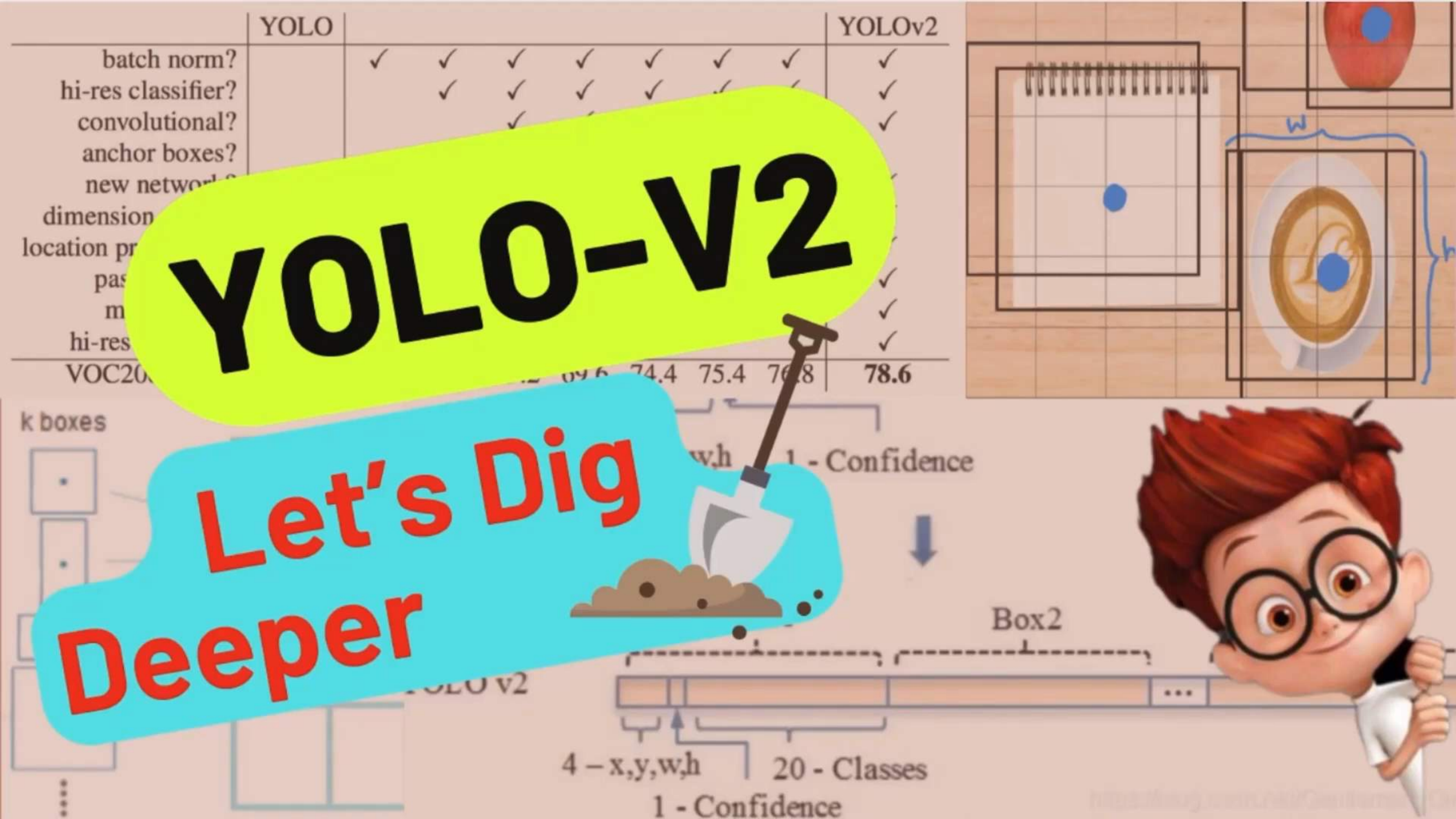
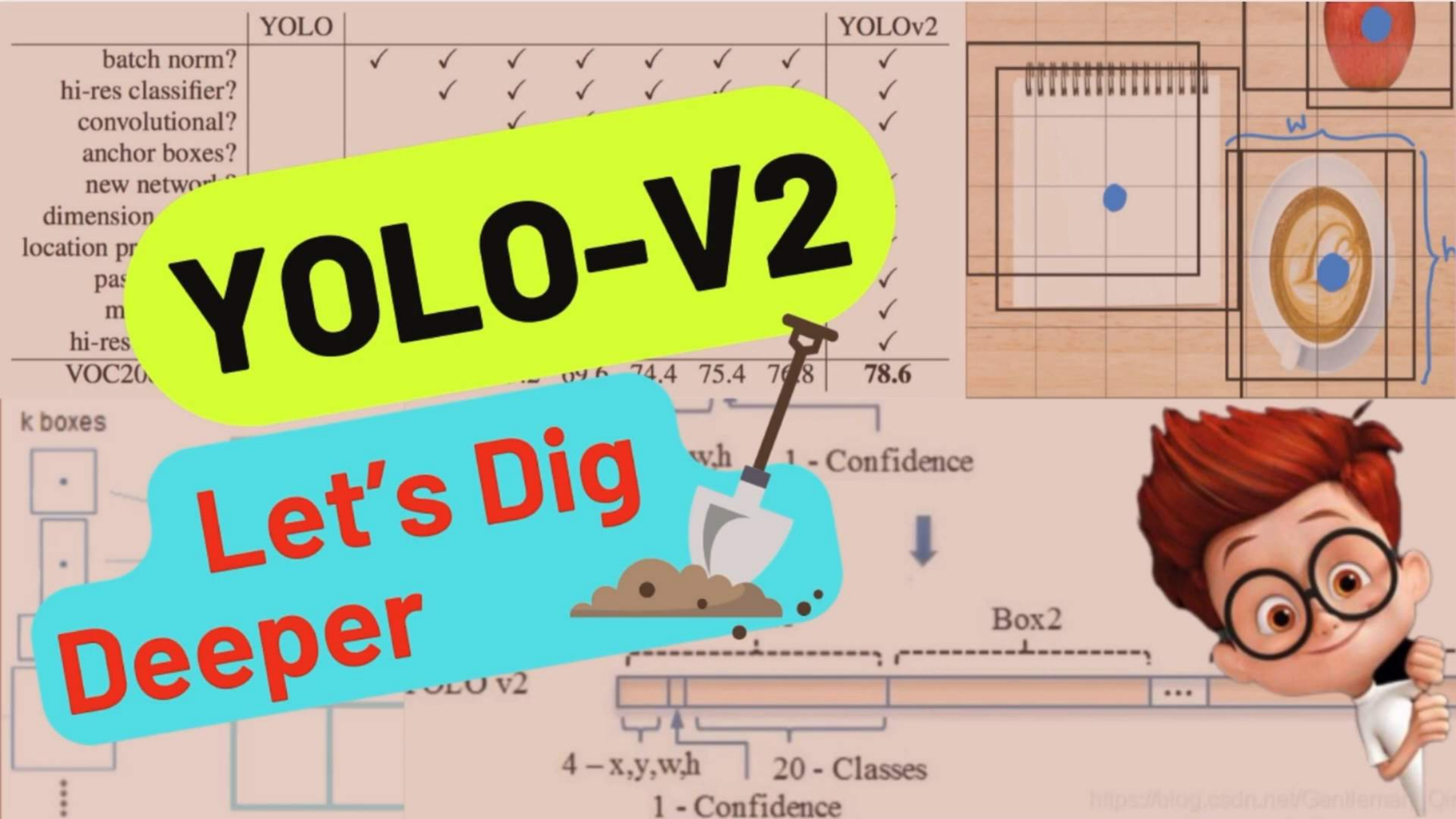
YOLO-V2

Let's Dig Deeper




YOLO
YOLOv2
batch norm?
hi-res classifier?
convolutional?
anchor boxes?
dimension
location pr
pas
m
hi-res
VOC20
75.4
78.6
YOLO-V2
Let's Dig Deeper
k boxes
w,h
1 - Confidence
Box2
4 – x,y,w,h
20 - Classes
1 - Confidence

Recap of Yolo-V1

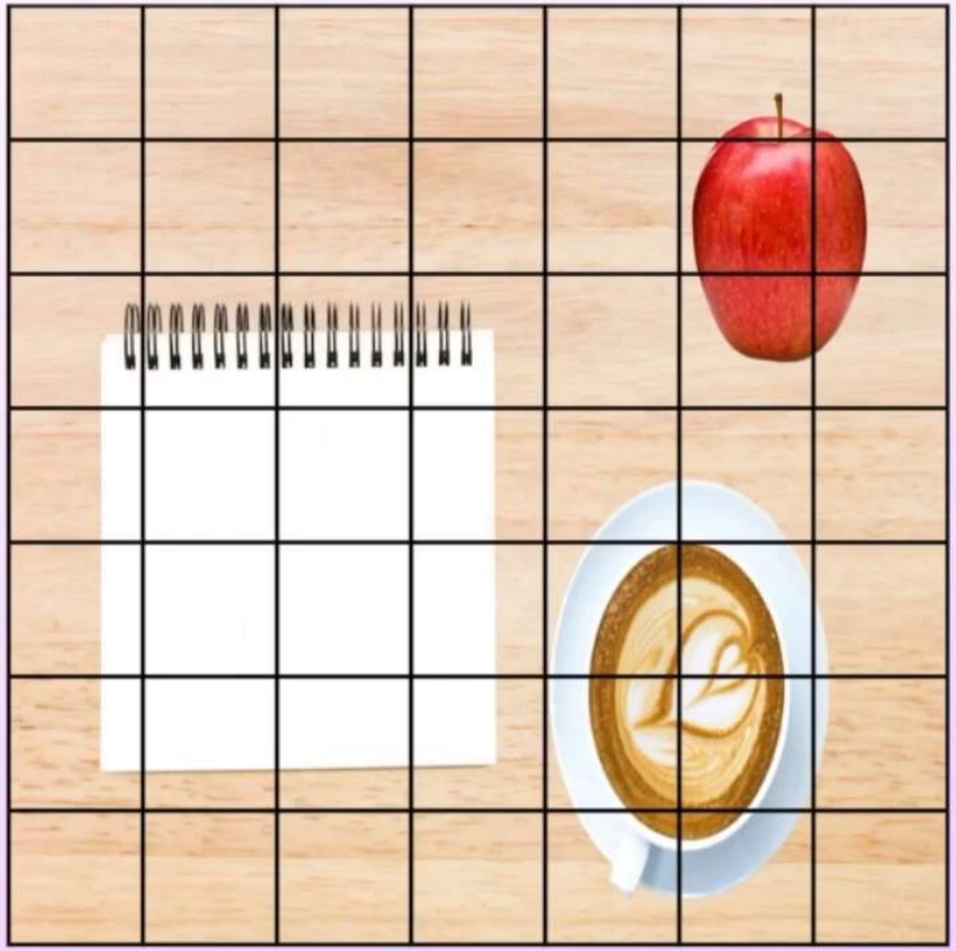
- Object Detection is formulated as regression
- Boxes and class probabilities are predicted in a single pass
- Trained on full images
- Learns contextual information better
- Features from whole image are used in detection of objects

Recap of Yolo-V1

- Object Detection is formulated as regression
- Boxes and class probabilities are predicted in a single pass
- Trained on full images
- Learns contextual information better
- Features from whole image are used in detection of objects

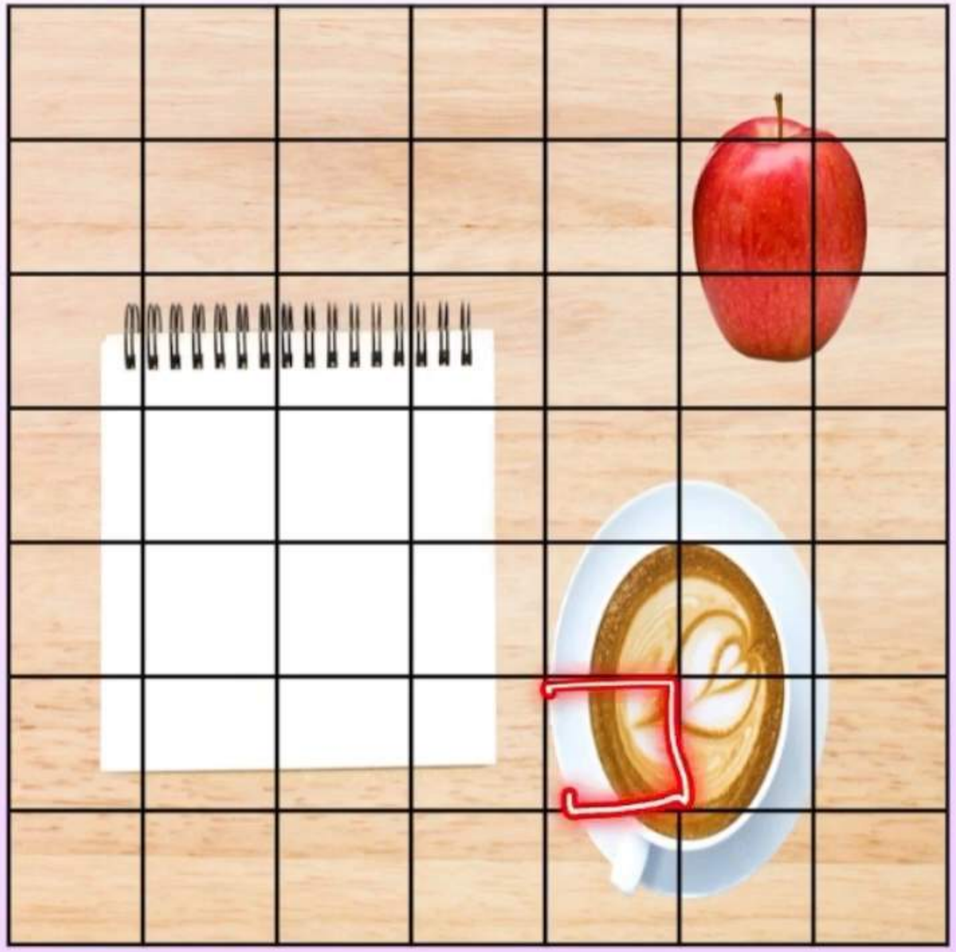
Watch at 1.5x speed

Recap of Yolo-V1



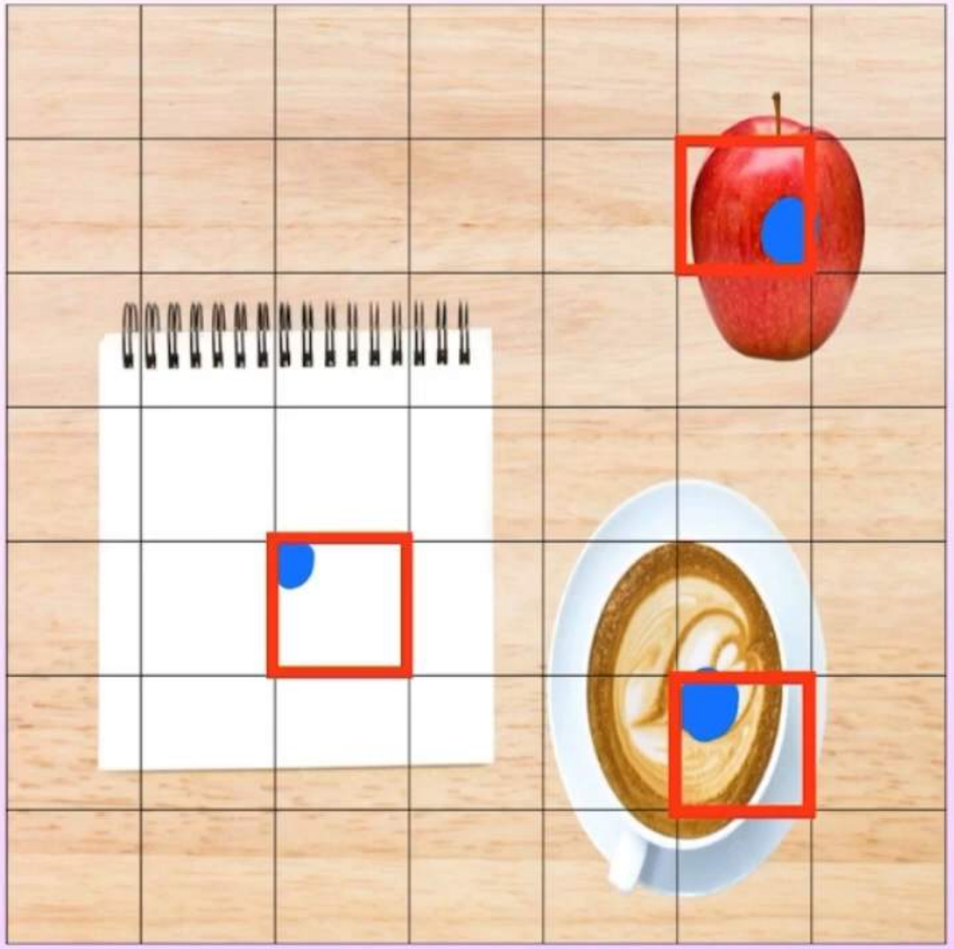
- Resize the image to 448x448
- Divide the image into SxS grids, S=7
- Each grid cell can detect 2 boxes - but only 1 with highest conf score is kept

Recap of Yolo-V1



- Resize the image to 448x448
- Divide the image into SxS grids, S=7
- Each grid cell can detect 2 boxes - but only 1 with highest conf score is kept

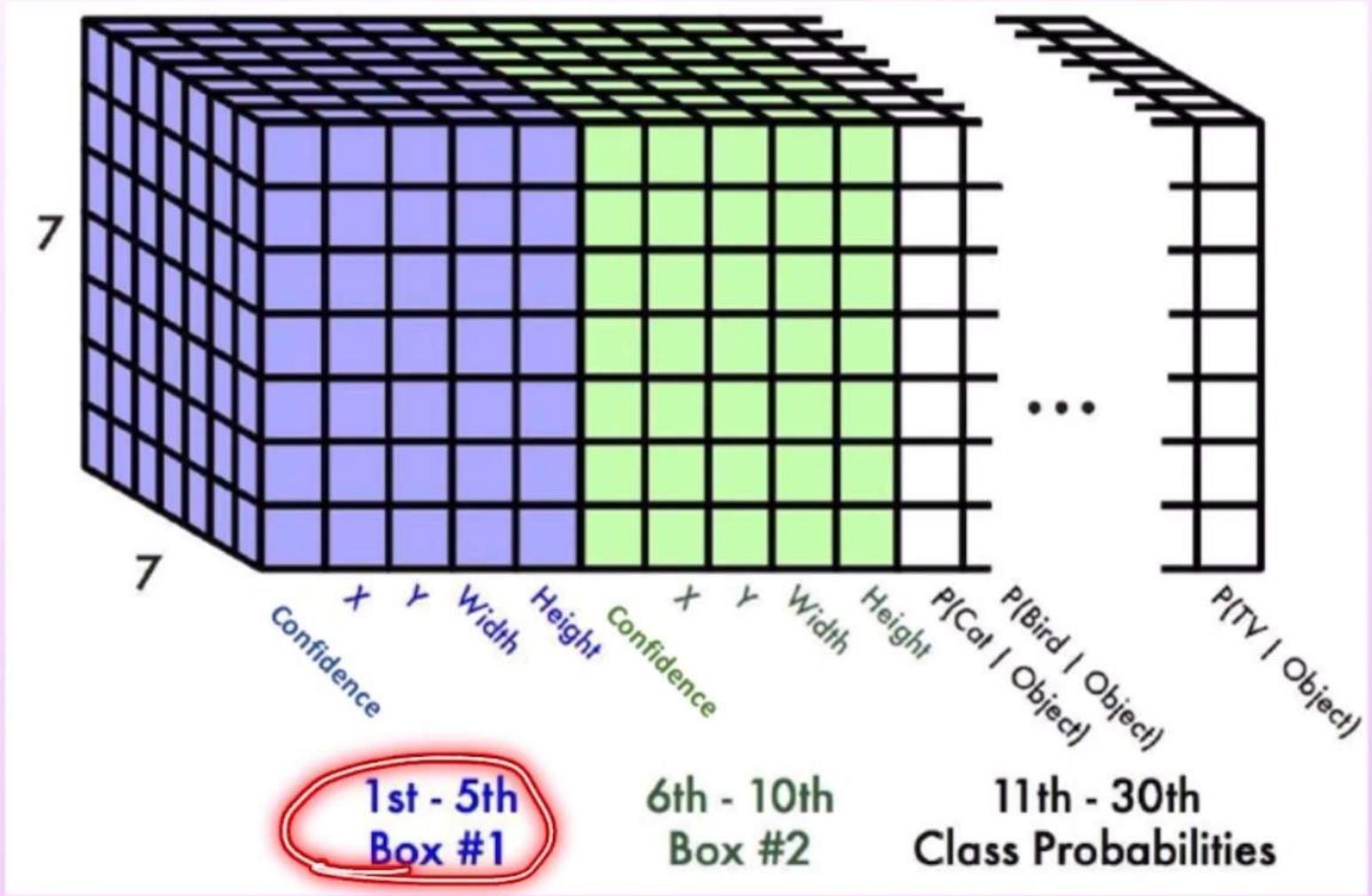
Recap of Yolo-V1



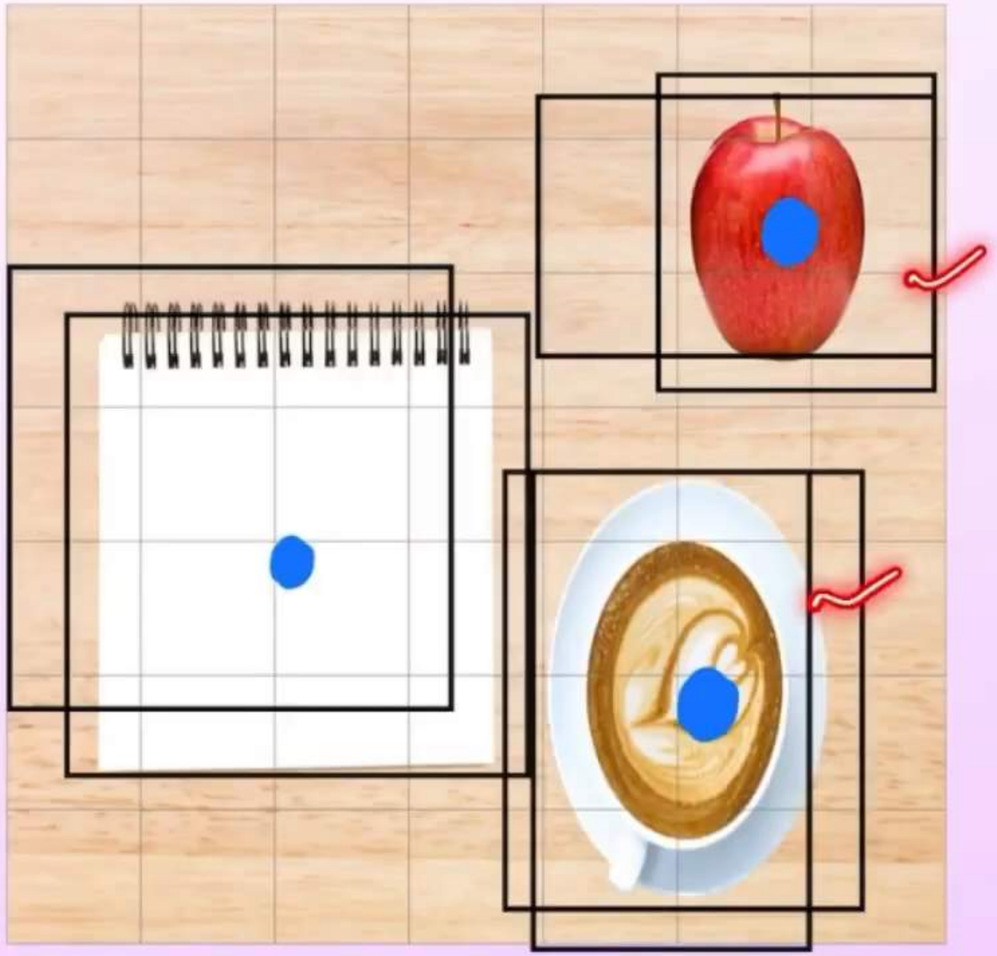
- Each Grid cell predicts 2 bounding boxes and a confidence score for each box

Recap of Yolo-V1

- Prediction Vector for all Grid Cells
- Each Grid cell - 30 Values



Recap of Yolo-V1



- Assume that network predicted these bounding boxes for 3 grid cells

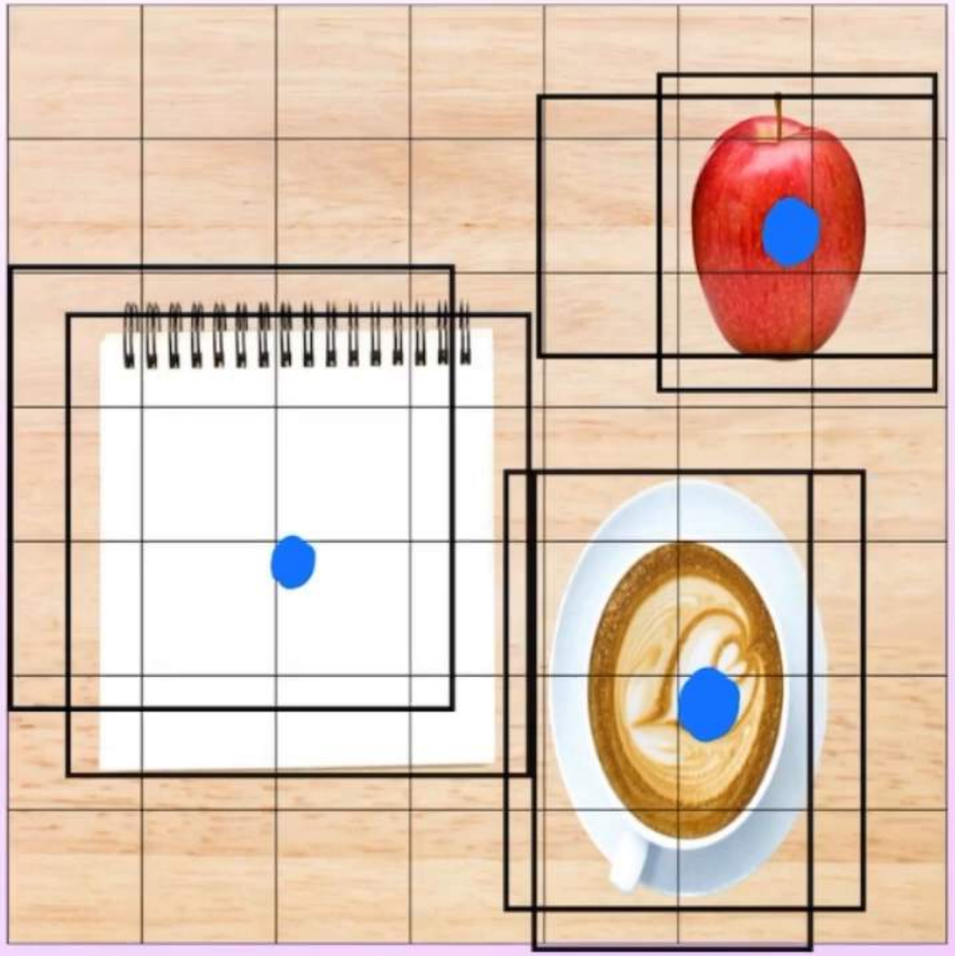
Recap of Yolo-V1

- For each box, a vector of size 5 values is predicted

X	Y	W	H	C

- Bounding box center as offsets to grid cell origin (top-left corner)
- X & y are normalised by grid cell size - 64 (448/7)

Recap of Yolo-V1

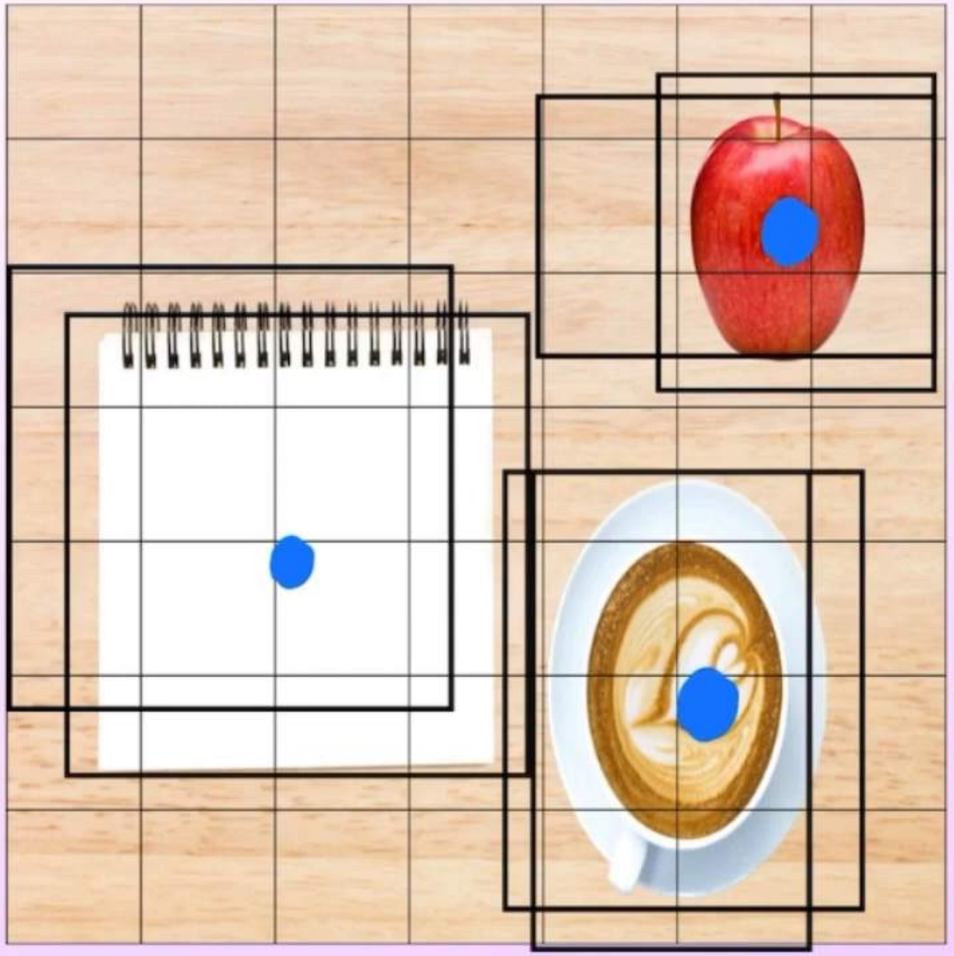


- For each box, a vector of size 5 values is predicted

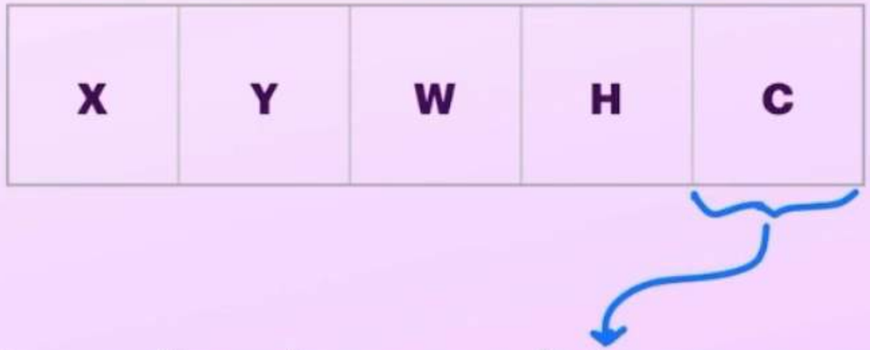
X	Y	W	H	C

- Bounding box confidence score
- This is **Objectness score** - How confident the is the model there is an object inside the box?

Recap of Yolo-V1

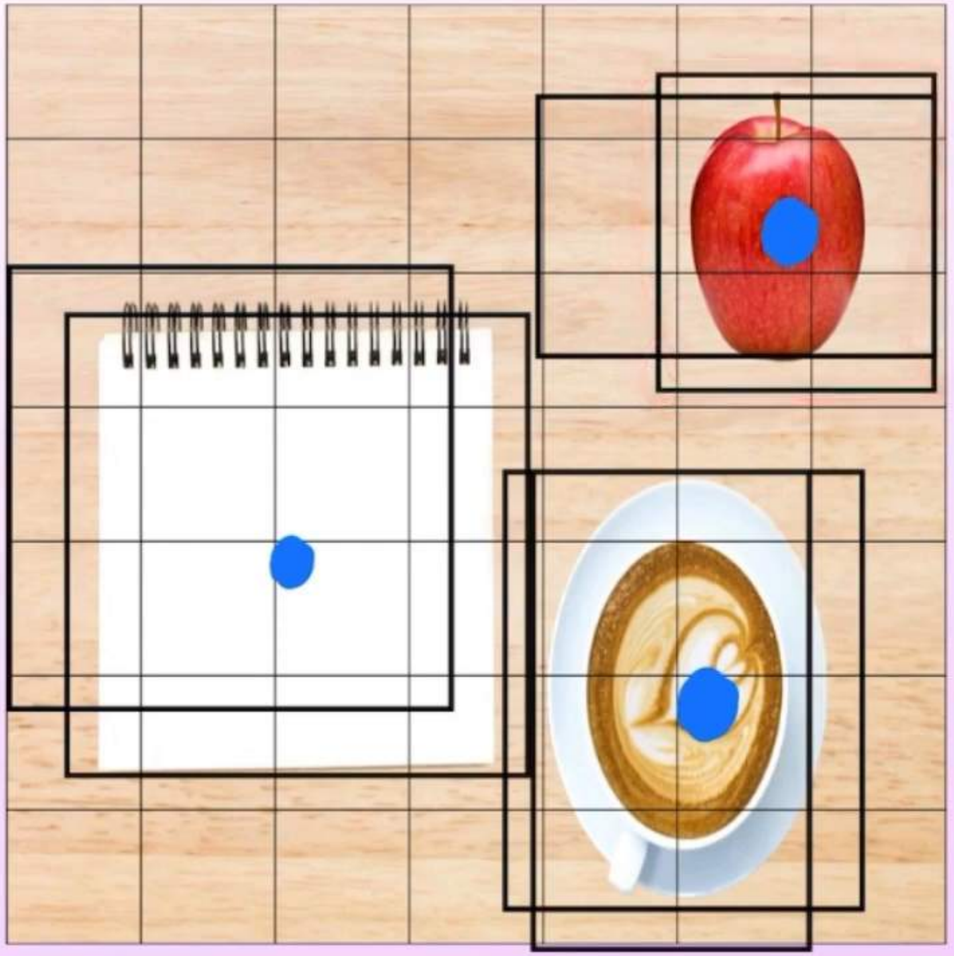


- For each box, a vector of size 5 values is predicted



- Bounding box confidence score
- This is **Objectness score** - How confident the is the model there is an object inside the box?

Recap of Yolo-V1

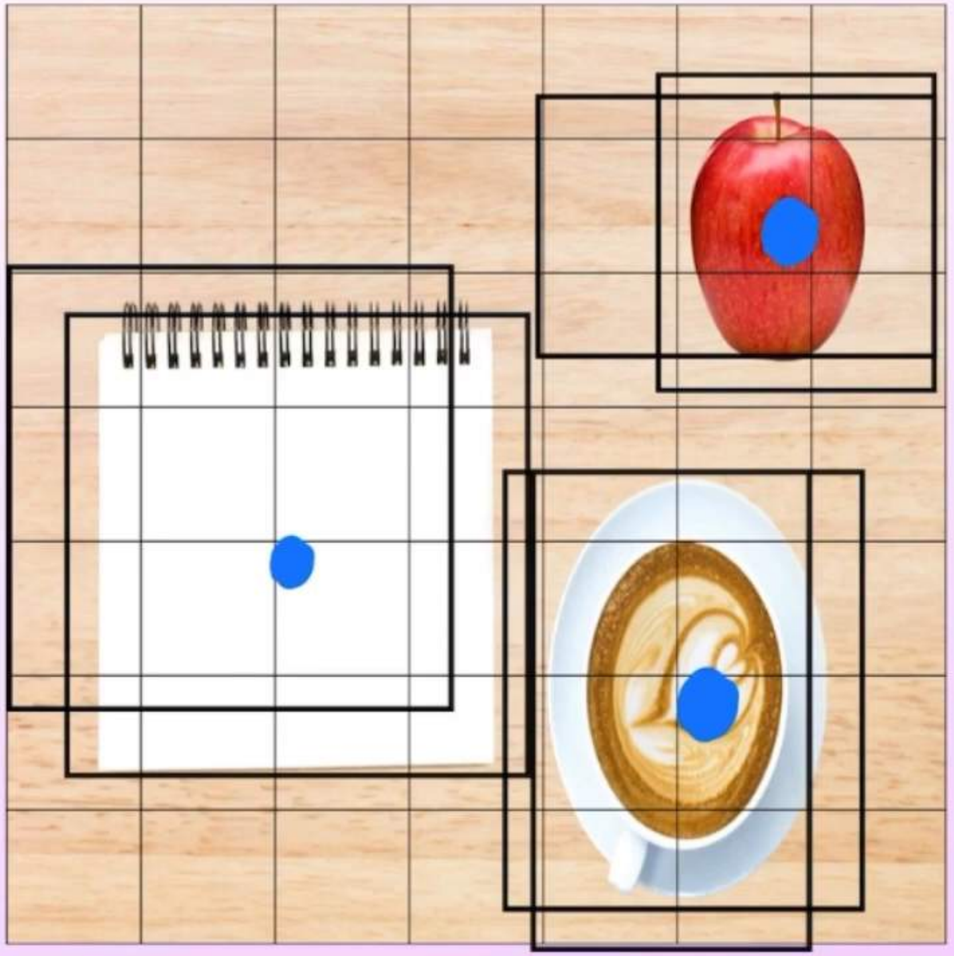


P1	P2	...	...	P20

X	Y	W	H	C

- Final Confidence Score for each box
- C’ = max(P1...P20) * C

Recap of Yolo-V1

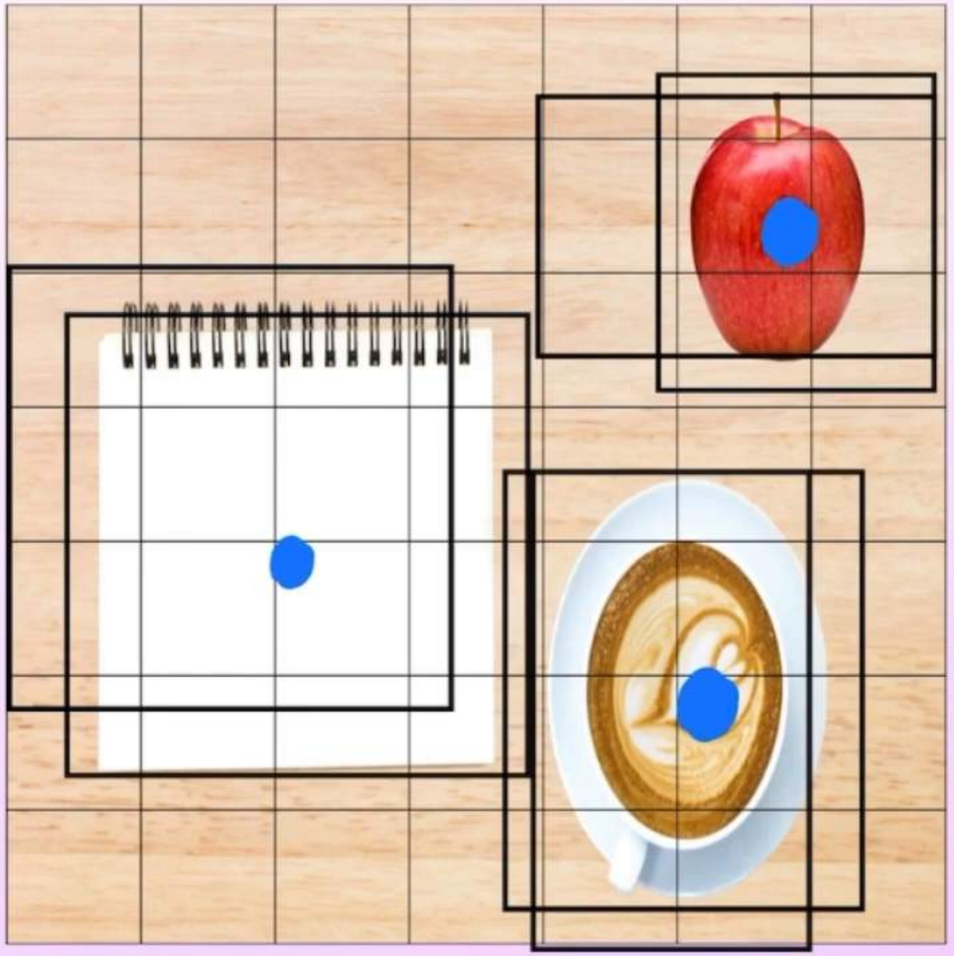


P1	P2	...	...	P20

X	Y	W	H	C

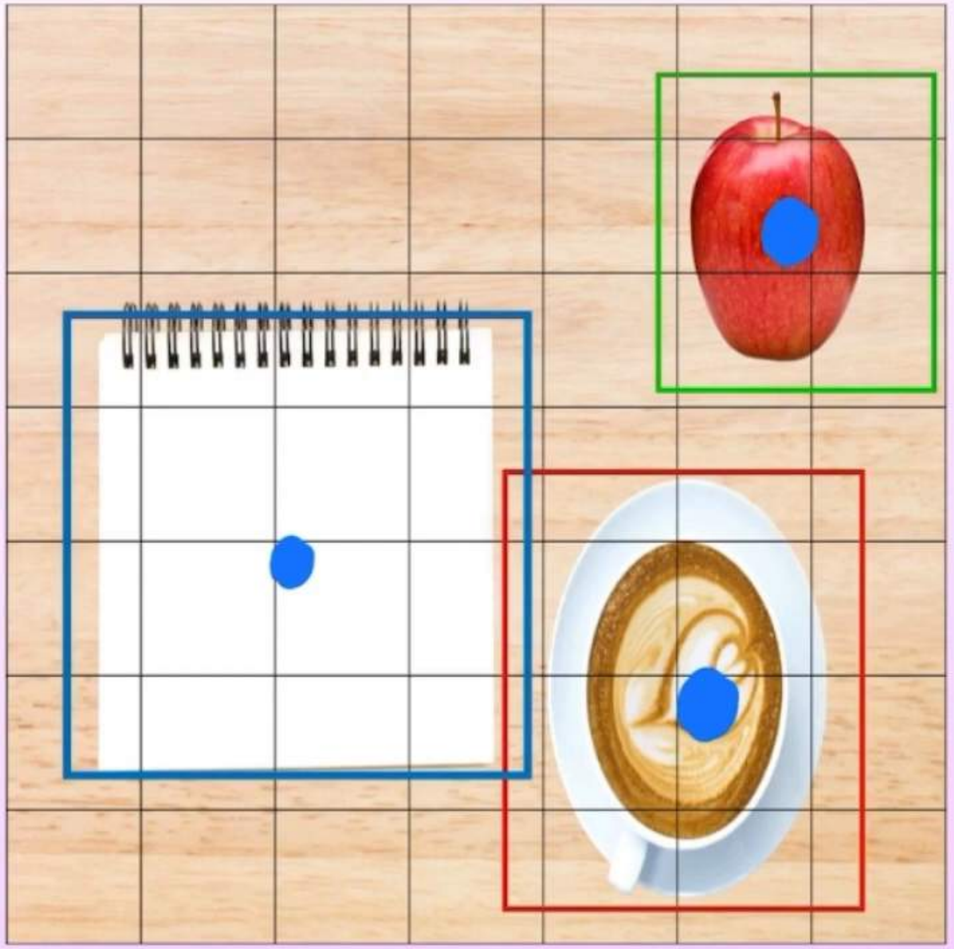
- Final Confidence Score for each box
- C’ = max(P1...P20) * C

Recap of Yolo-V1



- For both the boxes per grid cell
- C1′ = max(P1...P20) * C1
- C2′ = max(P1...P20) * C2

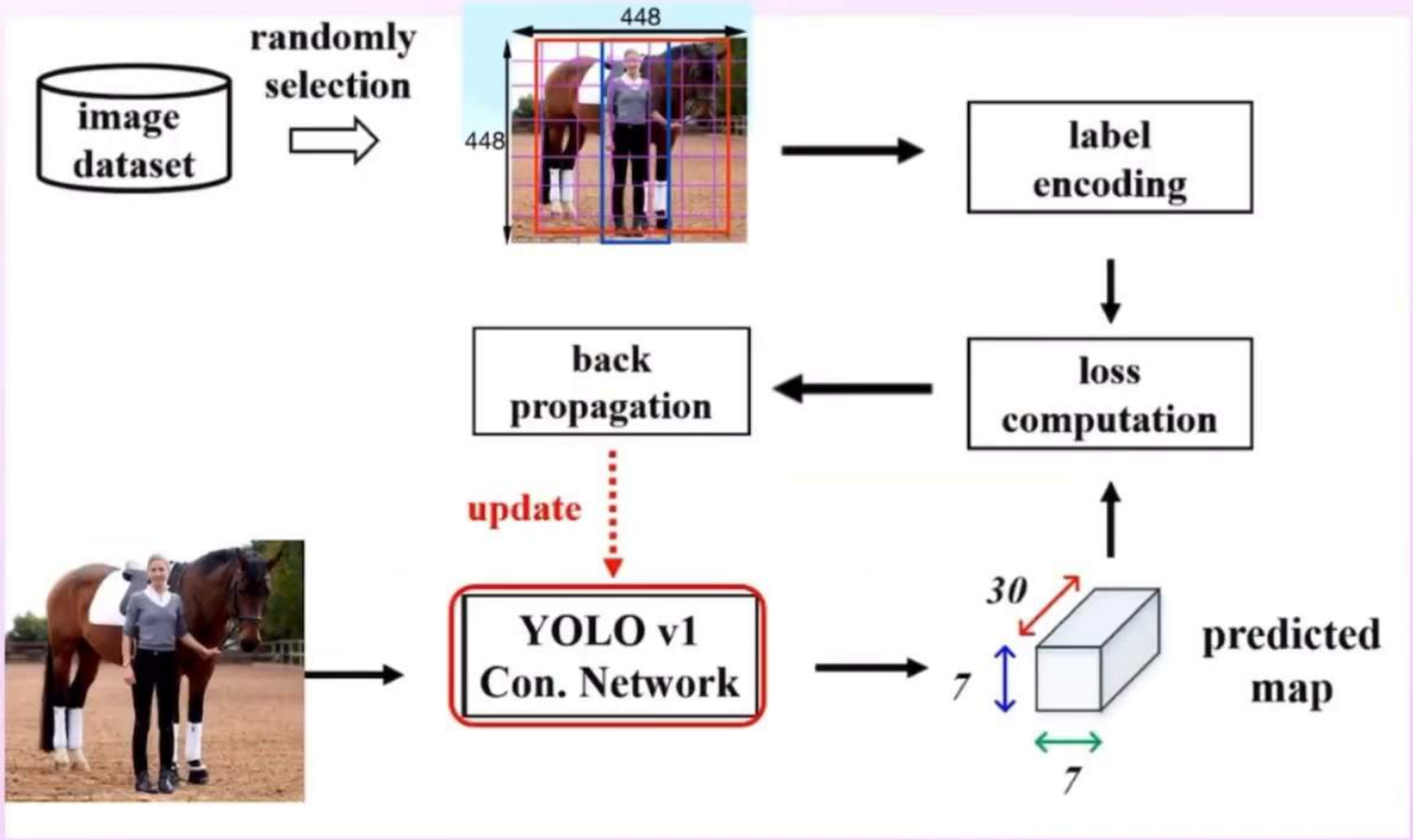
Recap of Yolo-V1



- Keep the box with highest confidence score
- Assign that classes:
- Apple ✔
- Cup
- Notebook

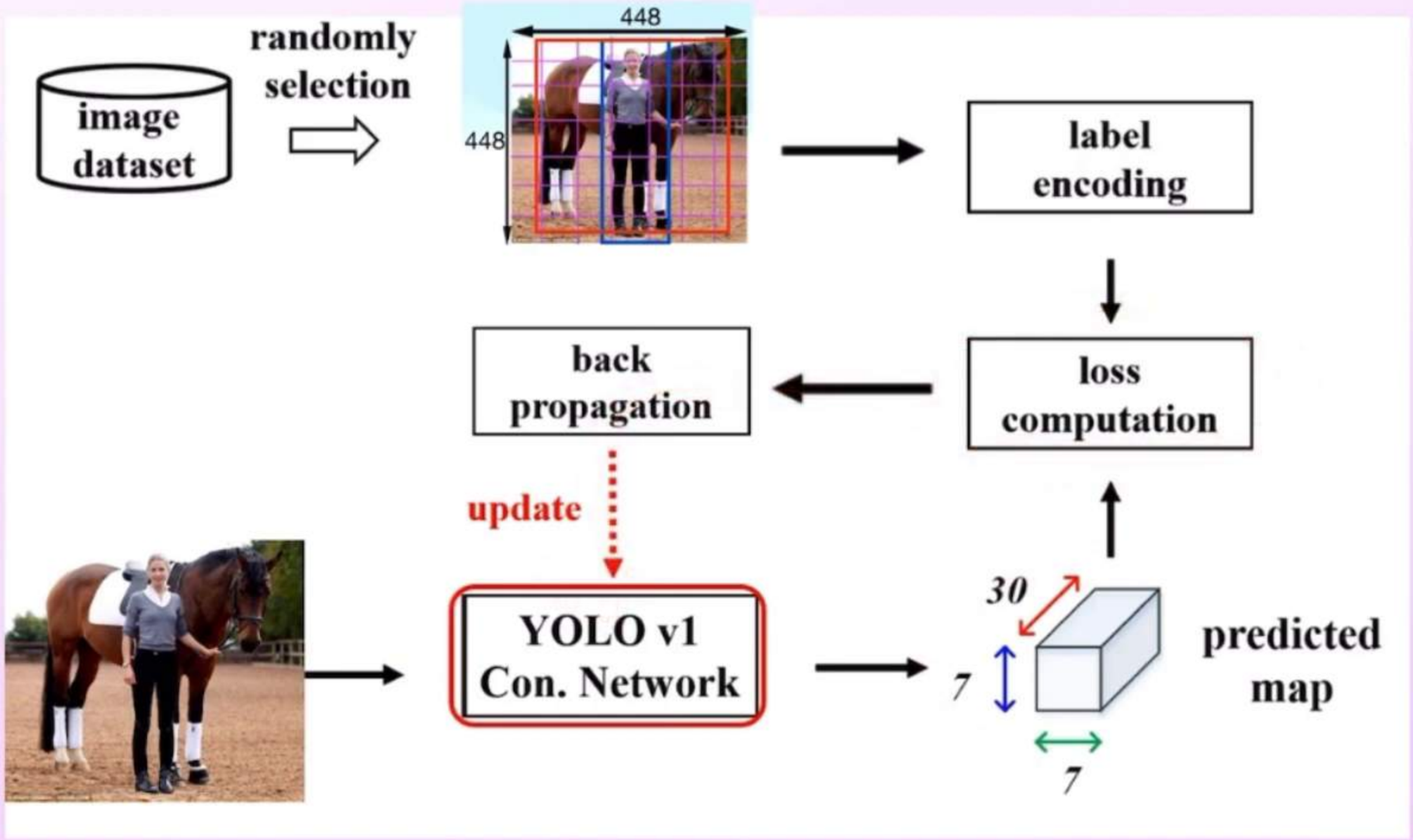
Recap of Yolo-V1

- Dataset: Pascal VOC - 20 classes



Recap of Yolo-V1

- Dataset: Pascal VOC - 20 classes



Recap of Yolo-V1

$$\boldsymbol{L} = \lambda_{coord} \times \sum_{i=1}^{S^2} 1_i^{obj} \times \left(\begin{array}{c} (\Delta x_i^* - \Delta \hat{x}_i)^2 + (\Delta y_i^* - \Delta \hat{y}_i)^2 + \\ \left(\sqrt{\Delta w_i^*} - \sqrt{\Delta \hat{w}_i}\right)^2 + \left(\sqrt{\Delta h_i^*} - \sqrt{\Delta h_i}\right)^2 \end{array} \right)$$

$$+ \sum_{i=1}^{S^2} 1_i^{obj} \times (c_i^* - \hat{c}_i)^2 + \sum_{i=1}^{S^2} 1_i^{obj} \times \sum_{c=1}^{20} \left(p_{i,c} - \hat{p}_{i,c}\right)^2$$

$$+ \lambda_{no\_obj} \sum_{i=1}^{S^2} 1_i^{no\_obj} \times \sum_{j=1}^{B} \left(c_{i,j} - \hat{c}_{i,j}\right)^2$$

YOLO9000:
Better, Faster, Stronger

Joseph Redmon*†, Ali Farhadi*†

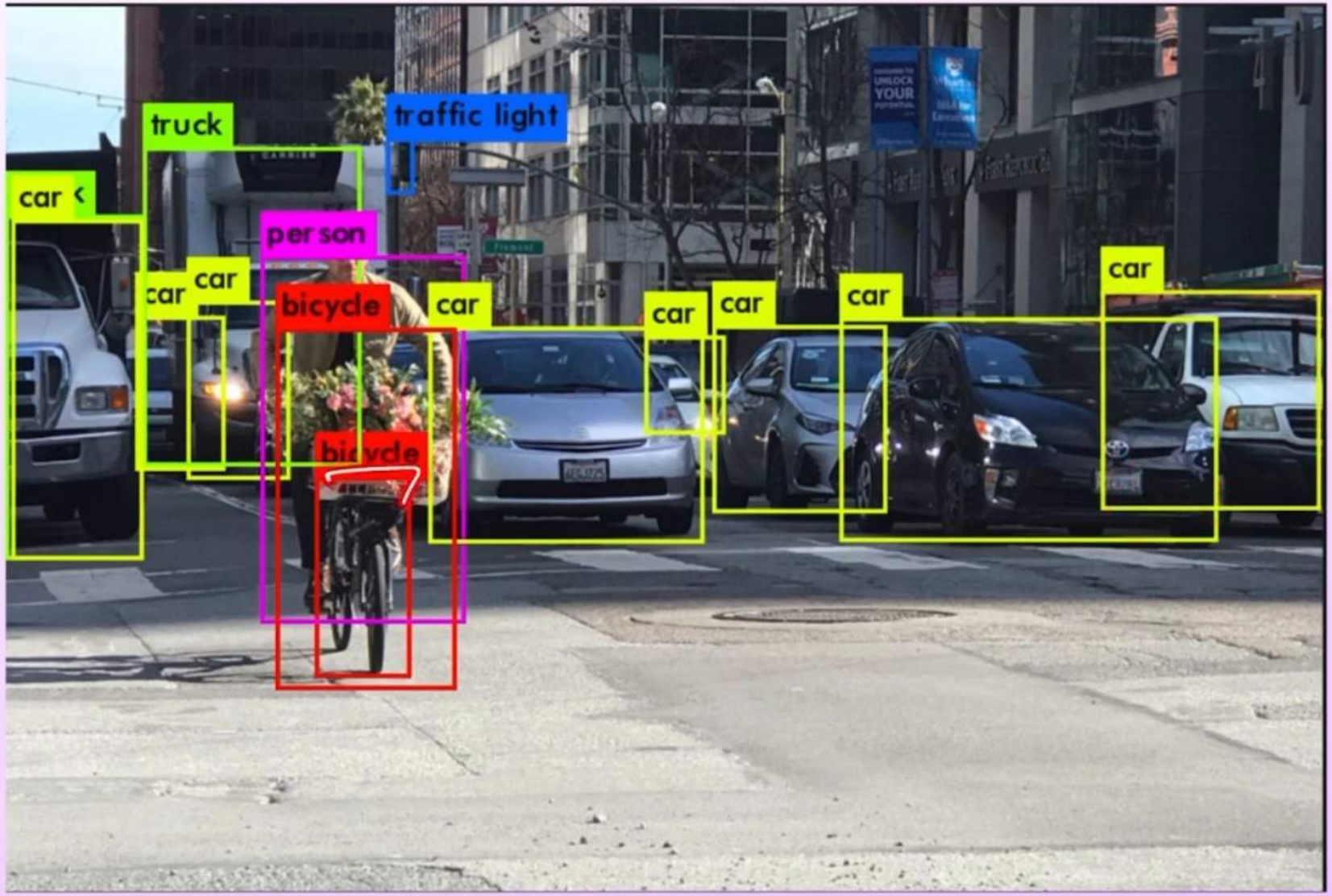
University of Washington*, Allen Institute for AI†

http://pjreddie.com/yolo9000/

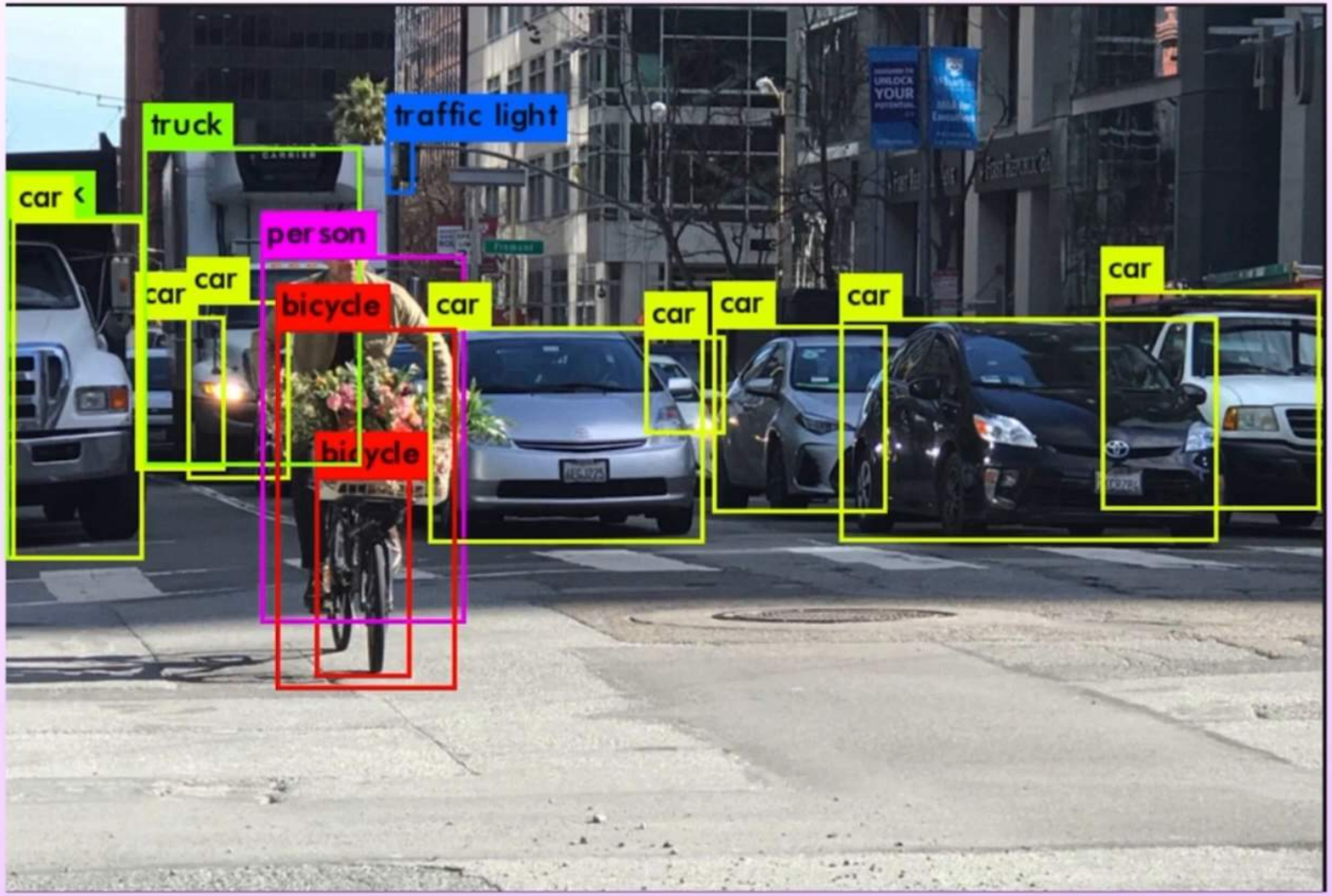
Yolo-V1 - Low Accuracy

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

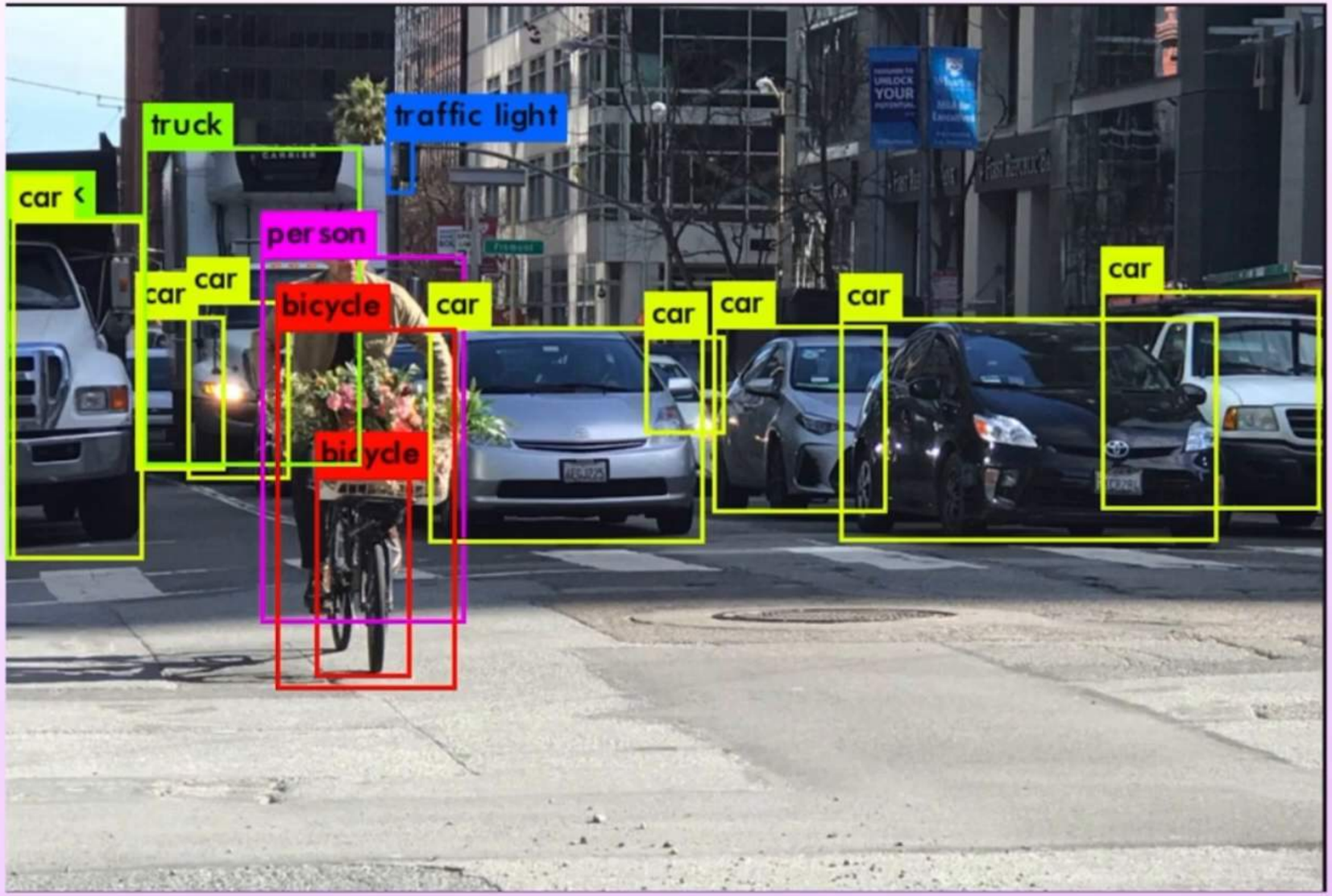
Yolo-V1 - Poor localization



Yolo-V1 - Poor localization



Yolo-V1 - Poor localization



Yolo-V1 vs SSD vs Faster-RCNN

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Authors

Joseph Redmon

Computer scientist

Ali Farhadi - Professor @ University of Washington -...

