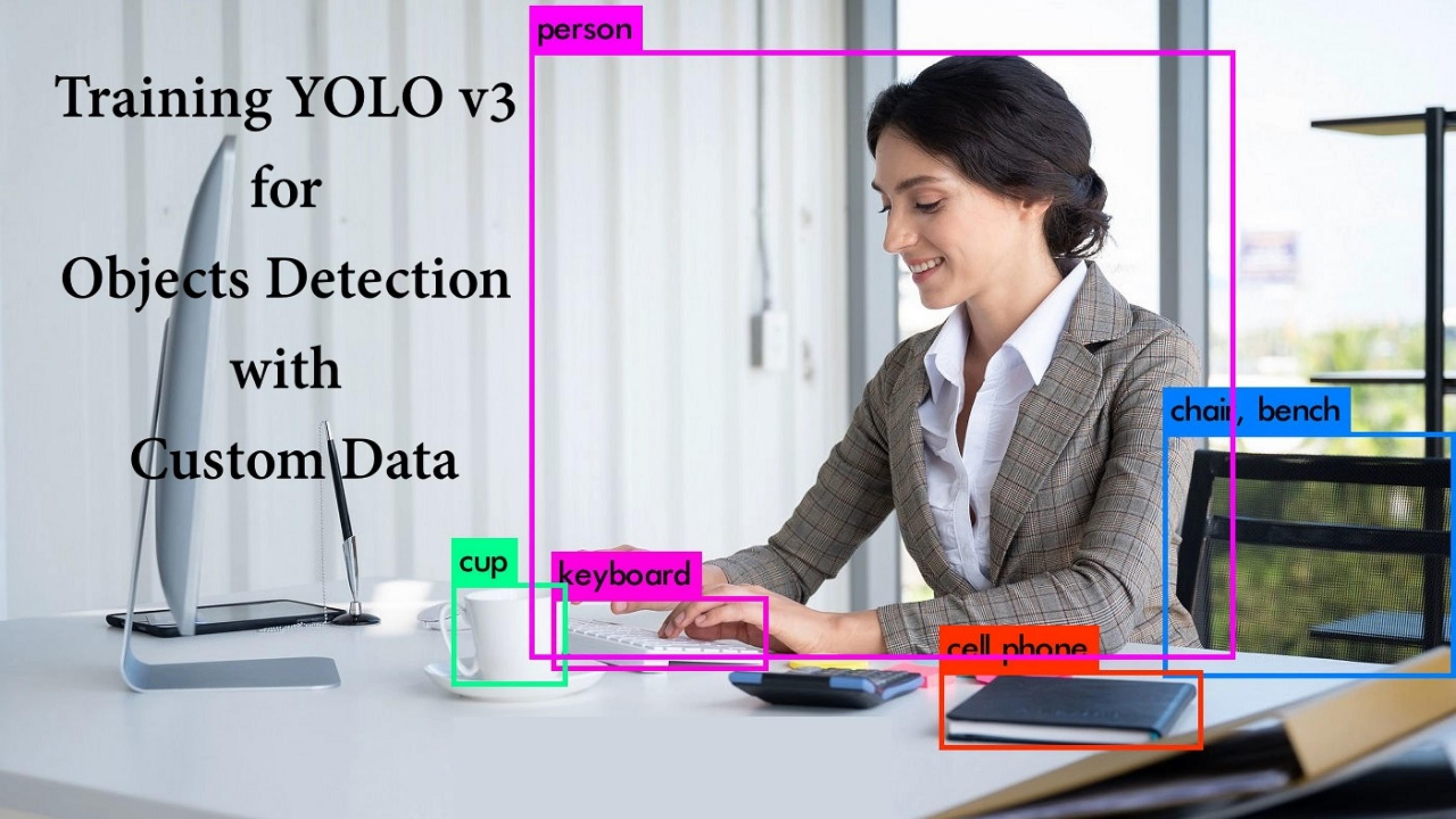


Training YOLO v3 for Objects Detection with Custom Data



Lecture organization

Content

- What is YOLO?
- Architecture of YOLO v3
- Input
- Detections at 3 Scales
- Detection Kernels
- Grid Cells
- Anchor Boxes
- Predicted Bounding Boxes
- Objectness Score
- Conclusion



**Simple
definition**

YOLO:

"You Only Look Once"

uses

**Convolutional Neural
Networks**

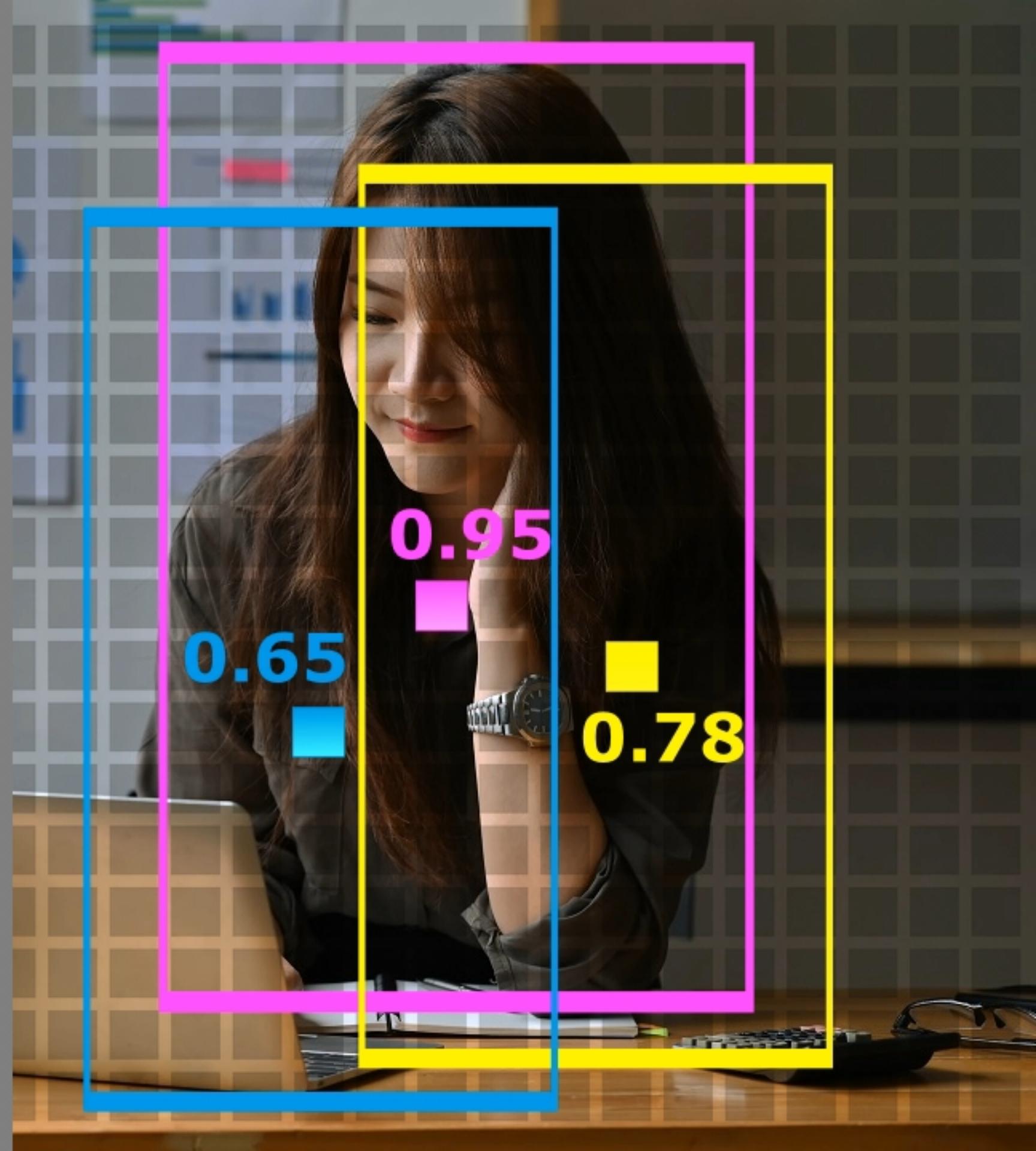
YOLO v3 can:

