An Efficient Traffic Management System to Detect Lane Rule Violation using Real-time Object Detection

by

Faed Ahmed Arnob
17301145
Md. Azmol Fuad
17301154
Abu Tahir Nizam
17101393
Arifin Tanjim Siam
17301123

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

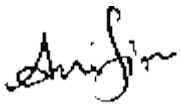
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Faed Ahmed Arnob
17301145

Md. Azmol Fuad
17301154

Abu Tahir Nizam
17101393



Arifin Tanzim Siam
17301123

Approval

The thesis titled "An Efficient Traffic Management System to Detect Lane Rule Violation using Real-time Object Detection" submitted by

1. Faed Ahmed Arnob (17301145)
2. Md. Azmol Fuad (17301154)
3. Abu Tahir Nizam (17101393)
4. Arifin Tanjim Siam (17301123)

Of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on .

Examining Committee:

Supervisor:
(Member)

Md. Motaharul Islam

Md. Motaharul Islam, PhD
Professor
Department of Computer Science and Engineering
United International Uniiversity

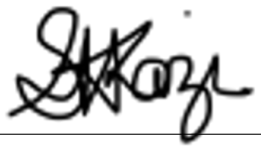
Co-Supervisor:
(Member)

Jannatun Noor
Lecturer
Department of Computer Science and Engineering
Brac University

Program Cooridinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of the Department:
(Chair)



Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

We the members, hereby and sincerely declare that this thesis has been done based on the findings of our extensive research. The materials which have been used, are properly noted and cited in this report. This research work, neither in full nor any part has never been submitted by any other person to another university or any institution for the award of any degree.

Abstract

There has been an upsurge in the number of issues with Bangladesh's present traffic control system. Hence, several accidents have occurred frequently. The two primary causes of a rise in the number of injuries are violations of traffic laws, such as illegal lane changes and excessive speeding. Here we have presented extensive research with an intention to resolve the current traffic management system using real-time object detection. In our proposed system, an edge node will detect the lane-based rule violation and send the data to the nearest intermediary node. Afterward, License plates as objects will be detected using YOLO object detection executed in the intermediary computing device. Finally, extracted license plate images from the intermediary nodes will be sent to BRTA traffic servers to detect the violator's Bangla license plate number using pytesseract. We have built a data set of 1450 images for object detection and achieved an accuracy of 91%. Our system will assist the traffic control department in identifying those responsible for traffic rule violations and ensuring that the laws are strictly enforced.

Keywords: Automatic License Plate Recognition (ALPR); Hough Line Transform; YOLO Object Detection; Fog Computing; Optical Character Recognition (OCR); Computer Vision.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption in this pandemic situation.

Secondly, We are hugely indebted to our supervisor Professor Md. Motaharul Islam of Department of Computer Science and Engineering at United International University for providing us with all the necessary guidance for this research. He always encouraged us to think innovatively and bring out new ideas. He has been a constant source of encouragement and enthusiasm throughout the time of our thesis.

We would also like to express our sincere gratitude to our co-supervisor Jannatun Noor, Lecturer of BRAC University, for sharing her ideas and expertise which helped us to make right choices in the implementation phase.

Thirdly, We want to take a moment to thank our supportive friends who have been there to give us mental support in the hard times. Also the cooperation and understanding of our team mates have helped us to finish our thesis without any hassle.

And finally to our parents, without their continuous support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

σ sigma

τ tau

AI Artificial Intelligence

$ALPR$ Automatic License Plate Recognition

$BLPRS$ Bangla License Plate Recognition System

$BRTA$ Bangladesh Road Transport Authority

CCA Connected Component Analysis

CNN Convolutional Neural Network

CV Computer Vision

$DCNN$ Deep Convolutional Neural Network

$DtBs$ Distance to Border

FCN Fully Convolutional Network

HT Hough Transform

IoT Internet of Things

ITS Intelligent Transport System

$IVIoT$ Intelligent Visual Internet of Things

KNN K-Nearest Neighbours

LPR License Plate Recognition

LSO Line Segmentation and Orientation

NAS Network Attached Storage

$NCPSRR$ National Advisory group to Secure Transportation, Streets and Railroads

PBT Pixel Brightness Trasformation

PHT Probabilistic Hough Transform

$R-CNN$ Region-based Convolutional Neural Network

$RCNN$ Recurrent Convolutional Neural Network

RPN Regional Proposal Network

SSD Single Shot Detector

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Thoughts behind the Traffic Management System

Traffic Management and Monitoring is one of the most significant issues in many countries. This concern is growing rapidly due to increasing daily commuters, overcrowded and narrow roads. Inappropriate or traditional architecture in the signal system and violation of traffic rules are influencing congestion, accidents, and deaths. Besides, traffic management often ends up suing the wrong individual for rule violation due to malfunction. Moreover, accidents are regularly occurring due to various traffic rule violations such as - illegal lane changes, overspeeding, vehicles without fitness, etc.

IoT-driven intelligent traffic management systems offer a proper monitoring system by applying strict punishments for rule violators, automated ticketing systems, smart signaling systems, etc. Advanced technology applications such as AI, Computer Vision, IoT, Machine Learning, Complex Computing, Big Data are now providing real-time solutions for the traffic management system.

1.2 Problem Statement

There's been an increase in road accidents in countries like Bangladesh despite having traffic rules stated by the government. If we look at the years of 2014-16, there's been an emergence of road accidents and fatalities. According to the Bangladesh police, more than 2,500 people had died in almost 2,600 road crashes in 2018. Furthermore, more than 2000 people had died in almost 2500 accidents in 2017 in Bangladesh [1, 2]. In addition, the NCPSRR of Bangladesh had uncovered the data in a press release that nearly 1,500 people died and more than 3,000 people had been injured in close to 1,500 road accidents throughout the country in the first four months of the year of 2019 [2, 3]. If we dive deep into these cases, it can be found that the main reasons for these road accidents were overspeeding, unpermitted lane changing. One of the main reasons behind these accidents, which is often overlooked, is unpermitted lane changing.
In our country, the lane-based traffic rule says, 'No vehicles can't cross the solid marked lanes whatsoever; they can only cross the dotted lane and while changing

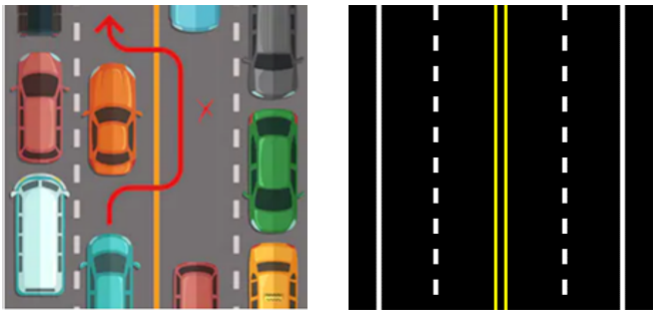


Figure 1.1: Traffic Lanes

the lane over the dotted line, they must use indicators. So in Figure 1.1, vehicles can't cross the straight white and double yellow lines displayed on the right side. On the left side of Figure 1.1, the lane change rule has been illustrated. To the best of our knowledge, there are no systems that have been implemented for detecting lane change rule violations.

In addition, to identify the rule violator, the information of that vehicle is necessary. This information can be extracted from the Bangla License Plate. So in this regard, Bangla ALPR becomes another concern of our research. There are various worldwide sources that supply ALPR solutions. However, these sources do not give support for the Bangla language. Besides, there are no prominent Bangla License Plate datasets to develop such ALPR models.

1.3 Motivation

The objective of our research is to minimize the number of accidents which are caused by the lane-based rule violation. The phenomenon which thrived us to perform this research is the personal experiences while commuting through the city. Two of our team members use a motorcycle for their everyday transportation. According to them the current traffic situation in our country is quite hostile. They faced some minor accidents and also got a glimpse of some traumatizing major accidents. Moreover, one of our classmates also died recently in a horrifying road accident. These experiences enabled us to distinguish the significant reasons behind the accidents. We felt it is high time we should develop an effective and efficient traffic monitoring system to minimize these unfortunate road accidents. Hence, these problems motivated us to develop a traffic management system that will help the people of our country and our team members to have a safe journey.

1.4 Research Aims and Objectives

As previously discussed in the problem statement, The goal of our study is to reduce the amount of accidents that occur on a regular basis in our environment. The only way that can be achieved is to have a strict traffic monitoring system and proper utilization of technologies. In order to do that, we are proposing a complete system where several algorithms have been used, such as - HT, YOLOv5, OCR, etc. We

have learned many new tools and implemented the new learnings to meet our research objectives.

The major aspects and contributions of our research are summarized as follows:

- **Novelty:** We have implemented the latest variant of YOLO using three tier system architecture - end device (pi), computation layer (YOLOv5) and BRTA server (penalization).
- **Performance:** Compared to other existing BLPRS, we have achieved better result on License Plate Recognition.
- **Accuracy:** Our trained model has achieved 91% accuracy on test set of License Plate which is 5% less than current-state-of-the-art (YOLOv2).

The rest of the paper has been organized in the following manner. Chapter 2 discusses the algorithmic analysis which has been done in our research. Chapter 3 discusses the previous researchers that have been done in this field. Our proposed method and our previous works have been explained in Chapter 4. Primary dataset formation and implementation details of our system have been narrated in Chapter 5. After that, we have evaluated our work in different criteria in chapter 6. Finally, chapter 7 concludes our thesis.

Chapter 2

Algorithmic Analysis

2.1 Computer Vision

Computer Vision is one of the emerging technology in the AI environments. It is considered as an inter-disciplinary area of artificial intelligence which briefly focuses on developing techniques for computers to gain a high-level understanding from digital images and video content. It is often denoted as CV. This field of study basically enables the computers to see and observe digital contents such as images and videos like the humans do. With the rapid development of this field, many well accurate techniques of seeing things are also being invented. Soon our whole world will be full of text and images. Text in a sense, is rather easy to search and incorporate . On the other hand, images and videos need different types of well developed algorithms to extract correct information from it. In this section, five major topics of CV will be discussed.

2.1.1 Image Classification

Image classification is the process in which any smart digital computing devices like computers or smartphones can identify any sort of object from an image or video. This is done with the help of image classification and foremostly AI. There are two techniques of classifying an image. Those are pixel level and the other one is object level. Pixel-based classification methods only evaluate spectral information (a pixel's intensity), while object-based classification methods analyze both pixel spectral and spatial information [4, 5]. Object-based classification is more reliable in modern day and the most used one. Image classification follows a structured way to fulfil its purpose. Classifying an image and detecting objects from it is the very end goal of image classification. Right after the data is labeled and preprocessed, it is ready to be fed into the machine learning algorithm. The most commonly used algorithms are KNN, SVM, CNN.

K-nearest Neighbours

Like most of the algorithms in the machine learning field, KNN can be used both for classification and regression [6]. KNN is a classification model that classifies data points based on how close they are to each other. It makes an "informed guess" on what an unclassified point should be classified as based on test results. The

ease of this model is that it is easy to interpret in terms of output. Moreover, the algorithms' calculation time is much more reasonable and the predictive power is quite good.

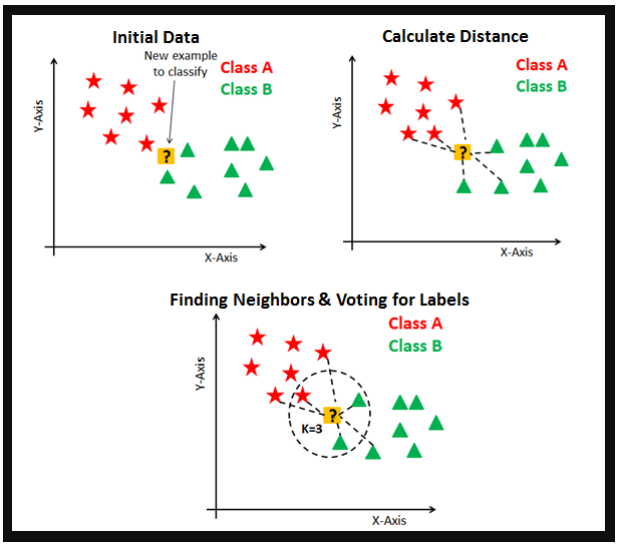


Figure 2.1: Three Stages of KNN

For instance in Figure 2.1, the new example marked as yellow square box needs to be classified . Here, the red star and green triangle are two classes and the question marked region needs to be identified between the two classes. KNN will give the prediction in which class the square box will fall. Following the process, first the distance of the classified points will be calculated from the unclassified one. This can be done by following two methods - Euclidean distance and Manhattan distance. Next it ranks the distance between points by increasing distance values. The shortest distance is considered as the closest neighbour. Finally, vote on the labeled class begin and the closest to k=1,2,3..N nearest neighbors are declared as the predicted class.

Support Vector Machine

SVM stands for support vector machine and is a classification and regression algorithm. Its aim is to improve predictive accuracy while avoiding data overfitting. It is used for applications such as handwriting, face, text and hypertext classification, bioinformatics etc. To achieve the greatest separation between data points, SVM is used. Hyperplane is a component of SVM that maximizes data point separation by increasing line width in iterations [7]. It begins with a line and two parallel lines that are equidistant. Next, the algorithm chooses a stopping point to avoid entering an infinite loop, as well as an expanding factor closer to 1. In this case perfect is 0.99. SVM's main goal is to find a hyperplane in an N-dimensional space that clearly classifies data points, where n is the number of features to use.
Although there are many hyperplanes in the support vector algorithm, the hyperplane with the greatest margin is the highest. To put it another way, the best hyperplane is one that offers the greatest distance between data points for both groups. As a consequence, some reinforcement can occur, allowing potential data points to be classified with greater confidence.The loss function that helps maximize

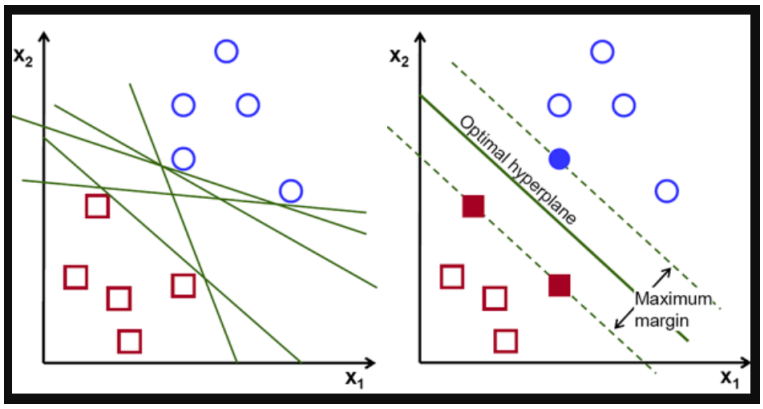


Figure 2.2: SVM Procedure

the margin is hinge loss. The equation that has been used in this algorithm is:

$$c(x, y, f(x)) = 1 - y * f(x)$$

The hinge loss function is equal to 0 when $y * f(x)$ becomes greater or equal 1. Despite the fact that the support vector algorithm is extremely useful, it took a long time to run in Quantopian. Despite its many benefits, we decided not to include it in our ensemble learning.

Convolutional Neural Network

Making computers see things like humans perceive is the main goal of any supervised machine learning algorithm. The advancements in computer vision have come to existence over one particular algorithm which is CNN [8]. CNN works on a multitude of tasks such as image & video recognition, image analysis & classification, media recreation, recommendation systems, natural language processing, etc. This specialized neural network model is designed to work with one-dimensional, two-dimensional, three-dimensional image data. In our case we are going to work with two-dimensional image data. This is performed in multiple layers and the operation here is called convolution. It is a linear operation which consists of multiplication of a set of weights with the input.