Vis

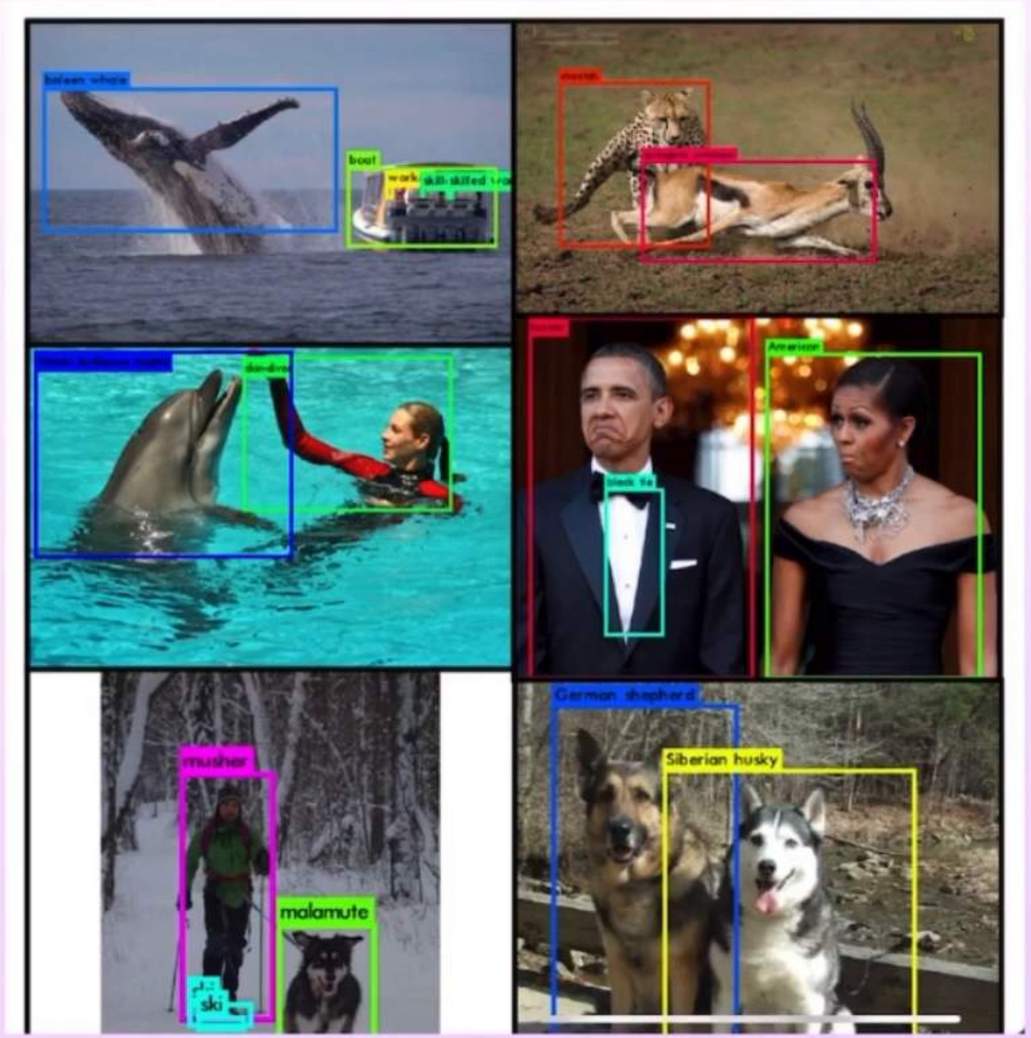
Modifications of yolo-v2

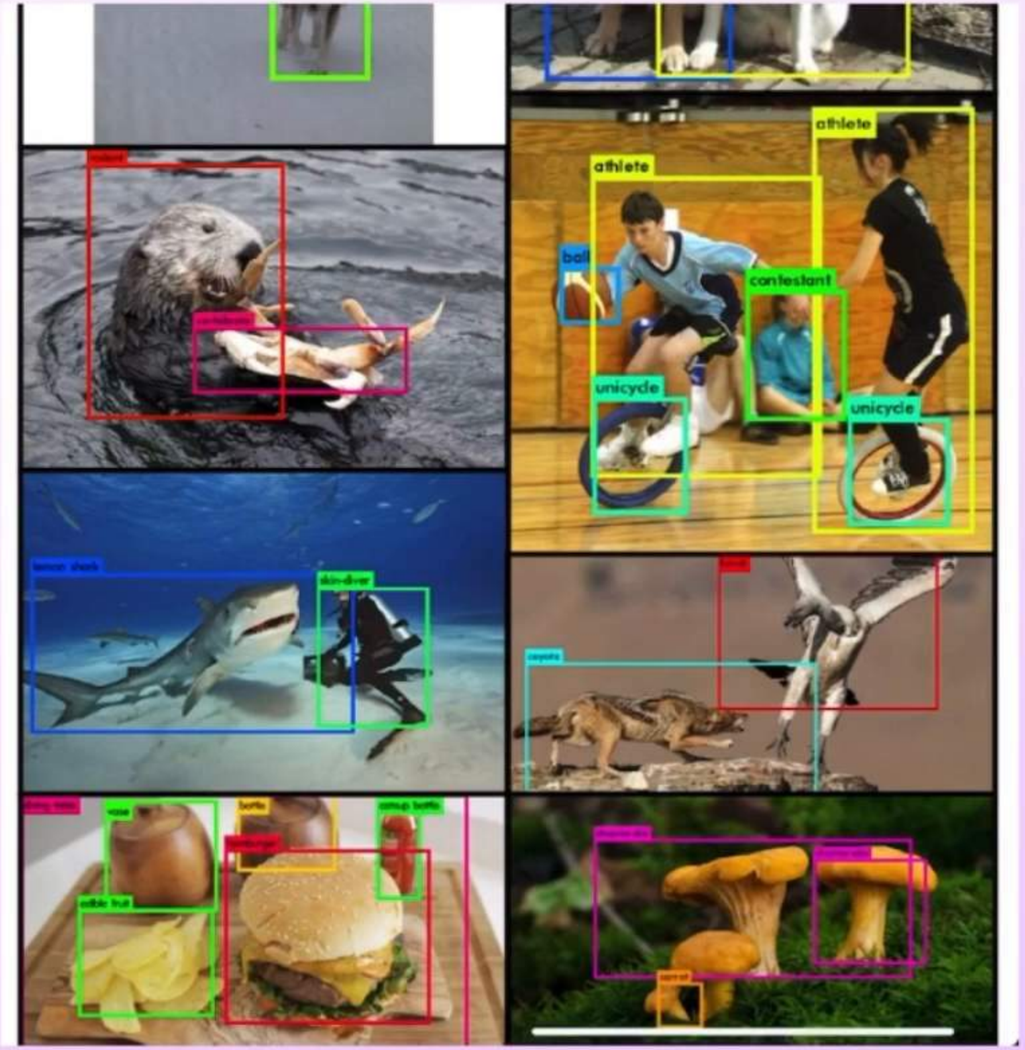
	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Darknet-19 backbone

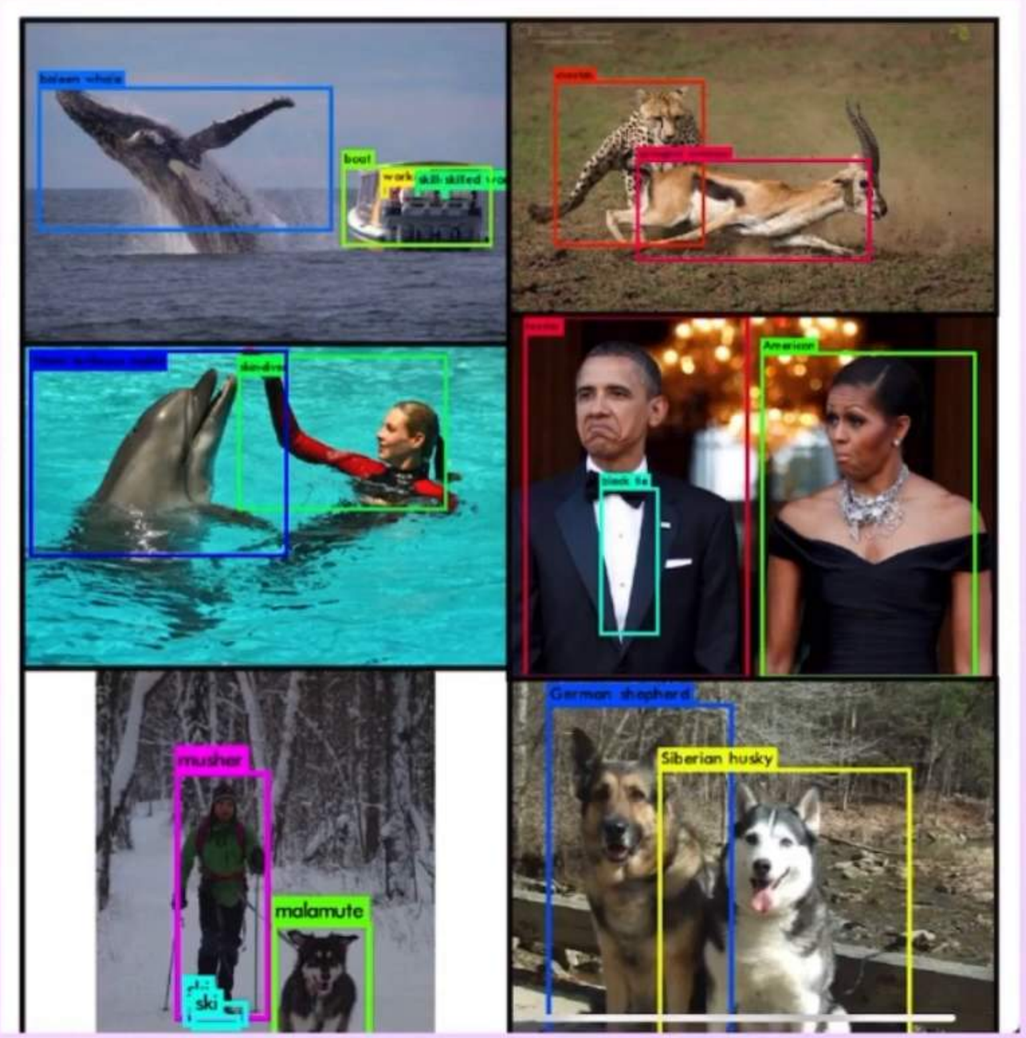
	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

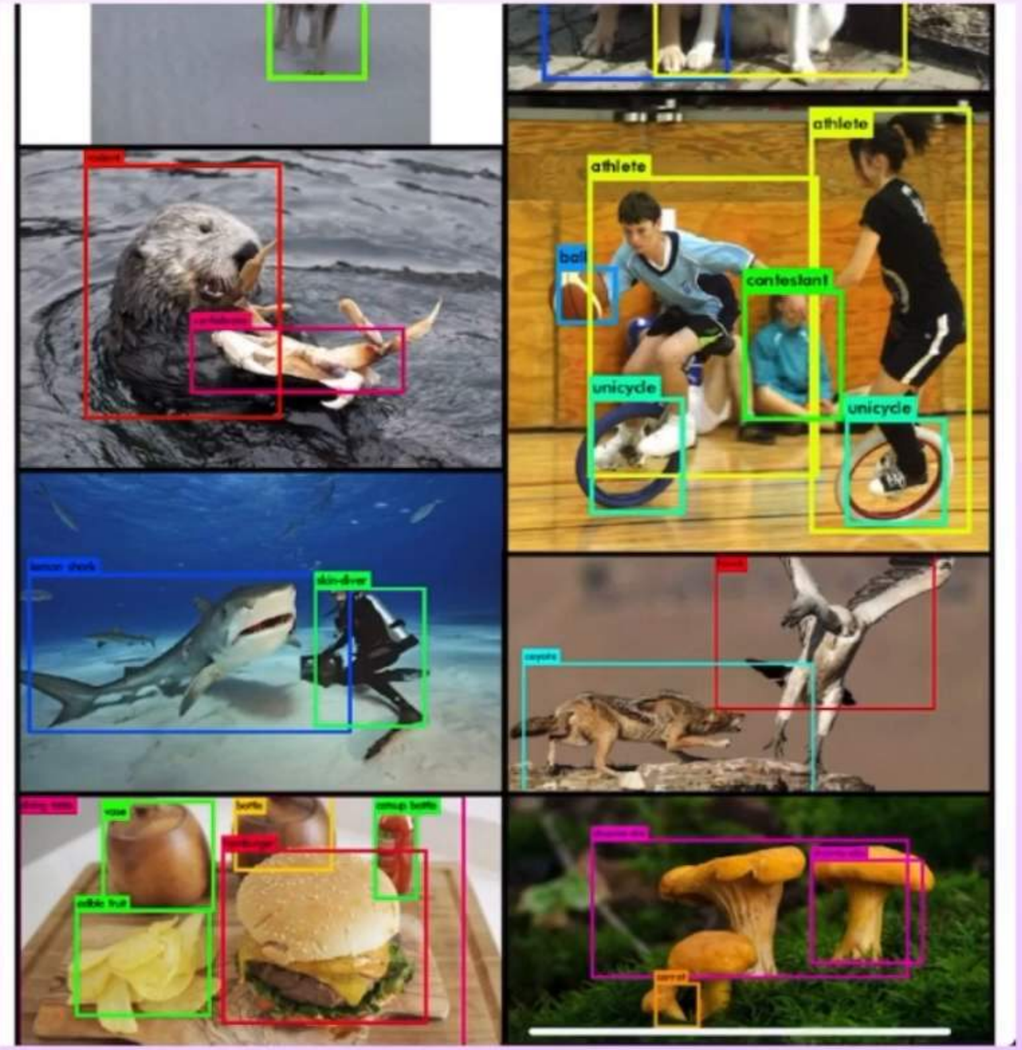
Yolo9000


boat
musher
malamute
ski
Siberian husky

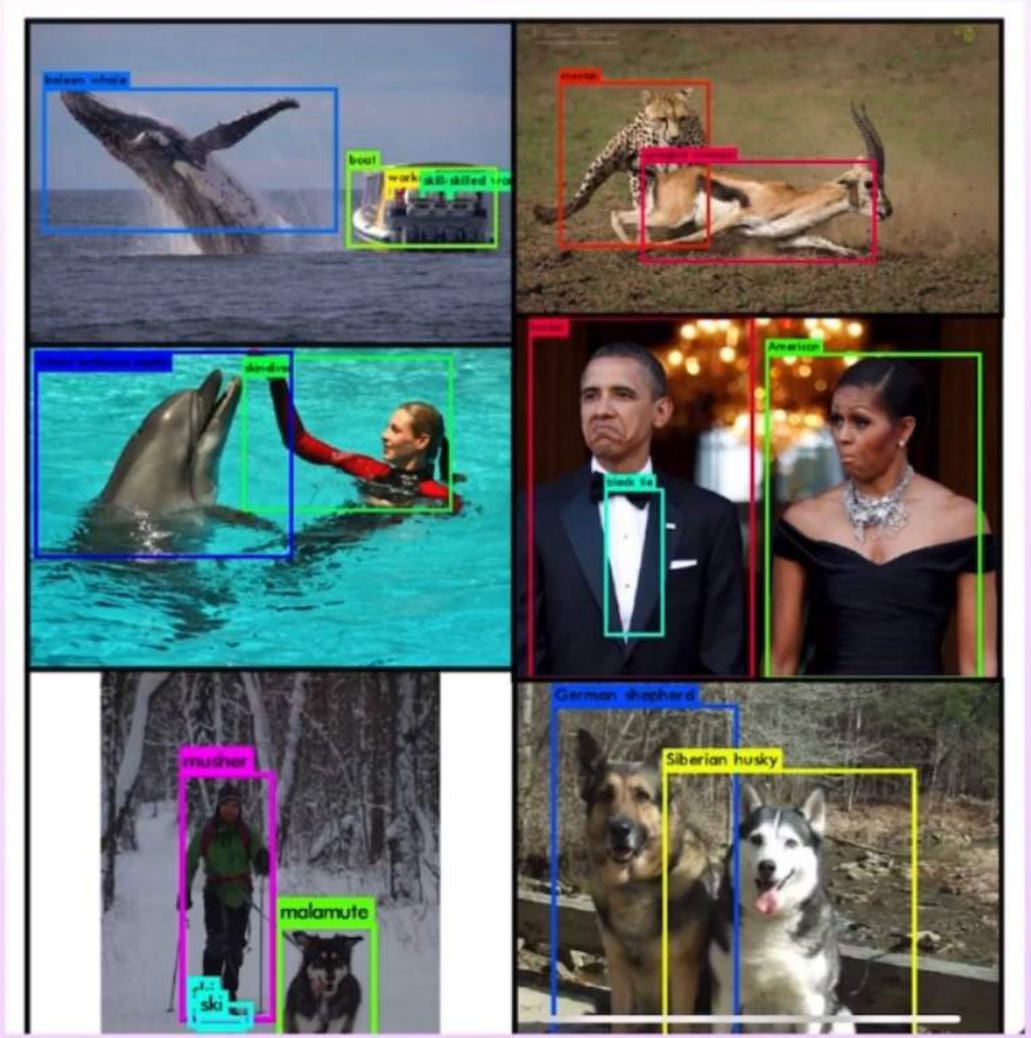

athlete
athlete
ball
contestant
unicycle
unicycle
skin-diver

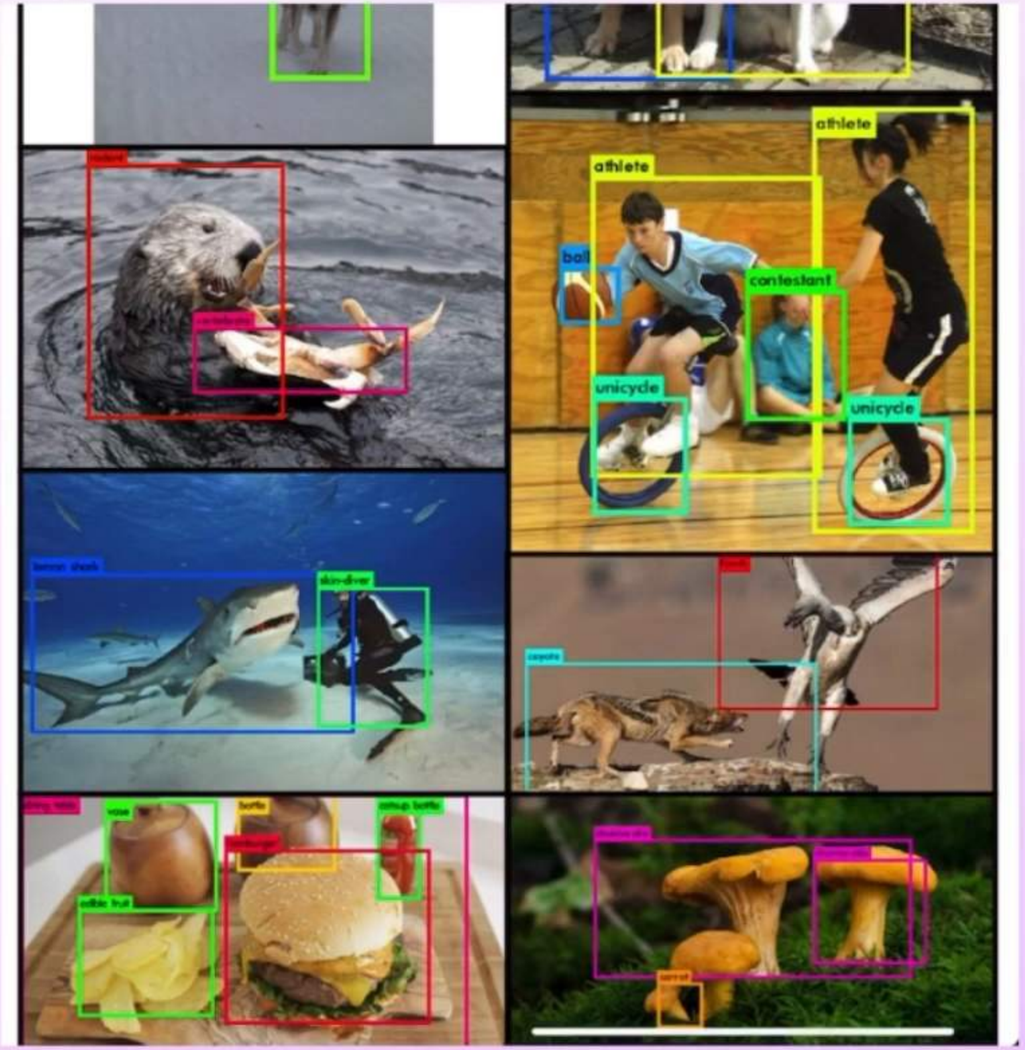
Yolo9000


boat
American
musher
malamute
ski
German shepherd
Siberian husky


athlete
athlete
ball
contestant
unicycle
unicycle
skin-diver
vase
bottle

Yolo9000

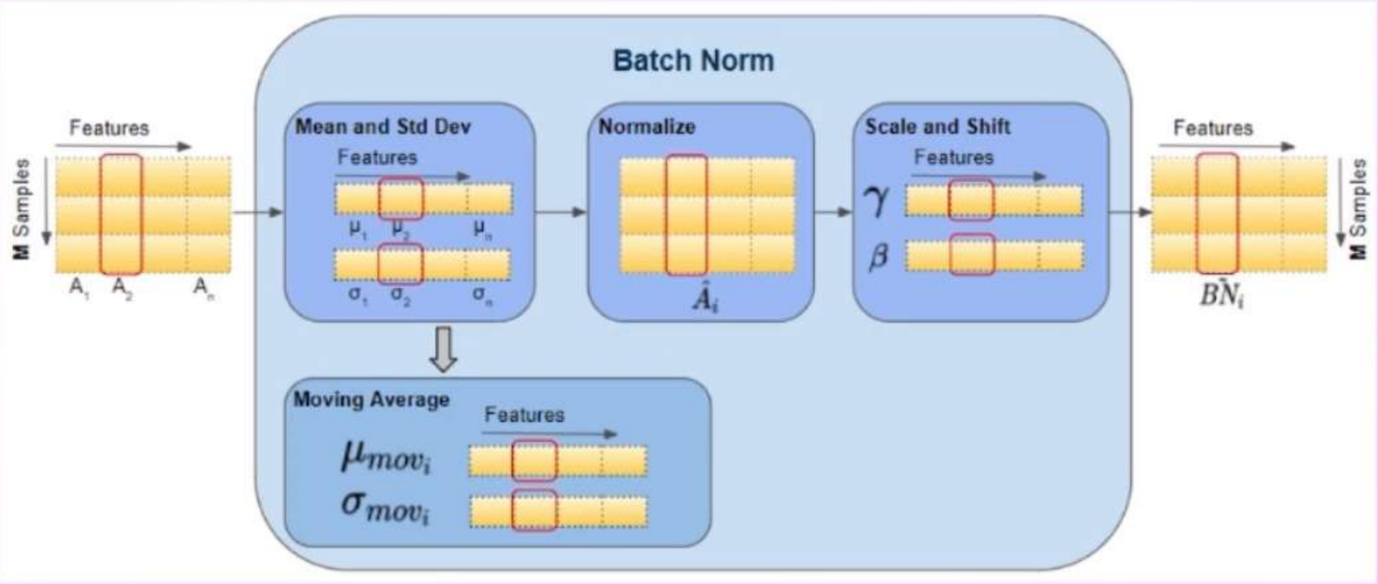

boat
American
musher
malamute
German shepherd
Siberian husky


athlete
athlete
ball
contestant
unicycle
unicycle
skis-diver

Batch Normalization

Applied batch normalization to all convolution layers

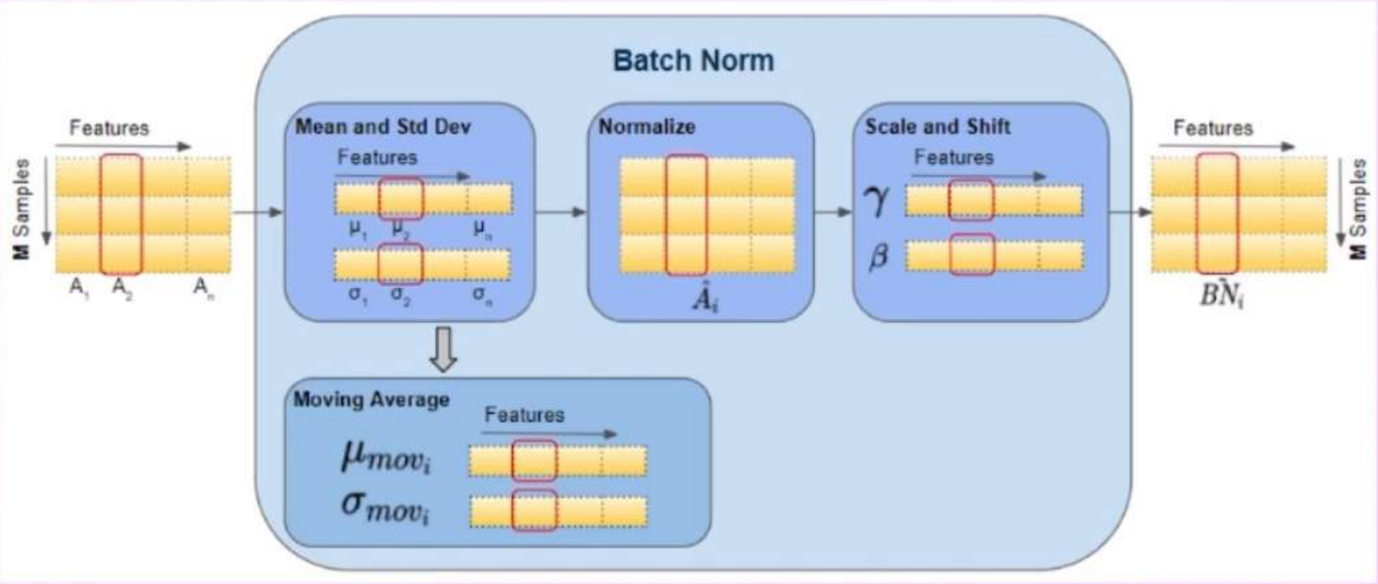
- Improved mAP by 2%
- Helped the model to avoid over fitting
- Regularization effect - removed dropout layers



Batch Normalization

Applied batch normalization to all convolution layers

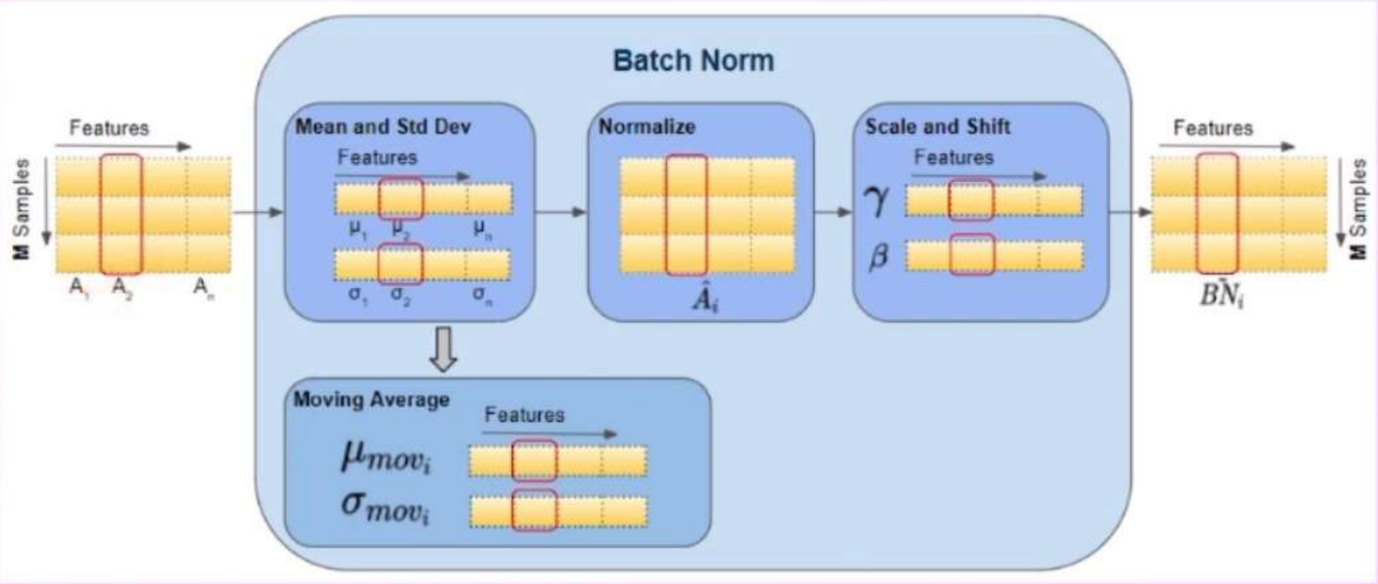
- Improved mAP by 2%
- Helped the model to avoid over fitting
- Regularization effect - removed dropout layers



Batch Normalization

Applied batch normalization to all convolution layers

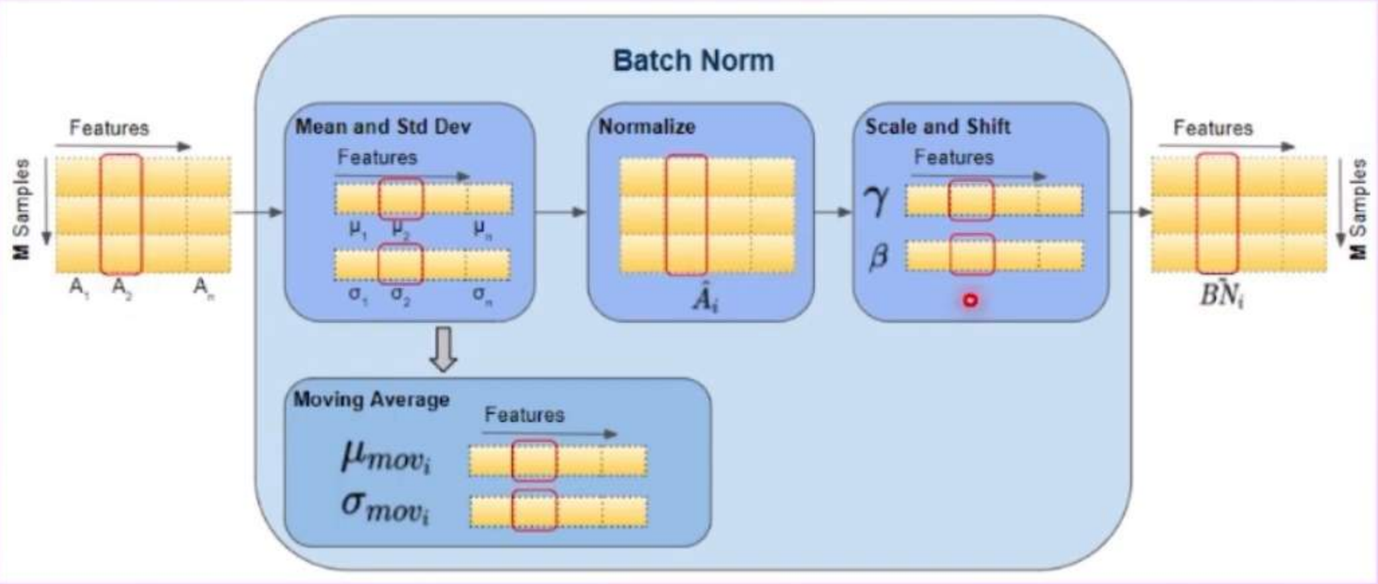
- Improved mAP by 2%
- Helped the model to avoid over fitting
- Regularization effect - removed dropout layers



Batch Normalization

Applied batch normalization to all convolution layers

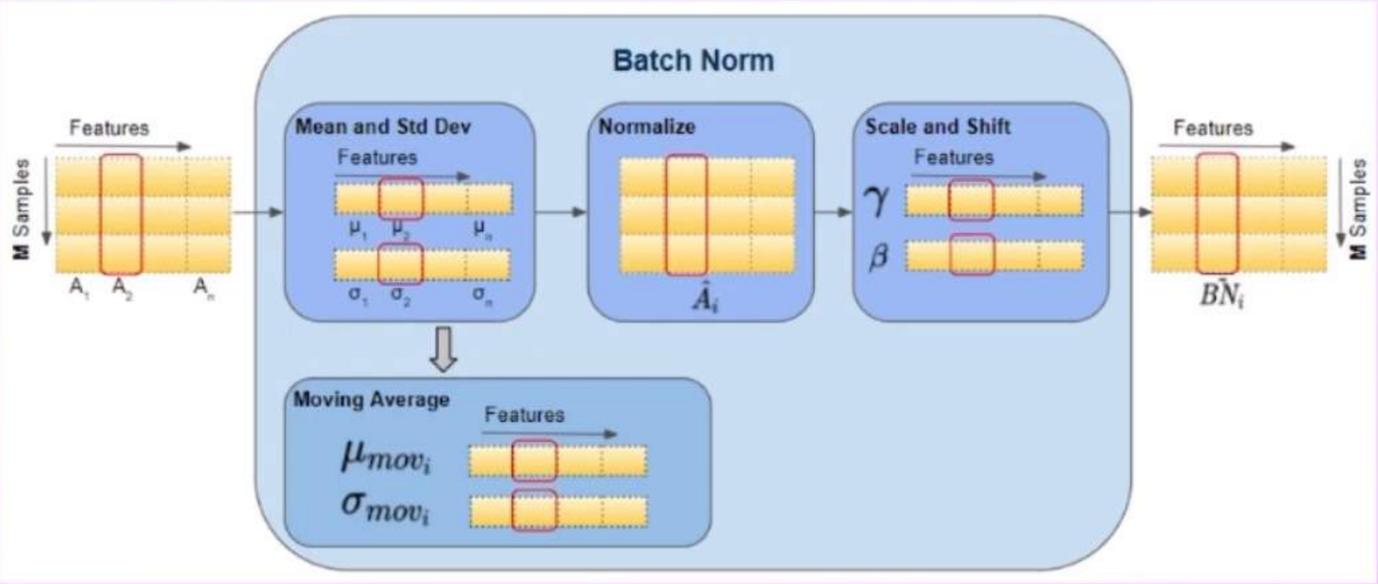
- Improved mAP by 2%
- Helped the model to avoid over fitting
- Regularization effect - removed dropout layers



Batch Normalization

Applied batch normalization to all convolution layers

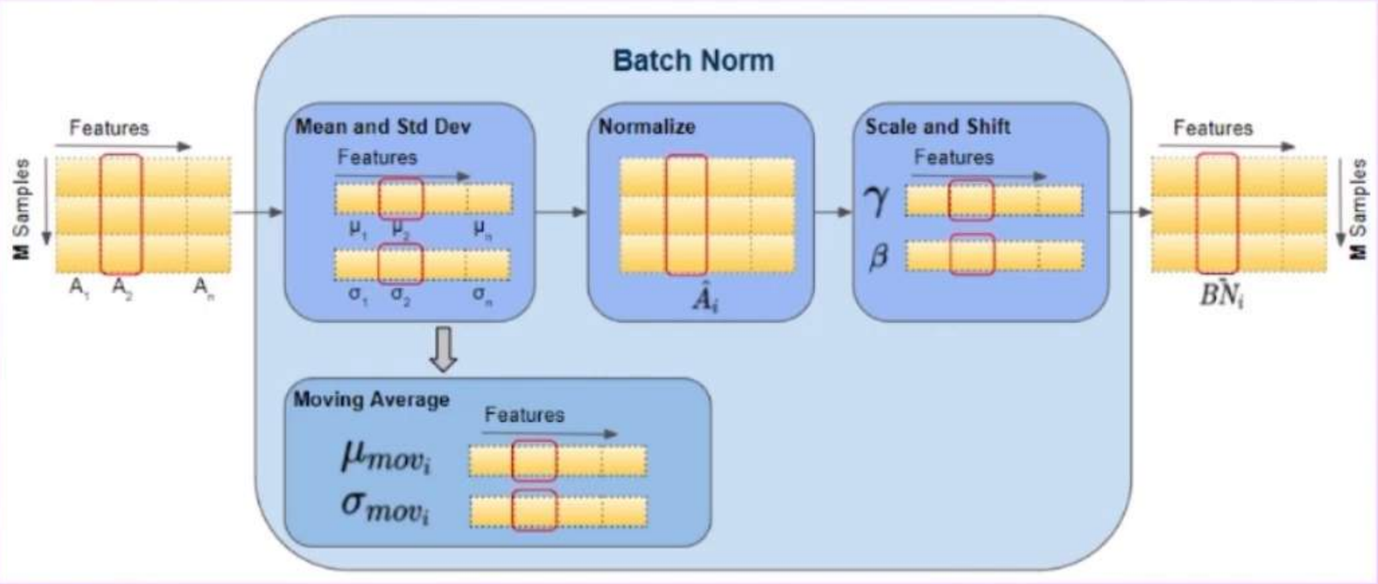
- Improved mAP by 2%
- Helped the model to avoid over fitting
- Regularization effect - removed dropout layers



Batch Normalization

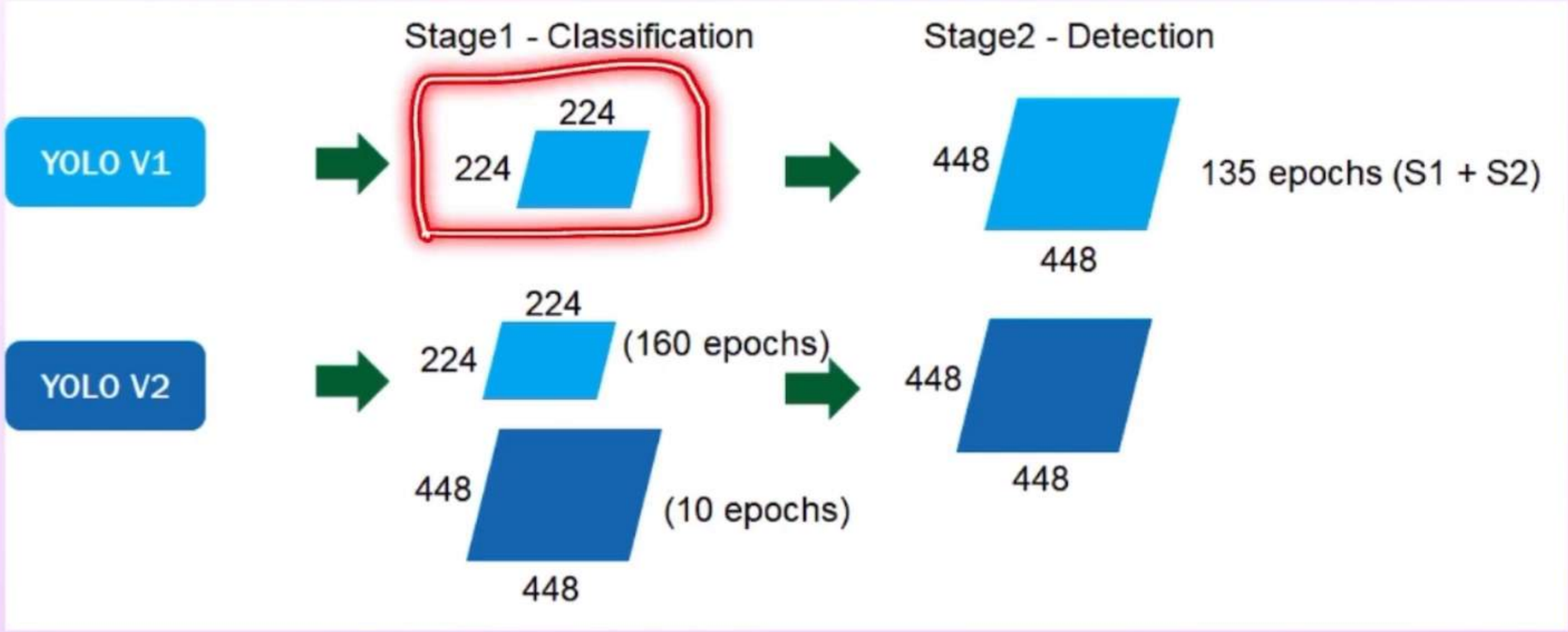
Applied batch normalization to all convolution layers

- Improved mAP by 2%
- Helped the model to avoid over fitting
- Regularization effect - removed dropout layers



