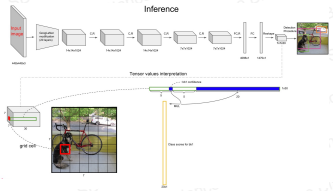
39

YOLO: inference



40

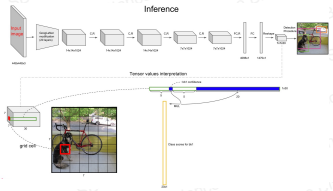
YOLO: inference



- For each BBox in each grid cell we estimate 5 values:
  [x, y, w, h, confidence]
- Then for each grid cell we also estimate C class probabilities (one for each class):

41

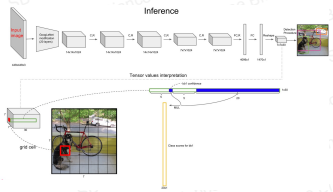
YOLO: inference



- For each BBox in each grid cell we estimate 5 values:
  [x, y, w, h, confidence]
  $\Pr(Object) \cdot IOU_{pred}^{truth}$
  BBox offset corrections for a more precise localization
- Then for each grid cell we also estimate C class probabilities (one for each class):
  $\Pr(Class_i|Object)$

42

YOLO: inference

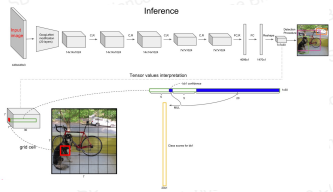


These confidence scores reflect how confident the model is that the box contains an object

- For each BBox in each grid cell we estimate 5 values:
  [x, y, w, h, confidence]
  $\Pr(Object) \cdot IOU_{pred}^{truth}$
  BBox offset corrections for a more precise localization
- Then for each grid cell we also estimate C class probabilities (one for each class):
  $\Pr(Class_i|Object)$

43

YOLO: inference

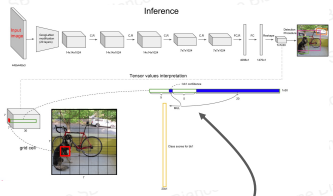


These confidence scores reflect how confident the model is that the box contains an object, and also how accurate it thinks the box is that it predicts.

- For each BBox in each grid cell we estimate 5 values:
  [x, y, w, h, confidence]
  $\Pr(Object) \cdot IOU_{pred}^{truth}$
  BBox offset corrections for a more precise localization
- Then for each grid cell we also estimate C class probabilities (one for each class):
  $\Pr(Class_i|Object)$

44

YOLO: inference

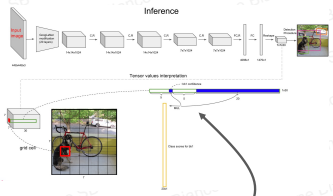


- Thus, in our multiplication, we are actually computing:

$$\Pr(Class_i|Object) \cdot \Pr(Object) \cdot IOU_{pred}^{truth} = \Pr(Class_i) \cdot IOU_{pred}^{truth}$$

45

YOLO: inference



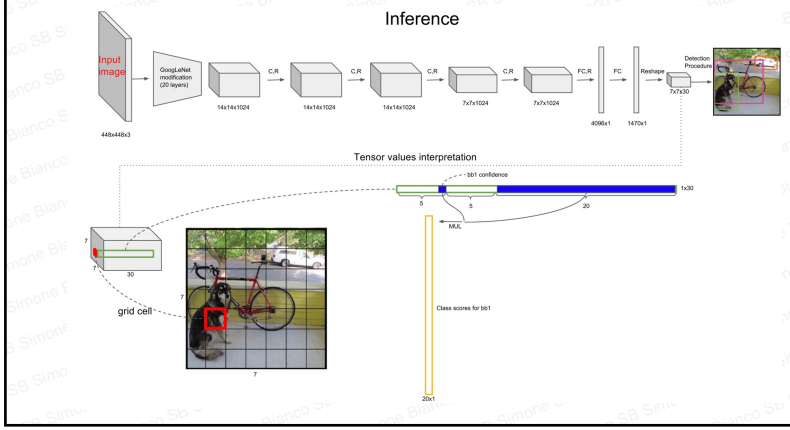
- Thus, in our multiplication, we are actually computing:

$$\Pr(Class_i|Object) \cdot \Pr(Object) \cdot IOU_{pred}^{truth} = \Pr(Class_i) \cdot IOU_{pred}^{truth}$$

