Object Classification, Localization and Detection
Part 2

CS8001: Deep Learning and Applications

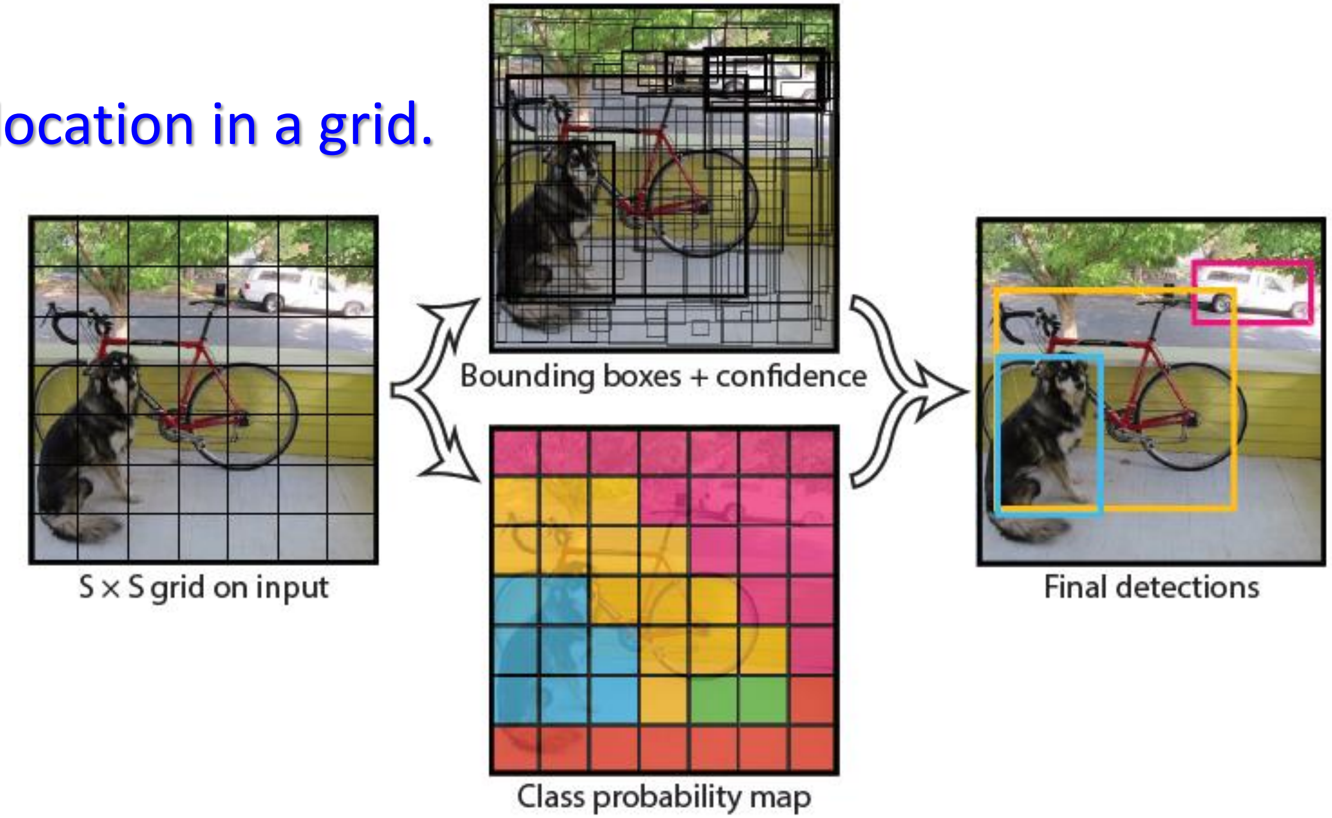
Slides based on : guillaumegenthial.github.io

YOLO

- A Single shot detector that trains a single CNN once only for all the objects in the scene.

Yolo : You Only Look Once

Basic Idea :
Predict a class and a
Bounding box for every location in a grid.



https://arxiv.org/abs/1506.02640 Redmon et al. CVPR 2016.

YOLO Features

- Computationally Very Fast, can be used on real time environment.
- Globally processing the entire image once only with a single CNN.
- Learn generalizable representations
- Maintains a high accuracy range.

How Does YOLO Work?

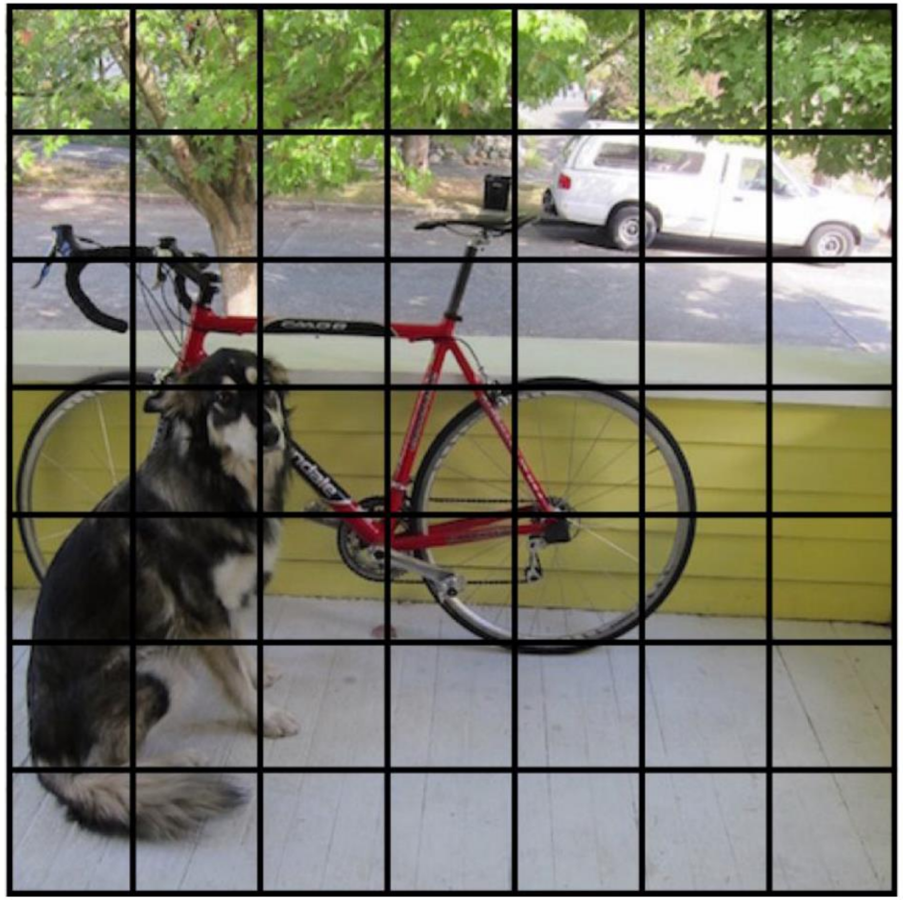
- Video from CVPR 2016

How Does YOLO Work?