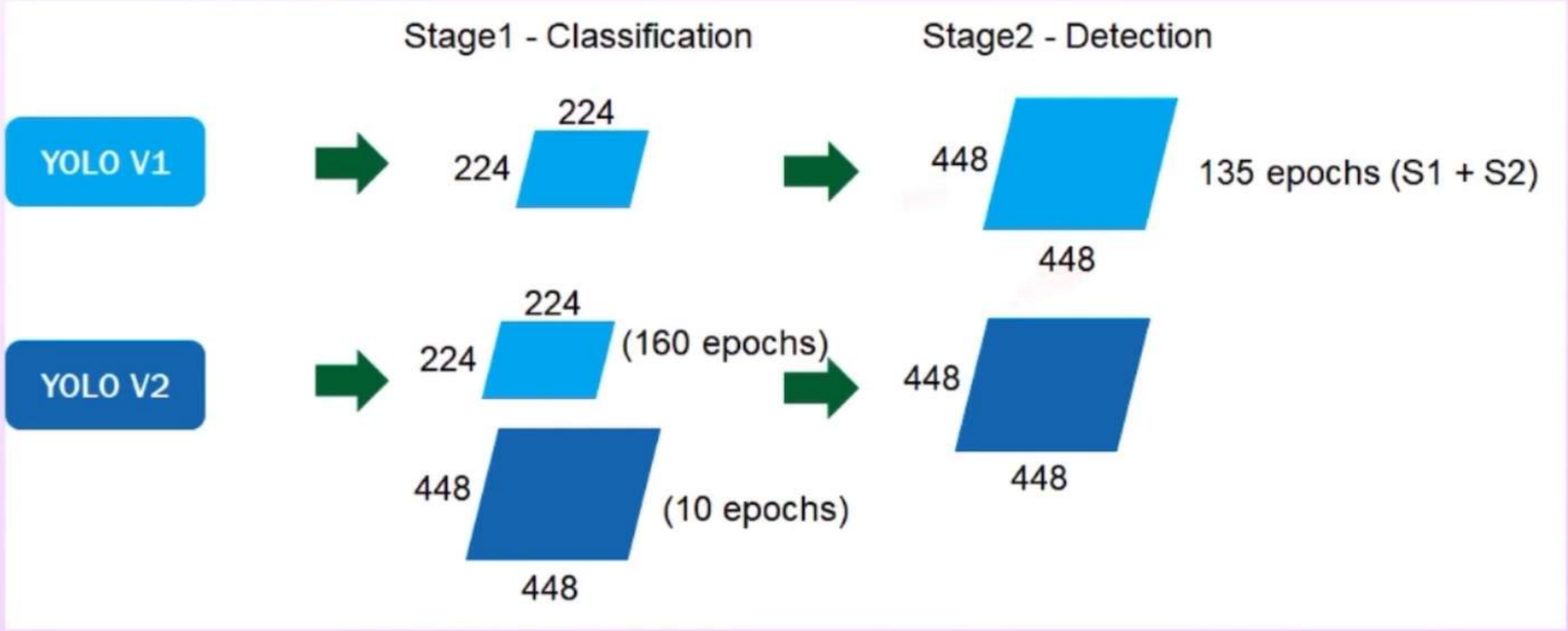
High Resolution Classifier

Improved the mAP by 4%



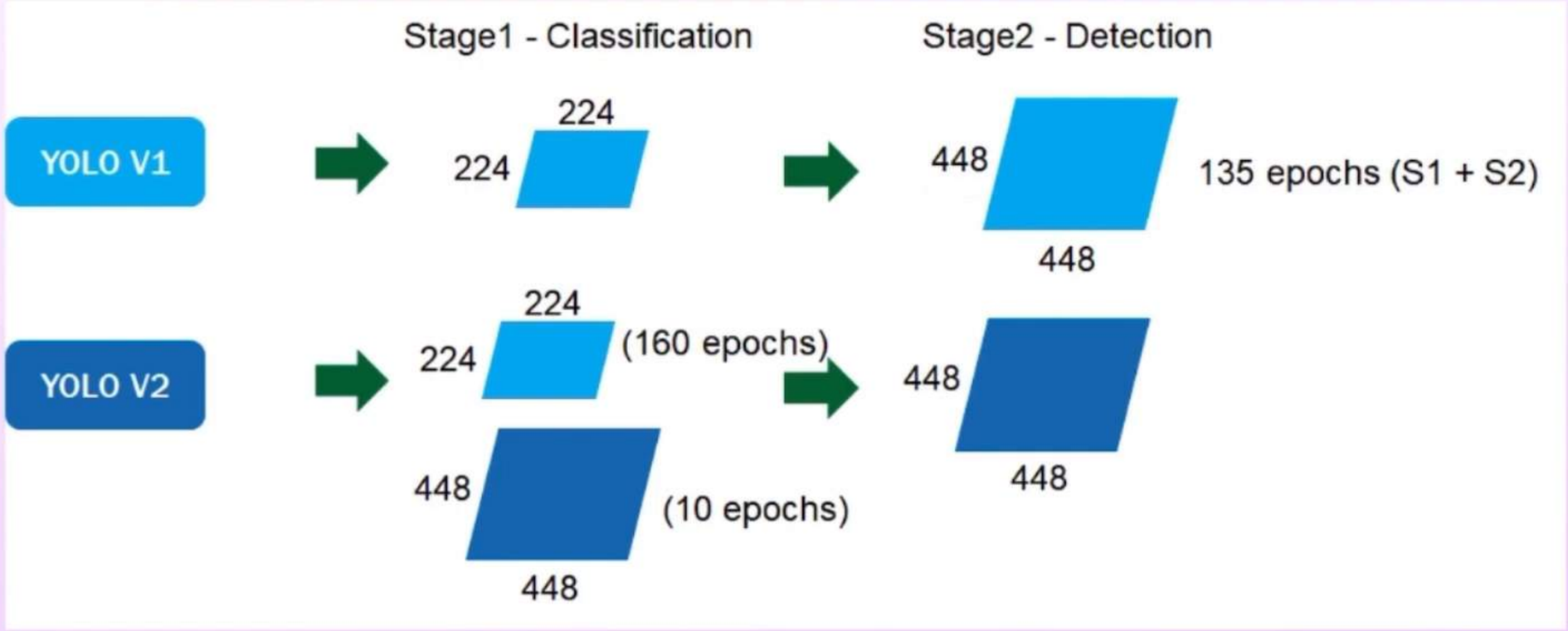
High Resolution Classifier

Improved the mAP by 4%



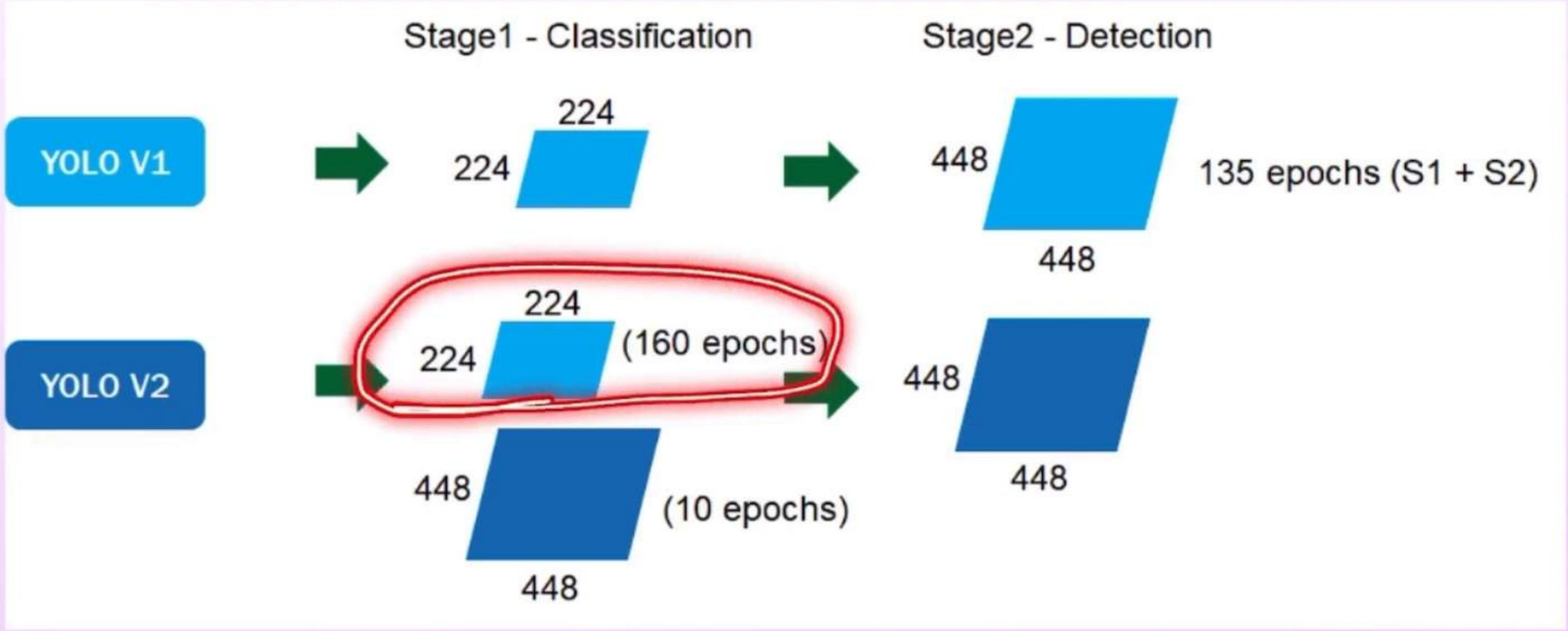
High Resolution Classifier

Improved the mAP by 4%



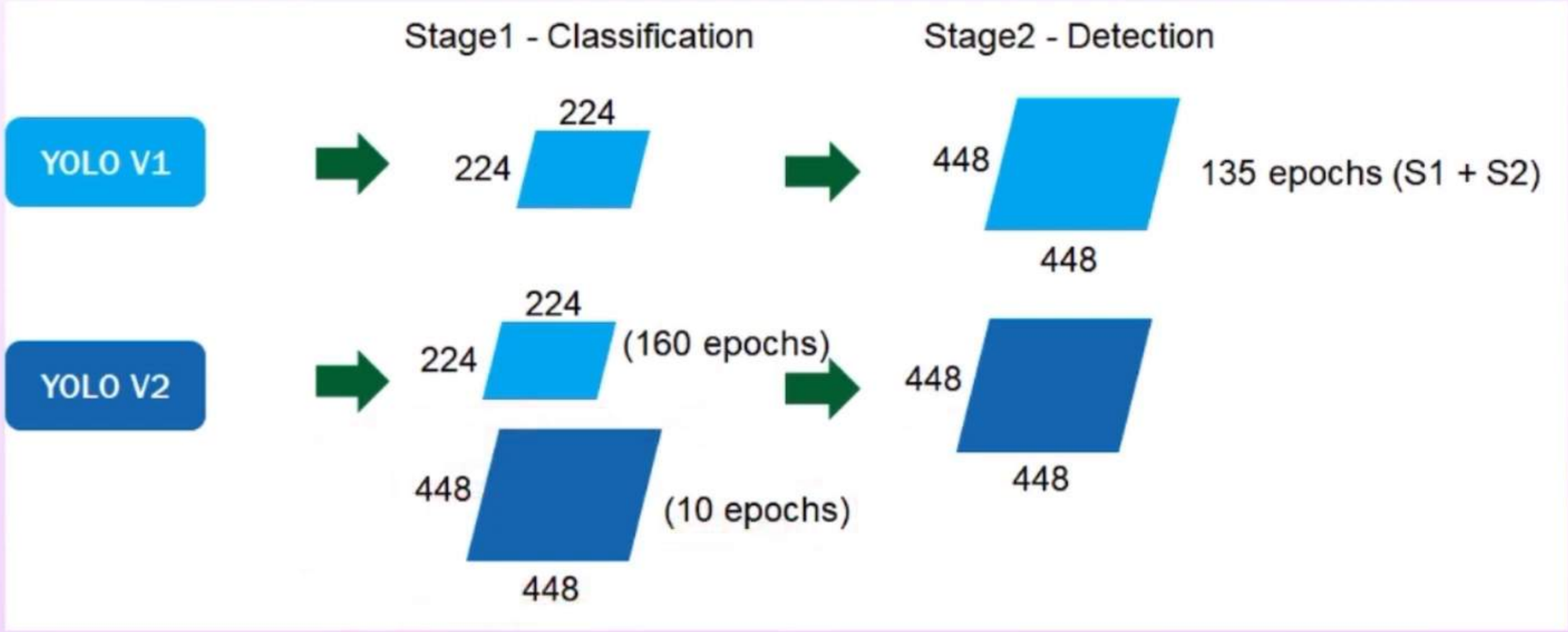
High Resolution Classifier

Improved the mAP by 4%



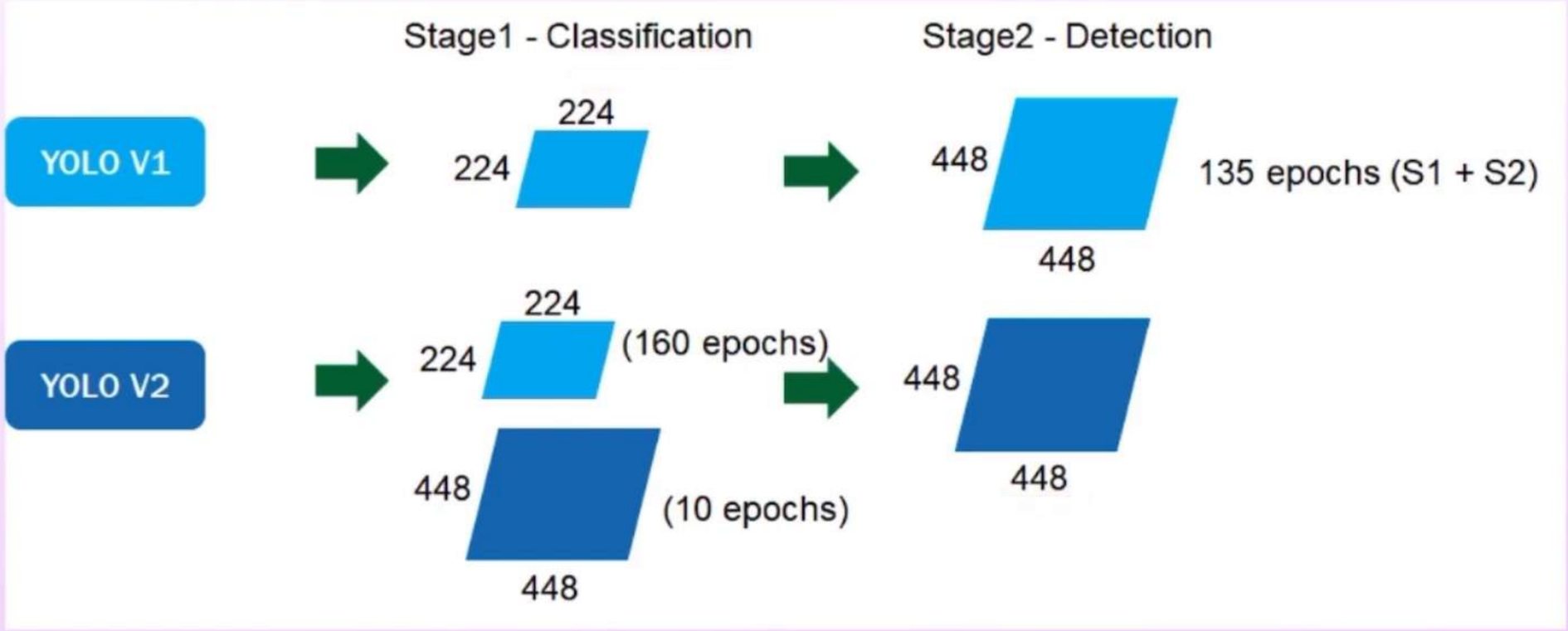
High Resolution Classifier

Improved the mAP by 4%



High Resolution Classifier

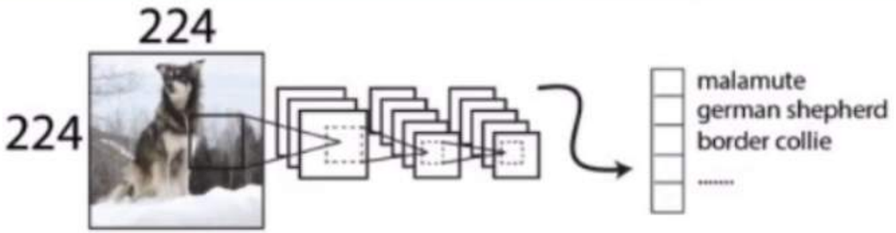
Improved the mAP by 4%



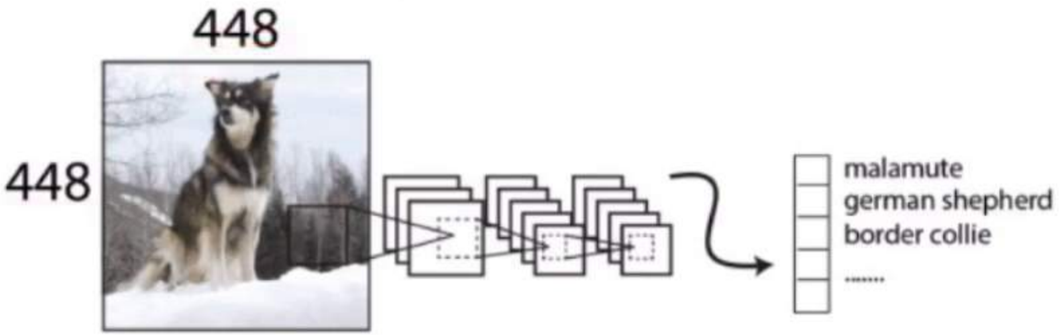
High Resolution Classifier

Improved the mAP by 4%

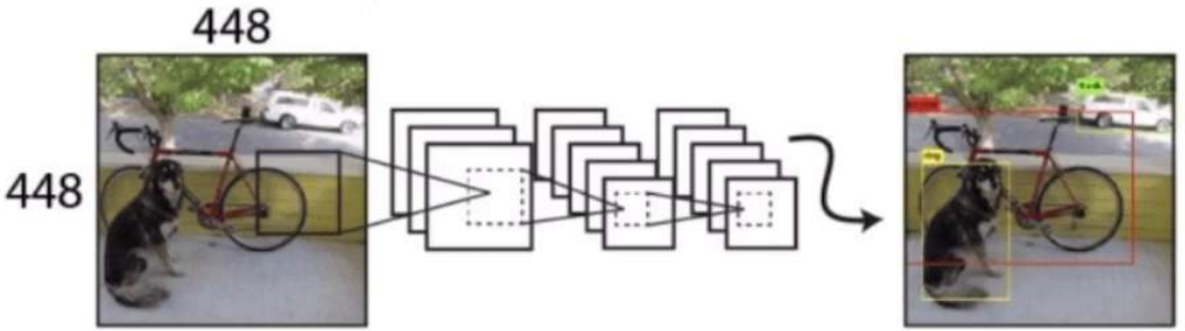
Train on ImageNet



Resize, fine-tune on ImageNet



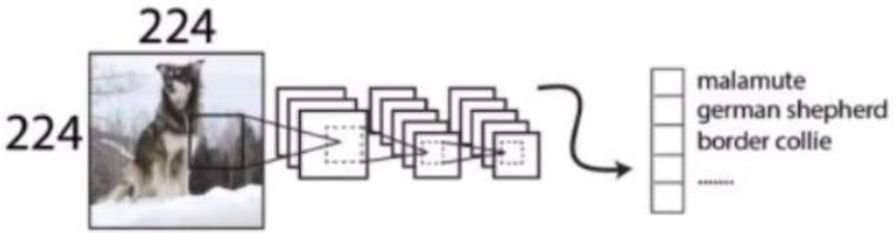
Fine-tune on detection



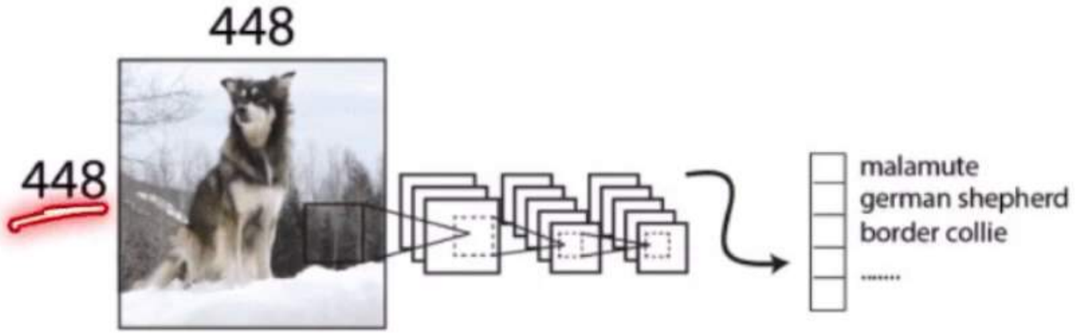
High Resolution Classifier

Improved the mAP by 4%

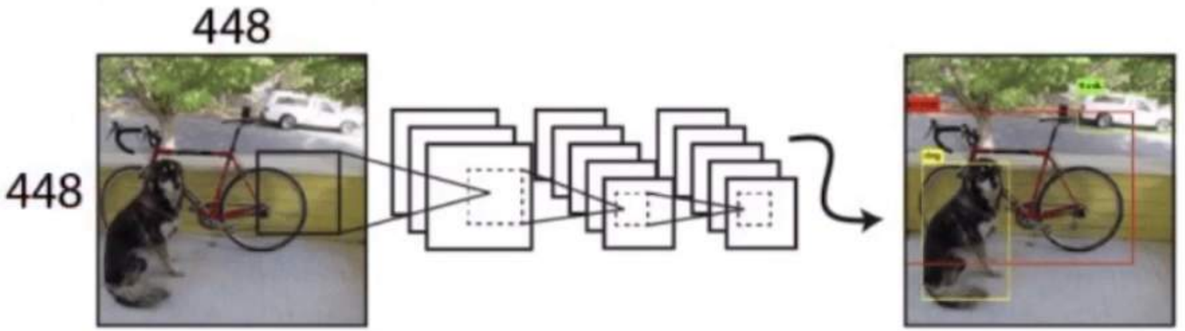
Train on ImageNet



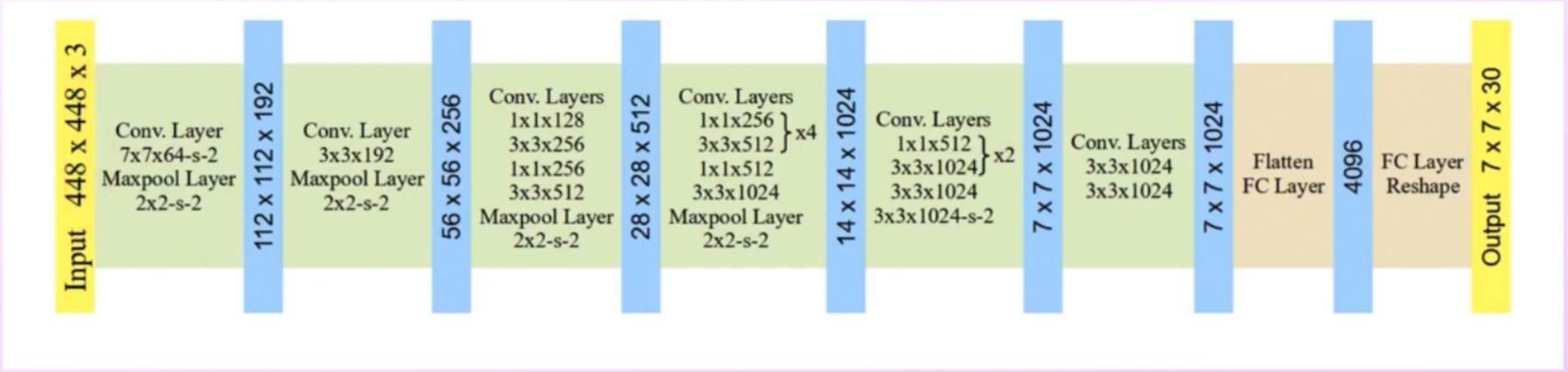
Resize, fine-tune on ImageNet



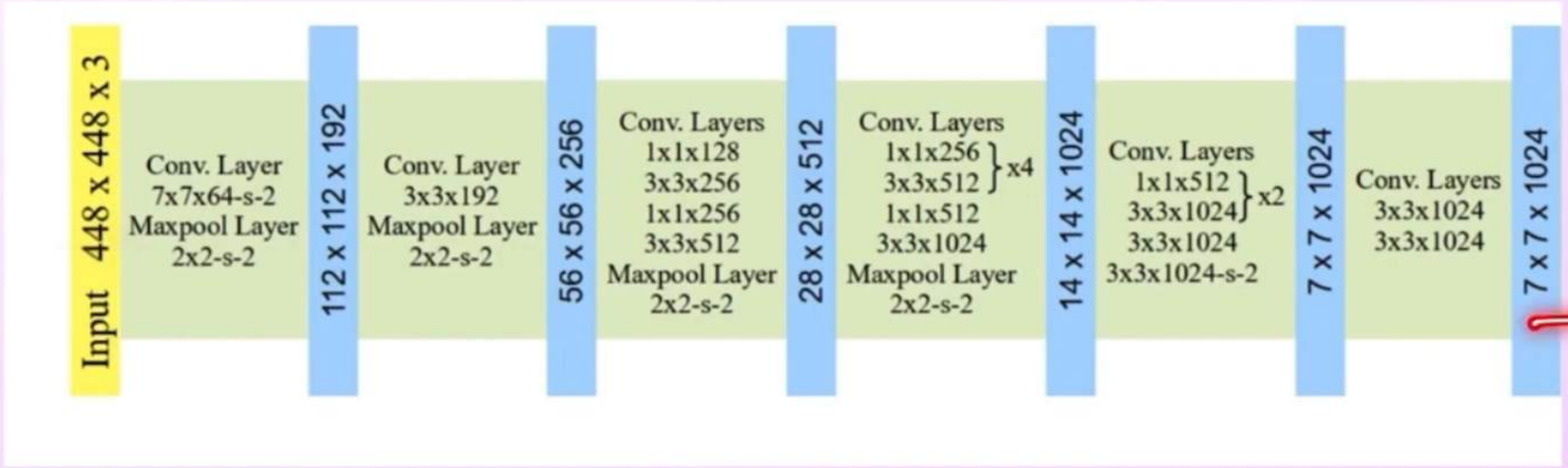
Fine-tune on detection



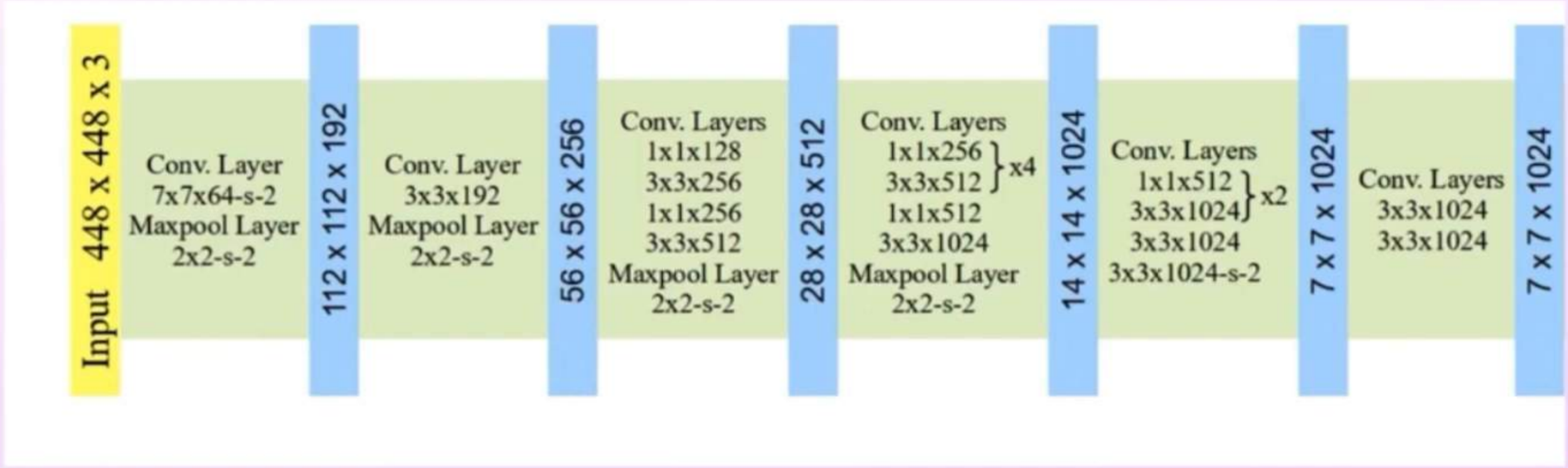
High Resolution Feature maps



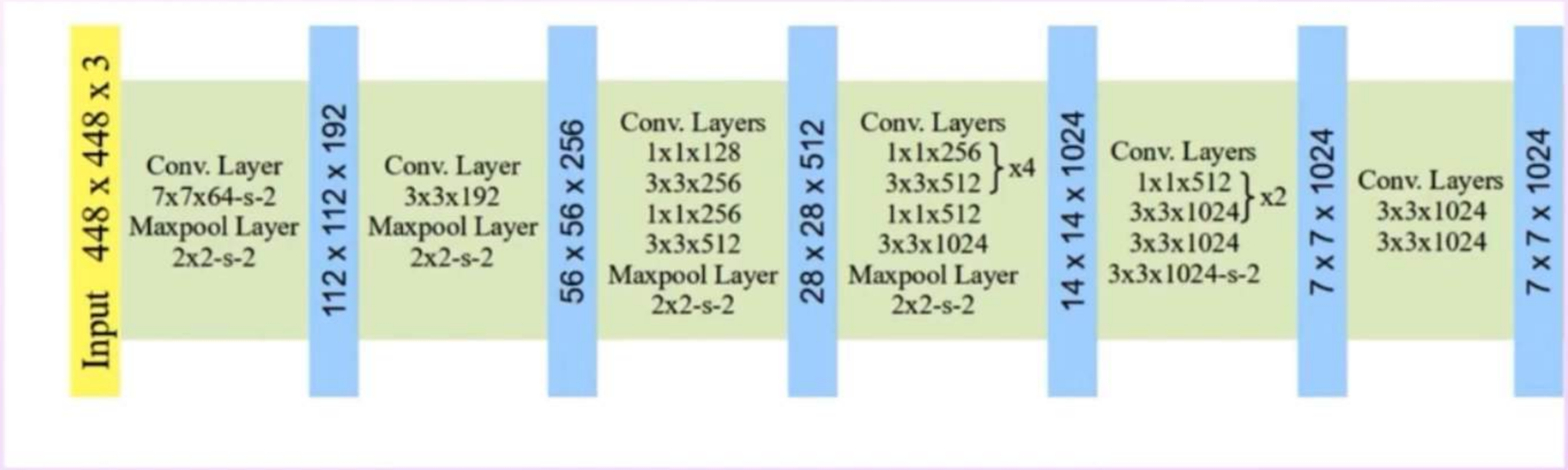
High Resolution Feature maps



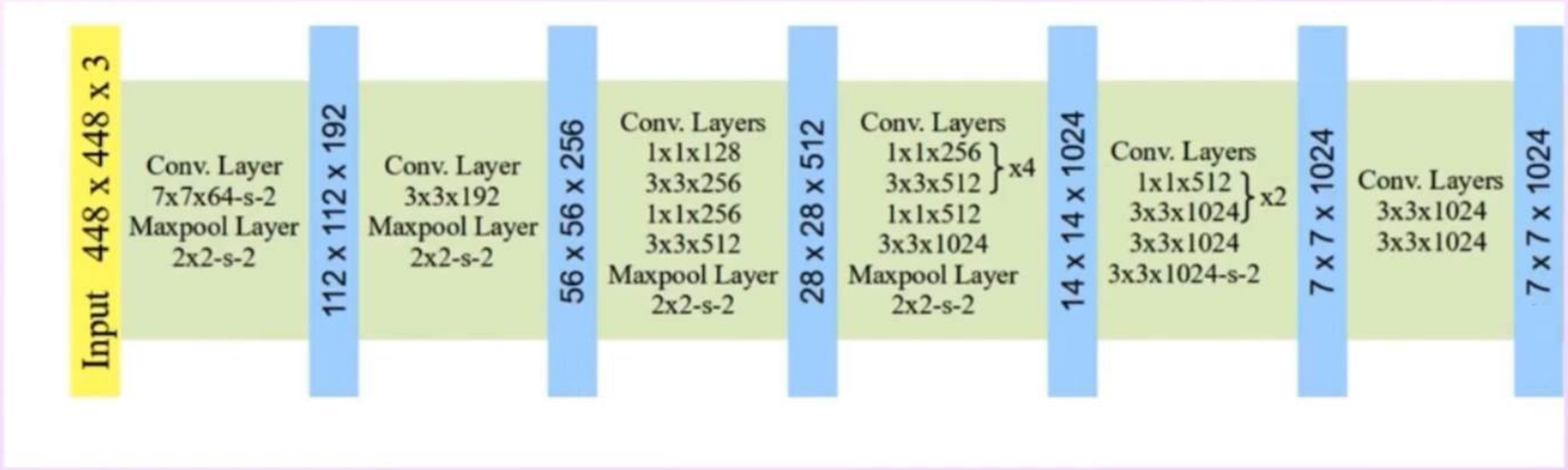
High Resolution Feature maps



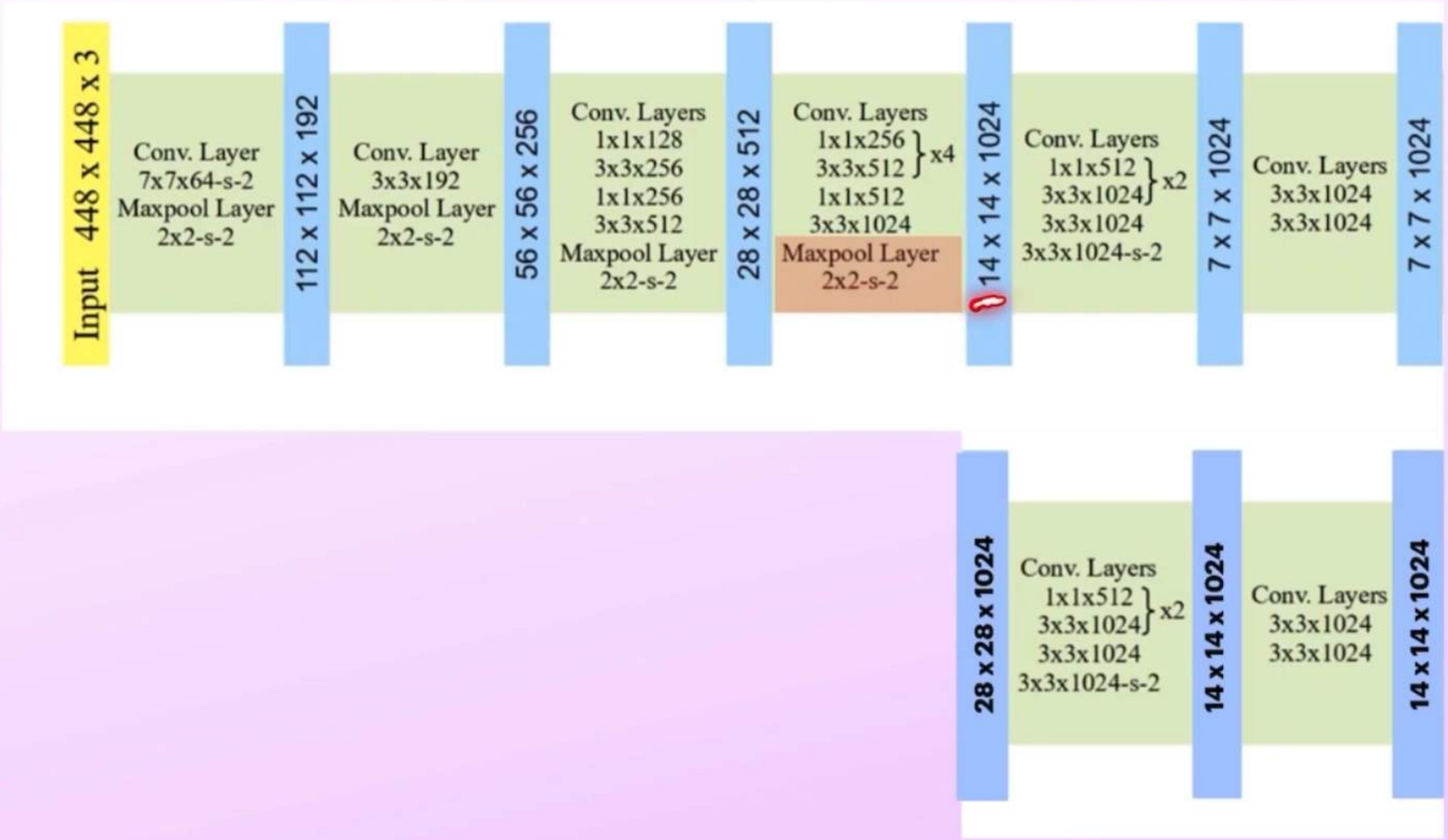
High Resolution Feature maps



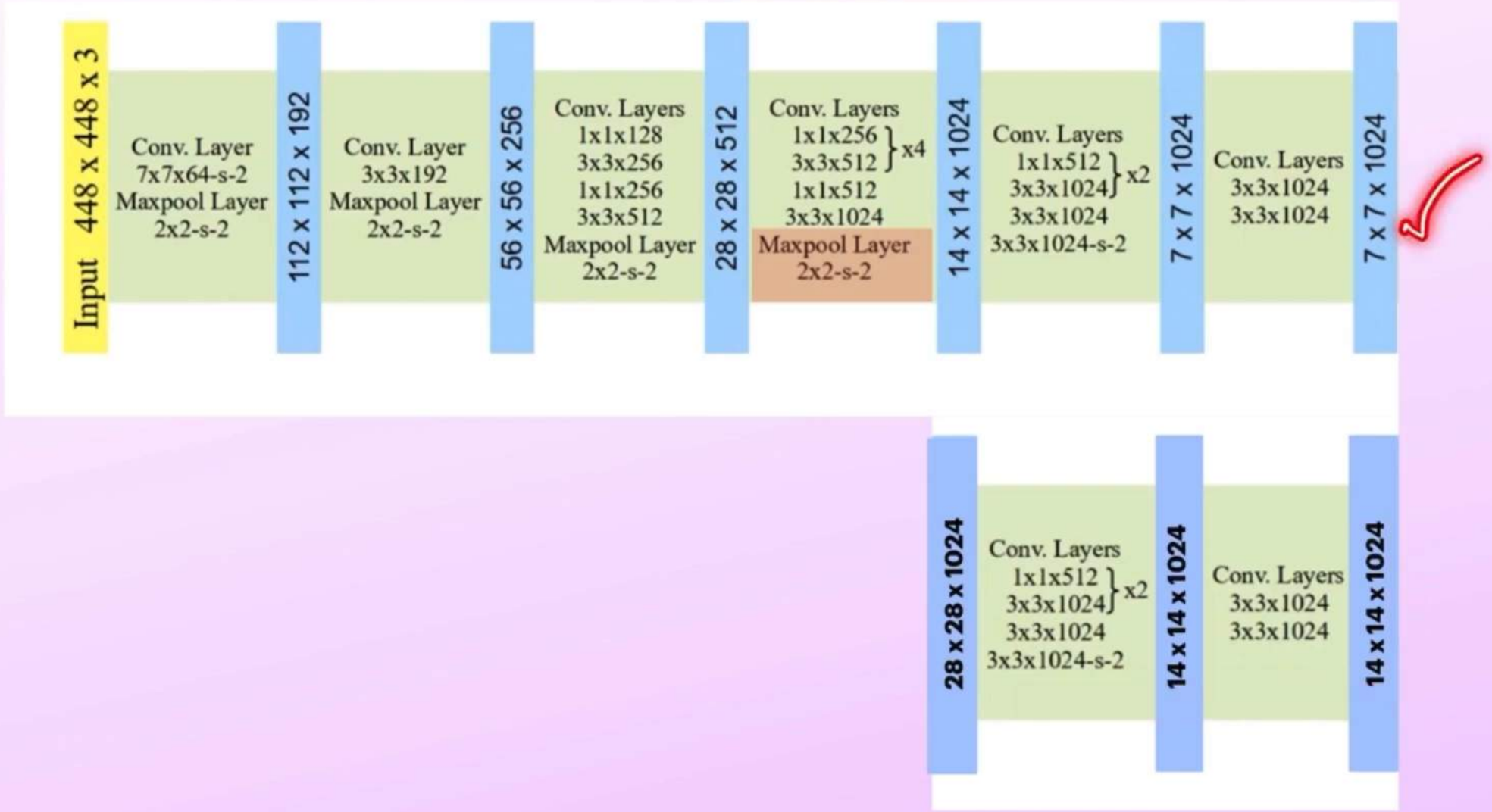
High Resolution Feature maps



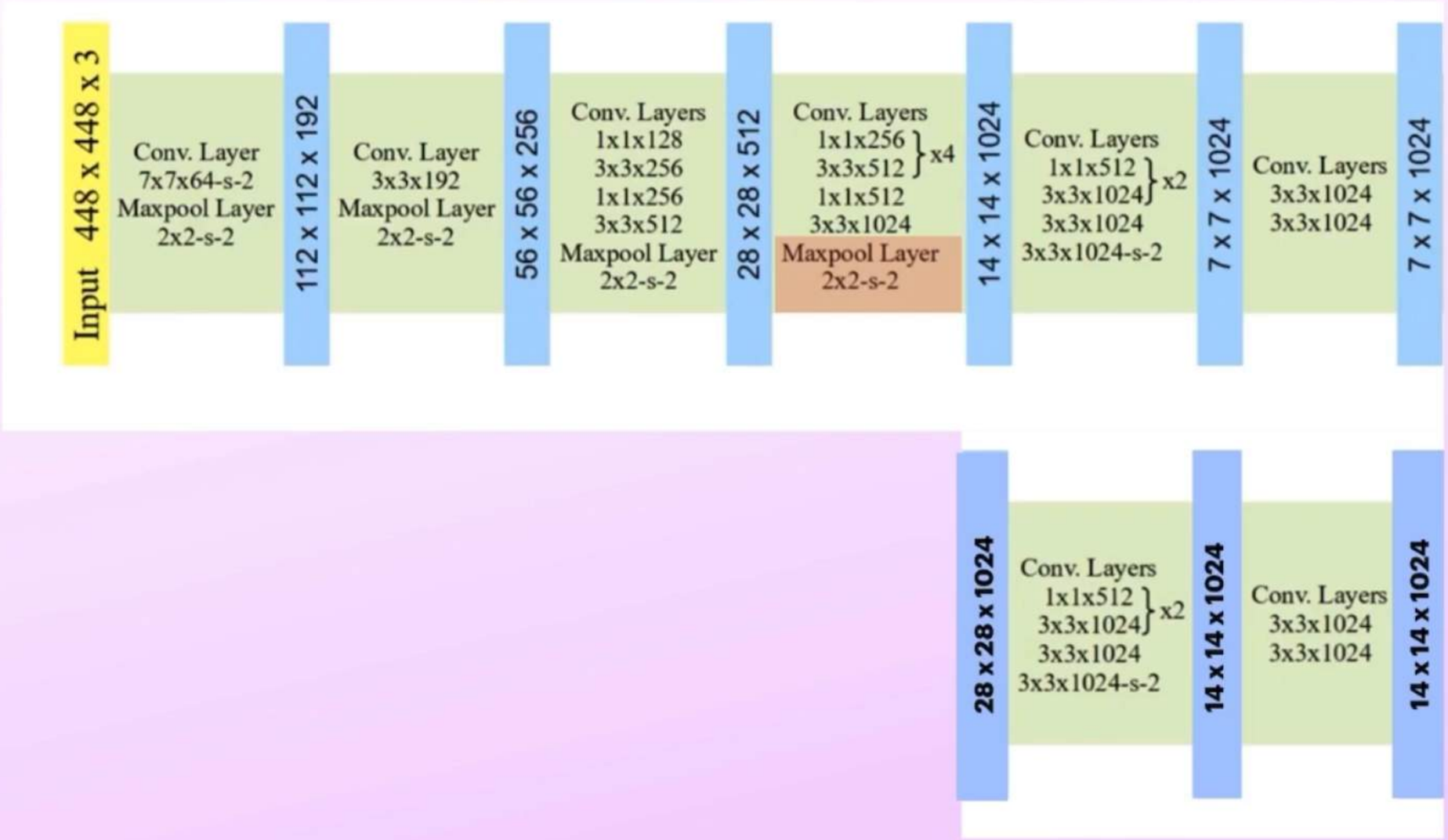
High Resolution Feature maps



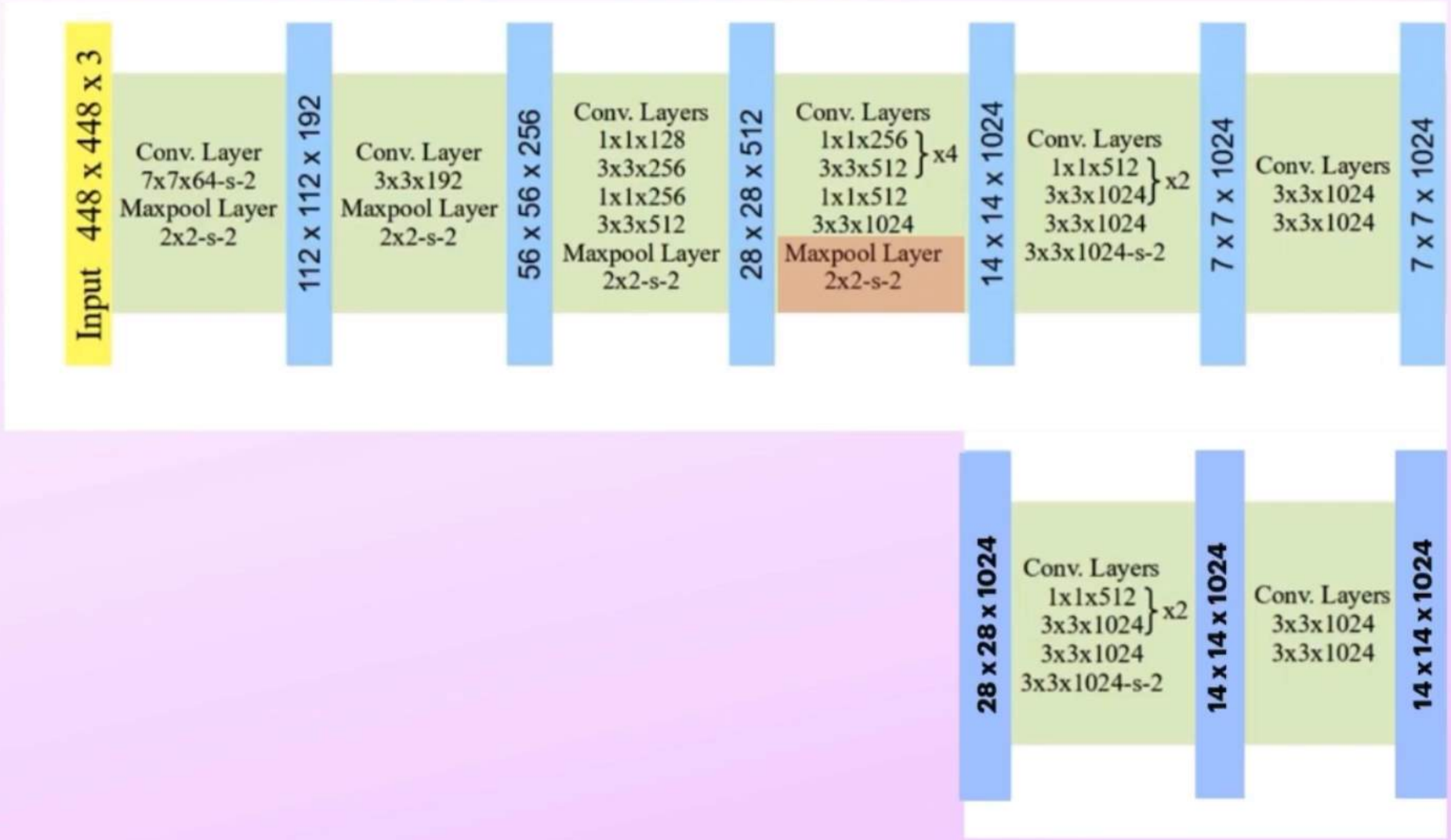
High Resolution Feature maps



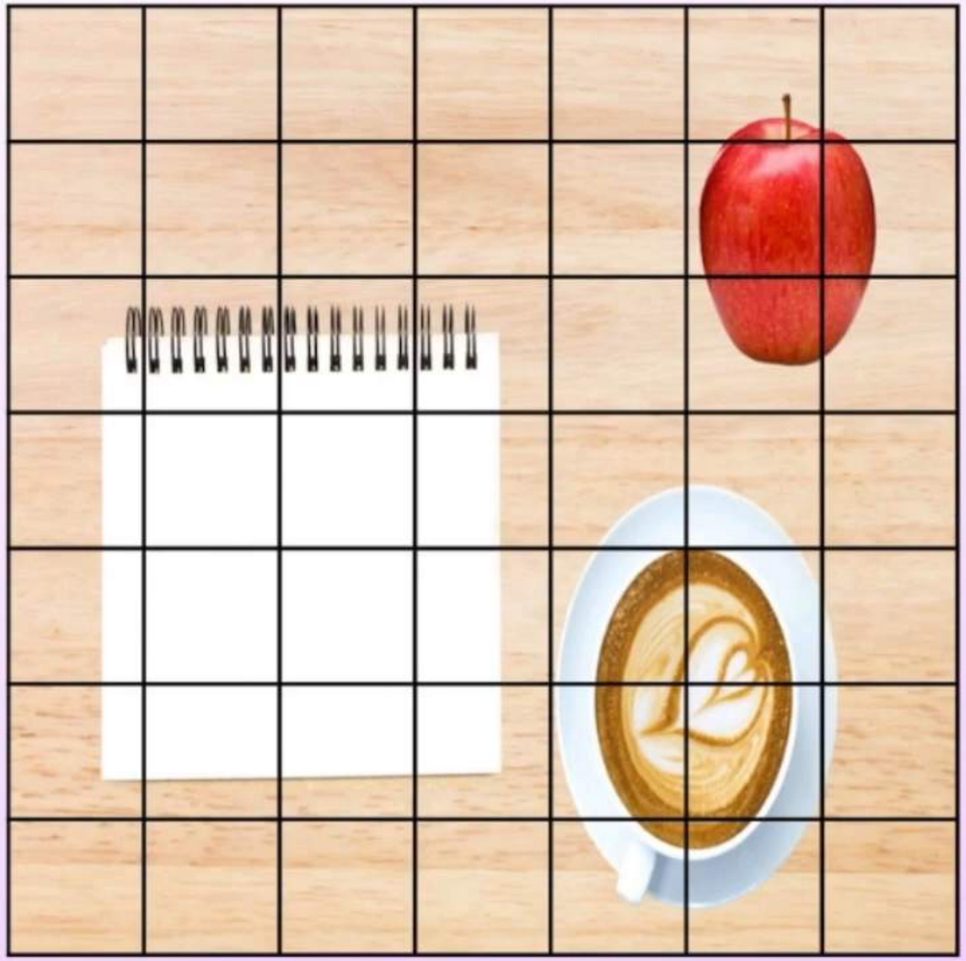
High Resolution Feature maps



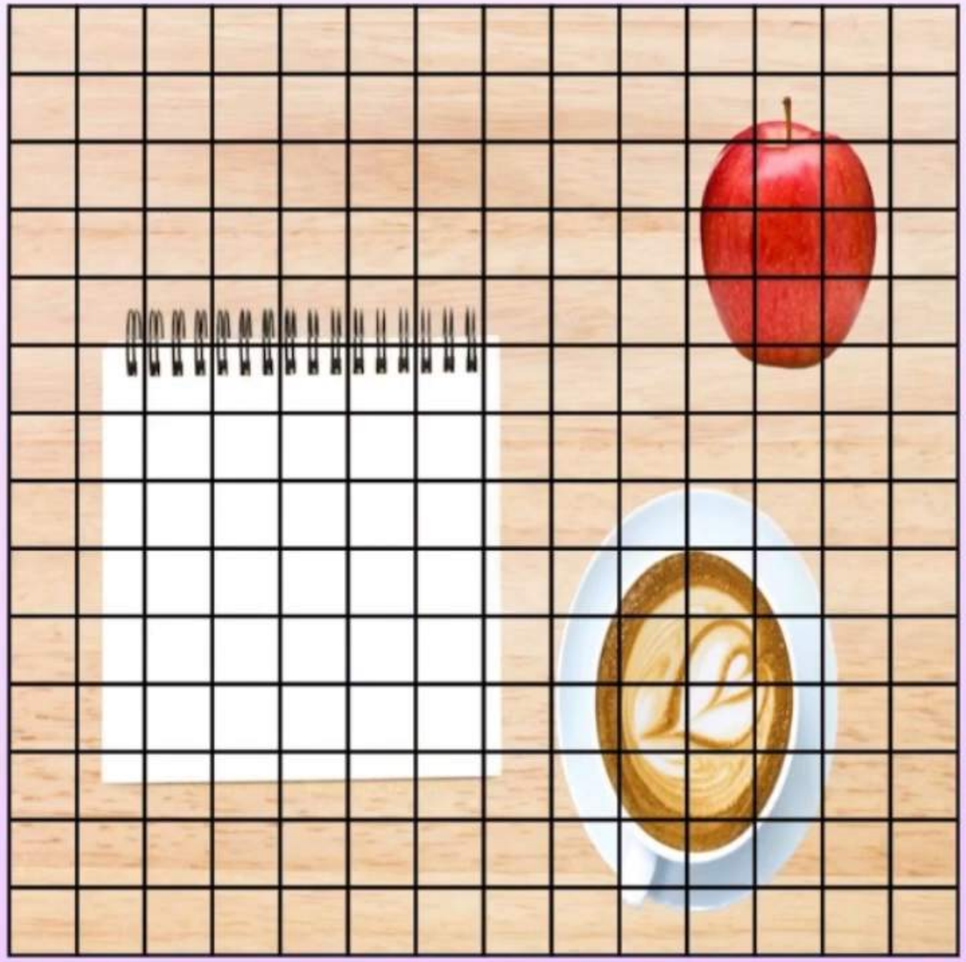
High Resolution Feature maps



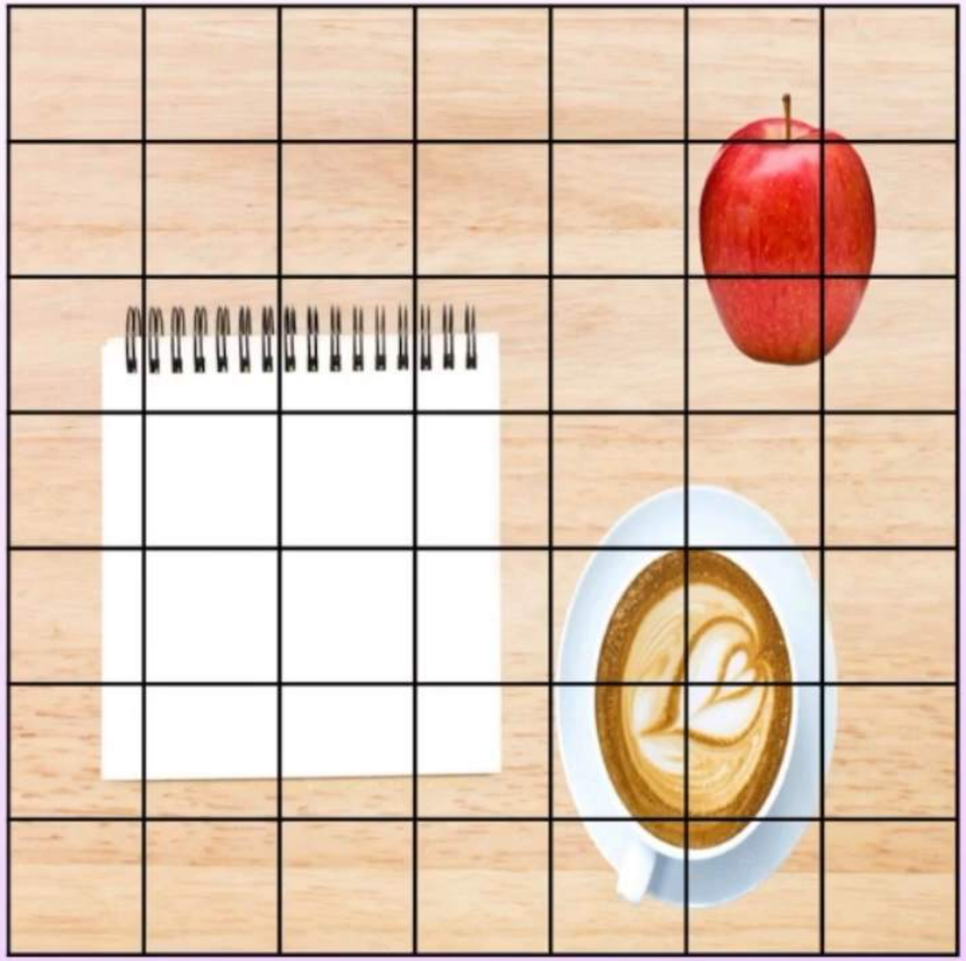
YOLO V1



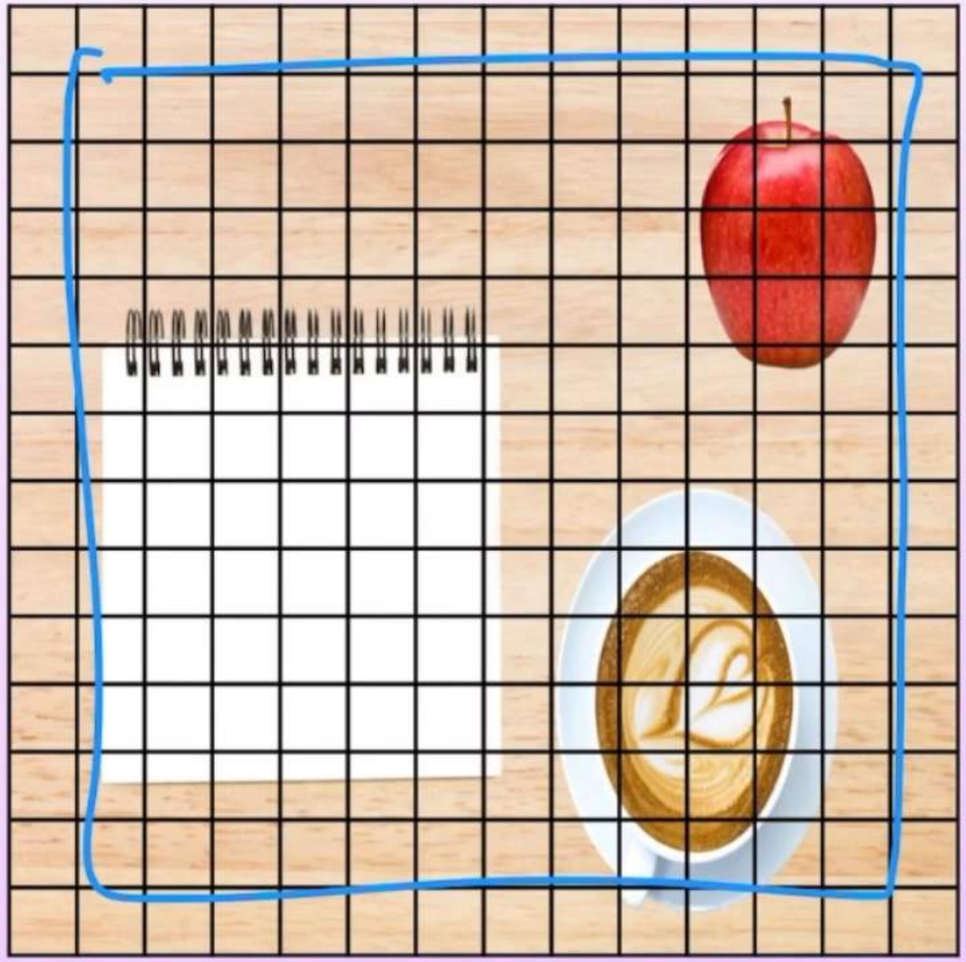
YOLO V2



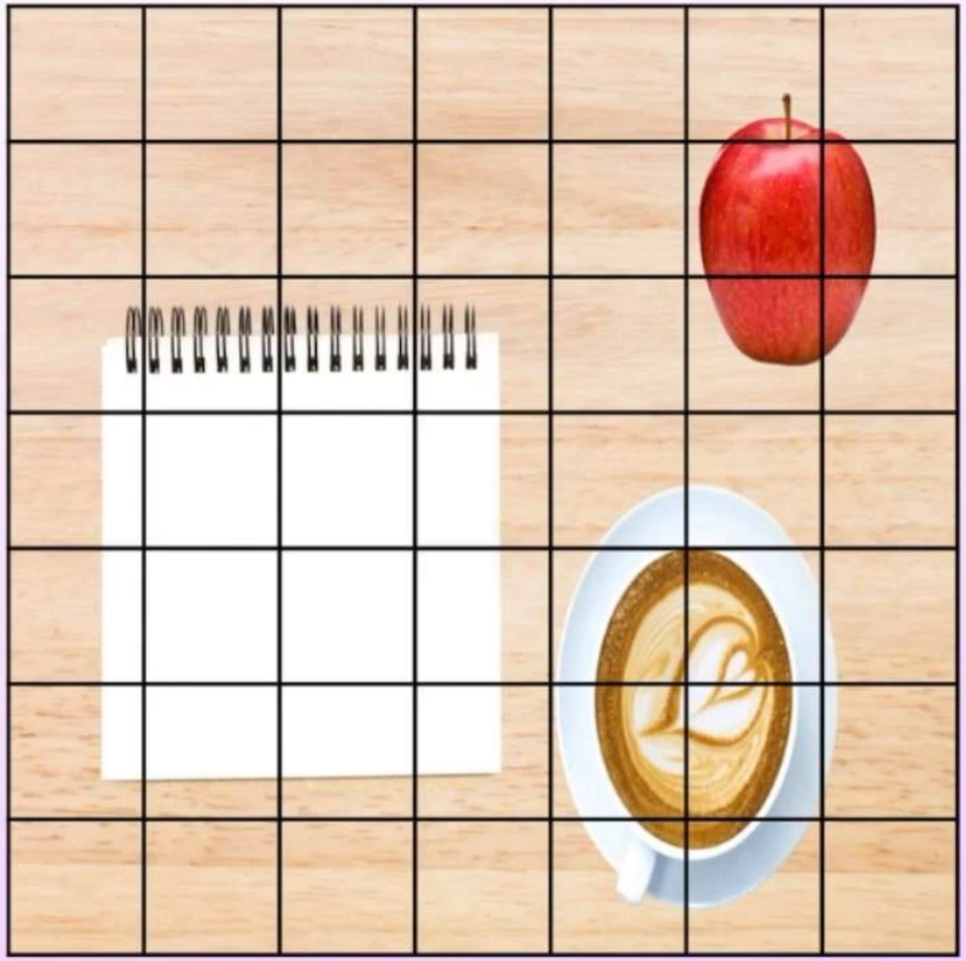
YOLO V1



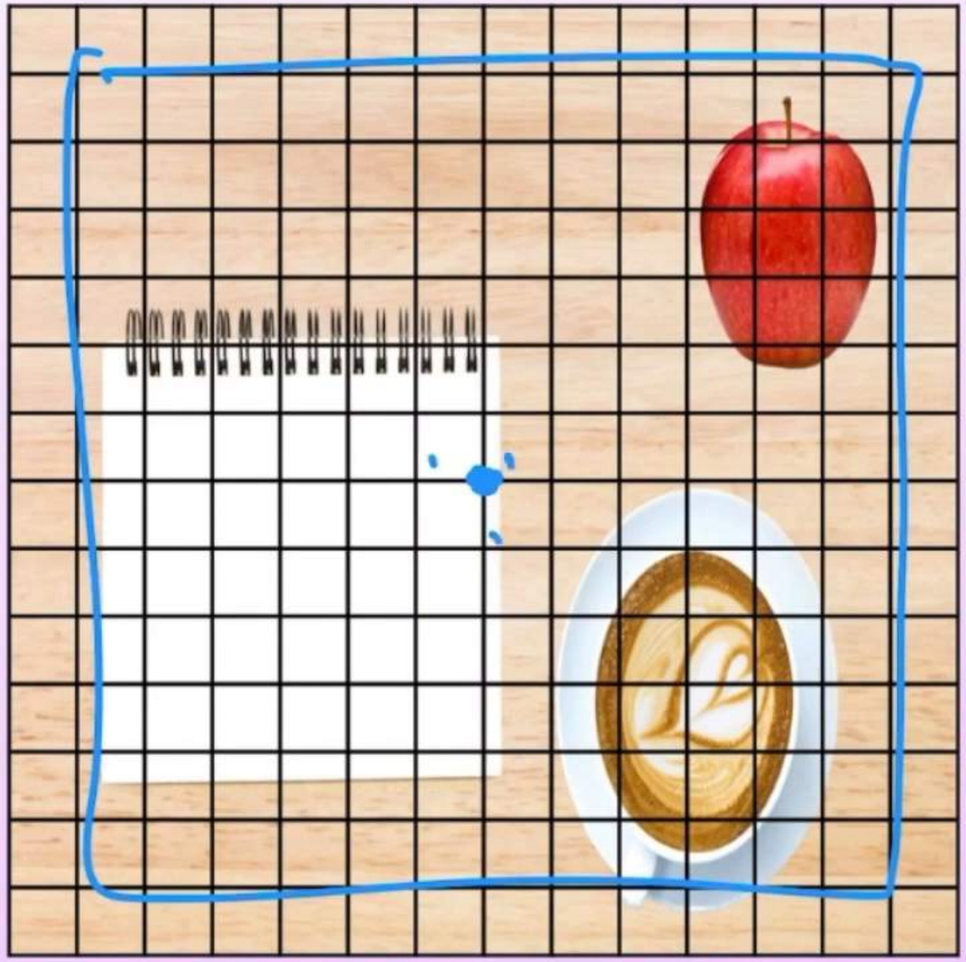
YOLO V2



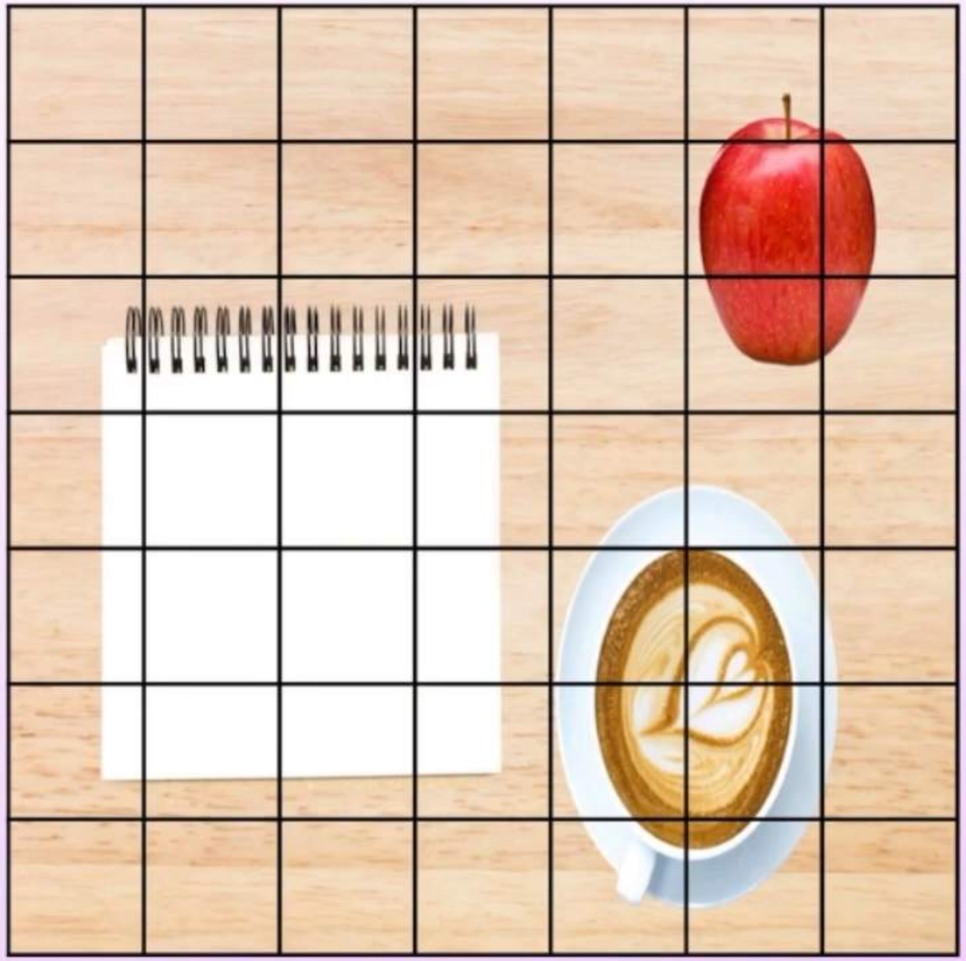
YOLO V1



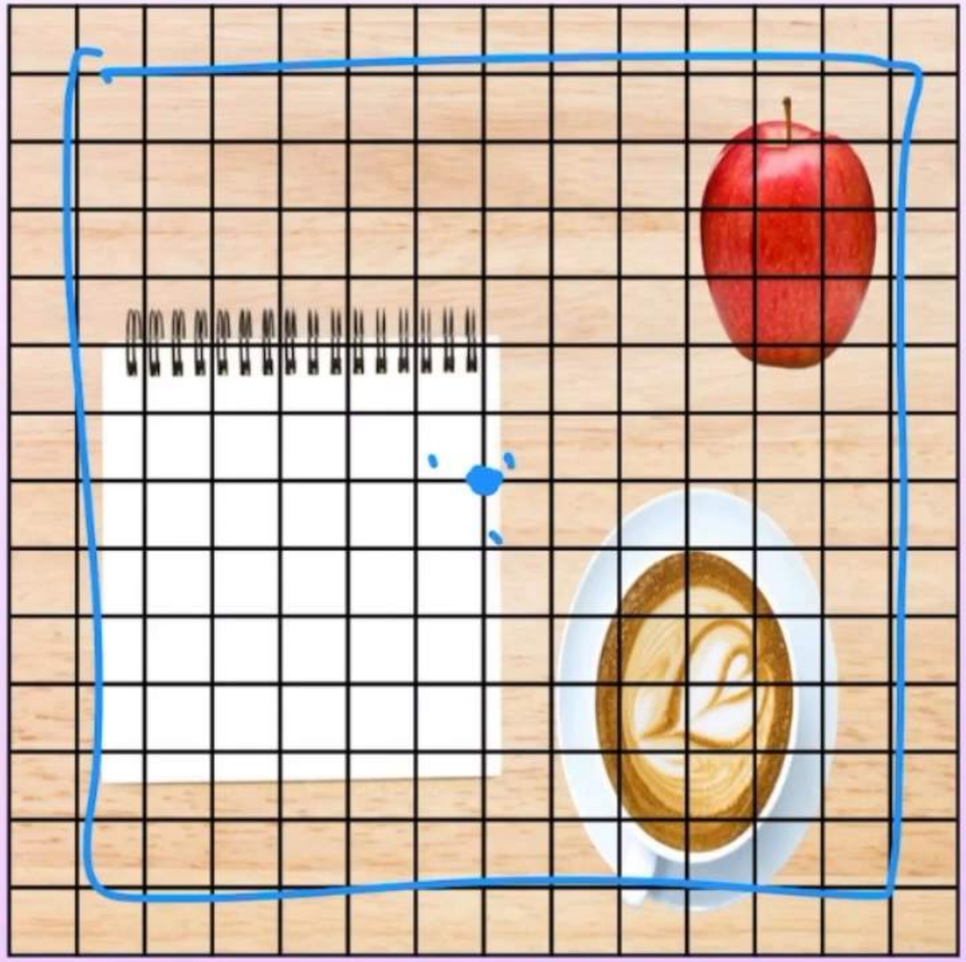
YOLO V2



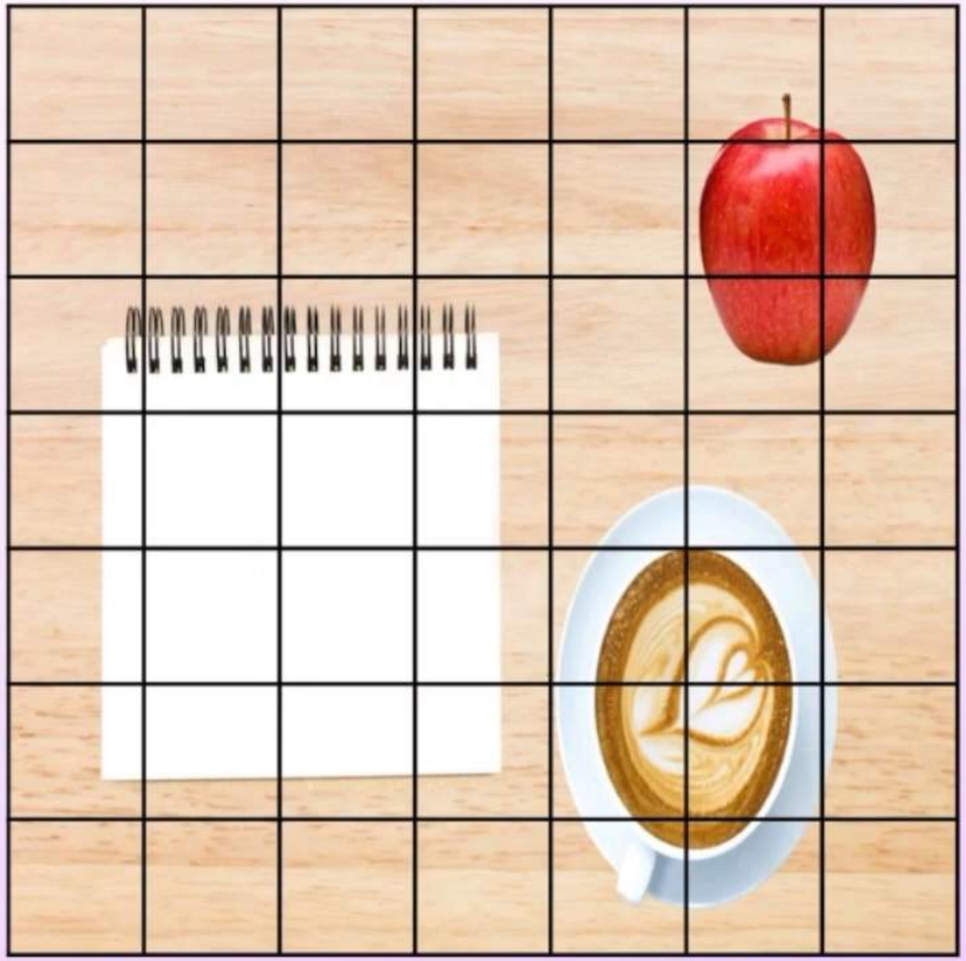
YOLO V1



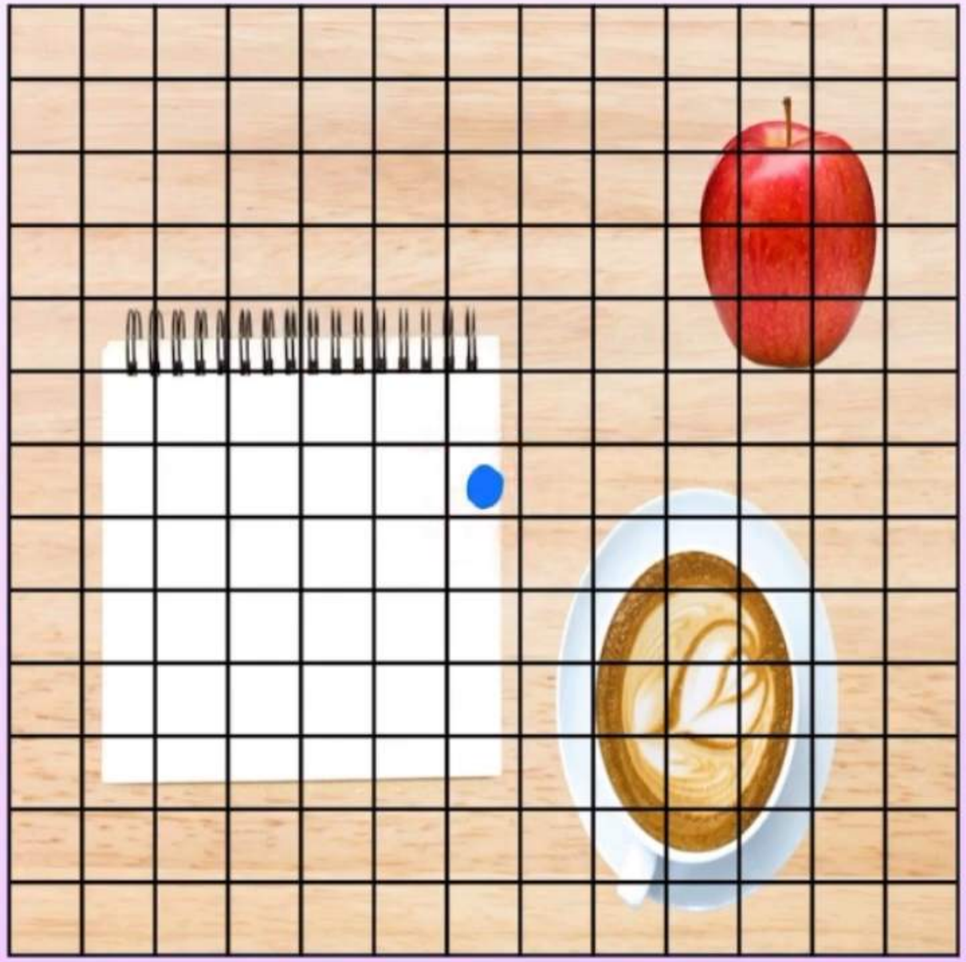
YOLO V2



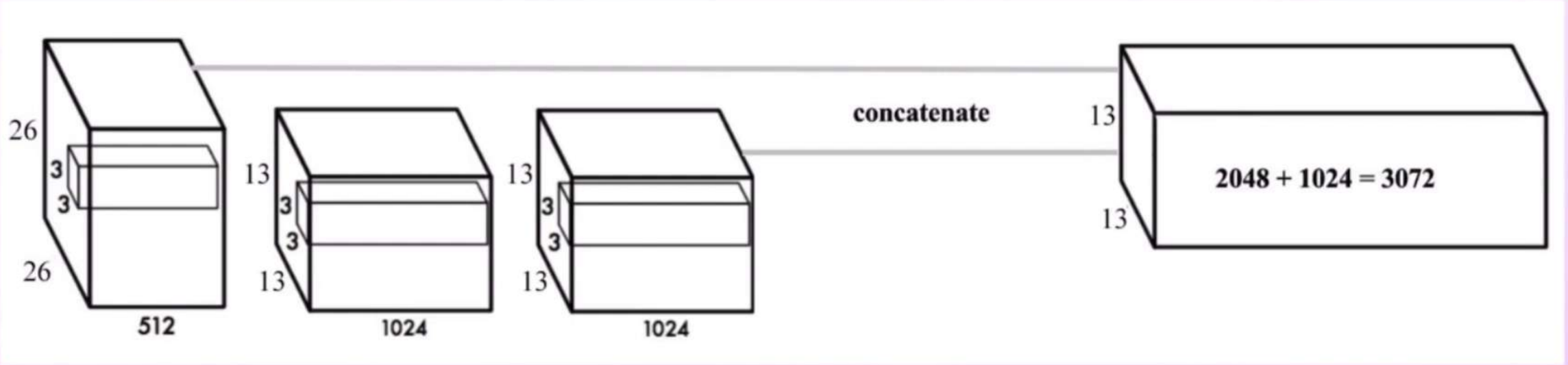
YOLO V1



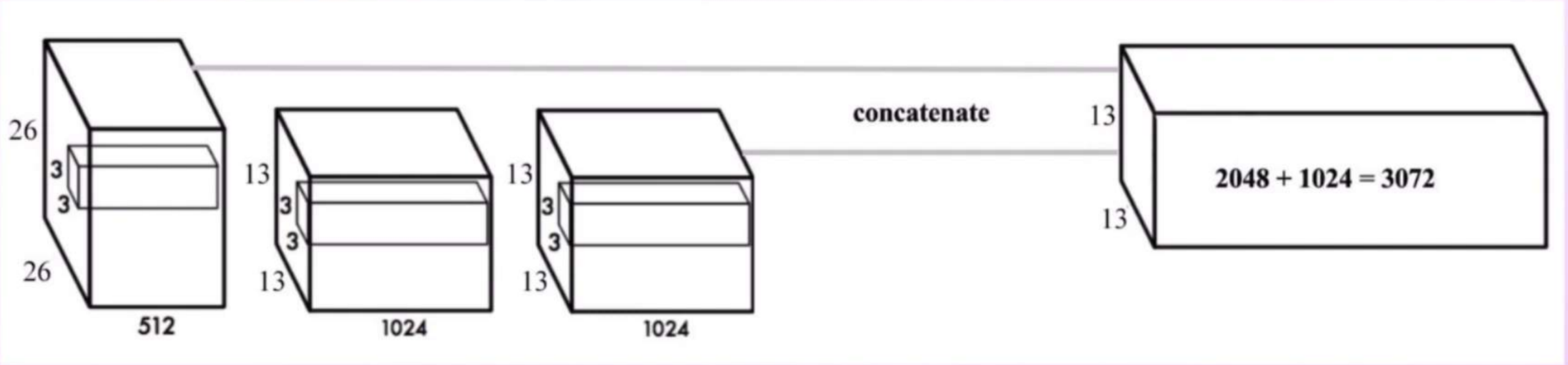
YOLO V2



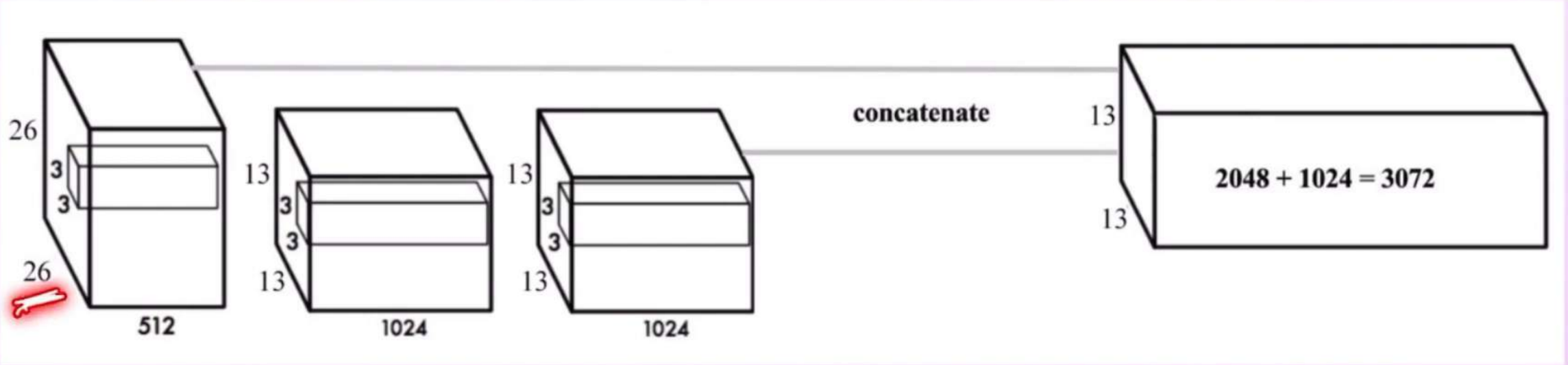
Pass-through layer



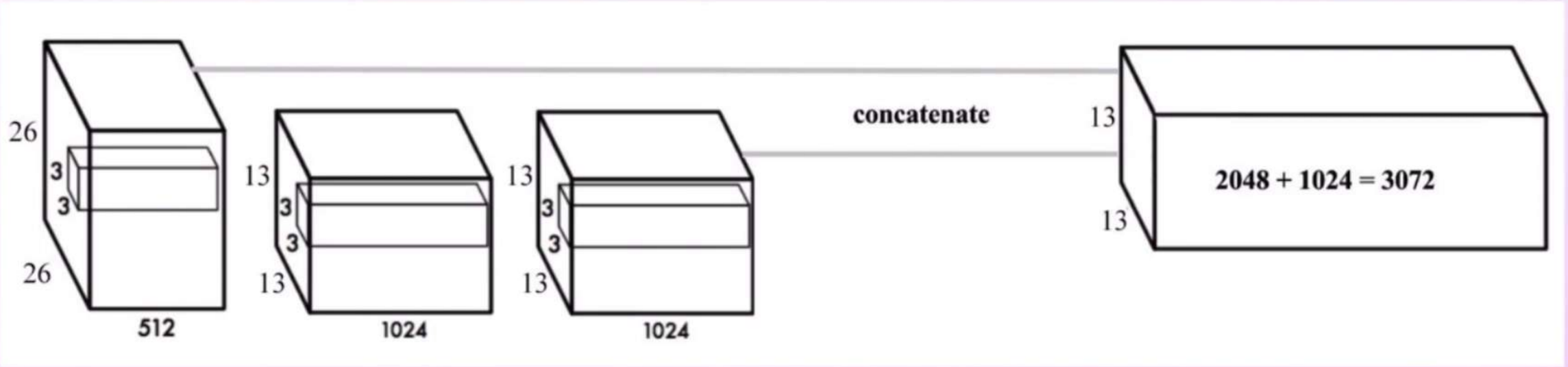
Pass-through layer



Pass-through layer

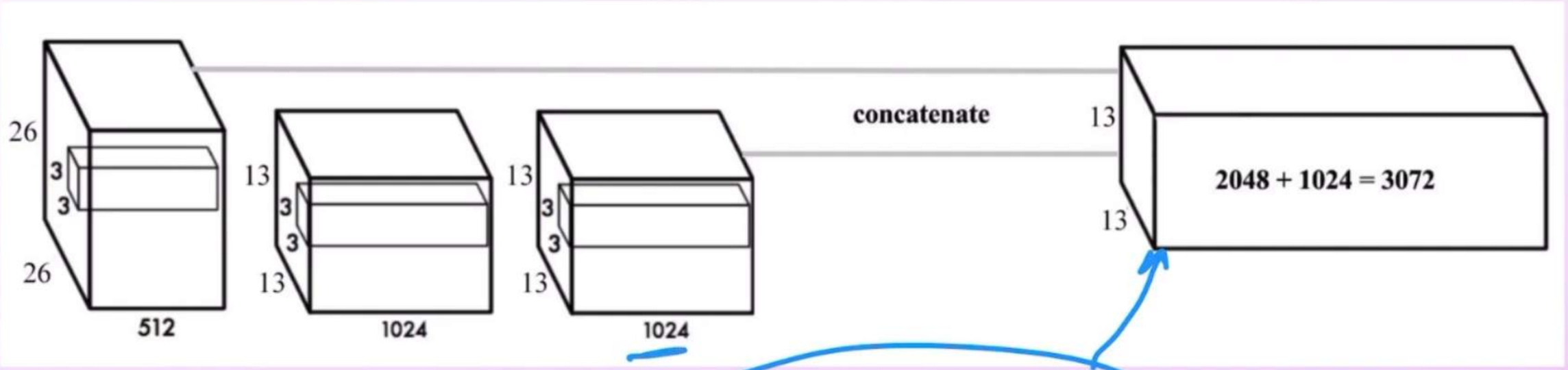


Pass-through layer

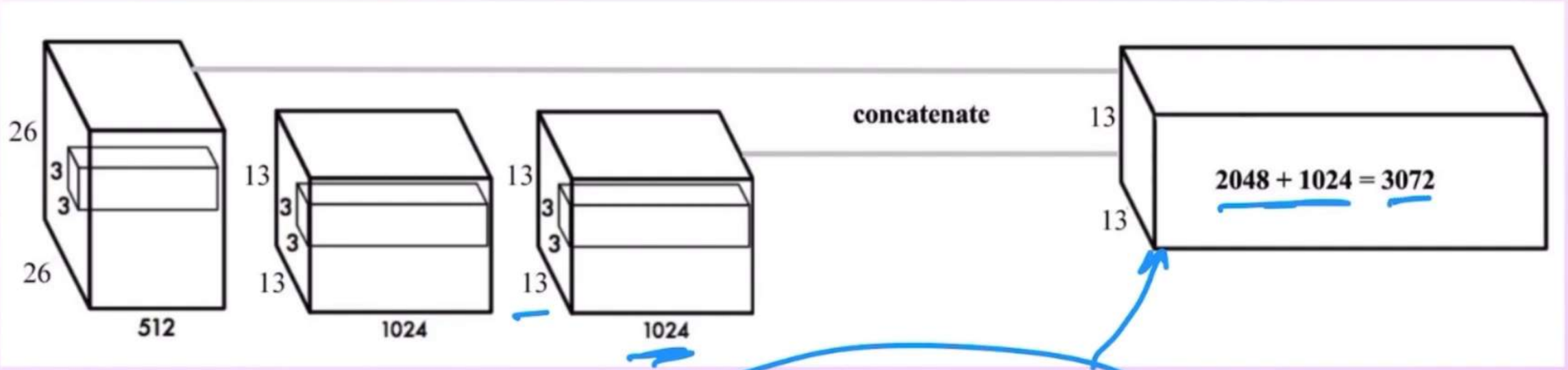


26 × 26 × 5

Pass-through layer

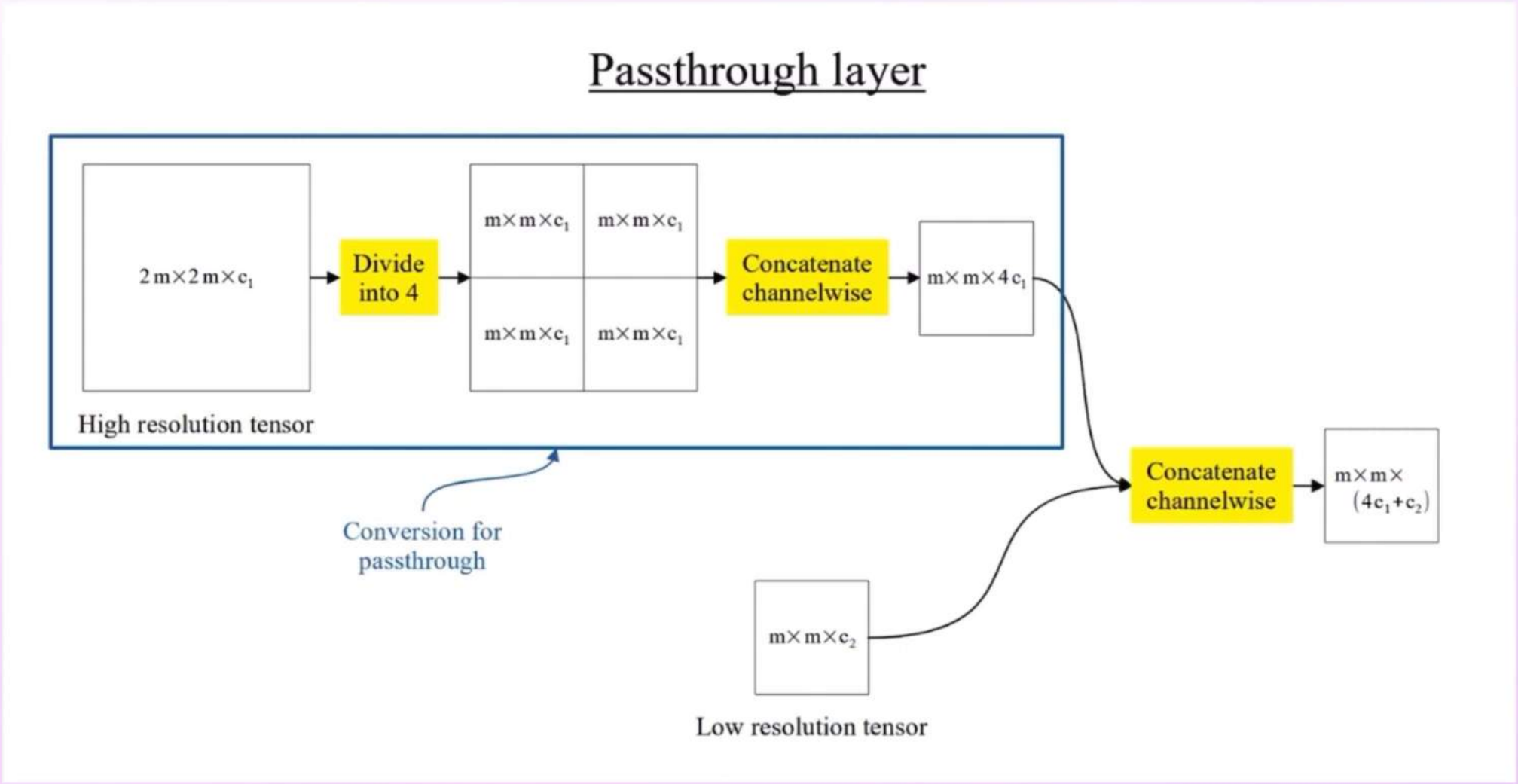


Pass-through layer



