VEHICLE DETECTION, SPEED MONITORING AND CLASSIFICATION FROM VIDEO STREAM

By

Hoor Ul Ain Tahir	**01-132162-031**
Abdullah Waqar	**01-132162-001**

Supervised by

Engr. Waleed Manzoor

Submitted to the department of computer engineering in the partial fulfillment of the requirements for the degree of bachelor's in computer engineering.

Department of Computer Engineering

Bahria University Islamabad

2020

Bahria University Islamabad
Department of Computer Engineering

Dated: 15-07-2020

CERTIFICATE

We accept the work contained in the report titled "Vehicle Detection, Speed Monitoring and Classification From Video Stream" as a confirmation to the required standard for the partial fulfillment the degree of Bachelor of Computer Engineering.

Mr. Waleed Manzoor
Project Coordinator
Date: **15-07-2020**

Mr. Waleed Manzoor
Supervisor

Date: **15-07-2020**

Mr. Kamran Saeed
Internal Examiner

Date: **15-07-2020**

Dr. Usman Hashmi
External Examiner

Date: **15-07-2020**

Head of Department (CE)
Date: **15-07-2020**

UNDERTAKING

I certify that research work titled "*Vehicle Detection, Speed Monitoring and Classification from Video Stream*" is my own work. The work has not been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged / referred.

Hoor Ul Ain Tahir

01-132162-031

Abdullah Waqar

01-132162-001

DEDICATION

We dedicate this piece of work to our parents, foremost, for their unfathomable support and to everyone who was there to guide us and help us through every thick and thin. Each and every person whom we met during this journey had a great impact on us and helped us in reaching at this point of success.

ACKNOWLEDGEMENTS

All praises to Allah (SWT), The Most Gracious and The Empathetic, whose blessings and guidance remained with us during the execution of the project and without Whose help I could not have been where I am today. We would like to express our gratitude to our project supervisor Engr Waleed Manzoor for his precious time. This project task could not have come about without his technical help, moral encouragement and guidance. He provided us lots of advice and inspiration on our project. We would like to thank our parents for supporting and encouraging us with their best wishes.

ABSTRACT

Detection of vehicle and its license number plate is an important part in real-time applications. Many of the approaches initially detect a vehicle and number plate, after that recognize the characters by using only an image of the specific vehicle. This approach has low recognition rate because of the noise present in that specific frame. However, in our proposed system a different real-time approach is used to detect the vehicle and localize its license number plate instead of selecting a single frame to perform the recognition. In live video stream vehicle was detected, then by applying Laplacian filter having a certain threshold value, clear frames were obtained for further processing. Many of the existing solutions are not fast for real-world situations because of several constraints. In this thesis we present a robust and efficient vehicle detection and license number plate recognition system based on the state-of-the-art SSD object detection. By using transfer learning a new model was trained based on the collected dataset using SSD (coco dataset) pretrained model. This dataset contains 1693 images having 800x600 resolution also vehicles are of different types (cars, motorcycles, buses, and trucks). The SSD and KNN models are trained and finetuned for each stage of vehicle detection, license plate localization, character recognition and Color detection, so the system is robust under different conditions i.e. due to camera, lighting or background. The extracted information of vehicle number plate, type, color, time, speed, location, cropped images of vehicle and number plate will be stored in the database created using MongoDB. Also, specific data can be accessed from database using search option on frontend GUI application. The success of our system greatly depends on the quality of acquired video. Our real-time system efficiently and successfully localizes license number plate under different environmental conditions i.e. indoor, outdoor, or daytime. It has ability to detect license number plates of different provinces. These detected number plates can be of different colors, having different fonts, number plates may be having solid color background or an image as background. Depending upon all the mentioned factors accuracy of vehicle detection is 97.23 % and number plate localization is 89.84%. The live video stream is having 30 fps for which it takes 15ms to detect a vehicle.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

As there is manifold increase in traffic, which demands better road safety measures. One of the main contributing factors in road safety is speed of vehicles which if not observed properly may cause numerous fatal road accidents. Hence there is much need to monitor the speed of vehicles to enhance the road safety vis a vis control of accidents. Hence there is an urgent requirement to devise method through which we can identify the over speeding vehicle for that purpose we need to know type, color, and registration number of the vehicle.

Many industrial systems use dedicated camera systems which provide video input for monitoring of vehicles by retaining the record of their license number plate. Vehicle detection and Number Plate Recognition is an important research [1]–[3] because of its practical implementation. Number Plate detection has been applied to smart parking area system, automatic toll tax collection, traffic law enforcement, however, these surveillance systems face several challenges when there is heavy traffic vehicles having fancy license plates, tilted license plates and some known factors contribute in false positives results of Vehicle detection as well as license plate detection.[4]

The installation of High Definition (HD) cameras along with speed Gun on the roads assist in detection and speed monitoring of vehicle. However, the current systems most of the times is not been able to capture clear images due to speed mismatch of vehicles and camera shutter speed. As a result, system is not able to detect its feature i.e. Number Plate of vehicle. Since failing to detect Vehicle and Number Plate will lead to failures in next stages. So, detection of Vehicle and Number Plate requires higher accuracy or almost perfection. [5]

Deep learning is a vast topic of study in machine learning, it has a closer resemblance to Artificial Intelligence. By using Machine learning techniques several algorithms were created that can train computers and further capable to differentiate several instances of the same object. As there is increase in number of drivers and number of

vehicles on the road which eventually causes more problems related to traffic. Number Plate Recognition algorithm helps in the detection of vehicles and its number plates in effective manner by using fewer human resources. There are numerous reasons for the increase in importance as there are an increased number of vehicles on the road and each of the vehicle contains the number plate. Rapid growth in Digital Image Processing studies has made it possible to detect vehicle and its license plates at a speedy rate. The problems are detection of over speeding vehicle, highway monitoring, parking area administration are currently tackled using machine learning techniques.

In this project we will explore the use of deep learning, aiming at vehicle Detection and number plate recognition. This project gave the basic understanding of the modern neural network and its application working with the computer vision. By using the neural networks as building blocks, we were able to boost the accuracy of a model with its pre-trained model.

Transfer learning is the machine learning technique in which a model created for a task is reprocessed as the initial point for another model on the second task. As it is a widespread approach in deep learning, pre trained models/networks are used as the initial point on computer vision tasks, huge computation and time resources are required to develop neural network models.

In computer vision the neural networks typically try to detect edges in starting layer, middle layer is responsible for the shape and some task specific features in end layers. In transfer learning, starting and middle layers are used from the pre trained model and the final layers are trained again.

Transfer learning has numerous advantages, the main benefits are that it takes less time to train and also less amount of dataset it required whereas to train a new model we need a lot of time and a really huge dataset. The collect this much huge dataset a alone a really huge task and time taking. So that’s where transfer learning helps us and save our time and there is no need to collect huge dataset.

Object detection is one of the most common application of computer vision. Object detection means detecting objects in an image or a scene by software system or computer and locating objects in it accurately. In earlier years, object detection has been used in many fields like security systems, self-driving cars, face recognition and

pedestrian counting. With the recent development in terms of hardware and software, object detection has seen a flourishing development in some of the most advance systems.

After object detection, another task that one need to solve is to accurately draw boundary boxes around all the detected object. Today, if one can look at the technologies which depend on on object detection as part of the task, accurate boundary boxes help them to lessen the hardware cost. Robotic systems and Self-driving cars use a lot of hardware sensors apart from object detection software to exactly locate the position of object.

When someone's life is given in the hand of technology, even a millimeter of precision matters. For example, in case of self-driving car, object detection is the main task. So, if object detection software is capable of accurately drawing boundary boxes around the object it will further help in reducing hardware cost and decreases overall cost for a normal consumers and industries. So there comes a requirement for having single system and quicker execution. So, Our SSD model consists of 6 extra auxiliary convolution layers after the five layers of truncated VGG16. Two of them are fully connected. Five of them are used for object detection and in which three of them make 6 predictions and rest 2 layers make 4 predictions. SSD process is speed-up by eliminating the requirement for region proposal. As the accuracy was dropped so to enhance it SSD makes default boxes and works on the principle of multi-scale feature. After the enhancements made in SSD, it matched to the accuracy of Faster R-CNN's when lower resolution images were used, further the speed was improved as well. The SSD object detection composes of 2 parts: i) It extracts multi scale feature maps and ii) For object detection it applies depthwise separable convolution filters. [6]

We can understand the meaning and purpose of multiscale feature map by this simple example. If the distance between the car and camera is 5 meters, then it looks much bigger in the image, compared to the case if the distance is 50 meters between the camera and the car. Thus, for this example, we need two feature maps with different spatial scales where one is suitable when the distance is 5 meters between the camera and the car and the other one for when the distance is 50 meters from the camera. Note that even the feature map with the larger spatial scale (for the first case in which the distance is 5

meters between the car and the camera) is also able to capture the image of the car when it is 50 meter away from the camera, it does not provide accurate representation for the 50 meter away car since its spatial scale is not tight enough around the image car and contains substantial leakage of information from the other objects in the scene.

Major building block used for Convolutional Neural Network (CNN) is the Convolutional layers. Most simple function of convolution is to apply filter on the input Resulting in an activation. When same filter is applied multiple times to the input it results in an activation map also known as feature map that indicates the locations and strength of a detected feature in the input. Convolutional Neural Network is an innovative network that has the ability to discover a several filters in parallel particular to the training dataset under the limitations of the particular predictive modeling problem i.e. image classification. So, results are highly specific features that can be found anywhere in an input image.

Considering the convolutional neural network in which the convolution is basically linear multiplication operation between input and set of weights shows the behavior of traditional neural network structure. In this method we consider two-dimensional input for which the multiplication is carried out between two-dimensional weights and input data array it is called as filter or kernel.

The used kernel or filter is kept smaller than the input image, the multiplication carried out between a filter and filter sized area of an input image is dot product. Element wise multiplication between filter and filter sized area of an is a basically a dot product that is incorporated at the end and it gives a single output value input.

Intentionally smaller filter size is used as compared to the input in this way same filter is multiplied to the input array multiple times at the different points of the input image. Systematically filter is applied to each of the overlapping part or that particular filter sized area of an input image. The filter moves from left to right and top to bottom.

The application of same filter to the input image is a good approach. The filter that is designed to identify any specific type of feature in the given input, then by applying the filter systematically on the entire input image, then the filter learns the specific feature at any point of the image.

After multiplying input array and filter the output will always be a single value. When the feature map is created each value is sent in the feature map by a nonlinearity activation function i.e. ReLU.

Multiple features are learnt in parallel for the given input image. It is common for a convolutional layer to learn from 32 to 512 filters in parallel for a given input. In this method the model takes out the features from an input image in 32 or even 512 different ways.

There are multiple channels for the Colored images characteristically one for each color i.e. Red, green, and Blue (RGB). In data perspective the single image, which is provided as an input are three images of different colors. The number of channels as the input determine the size of filter as it must have same number of channels. As an input image have 3 channels or depth of 3 then a filter applied on that image must consist of 3 channels. In this instance, a 3 × 3 filter would be for 3 x 3 x 3 or [3, 3, 3] for rows, columns, and depth. This means that if a convolutional layer has 32 filters, these 32 filters are not just two-dimensional for the two-dimensional image input, but are also three-dimensional, having definite filter weights for each of the three channels. Yet, each filter gives a separate single feature map. Which means that the depth of the output of applying the convolutional layer with 32 filters is 32 for the 32 feature maps made.

In case of SSD instead of doing simple convolution, we apply different method i.e depthwise separable convolutions which includes depthwise and pointwise convolution. In this way the computation / Number of multiplications are reduced which results in the increase in the overall speed of the model. In depthwise convolution, we apply a 2D filter at each depth level of input image. For example. Suppose our input image is 8 x 8 x 3 Filter is 3 x 3 x 3. In a simple convolution we would directly convolve in depth dimension as well. Whereas in depth-wise convolution, we use each filter channel only at one input channel. To produce same effect with normal convolution, we select a channel, make all the elements zero in the filter but that channel and then preform convolution. For each channel we will need three different filters. Even though parameters are remaining same, by using an only one 3 channels filter this convolution gives us three output channels.[5]

We can understand the concept of pointwise convolution by this example. Suppose you are trying to locate some old photes of your childhood days which are held in one of the albums inside one of your closets at home. Now you don't know where that exact photo which you are looking for is. You took almost all the compartments from all the rooms to your room and you can start searching for it. And after searching for a while at last you found it. But now the room is a mess and while searching you mixed all the stuff in one place. Now the job is to keep the things correctly at their particular places. Now that's exactly how important a 1 x 1 is for our model. A 1 x 1 kernel does three main roles in any model, decrease the number of channels in a compute/ memory constrained environment, Increase the number of channels, decrease the number of channels. You may not have to isolate things accurately if the total number of rooms in your home are equal to the number of compartments. In that situation you can just keep one compartments per room. Likewise, you may be able to manage with images of size 12 x 12 now. But if you need to process image of size 1280 x 720 and a greater number of them altogether then your GPU will not be able to handle this much of computations. So, it's very vital to keep the number of parameters as less as we can, and to accomplish it we need to understand and make proper use of 1 x 1 kernels (pointwise).

We used K nearest neighbor KNN [7] technique to detect the color of the vehicle. It is a type of supervised machine learning algorithm which can be used for regression and also classification predictive problems. The advantage of using KNN is that it does not have a specific training phase but it uses all the training data while preforming classification. It uses feature similarity to predict the values of new datapoints. Other advantages our that its implementation is very simple, its robust with regard to space search space and takes very few parameters to tune.

CHAPTER 2

Literature Review

Vehicle Detection and Number Plate recognition has been researched often in many studies, already existing solutions are still not robust enough on real-time situations. The existing Vehicle Detection and Number Plate recognition systems are not efficient for real-time video stream obtained via HD Cameras. Such systems do not indicate accuracy or work efficiently because given solutions depend on limitations such as search of vehicle or number plate in fixed region, Lightning conditions, viewing angle of camera. Most important factor of the system is clarity of image or certain frame of video stream on which processing is to be carried out. Computer vision [3] and pattern recognition previously existing algorithms of Vehicle Detection and Number Plate recognition systems deal with many challenges due to the above stated constraints.