- The algorithm "only looks once" at the image.
- Needs only one forward propagation pass through the network to make predictions. The network reasons globally about the full image and all the objects in the image in one go.
- It uses features from the entire image to predict each bounding box for objects.
- It also predicts all bounding boxes across all classes for an image simultaneously.
- It then outputs recognized objects together with the bounding boxes after a process called non-max suppression (We will see what is non-max suppression soon).
- The YOLO design enables end-to-end training and realtime speeds while maintaining high average precision.

YOLO

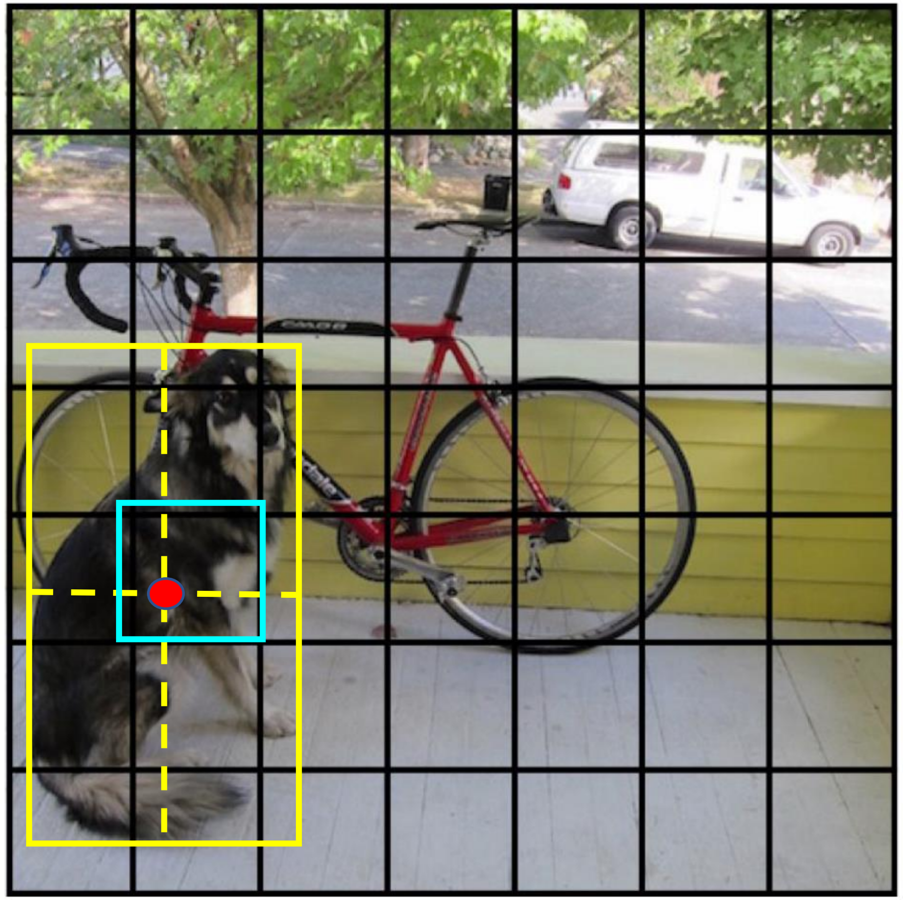
We split the image into an S*S grid



7*7 grid

YOLO

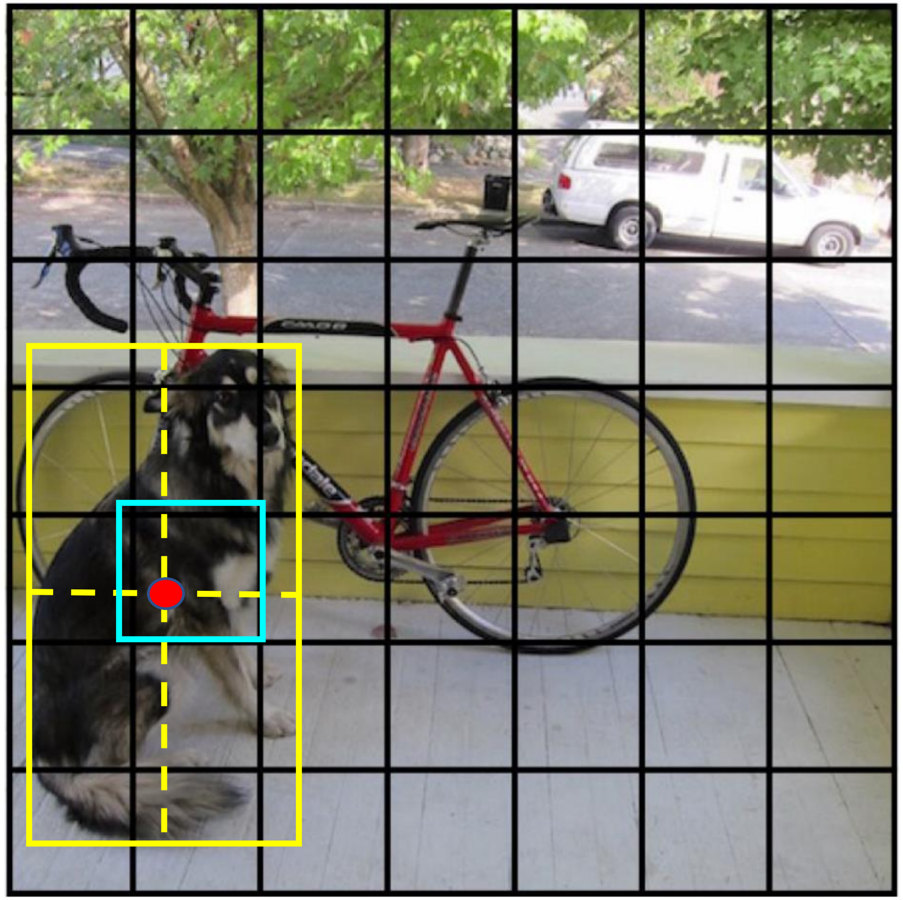
We split the image into an S*S grid



If the center/midpoint of an object falls into a grid cell, that grid cell is responsible for detecting that object.