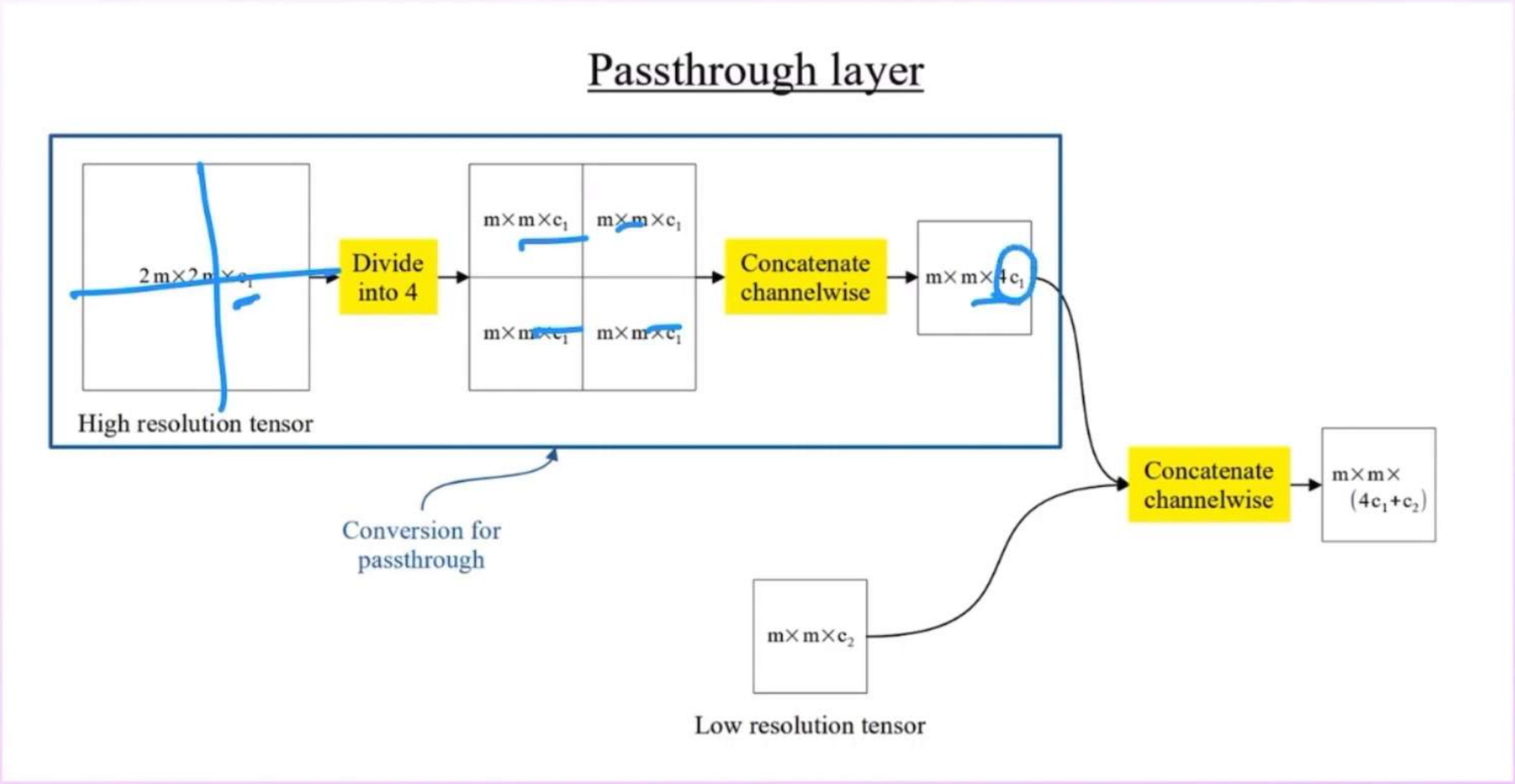
Pass-through layer

1% mAP rise



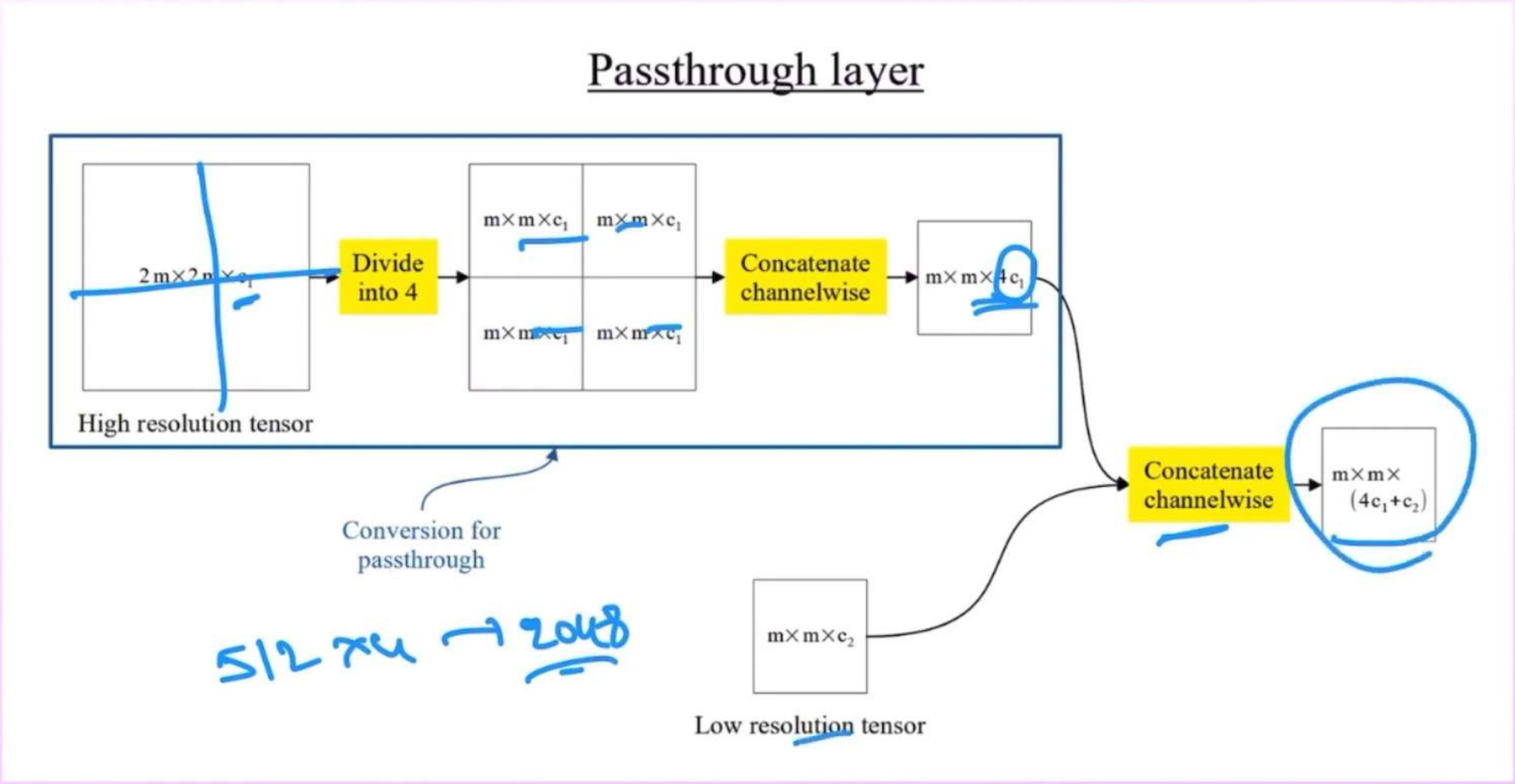
Pass-through layer

1% mAP rise

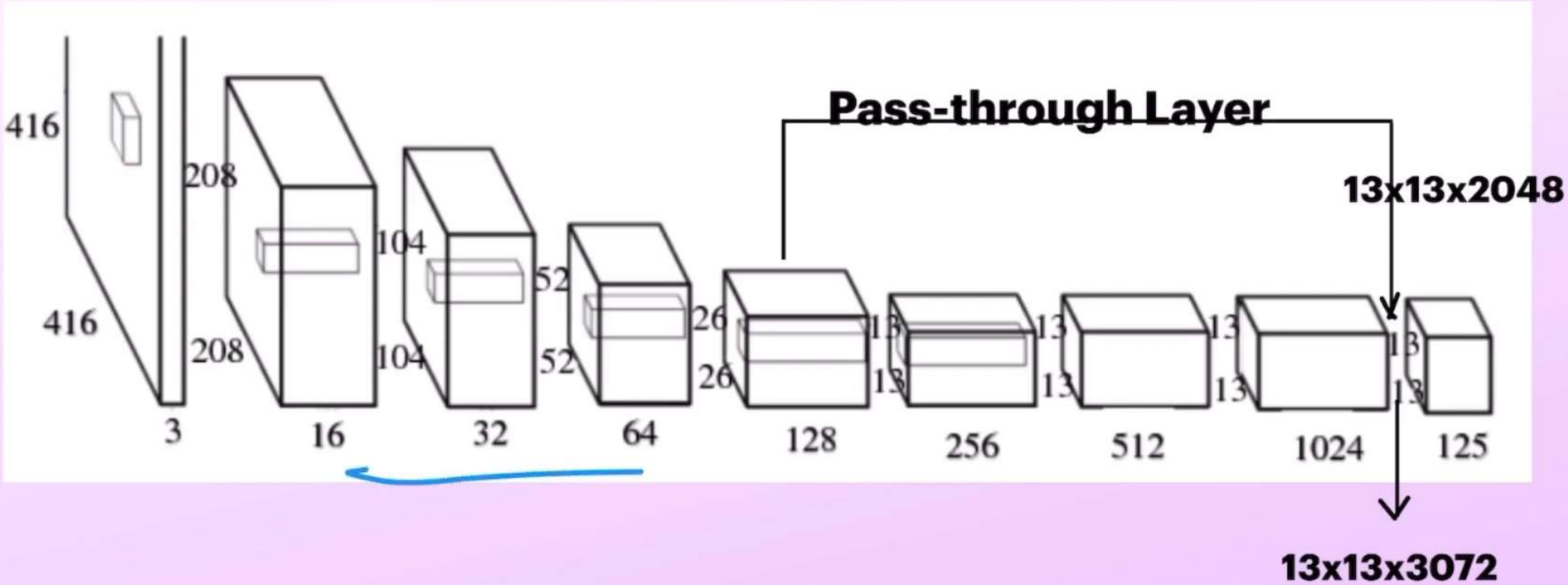


Pass-through layer

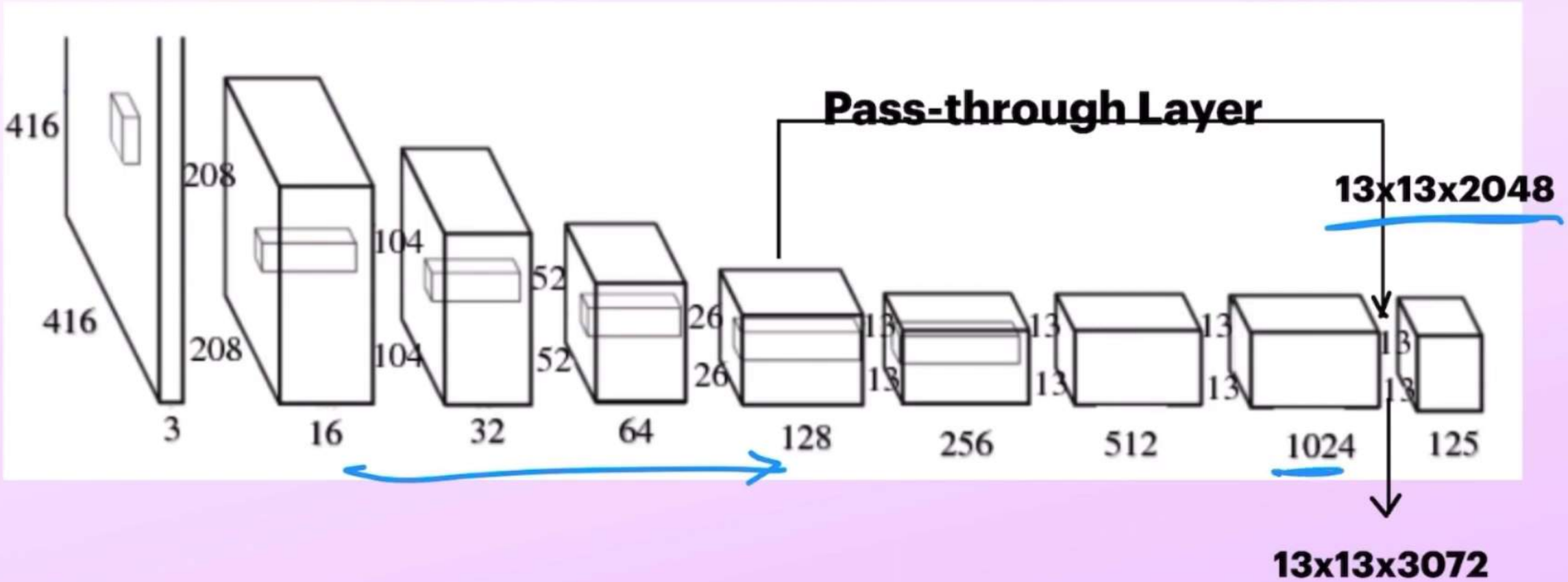
1% mAP rise



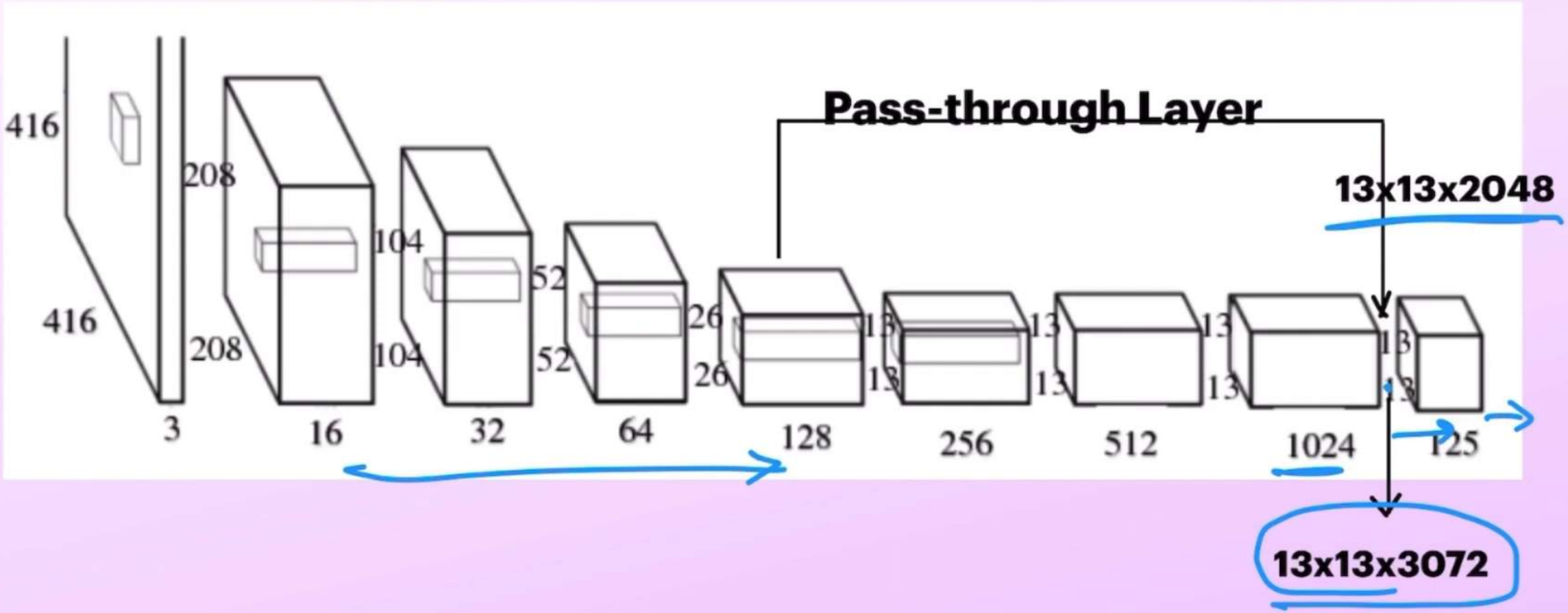
YOLO-V2 Model



YOLO-V2 Model



YOLO-V2 Model



Problem with bounding boxes

- 2 boxes per grid cell - 7x7x2=98

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Problem with bounding boxes

- 2 boxes per grid cell - 7x7x2=98

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Problem with bounding boxes

- 2 boxes per grid cell - 7x7x2=98

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Problem with bounding boxes

- 2 boxes per grid cell - 7x7x2=98

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Problem with bounding boxes

- 2 boxes per grid cell - 7x7x2=98

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

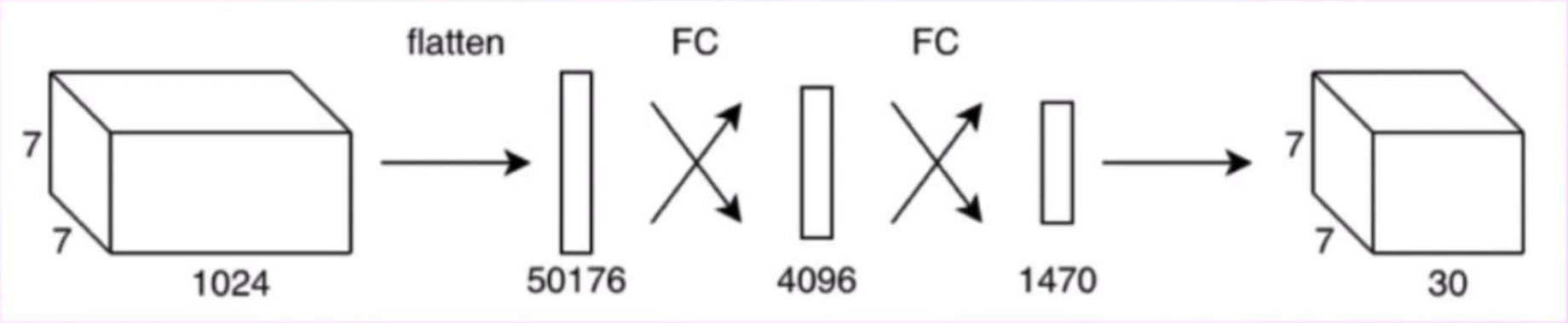
Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

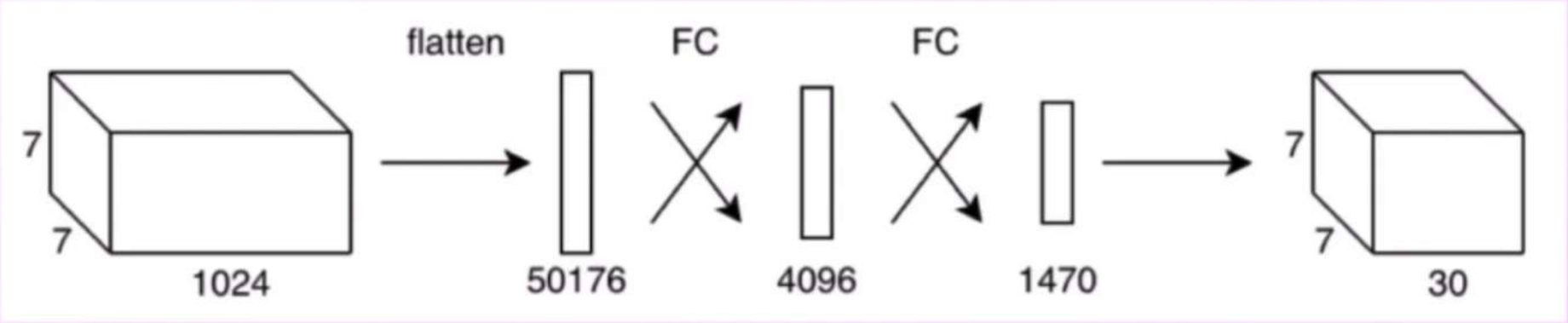
Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50?



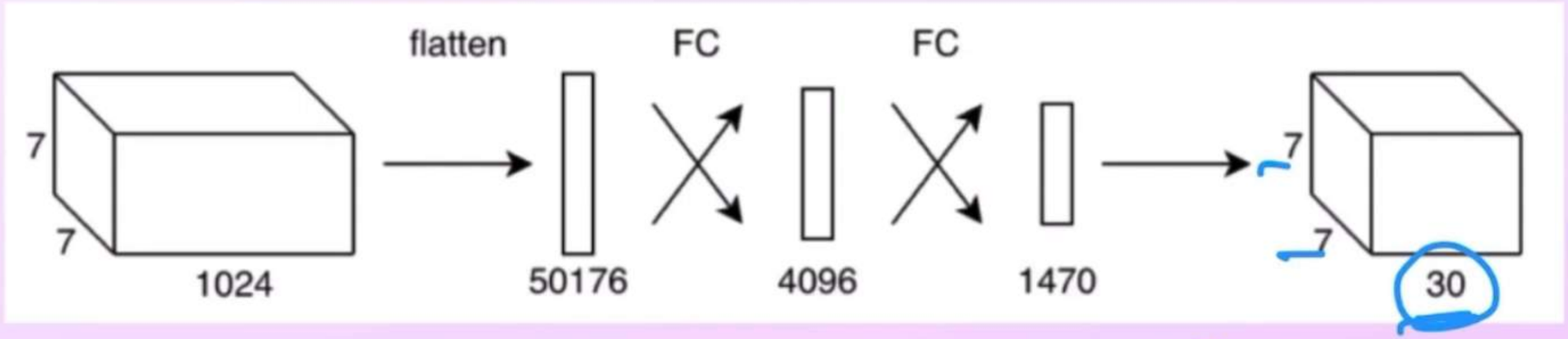
Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50? → 5000



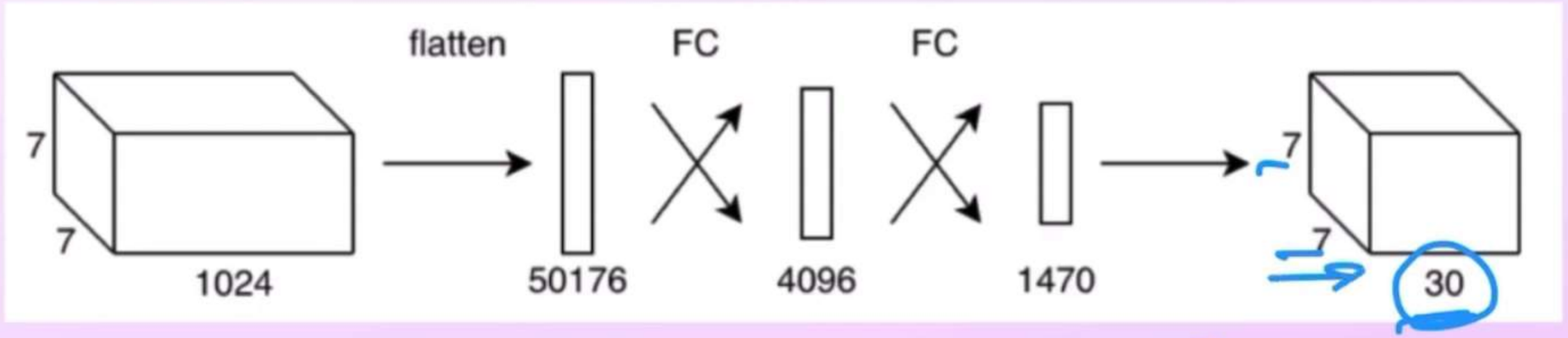
Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50?



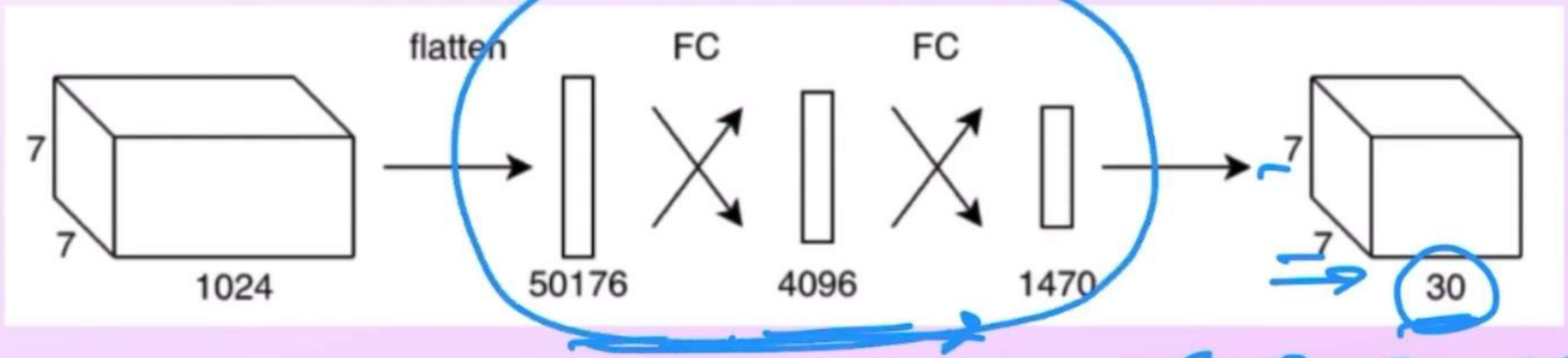
Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50?



Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50?



Million

50 x 50 x 30 → 75K

Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50? - Not Feasible

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50? - Not Feasible

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Problem with bounding boxes

- With 13x13 grid cells - 13x13x2=338
- Should we increase the grids to 50x50?
- Problem is with fully connected layers

Fully Convolutional Network

- Last 2 fully connected layers are removed
- Number of parameters is negligible - kernel size is fixed
- Yolo-V2 is a fully Convolutional network

Fully Convolutional Network

- Last 2 fully connected layers are removed
- Number of parameters is negligible - kernel size is fixed
- Yolo-V2 is a fully Convolutional network

Fully Convolutional Network

- Last 2 fully connected layers are removed
- Number of parameters is negligible - kernel size is fixed
- Yolo-V2 is a fully Convolutional network

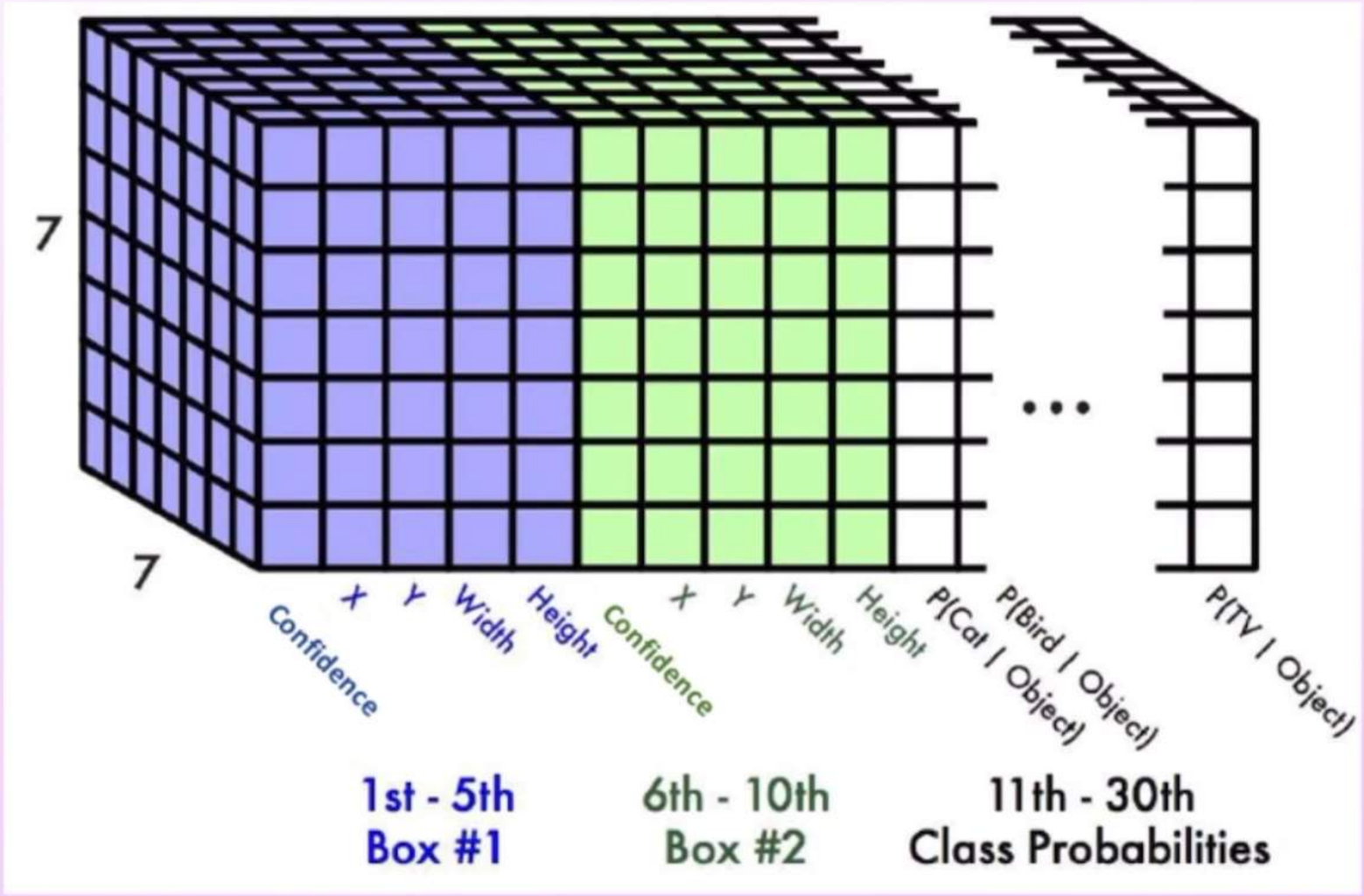
Fully Convolutional Network

- Last 2 fully connected layers are removed
- Number of parameters is negligible - kernel size is fixed
- Yolo-V2 is a fully Convolutional network