Many of the computer visions tasks have lately accomplished betterment in performance because of a huge annotated dataset and the hardware Graphical Processing Units (GPUs) efficient of handling enormous data. For such situations Deep Learning studies are used. However despite the notable progress of Deep Learning approaches in Vehicle Detection and Number Plate recognition [4]–[6], there is still a great need for Vehicle and Number Plate annotated datasets. Trained model determines the outcomes of Deep Learning technique. The more amount of data allows use of more fast and strong network architectures having more enhanced info of parameters and layers. Existing systems do not show much accuracy because of their datasets. Models which are trained on these datasets have some limitations that it may always use camera mounted statically in the same position so all the images will have similar and might have simple backgrounds as there are chances no motorcycles is present and only in few of the cases where the Number Plates might not be well aligned.

In recent researches remarkable outcomes in several computer vision tasks like object detection [8] [9] [10] has been seen by applying deep learning-based approaches. Deep networks learn the discriminative feature by itself automatically directly from the dataset provided to it this causes the main reason behind its extraordinary performance. For the object detection the deep neural network has outpaced old typical traditional methods of searching. Deep learning approaches used for object detection are mainly divided into two types that is region proposal based method and regression-based method. [11]

Earlier implemented Deep networks, Convolutional Neural Network (CNN)[12] based detection methods like R-CNN starts localizing the location of the objects and object scales that are according to the given test image which is basically an input for the object so when training it returns resultant proposed region to the classifier to detect the object. When classification is completed then the post-processing is applied so that the bounding boxes can be rectified, along with re-scoring bonding boxes which is created on the other objects in that particular frame.[13]

Lots of improved techniques based on CNN[12] emerged for object detection which include RCNN[13], fast RCNN, faster RCNN and R-FCN. These new emerged techniques attained better accuracies, but they lack due to complex network structure. To increase the speed researchers recommended techniques consisting of a single network which will directly predict bounding boxes instead of searching for region of proposals. YOLO being a regression based technique which returns object borders and directly give their recognition confidence. As YOLO has refined network structure, capable of achieving real-time processing on Graphical Processing Units (GPUs) for object detection but still this technique has the problem of inaccurate positioning of object.

You Only Look Once (YOLO)[9] and Single Shot Detector (SSD)[3] technique in which object detection task steps of classification and localization is converted into the regression problem, that can perform object detection with a single neural network. In YOLO, the problem for improving speed for real-life scenario was overcome in this system by combining a region of proposal with that of classification to the exceptional

regression problem through an image pixel to bounding box coordinates consisting of all class probabilities that will assess the whole image in the single run.

As YOLO still faces the problem in detecting smaller objects in a particular frame. This issue was resolved by using SSD approach in which it makes default boxes and uses multi-scale features for the convolutional detection layers. [9]

SSD a deep learning algorithm that is based on the Anchor method idea of Faster R-CNNs [14]. It consists of the traditional classification networks i.e. VGG16 [15] and after the VGG Network extra auxiliary layers also introduced for feature extraction in which pointwise and depthwise convolution takes place. The scale of extra layers change so that it can detect multiscale objects present in the frame. For training SSD[4] model a dataset images with accurate labels are needed., A large number of datasets is required in order to make weights more fitted. Collection of dataset images and then labelling them is a time taking process.[11]

Both the algorithms YOLO and SSD use convolutional neural network as basic network for feature extraction which make them far better than traditional object detection methods in accuracy and speed. The difference between YOLO and SSD is that the YOLO[9] algorithm uses multiple convolutional layers for feature extraction and then via fully connected layer predicts the output probability. But SSD algorithm uses different size auxiliary convolutional layers for the feature extraction after that calculates the location-loss, confidence-loss and predicts the bounding box. The SSD[3] algorithm do not consist of full connected layers which eventually helps in making the detection speed more faster and it uses the features that are obtained from multiple size convolution layers so in this technique less features are lost and detection accuracy is improved.[16] SSD showed improvised result as the models gives less detection false-rate and accuracy-rate about 0.97.

CHAPTER 3

Design and Methodology

Our project is divided into five main parts. Vehicle detection, color detection, number plate localization, number plate recognition and speed monitoring. In first part we detect whether the input stream has a vehicle in it or not. If the vehicle is detected by the algorithm in the input stream, then the defined method will exactly find and crop the vehicle from the input stream. Now the output of phase one will be considered as input of phase two and three, Phase two will detect color of the vehicle using a defined algorithm explained below and phase three will search for the vehicle's number plate and crop out the number plate, now the output of the phase three will be the input of phase four in which it will recognize the characters on the vehicle's numbers plate. Through researching, we found that vehicle detection is already pretty developed and there are number of methods exist online but still, we decided to use deep learning for this project. We improved the accuracy with some pre-trained models.

We faced many challenges during our project, the first challenge was the gather the pictures of different type of vehicles which should include Cars, Buses, Motorcycles and Trucks. We labeled those images one by one and generated XML and CSV files Using this data, we trained our SSD model which can detect/localize and classify vehicles in an input live video stream but the accuracy was not significant. We increased the number of pictures and trained again and good phenomenal results. The next challenge was to detect the number of each vehicle. For this purpose, we applied different image processing techniques explained below. The third challenge was to recognize the characters on the number and also at to exclude extra writings on the license plate.

The flow of out project is shown in the figure 1 below.

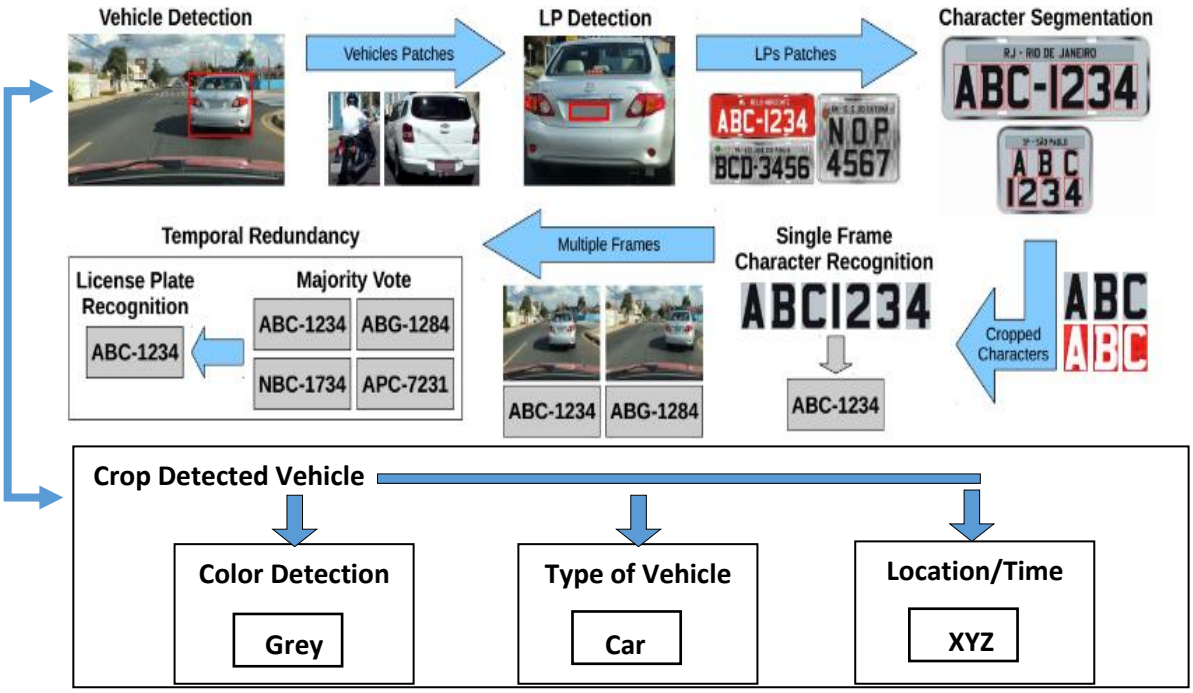


Figure 1: Flow of Project

3.1 Stages of the system:

Automatic license plate reorganization ALPR systems typically have Five stages: Acquisition of live video stream, Vehicle detection, License Number Plate Extraction, Color Detection, and type of vehicle. The earlier stages needed more accuracy or almost perfection, since failing to detect the number plate would mean failure in the next stages as well. Many methods search first for the vehicle and then its number plate to reduce processing time and eliminate false positives. [15]

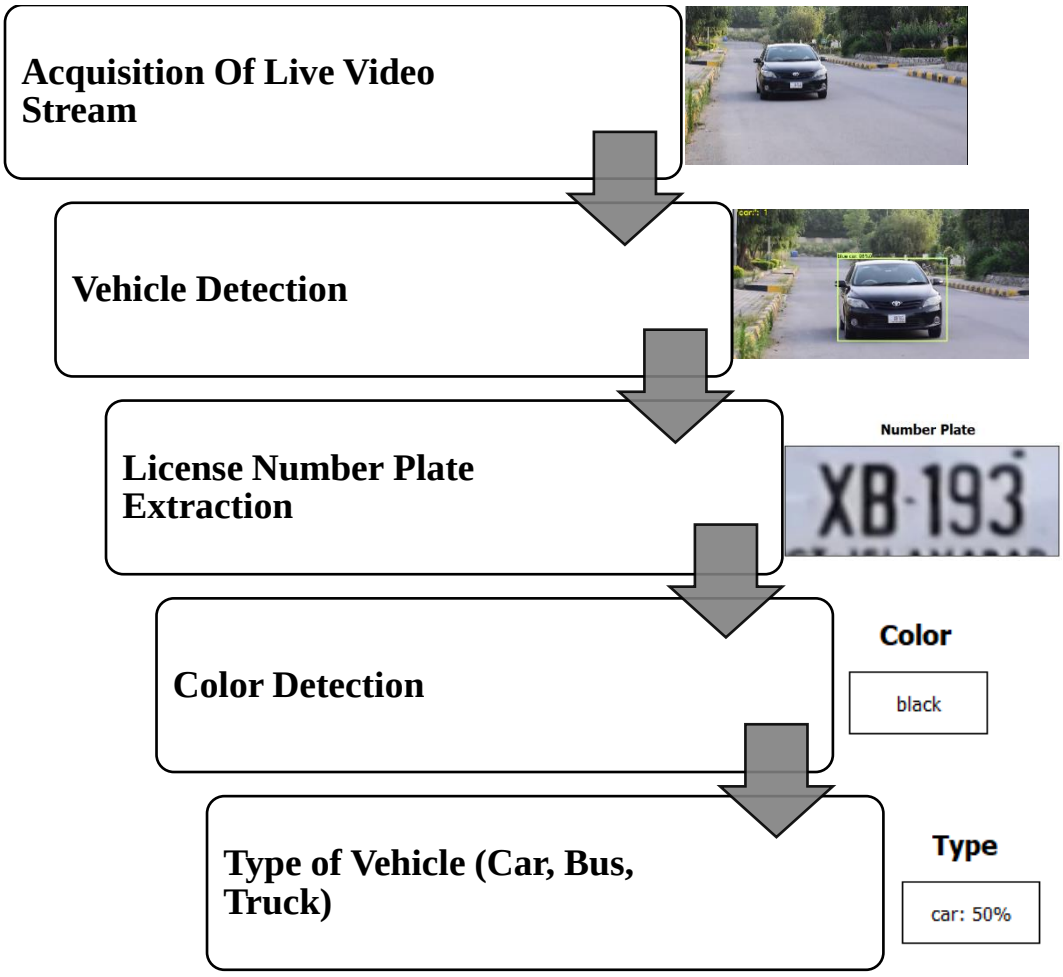


Figure 2: Stages of ALPR

Another Important aspect of this project is storing all the gathered information in a search able database. For which we have used MongoDB because of its faster speed of storing and retrieving the data. Also, almost no actual constrains as compared to MYSQL or oracle. A GUI was designed using Qt designer and implement in our project using PyQt5 library of python and embedded our database with it to make it searchable, it is an advanced search in which we can search vehicles according to our requirement. Each time when a vehicle is detected all its information i.e. Color, Type Location, Speed, and Time will be extracted and stored in our database and can be searched anytime using this GUI.

3.2 Initial Plan

We planned to first build our own CNN model for detection and classification of vehicles in a live video stream but this technique was too slow for our real time system and also accuracy was not satisfactory.

We applied some data augmentation on the collected dataset as a reasonable method to improve the prediction accuracy. Based on this idea we completed the research on currently prevailing solutions for vehicle detection.

3.3 Collection of datasets

Firstly, we collected 976 images of different types of vehicles which includes Cars, Motorcycles, Buses and Trucks and by using a pre-trained model we trained our own SSD model but the accuracy was not got as expected. Then we increased our dataset to 1693. These images were collected from internet and some organizations which kept record of the vehicles. The average image size of our dataset was 960 x 734, and the largest image resolution is 3424 x 2428. The main idea for collecting this type of dataset over some other small dataset is that we had to crop the detected vehicle and pass the output to the process where the number plate is localized and character recognition is applied. The whole program will lose accuracy if resolution of initial input is too low. So, there is a tradeoff between speed and accuracy/performance. Large resolution images required extra training/testing time and more effective device, but it gives us a higher chance to detect a car and its number plate.

The problem occurred when using high-resolution images as a test set, as the memory was not sufficient enough to hold them. So, instead of passing images of different sizes we collectively resized the image to 800x600 resolution. Code attached in appendix 1.