YOLO

We split the image into an S*S grid

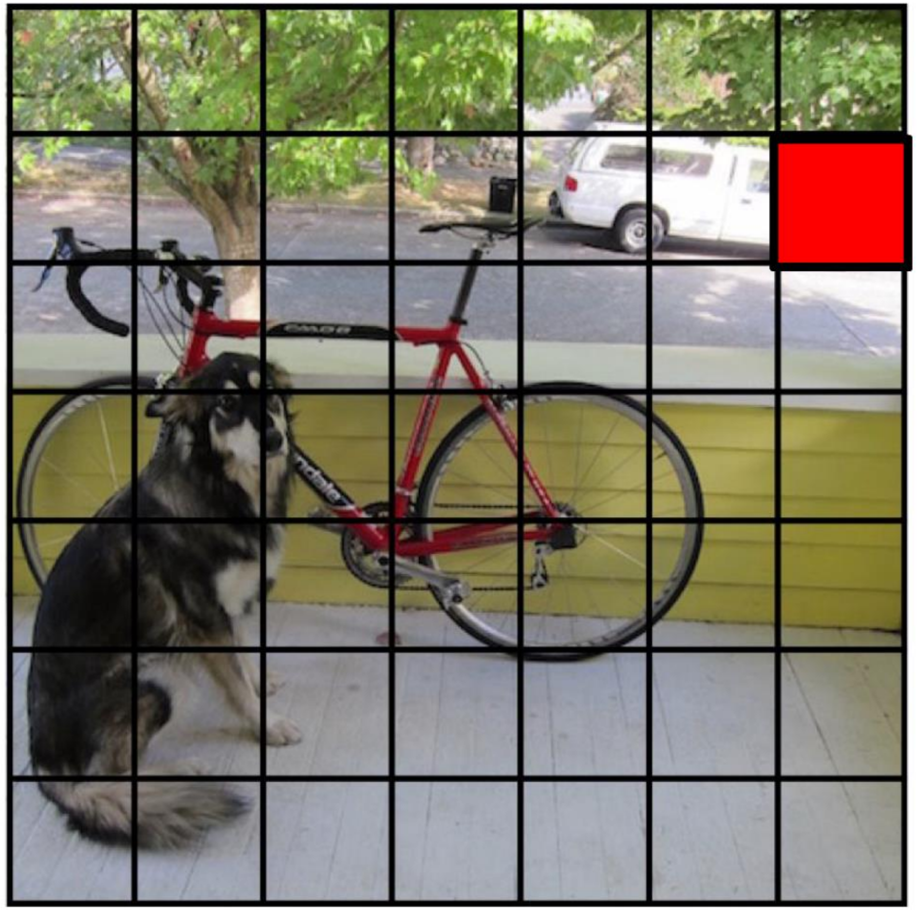


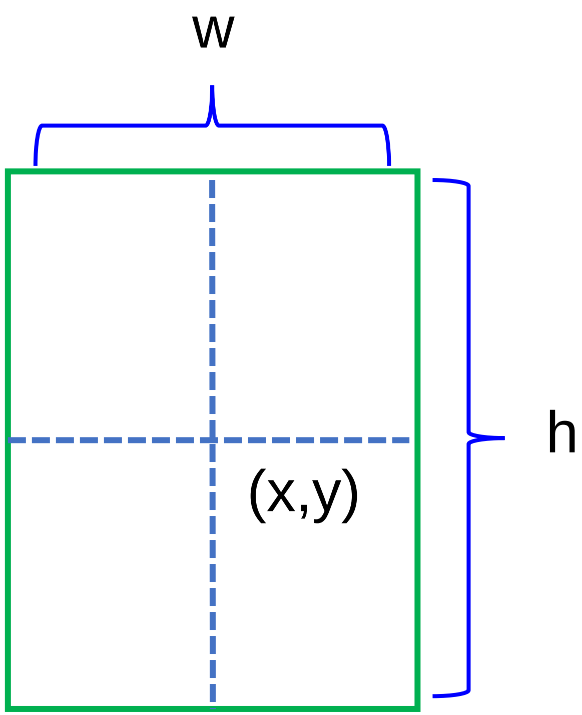
Each grid cell predicts B bounding boxes and confidence scores for those boxes.

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

YOLO Algorithm

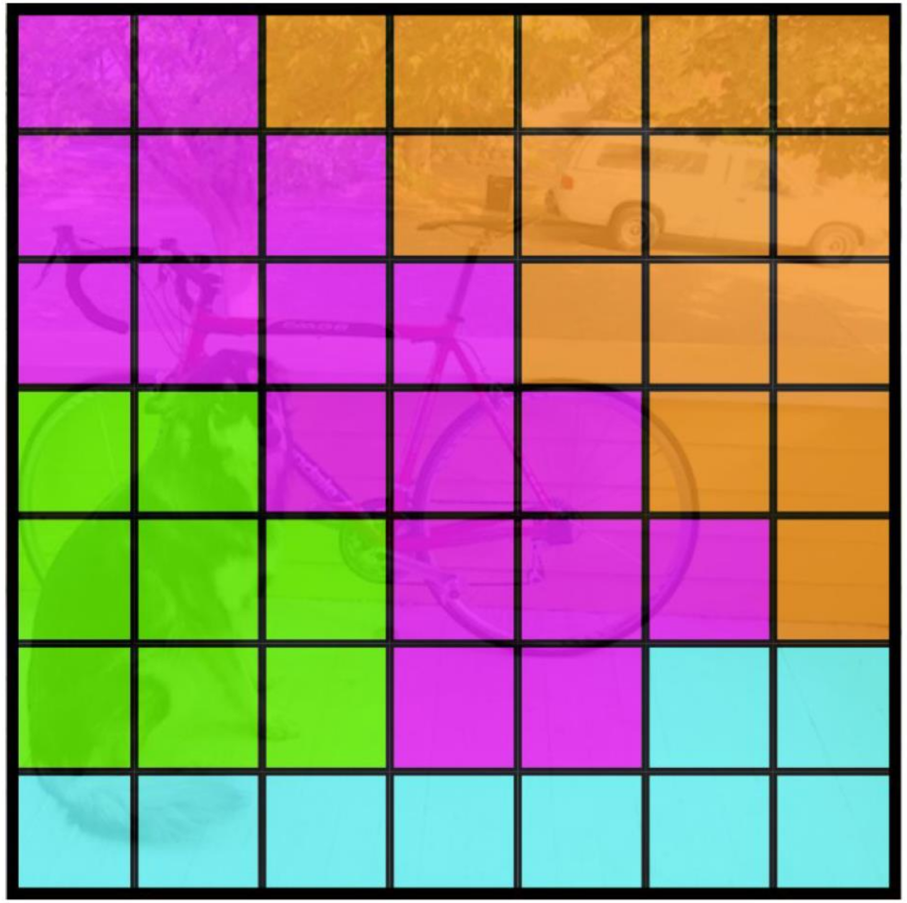
Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