Problem with bounding boxes

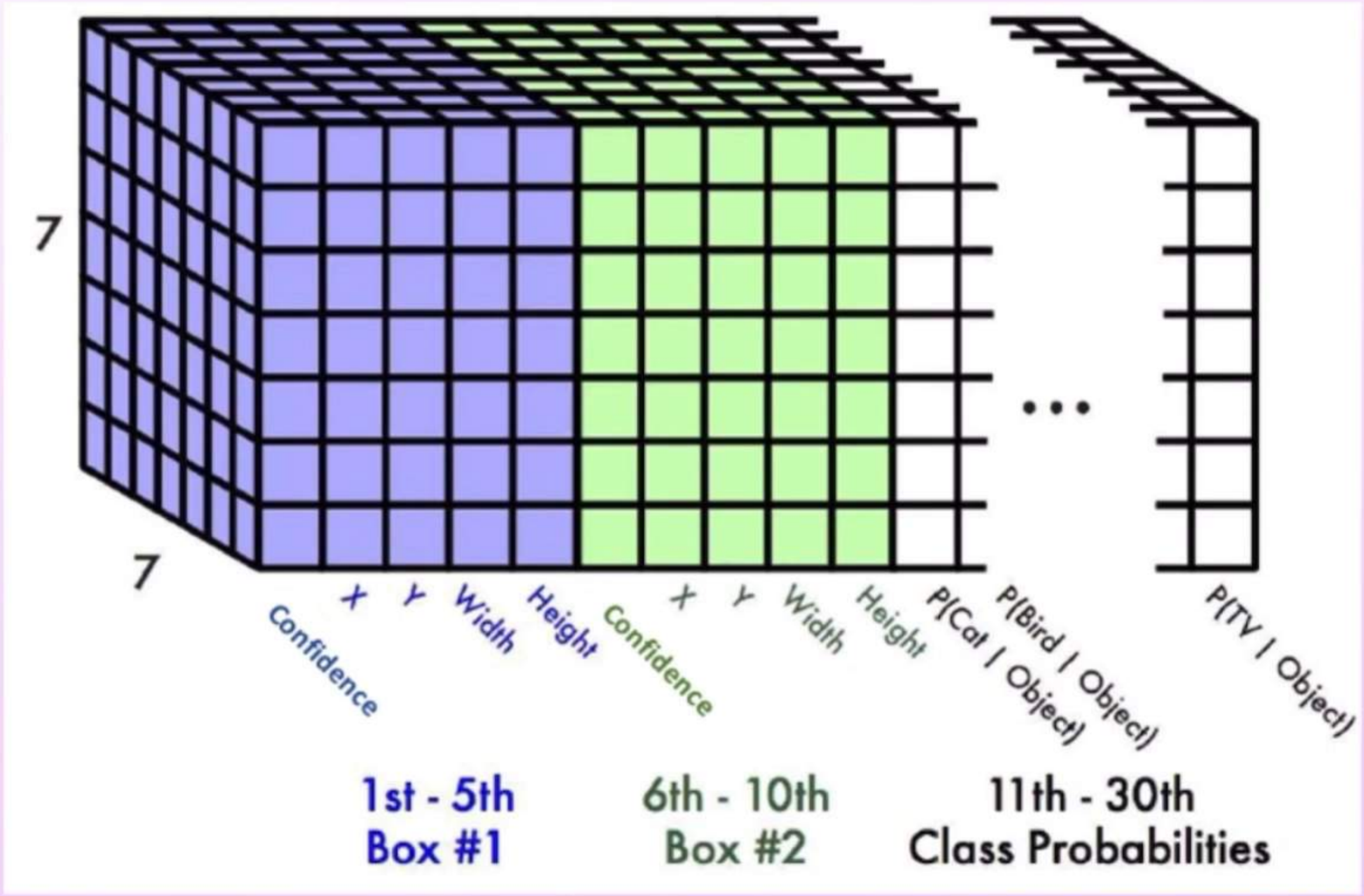
- 1 class per grid cell

10



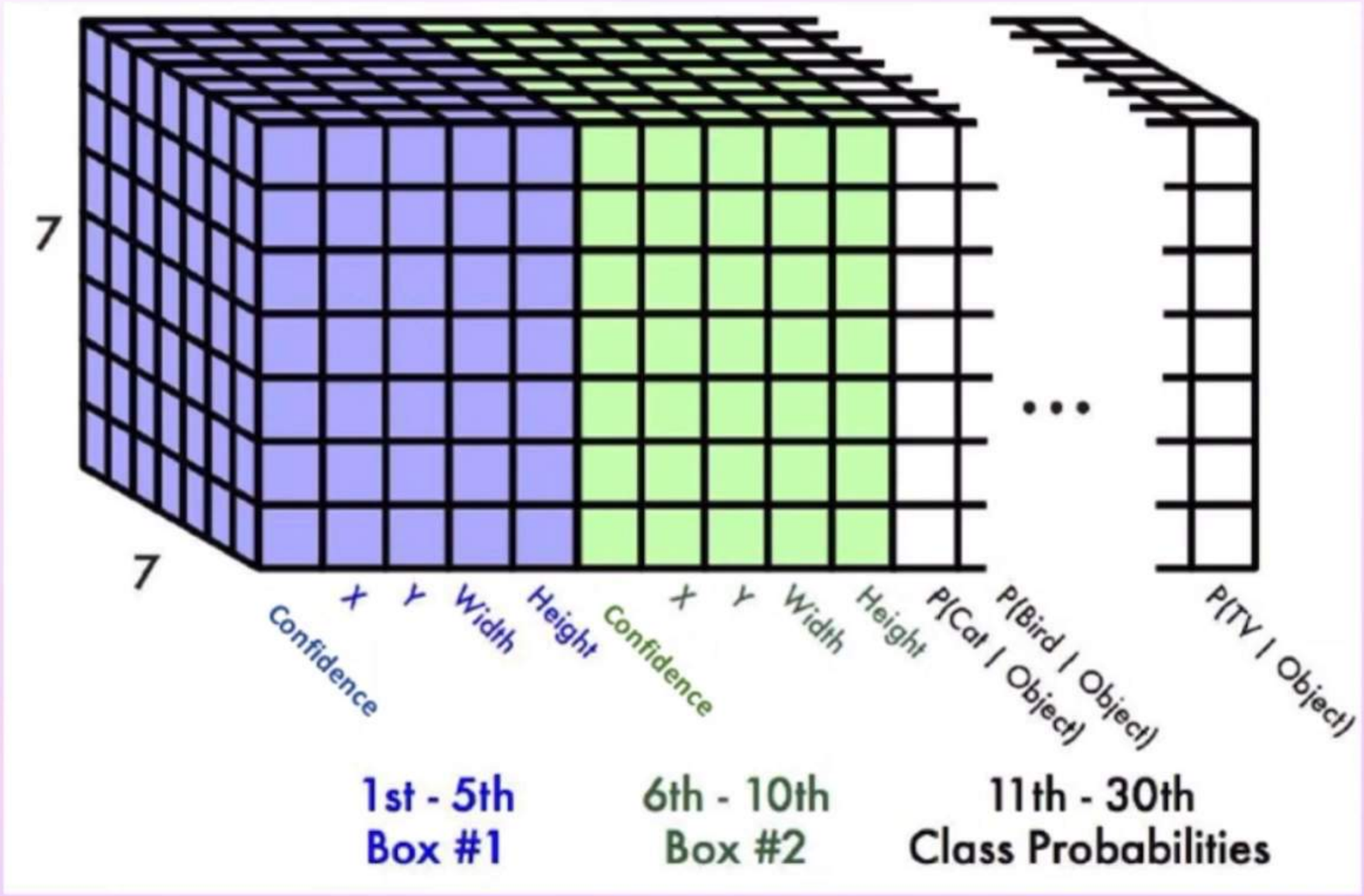
Problem with bounding boxes

- 1 class per grid cell



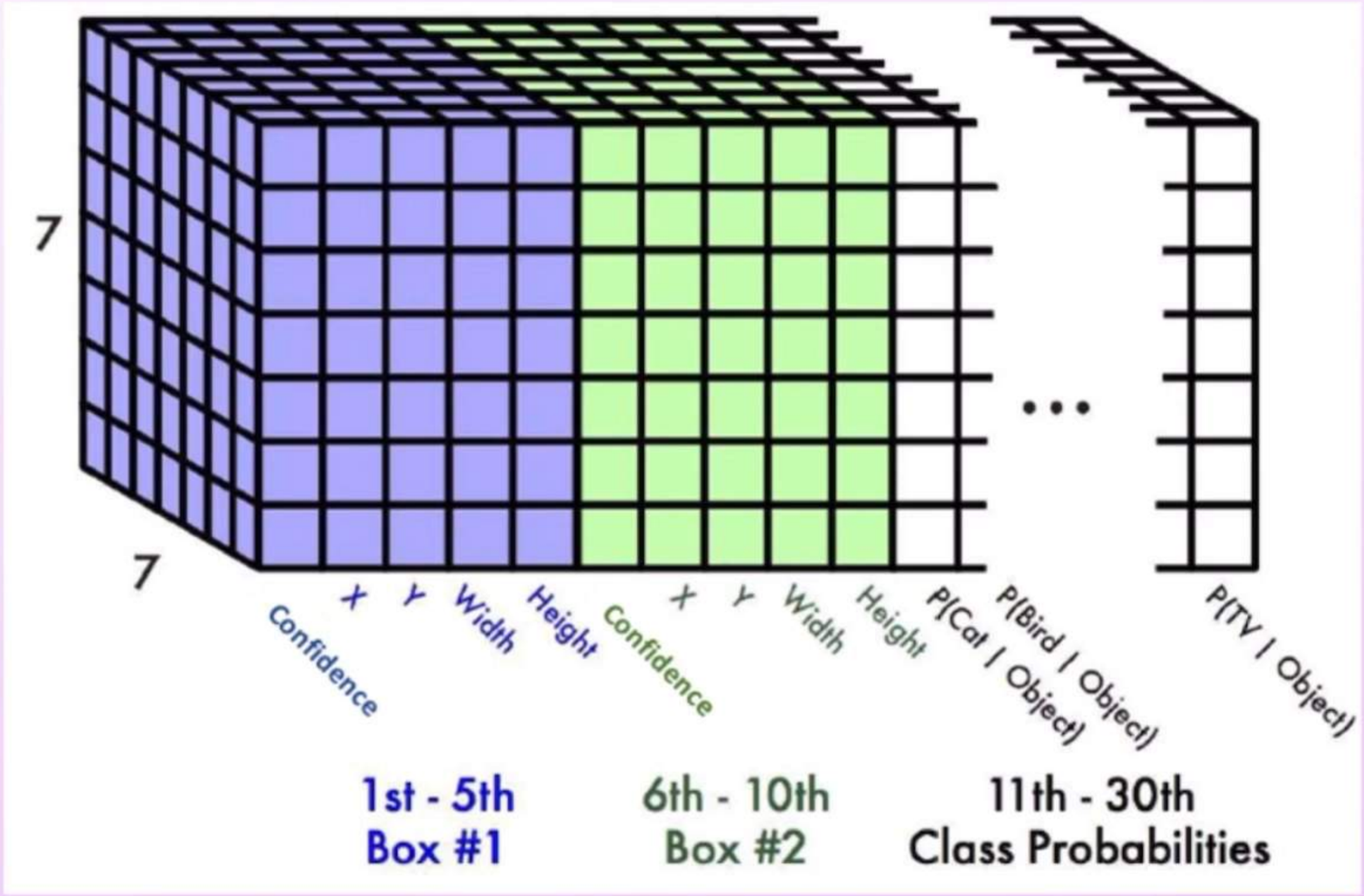
Problem with bounding boxes

- 1 class per grid cell
- Limits the number of objects detected - 49
- For 13x13 cells - 169



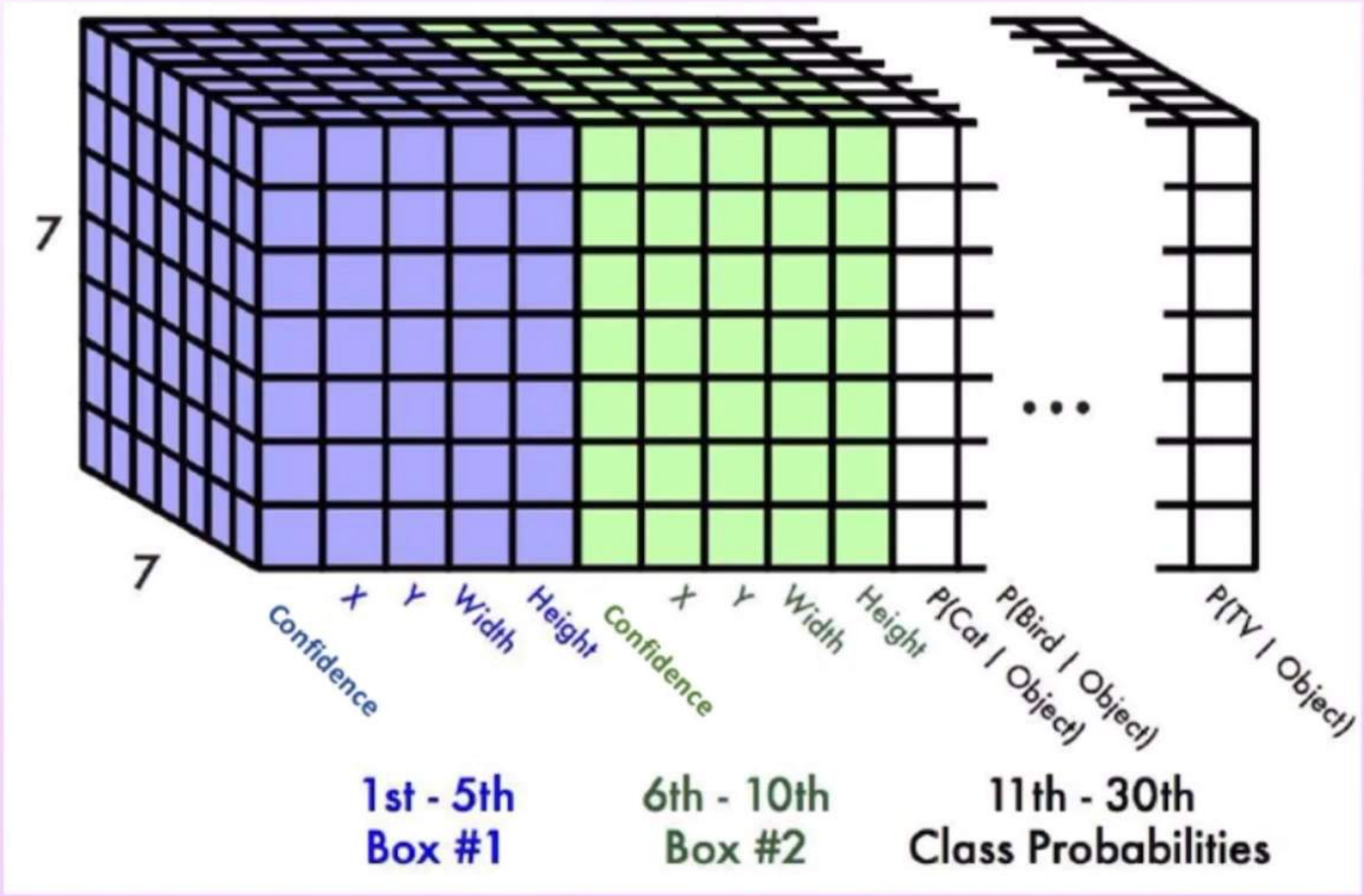
Problem with bounding boxes

- 1 class per grid cell
- Limits the number of objects detected - 49
- For 13x13 cells - 169



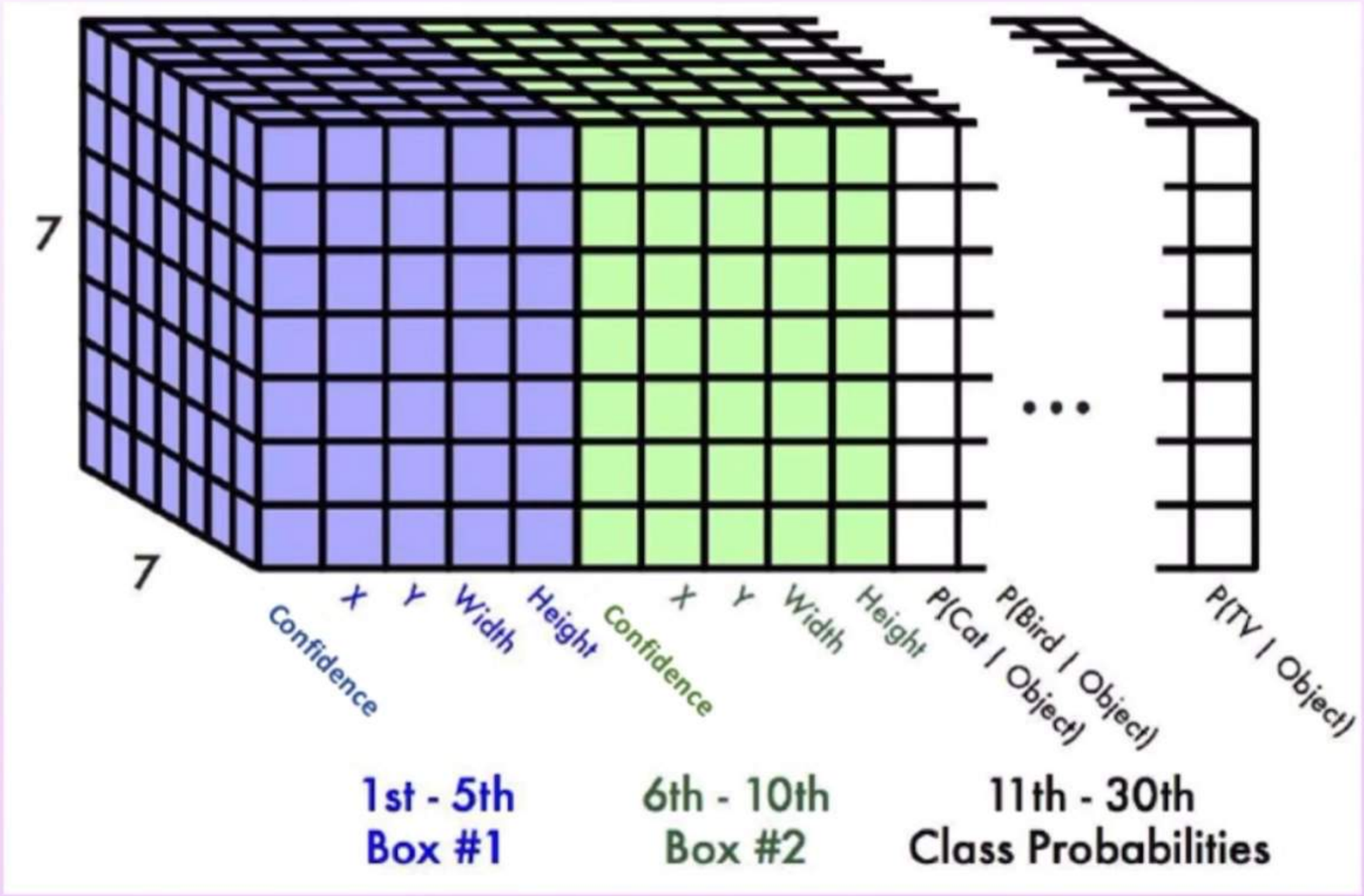
Problem with bounding boxes

- 1 class per grid cell
- Limits the number of objects detected - 49
- For 13x13 cells - 169



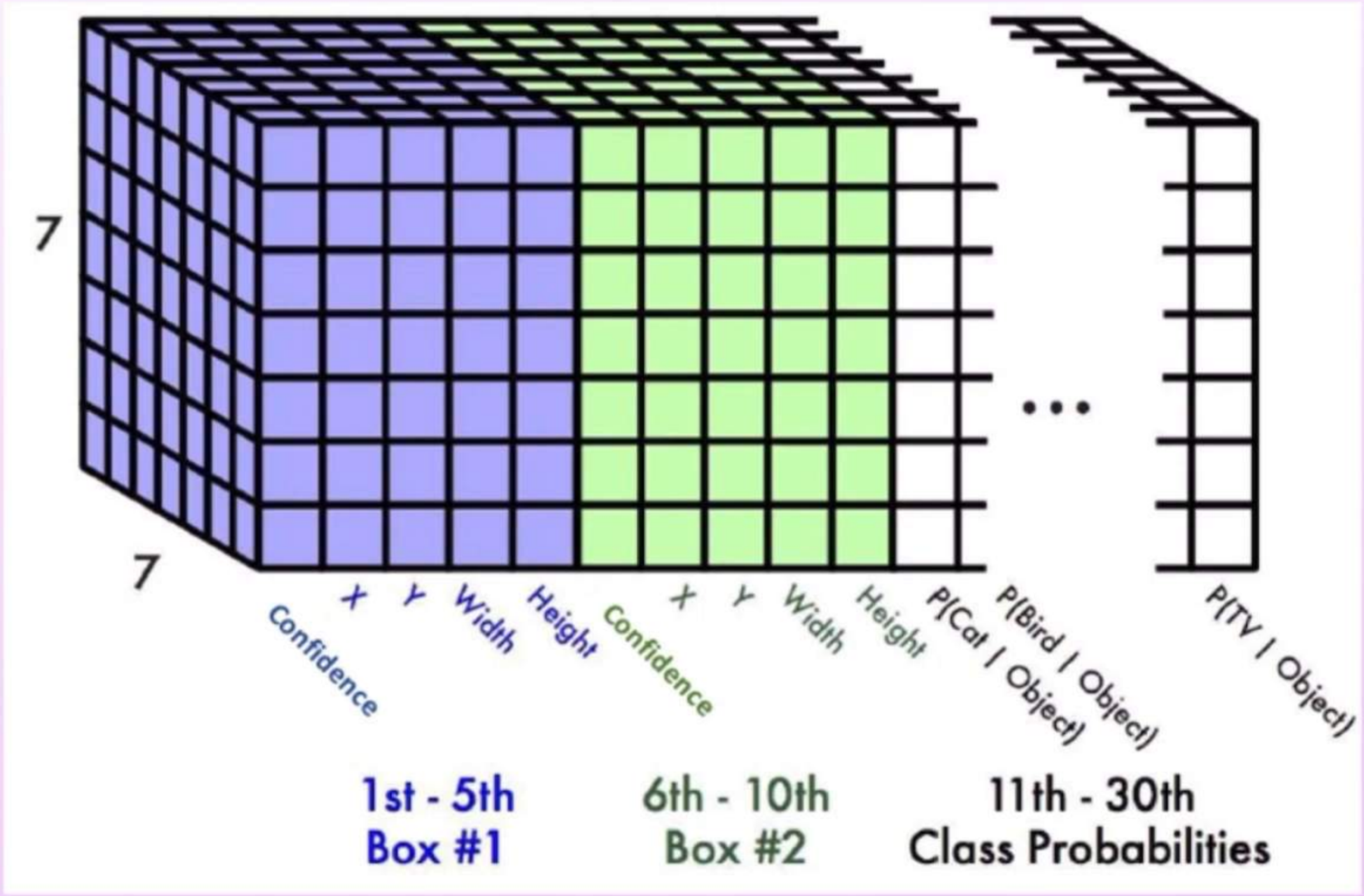
Problem with bounding boxes

- 1 class per grid cell
- Limits the number of objects detected
- Solution: class prediction per box



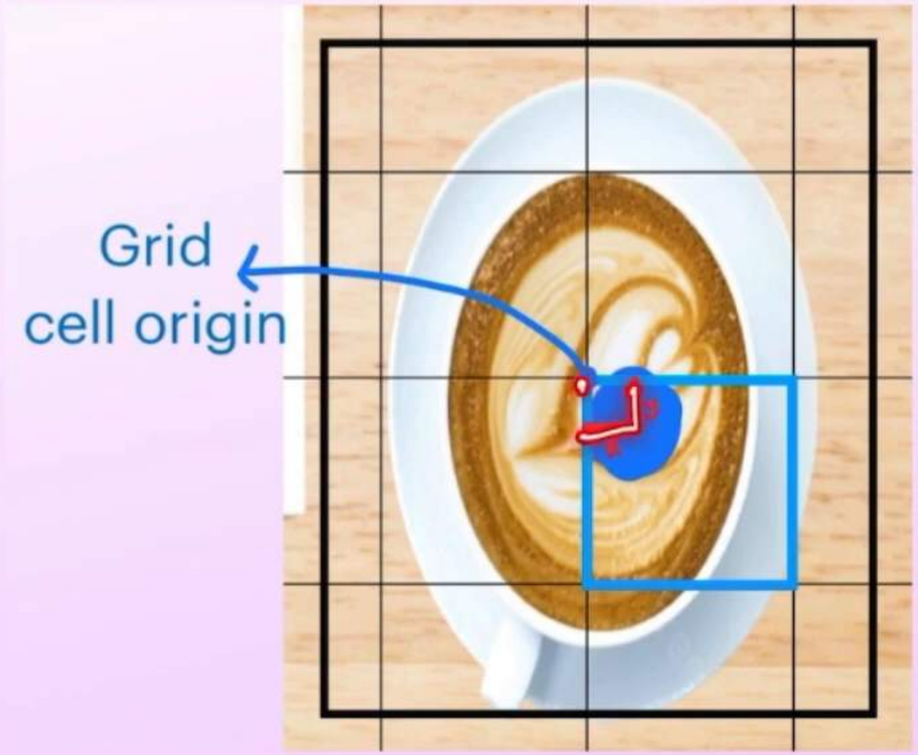
Problem with bounding boxes

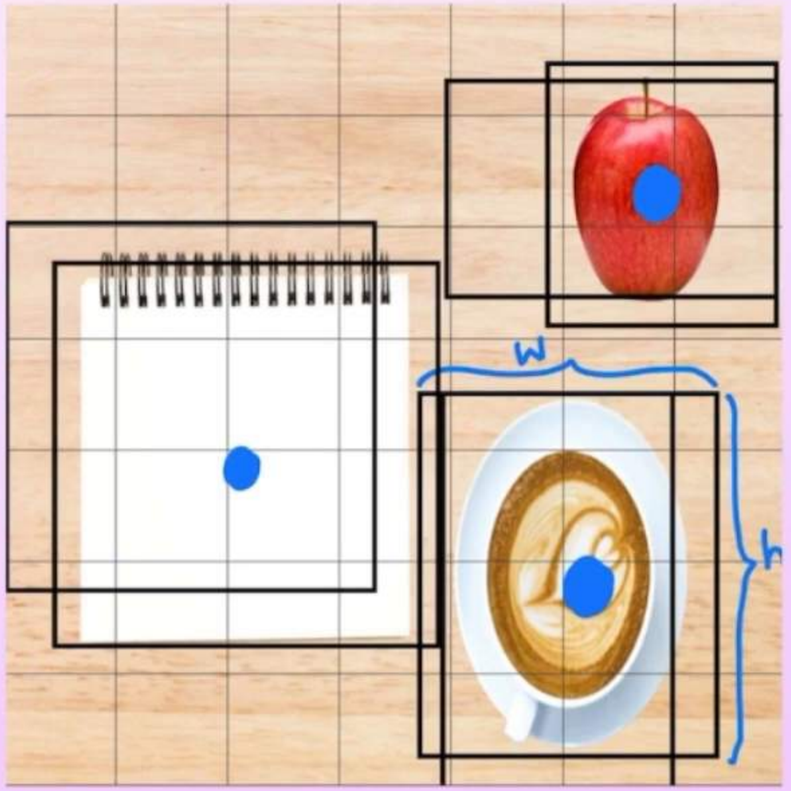
- 1 class per grid cell
- Limits the number of objects detected
- Solution: class prediction per box



Reason for Poor Localization

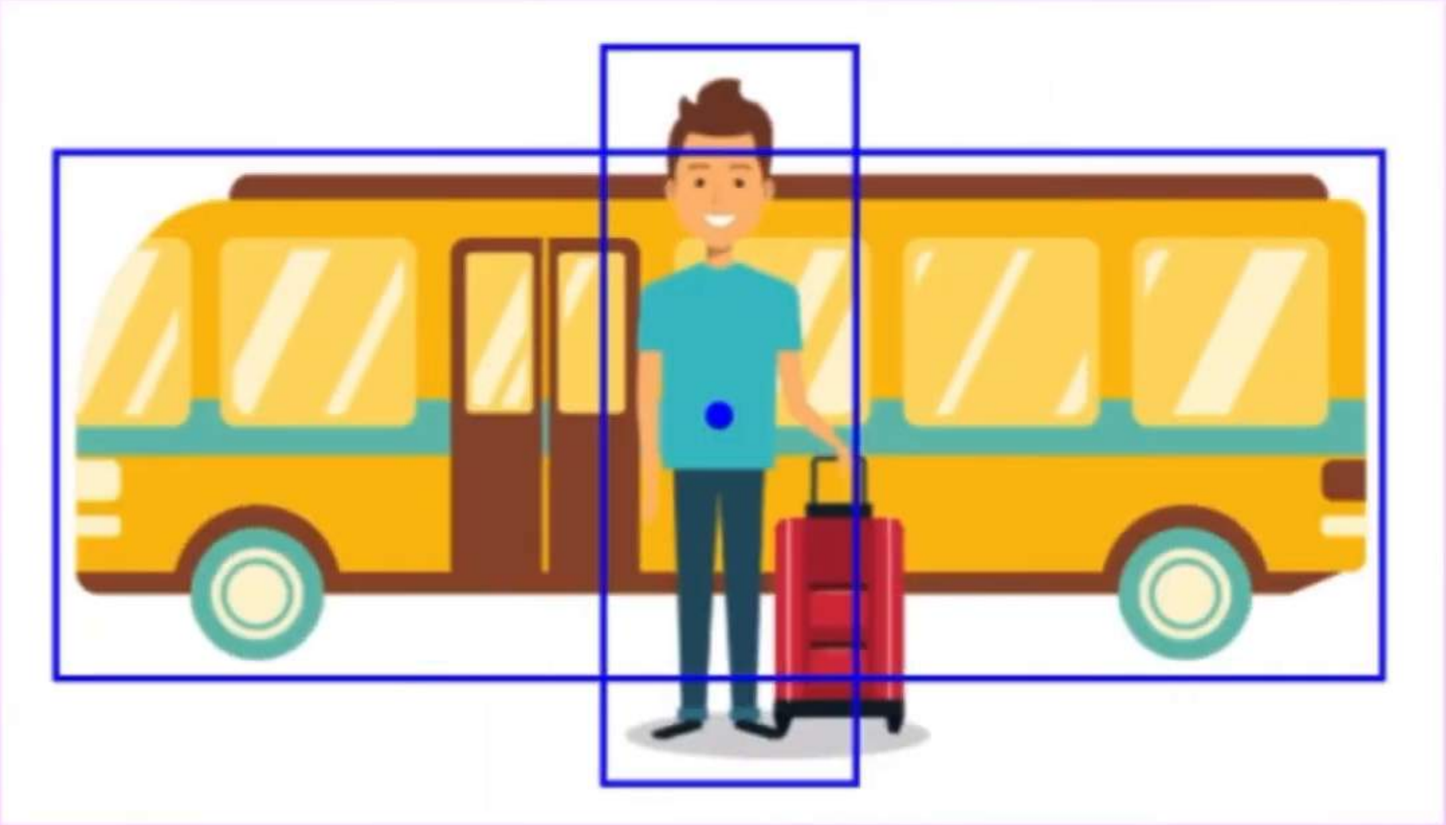
- Boxes are learnt relative to grid cell





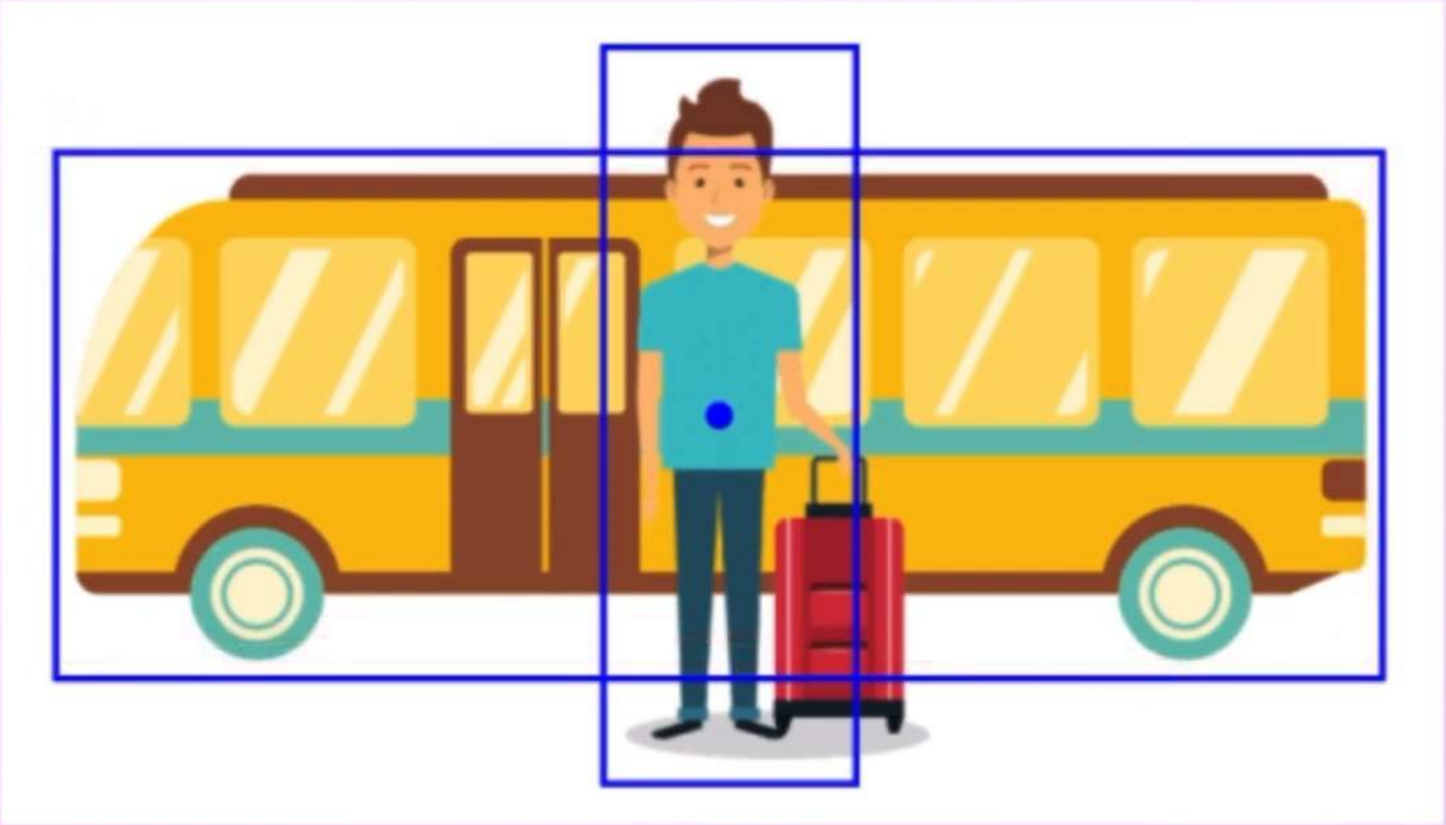
Reason for Poor Localization

- Boxes are learnt relative to grid cell
- Objects can be of different shapes



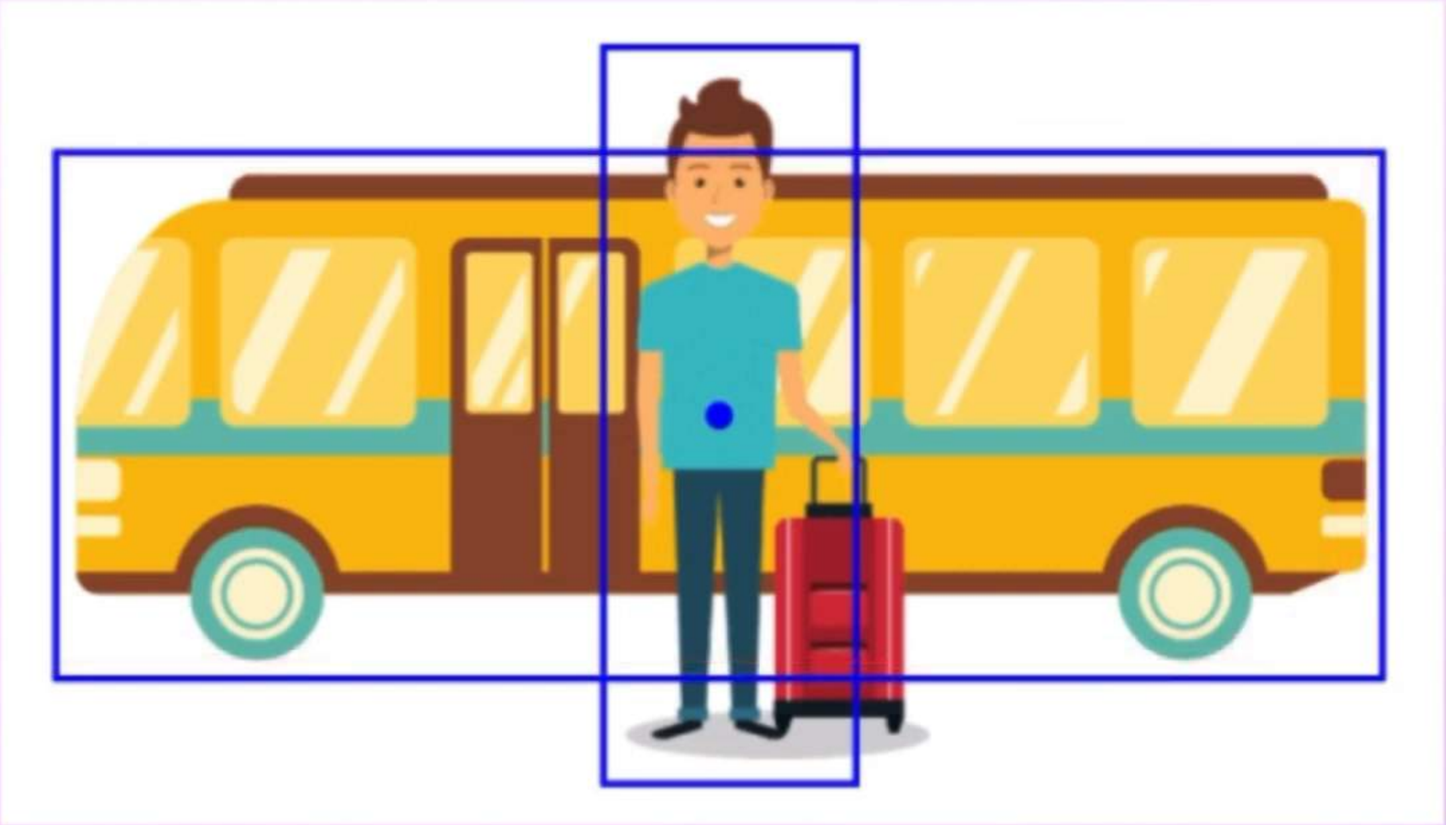
Reason for Poor Localization

- Boxes are learnt relative to grid cell
- Objects can be of different shapes



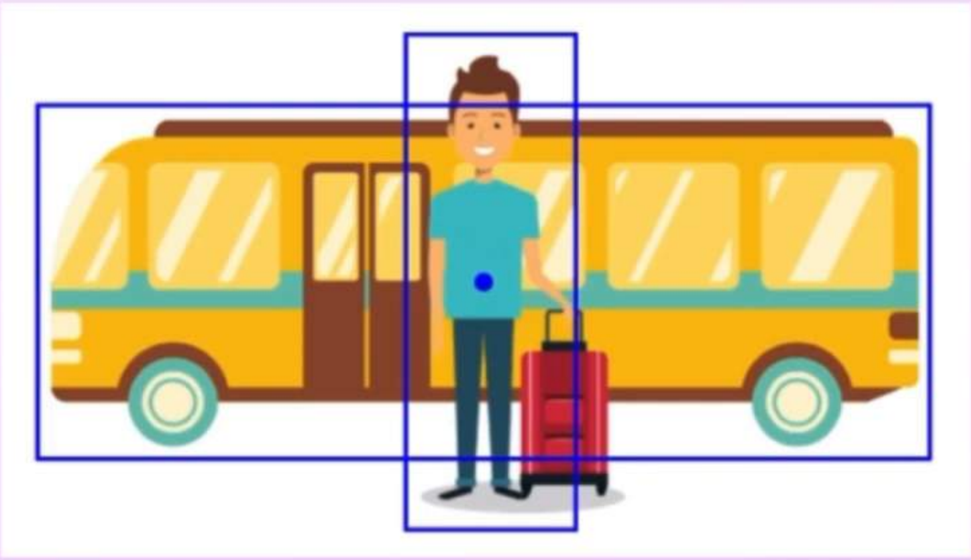
Reason for Poor Localization

- Boxes are learnt relative to grid cell
- Objects can be of different shapes



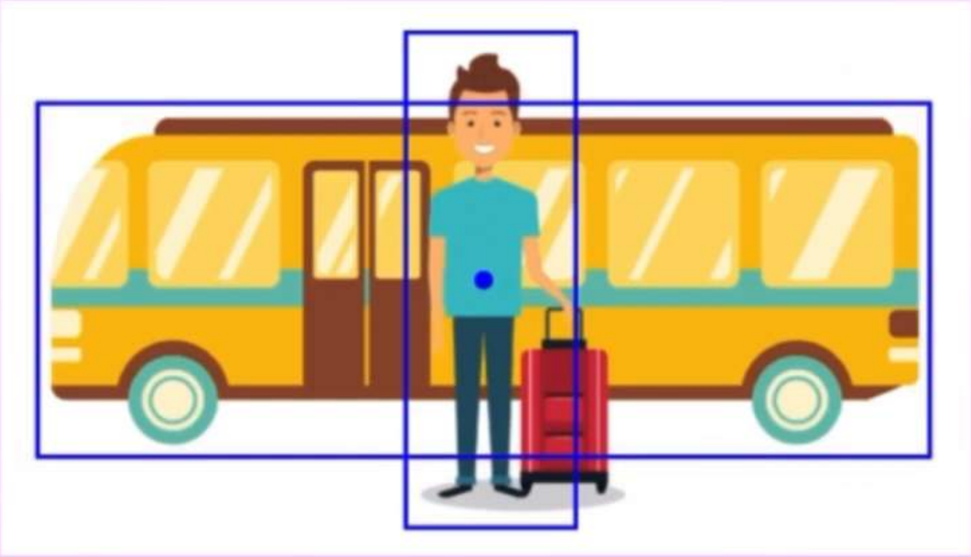
Reason for Poor Localization

- Boxes are learnt relative to grid cell
- Objects can be of different shapes
- Bounding box regression is difficult



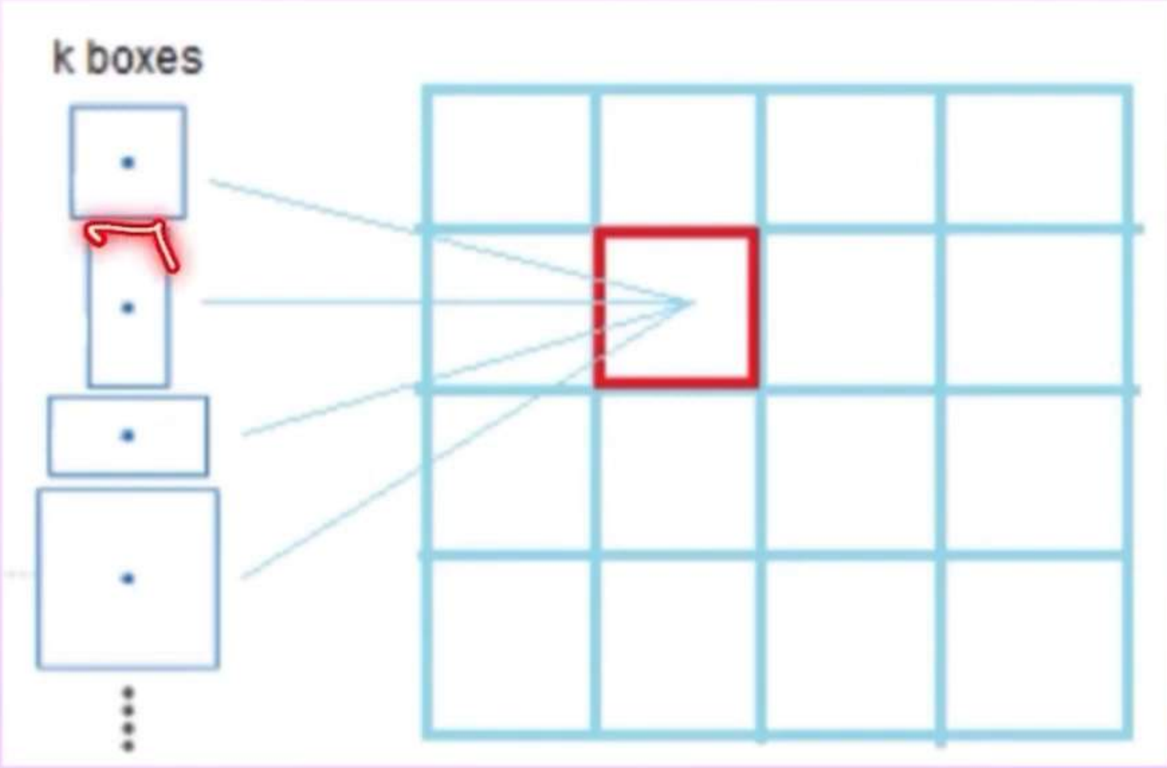
Reason for Poor Localization

- Boxes are learnt relative to grid cell
- Objects can be of different shapes
- Bounding box regression is difficult
- Solution: Anchor boxes



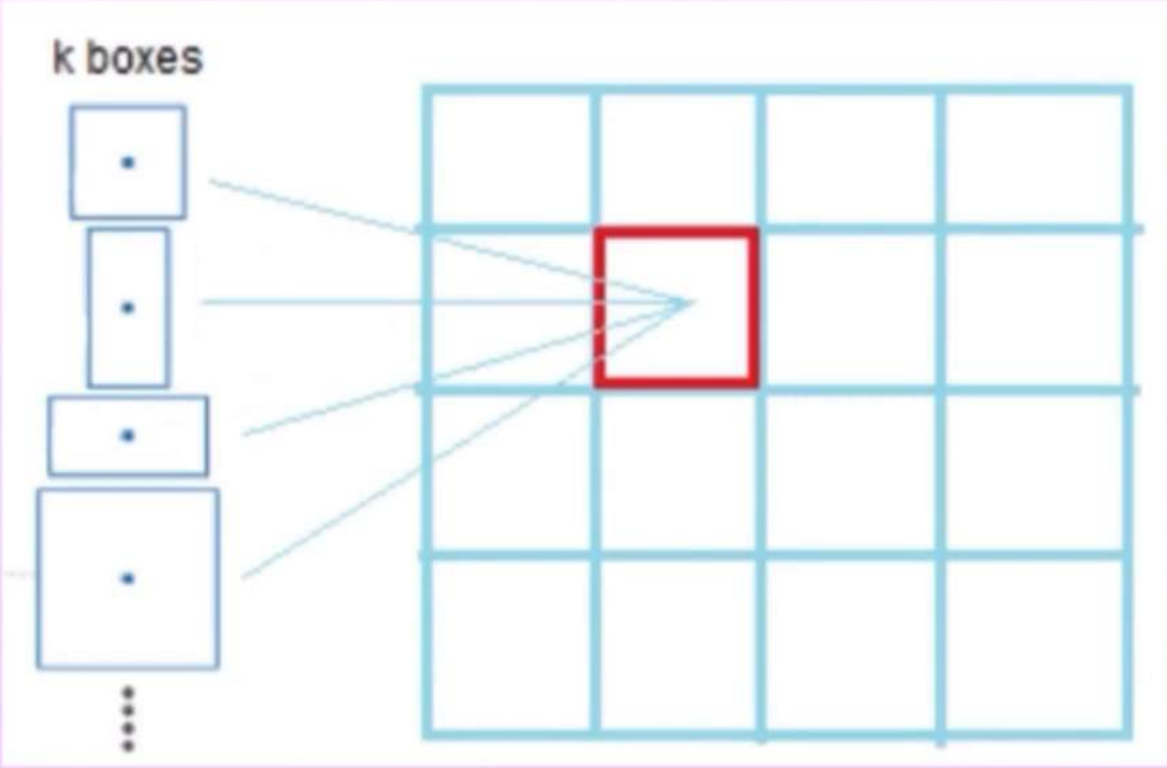
Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors



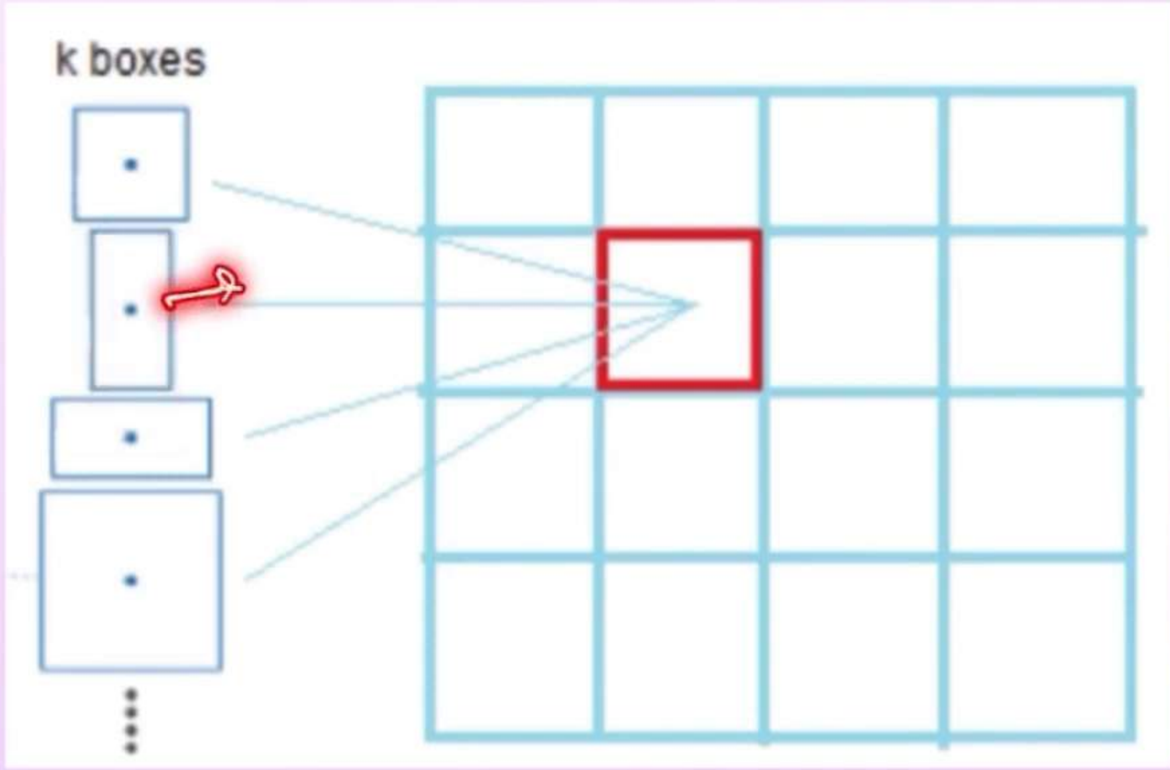
Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors



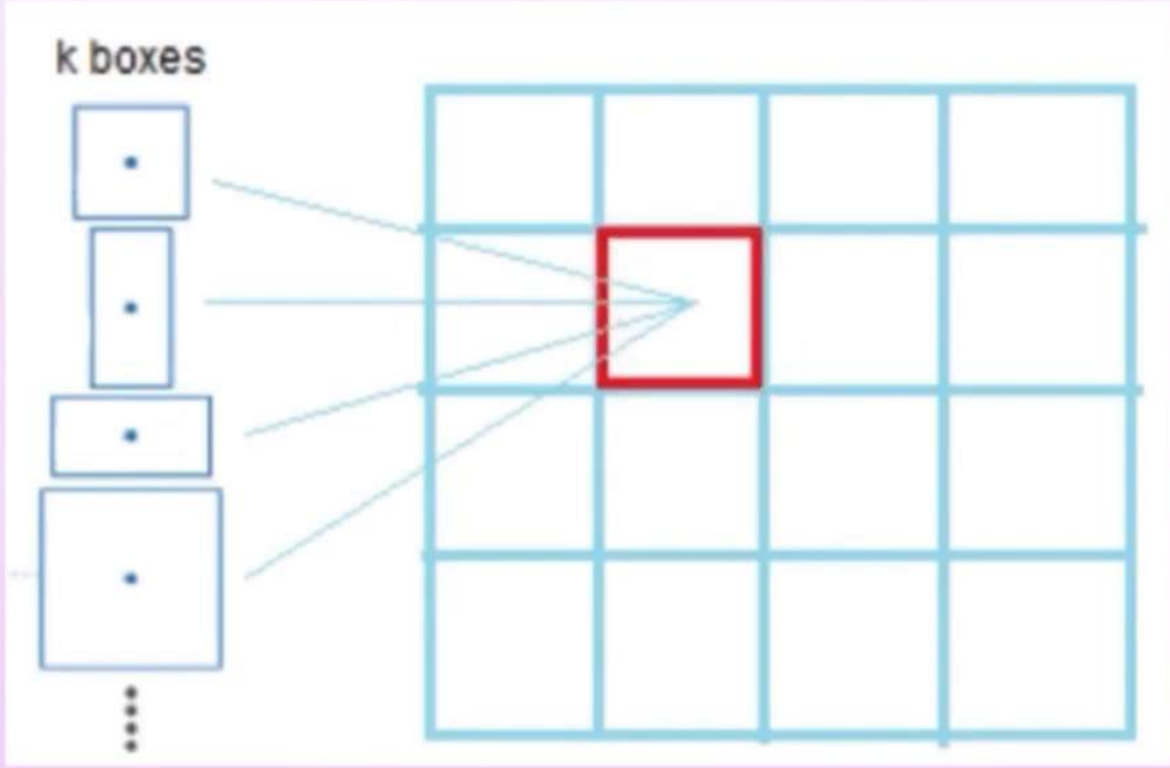
Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors



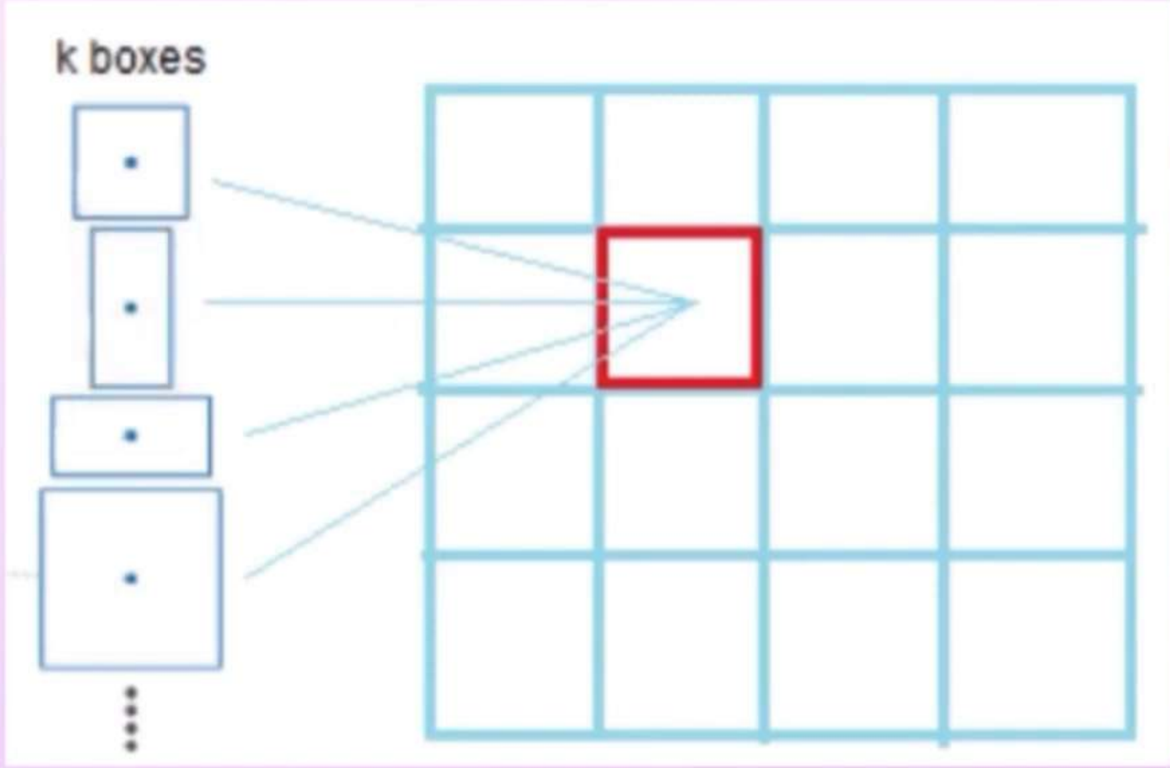
Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors



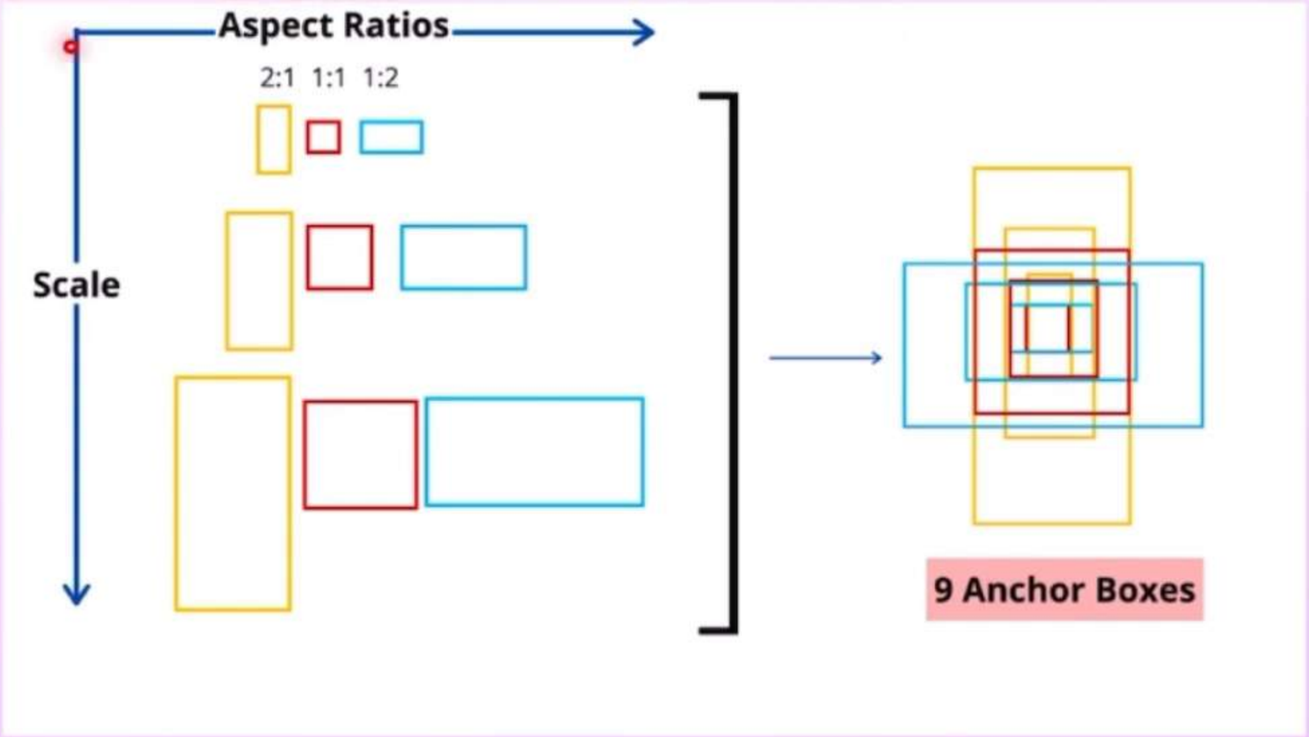
Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors



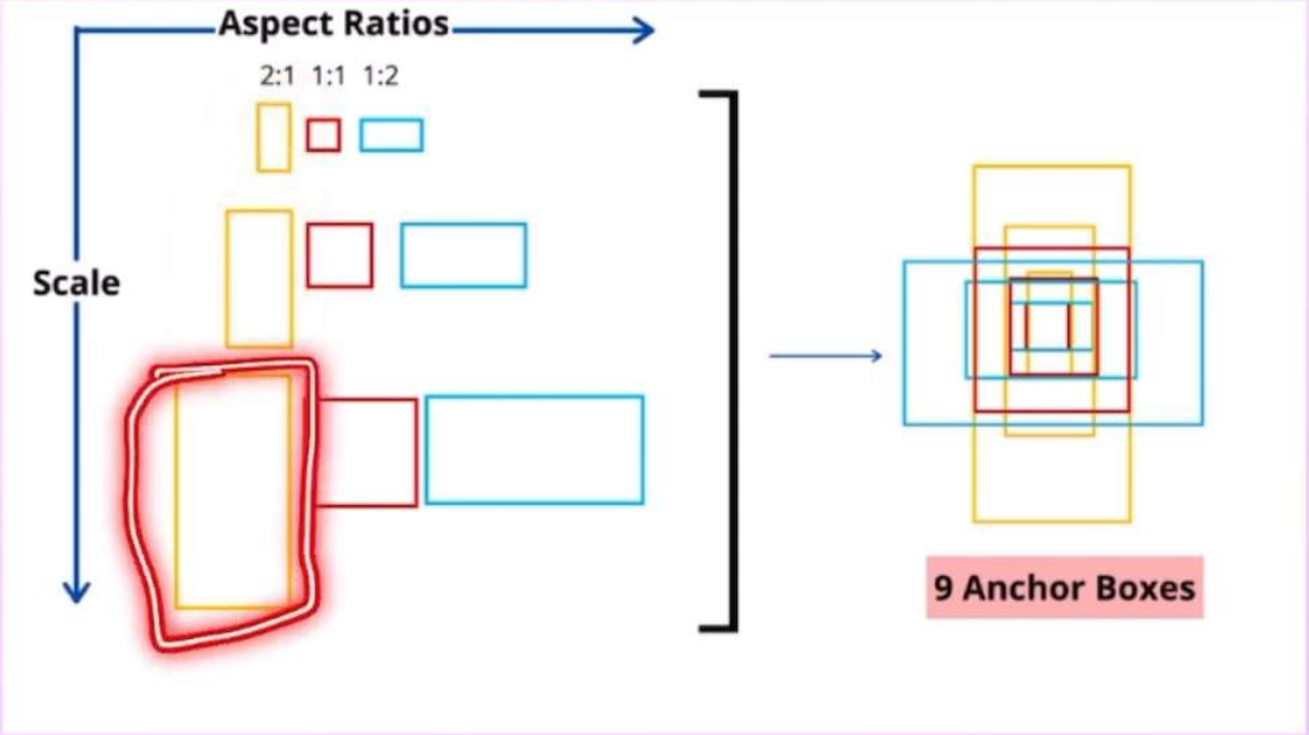
Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors
- Faster-RCNN used 9 anchor boxes at each location



