1 Introduction

YOLO is an approach to object detection. YOLO is extremely fast, their base network runs at 45 frames per second with no batch processing on a Titan X GPU. YOLO reasons globally about the image when making predictions while other previous methods used sliding windows and region proposal. YOLO learns generalizable representations of objects. While it can quickly identify objects in images it struggles to precisely localize some objects, especially small ones.

2 YOLOv1

2.1 Method

YOLO divides the input image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.
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. So, they define confidence as $Pr(Object) * IOU_{pred}^{truth}$.
Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. The confidence prediction represents the IOU between the predicted box and any ground truth box. The predictions that have IOU <= threshold are discarded. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression is used to fixed this. The algorithm of NMS is presented in Appendix A.
In summary, YOLO models detection as a regression problem. It divides the image into a $S \times S$ grid and for each cell, it predicts B bounding boxes and confidence scores for those boxes, and C class probabilities. These predictions are encoded as a $S \times S \times (B * 5 + C)$. For evaluating on PASCAL VOC, $S = 7$, $B = 2$, $C = 20$.

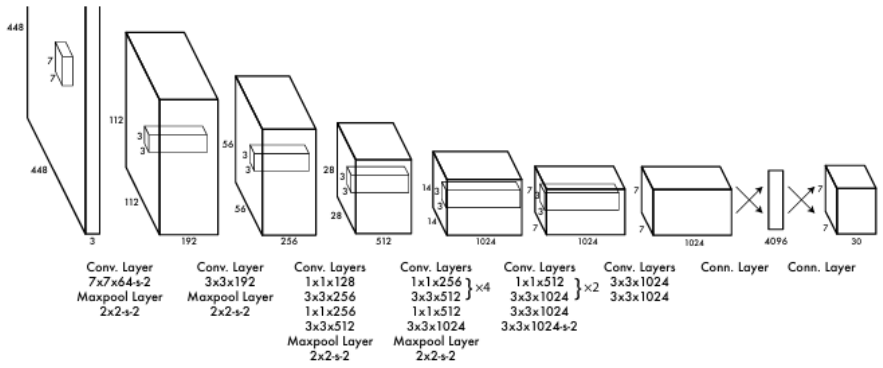

Figure 1: The architecture

2.2 Training

The network architecture is inspired by the GoogLeNet model for image classification. Their network has 24 convolutional layers followed by 2 fully connected layers. The full network in shown in Figure 1. Detection often requires fine-grained visual information so the input resolution of the network is increased to 448×448.

The final layer predicts both class probabilities and bounding box coordinates. They normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. They parameterize 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.

A linear activation function is used for the final layer and all other layers use the following leaky rectified linear activation.

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

The output of network is optimized for sum-squared error. It is easy to optimize, however it does not perfectly align with the goal of maximizing average precision.

It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model insta-

bility, causing training to diverge early on.

To remedy this, they increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. They use two parameters, $\lambda_{coord} = 5$ and $\lambda_{noobj} = .5$ to accomplish this.

Sum-squared error also equally weights errors in large boxes and small boxes. The error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this by normalizing, they predict the square root of the bounding box width and height instead of the width and height directly.

YOLO predicts multiple bounding boxes per grid cell. At training time, only one bounding box predictor is responsible for each object. They assign one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

The multi-part loss function is:

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

where $\mathbb{1}_i^{obj}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i is responsible for that prediction.

2.3 Limitations

- Imposing 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 near by objects that the model can predict. The model struggles with small objects that appear in groups.
- Since the model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. The model also uses relatively coarse features for predicting bounding boxes since the architecture has multiple downsampling layers from the input image.
- While training on a loss function that approximates detection performance, the loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. The main source of error is incorrect localizations.

