



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A
PROJECT REPORT
ON
TRAFFIC VIOLATION DETECTION WITH COMPUTER VISION

SUBMITTED BY:

NEHA DAHAL (PUL075BEI019)

SIMON G.C. (PUL075BEI036)

SUBASH MAINALI (PUL075BEI039)

UDAYA RAJ SUBEDI (PUL075BEI047)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

April, 2023

Page of Approval

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled ”**Traffic Violation Detection With Computer Vision**” submitted by **Neha Dahal, Simon G.C., Subash Mainali, Udaya Raj Subedi** in partial fulfillment of the requirements for the Bachelor’s degree in Electronics & Computer Engineering.

.....

Supervisor

Dibakar Raj Pant

Associate Professor

Department of Electronics and Computer

Engineering,

Pulchowk Campus, IOE, TU.

.....

Internal examiner

Person B

Assistant Professor

Department of Electronics and Computer

Engineering,

Pulchowk Campus, IOE, TU.

.....

External examiner

Person C

Assistant Professor

Department of Electronics and Computer Engineering,

Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

The author has agreed that the Library, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purposes may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or the other use of this report for financial gain without approval of to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering and author's written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head

Department of Electronics and Computer Engineering

Pulchowk Campus, Institute of Engineering, TU

Lalitpur, Nepal.

Acknowledgments

We would like to express our sincere gratitude and appreciation to all those who contributed to the success of our final year project.

First and foremost, we would like to thank our project supervisor, **Assoc. Prof. Dibakar Raj Pant** for providing us with invaluable guidance and continuous support throughout the project. His knowledge in this field has been crucial in shaping our project and achieving its objectives.

We express our gratitude to the administration and faculty members of **Department of Electronics and Computer Engineering** for providing us with the necessary resources and facilities to undertake this project.

We also extend our heartfelt appreciation to our classmates and friends for their unwavering support and collaboration.

Furthermore, we would like to acknowledge the contribution of our family and friends for their continuous support throughout the project.

Once again, we would like to extend our sincere thanks to everyone who has contributed to the successful completion of our project.

Abstract

In this project we used YOLOv5s that was trained on custom dataset collected by us which consisted of 2193 images of 6 classes which was augmented to extend our dataset to 5259 images and was split in the ratio of 70:20:10 for train, validation, and test respectively. For tracking the detected objects in the video, we used DeepSORT which tracks and outputs the bounding box for the object with respective track IDs. Then if the detected and tracked object have violated traffic lights the corresponding license plate of the object in question is sent as input for segmentation program. The image of the license plate undergoes HSV color space conversion, color masking and perspective transformed in that order before it is preprocessed for profiling the different types of license plate in the dataset. The image undergoes horizontal projection profiling and vertical projection profiling which is then validated to separate the characters of the license plate. The segmented characters are then fed to a CNN trained on our custom dataset of characters of license plate, for now which consists of only private vehicles of Bagmati province.

Keywords: *YOLO, DeepSORT, tracking, character segmentation, CNN*

Contents

- Page of Approval ii
- Copyright iii
- Acknowledgements iv
- Abstract v
- Contents vii
- List of Figures ix
- List of Tables x
- List of Abbreviations xi

- 1 Introduction 1**
 - 1.1 Background 1
 - 1.2 Problem Statements 1
 - 1.3 Objectives 2
 - 1.4 Scope 2

- 2 Literature Review 4**
 - 2.1 Related work 4
 - 2.2 Related theory 6
 - 2.2.1 History of YOLO 6
 - 2.2.2 YOLOv5 Architecture 8
 - 2.2.3 Nepali Vehicle Classification and License Plate 11

- 3 Methodology 17**

- 4 System design 18**

- 5 Results & Discussion 19**
 - 5.1 Object Detection 19

5.1.1	Data Acquisition	20
5.1.2	Data Preprocessing	21
5.1.3	Data Augmentation	21
5.1.4	Model Training	22
5.2	DeepSORT Tracking	30
5.3	Violation Detection	34
5.4	Plate Localization	35
5.5	Processing ROI	36
5.6	Character Segmentation	38
5.7	Plate Recognition	48
5.7.1	Character Pre-processing	49
5.7.2	Character Recognition	55
5.7.3	Multi-class classification model evaluation metrics	65
6	Conclusions	72
7	Limitations and Future enhancement	73
	References	73

List of Figures

2.1	Yolov5 Architecture	9
2.2	Applying CSPNet to ResNet and DenseNet	10
2.3	Zonal Plate Format	13
2.4	Provincial Plate Format	16
3.1	Project Methodology	17
4.1	Designed System	18
5.1	Object Detection	19
5.2	Confusion Matrix Yolov5s Model	24
5.3	Recall Confidence Plot	25
5.4	Precision Confidence Plot	25
5.5	YOLOv5s Training Result	28
5.6	Validation Batch	29
5.7	DeepSORT Tracking	30
5.8	Deep SORT Algorithm	32
5.9	Tracking Frame 0	33
5.10	Tracking Frame 1	33
5.11	Red light violation	35
5.12	License Plate Localized by YOLOv5s	36
5.13	Pre-processing Detected License Plate ROI	38
5.14	Overview of Character Segmentation Steps	39
5.15	Histogram of Gray Image	41
5.16	Grayscale License Plate	41
5.17	Equalized Histogram Plot	41
5.18	Histogram Equalized Gray Image	41
5.19	Contrast Limited Adaptive Histogram Equalization	41
5.20	Character Segmentation Stage 1	44
5.21	Horizontal Profile Projection	45
5.22	Vertical Profile Projection	45
5.23	Segmented Characters, final stage	47

5.24	Character Recognition Process Overview	48
5.25	Segmented Character	49
5.26	Segmented Characters after filtering	49
5.27	Character Closing	50
5.28	Character Erosion	51
5.29	Otsu's Thresholding Character	52
5.30	Cleaned Character	53
5.31	Character, Thinned	54
5.32	Convolution Operation	56
5.33	Padding	57
5.34	Maxpool layer	57
5.35	ReLU Activation Function	58
5.36	Sigmoid Activation Function	59
5.37	Fully Connected Layer	59
5.38	Categorical Cross-Entropy Loss	60
5.39	Model Architecture	64
5.40	Model Summary	64
5.41	Train/Validation Model Error and Accuracy	67
5.42	Model Confusion Matrix for Test Set	68
5.43	Model Confusion Matrix for Validation Set	69
5.44	Test Classification Report for Model	70
5.45	Validation Classification Report for Model	71

List of Tables

2.1	Devaganari Zonal License Plate Characters	14
2.2	Province number and names	15
5.1	Confusion Matrix Table	23

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CCTV	Closed Circuit Television
DNN	Deep Neural Network
FPS	Frames per second
HOG	Histograms of Oriented Gradients
OCR	Optical Character Recognition
OpenCV	Open-Source Computer Vision Library
PC	Personal Computer
R-CNN	Region Based Convolutional Neural Network
RGB	Red, Green, and Blue
SSD	Single Shot Detector
SVM	Support Vector Machine
WHO	World Health Organization
YOLO	You Only Look Once

1. Introduction

The study of computer vision focuses on simulating some of the complexity of the human visual system so that computers can recognize and analyze items in pictures and videos in a similar manner to how people do. Until recently, computer vision only worked in limited capacity. Artificial intelligence (AI) has made enormous strides in recent years, and is now capable of outperforming humans in various tasks involving object detection and object classification. This is due to developments in deep learning, neural networks, and artificial intelligence. Computer vision has a wide range of applications, one of them is traffic monitoring and road safety. Today, computer vision and machine learning strategies utilizing artificial intelligence provide fascinating and encouraging remedies for boosting road safety and traffic monitoring.

1.1 Background

Accidents on the road are a major problem in today's world. Every year, several countries invest millions of dollars to reduce traffic accidents. To keep road users from being badly wounded or killed, many road safety systems and measures are in place. Nevertheless, countries face a significant number of vehicle accidents every day. The majority of these incidents are caused by persons who are unwilling to respect traffic laws and a lack of supervision. According to research, improvements to road infrastructure, particularly design standards that include the safety of all road users, are crucial to keeping roads safe.112 countries have national design standards for the management of speed[1]. But infrastructure alone cannot make the roads safer without monitoring and adherence to traffic rules. The problem is particularly acute in countries like Nepal where people tend to follow rules only when traffic police are watching. If there was a system that could detect traffic rules violations using CCTV cameras, it would definitely contribute to road safety. Our focus for this project will be in Kathmandu, capital city as well as the largest city of Nepal. We intend to focus our work in and around the ring road of Kathmandu Valley.

1.2 Problem Statements

According to the latest WHO data published in 2020 Road Traffic Accidents Deaths in Nepal reached 4,654 or 2.90 % of total deaths[2]. The age adjusted Death Rate is 20.65 per 100,000

of population ranks Nepal 72 in the world. In the fiscal year 2077/078, there were 9545 recorded road accidents out of which 229 were serious accidents, while 7095 were normal accidents [3]. In fact, more people die on Nepal's highways every year than in all natural disasters combined [4]. Nearly 1,800 people were killed in road traffic accidents all over Nepal in 2014-15. The reason for the high fatality rate is poorly maintained and risky mountain roads, overloading of vehicles, lack of road discipline and poor training of drivers [5].

Given the situation, something must be done to decrease the frequency of road accidents. We want to focus on one of the main causes of road accidents i.e., lack of road discipline. There are rules in place, traffic police do their best to enforce them, but it's the attitude of the people we need to change. If people knew that city roads are equipped with cameras, and their actions are being monitored it would encourage them to play by the rules, otherwise face legal consequences.

1.3 Objectives

The main objectives of this project are as follows:

- To detect traffic light violation.
- To recognize motorcycle, car, bus, truck, and license plates.
- To recognize the characters of detected license plates.

1.4 Scope

The idea presented in the project can be used to monitor traffic rules violations using road side camera. People often tend to ignore rules, and regulations whenever they feel like there won't be any consequences. Many road accidents are caused in Nepal every year by sheer ignorance. It not only compromises the safety of the driver but also the safety of other people on the road. Especially in the Kathmandu valley, traffic jams during rush hour have become a norm. Undoubtedly, there is poor road infrastructure in the valley, however it is not the sole cause of the problem. Lack of road discipline has a significant role in it, a simple lane violation has caused huge traffic jams in the past. Police are finding traffic management increasingly complex as the number of vehicles increases every year. Government has prioritized installation of CCTV cameras for security, and surveillance in recent years. According to Nepal police records, there are 1249 cameras in the valley [6]. Implementation of computer vision based automatic traffic violation systems utilizing the

footage from these already installed cameras will increase their utility as well as help in betterment of overall road safety.

2. Literature Review

2.1 Related work

Traffic management must rely on a system for estimating traffic parameters in real-time. Currently, there is a lot of work going on to detect traffic parameters in the research field. Parallel process techniques to detect traffic offenses in real time-time traffic [7]. Their work implemented real-time traffic violation detection in a monitoring stream using simultaneous video streams from multiple cameras. The system was designed to detect three types of vehicles- speeding vehicles, blacklisted vehicles, and plate-cloned vehicles. The system was implemented and evaluated using both real and synthetic traffic data.

Beymer [8] proposed a model that deals with traffic and lighting conditions while using segmentation, classification, and tracking methodologies. The model focuses on vehicle segmentation and tracking and the computation of traffic parameters from tracking data. They have used a feature-based traffic tracking approach to track vehicles under congestion. The system tracks vehicle sub-features, which makes it less susceptible to partial occlusion. A network of C40 DSP (Digital Signal Processor) chips linked to a host PC has been used to construct a real-time version of the system. Wang [9] proposed an approach to use an enhanced background-updating algorithm and feature-based tracking method. In their paper, they have used video-based traffic detection through an improved background-updating algorithm, then tracking the moving vehicles by a feature-based tracking method.

Zhu [10] proposed a vehicle detection and tracking volume statistics algorithm based on an improved single Gaussian model. The algorithm has three sections: moving target detection, shadows suppression, and traffic volume count. The single Gaussian model detects moving vehicles while shadow suppression is performed in the RGB feature space. Traffic volume was calculated in the virtual lanes. Deng [11] proposed a model that can be used for object segmentation, classification, and tracking methodologies to know the real-time measurements in urban roads. The model used the background subtraction method to detect stationary or slow-motion objects. Experimental results showed a diminishing 20% degradation of infrastructure capacities.

Wen-juan [12] proposed a model that can locate and track the vehicles in a video and calculate the traffic flow, queue length, queue waiting time, the average speed, and other vital

parameters. The model used online learning mechanisms and optical flow to track objects. Experimental results showed the accuracy of daytime detection was 98%, and nighttime detection was 92%. The average speed detection of daytime was 95%, while nighttime was 90%.

Kim [13] compared different Artificial Neural Network (ANN) models for real-time vehicle type recognition. They compared deep learning-based object detection models R-CNN (Region Based Convolutional Neural Network), Fast R-CNN, Faster R-CNN, YOLO, and SSD (Single Shot Detector) in processing speed and accuracy for best performance. Faster R-CNN had less FPS (frames per second) with better accuracy, while SSD had less accuracy with better FPS. The YOLO model was a middle ground.

Shubho [14] Implemented a traffic offense detection module that analyzes traffic patterns and detects different types of traffic violations in real-time. The entire system is implemented using OpenCV Deep Neural Network (DNN) module. They have used YOLOv4 to detect vehicles on the roads with high accuracy. For motorbike riders without helmets, they have used a fast YOLOv4-tiny model. The DeepSORT algorithm is used to track vehicles in real-time. Obtained accuracies are 86% in YOLOv4 for Vehicle and 92% in YOLOv4-tint for Helmet Detection.

Nepali license plate recognition, [15] used Support Vector Machine (SVM) based learning and prediction on calculated Histograms of Oriented Gradients (HOG) features from each character. The system was evaluated on Nepali number plate dataset created by the authors. Evaluation accuracy of the number plate character dataset was obtained as; 6.79% of average system error rate, 87.59% of average precision, 98.66% of average recall and 92.79% of average f-score. The accuracy of the complete number plate labeling experiment was obtained 75.0%. Accuracy of the automatic number plate recognition was greatly influenced by the segmentation accuracy of the individual characters along with the size, resolution, pose, and illumination of the given image.

Bhalerao [16] put forth an effective design for forecasting driving behavior by the driver behavior analysis and the driver behavior. The driver's background data was analyzed using logistic regression, the driver's behavior while driving was analyzed using computer vision based on various environmental factors, and drowsiness was analyzed using SVM. The given factors were analyzed, and HMM was utilized to forecast driving behavior. The proposed model identified faces more accurately with the accuracy of 85% compared to the traditional method. The experimental results were measured based on the sensitivity, specificity and precision using the lfw-deep funneled dataset.

Ghandour [17] implemented four machine learning classification methods and compared them to identify drivers' behavior and distraction situations based on real data corresponding to various behaviors such as aggressive, drowsy and normal. Due to its flexibility and effectiveness in using clone decision trees, the Gradient Boosting method outperformed other commonly used classifiers like Logistic Regression, ANN, and Random Forest Classifiers. Driving behavior was examined using a large dataset called UAH-DriveSet, which was collected from six distinct drivers and cars.

2.2 Related theory

2.2.1 History of YOLO

The history of the YOLO (You Only Look Once) object detection algorithm can be traced back to 2016 when it was first introduced by Joseph Redmon in a paper titled "You Only Look Once: Unified, Real-Time Object Detection". The paper proposed a single-shot object detection algorithm that could detect objects and classify them in real-time using a single neural network. This approach was faster and more efficient than other object detection algorithms that used multiple networks and stages [18][19].

Redmon's original YOLO algorithm had some limitations in terms of object localization accuracy, but subsequent versions of the algorithm addressed these issues. In 2017, Redmon and his team released YOLOv2, which used a new detection architecture called Darknet-19 and introduced features like anchor boxes to improve object localization accuracy. YOLOv2 also achieved faster detection speeds than the original YOLO algorithm [19].

In 2018, Redmon and his team released YOLOv3, which further improved object detection accuracy and added new features like multi-scale detection and improved feature extraction. YOLOv3 was faster and more accurate than its predecessors, and it set new benchmarks in object detection performance [19] [20].

The development of YOLO did not stop there. In 2020, YOLOv4 was released, which introduced new techniques like Mish activation function and Spatial Pyramid Pooling (SPP) block to further improve the algorithm's performance. YOLOv4 achieved state-of-the-art results in terms of object detection accuracy and speed, and it set new benchmarks in the field [18] [19].

The most recent version of YOLO is YOLOv5, which was released in 2020. YOLOv5 introduced a new anchor-based prediction system, a lightweight architecture, and a focus on model compression to reduce memory usage. YOLOv5 is faster and more accurate than its

predecessors, and it has been used in a wide range of applications, including autonomous driving and robotics.

In conclusion, the history of YOLO object detection has been characterized by a series of improvements and innovations that have led to faster and more accurate object detection algorithms. From its original release in 2016 to the most recent version of YOLOv5, the algorithm has continued to evolve and set new benchmarks in the field of object detection. State-of-the-art performance on several object detection benchmarks while being much faster than previous versions.

Overall, YOLO has become one of the most widely used object detection algorithms due to its speed, accuracy, and simplicity. YOLO has been used in a variety of applications, from detecting objects in images and videos to tracking objects in real-time. The algorithm has undergone several improvements over the years, and the latest version, YOLOv5, is a significant improvement over previous version.

Here is a step-by-step algorithm for the YOLO (You Only Look Once) object detection algorithm:

1. Input: a digital image I with width W and height H .
2. Preprocessing: resize the image I to a fixed size and normalize its pixel values to range $[0, 1]$.
3. Divide the image into a grid of $S \times S$ cells, where S is determined, depending on the network architecture.
4. For each cell, predict B bounding boxes with confidence scores and (class probabilities for each bounding box) for K object classes, using a convolutional neural network (CNN) model.
5. Calculate the confidence score for each bounding box by multiplying the conditional class probability with the intersection over union (IoU) between the predicted box and the ground truth box, if any.
6. Apply a threshold to the confidence scores to remove low-confidence predictions.
7. Non-maximum suppression (NMS): for each class, remove overlapping bounding boxes by keeping only the one with the highest confidence score. This results in a final set of predictions for all classes.

8. Output: a list of predicted bounding boxes with their corresponding class labels and confidence scores.

Object detection algorithms in YOLOv5 are now quicker and more accurate thanks to a number of advancements and breakthroughs. At the time of its initial release in 2016 to the most recent iteration of YOLOv5, the algorithm has advanced and established new standards for object detection. For our use case in this project, we used the YOLOv5s model.

2.2.2 YOLOv5 Architecture

The architecture of YOLOv5 consists of a backbone network called CSPDarknet53, which is a variant of the Darknet architecture used in previous versions of YOLO. This network is composed of a series of convolutional and residual blocks that help to extract features from the input image. The CSPDarknet53 architecture was chosen for its efficiency and effectiveness in feature extraction, making it well-suited for real-time object detection applications.

Up to the day of writing this report, there is no research paper that was published for YOLO v5.

All of the YOLOv5 models are composed of the same 3 components: CSPDarknet53 as a backbone, SPP and PANet in the model neck and the head used in YOLOv4 [21].