- ✓ **detect multiple objects**
- ✓ **predict classes**
- ✓ **identify locations**



YOLO v3:

- ✓ applies single NN
- ✓ divides image into grid cells
- ✓ produces cells' probabilities
- ✓ predicts Boxes



Prerequisites

Terminology

- CNNs
- Residual Blocks
- Skip connections
- Up-sampling
- Leaky ReLU
- IoU
- Non-maximum suppression



Layers' Structure

YOLO v3

- 53 CNNs layers (Darknet-53)
stacked with
- 53 more layers
producing
- 106 layers for YOLO v3



Darknet framework:

✓ loads 106 layers

✓ detections at
layers:

82, 94 and 106

```
velentyn@velentyn-asus: ~/Downloads/darknet
 90 conv    512      3 x 3/ 1      26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
 91 conv    256      1 x 1/ 1      26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
 92 conv    512      3 x 3/ 1      26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
 93 conv    255      1 x 1/ 1      26 x 26 x 512 -> 26 x 26 x 255 0.177 BF
 94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
 95 route  91
                                     -> 26 x 26 x 256
 96 conv    128      1 x 1/ 1      26 x 26 x 256 -> 26 x 26 x 128 0.044 BF
 97 upsample
                                     2x   26 x 26 x 128 -> 52 x 52 x 128
 98 route  97 36
                                     -> 52 x 52 x 384
 99 conv    128      1 x 1/ 1      52 x 52 x 384 -> 52 x 52 x 128 0.266 BF
100 conv    256      3 x 3/ 1      52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
101 conv    128      1 x 1/ 1      52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
102 conv    256      3 x 3/ 1      52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
103 conv    128      1 x 1/ 1      52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
104 conv    256      3 x 3/ 1      52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
105 conv    255      1 x 1/ 1      52 x 52 x 256 -> 52 x 52 x 255 0.353 BF
106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.864
Loading weights from weights/yolov3.weights...
seen 64
```

Improvements

Essential elements:

- * Residual Blocks
- * Skip Connections
- * Up-Sampling

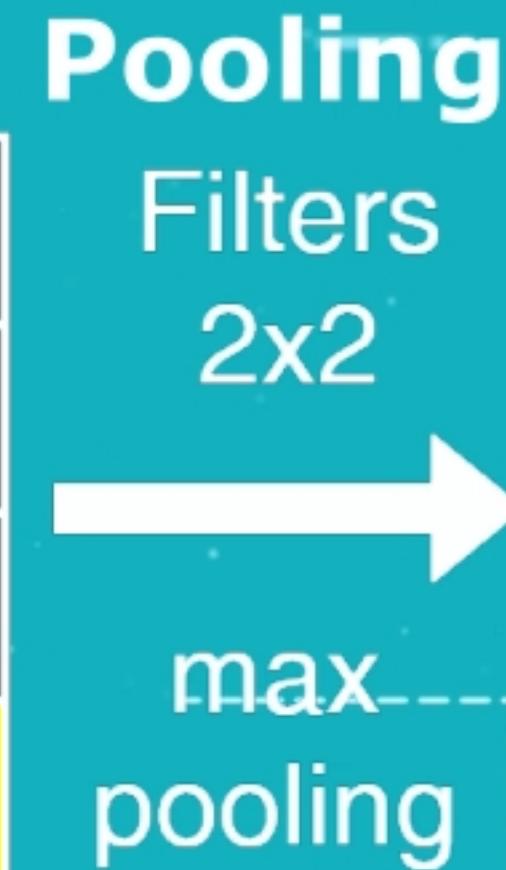
CNN Layers followed by

- * Batch Normalization
- * Leaky ReLU

YOLO v3:

- ✓ **No Pooling layers**
- ✓ **Convolutional layers instead**
- ✓ **Prevents loss of low-level features**
- ✓ **Ability to detect small objects**

