Vehicle Detection

Samyuktha Venkatesan, SCU ID: 07700006461, mAP Score: 86.94%, Rank: 1

Abstract

Object Detection has numerous applications, and Transfer Learning stands out as a common approach to customize pretrained models for specific tasks through selective training and meticulous hyperparameter adjustment. This study aims to detect three types of vehicles namely cars, medium sized trucks, and large trucks by employing Transfer Learning with models like Faster R-CNN and YOLO. Additionally, hyperparameter optimization is conducted through grid search, leading to the selection of the optimal model. The experiment attains a mean Average Precision (mAP) score of approximately 86.94% on testing data, particularly leveraging the two-stage detector Faster R-CNN.

1. Introduction

Object detection refers to the task of detecting and locating the objects of interest in an image or video. This involves identifying the position and boundaries of object and classifying the object of interest into different groups [1]. From medical image analysis to autonomous driving, the applications of object detection are vast and diverse. Various models are available to tackle these tasks, broadly classified into two categories: single-stage detectors like YOLO, which offer good mAP scores and high inference speed suitable for real-time object detection, and two-stage detectors such as Faster R-CNN, which offer relatively higher mAP scores although at a slower pace.

Transfer Learning is a technique in Deep Learning where a model developed for a task is reused as the starting point for a model on a second task [2]. Large deep learning models require extensive datasets and computational resources. Transfer Learning enables us to repurpose these models without the need to train them from scratch, thereby optimizing resource utilization and accelerating training. The choice of model selected for transfer learning depends on the specific application of object detection. While both single-stage detector and two-stage detectors were implemented to achieve the objective, the latter proved to be more suitable due to its ability to provide higher mAP scores.

2. Approach

The initial phase involved comprehending the dataset and annotating bounding boxes for the training data based on provided labels, while also examining the classes of vehicles. Subsequently, established the foundational models for YOLO and two Faster R-CNN models with different backbones. Implementations were then added to preprocess the input data as required by each model. Upon scheduling the models, a notable difference between Faster R-CNN and the others was observed, prompting the decision to proceed with Faster R-CNN model with ResNet-50 backbone for further hyperparameter tuning. The subsequent phase involved fine-tuning various hyperparameters such as batch size, the number of trainable layers, optimizer, and learning rate.

3. Methodology

3.1. YOLO

The YOLOv5 model was employed, requiring adjustments to the code to process the input labels in their given format rather than conforming to YOLO's specific requirements. Training was conducted for 20 epochs, with all 10 backbone layers frozen and the number of classes set to 3. A learning rate of 0.01 was specified, utilizing pretrained weights from the YOLOv5 small model to initiate training. Upon evaluation, a validation mean Average Precision (mAP) of 61% was observed.

3.2. Faster R-CNN

Several types of Faster R-CNN models were available in torchvision package. Two models were chosen based on the number of parameters, GFLOPs and box mAP scores mentioned. Implemented data loaders, pre-processing, training, validation, and visualizations for this task.

3.2.1. Choosing Models

First, a Faster R-CNN model with a ResNet-50-FPN backbone was chosen for which the number of parameters were 41M, GFLOPS was 4.49 and the box mAP on COCO-val2017 was 37.0. The second model was Faster R-CNN model with a MobileNetV3-Large FPN for which the number of parameters were 19M, with a GFLOP of 0.72 and a lower mAP value of 22.8 on COCO-val2017 [3].

3.2.2. Hyperparameter Values

The model weights were set to "DEFAULT" indicating pretrained model weights on COCO dataset. Initially, started out by setting the default value for trainable backbone layers to 3. Began with a batch size of 4, a learning rate of 0.005 and an SGD optimizer. Both the models were trained for 20 epochs and the training dataset fed to the models were shuffled to improve performance.