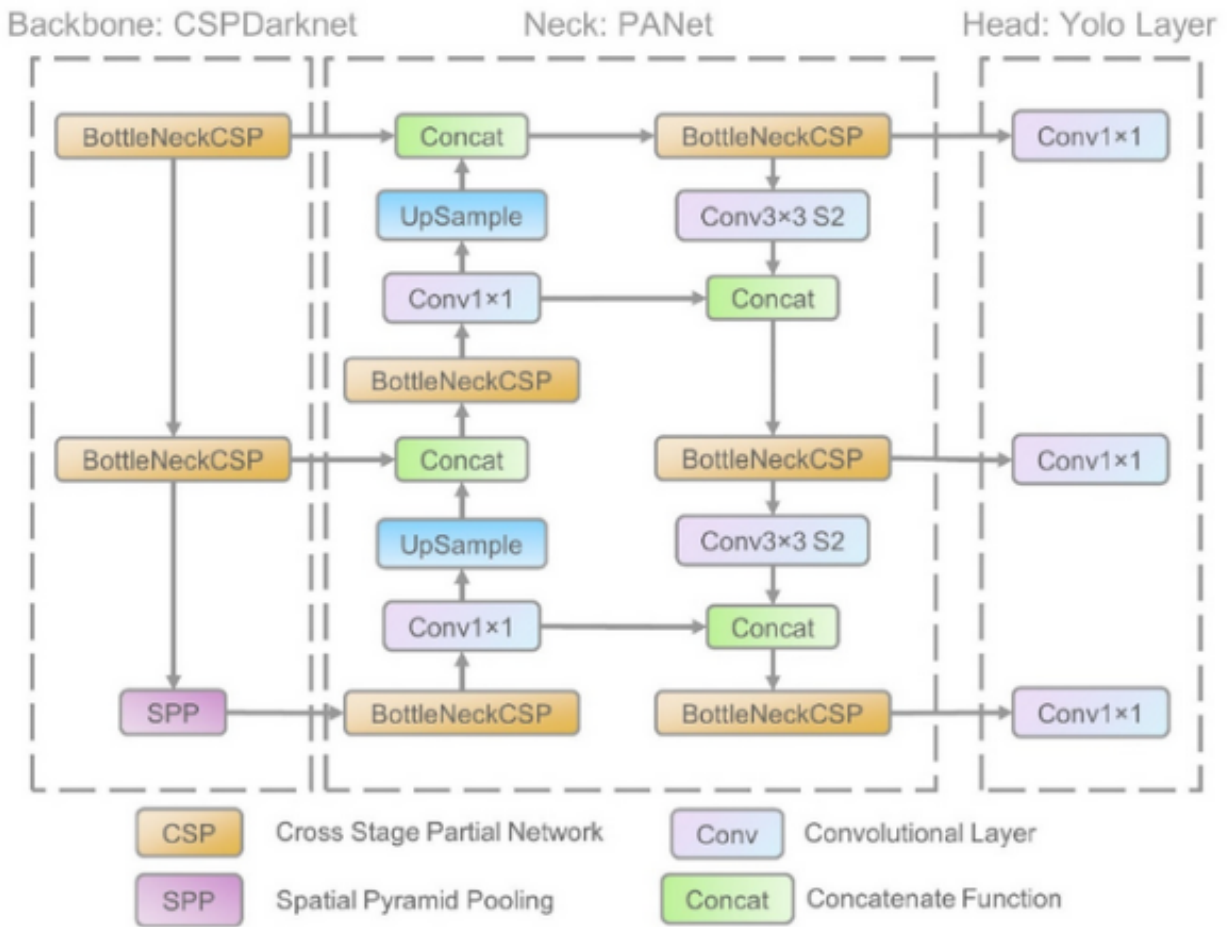


Figure 2.1: YOLOv5 Architecture

In addition to the CSPDarknet53 backbone, YOLOv5 also includes several new architectural features. One of the key additions is the use of a Spatial Pyramid Pooling (SPP) block, which helps to capture multi-scale features from the input image. This block is composed of several pooling layers that operate at different scales, allowing the model to capture information at different levels of detail [21].

Cross Stage Partial Network

YOLO is a deep network; it uses residual and dense blocks in order to enable the flow of information to the deepest layers and to overcome the vanishing gradient problem. However, one of the perks of using dense and residual blocks is the problem of redundant gradients. CSP-Net helps tackling this problem by truncating the gradient flow. According to the authors of [21]: CSP network preserves the advantage of DenseNet's feature reuse characteristics

and helps reduce the excessive amount of redundant gradient information by truncating the gradient flow.

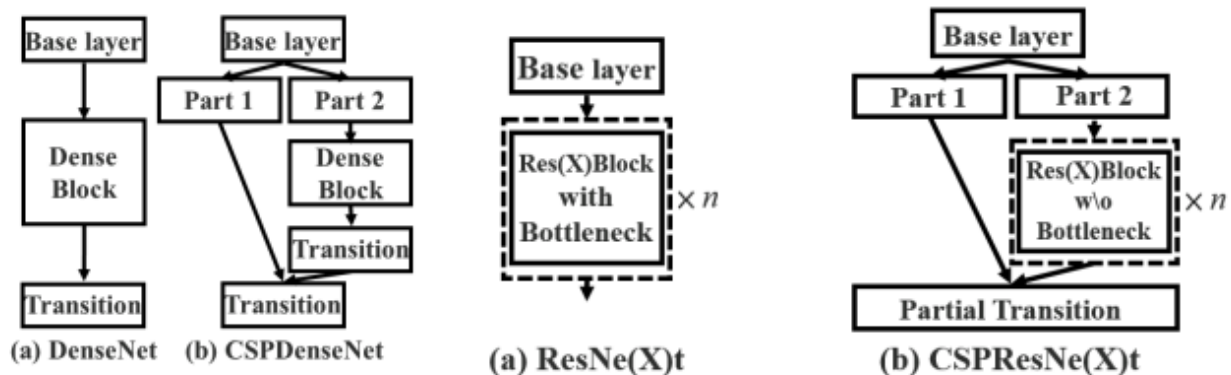


Figure 2.2: Applying CSPNet to ResNet and DenseNet

Head of the network

YOLOv5 uses the same head as YOLOv3 and YOLOv4. It is composed from three convolution layers that predicts the location of the bounding boxes (x, y, height, width), the scores and the objects classes. The activation function used in YOLOv5 is the Mish activation function, which has been shown to outperform other popular activation functions such as ReLU and Swish in terms of both training speed and accuracy.

The optimization function used in YOLOv5 is the Adaptive Moment Estimation (Adam) optimization algorithm, which is an extension of the stochastic gradient descent algorithm. This algorithm is well-suited for training deep neural networks and has been shown to converge quickly and efficiently [21].

The cost function used in YOLOv5 is a combination of several loss functions, including the binary cross-entropy loss function for objectness prediction, the mean squared error loss function for bounding box prediction, and the cross-entropy loss function for class prediction. The model is trained using backpropagation with stochastic gradient descent, and the weights and biases of the model are updated using the gradients computed during the training process.

The final YOLOv5 model has 85.2 million trainable parameters, making it smaller and faster than previous versions of YOLO. This model achieves state-of-the-art performance in object detection and has been shown to outperform other popular object detection models such as Faster R-CNN and RetinaNet.

Overall, the YOLOv5 architecture and network structure are optimized for efficiency and accuracy, making it well-suited for real-time object detection applications like ours to detect vehicles and their license plate in a traffic. Ultimately, how well the model performs in our custom dataset is entirely based on how well we tune the hyperparameters for your model.

Hyperparameters in machine learning refer to the parameters that are not learned by the model during training but rather set by the user before training. These parameters can greatly affect the performance of the model and can be adjusted to improve its accuracy [22].

For example, in our case the hyper parameters that we had to tune were SGD momentum/Adam beta1, weight decay, warmup epochs, warmup bias learning rate and 26 more. These hyperparameters need to be set before the training process begins, and can be tuned through a process called hyperparameter tuning, which involves trying out different combinations of hyperparameters to find the best performing model.

2.2.3 Nepali Vehicle Classification and License Plate

In Nepal, all road vehicles with or without a motor (except bicycles) are tagged with a registration number. This is issued by the state-level Transport Management Office, previously by a zonal-level Transport Management Office, a government agency under the Department of Transport Management. The license plates must be placed in the front as well as back of the vehicle. The international vehicle registration code for Nepal is NEP. We are specially focused on recognition of Devnagari plates; therefore, we won't be discussing about emboss plates.

Vehicle classification: For the purpose of vehicle registration Vehicle & Transport Management Act, 2049 (1992) and Vehicle & Transport Management Rule, 2054 (1997) of Nepal [ref law], classifies vehicles into the following 5 main categories on the basis of size and capacity:

1. Heavy and medium-sized vehicle:

This includes bus, truck, dozer, dumper, loader, crane, Fire engine, tanker, roller, pick-up, van, mini bus, mini truck, minivan etc. having the capacity to carry more than 14 people (for passenger vehicle) or more than 4 tons (for cargo vehicle).

2. Light vehicle:

This includes car, SUV, van, pick-up, micro bus etc. having the capacity to carry less than 24 people or less than 4 tons.

3. Two-wheeler:

This includes vehicle having two wheels like motor cycle, scooter etc.

4. Tractor and power-trailer

5. Three-wheeler:

This includes vehicle having three wheels like electric-safari, rickshaw etc.

The above-mentioned each category are further divided into 5 sub categories on the basis of ownership and service-type which are as follows:

1. Private vehicle: Vehicles which are for entirely personal purpose.

2. Public vehicle: Vehicles which are for public transport purpose.

3. Government vehicle: Vehicles owned by government agencies and constitutional bodies such as ministries, departments, directorates, along with the police, military, etc.

4. National Corporation vehicle: Vehicles which are registered under the name of public corporations that are fully or partially owned by the government fall under this category.

5. Tourist vehicle: Vehicles which are registered for tourist transport.

There are three types of license plate currently in use; emboss plate, Devnagari provincial, and Devnagari Zonal. This classification is made for the sake of convenience only. The emboss number plate adoption has been very slow, only about 32 thousand vehicles have emboss number plate installed [Kantipur Article]. Our custom yolov5 model can detect both emboss, and Devnagari plates, however due to less representation in the dataset, emboss plate recognition can fail sometimes. Since, the majority of vehicles in Nepal are yet to switch to emboss system, and it's likely to take few more years practically speaking despite legal obligation, we have focused on Devanagari plates only.

Devnagari Zonal Plates:







The previous system of the license plate of Nepal consisted of four parts composed of letters (L) and numbers (N) in the L N L NNNN format:



Figure 2.3: Zonal Plate Format

- L: indicates the zonal code, signifying the zone in which the vehicle is registered.
- N: is a 1- or 2-digit number which is prefixed when the four-digit number runs out from the last part.
- L: indicates vehicle category, whether it is a privately owned vehicle, public commercial, governmental, etc., as well as whether it is a heavy vehicle, medium-sized vehicle, or a light vehicle.
- NNNN: signifies four digits running in sequence.

Table 2.1: Devanagari Zonal License Plate Characters

Numbers		Zonal Representation		Vehicles' Category				
				Plate Color		Heavy Size	Middle Size	Light Size
0	०	मे	Mechi	Private		क	च	प
1	१	को	Koshi	Government		ग	झा	ब
2	२	स	Sagarmatha	Public		ख	ज	थ
3	३	ज	Janakpur	Diplomatic		सि डी	सि डी	सि डी
4	४	ना	Narayani	Tourist		य	य	
5	५	बा	Bagmati	Public/National Corporation		घ	ञ	
6	६	ग	Gandaki					
7	७	लु	Lumbini					
8	८	ध	Dhaulagiri					
9	९	भे	Bheri					
		रा	<u>Rapti</u>					
		क	Karnali					
		से	Seti					
		मा	Mahakali					

Devanagari Provincial Plates

Vehicle Transport Management Rule, 2054 (1997) was amended on 2075/07/08 to start issuing license plate on provincial names. The step was taken after dissolution of fourteen zones, and establishment of seven provinces. The provinces didn't have name at the beginning, they were called Province1, to Province 7. Provinces replaced the zonal code from license plate with province names, and new format of Devnagari plates were defined.

Table 2.2: Province number and names

Provinces Number	Province Names
प्रदेश १	कोशी प्रदेश
प्रदेश २	मधेश प्रदेश
प्रदेश ३	बागमती प्रदेश
प्रदेश ४	गण्डकी प्रदेश
प्रदेश ५	लुम्बिनी प्रदेश
प्रदेश ६	कर्णाली प्रदेश
प्रदेश ७	सुदूरपश्चिम प्रदेश

The new format is: L# NN NNN L NNNN, where # can be either letter or number.

- L indicates the provincial code, signifying the province in which the vehicle is registered. It is either province number or province name as shown in above table.
- #: Before provinces were named, it used to be a number from 1-7, now it is replaced by Nepali word for province ().
- NN: Indicates the serial number of Transport Management office inside the province where the vehicle is registered. The number is given by Transport Management Department.
- NNN: is a 3-digit number which is prefixed when the four-digit number runs out from the last part. Extra zeros are added to maintain the format.
- L: indicates vehicle category, whether it is a privately owned vehicle, public commercial, governmental, etc., as well as whether it is a heavy vehicle, medium-sized vehicle, or a light vehicle.
- NNNN: NNNN signifies four digits running in sequence, extra zeros are padded at the front to maintain the format.



Figure 2.4: Provincial Plate Format

3. Methodology

Our system will monitor and detect traffic violations by processing the video as shown in the following figure. YOLOv5s model trained on our custom dataset detects motorcycle, car, bus, truck, license plate, and no-parking sign. The detected objects are tracked using DeepSORT and the object being tracked are analyzed for violation, motorcycle, car, bus and truck. Any vehicles overrunning traffic light will be annotated by red bounding box in the frame. For every license plate detected in a frame, we keep track of id, given by DeepSORT, and send it for recognition if it is continuously seen for three consecutive frames. The license plate is fed to character segmentation module, and the segmented characters are sent for recognition. The output given by the character recognition model is annotated along with the plate in the video frame.

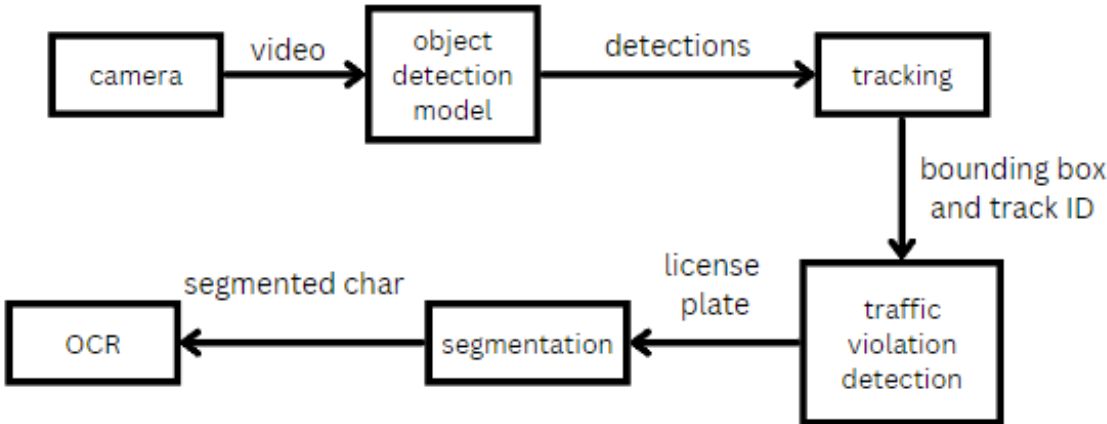


Figure 3.1: Project Methodology

4. System design

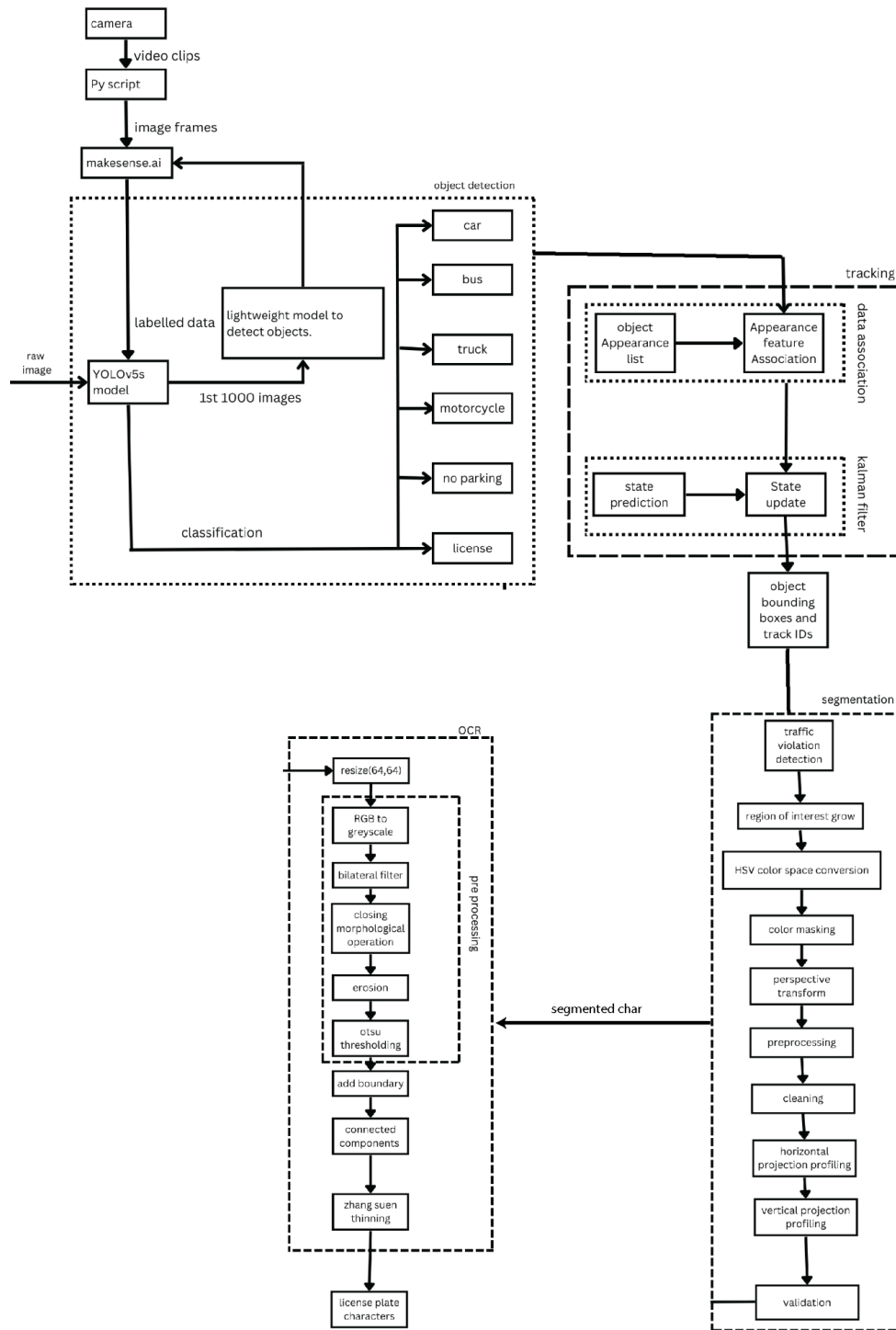


Figure 4.1: Designed System

5. Results & Discussion

5.1 Object Detection

Before the image is fed to the YOLOv5 CNN model in our project the data is preprocessed.

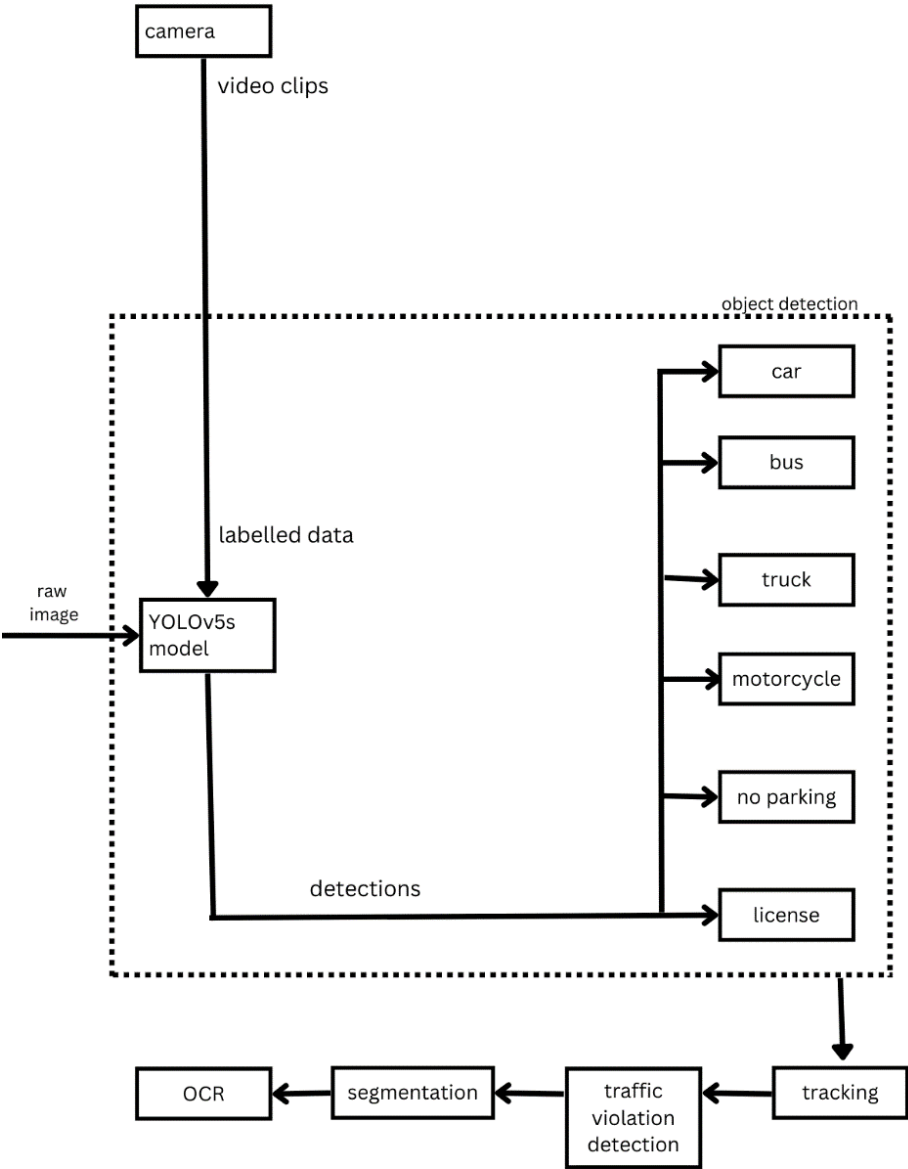


Figure 5.1: Object Detection

The general practices for preprocessing were practiced and for robustness data augmentation was implemented too, but before we dive into data preprocessing and its augmentation we need to talk about the methods and hurdles we faced during data acquisition.

