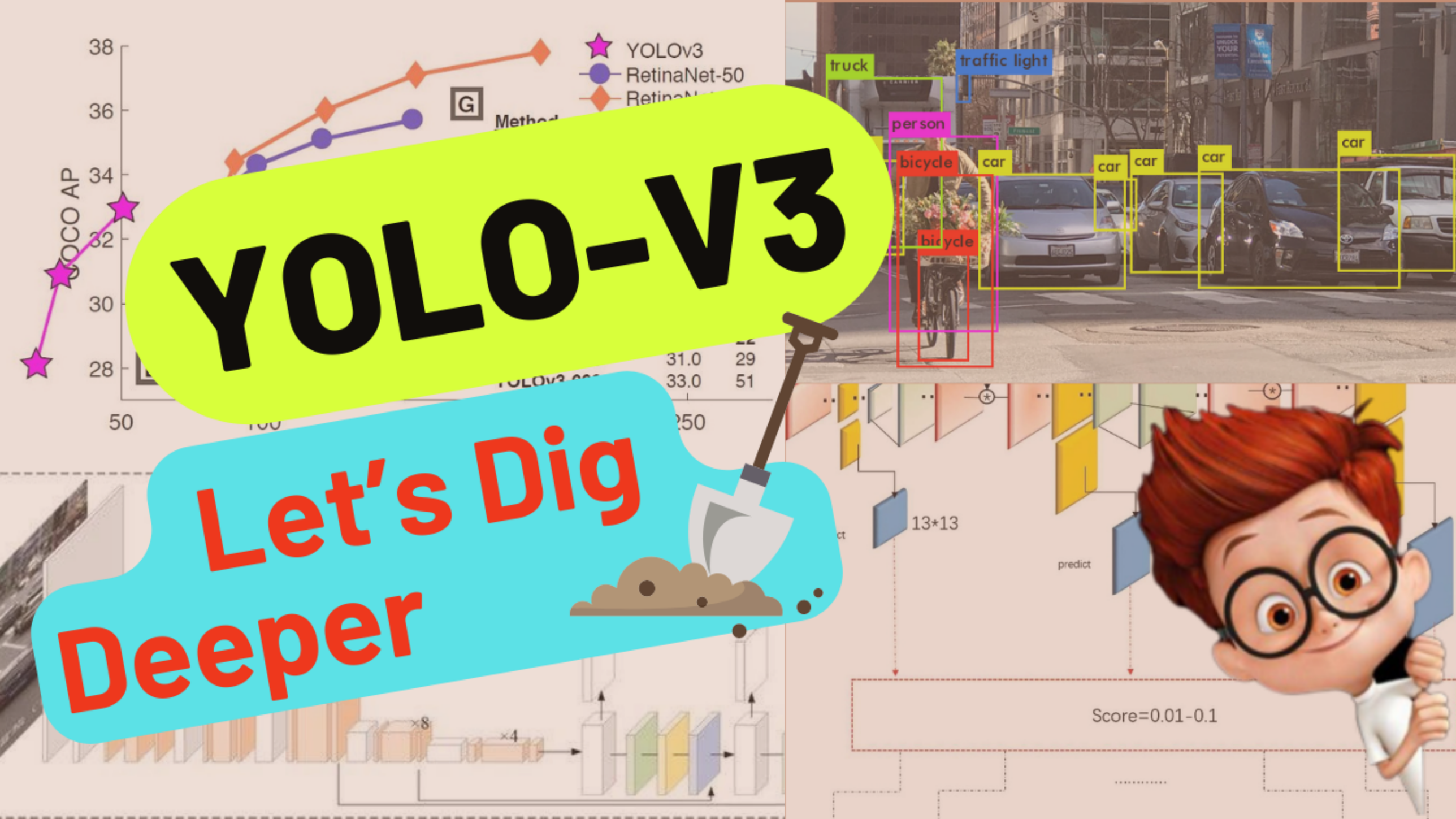

YOLO-V3
Let's Dig Deeper
YOLOv3
RetinaNet-50
COCO AP
38
36
34
32
30
28
50
31.0
29
33.0
51
truck
traffic light
person
bicycle
car
car
car
car
bicycle
13*13
predict
Score=0.01-0.1
×8
×4

Accuracy - COCO

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

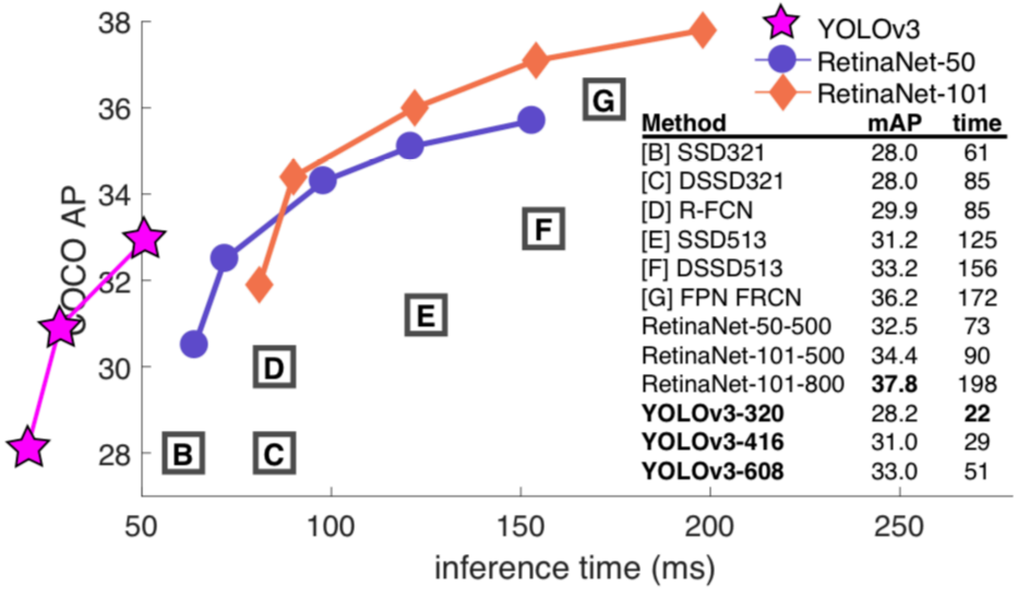
YOLOv3: An Incremental Improvement

Joseph Redmon Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

1. Introduction



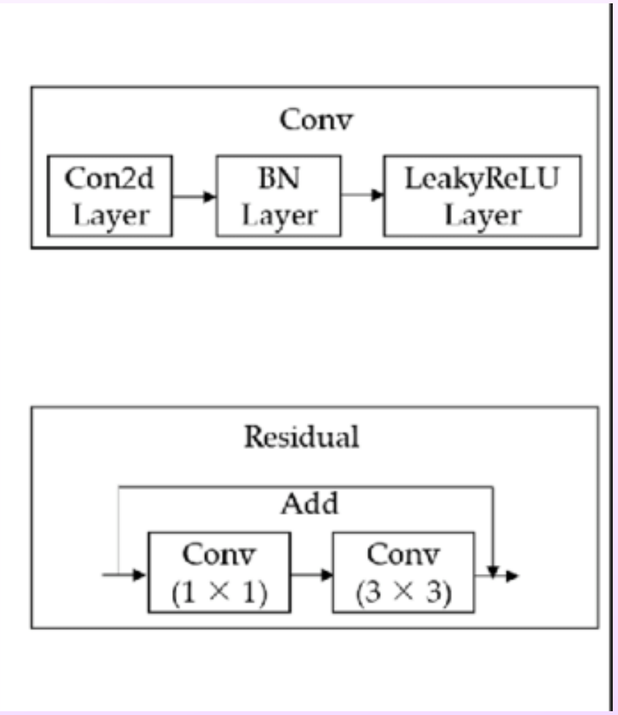
Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

