Vehicle Detection and Counting in Images

Ryan Furtado
Indiana University
Bloomington
rfurtado@iu.edu

Anurag Hambir
Indiana University
Bloomington
ahambir@iu.edu

FNU Himanshu
Indiana University
Bloomington
hhimansh@iu.edu

Akash Bhapkar
Indiana University
Bloomington
abhapkar@iu.edu

Abstract

In order to alleviate the issue of congested streets and highways, this study suggests real-time counting of vehicles and bikes and providing drivers with alternate routes in advance. The performance of three well-known deep learning models—YOLOv7, RetinaNet, and Faster R-CNN—is compared using transfer learning and fine-tuning on a dataset for detecting vehicles and bicycles. We investigate the efficiency of deep learning-based models for counting and detecting vehicles and give a thorough explanation of their dataset [5], format conversion for the annotations [8], and model training. The merits, shortcomings, and corresponding performance indicators of the three models are covered in the paper's conclusion. Our goal is to shed light on the top model for item detection and counting, as well as the reasons for the differences in their performance.

1. Introduction

This project is inspired by the problem of heavy traffic observed on streets and highways which tends to be a major problem faced by a lot of metropolitan cities these days especially during holiday periods or during peak work hours. We aim to count the number of vehicles on such roads so that we can take preventive measures to control the flow of traffic by detecting the presence of traffic in real time and suggesting alternative routes to drivers beforehand.

To solve this problem, we will use transfer learning and fine tuning on some well-known models with pre-trained weights on our dataset [5] and compare their performance in order to understand their strengths and weaknesses.

2. Background and Related Work

There are several applications for object identification and counting in computer vision, including traffic control, surveillance, and autonomous vehicles. When compared to conventional methods, deep learning techniques have demonstrated a significant improvement in these tasks. Several deep learning-based models for object detection and counting have been put forth in recent years.

Among them, the most well-liked models for object detection and counting in photos are YOLOv7, RetinaNet, and Faster R-CNN. The most recent member of the You Only Look Once (YOLO) family of object detection models is called YOLOv7. RetinaNet addresses the class imbalance issue with a unique focus loss function, and Faster R-CNN enhances detection performance with a region proposal network.

Several studies have looked into the effectiveness of deep learning-based models for vehicle detection and counting in the literature. An extremely overlapping vehicle counting technique based on YOLOv2, was proposed by Guerrero-Gomez-Olmedo et al. and achieves state-of-the-art performance on the TRANCOS data-set [3]. A perspective-free object counting technique based on YOLOv2 was proposed by Onoro-Rubio and Lopez-Sastre, who also demonstrated competitive performance on the NWPU-Crowd data-set [7].

Additionally, Song et al. proposed a deep learning-based vision-based vehicle detection and counting system on the highway scene, which had a 93.9% accuracy rate [10]. A YOLO-based vehicle detection and counting system with a Deep SORT algorithm was proposed by Kejriwal et al. and obtained 96.5% accuracy on a bespoke data-set [6].

In summary, some of the well-liked models for object detection and counting include YOLOv7, RetinaNet, and Faster R-CNN. Our goal with in the course of this research project is to see how well YOLOv7 actually performs when compared against RetinaNet and Faster R-CNN as well as understand the reasons for the differences in their

respective performance metrics.

3. Methods

The focus of our research project is to study and evaluate the performance of different object detection models on our vehicle and bike detection dataset. We chose two single stage object detection models: YOLOv7 and RetinaNet and a two stage detection model : Faster R-CNN and compared their speed, accuracy, and computational power. Please refer to the source code on GitHub [4] In the subsequent topics, we provide a detailed explanation about our dataset, and the three models that we have used.

3.1. Data Exploration and Analysis

We utilized the Traffic Image Dataset [5] available on Kaggle, which comprises nearly 3000 images depicting traffic in Bangladesh. The dataset was annotated with labels and coordinates of different types of vehicles and bikes in txt format. Initially, the dataset [5] had 21 labels representing various types of vehicles, which we converted to two labels - "vehicle" and "bike". It is an imbalanced dataset [5] comprising approximately 2400 vehicles and 300 bikes.

To employ RetinaNet and Faster R-CNN models, we had to convert the annotations in txt format to XML in PASCAL VOC format. This task was accomplished with the assistance of an online format conversion tool in Roboflow [8]. Subsequently, we had to arrange the dataset [5] based on the requirements of each model.

For YOLOv7, we established distinct folders for training, validation, and testing, each containing a separate folder for images and annotations. Whereas for RetinaNet and Faster R-CNN, we created three folders: one for images, one for annotations, and another with three text files for training, validation, and testing. Each text file comprises the paths of the images utilized for training (80%), validation (10%), and testing (10%).