5.1.1 Data Acquisition

Data had to be collected for training dataset and to collect data for creating a dataset for YOLOv5 to detect vehicles and license plates, we used a NIKON camera body with a 65mm lens to capture video and we used a tripod to ensure stability and accuracy in capturing the images. We positioned the camera at various angles and heights to capture different views of the vehicles. We ensured that each image contained at least one vehicle, and the license plate was visible, which was especially challenging due to the high volume of traffic on the roads of Nepal.

When using a camera to collect data, it is important to be aware of the artifacts that might be introduced to the images and data. For instance, the depth of field blur might cause certain parts of the image to be out of focus which might make it hard to see the characters of the license plate. The motion blur might result from the movement of the vehicle or camera and could also lead to image distortions. These artifacts can significantly affect the dataset's quality and accuracy, making it challenging to detect the vehicles and license plates.

In order to address the impact of these unwanted visual effects on the quality of our dataset, we took measures to appropriately tweak camera settings. We made sure that depth-of-field was appropriate by adjusting aperture size and avoided motion blur through control over shutter speed. Furthermore, lighting conditions were closely monitored in order to ensure optimal illumination so as not obscure any license plate details captured in images taken for this project's purposes. By carefully following such procedures during data collection processes overall resulting images obtained had minimal defects thus allowing YOLOv5 algorithms capabilities accurate detection of objects like vehicles and their associated label information with reliability being paramount throughout each step.

To stay consistent, we took the video in the same resolution and same fps throughout the process. The video was then parsed by a python script we wrote that would take the video and take images from it after every 6 frames and store the frames that will be used to create a labeled dataset as well as for test dataset.

The images collected thus far need to be labeled properly in order to turn it into useful data to create a dataset for our training model. In order to label the images, we used an

open-source tool called “makesense.ai” which is free to use under APLv3 license. It does not store any of our images as it runs locally and supports multiple output formats like YOLO, VOC XML, VGG JSON, CSV. More importantly it allows us to use a custom model that we trained on the 1st batch of 1000 images to label the next batch of images which makes our manually labeling job a bit easier. We used a lightweight YOLOv5 model to help us label the images we acquired.

5.1.2 Data Preprocessing

Regardless of all the measures we took while acquiring the data some unwanted artifacts are introduced in the image which hinders the accuracy of the model. To minimize its effects the images 1st goes through histogram equalization which is a commonly used technique for image preprocessing to adjust the image’s pixel intensities, enhance the contrast of the image and increase its visual features.

The process of histogram equalization involves computing a histogram of the image’s intensity values and then redistributing these values to create a more uniform distribution. The result is an image with enhanced contrast, where the dark and light areas of the image are more distinguishable. This technique is especially useful for images that have a limited range of pixel intensities, resulting in low contrast and making it challenging for YOLOv5 to detect objects accurately.

5.1.3 Data Augmentation

To expand the amount of data that is at disposal when training a model in machine learning, it’s common to apply a technique known as data augmentation. The basic idea behind this strategy involves deriving fresh sets of training samples by subjecting original datasets to diverse transformations. Such manipulations can include completely random rotations and flips or modifications involving crops with different brightness levels, contrast variations or even changes in saturation levels too. This method offers an opportunity for models’ exposure towards more complex dynamics present amongst input information thus enhancing their capacity for generalization during tests with unseen data inputs.

The implementation of data augmentation techniques can prevent overfitting, whereby the model becomes prone to retaining information in the training dataset rather than acquiring generalizable patterns. Generating supplementary samples for training purposes not only decreases the likelihood of overfitting but also bolsters performance efficiency. Moreover, by using data augmentation practices that balance class distribution within a given dataset

leads to adequate exposure of examples from all classes before modeling commences.

We followed generally followed practices for augmentation of data. The data augmentation techniques we used for our dataset are:

1. Rotation: We introduced rotation to the original images anywhere between - 15degree to +15degree.
2. Shear: We introduced shearing to the original images anywhere between +-15 degree horizontal, +-15 degree vertical.
3. Grayscale: 20% of the images we used for the dataset we applied grayscale filters.
4. Hue: The Hue value of the image was adjusted anywhere between +-25 degree.
5. Saturation: The saturation value of the images was changed to be anywhere between +-25%
6. Brightness: The brightness value of the images was changed to be anywhere between +-25%
7. Exposure: The exposure value of the images was changed to be anywhere between +-25%
8. Blur: Up to 0.75px blur was introduced to images
9. Noise: Up to 3% of pixel noise was introduced to images

After the dataset is ready, we split the dataset into training dataset, validation dataset and finally test dataset. We used YOLOv5 model from [22] to train on our custom dataset.

5.1.4 Model Training

YOLOv5s model was trained for detecting vehicles and number plates and no parking signs on a dataset of 2193 images of 6 classes which was augmented to extend our dataset to 5259 images and was split in the ratio of 70:20:10 for train, validation, and test respectively. Instead of initializing the weights randomly, pretrained weights on COCO2017 dataset was used. After training the model for 348 epochs, there was no improvement in the model, hence training process was terminated.

When we evaluate the accuracy of any model, we consider evaluation metrics like mAP (mean accuracy precision), recall and precision, confusion matrix, object loss, classification loss.

Confusion matrix

A confusion matrix is a table that is often used to evaluate the performance of a classification model. It provides a detailed breakdown of how many instances were correctly or incorrectly classified by the model, allowing us to measure the model's overall accuracy.

A confusion matrix is typically composed of four different metrics:

1. True Positives (TP): This refers to the number of positive instances that were correctly classified by the model.
2. False Positives (FP): This refers to the number of negative instances that were incorrectly classified as positive by the model.
3. False Negatives (FN): This refers to the number of positive instances that were incorrectly classified as negative by the model.
4. True Negatives (TN): This refers to the number of negative instances that were correctly classified by the model.

By combining these metrics, we can create a table that summarizes the performance of the classification model. The table is organized into a grid, with the actual class labels on one axis and the predicted class labels on the other.

Table 5.1: **Confusion Matrix Table**

	Actual Positive	Actual Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Using this table, we can calculate a number of different metrics that are useful for evaluating the performance of a classification model. For example:

1. Accuracy: This refers to the overall percentage of instances that were correctly classified by the model. It is calculated as $(TP + TN) / (TP + TN + FP + FN)$ [9].
2. Precision: This refers to the percentage of instances that were classified as positive by the model that were actually positive. It is calculated as $TP / (TP + FP)$ [9].
3. Recall: This refers to the percentage of actual positive instances that were correctly identified by the model. It is calculated as $TP / (TP + FN)$ [9].

- F1-Score: This is a weighted average of precision and recall, with a higher score indicating better performance. It is calculated as $2 * ((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$.

Overall, the confusion matrix is a powerful tool for evaluating the performance of a classification model. By breaking down the model's predictions into a detailed table, we can gain valuable insights into its strengths and weaknesses, and identify areas for improvement. The confusion matrix for our trained model and precision, recall graphs are as follows:

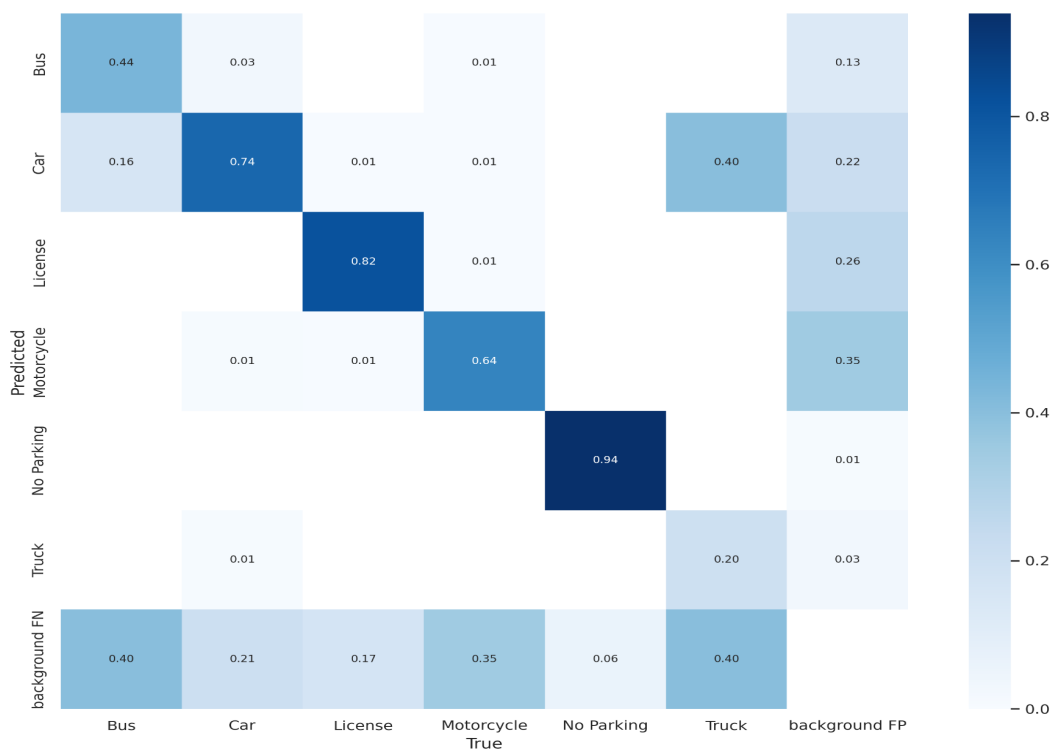


Figure 5.2: Confusion Matrix YOLOv5s Model

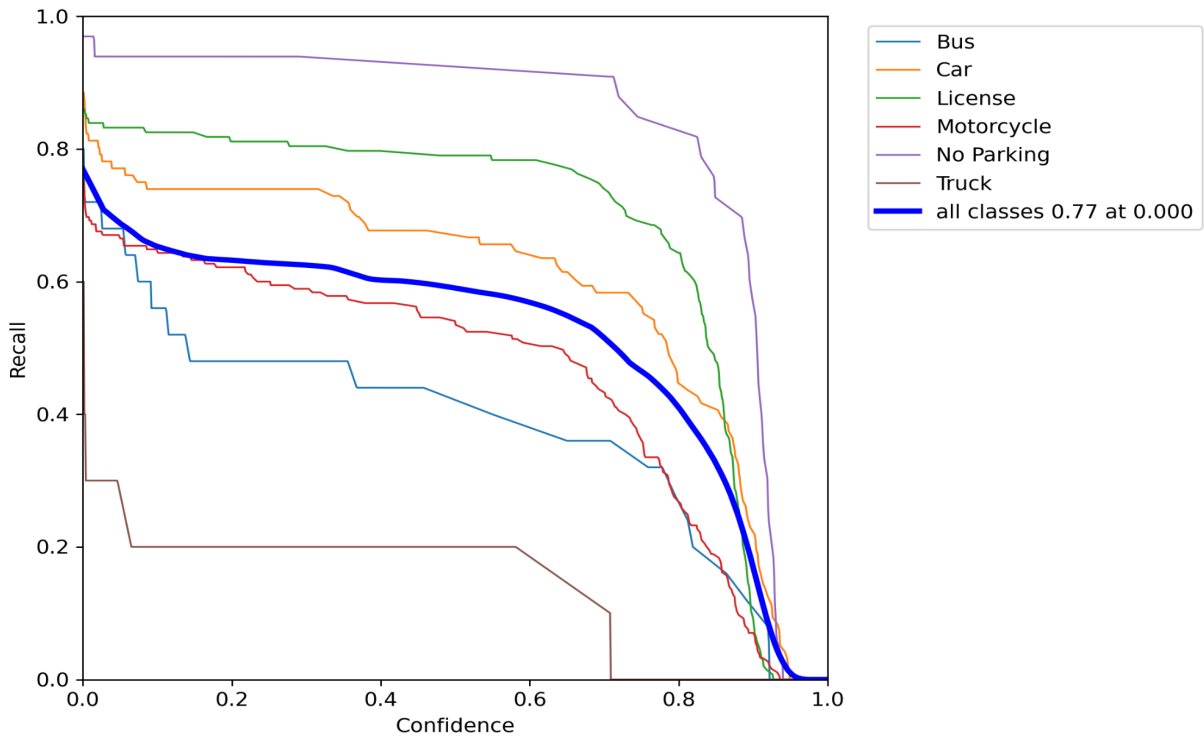


Figure 5.3: Recall Confidence Plot

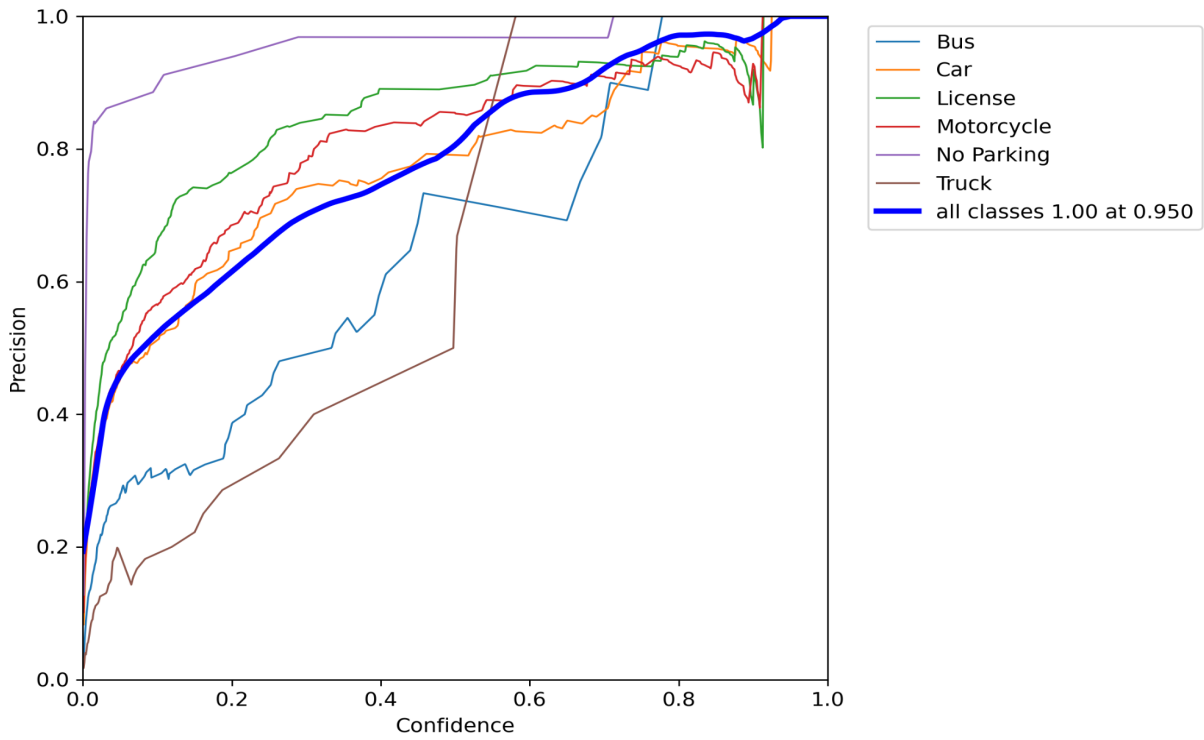


Figure 5.4: Precision Confidence Plot

mAP

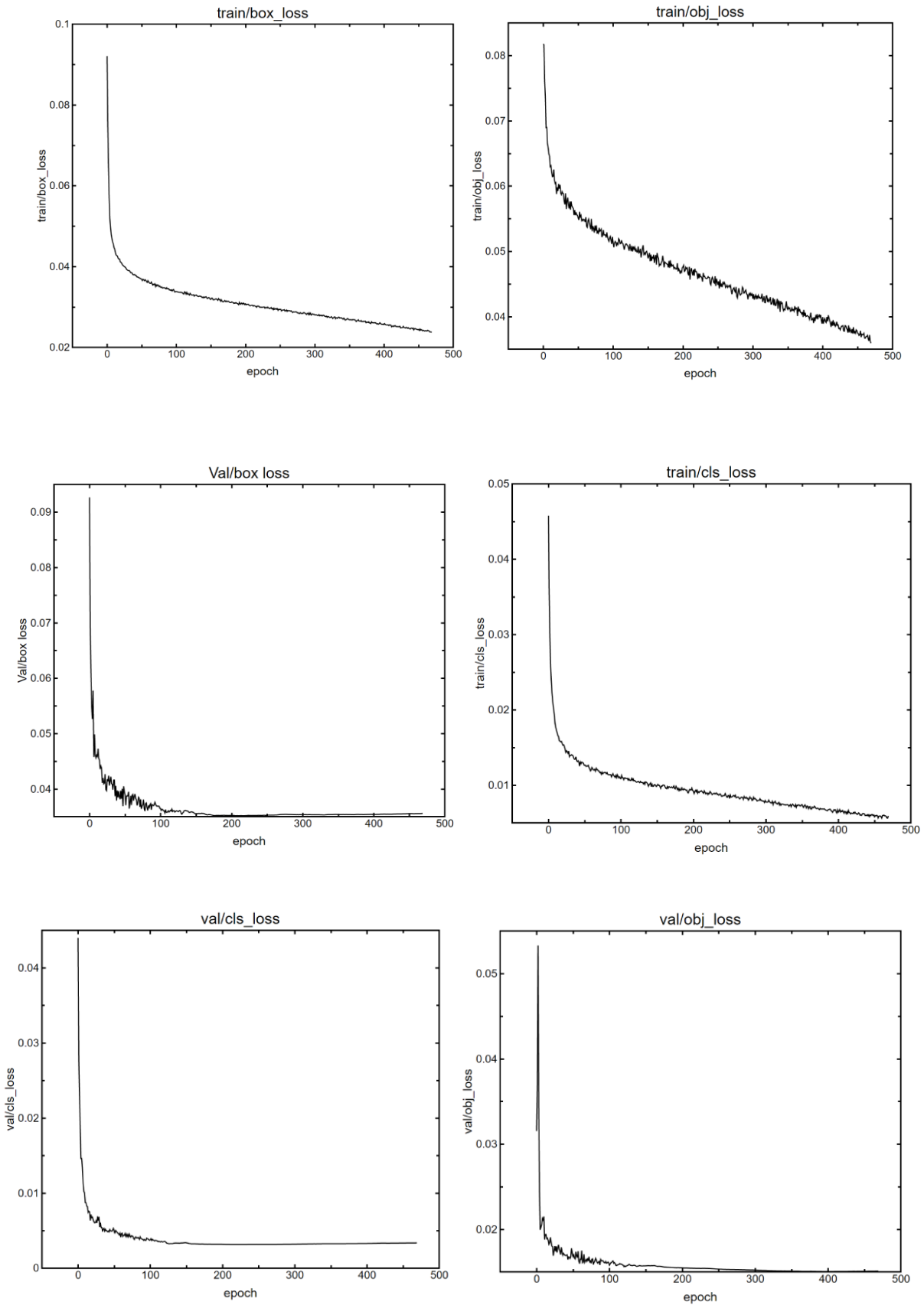
mAP(mean Average Precision), is an evaluation metric in object detection and image classification tasks. It measures the accuracy of an algorithm in identifying objects within an image and assigning a level of confidence to the identified objects.

The value of mAP is calculated by taking the average of the precision values at different recall levels. Here, the recall level represents the percentage of objects detected correctly out of all the objects that are actually present in the image. mAP at 0.5 is a specific threshold used in object detection tasks. It refers to the minimum level of intersection over union (IoU) required between the predicted bounding box and the ground truth bounding box for the detection to be considered correct.

In other words, if the predicted bounding box overlaps with the ground truth bounding box by at least 50%, then the detection is considered correct and contributes to the mAP at 0.5 score. This threshold of 0.5 is commonly used in object detection tasks as it strikes a balance between precision and recall.

Our model's performance upon referring to the mAP curve at 0.5 threshold shows that the average precision is more than 0.85 at approximately 460 epochs.