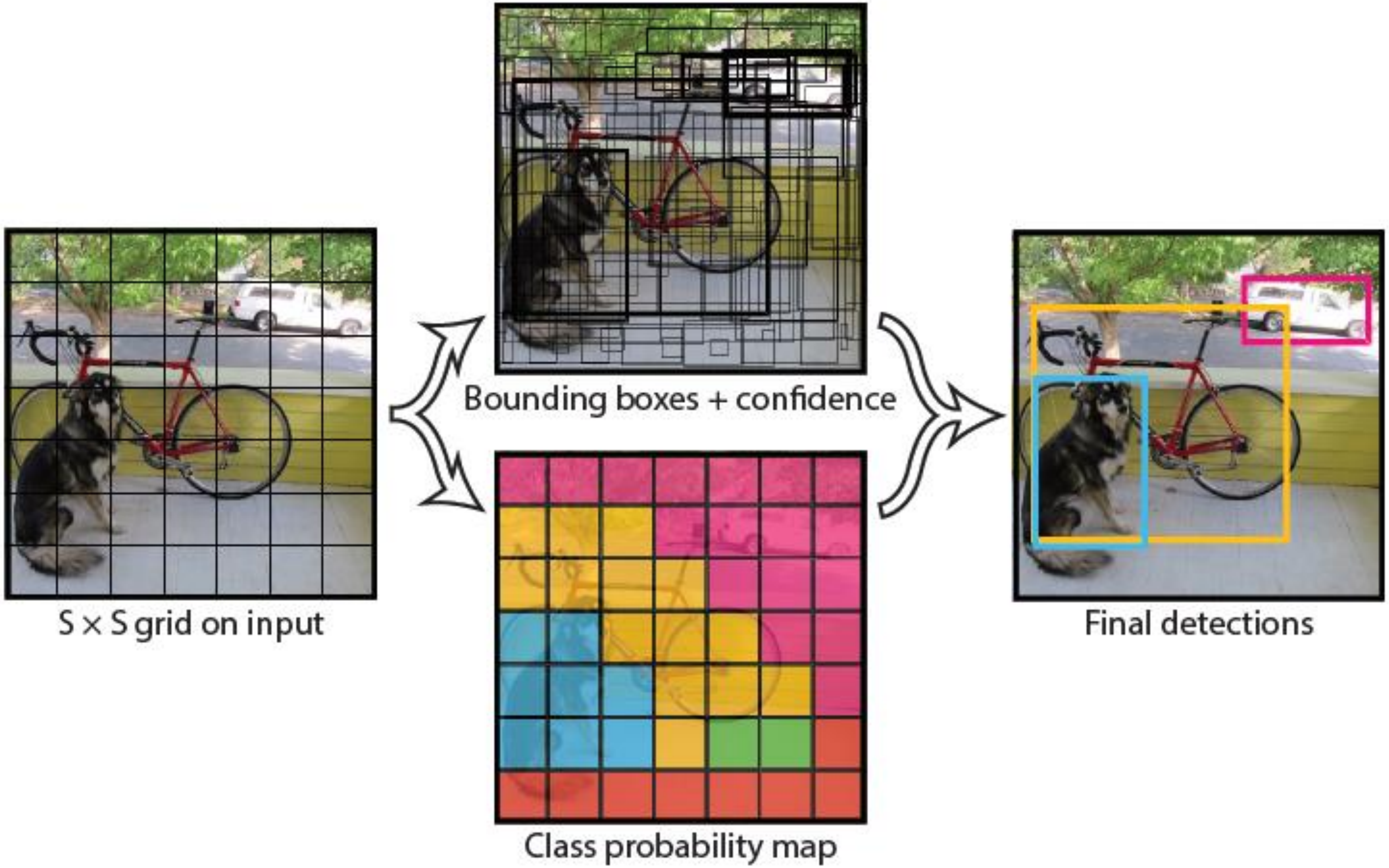




YOLO Algorithm

Each cell also predicts a class probability.




Bounding boxes + confidence
S × S grid on input
Final detections
Class probability map