Darknet-19

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Darknet-53



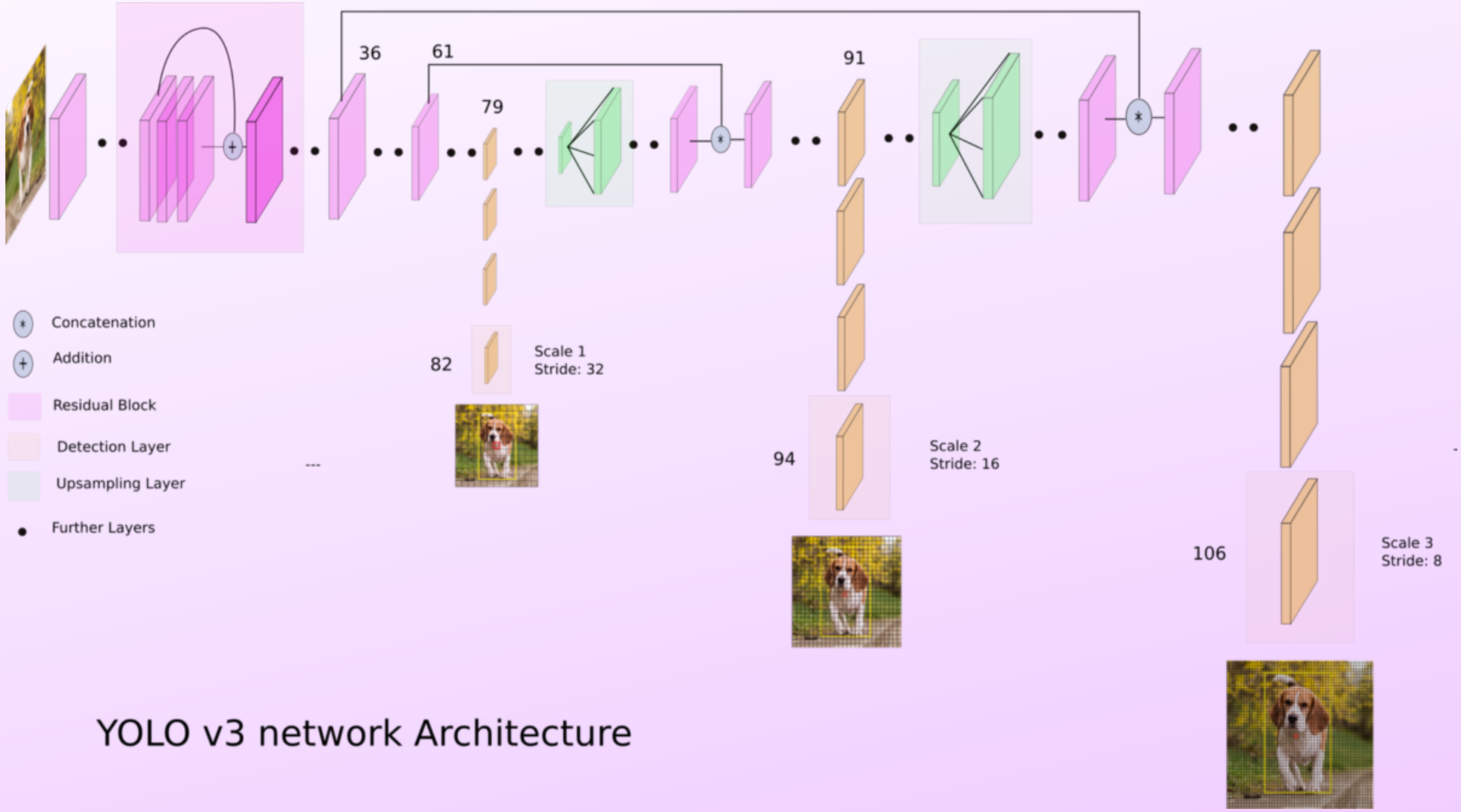
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Darknet-53

This new network is much more powerful than Darknet-19 but still more efficient than ResNet-101 or ResNet-152. Here are some ImageNet results:

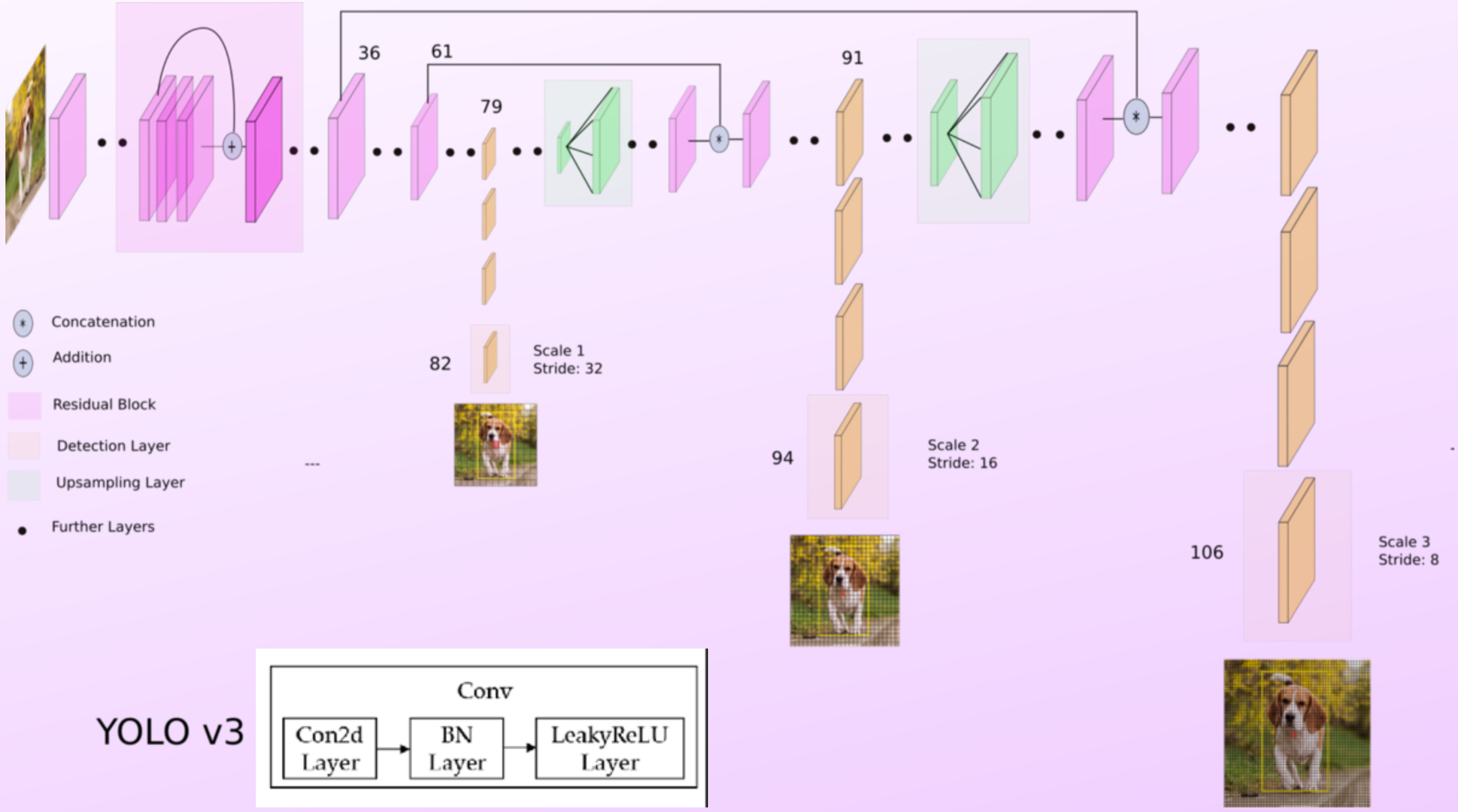
Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