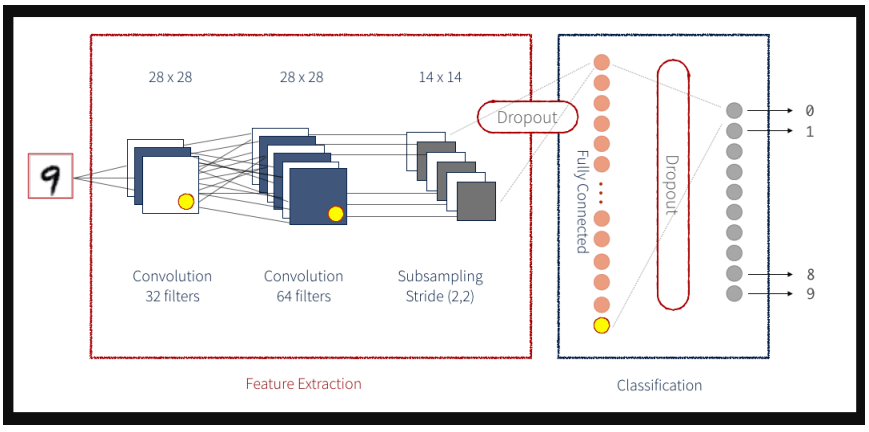


Figure 2.3: CNN Framework

Here, for the multiplication a filter or a kernel is used to perform the process. Meaning multiplication is performed between a two-dimensional array of weights and an array of input data. A scalar product operation is being done between the filter-sized patch of the input and the filter. After that it is summed to a single value. In order to multiply the same filter by the input array multiple times at different points on the input, the filter size must be kept small. Therefore, the filter is applied serially to each overlapping part of the input data from left to right and top to bottom. This powerful idea allows repeated overlapping of the application to build a feature map. Moreover, a 2D convolution layer in Figure 2.3 is meant to have the input of three-dimensional operation. Despite having three-dimensional convolutional operation it is called 2D because the filter run process is done in two dimensions only.

2.1.2 Semantic Segmentation

Semantic segmentation is a gradual process in the progression from rough to delicate inference, with its origins potentially in classification, which entails making a prediction for the entire process. The following move is identification or translation, that includes the classes and other supplementary knowledge about their spatial location[9]. Segmentation is a crucial part of our visual comprehension process when we don't know the exact identity of all objects in an image and It's being used to improve or complement current computer vision techniques. Conventional image segmentation algorithms rely on clustering, which is also supplemented with information from contours and edges. Many older approaches have become outdated as a result of modern advancements. There are few semantic segmentation methods are available.

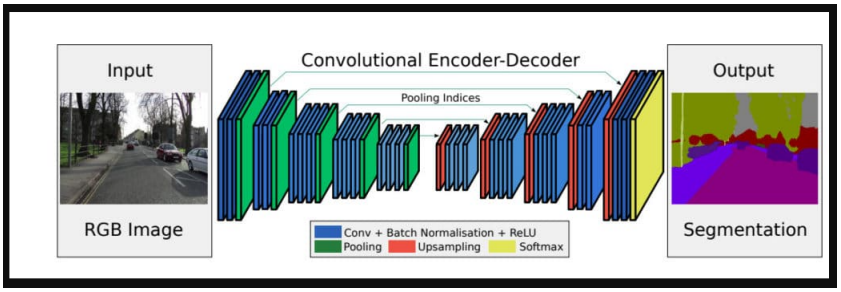


Figure 2.4: Semantic Segmentation

The recent semantic segmentation methods are divided into three groups based on the main component: Region-based semantic segmentation (R-CNN), Fully Convolutional Network (FCN) based semantic segmentation, and Weakly Supervised Segmentation.

R - CNN

The segmentation utilizing recognition technique, which extracts and recognizes free-form sections from an image before categorizing them using region-based techniques, is frequently followed by region-based approaches [10]. During the testing phase, these projections are transformed to pixel estimations, typically by marking

a pixel with the highest scoring region that occupies it. Based on the object detection data, R-CNN performs semantic segmentation. It begins by employing selective search to extract a huge number of object suggestions, after which it computes CNN characteristics for each of them. Lastly, each region is classified using class-specific linear Support vector machines. R-CNN collects two types of characteristics for each area in image segmentation tasks: full region feature and foreground feature, then Concatenating them as the region feature has been discovered to boost outcomes. There are also some drawbacks such as This feature is incompatible with the task of segmentation. Moreover, It lacks sufficient spatial information to generate precise boundaries. Lastly, R-CNN proposals are quite time consuming which will affect the final result.

Fully Convolutional Network

FCN-based approaches are based on the concept of learning a portrayal of pixels to pixels mapping without taking out area proposals. A version of the regular CNN network pipeline is the FCN network pipeline. The main idea is to produce the traditional CNN accepting arbitrary-sized images as input. As FCNs are built up of convolutional, pooling, and up-sampling layers, and are based on the idea of a loss function, they may well be end-to-end trainable. DeepLab-CRF [11] provided an alternative to the de-convolutional layers for increasing output resolution.

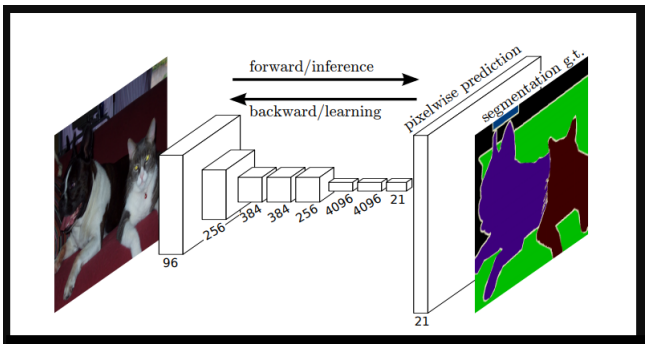


Figure 2.5: Fully Convolutional Network Framework

It employed atrous convolution to boost feature resolution before up-sampling the score map to the original picture resolution with bi-linear interpolation. In FCN techniques, the output function maps' resolution is down-sampled by propagating through several alternating convolutional and pooling layers. As a result, FCN's direct predictions are usually of low resolution. The object boundary was then refined using the CRF method.

Weakly Supervised Segmentation

To manually annotate the segmentation is time-consuming, tedious, and financially costly. As a result, some weakly supervised segmentation methods have suggested the use of annotated bounding boxes or image-level labels to fulfill the semantic segmentation. Dai et al. [12] have proposed to use the bounding box annotations which will act as a supervision to train the system, and Iteratively, the approximation masks for semantic segmentation were improved.

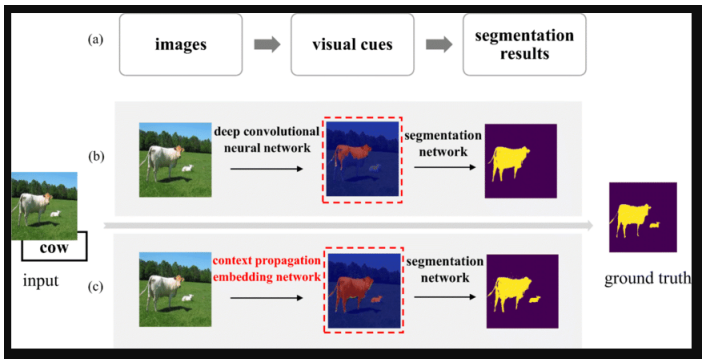


Figure 2.6: Weakly Supervised Segmentation Framework

Papandreou et al. [13] have suggested an expectation–maximization (EM) method for training semantic segmentation models on data that is weakly annotated, for example, image-level or bounding box annotation. They also discovered that only image-level annotation was not sufficient for training a better segmentation model, Although pixel-level annotation could provide a competitive model, bounding box annotation could not. The above methods used slightly different networks and training protocols for semantic segmentation with a view to fit well to the weakly supervised semantic segmentation challenge. Finally, one of the most major disadvantages of adopting image-level supervision is the absence of object localization.

2.1.3 Image Enhancement

The initial and most important step for any image classification or machine learning model is image enhancement. It is mostly known as image preprocessing. The main goal of image preprocessing is enhancement of a given image data by removing unnecessary information and distortions or improving relevant features for further processing. Preprocessing techniques can include several work such as - pixel brightness transformation, image edge detections, image resizing, data augmentations, image filtering etc. The most common image preprocessing for any classification or machine learning model are - pixel brightness transformation, image edge detections and image resizing.

Pixel Brightness Transformation

PBT refers to the correction of brightness, contrast of an image depending on the image pixel information. It is one of the most extensively used techniques for medical and image or video applications, text or speech recognition etc. There are two types of PBT operations - Brightness Correction and Grayscale Transformation [14].

$$O = (\frac{I}{255})^{\gamma} \text{ x } 255$$

Gamma Correction is widely used for brightness correction for individual pixel of an image. This technique executes an non-linear process on a given image and adjust saturation of the image as well [15].

For instance, a side by side comparison of Gamma Correction method is given in Figure 2.7. In the original image, some of the pixels of the image are darker and not

visible. Using Gamma Correction i.e, $g = 2.0$, Most of the objects are visible in the image.

Figure 2.7: Gamma Correction

Due to its ability to work on almost any type of image, histogram equalization is a well-known contrast enhancement technique. Histogram equalization is a complex method for adjusting an image's dynamic range and contrast by changing the image's intensity histogram to the required shapes. Histogram modelling operators can use non-linear and non-monotonic transfer functions to map between pixel intensity values in input and output images.

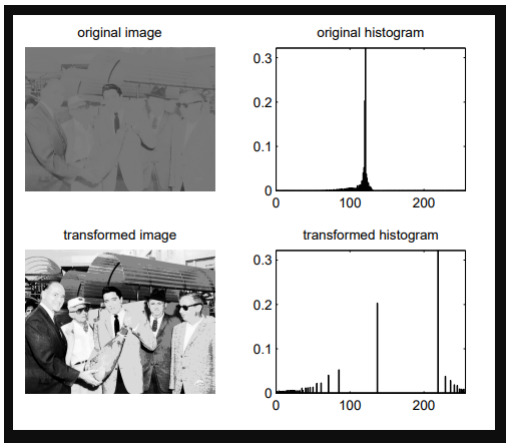


Figure 2.8: Histogram Equalization

Edge Detections

Edge detection refers to the segmentation of image in terms of edge. The concept of this detection is to locate edges in an image by modifying its intensity. It is considered as the main instrument in pattern recognition, segmentation, analysis, feature extraction etc. Edge detection acknowledges the features of an image in terms of drastic changes in grey area. There are mainly two types of edge detection operations- Gradient based and Gaussian based.

Canny Edge Detector is one of the most widely used and famous algorithms for edge detection. It is a Gaussian-based edge operator which was developed by John F. Canny [16]. This algorithm works on achieving three major goals -

- Edge detection with low error rate.
- Accurate localization of edge points.
- One single response for a given edge.

The major advantage of this algorithm is it can extract features without making major modification in the image. Moreover, this algorithm can provide less false edge due to less sensitivity to image noise. There are five steps of Canny Edge algorithm [17].

The procedure starts with Gaussian Filter. As every image contains noise, it can badly affect in detecting edges [18]. Therefore, applying smoothness to the image can improve edge detection. Gaussian filter can smooth or blur image by reducing the evident noises. The equation of Gaussian filter is -

$$G_{(x,y)} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Second step is to find the intensity gradient of the image. As an edge can direct to many point directions, Canny uses four filters to identify 2D edge in the gaussian blurred image.

$$G = \sqrt{G_{x^2} + G_{y^2}}$$

$$\theta = arctan(G_y/G_x)$$

Third step is to apply edge thinning technique. Gradient magnitude thresholding or cutting off lower bound threshold can be applied in this case. It helps to find out the major change of intensity value. However, some edges still get affected by noise and color variation. Therefore, double threshold is applied to conserve edges with high gradient value and remove lower gradient value. The final stage is tracking the edges by hysteresis. An example of the whole process can be illustrated below -

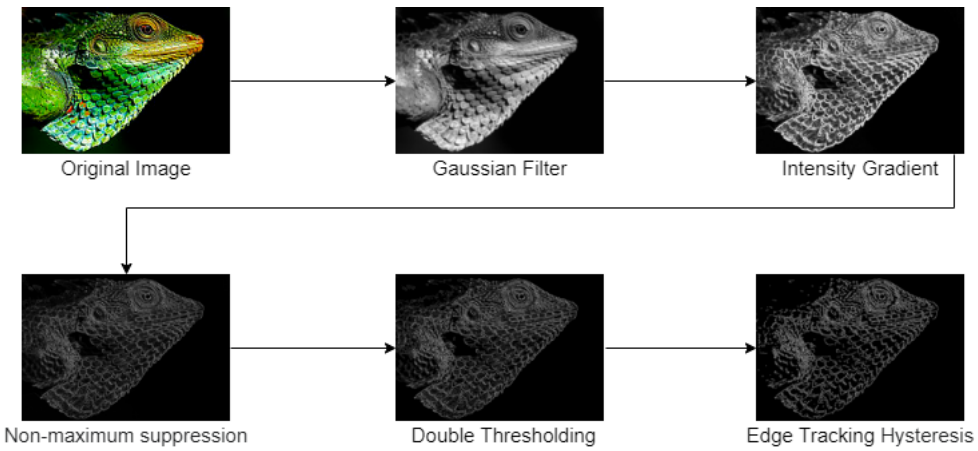


Figure 2.9: Canny Edge Detection Process

2.1.4 Object Detection

In Computer Vision technology, there has been a rapid development regarding object detection. Due to its quick advancement and versatility, object detection has become one of the popular techniques among the researchers. Basically, it is an integration of two procedures - Object Classification and Object Localization. The main objective of object detection is to determine objects in a given image and categorize them. Here, detection of the object is known as Object Localization and categorization of the object is known as Object Classification. However, there has been some conflict of interest in terms of object detection algorithms and classification algorithms. In object detection, a bounding box is drawn around the object in an image for localization. But, there may be different types of objects in the image and several bounding boxes correspond to those objects. And we can not know about this issue beforehand. If regular classification algorithms such as - CNN, KNN, ANN etc. are applied in this case, it will create some major issues. The length of output layer determined by the regular classification algorithms will not be constant. This happens because the object of interest in the image is not fixed and several bounding box are drawn around the objects. Moreover, different objects in an image will have different aspect ratio and locations. Therefore, algorithms like - Faster R-CNN, SSD and YOLO have been developed to deliver real-time object detection.

Faster R-CNN

Faster R-CNN is the third iteration of R-CNN. The problems with the previous iterations - R-CNN and Fast R-CNN were; both of the algorithms apply the selective search approach in order to find the region proposal. Selective search is a algorithm based on hierarchical grouping of objects with similar regions based on colour, size, shape of objects. This algorithm uses an over-segmenting method to the given image and then applies bounding box to the segmented objects. thus the algorithm lists the regional proposal and group them based on the similarity of objects. However, this process is very time-consuming considering the performance of the network.

