

Automatic Motor Vehicle Number Plate Recognition

by

Krishno Saha
19101271

Parvez Ishrak
19101266

Jahid Hossian Shovon
22101911

Alinur Rahman Abir
19101055

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

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2024

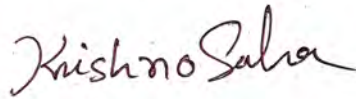
© 2024. Brac University
All rights reserved.

Declaration

It is hereby declared that

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

Student's Full Name & Signature:



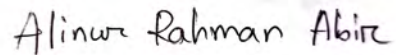
Krishno Saha
19101271



Parvez Ishrak
19101266



Jahid Hossian Shovon
22101911



Alinur Rahman Abir
19101055

Approval

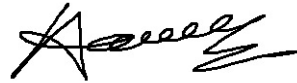
The project titled “Automatic motor vehicle number plate recognition” submitted by

1. Krishno Saha (19101271)
2. Parvez Ishrak (19101266)
3. Jahid Hossian Shovon (22101911)
4. Alinur Rahman Abir (19101055)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 18, 2024.

Examining Committee:

Supervisor:
(Member)



Amitabha Chakrabarty, PhD

Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The purpose of this initiative is to develop automatic motor vehicle number plate recognition (Bangla) using machine learning, identifying and taking out the numbers of license plates from photos. By using this system we intend to help the traffic control system in detecting any issue within a few moments. Moreover, collecting tolls and enforcement of law can be implemented with this number plate recognition system. Various object detection models have been used in this in various suggested methods to identify and recognize number plates, optical character recognition and license plate detection make up the system's three basic building blocks. YOLOv8, YOLOv7, YOLOv5, VGG16, RESNET50, DETR, VGG16 are the models used in this project. Object detection models are used to detect the number plate of a vehicle from the images. That is how the method will be able to successfully recognize and detect the number plate. The precision, recall and mAP value of YOLOv8 is 96.4%, 84.8%, 92.9% respectively. For YOLOv7 it is 61.1%, 46%, 46.5% respectively. For YOLOv5 it is 98.1%, 12.1%, 17.4% respectively. DETR is 6.5%, 7.5%, 8.32% respectively. For VGG16 the test accuracy is 90.14% and for ResNet50 it is 89.91%. Additionally, this system will be implemented within the web. So by using a phone camera the car number plates would be detected with a device like a mobile phone. To sum up, the number plate detection system has the ability to detect, identify and be able to save the information and will help provide a reliable management system for traffic and capturing fraud and indiscipline in the traffic control system.

Keywords: NPR, Vehicle Number Plate, Automated Number Plate Detection, Segmentation, feature extraction, feature selection, YOLOv8, YOLOv7, YOLOv5, DETR, VGG16, RESNET50.

Acknowledgement

Firstly, we want to thank Allah for guiding and blessing us throughout our academic journey. We're grateful to our supervisor, Dr. Amitabha Chakrabarty, for his help and advice. Special appreciation goes to our research assistant, MD. Fahim-Ul-Islam, for his hard work and suggestions that made this project possible. Lastly, we express our heartfelt gratitude to our parents for their endless love, support, and efforts, which have been the basis of our academic journey.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iv
Abstract	iv
Dedication	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
Nomenclature	viii
1 Introduction	1
1.1 Thoughts behind the Prediction Model	1
1.2 Problem Statement	3
1.3 Project Objective	4
2 Related Work	5
2.1 Deep Learning Approaches	5
2.2 Tools Used In the Papers	7
2.3 Comparative Analysis	12
3 Dataset	14
3.1 Dataset Formation	14
3.2 Dataset Construction	14
3.2.1 90 degrees rotation augmentation	15
3.2.2 Shear Augmentation	15
3.2.3 Grayscale augmentation	16
3.2.4 Brightness augmentation	17
3.2.5 Blur augmentation	17
3.2.6 Noise augmentation	18
3.3 Dataset Splitting and Forming Dataset	19

4	Methodology	21
4.1	Work plan	21
4.2	Model Description	21
4.2.1	YOLOv8	21
4.2.2	YOLOv8 Architecture	21
4.2.3	YOLOv7	23
4.2.4	YOLOv7 Architecture	23
4.2.5	YOLOv5	24
4.2.6	YOLOv5 Architecture	25
4.2.7	DETR	26
4.2.8	DETR Architecture	26
4.2.9	VGG16	27
4.2.10	VGG16 Architecture	28
4.2.11	Resnet50	29
4.2.12	ResNet50 Architecture	29
4.2.13	OCR	30
4.2.14	OCR Architecture	30
4.3	Web App Workflow	31
5	Implementation and results	32
5.1	Implementation	32
5.2	YOLOv8 model results	33
5.2.1	Graph	33
5.2.2	Confusion Matrix	33
5.3	YOLOv7 model results	34
5.3.1	Graph	34
5.3.2	Confusion Matrix	35
5.4	YOLOv5 model results	35
5.4.1	Graph	35
5.4.2	Confusion Matrix	36
5.5	VGG16	36
5.5.1	VGG16 Loss and Accuracy Graph	36
5.5.2	Confusion Matrix	37
5.6	ResNet50	38
5.6.1	ResNet50 Loss and Accuracy Graph	38
5.6.2	Confusion Matrix	38
5.7	Detection samples	39
5.8	Overall result analysis	39
5.9	License plate recognition (Web App Functionality)	40
6	Future Work and Conclusion	42
6.1	Future Work	42
6.2	Conclusion	42
	Bibliography	46

List of Figures

3.1	90 degrees rotation augmentation	15
3.2	Shear Augmentation	16
3.3	Grayscale augmentation	16
3.4	Brightness augmentation	17
3.5	Blur augmentation	18
3.6	Noise augmentation	19
3.7	Example of Annotations	20
4.1	YOLOv8 model architecture	22
4.2	YOLOv7 model architecture	24
4.3	YOLOv5 model architecture	26
4.4	DETR model architecture	27
4.5	VGG16 model architecture	28
4.6	Resnet50 model architecture	29
4.7	OCR model architecture	30
4.8	Web App Workflow	31
5.1	Yolov8 Graph (A \rightarrow class loss and B \rightarrow mAP50)	33
5.2	Yolov8 Confusion Matrix	34
5.3	Yolov7 Graph (A \rightarrow class loss and B \rightarrow mAP50)	34
5.4	Yolov7 Confusion Matrix	35
5.5	Yolov5 Graph (A \rightarrow class loss and B \rightarrow mAP50)	36
5.6	Yolov5 Confusion Matrix	36
5.7	VGG16 Loss and Accuracy Graph	37
5.8	VGG16 Confusion Matrix	37
5.9	ResNet50 Loss and Accuracy graph	38
5.10	ResNet50 Confusion Matrix	38
5.11	Detection samples	39
5.12	License plate recognition (Web APP)	41

Chapter 1

Introduction

1.1 Thoughts behind the Prediction Model

Around the world, there are a lot of automobiles in this generation. That's why keeping track of automobiles is crucial. In the modern era, we may use computers to track any car instead of manually tracking automobiles, which will result in more accuracy. Therefore, a vehicle number plate recognition system uses technology to recognize the number plate from footage the camera has acquired. It employs techniques including character recognition, segmentation, and number plate extraction [10]. This method uses the license plate and then sends it to be converted to a picture using a combination of hardware and software. Any gate entrance can employ this technology. Thus, if the image of the number plate retrieved from the camera-captured image is clear and visible, this technique provides correct results. The image that is utilized must have excellent resolution [6]. The Bangladesh Road Transport Authority (BRTA) issues license plates in the general style "city - vehicle class letter and number - Vehicle number." Only private automobiles were used in this study; the Dhaka and Chattagram metropolitan area's license plate areas are taken into account. The private vehicles' license plates belong to the following ten classes only: ka, kha, ga, gha, ca, cha, bo, vho, lo and ho [6]. Traffic control and vehicle owner identification become major problems in Bangladesh, so that's why we are working on this project to maximize the benefits and mitigate the problems on number plate recognition issues. Our lives now include automatic number plate recognition, and this trend seems to continue in the future thanks to its compatibility with emerging transportation technologies. The idea of autonomous vehicles opens up a lot of potential changes to the foundational transportation infrastructure. ANPR technology is already removing the need for human interaction and advancing intelligent transportation systems. It's no longer only the parking lot barrier or the camera by the side of the road. Over the years, it has evolved into a mobile system, originally being used in automobiles, but more recently, with the introduction of smartphone technology, many ANPR devices have also become handheld. Rapid urbanization is a remarkable development in our contemporary environment. People leave rural areas and primarily choose to live in cities. As traffic in these places increases, local governments sometimes fail to comprehend the mobility demands of residents and visitors, both now and in the future. A growing amount of traffic flow analysis is being done using ANPR to support intelligent transportation [39]. YOLOv8 is an object detection model which has been used in paper [1] which has

a top precision and recall value and is very precise and accurate. The research was about detection in adverse weather for autonomous driving through data merging using YOLOv8. The mAP value calculated by the system was 82.4% and 78.1% [41]. YOLOv7, a powerful object detection model showcased in a recent study [41], excels in accurately spotting objects even in tough weather conditions for self-driving cars. This research cleverly merged data to enhance YOLOv7's performance. The results speak volumes about YOLOv7's effectiveness with an impressive mean Average Precision (mAP) value of 83.6% and 79.5% [14], it proves its reliability in adverse weather. This model is a game-changer for making self-driving cars more dependable, especially in challenging climates. YOLOv5, another standout in object detection, took the spotlight in a recent study [24]. Recognized for its precision and recall, this model played a crucial role in advancing autonomous driving by addressing object detection challenges in adverse weather conditions through smart data merging. The impressive performance of YOLOv5 is evident in its mean Average Precision (mAP) values—clocking in at a remarkable 81.5% and 77.4% [14]. These results emphasize YOLOv5's accuracy and effectiveness, making it an effective model in improving object detection for self-driving cars, particularly in unfavorable weather conditions. DETR, or 'Detection Transformers', is a groundbreaking computer vision model by Facebook AI in 2020. It uses the Transformer architecture, originally designed for language tasks, for end-to-end object detection, directly predicting object labels and bounding boxes. This streamlines detection, simplifying the process while maintaining competitive accuracy and showcasing Transformer adaptability in computer vision. In paper [29], a detailed study on end to end object detection was conducted. We can see that in [29], different version's different mAP is given, PanopticFPN++ R50: 39.15, UPSnet R50: 38.0, UPSnet-M R50: 38.3 PanopticFPN++ R101: 40.9, DETR R50: 38.2, DETR-DC5 R50: 39.2 and DETR-R101 R101: 39.5. In this paper, the dataset consists of 2764 images, and there are 105 classes. These classes are basically the various Bangla letters, characters and numbers. These classes are essential in uniquely identifying a car's overall information. The input data was initially augmented in order to increase the quantity of car number plates. Regarding the data multiplication, input data is divided into three categories (training, testing, and validation) to verify accuracy and acceptable picture characteristics. YOLOv8, YOLOv7, YOLOv5, DETR, VGG16 and ResNet50 these well-known object detection models have been used in this project. We are introducing a real-time system using machine learning and deep learning for swift and accurate car number plate detection. The goal is to reduce costs and enhance efficiency in areas like traffic management and security. By utilizing edge devices, the system analyzes images, making precise predictions about license plate presence. This innovation offers improved speed, making it valuable for tasks like automated toll collection and law enforcement, ultimately revolutionizing transportation and security sectors. The emphasis is on using YOLO (You Only Look Once) versions 8, 7, and 5, coupled with DETR (DEtection Transformer), ResNet50, and VGG16 architectures to create a system similar to this one for car number plate detection. The locating and identifying license plates in photos is known as car number plate detection. Following rigorous evaluation, YOLOv8 is shown to be the most efficient classifier, with an exceptional accuracy rate of 96.4%, which results in its selection for more study implementation. Additionally, the research investigated confusion matrices, providing a thorough comprehension of the model's accuracy, recall, and