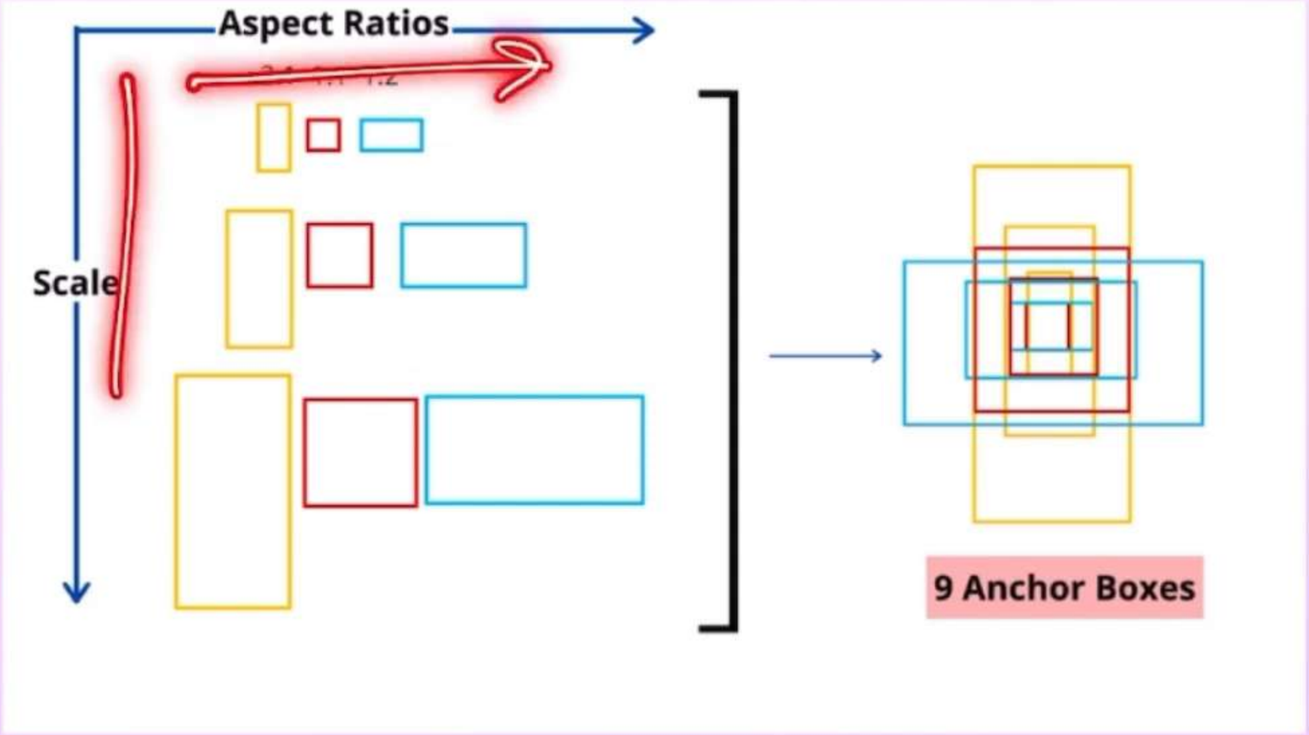
Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors
- Faster-RCNN used 9 anchor boxes at each location



Anchor boxes

- Predefined set of shapes
- Bounding boxes will be relative to these anchors
- Faster-RCNN used 9 anchor boxes at each location



Anchor boxes

- Yolo adopted the anchors concept
- mAP decreased by 0.3% - 69.5 to 69.2
- Recall has improved from 81% to 88%
- Increase in recall indicates that the model has more room to improve

Anchor boxes

- Yolo adopted the anchors concept
- mAP decreased by 0.3% - 69.5 to 69.2
- Recall has improved from 81% to 88%
- Increase in recall indicates that the model has more room to improve

Anchor boxes

- Yolo adopted the anchors concept
- mAP decreased by 0.3% - 69.5 to 69.2
- Recall has improved from 81% to 88%
- Increase in recall indicates that the model has more room to improve

Anchor boxes

- Yolo adopted the anchors concept
- mAP decreased by 0.3% - 69.5 to 69.2
- Recall has improved from 81% to 88%
- Increase in recall indicates that the model has more room to improve
- How can we increase mAP?
- Solution: Choose anchors based on the dataset

Anchor boxes

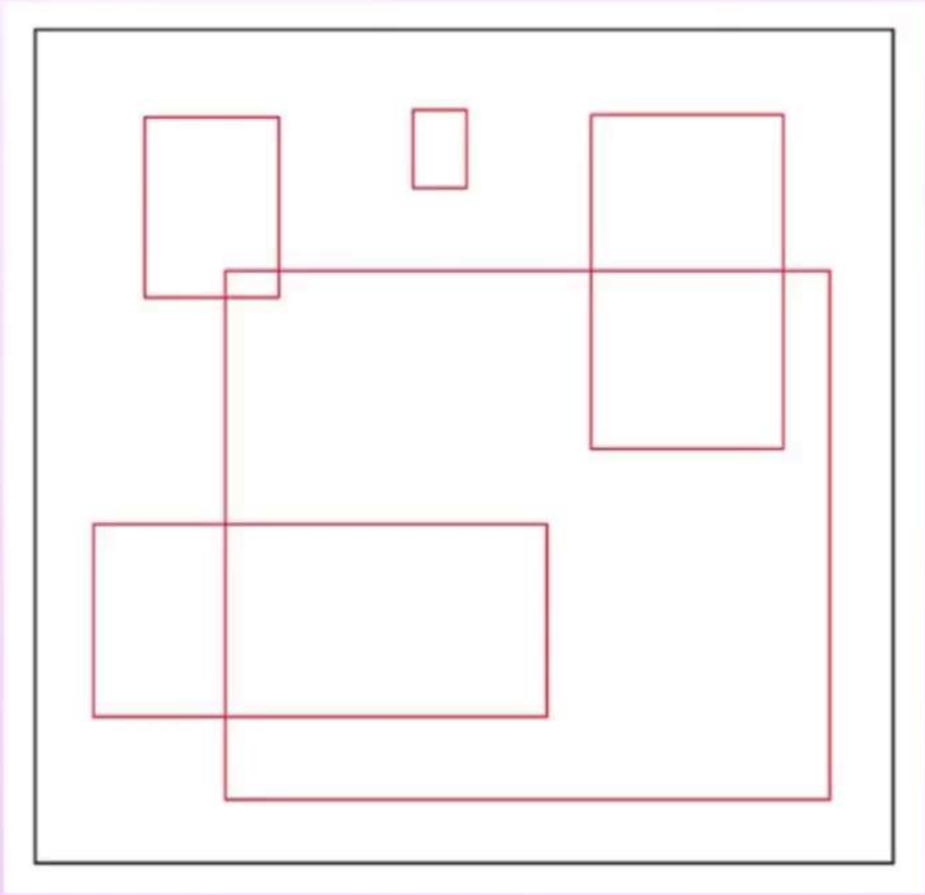
- Yolo adopted the anchors concept
- mAP decreased by 0.3% - 69.5 to 69.2
- Recall has improved from 81% to 88%
- Increase in recall indicates that the model has more room to improve
- How can we increase mAP?
- Solution: Choose anchors based on the dataset

Anchor boxes

- Yolo adopted the anchors concept
- mAP decreased by 0.3% - 69.5 to 69.2
- Recall has improved from 81% to 88%
- Increase in recall indicates that the model has more room to improve
- How can we increase mAP?
- Solution: Choose anchors based on the dataset

Anchor boxes - Clustering

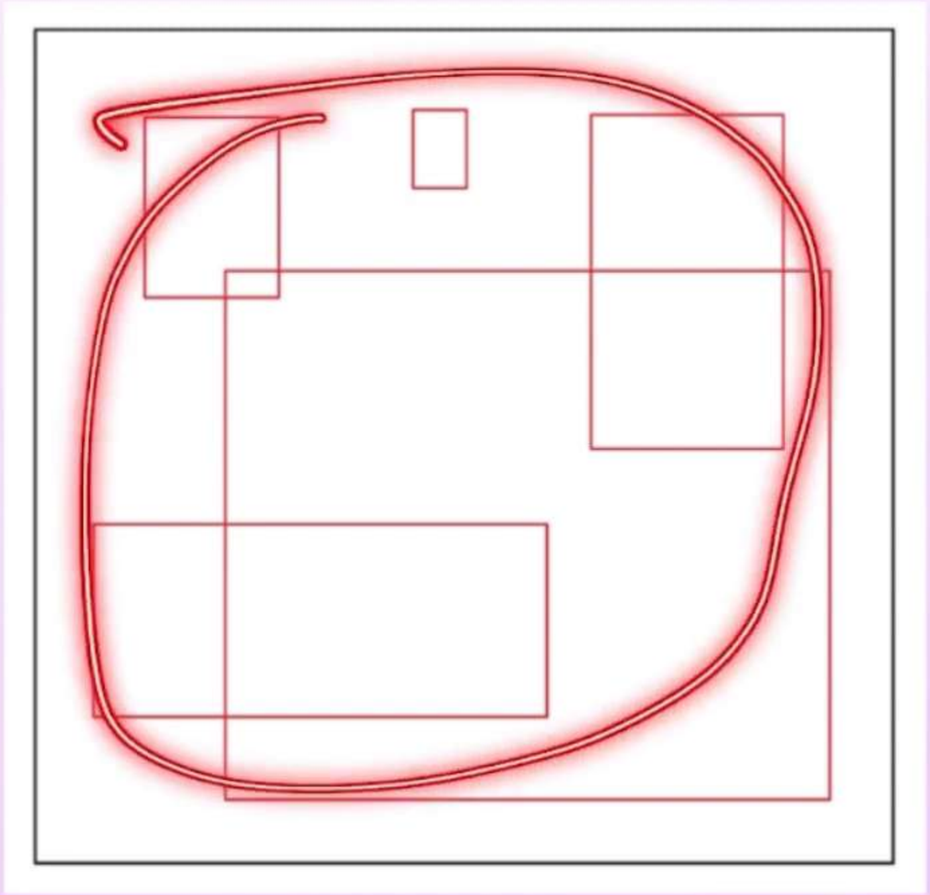
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

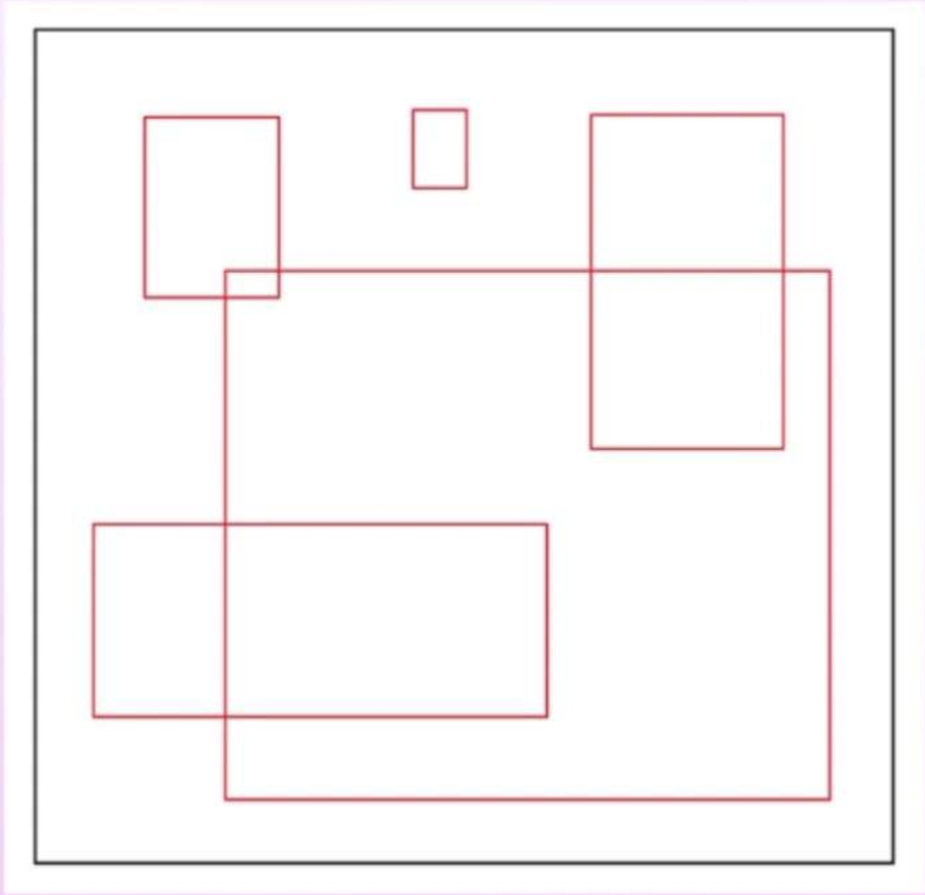
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

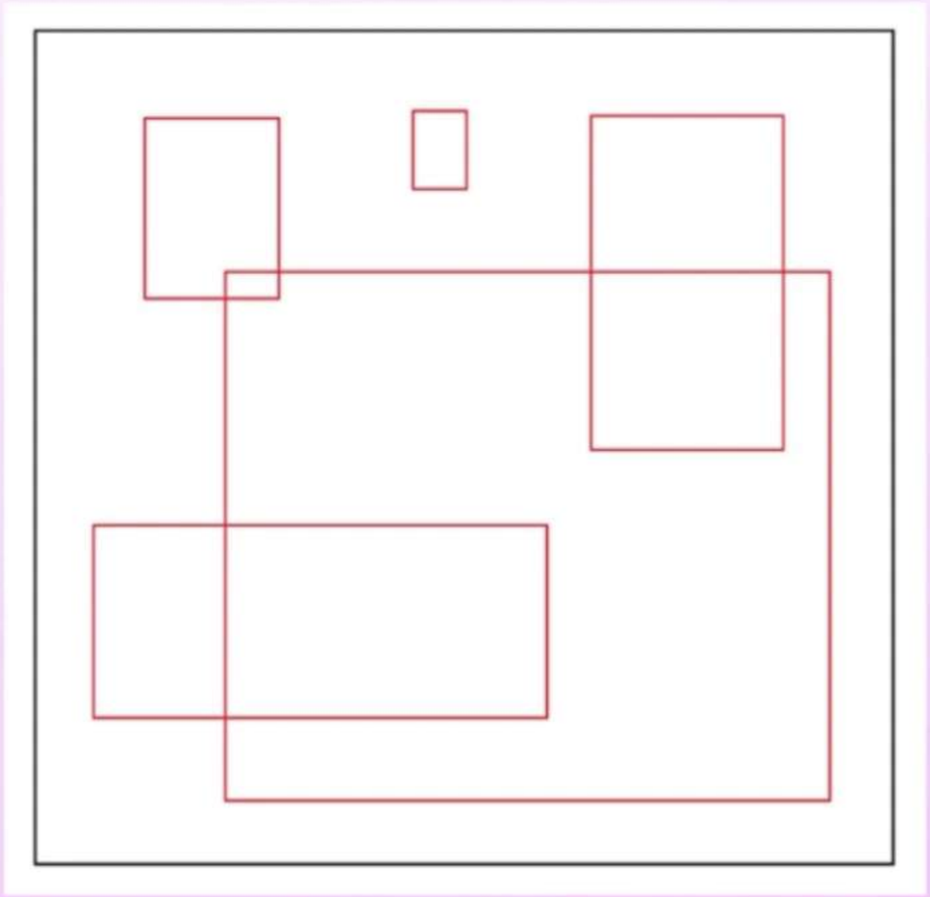
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

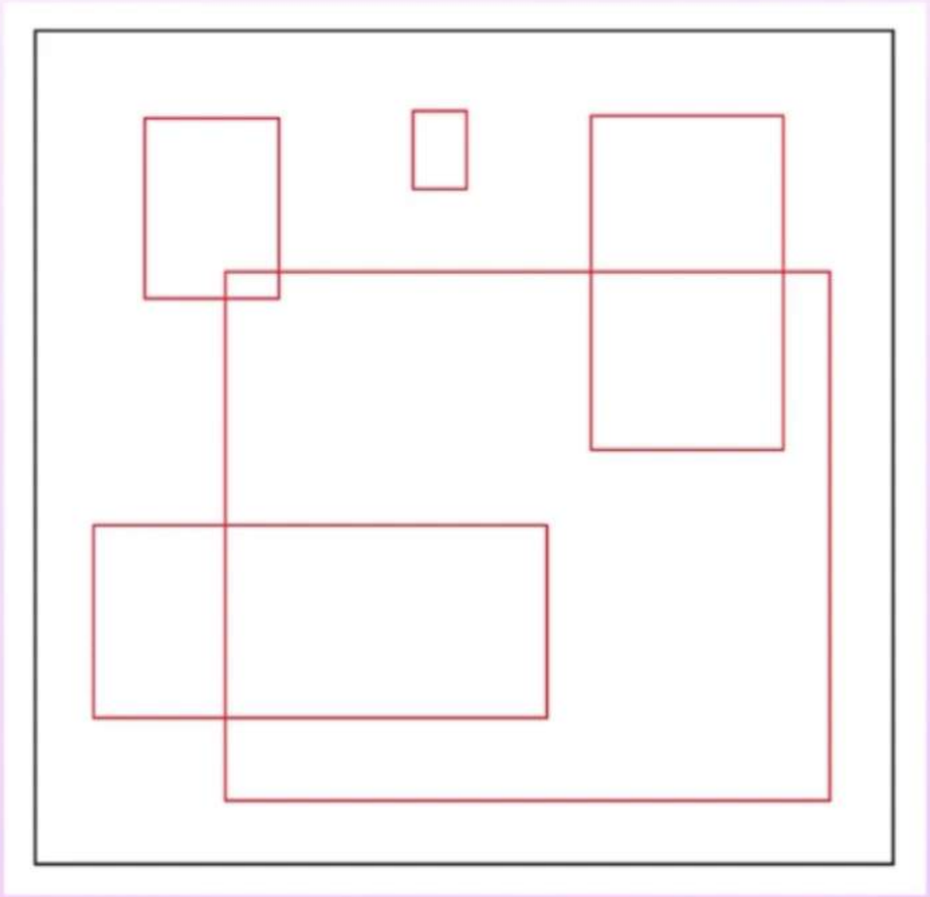
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

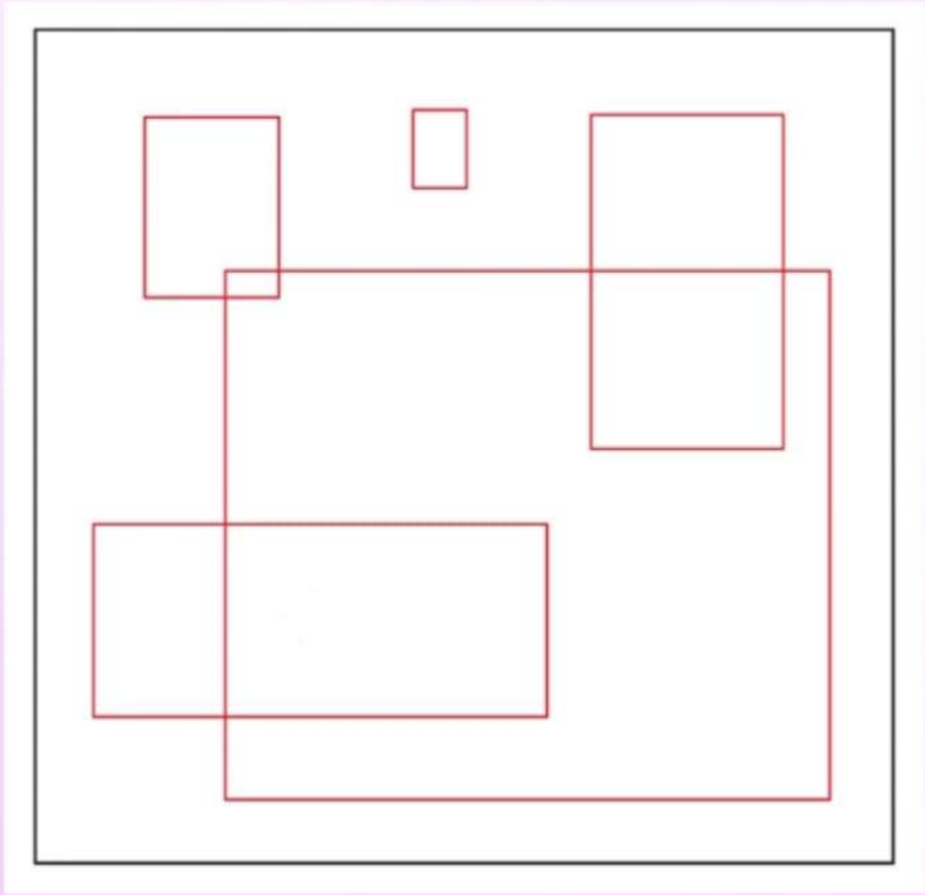
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

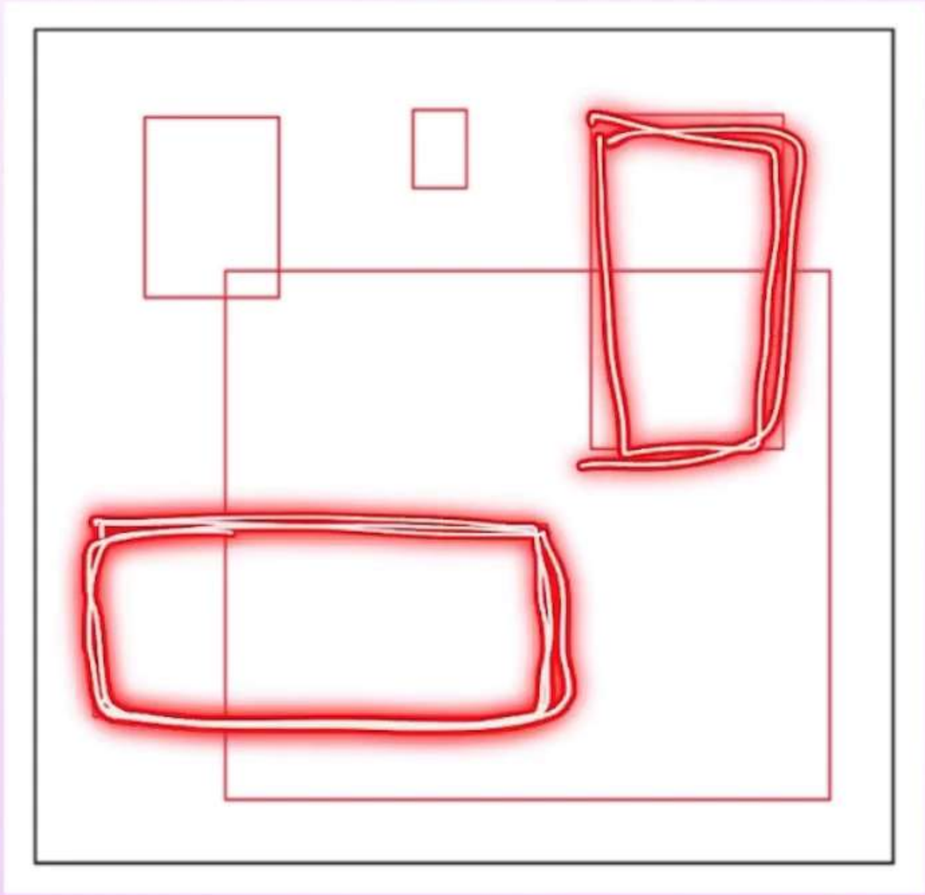
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

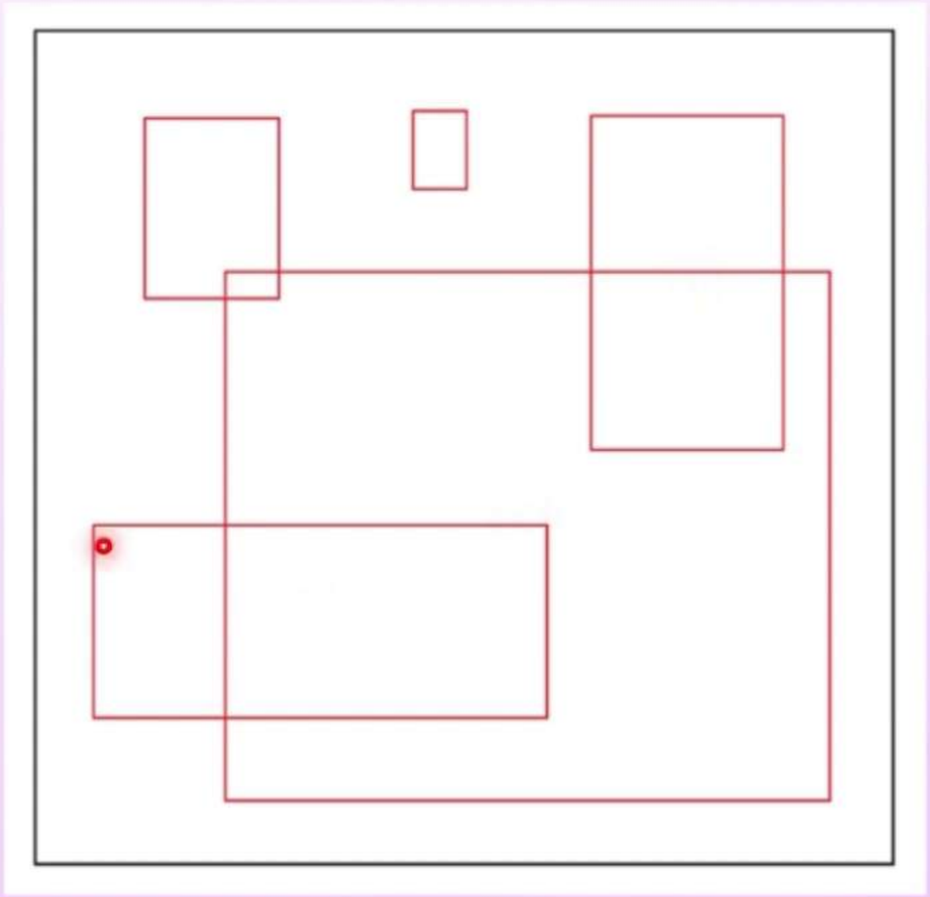
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

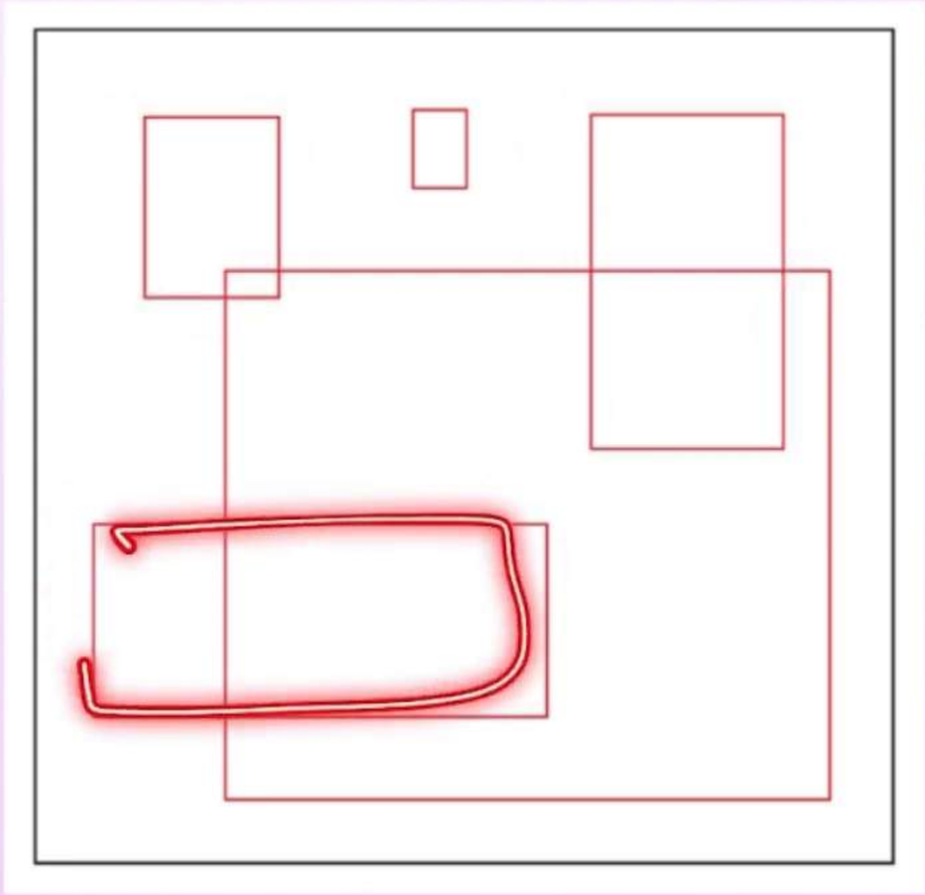
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

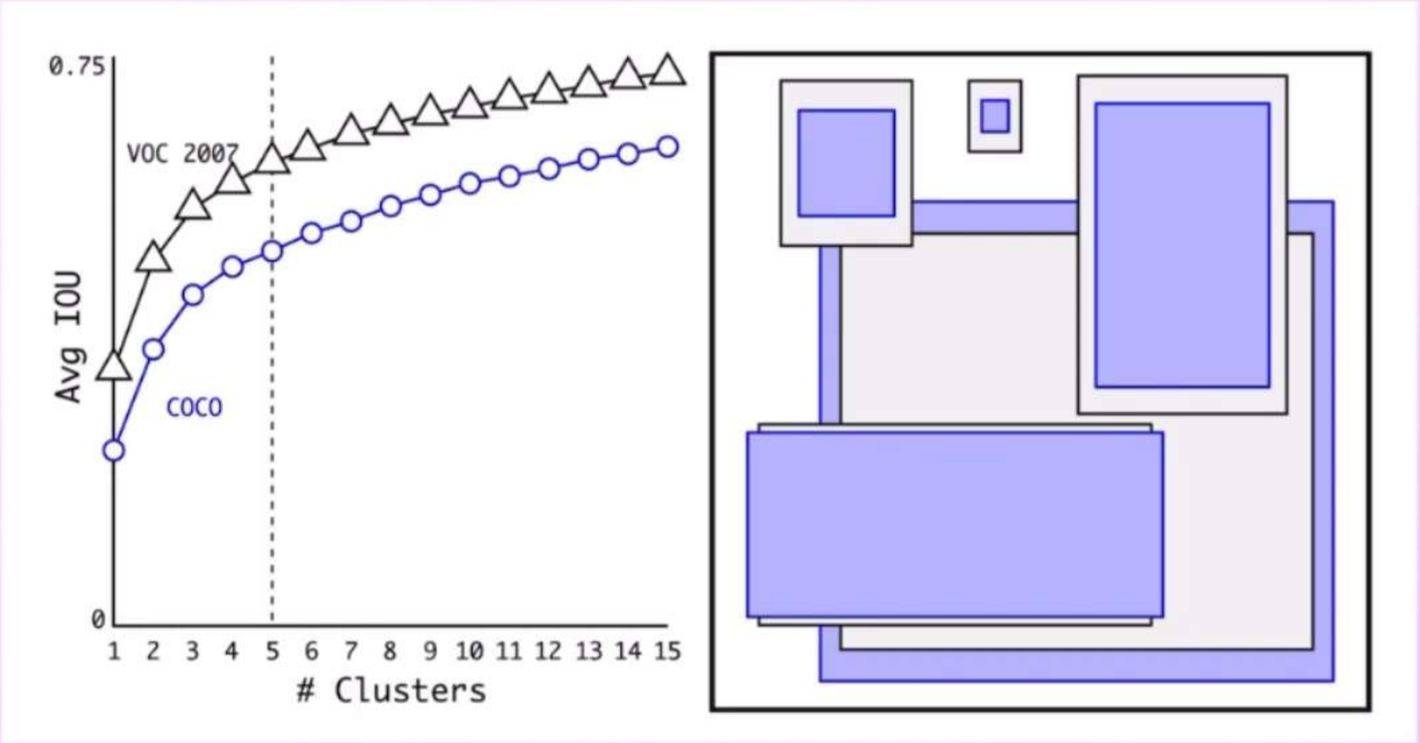
- Instead of predefined boxes, Choose the shapes depending on dataset
- K-means clustering for getting the average shapes and locations



Average shapes and locations of objects for pascal voc dataset

Anchor boxes - Clustering

- K-means clustering for getting the average shapes and locations
- On Pascal VOC and COCO
- 5 boxes are selected



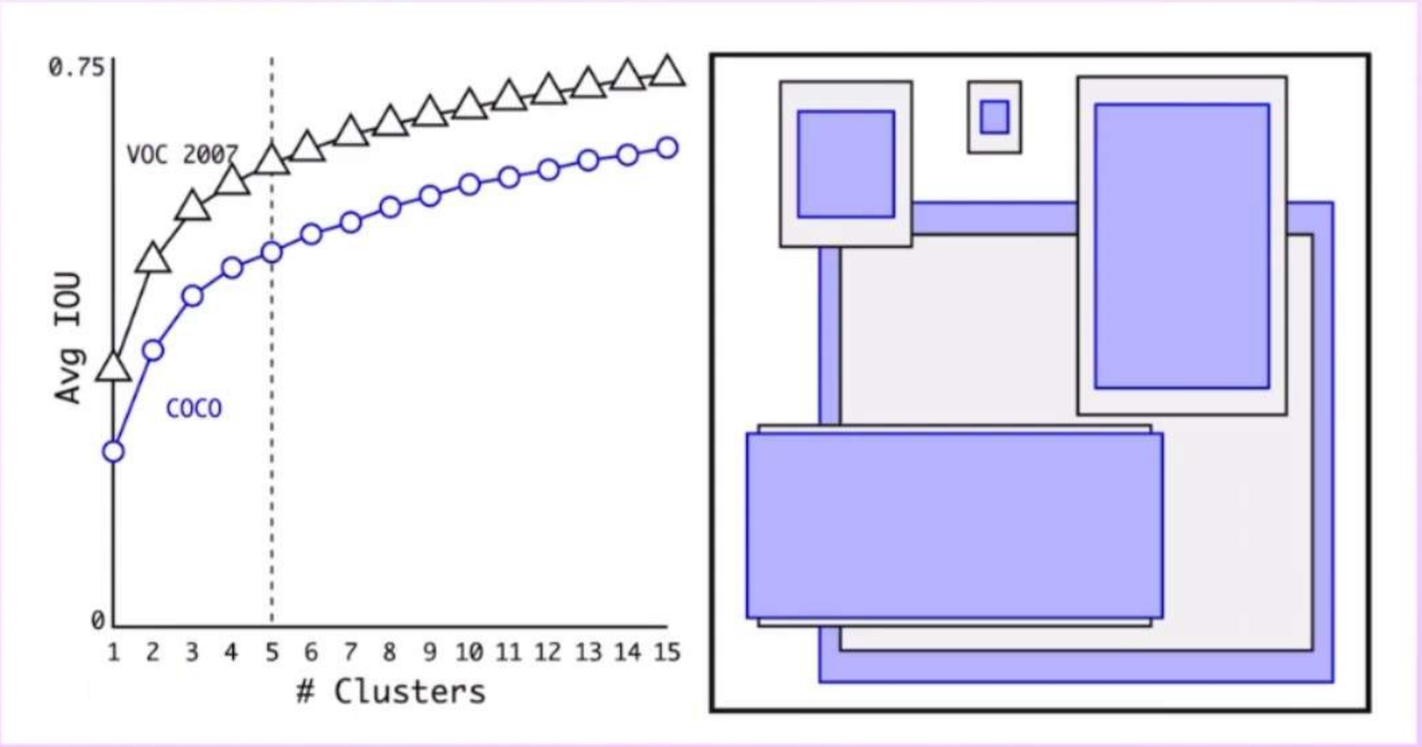
Anchor boxes - Clustering

- Faster-RCNN - 9 anchors
- Yolo-v2 - 5 anchors
- Reduced computations by almost half

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2

Anchor boxes - Clustering

- K-means clustering for getting the average shapes and locations
- On Pascal VOC and COCO
- 5 boxes are selected



Anchor boxes - Clustering

- Faster-RCNN - 9 anchors
- Yolo-v2 - 5 anchors
- Reduced computations by almost half

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2

Problem with bounding boxes

- YOLO-V1 uses linear activation in the final layer - regression
- Predictions range - [-inf to +inf]
- Targets relative to grid cell - normalized to range of [0,1]

Problem with bounding boxes

- YOLO-V1 uses linear activation in the final layer - regression
- Predictions range - [-inf to +inf]
- Targets relative to grid cell - normalized to range of [0,1]

Problem with bounding boxes

- YOLO-V1 uses linear activation in the final layer - regression
- Predictions range - [-inf to +inf]
- Targets relative to grid cell - normalized to range of [0,1]

Problem with bounding boxes

- YOLO-V1 uses linear activation in the final layer - regression
- Predictions range - [-inf to +inf]
- Targets relative to grid cell - normalized to range of [0,1]

Problem with bounding boxes

- YOLO-V1 uses linear activation in the final layer - regression
- Predictions range - [-inf to +inf]
- Targets relative to grid cell - normalized to range of [0,1]
- Loss - Very high
- Unstable training

Problem with bounding boxes

- YOLO-V1 uses linear activation in the final layer - regression
- Predictions range - [-inf to +inf]
- Targets relative to grid cell - normalized to range of [0,1]
- Loss - Very high
- Unstable training

Constrained predictions

- Used logistic activation on location coordinates
- Range - [0,1]
- ***t_x = sigmoid(t_x), t_y = sigmoid(t_y)***
- Used exponential terms of width and height predictions
- Range - [0, +inf]
- ***t_w = exp(t_w), t_h = exp(t_h)***

Constrained predictions

- Used logistic activation on location coordinates
- Range - [0,1]
- ***t_x = sigmoid(t_x), t_y = sigmoid(t_y)***
- Used exponential terms of width and height predictions
- Range - [0, +inf]
- ***t_w = exp(t_w), t_h = exp(t_h)***

Constrained predictions

- Used logistic activation on location coordinates
- Range - [0,1]
- ***t_x = sigmoid(t_x), t_y = sigmoid(t_y)***
- Used exponential terms of width and height predictions
- Range - [0, +inf]
- ***t_w = exp(t_w), t_h = exp(t_h)***

Constrained predictions

- Used logistic activation on location coordinates
- Range - [0,1]
- ***t_x = sigmoid(t_x), t_y = sigmoid(t_y)***
- Used exponential terms of width and height predictions
- Range - [0, +inf]
- ***t_w = exp(t_w), t_h = exp(t_h)***

Constrained predictions

- Used logistic activation on location coordinates
- Range - [0,1]
- ***t_x = sigmoid(t_x), t_y = sigmoid(t_y)***
- Used exponential terms of width and height predictions
- Range - [0, +inf]
- ***t_w = exp(t_w), t_h = exp(t_h)***

Problem with bounding boxes

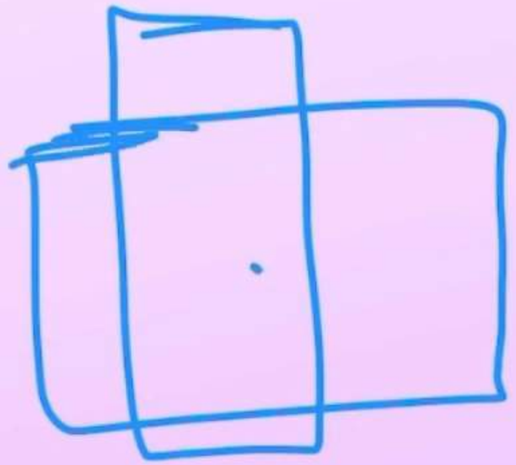
- In Faster-RCNN, location offsets ***t_x, t_y*** are calculated relative to anchor boxer
- ***x_a, y_a, w_a, h_a*** are anchor box values
- Adopting this caused instability in training during early iterations

$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

Problem with bounding boxes

- In Faster-RCNN, location offsets ***t_x, t_y*** are calculated relative to anchor boxer
- ***x_a, y_a, w_a, h_a*** are anchor box values
- Adopting this caused instability in training during early iterations

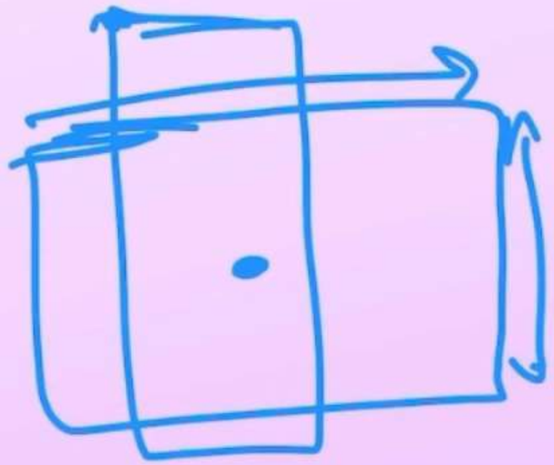
$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$



Problem with bounding boxes

- In Faster-RCNN, location offsets ***t_x, t_y*** are calculated relative to anchor boxer
- ***x_a, y_a, w_a, h_a*** are anchor box values
- Adopting this caused instability in training during early iterations

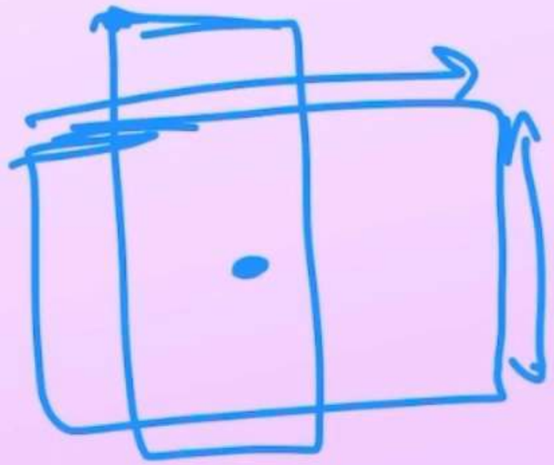
$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$



Problem with bounding boxes

- In Faster-RCNN, location offsets ***t_x, t_y*** are calculated relative to anchor boxer
- ***x_a, y_a, w_a, h_a*** are anchor box values
- Adopting this caused instability in training during early iterations

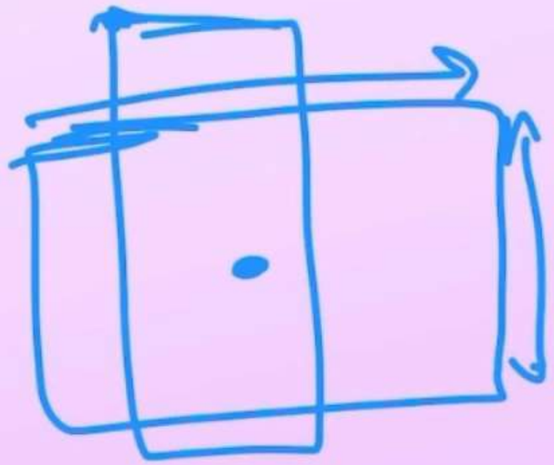
$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$



Problem with bounding boxes

- In Faster-RCNN, location offsets ***t_x, t_y*** are calculated relative to anchor boxer
- ***x_a, y_a, w_a, h_a*** are anchor box values
- Adopting this caused instability in training during early iterations

$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$



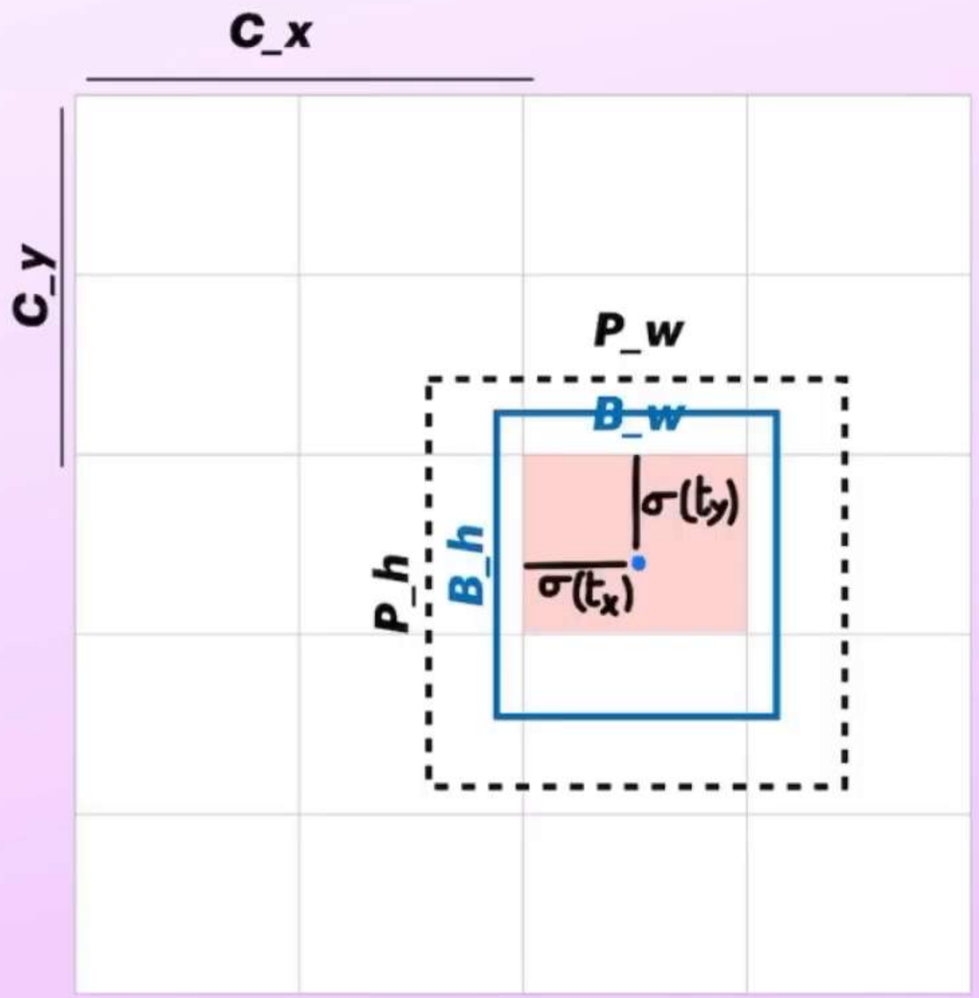
Problem with bounding boxes

- In Faster-RCNN, location offsets ***t_x, t_y*** are calculated relative to anchor boxer
- ***x_a, y_a, w_a, h_a*** are anchor box values
- Adopting this caused instability in training during early iterations
- Solution: Adopted grid based predictions for location coordinates ***t_x, t_y***

$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

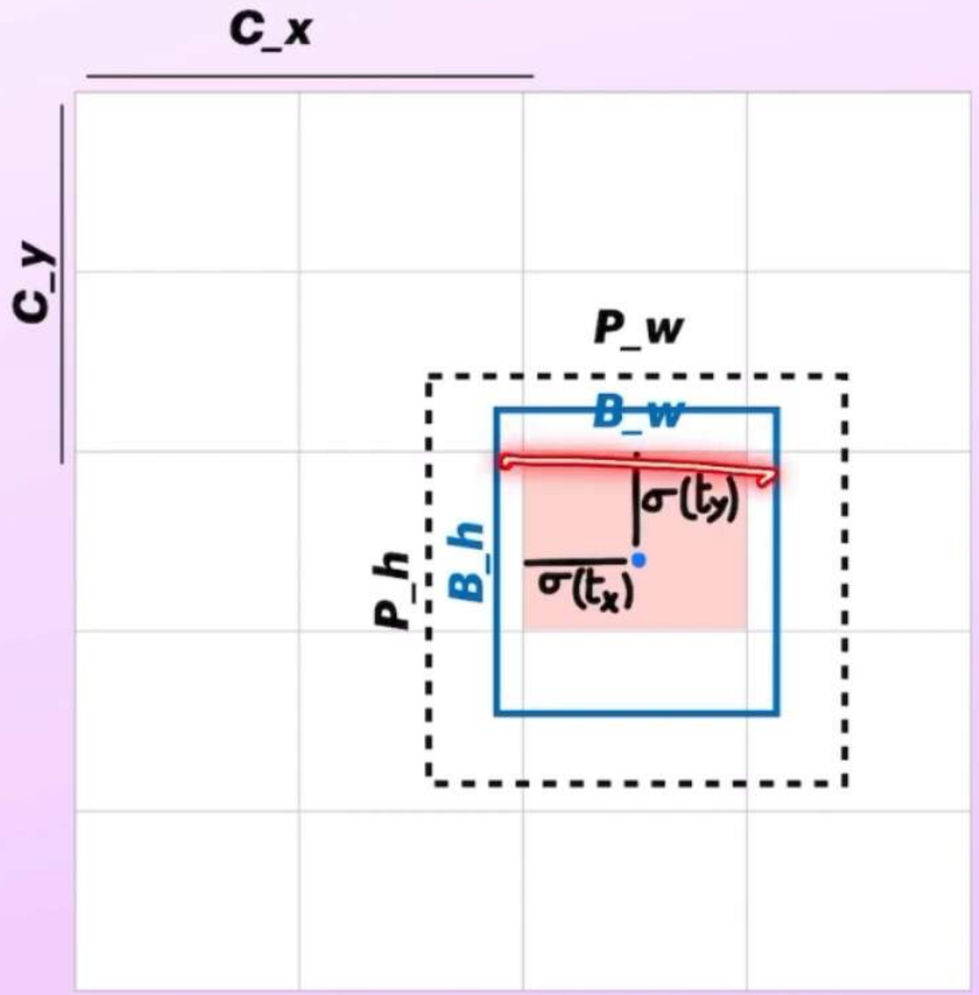


——————— Anchor box

——————— Prediction

Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

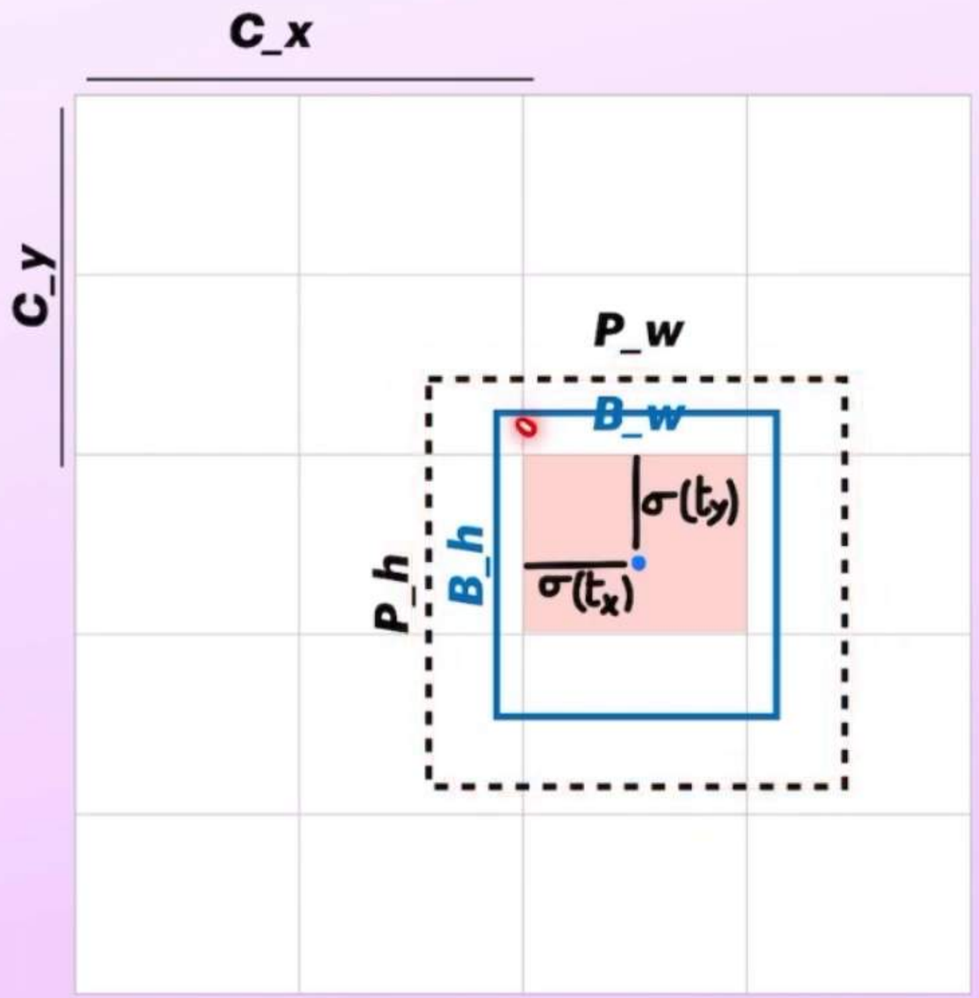


——————— Anchor box

——————— Prediction

Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

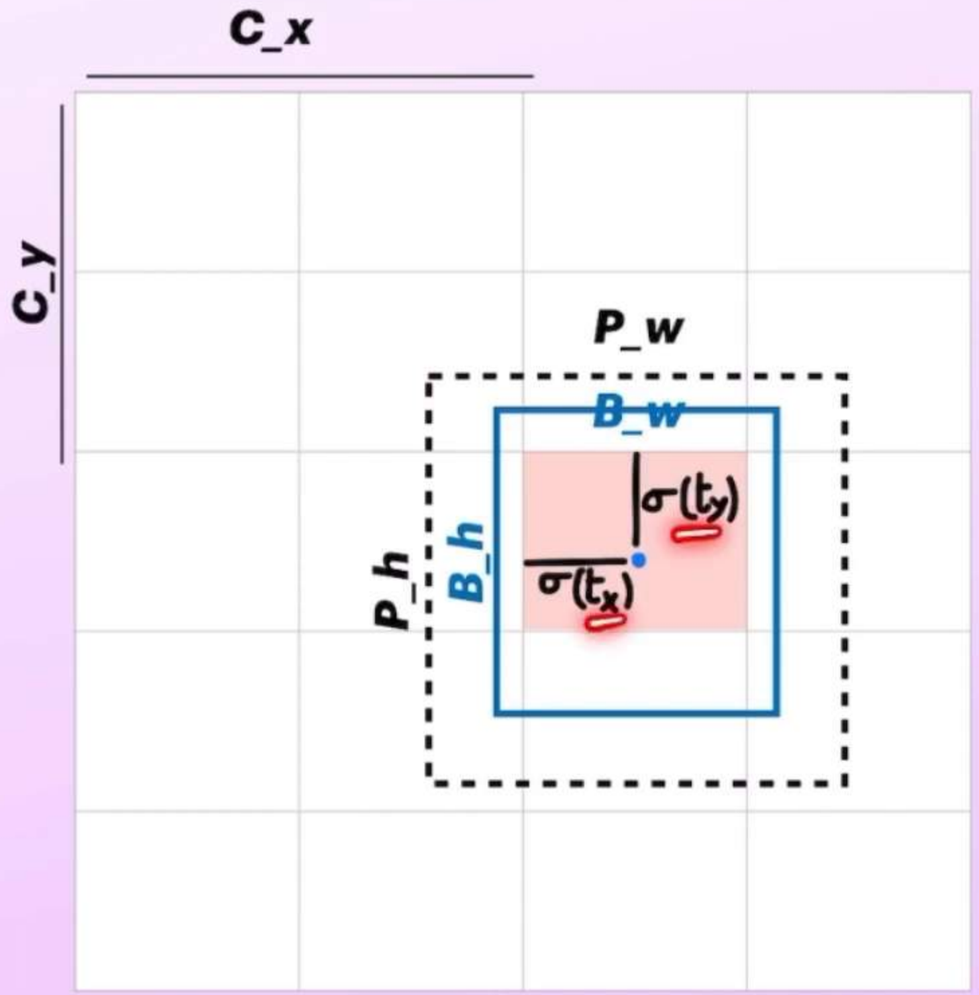


——————— Anchor box

——————— Prediction

Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

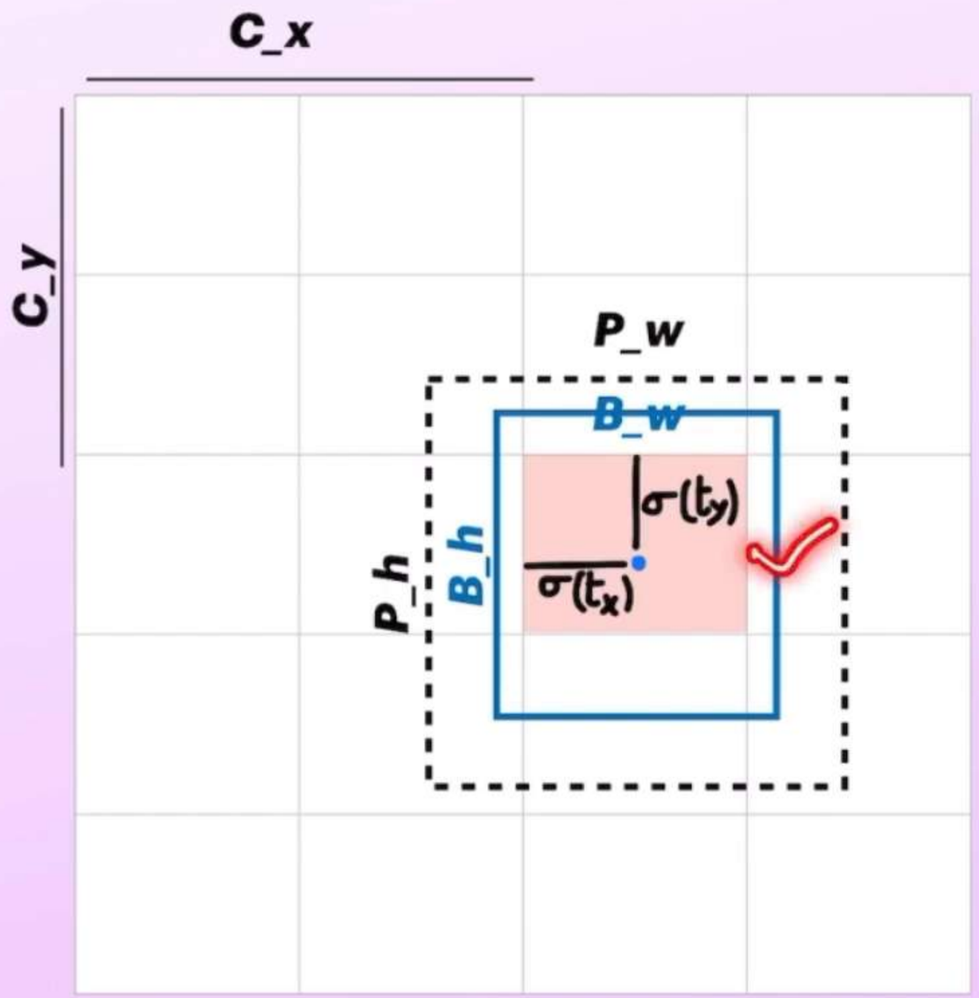


——————— Anchor box

——————— Prediction

Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***



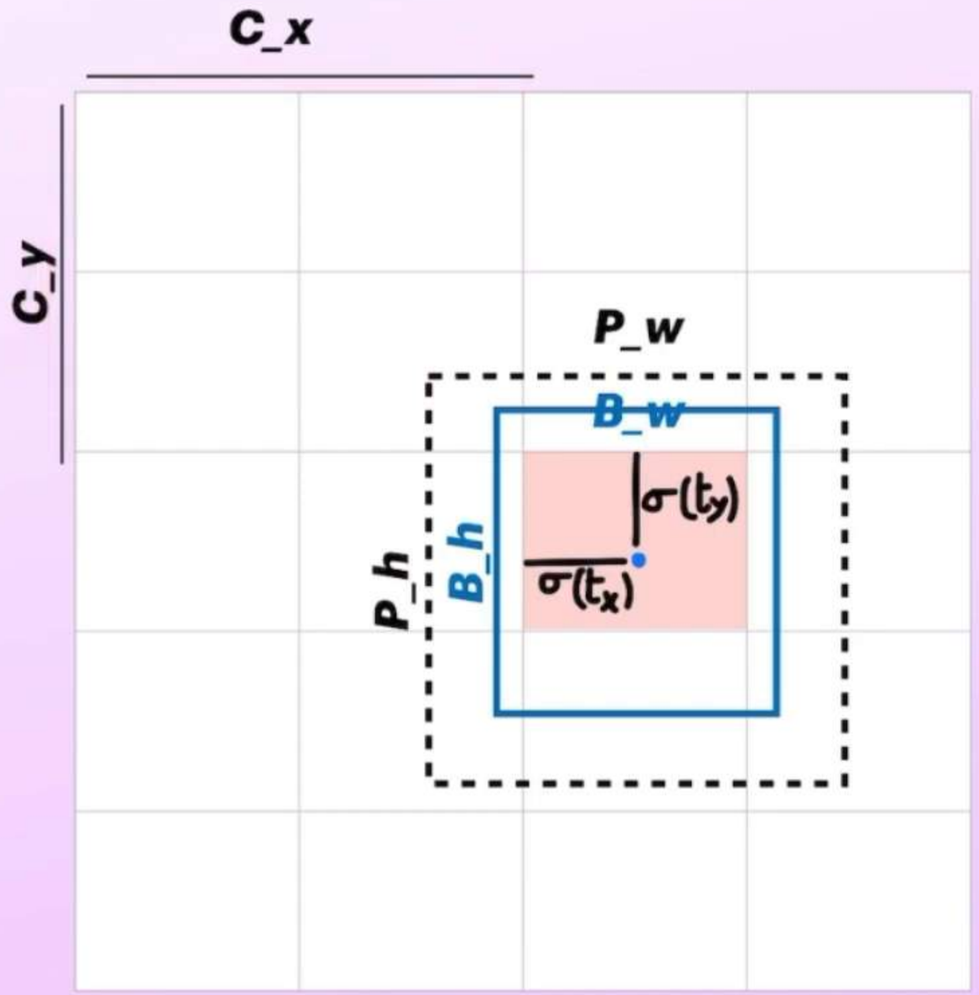
——————— Anchor box

——————— Prediction

Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

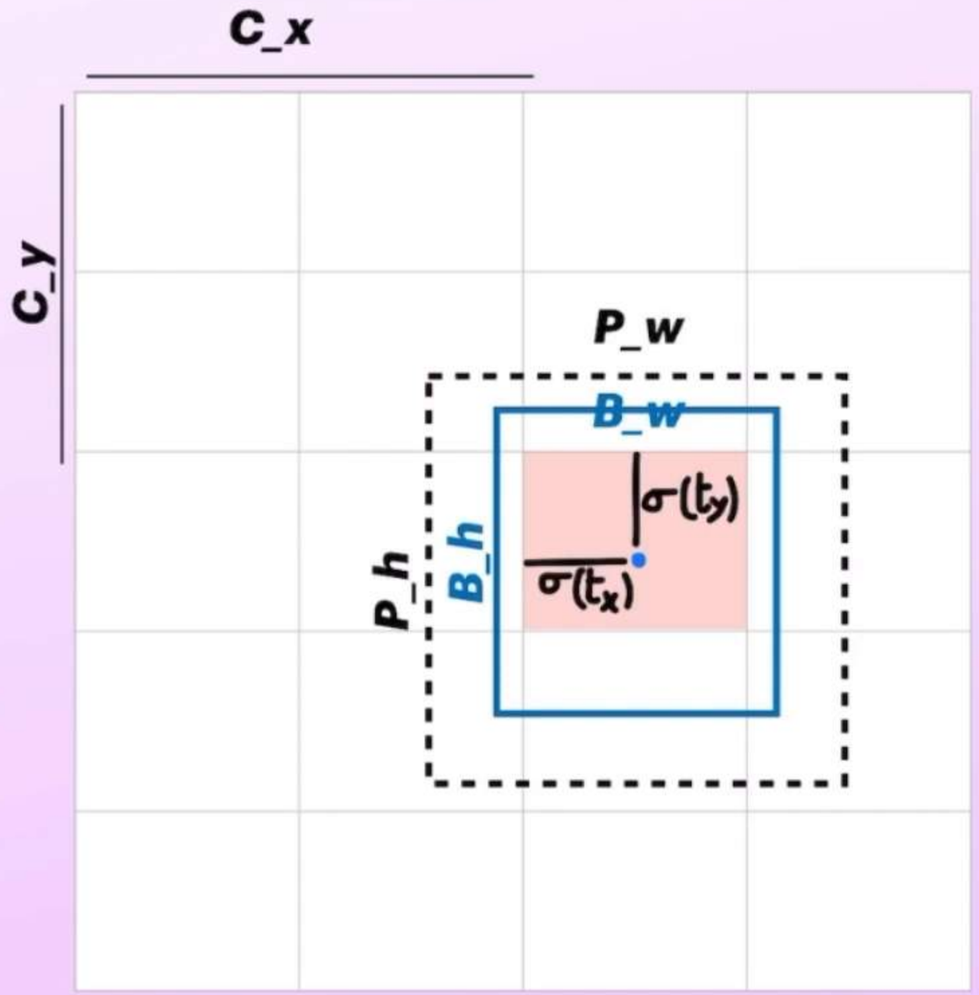
$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$



Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

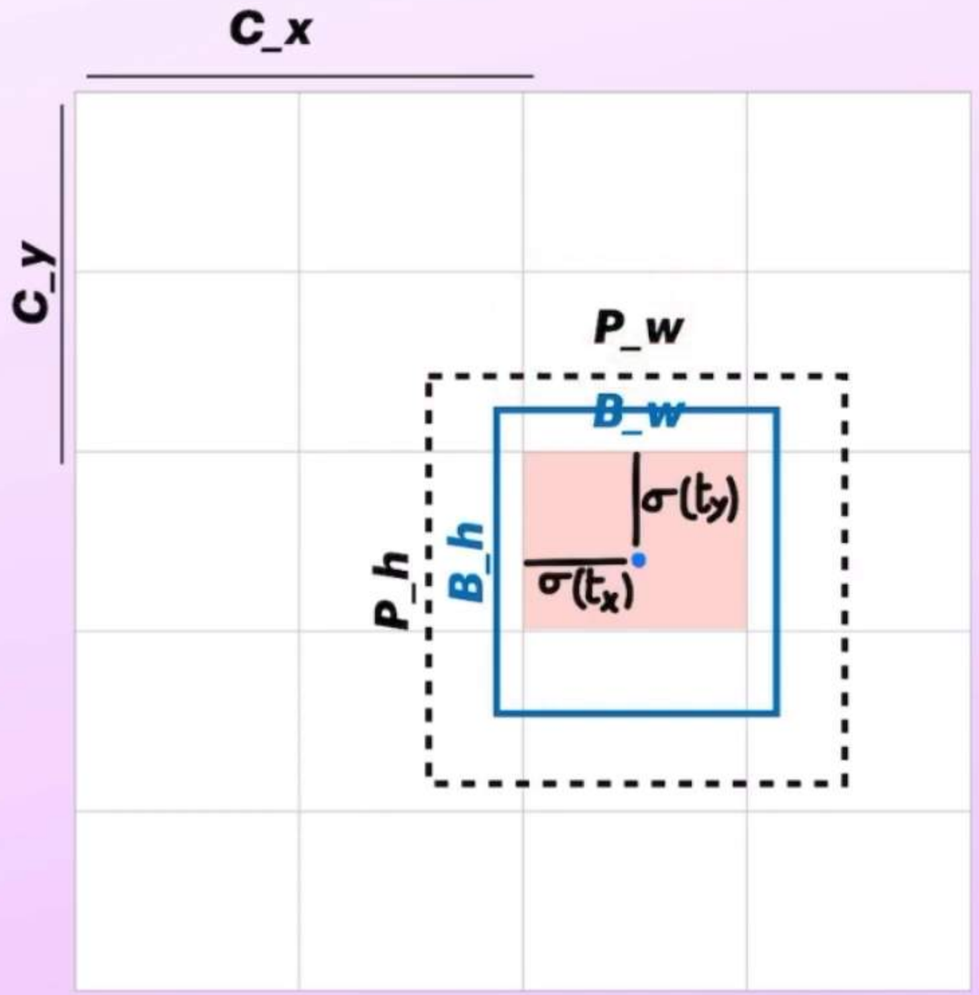
$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$



Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

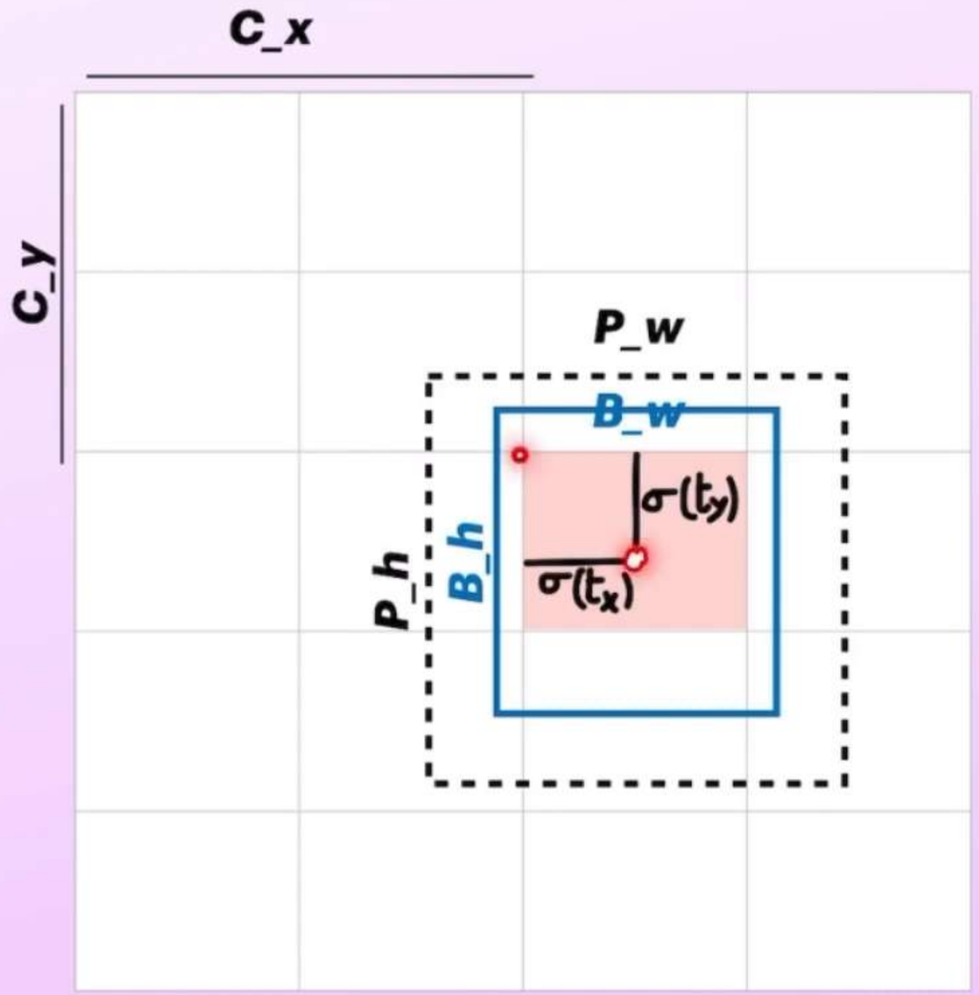
$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$



Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

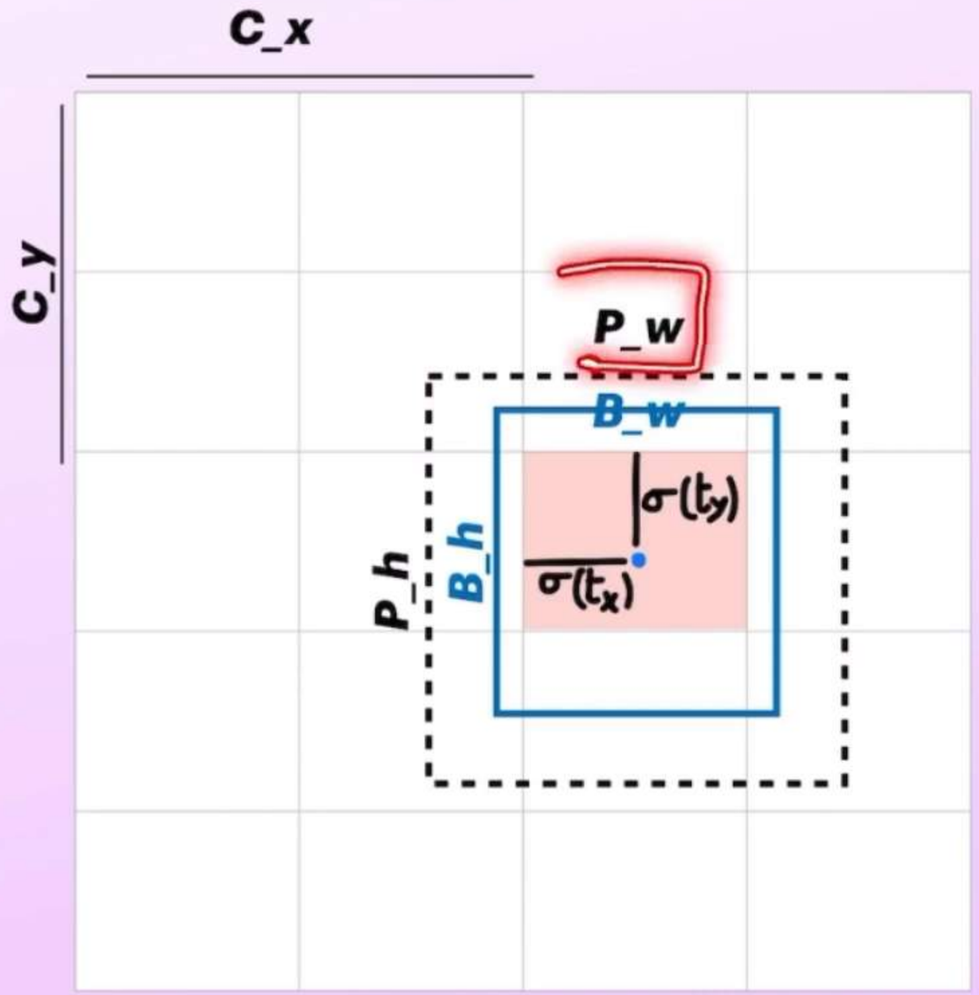
$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$



Modified Box Predictions

- ***C_x, C_y*** are cell position from top-left corner of the image
- ***P_w, P_h*** are width and height of anchor box
- Network Predictions: ***t_x, t_y, t_w, t_h***

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

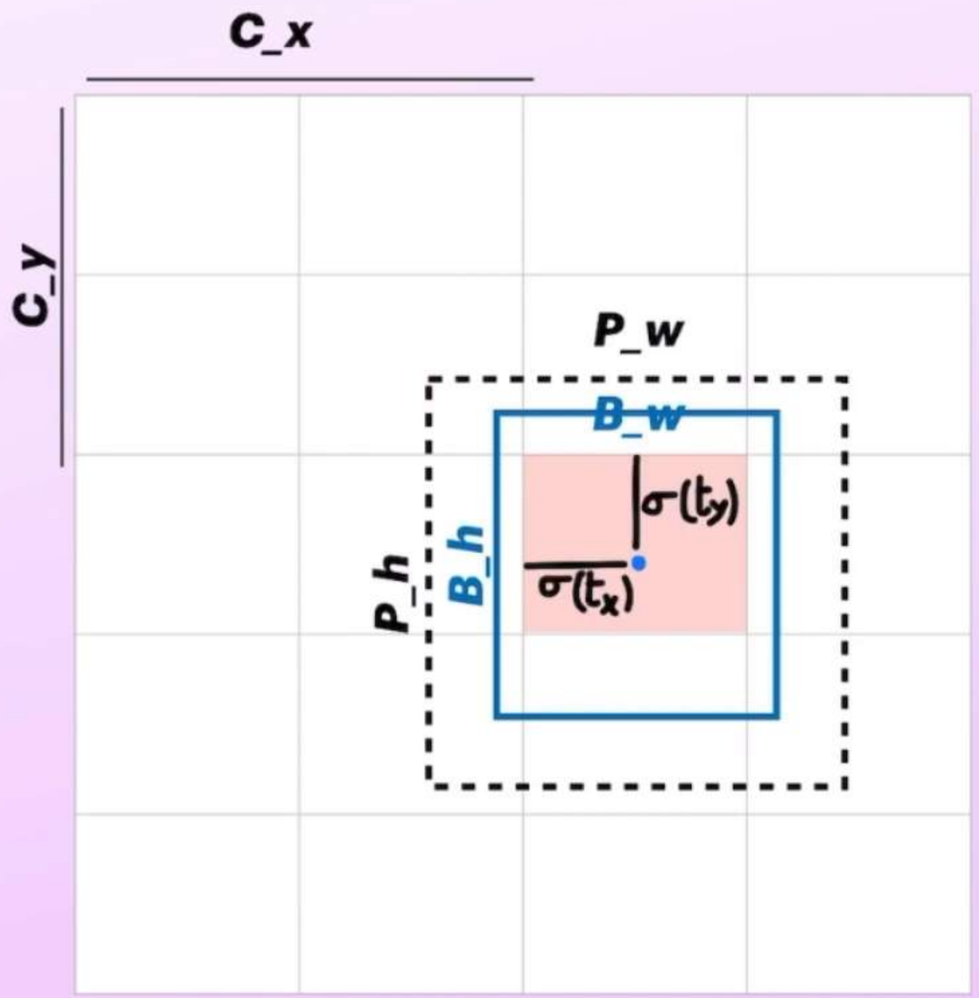


Modified Box Predictions

- Network Predictions: **t_x, t_y, t_w, t_h, t_o**
- Objectness score: **t_o**

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

- Class probabilities for each box

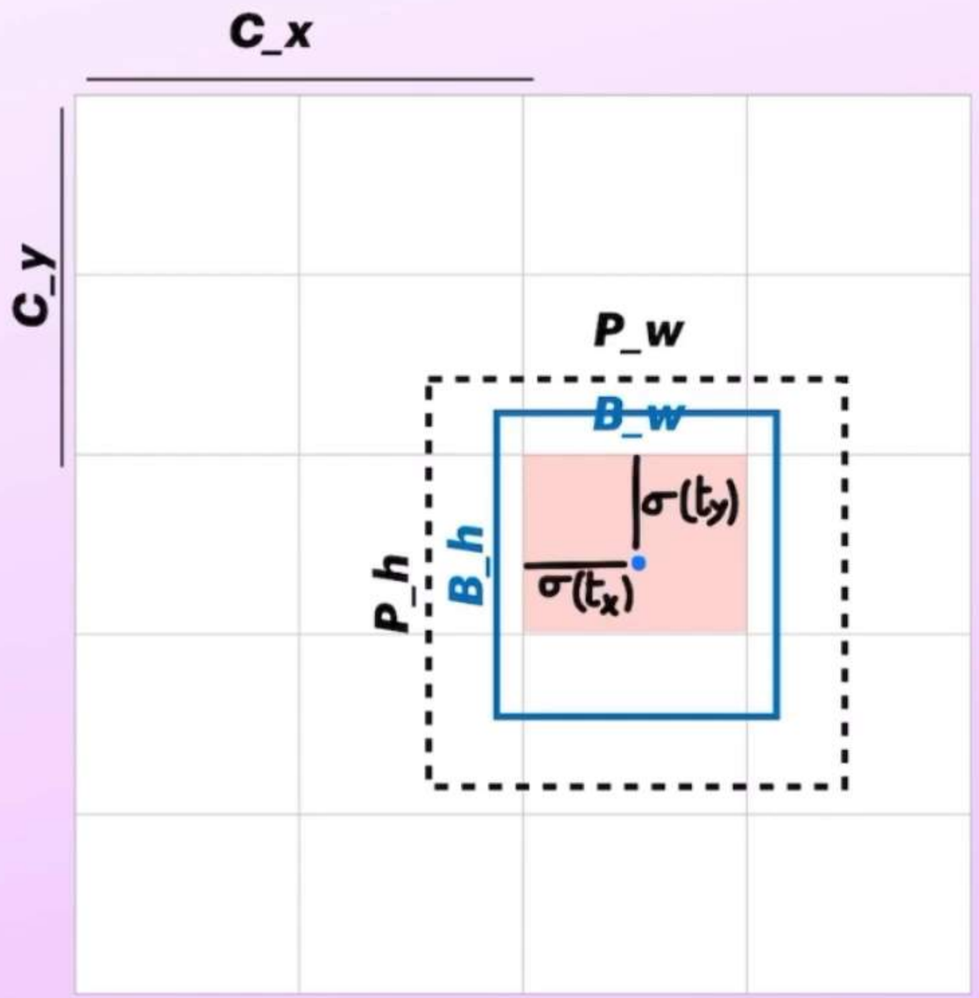


Modified Box Predictions

- Network Predictions: ***t_x, t_y, t_w, t_h, t_o***
- Objectness score: ***t_o***

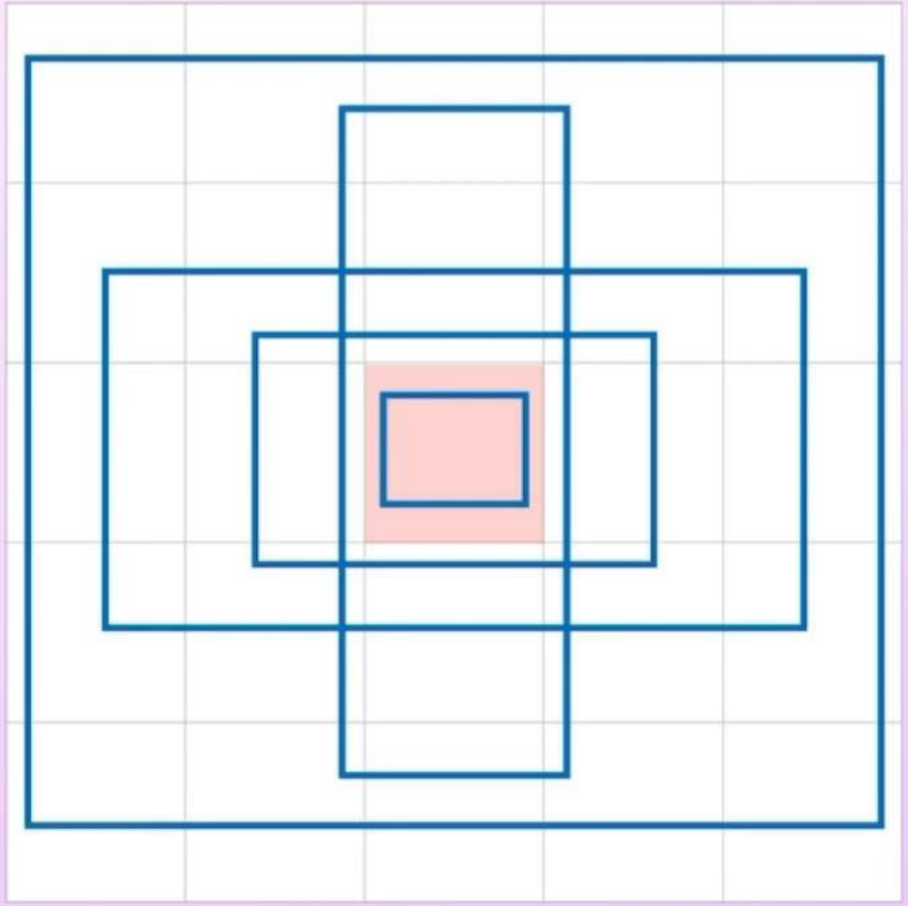
$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

- Class probabilities for each box



Modified Box Predictions

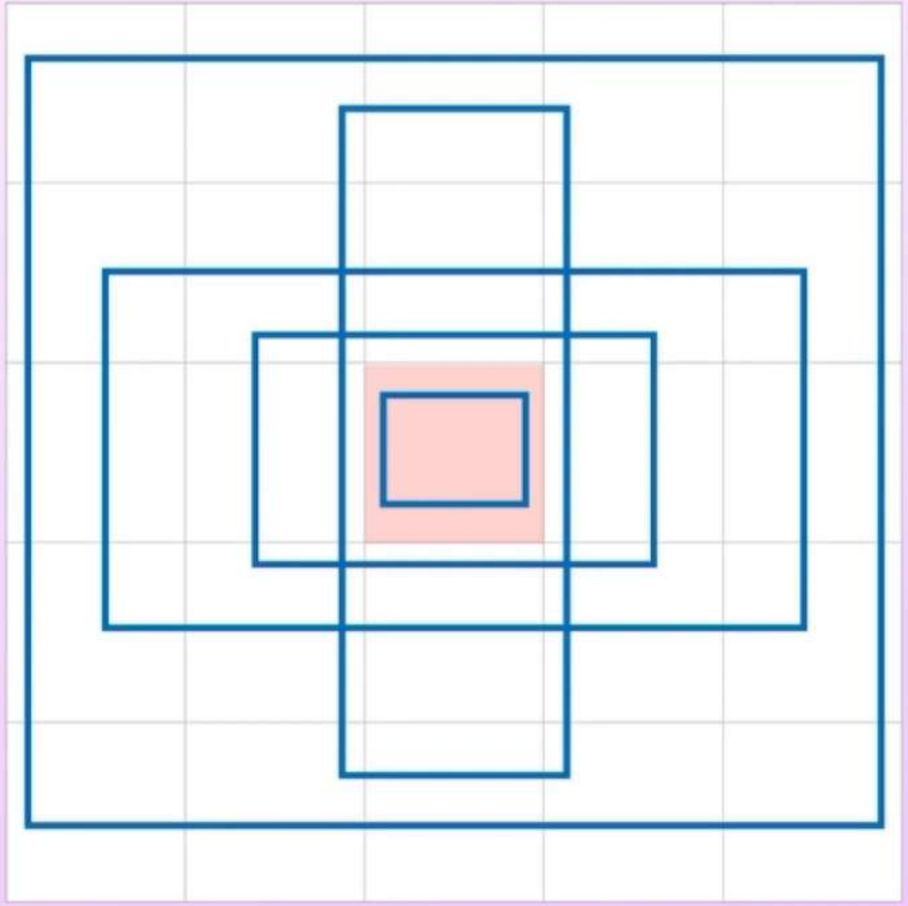
- For every Grid cell - 5 boxes predicted
- For each box:
  - Box Predictions - (***t_x, t_y, t_w, t_h, t_o)***
  - Class Probabilities - ***(P_1,P_2,....P_20)***
- Parameters per Grid cell:
  - 5 x (5 + 20) = **125** (For Pascal VOC)
- Total Grid cells - 13 x 13
- Total parameters - **13 x 13 x 125**



Sample Anchor boxes

Modified Box Predictions

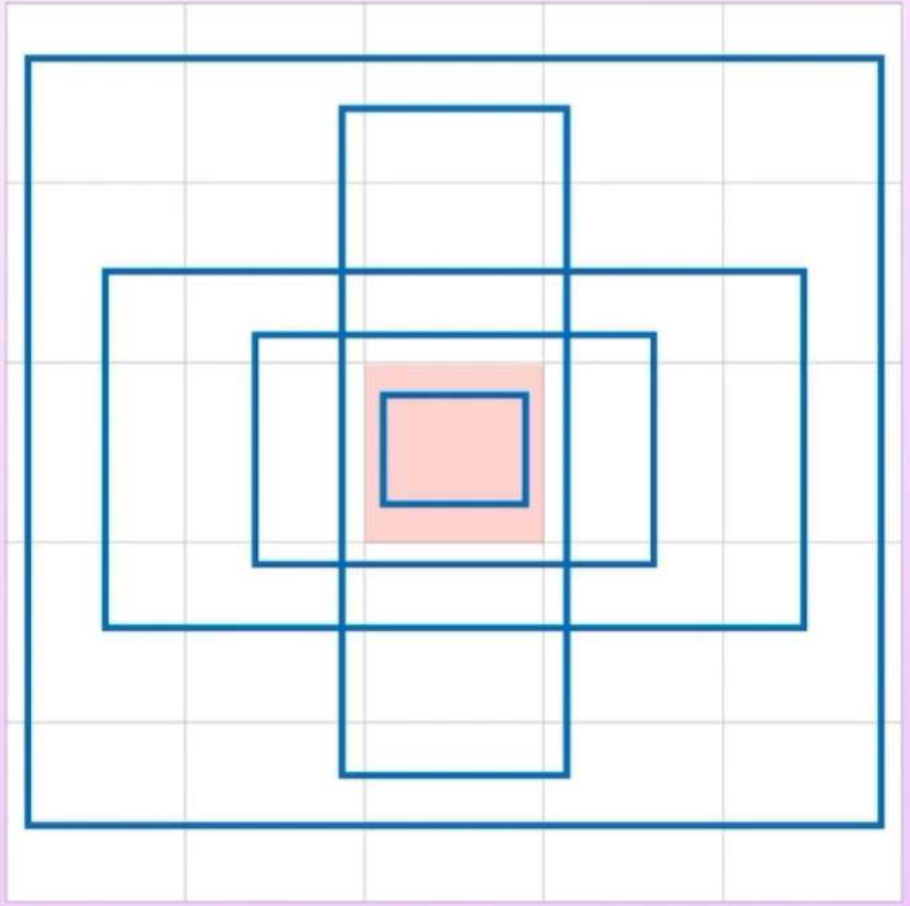
- For every Grid cell - 5 boxes predicted
- For each box:
  - Box Predictions - (***t_x, t_y, t_w, t_h, t_o)***
  - Class Probabilities - ***(P_1,P_2,....P_20)***
- Parameters per Grid cell:
  - 5 x (5 + 20) = **125** (For Pascal VOC)
- Total Grid cells - 13 x 13
- Total parameters - **13 x 13 x 125**



Sample Anchor boxes

Modified Box Predictions

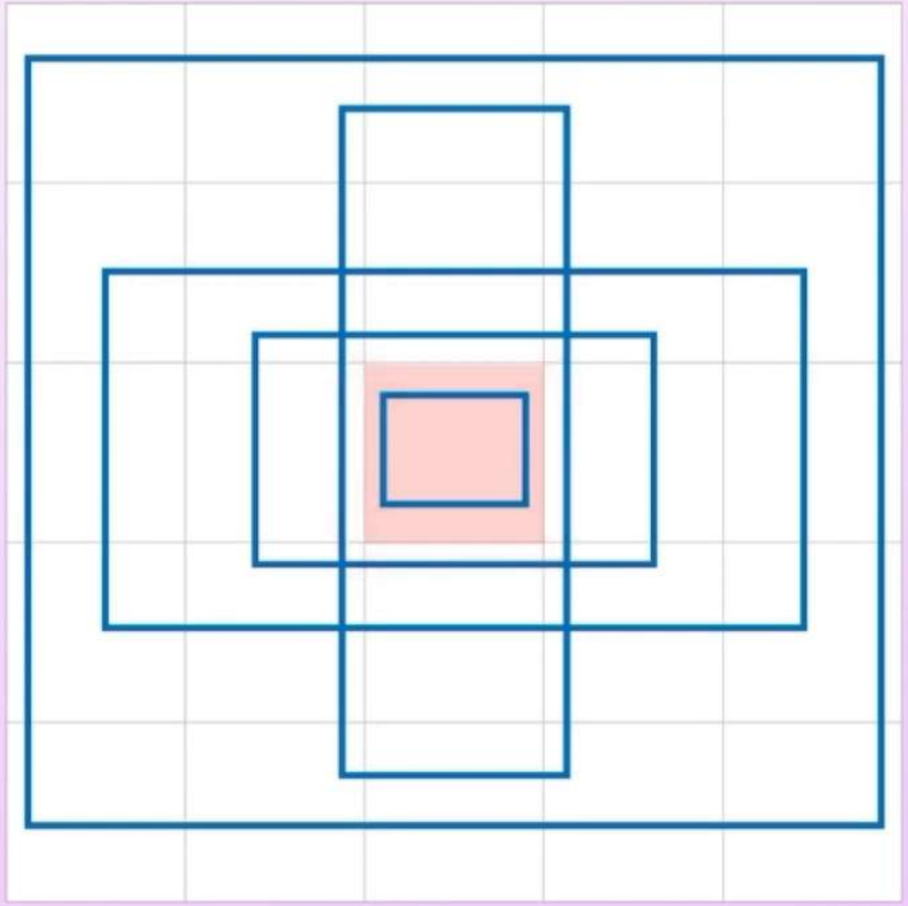
- For every Grid cell - 5 boxes predicted
- For each box:
  - Box Predictions - ***(t_x, t_y, t_w, t_h, t_o)***
  - Class Probabilities - ***(P_1,P_2,....P_20)***
- Parameters per Grid cell:
  - 5 x (5 + 20) = **125** (For Pascal VOC)
- Total Grid cells - 13 x 13
- Total parameters - **13 x 13 x 125**



Sample Anchor boxes

Modified Box Predictions

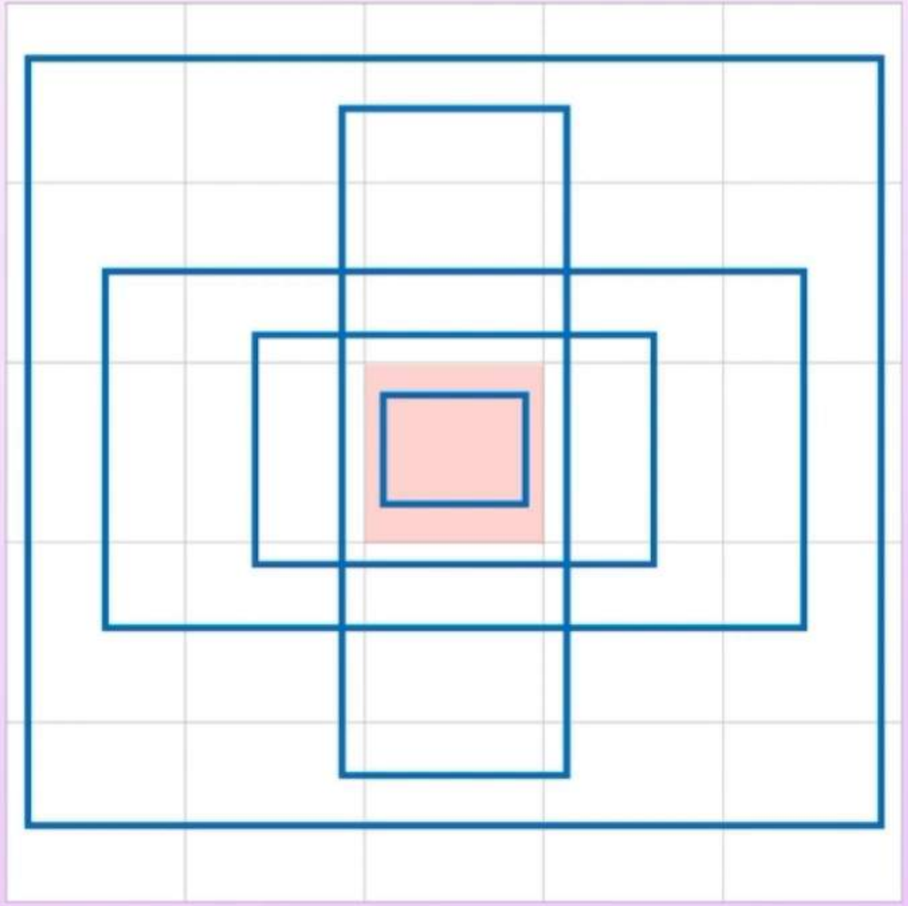
- For every Grid cell - 5 boxes predicted
- For each box:
  - Box Predictions - ***(t_x, t_y, t_w, t_h, t_o)***
  - Class Probabilities - ***(P_1,P_2,....P_20)***
- Parameters per Grid cell:
  - 5 x (5 + 20) = **125** (For Pascal VOC)
- Total Grid cells - 13 x 13
- Total parameters - **13 x 13 x 125**



Sample Anchor boxes

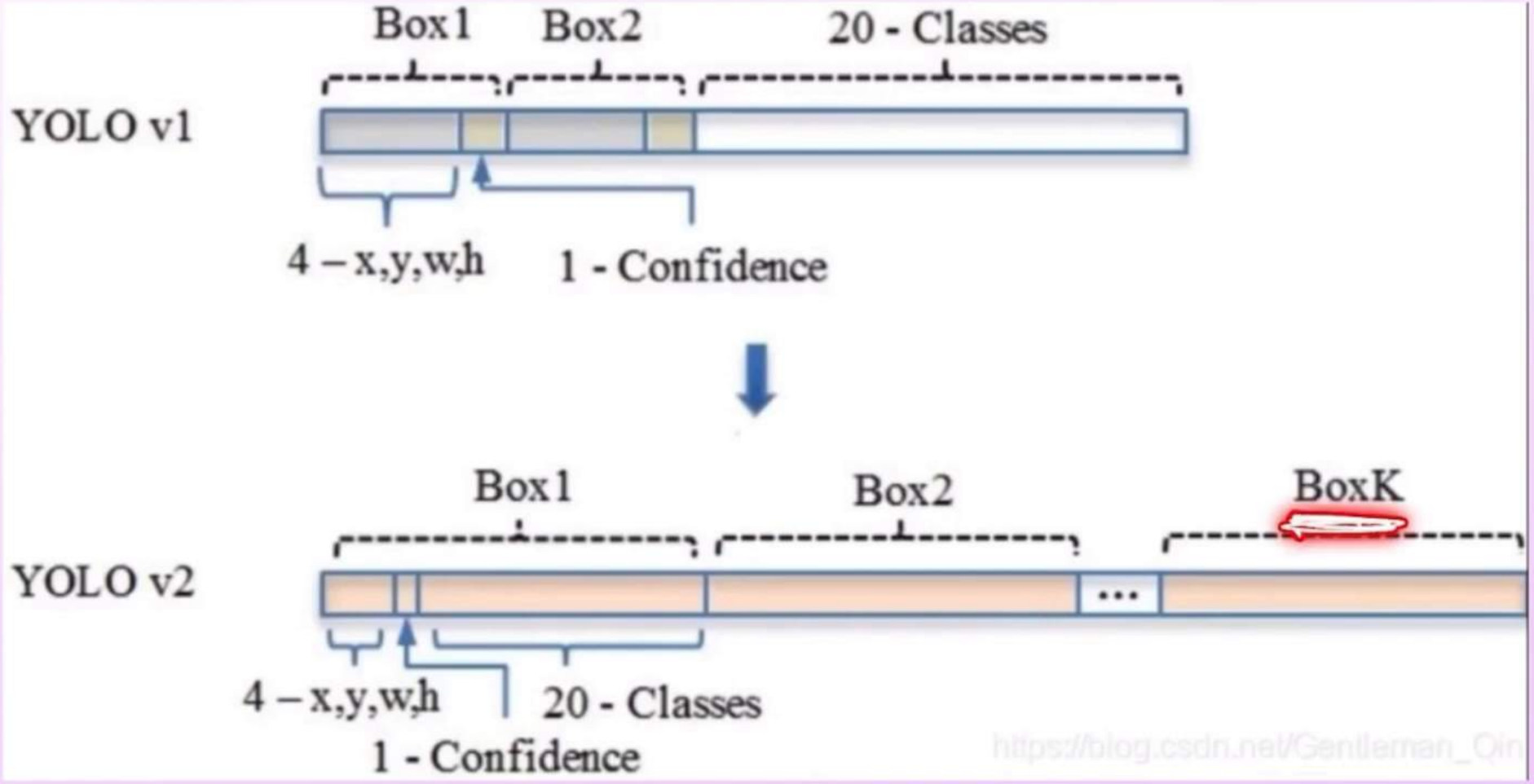
Modified Box Predictions

- For every Grid cell - 5 boxes predicted
- For each box:
  - Box Predictions - ***(t_x, t_y, t_w, t_h, t_o)***
  - Class Probabilities - ***(P_1,P_2,....P_20)***
- Parameters per Grid cell:
  - 5 x (5 + 20) = **125** (For Pascal VOC)
- Total Grid cells - 13 x 13
- Total parameters - **13 x 13 x 125**

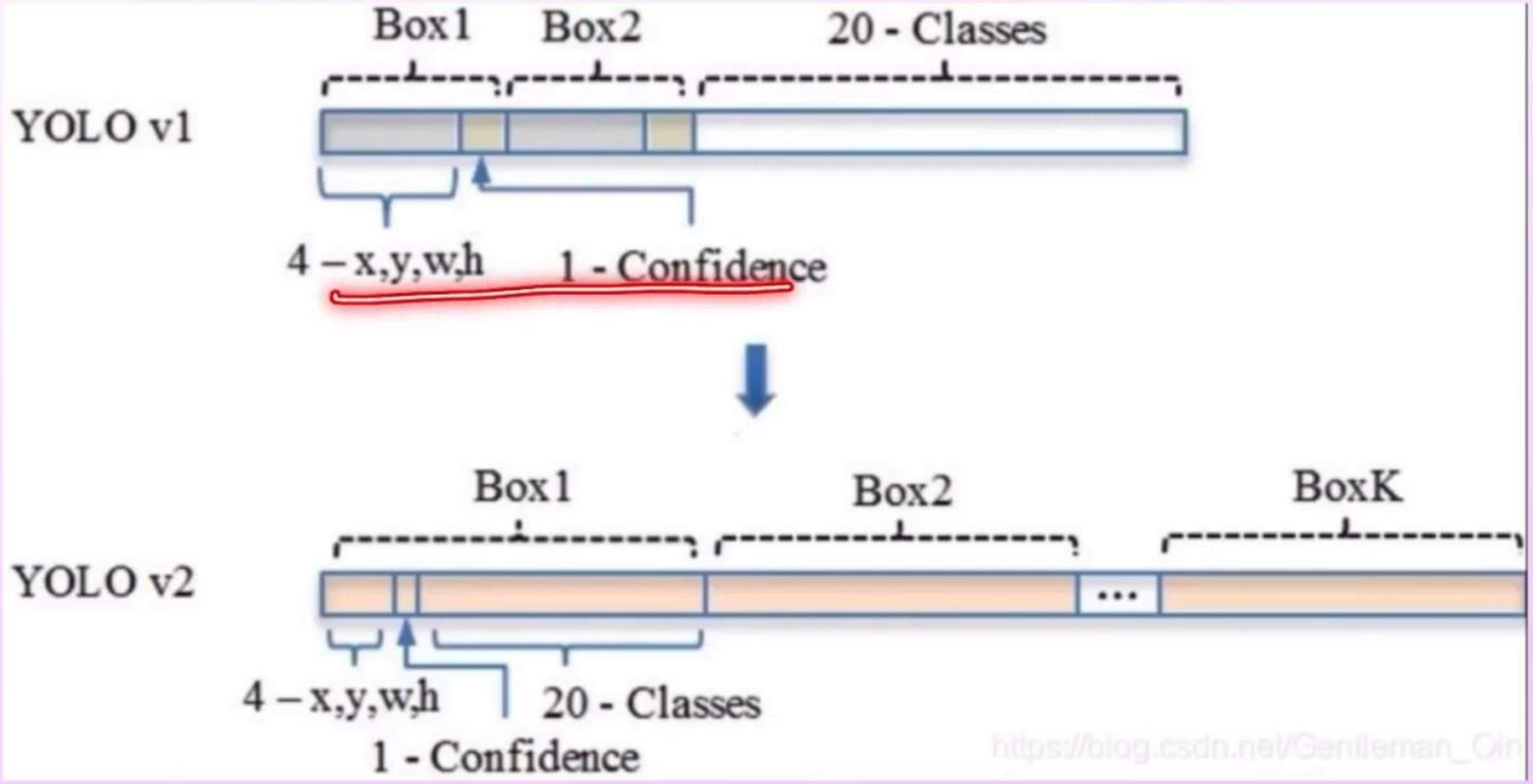


Sample Anchor boxes

Output Predictions - Grid Cell

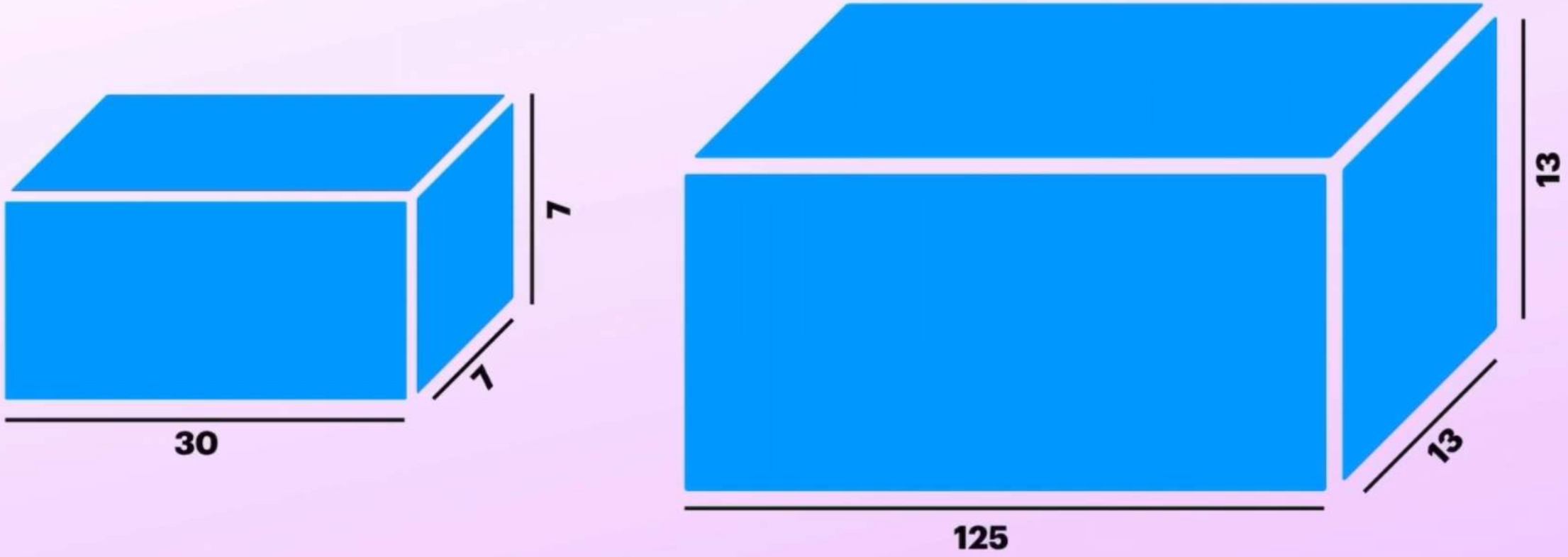


Output Predictions - Grid Cell



Output Predictions

5% mAP rise

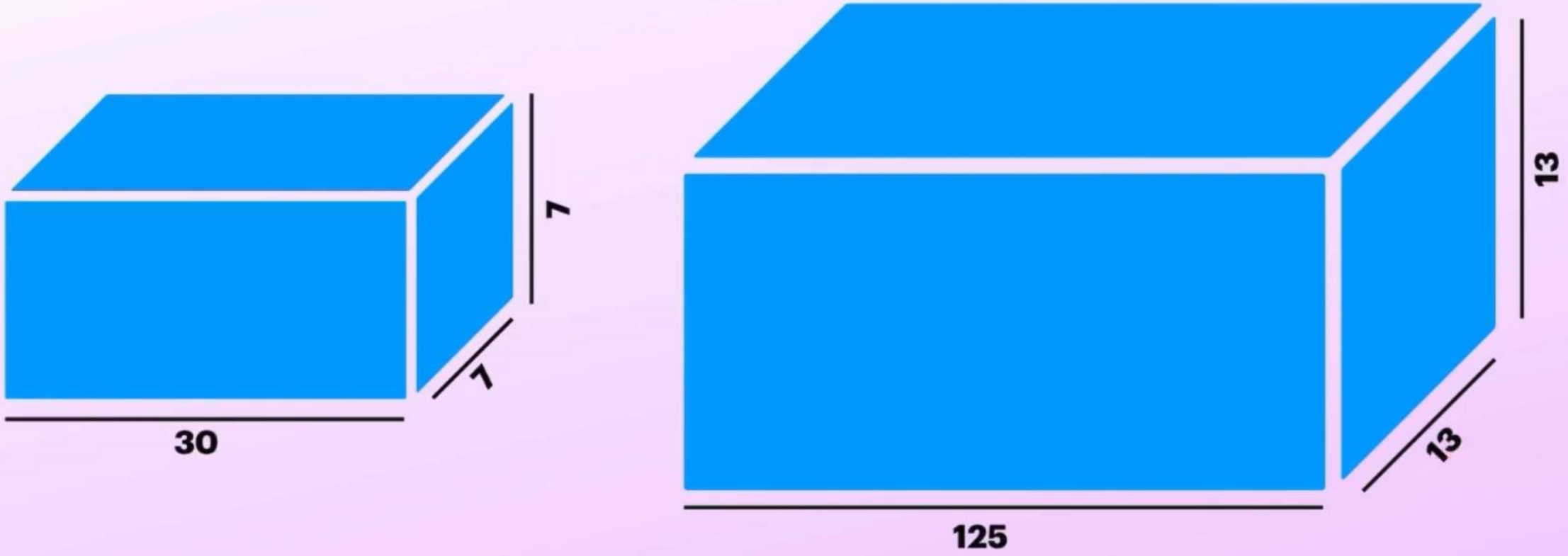


YOLO-V1

YOLO-V2

Output Predictions

5% mAP rise

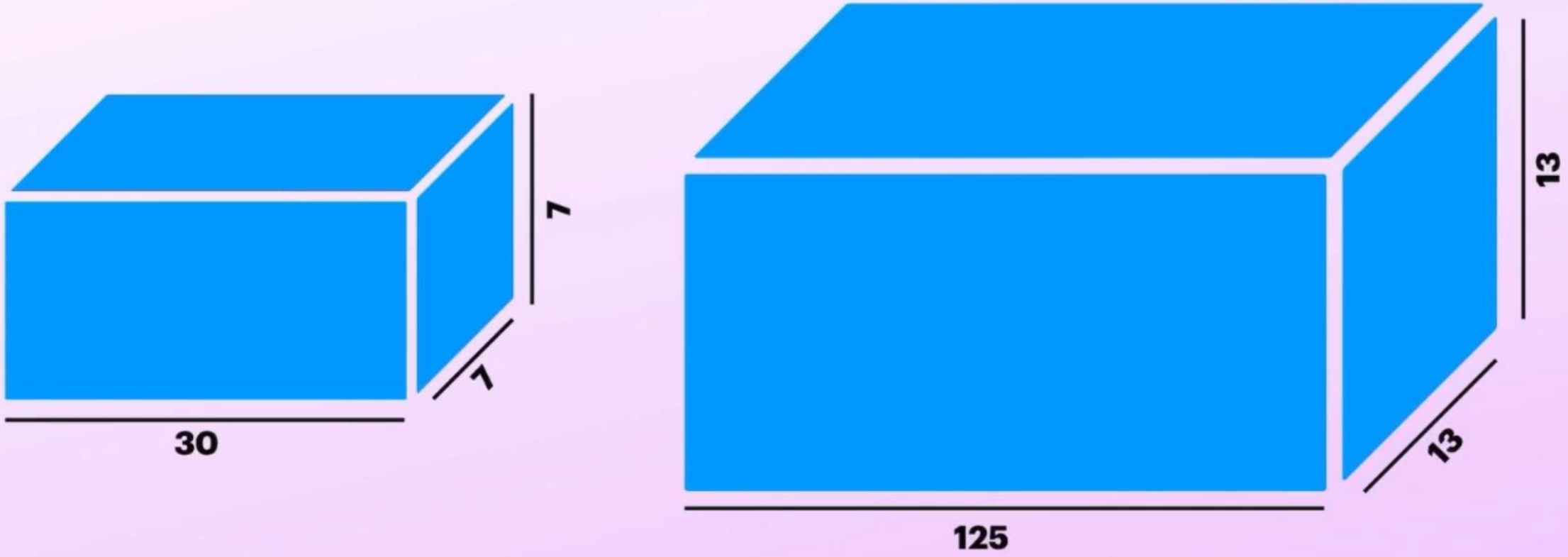


YOLO-V1

YOLO-V2

Output Predictions

5% mAP rise

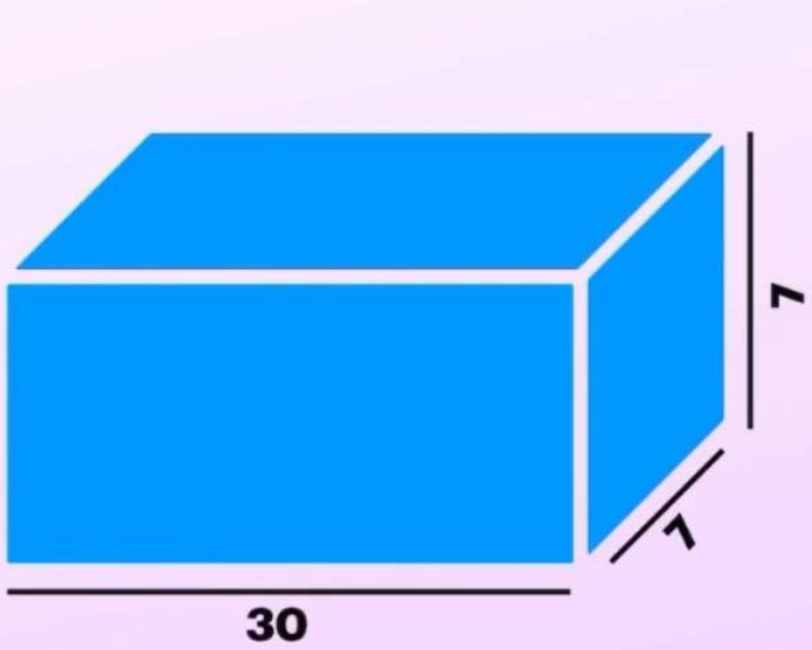


YOLO-V1

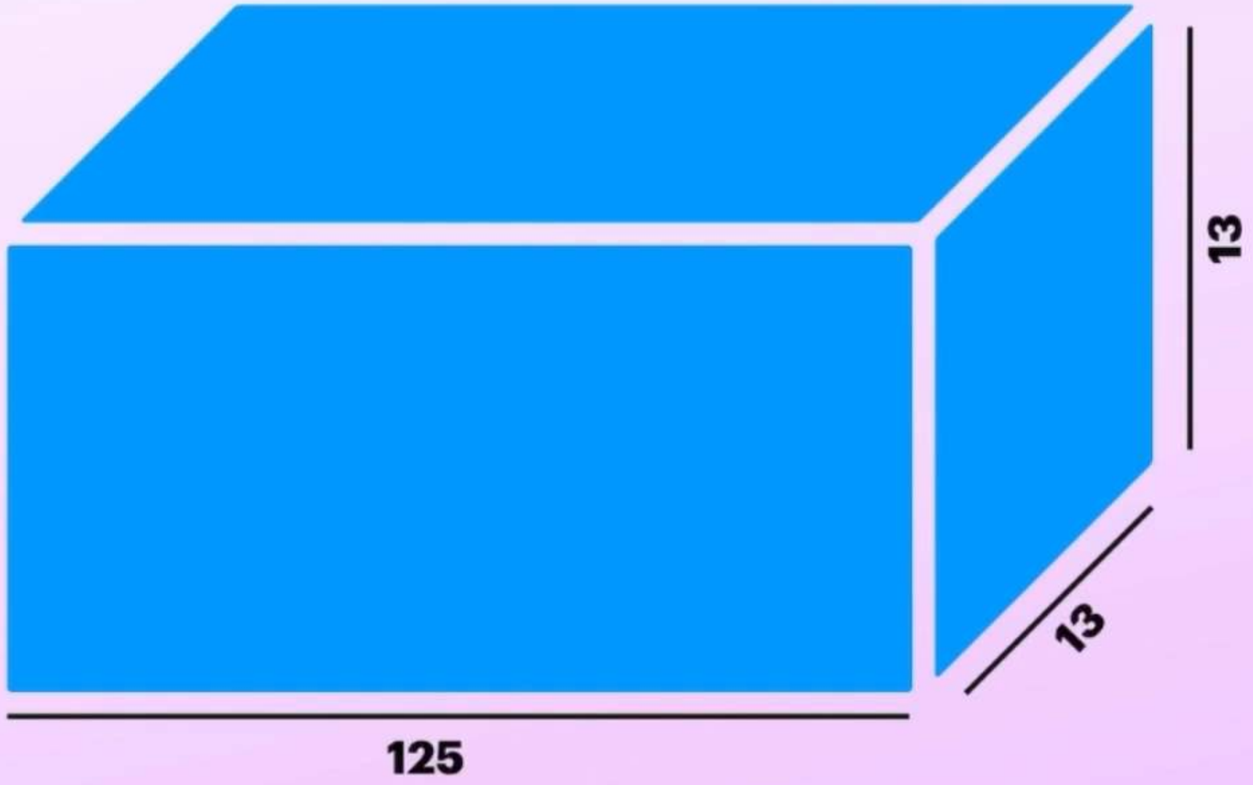
YOLO-V2

Output Predictions

5% mAP rise



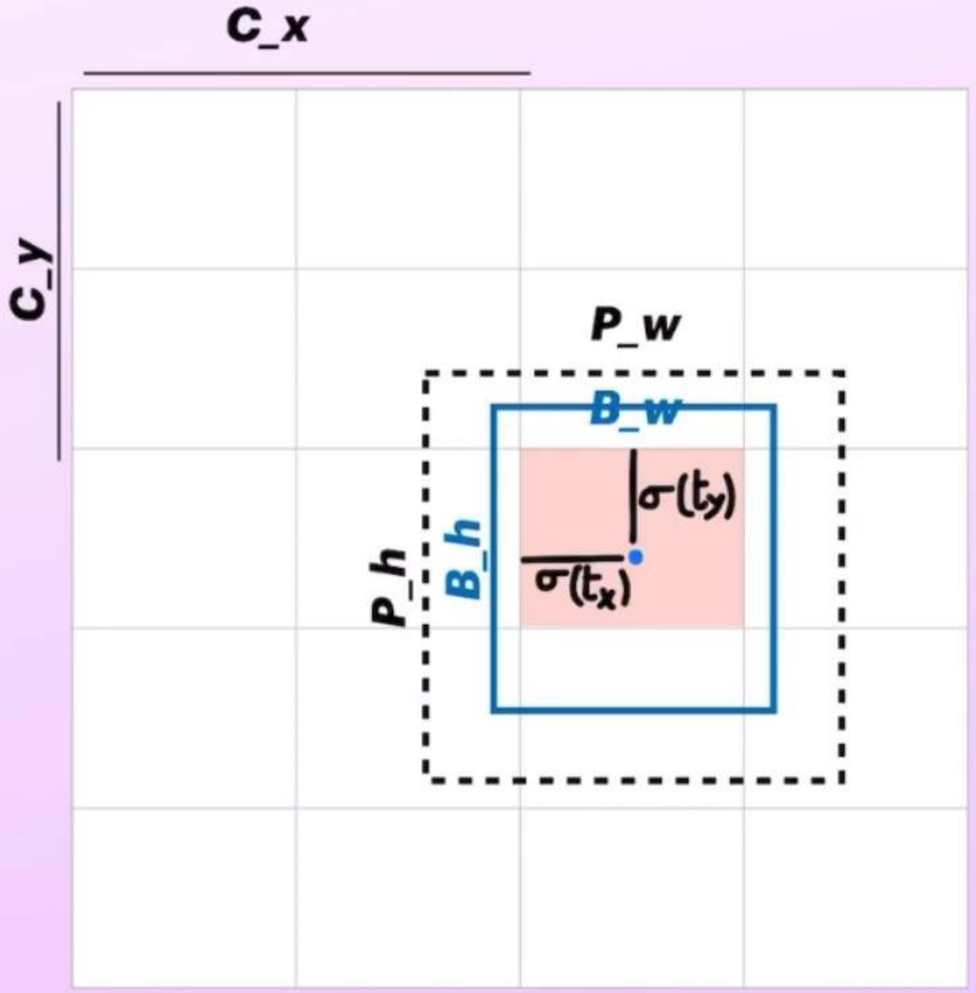
YOLO-V1



YOLO-V2

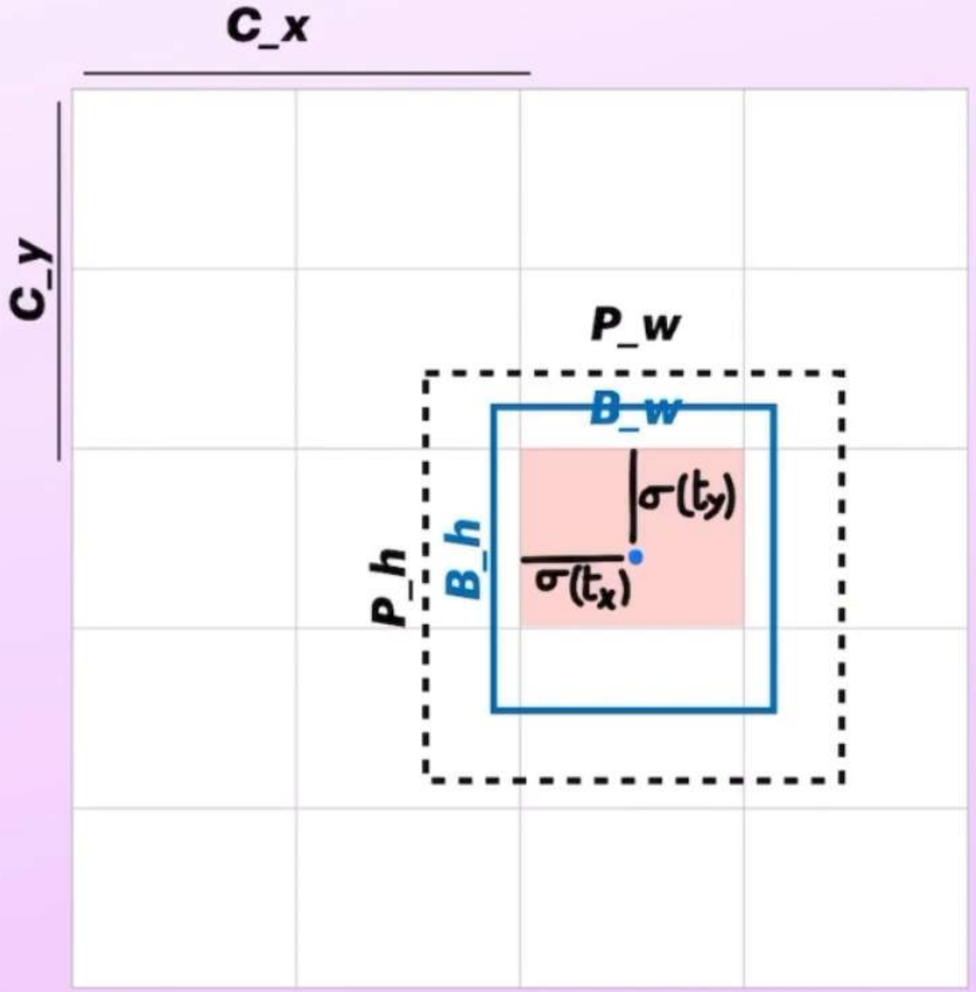
Target Calculations

- Anchor box - Highest IOU with Groundtruth box
- Grid cell - where the object center falls
- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box