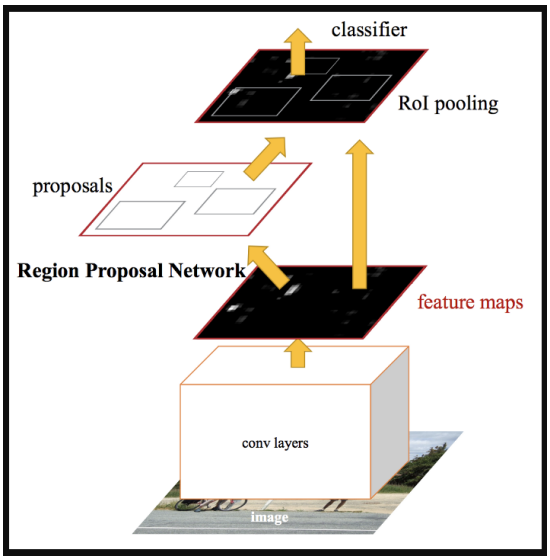


Figure 2.10: Faster R-CNN Working Procedure

In this issue, S. Ren et al. [19] have introduced a new object detection algorithm that removes the selective search approach and uses the Region Proposal Network (RPN). In RPN, they provide the convolutional features of the full image as well as the detection network. The RPN can determine the bounding box of object and objectness in every corner of the image simultaneously using the fully convolutional network. They have then merged the RPN with Fast R-CNN in order to provide the high quality region proposal. Thus, avoiding the selective search approach and using a feature map to identify the region proposals, it become much faster compared to R-CNN and Faster R-CNN. A side by side comparison between the three algorithms in terms of test time speed is given below -

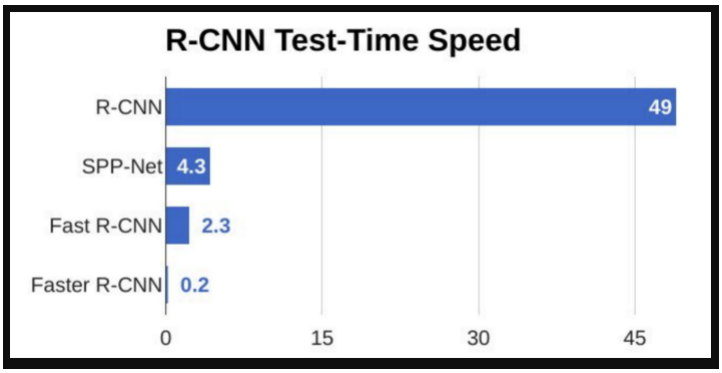


Figure 2.11: Test-Time Speed of Faster R-CNN

Single Shot Detector

SSD is the recently developed object detection algorithm introduced by W. Liu et al. [20] This algorithm uses a single deep neural network to separate the output space of bounding box from the set of default boxes in terms of various aspect ratios. After the separation, the algorithm scale the location of object according to feature map. In order to handle different sizes of different kinds of objects, SSD merges the prediction output from several feature maps with discrete resolutions.

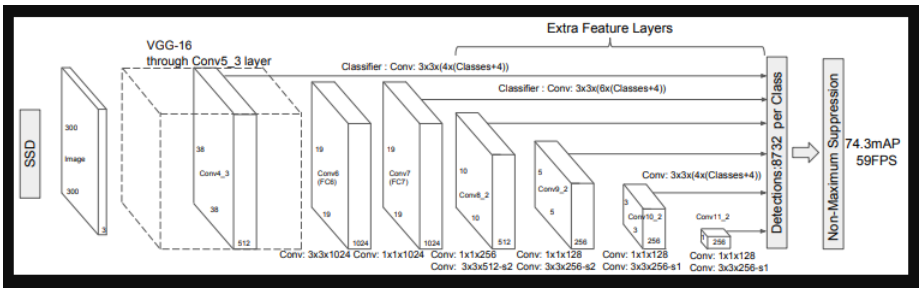


Figure 2.12: Single Shot Detector Model

The major advantage of using SSD in object detections is that this algorithm removes several stages of network such as- proposal generation, feature re-sampling etc. and compress the whole computation in one network, hence the name is single shot detection. Moreover, among all the other algorithms, this is easy to implement into system for training of any object detection.

You Only Look Once

You Only Look Once or renowned as YOLO is one of the most newly developed and efficient approach for object detection. YOLO is a regression-based algorithm. This means that, instead of applying classification approach, this method defines the object detection as a problem of regression and determine the classes and bounding box around the object in one run. Unlike other algorithms, YOLO network looks the whole image and predict the classes and bounding boxes of objects. This algorithm also uses a single neural networks for prediction.

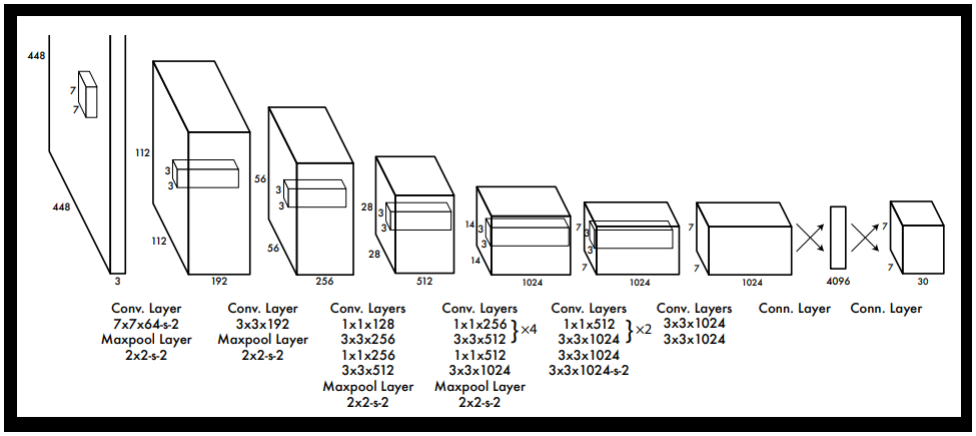


Figure 2.13: YOLO Network

The full YOLO network is consists of 24 convolutional layers continued by 2 fully connected layers. By applying 1 X 1 reduction layers, it helps to eliminate the features space of the preceding layers. After that a 3 X 3 convolutional layers is applied which are pretrained on ImageNet Classification. For the pretrained layer, the algorithm reduces the input image resolution (224 X 224) and for the detection, the resolution is doubled (448 X 448) [21]. The full YOLO network is illustrated in Figure 2.13.

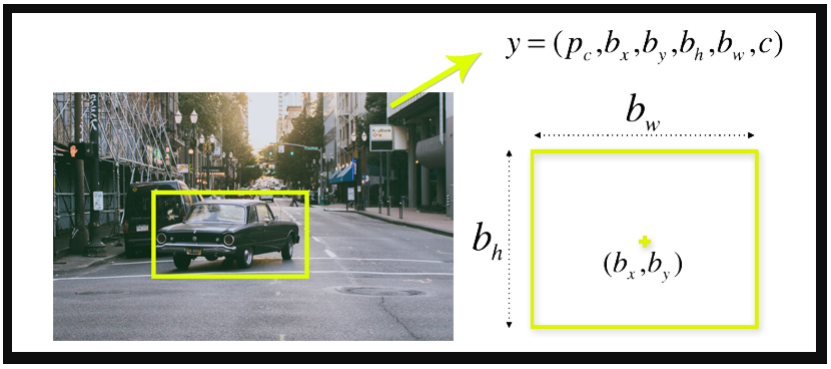


Figure 2.14: YOLO Bounding Box

In YOLO detection model, the input image is cut down in into cells using a S X S grid. This method is implemented because YOLO does not look for interesting regions like other models in an image that can have an object potentially. Each cut down cell is accountable for determining 5 bounding boxes. Here, 5 bounding boxes

are used if there is more than one object within that cell. It is very important for YOLO to determine the bounding box that specify the location of object. In the YOLO bounding box, there is a center (b_x, b_y), width (b_w), height (b_h) and a class of the object (c). YOLO also predict the probability of an object within the bounding box p_c [22].

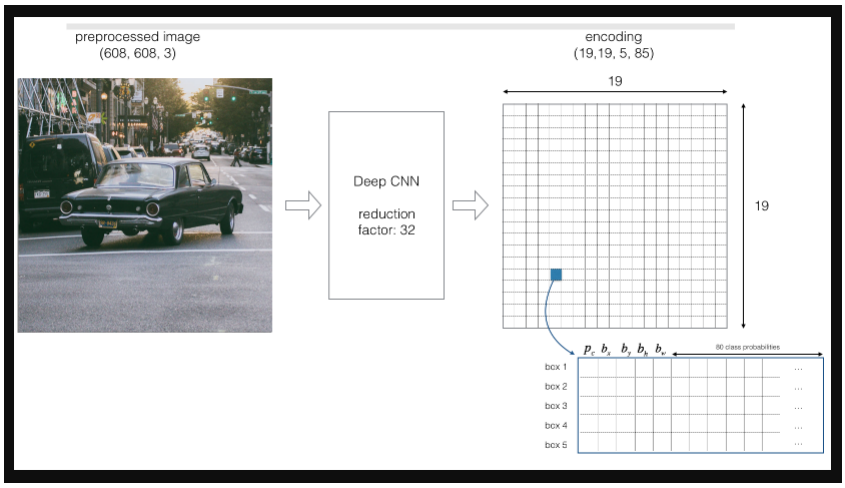


Figure 2.15: YOLO Processing Image

$$y = (p_c, b_x, b_y, b_w, b_h, c)$$

each cell is responsible for detecting all the elements of bounding box. But most of the time, all cells will not contain the object. Therefore, the p_c helps to remove the cells and bounding box with less probability. Bounding box with highest probability will be determined as an object. This is called non-max suppression.

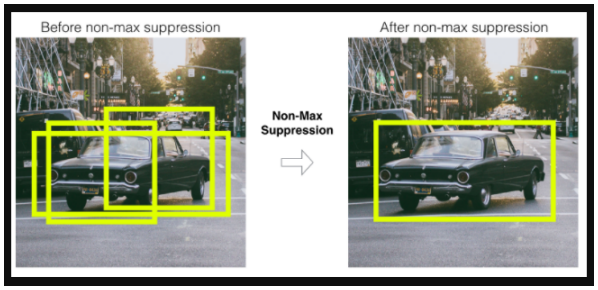


Figure 2.16: YOLO Non-max Suppression

In terms of magnitude and real time objection, YOLO is faster than other algorithms with obtaining detection rate in 45 frames per second. Fast YOLO can detect object 155 frames per second. However, YOLO falls behind in terms of detecting very small objects in the image.

2.1.5 Object Tracking

The method of following a specific object of interest or several objects is referred to as object tracking. It has commonly been used in video and real-world encounters

where observations are made after initial object detection. According to the observation model, it can be divided into two groups. One is the generative approach which is used to explain appearing characteristics by a generative model [23]. Second one is the discriminative approach which is used to distinguish between the object and the background. Taking an initial series of object detections such as - an input set of bounding box coordinates, is the method of object tracking. The bounding box provides a unique ID for each of the preliminary detections, and then monitors each object as they pass through frames in a video while preserving the unique ID allocation. Object tracking also allows us to assign a unique ID to each tracked object as well as to count the number of unique objects in a video. A good object tracking algorithm uses the object detection process only once [24]. It also deals with the tracked object "disappearing" or moving beyond the video frame's boundaries and withstand occlusion. The algorithm possesses the ability to reclaim things that have been "lost" in between frames. There are several methods for object tracking which can be classified into four categories.

Feature based Methods

It is one of the most elementary object tracking techniques [25]. In order to identify objects, properties such as color, texture, and optical flow are extracted first. After extracting the features, the next step is to use those features to discover the most similar object in the next image based on some similarity. In terms of color as a feature, The color histogram illustrates the color distribution in a picture. It displays the various colors in the image as well as the number of pixels for each color. A texture is an informational pattern or a consistent structural layout that is repeated at regular intervals. It can be used in comparison with the color function to define an image's contents. Gabor Wavelet filter is one of the most renowned methods for texture features in object tracking [26]. Gabor Wavelet is a Gaussian modulated function by a complex sinusoid that can be defined as -

$$G_x(t, f) = \int_{-\infty}^{\infty} e^{\pi(\tau-t)^2} e^{-j2\pi f\tau} x(\tau) d\tau$$

The apparent motion of brightness patterns in an image is known as optical flow. Lighting adjustments can create the illusion of motion without actually moving something. The optical flow algorithm measures the brightness pattern displacement from one frame to the next.

Segmentation based Methods

Segmenting foreground items from a video frame is the most critical stage in visual tracking. Foreground segmentation is used to separate foreground items from the background scene [27]. Bottom-Up based method and joint based method are one of the most used object tracking methods where segmentation is examined. The Bottom-Up based method necessitates the separation of two tasks: foreground segmentation and object tracking [28]. As shown in Figure 2.17, the foreground segmentation uses a low-level segmentation algorithm to identify areas across all frames, after which some characteristics from the foreground areas are recovered and tracked using those features.
To solve segmentation error in Bottom-Up method, the researchers combined the

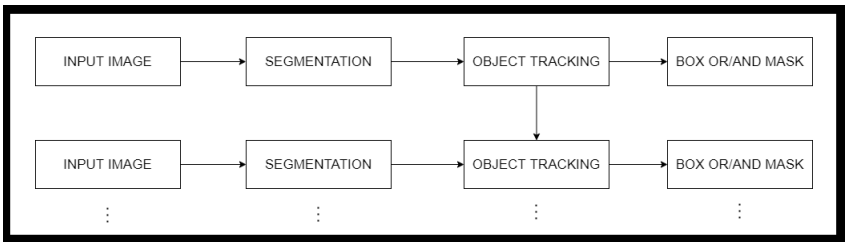


Figure 2.17: Bottom-Up Method Cycle

foreground segmentation and tracking methods named as joint method, as shown in Figure 2.18, which improved tracking accuracy [29].

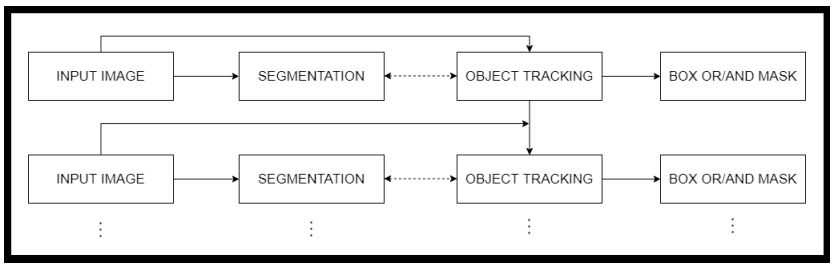


Figure 2.18: Joint based Method Cycle

Estimation based Methods

The tracking problem is transformed into an estimating issue using a state vector to represent the object using estimation methods. Then the state vector defines a system's dynamic action, such as an object's location and velocity.For the dynamic mode estimation issue, Bayesian approaches provide a general framework. The bayesian filter adjusts the target's location on the coordinate system using the most recent sensor data. This strategy is exemplified by the Kalman filter and particle filters. When calculating the position of a linear system with Gaussian errors, the Kalman filter is utilized [30]. The majority of tracking problems are non-linear.

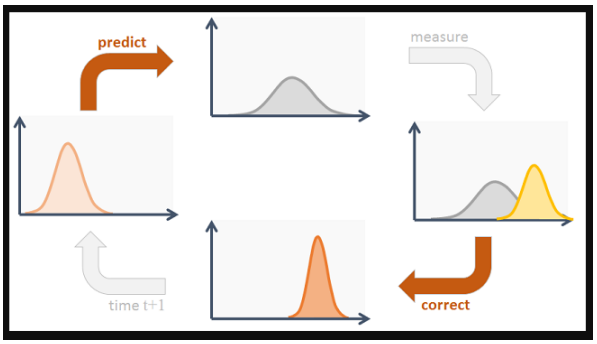


Figure 2.19: Kalman Filter Cycle

As a result, particle filters have been considered as a solution to such issues. The particle filter is a recursive Monte Carlo statistical computation model for non-gaussian noise measurement. The primary goal of a particle filter is to display the distribution of a set of fragments [31].

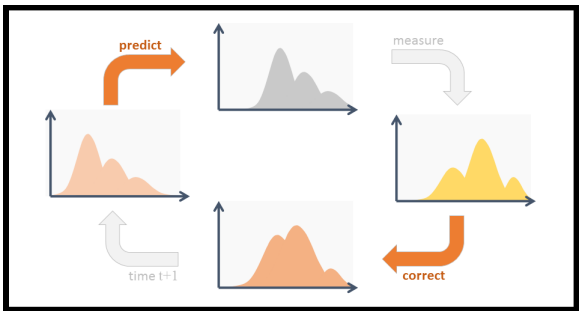


Figure 2.20: Particle Filter Cycle

Learning based Methods

Tracking is commonly conceived of as a classification problem in which discriminative trackers isolate the target from the background. Shallow and Deep learning are the two categories of discriminative learning. Considering shallow learning, when the object is tested after the preparation, it determines whether it is a target object or not. Support vector machines and other classification algorithms can also be used to extract features from diverse objects. Shallow learning, which has fewer parameters, predicts the model, but deep learning also has more layers. Another distinction is that shallow learning allows specialists to extract critical and discriminatory features, while deep learning extracts these features itself. The generative approach looks for places that are more similar to the object. One example of these trackers is Correlation filter-based approaches shown in Figure 2.21.