3.4 Training

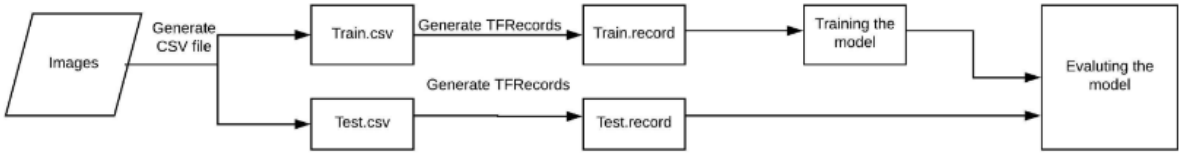


Figure 3: Basic Flow diagram of training

To train a Deep learning model we have to preform some operations on our dataset on the bases of which our network extracts the features and learn. For this purpose, we have to first label our dataset and generate XML files for each image, by using those XML files we generate CSV files for training and testing images separately. Then from these CSV files we generate tensor flow records (tf records) separately for training and testing and by using these tf records we train our model (figure 3). Details of each step is given below.

3.4.1 Labeling and XML creation

The first challenge of our project was to pre-process our custom datasets. The dataset was gathered from different resources i.e. from internet, Organizations that keep record of vehicles. We had 1693 image, spilt them into training and testing set. As there were 1293 images in training set and 400 images for the testing set. So, for vehicle detection, we needed to manipulate our dataset and generate the information of each image.

We used the “labelImg” labeling software and manually drew the bounding box as shown in Figure 4. This software records the appearances of cars from each image and also the exact coordinates of bounding boxes. Necessarily accuracy of training depends on labelling in this case. The labeling tool generated XML files for each image. In these XML files the information about four coordinates of the bounding box that we drew are saved in the form of HTML code.

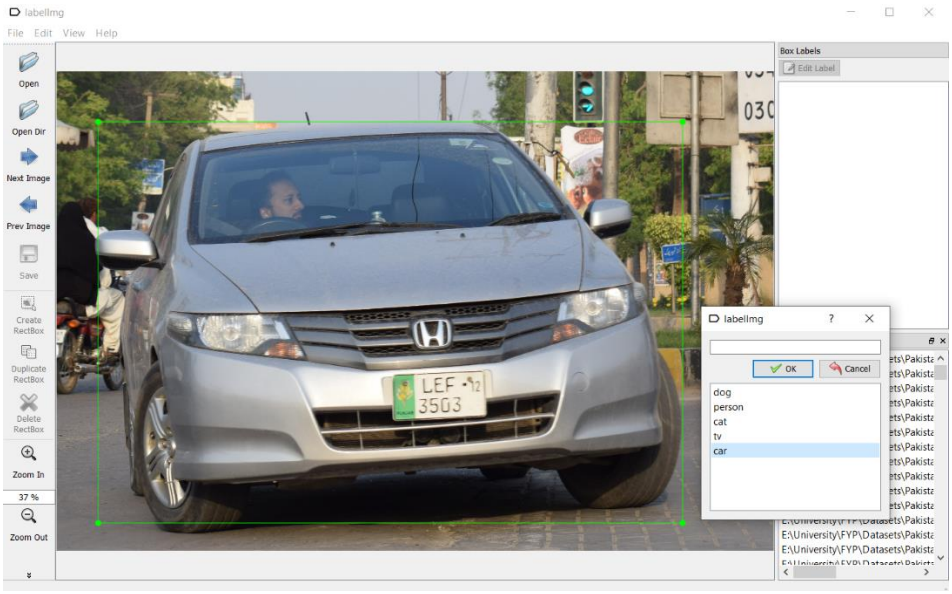


Figure 4: labelling using LabelImg

3.4.2 Training on Google Colab:

With a significant increase in size of datasets for deep learning models to fit as a result there has been an increase in demand for Graphical Processor Units to train the model faster.

We used Google Colab for training. Google provides a free GPU i.e Tesla K80 GPU on the cloud for running large scale machine learning projects and also 13 GBs of RAM. In case of uploading our dataset on this platform there is a drawback that the dataset will be removed when session is restarted. So by mounting Google Drive to the Notebook and using this feature, instead of uploading the file each time we open the notebook.

3.4.3 Uploadation on google drive:

Firstly, upload the whole dataset which includes train and test images along with their XML files on google drive. So, we can access them during the training on our google colab notebook. Afterwards mount your drive with your google colab notebook to easily access the dataset and XML files for further processing.

3.4.4 Converting XML into CSV:

Once drive is mounted with google colab, now start the training process. First, convert the XML files of training and testing images into CSV files, which stores the name of the image, dimensions, the name of the class (the bounding box we created), and information about the four coordinates of the bounding box.

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	0	800	600	No_plate	375	377	480	434
3	1	800	600	No_plate	185	400	271	455
4	10	800	600	No_plate	131	399	206	453
5	100	800	600	No_plate	624	309	704	347
6	101	800	600	No_plate	447	430	536	474
7	102	800	600	No_plate	439	329	516	376

Figure 5: XML into CSV

We will create separate csv files for given training images and testing images XMLs.

3.4.5 Generating TensorFlow records and label map:

The TFRecord is a format for storing data in a sequence of binary records. To read data efficiently and faster it is helpful to serialize the data and store it in the set of files that can be read by a computer system easily.

When working with a large dataset it is necessary to store data in such format which is easy for the computer system to understand read. Eventually by doing so the system can read the data faster and takes less time to copy and takes less space in memory hence speeds up the training process.

We have to generate separate TFRecord for training set and the testing set to train and also evaluate our model.

3.4.6 Batch size:

Set batch size to 12, it fits with google colab's K80 Tesla GPU memory for particular model. Batch size is number of images CPU or GPU can handles at a time. As our dataset has 1693 images, it divides those 1693 images into sets of 12.

Batch size impact

The batch will affect following

- Model performance
- Step duration

The batch size will not affect:

- Training duration

3.4.7 Epochs:

We have dataset of 1693 images and divided into 142 batches, and then 12 iterations will complete 1 epoch.

We set the number of training steps/epochs to 50.

The more the epochs the better the training.

3.4.8 Configuring a Training Pipeline:

First, install required packages i.e. pillow, matplotlib then rather than training model from scratch, transfer learning will be done from a pre trained object detection model.

Transfer learning will take small amount of dataset as compared to training from scratch. In this case base model that we used was trained with coco dataset of common objects.

As we are using tensorflow object detection API so it necessary to download the pretrained model weights and checkpoints and then configure the corresponding pipeline config file to inform trainer about following information.

- The path to the pretrained model checkpoint.
- The path to training and testing tfrecord files.

- The path to the label map file.
- The training batch size.
- The number of training steps.
- The number of classes of our model.

We have used ssd_mobilenet v1 coco_2018 as our pre trained model. And its pipline file from tensorflow object detection API ssd mobilenet v1 coco.config. It took around 12-14 hours to complete the training.

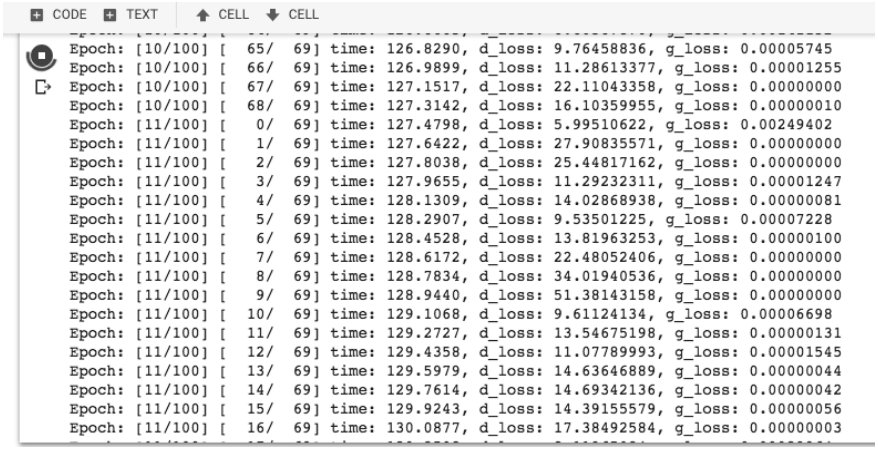
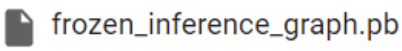


Figure 6: Epoch during training

3.4.9 Exporting Trained Inference Graph

After completing the training, we extracted the freshly trained inference graph, which will help us to perform vehicle detection.

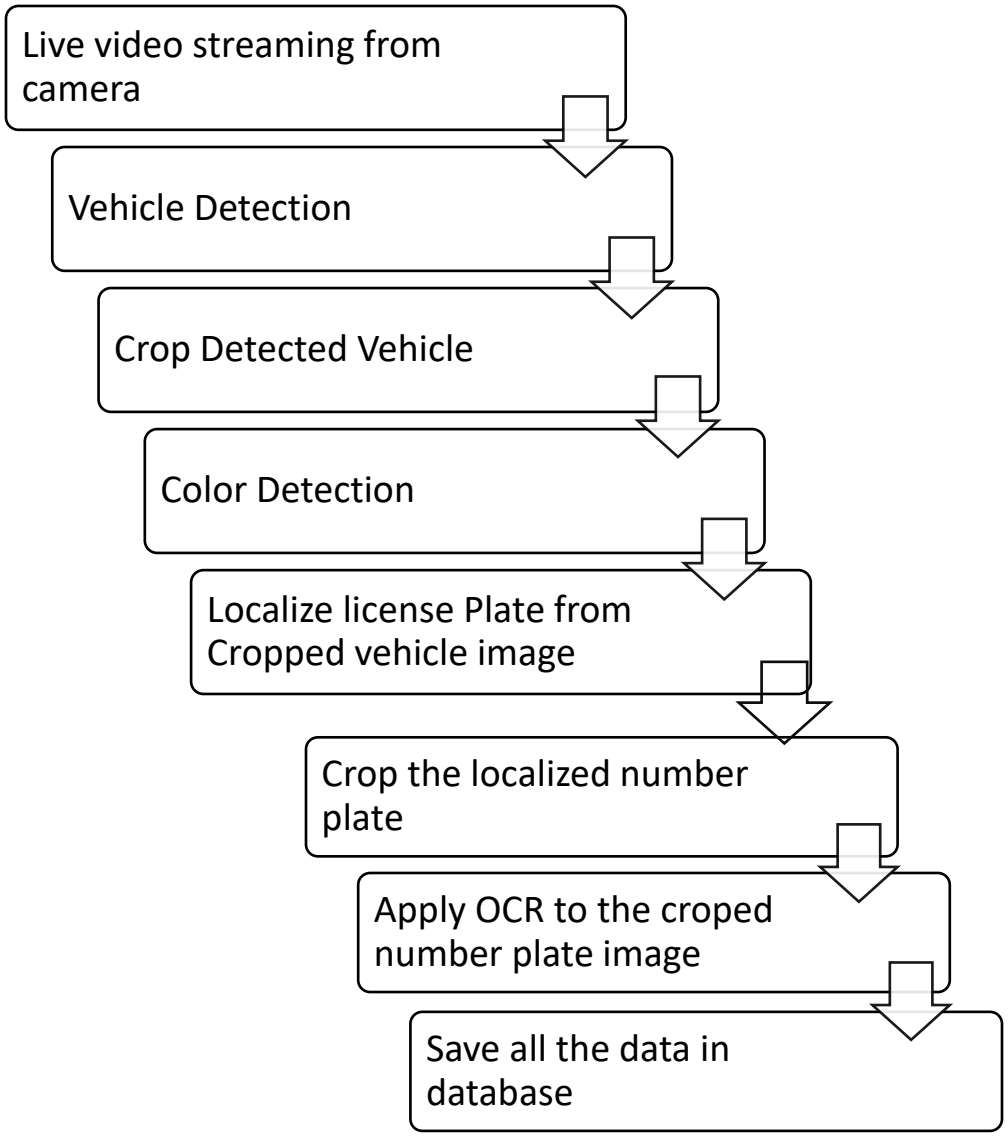
After exporting the newly trained inference graph, we need to download the .pb file.



That is our model file and we will use this file in our algorithm.

3.5 Project chronological Steps

Once the model is trained and downloaded, now comes the part where we use our trained model in real-time scenarios and get the required results.



3.5.1 Live stream from camera

Hanse 260x is a CCTV camera, we will take live video stream from this camera. The reason behind using this model is that it has good motion detection and tracking auto focus which help us in getting better results. After connecting the camera with our system, we will pass the live stream to our algorithm and we will process the live stream frame by frame. [10]

Figure 7: Hanse Camera

Features of our camera:

- Motion Detection, AWB, ALC, AES, AGC, Tracking auto focus
- 64 Presets (Zoom, Focus), Reverse (H/V) & Negative Function
- RS232 / 485, OSD Function
- Night vision Function
- 470 TV Lines
- Slow Scan : 0.05 Lux
- 10X Digital Zoom
- Digital CCD Color Zoom Camera
- ¼ ± CCD 410,000 (NTSC), 470,000 (PAL)
- 26X Optical Zoom

Specifications:

Specifications		
Description	**CA-Z260N(260X)**	**CA-Z260P(260X)**
TV System	NTSC 60Hz	PAL 50Hz
Image Device	1/4¡±CCD	1/4¡±CCD
Total Pixels	410,000	470,000
Effective Pixels	380,000	430,000
Horizontal Frequency	15,734Khz Standard	15,625KHz standard
Vertical Frequency	59.94Hz	50.00Hz
Total Zoom	260X (Optical 26X, Digital 10X)	
Focal Lenth	F1.6~F3.8, f=3.5mm~91mm	
Focus	Auto / Manual	
Sensitivity	1 Lux (F1.6, 30IRE)	
Slow Scan	0.05 Lux	
White Balance	Auto (ATW or AWB) / Fix	
Operating Temperature	PAL or NTSC	
Power Source	DC12V ¡¾ 1.0V	
Dimension	130(W) X 62(H) X 67(D)mm	

3.5.2 Detection of vehicle using SSD Model

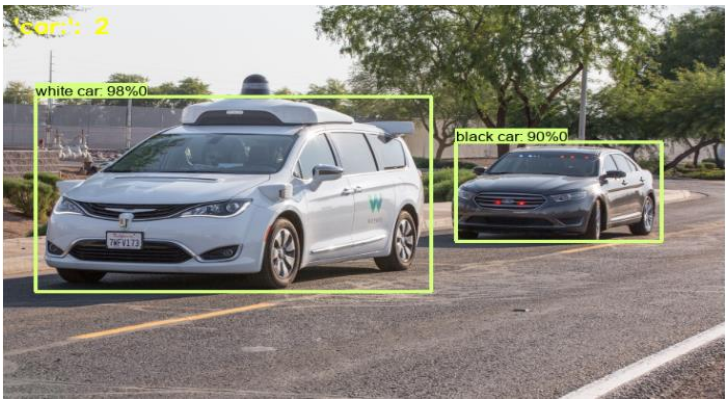


Figure 8: Detection of vehicle using SSD model

In the real time scenarios SSD [3] is created for object detection. For Faster R-CNN a technique of region proposal is used to make boundary boxes and further these boxes are used for the classification of objects. SSD process is speed-up by eliminating the requirement for region proposal. As the accuracy was dropped so to enhance it SSD makes default boxes and works on the principle of multi-scale feature. After the enhancements made in SSD, it matched to the accuracy of Faster R-CNN's when lower resolution images were used, further the speed was improved as well. As per given comparison, it achieved faster processing speed and more accuracy than Faster R-CNN in real-time scenario.