YOLOv5s Training Result



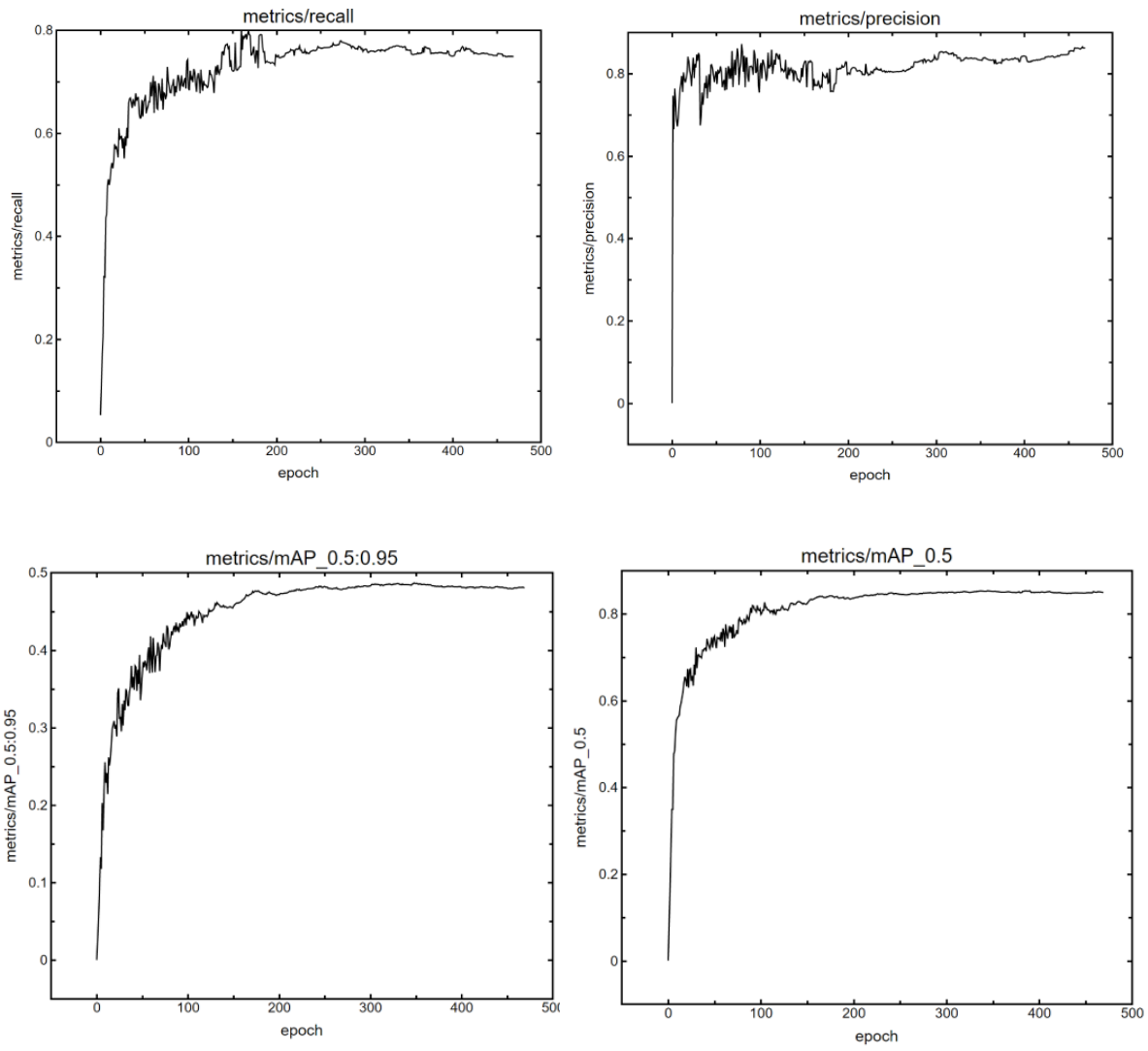


Figure 5.5: YOLOv5s Training Result



Figure 5.6: Validation Batch

5.2 DeepSORT Tracking

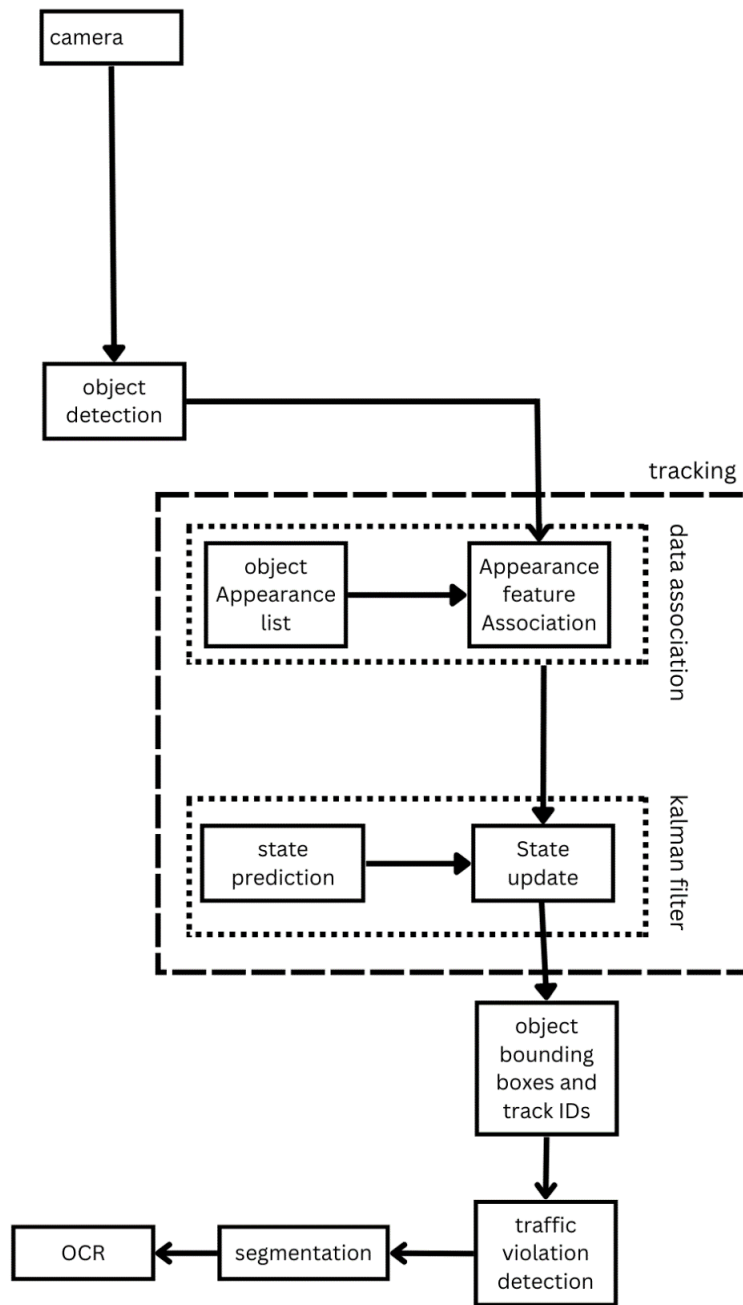


Figure 5.7: DeepSORT Tracking

Tracking is the process of predicting an object's position over the course of a video with the use of both spatial and temporal features. It is a two-step procedure:

- A detection module for target localization: It is responsible for locating the object in

the frame using an object detector, in our case YOLOv5s.

- A motion predictor: It is responsible for making motion predictions based on the object's past motion.

SORT (Simple Online Realtime Tracking)

The state of the objects being tracked is continuously calculated by the SORT algorithm using the KF algorithm. It uses Hungarian algorithm to accurately link the modeled objects that are being tracked, to new measurements acquired by an object detector.

KF Estimation

The SORT KF Estimation module uses the KF for each track to assign the state of the object's bounding box with a linear constant velocity model which is independent to other objects and camera motion. The track state vector ($x = [u, v, s, r, u', v', s']^T$) is composed by central bounding box coordinates (u and v), bounding box scale and aspect ratio (s and r), velocities of central bounding box coordinates (u' and v') and bounding box scale velocity (s'). Here, the aspect ratio is considered to be constant.

The KF Estimation module calculates a priori estimation of the state (x_k^-) for every active track assigned on the previous frame step. Then, an association between KF predicted bounding boxes and measurements is performed. The object state is updated by using the matched measurements on the KF algorithm. Existing tracks that were not associated with measurements, do not go through update stage, instead prior estimation of the state (x_k^-) is used as the state of the object in that frame.

Deep SORT method

DeepSORT is a computer vision tracking algorithm used to track objects while giving each one a unique ID. It is an addition to the SORT algorithm. It incorporates deep learning into the SORT algorithm by adding an appearance descriptor to reduce identity switches. This makes tracking more efficient.

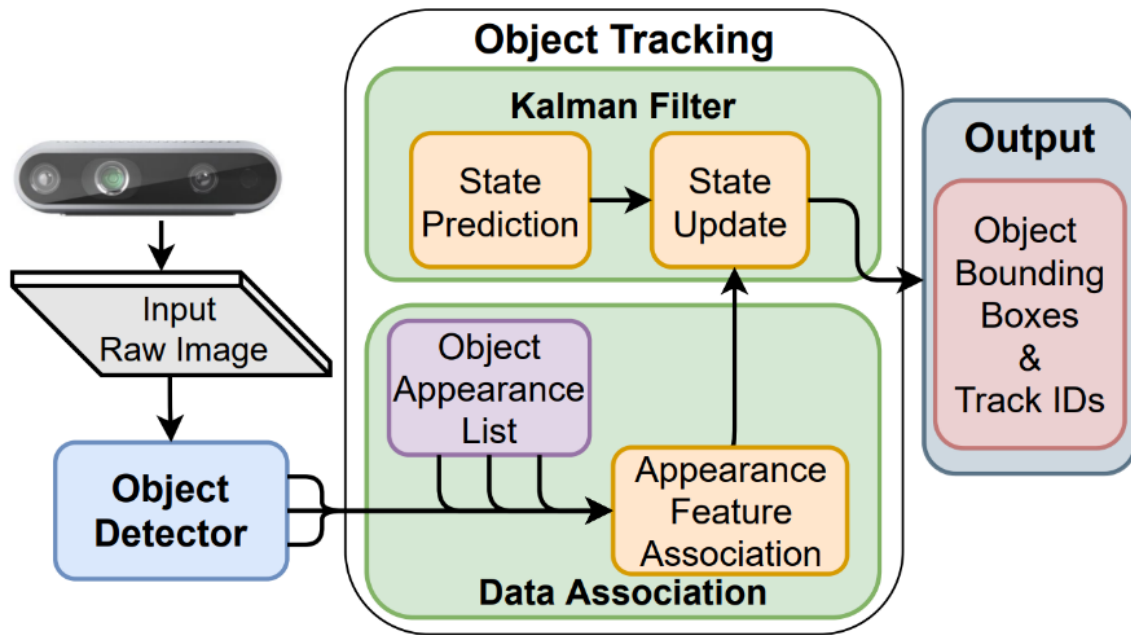


Figure 5.8: Deep SORT Algorithm

The SORT and Deep-SORT are the two well-known Kalman Filter - based tracking methods. The Deep-SORT is an extended version of the SORT algorithm as it uses DL based association metrics on the Data association module. Both the SORT and Deep-SORT have 3 main modules: KF Estimation, Data Association and Track Management.

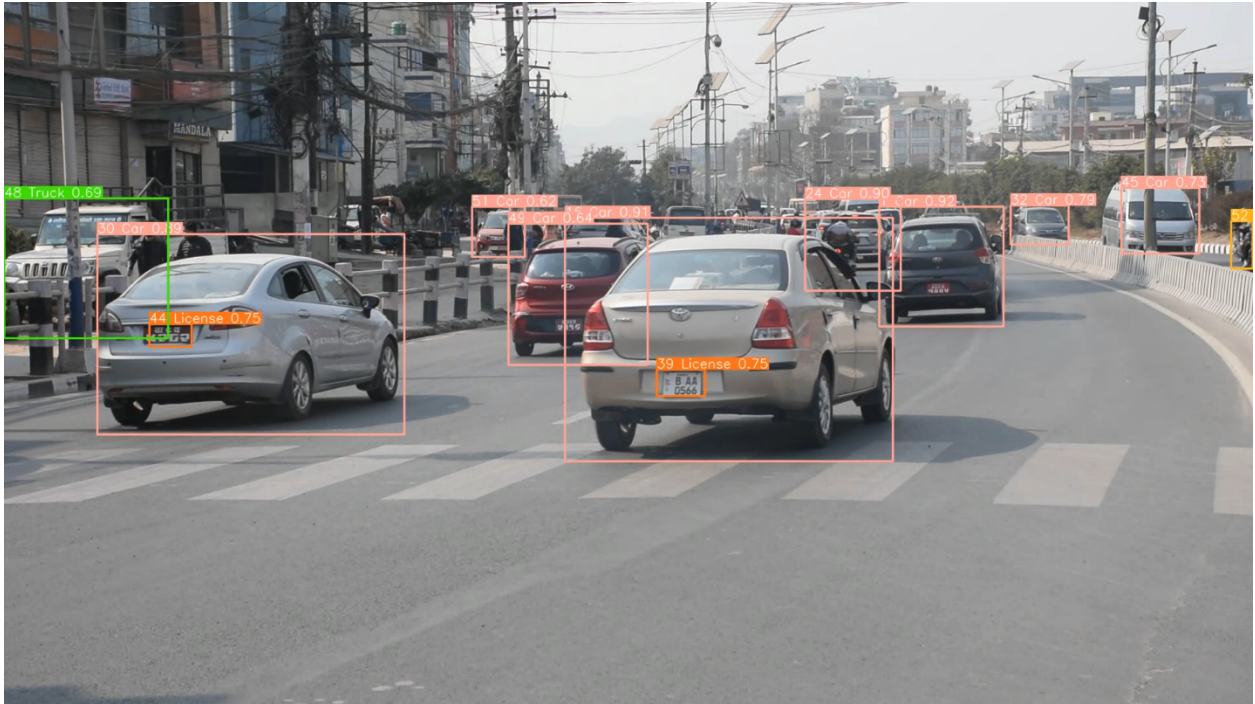


Figure 5.9: Tracking Frame 0

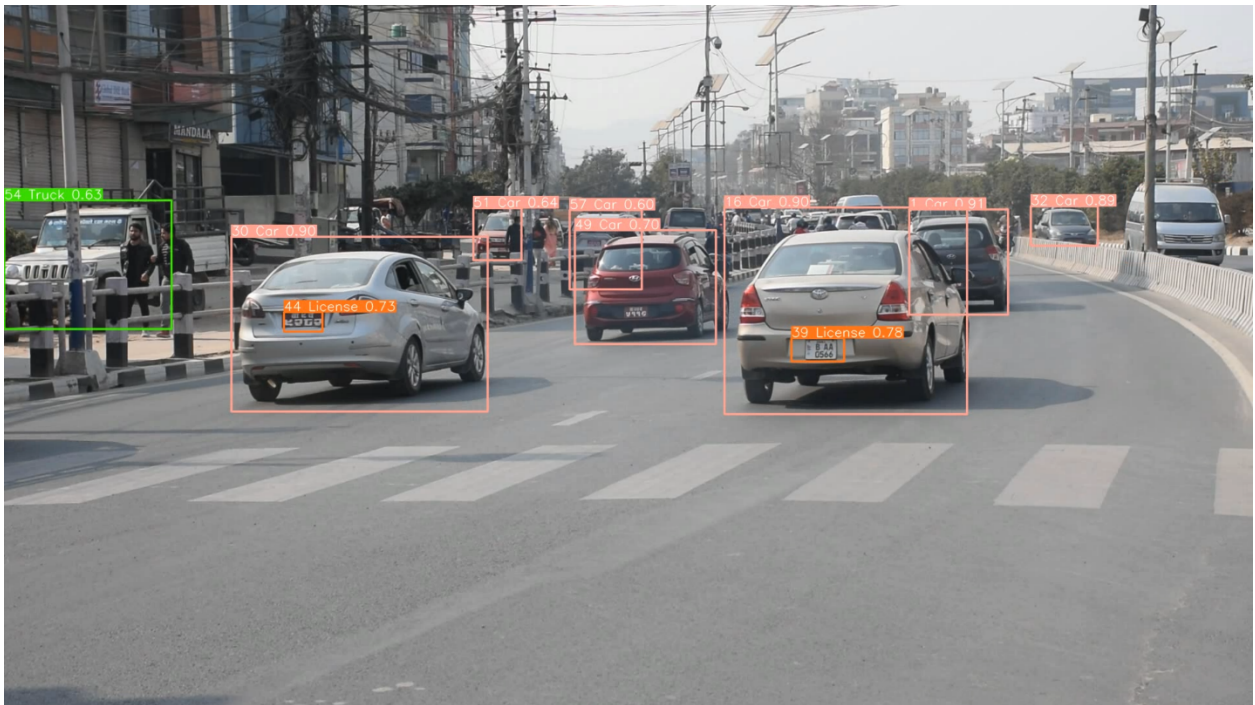


Figure 5.10: Tracking Frame 1

5.3 Violation Detection

A line is drawn on the video frame, if any vehicle crosses that line when the light is either yellow, or red, it is annotated with red bounding box. The python pseudo code of line crossing implementation is given below:

1. `tracker_list = { }`
2. For each frame in the video:
3. Draw (a, b) to (c, d)
4. `detections = All 'bounding-boxes' and associated 'tracker_ids' from DeepSORT`
5. For bounding-box, `tracker_id` in `detections`:
 - a. `x1, y1, x2, y2 = bounding-box`
 - b. `anchors = [(x1, y1), (x1, y2), (x2, y1), (x2, y2)]`
 - c. `triggers = []`
 - d. for (x, y) in anchors:
`v1 = (c - a), (d - b)`
`v2 = (x - a), (y - b)`
`cross_product = v1 x v2 < 0`
`state = True if cross_product else False`
Add 'state' to 'triggers'
 - e. If all 'triggers' are not same:
continue
 - f. `tracker_state = triggers [0]`
 - g. If 'tracker_id' is not in 'tracker_list':
Add 'tracker_id' and 'tracker_state' to 'tracker_list':
continue
 - h. If 'tracker_id' has same 'tracker_state' in 'tracker_list':
continue
 - i. Add 'tracker_id' and 'tracker_state' to 'tracker_list'

This the algorithm, we have proposed to determine whether the line was crossed or not. It needs to be tested. The scene should look something like in the following figure:

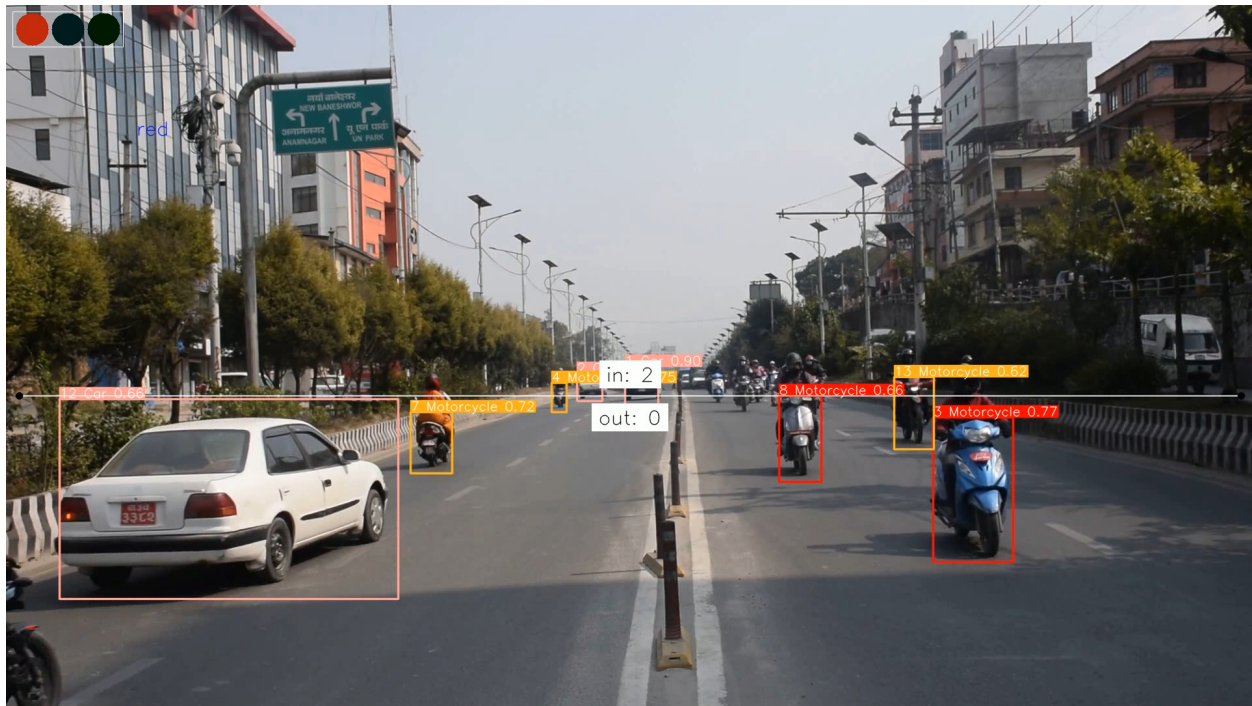


Figure 5.11: Red light violation

5.4 Plate Localization

Localization refers to identifying the location of an object in the image. In any ANPR system, plate localization is the first stage. There are many approaches to this essential task, before getting ready for plate recognition. Basic image processing techniques can be used in ANPR when done under regulated lighting settings with predictable license plate types. P. R. Sanap and S. P. Narote [23] have summarized various methods of license plate localization; histogram, morphological processing, texture, edge detection, and transformation are the bases for the number plate detection techniques.

Karthikeyan and V.J. Vijayalakshmi [24] used morphological operation for license plate localization. The algorithm used morphological operations on the preprocessed, edge images of the vehicles. Characteristic features such as license plate width and height, character height and spacing are considered for defining structural elements for morphological operations.

Traditional approach of plate localization using morphological operations, and four-point contour detection is difficult to generalize given the various types of license plate being used in Nepal right now. [25][26] Law clearly defines the size, color, character size and spaces

between lines and character, but people have taken liberty to use number plate that fits their vehicle. Introduction of emboss number plate has certainly brought some standardization, but its adoption rate has been very slow.