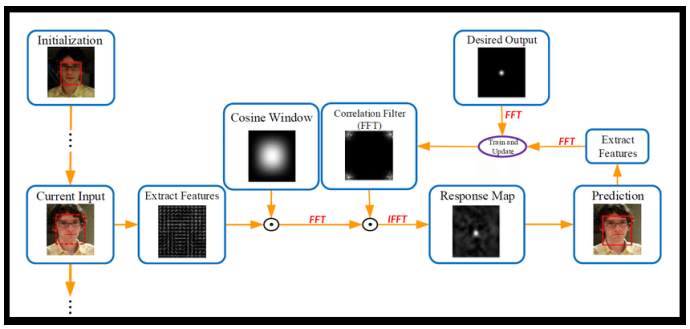


Figure 2.21: Correlation Filter

The correlation filter's main goal is to find an optimal image filter that will achieve the best output on the input image [32]. Considering reinforcement learning problems, we observe in figure 2.22, an agent that learns to choose the best action to achieve a goal by interacting with the environment through trial and error.

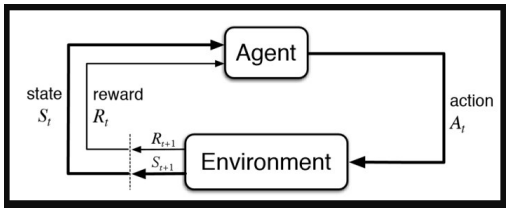


Figure 2.22: Reinforcement learning

2.2 Hough Transform

When an image is used in the field of image processing such as - shape or line detection, object recognition etc. , it is a prerequisite to remove the redundant data from the images while focusing on the important characteristics. Line detection from images is a definitive problem in computer vision as well as in image processing. In terms of real-time applications, line extraction is an essential job.

The Hough transform (HT), introduced by Paul V.C. Hough in 1962, is a systematic algorithm that detect targeted straight lines [33]. HT can detect straight lines efficiently from images even if the image contains noise, blurriness, redundant or missing data. this algorithm works on four systematic steps - Edge detection, Hough Space, Line representation and finally Line detection. The four steps of HT are explained below.

2.2.1 Edge Detection

In order to detect lines from an image using HT, the first and most important task is to detect edges from an image. HT performs its best on binary image. So the primary task is to grayscale the given image and then mask the image in terms of edge. The advantage of grayscaling an image is it reduces the model complexity as well as dimension. RGB images contains three color channels while grayscaled images are single dimensional image. For instance, we will use the Figure 2.23 as an experimental image. So according to the primary task, the grayscaled image is illustrated in Figure 2.24. The next step is edge detection.

Figure 2.23: Original Image

Edge detection algorithm helps to detect edges from an image by finding out the substantial changes of the image in terms of brightness and intensity. As previously mentioned in section 2.1, there are several edge detection algorithms such as - Canny Edge Detector, Sobel Operator, Laplacian Operator etc. However, it is very common in HT algorithm to use Canny edge detector algorithm in order to to detect edges from an image. Here is an example of edge image of Figure 2.25.

Figure 2.24: Grayscaled Image

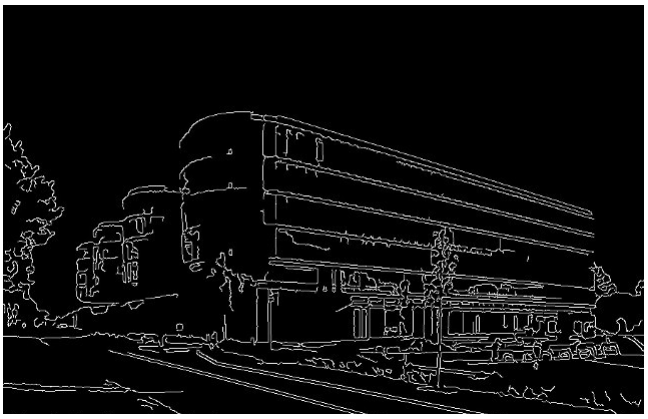


Figure 2.25: Edge Image

2.2.2 The Hough Space

If we consider Figure 2.23, we can say that the image is 2D matrix in terms of x and y coordinates. A straight line from that image parameter space can be represented in Cartesian coordinate system (a, b) -

$$y = ax + b \tag{2.1}$$

The line from the image parameter space can be illustrated as -

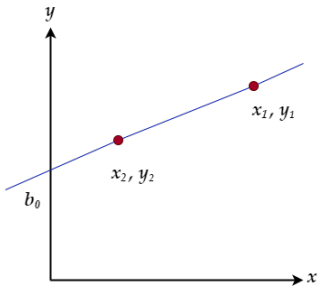


Figure 2.26: Line in Parameter Space

The Hough space is similar to the image space containing a horizontal axis and vertical axis. The horizontal axis represents the slope and the vertical axis represents the intercept of a line on the edge image [34]. In the Hough space, a line is denoted

by its slope a and intercept point b as shown in equation 2.1. A line from the edged image can create a point in Hough space which can be defined as edge point (x_i, y_i). There are infinite number of lines going though a certain edge point. So, if we consider two edge points - (x_1, y_1) and x_2, y_2 as shown in Figure 2.26, we will find that the edge points generate lines in the Hough space in form of $b = ax_1 + y_1$ and $b = ax_2 + y_2$ respectively. The mapping of the edge points are illustrated in Figure 2.27.

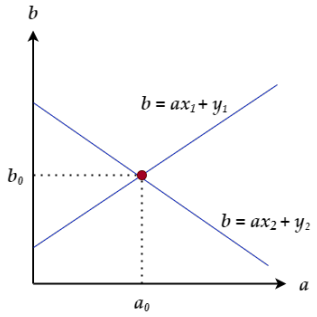


Figure 2.27: Mapping Edge Points in Hough Space

2.2.3 Alternative Line Representation

The Cartesian coordinate system for line presentation has a issue in the Hough space. By using the line representation in equation 2.1 for HT, the algorithm can not detect the slope a because the slope is undefined in terms of vertical lines. It creates unbounded values for all possible values of slope a. Thus, using the equation 2.1 for HT, the algorithm will not be able to detect vertical lines. To avoid this problem, the line is represented using Polar coordinate system (r, θ). Here, a normal line is considered that goes through the origin and the normal line is perpendicular to the previous straight line. the normal line equation can be represented as -

$$p = xcos(\theta) + ysin(\theta) \tag{2.2}$$

Here, p is the perpendicular distance from the origin or the normal line and θ is the angle between the normal line and the x axis. The equation 2.2 can be illustrated as follows -

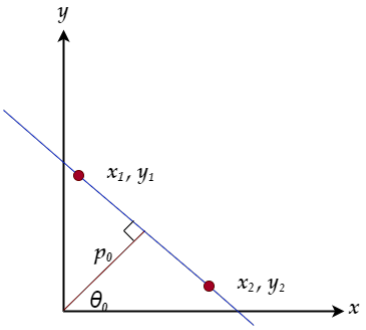


Figure 2.28: Polar Coordinate System of Line Representation

Now in the Hough space, θ will represent the horizontal axis and the p will represent the vertical axis. Now the slope can be calculated using -

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \tag{2.3}$$

The mapping of the two edge points - (x_1, y_1) and (x_2, y_2) will be same as previously shown. This time the edge points will create cosine curve in the Hough Space as shown in Figure 2.29. Thus this method removes the unbounded values of all possible values of a while dealing with vertical lines.

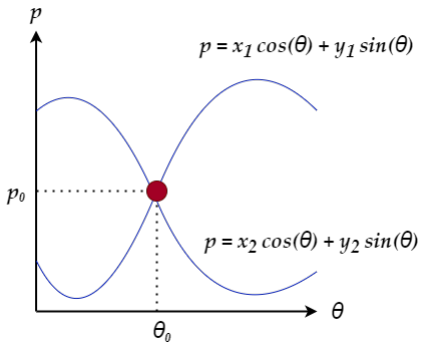


Figure 2.29: New Mapping of Edge Points in Hough Space

2.2.4 Line Detection

Now, if we consider all the edge points and map them into the Hough space, they will generate a lot of cosine curves as shown in Figure 2.30.

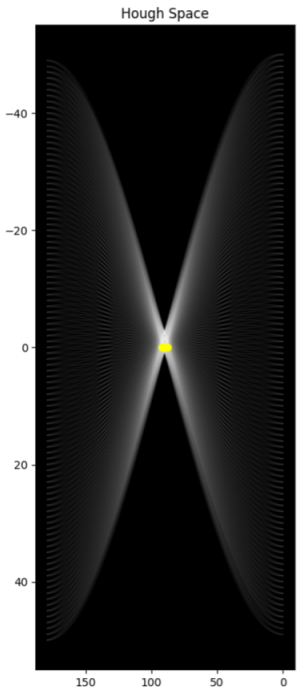


Figure 2.30: Line Detection Process of Hough Space

If two edge points are situated on the same line, the corresponding cosine curves of the edge points will intersect each other. These two edge points can be considered a specific (p, θ) pair. The HT algorithm then finds the (p, θ) pairs that has a number of intersections larger than a certain threshold and combine them to detect lines. So if we follow these steps we will have the following results from our experimental image.

Figure 2.31: Hough Line Detection from Given Image

It is to be mentioned that, there are some redundant lines detected in the Figure 2.31. This can be overcome by using the Probabilistic Hough Transform (PHT) which has more efficient implementation of the HT algorithm [35]. If PHT is used, following output of the given image is extracted -

Figure 2.32: Probabilistic Hough Line Detection

2.3 Fog Computing

Cisco came up with the term "fog computing" to describe distributed cloud infrastructures in 2012. The aim was to support IoT scalability, or the ability to manage a large number of IoT devices and large amounts of data for real-time low-latency

applications [36]. It connects terminal nodes and cloud servers to provide processing, analysis, storage, and a variety of other services. Fog takes cloud services closer to the data-generating nodes. Fog is a dense computational architecture at the network's edge that provides real-time analytics and enhanced security [37]. A fog node is a device that is connected to a network and has computing power and storage space. Switches, routers, servers, security cameras, and so on are all examples of fog nodes.

2.3.1 Background Overview

A Fog environment is a three-tier architecture. The three individual tiers consist of the IoT device tier, the Fog tier, and the cloud tier illustrated in Figure 2.33. The IoT tier consists of different types of IoT devices, sensors, and actuators. The data generated from the IoT devices through sensors are collected and sent to the fog tier for further processing. In the fog tier, some data processing is executed in order to reduce time and latency. The fog tier consists of several intermediate devices known as fog nodes. These fog nodes are linked to the cloud data centers through cloud gateways and resend the data to the cloud tier.

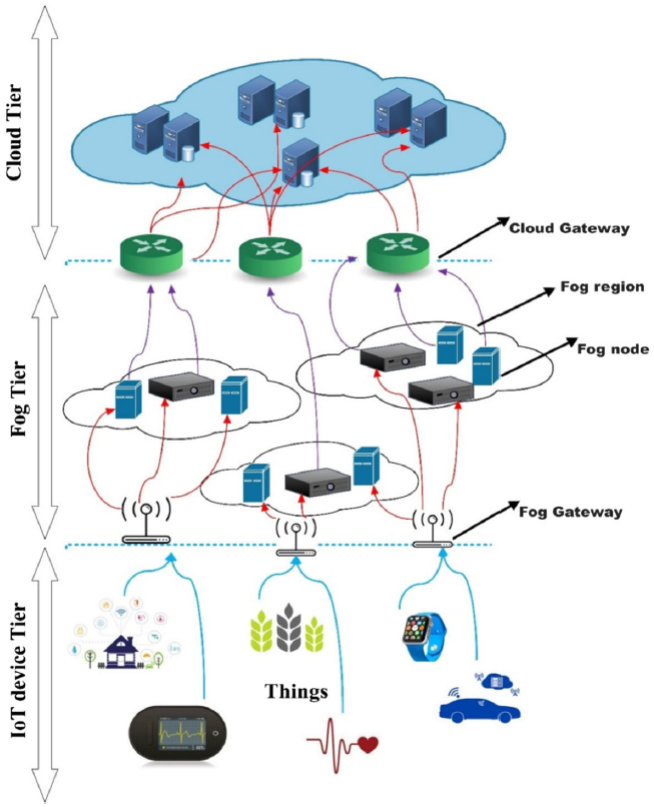


Figure 2.33: Three Tier Fog Architecture

The cloud tier is composed of different high-level servers and storage centers. The storage centers provide high-end processing for the data collected from the IoT devices. Thus the workload is divided into three tiers in order to reduce latency and bandwidth. The core application of the fog layer is to collect data from edge nodes, process the data to some extent, store some necessary data and overheads, route the data to the cloud servers [38]. Traditional heterogeneous network architectures

are incapable of handling the massive amounts of data traffic and computational demands posed by billions of IoT nodes [39]. As a result, fog computing is seen as a promising architecture for meeting the service requirements of these IoT nodes. Cloudlets or fog nodes are intermediary devices with plenty of storage, communication, and computing capabilities. Fog networking, also known as fog computing, is a concept that uses these tools for the edge nodes to get the cloud closer to the IoT devices in a decentralized manner.

2.3.2 Architecture of Fog Computing

In terms of layered architecture, fog computing has six layers. Physical and Virtualization layer, Monitoring layer, Pre-processing layer, Temporary storage layer, Security layer, and Transport layer are the six layers that make up the Fog architecture [40]. In Figure 2.34, The primary task of the physical and virtualization layers is to collect data from the edge nodes. The Monitoring layer performs node monitoring related to various tasks, where fog nodes are checked for power consumption and current state. The pre-processing layer is responsible for data analysis and redundant data removal. This layer compiles analysis of massive amounts of data collected from end devices. The temporary storage layer is associated with data distribution and replication in the short term.

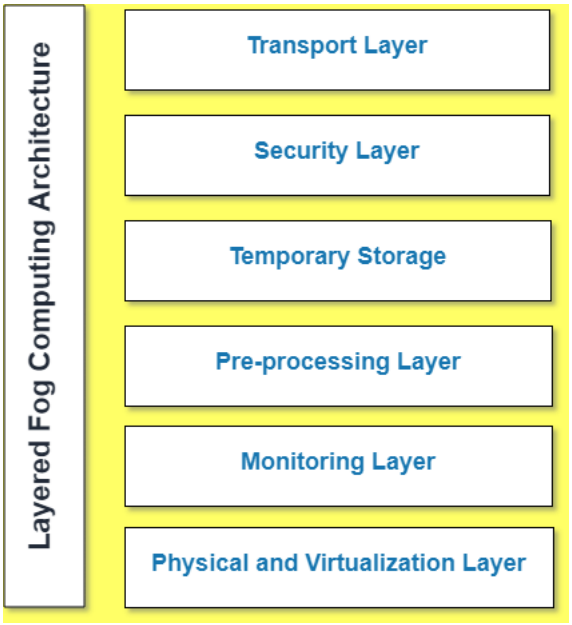


Figure 2.34: Six Layer Fog Architecture

Once the data is uploaded in the cloud, data is removed from this layer. The integrity, encryption, decryption and privacy is maintained in the Security layer. This layer ensures the preservation and security of the data collected from the fog nodes. Lastly, the transport layer securely uploads the final or partial data to the cloud for permanent storage. Smart gateways are used to refine the data before uploading to the cloud.

2.4 Optical Character Recognition

OCR is a modern day scanner tool which helps to recognize text from any surface that contains any language in this world. From scanning documents to having your signature validated by any commercial bank, everything nowadays is done with the help of OCR. The extracted text from any image or video will be provided in digital form [41]. This text recognition system used to be quite expensive a few decades ago but now it is quite in reach to ordinary people. The advancement in the field of deep learning and machine learning helped this recognition system to flourish. However, building and deploying an OCR is not an easy task. Every language has a different set of alphabets and writing styles and it can be handwritten or printed text. Therefore, building an OCR is a challenging task.