(Accuracy is measured as the mean average precision mAP: the precision of the predictions). [18]

System	VOC2007 test *mAP*	FPS (Titan X)	Number of Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	~6000	~1000 x 600
YOLO (customized)	63.4	45	98	448 x 448
SSD300* (VGG16)	77.2	46	8732	300 x 300
SSD512* (VGG16)	**79.8**	19	24564	512 x 512

The SSD object detection comprises of 2 parts:

- It extracts multi scale feature maps.
- For object detection it applies depthwise separable convolution filters.

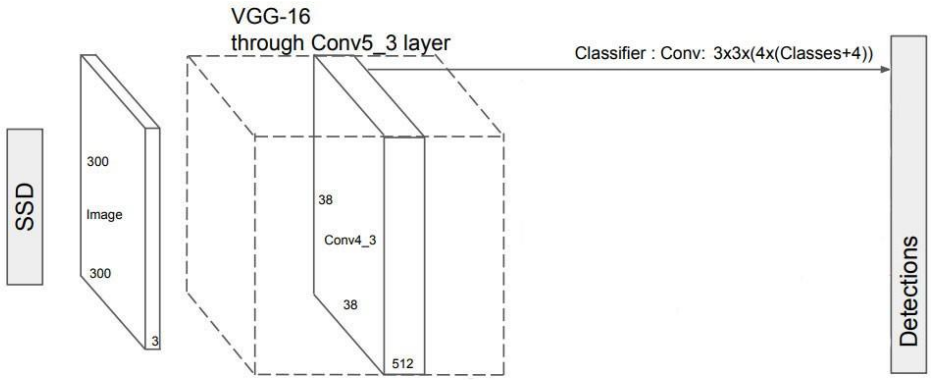


Figure 9: SSD: Single Shot MultiBox Detector.

SSD using truncated VGG16, it uses first 5 layers of VGG16 to extract feature map. Rest of the fully connected layers of VGG16 are discarded. The network then detects the objects using the Conv4_3 layer. Each of the prediction comprises of a boundary box and 4 probability scores for each class one extra score for no object and then we pick the highest probability score as the class for that bounded object. At Conv4_3 layer, 4 predictions are made per cell and it makes total 38x38x4 = 5776 predictions regardless of the depth of the feature maps. And many of the predictions doesn't contain an object. It keeps a separate class "0" to indicate that there is no object. [19]

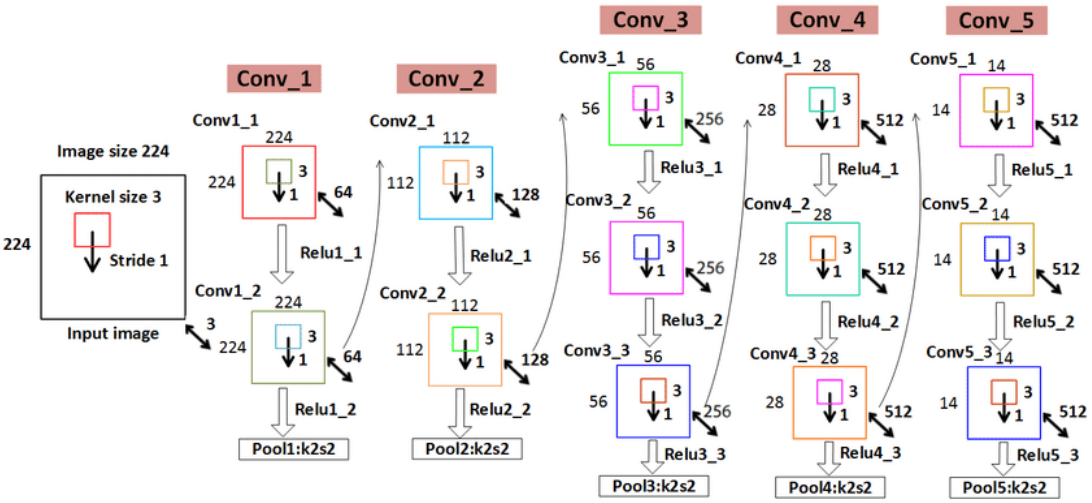


Figure 10: Truncated VGG16

Predictors of our object detection model

When the feature map is extracted SSD network applies 3 × 3 convolution filters on each cell for making predictions. Each filter gives 4 probability scores for each class (i.e. Car, bus, truck. Bike) plus one boundary box to locate the object and to identify the class. SSD uses multi scale feature maps to detect objects independently. Which helps the network to detect objects even in low resolution.

Our SSD model consists of 6 extra auxiliary convolution layers after the five layers of truncated VGG16.Two of them are fully connected. Five of them are used for object detection and in which three of them make 6 predictions and rest 2 layers make 4 predictions [19].

Total number of predictions:

- Conv4_3 (from VGG16) makes 38 x 38 x 4 = 5776 predictions
- Conv7 makes 19 x 19 x 6 = 2166 predictions
- Conv8_2 makes 10 x 10 x 6 = 600 predictions
- Conv9_2 makes 5 x 5 x 6 = 150 predictions
- Conv10_2 makes 3 x 3 x 4 = 36 predictions
- Conv 11_2 makes 1 x 1 x 4 = 4 predictions

If we sum them up, we get 5766 + 2166 + 600 + 150 + 36 + 4 = 8732 predictions. The greater the number of predictions, the better the accuracy.

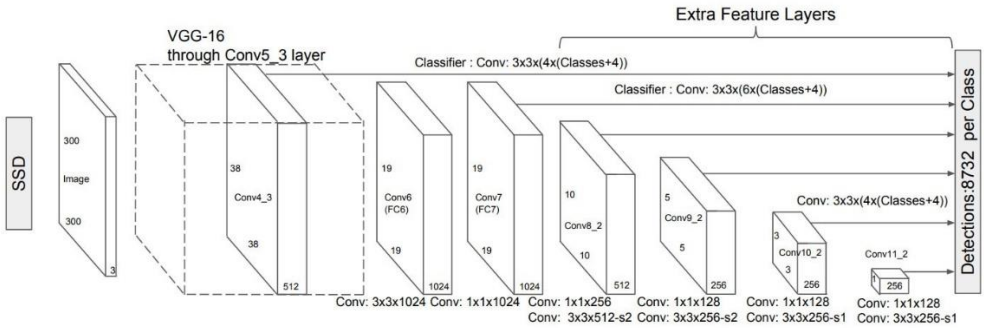


Figure 11: Layer of SSD

Lower resolution feature maps (right) identifies larger scale objects. Multi-scale feature maps enhance accuracy considerably.

The predictions of our SSD model are classified as, negative matches or positive matches. For calculating a localization cost (the mismatch of the boundary box) SSD network uses the positive matches only. If the resultant default boundary box has an IoU (intersection over the union) greater than 0.5 the match is positive. Otherwise it is negative. [17]

The main idea of using a SSD network is due to its faster speed. The process is speed up by using depth wise separable convolution because this reduces the number of computations/Multiplication. It include two steps. Depthwise convolution and pointwise convolution.

1. **Depth wise convolution:**

   In depthwise convolution, we apply a 2D filter at each depth level of input image. For example. Suppose our input image is 8 x 8 x 3 Filter is 3 x 3 x 3. In a simple convolution we would directly convolve in depth dimension as well. Whereas in depth-wise convolution, we use each filter channel only at one input channel. To produce same effect with normal convolution, we select a channel, make all the elements zero in the filter but that channel and then preform convolution. For each channel we will need three different filters. Even though parameters are remaining same, by using an only one 3 channels filter this convolution gives us three output channels. [3]

2. **Point wise convolution:**

We can understand the concept of pointwise convolution by this example. Suppose you are trying to locate some old photes of your childhood days which are held in one of the albums inside one of your closets at home. Now you don't know where that exact photo which you are looking for is. You took almost all the compartments from all the rooms to your room and you can start searching for it. And after searching for a while at last you found it. But now the room is a mess and while searching you mixed all the stuff in one place. Now the job is to keep the things correctly at their particular places.

Now that's exactly how important a 1 x 1 is for our model. A 1 X 1 kernel does three main roles in any model, decrease the number of channels in a compute/ memory constrained environment, Increase the number of channels, decrease the number of channels. You may not have to isolate things accurately if the total number of rooms in your home are equal to the number of compartments. In that situation you can just keep one compartments per room. Likewise, you may be able to manage with images of size 12 x 12 now. But if you need to process image of size 1280 x 720 and a greater number of them altogether then your GPU will not be able to handle this much of computations. So, it's very vital to keep the number of parameters as less as we can, and to accomplish it we need to understand and make proper use of 1 x 1 kernels (pointwise).

3.5.2.1 Laplacian Filter

As we are taking video as an input, we may get redundant frames in which the object/ vehicle might not be clear due to some unknown factors. further processing on those frames will decrease the overall accuracy of system and we will get redundant data. So, in order to save our system from processing those unclear/ blur frames we will discard them and only use process potential frames. For this purpose, we used Laplacian filter to detect the potential frame which have sharp features. The Laplacian filter return a value. This value is the average number of sharp edges detected in the image. The greater the value the

sharper and clearer the image. We set this threshold to 800 on hit and trail bases. It may vary in different situations or different specifications of the camera used for the input video stream.

3.5.3 KNN Color Detection

We used K nearest neighbor KNN technique to detect the color of the vehicle. It is a type of supervised machine learning algorithm which can be used for regression and also classification predictive problems. The advantage of using KNN is that it does not have a specific training phase but it uses all the training data while preforming classification. It uses feature similarity to predict the values of new datapoints. Other advantages our that its implementation is very simple, its robust with regard to space search space and takes very few parameters to tune.

To implement this technique, we first have to generate out training data. In this case our training data will be the intensity values of different colors which we want to classify. So, we extracted the RGB values of different colors of different shades and stored them as our training set. Now the cropped vehicle image is passed to this classifier. It extracts the RGB values of the input image and compare with the training data we generated earlier by using the Euclidean distance formula

$$D(a,b) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2}$$

After finding the distance from all the training data we pick the 3 least distance values and check their classes, the class which occurs the most is assigned to the vehicle. The process in which it determines the least 3 values is pre-defined. It is the part in which we set the value of K and in our algorithm, we set it at 3.

3.5.3.1 Challenges faced in vehicle color detection

Vehicle color detection is not an easy task. The color may vary in different conditions. These conditions are as follows:

I. Sunlight reflection on the vehicle.
II. Shadow on the vehicle.
III. Weather may be cloudy of rainy.
IV. Dirt on the vehicle.

All these conditions may affect the accuracy of our color detection algorithm, to overcome this anomaly we created a more bigger training data with more variety of shades and more variety of colors in it. This helped us to overcome the problems occurred in these conditions and improved the accuracy as well.

3.5.4 Localization of Number plate

After the vehicle is detected in a video frame, we draw a final boundary box around it and crop the detected vehicle from the main input frame. To find the ROI in the image (area having License plate) we applied multiple image processing techniques shown in figure 12.