YoloV3 Architecture

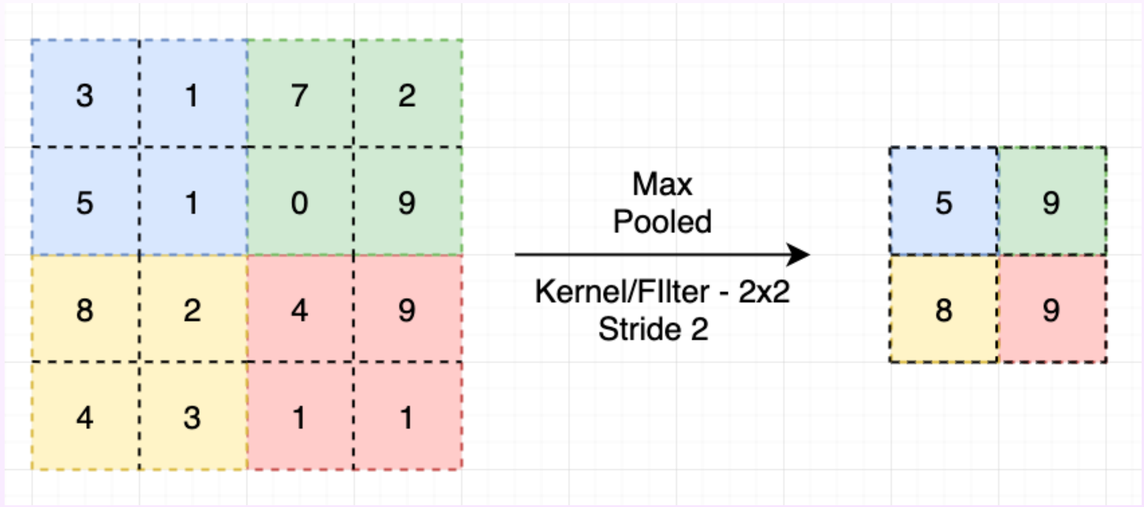


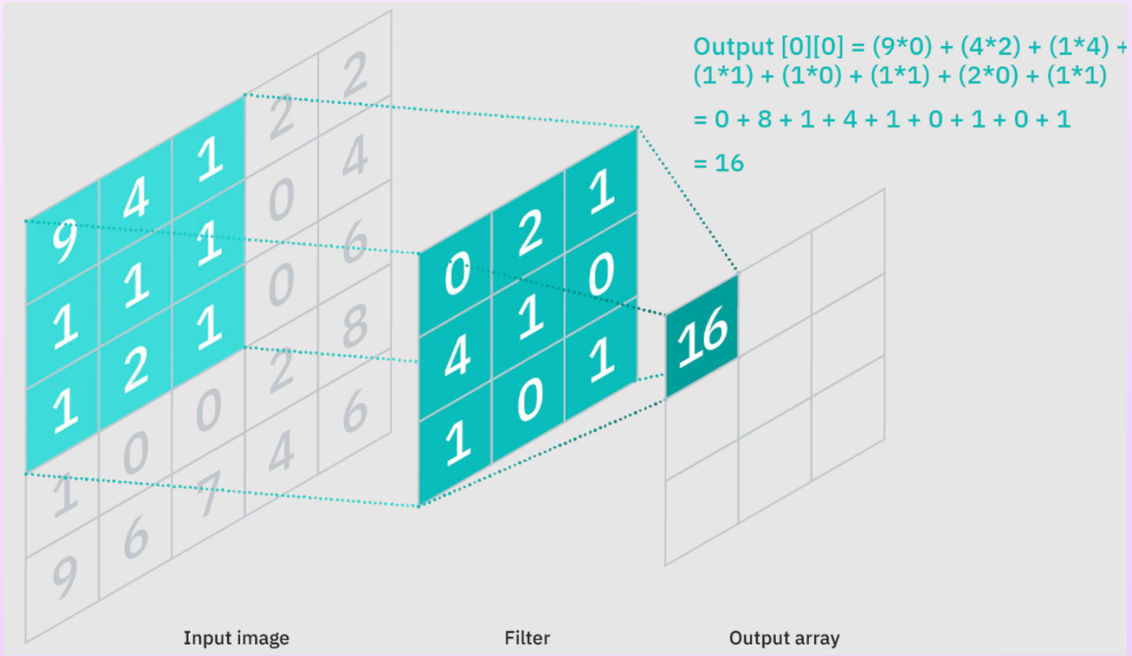
YOLO v3 network Architecture

YoloV3 Architecture



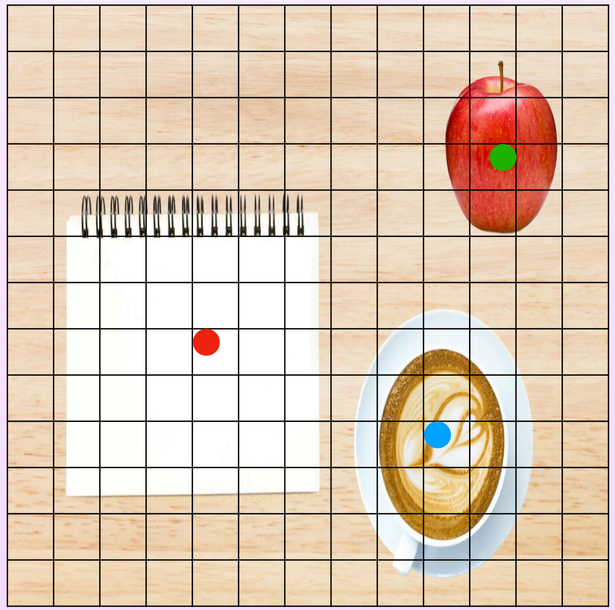
Max Pooling Vs Conv Layer





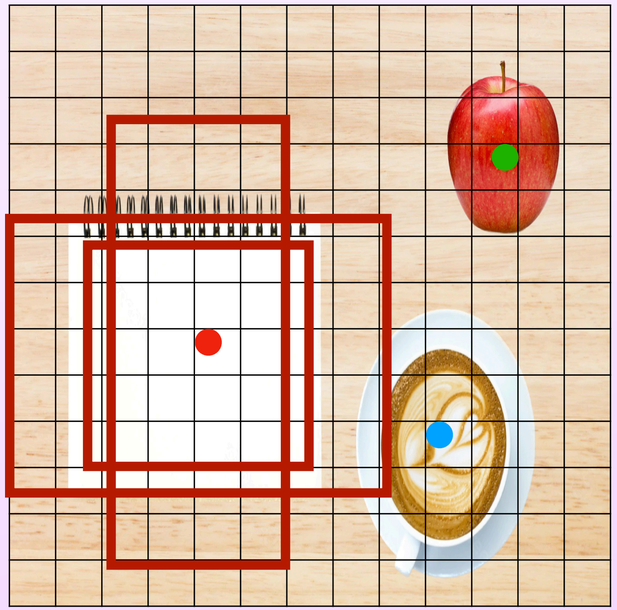
Grid Cells

- Image is divided into grids



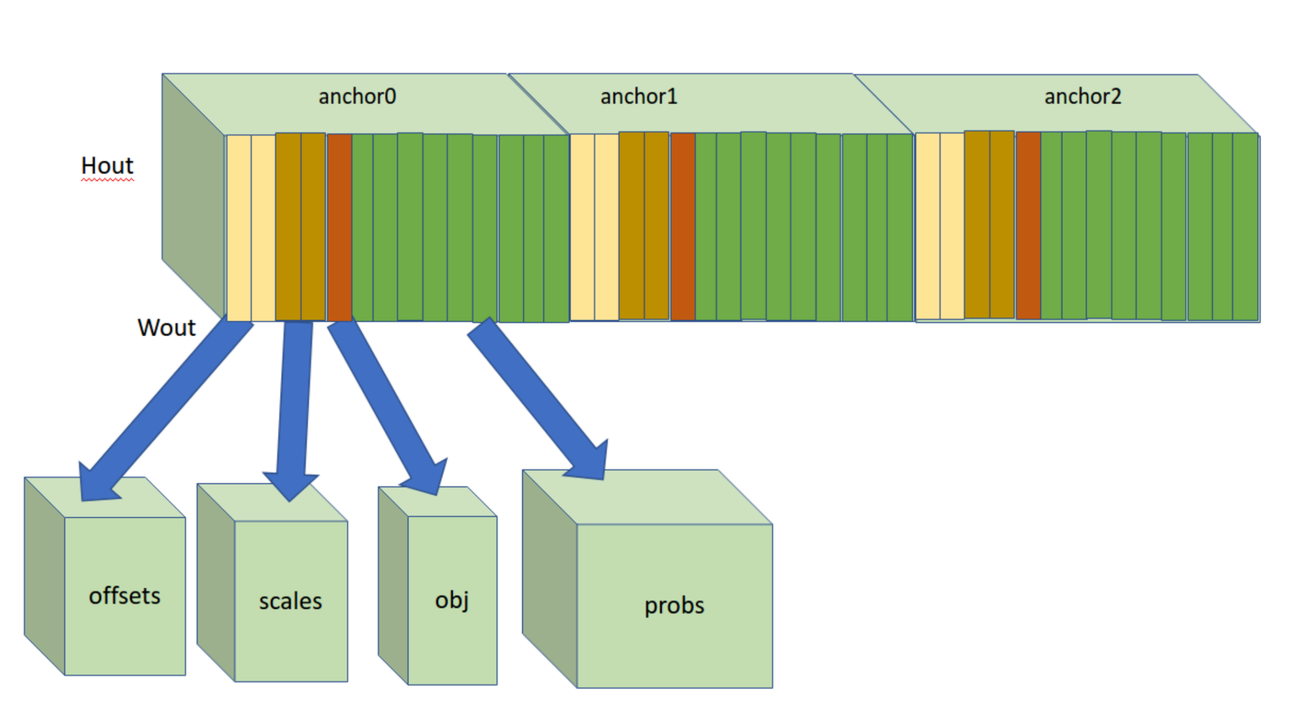
Grid Cells

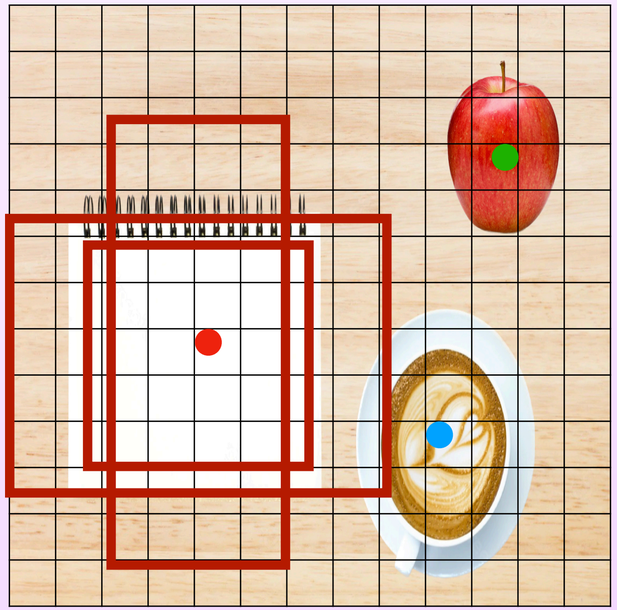
- Image is divided into grids
- Anchor boxes - 3 per grid cell



Grid Cells

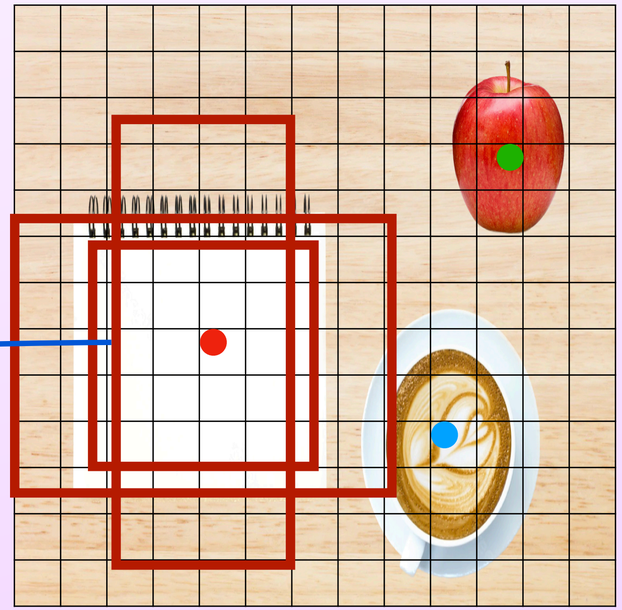
- Image is divided into grids
- Anchor boxes - 3 per grid cell