Now let's go back to our computation…

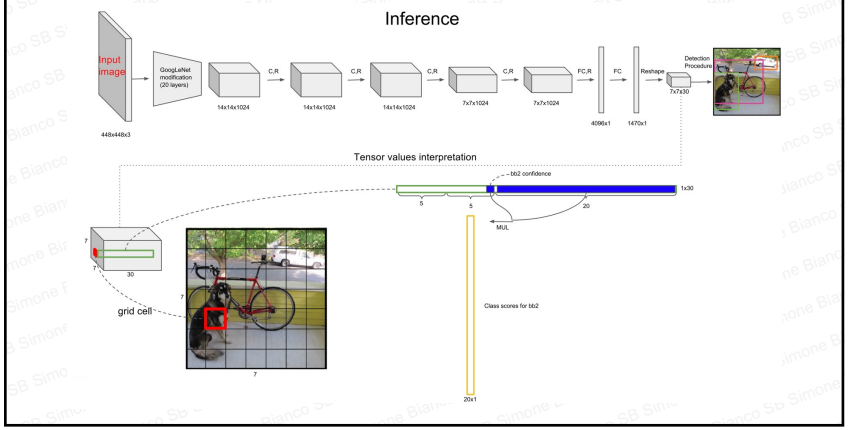
46

YOLO: inference

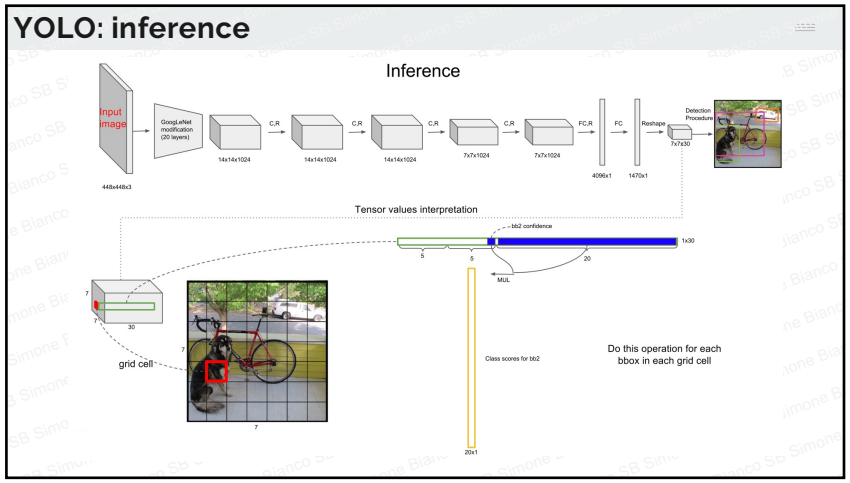


47

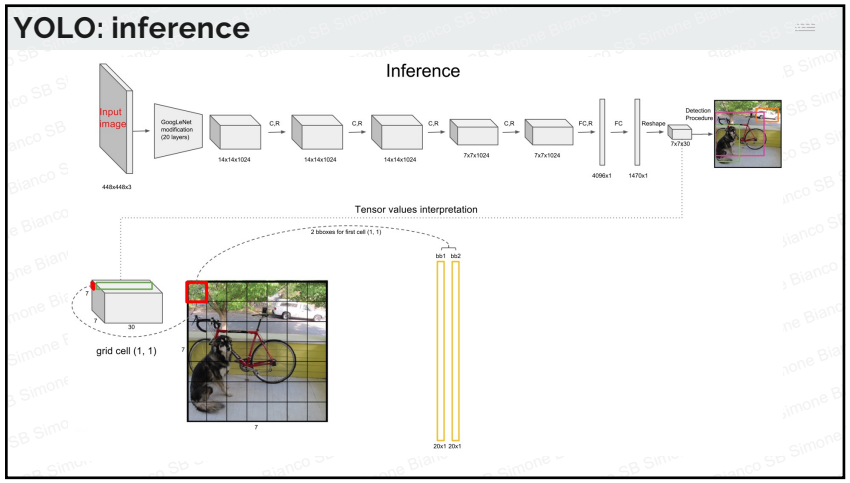
YOLO: inference



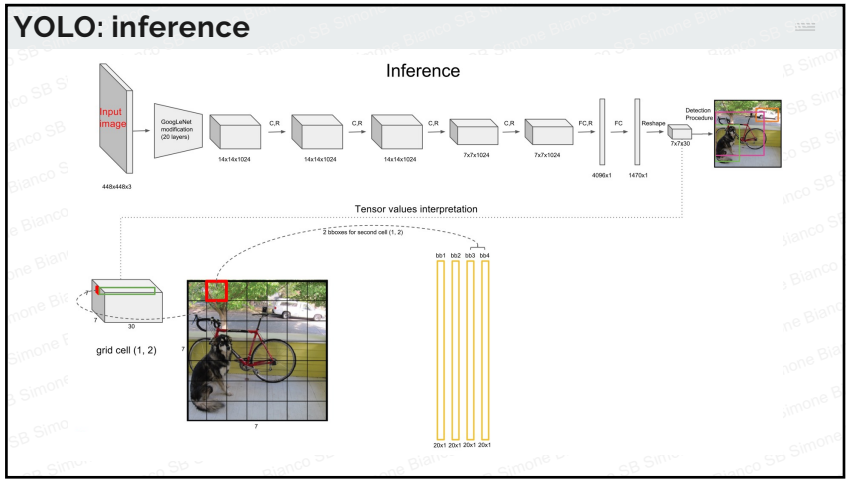
48


YOLO: inference
Inference
Tensor values interpretation
grid cell
Class scores for bb2
Do this operation for each bbox in each grid cell

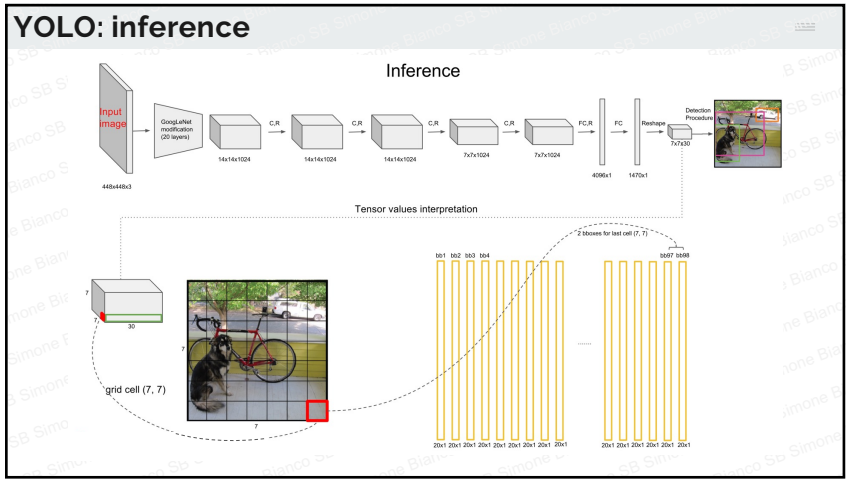
49


YOLO: inference
Inference
Tensor values interpretation
grid cell (1, 1)

50

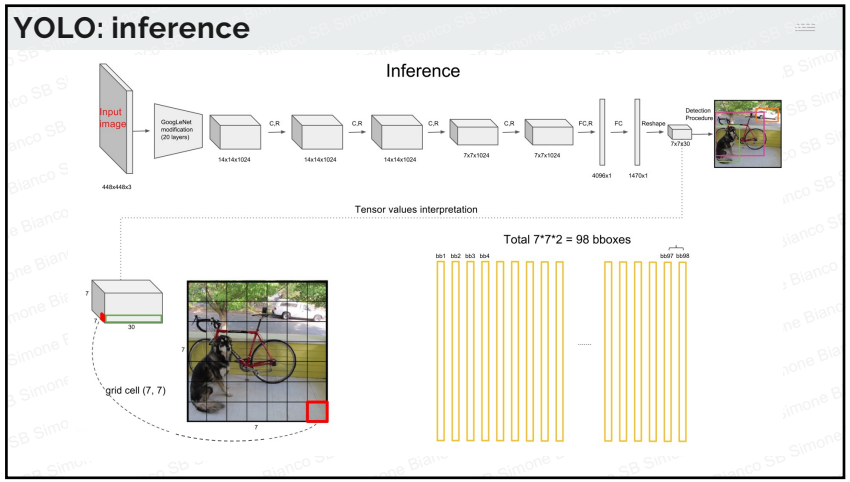

YOLO: inference
Inference
Tensor values interpretation
grid cell (1, 2)

51


YOLO: inference
Inference
Tensor values interpretation
grid cell (7, 7)