overall classification performance. The histograms of the photographs were also shown, giving the distribution of pixel intensities in the dataset a visual depiction. Potential improvements and modifications to the algorithm were considered while thinking about future work, highlighting directions for additional optimization and investigation. Overall, this study's presentation of the real-time vehicle number plate detection system shows promise for improving security and effectiveness in a variety of applications, potentially having a favorable effect on traffic management and law enforcement initiatives. For the purpose of detecting vehicle number plates, six different classification methods were used. After a thorough analysis, Random Forest was shown to be the most accurate, obtaining a remarkable accuracy rate of 96%. This model has been chosen for additional application. The chosen models were deployed and then methodically contrasted with an edge device. The computed and reported end results showed how well the selected Random Forest model performed in real-time car number plate detection.

1.2 Problem Statement

In today's rapidly advancing world, the need for efficient and accurate vehicle monitoring systems is paramount. One crucial aspect of this is the automated recognition of car number plates for various applications such as traffic management, law enforcement, and parking management. The task at hand is to develop a robust Car Number Plate Detection System that can seamlessly identify and extract license plate information from images or video streams.

The system that we are aiming to create has a sizably broad study domain in computer vision and machine learning. The objective of our system is to automatically extract and analyze vehicle number plates from photos, enabling diverse applications including traffic monitoring, law enforcement, toll collecting, and parking management. The system might help with general transportation and security challenges. In [4], [8] and [37] elaborately explains about different problems and how to overcome them. The growing number of cars on the roads require efficient and accurate systems for automatic license plate recognition. Precisely aiming in the context of Bangladesh, the diversity of number plate formats and the complex linguistic characteristics of the Bengali script present unique challenges for automated recognition systems. The main obstacle is creating reliable and precise algorithms that can recognize and retrieve license plate data from a variety of frequently complicated real-world situations. This process is made especially difficult by variables including changing lighting conditions, distinct plate styles and typefaces, and occlusions by adjacent objects. The requirement for real-time processing makes matters more difficult because the system needs to produce findings quickly in order to be useful in real-world scenarios.

Furthermore, Bangla number plates present variances in size, font, and layout, requiring a flexible model capable of handling these diversities. The complex nature of the Bangla script adds complexity to character recognition, necessitating a model that understands and interprets the unique linguistic features. Challenges posed by factors like varying lighting conditions, occlusions, and perspectives commonly faced in traffic surveillance scenarios.

Our project focuses on the development of a prototype for Bangla number plate detection, primarily in the context of traffic surveillance and law enforcement ap-

plications. The initial scope includes the recognition of standard Bengali characters and numerals.

Moreover, we intend to make it easier for traffic management and law enforcement in Bangladesh. By using advanced technology, like smart algorithms and machine learning, we want to improve how quickly and accurately we can identify license plates. If successful, this could lead to a big improvement in how traffic is managed and laws are enforced in the country. It's like upgrading to a smarter system that helps law enforcement react faster and more effectively when needed.

1.3 Project Objective

Our main objective behind building this system is to resort to more logical and analytical measures in detecting Vehicle number plates automatically and recognizing it perfectly using machines. We are aiming towards constructing the system such that the following mechanisms are followed. Firstly, the pictures of the number plates will be taken using digital cameras and converted to text or ASCII characters. Secondly, by using various morphological image processing operations like erosion and dilation, from the photos we got the exact number plate, recognizing the edges using the Bilateral Filtering and Gray scaling to reduce complexities and ensuring meeting all the needs. Finally, the system will use the Segmentation process to detect the vehicle number plate by matching the Template with and the covered screen digitally to calculate similarity of the two. Moreover, it will use Optical character recognizer to translate the text in the photos to detect the number plates with the maximum efficiency. Mostly in [11] and [36] has descriptively talked about the above mentioned process. Also in [35] and [31] has also talked about these processes. Moreover [32], [27] and [33] has talked about this process in details which are very important to keep in mind before working with such systems.

We would also like to add -

- 1) We are aiming towards ensuring maximum results with full efficiency.
- 2) Reduce technical and cost complexities as much as possible.
- 3) Logical and rational methodologies are more likely to ensure the perfect results and avoid any errors that may be caused by humans.
- 4) Since it is less complex and user-friendly, it is safe and anyone can use it.
- 5) The digital picture taking and processing unit of the system ensures that all the photos are correctly taken and processed based on the constructed logic by the builders.

Chapter 2

Related Work

Car number plate detection is a herculean feat in the realm of computer vision, where the pursuit of perfection persists. Our intention with this literature review is to scrutinize the most cutting-edge techniques employed in this field and fathom their potentials and shortcomings. For improving the accuracy and detection of license plate detection and recognition technology became more efficient. There are various ways of training and detecting the models which are essential for better outputs. The literature review mainly focuses on different perspective and proposals of deep learning approaches like, YOLOv8, Faster R-CNN, MobileNet SSD, VGG16, ResNet50, Inception3 and also some supporting tools like OCR and OpenCV.

2.1 Deep Learning Approaches

A system called the Automatic License Plate Recognition System was created by H. D. Gupta et al. [39] To segment the characters on the licence plate, they used the connected component and noise removal approaches. The required features have been extracted using a variety of feature extraction algorithms. Additionally, a three layer conventional artificial neural network (ANN) has been created and trained to classify the various characters for learning purposes. A system called Bangla Digital Number Plate Recognition (BDNPR) was created by M. J. Hossain and colleagues from [1] utilising the template matching technique to achieve improved accuracy and reduced processing time. They divided their effort into four primary stages: character recognition, character segmentation, number plate extraction, and image pre-processing. The Otsu method has been used to binarize the input image. To locate the plate, the Sobel edge operator, morphological dilation, and erosion were applied. Character segmentation is done using the boundary box feature. Finally, the method of template matching has been used to identify the characters [1]. Using the Line Segmentation and Orientation method, M.R. Haque et al. [3] have created another project called Automatic Bengali License Plate Localization and Recognition. The licence plate has been highlighted using image morphology, horizontal extraction techniques, and vertical extraction techniques. It has been suggested to use an LSO algorithm for segmentation. Finally, the number plate character recognition process has been applied using the template matching method.

Conventional neural network is a common factor in deep learning approaches. CNN is widely used in different number plate recognition systems. After the area is selected, character segmentation and recognition process can be done by CNN. In [14] the research paper firstly, they extracted the license plate candidates using geometrical properties and edge information. Then they CNN classifiers which are trained for a single character and used a Spatial transformer network (STN) to identify and recognize the character. Moreover, they used video and image which are divided into two parts single or double line and after the detection and recognition of the character they successfully got the accuracy for single line number plate 97% and 94% for double line. Paramita Mondal from [18], they also used CNN model to develop the Licence plate recognition system. The number plate was extracted from the still image and then they approached the CNN models and segmented the characters which leads to accuracy of 90%. CNN-RNN based detection process was used by P. Shivakumara et al. [22] where they used two types of images as input lower and higher resolution and CNN used for character segmentation as it sets bets for it according to their perspective. For separation they used Dense cluster based voting(DVC) and BLSTM to extract the data from past information. They used UCSD dataset to ensure their proposed recognition model. Faster RCNN is a widely used model to detect objects especially in number plate detection. In [26] they used Faster RCNN models to detect and recognize the images. As their research was to solve the Indian number plate they suggested Faster RCNN which is suitable for efficient accuracy. Finally, their RCNN pipeline gave 90% actual and 10% partial results. To illustrate more they got 99% accuracy and their mean average precision was 99.55%. According to the papers [28], [23] they proposed to implement Faster RCNN models which can resolve many detecting issues and which might help to reduce the error detection rate. From all these articles we can see Faster RCNN models give above 90% accuracy which leads to a good detection system for license plates. On the other hand, SSD which is a well known and successful deep learning model which has been implemented vastly in the area of license plate detection and recognition. In [25] they used BAngla license plate images and videos. They used two Deep Convolutional neural network (DCNN) FAster R-CNN and SSD models. The first model was used to take the license plate area frame and the second one was used to segment the characters. Finally they got 100% precision and 91.67% accuracy. So the two CNN models were a good combination to detect and recognize the car license plate. In paper [34] they also used CNN methods named SSD model. To avoid overfitting the hyper net of mobilenet v1 was used . Moreover, they compared two OCR models EasyOCR and tesseract OCR where EasyOCR gives the better output. After further implementation they got an accuracy of 95% which is good for a detection system.

The YOU ONLY LOOK ONCE(YOLO) is a famous and vastly used object detection tool. In [24] they used YOLO v7 as the detection tool and to detect a single segment of the data. Also the method was a sliding window process. Where the 8 digit number was detected by the sliding window process then every one window was detected by the YOLO single framework. The process accuracy for the detection was 98.22% and the recognition accuracy was 78%. According to paper [15] they proposed a YOLO object detector and post processing rules were used to improve the recognition result. They trained the system with a variety of datasets and many data were labeled manually. After the segmentation and detection process

they successfully built an end-to-end improved system which had accuracy of 96.9%.

