

To Monitor Traffic By Vehicle Identification and Classification Using UA-DETRAC Dataset

Aniruddha Kulkarni

Dept of computing
Dublin City University
Dublin, Ireland

aniruddha.kulkarni4@mail.dcu.ie

Jasmeet Singh Narula

Dept of computing
Dublin City University
Dublin, Ireland

jasmeet.narula2@mail.dcu.ie

Abstract—This paper demonstrates the importance of traffic motoring using latest identification and segmentation models. We intend to compare the results obtained by identifying objects using state of the art one stage object detection systems such as YOLO V4 and latest iteration which is in beta phase, YOLO V5. Extensive Experiments were performed on the UA-DETRAC benchmark dataset in various ways such as using a model pre-trained on a recognised COCO dataset as well as creating our custom dataset by manually annotating each frame to train our dataset and later test it to show the accuracy and superiority of the methods. It was found that Yolo V4 using DarkNet was the stable version with results giving high accuracy and performance. The results obtained from the classified classes are then used to gain the count of vehicle movement which would help in identifying traffic situation in different parts of the city.

Index Terms—Convolutional Neural Network, YOLO, Image Detection, Deep Learning, Classification.

I. INTRODUCTION

Transportation in any city plays an important part in the smooth functioning of day today life of an individual living or travelling in that city. As per the central statistics office of Ireland, In the year 2018, A licensed vehicle travelled about 47 billion kilometres. On an average that's 17,422 kms travelled by each vehicle where private cars account to 75% of the total travelled distance. [1] Due to the large number of registered vehicles including the commercial as well as private vehicles on the road, it is bound to affect the traffic conditions of a particular city. Hence It has become an important task for the road safety authority of each country to maintain order by monitoring traffic at every moment.

Technology has come a long way from assisting to manually observe and identify objects. With the idea of smart city coming up at a rapid speed, cameras have been put up at every signal in major cities to observe the traffic flow, any anomalies as well as assist in case of vehicle mishap. While these observations were being done manually, Vehicle detection, identification, and tracking has become an important part of any system to maintain order. With the introduction of Deep neural networks, it has emerged and is known to be the foundation of new era of machine learning models altogether. Its main advantage was the deep architecture which had the capacity to handle difficult models. This further increased the capacity of the learning model to train on a particular set

of data and later to execute the results. With the increase of usage of deep learning models, we now have Even more efficient, refined models which assist with object identification and classification with the introduction of Regions with CNN features (R-CNN) [2]. With the introduction of R-CNN, many improvements have been suggested, including the Fast R-CNN [3] and later, Faster R-CNN [4] where the CNN and its Bounding box regressor was combined into a single architecture giving the result in terms of a feature map. Faster R-CNN is ground breaking at the time but if we talk about speed, the factor of accuracy and real time performance plays an important role as well. This was looked into and taken care by You only look once, J Redmon et al. [5].

In this paper, we tend to compare different implementation and versions of a fast, reliable and an efficient object identification and classification model, You Only Look Once (YOLO). The paper comprises of 1. Dataset Review and Processing, 2. Vehicle detection. 3. Classification of the vehicles while training the model and 4. Test the model on the dataset.

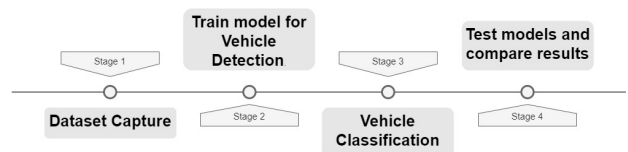


Fig. 1. Project Flow

A. Data Capture

To implement the project, the most important task the data collection on which the model would be trained and later tested on. Hence having the correct dataset is essential. For this the dataset depends on the task that needs to be achieved. Hence after reviewing different types of dataset, such as random images, videos, and also textual data, we came across a data set which would assist in achieving the goal of traffic monitoring.

B. Vehicle Detection

To prepare the task of detection, various image detection models are present to compute the required task. We review different models such as R-CNN, Faster R-CNN and YOLO. On experimenting with different models we understand the

model to be used for our further experiments and prepare the dataset accordingly. To train the model the important requirement is the annotation of the dataset would help in teaching the model the different object classes.

C. Vehicle Classification

Once the different object classes have been identified, we manually annotate multiple frames from the dataset which have different conditions affecting the quality of identifying the object. Once the dataset is ready, we train the deep learning model to understand the object which have been identified manually.

D. Model testing

Once the model is trained, the test data would be used to test the efficiency of the model. This would give us the results of different vehicles being identified for further monitoring of the traffic.

