YOLO Object Detection with PyTorch

Nemekhbayar Nomin[1],
[1]Computer Science Engineering Dep. of PNU.

1 Introduction

YOLO (You Only Look Once) is a real-time object detection model that reframes the detection problem as a single regression problem, directly predicting bounding boxes and class probabilities from image pixels [1]. This report documents the process of building and training a YOLO model using PyTorch, analyzing the results, and comparing it with other object detection approaches.

2 Difference Between YOLO and Previous Detectors

YOLO differs from traditional detectors (e.g., R-CNN variants) by using a single neural network to make predictions across the entire image [2]. This approach is faster but may trade off some accuracy. Key differences are summarized below:

- **Single-stage approach:** Unlike R-CNN which uses region proposals, YOLO directly predicts bounding boxes and classes.
- **Speed:** YOLO can process images in real-time, whereas traditional methods are computationally intensive.
- **Trade-offs:** YOLO's design emphasizes speed, which may slightly reduce localization accuracy, particularly for small objects.

3 YOLO Network Architecture

The YOLO model architecture used in this project consists of 22 convolutional layers, 4 max-pooling layers, and 2 fully connected layers. Leaky ReLU activations are applied to all layers except the output layer.

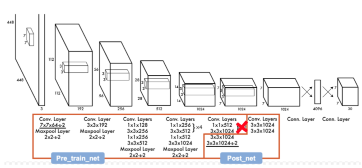

Figure 1: YOLO Network Architecture

4 Dataset Preparation

The Pascal VOC dataset was used, consisting of images labeled with bounding boxes and class labels. Each image is divided into a 7×7 grid, with each grid cell predicting 2 bounding boxes and 20 class probabilities. The `generate_target` function generates output vectors for each image based on bounding boxes.

5 Loss Function

The loss function combines three main components:

- **Localization loss:** Calculated for bounding box coordinates (x, y, w, h), with width and height adapted by square root.
- **Confidence loss:** Computes the confidence score using Intersection over Union (IoU) between predicted and ground-truth boxes.
- **Classification loss:** Applied to class predictions, penalized only when an object is present.

$$\text{Total Loss} = \lambda_{coord} \cdot \text{Localization Loss} + \text{Confidence Loss} + \lambda_{noobj} \cdot \text{No-object Loss} \tag{1}$$

6 Training the Model

The model was trained for 15 epochs using a learning rate of 1×10^{-4}. The optimizer used was Stochastic Gradient Descent with momentum set to 0.9 and weight decay to 0.0005. A pre-trained model with 0.685 mAP was provided as a starting point. During the last epoch of training, the final loss achieved was:

```
Loss: 3.5474, average_loss: 3.3139
```

7 Inference and Visualization

After training, the model was tested on unseen data. The qualitative results showed bounding boxes drawn around detected objects in the test images. Quantitative results were evaluated using Mean Average Precision (mAP).

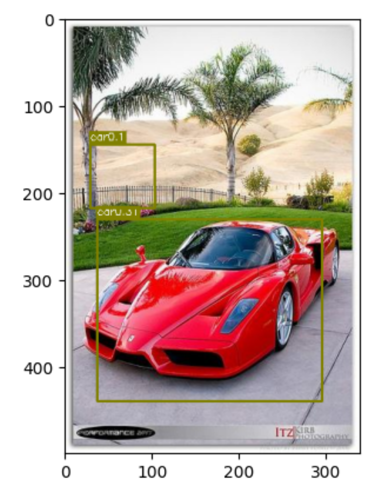

Figure 2: Example inference result showing detected objects

8 Results and Analysis

The model achieved a mAP of 55.11%, indicating reliable performance across multiple object classes. This demonstrates YOLO's ability to deliver real-time detection with good accuracy, although some classes such as "bus" and "car" showed relatively lower AP, likely due to smaller object sizes or occlusions. The gradual improvement in loss values throughout training suggests that further hyperparameter tuning or model adjustments could enhance performance.

9 Conclusion

This project implemented YOLO for object detection, achieving real-time performance with reasonable accuracy. Future work could explore more complex YOLO variants or combine it with other methods to enhance detection of small or overlapping objects.

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[2] Towards Data Science. R-cnn, fast r-cnn, faster r-cnn, yolo - object detection algorithms, 2018. URL https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. Accessed: 2024-10-28.