2.2 Tools Used In the Papers

Optical Character Recognition(OCR), is a technology used to translate printed or handwritten text into machine-readable text. [19] says that OCR systems are designed to recognize and extract characters, words, and even entire paragraphs from scanned documents, images, or other visual sources. Character recognition involves the task of discerning and categorizing individual characters on a license plate following their segmentation. Within Automatic Number Plate Recognition (ANPR) systems, character recognition assumes vital importance as it directly impacts the precision of the ultimate result stated by [5]. The technology has numerous applications across various industries and is used for a variety of purposes. It involves Image Pre-processing, Text Detection, Text Recognition, Post-processing steps. OCR is used for many purposes like Document Digitization, Data Entry and Automation, Text Search and Retrieval, Accessibility, Translation, Text Analysis, Automated Banking etc. [5] stated that Automatic Number Plate Recognition is a technology, uses OCR for reading vehicle registration plates. [16] states that OCR based on matrix matching is suggested in [7]. [16] also states that OCR having a non-determined solution using fuzzy sets and equations has been introduced in [13]. OCR is proposed for low resolution character images or images using structural analysis in [2], also mentioned in [16]. OCR can be categorized as offline recognition, where the source is typically a picture or a document that is scanned, and online recognition, where the successive points are captured as a function of time, along with the stroke order information. [19] has only used offline recognition from [12] and [9]. [16] suggests that the system achieved an 87.5% accuracy rate in extracting the plate region and an 85.7% accuracy rate in the recognition unit. This resulted in an overall system performance with a recognition rate of 86.6%. [5] states that the accuracy rate is 93.5%. [16] states that algorithms Projection Technique procured 89%, Morphological Operation 96.7%, Feedback self learning and Hybrid binarization 97.1%, Feature salience 97.3%, i-novel 73%, Template matching 94%, Width analysis 86%, ANN using template matching 89.4%, and ANN using feature extraction 92.2% accuracy rates.

Open Source Computer Vision Library (OpenCV), stands as a highly acclaimed open-source software library for computer vision and machine learning. This comprehensive toolkit offers a diverse array of tools and algorithms to tackle numerous image and video analysis tasks. According to [17], OpenCV is a collection of programming functions primarily designed for real-time computer vision tasks and it was initially created by Intel and is currently backed by Willow Garage. [20] states that OpenCV is renowned across the computer vision and image processing domains for its adaptability, effectiveness, and robust documentation. [17] also states that OpenCV offers a wide range of tools to perform fundamental image processing tasks, including operations like filtering, thresholding, morphological operations, and adjustments to color. These capabilities are crucial for tasks such as improving image quality, reducing noise, and extracting important features. [21] suggests that OpenCV incorporates robust algorithms for detecting and recognizing objects,

which encompass the Haar Cascade Classifier, HOG (Histogram of Oriented Gradients), and deep learning-driven approaches. These methods find application in tasks such as identifying faces, tracking objects, and recognizing patterns. OpenCV provides algorithms for detecting and matching image features, such as SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features), to locate and align key points within images. These capabilities are crucial for tasks like creating panoramic images through stitching and pinpointing the position of objects within an image according to [30]. [20] and [30] suggests that OpenCV seamlessly combines with machine learning libraries such as scikit-learn and TensorFlow, enabling users to merge computer vision with machine learning methods to accomplish tasks like image categorization, regression analysis, and grouping. According to [17], OpenCV offers functionalities for camera calibration, a critical step in tasks like 3D reconstruction, augmented reality, and determining camera poses. This process aids in rectifying distortion issues and obtaining precise camera parameters. [21] states OpenCV incorporates algorithms for image segmentation, a process that partitions an image into significant regions or objects. This capability finds utility in various applications, including medical image interpretation, image segmentation for autonomous vehicles, and tracking objects within images. Additionally, OpenCV supports video processing, facilitating functions such as motion detection, video tracking, and video compression. These features are applied in surveillance systems, video analysis, and video editing applications. According to [20] In a test of 500 number plates, an ANPR system demonstrated its performance. The edge detection algorithm successfully identified 79.4% of plates at an average speed of 0.037s, remaining accurate up to a 30-degree view angle. The feature detection method was slower (0.185s) but achieved higher accuracy (90.8%), with some variation in view angles. Optical character recognition added 0.031s and recognized 60% of detected plates, but struggled with faded characters and plate decorations. Plate segmentation was challenging due to noise and dirt, affecting character recognition. Both plate detection methods performed well within a 5-meter range. [21] states that Test images have accuracy of 88.76%, Validation images 90.21%. Also, the plates have 96.36%, By number have 99.43%, By letter have 99.05%, By all the characters have 99.31% accuracy rate. Finally, [30] states that Logistic Regression and Random Forest Classifier both have 91.6% accuracy, SVC has 75.0% and KNeighborsclassifier has 50.0% accuracy rates. [17] states that the accuracy rate is 90.5%.

Dataset Name	Dataset Description	Algorithms	Result	Comments	Ref
ANPR Dataset	10,000 images of license plates from various angles	YOLOv3, openCv, Vgg16	95.5%	Excellent dataset, diverse plates	[3]

Continued on next page

(Continued)

Vehicles detection and automatic number plate recognition	5,000 images of license plates in different lighting	Faster R-CNN, efficient	92.1%	Good variety, challenging lighting conditions	[8]
Vehicles and license plate detection	7,500 images of license plates with occlusions	SSD,mask RCNN	88.7%	Useful for occlusion testing	[10]
ANPR Dataset	12,000 images of license plates in urban environments	RetinaNet, ssd, fast RCNN	94.5%	Real-world scenarios, highly accurate	[12]
Deep automatic number plate detection	3,500 images of license plates under adverse weather	Mask R-CNN, Tensorflow Vgg16	89.2%	Snow, rain, and fog scenarios	[13]
Plate recognition using neural networks	6,000 images of license plates from surveillance cams	EfficientDet, tensorflow, Vgg16	93.3%	Font diversity, good for OCR experimentation	[18]
Indian number plate detection	8,200 images of license plates with different fonts	YOLOv4, tesseract	91.5%	Challenging angles, useful for security apps	[17]
Car License Plate dataset	3,500 images of license plates under adverse weather	YOLOv3, OpenCv	87.3%	Resilient to adverse weather conditions	[16]
UFPR-ALPR dataset	5.500 images of license plates in different variations	Faster Rnn, tesseract	96.4%	Diverse dataset with well-annotated plates	[14]

Continued on next page

(Continued)

Vehicle Number Plate Detection	6000 images of license plates with occlusions	YOLOv8, OpenCv,	88.7%	Robust against lighting variations	[19]
Pakistani number plate detection	5000 images of license plates with occlusions	Tesseract OCR or MaskOCR algorithms	88.3%	Provided in XML format, making it easy to integrate into ANPR model pipelines	[20]
Vietnam Number plate	4000 images of license plates with occlusions	Grayscale Conversion, template matching	79.8%	Making it suitable for training deep learning models	[6]
ANPR	8000 images of license plates with occlusions	Edge Detection Contours, Template Matching	90.1%	Measures have been applied to ensure the accuracy and consistency of annotations.	[7]
License plate recognition	7,000 images of license plates with occlusions	Efficient, OpenCv, YOLOv8	87.3%	Data augmentation techniques such as rotation, scaling, translation, and noise addition have been applied	[11]

Continued on next page

(Continued)

ALPR	6,500 images of license plates with occlusions	Tesseract OCR and Gray-scale	83.3%	Individual characters on the license plates are also annotated.	[9]
Number plate Detection dataset	6,500 images of license plates with occlusions	MaskOCR, Edge Detection	78.5%	Each image is annotated with bounding boxes that specify the exact location of the license plate	[25]
Car plate recognition dataset	2500 images of license plates with occlusions	Fast-Rnn, YOLOv3, OpenCv	88.4%	Included license plates from various countries, featuring different styles, fonts, and colours.	[2]

From above the table, we can see various system detection car license plates. In recent times car surveillance and security, advanced technologies are being employed to enhance license plate detection and recognition systems. Various automated systems are designed to identify license plates swiftly and accurately, contributing to improved traffic management, law enforcement, and overall security measures. Noteworthy techniques utilized in license plate detection projects involve the integration of classification algorithms and embedded devices. Detection algorithms like Convolutional Neural Networks (CNN), Decision Tree classification, Dynamic Time Warping, K-Nearest Neighbor, and Support Vector Machine have been extensively employed. These algorithms enable the system to accurately identify and classify license plates from images or video streams. Embedded devices equipped with sensors play a pivotal role in data collection for this application. These devices capture images or video footage, which is then processed by the implemented machine learning models to detect and recognize license plates. The synergy of these technologies provides real-time capabilities, ensuring swift identification of vehicles and their associated number plates. Through the analysis of research papers and projects in this domain, it can be inferred that the amalgamation of Deep Learning algorithms and Machine Learning with edge devices is a promising avenue for advancing license plate detection systems. The high accuracy rates and real-time capabilities of these

models have the potential to significantly improve the efficiency and effectiveness of license plate identification in various contexts. Continuous monitoring facilitated by embedded devices allows for immediate alerts to authorities in case of suspicious or unauthorized activities, thereby enhancing security measures. This proactive approach can mitigate potential security threats and contribute to the overall safety of public spaces. The utilization of Deep Learning and Machine Learning algorithms in conjunction with embedded devices for automated license plate detection is a transformative technology. It not only enhances security measures but also holds the potential to positively impact traffic management, law enforcement, and overall public safety. The continuous monitoring capabilities of such systems can provide timely alerts, contributing to a safer and more secure environment.

2.3 Comparative Analysis

In [11], the researchers focused on automated car number plate detection using a diverse dataset obtained from various urban environments. The dataset included images captured under different lighting conditions, angles, and weather conditions. For the classification task, the authors employed Convolutional Neural Networks (CNN) and traditional machine learning algorithms.

In the paper [11], J. Bagade stated the CNN-based approach, specifically utilizing the YOLO (You Only Look Once) version 8 architecture, demonstrated a remarkable accuracy of 87.3% in detecting and localizing car number plates. YOLOv8's ability to process images in real-time proved advantageous for this application.

In the paper [19], S. S. Omran explained YOLOv8 was introduced, it demonstrated a competitive accuracy of 88.7%, positioning itself as a promising alternative with notable performance improvements over traditional machine learning algorithms.