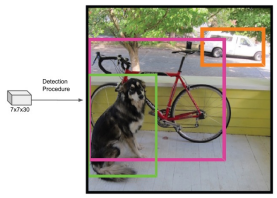
52

YOLO: inference



53

YOLO: inference

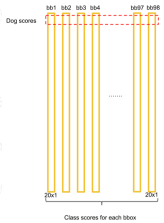


54

YOLO: inference

55

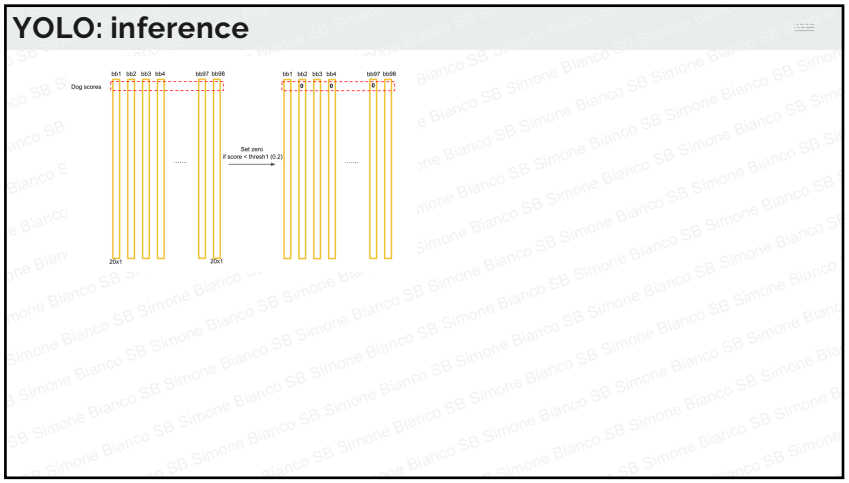
YOLO: inference



Get first class scores for each bbox

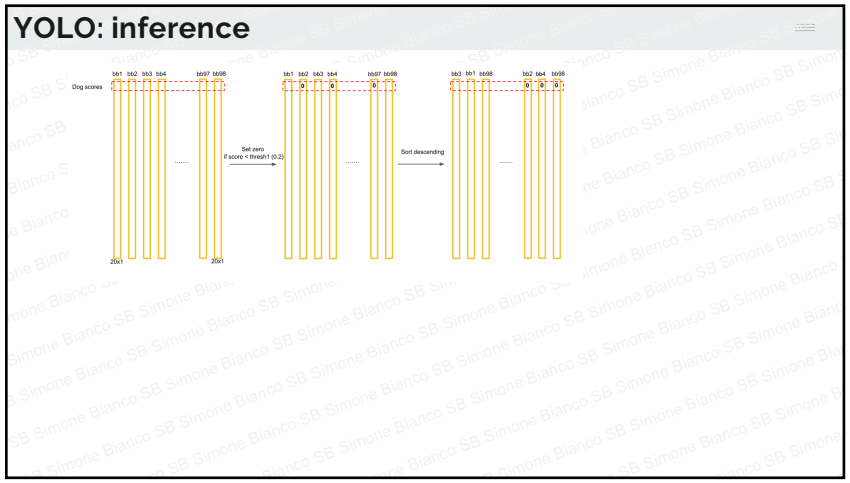
56

YOLO: inference



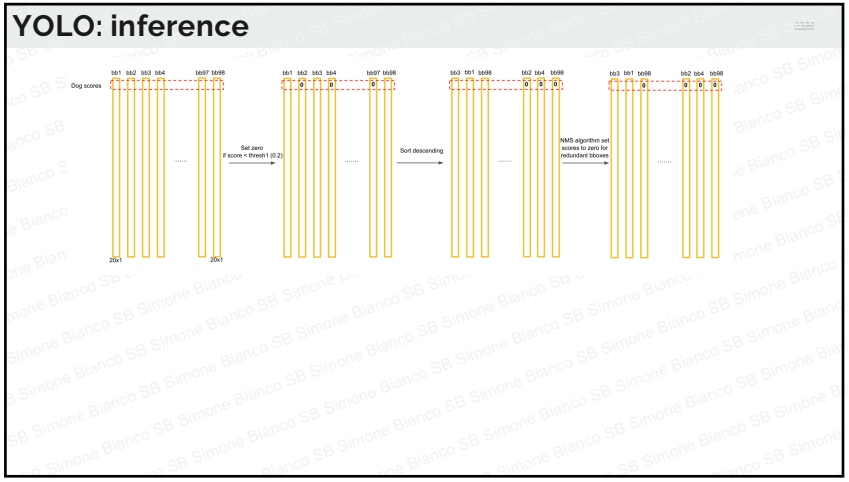
57

YOLO: inference



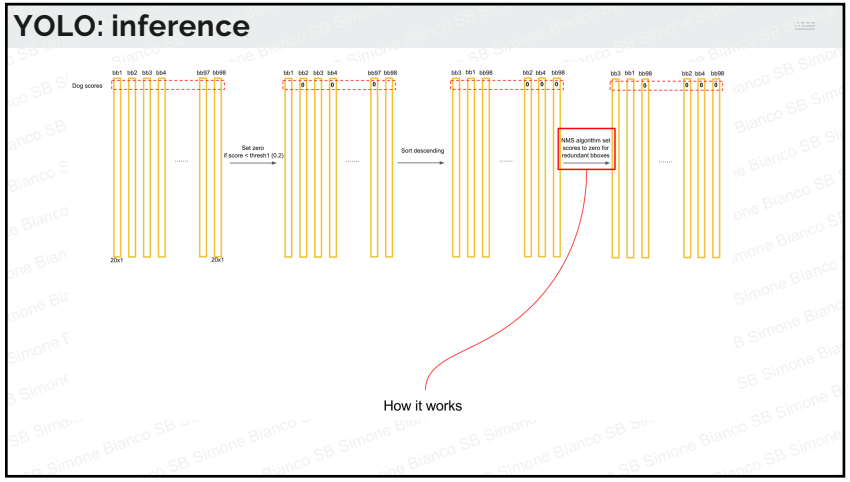
58

YOLO: inference



59

YOLO: inference



60

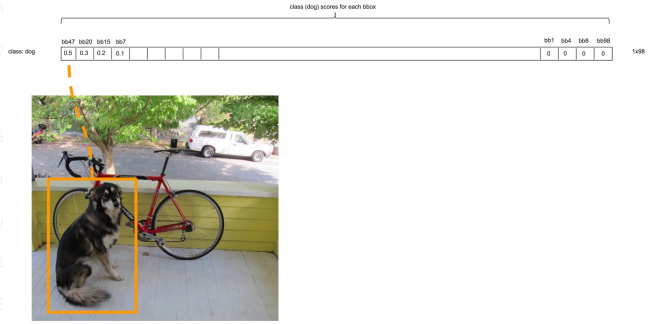
YOLO: inference

Non-Maximum Suppression: intuition

61

YOLO: inference

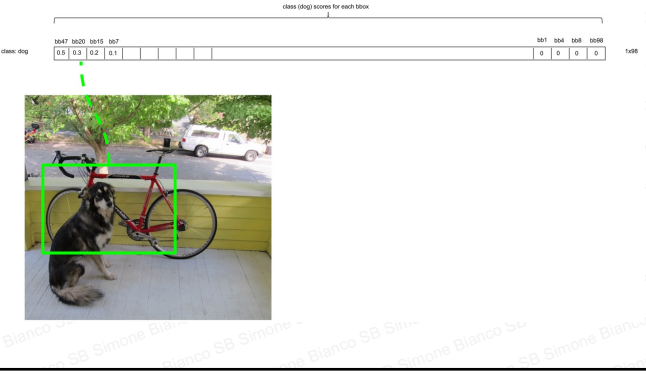
Non-Maximum Suppression: intuition



62

YOLO: inference

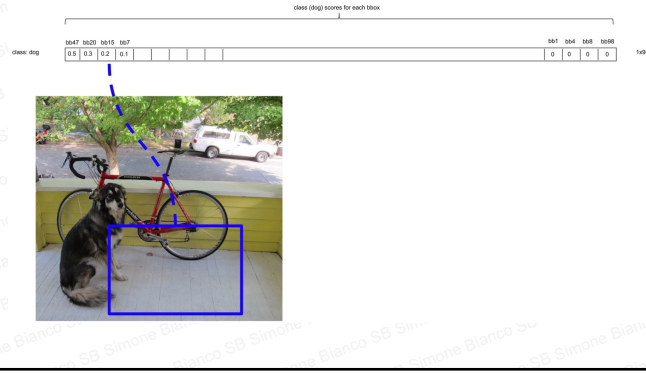
Non-Maximum Suppression: intuition



63

YOLO: inference

Non-Maximum Suppression: intuition



64

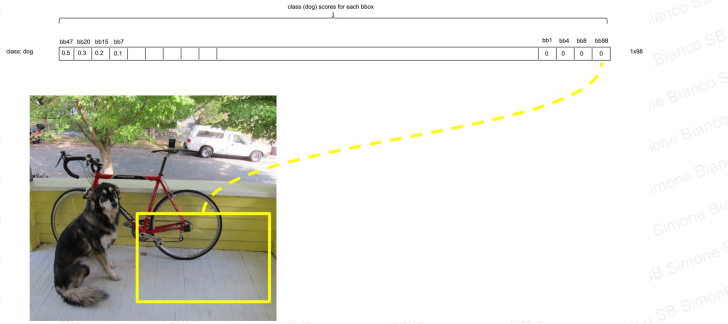
YOLO: inference

Non-Maximum Suppression: intuition

65

YOLO: inference

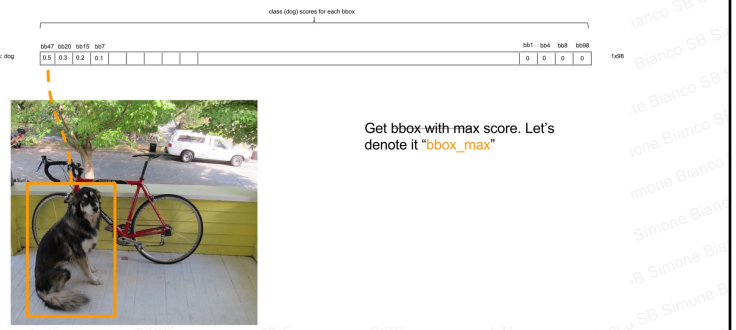
Non-Maximum Suppression: intuition



66

YOLO: inference

Non-Maximum Suppression: intuition

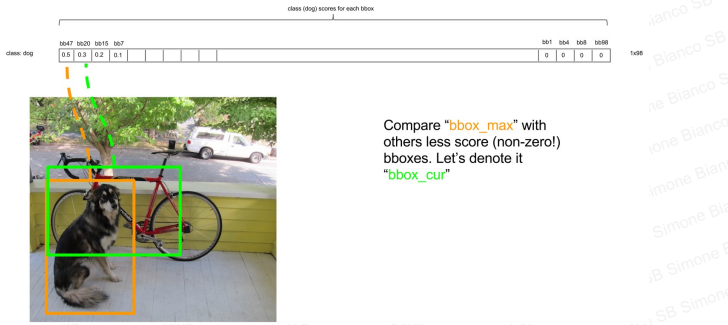


Get bbox with max score. Let's denote it "bbox_max"