II. LITERATURE REVIEW

A desirable smart traffic-monitoring and street-safety system can support the intervention of law enforcement agencies. In recent years there has been dramatic high demand for smart systems. UA-Detrac 2018 paper look towards evaluation of different detection and tracking algorithms that are available for use. Here, 5 different detection and 7 tracking methods were evaluated, and the results were made public. [6] Object detection is identifying the object in a video or image whereas tracking is to lock on object in real time. The object detection method that were used here are R-CNN, IMIVD-TF, YOLOv3, MYOLO.

Most widely used object detection method is R-CNN which was developed by Ross Girshick. Based on the traditional R-CNN, R Girshick further proposed Fast R-CNN [3]. It is built on previous R-CNN to classify object proposals using convolutional neural networks. Fast R-CNN improved the training and testing speed of the model which results in improved accuracy. According to R. Girshick. Fast R-CNN [3], Fast R-CNN trains the deep VGG16 network 9x times faster than R-CNN. The flexibility of using Python and C++ makes Fast R-CNN comparatively easy to implement. Here, for our research we will be using Fast R-CNN using two different evaluation matrixes: IOU and mAP.

Further processing with the image, there were also question raised such as how density of traffic can be calculated using image processing. To answer this problem Freddy Kurniawan et al. [7] came up with a solution. Here, they used a technique where they estimate the traffic density and background construction. Here, traffic density can be estimated by calculating road area covered by vehicle and total road area. The experiments were carried out at an intersection controlled by a conventional traffic light and many types of vehicles running on it. The traffic flow was recorded by a frame rate of 30 frames per second. The video size used in the experiment is 640×480 pixels. The data were extracted into a sequence

frame and the selected images were processed by a MATLAB commercial software

Vehicle detection is a challenging problem due to structural and appearance variations. Multi-Task Vehicle Detection with ROI voting [8] suggest an object detection method which is based on multi task deep CNN and ROI Voting. Here, in this technique we enrich the information with bounding box regression, region overlap, and then category of each training ROI as multi task framework. This makes the object detection using CNN more robust. Here the results from each ROI is compared with the ground truth. This experiment results on real world machine vision benchmark KITTI and PASCAL2007 dataset.

Vehicle and traffic monitoring are only a small implementation of a large idea of using artificial intelligence in the real world. City brain, a practice of large-scale artificial intelligence in the real world [9] brings the concept collecting large amount of city data in terms of mainly videos and training the system for decision making. The DAMO academy plans to implement an end to end system from cognition to optimisation so that the data collected by the system can be used by different departments such as traffic, police, health and safety, etc. to implement such a system, RV-CNN multi task framework is used where Roi (Region of interest) pooling layer is proposed to be used to extract features from each Roi. This pool of ROI will be used for classification, identification and overlap prediction.

Convolutional Neural Networks (CNN) based method have significantly improved the performances of object detection task in recent years [10]. However due to different ranging sizes of the images captured has led to a decline in the performance of detection. Previewer for Multi-Scale Object Detector [10] discusses the option of adding a previewer block for identification of a possible regression region using the stronger features with larger receptive fields and more contextual information for better predictions [10]. This method can easily be implemented along with faster R-CNN which introduced the concept of prior box into CNN object detection algorithm making annotation and classification less complex.

While Faster R-CNN was being used and proved the speed being the important factor in object detection, A new model was introduced which also paid equal importance to accuracy and real time monitoring of objects. You Only Look Once (YOLO) was introduced by author J Redmon et al. [5] In the Paper, The author describes the importance of an extremely fast unified architecture where the model can be used to process data which arrives at a speed of 45 frames every second. This would mean that the model can not only process image data but also the data in video form. In the experiments performed, the model was compared to many claimed to be best in class detection models and YOLO proved to outperform each and every model as even though it does make localization errors but it less likely predicts false positives on the background of the frames. Hence YOLO is an ideal application when it comes to fast reliable object detection.

To have a stable and accurate image count, the differ-

ent sizes of vehicles and their detection always remains a challenge. Hence to address this issue, Author Huansheng Song et al. [11] propose a vision-based vehicle detection and counting system. For the same, the study required a large number of high definition data images which were annotated which provided the root to the data foundation helping in deep learning. The network structure adopted by the machine learning model YOLO V3 was Darknet -53. This helps in directly connecting the CNN to the existing structure and later connecting the input to deep layer of network. Along with identification, the authors found out the trajectory of the vehicle to find the direction of the moving object.

We also observe that every neural network model requires a lot of computational knowledge and resources. This could be a problem if the computation takes place on a simple device. Hence Chien et al. [12] suggest the introduction of Cross Stage Partial Network (CSPNet) to mitigate such problems. The paper proposes introducing feature maps from start till the end of network which would reduce the computation significantly with increase in the accuracy.

While we perform computation during training, the issues observed is the loss of important informative pixels which is mistook for random patch of black pixel or noise. This leads to information loss and inefficiency. Hence author Sangdoo et al. [13] proposed CutMix where the patches cut and pasted among the training images where the ground truth annotations are also mixed in the area of patches. It was proved that CutMix improves the model's robustness against errors and improves the detection performance.