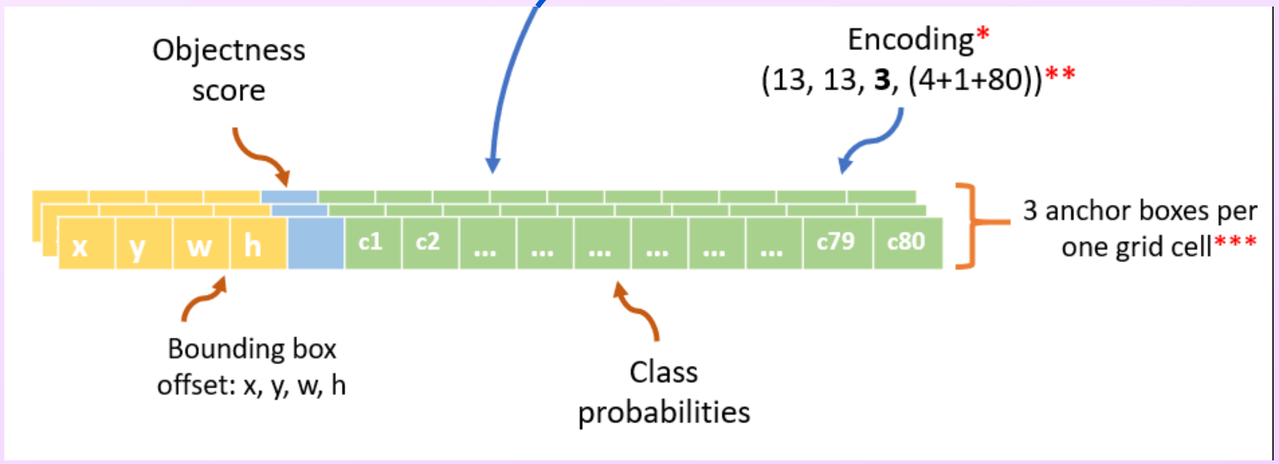




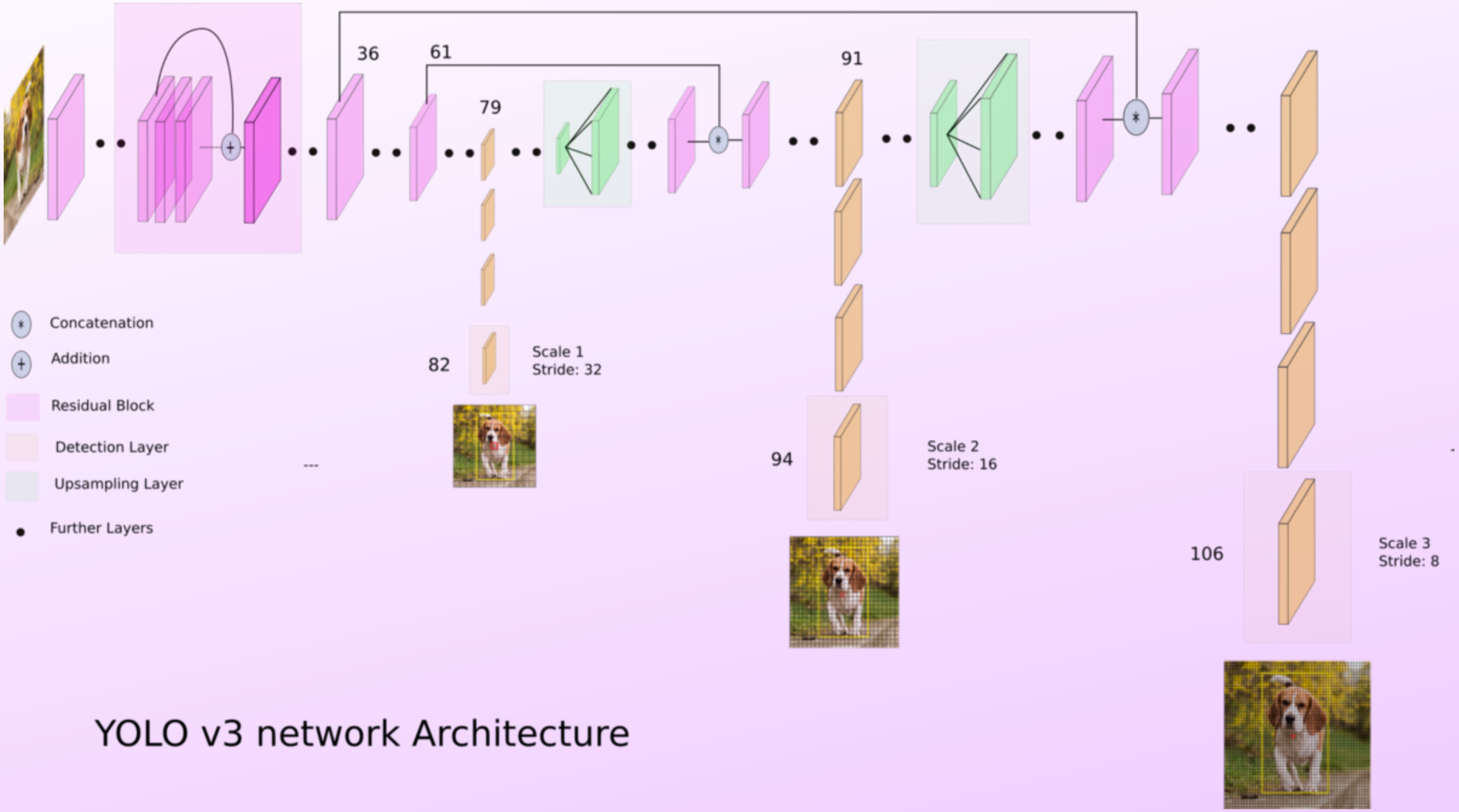
Grid Cells

- Image is divided into grids
- Anchor boxes - 3 per grid cell



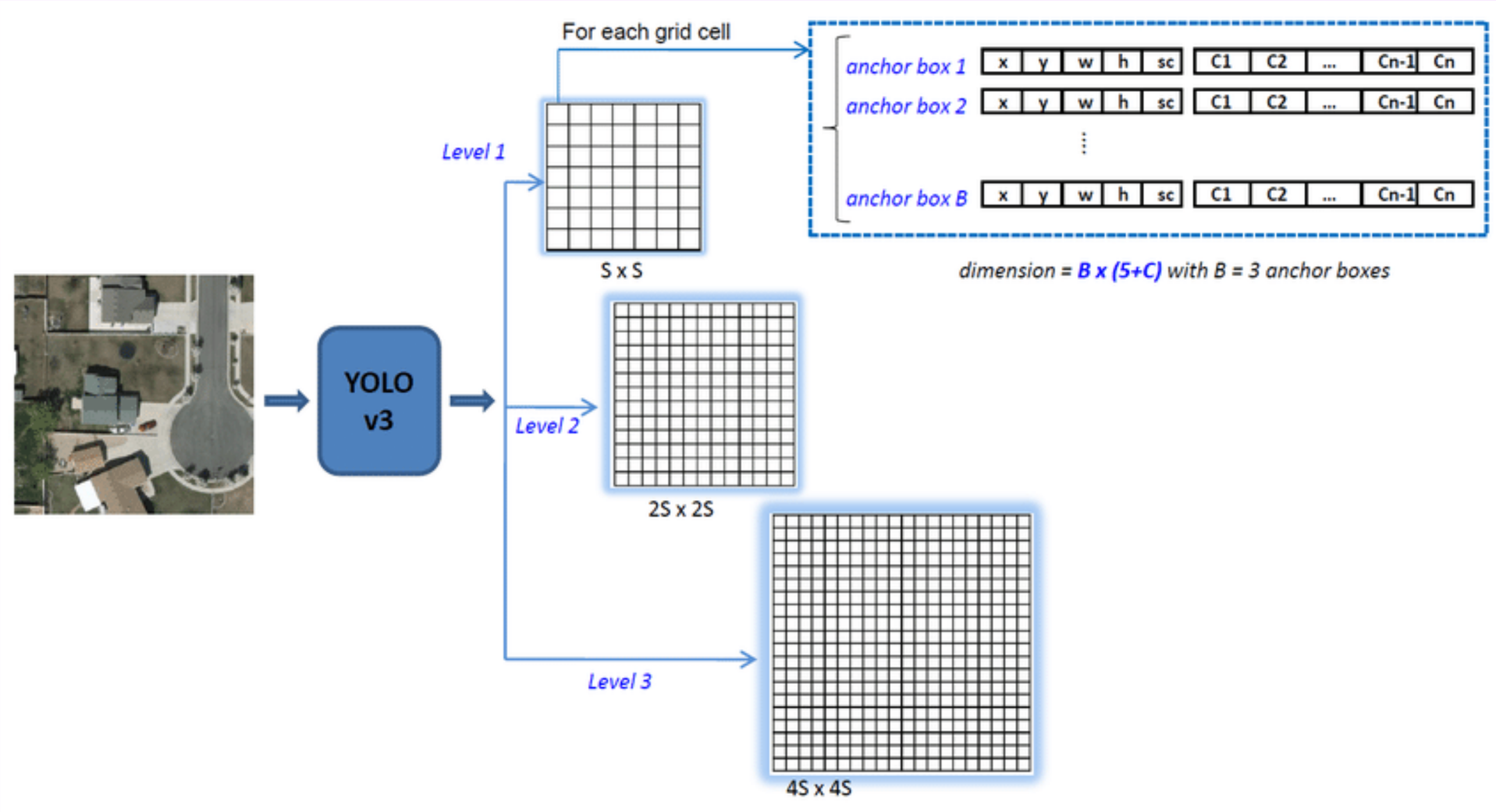


Predictions across Scales

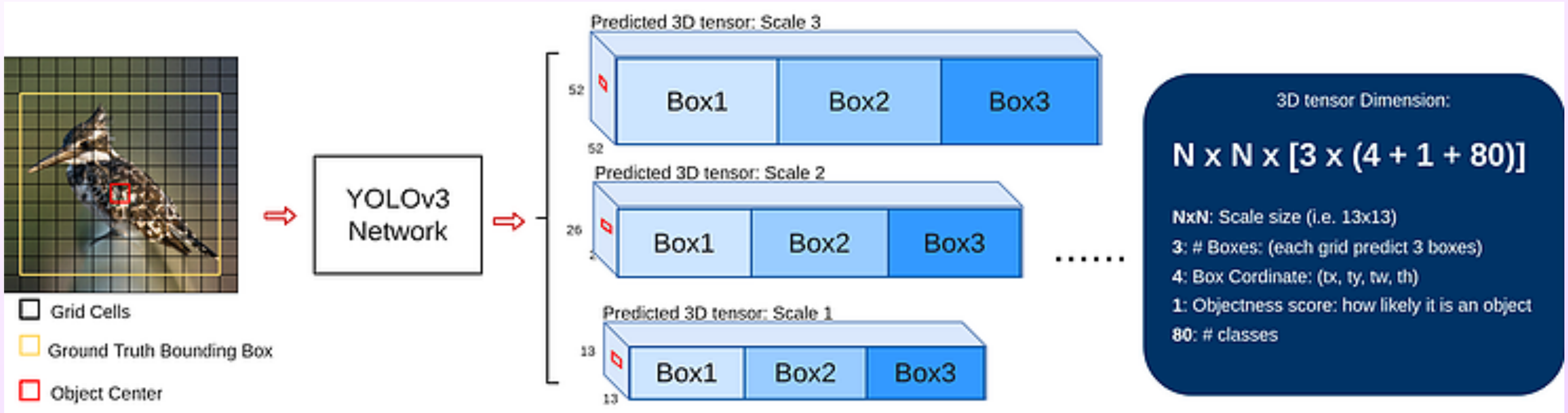


YOLO v3 network Architecture

Predictions across Scales



Predictions across Scales



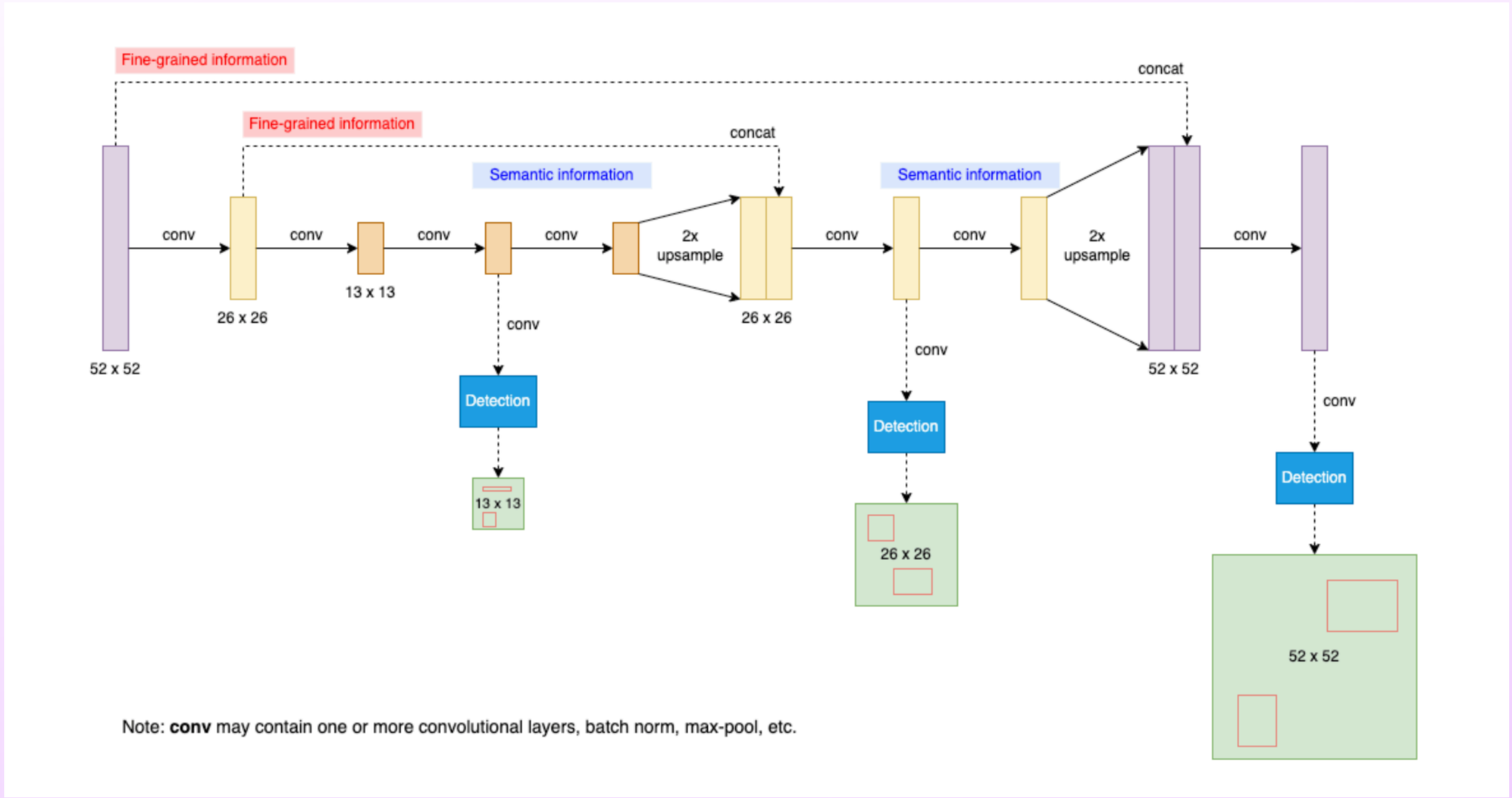
Total Predictions

- Grids - 13x13, 26x26, 52x52
- Total Grids - 3549
- Each grid has 3 boxes