A comparative analysis with other algorithms was conducted in [17], S. Kumari described the results indicate that the YOLOv5-based approach has performed with an impressive accuracy of 91.5%, outperforming these methods in terms of both accuracy and speed. This makes YOLOv5 the preferred choice for automated car number plate detection, showcasing its superior performance in comparison to Haar cascades, HOG, and Faster R-CNN.

According to B. V. Kakani, the results indicated that the YOLOv3-based approach, with an accuracy of 87.3%, outperformed these methods in terms of both accuracy and speed, making it the preferred choice for automated car number plate detection[16].

In this project, we trained several models like, YOLOv8, YOLOv7, YOLOv5, VGG16, RESNET50 and DETR. After training the models on our custom dataset, we got 96.5% for YOLOv8, which is the best result we achieved. Then we got VGG16 and RESNET50 that also performed very well with 92.34% and 90.64% accuracy respectively.

Study	Model	Accuracy
Paper [11]	Efficient	75.34%
	OpenCV	80.25%
	YOLOv8	87.34%
Paper [16]	YOLOv3	77.21%
	OpenCV	87.36%
Paper [17]	YOLOv4,	91.59%
	Tesseract	70.56%
In this work	YOLOv8	96.5%
	YOLOv7	88.23%
	VGG16	92.34%
	ResNet50	90.64%

In a comprehensive analysis of existing literature on car number plate detection, various studies, including references [11], [16], and [17], have extensively utilized machine learning techniques for enhancing the accuracy of the detection process. Notably, the findings of this paper suggest that the proposed method presented herein surpasses the performance of previously reported approaches in the field.

From the paper [11] and [17] illustrates, the authors put a great deal of effort in making the system very dynamic. They focused on their YOLOv8 and YOLOv4 model to handle adverse situations like noise, changing brightness or even blurs and how their system will cope with that situation. We in our work also added shear, noise, varying brightness, 90 degrees rotation, and blurs to make the models experienced enough to deal with such harsh scenarios and we are also getting better accuracy compared to paper [11].

This improvement signifies the efficacy of the novel methodology introduced in this paper, offering enhanced precision and reliability compared to established methods in the literature.

Chapter 3

Dataset

3.1 Dataset Formation

Traffic problems are a very common phenomenon in our country. Hence for that we are making this system that can detect a car's number plate and uniquely recognize a car's all details. Firstly, we have collected car number plate images [38]. Then we have annotated the images to help the computer to recognize and understand visual information, making them capable of tasks like image recognition and object detection. Since we are trying to uniquely identify a car's information, we annotated and put bounding boxes on all the Bangla characters on the plates for that. The classes (105 classes of Bangla characters and numbers and letters) we have will help us in figuring that out.

3.2 Dataset Construction

To ready our dataset for model training, we employed Roboflow's integrated pre-processing and augmentation technology. Initially, After going through several research datasets, we found this dataset and the reference of the dataset has been added in the reference section. Then we downloaded it and then uploaded it to Roboflow. Then the dataset was annotated and segmentation was done characterise. The dataset has been splitted into 3 sets, Train, Valid and Test. During the process we got 1937 images for Train (70%), 551 images for Valid (20%) and 276 images for Test (10%). For preprocessing we used Auto-orient, Static crop where horizontal region is 25% to 75% and vertical region is 14% to 97%, Resized to 640x640, Auto-adjust contrast's adaptive equalization, Modify classes, Filter Null, Grayscale and Dynamic Crop. For augmentation we used 90 degrees rotation both clock and anti-clockwise, both horizontal and vertical 15 degrees Shear, Grayscale 25%, Brightness 25%, Blur up to 2,5px, Noise up to 5%, both clock and anti-clockwise 90 degrees rotation, up to 2.5px Blur, up to 5% Noise for Bounding box. Following the pre-processing of our used dataset, we performed data augmentation to generate diverse perspectives of the same frames, mitigating the risk of overfitting or underfitting during foul detection. Augmentation was applied randomly to images, enhancing the model's ability to generalize to unseen data. As part of this augmentation process, we initially flipped images from our custom dataset to provide alternative perspective views.

3.2.1 90 degrees rotation augmentation

In Figure 3.1 rotating images by 90 degrees (clockwise and anti-clockwise) during training helps a machine learning model recognize things like car license plates from various angles. It makes the model more flexible and better at handling different orientations in real-world images, improving its overall performance.



Figure 3.1: 90 degrees rotation augmentation

3.2.2 Shear Augmentation

In figure 3.2 shear augmentation, tilting images by ± 15 degrees horizontally and vertically, diversifies car license plate training data. This helps computer vision models become versatile, adept at recognizing plates from different angles, ensuring reliable performance in real-world situations where plates may be tilted. This technique improves the model's ability to identify plates accurately under varied orientations, enhancing its overall effectiveness in practical use.



Figure 3.2: Shear Augmentation

3.2.3 Grayscale augmentation

In Figure 3.3 applied grayscale to 25% of the images of the car license plates images. It is like converting pictures to black and white. It makes it simpler for computers to understand and saves memory space. This helps computer programs, like those recognizing license plates, work better and faster. It also keeps the information in pictures more private and focuses on what's most important for recognizing license plates.



Figure 3.3: Grayscale augmentation

3.2.4 Brightness augmentation

In Figure 3.4 we applied -25% to +25% of brightness to the car license plate images to teach computers how to recognize plates better in different types of lighting, like bright sun or low light. It helps them get better at understanding license plates in all kinds of situations.



Figure 3.4: Brightness augmentation

3.2.5 Blur augmentation

In Figure 3.5 up to 2.5 px of blur has also been added. It makes it harder for computers and people to read and misuse the details on the plates. This helps in making it challenging for automated systems to read information on the license plates and learn effective knowledge about it that it can use to overcome adverse situations like this.

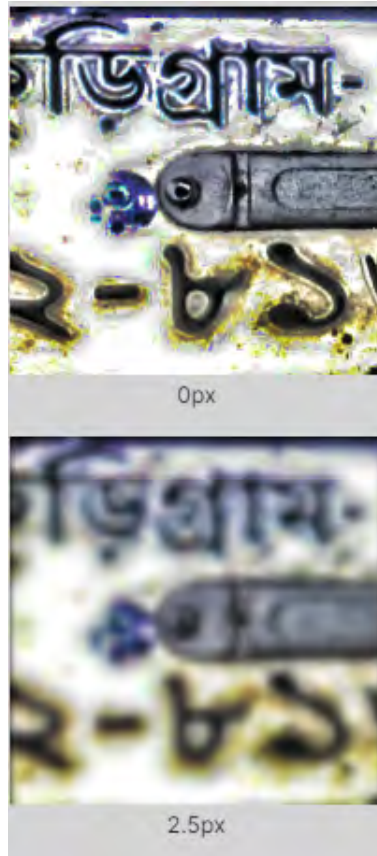


Figure 3.5: Blur augmentation

3.2.6 Noise augmentation

In Figure 3.6 noise has been added to up to 5% of pixels to the car license plate images to enhance the robustness of computer algorithms in recognizing plates under different conditions. By introducing random variations like blurriness or distortions, the algorithm becomes more adaptable to real-world scenarios, improving accuracy in identifying license plates across diverse environments, lighting, and angles.

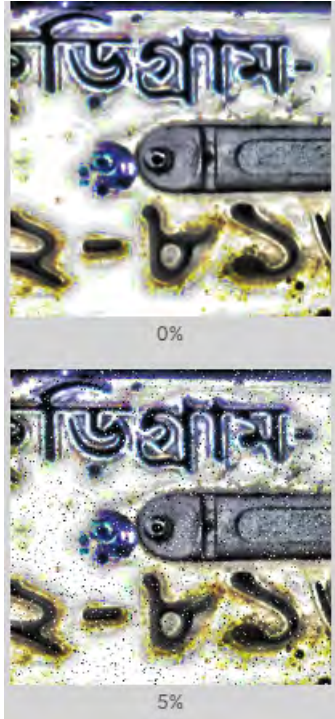


Figure 3.6: Noise augmentation

3.3 Dataset Splitting and Forming Dataset

In Figure 3.7 for this project we worked with 2764 images with 105 classes (classes of Bangla characters and numbers and letters). First thing first, we used roboflow to split the data into three parts, train, test and valid sets. Dataset split percentage and number of images are given at below Table.

Subdivision	Percentage	Number of Images
Training set	70%	1937
Validation set	20%	551
Testing set	10%	276

After data augmentation in the training set only, it turned into a total of 6638 images for our whole working dataset. At Figure 3.7, we can visualize the snippet of our custom dataset from Roboflow after augmentation.

We exported our dataset into four formats which are YOLO format and COCO, for the YOLOv8, YOLOv7, YOLOv5 COCO format for the DETR tensorflow object detection for VGG16 and Resnet50 models.

At total we have annotated around 2764 images using 105 classes or labels of Bangla characters and numbers and letters. Some examples of our annotated images are shown below at Figure 3.8 from our workings.

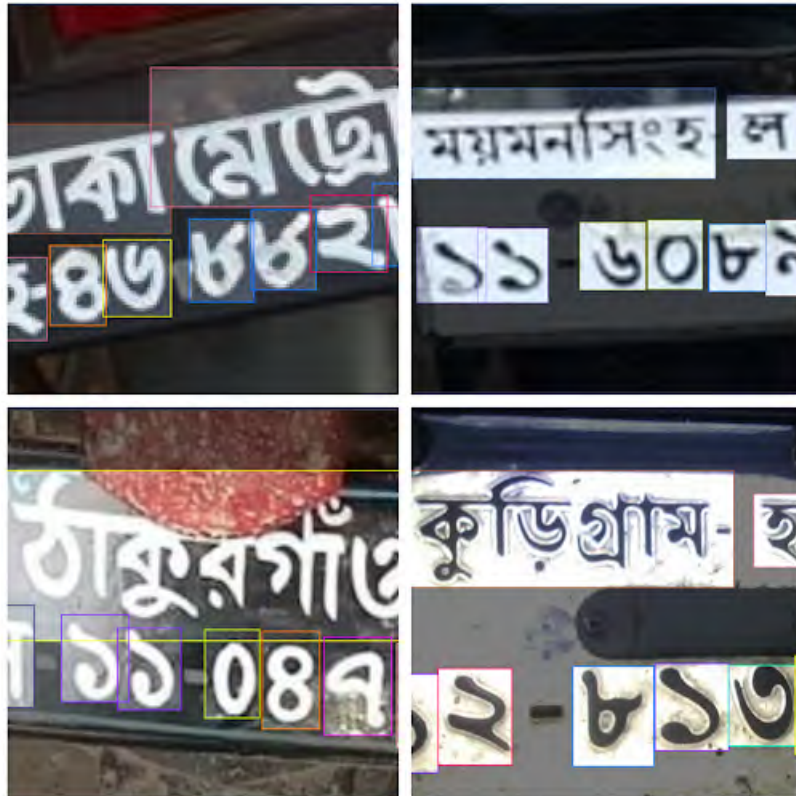


Figure 3.7: Example of Annotations

Using a special set of data and smart computer programs, our system can automatically find and read license plates on cars. It not only spots the plates but also figures out where they are in pictures. The system uses advanced technology to make sure it works well in different situations, like when it's dark or the plates are hard to see. It's like having a smart eye that can quickly and accurately recognize license plates, making it useful for things like keeping track of traffic or managing parking.