Example

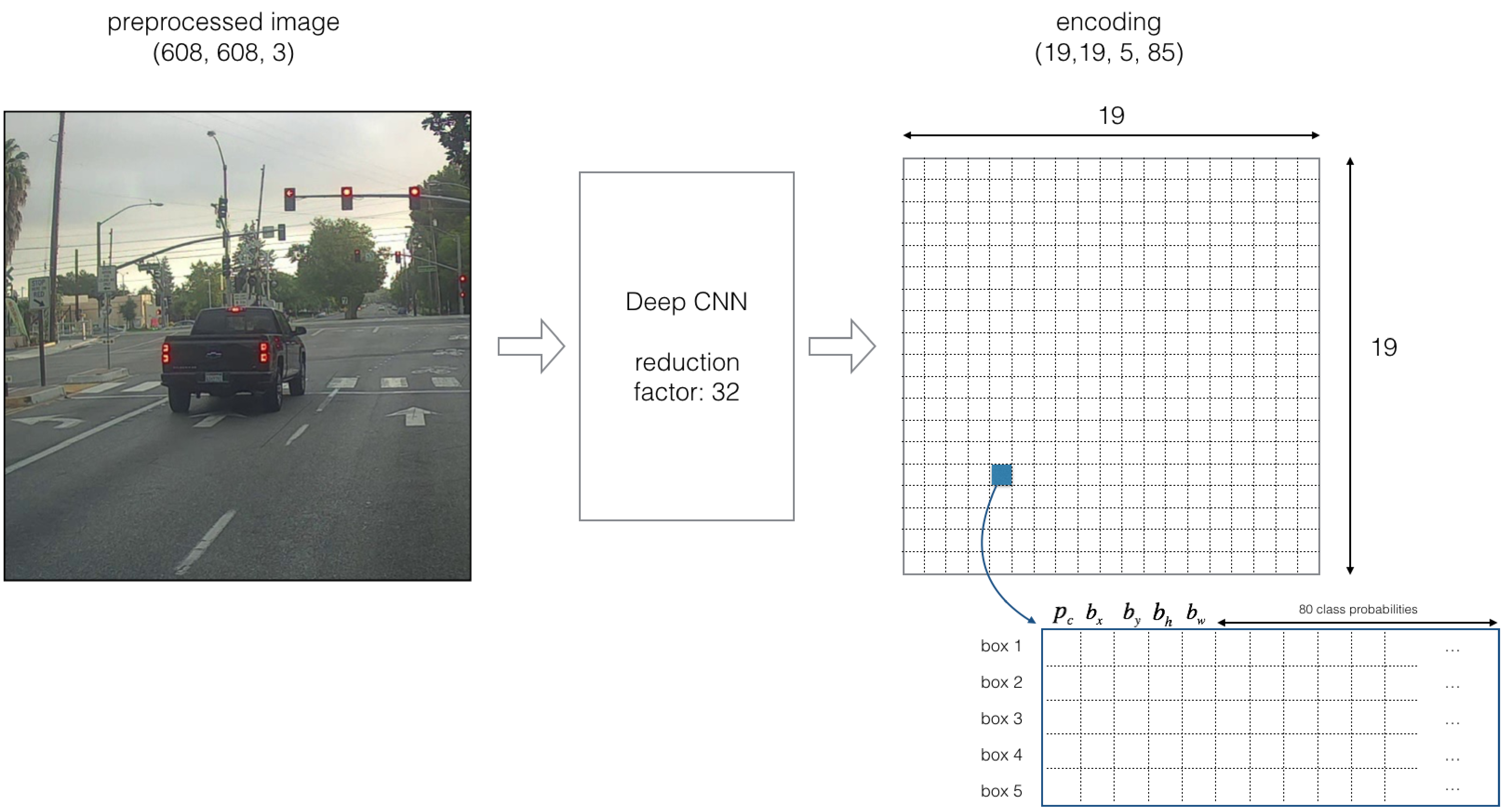
- Suppose the input images is of shape (608, 608, 3) and the batch size for CNN is m.
- Suppose a cell is of size 32×32 and there are 80 classes of objects in the dataset.
- The **output** is a list of bounding boxes along with the recognized classes.
- Each bounding box is represented by 6 numbers $\left(p_c, b_x, b_y, b_h, b_w, c\right)$
- The class probability c is a number (one of the 80 classes) or a 1-hot vector.
- If you expand c into an 80-dimensional vector, each bounding box is then represented by 85 numbers.
- Suppose B = 5, that means we will use 5 anchor boxes.
- Then the YOLO architecture has following input / output :

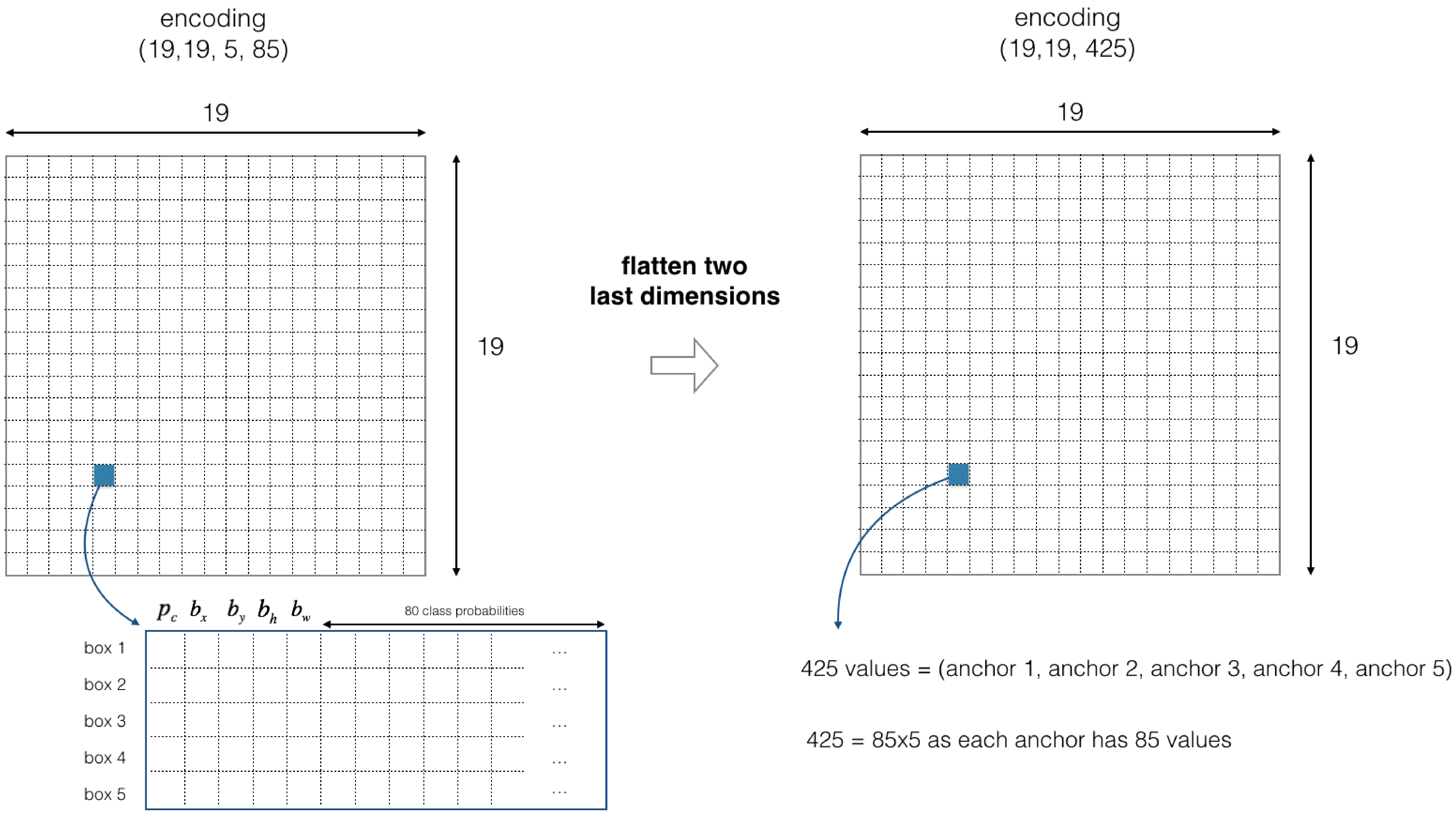
IMAGE (m, 608, 608, 3) -> YOLO CNN -> ENCODING (m, 19, 19, 5, 85).