First the cropped vehicle image is converted to gray and applied top and black hat. Black hat enhances the dark object of interest in a bright background where as top hat enhances the bright object of interest in a dark background. This step is very important because to detect fancy and different color number plates we needed to a make a new and unique method. As different provinces has different color of number plates and different color of characters. After this step we applied gaussian blur and thresholding and then find the contours, this gave us the required edges in the image. Then comes the most important step of finding the possible characters. For this we first generated an identification data. We stored the flattened image matrix of each character from 0 to 9 and A to Z in different fonts. By comparing these matrixes values with the parts of the input image where the edges are found, if any similarity is observed. We label it as a character and where ever the characters are found in series we label it as number plate and draw a bounding box around it

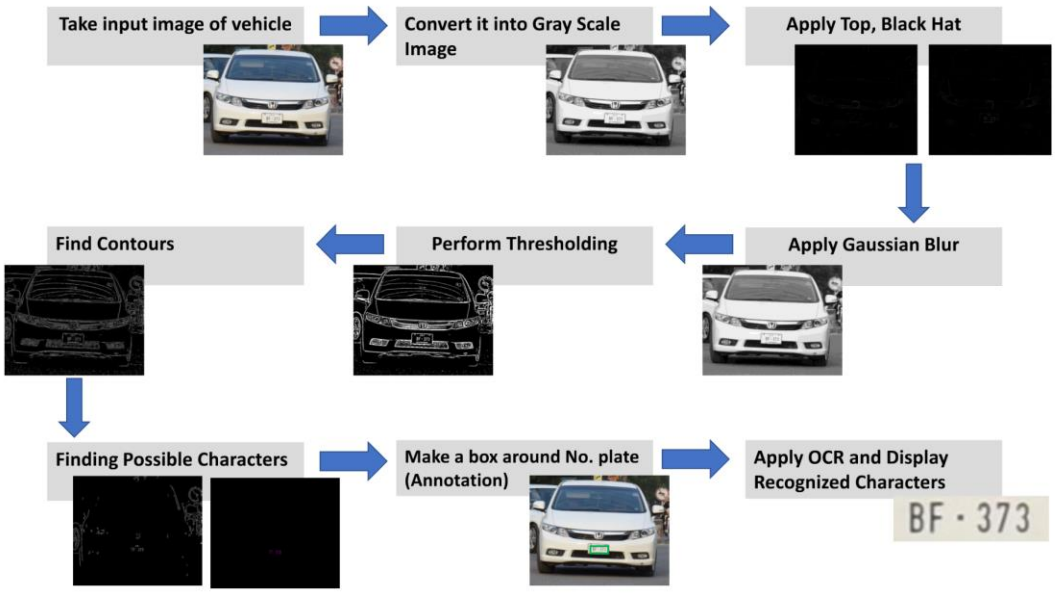


Figure 12: Localization of Number Plate

3.5.4.1 Challenges faced in the Detection of License Plates

As its very important to have accurately localized number plate of the vehicle for our system's accuracy. There are different variations of number plates or environments causes challenges in the Localization and recognition of Characters. [5] These challenges are mentioned as follow:

A. Plate variations:

1. Number plates can exist anywhere in the frame.
2. There could be multiple number of plates in one frame.
3. There are different sizes of number plates as there is no standard size. Also, the size of number plate varies in different type of vehicles (i.e. Car, Motor cycle, Bus, Truck).
4. Color of number plate varies. And also, the color of characters written on the plate are different. As there are different color of numbers plate for different provinces.
5. There is no standard font or font size for the characters on the number plate.

6. The standard license plate of ICT(Islamabad), can be seen in fig. 13(b). Fancy/fake license plates may have any number of characters without any regulations, as shown in Fig. 13(a).
7. Number plates can be covered by dirt.
8. Number plates may be tilted or not placed correctly.
9. Number plate may have frames and screws in addition to characters.

Figure 13 (a)Fancy/Fake number plate *(b)Standard number plate*

B. Environment variations:

1. Captured/selected frame of video stream may have clarity but can have different types of illumination primarily due to different lighting environmental or headlights of the vehicle.
2. The background of Captured/selected frame of video stream may contain patterns like plates, such as numbers printed on a vehicle, bumper with patterns on it and text written on car body.

3.5.5 OCR (Optical Character Recognition)

After the number plate is detected the next step is ko recognize the characters written on it. For this purpose, we pass the cropped number plate image to our algorithm. In which we first convert the image into gray scale and binarize the image. After wards the same identification data used to localize the number plate was used to identify the characters. We applied KNN technique which is discussed above.

We created the horizontal and vertical histograms of each character in the input image and also the identification as shown in the figure 14. By using these histograms values we applied KNN classifier and set the value of K to 3 and recognized the characters on the number plate.

Figure 14: Histogram of pixels

3.5.5.1 Temporal Redundancy

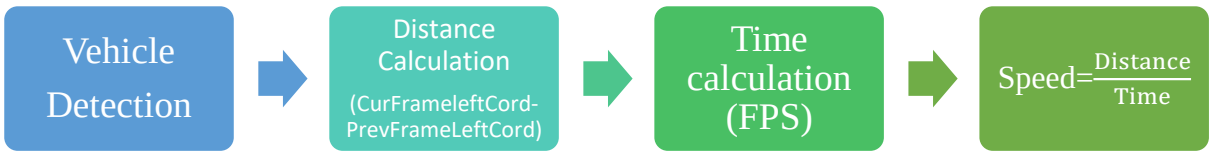
The characters written on the number plate are of many different fonts and sizes. And there are many characters which look alike and may have very close histogram values like the character “8” and “B”, or the character “0” and “D” etc. we get variety of registration numbers for a single vehicle as the number plate is localized continuously and OCR is being applied on every frame until the vehicle has passed the camera. To overcome this anomaly, we used a technique called temporal redundancy.

For this purpose, we made a queue and saved all the registration numbers extracted of a single vehicle and then checked the probability of each character and selected the most frequent one. This helped us improve the accuracy of OCR.

3.5.6 Speed Monitoring

Another main part of our project is to record the speed of each vehicle. As the vehicle is been detected in each frame. The location of the vehicle in each frame is different as the vehicle is moving, we have to calculate the change in the location of the vehicle in each frame in order to calculate the distance. As we know the coordinates of the bounding box around the vehicle, we simply subtracted the coordinates from the previous frame from the coordinates of the current frame. We saved these subtracted values in a list for each vehicle and then calculated their average to get the approximate distance. For the time we

knew that our video is 30 FPS (30 frames per second). This means that one frame is processed in 0.033 seconds. As we have both the values i.e. distance and the time. We will apply simple formula of speed which is distance over time.



3.5.7 Storing the data

we used MongoDB database to store all the extracted information which includes the type of vehicle, color, registration number the time at which the vehicle is passed, the location/ name of the highway where this system is installed and the cropped picture of number plate. The reason why we used MongoDB is that it is faster in saving and retrieving data and because of almost no constrains like SQL. It stores data as JSON-like documents.

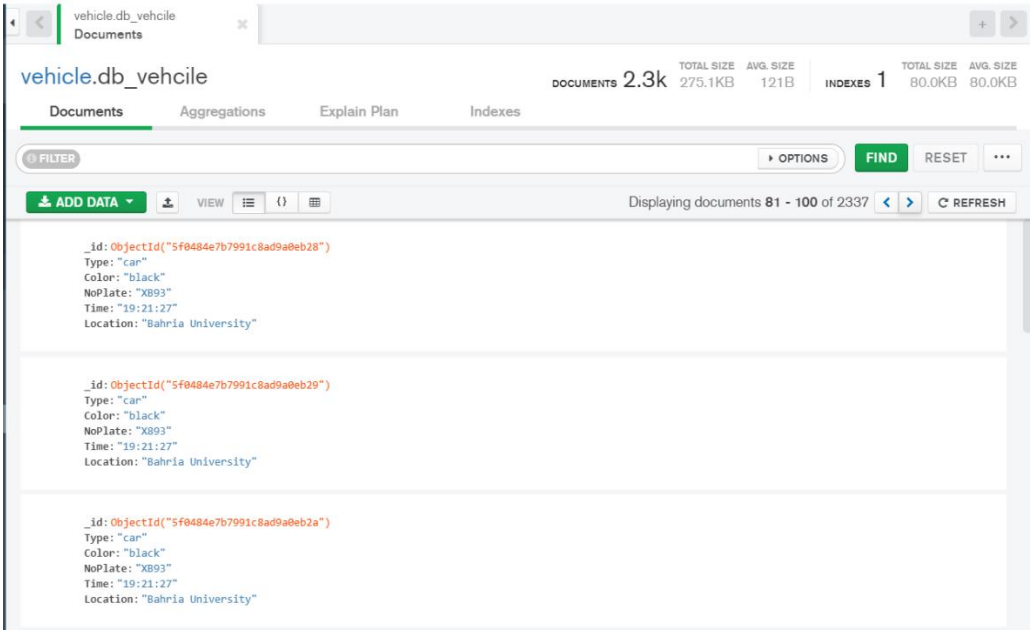


Figure 15: Vehicle Database

3.5.8 Graphical User Interface (GUI)

To show all the process and all the extracted information we created a graphical user interface GUI. We designed this GUI using Qt designer and implemented in our project using PyQt5 python library. This GUI includes the widget in which the live video stream is shown. It also shows the bounding boxes on the around detected vehicles in the stream. It also has a label in which the detected number plate is been shown. And then there are some labels in which it shows the type of the vehicle, its color, its registration number, the time at which it was passed and the location where this system is installed.

We also embedded our database with it. In which can access and get required data from the dataset by using advanced search. We applied mongo DB queries to retrieve the required data from our database and showed it in an organized tabular form.

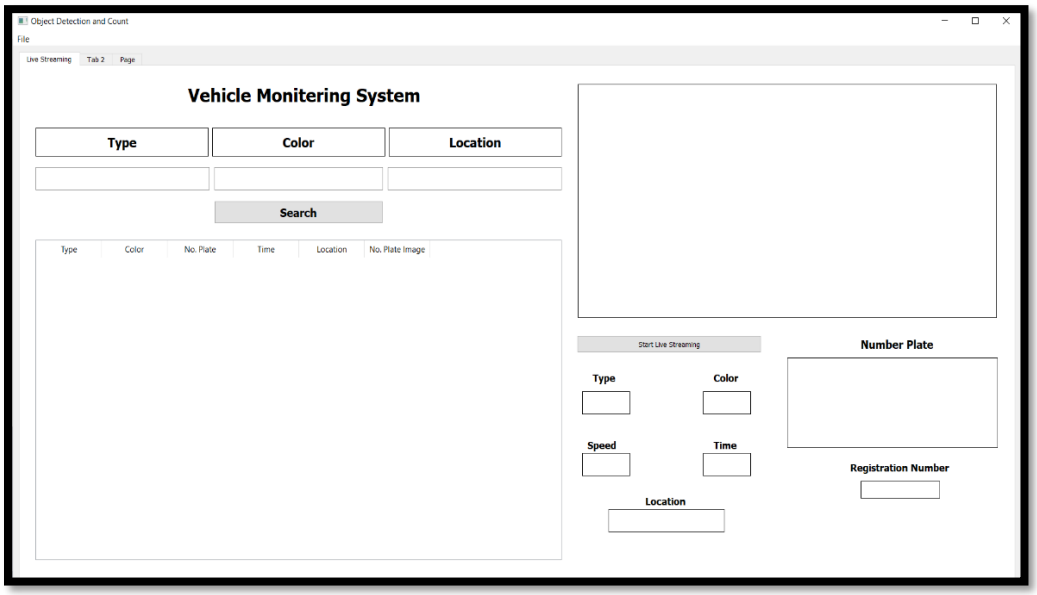


Figure 16: GUI Vehicle Monitoring System

CHAPTER 4

Results

In this section, we conducted experiments to verify the effectiveness of the proposed system.

Vehicle Detection:

We trained our model using a pre-trained model on SSD MobileNet V1, which was trained on coco dataset. We used transfer learning and trained the model on our dataset so it can detect vehicles. We trained our model on Google Colab, which gives variety of option, it provides you to select whether you want to use python 2 or python 3 and it also give you the option to use GPU or TPU. We used python 3 and GPU to train our model, Google colab provides 12GB NVIDIA Tesla K80 GPU and 13 Gbs or RAM.
Our dataset contains images of cars, buses, trucks, and bikes We used object detection algorithm, and use above mentioned trained model for detection of these vehicles. Depending on our dataset It took us 12-14 hours to train our model. We gained 97.23% accuracy.

Color detection:

After training our vehicle detection model, we build our color detection model, we used KNN classifier to detect vehicle colors, we trained that model on local machine which has Nvidia 1050Ti GPU and 8 GBs of RAM. we extracted the ranges of intensities of RGB colors using MATLAB and trained our model on that data and gave it a test dataset of different values of RGB intensities. We got 78.47% accuracy. But the accuracy varies when there is light reflecting on the vehicle because reflection changes the color completed.

License Plate:

From the cropped image of detected vehicle, then we localized the number plate of vehicle by applying several image processing techniques and cropped it. Then applied OCR on it.

OCR:

We applied OCR on the cropped license plate which is KNN based and the working of OCR is mentioned above in section 3, the character recognition depends upon weightages.

Improvement:

After applying all the above-mentioned procedures, still our overall accuracy was a bit low when we tested on a live video stream. Then we figured there was blurriness in the selected frames. We applied Laplacian filter on the obtained frames and by setting a particular Laplacian value we were able to deselect blurred frames.(Code attached in appendix 2) We set a threshold for it, that if the Laplacian value of the input frame at that time is higher than a certain value , then it means that the frame is clear enough on which we can perform the rest of the operations, like color detection, license plate detection and OCR otherwise we can eliminate that frame. That's how we increased our overall accuracy.

Live video stream:

We are getting the live video stream for the camera mentioned in the section 3, to get the maximum accurate results. According to the demand of industry the camera must be mounted on the roadside matching the level of the vehicle. In such a way that it only focusses on the front view of the vehicle. By doing so we will be able to get a clear view of the vehicle number plate. Also, by doing so the reflection due to sun light will be minimized and it will have least effect on the color on the vehicle. This system is just suggested for single lane operation.

Dataset:

The images were collected from internet and some organizations which kept record of the vehicles. The average image size of our dataset was 960 x 734, and the largest image resolution was 3424 x 2428. After collecting all the pictures, we resized the images to a standard resolution of 800x600. The code is also attached in the appendix 1.

Limitations

The limitations of our project are, the accuracy of color detection, that it varies under different light conditions due to which colors belonging from same intensity group might not be detected accurately. Second limitation is that there are many different fonts and sizes of characters on the license plate of the vehicle, due to which the accuracy of character recognition varies.

CHAPTER 5

Conclusion

In this thesis paper, we have presented a robust real-time end-to-end Vehicle Detection, speed monitoring and Number Plate recognition system by using the state-of-the-art SSD object detection model. We trained our own network for Vehicle Detection and Number Plate recognition stage.