Given the pretext, the task of plate localization was done by training YOLOv5 on a custom dataset, both images and video, taken at various location and from multiple angles in Kathmandu Valley. YOLOv5 model trained on our custom dataset identifies motorcycle, car, bus, truck, no-parking sign, and license plates. The object detection step draws a bounding box around the detected license plate above a set confidence threshold. The plate candidate detected by the object detector is called the region of interest (ROI).



Figure 5.12: License Plate Localized by YOLOv5s

5.5 Processing ROI

Preprocessing the acquired image is the first stage in any image processing system. Preprocessing involves some actions on the image to improve the area of interest, and as a result, these procedures are wholly dependent on context. In our situation, the area of interest is the

vehicle's license plate, so we put the following processes into place to transform the image into one that can be processed further and improve the effectiveness of the segmentation algorithm. Preprocessing of ROI includes, region grow, HSV Color Space Conversion, color masking, and perspective transform.

1. Region Grow: The object detector may not always perfectly bound the license plate in every image or frame; therefore, the detected bounding box is increased by ten pixels if the image size is not exceeded.
2. HSV Color Space Conversion: Colors are described in terms of Hue, Saturation, and Value according to the Hue, Saturation, and Value (HSV) color space model. When color description is important, the HSV color model is frequently chosen over the RGB model. Similar to how the human eye perceives color, the HSV model defines color [15]. Whereas RGB depicts color as a mixture of primary colors, HSV characterizes color using more relatable comparisons like color, vibrancy, and brightness. Color definitions are contained within a hexcone and the coordinate system is cylindrical. The range of the hue value H is 0° to 360° . The saturation S ranges from 0 to 1 and indicates the degree of strength or purity. The brightness, which similarly varies from 0 to 1, is represented by value V.
3. Color Masking: The image in HSV color space is now ready to be masked by using appropriate color mask. Nepali license plate can come in multiple colors as discussed earlier depending on the ownership of the vehicle. Our interest of vehicle is private vehicles, which have red color plate with white characters. Thus, we masked the red color from the image. In HSV model, the Hue value from around 0° - 10° and 350° - 360° can be approximated as the red color. Using these range, we masked the red color regions from the images or frames [15].
4. Perspective Transform: The color masked image is sent for external contour detection, the contours are arranged by area, and the largest contour by area is approximated by a minimum rectangle. This is where the characters in the plate are located. Due to variation in camera position, and angle, the plate could be oriented in any direction. If such plate is sent for segmentation, the algorithm won't perform well [27]. The four coordinate points of the minimum rectangle found earlier is arranged in order: top-left, top-right, bottom-right, and bottom-left. A blank image whose width is the maximum distance between bottom-right and bottom-left x-coordinate or the top-right and top-left x-coordinate, and height is the maximum distance between the top-right and bottom right y-coordinate or top-left and bottom-left y-coordinate is created.

Now we have the size of new image, a set of destinations points are obtained in the same order: top-left, top-right, bottom-right, and bottom-left. To obtain a “birds eye view” or top-down view of the plate, a perspective transform matrix is calculated using OpenCV’s `getPerspectiveTransform ()` method, and this matrix is used to compute the perspective transform.

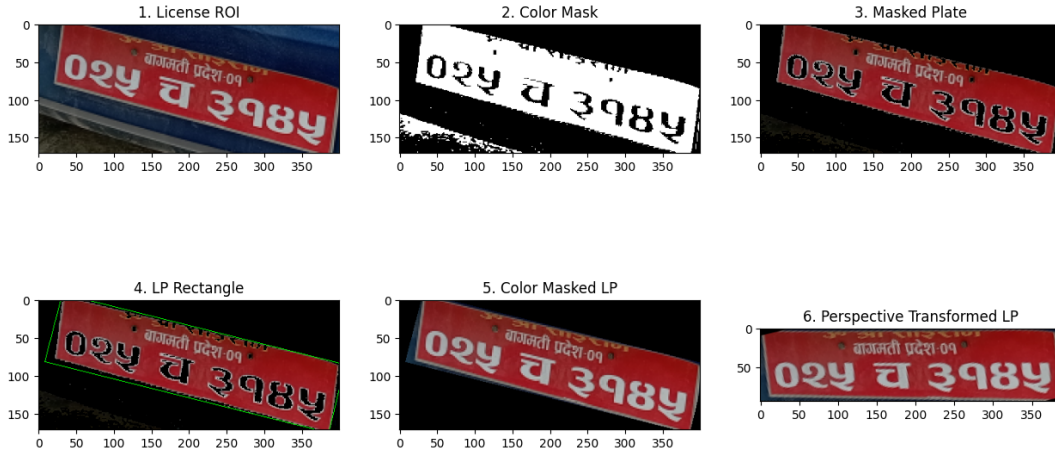


Figure 5.13: Pre-processing Detected License Plate ROI

5.6 Character Segmentation

The vehicles on the road are detected, and those vehicles violating the traffic rules are flagged, but it makes no sense if we cannot identify the offender. The license plate number or simply called vehicle number in common language, is unique to ever vehicle on the road. It conveys some additional information about the vehicle, such as vehicle type, and ownership [26] [25]. The detected license plate must be processed prior feeding it to a CNN trained on Nepali license plate characters for recognition. Character segmentation means extracting the characters from a given license plate.

Yungang Zang and Changshui Zhang developed a new algorithm [28] using Hough transformation and the prior knowledge in horizontal and vertical segmentation respectively. Advantages of this algorithm are; no need of rotation correction of plate images, influence of background is weakened and also the illumination variance.

Feng Yang, Zheng Ma, and Mei Xie [29] proposed a region growing based segmentation technique. Contrast stretching transformation is used to improve character areas, and then Laplacian Transformation is used to detect edges. With the aid of the region-growing algorithm, the locations of the potential regions are discovered. Using predetermined parameters,

the process of "region growing" groups pixels or smaller regions into bigger ones. The method was tested on 320 images with a success rate of 97.2%.

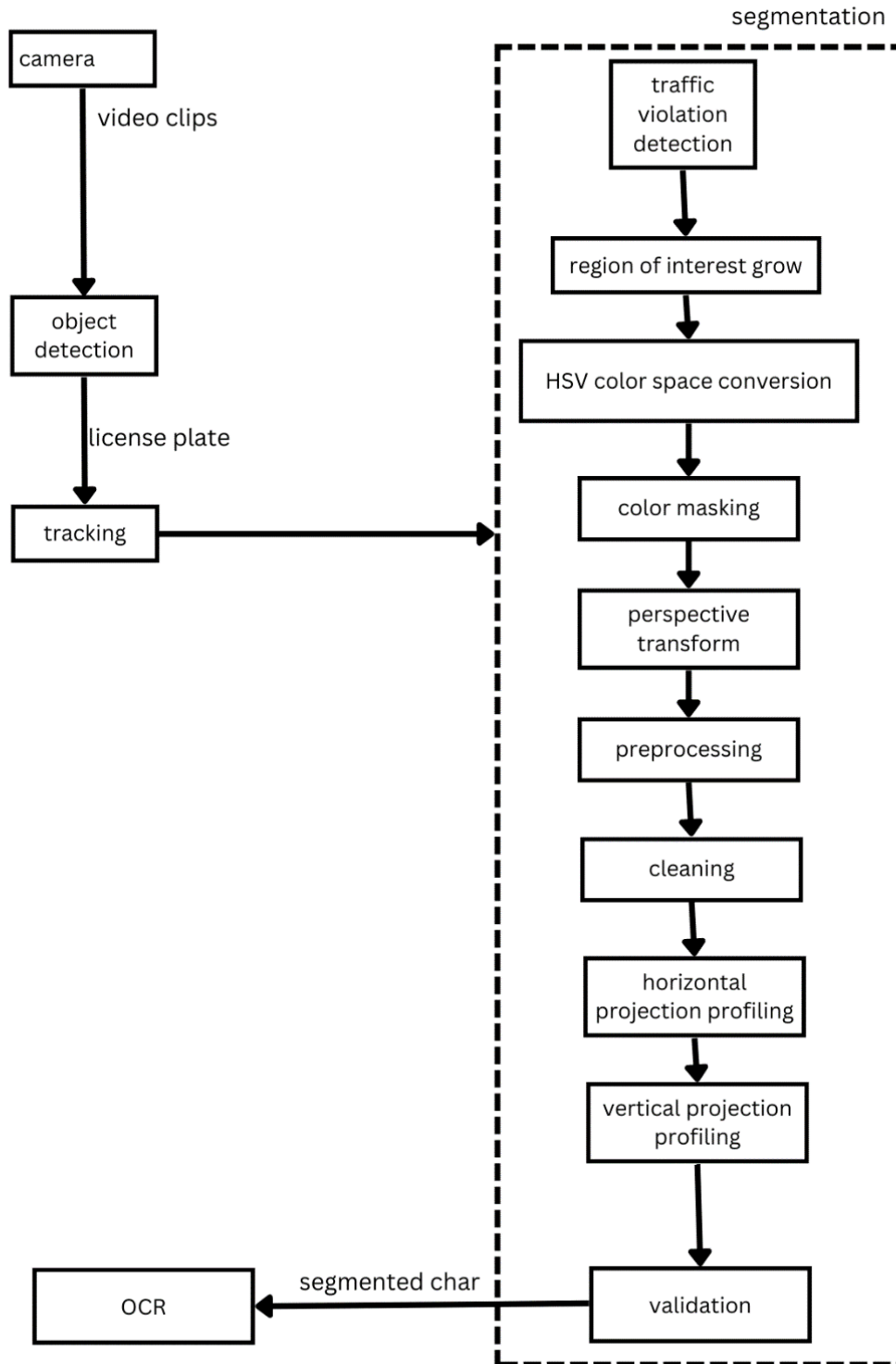


Figure 5.14: Overview of Character Segmentation Steps

Pre-processing

The object detector gives a region of interest, we process the region of interest and get the region of the plate where characters are present. The plate's perspective must be top-down view to ensure proper segmentation of characters. In the pre-processing step, we want to reveal the characters clearly from the background of the plate. Before extracting characters, raw plate image is fine-grained. It is the stage when unnecessary marks, noise, and blobs are removed. Dirt and dust on the plate, old discolored plates, and distorted characters cause problem in segmentation. The primary preprocessing steps are: histogram equalization (Clip Limited Adaptive Histogram Equalization), grayscale, median filter, morphological operation, and binarization (Otsu's method).

RGB to Grayscale Conversion

By adding the R, G, and B weighted sums, a 24-bit RGB image is transformed into an 8-bit grayscale image. The grayscale signal used to display images on monochrome televisions is chosen using the same weights as the NTSC color space [30]. The grayscale picture for the RGB image $f(x, y)$ is provided by the formula:

$$g(x, y) = 0.2989 * f_R + 0.5870 * f_G + 0.1140 * f_B \quad (5.1)$$

Where, f_R , f_G , and f_B are the red, green, and blue components of the RGB image $f(x, y)$, respectively.

Histogram Equalization

A histogram is a representation of frequency distribution of pixel intensity. This method usually increases the global contrast of many images, especially when the image is represented by a narrow range of intensity values. Global Histogram Equalization (GHE) is very simple and fast, but its contrast enhancement power is low.

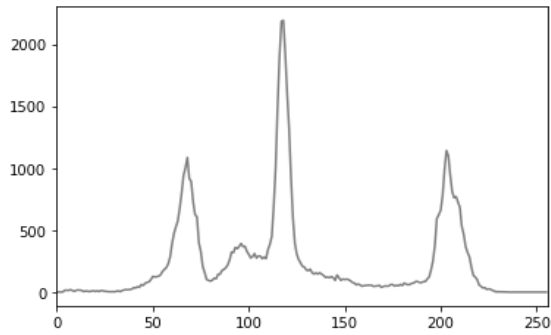


Figure 5.15: Histogram of Gray Image



Figure 5.16: Grayscale License Plate

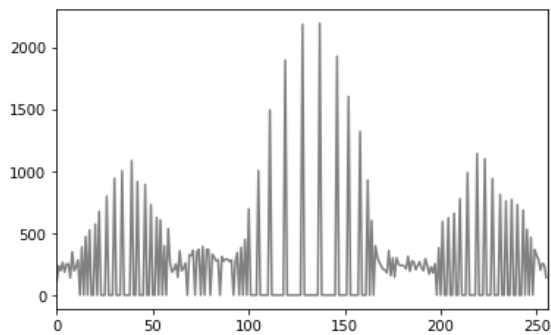


Figure 5.17: Equalized Histogram Plot



Figure 5.18: Histogram Equalized Gray Image

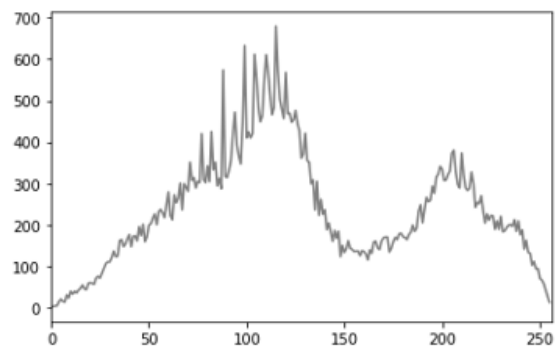


Figure 5.19: Contrast Limited Adaptive Histogram Equalization

Contrast Limited Adaptive Histogram Equalization (CLAHE)

This is the improvement over the Adaptive Histogram Equalization (AHE). AHE divides the image into distinct blocks and computes histogram equalisation for each section. Thus, AHE computes many histograms, each corresponding to a distinct section of the image. CLAHE

improvises AHE by employing bilinear interpolation to remove artificial boundaries caused by merging the different sections/tiles.

Median Filter

One of the crucial steps in image preparation for segmentation is noise removal. Filtering is used to remove noisy image pixels. The noise removal method utilized in this case is non-linear median filtering. An efficient way to reduce noise is to use a median filter, which can do so without obliterating sharp edges [30]. A pixel is replaced with a median filter with the neighborhood median value. The median filtered image for the digital image $f(x, y)$ is derived as,

$$g(x, y) = \text{median}\{f(i, j) \mid (i, j) \in w\} \quad (5.2)$$

Where, w is the neighborhood in the image centered on position (x, y) .

Binarization (Otsu's Method)

The central task of character segmentation is distinguishing the characters (foreground) from the plate (background). The segmented image $g(x, y)$ for the grayscale image $f(x, y)$ is produced using the image binarization procedure described below.

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) \geq T \\ 0 & \text{if } f(x, y) < T \end{cases} \quad (5.3)$$

where, T is the threshold value and it can be obtained using the Otsu's threshold selection technique for grayscale image segmentation [31]. The automatic threshold selection method developed by Otsu for gray level picture binarization is nonparametric and unsupervised. An optimal threshold is selected by the discriminant criteria i.e., by maximizing the interclass variance between white and black pixels [30].

Morphological Operation

Morphological transformations are some simple operations based on the image shape. It often takes place on binary images. It requires two inputs: our original image as the first, and the structural element or kernel as the second, which determines the type of operation. Erosion and dilation are two fundamental morphological operations. Then, its alternative forms, such as Opening, Closing, Gradient, etc., also come into play.

Erosion: The fundamental concept of erosion is similar to soil erosion, except it only removes the boundaries of foreground objects (always try to keep foreground in white). The kernel traverses the picture (as in 2D convolution). A pixel in the original image—whether it is 1 or 0—will only be treated as 1 if every pixel under the kernel is 1, otherwise it is eroded (made to zero).

Thus, depending on the size of the kernel, all pixels close to the boundary are eliminated. So, the foreground object's thickness or size, or just the white region itself, reduces in the image. It can be used to eliminate faint white noises, separate two related items, like the characters in our case, and more.

Dilation: It is directly opposed to erosion. A pixel element in this case is a "1" if at least one pixel under the kernel is "1". Therefore, either the size of the foreground object or the white area in the image grows. Typically, erosion is followed by dilation in situations like noise abatement. Because erosion reduces the size of our item while simultaneously removing white noise. So, we dilate it. They won't return because the noise is gone, but our object area grows. It can also be used to repair disconnected pieces of a character.

Cleaning

The preprocessed plate is not yet ready for segmentation, there might be plate boundary, dirt, dust, and noise left in the plate. These unwanted objects will drastically hamper the performance of segmentation algorithm. We clean them by finding contours all contours and filtering the unwanted objects based on area, aspect ratio, width, height, solidity and extent.

A mask is created for all those objects which pass these tests, and 'bitwise-and' of original image is taken with the mask. The threshold for above mentioned tests is determined based on experiment, and such a value is chosen that performs well for all types of plates.

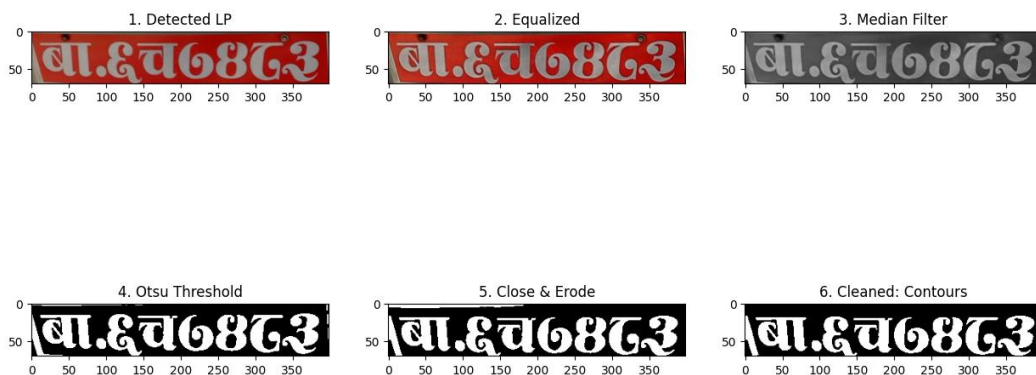


Figure 5.20: Character Segmentation Stage 1

Subsequent tests are heavily dependent on this step; therefore, it must be performed meticulously.

Horizontal Projection Profile

Nepali license plate can have one to three rows. The various types of number plates currently in use can either have a single row or up to three rows. The task of horizontal projection is determining the number of rows in the plate, and separate characters from each row. Horizontal projection analysis to separate double line plates, plates with aspect ratio 4:3, into two separate rows, and segment characters from each row [15]. Analysis of the horizontal projection showed two peaks representing each row and minima representing the boundary between them. In our case, we want to incorporate all kinds of plates being used so horizontal projection profile is used to determine the plate type first, and process each rows separately. The number of peaks can vary from one to three, and minima representing boundary between them.

Horizontal Profile

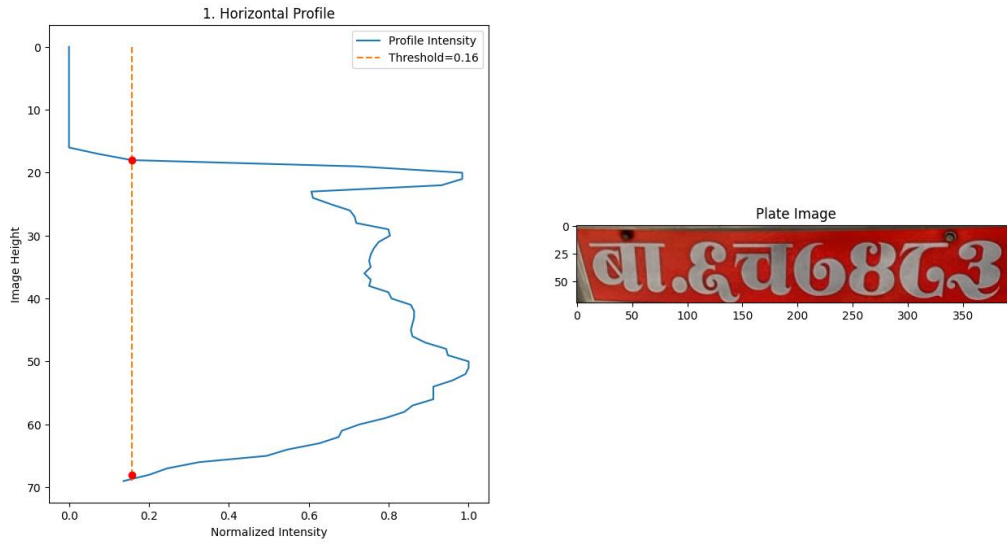


Figure 5.21: Horizontal Profile Projection

Vertical Projection Profile

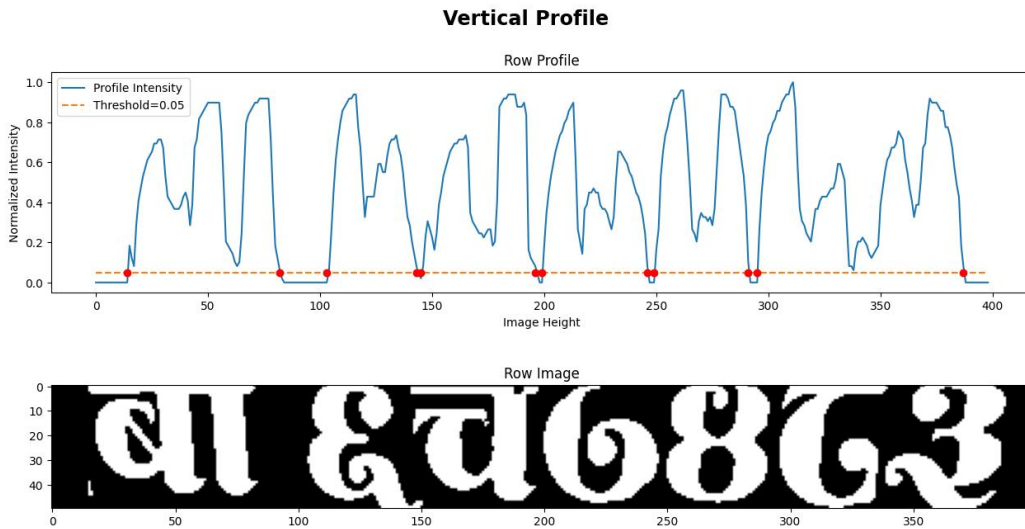


Figure 5.22: Vertical Profile Projection

Vertical projection profile was used to segment the Nepali license plate characters for recognition [15]. The peak of the projection indicated the presence of the characters and the deep valley represented the boundary between the characters. A width threshold was also determined by experiment to ensure valid characters, peaks smaller than the threshold was

considered a part of the previous character. The technique of vertical projection profile to find the gaps between characters in a license plate, zero point in the projection histogram indicated the boundary between two characters [32]. Top and bottom boundaries were removed to ensure the zero point that separated the plate into horizontal segments. Too small segments were linked together as a single character, and too big segments were split into two or more segments based on a preset threshold value. We utilize the exact same concept to segment each row (determined by horizontal projection) of a plate into character segments. The number of maxima peaks depends on the type and row of the plate.