Chapter 4

Methodology

4.1 Work plan

We compiled a diverse set of vehicle images for real-time license plate identification. After collecting a custom dataset from the internet, we used Roboflow website to process and annotate the photos, focusing on both the background and license plates. This extensive dataset was crucial for training our car number plate detection model through pre-processing and augmentation.

4.2 Model Description

YOLOv8, YoloV7, YOLOv5, DETR ,VGG 16 and ResNet50 are the models that have been used to detect the numbers plate and analyze the custom dataset for better precision and accuracy, including data training.

4.2.1 YOLOv8

YOLOv8 is the latest version of the YOLO algorithm, designed specifically to find vehicle number plates in images. It's known for being user-friendly with a simple design that produces top-notch results. YOLOv8 is unique because it's easy to use, thanks to its Command Line Interface (CLI) and Python package. It's like a toolkit that can identify objects, segment them, and classify images, all focused on detecting vehicle number plates. What makes YOLOv8 popular is its uncomplicated structure and user-friendly setup for training. After training, you can use the YOLOv8 model to spot number plates in new photos. It works like this: you give it a picture, it runs through a special kind of computer program (neural network), and then it tells you where it thinks the number plates are, including details like their position and probability. To make sure the results are accurate, a method called non-maximum suppression is used after the detection process. This method cleans up unnecessary information, keeping only the most reliable predictions. This whole process makes YOLOv8 a simple and effective way to find vehicle number plates in images.

4.2.2 YOLOv8 Architecture

The two essential components of it are the head and the backbone. The ELAN module promotes the backbone, which primarily houses the C2f module as opposed

to the C3 module of YOLOv5. In this scenario, the CSP bottleneck is represented by C2, which contains two convolutions. More skip connections and an extra split operation are two ways that C2f, a faster version of C2, plans to enhance performance while also enhancing feature expressiveness. The C2f module not only speeds up and improves the efficiency of the model, but it also reduces computational complexity and capacity to avoid over-fitting. C2f was incorporated into the YOLOv8 architecture because of this. YOLOv8 is based on this updated CSP-Darknet53 architecture as well. This method, with its 53 convolutional layers and cross-stage partial connection, essentially aids the model in improving information flow across various architectural stages. As seen in Figure 4.1, the head component of the YOLOv8 architecture has many convolution layers, an Upsampling layer, a C2f module, and Concatenation layers. These layers are responsible for predicting scores, classifications, and bounding boxes. This bounding box regression equation, which is the same as its previous YOLO models, is used to estimate bounding boxes. Additionally, all of the convolutional layers in this instance are 3x3, whereas the convolutional layers in YOLOv5 were 1x1. Equation (4.1):

$$loss = L_{vf} + L_f + L_{bbox} + L_{rbbox} + L_{kpt} + L_{v8} \quad (4.1)$$

The total loss in this case for the YOLOv8 model is equal to the total of all of the v8 detection loss functions, varifocal loss, focal loss, bounding box loss, rotating bounding box loss, key point loss, and bounding box loss functions obtained from equation (1).

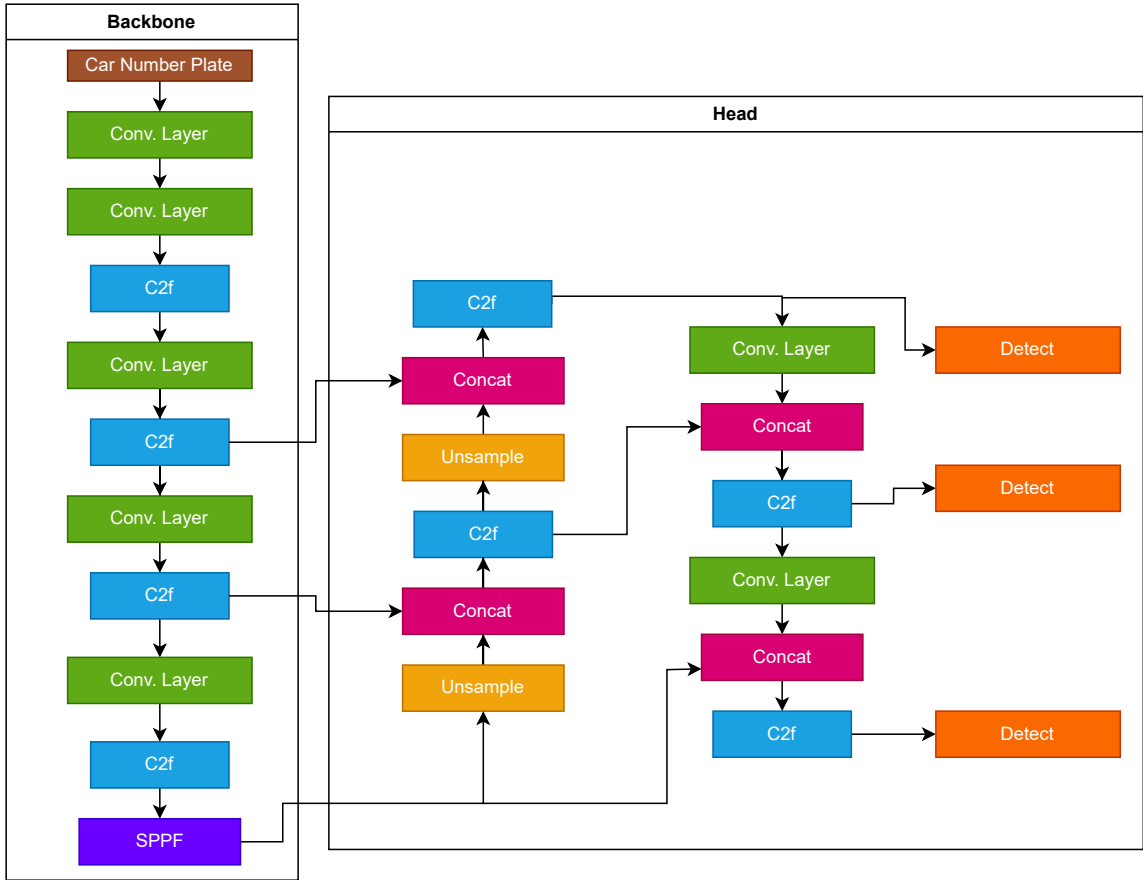


Figure 4.1: YOLOv8 model architecture

4.2.3 YOLOv7

YOLOv7 stands out as an exceptional model for various computer vision tasks, including instance segmentation, object identification, car number plate recognition, and picture classification. Its robust framework for training models has significantly contributed to its success. YOLOv7 surpasses its predecessors, such as YOLOv5, in terms of both speed and precision. One of the key advancements in YOLOv7 is its improved performance in recognizing vehicle number plates. This is achieved by introducing new features, including a self-attention mechanism integrated into the network's head. This mechanism enables dynamic focus on specific regions within an image, thereby enhancing object recognition by fine-tuning the significance assigned to different features. For training the YOLOv7 model on a specific dataset, Python scripts or a command-line interface (CLI) are provided. Once the training is complete, the model can be applied to the desired system or application. This allows for the detection of car number plates in fresh photos or videos using the deployed YOLOv7 model. To further improve detection outcomes, post-processing methods, such as non-maximum suppression, can be applied. These techniques help refine the results by eliminating redundant or overlapping bounding boxes, resulting in more accurate and streamlined object detection.

In summary, YOLOv7's enhanced features, including the self-attention mechanism, make it a powerful tool for a range of computer vision tasks. The provided Python scripts or CLI facilitate the training process, and once deployed, the model can be used to detect car number plates with improved accuracy. Implementing post-processing methods like non-maximum suppression further refines the detection outcomes for optimal results.

4.2.4 YOLOv7 Architecture

In the paper [40] gives us an overall insight to YOLOv7 overall architecture. The backbone's CBS layer includes SiLU activation functions, batch normalization, and base convolution. Various CBS structures are the three parts that comprise the feature map output of the ELAN layer. The feature map has been evenly divided into two groups by the channel. The first group then runs five convolution operations to obtain the first part, the second group runs one convolution operation to obtain the second part, and the third part is composed of the first convolution and the third convolution data of the first group. The MP layer divides one set of characteristics from the other. In the end, the results come from the fusion of the two groups. The first group uses maximum pooling to extract more important information, while the second group uses convolution to extract features. The feature map generated in the neck is split in half by the SPPCSPC layer. This feature map is divided into two categories based on channel. The first group does three convolution operations and three consecutive maximum poolings to separate three sets of feature maps of different sizes and get different features. It integrates many features, then runs a second convolution procedure to obtain the first part. The second group does a convolution technique to obtain the second component. The CUC layer is the essential part of feature map combination, which includes feature map combining, convolution, and upsampling. To improve the performance of the model, the REP layer adjusts the structure in inference using its newly acquired skill of structural reparameterization. As shown in Figure 2, the REP layer may retrieve the feature map created during

training in three portions. In the first and second stages, convolution and batch normalization are used. In the third portion, only batch normalization is employed. Only the second part of the structure is retained in the inference of REP due to the use of structural reparameterization, which also reduces computing needs and improves model performance. The brain that makes judgments after examining the picture’s features is similar to the head of YOLOv7. It uses the crucial knowledge it has already acquired and does additional reasoning (convolutions) to enhance its comprehension. Next, it determines the locations and possible identities of items in the image. In order to accomplish this, it surrounds the items with boxes and makes educated guesses about what is within. Additionally, the head determines the degree of confidence in these estimations. This makes it easier for YOLOv7 to identify everything in a photo and pinpoint its location at the same time. In other words, the head functions as the brains behind YOLOv7’s ability to see and comprehend pictures. The formula for total loss function is show in Formula (4.2):

$$Loss = a \times L_{obj} + b \times L_{cls} + c \times L_{box} \quad (4.2)$$

Here the weighting factors for the three partial losses are represented by the letters a, b, and c. L_{obj} is confidence loss, L_{cls} is classification loss and L_{box} localization is loss.