- Total boxes - 3549*3 = 10647 boxes

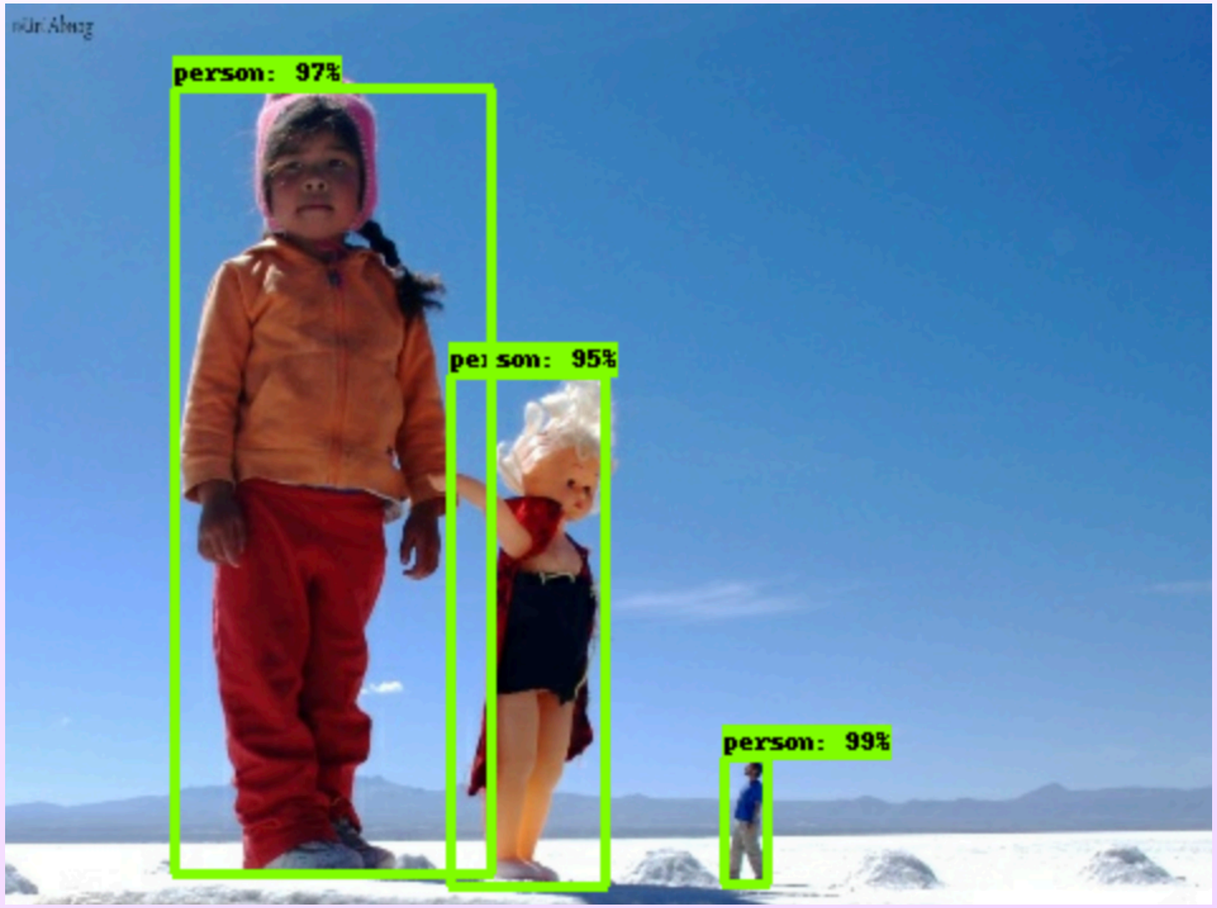
More boxes

- YoloV1 predicts **98** boxes (7x7 grid cells, 2 boxes per cell @448x448)
- YoloV2 predicts **845** boxes (13x13 grid cells, 5 anchor boxes per grid cell @416x416)
- YoloV3 predicts **10647** boxes (13x13, 26x26,52x52 grid cells, 3 anchor boxes per grid cell @416x416)
- YoloV3 predicts more than **10x** the number of boxes predicted by YoloV2

Small Object Detection



Small Object Detection