Validation

Binarized license plate after preprocessing went through cleaning, and both horizontal and vertical projection profile analysis. There is a chance that some characters might get missed out while cleaning if it is touching the boundary, if characters are joined together (usually observed in old discolored plates), and two or more characters might be grouped into one when threshold value fails in vertical projection profile. To take care of these problems, we run a series of validation steps to make sure the segmented region is indeed a character.

The first test is character size test. If the integer ratio of segmented character width and the median character width is greater than one, the region is divided into as many segments each of median width. It is then followed by spacing test. It finds the gap between the segmented characters. The gap at the start and end of the plate row is usually large compared to the gap between the characters. The ratio of start and end gap is not greater than 1.5, none of the gaps between character is greater than the median character width. Besides, if a character is less than 1.5 times the previous character, it is considered a part of previous character. Finally, to ensure that each segmented region contains only one character, connect component analysis is performed and masked by area, height, width, and aspect ratio set to a preset threshold. For all the tests, threshold values are determined by experiment.

Segmented Characters

Based on the number of horizontal, and vertical projection peaks, we determine the type of license plate. If the plate passes tests in all rows, characters are segmented from the license plate image at the specified bounding box of each character, and same for recognition.

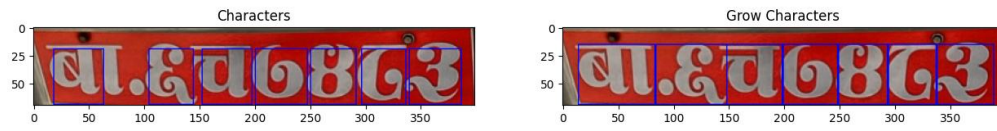


Figure 5.23: Segmented Characters, final stage

5.7 Plate Recognition

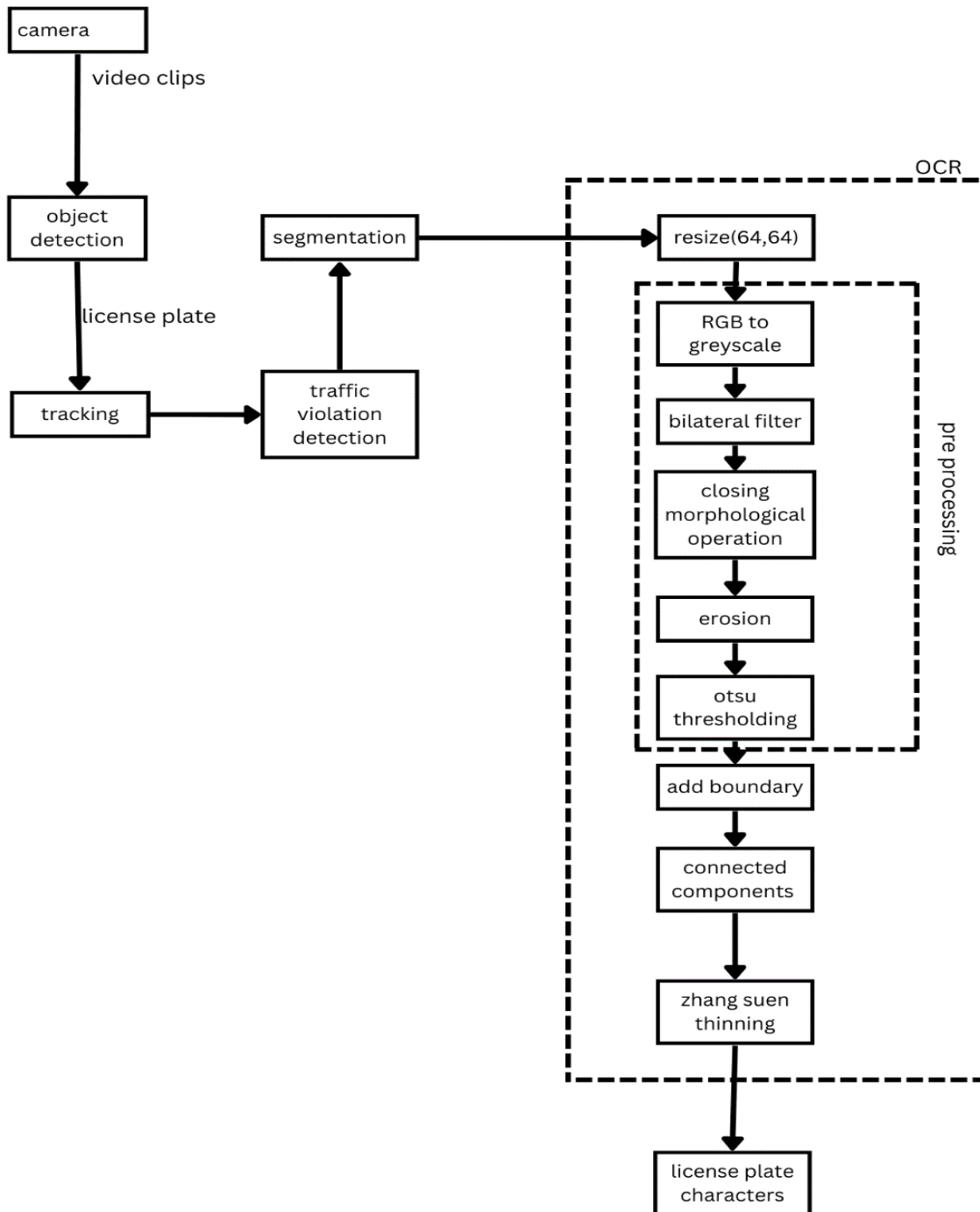


Figure 5.24: Character Recognition Process Overview

5.7.1 Character Pre-processing

The segmented characters are fed to a CNN neural network trained on license plate character dataset. Before, feeding the characters to the trained model, it must go through following steps:

1. Resize character to (64,64)

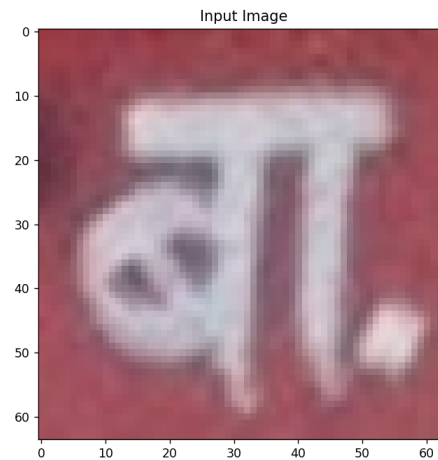


Figure 5.25: Segmented Character

2. Bilateral Filter:

Bilateral Filter is used for smoothening images and reducing noise, while preserving edges. There are other types of filters such as, averaging Filter and median Filter. These convolution results in loss of important edge information, since they blur out everything, irrespective of it being noise or an edge.

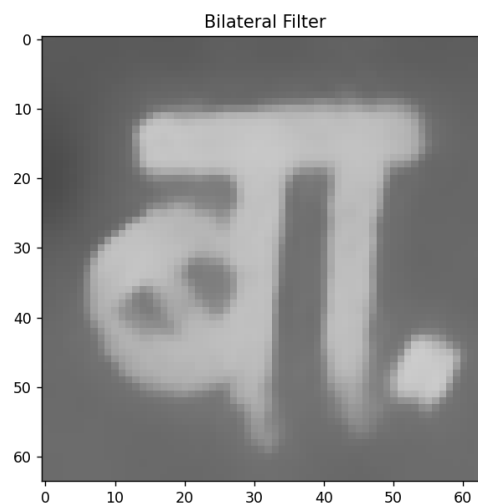


Figure 5.26: Segmented Characters after filtering

3. Closing Morphological Operation:

Morphological Operation provides the insights of shape of image objects present in image and extraction of image features. It is based on set theory. It is useful in binary images. Morphological operators take two pieces of data as input. One is the input image, which can either be binary or grayscale. The other is the Structuring elements. The structuring element is sometimes called the kernel. The structuring element consists of a pattern specified as the coordinates of a number of discrete points relative to some origin.

Closing is the dilation followed by erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.

4. Erosion:

The kernel/structuring elements slides through the image. A pixel in the original image will be considered 1 only if all the pixels under the kernel are 1, otherwise it is eroded (made to zero).

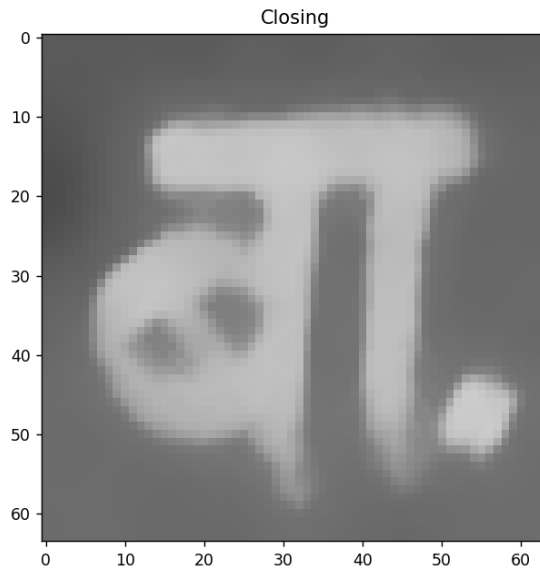


Figure 5.27: Character Closing

Let the pixel coordinate beneath the origin of the structuring element be (x, y) . Let the number of pixels of value 1 in the input image under the kernel be z and the number of pixels in the structuring element be k .

The output of erosion is given by,

$$g(x, y) = \begin{cases} 1 & \text{if } z = k \\ 0 & \text{if } z < k \end{cases} \quad (5.4)$$

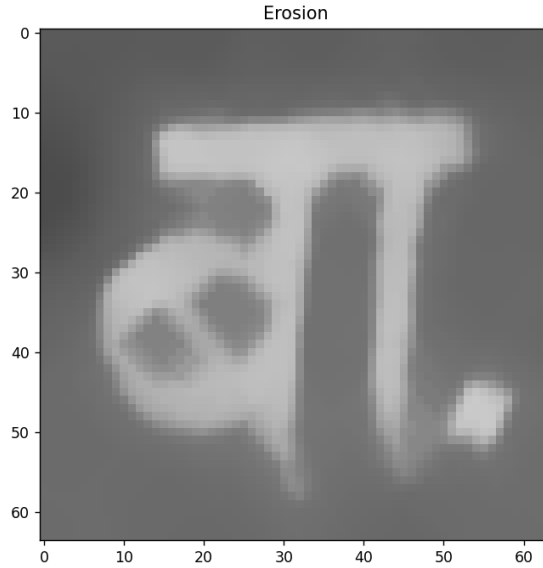


Figure 5.28: Character Erosion

5. Dilation:

It is the opposite of erosion. A pixel is 1 if at least one pixel under the kernel is 1. It increases the white region in the image or size of the foreground object increases.

$$g(x, y) = \begin{cases} 1 & \text{if } z > 0 \\ g(x, y) & \text{if } z < k \end{cases} \quad (5.5)$$

6. Otsu Thresholding:

Otsu thresholding is the global thresholding method which chooses a threshold value and determines it automatically. The steps involved in Otsu thresholding are as follows:

- a. Compute the histogram of the input image.
- b. Normalize the histogram so that it represents the probability distribution of the pixel intensities.
- c. Iterate over all possible threshold values, from 0 to 255, and compute the intra-class variance for each threshold value.
- d. The optimal threshold value is the one that minimizes the intra-class variance.

The formula for finding the within-class variance at any threshold t is given by:

$$\sigma^2(t) = w_{bg}(t)\sigma_{bg}^2(t) + w_{fg}(t)\sigma_{fg}^2(t) \quad (5.6)$$

Where, $w_{bg}(t)$ and $w_{fg}(t)$ represent the probability of the number of pixels for each class at threshold and σ^2 represents the variance of color values.

P_{all} be the total count of pixels in an image,

$P_{bg}(t)$ be the total count of background pixels at threshold t

$P_{fg}(t)$ be the total count of foreground pixels at threshold t

$$w_{bg}(t) = \frac{P_{bg}(t)}{P_{all}} \quad (5.7)$$

$$w_{fg}(t) = \frac{P_{fg}(t)}{P_{all}} \quad (5.8)$$

The variance can be calculated as:

$$\sigma^2(t) = \frac{\sum(x_i - \bar{x})^2}{N - 1} \quad (5.9)$$

for foreground and background

Where x_i is the value of pixel at i in the group (bg or fg)

\bar{x} is the means of pixel values in the group (bg or fg)

N is the number of pixels

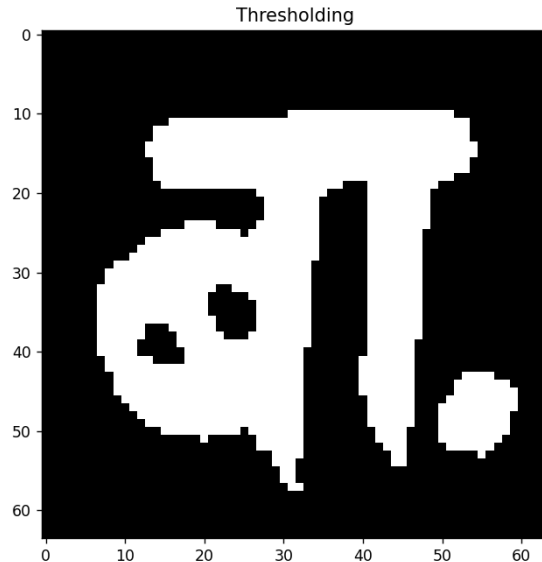


Figure 5.29: Otsu's Thresholding Character

7. Add boundary:

We need to pad the zero pixel in boundary before thinning so that foreground pixels don't touch the image boundary.

8. Connected Components Characters:

Segmented character images from the license plate may not be perfectly segmented.

Two or more characters may be in the same bounding box of segmented image. To eliminate such problems, we need to separate them before prediction.

- a. The connected components are given separate labels.
- b. Different statistics related to each component are computed. Stats are such as, area, connected components height, width, end points etc.
- c. We iterate over each extracted component and filter them based on statistics of corresponding components. E.g., Area of component can be used to filter to consider component as a character or not.
- d. Mask is created to extract the particular character.

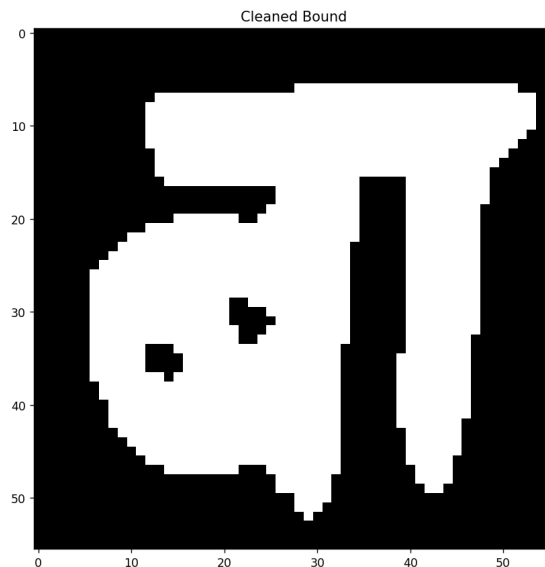


Figure 5.30: Cleaned Character

9. Zhang Suen Thinning:

Skeletonization provides a geometrical description that describes a shape using a smaller number of pixels than the original. Every skeleton is sensitive to noise. Noise removal is performed by a Bilateral filter.

Algorithm: The algorithm operates on all black pixels $P1$ that can have eight neighbours. The boundary of the image cannot have full eight neighbours.

$A(P1)$: the number of transitions from white to black in the circular fashion in the neighbour.

$B(P1)$: the number of black pixel neighbours of $P1$. It is a 2-pass algorithm, for each iteration it performs two sets of checks to remove a pixel.

Step 1 : All pixels are tested and pixels satisfying all the following conditions (simultaneously) are just noted at this stage.

- The pixel is black and has eight neighbours
- $2 \leq B(P1) \leq 6$
- $A(P1) = 1$
- At least one of the north, east, and south neighbours is white.
- At least one of the east, south, and west neighbours is white.

After iterating over the image and collecting all the pixels satisfying all step 1 conditions, all these condition satisfying pixels are set to white.

Step 2 : All pixels are again tested and pixels satisfying all the following conditions are just noted at this stage.

- The pixel is black and has eight neighbours
- $2 \leq B(P1) \leq 6$
- $A(P1) = 1$
- At least one of the north, east, and west neighbours is white.
- At least one of the north, south, and west neighbours is white.

After iterating over the image and collecting all the pixels satisfying all step 2 conditions, all these condition satisfying pixels are again set to white.

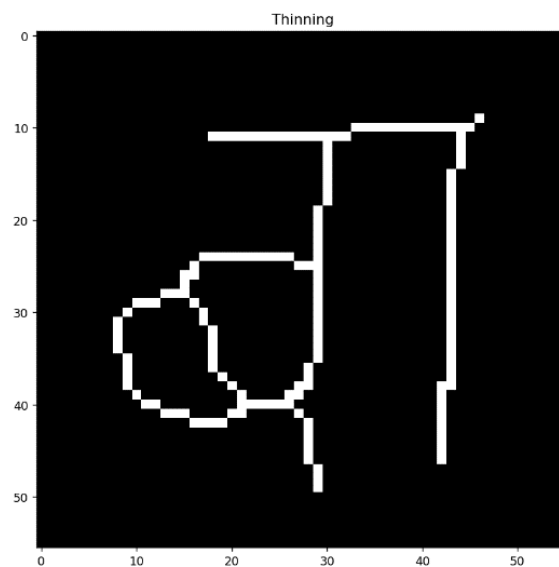


Figure 5.31: Character, Thinned

5.7.2 Character Recognition

Neural Networks

A neural network is a type of machine learning which models itself after the human brain. This creates an artificial neural network that via an algorithm allows the computer to learn by incorporating new data. Image is a 2-dimensional data, containing spatial information of pixel intensity. Before feeding the ANN, we need to extract the features from the input image. Feature extraction is a crucial step in image classification tasks.

Feature Extraction

A convolution operation with the kernel is performed to extract the features. This gives rise to a Convolution Neural Network. Convolution operation is the fundamental building blocks of a Convolution Neural Network. CNNs are state of the art algorithms for computer vision problems like object detection, localization, recognition and so on. Filters/kernels in the CNN layers are used to extract the features from the images. This difficult task of assigning the filters parameters is learned in a supervised manner.

We emphasize that all weights in all layers of a convolutional network are learned through training. Moreover, the network learns to extract its own features automatically.

Convolutional Layer

When designing a CNN, each convolutional layer within neural network should have the following attribute: Input is a tensor with shape (number of images) x (images width) x (images height) x (images depth) can convolutional kernels whose width and height are hyper-parameters, and whose depth must be equal to that of the image. In our case, image width = image height = 64, image depth = 1, number images in single step of feed forward are 32.