67

YOLO: inference

Non-Maximum Suppression: intuition



Compare "bbox_max" with others less score (non-zero!) bboxes. Let's denote it "bbox_cur"

68

YOLO: inference

Non-Maximum Suppression: intuition

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

69

YOLO: inference

Non-Maximum Suppression: intuition

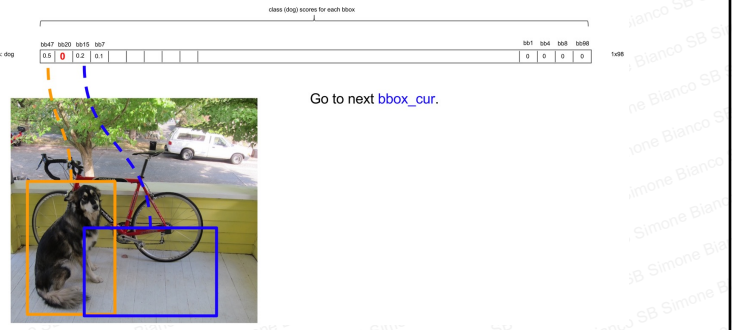
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: set to 0.

70

YOLO: inference

Non-Maximum Suppression: intuition

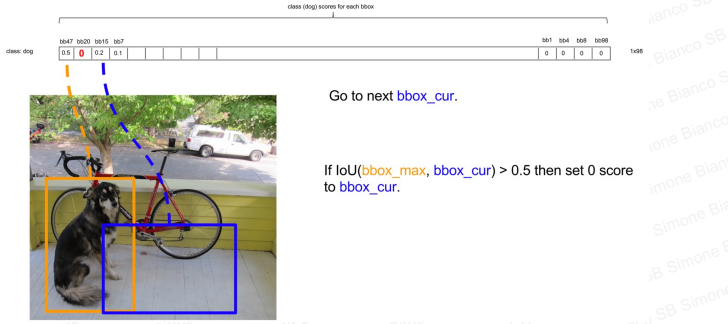


Go to next bbox_cur.

71

YOLO: inference

Non-Maximum Suppression: intuition



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

72

YOLO: inference

Non-Maximum Suppression: intuition

Go to next bbox_cur.

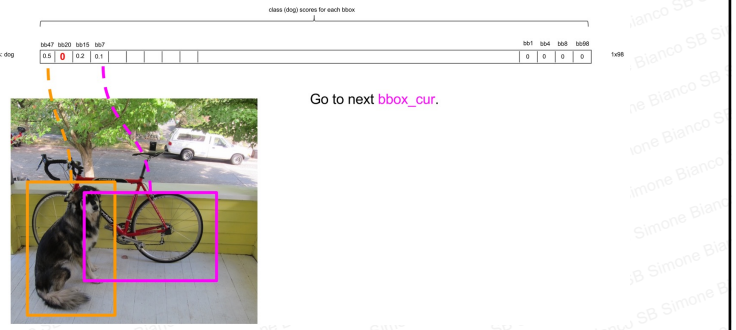
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

73

YOLO: inference

Non-Maximum Suppression: intuition

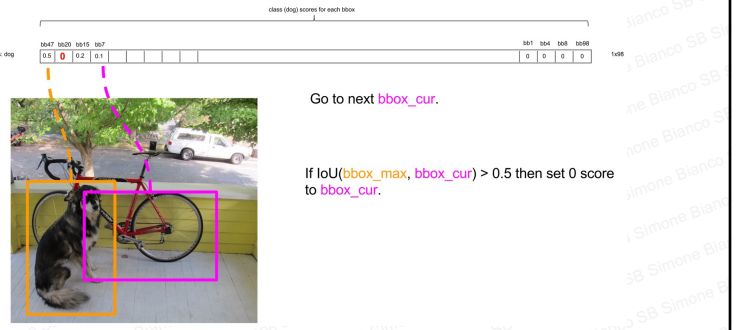


Go to next bbox_cur.

74

YOLO: inference

Non-Maximum Suppression: intuition



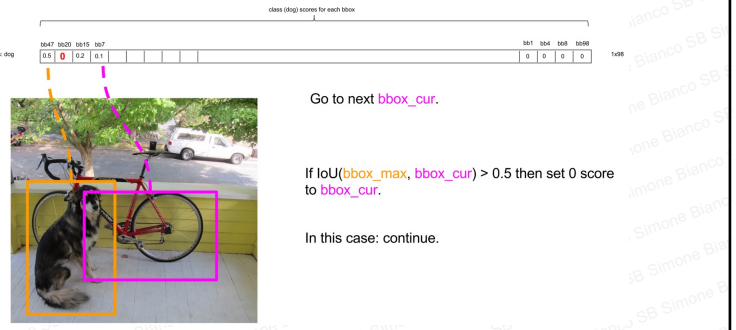
Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

75

YOLO: inference

Non-Maximum Suppression: intuition



Go to next bbox_cur.

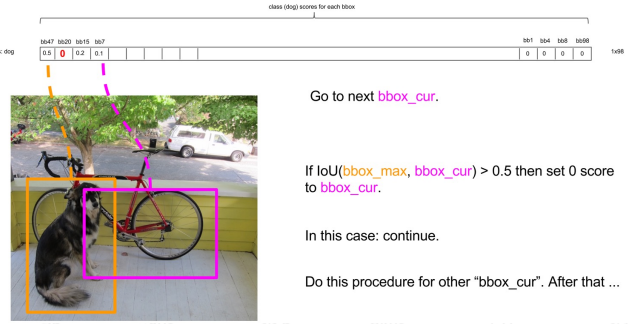
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

76

YOLO: inference

Non-Maximum Suppression: intuition



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

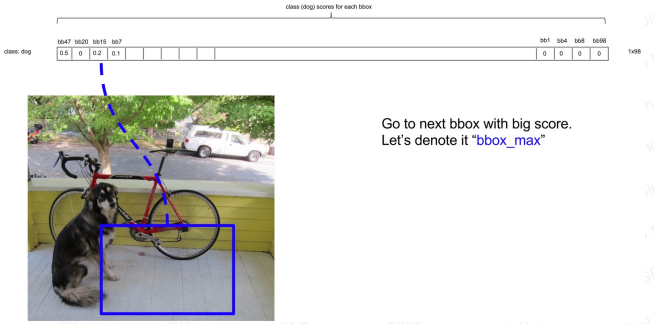
In this case: continue.

Do this procedure for other “bbox_cur”. After that ...