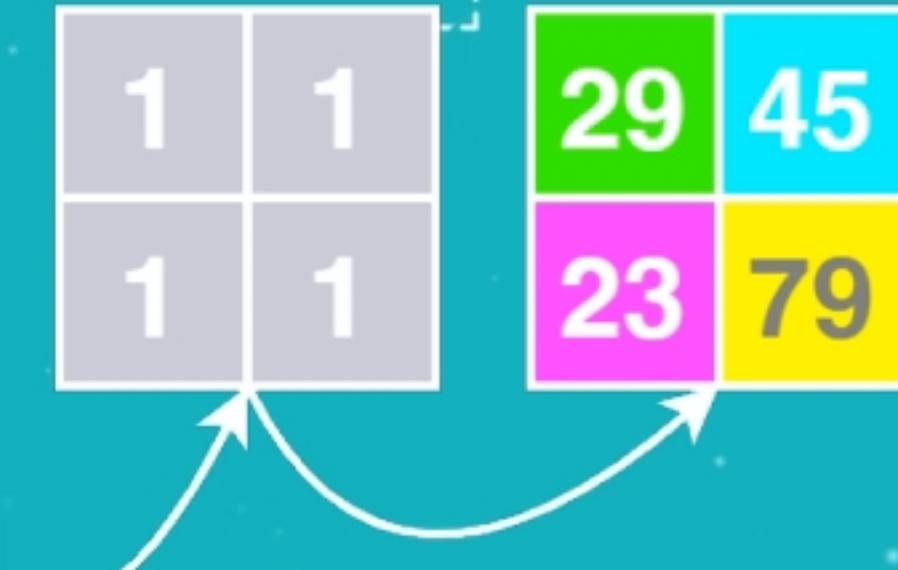
3	10	23	16
9	7	4	2
11	9	19	18
1	2	17	25



3	10	23	16
9	7	4	2
11	9	19	18
1	2	17	25

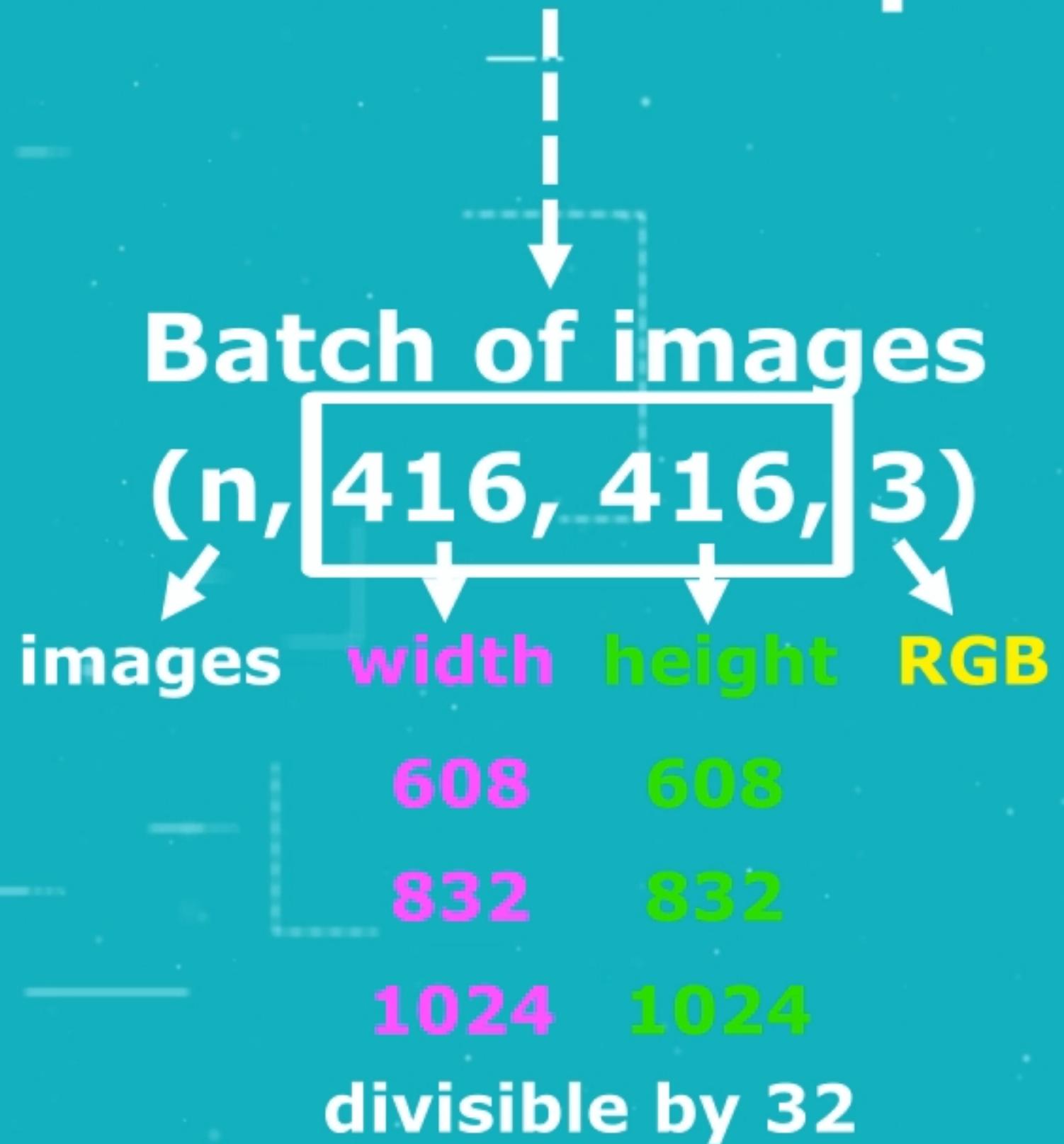
Convolution

Filters 2x2, stride 2



Input's
shape

Network's Input



Input images:

- ✓ Can be of any size
- ✓ Will be resized to network size

Aspect ratio:

- ✓ Can be kept or not
- ✓ Compare and choose best approach

608 x 608



not keeping aspect ratio

Layers of detection

Detections at:

82, 94, 106 layers

Downsamples input:

8

16

32



Network strides:

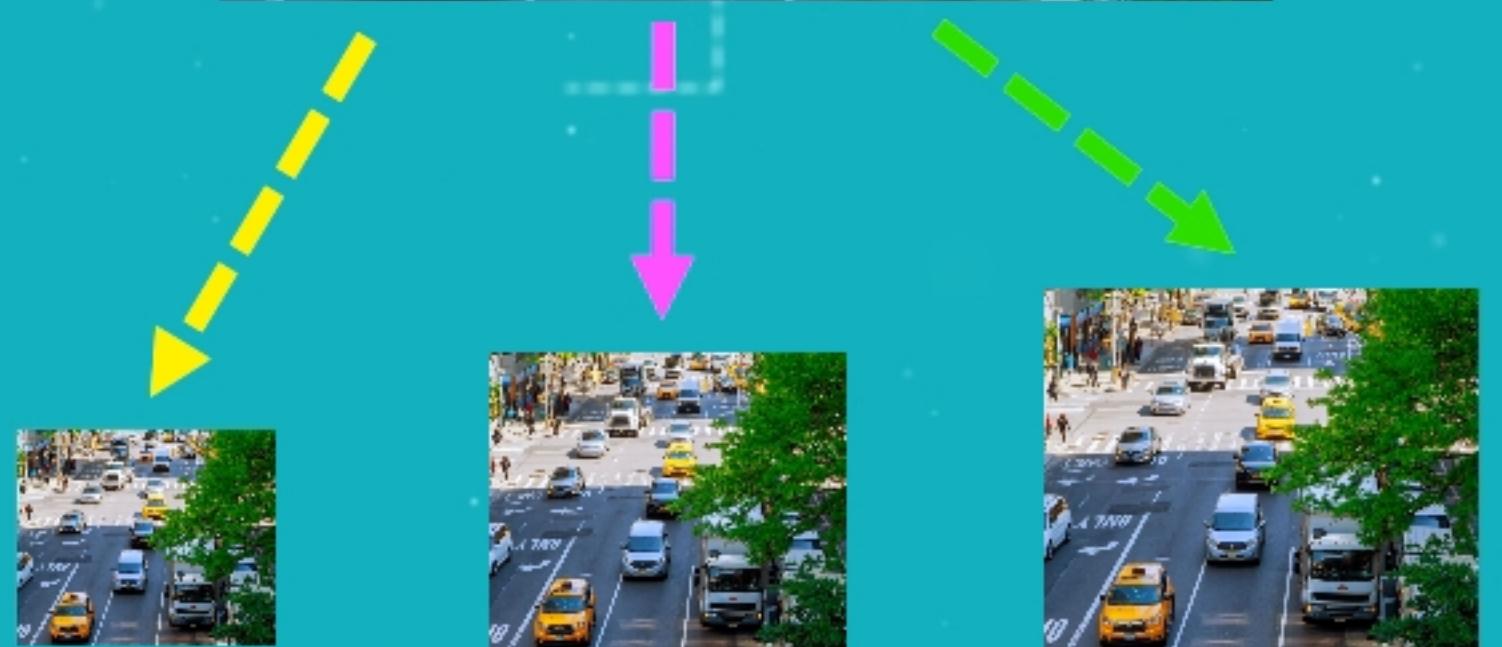
32, **16,** **8**

at layers
82, 94, 106

Size of outputs:

13x13, **26x26**, **52x52**
large medium small

Network's input
416 x 416



Possible inputs

Network's Input

Batch of images
 $(n, 416, 416, 3)$

images	width	height	RGB
608	608		
832	832		
1024	1024		

divisible by 32 -> 16 -> 8

**Filters to be
trained**

Detections at:

82, 94, 106 layers

By 1x1 kernels

applied to



Downsampled images:

52x52

13x13



26x26



Equation:

✓ $(b * (5 + c))$,

where:

✓ **$b = 3$, number of BB**

✓ **$(5 + c)$, BB attributes**

✓ **$c = 80$, COCO classes**

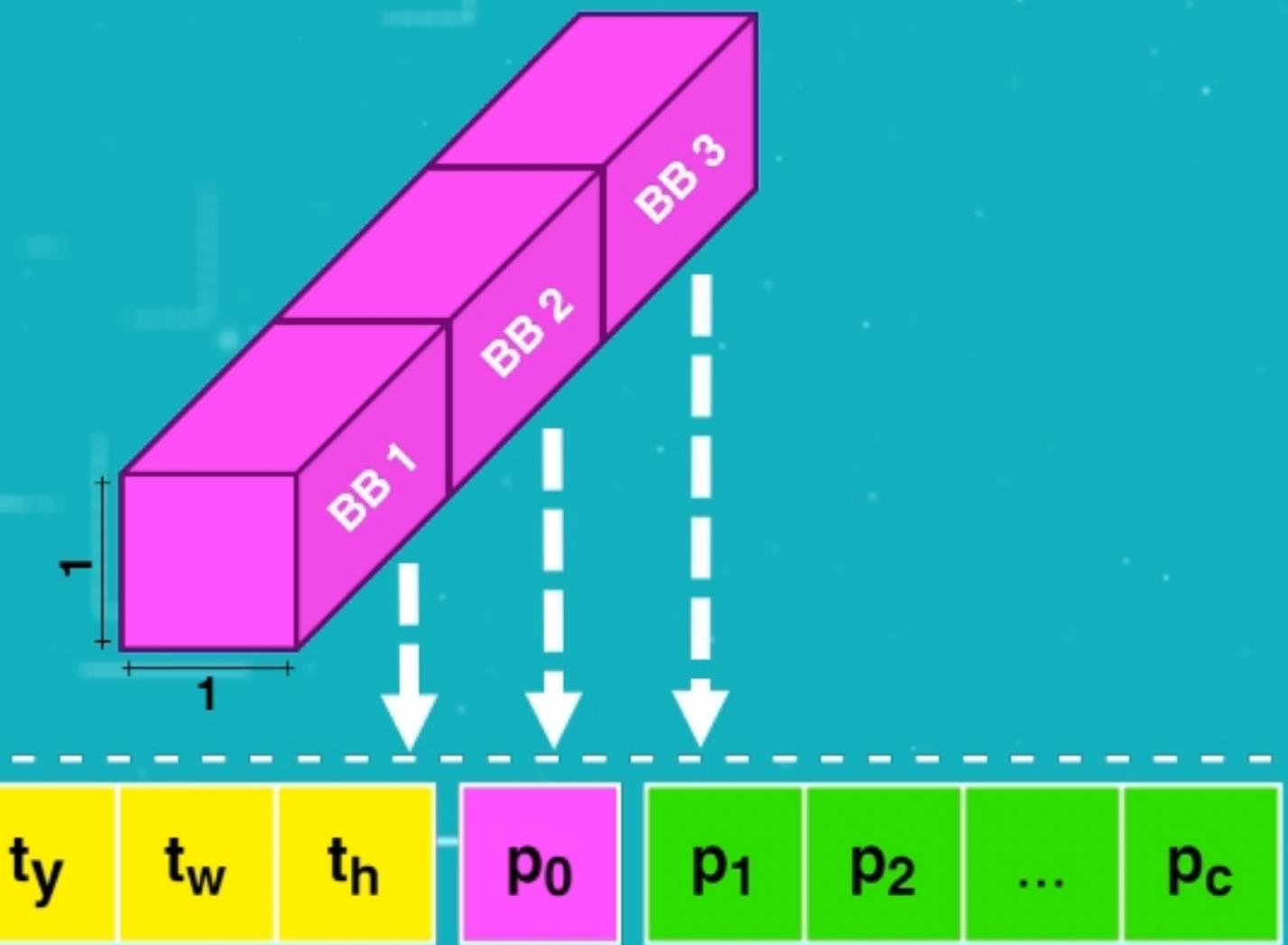
Result:

✓ **$(3 * (5 + 80)) = 255$**

Feature maps:

(13x13), (26x26), (52x52)

1x1 kernels' shape:



Maps' Shapes

Detections at:
82, 94, 106 layers

1x1 kernels
produce

Feature maps:

(13, 13, 255)

(26, 26, 255)

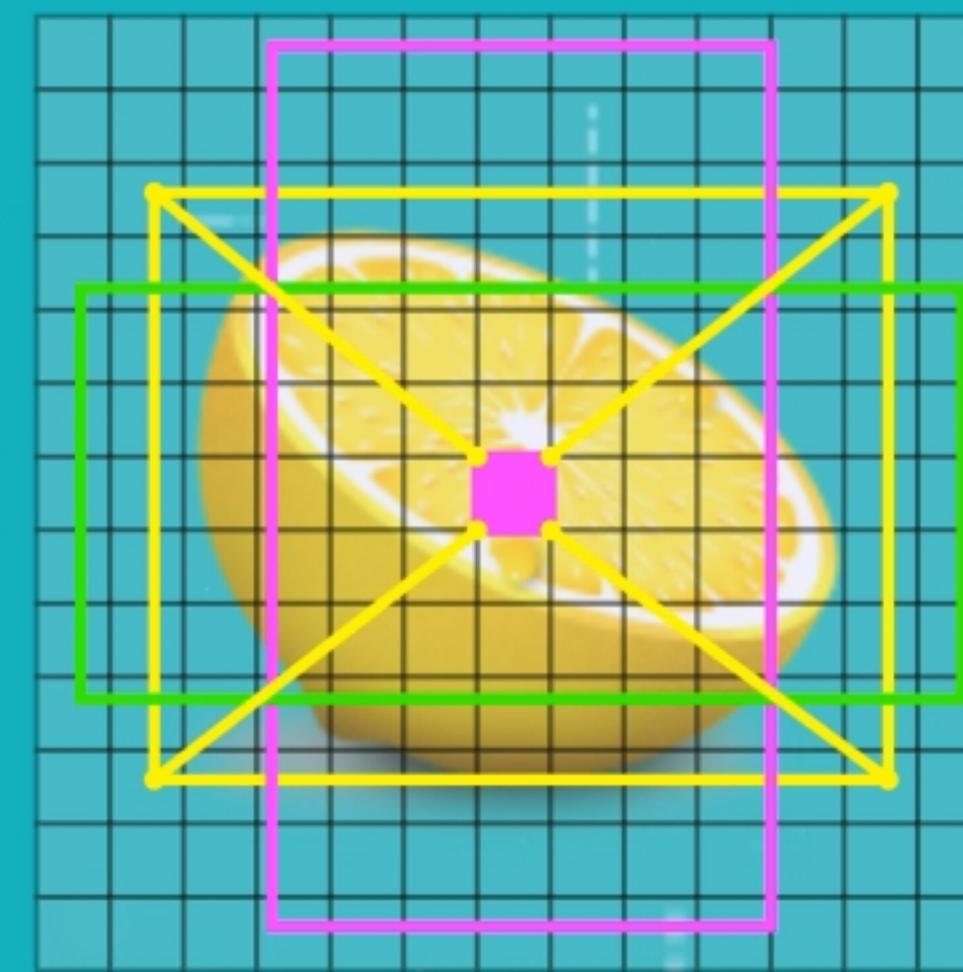
(52, 52, 255)

Detection cells

Feature maps:

(13, 13, 255), (26, 26, 255), (52, 52, 255)

3 BB for each cell



Training YOLO v3:

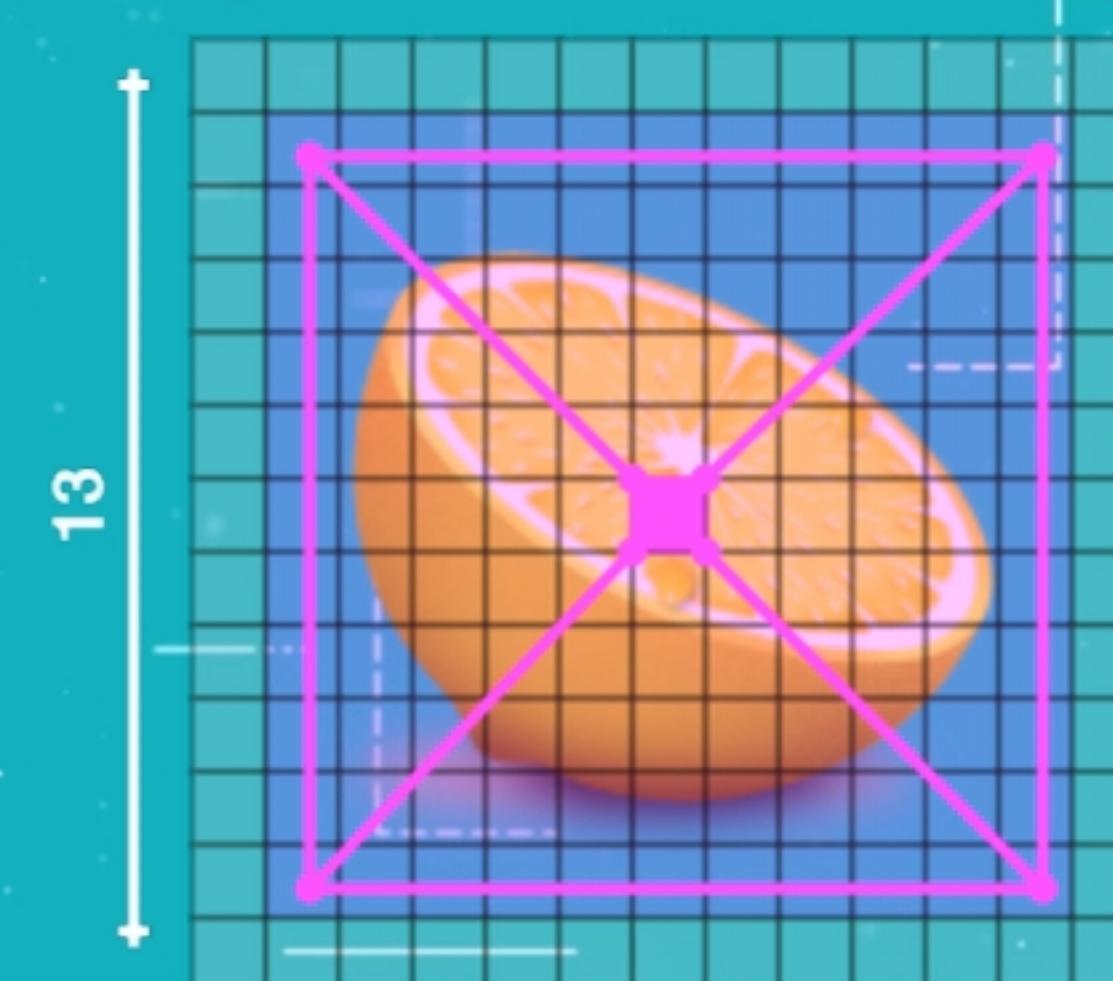
- ✓ One ground truth BB for one object
- ✓ Centre cell is assigned to predict this object
- ✓ Cell's objectness = 1

Input image:

416 x 416

Scale 1, stride 32:

13 x 13



Predictions by Anchor Boxes

Detections at layers:

82,

3
anchors

94,

3
anchors

106

3
anchors

9

Anchors (priors)
used to calculate

Predicted BB
real width and height

Calculations of Anchors

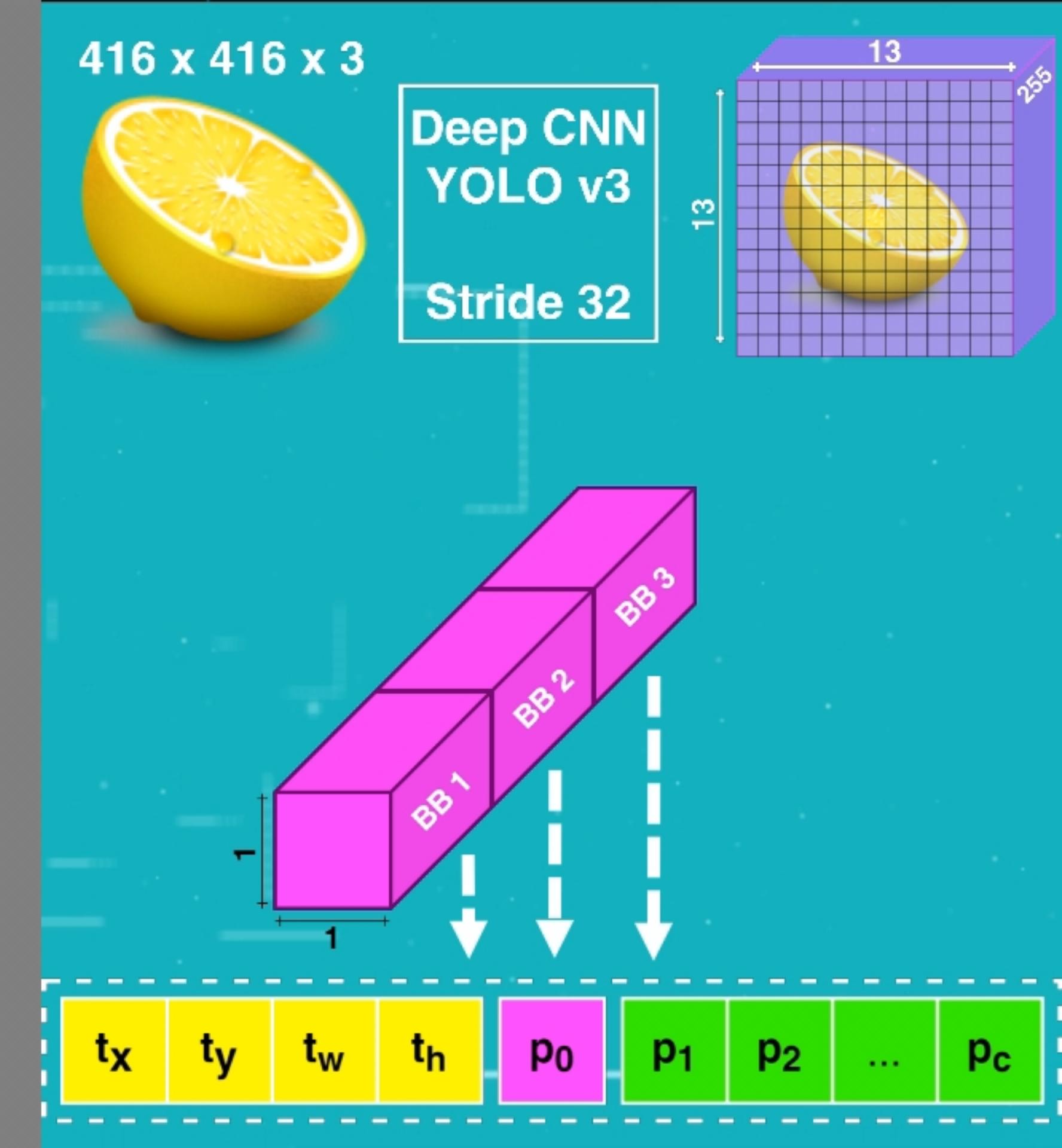
k-means clustering
applied to calculate

Anchors (priors)
for COCO dataset

Scale-1	Scale-2	Scale-3
(116×90)	(30×61)	(10×13)
(156×198)	(62×45)	(16×30)
(373×326)	(59×119)	(33×23)