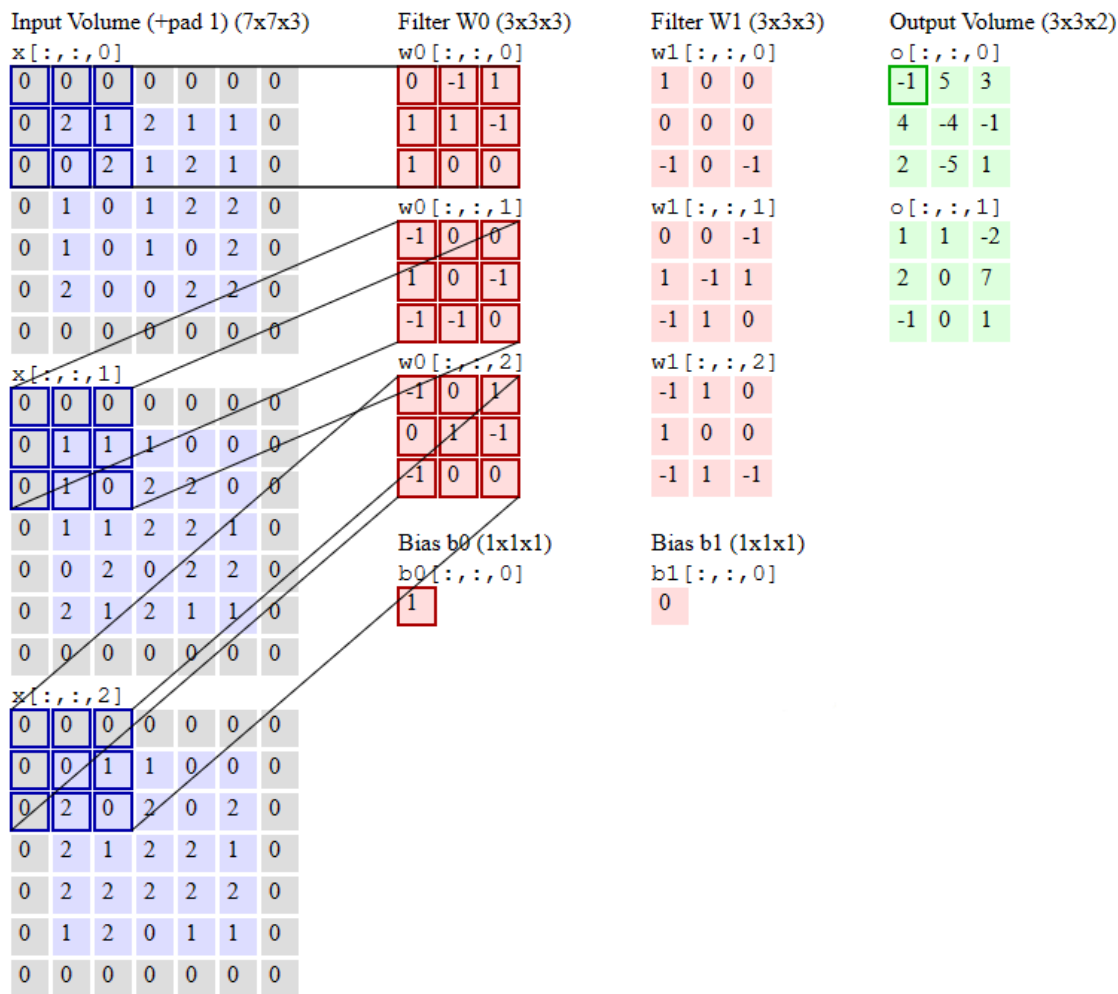


Figure 5.32: Convolution Operation

Padding layer

Every time we apply convolution, image size shrinks. Pixels in the corner are used very less than pixels in the middle for convolution. Pixels holding the shape/structure of a character in the boundary won't contribute much during convolution. So, padding of 0 values is done before feeding to the network. Convolution operation with $(n \times n)$ image with $(f \times f)$ gives $(n - f + 1) \times (n - f + 1)$ output. This shrinkage of output drastically reduces the size of output in the deep layers.

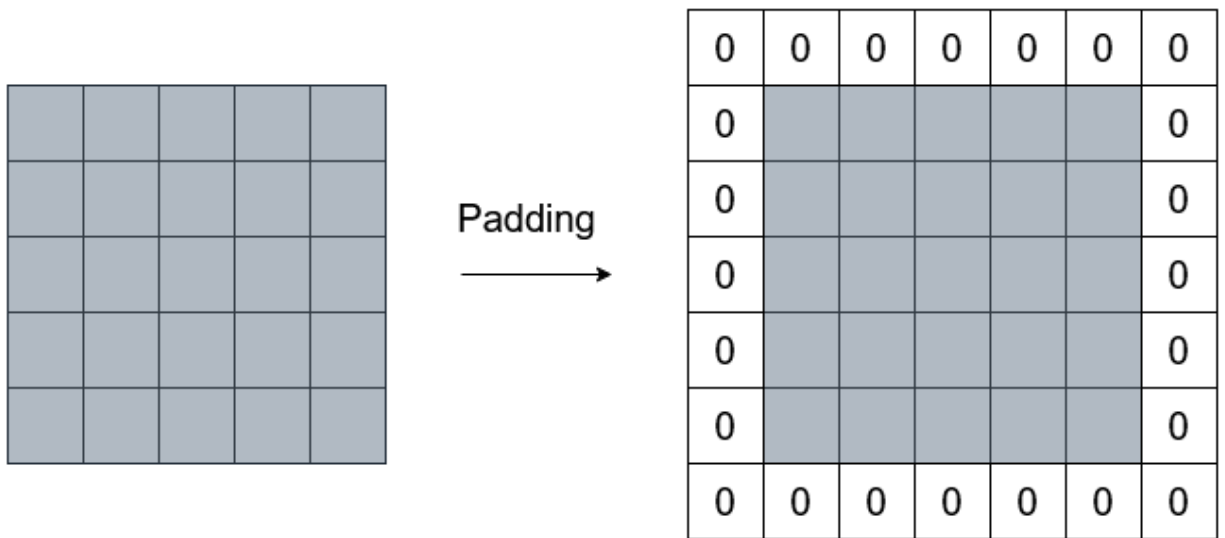


Figure 5.33: Padding

Pooling layer

ConvNet uses pooling to reduce the size of representation to speed the computation. This layer has no parameters to learn. It is based on the intuition that most of the features in the kernel size are held by pixels with the largest value (max pooling).

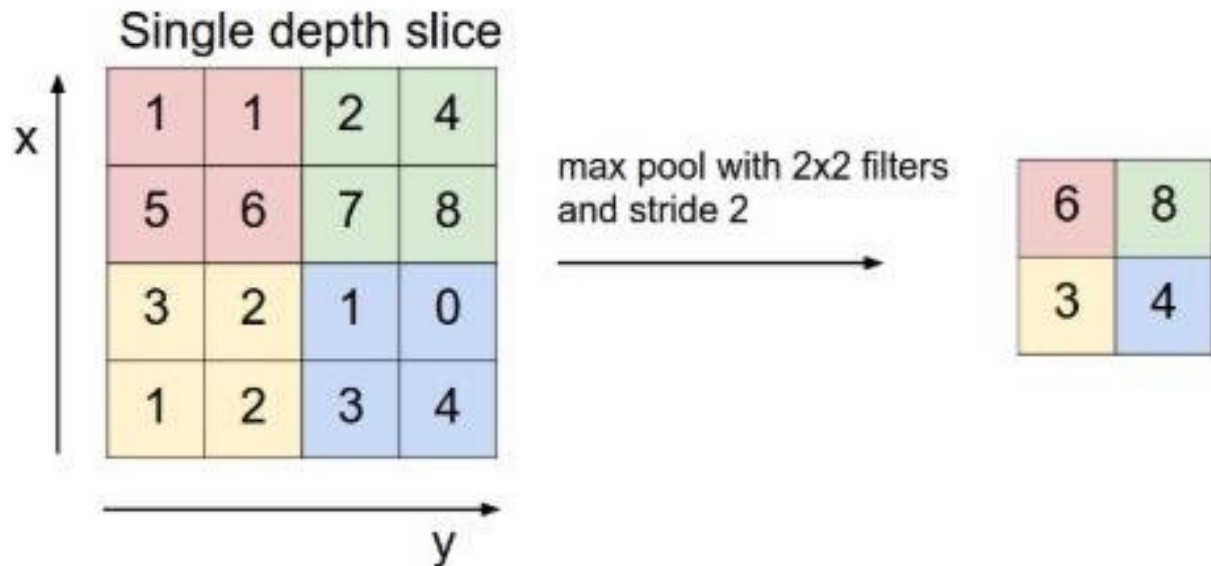


Figure 5.34: Maxpool layer

Activation Function

Activation function allows us to transform the given input into a required output that has a certain range. Activation function introduces non-linearity, so as to make the network learn complex patterns.

ReLU Layer

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The ReLU function has been found to be very good for networks with many layers because it can prevent vanishing gradients when training deep networks and fast computation.

$$f(x) = \max(0, x)$$

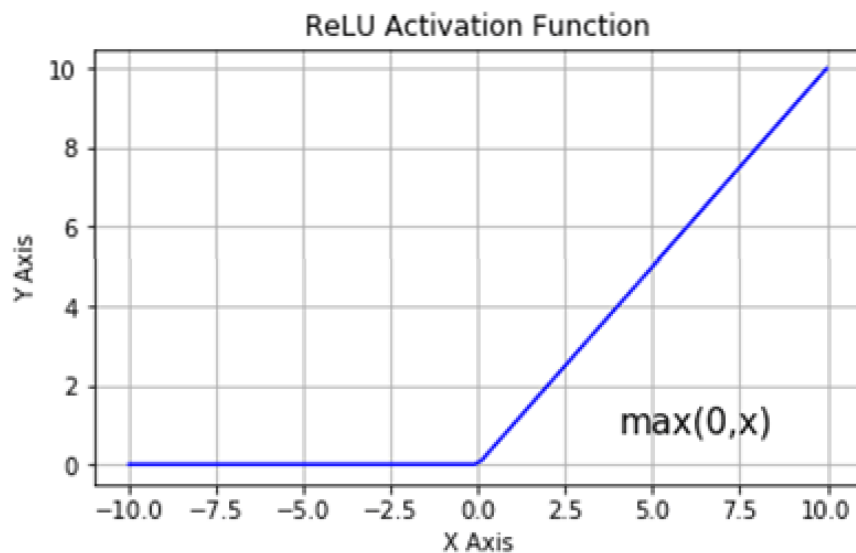


Figure 5.35: ReLU Activation Function

Sigmoid Activation

Sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. The Sigmoid function transforms a real value into one that can be interpreted as a probability, such as between 0 and 1, or -1 and 1.

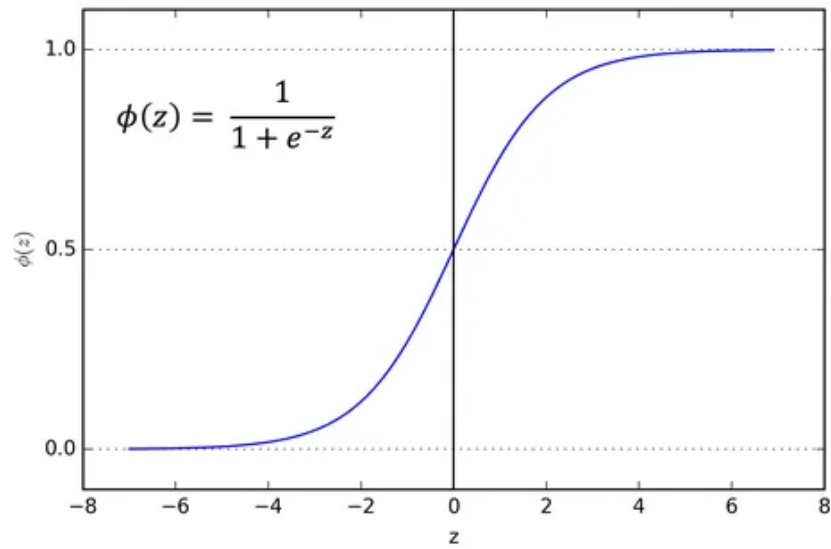


Figure 5.36: Sigmoid Activation Function

Fully Connected Layers

Features extracted from the Convolution Neural Network are flattened to form a 1-dimensional feature set. This 1-dimensional feature is fed to Fully connected layers, with input from earlier convolution layers (filter height x filter width x number of channels).

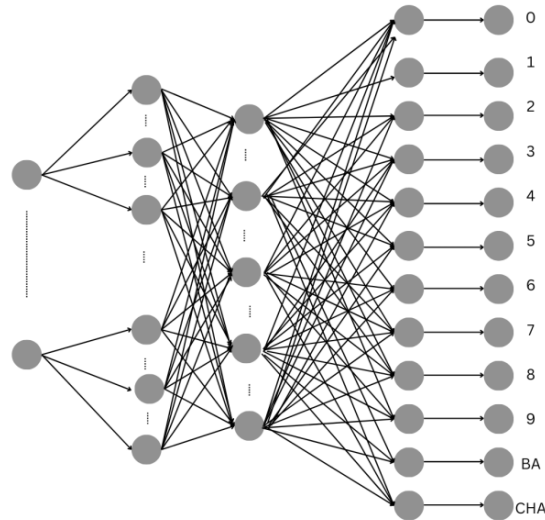


Figure 5.37: Fully Connected Layer

Softmax function

The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. Softmax function in the final layer of the Neural Network, which assigns the probability associated with each class. The probability from different classes sums up to 1.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (5.10)$$

\vec{z} - the input vector to the softmax function, made up of (Z_0, \dots, Z_k)

K - number of classes in the multi-class classifier

Cross-Entropy Loss

Loss functions are used to quantize the difference between the actual and predicted labels. Higher the loss, the worse the model is performing.

$$CE = - \sum_{i=1}^C t_i \log(s_i) \quad (5.11)$$

Where t_i and s_i are the growth truth and CNN score/ output vector of the earlier layer respectively.

Categorical Cross-Entropy Loss

It is a Softmax activation plus a Cross-Entropy loss.

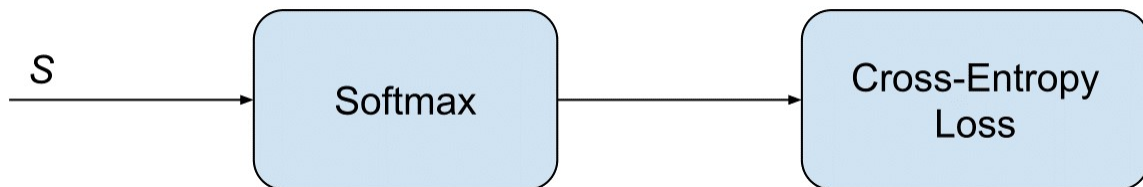


Figure 5.38: Categorical Cross-Entropy Loss

$$f(s)_i = \frac{e^{s_i}}{\sum_{j=1}^C e^{s_j}} \quad (5.12)$$

$$CE = - \log \left(\frac{e^{s_p}}{\sum_j e^{s_j}} \right) \quad (5.13)$$

Where s_p is the CNN score for positive class.

In the specific (and usual) case of multi-Class classification the labels are one-hot, so only the positive class C_p keeps its term in the loss. There is only one element of the Target vector t which is not zero $t_i = t_p$. So, discarding the elements of the summation which are zero due to target labels.

Optimization

Optimization is the iterative process of minimizing the model error. It concerns minimizing the loss. A loss function is often referred to as the objective function for the optimization problem. An optimum solution for the objective function is the global minimum value of the loss function for that model. Optimization algorithms: Gradient Descent, Mini-batch gradient descent, RMS prop, Adams Optimization algorithm.

ADAMs Optimization algorithm

Adaptive Moment estimation is an optimization algorithm formed with a combination of the ‘gradient descent with momentum’ algorithm and the RMS propagation algorithm. In the momentum gradient descent algorithm, the gradient descent algorithm is accelerated by taking the exponentially weighted average of the gradients reducing the oscillation while descending. By taking exponential weight, it makes the algorithm converge toward the minimum quicker. The gradient of complex functions in the neural network may be subjected to either vanish or explode as data propagates through the function. RMS prop deals with the above problem by using the moving average of squared gradients to normalize the gradient. We compute the decaying averages of past and past squared gradients m_t and v_t respectively as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (5.14)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (5.15)$$

Here, m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively. β_1 and β_2 represent the decay rates which are small (i.e., β_1 and β_2 are close to 1), g_t refers to a derivative of the loss function with respect to a derivative of weight at time t (dL/dW_t). For computing bias-corrected first and second moment estimations, Now, instead of our normal weight parameters m_t and v_t , we

take the bias corrected weight parameters \hat{m}_t and \hat{v}_t . Bias correction for the fact that first and second moment estimates start at zero. Now new weight (w_t) is calculated as

$$CE = -\log \left(\frac{e^{s_p}}{\sum_j^C e^{s_j}} \right) \quad (5.16)$$

$$\hat{v} = \frac{v_t}{1 - \beta_2^t} \quad (5.17)$$

$$w_t = w_{t-1} - \eta \frac{\hat{m}}{\sqrt{\hat{v} + \epsilon}} \quad (5.18)$$

Where ϵ is the very small positive number to avoid divide by zero.

Batch Normalization

When the distribution of the input to the layers of the neural network changes as the network is trained, which can slow down the learning process and make it difficult to optimize the model. By normalizing all the features, to take on a similar range of values that can speed up learning. Batch normalization reduces the amount the distribution of the hidden unit values shifts round. Batch norm reduces the problem of the input values changing on the deeper layer so that deeper layer does not have to train on changing input. Batch normalization processes your data in mini batches at a time.

$$\mu = \frac{1}{m} \sum_i z^i \quad (5.19)$$

$$\sigma^2 = \frac{1}{m} \sum_i (z^i - \mu)^2 \quad (5.20)$$

$$z_{norm}^i = \frac{z^i - \mu}{\sqrt{\sigma^2 + \epsilon}} \quad (5.21)$$

Where, z is the mini-batches of the training input features

m is the number of trainings set in that training batch

ϵ is the small positive value to avoid divide by zero

μ is the mean of the mini-batches

σ^2 is the variance within mini-batches

L2 regularization

L2 regularization is a technique used in machine learning to prevent overfitting of a model by adding a penalty term to the loss function. This penalty term is the sum of the squared values

of the model's weights, multiplied by a regularization parameter lambda (λ). By reducing the weights' magnitude, this penalty term forces the model to rely more on the input features and less on each individual weight. By lessening the impact of minor changes in the training data, this can enhance the model's generalization performance. When working with high-dimensional data, where the number of features is significantly more than the number of observations, L2 regularization, also known as Ridge regression, is especially helpful.

$$L2 \text{ regularization} = \lambda \|w\|^2 \quad (5.22)$$

$\|w\|^2$ is the square of the L2 norm of the weights. The L2 norm of weight vector is given by:

$$\|w\|^2 = w_1^2 + w_2^2 + \dots + w_n^2 \quad (5.23)$$

$w_1, w_2 \dots w_n$ are the weights in the weight vector. The L2 regularization term is added to the loss function in each layer.

Model

The architecture is inspired from the Lenet-5 architecture for handwritten and machine-printed character recognition in 1990's consisting of 5 different layers.

Model architecture consists of 3 sets of convolutional and max pooling layers, followed by a flattening convolutional layer, then three fully-connected layers and finally a SoftMax classifier. Convolution layer employs ReLu as the activation function, while dense layer (fully connected layers) employs sigmoid activation function. Adams optimization algorithm with the learning rate of 1e-4 is used.

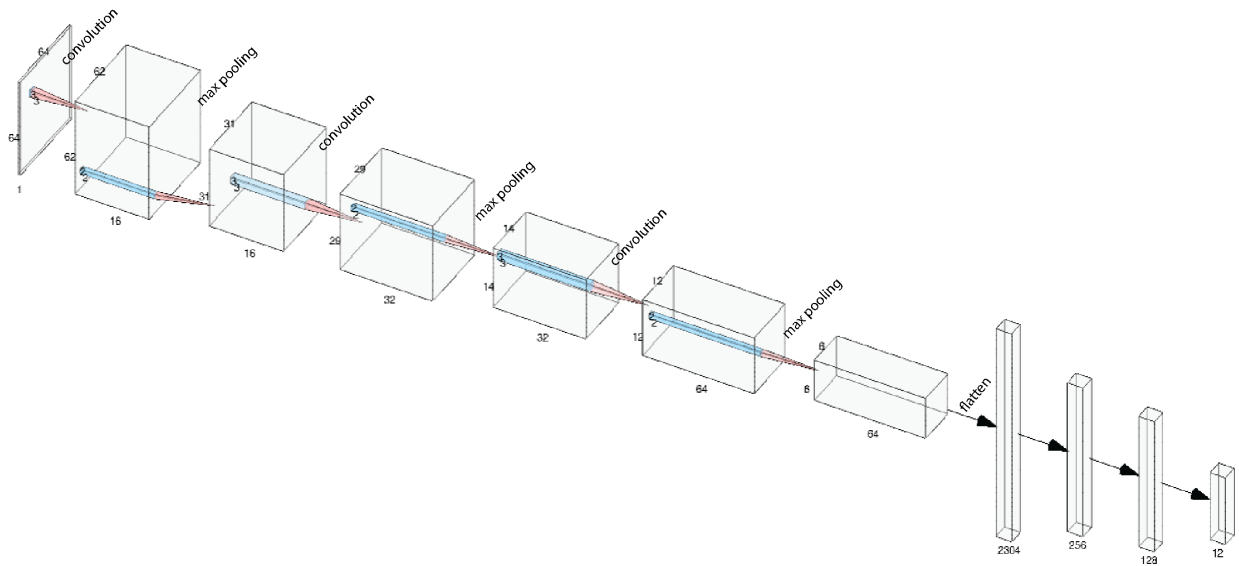


Figure 5.39: Model Architecture

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 16)	160
max_pooling2d (MaxPooling2D)	(None, 31, 31, 16)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 256)	590080
dense_1 (Dense)	(None, 128)	32896
dense_2 (Dense)	(None, 12)	1548

=====
Total params: 647,820
Trainable params: 647,820
Non-trainable params: 0
=====

Figure 5.40: Model Summary

5.7.3 Multi-class classification model evaluation metrics

Training loss and training accuracy

Training loss and accuracy are used to evaluate the performance of a machine learning model during the training phase.