III. DATASET AND EVALUATION MATRIX

We needed a dataset which would help us perform tests where we would be able to identify vehicles and classify the identified vehicles accurately. Hence the dataset that would suit best for our experiments on vehicle identification and traffic monitoring came down to using the large-scale University at Albany DETection and TRACking (UA-DETRAC) dataset. [6]



Fig. 2. Frame from the UA-DETRAC Dataset

The UA-DETRAC dataset includes 100 challenging videos with more than 140, 000 frames of real-world traffic scenes. The dataset consists of 10 hours of videos captured with a handheld camera at 24 different locations. The videos are recorded at 25 frames per seconds (fps), with resolution of 960x540 pixels. There are more than 140 thousand frames in

the UA-DETRAC dataset. The dataset consist of Training set (DETRAC-Train) which has 83,761 frames which have been taken from the train videos and Test dataset which consist of 56,340 frames. The images are in JPEG format 24 bit color. The dataset comes with a large variation in terms of the conditions in the images such as day, night, sunny, clustered. This will help in training the model based on the complexity of the images when detection takes places in different conditions.

The evaluation matrix that would be used is Average precision (AP). Average Precision is a known metric to measure the accuracy when image identification and classification is required. Every object detection models such as R-CNN, YOLO, etc. calculates the average precision to be between 0 to 1. To get to our average precision, we calculate the precision of the model which shows the percentage of correct prediction of identification.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

... Eq. 1

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

... Eq. 2

Recall lets us know how well did the model perform to give us the positives. The detection will be evaluated on the basis of the precision vs recall curve. To get the accuracy, Intersection over union (IoU) threshold is fixed which is shows the amount of predicted boundary intersects with the real boundary which is the ground truth. the larger AP score indicates the better performance of object detection algorithm.

$$\int_0^1 p(r)dx$$

... Eq. 3

We use the mean average precision (mAP) score of the precision vs. recall (PR) curve to indicate the performance of vehicle detection methods, where the hit/miss threshold of the overlap between a detected bounding box and a ground truth bounding box set to 0.7.

IV. METHODOLOGY

A. Faster R-CNN

The introduction of Region Proposal Network (RPN) by authors Shaoqing Ren [4] was game changing at the time in terms of the execution time compared to previous models such as R-CNN [2] and Fast R-CNN [3]. The basic architecture of Faster R-CNN is base on Region Proposal Network (RPN) for

generation of the proposals and a network that uses these to assist in the detection of the object. RPN is used to identify the rank boxes that are also known as Anchors to identify the objects.

1) **Anchors:** Anchors are rank boxes which help with identifying the objects. Usually all the objects that need to be identified lie in the Anchor boxes. With Faster R-CNN, there are 9 anchors divided in every image. For example, for an image which is of size 600 x 800 would have 9 anchors at position 320 x 320. There are 3 different anchors boxes creator of sizes 128 x 128, 256 x 256 and 512 x 512. The boxes are in ratios 1:1, 1:2 and 2:1 respectively [14].

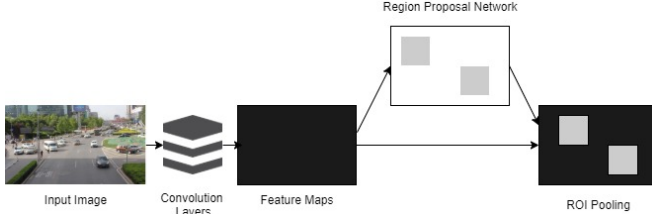


Fig. 3. Faster R-CNN Architecture

Region proposal Network is a group of boxes and proposals used by classifier to validate the number of times the object appears in that image. The depth of a feature map is 32. The overall loss of the RPN is a combination of the classification loss and regression loss as mentioned below.

$$L_{loc}(t^u, v) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

... Eq. 4

In Which,

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

... Eq. 5

B. YOLO v4 Using DarkNet

The basic aim is a fast and parallel computations for object detection. The proposed architecture is basically divided into head, Backbone and Neck. YOLO is known for giving good performance on CPU as well, but its faster on GPU. [15] The main objective of YOLO v4 is to find an optimal balance between convolutional layer number, parameter number and input network resolution.

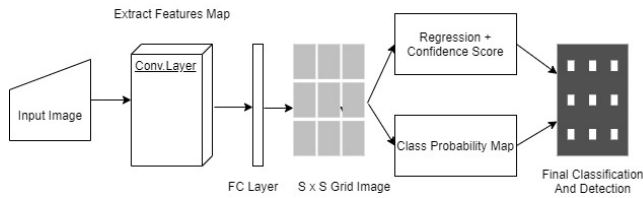


Fig. 4. YOLO Architecture

TABLE I
PARAMETERS OF NEURAL NETWORKS FOR IMAGE CLASSIFICATION

Backbone model	Input Network resolution	Receptive field size	Parameters
CSPResNext50	512x512	425x425	20.6M
CSPDarknet53	512x512	725x725	27.6M
EfficientNet-B3	512x512	1311x1311	12.0M

We have implemented YOLO v4, which has been pre-trained on COCO dataset. This model was designed based on following analysis. Below image is a comparison between different models of YOLO where version 4 outperformed other models.