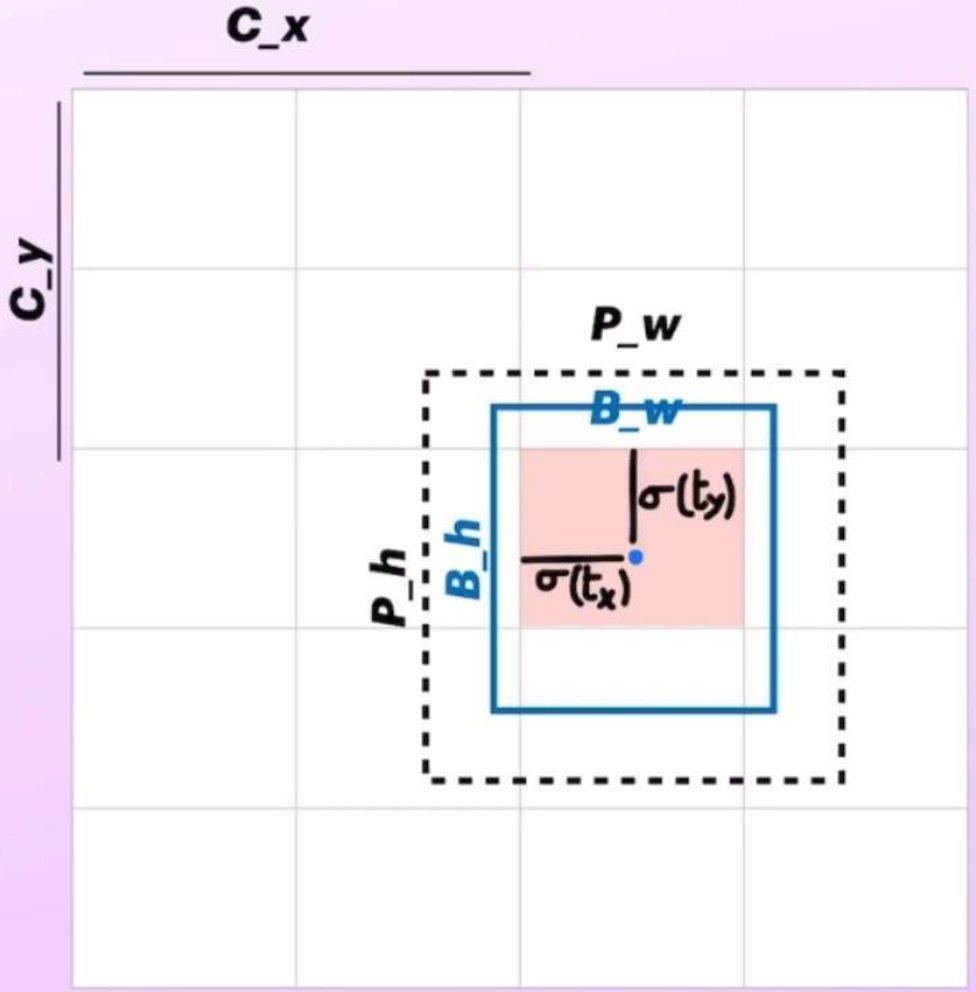
Target Calculations

- Anchor box - Highest IOU with Groundtruth box
- Grid cell - where the object center falls
- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box



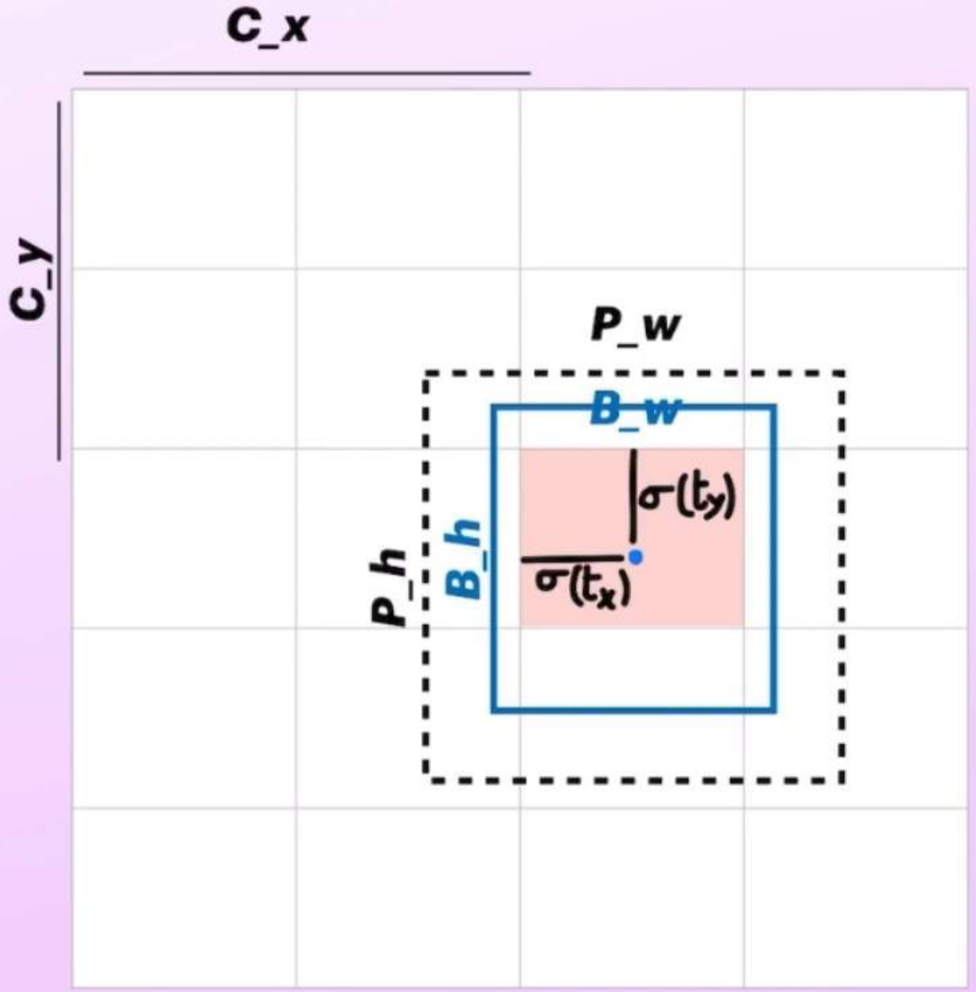
Target Calculations

- Anchor box - Highest IOU with Groundtruth box
- Grid cell - where the object center falls
- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box



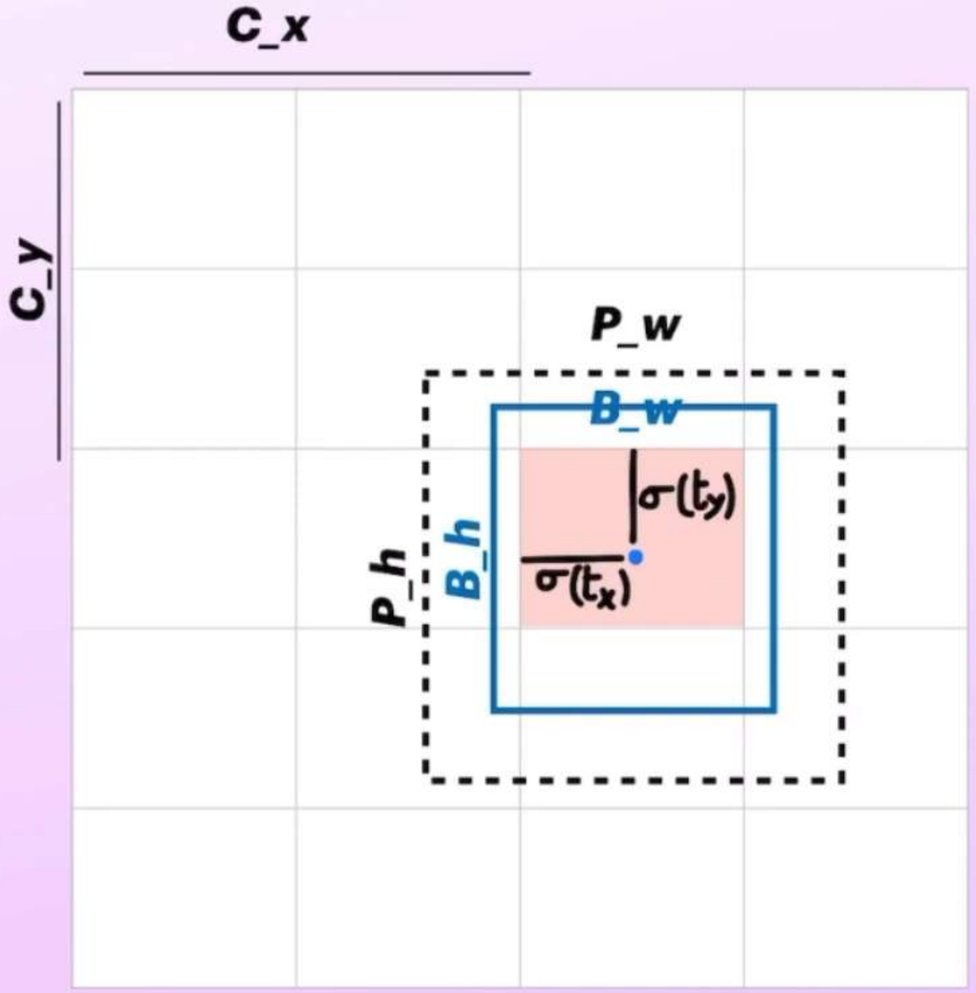
Target Calculations

- Anchor box - Highest IOU with Groundtruth box
- Grid cell - where the object center falls
- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box



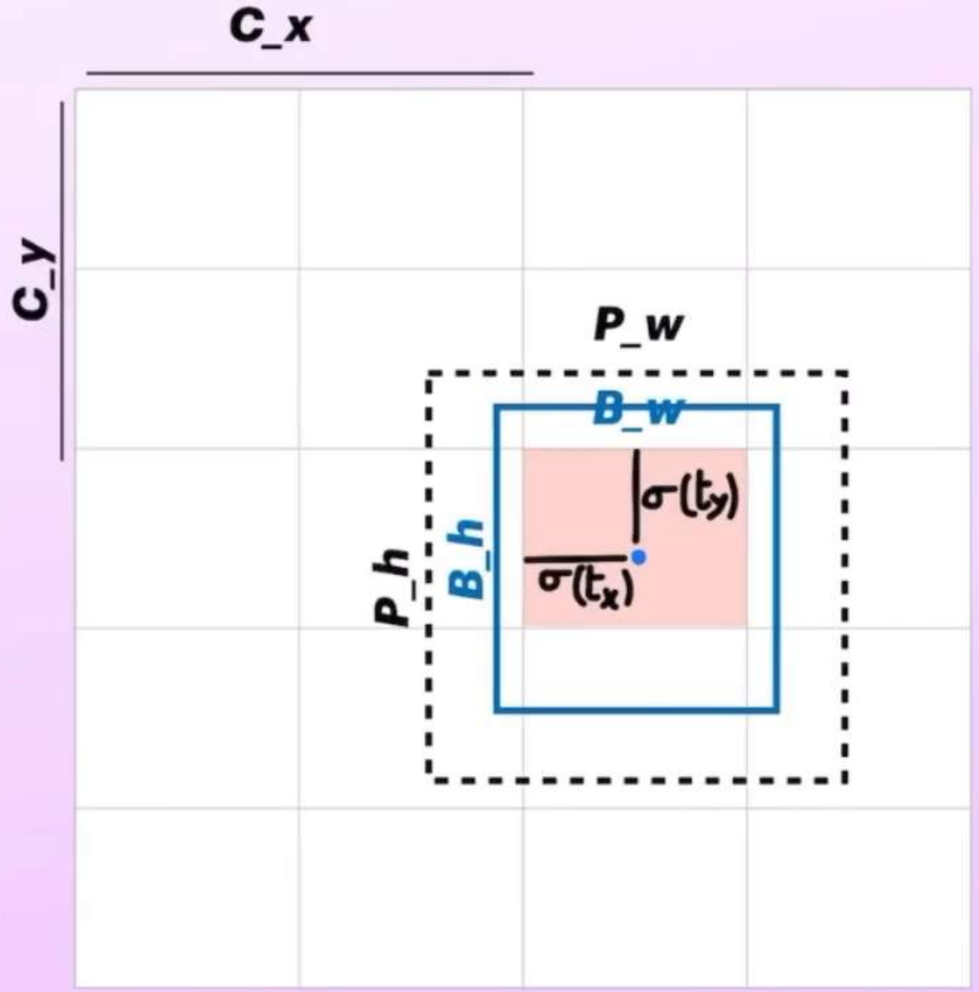
Target Calculations

- Anchor box - Highest IOU with Groundtruth box
- Grid cell - where the object center falls
- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box



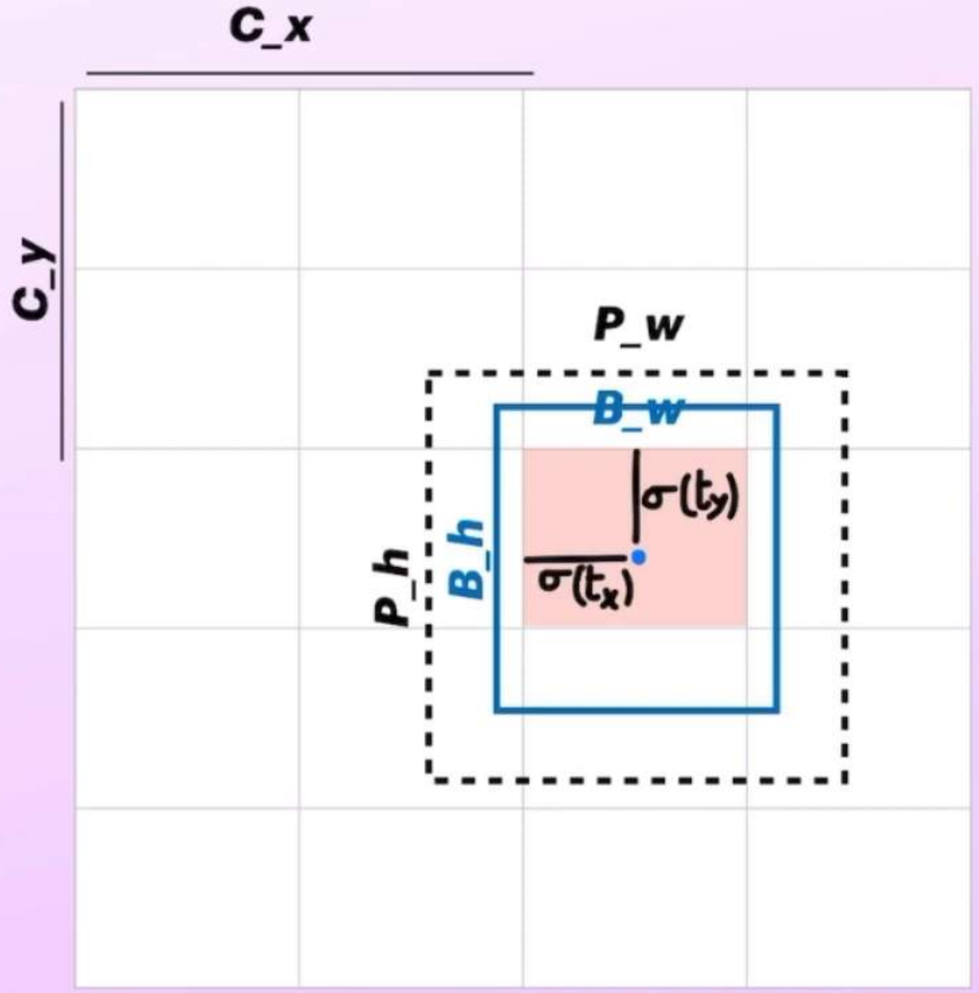
Target Calculations

- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box
  - Target for objectness score - 1 for cells with objects and 0 for cells without objects
  - Target for class probabilities - one-hot encoding vector



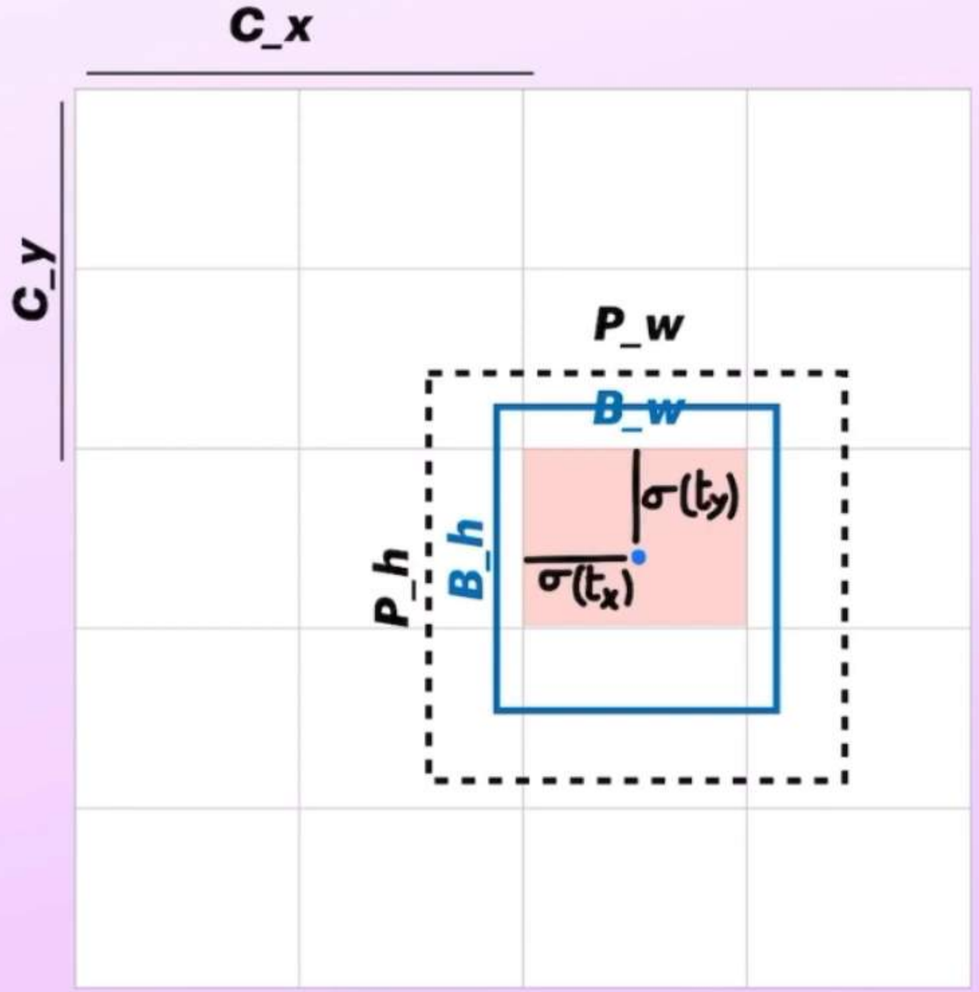
Target Calculations

- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box
  - Target for objectness score - 1 for cells with objects and 0 for cells without objects
  - Target for class probabilities - one-hot encoding vector



Target Calculations

- Targets for each grid cell and anchor box:
  - Targets for x,y - relative to grid cell
  - Targets for w,h - relative to anchor box
  - Target for objectness score - 1 for cells with objects and 0 for cells without objects
  - Target for class probabilities - one-hot encoding vector



Loss Function

$$loss_t = \sum_{i=0}^{W}\sum_{j=0}^{H}\sum_{k=0}^{A} 1_{Max\ IOU < Thresh}\ \lambda_{noobj} * \left(-b_{ijk}^{o}\right)^2$$
$$+ 1_{\ t<12800}\ \lambda_{prior} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(prior_k^r - b_{ijk}^r\right)^2$$
$$+ 1_k^{truth}\Big(\ \lambda_{coord} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(truth^r - b_{ijk}^r\right)^2$$
$$+ \lambda_{obj} * \left(IOU_{truth}^k - b_{ijk}^o\right)^2$$
$$+ \lambda_{class} * \Big(\sum_{c=1}^{C} \left(truth^c - b_{ijk}^c\right)^2\Big)\Big)$$

Loss Function

$$loss_t = \sum_{i=0}^{W}\sum_{j=0}^{H}\sum_{k=0}^{A} 1_{Max\ IOU < Thresh}\ \lambda_{noobj} * \left(-b^o_{ijk}\right)^2$$

$$+ 1_{\ t<12800}\ \lambda_{prior} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(prior^r_k - b^r_{ijk}\right)^2$$

$$+ 1^{truth}_k \Big(\lambda_{coord} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(truth^r - b^r_{ijk}\right)^2$$

$$+ \lambda_{obj} * \left(IOU^k_{truth} - b^o_{ijk}\right)^2$$

$$+ \lambda_{class} * \Big(\sum_{c=1}^{C} \left(truth^c - b^c_{ijk}\right)^2\Big)\Big)$$

Loss Function

- W,H are same as S x S - 13x13 grid cells
- A - Number of anchors
- Loss - for every grid cell and every anchor in that grid cell

$$loss_t = \sum_{i=0}^{S} \sum_{j=0}^{S} \sum_{k=0}^{A}$$

Loss Function

- W,H are same as S x S - 13x13 grid cells
- A - Number of anchors
- Loss - for every grid cell and every anchor in that grid cell

$$loss_t = \sum_{i=0}^{W} \sum_{j=0}^{H} \sum_{k=0}^{A}$$

Loss Function

- W,H are same as S x S - 13x13 grid cells
- A - Number of anchors
- Loss - for every grid cell and every anchor in that grid cell

$$loss_t = \sum_{i=0}^{W} \sum_{j=0}^{H} \sum_{k=0}^{A}$$

Loss For cells with No objects

- Loss for only objectness score

$$1_{Max\ IOU < Thresh}\ \lambda_{noobj}\ *\ \left(-b^{o}_{ijk}\right)^2$$

Loss For cells with No objects

- Loss for only objectness score

$$1_{Max\ IOU < Thresh}\ \lambda_{noobj} * \left(-b^{o}_{ijk}\right)^2$$

Loss For cells with No objects

- Loss for only objectness score

$$1_{Max\ IOU < Thresh}\ \lambda_{noobj} * \left(-b^{o}_{ijk}\right)^2$$

Loss between anchors and predictions

- Calculate loss between alignment of Anchors and Predictions
- Squared error loss between ***x,y,w,h***
- Calculated only for first 12800 iterations

$$1_{\,t<12800}\,\lambda_{prior} * \sum_{r\,\epsilon\,(x,y,w,h)} \left(prior_k^r - b_{ijk}^r\right)^2$$

Loss between anchors and predictions

- Calculate loss between alignment of Anchors and Predictions
- Squared error loss between ***x,y,w,h***
- Calculated only for first 12800 iterations

$$1_{\,t<12800}\,\lambda_{prior} * \sum_{r\,\epsilon\,(x,y,w,h)} \left(prior_k^r - b_{ijk}^r\right)^2$$

Loss between anchors and predictions

- Calculate loss between alignment of Anchors and Predictions
- Squared error loss between ***x,y,w,h***
- Calculated only for first 12800 iterations

$$1_{\,t<12800}\,\lambda_{prior} * \sum_{r\,\epsilon\,(x,y,w,h)} \left(prior_k^r - b_{ijk}^r\right)^2$$

Bounding box prediction loss

- Same as that of Yolo-V1
- Squared error loss
- Truth means ground truth targets

$$1_k^{truth}\left(\ \lambda_{coord}\ *\sum_{r\ \epsilon\ (x,y,w,h)}\left(truth^r\ -\ b_{ijk}^r\right)^2\right.$$

Bounding box prediction loss

- Same as that of Yolo-V1
- Squared error loss
- Truth means ground truth targets

$$1_k^{truth}\left(\ \lambda_{coord}\ *\ \sum_{r\ \epsilon\ (x,y,w,h)} (truth^r\ -\ b_{ijk}^r)^2\right.$$

Objectness and Class Losses

- Same as that of Yolo-V1
- Squared error losses
- Truth means ground truth targets

$$\lambda_{obj} * \left(IOU^{k}_{truth} - b^{o}_{ijk}\right)^2$$

$$\lambda_{class} * \left(\sum_{c=1}^{C} \left(truth^{c} - b^{c}_{ijk}\right)^2\right)$$

Objectness and Class Losses

- Same as that of Yolo-V1
- Squared error losses
- Truth means ground truth targets

$$\lambda_{obj} * \left(IOU^{k}_{truth} - b^{o}_{ijk}\right)^2$$

$$\lambda_{class} * \left(\sum_{c=1}^{C} \left(truth^{c} - b^{c}_{ijk}\right)^2\right)$$

Objectness and Class Losses

- Same as that of Yolo-V1
- Squared error losses
- Truth means ground truth targets

$$\lambda_{obj} * \left(IOU^{k}_{truth} - b^{o}_{ijk}\right)^2$$

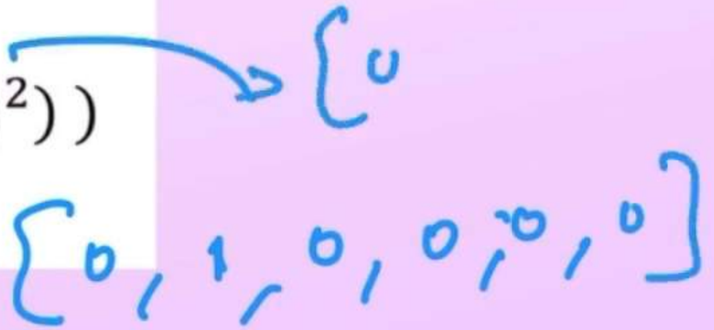
$$\lambda_{class} * \left(\sum_{c=1}^{C} \left(truth^{c} - b^{c}_{ijk}\right)^2\right)$$

Objectness and Class Losses

- Same as that of Yolo-V1
- Squared error losses
- Truth means ground truth targets

$$\lambda_{obj} * \left(IOU^{k}_{truth} - b^{o}_{ijk}\right)^2$$

$$\lambda_{class} * \left(\sum_{c=1}^{C} \left(truth^{c} - b^{c}_{ijk}\right)^2\right)$$

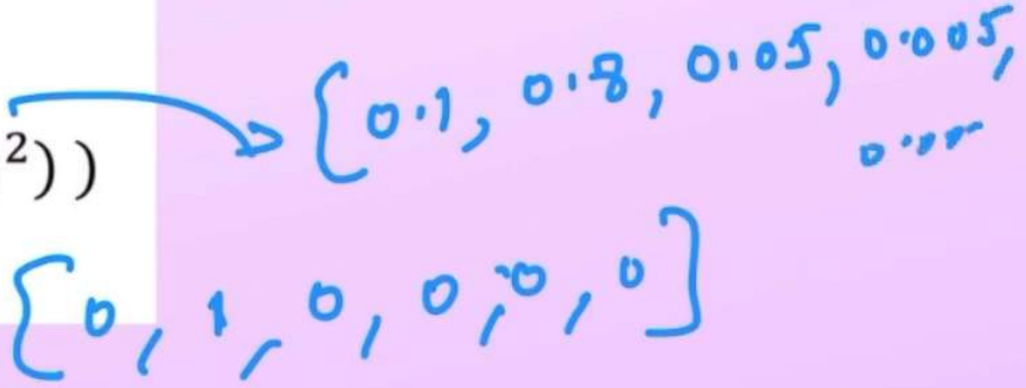


Objectness and Class Losses

- Same as that of Yolo-V1
- Squared error losses
- Truth means ground truth targets

$$\lambda_{obj} * \left(IOU^{k}_{truth} - b^{o}_{ijk}\right)^2$$

$$\lambda_{class} * \left(\sum_{c=1}^{C} (truth^{c} - b^{c}_{ijk})^2\right)$$

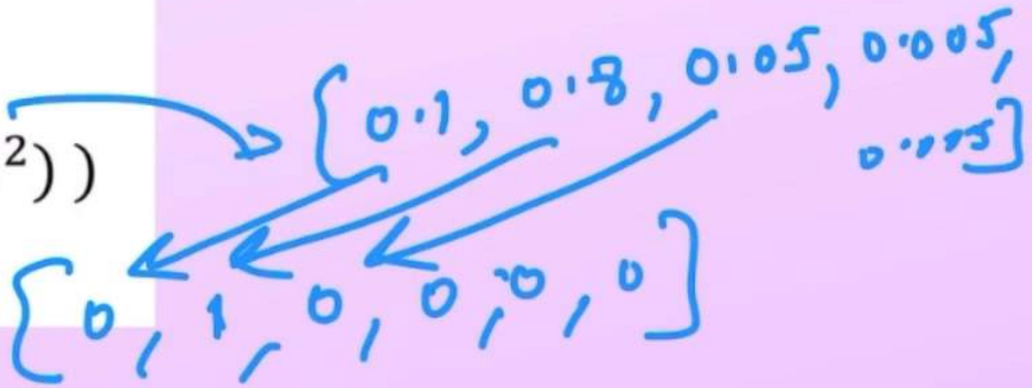


Objectness and Class Losses

- Same as that of Yolo-V1
- Squared error losses
- Truth means ground truth targets

$$\lambda_{obj} * \left(IOU^{k}_{truth} - b^{o}_{ijk}\right)^2$$

$$\lambda_{class} * \left(\sum_{c=1}^{C} \left(truth^{c} - b^{c}_{ijk}\right)^2\right)$$



Total Loss Function

$$loss_t = \sum_{i=0}^{W}\sum_{j=0}^{H}\sum_{k=0}^{A} 1_{Max\ IOU < Thresh}\ \lambda_{noobj} * \left(-b^o_{ijk}\right)^2$$

Loss for cells with No Objects

Loss for Anchor box alignment

$$+ 1_{t<12800}\ \lambda_{prior} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(prior^r_k - b^r_{ijk}\right)^2$$

Loss for cells with objects

$$+ 1^{truth}_k \Big(\lambda_{coord} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(truth^r - b^r_{ijk}\right)^2$$

$$+ \lambda_{obj} * \left(IOU^k_{truth} - b^o_{ijk}\right)^2$$

$$+ \lambda_{class} * \Big(\sum_{c=1}^{C} \left(truth^c - b^c_{ijk}\right)^2\Big)\Big)$$

Total Loss Function

$$loss_t = \sum_{i=0}^{W}\sum_{j=0}^{H}\sum_{k=0}^{A} 1_{Max\ IOU < Thresh}\ \lambda_{noobj} * \left(-b^o_{ijk}\right)^2$$

Loss for cells with No Objects

Loss for Anchor box alignment

$$+ 1_{\ t<12800}\ \lambda_{prior} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(prior^r_k - b^r_{ijk}\right)^2$$

Loss for cells with objects

$$+ 1^{truth}_k \Big(\lambda_{coord} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(truth^r - b^r_{ijk}\right)^2$$

$$+ \lambda_{obj} * \left(IOU^k_{truth} - b^o_{ijk}\right)^2$$

$$+ \lambda_{class} * \Big(\sum_{c=1}^{C} \left(truth^c - b^c_{ijk}\right)^2\Big)\Big)$$

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- YOLO-V2 is a fully Convolutional network
- Network can accept any image resolution
- Resolution should be multiple of 32
- Scales - [288, 320,....608]
- Scale is randomly picked for every 10 batches
- Acts as data augmentation

Multi-Scale Training

- Lower resolutions - low accuracy but good speed
- 288x288 - 69 mAP, 90FPS
- 544x544 - 78.6 mAP, 40 FPS
- We can choose the resolution based on our requirement

Multi-Scale Training

- Lower resolutions - low accuracy but good speed
- 288x288 - 69 mAP, 90FPS
- 544x544 - 78.6 mAP, 40 FPS
- We can choose the resolution based on our requirement

Multi-Scale Training

- Lower resolutions - low accuracy but good speed
- 288x288 - 69 mAP, 90FPS
- 544x544 - 78.6 mAP, 40 FPS
- We can choose the resolution based on our requirement

Accuracy

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Accuracy

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Accuracy

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Accuracy

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

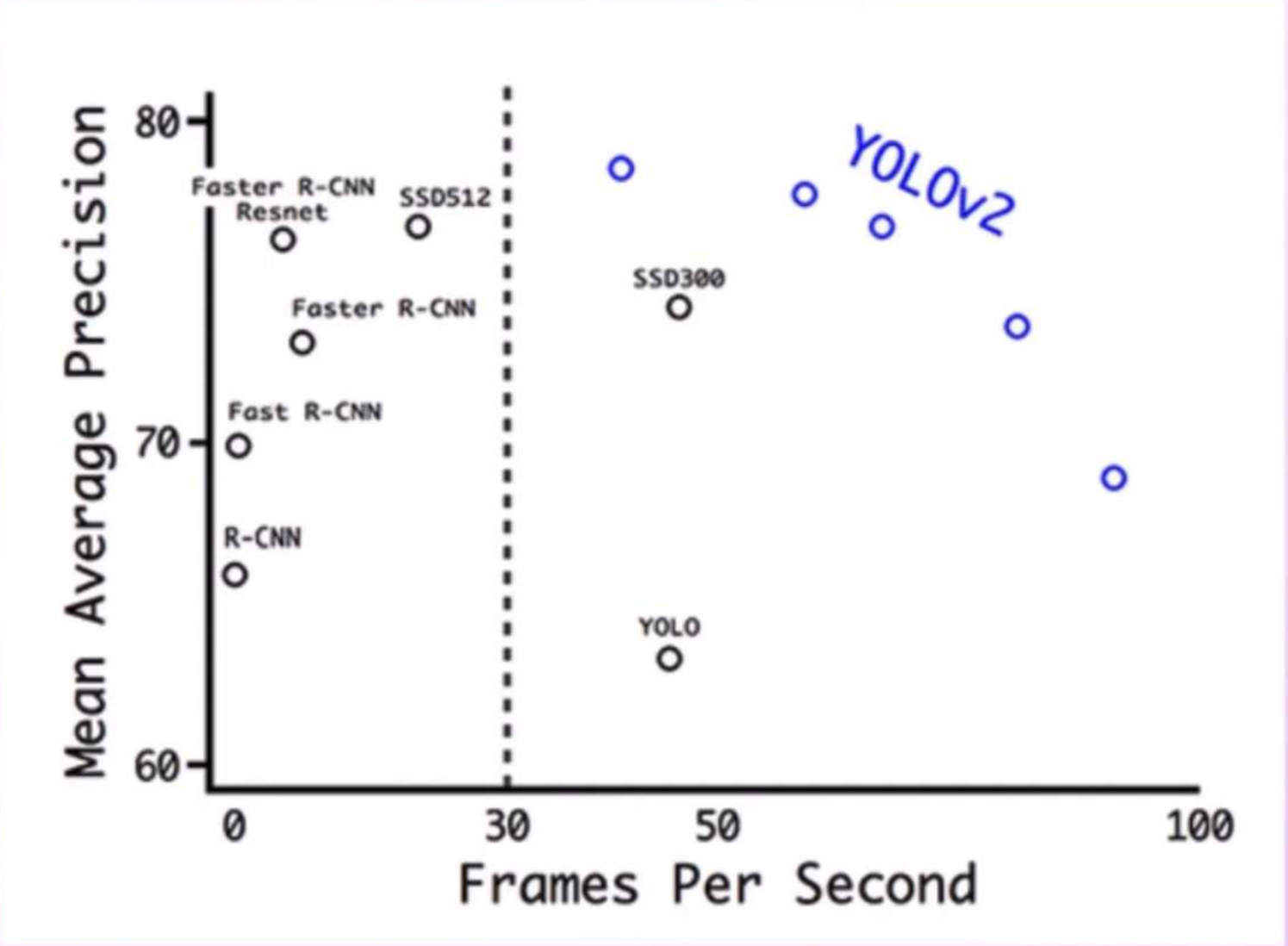
Accuracy

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

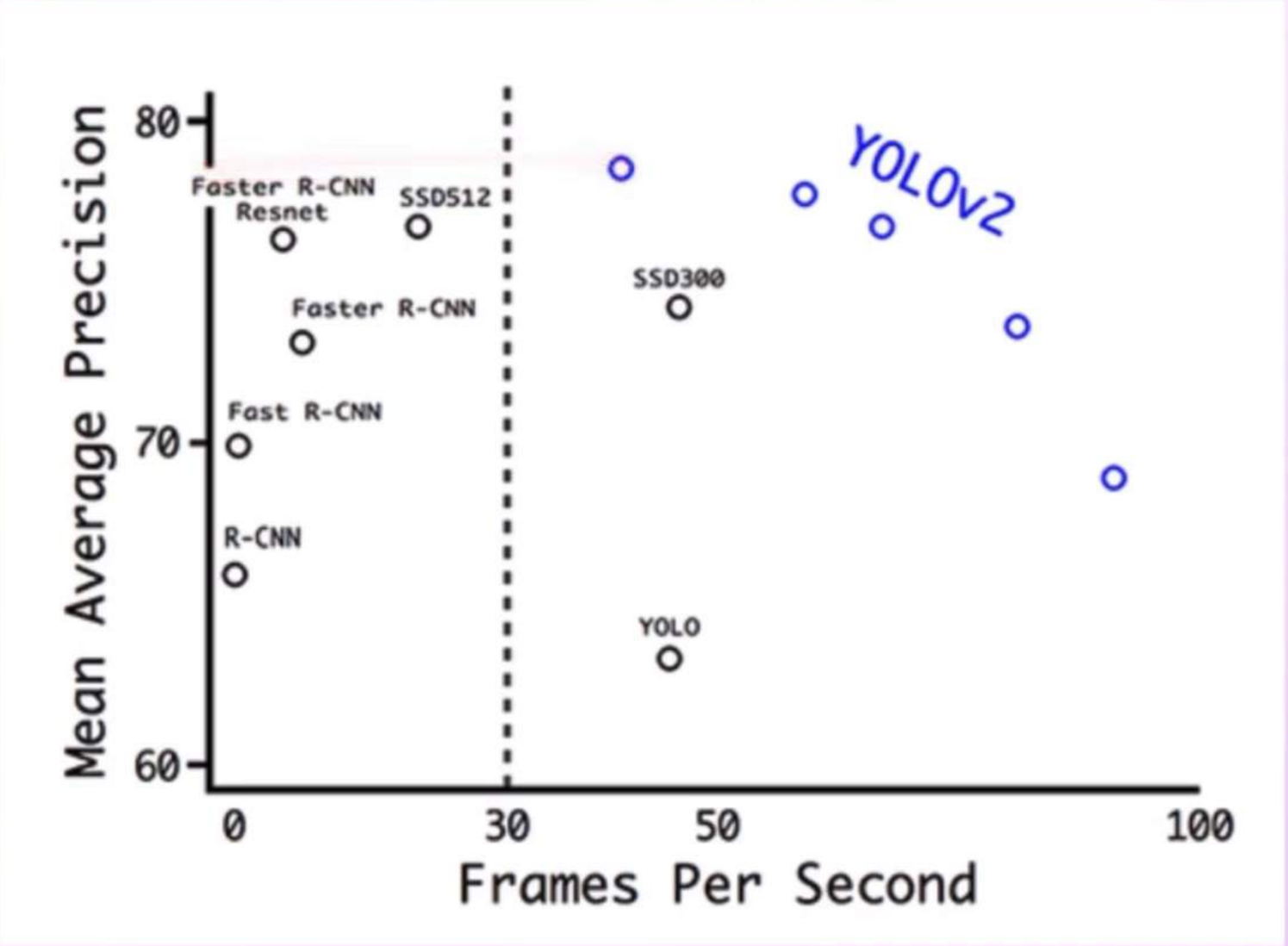
Accuracy

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

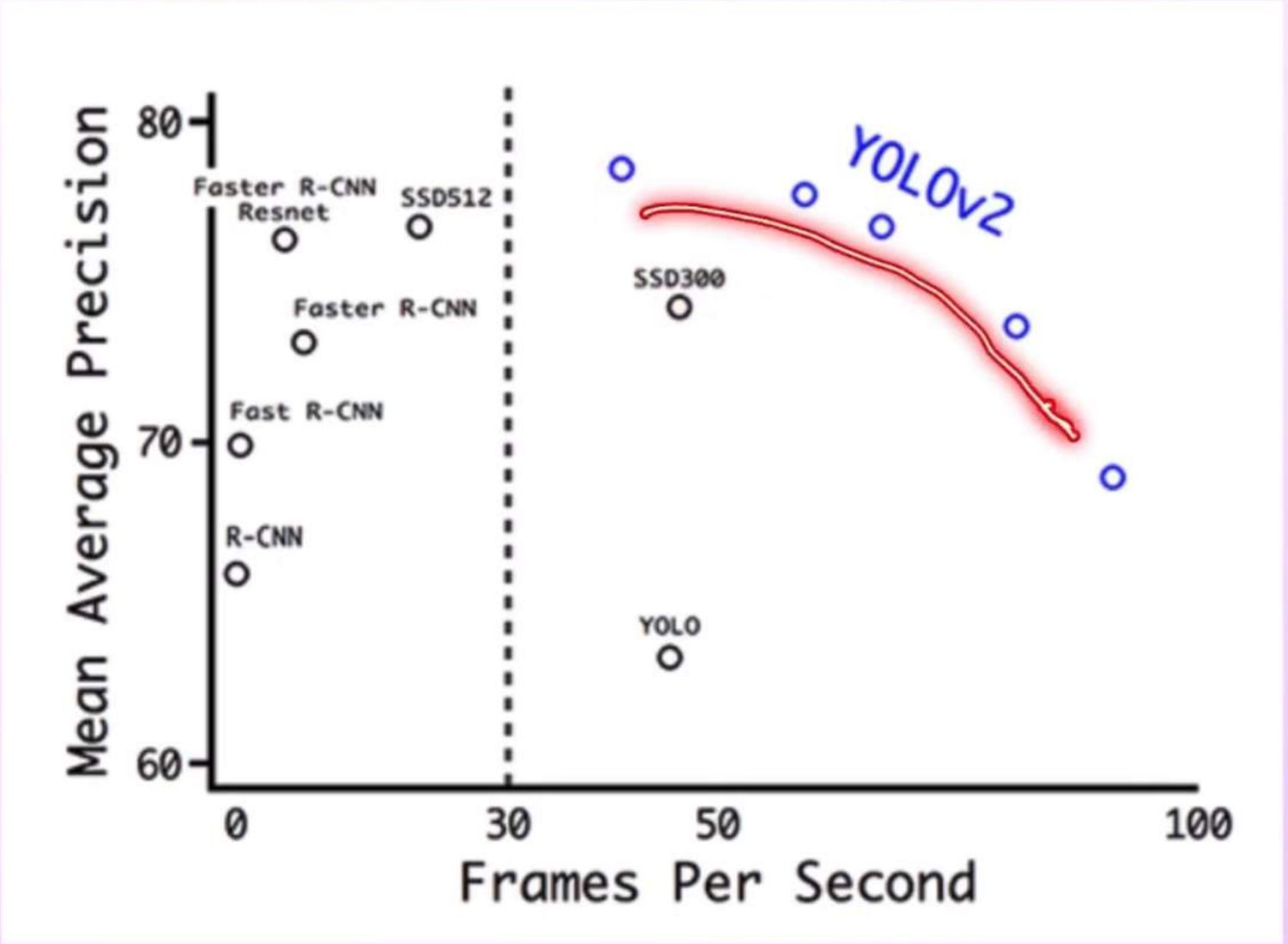
Accuracy - Pascal VOC



Accuracy - Pascal VOC



Accuracy - Pascal VOC



Accuracy - COCO

		0.5:0.95	0.5	0.75
Fast R-CNN [5]	train	19.7	35.9	-
Fast R-CNN[1]	train	20.5	39.9	19.4
Faster R-CNN[15]	trainval	21.9	42.7	-
ION [1]	train	23.6	43.2	23.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5
SSD300 [11]	trainval35k	23.2	41.2	23.4
SSD512 [11]	trainval35k	**26.8**	**46.5**	**27.8**
YOLOv2 [11]	trainval35k	21.6	44.0	19.2

Accuracy - COCO

		0.5:0.95	0.5	0.75
Fast R-CNN [5]	train	19.7	35.9	-
Fast R-CNN[1]	train	20.5	39.9	19.4
Faster R-CNN[15]	trainval	21.9	42.7	-
ION [1]	train	23.6	43.2	23.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5
SSD300 [11]	trainval35k	23.2	41.2	23.4
SSD512 [11]	trainval35k	**26.8**	**46.5**	**27.8**
YOLOv2 [11]	trainval35k	21.6	44.0	19.2

Accuracy - COCO

		0.5:0.95	0.5	0.75
Fast R-CNN [5]	train	19.7	35.9	-
Fast R-CNN[1]	train	20.5	39.9	19.4
Faster R-CNN[15]	trainval	21.9	42.7	-
ION [1]	train	23.6	43.2	23.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5
SSD300 [11]	trainval35k	23.2	41.2	23.4
SSD512 [11]	trainval35k	**26.8**	**46.5**	**27.8**
YOLOv2 [11]	trainval35k	21.6	44.0	19.2

Accuracy - COCO

		0.5:0.95	0.5	0.75
Fast R-CNN [5]	train	19.7	35.9	-
Fast R-CNN[1]	train	20.5	39.9	19.4
Faster R-CNN[15]	trainval	21.9	42.7	-
ION [1]	train	23.6	43.2	23.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5
SSD300 [11]	trainval35k	23.2	41.2	23.4
SSD512 [11]	trainval35k	**26.8**	**46.5**	**27.8**
YOLOv2 [11]	trainval35k	21.6	44.0	19.2

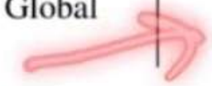
Darknet-19

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			

	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

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			



	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

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			

	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

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			

	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

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			

	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

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			

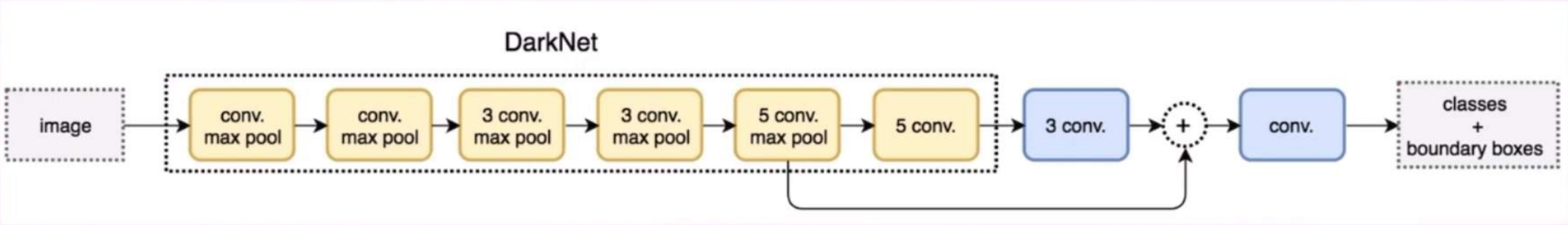
	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

Darknet-19

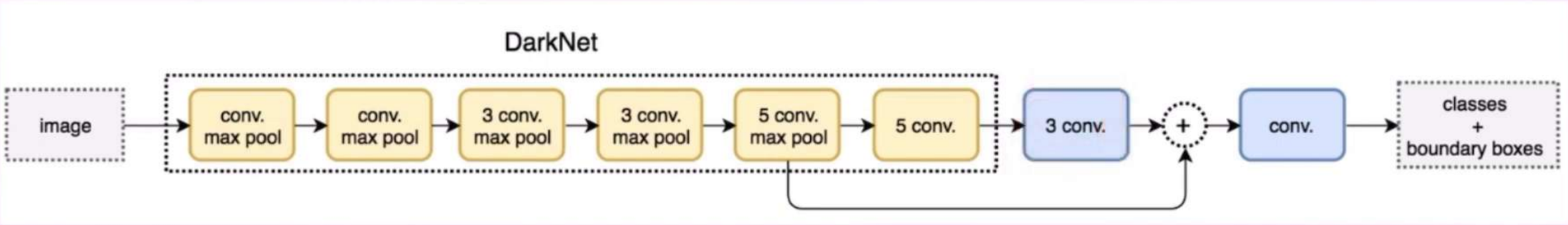
Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	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			

	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

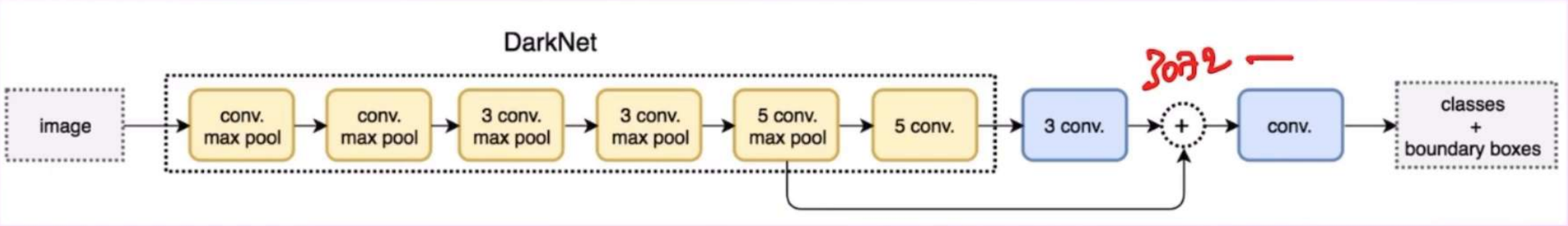
Darknet-19 for YOLO-V2



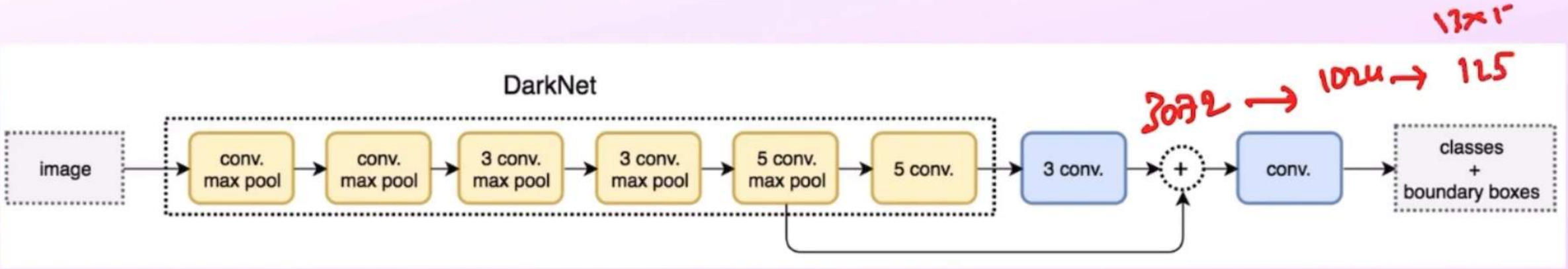
Darknet-19 for YOLO-V2



Darknet-19 for YOLO-V2



Darknet-19 for YOLO-V2



YOLO-9000


Thank You!
boat
mushe
malamute
ski
athlete
athlete
ball
contestant
unicycle
unicycle

YOLO-9000