Training loss is a metric that defines how well the model is able to fit the training data. It can be computed as the difference between the predicted and actual values for each training example and then averaged across all the training examples. The main goal is to minimize the training loss thereby meaning that the model is able to make more accurate predictions on the training data. The training loss is computed after every batch or epoch of training, and its value decreases over time as the model learns to make better predictions.

On the other hand, training accuracy is a metric that is used to measure the percentage of instances in the training set that were correctly classified. It can be calculated by dividing the number of correct predictions in the training set by the total number of instances. A high training accuracy indicates that the model is performing well on the training data set. But it doesn't necessarily ensure good performance on new, unseen data, as the model may have overfit to the training data.

Validation loss and validation accuracy

Validation loss and accuracy are used to evaluate the generalization performance of a machine learning model during the training phase. Validation loss is a metric that defines how well the model is able to generalize to new and unseen data. It can be calculated as the average difference between predicted values and true values over a validation set. The main goal is to minimize the validation loss thereby meaning that the model is able to make more accurate predictions on new data. The validation loss is computed after each epoch of training. On the other hand, validation accuracy measures the percentage of instances in the validation set that were classified correctly. It can be calculated by dividing the number of correct predictions by the total number of instances in the validation set. A high validation accuracy indicates that the model is able to generalize well to new and unseen data sets.

When validation loss is largely greater than the training loss, it may indicate that the model is underfitting. It occurs when the model can't accurately model the training data and produces large errors. In such cases, additional training is required to minimize the incurred loss during training. The training data can also be increased either by obtaining more

samples or augmenting the data.

If the training loss and accuracy continue to improve while the validation loss and accuracy start to degrade, it may indicate overfitting. In this case, adjustments need to be made to the model to improve its generalization performance such as reducing its complexity or adding regularization techniques. The validation set can also be used to compare different models and select the best one based on its validation performance.

When the training loss and validation loss both decrease and stabilize at a specific point, it indicates an optimal fit i.e., the model does not overfit or underfit.

Model:

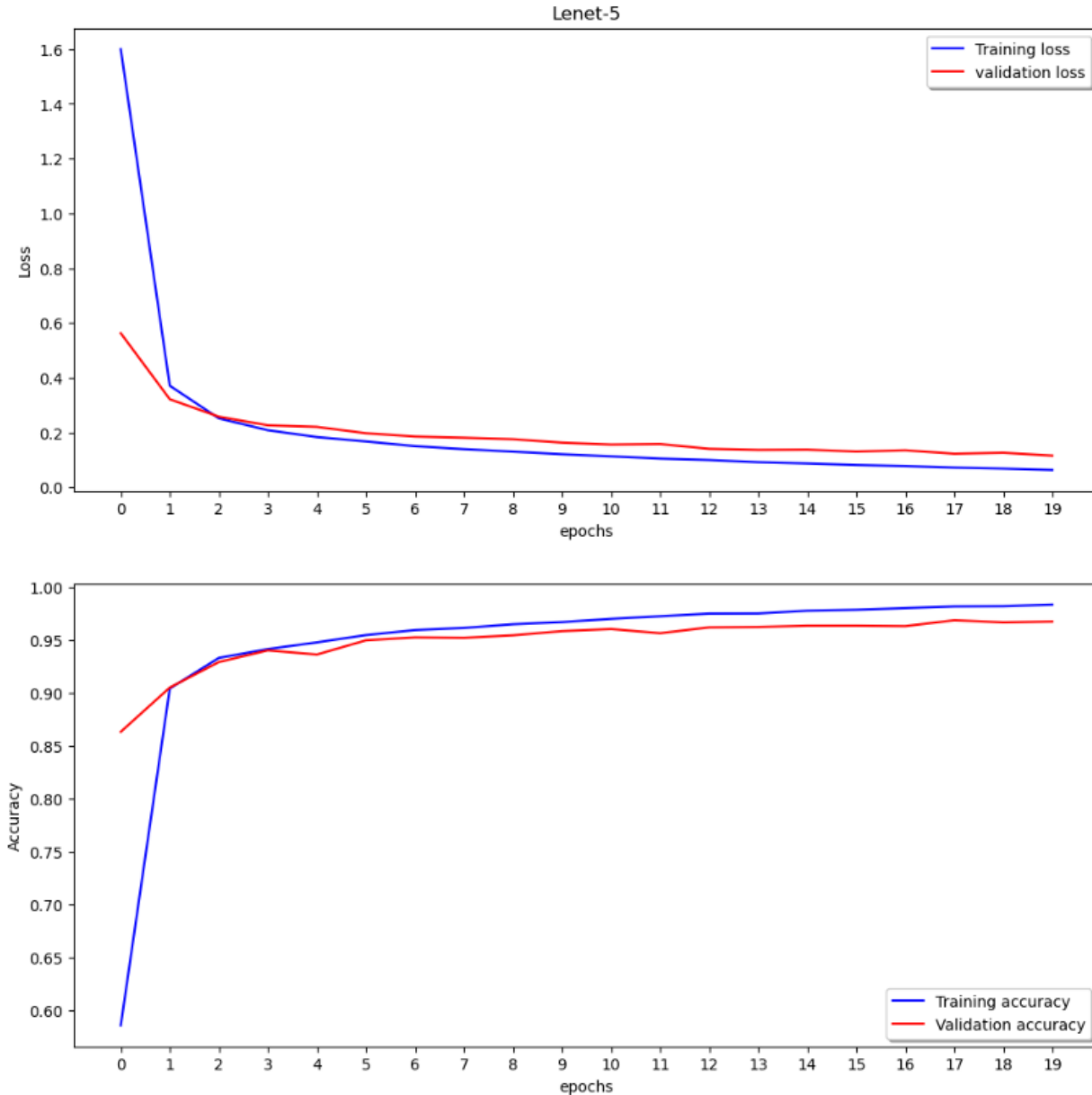


Figure 5.41: Train/Validation Model Error and Accuracy

Confusion Matrix

A confusion matrix is a table used to visualize and summarize the performance of a classification algorithm. It creates a table of all the predicted and actual values of a classifier. To create a 2 x 2 Confusion matrix, we can get four different combinations from the predicted and actual values of a classifier as:

1. True Positive: It represents the number of times our actual positive values are equal to the predicted positive i.e., we predict a positive value correctly.
2. False Positive: It represents the number of times our model predicts positive value but the actual value is negative.
3. True Negative: It represents the number of times our actual negative values are equal to predicted negative values i.e., we predicted a negative value and it is actually negative.
4. False Negative: It represents the number of times our model predicts a negative value but the actual value is positive.

Model

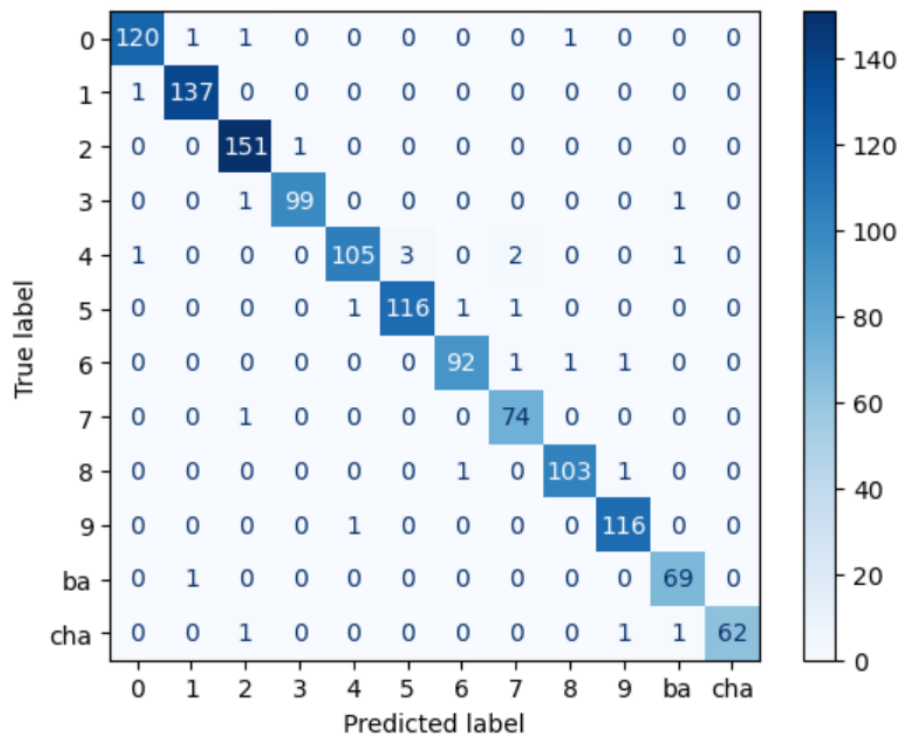


Figure 5.42: Model Confusion Matrix for Test Set

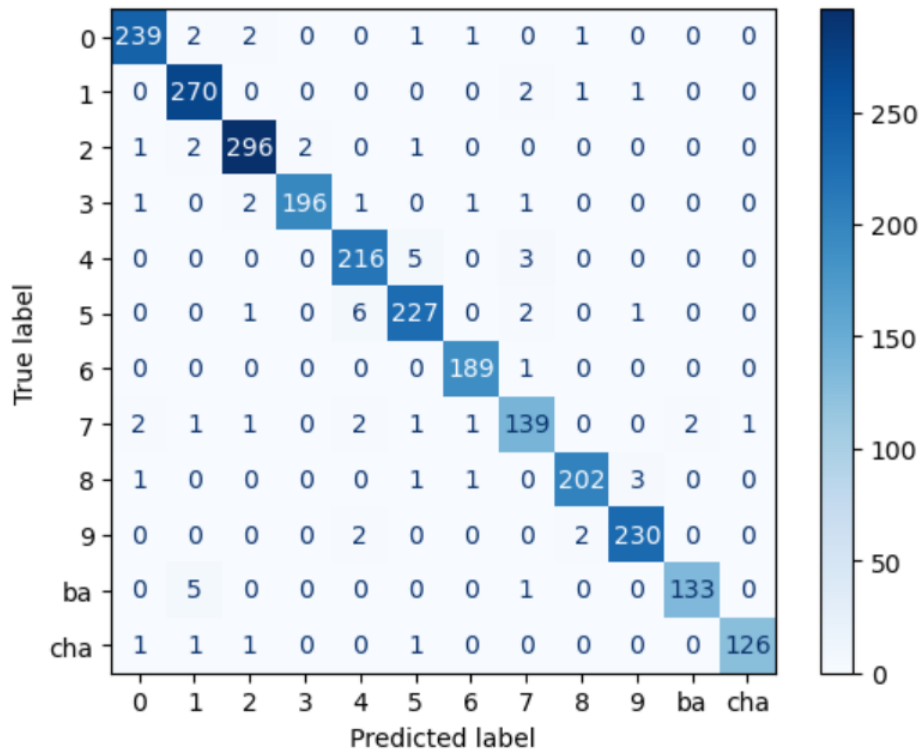


Figure 5.43: Model Confusion Matrix for Validation Set

Precision

Precision is a performance metric that is used to calculate the model’s ability to classify positive values correctly. In other words, it measures the proportion of true positives among the total number of positive predictions (i.e., sum of true positives and false positives). Its formula is given as follows:

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

A high precision score denotes that the model is making accurate positive predictions with a low rate of false positives. But it doesn’t necessarily mean that the model is accurately identifying all positive cases.

Recall(Sensitivity)

Recall is a performance metric that is used to calculate the model’s ability to predict positive values. In other words, it measures the proportion of true positives among the total number of actual positive cases (i.e., the sum of true positives and false negatives). Its formula is given as follows:

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

A high recall score denotes that the model is identifying a high proportion of positive cases with a low rate of false negatives. But it doesn't necessarily mean that the model is making accurate positive predictions.

F1-Score

It is difficult to compare two models that have low precision and high recall or vice versa. In such scenarios, we use F1-Score. F1-Score is a commonly used performance metric that takes both precision and recall into account. It is the harmonic mean of precision and recall which provides a single score which balances both measures. Its formula is given as follows:

$$\text{F1 score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

Its score ranges between 0 and 1, where 1 indicates perfect precision and recall whereas 0 indicates poor performance.

Classification Report for Model

Classification Report				
	precision	recall	f1-score	support
0	0.98	0.98	0.98	123
1	0.99	0.99	0.99	138
2	0.97	0.99	0.98	152
3	0.99	0.98	0.99	101
4	0.98	0.94	0.96	112
5	0.97	0.97	0.97	119
6	0.98	0.97	0.97	95
7	0.95	0.99	0.97	75
8	0.98	0.98	0.98	105
9	0.97	0.99	0.98	117
ba	0.96	0.99	0.97	70
cha	1.00	0.95	0.98	65
accuracy			0.98	1272
macro avg	0.98	0.98	0.98	1272
weighted avg	0.98	0.98	0.98	1272

Figure 5.44: Test Classification Report for Model

Classification Report				
	precision	recall	f1-score	support
0	0.98	0.97	0.97	246
1	0.96	0.99	0.97	274
2	0.98	0.98	0.98	302
3	0.99	0.97	0.98	202
4	0.95	0.96	0.96	224
5	0.96	0.96	0.96	237
6	0.98	0.99	0.99	190
7	0.93	0.93	0.93	150
8	0.98	0.97	0.98	208
9	0.98	0.98	0.98	234
ba	0.99	0.96	0.97	139
cha	0.99	0.97	0.98	130
accuracy			0.97	2536
macro avg	0.97	0.97	0.97	2536
weighted avg	0.97	0.97	0.97	2536

Figure 5.45: Validation Classification Report for Model

6. Conclusions

The project has successfully shown the possibilities of employing the YOLOv5s model for object identification, together with DeepSORT for tracking objects and afterwards checking for traffic signal violations. Also, the license plate segmentation and OCR program offered a useful method of extracting and reading the characters from the license plate. The project demonstrated the use of deep learning models in practical applications the project can be further enhanced for use in traffic surveillance systems.

7. Limitations and Future enhancement

Limitations:

1. If the object detection fails, the entire process of vehicle tracking and violation detection will fail. Hence, object detection is the key, and the most crucial part of the project.
2. The custom dataset doesn't have all classes represented equally, hence mAP is only 0.85 which can surely be improved with a balanced dataset.
3. If the object ID is switched due to occlusion, light violation fails.
4. Provincial number plates have many variations, hence currently the system doesn't recognize the province name and transport management office where the vehicle is registered.
5. If the license plates are very old, color washed out or unclear characters, character recognition won't perform well.
6. Embossed plates can be detected, but characters are not recognized.

Enhancement:

Object detection is the first crucial step in the project, a balanced dataset with all classes represented equally is definitely one way to improve the model. Adding new data, and having ample variation in the dataset representing real deployment environment, will help to improve model's performance. As far as recognition of license plate is concerned, only private vehicles of Bagmati province are considered hence, it can be extended to other vehicle types, and even other provinces. Due to existing variation in license plates, provincial plates are a real challenge. The project can be improved to deal with all variations of license plates, and even embossed plates written in English. To improve character recognition, detected license plate's image can be enhanced such as detecting motion blur, and taking the best image from 5-10 frames. With these enhancements, the project will be able to deal with all variations of number plates currently being used in Nepal.

References

- [1] W. Team, *Global status report on road safety 2018*, <https://www.who.int/publications/i/item/9789241565684>, 2018.
- [2] W. H. Rankings, *Road accidents in nepal*, <https://www.worldlifeexpectancy.com/nepal-road-traffic-accidents>, 2020.
- [3] *Nepal accidental description*, <https://www.traffic.nepalpolice.gov.np/index.php/news/traffic-activities/425-annually-accidental-descriptions>.
- [4] *Nepal's other pandemic : Road fatalitites*, <https://www.nepalitimes.com/multimedia/nepal-s-other-pandemic-road-fatalities>, 2021.
- [5] S. Phuyal, *Road kill, kathmandu: Nepali times*, <https://archive.nepalitimes.com/article/nation/traffic-accidents-continue-to-increase-worryingly-in-Nepal>, 2799.
- [6] S. Dhungana, *Authorities are installing more cctv cameras to increase surveillance in the capital city*, <https://kathmandupost.com/valley/2019/10/19/authorities-are-installing-more-cctv-cameras-to-increase-surveillance-in-the-capital-city>, 2019.
- [7] G. Ou, Y. Gao, and Y. Liu, “Real-time vehicular traffic violation detection in traffic monitoring stream,” in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, IEEE, vol. 3, 2012, pp. 15–19.
- [8] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, “A real-time computer vision system for measuring traffic parameters,” in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, IEEE, 1997, pp. 495–501.
- [9] X. Wang, L.-M. Meng, B. Zhang, J. Lu, and K.-L. Du, “A video-based traffic violation detection system,” in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, IEEE, 2013, pp. 1191–1194.
- [10] H. Zhu, C. Xu, and F. Li, “The traffic volume count algorithm based on computer vision,” in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, IEEE, vol. 1, 2013, pp. 130–133.

- [11] L. Y. Deng, N. C. Tang, D.-l. Lee, C. T. Wang, and M. C. Lu, "Vision based adaptive traffic signal control system development," in *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, IEEE, vol. 2, 2005, pp. 385–388.
- [12] X. Wen-juan and L. Jian-feng, "Application of vision sensing technology in urban intelligent traffic control system," in *2018 4th International Conference on Computer and Technology Applications (ICCTA)*, IEEE, 2018, pp. 74–77.
- [13] J.-a. Kim, J.-Y. Sung, and S.-h. Park, "Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition," in *2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia)*, IEEE, 2020, pp. 1–4.
- [14] F. H. Shubho, F. Iftexhar, E. Hossain, and S. Siddique, "Real-time traffic monitoring and traffic offense detection using yolov4 and opencv dnn," in *TENCON 2021-2021 IEEE Region 10 Conference (TENCON)*, IEEE, 2021, pp. 46–51.
- [15] A. K. Pant, P. K. Gyawali, and S. Acharya, "Automatic nepali number plate recognition with support vector machines," in *Proceedings of the 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, 2015, pp. 92–99.
- [16] J. Bhalerao, A. Kadam, A. Shinde, V. Mugalihar, and H. Bhan, "Proposed design on driver behavioral analysis," *International Journal of Engineering Research and Technology*, vol. 9, no. 5, pp. 554–557, 2020.
- [17] R. Ghandour, A. J. Potams, I. Boulkaibet, B. Neji, and Z. Al Barakeh, "Driver behavior classification system analysis using machine learning methods," *Applied Sciences*, vol. 11, no. 22, p. 10 562, 2021.
- [18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [19] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
- [20] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
- [21] C. Imane, *Yolov5 architecture explained*, <https://iq.opengenus.org/yolov5/>.
- [22] G. Jocher, A. Stoken, J. Borovec, *et al.*, "Ultralytics/yolov5: V5. 0-yolov5-p6 1280 models aws supervise. ly and youtube integrations," *Zenodo*, vol. 11, 2021.

- [23] P. Sanap and S. Narote, “License plate recognition system-survey,” in *AIP Conference Proceedings*, American Institute of Physics, vol. 1324, 2010, pp. 255–260.
- [24] V. Karthikeyan and V. Vijayalakshmi, “Localization of license plate using morphological operations,” *arXiv preprint arXiv:1402.5623*, 2014.
- [25] *Vehicle transport management rule, 2054 (1997)*, 1997.
- [26] *Motor vehicles and transport management act, 2049 (1993)*, <https://www.dotm.gov.np/MainData/ActRegulations>, 1993.
- [27] A. Rosebrock, *Point opencv getperspective transform example-pyimagesearch*, 2014, 4.
- [28] Y. Zhang and C. Zhang, “A new algorithm for character segmentation of license plate,” in *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No. 03TH8683)*, IEEE, 2003, pp. 106–109.
- [29] F. Yang, Z. Ma, and M. Xie, “A novel approach for license plate character segmentation,” in *2006 1st IEEE Conference on Industrial Electronics and Applications*, IEEE, 2006, pp. 1–6.
- [30] A. K. Pant, S. P. Panday, and S. R. Joshi, “Off-line nepali handwritten character recognition using multilayer perceptron and radial basis function neural networks,” in *2012 Third Asian Himalayas International Conference on Internet*, IEEE, 2012, pp. 1–5.
- [31] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [32] X. He, L. Zheng, Q. Wu, W. Jia, B. Samali, and M. Palaniswami, “Segmentation of characters on car license plates,” in *2008 IEEE 10th Workshop on Multimedia Signal Processing*, IEEE, 2008, pp. 399–402.