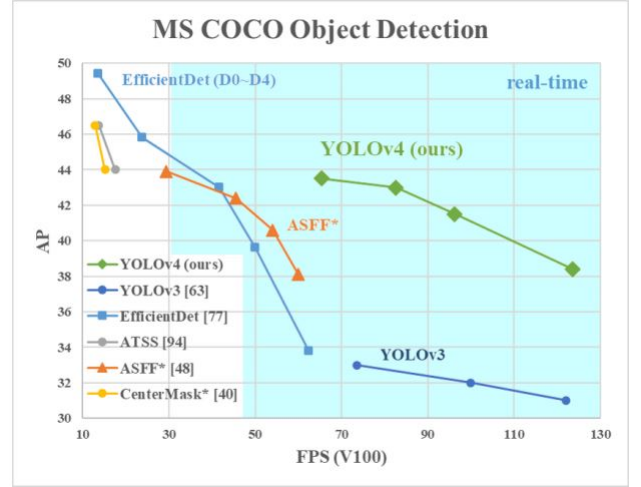


Fig. 5. Coco DataSet [15]

TABLE II
PARAMETERS OF NEURAL NETWORKS FOR IMAGE CLASSIFICATION PART (B)

Backbone model	Avg. size of layer output	BFLOPS	FPS
CSPResNext50	1058K	31	62
CSPDarknet53	950K	52	66
EfficientNet-B3	668K	11	26

Based on above results we assume that the model high parameter number and large receptive field size should be selected for backbone. Table 1 shows that CSPDarknet53 contains 29 convolutional layers [12], a 425 x 425 receptive field and 27.6 M Parameters. This theory justifies the selection of Darknet for this analysis. Author has also added SPP block over Darknet. SPP basically is used to separate significant features [16], increase the receptive field and have no effect on speed of network operations. For Neck, PANnet has been used as a method of parameter aggregation. PANet is a instance segmentation framework which was designed by Shu Liu et al. [17]

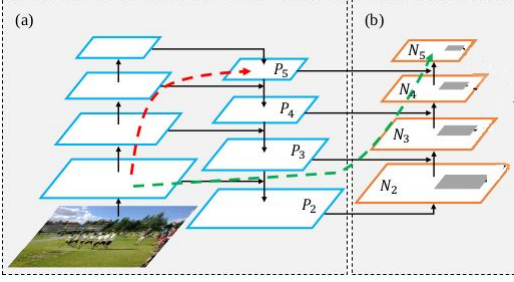


Fig. 6. PANet: (a) FPN Backbone; (b) Bottom-Up Path Augmentation [17]

PANet improve the entire Hierarchy of Features Precise bottom-up localisation signals in lower layers Path augmentation that shortens the path of information between the lower layers and the uppermost feature. In Fig. 1, we see that FPN uses top-down path where the path through which spatial information is propagated, is quite long. PANet on other hand shortens this path by direct links to top layer. Finally the architecture used for YOLOV4 comprised of SPP block over Darknet as backbone, PANet Path aggregation as Neck and YOLO v3 as head.

For increasing our accuracy of object detection, CNN usually uses following:

Activation: ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish

Bounding Box Regression Loss: MSE, IoU, GIoU, CIoU, DIoU

Data Augmentation: CutOut, MixUp, CutMix

Regularization Method: DropOut, DropPath, Spatial DropOut, or DropBlock [18]

1) **GIoU Loss:** A loss function acts as a signal for us to make adjustments in weights. Using IOU has also its shortcomings. Consider two predictions which do not overlap with ground truth. This makes it difficult for IOU function to tell which one is better.

IOU is usually calculated as

$$IOU = \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|}$$

... Eq. 6

$$\iota_{IOU} = 1 - \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|}$$

... Eq. 7

This can be fixed by using Generalised IOU which refines the loss as below:

$$\iota_{GIoU} = 1 - IOU + \frac{|C - B \cup B^{gt}|}{|C|}$$

... Eq. 8

2) **CIoU Loss:** Since GIoU had limitations, where it expanded bounding box to meet ground truths and later shrank to improve the accuracy of IoU. To help this Complete IoU was introduced [19]. It has following feature:

- It minimize the distance of centre point.
- It increase the overlapping area of ground truth box and predicted box.
- It maintains aspect ratio of the box.

$$R_{CIoU} = \frac{\rho^2(b, B^{gt})}{C^2} + \alpha\nu$$

... Eq. 9

Here b and B^{gt} are the center point for b and B^{gt} , ρ is Euclidean distance and c is diagonal length. α is positive trade-off parameter and ν is consistency of aspect ratio.

