Object Detection [1]

Loss function in Yolo: classification loss (CE) *(classification problem)* + localization loss (MSE/ L2_norm) *(Regression problem)* + confidence loss(objectness)(?)

Categorical CE?

Losses are weighted because sometimes localization loss could be larger than confidence loss or vice versa. Hence, we have these losses in weighted combinations using \alpha to make them approximately same order of magnitude so that it focuses equally on getting tight bbox with correct label.

Loss function in SSD:

Loss function in R-CNN:

Loss function in Faster R-CNN:

Two-stage detectors – R-CNN, Faster R-CNN

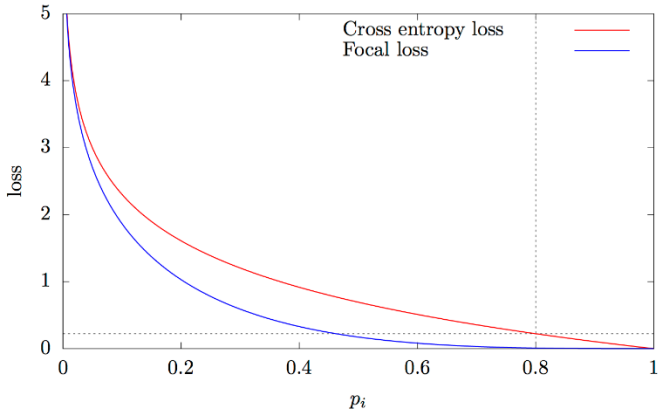
Advantage of two-stage methods?

Problem with two-stage methods? Computationally expensive

Advantage of SSD– no need of region proposals, i.e. speed/faster and less computationally expensive. Detects objects in *one single forward pass*

Problem with Single-shot methods like Yolo or SSD? Suffers from extreme class imbalance. Most popular fix is Focal loss introduced in RetinaNet. Presumably, this makes them as accurate as two-stage detectors (cite references)

Focal loss: Slightly modified version of CE loss. Gives more importance to new/difficult cases and not to already correct and too confident predictions. $-\Sigma y_i(1-p_i)^{\gamma}\ln p_i \,||\, \gamma=2$ used in official paper. As shown in the figure below, when network is too confident, probability>0.7 or so, the loss is very-very small. So, high confidence data points now would not update parameters a lot and the updates would now be driven by the less-confident datapoints.



Feature Pyramid Network (FPN) – Lateral connections? Top Down Pathway? Bottom up Pathway? In Yolo, Faster R-CNN

Difference between SSD and Yolo? Yolo uses fully connected layers instead of convolutional layers at the top of the network (like in SSD shown in Figure-1).

Receptive Field of an activation – Refer Figure-1

Metrics for object detection – mAP – average precision for recall value over 0 to 1.?

NMS (Non-Max Suppression) – technique of selecting one bounding box out of many overlapping bounding box for a single class.
Step1: sort the prediction confidence scores in decreasing order
Step2: start from top score, ignore current prediction if previous prediction has same class and IOU>thresh

Step3: Repeat until all predictions are checked.
Step4: Implement in code?

FLOPS – Floating point operations per second

Use of FLOPS for object detection? – does it measure efficiency of models, I don't know?

Size of predictions for object detection – 4 (xyxy or xyhw) *(regression problem)* + n(+1) (probabs for n classes + background) *(classification problem)*

L2 loss or L1 loss in predicting bbox coordinates – L1 loss could be better. L2 loss is sensitive to outlier. So, model might adjust to minimize single outlier cases at expense of normal examples that have a much smaller error.

Jaccard index or IOU index – basic intersection over union

Jonathan Lui in his medium articles on object detection recommends removing any ***Adaptive pooling*** layer prior to classification layers because they destroy spatial information, we need to regress the coordinates of edges of the bounding boxes.