Bounding Boxes Normalization

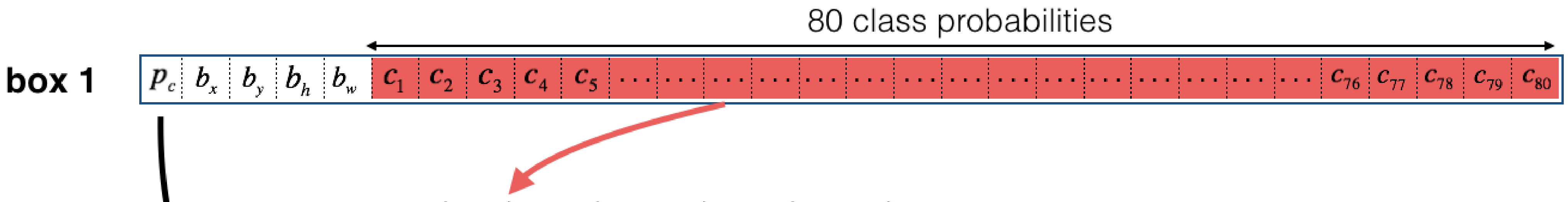
- Normalize the bounding box width and height by the image width and height so that they fall between 0 and 1.

- Parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.


preprocessed image
(608, 608, 3)
Deep CNN
reduction factor: 32
encoding
(19,19, 5, 85)
19
19
p_c b_x b_y b_h b_w
80 class probabilities
box 1
box 2
box 3
box 4
box 5


encoding
(19,19, 5, 85)
19
19
p_c b_x b_y b_h b_w
80 class probabilities
box 1
box 2
box 3
box 4
box 5
flatten two
last dimensions
encoding
(19,19, 425)
19
19
425 values = (anchor 1, anchor 2, anchor 3, anchor 4, anchor 5)
425 = 85x5 as each anchor has 85 values

For each anchor box, compute elementwise product to extract a probability that the box contains a certain class.



$$scores = p_c * \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{78} \\ c_{79} \\ c_{80} \end{pmatrix} = \begin{pmatrix} p_c c_1 \\ p_c c_2 \\ p_c c_3 \\ \vdots \\ p_c c_{78} \\ p_c c_{79} \\ p_c c_{80} \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.13 \\ 0.44 \\ \vdots \\ 0.07 \\ 0.01 \\ 0.09 \end{pmatrix}$$

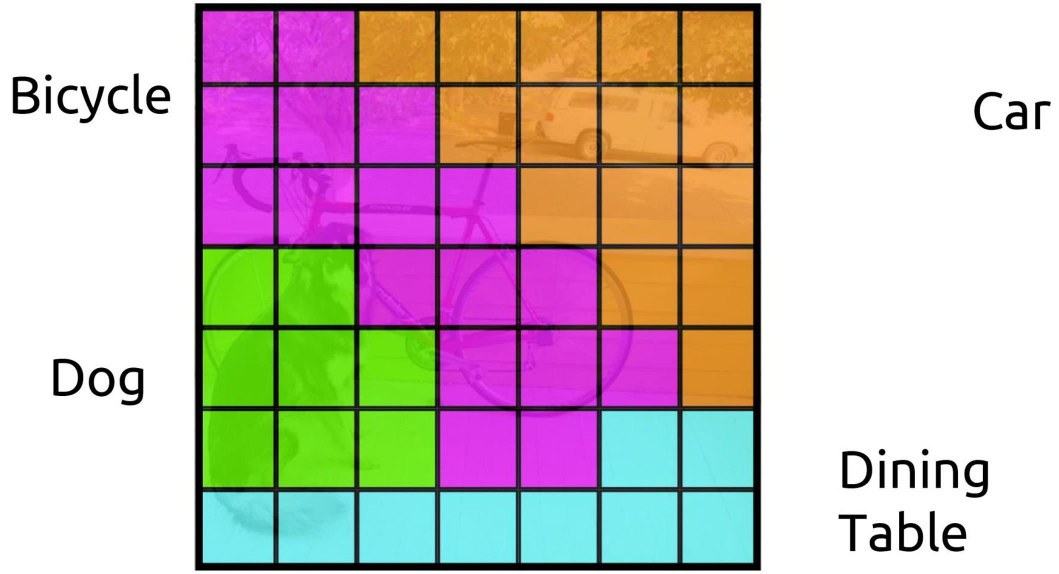
find the max $\longrightarrow$

score: 0.44
box: (b_x, b_y, b_h, b_w)
class: c = 3 (“car”)

the box (b_x, b_y, b_h, b_w) has detected c = 3 (“car”) with probability score: 0.44

YOLO’s Prediction

- For each of the 19x19 grid cells, the maximum of the probability scores (taking a max across both the 5 anchor boxes and across different classes).
- Color that grid cell according to what object that grid cell considers the most likely.



Too Many Boxes!

Dealing with Anchor Boxes

- Two stage filtering out of anchor boxes.

- Set a threshold on confidence of a box detecting a class.

  - Ignore boxes with a low score, that is, when the box is not very confident about detecting a class.

- Select only one box when several boxes overlap with each other and detect the same object. How ?

Box Confidence

- Remember the anchor box confidence depends on two factors.

    1. How confident the model is that the box contains an object, and
    2. how accurate it thinks the box is that it predicts.

- It is defined as $\Pr(Object) \times IoU$.

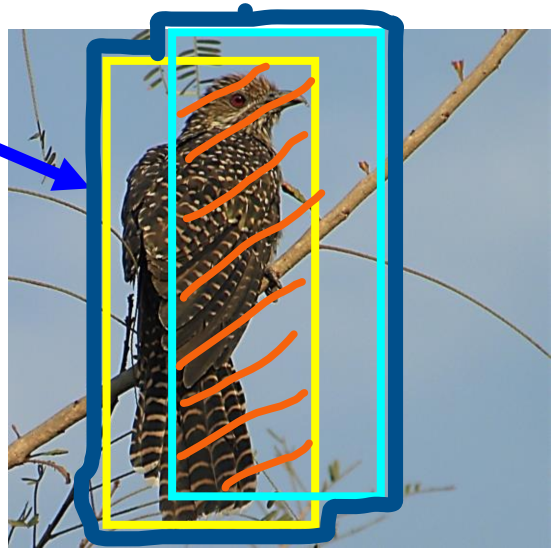
- What is IoU ?