We'll elaborate the specifics of YOLO v4 in this segment. Our YOLO v4 Model consist of:

Backbone: CSPDarknet53

Neck: SPP, PAN

Head: YOLO v3

YOLOv4 uses below BOF and BOS:

BOF of backbone: CutMix [13] and Mosaic Data Augmentation [20], DropBlock Regularization

BOS for Backbone: Mish Activation, Cross-Stage partial connection.

BOF for detector: CIoU loss, CmBN, DropBlock regularization, Mosaic data augmentation.

BOS for detector: Mish Activation, SPP-block, SAM-block, PANet

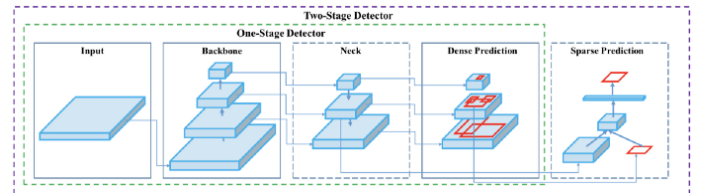


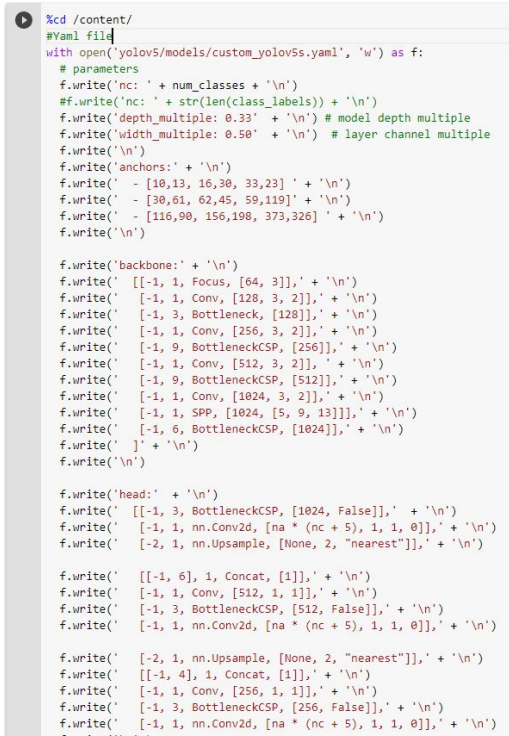
Fig. 7. Detector Architecture [13]

C. YOLO v5 Using PyTorch

YOLO v5 was released on 9th June with number of improvements which includes the improvements done in YOLO v4. YOLO v5 has been written in PyTorch and not in Darknet. This model is said to be faster and lighter than YOLO v4. Thus, we conducted our research on this newly launched model with our custom dataset [21]. The architecture of

YOLOv5 is basically a replica of YOLO v3 PYTorch repository that was published by Glenn Jocher et al. Many significant improvements made in PANet such as new heads, reduced parameters, faster inference. YOLO v5 basically comprises of Backbone: CSP- Cross Stage Partial Network [12] which is used for feature extraction, PANet is used for Neck to get feature pyramids. It help models on object scaling to generalize well. It helps to distinguish different sizes and scales of the same thing. It uses same model head as that in YOLO v4 and YOLO v3. Other changes that separates YOLO v5 from YOLO v4 is Activation Function and Optimization Function. YOLOv5 uses Leaky ReLU and Sigmoid activation function unlike YOLO v4. Default optimization function used here is SGD.