3.2.3 Performance Evaluation

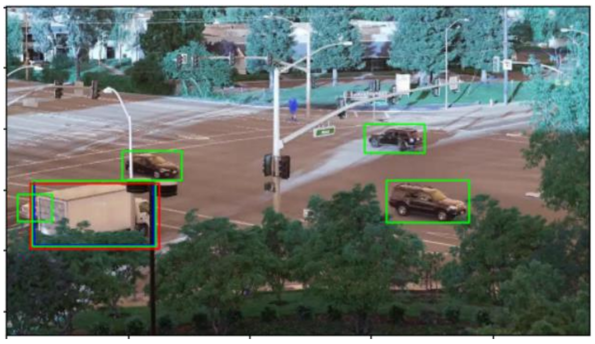
A difference in loss was observed between MobileNetV3 and ResNet-50-FPN. The total loss per image on validation for ResNet-50 was 0.2758, whereas for MobileNetV3 it was around 0.3213. This influenced the decision of choosing ResNet-50 over MobileNetV3. It was also noted that the model started to overfit after epoch 8 for ResNet-50.

3.2.3 Hyperparameter Tuning

Both models were run with a batch size of 16 and a significant difference in validation total loss was observed with a loss of 0.0730 per images for ResNet-50 and 0.1765 per image for MobileNetV3. The definite choice of ResNet-50 was made. ResNet-50 being a heavier model required NVIDIA V100 to run a batch size of 16 and took a training time of about 4.5 hours for 10 epochs. On the other hand, MobileNetV3 took only 2.5 hours for 10 epochs. Used grid search to tune hyperparameters such as learning rate (0.005, 0.0001), batch size (2-16), number of trainable layers (0-3) and optimiser (SGD, Adam). Faster R-CNN with ResNet-50 provided best results with a batch size of 16, learning rate of 0.005, number of trainable layers of 3 and an SGD optimizer.

3.2.4 Visualisation and other improvements

When visualising the data, it was evident that the model was clearly able to detect the objects and provide bounding boxes. But observed that sometimes a single image included several bounding boxes as shown below in Figure 1. To overcome this, tried using Non-Maximum Suppression (NMS). This method chose from overlapping bounding boxes, the one with higher confidence score given the IOU was above a threshold as shown in Figure 2. Despite tuning IOU threshold values, it produced suboptimal mAP scores (71%). Further, tried to experiment with confidence scores as shown in Figure 3, still achieved a lower mAP score (82%). In the below figures, green bounding box indicates cars, blue indicates medium sized trucks and red indicates large trucks.