This paper provided a concise study of recent deep learning approaches employed in Number Plate Recognition systems. System consists of five major steps, including vehicle detection, Color detection, Number Plate Detection, Optical Character Recognition (OCR) and saving the data into database and some recent studies have focused on each step independently. In this regard, we reviewed these researches based on the application of deep learning techniques in them. In our project we used Deep learning Neural Networks techniques to achieve the required results. We used SSD objection detection model to detect the vehicles through a live video stream, we used KNN classifier for the detection of the color of the vehicle and for the character recognition on the license plate. It is an efficient system which detects vehicle, its color, its license plate, and recognizes it through a live video stream, as future work, the method can be improved further by incorporating super-resolution techniques on the obtained multiple low-resolution instances of the number plate for enhanced recognition. Additionally, we plan to explore the vehicle's manufacturer and model in the Vehicle Detection and Number Plate recognition pipeline. Even though our system was conceived and evaluated on one country-specific dataset from Pakistan, we believe that the proposed system is robust to detect Vehicle, detect its color, localize its Number plate, recognize character written on it in variety of fonts, record its speed and save all the information in a searchable database in correct format which can also we be accessed later for further use.

REFERENCES

[1] Du, S., Ibrahim, M., Shehata, M. and Badawy, W., 2012. Automatic license plate recognition (ALPR): A state-of-the-art review. *IEEE Transactions on circuits and systems for video technology*, *23*(2), pp.311-325.

[2] Gou, C., Wang, K., Yao, Y. and Li, Z., 2015. Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines. *IEEE Transactions on Intelligent Transportation Systems*, *17*(4), pp.1096-1107.

[3] O. Bulan, V. Kozitsky, P. Ramesh, and M. Shreve, "Segmentation and annotation-free license plate recognition with deep localization and failure identification," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 9, pp. 2351–2363, Sept 2017

[4] Laroca, R., Severo, E., Zanlorensi, L.A., Oliveira, L.S., Gonçalves, G.R., Schwartz, W.R. and Menotti, D., 2018, July. A robust real-time automatic license plate recognition based on the YOLO detector. In *2018 international joint conference on neural networks (ijcnn)* (pp. 1-10). IEEE.

[5] Li, H. and Shen, C., 2016. Reading car license plates using deep convolutional neural networks and lstms. *arXiv preprint arXiv:1601.05610*.

[6] Li, H., Wang, P. and Shen, C., 2018. Toward end-to-end car license plate detection and recognition with deep neural networks. *IEEE Transactions on Intelligent Transportation Systems*, *20*(3), pp.1126-1136.

[7] P. Soucy and G. W. Mineau, "A simple KNN algorithm for text categorization," Proceedings 2001 IEEE International Conference on Data Mining, San Jose, CA, USA, 2001, pp. 647-648, doi: 10.1109/ICDM.2001.989592

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Advances in Neural Information Processing Systems (NIPS), 2015.

[9] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," arXiv preprint arXiv:1612.08242, 2016.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, A. C. Berg, SSD: Single Shot MultiBox

[11] Ning, C., Zhou, H., Song, Y. and Tang, J., 2017, July. Inception single shot multibox detector for object detection. In 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW) (pp. 549-554). IEEE.

[12] Cai Z., Fan Q., Feris R.S., Vasconcelos N. (2016) A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9908. Springer, Cham

[13] Chen, K.H., Shou, T.D., Li, J.K.H. and Tsai, C.M., 2018, July. Vehicles detection on expressway via deep learning: Single shot multibox object detector. In *2018 International Conference on Machine Learning and Cybernetics (ICMLC)* (Vol. 2, pp. 467-473). IEEE.

[14] S. Ren, K. He, R Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks., " IEEE Transactions on Pattern Analysis & Machine Intelligence, pp. 1-1, 2016.

[15] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition, " Computer Science, 2015.

[16] Single Shot MultiBox Detector for Vehicles and Pedestrians Detection and Classification Qiong WU, Sheng-bin LIAO

[17] Miao, F., Tian, Y. and Jin, L., 2019, August. Vehicle Direction Detection Based on YOLOv3. In *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)* (Vol. 2, pp. 268-271). IEEE.

[18] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[19] Wang, Y., Wang, C. and Zhang, H., 2018. Combining a single shot multibox detector with transfer learning for ship detection using sentinel-1 SAR images. *Remote sensing letters*, *9*(8), pp.780-788.

APPENDIX

Appendix 1: Resizing of the images

```
import os
import glob
import cv2
if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser(
        description ="Resize raw images to uniformed target size."
    )
    parser.add_argument(
        "--raw - dir",
        help= "Directory path to raw images.",
        default ="./data/raw",
        type=str ,
    )
    parser.add_argument(
        "--save - dir",
        help= "Directory path to save resized images.",
        default ="./data/images",
        type=str ,
    )
    parser.add_argument(
        "--ext", help ="Raw image files extension to resize.",
default="jpg", type=str
    )
    parser.add_argument(
        "--target-size",
        help="Target size to resize as a tuple of 2 integers.",
        default="(800, 600)",
        type=str,
    )
    args = parser.parse_args()
    raw_dir = args.raw_dir
    save_dir = args.save_dir
    ext = args.ext
    target_size = eval(args.target_size)
    msg = "--target-size must be a tuple of 2 integers"
    assert isinstance(target_size, tuple) and len(target_size) == 2,
msg
    fnames = glob.glob  (os.path. join(raw_ dir, "*.{}".format(ext)))
    os. makedirs(save_dir, exist _ok=True)
    print(
```

```
        "{} files to resize from directory `{}` to target
size:{}".format(
            len(fnames), raw_dir, target_size
        )
    )
    for i, fname in enumerate(fnames):
        print(".", end="", flush=True)
        img = cv2.imread(f import os
import glob
import cv2
if __name__ == "__main__" :
    import argparse

    parser = argparse.ArgumentParser  (
        description = "Resize raw images to uniformed target size."
    )
    parser.add_argument(
        "--raw-dir",
        help="Directory path to raw images.",
        default="./data/raw",
        type=str,
    )
    parser.add_argument(
        "--save-dir",
        help="Directory path to save resized images.",
        default="./data/images",
        type=str,
    )
    parser.add_argument(
        "--ext", help="Raw image files extension to resize.",
default="jpg", type=str
    )
    parser.add_argument(
        "--target-size",
        help="Target size to resize as a tuple of 2 integers.",
        default="(800, 600)",
        type=str,
    )
    args = parser.parse_args()

    raw_dir = args.raw_dir
    save_dir = args.save_dir
    ext = args.ext
    target_size = eval(args.target_size)
    msg = "--target-size must be a tuple of 2 integers"
    assert isinstance(target_size, tuple) and len(target_size) == 2,
msg
    fnames = glob.glob(os.path.join(raw_dir, "*.{}".format(ext)))
    os. makedirs  (save_dir, exist_ok = True)
```

```
    print(
        "{} files to resize from directory `{}` to target
size:{}".format(
            len(fnames), raw_dir, target_size
        )
    )
    for i, fname in enumerate(fnames):
        print(".", end="", flush=True)
        img = cv2.imread(fname)
        img_small = cv2.resize(img, target_size)
        new_fname = "{}.{}".format(str(i), ext)
        small_fname = os.path.join(save_dir, new_fname)
        cv2.imwrite(small_fname, img_small)
    print(
        "\nDone resizing {} files.\nSaved to directory:
`{}`".format(
            len(fnames), save_dir
```

Appendix 2: Laplacian Filter

```
import cv2
import numpy as np

#img = cv2.imread("0.jpg", cv2.IMREAD_GRAYSCALE)


def blurdetect(img):
    laplacian_var = cv2.Laplacian(img, cv2.CV_64F).var()
    # if laplacian_var < 5:
    #     print("Image blurry")

    #print(laplacian_var)
    return (laplacian_var)
```