New Model Configuration Files: YOLO v5 formulates the setup of the platform in.yaml, as opposed to.cfg files in Darknet. In .yaml file we just need to specify the different layers in network and then multiply those by number of blocks. Below image is an example of what yaml file would look like:



```
%cd /content/
#Yaml file
with open('yolov5/models/custom_yolov5s.yaml', 'w') as f:
    # parameters
    f.write('nc: ' + num_classes + '\n')
    #f.write('nc: ' + str(len(class_labels)) + '\n')
    f.write('depth_multiple: 0.33' + '\n') # model depth multiple
    f.write('width_multiple: 0.50' + '\n') # layer channel multiple
    f.write('\n')
    f.write('anchors: ' + '\n')
    f.write('  - [10,13, 16,30, 33,23]' + '\n')
    f.write('  - [30,61, 62,45, 59,119]' + '\n')
    f.write('  - [116,90, 156,198, 373,326]' + '\n')
    f.write('\n')

    f.write('backbone: ' + '\n')
    f.write('  [[-1, 1, Focus, [64, 3]],' + '\n')
    f.write('  [-1, 1, Conv, [128, 3, 2]],' + '\n')
    f.write('  [-1, 3, Bottleneck, [128]],' + '\n')
    f.write('  [-1, 1, Conv, [256, 3, 2]],' + '\n')
    f.write('  [-1, 9, BottleneckCSP, [256]],' + '\n')
    f.write('  [-1, 1, Conv, [512, 3, 2]],' + '\n')
    f.write('  [-1, 9, BottleneckCSP, [512]],' + '\n')
    f.write('  [-1, 1, Conv, [1024, 3, 2]],' + '\n')
    f.write('  [-1, 1, SPP, [1024, [5, 9, 13]]],' + '\n')
    f.write('  [-1, 6, BottleneckCSP, [1024]],' + '\n')
    f.write('  ]' + '\n')
    f.write('\n')

    f.write('head: ' + '\n')
    f.write('  [[-1, 3, BottleneckCSP, [1024, False]],' + '\n')
    f.write('  [-1, 1, nn.Conv2d, [na * (nc + 5), 1, 1, 0]],' + '\n')
    f.write('  [-2, 1, nn.Upsample, [None, 2, "nearest"]],' + '\n')

    f.write('  [[-1, 6], 1, Concat, [1]],' + '\n')
    f.write('  [-1, 1, Conv, [512, 1, 1]],' + '\n')
    f.write('  [-1, 3, BottleneckCSP, [512, False]],' + '\n')
    f.write('  [-1, 1, nn.Conv2d, [na * (nc + 5), 1, 1, 0]],' + '\n')

    f.write('  [-2, 1, nn.Upsample, [None, 2, "nearest"]],' + '\n')
    f.write('  [[-1, 4], 1, Concat, [1]],' + '\n')
    f.write('  [-1, 1, Conv, [256, 1, 1]],' + '\n')
    f.write('  [-1, 3, BottleneckCSP, [256, False]],' + '\n')
    f.write('  [-1, 1, nn.Conv2d, [na * (nc + 5), 1, 1, 0]],' + '\n')
```

Fig. 8. Example of the YAML file

V. EXPERIMENTS

We tried to implement and test different models for object detection. Many experiments were conducted to implement Faster R-CNN which is best know for its speed and efficiency for object detection. Later, we implemented YOLO v4 which was pre-trained on MS COCO dataset and then YOLO v5 on our custom dataset.

A. Faster R-CNN

Faster R-CNN is best know for it image classification and detection. But also with that setup requirement of faster R-CNN is also quite complex. Early in our research we went with the approach of using docker to implement a faster R-CNN. Using Docker usually solves the challenges of version issue as its a ready to go container which has entire image file required to run the model. Implementing docker came with a challenge of itself. The docker image itself had some version issues which were difficult to troubleshoot and led us to drop the plan of using docker. The model that we were implementing was designed using Tensorflow.

Setup: The basic requirement to implement Faster-RCNN was Caffe/Pycaffe (Software), GPU (Titan K20,K40, Nvidia), Tensorflow Implementation of caffe had its own challenges. There were missing packages that were not being installed. We then manually installed the package. Following changes were also made in makefile of the model:

```
USE-CUDNN = 1
WITH_PYTHON_LAYER = 1
```

After installation of Caffe, a compatible version of CUDA was also installed during the process. The graphics card that was being used here was MX250 Nvidia. We installed Tensorflow to get things going. Current version of tensorflow is Version 2 whereas all the models that were implemented prior were trained and designed using Tensorflow V1. This resulted in version issues and many libraries which were no longer used in version 2. We downgraded the version from version 2 to version 1 which also led us to issues where path of tensorflow was not found. Since, there were issues with the manual setup of infrastructure for this model to run, we then proceeded to run this model on Google Colab. Google Colab provides us a virtual environment with everything pre-installed along with a dedicated GPU. Google Colab provided us the required environment where we were using Tesla K80. The issue was still not resolved as it had a version issue between CUDA and GPU Architecture. We made the changes in makefile and added the required config in makefile. We then experimented by changing parameters like anchor boxes and learning rate for the model. We set the learning rate to 0.0001 with decay of 0.0005 per batch. We then train the data for 200 epochs and 500 epochs respectively.

B. YOLO v4 using DarkNet

Since, we faced issues implementing Faster R-CNN, we then moved on to next best known model which has a great accuracy and speed. For YOLO v4, we decided to use a pre-train model to identify our objects. The model is pre-trained on MS COCO dataset. The model is known for its fast speed and accuracy.