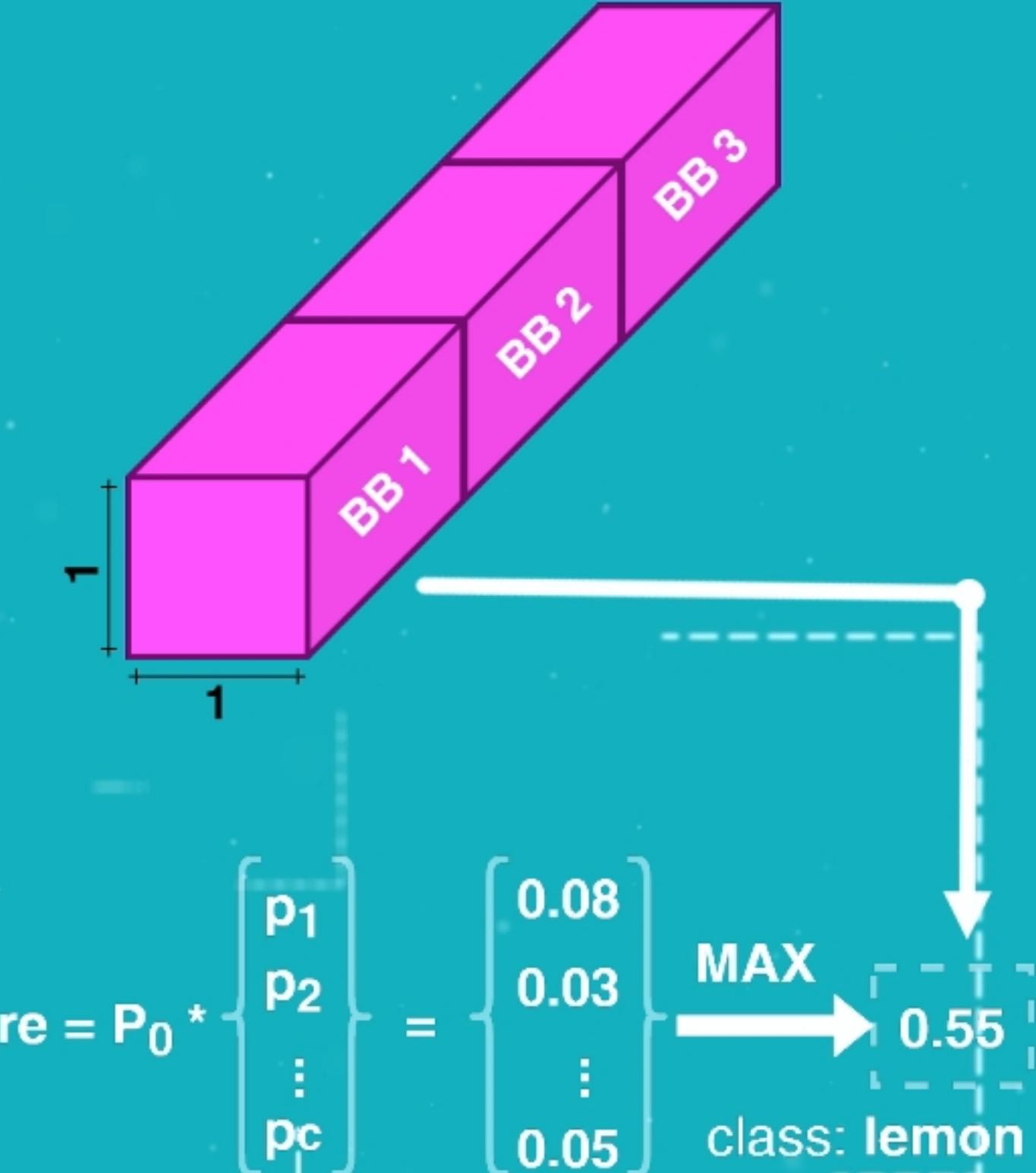
BB attributes obtained by:

- ✓ **Input (416, 416, 3)**
- ✓ **Deep CNN of YOLO v3**
- ✓ **Downsampled input image by stride 32**
- ✓ **(13, 13, 255) feature map at Scale-1**



BB probabilities:

- ✓ Extract probabilities of BB
- ✓ Compute elementwise product
- ✓ Find maximum probability



Number of Bounding Boxes

**13x13 cells across
3 BB and
80 classes' confidences**

predicts

Bounding Boxes

Scale-1

507

Scale-2

2028

Scale-3

8112

10 647

Dimensions of Anchors

Anchors
are



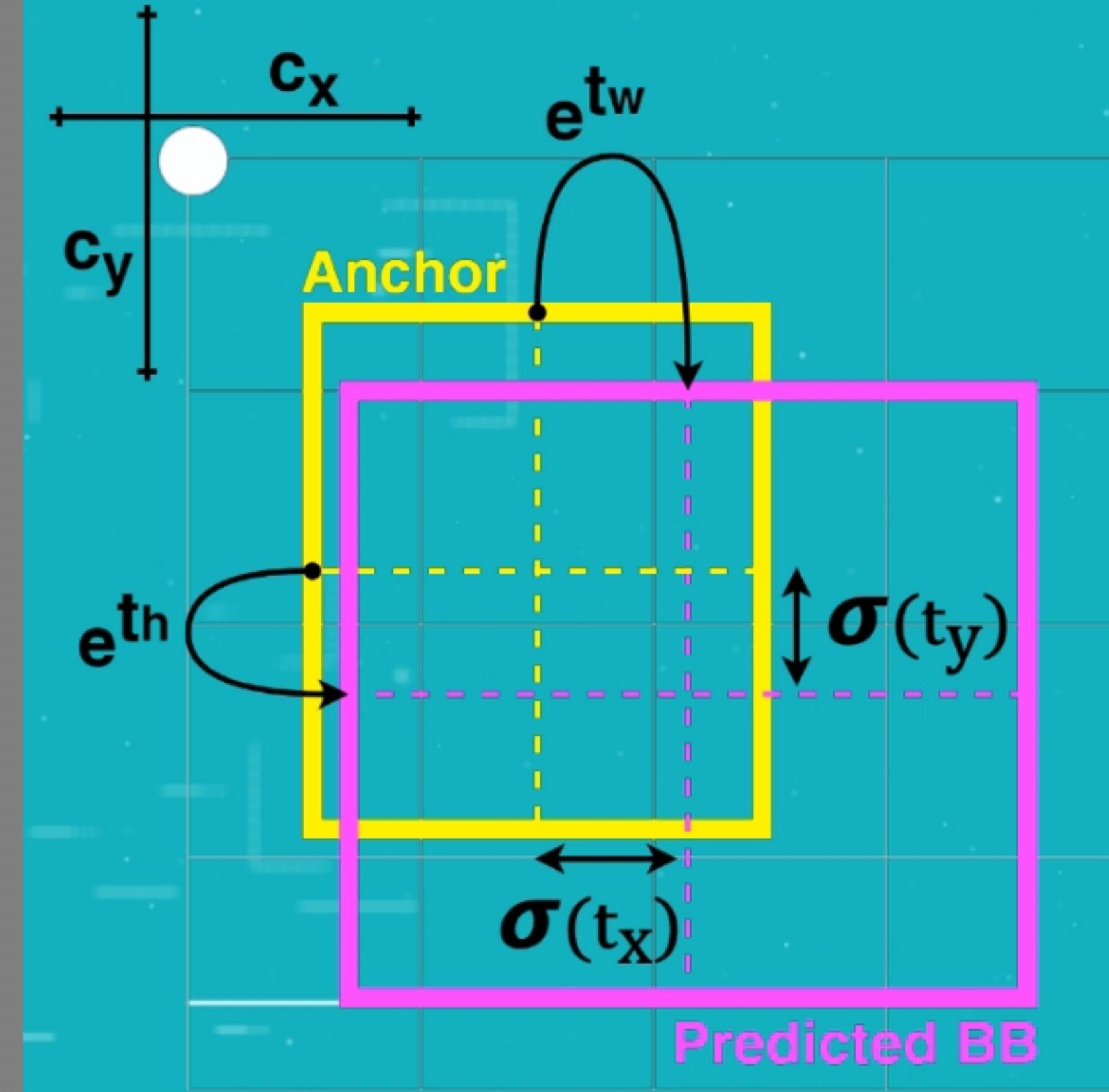
Bounding Boxes' priors
calculated by
K-means clustering