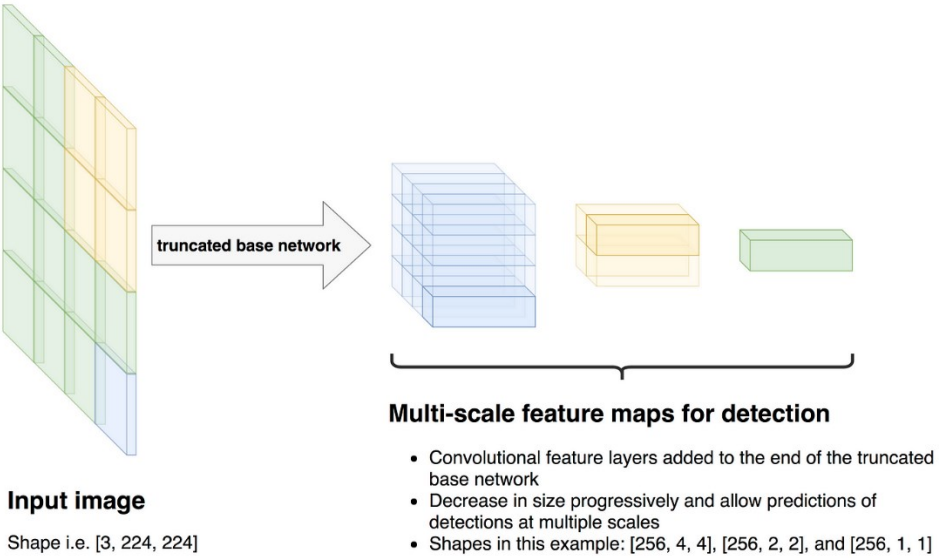


Figure 1Detection across multiple scales in SSD + Figure-3

Anchor boxes – Refer Figure-2 and Figure-1. Dots are anchors and dotted lines are anchor boxes. Colors of dotted lines represent anchor boxes for 3 different layers (Blue, Yellow, Green) which allows for detection across different scales. Why we require anchor boxes? – The idea is that an object that almost fills the entire image should best be detected by activations in the last (green layer) and an object that approximately fills the lower left quarter of the input image should be detected by activations in the yellow layer. Therefore some *default bboxes* aka *anchor boxes* are defined.

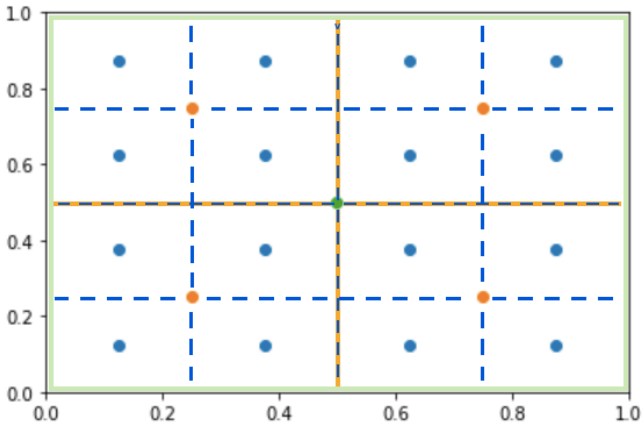


Figure 2 Anchor boxes

Per default box or anchor boxes, we need probability distribution for n+1 classes.

Hence, for blue: 4x4 grids: we want to predict – 4x4(boxes)*4(coordinates)*(n+1) (probability) numbers of values

For Yellow: 2x2 grids: 2x2(boxes)*4(coordinates)*(n+1) (probability) values

For Green: 1x1 grid: 1x1(boxes)*4(coordinates)*(n+1) (probability) values

How do we get these values – we feed each of the 3 feature maps (Blue, Green, Yellow) to 2 more conv layers (Figure-3). For yellow, we have 2x2 grids hence shape of output1 is 4x2x2 where 4 denotes bbox coordinates for 2x2 anchor boxes. And output2 shape is nx2x2 for n probabs for 2x2 grids.