77

YOLO: inference

Non-Maximum Suppression: intuition

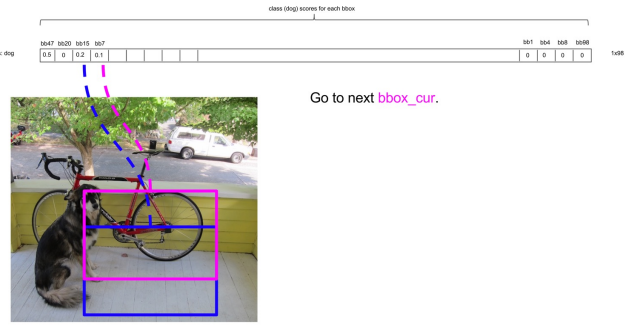


Go to next bbox with big score.
Let’s denote it “bbox_max”

78

YOLO: inference

Non-Maximum Suppression: intuition

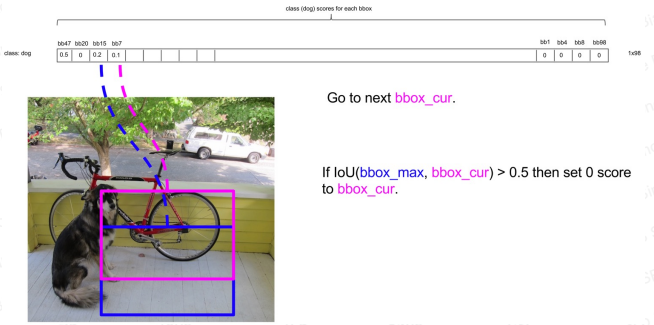


Go to next bbox_cur.

79

YOLO: inference

Non-Maximum Suppression: intuition



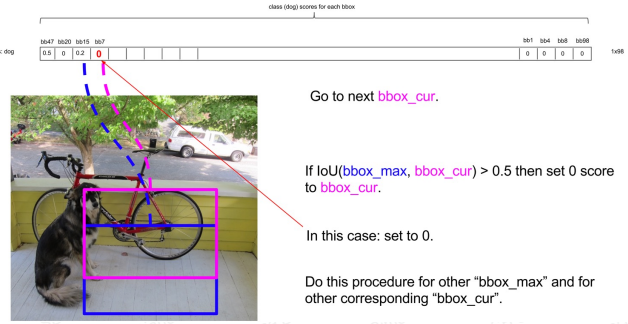
Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

80

YOLO: inference

Non-Maximum Suppression: intuition



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: set to 0.

Do this procedure for other “bbox_max” and for other corresponding “bbox_cur”.

81

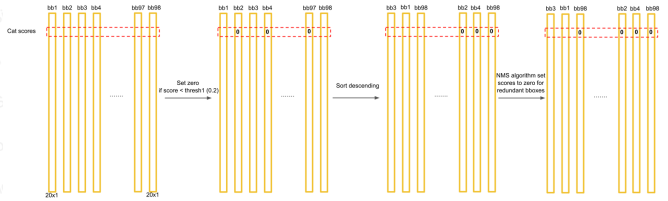
YOLO: inference

Non-Maximum Suppression: intuition

After comparison almost all pairs of bboxes the only two bboxes left with non-zero class score value.