Bounding Box Prediction

Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$

Bounding Box Prediction

Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$

Calculate Bx, By

Bounding Box Prediction

Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$

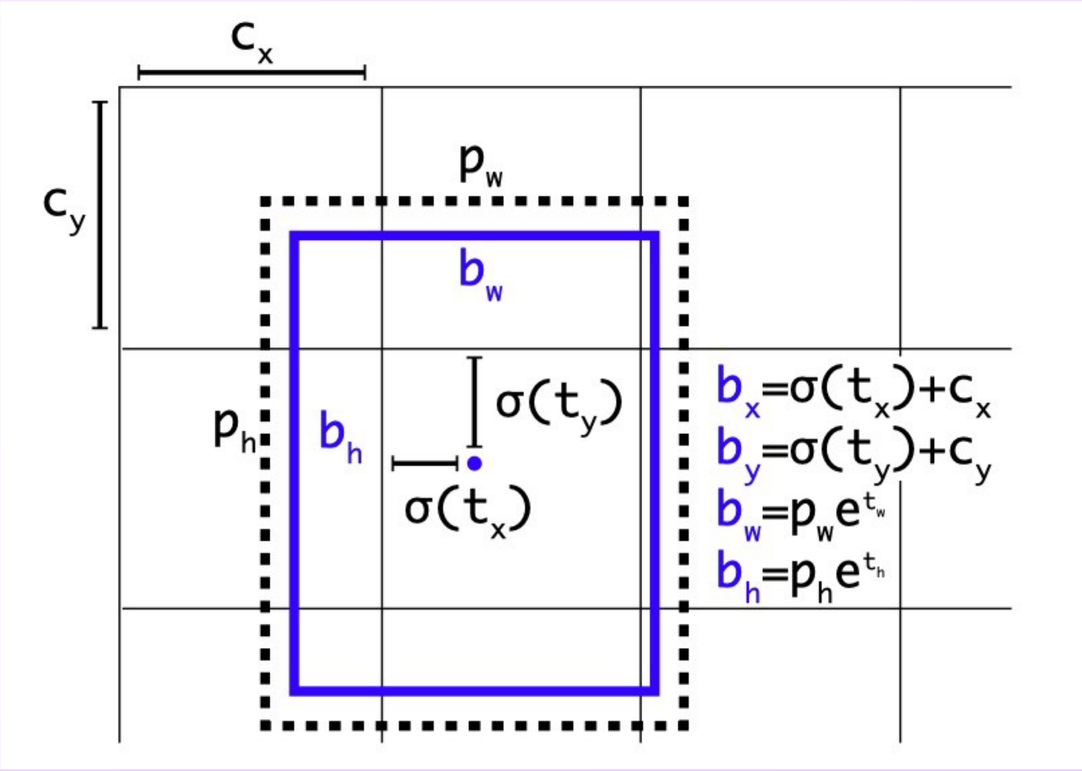
Calculate Bx, By

$$b_x = \sigma(t_x) + c_x \quad (c_x \text{ is the offset from the left of the image})$$
$$b_y = \sigma(t_y) + c_y \quad (c_y \text{ is the offset from the top of the image})$$