A Non-maximal suppression

NMS helps object detection model to filter a bunch of bounding boxed that near each other. Input:

- A list of proposed bounding boxes $\mathcal{B}$ with corresponding confidence score $\mathcal{S}$
- Overlap threshold N_t

Output:

- A list of filtered proposals $\mathcal{D}$

Algorithm:

1. Initialize an empty list $\mathcal{D}$ to contain filtered bounding boxes
2. Select the bounding box in $\mathcal{B}$ which has the highest confidence score $\mathcal{M}$, remove it in B and add to $\mathcal{D}$

3. Traverse through each bounding box in $\mathcal{B}$, if there are any bounding box that have IOU with one of box in $\mathcal{D}$ greater than N_t, remove this box

4. Return to step 2 until there aren't any proposals in $\mathcal{B}$

However, a problem with NMS is that it sets the score for neighboring detections to zero. Thus, if an object was actually present in that overlap threshold, it would be missed and this would lead to a drop in average precision.
To tackle this problem, paper[1] lowers the detection scores by Soft-NMS algorithm.

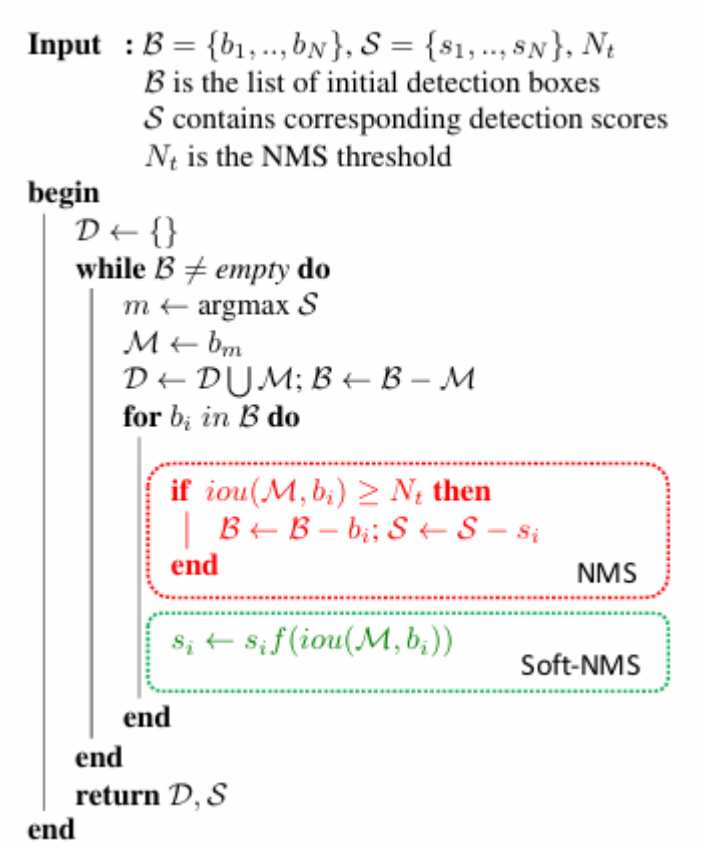

Figure 2: The pseudo-code of NMS and Soft-NMS

Initially, the paper uses $f(iou(\mathcal{M}, b_i)) = (1 - iou(\mathcal{M}, b_i))$ to decay the score

[1]Improving Object Detection With One Line of Code

of detections above threshold. This function makes detection boxes that are far away from $\mathcal{M}$ not affected much and those which are very close would be assigned a greater penalty.

However, it is not continuous because it apply a sudden penalty when the threshold N_t is reached. This could lead to abrupt changes to the ranked list of detections. A continuous penalty function should have no penalty when there is no overlap and very high penalty at a high overlap. Also, when the overlap is low, it should increase the penalty gradually, as $\mathcal{M}$ should not affect the scores of boxes which have a very low overlap with it. However, when overlap of a box b_i with $\mathcal{M}$ becomes close to one, b_i should be significantly penalized. Taking this into consideration, the pruning step is updated with a Gaussian penalty function as follows:

$$s_i = s_i e^{-\frac{iou(\mathcal{M},b_i)^2}{\sigma}}, \forall b_i \notin \mathcal{D}$$