Setup: The Implementation of model was done on Google Colab. The setup requirement for YOLO v4 is CUDA, GPU, Darknet. The backbone used for YOLO v4 is Darknet which is widely being used for research. We adopt the model where we first use a single image for testing the accuracy. We adopt

a batch size of 64 and subdivision = 8. Model was pre-trained using learning rate of 0.0013 and decay = 0.0005. We made a change in the model to pass multiple images at once and calculate the accuracy along with detecting the objects. Paths of all the images were added in a single .txt file which was imported in the model. We adopt a function which makes uploading and connecting our google drive to download predicted images easy. The weight file that was used is from MS COCO dataset. We experimented by changing the IoU threshold, subdivisions and IoU_loss = GIoU. As shown in Fig:3 after the experiment the best results we got were for following config: iou_thresh = 0.213, IoU_loss = ciou (complete IoU).

Fig. 9. Output from Yolo4.cfg

C. YOLO v5

YOLO v5 uses pytorch. As per the data available about YOLO v5, it is the pytorch version of YOLO v4.

VI. RESULTS

using their AP value and mAP along with their speed in time.

We also evaluate the performance of YOLO v5 which based on pytorch. Training speed was evaluated for both Faster R-CNN and YOLO v5 by training 50 images. The time taken by YOLO v5 was three times less when compared with Faster R-CNN. YOLO v5 gave us AP@.5= 0.908 and AP@.5:.95 = 0.606.

Fig. 10. Result from three detectors. 1. Yolo v5 using Pytorch 2. Yolo v4 using DarkNet 3. Faster R-CNN

Hence, YOLO v4 with DarkNet was used further to analyse the results obtained by the detection of classes from the test dataset. Based on the identification of the vehicles in classes such as Cars, Buses and Trucks, we got the results as mentioned in the table below:

TABLE IV
COUNT OF THE VEHICLES IDENTIFIED FROM THE FRAMES.

Unit	Video Frame	Classes	Count	Total
1	MVI 20012	Cars	22578	23863
		Bus	1285	
2	MVI 20032	Cars	12429	12430
		Bus	1	
3	MVI 39761	Cars	29262	30543
		Bus	1281	
4	MVI 39771	Cars	6644	7141
		Bus	497	
5	MVI 39781	Cars	15490	17068
		Bus	1578	
6	MVI 39801	Cars	11354	11355
		Bus	1	
7	MVI 40204	Cars	20767	21090
		Bus	323	
8	MVI 63544	Cars	12820	13013
		Bus	193	

count of 30543 vehicles identified which could mean there is a usual spike in traffic in that area while MVI-39771 has a count of 7141 meaning a free flowing traffic or a highway. We can not only find out the count of total vehicles but can also get a view of the type of vehicle such as large number of heavy vehicles such as Bus compared to the movement of light vehicles such as Cars. This data could later be used by the relevant authorities to monitor or divert traffic as and when required.

VII. LIMITATION

While experimenting with models, it was observed the most important limitation was working around with libraries and its versions which are different for each model implemented. While Faster R-CNN used PyTorch, YOLO v4 used DarkNet framework which led to inconsistencies during setup.

Once we had the setup in place, the next obstacle we identified was the annotation file which was part of the dataset was created for a dataset which was in a video form while the dataset was in frames in image (.jpg) format. This meant we either had to convert the existing annotation file to suit Faster R-CNN and YOLO separately as the type of input taken by each model is different. The second approach would be self annotating the images manually. Due to the complications while converting the existing annotations, we had to self annotate the train dataset which was time consuming.

While the approach we took to identify the object would have been the same, the approach the traffic flow would have been different if we were working around with a video dataset. The approach would have been the same to identify the vehicle and then classify based on the classes. The change would have been to get the count where a pixelated line would have been added to the video dataset where if the identified bounding box would have met with the line drawn would increase the count by 1. this would give us the count in real time surveillance scenario.

VIII. FUTURE WORK

As we were able to identify the vehicles of different classes and successfully obtained the count based on the frames in the form of images, the YOLO v4 framework with its high accuracy and performance speed would suit best for live traffic monitoring. Once we have successful identification and classification, additional data such as the location details, Weather, Time of the day could be later used to help create a model to predict the traffic condition of that part of the city which was be used by the traffic authority of that country to help divert traffic manually or automatically to avoid congestion and create an environment of smooth travel for each individual.

IX. CONCLUSION

In our research, we have successfully implemented 3 state of the art Object detection models and understood the working in various ways such as using it pre-trained on the MS COCO dataset as well as creating our custom dataset along with self-annotation of the Frames. This dataset was then used to train the YOLO v5 model and the Faster R-CNN model. Once all the models were trained, they were tested on the UA-DETRAC benchmark dataset. The results obtained were evaluated based on the accuracy and performance speed. It was observed that YOLO v5 model which is in its beta phase still gave us great results in terms of the mAP obtained along with the speed of execution which was 4 times faster than executing Faster R-CNN. YOLO v4 using DarkNet framework gave us promising results with an good mAP score and performance showing 3 times the speed of executing compared to Faster R-CNN. YOLO v5 still being in its beta phase is still unstable and hence YOLO v4 was used further for our analysis being the stable version. On executing the test dataset, we were able to get the count of Vehicles of different classes such as Cars, Buses, etc in different video frames letting us know the density of vehicle presence in a particular area compared to a different area of a city. This could be useful when the video is not present and monitoring needs to be executed on images.