82

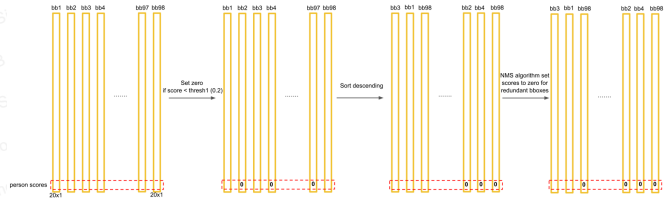
YOLO: inference



Do this procedure for next class

83

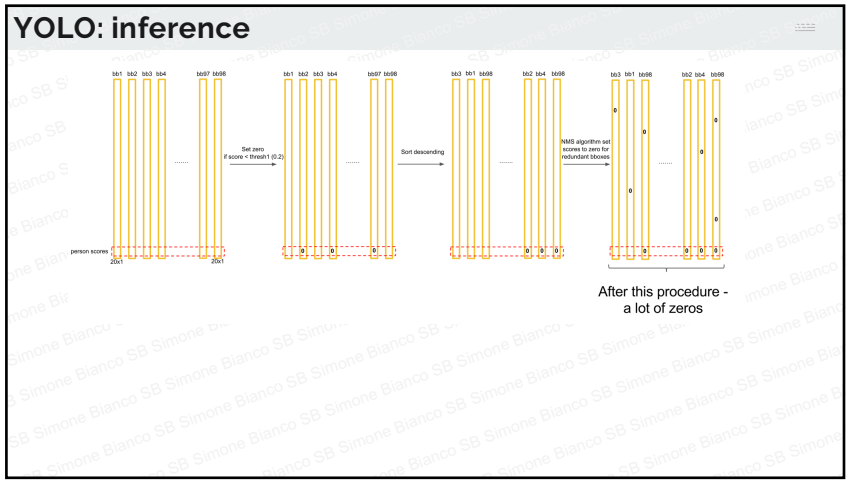
YOLO: inference



Do this procedure for all classes

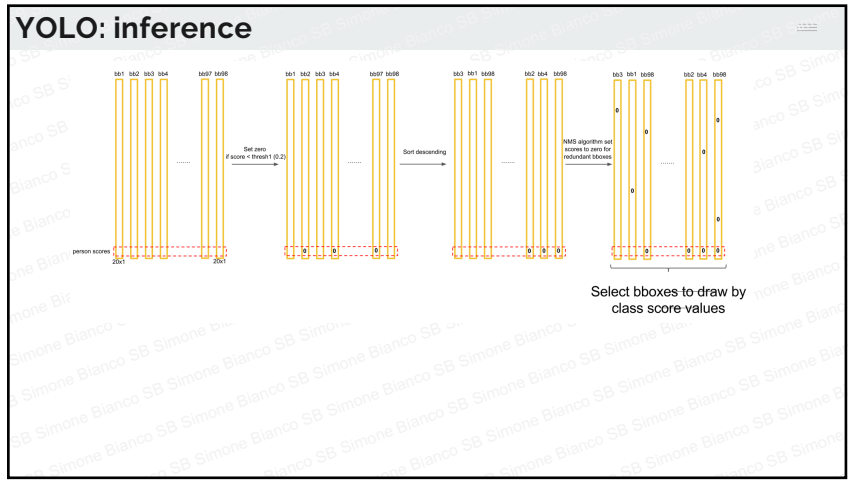
84

YOLO: inference



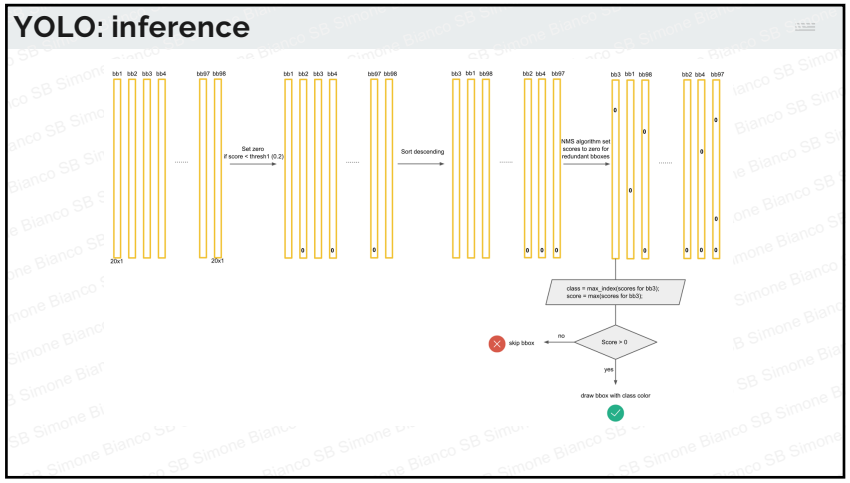
85

YOLO: inference



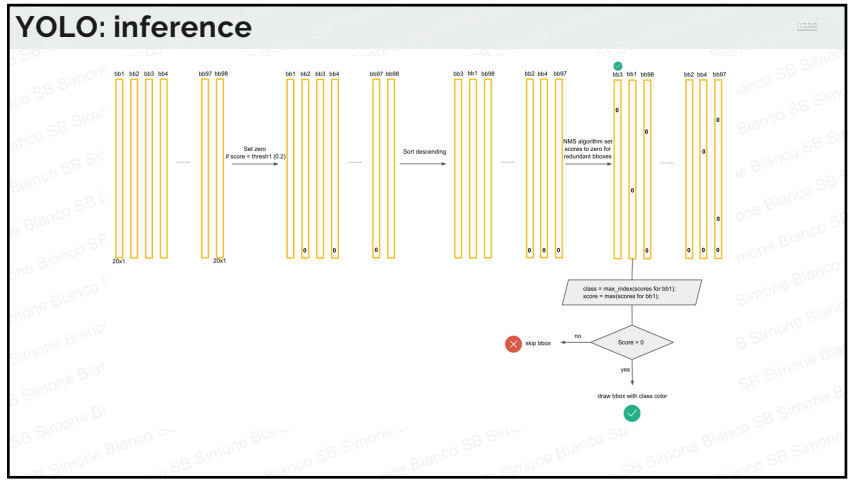
86

YOLO: inference



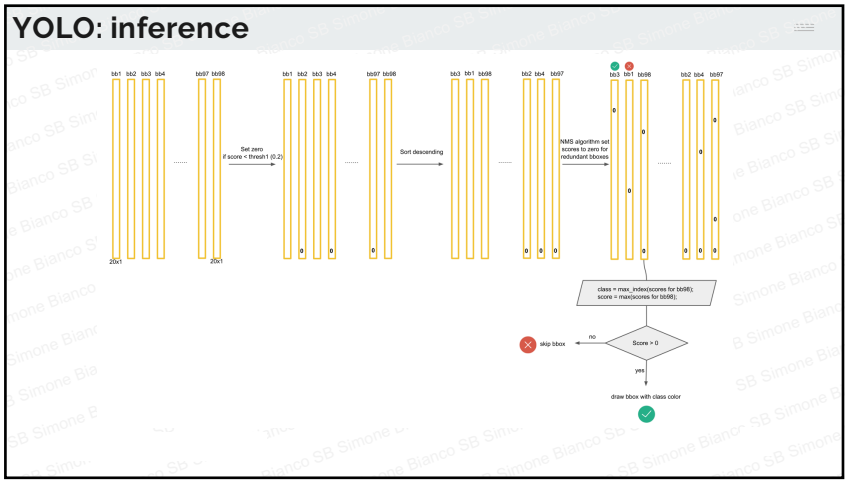
87

YOLO: inference



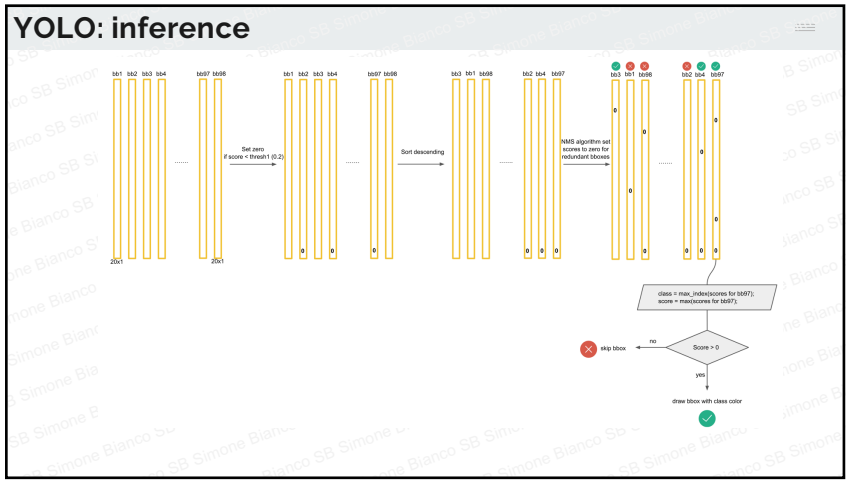
88

YOLO: inference



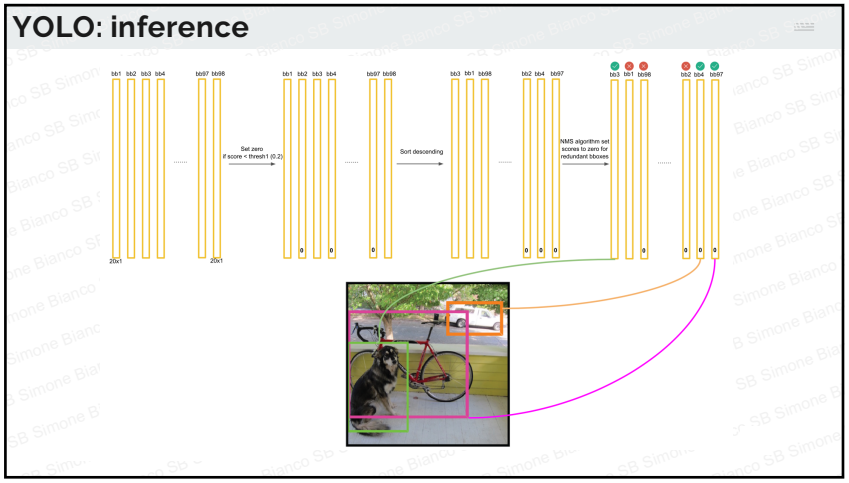
89

YOLO: inference



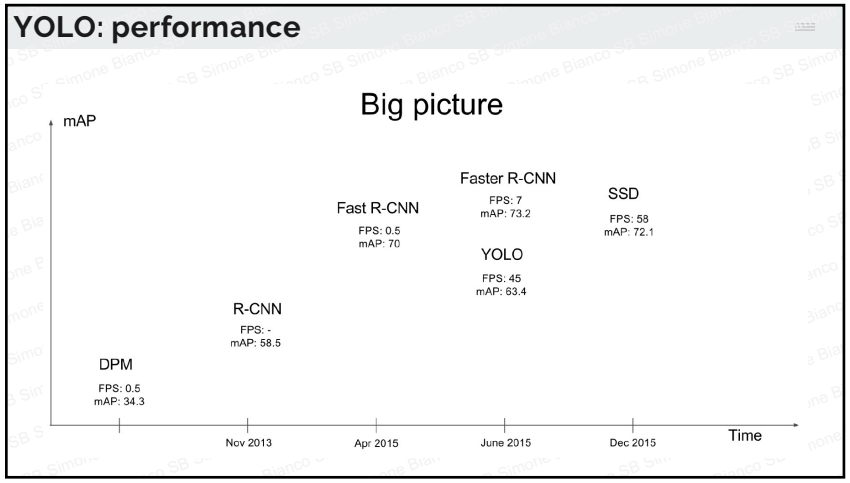
90

YOLO: inference



91

YOLO: performance



92

YOLO: comparison to other real-time approaches

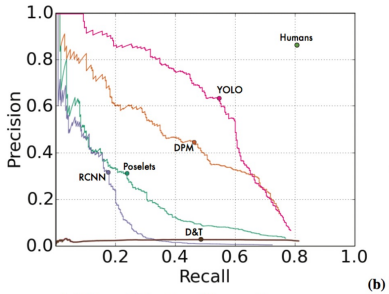
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

93

YOLO generalizability

Person detection in artwork



(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Figure 5: Generalization results on Picasso and People-Art datasets.