- IoU = Intersection over union. It is a measure of the overlap between the actual (ground truth) bounding box and predicted bounding box.

IoU

Estimated
bounding box

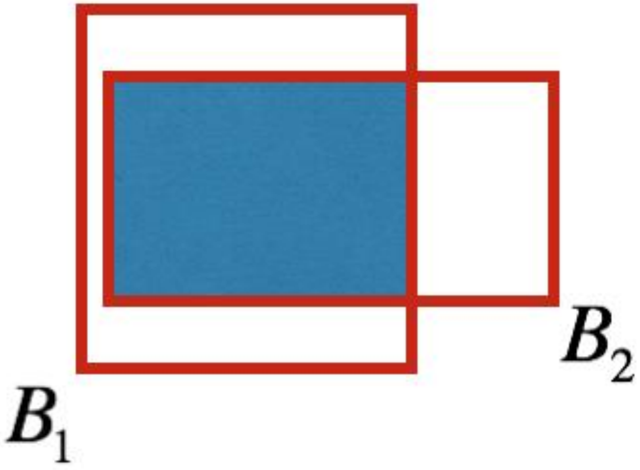
Ground truth
bounding box



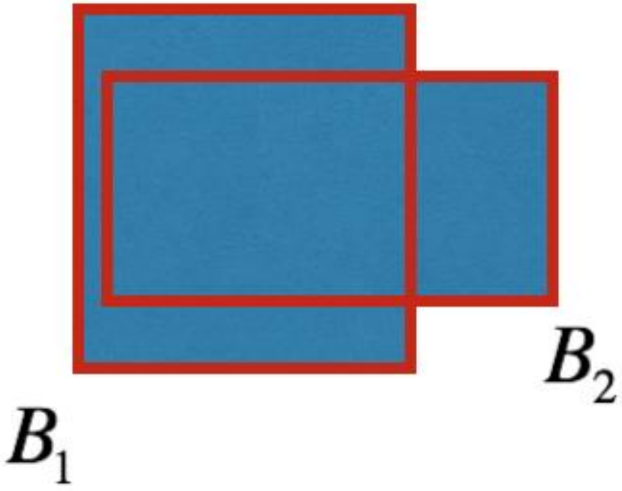
IoU

Formula for IoU

Intersection

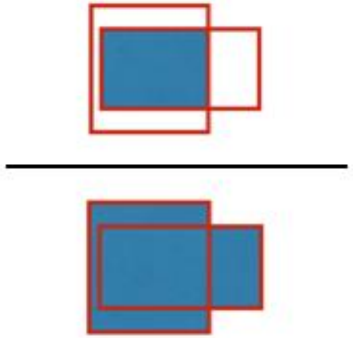


Union



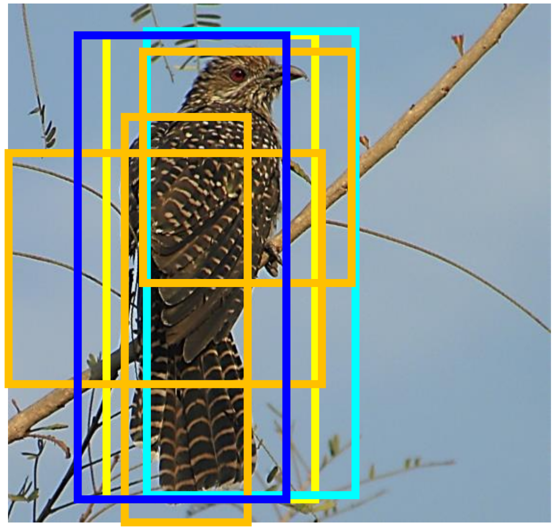
Intersection over Union

$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2} =$$



First Level Filtering Out (Boxes)

Remove all those boxes whose score is less than the threshold



Non Max Suppression

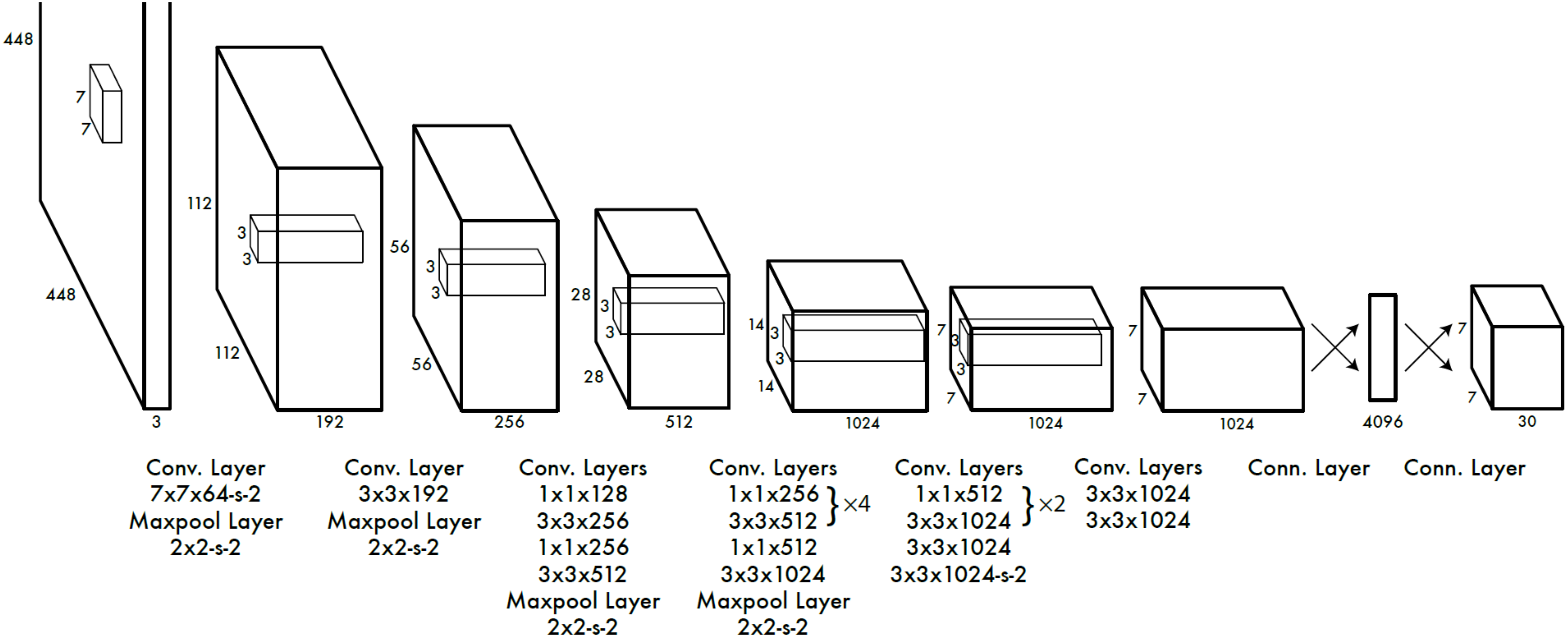
- Second level filter for selecting the right boxes.