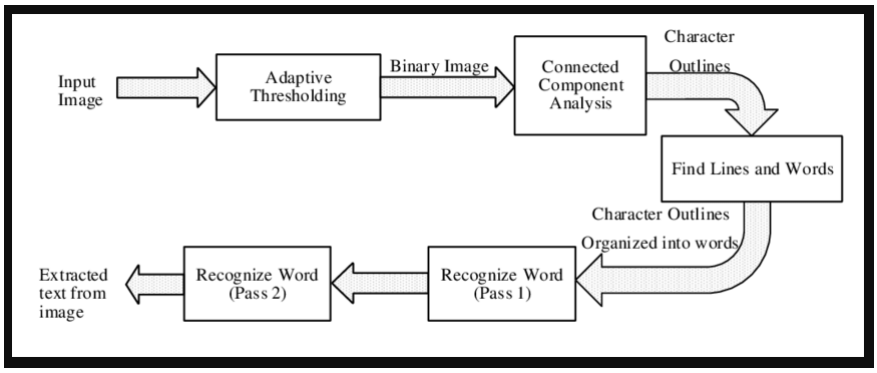


Figure 2.35: Optical Character Recognition Process

Working mechanism of any OCR consists of two basic and necessary steps. First one is 'Text detection'. Here, text is being detected inside the image. Second one is 'Text recognition' and in section text is being extracted and shown as the output of this system. In the Figure 2.35 the steps are shown for text extraction using OCR.

Chapter 3

Literature Review

3.1 Previous Researches

Ghosh et al. [42] have explained the format of a Bangla text, and later they have proposed an ALPR algorithm with three primary stages. The first stage was locating and extracting the license plate from an input image. The authors have utilized Sobel filters which had several steps such as - detecting license plates and removing noise from the input image to extract the region. After that, the authors have detected the vertical edge, located plate area, detected and corrected the skewed license plate to get the perfect view. Finally, the stage has been concluded by extracting the actual license plate by eliminating the extra lines using the Bounding Box analysis. The second stage was segmenting the characters without sacrificing any of their characteristics. Each character was separated and segmented under line segmentation, word segmentation, and character segmentation based on the collection of good character attributes during this process. The third stage was character recognition. This phase has determined the precision of character segmentation and identification work. A neural network has been used to classify each character. It turned out to be a complicated process due to the license plate occupying a small part of the whole photograph, the differences in license plate types, and environmental factors. Finally, the authors have stated that their results showed 84% accuracy in license plate extraction and 80% accuracy in terms of character recognition.

Baten et al. [43] authors have suggested a method for reading license plates that uses a function of the Bangla script known as "Matra". In the script, words have been partitioned as independent connected components and identified using template matching. Firstly, the authors have reduced noise and converted it to a binary image from the input image. After that, they have extracted whole words and single characters from that picture using the CCA method. Later, to detect the number line (second line in the license plate), the authors have verified the orientation and height. Moreover, they have compared detected components with previously arranged database templates to identify numbers. They also have classified numbers by comparing detected components to previously organized database models by finding the word "Metro", reading the city name, category name from the components, inserting hyphens where necessary. Finally, authors have claimed that, by using Matra's function in their proposed way, the algorithm's complexity can be reduced. Also, their proposed system has performed well for pictures captured under a variety

of conditions.

Haque et al. [44] have proposed a method with an algorithm known as the LSO algorithm to localize Bangla license plates automatically and template matching for recognizing. They have broken down the overall working procedure into four major parts - pre-processing, localization, license plate processing, and finally, recognition. Firstly, in the pre-processing part, the authors have modified the input photo by resizing. Secondly, for the localization part, they have used the Sobel filter method for detecting the edge. Along with that, they have also visualized the desired region and extracted the number plate from the cropped image using image morphology and connected component analysis. Thirdly, they have skewed and removed the noises from the extracted license plate in the license plate processing part using Otsu Method. Finally, in the recognition part, the LSO algorithm has been implemented for segmentation, and template matching has been used for recognition and extracted the output in text. Authors have claimed that their suggested model is successful with an 84.87% of classification accuracy. Moreover, the proposed approach not only be helpful for license plate recognition but also will be useful in other fields of Bangla optical character recognition.

Rahman et al. [45] have proposed a CNN-based method for the Bangla license plate recognition system. Besides, authors have claimed that this method can provide better accuracy and can be applied for various security purposes. They have used a dataset of 1750 sample images. Their whole system has four parts. The first part was data acquisition, where the infrared lighting system has been used for data collection, allowing photographs to be computed at day or night. The next step was pre-processing, where modification of the image detects the desired region from the photo, segmenting individual characters. After that, in recognizing parts, CNN has been used to identify the extracted partitioned characters by comparing and training their dataset. Besides, their CNN had used one input layer, two convolution layers, two sub-sampling layers, and a final layer for classification. The last part was the post-processing phase, where the output was shown in the display device. Authors have claimed that they have reached a research precision of 88.67% with the highest amount of training data, which is the highest in this application. Also, they have claimed that they have developed and published the first and most comprehensive BLPRS database.

Dhar et al. [46] have explained about ITS, LPR system. They have proposed a system design for the LPR application of Bangladeshi License Plates where edge detection and CNN have been utilized. The first step of the method was to identify the license plate from the input image by using several preprocessing tasks such as - edge detection, dilation, and extracting the desired region. After that, the verification phase has been done with the help of the DtBs vector technique- for getting better angles, skewness. Later, Character Segmentation methods have been performed to generate the components from the license plate. Lastly, the recognition phase has been done with the help of CNN through automatic feature extraction. The authors have concluded by showing their high success rate of 99.6%.

Oz et al. [47] have proposed a system that determines the license plates of vehicles

consisting of four steps. The first step of their proposed system was real-time image acquisition. Secondly, the authors have identified the area of the license plate from vehicles using an ANN. The third step was Extracting individual characters from license plates using image processing techniques. Lastly, each character in the license plate has been identified using a feed-forward ANN, and then the characters have been compiled. Later, the authors have tested their proposed system in real-time environments and have claimed that the accuracy rate is 95%. Finally, they have concluded by claiming that this system should be safe to use in real-time environments.

Li et al. [48] have proposed a vehicle recognition system based on IVIoT that can be applied on urban roads and mobile sensing vehicles. Moreover, they have proposed a method that is efficient to deal with the images taken under poor weather conditions. This proposed system has several steps; firstly, the authors have recognized the license plate and located it. Secondly, they have partitioned individual characters from the license plate. After that, they have identified the characters using the classifier of SVM that has two stages - feature extraction and training and recognition. Finally, they concluded their process by identifying vehicle color and type to get the Visual Vehicle Tags. According to their proposed system, the positive tested result consists of an 85.80% accuracy rate under various conditions. Besides, they have stated both advantages and disadvantages and the corresponding solutions for their proposed system.

Pavlidis et al. [49] have suggested a method for finding out the number of vehicles occupying the streets. They have focused on the fusion of near-infrared imaging signals and a fuzzy neural network classifier that can run on the merged near-infrared images and have implemented occupant identification and tracking. There are three facets of the issue that they have found. Firstly, the imaging aspect where the authors have developed a mechanism for providing premium imaging signals to the HOV system, with the imaging signal performance remaining high even in adverse weather or at night. Secondly, to detect car occupants in near-infrared imaging, they have developed and manufactured a fuzzy neural classifier. And lastly, following the previous two study methods, they have developed and implemented a prototype HOV counting system. They have concluded by mentioning the issue of not providing accurate face detectors for outdoor conditions, which the authors claim can be solved using their dual-band system.

Wen et al. [50] have suggested a license plate recognition algorithm based on a shadow removal approach and the SVM, a character recognition algorithm. Their proposed system had three steps: The first step was extracting the location and position of the license plate from the image. The second step was extracting individual characters from the license plate. The final step was recognizing the characters using SVM. They have claimed about the overall success rate of the results, which was 93.54%. In the end, they have concluded by saying the restrictions of their proposed system.

Lin et al. [51] have proposed an LPR system that uses CNN, YOLOv2, and SVM. Their proposed system consists of four steps. Firstly, they have identified the vehicle

from an input image. Then, they have generated the license plate from the identified vehicle. Moreover, they have segmented individual characters from the license plate by removing extra areas and filtering out the noises. Finally, they have identified the characters by using CNN. They have concluded that using CNN, YOLOv2, and SVM techniques could obtain 99.2% character recognition precision, demonstrating the superiority of their proposed approach in terms of accuracy and efficiency over the standard system.

Islam et al. [52] have proposed a method using five consecutive steps. Firstly, they have detected license plates by using an algorithm in terms of Region of Interest (ROI) from the input image. They have used a morphological approach in their proposed method. Secondly, they have extracted the ROI. Here, they have segmented the characters and digits by using horizontal and vertical projections. The ROI has been extracted from the license plate image using different geometric properties such as - area, bounding box, and aspect ratio. Thirdly, in terms of character localization, they have used edge extraction-based methods such as the connected component-based approach and bounding box technology that are applied to it to detect Bangla texts. The next step was the extraction of characters, where they have used vertical edge detection and filtering methods. Previously used connected component-based approach and bounding box technology can also detect non-text objects along with text objects. So, to filter out these non-text objects, they have changed the filtering parameters for different types of license plates. The extracted characters were resized to 30×30 pixels and labeled as their types, and stored in different folders named as training and test sets for training and testing the data set. The HOG process is used to extract features from each of the images. The final step was the recognition of those extracted characters. Here, the authors have used an SVM classifier where extracted Histogram of Oriented Gradient (HOG) features have been used as input for matching. Then they have produced a confusion matrix to get the accuracy.

Nazmus et al. [53] have proposed a system to recognize the license plates of the vehicles occupying urban areas. Firstly, they have constructed a training section where the system is trained with their own dataset for Bangla LRP. Their process has been done in three stages. The three chronological stages are - data acquisition (capturing images), data post-processing (cropping the license plate from ROI), individual character cropping (slicing the characters from the license plate). After successfully training the system, their testing samples have been used to assess the testing output using CNN, which consists of an input layer, several alternating convolutions, and max-pooling layers, as well as one fully connected and classification layer. The CNN that was introduced has six layers, one of which is completely connected, and only six and twelve feature maps in two convolution layers, respectively.

Abedin et al. [54] have proposed a Python OpenCV-based framework for an intelligent vehicle management system for the Bangla language. This proposed system has three steps - First, identifying license plate to recognize the plate region and crop region of interest (LP) from the input image by implementing several functions like filtering the image, converting it to binary, detecting and sorting the outlines to obtain the license plate's character outlines, tilt adjustment, and extracting the

region of interest from the input image. Secondly, The characters from the license plate have been extracted from the cropped filtered license plate area by the segmentation process. Finally, the character components have been designed and trained by the convolution layers in the recognition step, identifying the characters using DCNN. The authors have claimed about their algorithm that they have achieved good results in poor condition images too, and the performance rates for labeling the license plate, features extraction, and recognition have been 93%, 98%, and 98%, respectively. Also, The execution time is very satisfactory - 0.11 seconds, which is essential for real-time implementation.

Kim et al. [55] have proposed a network layered system that uses Deep Convolutional Neural Network which simultaneously performs the following two stages - first, a layer for determining whether or not a license plate exists in the input image and second, a layer for identifying license plate data where the training mechanism is based on Multi-Task Learning (MTL) method. Later, the authors have stated that as it gives a better result, the convolutional layer has been put at the bottom of the DCNN. There are three levels of the network. The first level determines whether or not the license plate exists. The classification of digits is the second level, and character classification is the third level. Finally, through experimentation with authentic images, the authors have concluded that the proposed system recognizes digits and characters more reliably than the DCNN using a traditional layer.

Rastegar et al. [56] have suggested an intelligent vehicle control framework based on a fast and accurate license plate detection and recognition method. The proposed system has several steps - Firstly, getting the captured colorful image. The next step is primary filtering with the introduced satisfactory functional threshold. After that, the authors have implemented candidate selection using morphological operators. Following that, the authors also have localized the license plate along with adjusting the angle and removed shadow, and extracted the license plate. Authors have completed the binarization process using the Otsu technique, then partitioning and extracting characters from LP. They have proposed an OCR engine for dilating the extracted essences, also, resized characters from the input vectors of ANN. Lastly, Multilayer Perceptron (MLP) has been used to classify the characters. Finally, they have concluded by saying that the proposed method performs better even for poor-quality images.

Silva et al. [57] have proposed an ALPR system that uses the SIFT algorithm to recognize the license plate from the input image. The author has talked about the steps of the proposed method as follows. The First step is identifying the license plates from the input image by using the Scale Invariant Feature Transform (SIFT) algorithm that consists of Scale-space extrema detection, Local extrema detection, Orientation assignment, and Keypoint description processes. Then, it recognizes the characters with two stages: character segmentation and recognition by using a traditional neural network. Finally, the authors have concluded by conducting two experiments with the suggested method, with the first yielding 414 successes out of 420 characters and the second yielding 370 successes out of 420 characters. They have claimed that their algorithm has an average time of 6.79 seconds for license plate recognition, and execution results obtained from their approach gave 88.33%

success rates.

3.2 Our Improvements

That being said, while all of the papers have been very useful they provide us with different approaches to LPR. Unfortunately their work is not linked to any automated traffic management and penalizing scheme. Since we are not stopping at LPR, our research differs from theirs. We have implemented YOLOv5 in the existing research field which has not been done before. Moreover, we are creating our own datasets for training our YOLOv5 model. Besides, We're identifying the convicted driver who is licensed under that license plate and breaking the law. Furthermore, with the help of our system the monitoring authority will be able to automatically penalize and control traffic laws.

Chapter 4

Proposed System with Architecture

We have proposed applying three tier architecture for developing the system. The first tier is the IoT layer. Here, we have used Raspberry Pi as an edge device. A 12 megapixel camera acting as a CCTV camera will be connected with the edge device to capture the traffic lane rule violation. From the video feed of the traffic, The edge device will capture the image of the traffic rule violation and send the data to the overlying tier. The overlying tier i.e, the fog tier will work out the necessary computation for extracting the license plate and character recognition. The text extracted from the license plate will be sent to the BRTA cloud server for penalizing the culprit. A basic architecture of our proposed system is illustrated below -

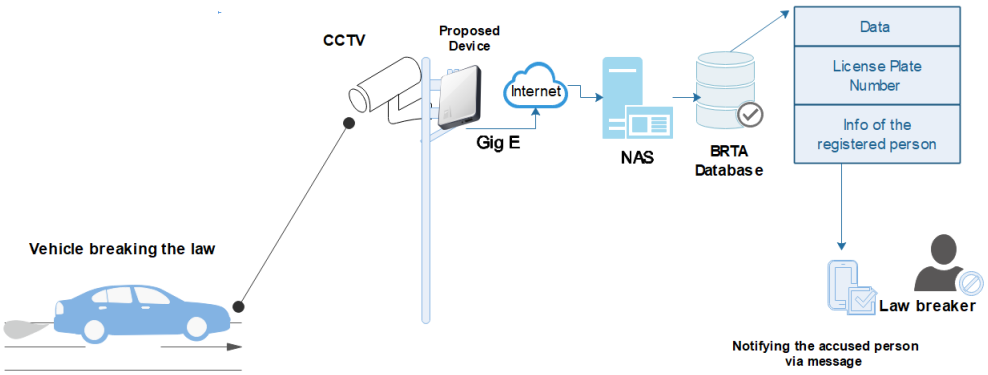


Figure 4.1: Architecture of Proposed System

4.1 Working Procedure

First, we have developed a workflow for our proposed to system to regulate and maintain the work process. According to the Figure 4.2, the first task of our proposed system is detecting traffic lane rule violation.
In order to do that, we have implemented HT to detect the traffic lanes. The main advantage of HT algorithm is the pixels placed on one line does not have to be adjacent to the other pixels of the line. Therefore, it is very beneficial for identifying lines with short breaks in spite of having noise in the image. As previously discussed, to detect the line from an image we need to detect the edges from the image. In

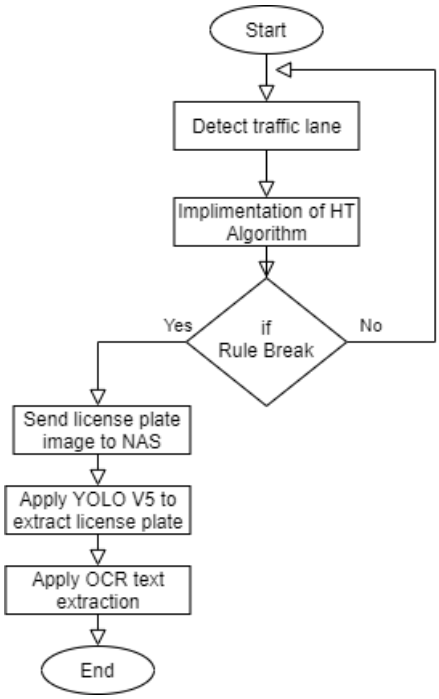


Figure 4.2: Work plan for Proposed System

our case, we have detected the edges of the traffic lanes. Figure 4.3 shows the traffic lanes which will be detected using HT algorithm. For testing the traffic lane rule violation, we have provided a test image as shown in Figure 4.4.