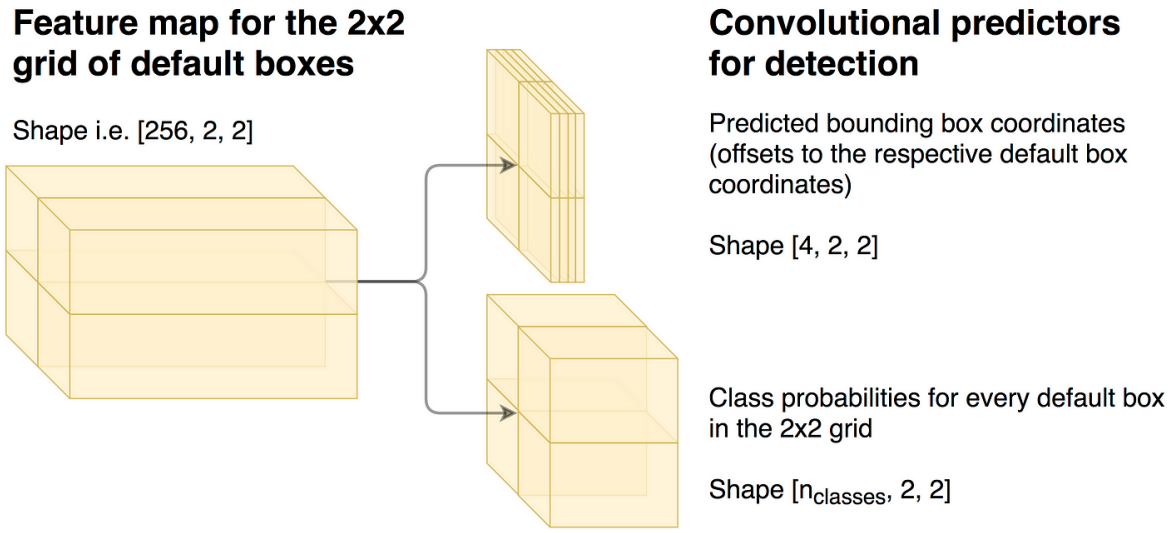
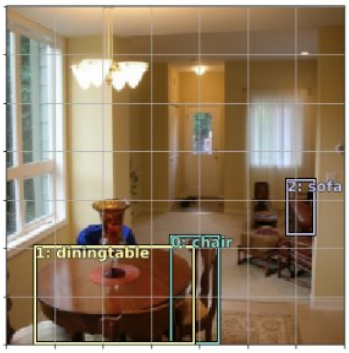
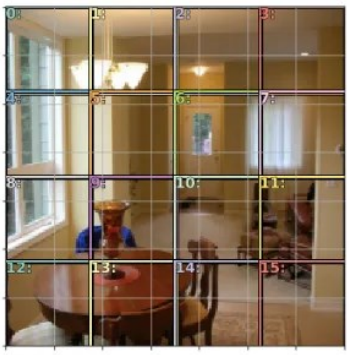


Figure 3 Each of 3 feature maps (B,Y,G) are fed to 2 more conv layers

Similarly, for Blue output1 will be of shape: 4x4x4 and output2 shape: nx4x4

	Q: The network predicts 4 coordinates (offsets) and *n* class probabilities for every of the 16 default boxes. But which of those predictions should we compare to the ground truth default boxes in our loss function while training? Ans: *We need to **match** each of the ground truth bounding boxes in our training example to (at least) one default box.*	

Matching: We want to match each ground truth bounding box to a default box (anchor box) that is 'as similar' to it as possible. Similarity is measured with IOU/ Jaccard index. Calculate IOU of every ground truth in training image with every default box in 4x4, 2x2 and 1x1 grids.

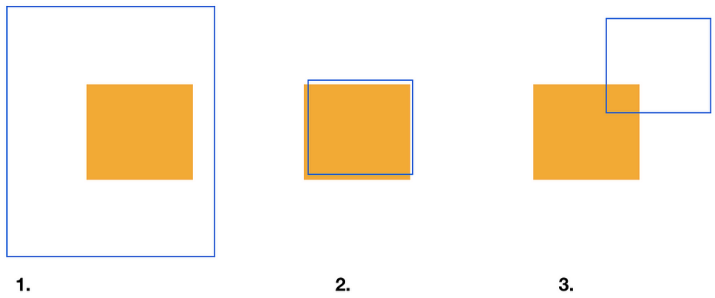


Figure 4Anchor box matching across different feature maps

We calculate the weighted sum of localization loss (L1 loss) and confidence loss (CE) for all anchor boxes and train to minimize it. (In reality for every anchor (dot), we define k anchor boxes of different aspect ratios and sizes)

References:

[1] A.H. Kumar, Interview Questions: Object Detection, Medium (2020). https://pub.towardsai.net/interview-questions-object-detection-9430d7dee763 (accessed June 18, 2024).

[2] L. Alzubaidi, J. Zhang, A.J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M.A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, Journal of Big Data 8 (2021) 53. https://doi.org/10.1186/s40537-021-00444-8.