1. Select the box that has the highest score.
2. Compute its overlap with all other boxes, and remove boxes that overlap it more than the threshold set for IoU.
3. Go back to step 1 and iterate until there's no more boxes with a lower score than the current selected box.

YOLO CNN Specification

- The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.
- Network architecture inspired by the GoogLeNet model for image classification.
- 24 convolutional layers followed by 2 fully connected layers.
- Instead of the inception modules used by GoogLeNet, 1×1 reduction layers are used followed by 3×3 convolutional layers.

CNN Architecture



Fast YOLO

- A fast version of YOLO designed to push the boundaries of fast object detection.

- Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers.

- Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

Training the Network

- Pretrained the convolutional layers on the ImageNet 1000-class competition dataset.

- For pretraining the first 20 convolutional layers were used followed by an average-pooling layer and a fully connected layer.

- Pretraining stopped when a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set was obtained comparable to the GoogLeNet models in Caffe's Model Zoo.

Training the Network

- After pretraining added four convolutional layers and two fully connected layers with randomly initialized weights.

- Input resolution increased from 224X224 to 448X448 as detection often requires fine-grained visual information.

Loss Function

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

YOLO Algorithm Limitations

- YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class.

- This spatial constraint limits the number of nearby objects that our model can predict.

- First version of YOLO model struggles with small objects that appear in groups, such as flocks of birds.

- It also struggles to generalize to objects in new or unusual aspect ratios or configurations.

YOLO9000: Better, Faster, Stronger

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

YOLO9000 CNN

- 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

YOLO3

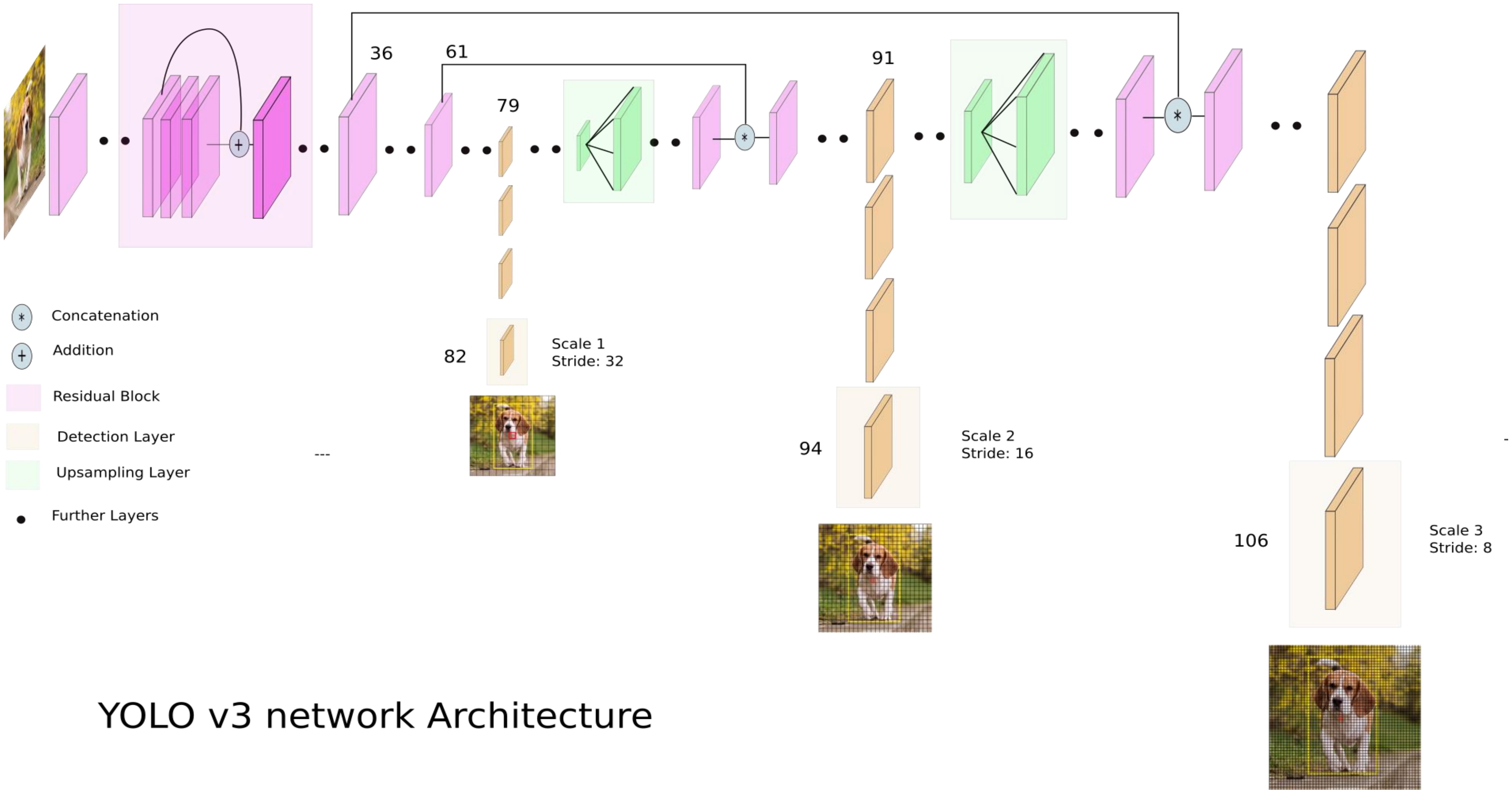


Image source: towardsdatascience.com