Figure 4.3: Traffic Lane

We have used Canny Edge detection for extracting the edges of the traffic lanes. We chose this algorithm for edge detection as it has low error rate in terms of edge detection. Moreover, this algorithm gives accurate localization of edge center for a detected edge points. Therefore, Canny Edge detector is suitable for detecting the traffic edge. Besides, we have also grayscaled our test image as HT algorithm works best on binary image. The grayscaled version and edged version of our test image illustrated in Figure 4.5 and 4.6 respectively.

Figure 4.4: Test Image of Rule Violation

Figure 4.5: Grayscaled Version of Test Image

After detecting the edges of the traffic lanes, we can trace the edges and map them into Hough space. They will return couple of cosine curves. The HT algorithm will find the edge pairs with larger number of intersection and combine them to detect lines. In Figure 4.6, we have traced the edge pairs of the yellow line for detecting the traffic lane. Here in the edged image, we are considering 0 as black threshold value for any kind of object and 1 as white threshold value for detecting edge points of the yellow traffic lane.

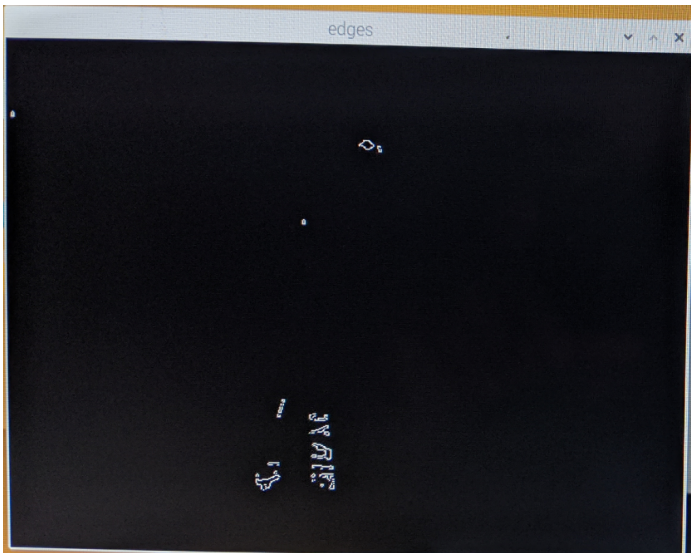


Figure 4.6: Edged Version of Test Image

If any object such as vehicle crosses the yellow traffic lane, there will be disruption in reading the traffic lane by the HT algorithm. In that moment, our modified algorithm in the HT process will capture the image and send it to the Network Attached Storage (NAS) server to store the image. We can see in Figure 4.6 that, as the motocycle was crossing the lane, there was disruption in tracking the edges of the traffic lane. At that moment, the original image which was shown in Figure 4.4 has been sent to NAS server. The output of line detection process from the HT algorithm is shown in Figure 4.7.

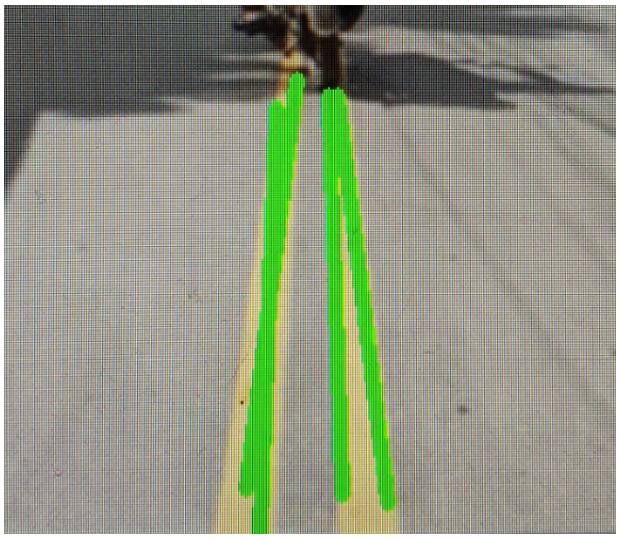


Figure 4.7: Line Detection of Test Image

The whole HT algorithm process has been fulfilled using python OpenCV platform. Moreover, python and pip3 were updated for package management system to install and manage software packages written in python version 3. Futhermore, Some extra dependencies for OpenCV and camera have been installed also in our raspberry pi 4. Here algorithm 1 as shown in Figure 4.8 has been followed to detect the traffic lanes and send images to the server.

Algorithm 1 Hough Line Detection

```
Input: Video(v)
Output: Detected Road Lane(l)
procedure VIDEO CAPTURE(v)
    while true do
        ret, src ←v.read()
        if ret = none then
            procedure VIDEO CAPTURE(v)
                continue
                frame ←GaussianBlur(src, ksize, sigmaX[sigmaY[]])
                hsv ← cvtColor(frame, bgr2hsv)
                lowyellow ←numpy.array([martix low value])
                upyellow ←numpy.array([matrix high value])
                mask ←inRange(hsv, lowyellow, upyellow)
                edge ←Canny(mask, horizontal value, vertical value)
                lines ←HoughLines(edge, hough rho res, hough theta res, hough votes
thresh)
                if lines = true then
                    for l in lines do
                        x1, y1, x2, y2 ←l[0]
                        procedure LINE(frame, (x1, y1), (x2, y2), (0, 255, 0), 5)
                            showframe ←frame
                            showedges ←edge
                            if lines = false then
                                while false do
                                    c = 1
                                    cam = device()
                                    cam.saveSnapshot(c.jpg)
                                    MainProcess(c)
                                    c++
```

Figure 4.8: HT Line detection Algorithm

After sending the image file to NAS server, the main process starts. NAS server acts as an file level storage service for our edge device. NAS is one of the three main storage service. NAS server acts as an intermediate layer between edge device and BRTA server. In our proposed system, BRTA acts as a cloud tier service as it contains all the necessary information related to vehicles' license plates such as - owner name, address, registration date etc. The NAS server will provide the necessary computational processing for extracting the license plate from the image. Here we have implemented YOLOv5 to extract the license plate. YOLO is the most renowned algorithm among all other learning models for object detection. The YOLOv5 model process images in real-time at 140 frames per second. In our case, we will be using license plate as an object in order to extract the license plate. The NAS server will compute the license plate extraction using YOLOv5s version of YOLOv5 as it is the smallest and fastest version among other four variations. Besides, we are extracting only one object from the image which is license plate. Therefore, we are using the smallest version of YOLOv5.

After successful extraction of the license plate, we have implemented the newly developed "Bangla" OCR to extract the text from the license plate. Here, we are not using the default pytesseract OCR - "bengali" as it is not fully compatible with our Bangla characters such as several juktakkhor etc. Besides the default OCR have been developed using Indian Bengali characters. Therefore, we have used the "Bangla" OCR which has been developed using Bangladeshi Bangla characters.

Finally , the extracted text from the license plate using OCR will be forward to BRTA server to match the text with BRTA license plate server. Thus, the necessary information about the rule violator will be extracted and penalyzed for rule violation.

4.2 Published Works

We have been fortunate to publish some of our previous research works regarding the issue - an Conference and a Journal. A small overview of our conference paper and journal paper is described below -

IEEE Conference

In the International Conference on Advances in The Emerging Computing Technologies hosted by IEEE we published our first work [58] regarding traffic mishaps. The idea of the paper was to complete all the necessary computation in the remote device which was Raspberry pi model 3B+ in our case. The road lane detection was still done with the help of HT. However, the plate detection and extraction was done with the help of Computer Vision. The whole process was based on English License Plates. So, KNN classification was used to extract the characters from the plates and contour detection technology was used to cut-out license plates from the scene. The accuracy of the proposed model was 78.83% which was good for starters. Accuracy percentage was poor because KNN detected the logo of a brand as a plate However, by doing all the computation on a single SoC it made the process slower and poorer. It took on an average of 10-12 seconds to complete the whole process for one scene.

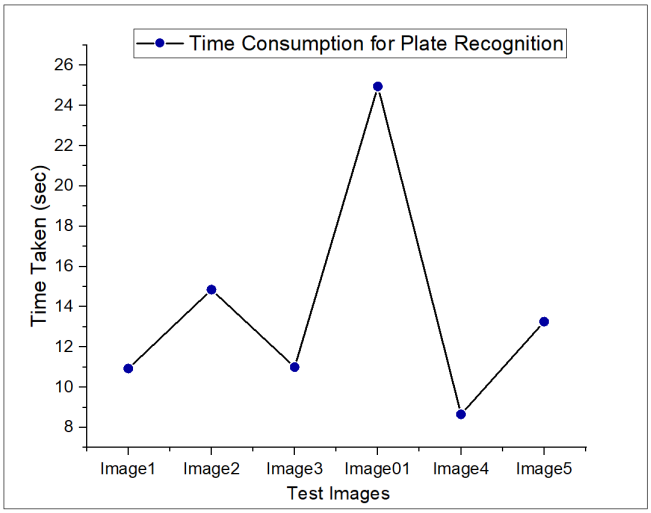


Figure 4.9: Time Consumption for Plate Recognition

AETiC Journal

A research article on the Annals of Emerging Technologies in Computing (AETiC) was published [59] and that was the continuation from the conference paper. After the new version release of the raspberry pi we shifted our whole system to pi 4

from previous model 3B. In this paper we did achieve the accuracy of 80% as the integrated CPU performance of pi-4 was better than 3B. Average time consumption decreased drastically while using pi-4 which dropped to less than 10 seconds for each successful process. However, there were few more advancements, like we proposed a central server to process and save final extracted data.

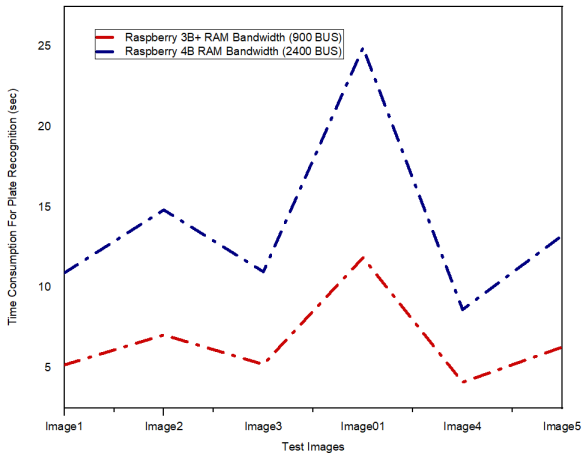


Figure 4.10: Time Comparison for Plate Recognition

However, above discussed materials have been already published, this thesis is an extensive research work in order to overcome the previous limitations for license plate detection.

Chapter 5

Primary Dataset Formation & Implementation

5.1 Data Collection

Our proposed system has been trained properly In order to accurately detect and comprehend the data from license plates. Unfortunately, there is no publicly available data set of Bengali license plates. As a result, we had to make a dataset on our own for our research. We have ridden through the city with a motorcycle while taking pictures of license plates for our dataset. We had taken almost 1450 pictures of license plates from both private and commercial vehicles. The pictures were taken from different angles with a view to properly train our system. As we had taken the pictures from other moving vehicles many pictures were unusable. As a result, we had to ride through the city for several days to collect the pictures of license plates. Later, we processed the collected images accordingly for our proposed system. Finally, we were able to assemble almost 1450 pictures of license plates taken from different angles for our research. Here is a sample preview of our collected dataset -

Figure 5.1: Collection of Data

5.2 Data Preprocessing & Augmentation

In order to train our model, the dataset needs to be preprocessed and augmented. Firstly, we annotated our dataset by applying bounding boxes. As we have proposed of using YOLOv5 model, we annotated every image with a bounding box of the object. Our class of object is license plate. Therefore, we have drawn the bounding boxes around each license plate of the images in the dataset. We have used Virtual Object Tracking Tool (VoTT) for the annotation. Figure 5.2 shows the sample preview of our annotation.

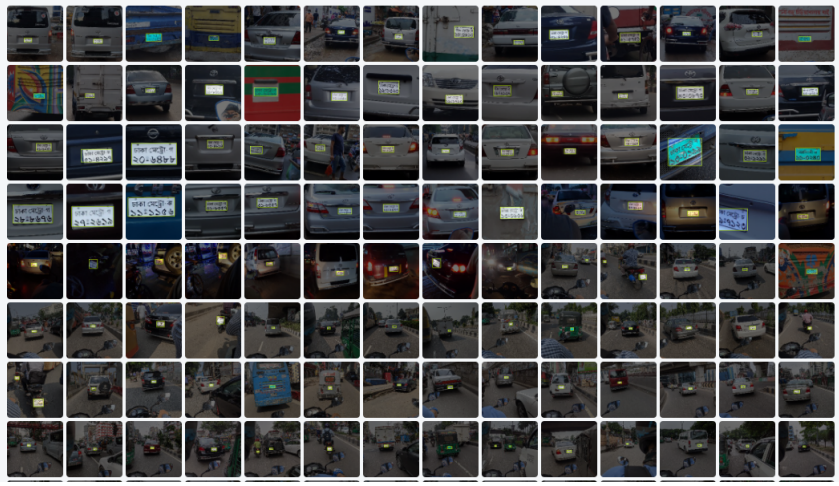


Figure 5.2: Data Annotation using VoTT

After that, we have preprocessed our images. In this stage, we grayscaled our images and applied Gaussian Blur filter in the images. Grayscaling and Blurring an image helps to eliminate the unnecessary and redundant informations from the image. A sample preview of our grayscaled image is given below -

Figure 5.3: Data Preprocessing - Grayscaling

We have also rescaled our images. In this case, we have stretched the images to 416 X 416 size. Resizing an image helps the model to learn faster about the object in the bounding box. A sample of the resized image is shown in Figure 5.4. We have also rotated some of our images so that the model can have variation in the learning phase.

Figure 5.4: Data Preprocessing - Resizing

Finally, in the augmentation stage, we have applied 30% zoom to the images so that any distant image can be zoomed by the model to learn about the license plate. We have also incremented the quantity of images in the dataset through data augmentation. A sample of zoomed images is shown in Figure 5.5.

Figure 5.5: Data Augmentation - Zoom

5.3 Data Training

For training, we have installed all the dependencies for the YOLOv5 model. We have using PyTorch in this case and created the dataset format according to the Pytorch. We have also used the GPU model - Tesla T4 for training purposes. We have chosen YOLOv5s model as it is the fastest and smallest base model. In our training command, we have provided these following options -

img was the input image size which is 416 X 416 format. We have used **batch** size of 32. **batch** is the number of training model used in each iteration. Sample batch images from our training model is given below -
Then we have used **epoch** level of 100. **epoch** refers to the number of complete passes of the training dataset. our 100 **epochs** have been completed in 0.36 hours. Then, **data** is the path to our yaml formatted dataset and **cfg** is the model configuration. We have provided our custom YOLOv5s model configuration where the

Figure 5.6: Data Train Batch Sample

nc is the number of classes. As we have only one class which is license plate, we have used **nc**=1. Our training of the whole model for license plate recognition took approximately 22 mins.