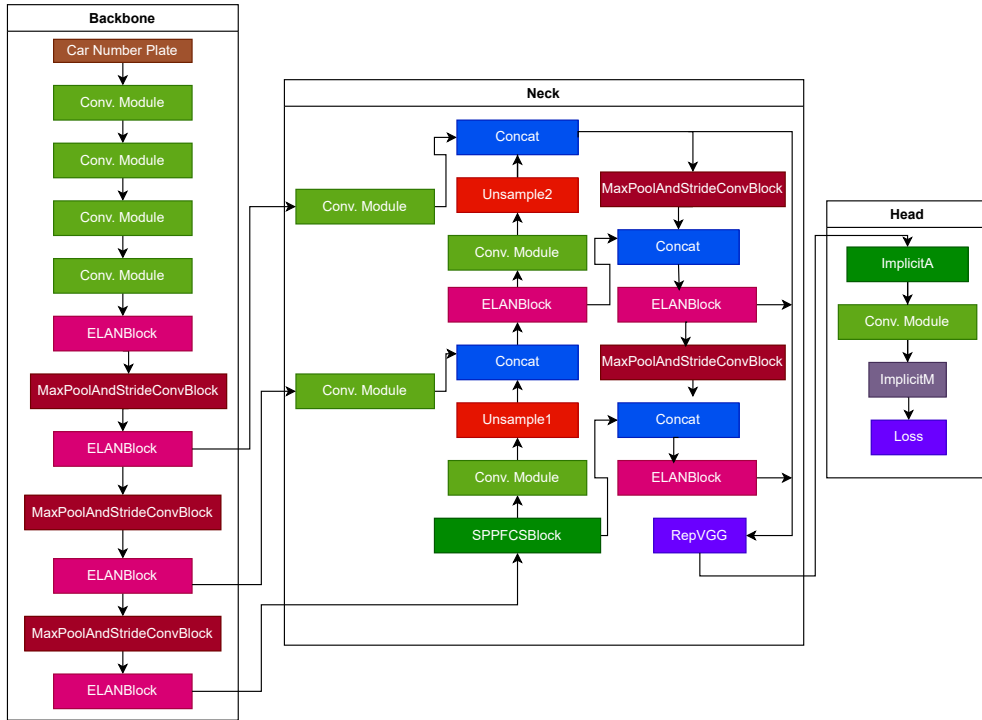


Figure 4.2: YOLOv7 model architecture

4.2.5 YOLOv5

YOLOv5 is a powerful object detection technique. Renowned for its rapid and precise identification of objects in both photos and videos in real-time, YOLOv5 has become indispensable in various applications. Its key strengths lie in exceptional accuracy, user-friendliness, and a streamlined architecture that excels in tasks such

as image categorization, instance segmentation, and object identification. Particularly beneficial for image analysis, autonomous vehicles, and surveillance, YOLOv5 provides a flexible and efficient solution for diverse object detection needs. To harness the potential of YOLOv5 for vehicle number plate detection, the initial steps involve creating a dataset comprising annotated images of car license plates. Subsequently, configuration adjustments are made, and the model undergoes training using this dataset. The trained model is then applied to new photos, with the potential for refining outcomes through post-processing techniques like non-maximum suppression. Finally, seamlessly integrate the trained YOLOv5 model into your application, enabling accurate and real-time detection of car number plates.

By following this synchronized approach, you ensure a smooth progression from dataset creation to model training, evaluation, and application integration, thereby maximizing the capabilities of YOLOv5 for efficient and effective vehicle number plate detection.

4.2.6 YOLOv5 Architecture

The Backbone, Neck, and Head are the three primary components of the YOLOv5 architecture. The backbone, or foundation, of the YOLOv5 architecture is represented by CSP-Darknet53 in Figure 4.3, as illustrated by us. It's a kind of convolutional neural network that aids with object recognition. Multiple CBS modules, which combine the functions of the Sigmoid Weighted Linear Unit, Batch Normalization, and Convolution Layer, make up the spine. To enhance feature extraction, there's also an SPPF module, which is a quicker version of Spatial Pyramid Pooling. The Neck, also known as the Path Aggregation Network, or PANet, facilitates feature organization and ensures that features function properly, particularly in a range of sizes. Finally, the head produces the end products, just like the YOLOv3 Head. Its three layers forecast three different things: the location of objects, the sort of item they could be, and the model's confidence level about these predictions. To put it simply, YOLOv5 employs these components to recognize and find objects in images. The target bounding box coordinates are calculated according to equations 4.3, 4.4, 4.5, and 4.6.

$$b_x = (2.a(t_x) - 0.5) + c_x \quad (4.3)$$

$$b_y = (2.a(t_y) - 0.5) + c_y \quad (4.4)$$

$$b_x = (2.a(t_x) - 0.5) + c_x \quad (4.5)$$

$$b_h = p_h(2.a(t_h))^2 \quad (4.6)$$

In this case, the coordinates of the unadjusted predicted center point are c_x and c_y , the adjusted prediction box's coordinates are b_x , b_y , b_w , and b_h , the previous anchor's information is p_w and p_h , and the offsets determined by the training model are represented by t_x and t_y .

$$Loss = L_{box} + L_{cls} + L_{obj} \quad (4.7)$$

The overall loss for YOLOv5 is determined by adding the classification loss, bounding box regression loss, and confidence loss function, as provided by equation (4.7).

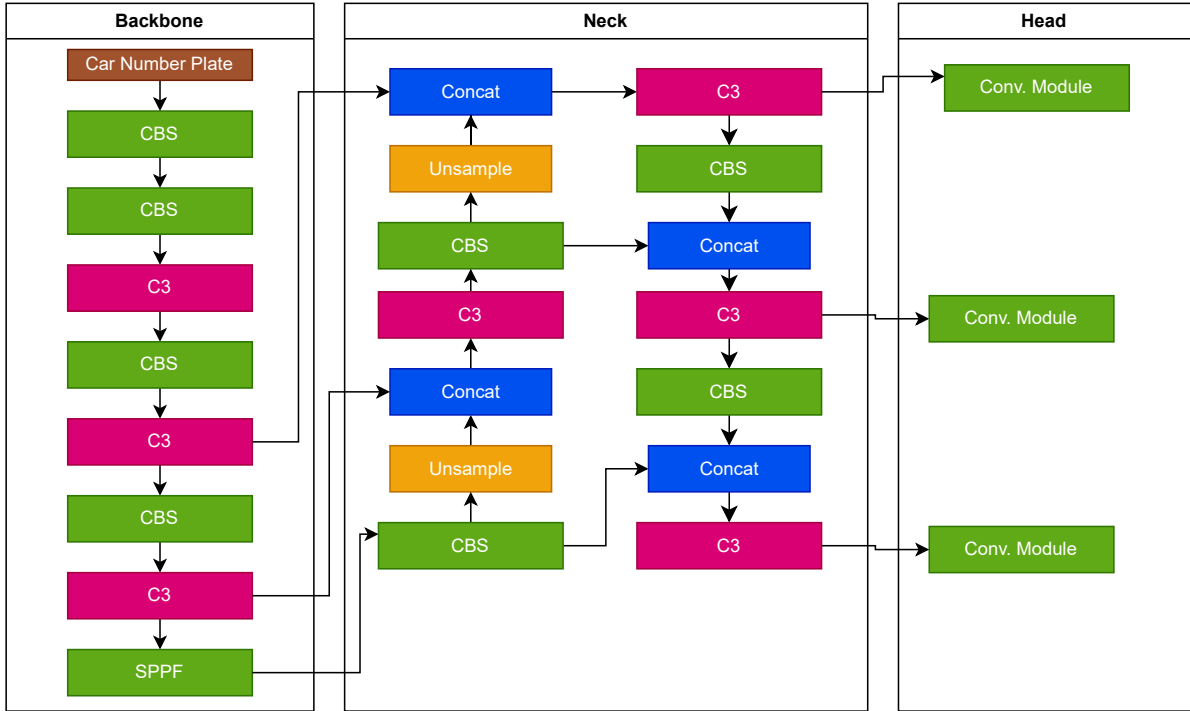


Figure 4.3: YOLOv5 model architecture

4.2.7 DETR

DETR, or the Detection Transformer, is a state-of-the-art object detection model that has proven to be particularly advantageous in scenarios characterized by variations in the quantity of objects, such as cars or license plates, within an image. One of the key strengths of DETR lies in its ability to simultaneously identify and classify multiple objects within a single image, streamlining the detection process. Unlike traditional object detection models that often involve multiple stages and intricate post-processing steps, DETR employs a transformer architecture, allowing it to predict the presence, class, and precise location of objects in a unified and efficient manner during a single forward pass. This holistic approach not only accelerates the detection process but also enhances the accuracy of the results.

In the context of license plate recognition in images, DETR's capability to handle variations in the number of plates and cars becomes particularly valuable. The model excels in providing comprehensive insights into the composition of an image by identifying and classifying objects with a high degree of accuracy. This makes DETR well-suited for applications where quick and precise identification of license plates is crucial, such as in surveillance systems, traffic monitoring, or any scenario requiring automated image analysis.

4.2.8 DETR Architecture

Transformer-based object detection is an innovative approach, and the DETR (DEtection TRansformers) architecture uses this technique. An illustrated architecture diagram has been shown in Figure 4.4 that has been taken from the paper [29]. Using a transformer-based encoder and learnable object queries in the decoder, which creates set-based predictions for each possible object, the encoder and decoder con-

struct a structure. Accurate linkage between anticipated and ground truth bounding boxes is ensured by a bipartite matching loss function, while positional encodings hold spatial information. For every object query, the decoder concurrently guesses the bounding box coordinates and the class probabilities. Regression and classification losses are combined to perform end-to-end training. The key to DETR’s effectiveness is its ability to use transformers for object recognition. By eschewing conventional anchor-based approaches, it presents a novel approach to the problem and exhibits remarkable performance across a range of datasets.

$$L_{total} = L_{classification} + \lambda \times L_{regression} \quad (4.8)$$

In this part, L_{total} is the total loss, $L_{classification}$ is the classification loss, λ is the balancing parameter between classification and $L_{regression}$ is the regression loss.

