You Only Look Once:
Unified, Real-Time Object Detection

- Saloni
- Ankit

a) What is YOLO and how is it different from other detectors? [3 marks]

You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm introduced in 2015.Compared to the approach taken by object detection algorithms before YOLO, which repurpose classifiers to perform detection, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once.

In addition to increased accuracy in predictions, better generalization and a better Intersection over Union in bounding boxes (compared to real-time object detectors), YOLO has the inherent advantage of speed.
YOLO is a much faster algorithm than its counterparts, running at as high as 45 FPS. While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then perform recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer and gets away with a single iteration.

b) Explain the following terms briefly: [6 marks]

i) Bounding box

In object detection, we usually use a bounding box to describe the spatial location of an object. We can have as many bounding boxes as there are objects within a given image.
YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

Example:

Y = [pc, bx, by, bh, bw, c1, c2]

- pc corresponds to the probability score of the grid containing an object.
- bx, by are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell.
- bh, bw correspond to the height and the width of the bounding box with respect to the enveloping grid cell.
- c1 and c2 correspond to the two classes Player and Ball. We can have as many classes as your use case requires.

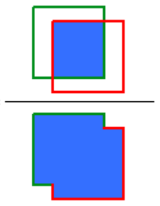
ii) Intersection Over Union

Intersection over Union is a popular metric to measure localization accuracy and calculate localization errors in object detection models.

To calculate the IoU with the predictions and the ground truth, we first take the intersecting area between the bounding boxes for a particular prediction and the ground truth bounding boxes of the same area. Following this, we calculate the total area covered by the two bounding boxes—also known as the Union.

The intersection divided by the Union gives us the ratio of the overlap to the total area, providing a good estimate of how close the bounding box is to the original prediction.

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



iii) Non Max Suppression

Non-maximal suppression is a critical step in detecting objects in an image. NMS is a classical algorithm that analyzes bounding boxes and keeps only the best one. However, it is computationally expensive, making it difficult to accelerate with traditional hardware architectures.

Algorithm:

1. Define a value for Confidence_Threshold, and IOU_Threshold.
2. Sort the bounding boxes in a descending order of confidence.
3. Remove boxes that have a confidence < Confidence_Threshold
4. Loop over all the remaining boxes, starting first with the box that has the highest confidence.
5. Calculate the IOU of the current box, with every remaining box that belongs to the same class.
6. If the IOU of the 2 boxes > IOU_Threshold, remove the box with a lower confidence from our list of boxes.
7. Repeat this operation until we have gone through all the boxes in the list.

This procedure for non max suppression can be modified according to the application.

c) Explain the YOLO loss function. **[5 Marks]**

The YOLO loss function is broken into three parts:

i) the one responsible for finding the bounding-box coordinates,

ii) the bounding-box score prediction, and

iii) the class-score prediction.

All of them are Mean-Squared error losses and are modulated by some scalar meta-parameter or IoU score between the prediction and ground truth:

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

Loss function: sum-squared error

a. Reason: Easy to optimize b. Problem: (1) Does not perfectly align with our goal of maximize average precision. (2) In every image, many grid cells do not contain any object. This pushes the confidence scores of those cells towards 0, often overpowering the gradient from cells that do contain an object. c. Solution: increase loss from bounding box coordinate predictions and decrease the loss from confidence predictions from boxes that don't contain objects. We use two parameters

$$\lambda_{coord} = 5$$

and λ_{noobj} = 0.5 d. Sum-squared error also equally weights errors in large boxes and small boxes

Only one bounding box should be responsible for each object. We assign one predictor to be responsible for predicting an object based on which prediction has the highest current IOU with the ground truth.

- x_i, y_i, which is the location of the centroid of the anchor box
- w_i, h_i, which is the width and height of the anchor box
- C_i, which is the *Objectness*, i.e. confidence score of whether there is an object or not, and
- $p_i(c)$, which is the classification loss.
- We not only need to train the network to detect an object if there is an object in a cell, we also need to punish the network, it if predicts an object in a cell, when there wasn't any. How do we do this? We use a mask ($\mathbb{1}_i^{obj}$ and $\mathbb{1}_i^{noobj}$) for each cell. If originally there was an object $\mathbb{1}_i^{obj}$ is 1 and other *no-object* cells are 0. $\mathbb{1}_i^{noobj}$ is just inverse of $\mathbb{1}_i^{obj}$, where it is 1 if *there was* ***no*** *object* in the cell and 0 if there was.

a. Loss from bound box coordinate (x, y) Note that the loss comes from one bounding box from one grid cell. Even if obj not in grid cell as ground truth.

$$\begin{cases} \lambda_{coord} \sum_{i=0}^{S^2}[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] & \textbf{responsible bounding box} \\ 0 & \textbf{other} \end{cases}$$

b. Loss from width w and height h. Note that the loss comes from one bounding box from one grid cell, even if the object is not in the grid cell as ground truth.

$$\begin{cases} \lambda_{coord} \sum_{i=0}^{S^2}[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] & \textbf{responsible bounding box} \\ 0 & \textbf{other} \end{cases}$$

c. Loss from the confidence in each bound box. Not that the loss comes from one bounding box from one grid cell, even if the object is not in the grid cell as ground truth.

$$\begin{cases} \sum_{i=0}^{S^2}(C_i - \hat{C}_i)^2 & \textbf{obj in grid cell and responsible bounding box} \\ \lambda_{noobj} \sum_{i=0}^{S^2}(C_i - \hat{C}_i)^2 & \textbf{obj not in grid cell and responsible bounding box} \\ 0 & \textbf{other} \end{cases}$$

d. Loss from the class probability of **grid cell**, only when object is in the grid cell as ground truth.

$$\begin{cases} \sum_{i=0}^{S^2} \sum_{c \in classes}(p_i(c) - \hat{p}_i(c))^2 & \textbf{obj in grid cell} \\ 0 & \textbf{other} \end{cases}$$

Loss function only penalizes classification if obj is present in the grid cell. It also penalize bounding box coordinate if that box is responsible for the ground box (highest IOU)