Chapter 6

Performance Evaluation

In this chapter, we have discussed about the evaluation of the trained model's performance. In order to evaluate our trained model, we have set four main parameters - mAP, confusion matrix, precision and recall. We have also discussed about the side by side comparison of the four variations of the YOLOv5 model.

6.1 mAP

Firstly, we have used mean average precision (mAP) parameter for the evaluation of the trained model. The mAP differentiates between the ground truth bounding box and detected bounding box and then returns a value. The higher the value, the more accurate the model is.

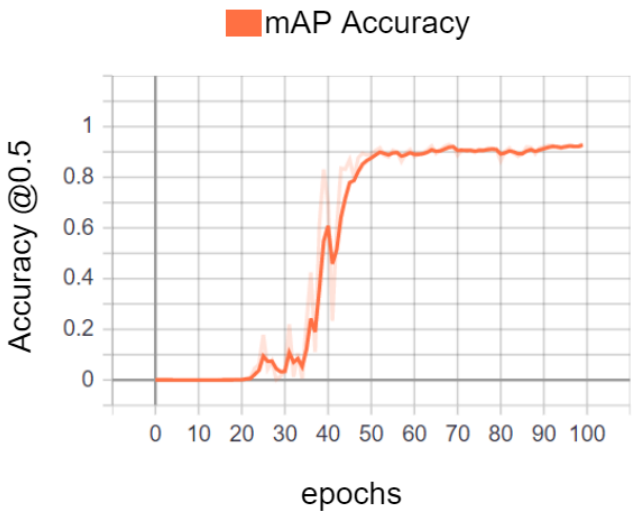


Figure 6.1: mAP Accuracy of YOLOv5s Model @0.5

In our case, we have achieved an accuracy of 91% in mAP@0.5 threshold. This means that, our model has been successful with the accuracy of 91% in detecting the bounding box around the license plate. The illustration of the accuracy of mAP@0.5 is given in Figure 6.1.

We have also achieved an accuracy of 55% in mAP@0.5:0.95 threshold. The accuracy has been illustrated in Figure 6.2.

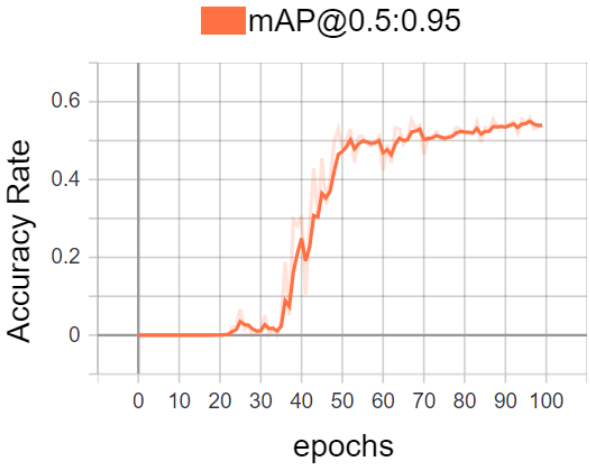


Figure 6.2: mAP Accuracy of YOLOv5s Model @0.5:0.95

6.2 Confusion Matrix

Secondly, we have evaluated our model using confusion matrix for binary classification. In this parameter, we have defined two classes for the ground truth bounding box and the predicted bounding box - positive and negative. In the confusion matrix, there are four metrics - True Positive (T_p), True Negative (T_n), False Positive (F_p) and False Negative (F_n).

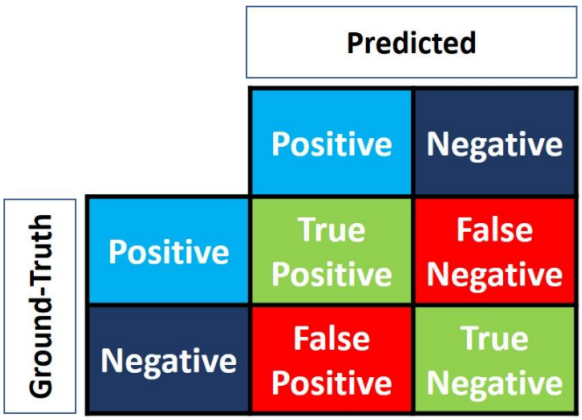


Figure 6.3: Metric of Confusion Matrix

First is True Positive, which represents the correct classification of a positive license plate. Second is True Negative, which refers to the correct classification of a negative license plate. This means that the model correctly predicted no license plate in the input image. Third is False Positive, which represents the incorrectly classified of the negative license plate. This means that the model has detected a license plate but the image does not contain any license plate. Last one is False Negative, that refers to the incorrectly classified of the positive license plate. This means that the model did not detect any license plate while the image contained a license plate.

Finally, in our confusion matrix illustrated in Figure 6.4, we have gained True Positive = 0.91, True Negative = 0, False Positive = 0.09 and False Negative = 1.0.

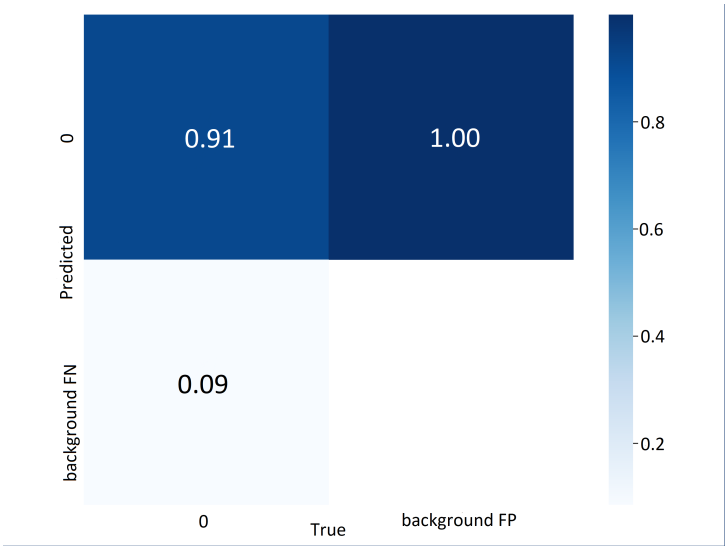


Figure 6.4: Confusion Matrix of YOLOv5s Model

6.3 Precision

Precision is defined as the ratio of the number of True Positive (T_p) and the total number of True Positive (T_p) and False Positive (F_p). We have used the precision parameter to find out the reliability of our model in terms of detecting license plate. the equation of precision is -

$$Precision = \frac{T_p}{(T_p + F_p)} \tag{6.1}$$

In Figure 6.5, we can see that our model has obtained 98% precision and reliability in terms of the license plate detection.

6.4 Recall

We have also used recall parameter. Recall is measured in order to detect the ability of the model in detecting any positive object. The parameter is calculated using equation 6.2 -

$$Recall = \frac{T_p}{(T_p + F_n)} \tag{6.2}$$

In our license plate detection process, our model has achieved 89% ability to detect positive license plates in input images. The recall parameter for our case is illustrated in Figure 6.6.

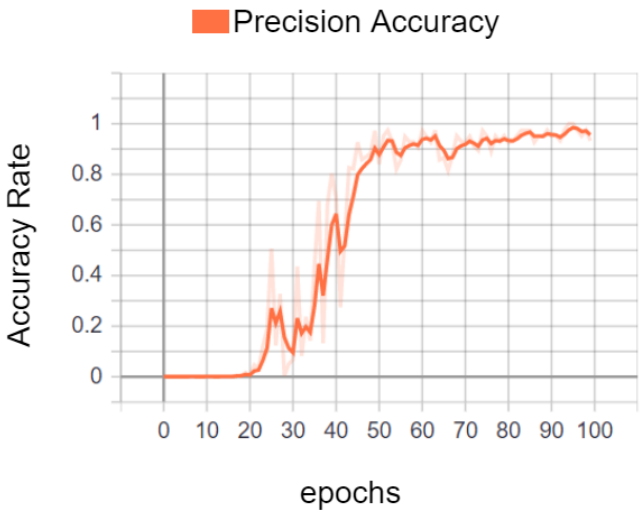


Figure 6.5: Precision of YOLOv5s Model

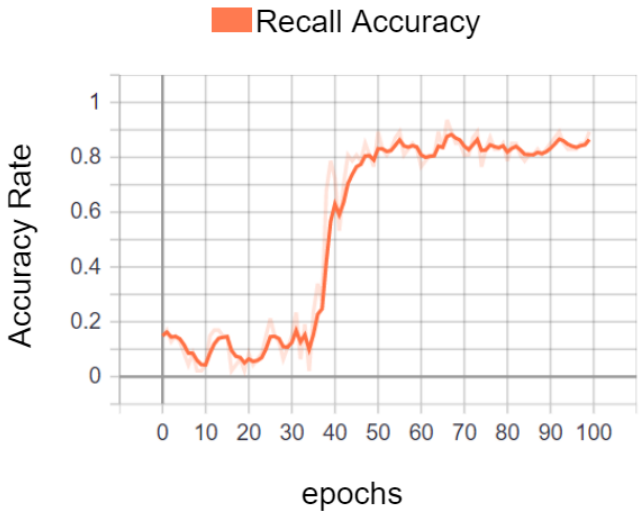


Figure 6.6: Recall of YOLOv5s Model

6.5 Overall Comparison

Moreover, we have compared our trained YOLOv5s model with the other variations of YOLOv5 model. There are mainly four variations of YOLOv5 model - YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x.
Figure 6.7 illustrates the comparison of four variation of YOLOv5 models. From the Figure 6.7, we can clearly say that, YOLOv5s is the fastest model in terms of license plate detection with obtaining an accuracy of 91%. We have used all the previous parameters - mAP, precision and recall for this comparison. We have also calculated the time required for the training of each variation of the YOLOv5 model. In our training YOLOv5s model took 21mins 55s. The other three variations - YOLOv5m, YOLOv5l, YOLOv5x took 12mins 19s, 20mins 39s and 37 mins 12s respectively. We have also calculated the time for testing our trained model. In testing, our trained YOLOv5s model provided real-time license plate detection of 0.36s. Though the model was the smallest variation among the four and took longer time for training, it was able to deliver real-time license plate detection. The other three variants -

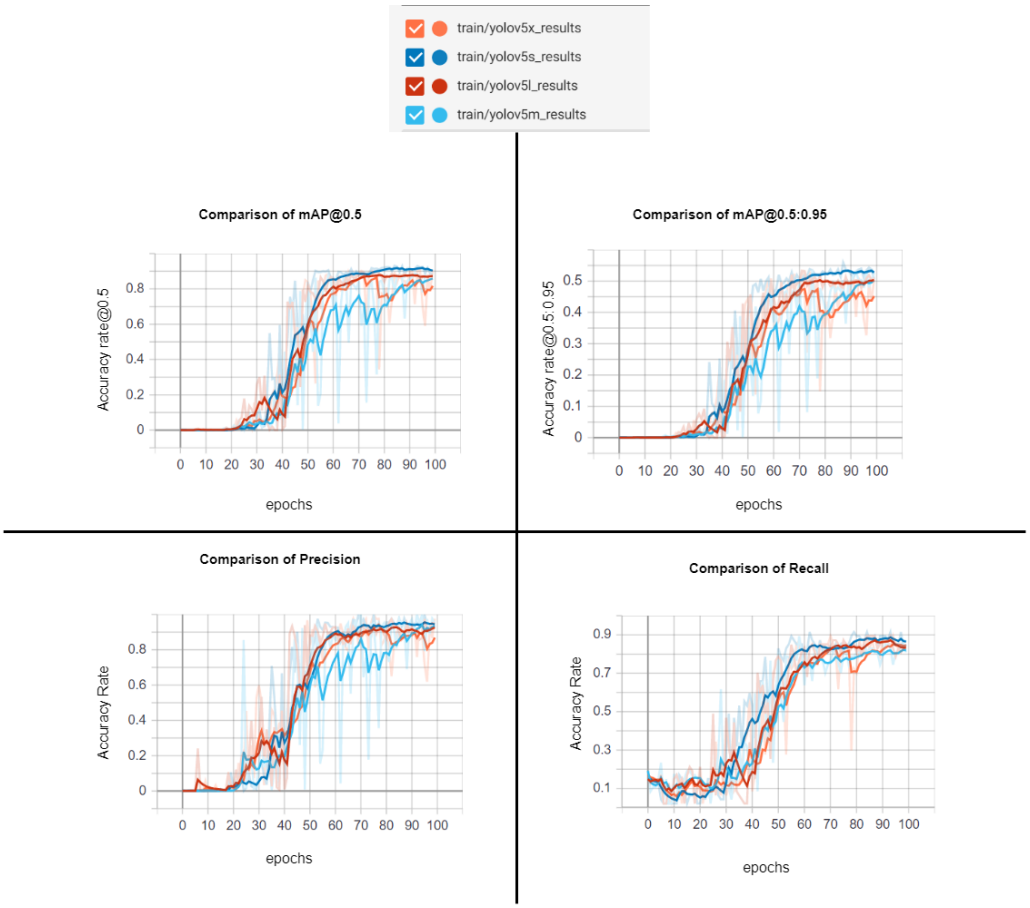


Figure 6.7: Comparison of Four Variation of YOLOv5 Model

YOLOv5m, YOLOv5l, YOLOv5x took 0.42s, 0.51s and 0.61s respectively. A side by side comparison of time consumption in testing the models is given in Figure 6.8. Table 6.1 describe the side by side comparison of time consumption for each variation of the YOLOv5 model in terms of training and testing.

YOLOv5 Model	**Training Time**	**Testing Time**
YOLOv5s	21mins 55s	0.358s
YOLOv5m	12mins 19s	0.421s
YOLOv5l	20mins 39s	0.510s
YOLOv5x	37mins 12s	0.609s

Table 6.1: Training and Testing Time Comparison of YOLOv5 Model

Moreover, we have compared our trained model with the existing works on Bangla License Plate Detection [42–45, 60]. We have provided a side by side comparison of our trained with the existing works in terms of dataset size, license plate recognition, character recognition and time consumption in Table 6.2.

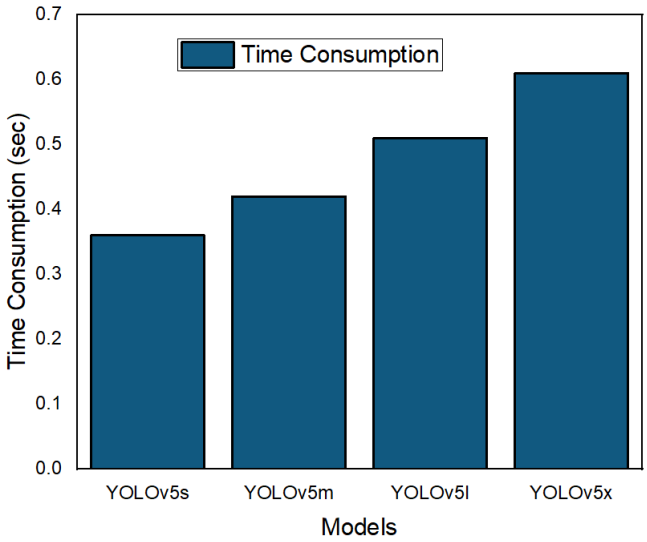


Figure 6.8: Time Comparison of Four Variation of YOLOv5 Model

Paper Citation	**Dataset Size**	**LP Recognition**	**Character Recognition**	**Test Time**
Ghosh et al [42]	Test:300	80%	84%	Not Provided
R.A. Baten et al[43]	Not Provided	Not Provided	Not Provided	1.3s
M.R. Haque et al[44]	Test:119	84.9%	95.8%	Not Provided
M.Rahman et al[45]	Train:1650 Test:350	Not Provided	88.7%	Not Provided
M.M.A. Joarder et al[60]	Not Provided	84.2%	92.1%	1.2s
Our YOLOv5s Model	Train:1200 Test:250	91%	97%	0.35s

Table 6.2: Comparative studys of YOLOv5 and Reviewed Literature

From the comparative analysis of Table 6.2, we can say that, our model has outperformed the other existing works on the Bangla License Plate Detection and Character Recognition.