REFERENCES

- [1] C. S. Office. (2018) Transport omnibus 2018. [Online]. Available: <https://www.cso.ie/en/releasesandpublications/ep/p-tranom/transportomnibus2018/roadtrafficvolumes/>
- [2] S.-t. X. Zhong-Qiu Zhao and X. Wu. (2019) Object detection with deep learning: A review. [Online]. Available: [IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS FOR PUBLICATION](https://arxiv.org/pdf/1506.01497.pdf)
- [3] R. Girshick. (2015) Fast r-cnn. [Online]. Available: <https://ieeexplore.ieee.org/document/7410526/authors/authors>
- [4] R. G. Shaoqing Ren, Kaiming He and J. Sun. (2016) Faster r-cnn: Towards real-time object detection with region proposal networks. [Online]. Available: <https://arxiv.org/pdf/1506.01497.pdf>
- [5] R. G. A. F. Joseph Redmon, Santosh Divvala. (2016) ;. [Online]. Available: <https://arxiv.org/abs/1506.02640>
- [6] Z. C. Z. L. M.-C. C. H. J. L. M.-H. Y. S. L. Longyin Wena, Dawei Dub. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. [Online]. Available: [Computer Vision and Image Understanding. journal homepage: www.elsevier.com](http://www.elsevier.com), year =
- [7] O. D. Freddy Kurniawan, Haruno Sajati. (2020) Image processing technique for traffic density estimation. [Online]. Available: [International Journal of Engineering and Technology](https://www.elsevier.com)

- [8] S. C. Chu W, Liu Y. Multi-task vehicle detection with region-of-interest voting. [Online]. Available: IEEE Transactions on Image Processing, 2018, 27(1): 432-441.
- [9] J. H. X. S.-J. C. Q. Z. Z. F.-Y. Z. Jianfeng Zhang, Xian-Sheng Hua. City brain: practice of large-scale artificial intelligence in the real world. [Online]. Available: The Institution of Engineering and Technology, 2019
- [10] G.-J. Q. C. S. Zhihang Fu, Zhongming Jin. Previewer for multiple-scale object detector. [Online]. Available: ACM on Multimedia Conference, 2018.
- [11] H. L. Z. D. Huansheng Song, Haoxiang Liang and X. Yu. Vision-based vehicle detection and counting system using deep learning in highway scenes. [Online]. Available: European Transport Research Review. Song European Transport Research Review <https://doi.org/10.1186/s12544-019-0390-4>
- [12] I.-H. Y. Y.-H. W. P.-Y. C. J.-W. H. Chien-Yao Wang, Hong-Yuan Mark Liao. Cspnet: A new backbone that can enhance learning capability of cnn. [Online]. Available: <https://github.com/WongKinYiu/CrossStagePartialNetworks>.
- [13] S. J. O. S. C. J. C.-Y. Y. Sangdoo Yun, Dongyoon Han. Cutmix: Regularization strategy to train strong classifiers with localizable features. [Online]. Available: <https://arxiv.org/pdf/1905.04899.pdf>. Source code available at <https://github.com/clovaai/CutMix-PyTorch>.
- [14] H. Gao. Faster r-cnn explained. [Online]. Available: <https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8>
- [15] H.-Y. M. L. Alexey Bochkovskiy, Chien-Yao Wang. Yolov4: Optimal speed and accuracy of object detection. [Online]. Available: Source code is at <https://github.com/AlexeyAB/darknet>.
- [16] S. R.-J. S. Kaiming He, Xiangyu Zhang. Spatial pyramid pooling in deep convolutional networks for visual recognition. [Online]. Available: <https://arxiv.org/abs/1406.4729>
- [17] H. Q.-J. S. J. J. Shu Liuy, Lu Qiy. Path aggregation network for instance segmentation. [Online]. Available: arXiv:1803.01534v4
- [18] G. Ghiasi, T.-Y. Lin, and Q. V. Le. (2018) Dropblock: A regularization method for convolutional networks.
- [19] W. L.-J. L. R. Y. D. R. Zhaohui Zheng, Ping Wang. (2019) Distance-iou loss: Faster and better learning for bounding box regression.
- [20] S.-D. K. Ju-Hyun Cho. (2015) Object detection using multi-resolution mosaic in image sequences. [Online]. Available: Signal Processing: Image Communication, Volume 20, Issue 3. <http://www.sciencedirect.com/science/article/pii/S0923596504001328>
- [21] G. Jocher. (2020) Yolo v5. [Online]. Available: <https://github.com/ultralytics/yolov5>