Bounding Box Prediction

Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$

$$b_x = \sigma(t_x) + c_x \quad (c_x \text{ is the offset from the left of the image})$$
$$b_y = \sigma(t_y) + c_y \quad (c_y \text{ is the offset from the top of the image})$$

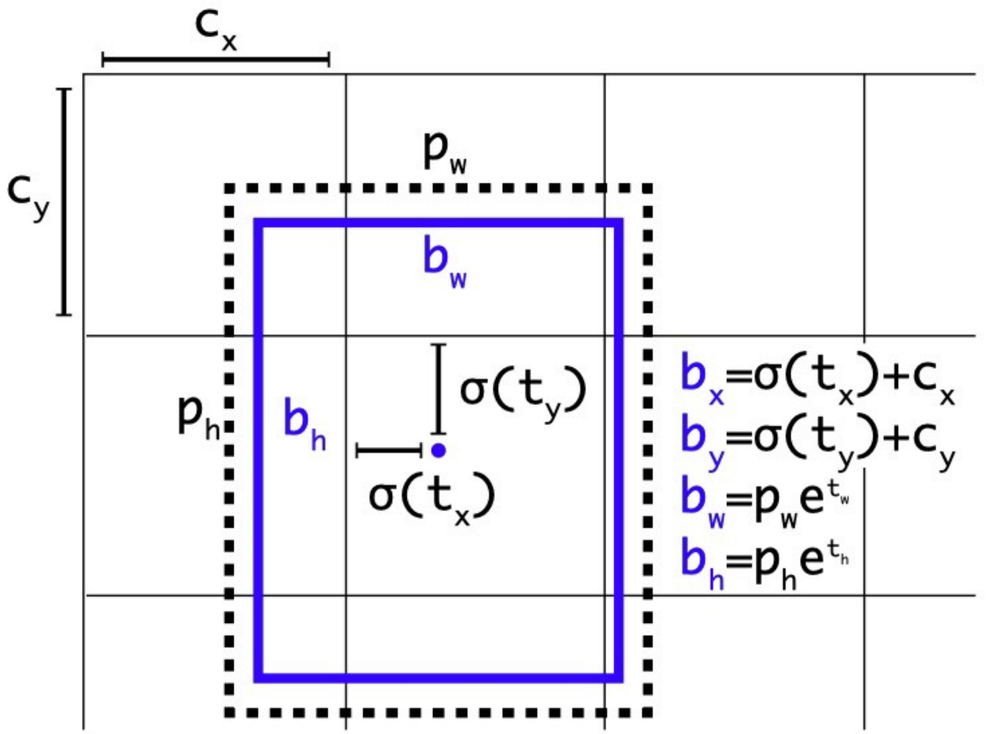


Bounding Box Prediction

Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$

$$b_x = \sigma(t_x) + c_x \quad (c_x \text{ is the offset from the left of the image})$$
$$b_y = \sigma(t_y) + c_y \quad (c_y \text{ is the offset from the top of the image})$$

Direct Location Prediciton



Bounding Box Prediction

Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$

Calculate Bw, Bh

Bounding Box Prediction

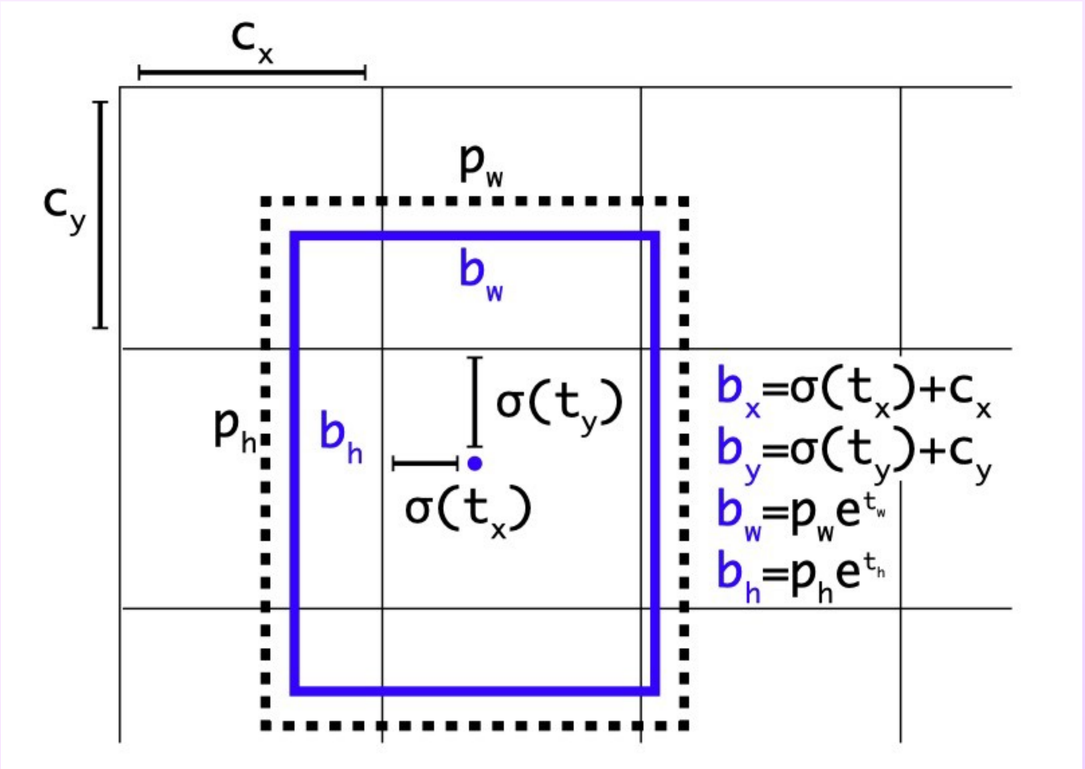
Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$

Calculate Bw, Bh

$$b_w = p_w e^{t_w} \quad (p_w \text{ is the width of anchor box})$$
$$b_h = p_h e^{t_h} \quad (p_h \text{ is the height of anchor box})$$