3.2. YOLOv7

While YOLOv5 is the latest official version of the single-stage object detection algorithm YOLO, we wanted to test the accuracy and speed of YOLOv7, which is currently under research and not an official version. We tested YOLOv7 in two modes - one with fixed resolutions for images and another with multiple resolutions. To achieve faster results, we used a batch size of 16 and compared the model's performance with others using 10 and 20 epochs.

YOLOv7 is made up of a neck network, which fuses the features from the input image, a head network, which forecasts the classes and locations of the objects, and a backbone network, which pulls features from the input image. As seen in Figure 1 E-ELAN architecture, which is comparable to ELAN blocks but goes one step further by using group convolution to expand the channels and cardinality of the computing block, forms the backbone of the system. By managing the shortest and longest gradient paths, employing expand, shuffle, and merge cardinality, and using group convolution to increase the number of channels and the cardinality of the computational block, YOLOv7 is made to be an effective network.

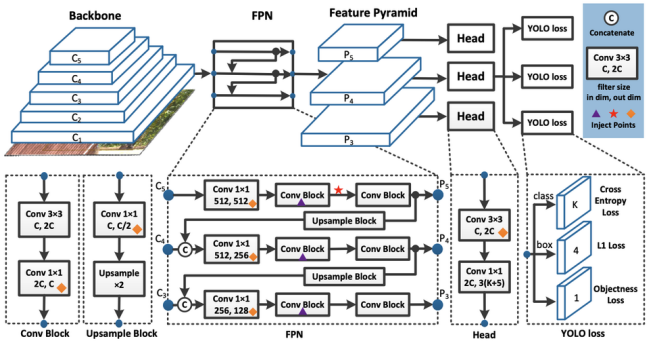

Figure 1. YOLOv7 Architecture [9]

It is made up of a neck network, which fuses the features from the input image, a head network, which forecasts the classes and locations of the objects, and a backbone network, which pulls features from the input image. The E-ELAN architecture, which is comparable to ELAN blocks but goes one step further by using group convolution to expand the channels and cardinality of the computing block, forms the backbone of the system. By managing the shortest and longest gradient paths, employing expand, shuffle, and merge cardinality, and using group convolution to increase the number of channels and the cardinality of the computational block, YOLOv7 is made to be an effective network.

3.2.1 Fixed Resolution

For this portion of the experiment, the image size is set at 640x640, and object detection is then trained. We trained YOLOv7 in this default configuration for 10 and 20 epochs with a batch size of 16. The multiple scaling of images for better detection is not considered in this.

3.2.2 Multi-Resolution

We must choose a base resolution, such as 640x640, to employ multi-resolution training. During training, the photos will then be adjusted to be between 320x320 and 1280x1280 pixels, or within 50% of this base resolution. In

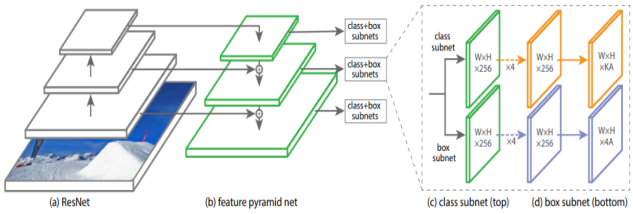

Figure 2. RetinaNet Architecture [2]

especially for smaller objects in the dataset, this strategy can increase the resilience of the model. However, given the increased difficulty of the dataset brought on by the different image sizes, training time might need to be extended.

3.3. RetinaNet

RetinaNet is a single stage object detection model and comprises three crucial components. As can be seen in Figure 2, these components include a backbone CNN, which in our case is Resnet50, a feature pyramid network (FPN), and two subnets, one for classification of the images and second one that enables accurate placement of bounding boxes for each detected object.

The backbone network is responsible for calculating the feature maps at varying scales, regardless of the input image size or backbone. The FPN works by upsampling the spatially coarser feature maps from higher pyramid levels, and the lateral connections combine the top-down and bottom-up layers with the same spatial size. The classification subnet predicts the probability of object presence for each anchor box and object class at each spatial location. On the other hand, the regression subnet is utilized for regressing the offset of the bounding boxes from the anchor boxes for each ground-truth object.