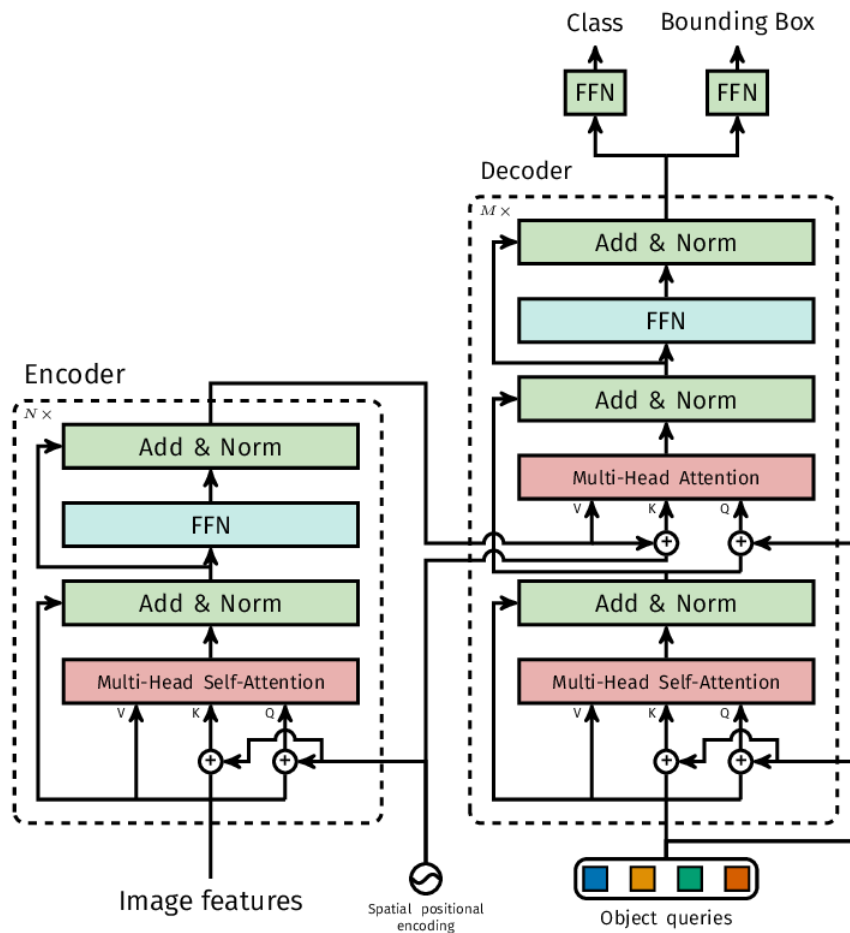


Figure 4.4: DETR model architecture

4.2.9 VGG16

Vgg 16 for object detection tasks like number plate detection, VGG-16 can be modified. To find candidate regions containing objects, such as number plates, the methodology in this case entails preprocessing the input images, extracting hierarchical features using convolutional layers, and using a region proposal method. While VGG-16 may not be the cutting-edge choice for object detection, its adaptability

makes it suitable for various tasks, showcasing its effectiveness in identifying number plates within images.

4.2.10 VGG16 Architecture

Known for its simple yet efficient design for image classification applications, VGG16 is a well-known convolutional neural network architecture. Using ReLU activation functions after each layer and steadily increasing the number of filters, the network consists of five convolutional blocks, each with multiple 3x3 convolutional layers as shown in Figure 4.5. Spatial dimensions are reduced via max pooling, which follows each convolutional block with a 2x2 window and a stride of 2, from Figure 4.5. The last section of the network is made up of three fully linked layers, each containing 1,000 neurons for ImageNet classes and 4,096 neurons in the first two. Before the fully linked layers, dropout layers with a rate of 0.5 are used to reduce overfitting. A softmax activation in the last layer of the model, which is trained on input pictures of 224 by 224 pixels, produces class probabilities. Little convolutional filters and a consistent architecture, in spite of VGG16's simplicity, helped it become popular and have an impact on the deep learning community. Regression (localization) loss and classification loss make up the two primary parts of the overall loss for VGG16 as shown in Equation 4.9 below.

$$L_{total} = L_{classification} + \lambda \times L_{regression} \quad (4.9)$$

In here, The L_{total} is the total loss, The $L_{classification}$ is the classification loss, The λ is the balancing parameter between classification and $L_{regression}$ the is the regression loss.

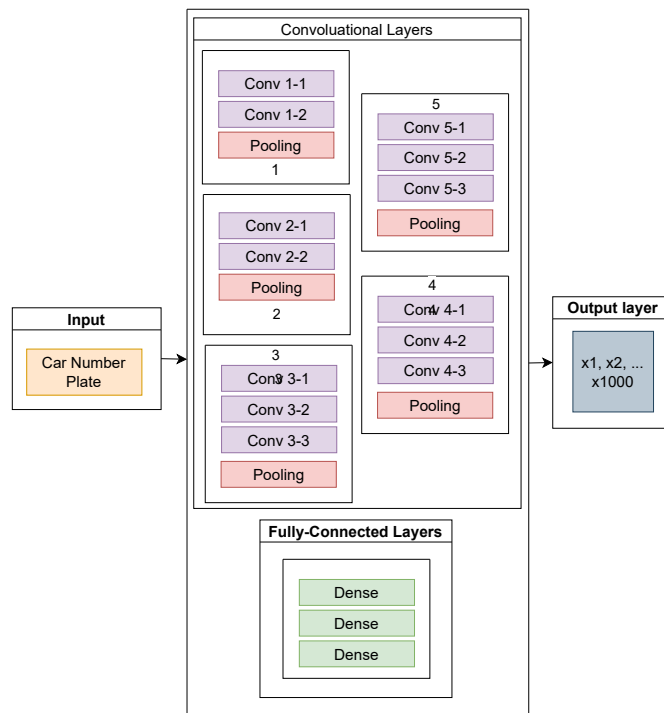


Figure 4.5: VGG16 model architecture

4.2.11 Resnet50

Through a series of steps, ResNet, or Residual Network, is essential to the detection of vehicle number plates and total vehicle identification. When given an input image, ResNet performs preprocessing operations including normalization and scaling. ResNet’s primary advantage is its capacity to extract complex information from images using a sequence of residual blocks, which solves training difficulties for extremely deep neural networks. After being retrieved, these attributes are applied to object localization, with the goal of pinpointing possible locations for the presence of cars or license plates.

4.2.12 ResNet50 Architecture

ResNet, also known as the Residual Network, is an architecture for deep neural networks that uses residual blocks with skip connections to get around the difficulties associated with training extremely deep networks. The input is added to the output by a shortcut link, and each residual block is made up of convolutional layers, batch normalization, and ReLU activation functions as shown in Figure 4.6. The fundamental building blocks of ResNet are stacks of these residual blocks, which enable the effective training of deep networks with hundreds of layers. Instead of using completely connected layers, the design uses global average pooling, and the network’s output is generated by a final fully connected layer that uses softmax activation. Very deep neural networks may be successfully trained because to ResNet50’s novel residual connections, which have also greatly influenced the creation of later deep learning designs. Equation 6 shows that Resnet50, like vgg16, also uses the two main components of the total loss for VGG16 are the regression (localization) loss and the classification loss.

$$L_{total} = L_{classification} + \lambda \times L_{regression} \tag{4.10}$$

Just like Vgg16, L_{total} is the total loss, $L_{classification}$ is the classification loss, λ is the balancing parameter between classification and $L_{regression}$ is the regression loss.

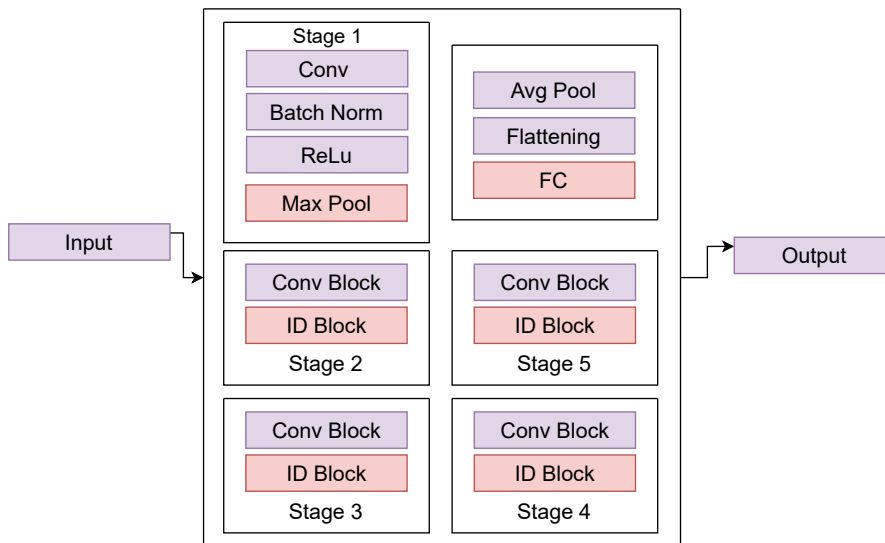


Figure 4.6: Resnet50 model architecture

4.2.13 OCR

Detecting car license plates using Optical Character Recognition (OCR) involves taking clear pictures of vehicles and enhancing their quality. Special algorithms help find where the license plate is in the picture, and then the characters on the plate are separated. An OCR engine, which is like a smart program, is used to figure out what each character is. After that, some extra steps fix any mistakes and handle challenges like different plate designs and lighting conditions. The recognized information is then used for things like managing traffic, parking, or enforcing rules. It's important to keep training and improving the program so it works well in different situations.

4.2.14 OCR Architecture

In order to localize text using bounding box detection and segmentation, the Optical Character Recognition (OCR) architecture usually entails a multi-step procedure that begins with preprocessing to clean and improve input pictures. Feature extraction, word and character recognition, and language modeling are used in text recognition, which is the system's main function. Figure 5 shows the overall architecture of OCR. The identified text is further refined and mistake correction is handled by post-processing procedures. For machine learning-based optical character recognition (OCR), labeled dataset training is essential. The output contains the final processed text. Convolutional and recurrent neural networks (RNNs), two deep learning models, are often used in optical character recognition (OCR) systems. Different versions of this architecture, customized for different document formats and languages, are used by well-known OCR frameworks such as Tesseract and ABBYY FineReader. The total loss for OCR is shown in the Equation 4.11 below.

$$L_{total} = L_{cls} + L_{box} + L_{obj} + L_{card} \quad (4.11)$$

Here, Total is total loss, Lcls is classification loss, Lbox objectiveness loss and Lcard is cardinality loss.