COCO dataset:

Scale-1	Scale-2	Scale-3
(116×90)	(30×61)	(10×13)
(156×198)	(62×45)	(16×30)
(373×326)	(59×119)	(33×23)

To predict BB:

- ✓ Calculate offset to anchor
- also called as:
- ✓ Log-space transform
- ✓ Pass centre through sigmoid function

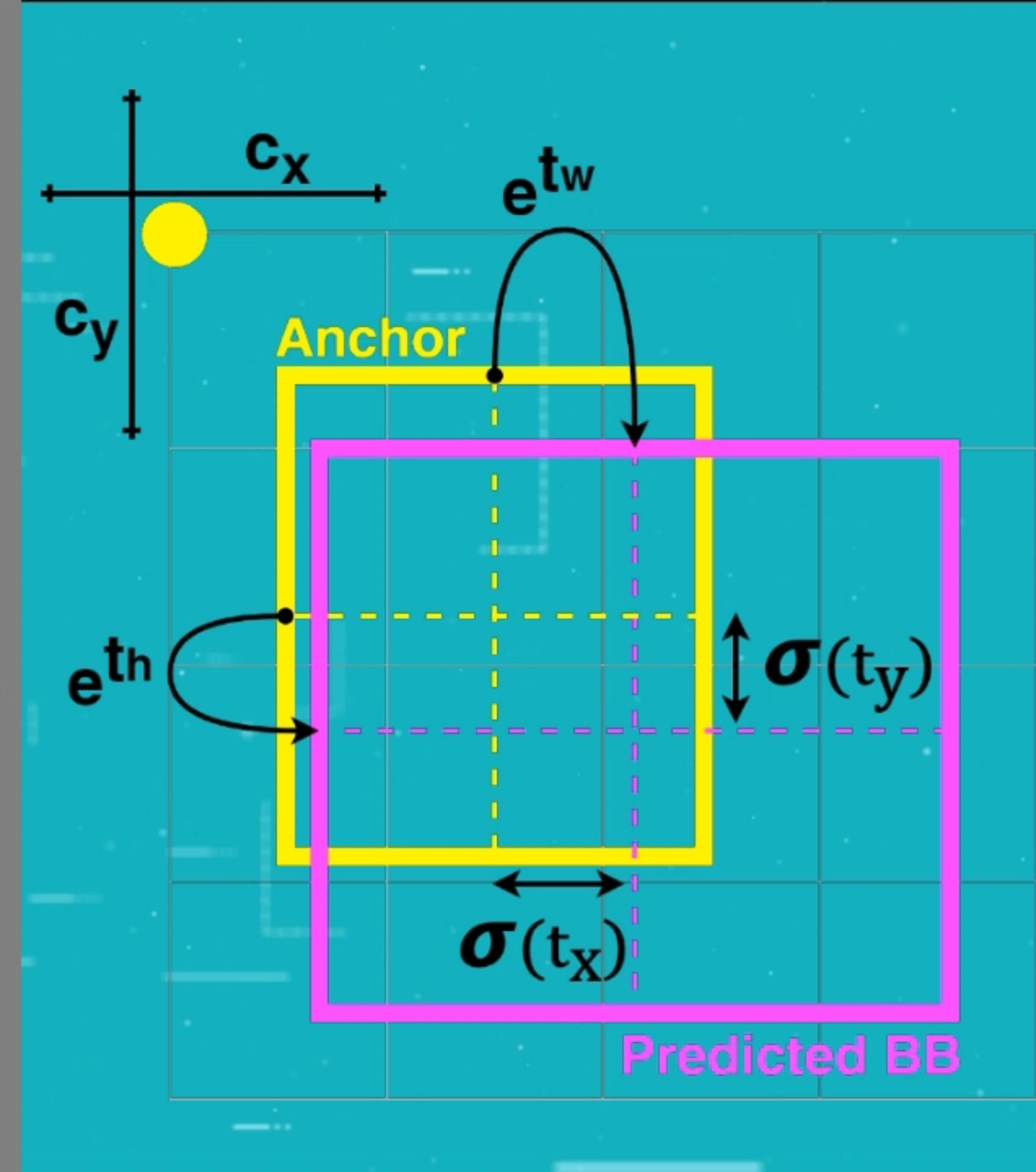


Equations:

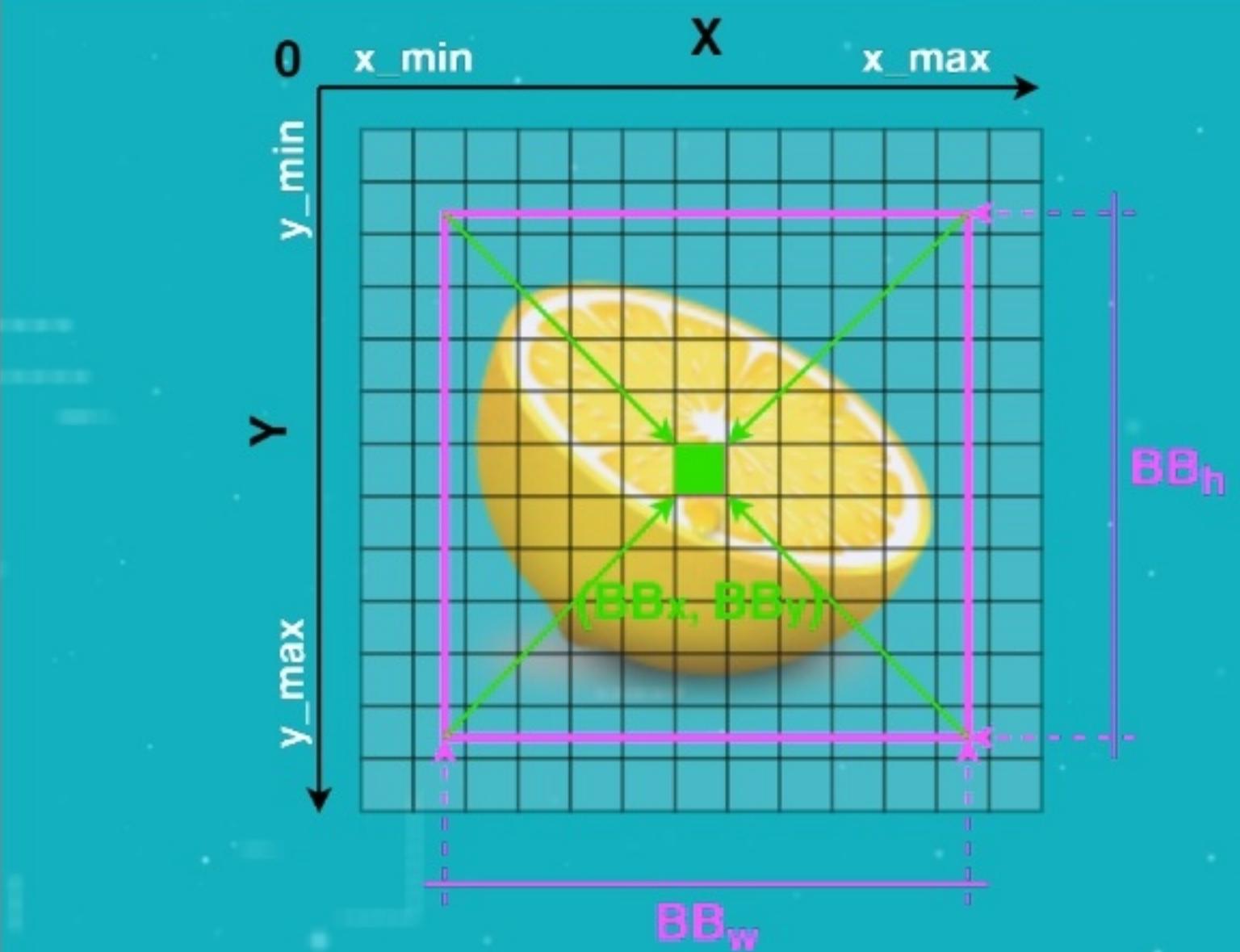
$$\left\{ \begin{array}{l} b_x = \sigma(t_x) + c_x \\ b_y = \sigma(t_y) + c_y \\ b_w = p_w * e^{t_w} \\ b_h = p_h * e^{t_h} \end{array} \right.$$

where:

- ✓ b_x, b_y, b_w, b_h - centre, width, height of predicted BB
- ✓ t_x, t_y, t_w, t_h - outputs of NN
- ✓ c_x, c_y - cell's top left corner of the anchor box
- ✓ p_w, p_h - anchor's width, height



Coordinates of Bounding Boxes



$$BB_x = b_x * \text{width of image}$$

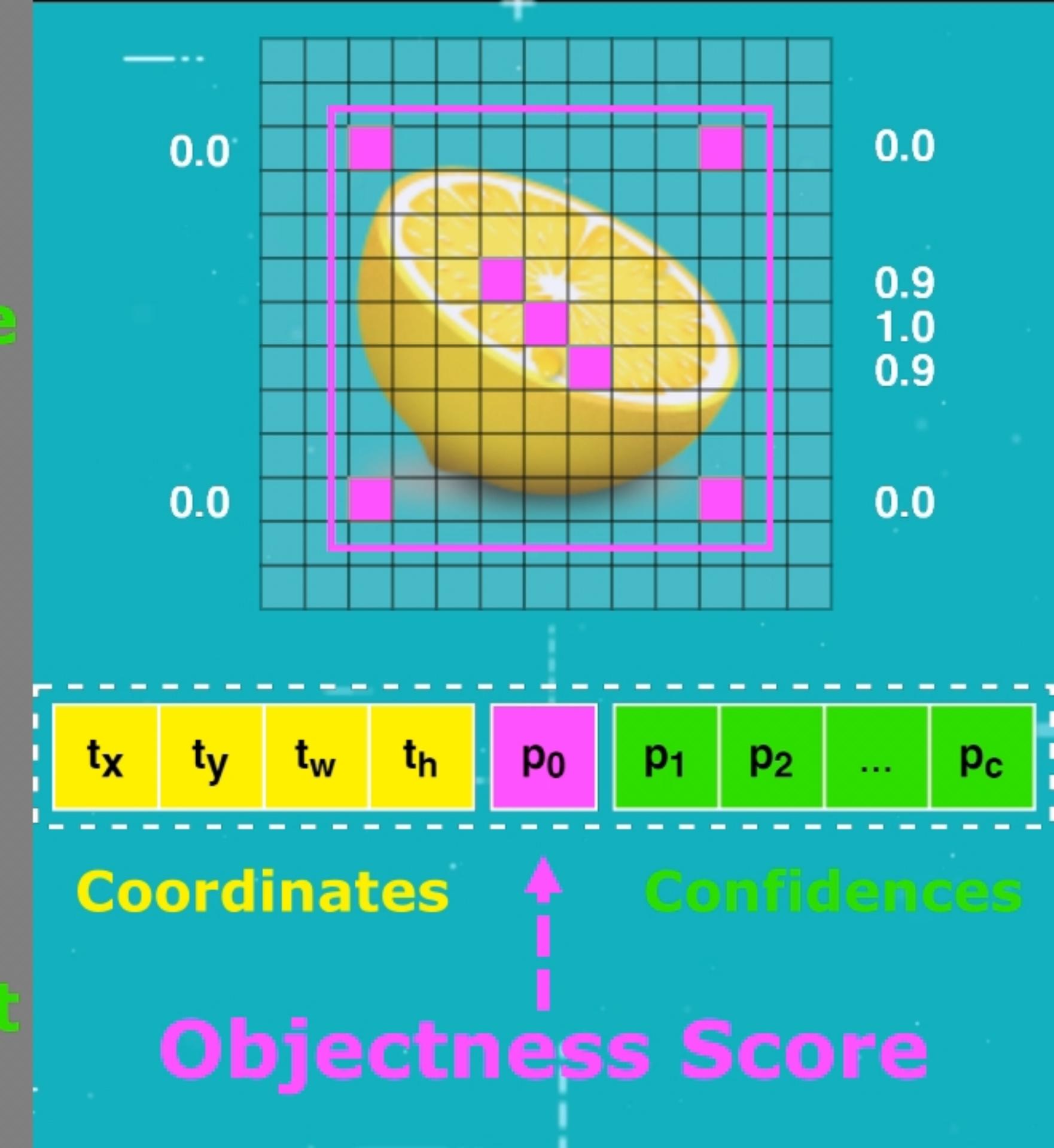
$$BB_y = b_y * \text{height of image}$$

$$BB_w = b_w * \text{width of image}$$

$$BB_h = b_h * \text{height of image}$$

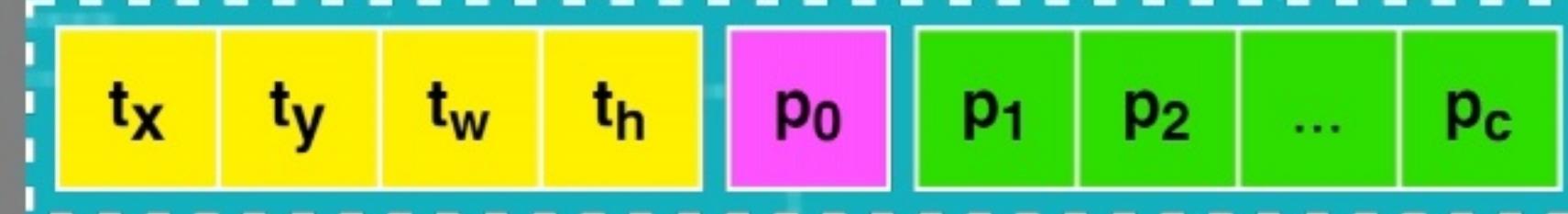
BB's attribute:

- ✓ p₀ **is Objectness Score**
- ✓ **Centre cell of ground truth BB has p₀ = 1**
- ✓ **Corner cells of ground truth BB has p₀ = 0**
- ✓ p₀ **is a probability that BB contains object**



Objectness Score

P_0 vs $\{P_1, P_2, \dots, P_c\}$



Detected BB
contains
object



Object
belongs to:

Person

Car

Cat

Calculating Objectness Score

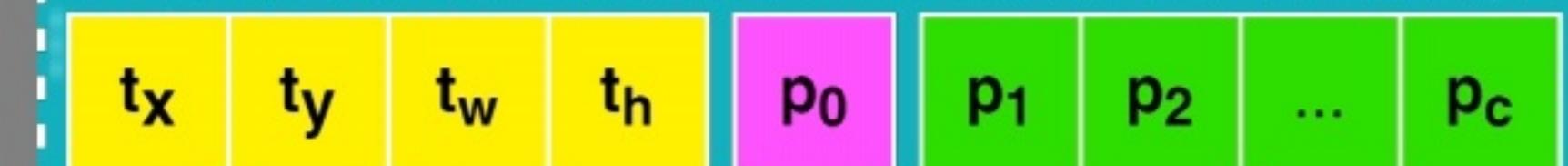
Equation for objectness score:

$$P_{\text{object}} * \text{IoU} = \sigma(t_0) = P_0$$

where

P_{object} - predicted probability

IoU - between predicted BB2 and ground truth BB1



[0..1]