To tackle the issue of class imbalance in single-stage object detection models, RetinaNet uses the focal loss function, which is an improvement over Cross-Entropy Loss (CE). Dense sampling of anchor boxes (possible object locations) in single-stage models results in an extreme foreground-background class imbalance. In RetinaNet, thousands of anchor boxes can be present at each pyramid layer, with only a few assigned to a ground-truth object, resulting in a vast majority of boxes belonging to the background class. Easy examples, such as detections with high probabilities, can collectively overwhelm the model, despite their small loss values. Focal Loss addresses this problem by decreasing the loss contribution from easy examples and increasing the importance of correcting misclassified examples. Our dataset, a highly imbalanced one is a classic example to test the performance of RetinaNet

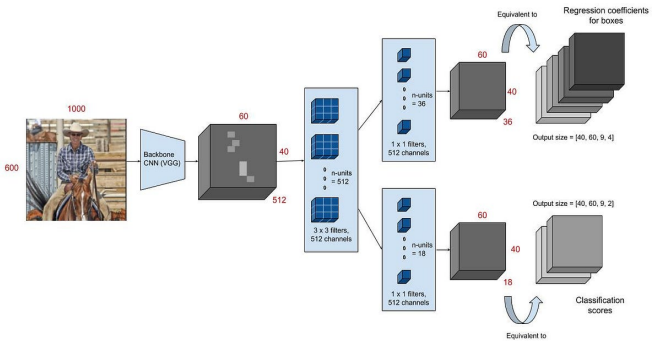

Figure 3. Faster R-CNN Architecture [1]

model.

3.4. Faster R-CNN

For our research purpose and given the hardware limitations, we chose to proceed with Faster R-CNN. Regular R-CNN makes use of a selective search algorithm that scans each image and selects 2000 proposal regions. These regions are usually parts of the image where it's highly likely for an image to exist. Each of these proposals are then snapped to a 1:1 square width and height before being passed through a Convolutional Neural Network (CNN). The CNN acts a feature extractor, and the extracted features are then used by the SVM to classify the proposal region. This process is inherently slow since each time an image is encountered, we select 2000 proposal regions and then each of those regions are passed into the CNN.

Faster R-CNN as seen in Figure 3 instead passes the whole image through the CNN first to generate feature maps after which we do region proposal which in turn generates anchors. These anchors are then sent to the classifier. Since the image is sent through the CNN only once instead of 2000 different regions of the image, it runs a lot faster than R-CNN.

Another reason to use Faster R-CNN for comparison against RetinaNet and YOLOv7 in terms of training performance and detection accuracy, was that R-CNN is a 2 stage object detection model and we wanted to see if it's architecture produced better results.

Since training the model itself takes a lot of time, we used a Faster R-CNN model which had been pretrained on the COCO data-set and fine-tuned the model using our

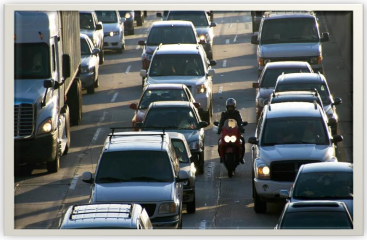

Figure 4. Original Input Image

Traffic Image Dataset from Kaggle which made the process relatively faster. The annotations were supplied to the model using the Pascal VOC format which involves parsing the annotations as xml files for each image.

4. Results

The experiment was run on a Google Collab CUDA processor with 12 GB RAM and a Tesla K80 GPU. Bangladesh's traffic photos are included in the data set. Training data set has 2700 images and testing data contains 300 images. This data set has a bias in favor of vehicles with a vehicle to bike ratio of 10:1. All of the parameters are set to the same values for a better comparison between the four models, with the learning rate set to 0.001, the batch size to 1, and the step size to 500. The IOU used as a benchmark for the models was 0.5. We carefully analyzed a variety of criteria, including the cost of computing and the amount of training time, before settling on 20 epochs for the experiment. To compare the accuracy of each model, we used Mean Average Precision (mAP). Table 1 compares mAP and demonstrates which model has better accuracy after 20 epochs.

Table 1. Mean Average Precision (mAP) @ 0.5

Number Of Epochs	10	20
YOLOv7 (Fixed Resolution)	0.65	0.72
YOLOv7 (Multi-Resolution)	0.69	0.76
Retina Net	0.37	0.38
Faster R-CNN	0.51	0.55

The YOLOV7 Multi-Resolution provides a far better outcome than other models, as shown in the aforementioned Table 1.

We used Figure 4 as input image for showing result of each model. Figure 5 shows output of YoloV7 Fixed Resolution model where as Figure 6 shows output of YoloV7 Multi-Resolution model. Figure 7 displays output for Retinanet model. Figure 8 displays output for Faster R-CNN model.