Appendix 3: Main File + GUI code

```
import cv2
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
import os.path
import time
import pprint

import vlc
import tensorflow as tf
import csv
import numpy as np
from utils import backbone
from utils import visualization_utils as vis_util

from pymongo import MongoClient
#creating DB
democlient = MongoClient()
myclient = MongoClient('localhost',27017)
mydb = myclient["vehicle"]
mycoll=mydb["db_vehcile"]


t=vis_util.typee
c=vis_util.colorr
nom=str(vis_util.n)
#current_time="0"
#Locationnn="None"
#f=0


class Ui_VideoPlayer(object):
    def setupUi(self, VideoPlayer):
        self.instance = vlc.Instance()

        self.mediaplayer = self.instance.media_player_new()
        self.isPaused = False
        VideoPlayer.setObjectName("VideoPlayer")
        VideoPlayer.resize(1920, 1050)
        self.centralwidget = QtWidgets.QWidget(VideoPlayer)
        self.centralwidget.setObjectName("centralwidget")
        self.horizontalLayout_2 =
QtWidgets.QHBoxLayout(self.centralwidget)
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
```

```
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base,
brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(240, 240, 240))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.Base, brush)
        self.tabWidget.setPalette(palette)
        self.tabWidget.setObjectName("tabWidget")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.frame = QtWidgets.QFrame(self.tab)
        self.frame.setGeometry(QtCore.QRect(1050, 20, 821, 441))
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.frame)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.LiveVideStreamFrame = QtWidgets.QLabel(self.frame)
        self.LiveVideStreamFrame.setFrameShape(QtWidgets.QFrame.Box)
        self.LiveVideStreamFrame.setText("")

self.LiveVideStreamFrame.setObjectName("LiveVideStreamFrame")
        self.horizontalLayout.addWidget(self.LiveVideStreamFrame)
        self.pushButton = QtWidgets.QPushButton(self.tab)
        self.pushButton.setGeometry(QtCore.QRect(1060, 480, 351,
31))
        self.pushButton.setObjectName("pushButton")
        self.label = QtWidgets.QLabel(self.tab)
        self.label.setGeometry(QtCore.QRect(1460, 520, 401, 161))
        self.label.setFrameShape(QtWidgets.QFrame.Box)
        self.label.setText("")
        self.label.setScaledContents(True)
        self.label.setObjectName("label")
        self.label_16 = QtWidgets.QLabel(self.tab)
        self.label_16.setGeometry(QtCore.QRect(1600, 480, 171, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.label_16.setFont(font)
        self.label_16.setObjectName("label_16")
        self.label_17 = QtWidgets.QLabel(self.tab)
        self.label_17.setGeometry(QtCore.QRect(1580, 700, 191, 31))
```

```
font = QtGui.QFont()
font.setPointSize(11)
font.setBold(True)
font.setWeight(75)
self.label_17.setFont(font)
self.label_17.setObjectName("label_17")
self.label_18 = QtWidgets.QLabel(self.tab)
self.label_18.setGeometry(QtCore.QRect(1600, 740, 151, 31))
self.label_18.setFrameShape(QtWidgets.QFrame.Box)
self.label_18.setText("")
self.label_18.setAlignment(QtCore.Qt.AlignCenter)
self.label_18.setObjectName("label_18")
self.label_19 = QtWidgets.QLabel(self.tab)
self.label_19.setGeometry(QtCore.QRect(1090, 540, 91, 31))
font = QtGui.QFont()
font.setPointSize(11)
font.setBold(True)
font.setWeight(75)
self.label_19.setFont(font)
self.label_19.setObjectName("label_19")
self.label_20 = QtWidgets.QLabel(self.tab)
self.label_20.setGeometry(QtCore.QRect(1320, 540, 91, 31))
font = QtGui.QFont()
font.setPointSize(11)
font.setBold(True)
font.setWeight(75)
self.label_20.setFont(font)
self.label_20.setObjectName("label_20")
self.label_21 = QtWidgets.QLabel(self.tab)
self.label_21.setGeometry(QtCore.QRect(1080, 660, 91, 31))
font = QtGui.QFont()
font.setPointSize(11)
font.setBold(True)
font.setWeight(75)
self.label_21.setFont(font)
self.label_21.setObjectName("label_21")
self.label_22 = QtWidgets.QLabel(self.tab)
self.label_22.setGeometry(QtCore.QRect(1190, 760, 91, 31))
font = QtGui.QFont()
font.setPointSize(11)
font.setBold(True)
font.setWeight(75)
self.label_22.setFont(font)
self.label_22.setObjectName("label_22")
self.label_23 = QtWidgets.QLabel(self.tab)
self.label_23.setGeometry(QtCore.QRect(1320, 660, 91, 31))
font = QtGui.QFont()
font.setPointSize(11)
font.setBold(True)
font.setWeight(75)
```

```
        self.label_23.setFont(font)
        self.label_23.setObjectName("label_23")
        self.type = QtWidgets.QLabel(self.tab)
        self.type.setGeometry(QtCore.QRect(1070, 580, 91, 41))
        self.type.setFrameShape(QtWidgets.QFrame.Box)
        self.type.setText("")
        self.type.setAlignment(QtCore.Qt.AlignCenter)
        self.type.setObjectName("type")
        self.color = QtWidgets.QLabel(self.tab)
        self.color.setGeometry(QtCore.QRect(1300, 580, 91, 41))
        self.color.setFrameShape(QtWidgets.QFrame.Box)
        self.color.setText("")
        self.color.setAlignment(QtCore.Qt.AlignCenter)
        self.color.setObjectName("color")
        self.location = QtWidgets.QLabel(self.tab)
        self.location.setGeometry(QtCore.QRect(1120, 790, 221, 41))
        self.location.setFrameShape(QtWidgets.QFrame.Box)
        self.location.setText("")
        self.location.setAlignment(QtCore.Qt.AlignCenter)
        self.location.setObjectName("location")
        self.speed = QtWidgets.QLabel(self.tab)
        self.speed.setGeometry(QtCore.QRect(1070, 690, 91, 41))
        self.speed.setFrameShape(QtWidgets.QFrame.Box)
        self.speed.setText("")
        self.speed.setAlignment(QtCore.Qt.AlignCenter)
        self.speed.setObjectName("speed")
        self.time = QtWidgets.QLabel(self.tab)
        self.time.setGeometry(QtCore.QRect(1300, 690, 91, 41))
        self.time.setFrameShape(QtWidgets.QFrame.Box)
        self.time.setText("")
        self.time.setAlignment(QtCore.Qt.AlignCenter)
        self.time.setObjectName("time")
        self.label_15 = QtWidgets.QLabel(self.tab)
        self.label_15.setGeometry(QtCore.QRect(320, 20, 491, 61))
        font = QtGui.QFont()
        font.setPointSize(20)
        font.setBold(True)
        font.setWeight(75)
        self.label_15.setFont(font)
        self.label_15.setObjectName("label_15")
        self.horizontalLayoutWidget = QtWidgets.QWidget(self.tab)
        self.horizontalLayoutWidget.setGeometry(QtCore.QRect(30,
110, 1001, 51))

self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")
        self.horizontalLayout_4 =
QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)
        self.horizontalLayout_4.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout_4.setObjectName("horizontalLayout_4")
```

```
        self.label_27 =
QtWidgets.QLabel(self.horizontalLayoutWidget)
        font = QtGui.QFont()
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.label_27.setFont(font)
        self.label_27.setFrameShape(QtWidgets.QFrame.Box)
        self.label_27.setAlignment(QtCore.Qt.AlignCenter)
        self.label_27.setObjectName("label_27")
        self.horizontalLayout_4.addWidget(self.label_27)
        self.label_26 =
QtWidgets.QLabel(self.horizontalLayoutWidget)
        font = QtGui.QFont()
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.label_26.setFont(font)
        self.label_26.setFrameShape(QtWidgets.QFrame.Box)
        self.label_26.setAlignment(QtCore.Qt.AlignCenter)
        self.label_26.setObjectName("label_26")
        self.horizontalLayout_4.addWidget(self.label_26)
        self.label_30 =
QtWidgets.QLabel(self.horizontalLayoutWidget)
        font = QtGui.QFont()
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.label_30.setFont(font)
        self.label_30.setFrameShape(QtWidgets.QFrame.Box)
        self.label_30.setAlignment(QtCore.Qt.AlignCenter)
        self.label_30.setObjectName("label_30")
        self.horizontalLayout_4.addWidget(self.label_30)
        self.pushButton_2 = QtWidgets.QPushButton(self.tab)
        self.pushButton_2.setGeometry(QtCore.QRect(370, 240, 321,
41))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setObjectName("pushButton_2")
        self.tableWidget = QtWidgets.QTableWidget(self.tab)
        self.tableWidget.setGeometry(QtCore.QRect(30, 310, 1001,
571))
        self.tableWidget.setObjectName("tableWidget")
        self.tableWidget.setColumnCount(5)
        self.tableWidget.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
```

```
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(3, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(4, item)
        self.fromtime = QtWidgets.QLineEdit(self.tab)
        self.fromtime.setGeometry(QtCore.QRect(700, 180, 331, 41))
        self.fromtime.setObjectName("fromtime")
        self.typeser = QtWidgets.QLineEdit(self.tab)
        self.typeser.setGeometry(QtCore.QRect(370, 180, 321, 41))
        self.typeser.setObjectName("typeser")
        self.colorser = QtWidgets.QLineEdit(self.tab)
        self.colorser.setGeometry(QtCore.QRect(30, 180, 331, 41))
        self.colorser.setObjectName("colorser")
        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.frame_2 = QtWidgets.QFrame(self.tab_2)
        self.frame_2.setGeometry(QtCore.QRect(10, 10, 231, 411))
        self.frame_2.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_2.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_2.setObjectName("frame_2")
        self.horizontalLayout_3 =
QtWidgets.QHBoxLayout(self.frame_2)
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        self.FilelistWidget = QtWidgets.QListWidget(self.frame_2)
        self.FilelistWidget.setObjectName("FilelistWidget")
```

```
        self.horizontalLayout_3.addWidget(self.FilelistWidget)
        self.Video_frame = QtWidgets.QFrame(self.tab_2)
        self.Video_frame.setGeometry(QtCore.QRect(250, 10, 481,
371))
        self.Video_frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.Video_frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.Video_frame.setObjectName("Video_frame")
        self.PlayPauseButton = QtWidgets.QPushButton(self.tab_2)
        self.PlayPauseButton.setGeometry(QtCore.QRect(250, 390, 61,
23))
        self.PlayPauseButton.setObjectName("PlayPauseButton")
        self.StopButton = QtWidgets.QPushButton(self.tab_2)
        self.StopButton.setGeometry(QtCore.QRect(320, 390, 61, 23))
        self.StopButton.setObjectName("StopButton")
        self.videohorizontalSlider = QtWidgets.QSlider(self.tab_2)
        self.videohorizontalSlider.setGeometry(QtCore.QRect(390,
390, 341, 22))

self.videohorizontalSlider.setOrientation(QtCore.Qt.Horizontal)

self.videohorizontalSlider.setObjectName("videohorizontalSlider")
        self.tabWidget.addTab(self.tab_2, "")
        self.tab_4 = QtWidgets.QWidget()
        self.tab_4.setObjectName("tab_4")
        self.label_2 = QtWidgets.QLabel(self.tab_4)
        self.label_2.setGeometry(QtCore.QRect(20, 20, 311, 151))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.tab_4)
        self.label_3.setGeometry(QtCore.QRect(400, 20, 61, 21))
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.tab_4)
        self.label_4.setGeometry(QtCore.QRect(400, 50, 55, 16))
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(self.tab_4)
        self.label_5.setGeometry(QtCore.QRect(570, 20, 55, 16))
        self.label_5.setObjectName("label_5")
        self.label_6 = QtWidgets.QLabel(self.tab_4)
        self.label_6.setGeometry(QtCore.QRect(570, 40, 55, 16))
        self.label_6.setObjectName("label_6")
        self.label_7 = QtWidgets.QLabel(self.tab_4)
        self.label_7.setGeometry(QtCore.QRect(710, 20, 55, 16))
        self.label_7.setObjectName("label_7")
        self.label_8 = QtWidgets.QLabel(self.tab_4)
        self.label_8.setGeometry(QtCore.QRect(710, 40, 55, 16))
        self.label_8.setObjectName("label_8")
        self.label_9 = QtWidgets.QLabel(self.tab_4)
        self.label_9.setGeometry(QtCore.QRect(400, 100, 55, 16))
        self.label_9.setObjectName("label_9")
        self.label_10 = QtWidgets.QLabel(self.tab_4)
        self.label_10.setGeometry(QtCore.QRect(400, 130, 55, 16))
```

```
        self.label_10.setObjectName("label_10")
        self.label_11 = QtWidgets.QLabel(self.tab_4)
        self.label_11.setGeometry(QtCore.QRect(570, 100, 55, 16))
        self.label_11.setObjectName("label_11")
        self.label_12 = QtWidgets.QLabel(self.tab_4)
        self.label_12.setGeometry(QtCore.QRect(570, 120, 55, 16))
        self.label_12.setObjectName("label_12")
        self.label_13 = QtWidgets.QLabel(self.tab_4)
        self.label_13.setGeometry(QtCore.QRect(710, 100, 55, 16))
        self.label_13.setObjectName("label_13")
        self.label_14 = QtWidgets.QLabel(self.tab_4)
        self.label_14.setGeometry(QtCore.QRect(710, 120, 55, 16))
        self.label_14.setObjectName("label_14")
        self.tabWidget.addTab(self.tab_4, "")
        self.horizontalLayout_2.addWidget(self.tabWidget)
        VideoPlayer.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(VideoPlayer)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1920, 26))
        self.menubar.setObjectName("menubar")
        self.menuFile = QtWidgets.QMenu(self.menubar)
        self.menuFile.setObjectName("menuFile")
        VideoPlayer.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(VideoPlayer)
        self.statusbar.setObjectName("statusbar")
        VideoPlayer.setStatusBar(self.statusbar)
        self.actionOpen = QtWidgets.QAction(VideoPlayer)
        self.actionOpen.setObjectName("actionOpen")
        self.actionExit = QtWidgets.QAction(VideoPlayer)
        self.actionExit.setObjectName("actionExit")
        self.actionAdd_Channels = QtWidgets.QAction(VideoPlayer)
        self.actionAdd_Channels.setObjectName("actionAdd_Channels")
        self.menuFile.addAction(self.actionOpen)
        self.menuFile.addAction(self.actionAdd_Channels)
        self.menuFile.addAction(self.actionExit)
        self.menubar.addAction(self.menuFile.menuAction())

        self.retranslateUi(VideoPlayer)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(VideoPlayer)


        self.imgpath =
"E:/University/FYP/All_Codes/DEFENCE/do_model,W_O-
M,clear_GUI/data/1.jpg"

        self.timer = QtCore.QTimer()
        self.timer.setInterval(200)
        self.timer.timeout.connect(self.updateUI)
        self.pushButton.clicked.connect(self.setup_liveStreaming)
        self.actionOpen.triggered.connect(self.viewFilesList)
```

```
        self.actionExit.triggered.connect(self.exit)
        self.FilelistWidget.itemClicked.connect(self.OpenFile)
        self.PlayPauseButton.clicked.connect(self.PlayPause)

self.videohorizontalSlider.sliderMoved.connect(self.setPosition)
        self.StopButton.clicked.connect(self.Stop)
        self.pushButton_2.clicked.connect(self.searchh)

    def retranslateUi(self, VideoPlayer):
        _translate = QtCore.QCoreApplication.translate
        VideoPlayer.setWindowTitle(_translate("VideoPlayer", "Object
Detection and Count"))
        self.pushButton.setText(_translate("VideoPlayer", "Start
Live Streaming"))
        self.label_16.setText(_translate("VideoPlayer", "Number
Plate"))
        self.label_17.setText(_translate("VideoPlayer",
"Registration Number"))
        self.label_19.setText(_translate("VideoPlayer", "Type"))
        self.label_20.setText(_translate("VideoPlayer", "Color"))
        self.label_21.setText(_translate("VideoPlayer", "Speed"))
        self.label_22.setText(_translate("VideoPlayer", "Location"))
        self.label_23.setText(_translate("VideoPlayer", "Time"))
        self.label_15.setText(_translate("VideoPlayer", "Vehicle
Monitering System"))
        self.label_27.setText(_translate("VideoPlayer", "Type"))
        self.label_26.setText(_translate("VideoPlayer", "Color"))
        self.label_30.setText(_translate("VideoPlayer", "Location"))
        self.pushButton_2.setText(_translate("VideoPlayer",
"Search"))
        item = self.tableWidget.horizontalHeaderItem(0)
        item.setText(_translate("VideoPlayer", "Type"))
        item = self.tableWidget.horizontalHeaderItem(1)
        item.setText(_translate("VideoPlayer", "Color"))
        item = self.tableWidget.horizontalHeaderItem(2)
        item.setText(_translate("VideoPlayer", "No. Plate"))
        item = self.tableWidget.horizontalHeaderItem(3)
        item.setText(_translate("VideoPlayer", "Time"))
        item = self.tableWidget.horizontalHeaderItem(4)
        item.setText(_translate("VideoPlayer", "Location"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
_translate("VideoPlayer", "Live Streaming"))
        self.PlayPauseButton.setText(_translate("VideoPlayer",
"Play"))
        self.StopButton.setText(_translate("VideoPlayer", "Stop"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2),
_translate("VideoPlayer", "Tab 2"))
        self.label_2.setText(_translate("VideoPlayer", "TextLabel"))
        self.label_3.setText(_translate("VideoPlayer", "TextLabel"))
```

```
        self.label_4.setText(_translate("VideoPlayer", "TextLabel"))
        self.label_5.setText(_translate("VideoPlayer", "TextLabel"))
        self.label_6.setText(_translate("VideoPlayer", "TextLabel"))
        self.label_7.setText(_translate("VideoPlayer", "TextLabel"))
        self.label_8.setText(_translate("VideoPlayer", "TextLabel"))
        self.label_9.setText(_translate("VideoPlayer", "TextLabel"))
        self.label_10.setText(_translate("VideoPlayer",
"TextLabel"))