94

YOLO generalizability

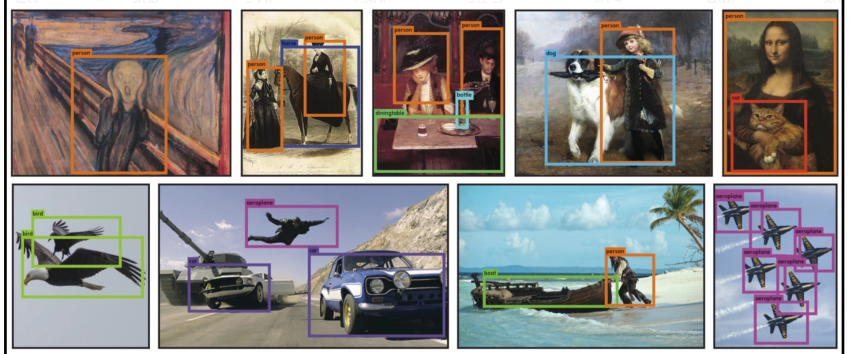


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

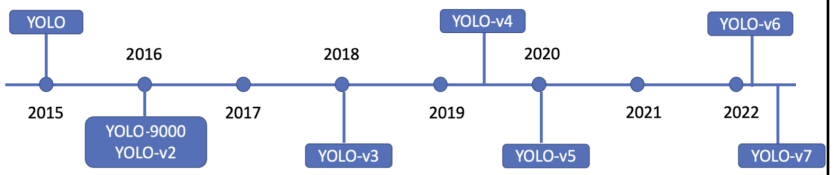
95

AA 2022/2023

YOLO timeline

96

YOLO timeline



98

YOLO-v2

99

YOLO-v2

- YOLO-v2 (a.k.a. YOLO9000) [v2], was introduced in 2016 as an improvement over the original YOLO algorithm.
- It was designed to be faster and more accurate than YOLO and to be able to detect a wider range of object classes.
- This updated version also uses a different CNN backbone called Darknet-19, a variant of the VGGNet architecture with simple progressive convolution and pooling layers.
- One of the main improvements in YOLO-v2 is the use of anchor boxes. Anchor boxes are a set of predefined bounding boxes of different aspect ratios. When predicting bounding boxes, YOLO-v2 uses a combination of the anchor boxes and the predicted offsets to determine the final bounding box. This allows the algorithm to handle a wider range of aspect ratios.

[v2] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE CVPR (pp. 7263-7271).

100

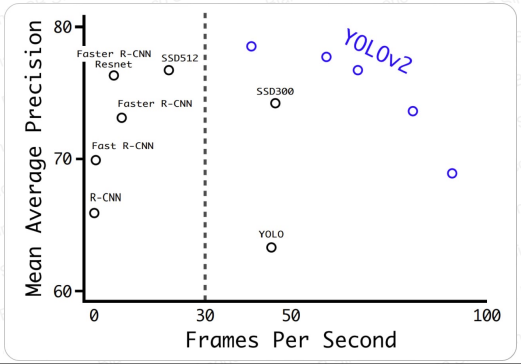
YOLO-v2

- Another improvement in YOLO-v2 is the use of batch normalization, which helps to improve the accuracy and stability of the model.
- YOLO-v2 also uses a multi-scale training strategy, which involves training the model on images at multiple scales and then averaging the predictions. This helps to improve the detection performance of small objects.
- YOLO-v2 also introduces a new loss function better suited to object detection tasks. The loss function is based on the sum of the squared errors between the predicted and ground truth bounding boxes and class probabilities.

101

YOLO-v2

- The results obtained by YOLO-v2 compared to the original version and other contemporary models are shown below.



102

YOLO-v2

103

YOLO-v3

104

YOLO-v3

- YOLO-v3 [v3] is the third version of the YOLO object detection algorithm. It was introduced in 2018 as an improvement over YOLO-v2, aiming to increase the accuracy and speed of the algorithm.
- One of the main improvements in YOLO v3 is the use of a new CNN architecture called Darknet-53. Darknet-53 is a variant of the ResNet architecture and is designed specifically for object detection tasks. It has 53 convolutional layers, and it is able to achieve state-of-the-art results on various object detection benchmarks.
- Another improvement in YOLO-v3 are anchor boxes with different scales and aspect ratios: In YOLO-v2, the anchor boxes were all the same size, which limited the ability of the algorithm to detect objects of different sizes and shapes. In YOLO-v3 the anchor boxes are scaled, and aspect ratios are varied to better match the size and shape of the objects being detected.

[v3] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

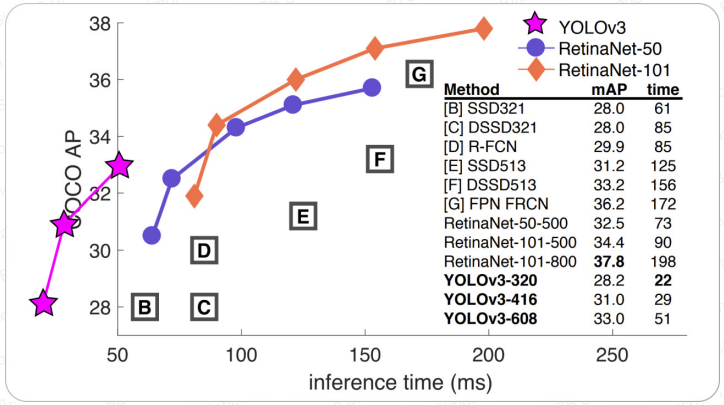
105

YOLO-v3

- YOLO-v3 also introduces the concept of "feature pyramid networks" (FPN). FPNs are a CNN architecture used to detect objects at multiple scales. They construct a pyramid of feature maps, with each level of the pyramid being used to detect objects at a different scale. This helps to improve the detection performance on small objects, as the model is able to see the objects at multiple scales.
- In addition to these improvements, YOLO-v3 can handle a wider range of object sizes and aspect ratios. It is also more accurate and stable than the previous versions of YOLO.

106

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

107

YOLO-v4

108

YOLO-v4

Note: Joseph Redmond, the original creator of YOLO, has left the AI community a few years before, so YOLO-v4 and other versions past that are not his official work. Some of them are maintained by co-authors, but none of the releases past YOLO-v3 is considered the "official" YOLO.

- YOLO-v4 [v4] is the fourth version of the YOLO object detection algorithm introduced in 2020 by Bochkovskiy et al. as an improvement over YOLO-v3.
- The primary improvement in YOLO-v4 over YOLO-v3 is the use of a new CNN architecture called CSPNet ("Cross Stage Partial Network")
- CSPNet is a variant of the ResNet architecture designed specifically for object detection tasks. It has a relatively shallow structure, with only 54 convolutional layers. However, it can achieve state-of-the-art results on various object detection benchmarks.

[v4] Bochkovskiy, A., et al. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934

109

YOLO-v4

- Both YOLO-v3 and YOLO-v4 use anchor boxes with different scales and aspect ratios to better match the size and shape of the detected objects.
- YOLO-v4 introduces a new method for generating the anchor boxes, called "k-means clustering": it involves using a clustering algorithm to group the ground truth bounding boxes into clusters and then using the centroids of the clusters as the anchor boxes. This allows the anchor boxes to be more closely aligned with the detected objects' size and shape.
- While both YOLO-v3 and YOLO-v4 use a similar loss function for training the model, YOLO-v4 introduces a new term called "GHM loss." It's a variant of the focal loss function and is designed to improve the model's performance on imbalanced datasets.
- YOLO-v4 also improves the architecture of the FPNs used in YOLO-v3.

110

YOLO-v4

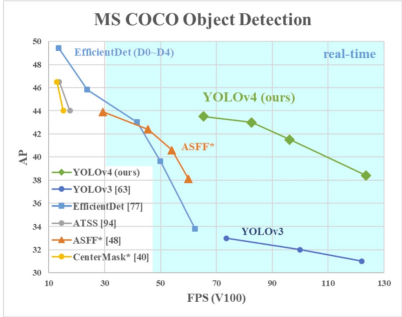


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

111

AA 2022/2023

YOLO-v5

112

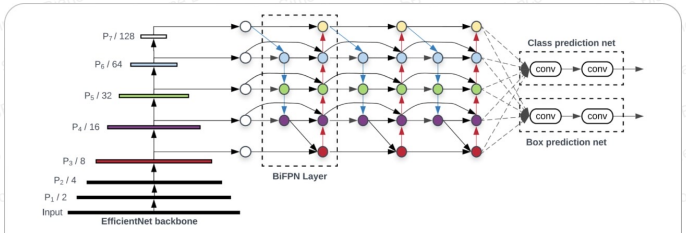
YOLO-v5

- YOLO-v5 was introduced in 2020 by the same team that developed the original YOLO algorithm as an open-source project. YOLO-v5 builds upon the success of previous versions and adds several new features and improvements.
- Unlike YOLO, YOLO-v5 uses a more complex architecture called EfficientDet, based on the EfficientNet network architecture.
- Using a more complex architecture in YOLO v5 allows it to achieve higher accuracy and better generalization to a wider range of object categories.