Figure 5. YoloV7 Fixed Resolution Result

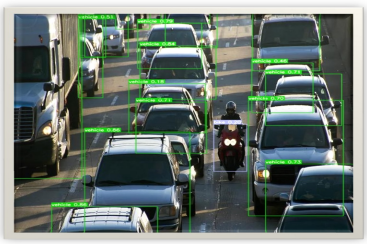

Figure 6. YoloV7 Multi Resolution Result

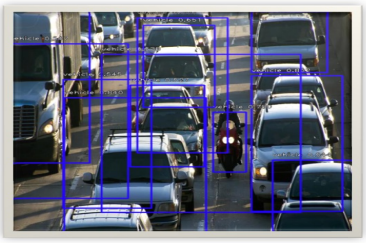

Figure 7. Retinanet Result

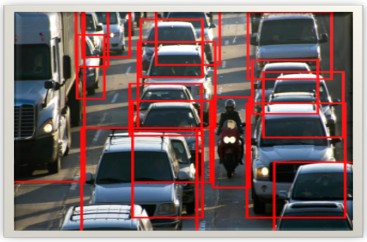

Figure 8. Faster R-CNN Result

5. Discussion

The objective of our project was to contrast the effectiveness and precision of one-stage and two-stage object identification models. For this, we chose YOLOv7 and RetinaNet as the single-stage models, and the popular Faster RCNN as the two-stage model. We used the Bangladesh Traffic Dataset [5] to train the models, and then we went one step further and tested the output on photos from several nations to make sure the findings

weren't skewed.

The mAP (Mean Average Precision) metric, which evaluates the precision of the correctly predicted objects and divides it by the number of labels—in our case, two (vehicles and bikes)—was used to assess the performance of the models. Surprisingly, we discovered that YOLOv7 with multi-scale resolutions offered the most precise findings. This is due to the photos' upscaling and downscaling, which enables more accurate object detection.

Furthermore, the Spatial Pyramid Pooling implementation in YOLOv7 creates a fixed-length representation of the feature map, independent of the input image size, making the network invariant to object size and scale. This implies that items of various sizes can be easily detected. Additionally, YOLOv7 makes use of focal loss to increase the precision of detecting smaller objects. Overall, our results indicate that YOLOv7 beats the well-known Faster RCNN two-stage model as a reliable and effective one-stage object identification model.

6. Conclusion

In our project, we put the power of transfer learning to the test by putting three different models to the test. Due to its effective backbone architecture, use of spatial pyramid pooling, incorporation of focus loss, and quicker outputs, we discovered that YOLOv7 with multi-scale resolutions outperformed both RetinaNet and Faster RCNN. This model was the best fit for our project because of its versatility, accuracy in detecting small objects, and capacity to provide exact feature maps.

References

[1] Shilpa Ananth. Faster r-cnn for object detection. https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46, 2019. 3

[2] ArcGIS Developers. How retinanet works? https://developers.arcgis.com/python/guide/how-retinanet-works/. 3

[3] Ricardo Guerrero-Gomez-Olmedo, Beatriz Torre-Jimenez, Roberto Lopez-Sastre, Saturnino Maldonado-Bascon, and Daniel Onoro-Rubio. Extremely overlapping vehicle counting. https://gram.web.uah.es/data/publications/ibpria2015-guerrero.pdf, 2015. GRAM, University of Alcala, Spain. 1

[4] Anurag Hambir, Ryan Furtado, FNU Himanshu, and Akash Bhapkar. Project source code. https://github.iu.edu/hhimansh/cs-b657-final-project. 2

[5] Kaggle. Road vehicle images dataset. https://www.kaggle.com/datasets/ashfakyeafi/road-vehicle-images-dataset, 2021. 1, 2, 4

[6] Rahul Kejriwal, Ritika H J, Arpit Arora, and Mohana. Vehicle detection and counting using deep learning basedyolo and deep sort algorithm for urban traffic management system. https://ieeexplore.ieee.org/document/9768653, 2022. 1

[7] Daniel Onoro-Rubio and Roberto J. Lopez-Sastre. Towards perspective-free object counting with deep learning. https://gram.web.uah.es/data/publications/eccv2016-onoro.pdf, 2016. GRAM, University of Alcala, Alcala de Henares, Spain. 1

[8] Roboflow. Annotation conversion to different formats. https://roboflow.com/formats, 2022. 1, 2

[9] Jacob Solawetz. What is yolov7? a complete guide. https://blog.roboflow.com/yolov7-breakdown/, 2022. 2

[10] Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai, and Xu Yun. Vision-based vehicle detection and counting system using deep learning in highway scenes. https://etrr.springeropen.com/articles/10.1186/s12544-019-0390-4, 2019. 1