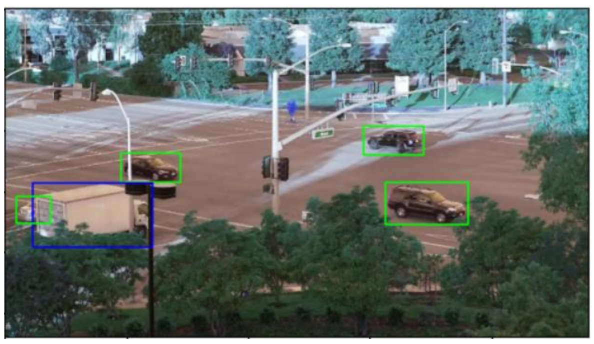


Figure 1: *Test Image 01002.jpeg before NMS*

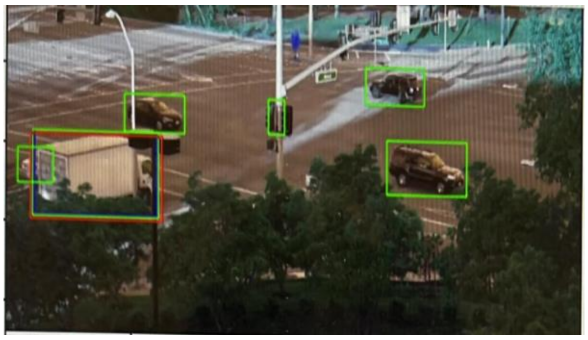


Figure 3: *Without filtering on confidence score*

Figure 2: *Test Image 01002.jpeg after NMS*

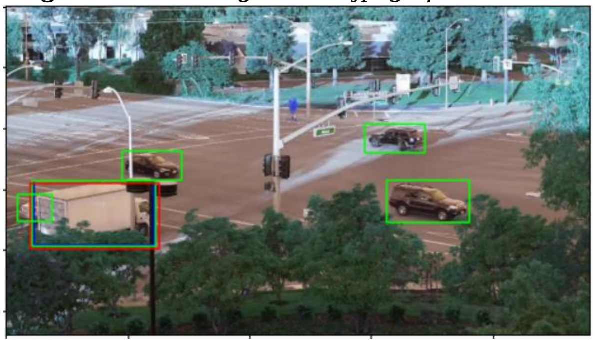


Figure 4: *After filtering on confidence score (0.1)*

4. Results

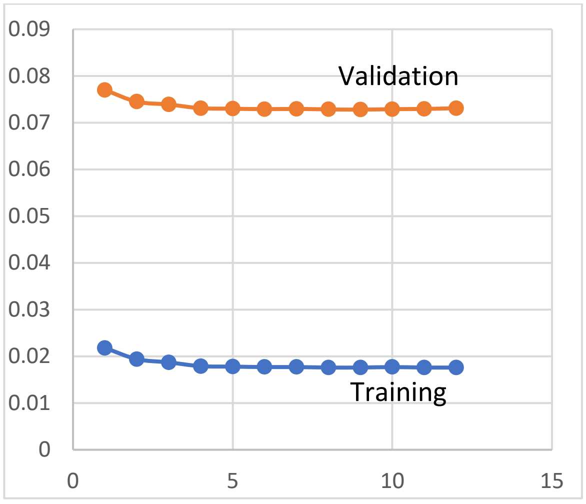
4.1. Tables

The table below shows results of several experiments conducted.

MODEL	Number of frozen layers in backbone	LR	Epoch	Batch Size	Best Results on Epoch	Total Loss per image on validation	mAP on test	Bias/Variance Trade Off	Hardware Requirement	Time Taken to Train
Yolov5	9	0.01	10	16	10	0.285	62.24 (on val)	-	Runs on any NVIDIA GPU. V100 preferred	3h
FasterRCNN MobileNet	2	0.005	30	16	3	0.1765	-	Starts to overfit after epoch 3	Requires NVIDIA GPU V100	4h
FasterRCNN	0	0.005	10	4	5	0.275835	-	Starts to overfit after epoch 5	Runs on any NVIDIA GPU.	4h 30m
FasterRCNN	2	0.005	10	4	7	0.27555	-	Starts to overfit after epoch 7	Runs on any NVIDIA GPU.	4h 30m
FasterRCNN	2	0.0001	10	8	7	0.3028	-	Starts to overfit after epoch 8	Requires NVIDIA GPU V100	4h 30m
FasterRCNN	2	0.005	20	16	7, 9	0.0730	88.17	Starts to overfit after epoch 9	Requires NVIDIA GPU V100	8h
FasterRCNN - loading the model that gave best results and decreased the learning rate	2	0.0001	10	16	3	0.1455	-	Starts to overfit the best model.	Requires NVIDIA GPU V100	4h 30m

4.2 Graphs

Below graph shows sum of all losses (box, classification, object) per image for the best model.



Graph 1*: Faster R-CNN with ResNet50 backbone*

5. Conclusion

Object Detection for different types of vehicles was successfully implemented using both a single-stage detector and a two-stage detector. Observed a higher mAP score for a two-stage detector faster R-CNN with ResNet50 backbone with a testing mAP score of 86.94%.

References

1. https://paperswithcode.com/task/object-detection
2. https://machinelearningmastery.com/transfer-learning-for-deep-learning/
3. https://pytorch.org/vision/main/models/faster_rcnn.html
4. https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html
5. https://chat.openai.com/