Finally, we have implemented pytesseract OCR for the character recognition of the detected license plate. Initially, we have used the default built-in OCR for Bangla text extraction. As the default Bangla OCR is developed using the "Indian Bengali" characters, it sometimes has given negative results. Moreover, it was not able to detect "Bangla Juktakkhor" such as - Metro, Chotro etc. Therefore, we have used newly developed "Bangla" OCR which is fully developed using Bangladeshi alphabets. Using the "Bangla" OCR, we have achieved an accuracy of 97% for character recognition. In Figure 6.9, a sample result has been shown using "Bangla"

OCR.

Figure 6.9: Text Extraction using Bangla OCR

Chapter 7

Conclusion

In this chapter we will conclude the paper by discussing the limitations of our work and the possible outcomes of our extensive research. At long last, we will also discuss the possible development sectors of our work.

7.1 Conclusion

Throughout our research we have tried to solve a problem with the help of technology and yes, we did achieve some good results. We have selected this sector as our preferred research field as this is not a volatile topic and needed some serious attention. In this thesis, We have contributed in the field of traffic management systems to detect lane rule violations properly. We have successfully implemented YOLOv5 to detect license plates from scenes which is the current most advanced object detection method. The implementation of intermediary computing devices makes our system more efficient and obtains an accuracy of 91% in terms of license plate detection. Moreover, the use of trained pytesseract Bangla OCR helped us to successfully extract the characters from the Bangladeshi license plates. Furthermore, we have also created a primary dataset of Bangladeshi license plates, which play an important role in the research community as there are no free and enriched datasets available.

7.2 Limitations

Like every other research work, we also faced many challenges and we overcame many of that. This whole thesis was done under the pandemic situation without any sort of access to the university research lab. The situations were unreal and two of our team members were tested positive for covid-19 also. On the other hand, during the implementation period, we faced the absence of a proper Bangla license plate dataset. We overcame this by creating our very own primary dataset from the streets. Taking pictures in Dhaka road conditions is not everyone's piece of cake but we did it somehow and created one. Moreover, YOLO is definitely a fast model to detect objects but, for being a single network it misses some small objects in the images. If the objects are really small, the bounding boxes become even smaller resulting in no object detection.

7.3 Future Works

We have done our level best in this research work but still, there are some scope for improvements. The one we definitely want to develop is an OCR specially to detect Bangladeshi characters from any surface. The accuracy of our model may increase if there is any newer version of YOLO which can be the new state-of-the-art. Lastly, we want to enrich our dataset by importing more photos into it. Till now, we have included the license plates of Dhaka city only. In the near future, we aim to collect plates from other mega cities like Chittagong, Rajshahi etc. As per the feedback received during our defense, we should consider motion parallax for detecting moving objects as fast moving vehicles can get away from camera viewing frame. Our model should have filtering process for emergency vehicles like Ambulances, Police Cars and Fire Trucks etc.

Bibliography

[1] "NCPSRR: 1,212 killed in road accidents in three months". In: *Dhaka Tribune* Bangladesh Nation (2019). URL: https://www.dhakatribune.com/bangladesh/nation/2019/04/02/ncpsrr-1-212-killed-in-road-accidents-in-three-months.

[2] T. S. Adhikary. "Road crashes on rise despite govt measures". In: *The Daily Star* (2019). URL: https://www.thedailystar.net/frontpage/news/road-crashes-rise-despite-govt-measures-1717696.

[3] "Road Accidents: '1,552 killed, 3,039 hurt in four months". In: *The Daily Star* (2019). URL: https://www.thedailystar.net/frontpage/news/road-accidents-1552-killed-3039-hurt-four-months-1742266.

[4] Taru Jain. "Basics of Image Classification Techniques in Machine Learning". In: *Opengenus.org* (2019). URL: https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/.

[5] I. Kanellopoulos and G. G. Wilkinson. "Strategies and best practice for neural network image classification". In: *International Journal of Remote Sensing* 18.4 (1997), pp. 711–725. DOI: 10.1080/014311697218719. eprint: https://doi.org/10.1080/014311697218719. URL: https://doi.org/10.1080/014311697218719.

[6] Avinash Navlani. "KNN Classification using Scikit-learn". In: *Datacamp* (2018). URL: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn.

[7] Theodoros Evgeniou and Massimiliano Pontil. "Support Vector Machines: Theory and Applications". In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.

[8] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[9] Yanming Guo et al. "A review of semantic segmentation using deep neural networks". In: *International Journal of Multimedia Information Retrieval* 7 (June 2018). DOI: 10.1007/s13735-017-0141-z.

[10] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. "Region-Based Semantic Segmentation with End-to-End Training". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 381–397. ISBN: 978-3-319-46448-0.

[11] Liang-Chieh Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2018), pp. 834–848. DOI: 10.1109/TPAMI.2017.2699184.

[12] Jifeng Dai, Kaiming He, and Jian Sun. “BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation”. In: *CoRR* abs/1503.01640 (2015). arXiv: 1503.01640. URL: http://arxiv.org/abs/1503.01640.

[13] George Papandreou et al. “Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1742–1750. DOI: 10.1109/ICCV.2015.203.

[14] Shanto Rahman et al. “An adaptive gamma correction for image enhancement”. In: *EURASIP Journal on Image and Video Processing* 35 (Oct. 2016). DOI: 10.1186/s13640-016-0138-1.

[15] Marcelo Bertalmío. “Chapter 5 - Brightness perception and encoding curves”. In: *Vision Models for High Dynamic Range and Wide Colour Gamut Imaging*. Ed. by Marcelo Bertalmío. Computer Vision and Pattern Recognition. Academic Press, 2020, pp. 95–129. ISBN: 978-0-12-813894-6. DOI: https://doi.org/10.1016/B978-0-12-813894-6.00010-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128138946000107.

[16] John Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[17] Ehsan Akbari Sekehravani, Eduard Babulak, and Mehdi Masoodi. “Implementing canny edge detection algorithm for noisy image”. In: *Bulletin of Electrical Engineering and Informatics* 9 (Aug. 2020), pp. 1404–1410. DOI: 10.11591/eei.v9i4.1837.

[18] Pyushi Singhal, Akhilesh Verma, and Akash Garg. “A study in finding effectiveness of Gaussian blur filter over bilateral filter in natural scenes for graph based image segmentation”. In: *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. 2017, pp. 1–6. DOI: 10.1109/ICACCS.2017.8014612.

[19] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[20] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46448-0.

[21] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[22] Jedrzej Świeżewski. “YOLO Algorithm and YOLO Object Detection: An Introduction”. In: *Applison* (2020). URL: https://appsilon.com/object-detection-yolo-algorithm/.

[23] Rachna Verma. "A Review of Object Detection and Tracking Methods". In: *International Journal of Advance Engineering and Research Development* 4 (Oct. 2017), pp. 569–578.

[24] Mukesh Tiwari and Rakesh Singhai. "A Review of Detection and Tracking of Object from Image and Video Sequences". In: *Int. J. Comput. Intell. Res* 13.5 (2017), pp. 745–765.

[25] Zahra Soleimanitaleb, Mohammad Ali Keyvanrad, and Ali Jafari. "Object Tracking Methods:A Review". In: *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*. 2019, pp. 282–288. DOI: 10.1109/ICCKE48569.2019.8964761.

[26] Muzammil Abdulrahman et al. "Gabor wavelet transform based facial expression recognition using PCA and LBP". In: *2014 22nd Signal Processing and Communications Applications Conference (SIU)*. 2014, pp. 2265–2268. DOI: 10.1109/SIU.2014.6830717.

[27] Qiang Wang et al. "Fast online object tracking and segmentation: A unifying approach". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1328–1338.

[28] Meihui Li et al. "A Bottom-Up and Top-Down Integration Framework for Online Object Tracking". In: *IEEE Transactions on Multimedia* 23 (2020), pp. 105–119.

[29] Chad Aeschliman, Johnny Park, and Avinash C Kak. "A Probabilistic Framework for Joint Segmentation and Tracking". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1371–1378.

[30] Qiang Li et al. "Kalman Filter and Its Application". In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. 2015, pp. 74–77. DOI: 10.1109/ICINIS.2015.35.

[31] Georges Oppenheim, Anne Philippe, and Jean Rigal. "The Particle Filters and their Applications". In: *Chemometrics and Intelligent Laboratory Systems* (Mar. 2008), pp. 87–93. DOI: 10.1016/j.chemolab.2007.09.010.

[32] Erika M. Ramos-Michel and Vitaly Kober. "Correlation Filters for Detection and Localization of Objects in Degraded Images". In: *Progress in Pattern Recognition, Image Analysis and Applications*. Ed. by José Francisco Martínez-Trinidad, Jesús Ariel Carrasco Ochoa, and Josef Kittler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 455–463. ISBN: 978-3-540-46557-7.

[33] Virginio Cantoni and Elio Mattia. "Hough Transform". In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky et al. New York, NY: Springer New York, 2013, pp. 917–918. ISBN: 978-1-4419-9863-7. DOI: 10.1007/978-1-4419-9863-7_1310. URL: https://doi.org/10.1007/978-1-4419-9863-7_1310.

[34] Priyanka Mukhopadhyay and Bidyut B Chaudhuri. "A survey of Hough Transform". In: *Pattern Recognition* 48.3 (2015), pp. 993–1010.

[35] Jiri Matas, Charles Galambos, and Josef Kittler. "Robust Detection of Lines using the Progressive Probabilistic Hough Transform". In: *Computer vision and image understanding* 78.1 (2000), pp. 119–137.

[36] Ketanpreet Kaur and Monika Sachdeva. "Fog computing in IOT: An Overview of New Opportunities". In: *Proceedings of ICETIT 2019* (2020), pp. 59–68.

[37] Flavio Bonomi et al. "Fog Computing and Its Role in the Internet of Things". In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. MCC '12. Helsinki, Finland: Association for Computing Machinery, 2012, 13–16. ISBN: 9781450315197. DOI: 10.1145/2342509.2342513. URL: https://doi.org/10.1145/2342509.2342513.

[38] Mostafa Ghobaei-Arani, Alireza Souri, and Ali Rahmanian. "Resource Management Approaches in Fog Computing: a Comprehensive Review". In: *Journal of Grid Computing* 18 (Mar. 2020). DOI: 10.1007/s10723-019-09491-1.

[39] Md Sipon Miah, Michael Schukat, and Enda Barrett. "An enhanced sum rate in the cluster based cognitive radio relay network using the sequential approach for the future Internet of Things". In: *Human-centric Computing and Information Sciences* 8 (Dec. 2018). DOI: 10.1186/s13673-018-0139-4.

[40] Hany F. Atlam, Robert J. Walters, and Gary B. Wills. "Fog Computing and the Internet of Things: A Review". In: *Big Data and Cognitive Computing* 2.2 (2018). ISSN: 2504-2289. DOI: 10.3390/bdcc2020010. URL: https://www.mdpi.com/2504-2289/2/2/10.

[41] Ashish Ranjan, Varun Nagesh Jolly Behera, and Motahar Reza. "OCR Using Computer Vision and Machine Learning". In: *Machine Learning Algorithms for Industrial Applications*. Ed. by Santosh Kumar Das et al. Cham: Springer International Publishing, 2021, pp. 83–105. DOI: 10.1007/978-3-030-50641-4_6. URL: https://doi.org/10.1007/978-3-030-50641-4_6.

[42] Ashim Ghosh et al. "Automatic License Plate Recognition (ALPR) for Bangladeshi Vehicles". In: *Global Journals Inc. (USA)* 11 (Dec. 2011), pp. 69–73.

[43] Raiyan Abdul Baten, Zunaid Omair, and Urmita Sikder. "Bangla license plate reader for metropolitan cities of Bangladesh using template matching". In: *8th International Conference on Electrical and Computer Engineering*. 2014, pp. 776–779. DOI: 10.1109/ICECE.2014.7026925.

[44] Md. Rokibul Haque et al. "Line Segmentation and Orientation Algorithm for Automatic Bengali License Plate Localization and Recognition". In: *International Journal of Computer Applications* 154 (2016), pp. 21–28.

[45] M M Shaifur Rahman et al. "Bangla License Plate Recognition Using Convolutional Neural Networks (CNN)". In: *2019 22nd International Conference on Computer and Information Technology (ICCIT)*. 2019, pp. 1–6. DOI: 10.1109/ICCIT48885.2019.9038597.

[46] Prashengit Dhar et al. "A System Design for License Plate Recognition by Using Edge Detection and Convolution Neural Network". In: *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*. 2018, pp. 1–4. DOI: 10.1109/IC4ME2.2018.8465630.

[47] Cemil Oz and Fikret Ercal. "A Practical License Plate Recognition System for Real-Time Environments". In: *Computational Intelligence and Bioinspired Systems*. Ed. by Joan Cabestany, Alberto Prieto, and Francisco Sandoval. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 881–888. ISBN: 978-3-540-32106-4.

[48] Qingwu Li et al. “Road Vehicle Monitoring System Based on Intelligent Visual Internet of Things”. In: *Journal of Sensors* 2015 (July 2015). DOI: 10.1155/2015/720308.

[49] I. Pavlidis, V. Morellas, and N. Papanikolopoulos. “A vehicle occupant counting system based on near-infrared phenomenology and fuzzy neural classification”. In: *IEEE Transactions on Intelligent Transportation Systems* 1.2 (2000), pp. 72–85. DOI: 10.1109/TITS.2000.880964.

[50] Ying Wen et al. “An Algorithm for License Plate Recognition Applied to Intelligent Transportation System”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.3 (2011), pp. 830–845. DOI: 10.1109/TITS.2011.2114346.

[51] Cheng-Hung Lin, Yong-Sin Lin, and Wei-Chen Liu. “An efficient license plate recognition system using convolution neural networks”. In: *2018 IEEE International Conference on Applied System Invention (ICASI)*. 2018, pp. 224–227. DOI: 10.1109/ICASI.2018.8394573.

[52] Rashedul Islam, Md. Rafiqul Islam, and Kamrul Talukder. “An efficient method for extraction and recognition of bangla characters from vehicle license plates”. In: *Multimedia Tools and Applications* 79 (July 2020). DOI: 10.1007/s11042-020-08629-8.

[53] Nazmus Saif et al. “Automatic License Plate Recognition System for Bangla License Plates using Convolutional Neural Network”. In: Dec. 2019. DOI: 10.1109/TENCON.2019.8929280.

[54] Md. Zainal Abedin et al. “License plate recognition system based on contour properties and deep learning model”. In: *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. 2017, pp. 590–593. DOI: 10.1109/R10-HTC.2017.8289029.

[55] Hong-Hyun Kim et al. “Multi-task convolutional neural network system for license plate recognition”. In: *International Journal of Control, Automation and Systems* 15.6 (2017), pp. 2942–2949.

[56] Saeed Rastegar et al. “An intelligent control system using an efficient License Plate Location and Recognition Approach”. In: *International Journal of Image Processing (IJIP) Volume (3)* 3.5 (2009), pp. 252–264.

[57] Francisco Assis da Silva et al. “ALPRs-A new approach for license plate recognition using the SIFT algorithm”. In: *arXiv preprint arXiv:1303.1667* (2013).

[58] Faed Ahmed Arnob et al. “An Intelligent Traffic System for Detecting Lane Based Rule Violation”. In: *2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*. 2020, pp. 1–6. DOI: 10.1109/AECT47998.2020.9194163.

[59] Md Fuad et al. “A Novel Traffic System for Detecting Lane-Based Rule Violation”. In: *Annals of Emerging Technologies in Computing (AETiC)* 4.3 (2020), pp. 29–41.

[60] M. A. Joarder et al. “Bangla automatic number plate recognition system using artificial neural network”. In: 2012.