[v5] Jocher, G., et al (2020). ultralytics/yolov5: v3. 0. Zenodo.

113

YOLO-v5



EfficientDet architecture – It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

114

YOLO-v5

- Another difference between YOLO and YOLO-v5 is the training data used to learn the object detection model:
  - YOLO was trained on the PASCAL VOC dataset, which consists of 20 object categories.
  - YOLO-v5, on the other hand, was trained on a larger and more diverse dataset called D5, which includes a total of 600 object categories.
- YOLO-v5 uses a new method for generating the anchor boxes, called "dynamic anchor boxes": It involves using a clustering algorithm to group the ground truth bounding boxes into clusters and then using the centroids of the clusters as the anchor boxes. This allows the anchor boxes to be more closely aligned with the detected objects' size and shape.

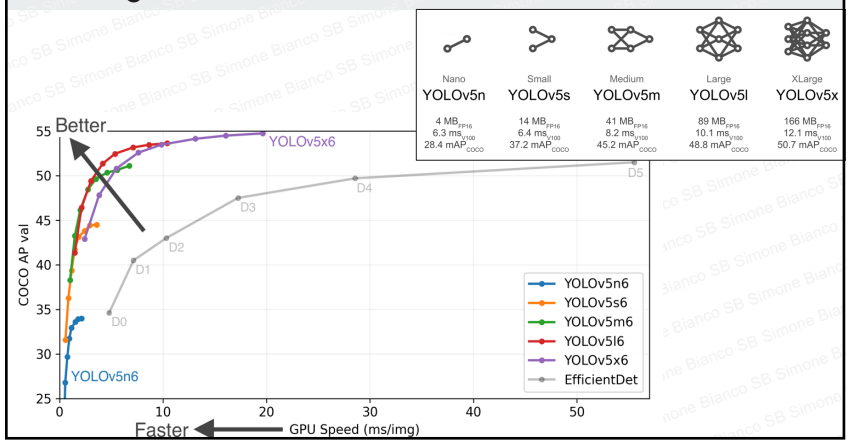
115

YOLO-v5

- YOLO-v5 also introduces the concept of "spatial pyramid pooling" (SPP), a type of pooling layer used to reduce the spatial resolution of the feature maps.
- SPP is used to improve the detection performance on small objects, as it allows the model to see the objects at multiple scales. YOLO-v4 also uses SPP, but YOLO-v5 includes several improvements to the SPP architecture that allow it to achieve better results.
- YOLO-v4 and YOLO-v5 use a similar loss function to train the model. However, YOLO-v5 introduces a new term called "CIoU loss," which is a variant of the IoU loss function designed to improve the model's performance on imbalanced datasets.

116

YOLO-v5



117

YOLO-v6

118

YOLO-v6

- YOLO-v6 was proposed in 2022 by Li et al [v6]. as an improvement over previous versions.
- One of the main differences between YOLO-v5 and YOLO-v6 is the CNN architecture used. YOLO v6 uses a variant of the EfficientNet architecture called EfficientNet-L2. It's a more efficient architecture than EfficientDet used in YOLO-v5, with fewer parameters and a higher computational efficiency. It can achieve state-of-the-art results on various object detection benchmarks.

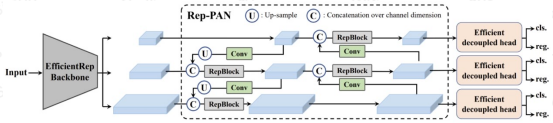


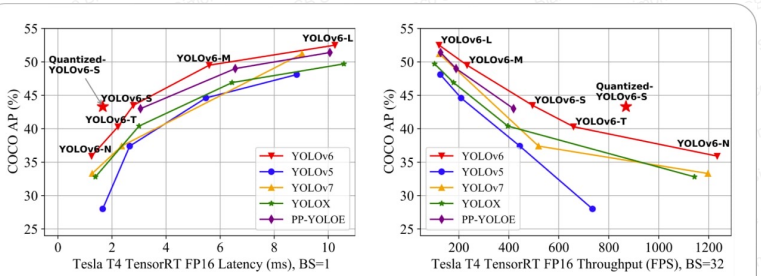
Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

[v6] Li, C., et al (2022). YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.

119

YOLO-v6

- YOLO-v6 also introduces a new method for generating the anchor boxes, called "dense anchor boxes."



Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7 except that the quantized model is with TensorRT 8.

120

YOLO-v7

121

YOLO-v7

- YOLO-v7 [v7], the latest version of YOLO, has several improvements over the previous versions. One of the main improvements is the use of anchor boxes.
- Anchor boxes are a set of predefined boxes with different aspect ratios that are used to detect objects of different shapes. YOLO-v7 uses nine anchor boxes, which allows it to detect a wider range of object shapes and sizes compared to previous versions, thus helping to reduce the number of false positives.
- A key improvement in YOLO-v7 is the use of a new loss function called “focal loss.” Previous versions of YOLO used a standard cross-entropy loss function, which is known to be less effective at detecting small objects. Focal loss battles this issue by down-weighting the loss for well-classified examples and focusing on the hard examples—the objects that are hard to detect.

[v7] Wang, C. et al. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696.

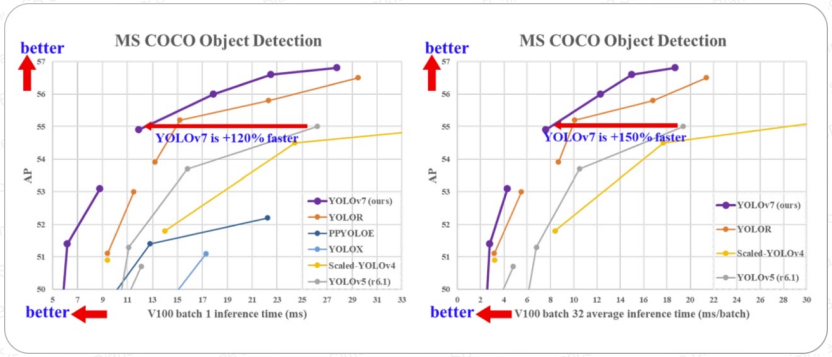
122

YOLO-v7

- YOLO-v7 also has a higher resolution than the previous versions. It processes images at a resolution of 608 by 608 pixels, which is higher than the 416 by 416 resolution used in YOLO-v3. This higher resolution allows YOLO-v7 to detect smaller objects and to have a higher accuracy overall.
- One of the main advantages of YOLO-v7 is its speed. It can process images at a rate of 155 frames per second, much faster than other state-of-the-art object detection algorithms.
- Even the original baseline YOLO model was capable of processing at a maximum rate of 45 frames per second. This makes it suitable for sensitive real-time applications such as surveillance and self-driving cars, where higher processing speeds are crucial.

123

YOLO-v7



124

YOLO-v7

- Regarding accuracy, YOLO-v7 performs well compared to other object detection algorithms. It achieves an average precision of 37.2% at an IoU (intersection over union) threshold of 0.5 on the COCO dataset, which is comparable to other state-of-the-art object detection algorithms.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-45%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

125

YOLO-v7

- However, it should be noted that YOLO-v7 is less accurate than two-stage detectors such as Faster R-CNN and Mask R-CNN, which tend to achieve higher average precision on the COCO dataset but also require longer inference times.

126