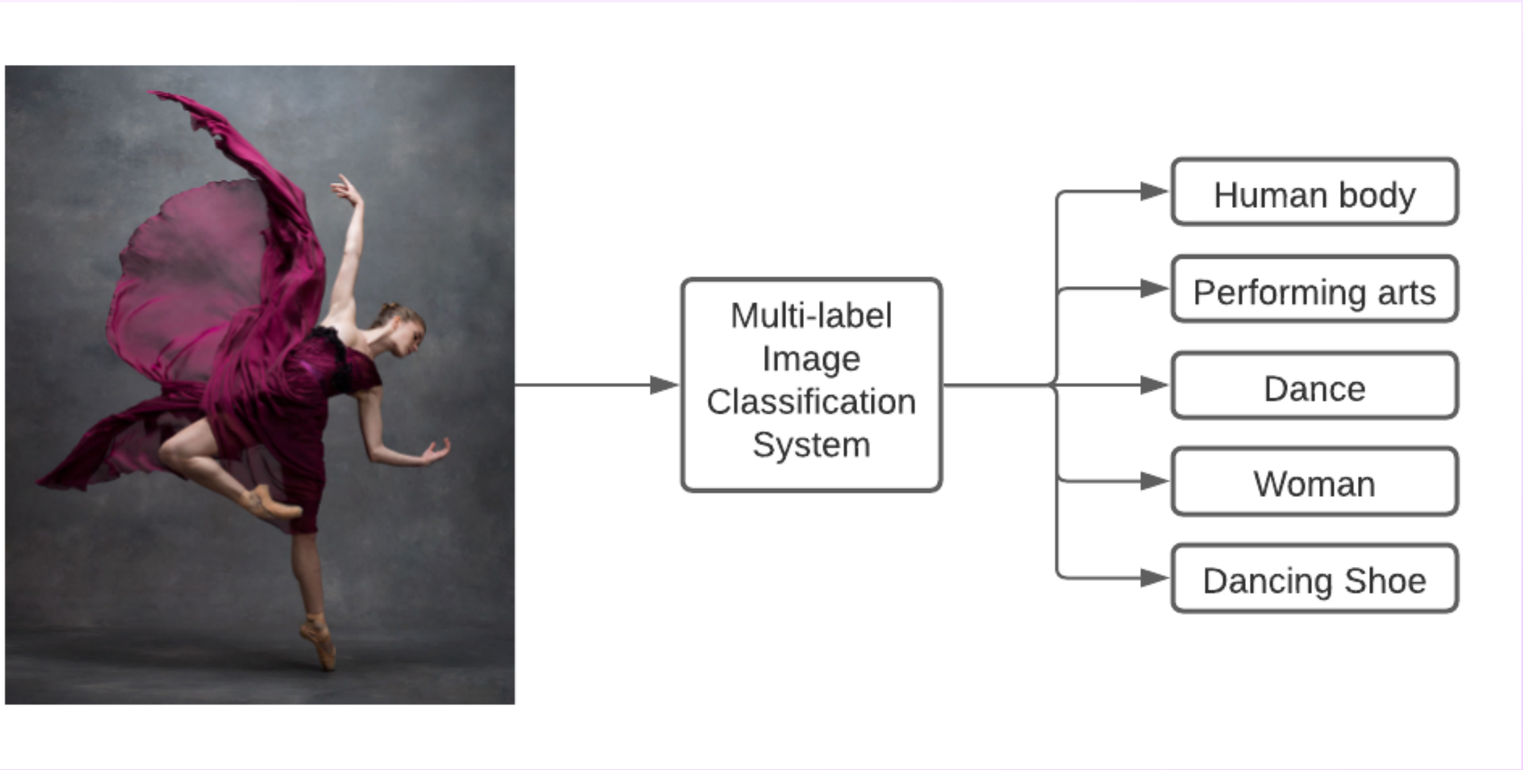
Bounding Box Prediction

Model outputs - $t_x, \quad t_y, \quad t_w, \quad t_h, \quad t_o$



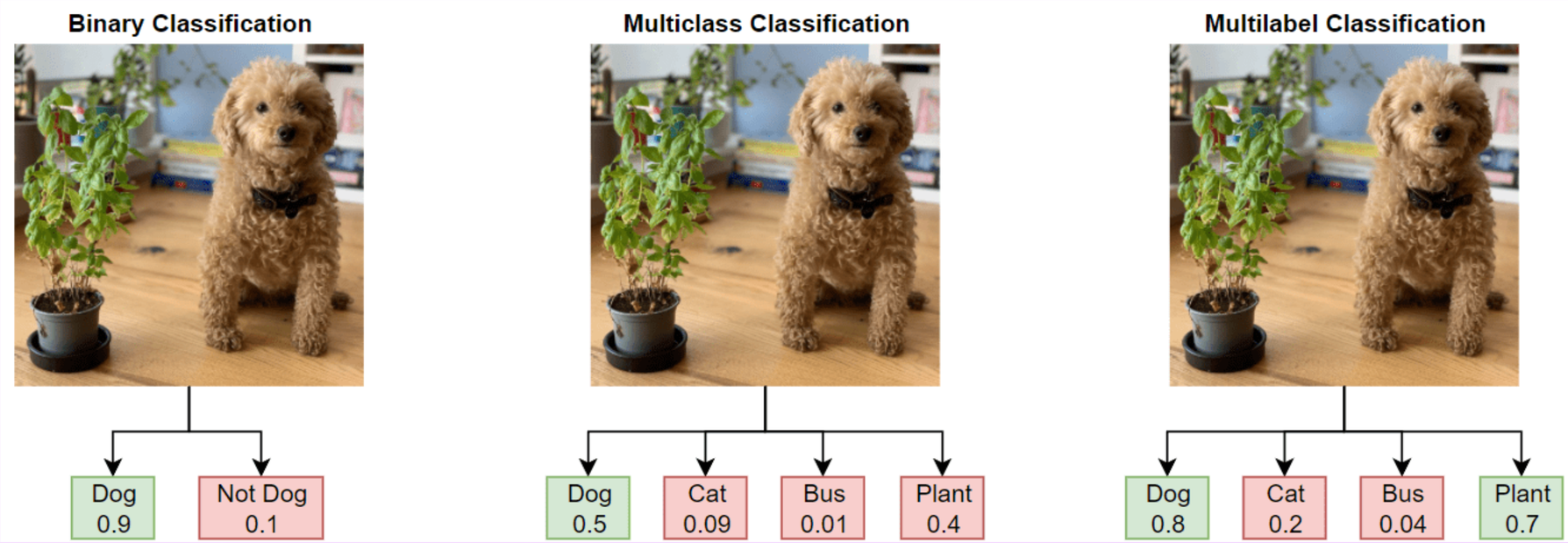
Class Prediction

Multi-label classification



Class Prediction

Multi-label classification



Loss Function

Regression loss

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

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

Confidence loss

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

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

Classification loss

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

Loss Function

Regression loss

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

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

Confidence loss

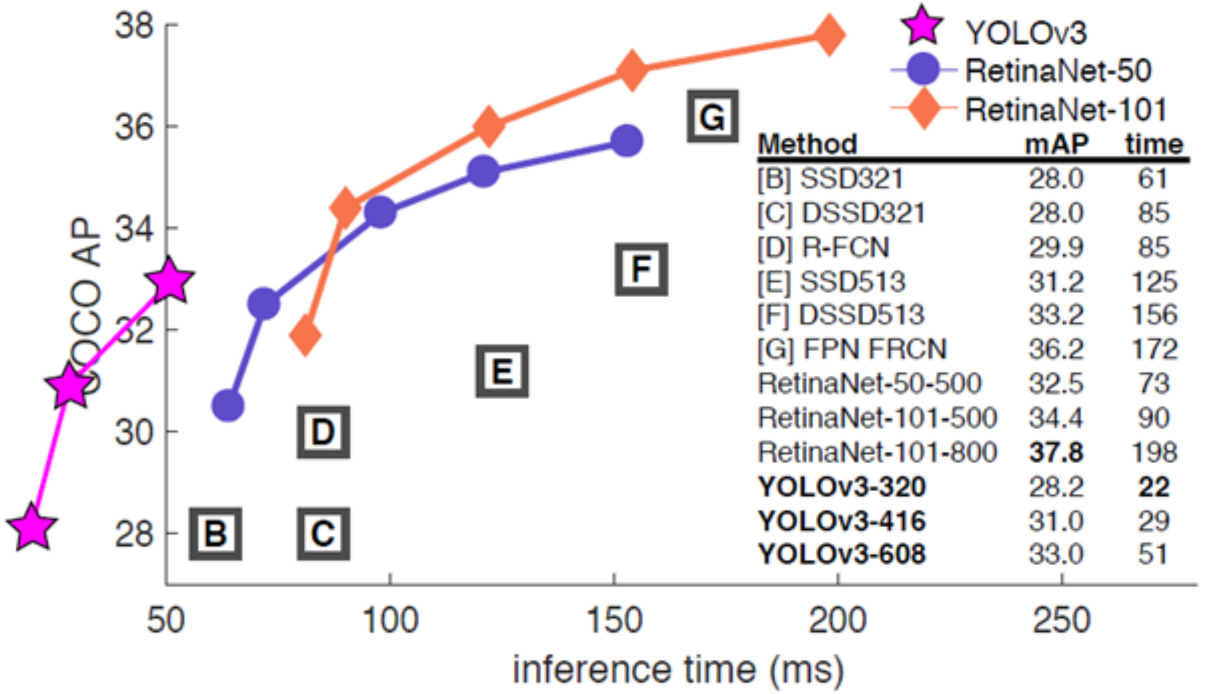
$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i \log(C_i) + \left(1 - \hat{C}_i \right) \log\left(1 - C_i \right) \right] +$$

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{noobj} \left[\hat{C}_i \log(C_i) + \left(1 - \hat{C}_i \right) \log(1 - C_i) \right] +$$

Classification loss

$$\sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in classes} \left[\hat{p}_i(c) \log(p_i(c)) + \left(1 - \hat{p}_i(c) \right) \log(1 - p_i(c)) \right]$$

Performance



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

Figure 1. We adapt this figure from the Focal Loss paper [7]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

Conclusion

- A Good Object Detector
- It’s fast and accurate
- It’s not the State of the art in terms of accuracy
- Fastest Object Detector