        self.label_11.setText(_translate("VideoPlayer",
"TextLabel"))
        self.label_12.setText(_translate("VideoPlayer",
"TextLabel"))
        self.label_13.setText(_translate("VideoPlayer",
"TextLabel"))
        self.label_14.setText(_translate("VideoPlayer",
"TextLabel"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_4),
_translate("VideoPlayer", "Page"))
        self.menuFile.setTitle(_translate("VideoPlayer", "File"))
        self.actionOpen.setText(_translate("VideoPlayer", "Open"))
        self.actionExit.setText(_translate("VideoPlayer", "Exit"))
        self.actionAdd_Channels.setText(_translate("VideoPlayer",
"Add Channels"))




    def PlayPause(self):

        if self.mediaplayer.is_playing():
            self.mediaplayer.pause()
            self.PlayPauseButton.setText("Play")
            self.isPaused = True
        else:
            if self.mediaplayer.play() == -1:
                self.OpenFile()
                return
            self.mediaplayer.play()
            self.PlayPauseButton.setText("Pause")
            self.timer.start()
            self.isPaused = False

    def Stop(self):

        self.mediaplayer.stop()
        self.PlayPauseButton.setText("Play")

    def OpenFile(self, item):
```

```
        self.filename = self.path + "/" + item.text()

        if sys.version < '3':
            self.filename = unicode(self.filename)
        self.media = self.instance.media_new(self.filename)

        self.mediaplayer.set_media(self.media)


        self.media.parse()


        if sys.platform.startswith('linux'):
            self.mediaplayer.set_xwindow(self.Video_frame.winId())
        elif sys.platform == "win32":
            self.mediaplayer.set_hwnd(self.Video_frame.winId())
        elif sys.platform == "darwin":

self.mediaplayer.set_nsobject(int(self.Video_frame.winId()))
        self.PlayPause()

    def setPosition(self, position):

        self.mediaplayer.set_position(position / 1000.0)


    def updateUI(self):


self.videohorizontalSlider.setValue(self.mediaplayer.get_position()
* 1000)

        if not self.mediaplayer.is_playing():

            self.timer.stop()
            if not self.isPaused:

                self.Stop()







    def setup_liveStreaming(self):
        # self.capture = cv2.VideoCapture(0)
        # self.capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        # self.capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
        # self.timer = QtCore.QTimer()
        # self.timer.timeout.connect(self.live_streaming)
```

```
        # self.timer.start(10)
        detection_graph, category_index =
backbone.set_model('model', 'label_map.pbtxt')
        is_color_recognition_enabled = 1
        total_passed_vehicle = 1
        speed = "waiting..."
        direction = "waiting..."
        size = "waiting..."
        color = "waiting..."
        counting_mode = "..."
        width_heigh_taken = True
        height = 0
        width = 0
        time.sleep(1)
        with detection_graph.as_default():
            with tf.Session(graph=detection_graph) as sess:

                image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')


                detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')


                detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
                detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')
                num_detections =
detection_graph.get_tensor_by_name('num_detections:0')
                self.cap =
cv2.VideoCapture("E:/University/FYP/All_Codes/DEFENCE/do_model,W_O-
M,clear_GUI/Videos/DSC_1917(e).mp4")



                f=0
                queueu=[]

                while (self.cap.isOpened()):

                    (ret, frame) = self.cap.read()

                    if not ret:
                        print("end of the video file...")
                        break

                    input_frame = frame
```

```
                    image_np_expanded = np.expand_dims(input_frame,
axis=0)


                    (boxes, scores, classes, num) = sess.run(
                        [detection_boxes, detection_scores,
detection_classes, num_detections],
                        feed_dict={image_tensor: image_np_expanded})


                    font = cv2.FONT_HERSHEY_SIMPLEX


                    #starttt=time.process_time()

                    counter, csv_line, counting_mode =
vis_util.visualize_boxes_and_labels_on_image_array(
                        self.cap.get(1),
                        input_frame,
                        1,
                        is_color_recognition_enabled,
                        np.squeeze(boxes),
                        np.squeeze(classes).astype(np.int32),
                        np.squeeze(scores),
                        category_index,
                        use_normalized_coordinates=True,
                        line_thickness=4)


                    #print(time.process_time() - starttt)



                    if (len(counting_mode) == 0):
                        cv2.putText(input_frame, "...", (10, 35),
font, 0.8, (0, 255, 255), 2, cv2.FONT_HERSHEY_SIMPLEX)
                    else:
                        cv2.putText(input_frame, counting_mode, (10,
35), font, 0.8, (0, 255, 255), 2,
                                    cv2.FONT_HERSHEY_SIMPLEX)

                    input_frame = cv2.resize(input_frame, (797,
417))
                    input_frame = cv2.cvtColor(input_frame,
cv2.COLOR_BGR2RGB)
                    image = QtGui.QImage(input_frame,
input_frame.shape[1], input_frame.shape[0],
                                         input_frame.strides[0],
QtGui.QImage.Format_RGB888)
```

```
self.LiveVideStreamFrame.setPixmap(QtGui.QPixmap.fromImage(image))
                    # print(image)

                    global t
                    global c
                    global nom
                    t = vis_util.typee
                    c = vis_util.colorr
                    nom = str(vis_util.n)

                    self.imageee = QtGui.QImage(self.imgpath)
                    self.pixmapimage =
QtGui.QPixmap.fromImage(self.imageee)
                    self.label.setPixmap(self.pixmapimage)

                    #global current_time
                    #global Locationnn

                    tim = time.localtime()
                    current_time = time.strftime("%H:%M:%S", tim)

                    Locationnn="Bahria University"




                    if "car" in t:
                        t="car"
                    elif "truck" in t:
                        t="truck"
                    elif "bus" in t:
                        t="bus"
                    elif "motorcycle" in t:
                        t="bike"




                    queueu.append(nom)

                    if len(queueu)>100:
                        queueu.pop(0)


                    #mostno=most_frequent(queueu)
                    counter = 0
                    num = queueu[0]

                    for i in queueu:
```

```
                        curr_frequency = queueu.count(i)
                        if(curr_frequency> counter):
                            counter = curr_frequency
                            num = i

                    mostno=num



                    # print("pro",mostno)
                    # print("num",nom)


                    # dblist = myclient.list_database_names()

                    # row_number=0
                    # column_number=0

                    # for alll in dblist:
                    #     #self.tabWidget.setTabText(row_number+1)
                    #     for dataa in alll:
                    #
cell=QtWidgets.QTableWidgetItem(str(dataa))
                    #
self.tabWidget.setTabText(row_number,alll)
                    #         column_number=column_number+1
                    #     row_number=row_number+1


                    # print(dblist)
                    # qt=QtWidgets.QTableWidgetItem(str(t))
                    # qc=QtWidgets.QTableWidgetItem(str(c))
                    # qnom=QtWidgets.QTableWidgetItem(str(nom))
                    #vbox=QVBoxLayout()
                    #rr=mydb.mycoll.count()


                    # d=mycoll.find_one()
                    # #pprint.pprint(d)
                    # lii=[]
                    # for dooo in d:
                    #     lii.append(dooo)

                    # #print(lii)
                    # yoo=0

                    # for o in lii:
                    #
self.tableWidget.setItem(1,yoo,QTableWidgetItem(str(lii(yoo+1))))
                    #     yoo=yoo+1
```

```
                    #vbox.addWidget(tabWidget)




                    # print(vis_util.noplatecropped)

                    # if vis_util.noplatecropped != None:
                    #     print(vis_util.noplatecropped)
                    #     cv2.imshow("m",vis_util.noplatecropped)
                    #     iimage =
QtGui.QImage(vis_util.noplatecropped,
vis_util.noplatecropped.shape[1], vis_util.noplatecropped.shape[0],
                    #
vis_util.noplatecropped.strides[0], QtGui.QImage.Format_RGB888)
                    #
self.label.setPixmap(QtGui.QPixmap.fromImage(iimage))

                    # else:
                    #     self.label.setText("no image")







                    self.type.setText(t)
                    self.color.setText(c)
                    self.label_18.setText(nom)
                    self.location.setText(Locationnn)
                    self.time.setText(str(current_time))



                    if t!="" and c!="" and nom!="" and
current_time!="" and Locationnn!="":
                        mylist = [
                            {
                             "Type": t,
                             "Color": c,
                             "NoPlate": nom,
                             "Time": current_time,
                             "Location": Locationnn,
```

```
                        }
                    ]

                    x = mycoll.insert_many(mylist)


                # self.tableWidget.setRowCount(3)
                #
self.tableWidget.setItem(1,0,QTableWidgetItem(str(t)))
                #
self.tableWidget.setItem(1,1,QTableWidgetItem(str(c)))
                #
self.tableWidget.setItem(1,2,QTableWidgetItem(str(nom)))



                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break




    def searchh(self):
        typedata=self.colorser.text()
        colordata=self.typeser.text()
        locationdata=self.fromtime.text()
        self.tableWidget.setRowCount(3)



        # imageg = QtGui.QPixmap(self.imgpath)
        # self.tableWidget.setItem(1, 5, QTableWidgetItem(imageg))
        lengthh=0
        down=1
        right=-1




        if typedata!="" and colordata=="" and locationdata=="":

            xx=mydb.db_vehcile.find ( { "Type" : typedata } )
            #x2=xx
            ttt=[]
            for items in xx:
                lengthh=lengthh+1
                ttt.append(items)

            #print(ttt)
```

```
            self.tableWidget.setRowCount(lengthh)


            for itemss in ttt:


self.tableWidget.setItem(down,0,QTableWidgetItem(itemss["Type"]))

self.tableWidget.setItem(down,1,QTableWidgetItem(itemss["Color"]))

self.tableWidget.setItem(down,2,QTableWidgetItem(itemss["NoPlate"]))

self.tableWidget.setItem(down,3,QTableWidgetItem(itemss["Time"]))

self.tableWidget.setItem(down,4,QTableWidgetItem(itemss["Location"])
)
                #    right=right+1
                down=down+1
            #self.tableWidget.setItem(1,0,QTableWidgetItem("type"))



        elif typedata=="" and colordata!="" and locationdata=="":

            xx=mydb.db_vehcile.find ( { "Color" : colordata } )
            #x2=xx
            ttt=[]
            for items in xx:
                lengthh=lengthh+1
                ttt.append(items)

            #print(ttt)

            self.tableWidget.setRowCount(lengthh)


            for itemss in ttt:


self.tableWidget.setItem(down,0,QTableWidgetItem(itemss["Type"]))

self.tableWidget.setItem(down,1,QTableWidgetItem(itemss["Color"]))

self.tableWidget.setItem(down,2,QTableWidgetItem(itemss["NoPlate"]))

self.tableWidget.setItem(down,3,QTableWidgetItem(itemss["Time"]))

self.tableWidget.setItem(down,4,QTableWidgetItem(itemss["Location"])
)
```

```
                #    right=right+1
                down=down+1


        elif typedata=="" and colordata=="" and locationdata!="":

            xx=mydb.db_vehcile.find ( { "location" :
locationdata } )
            #x2=xx
            ttt=[]
            for items in xx:
                lengthh=lengthh+1
                ttt.append(items)

            #print(ttt)

            self.tableWidget.setRowCount(lengthh)


            for itemss in ttt:


self.tableWidget.setItem(down,0,QTableWidgetItem(itemss["Type"]))

self.tableWidget.setItem(down,1,QTableWidgetItem(itemss["Color"]))

self.tableWidget.setItem(down,2,QTableWidgetItem(itemss["NoPlate"]))

self.tableWidget.setItem(down,3,QTableWidgetItem(itemss["Time"]))

self.tableWidget.setItem(down,4,QTableWidgetItem(itemss["Location"])
)
                #    right=right+1
                down=down+1


        elif typedata!="" and colordata!="" and locationdata=="":

            q={'$and': [{ 'Type' : typedata }, { 'Color' :
colordata}]}

            xx=mydb.db_vehcile.find(q)
            #x2=xx
            ttt=[]
            for items in xx:
                lengthh=lengthh+1
                ttt.append(items)

            #print(ttt)
```

```
            self.tableWidget.setRowCount(lengthh)


            for itemss in ttt:


self.tableWidget.setItem(down,0,QTableWidgetItem(itemss["Type"]))

self.tableWidget.setItem(down,1,QTableWidgetItem(itemss["Color"]))

self.tableWidget.setItem(down,2,QTableWidgetItem(itemss["NoPlate"]))

self.tableWidget.setItem(down,3,QTableWidgetItem(itemss["Time"]))

self.tableWidget.setItem(down,4,QTableWidgetItem(itemss["Location"])
)
                #    right=right+1
                down=down+1

        elif typedata!="" and colordata=="" and locationdata!="":

            q={'$and': [{ 'Type' : typedata }, { 'Location' :
locationdata}]}

            xx=mydb.db_vehcile.find(q)
            #x2=xx
            ttt=[]
            for items in xx:
                lengthh=lengthh+1
                ttt.append(items)

            #print(ttt)

            self.tableWidget.setRowCount(lengthh)


            for itemss in ttt:


self.tableWidget.setItem(down,0,QTableWidgetItem(itemss["Type"]))

self.tableWidget.setItem(down,1,QTableWidgetItem(itemss["Color"]))

self.tableWidget.setItem(down,2,QTableWidgetItem(itemss["NoPlate"]))

self.tableWidget.setItem(down,3,QTableWidgetItem(itemss["Time"]))

self.tableWidget.setItem(down,4,QTableWidgetItem(itemss["Location"])
)
                #    right=right+1
```

```
                down=down+1


        elif typedata=="" and colordata!="" and locationdata!="":

            q={'$and': [{ 'colordata' : colordata }, { 'Location' :
locationdata}]}

            xx=mydb.db_vehcile.find(q)
            #x2=xx
            ttt=[]
            for items in xx:
                lengthh=lengthh+1
                ttt.append(items)

            #print(ttt)

            self.tableWidget.setRowCount(lengthh)


            for itemss in ttt:


self.tableWidget.setItem(down,0,QTableWidgetItem(itemss["Type"]))

self.tableWidget.setItem(down,1,QTableWidgetItem(itemss["Color"]))

self.tableWidget.setItem(down,2,QTableWidgetItem(itemss["NoPlate"]))

self.tableWidget.setItem(down,3,QTableWidgetItem(itemss["Time"]))

self.tableWidget.setItem(down,4,QTableWidgetItem(itemss["Location"])
)
                #    right=right+1
                down=down+1

        elif typedata!="" and colordata!="" and locationdata!="":

            q={'$and': [{ 'Type' : typedata }, { 'Color' :
colordata }, { 'Location' : locationdata}]}

            xx=mydb.db_vehcile.find(q)
            #x2=xx
            ttt=[]
            for items in xx:
                lengthh=lengthh+1
                ttt.append(items)

            #print(ttt)
```

```
            self.tableWidget.setRowCount(lengthh)


            for itemss in ttt:


self.tableWidget.setItem(down,0,QTableWidgetItem(itemss["Type"]))

self.tableWidget.setItem(down,1,QTableWidgetItem(itemss["Color"]))

self.tableWidget.setItem(down,2,QTableWidgetItem(itemss["NoPlate"]))

self.tableWidget.setItem(down,3,QTableWidgetItem(itemss["Time"]))

self.tableWidget.setItem(down,4,QTableWidgetItem(itemss["Location"])
)
                #    right=right+1
                down=down+1

    def live_streaming(self):
        ret, frame = self.capture.read()
        frame = cv2.resize(frame, (695, 380))
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image = QtGui.QImage(frame, frame.shape[1], frame.shape[0],
                             frame.strides[0],
QtGui.QImage.Format_RGB888)

self.LiveVideStreamFrame.setPixmap(QtGui.QPixmap.fromImage(image))


    def viewFilesList(self):
        self.path = QtWidgets.QFileDialog.getExistingDirectory(None,
"Select Directory")
        fileNames = os.listdir(self.path)
        for items in fileNames:
            self.FilelistWidget.addItem(items)

    def exit(self):
        self.cap.release()
        cv2.destroyAllWindows()
        print("Vehicles")

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    VideoPlayer = QtWidgets.QMainWindow()
    ui = Ui_VideoPlayer()
    ui.setupUi(VideoPlayer)
    VideoPlayer.show()
    sys.exit(app.exec_())
```