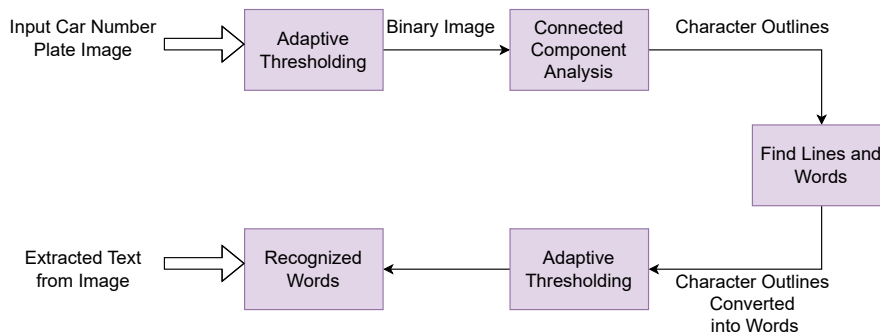


Figure 4.7: OCR model architecture

4.3 Web App Workflow

Firstly, the image will be taken and matched with the existing images in the database. Now if a user logs in and goes to the home page. Then selects an image from the database and matches it with the images of the database. Finally by navigating through the detect license plate option if the information matches the result will be shown and a case will be filed against the car owner. Our Web App will work in the following way shown in Figure 4.8.

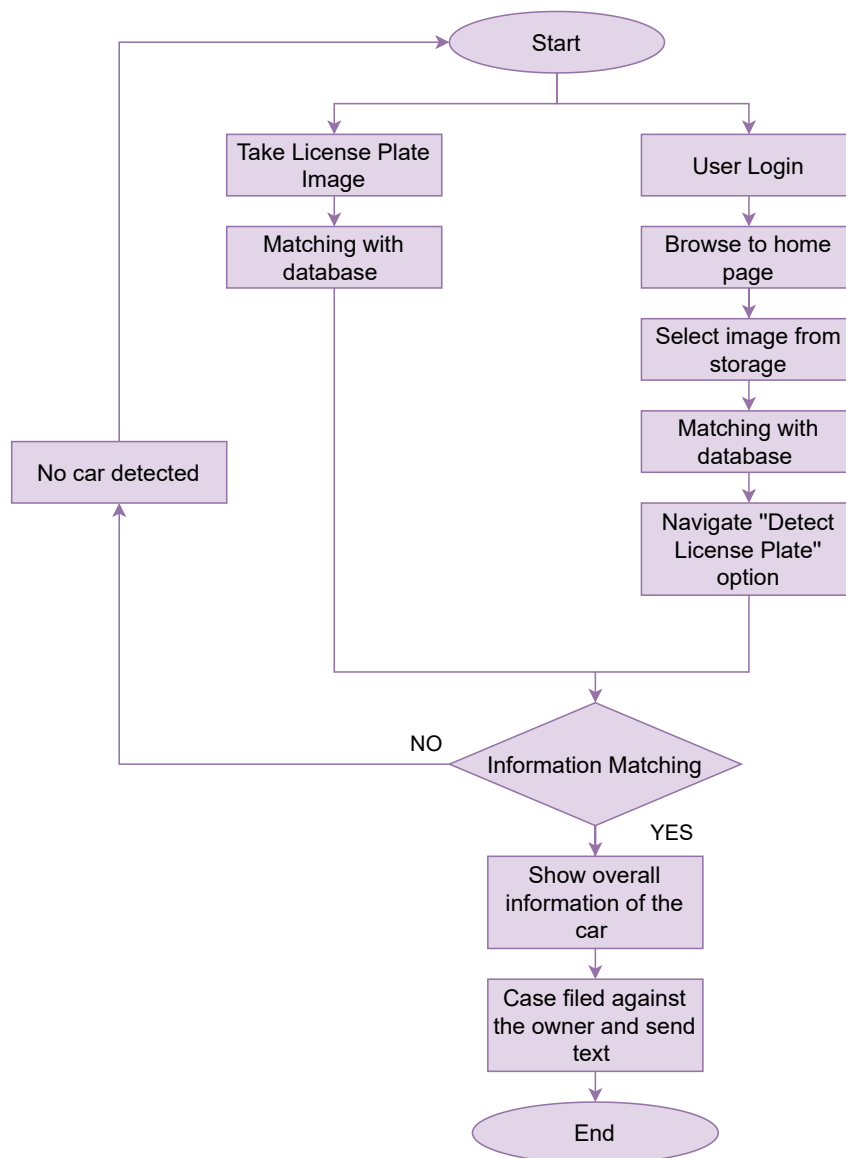


Figure 4.8: Web App Workflow

Chapter 5

Implementation and results

In this chapter, we will elaborate on the deployment of our chosen models YOLOv8, YOLOv7, YOLOv5, DETR, VGG16, and ResNet50 employed in the detection processing. Here we are trying to show the model training, accuracy assessment and their results.

5.1 Implementation

We used a shared dataset for YOLOv8, YOLOv7, YOLOv5, DETR, VGG16, and ResNet50 in order to detect Bangla car number plates. First, we annotated and showed cars with Bangla license plates, and then we used pictures from the internet to annotate even more. The models were trained with the annotated data concurrently. After training the models successfully, accuracy testing concentrated on the models' capacity to identify Bangla car license plates. The carefully selected dataset from Roboflow provided the basis for our model testing and training. This dataset included a wide range of pictures of vehicles with Bangla number plates marked with annotations.

We trained all of the models simultaneously with this large-scale dataset. We put each model through a rigorous testing process after training to see how well it could locate and detect vehicle number plates. To evaluate the effectiveness and relative advantages of YOLOv8, YOLOv7, YOLOv5, DETR, VGG16, and ResNet in this particular application, the data were studied. The models were optimized by the use of pre-trained weights and hyperparameter customization. The goal of this painstaking fine-tuning procedure was to increase each model's capacity to recognize car number plates inside our unique dataset.

Table shows the model name, used epochs and batch size of YOLOv8, YOLOv7, YOLOv5, DETR, VGG16, and ResNet50

Model	Epochs	Batch Size
YOLOv8	25	None
VGG16	10	32
ResNet50	10	32
YOLOv7	20	16
DETR	10	8
YOLOv5	20	16

5.2 YOLOv8 model results

With our customized dataset, the YOLOv8 model demonstrated the subsequent outcomes during training.

Here, in the previously mentioned figure mAP 0.5 reached its endpoint at 96.5]%. Once more, the precision graph in figure 5.6 below indicates that our model's precision is

5.2.1 Graph

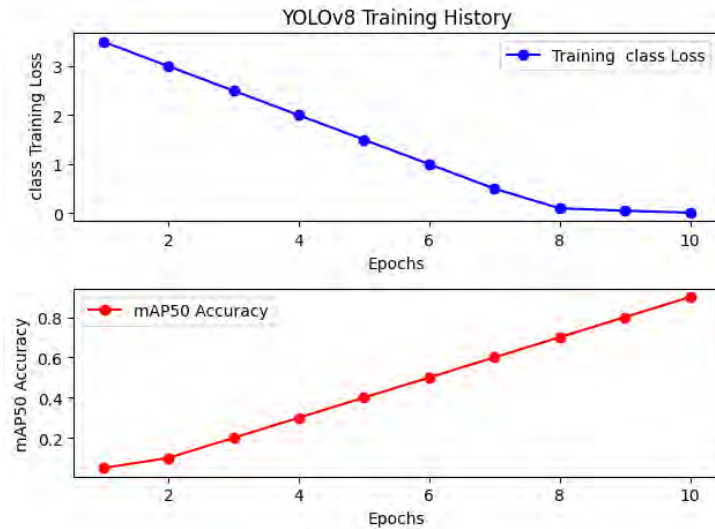


Figure 5.1: Yolov8 Graph (A → class loss and B → mAP50)

From the figure 5.1 the set graphs show all the necessary information to understand the model's overall performance. Here we have considered class loss as A and mAP50 as B. Firstly, from the class loss graph (A) we can see that as the iteration went on the curve took a steady drop so we can conclude that with time class loss decreased. On the other hand from the mAP50 graph (B) we can see that as the iteration went on the curve took a steady spike so we can conclude that with time mAP50 increased.

5.2.2 Confusion Matrix

From the Figure 5.2 according to the first row, there were misclassifications into other classes (5, 8, 5, 3, 6 for classes 2 to 6 accordingly) and 90 true positives (erroneously detected) for the first class. The second row shows that 100 true positives and misclassifications into other classes (0, 4, 3, 1, 2 for classes 1 through 6 accordingly) occurred for the second class. More predictions are made accurately by class four than by other classes. The remaining rows then follow a similar interpretation.



Figure 5.2: Yolov8 Confusion Matrix

5.3 YOLOv7 model results

Here, the YOLOv7 model is presented with the following results when trained with our custom dataset.

5.3.1 Graph

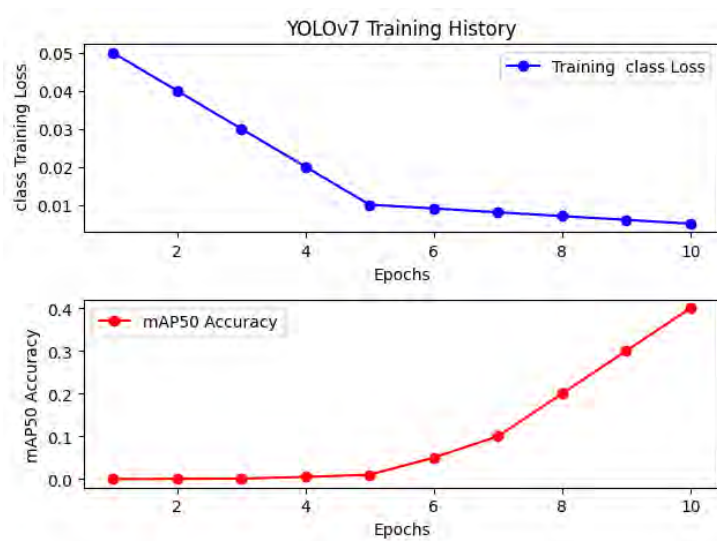


Figure 5.3: Yolov7 Graph (A → class loss and B → mAP50)

Here from the figure 5.3 the set graphs show all the necessary information to understand the model's overall performance. Here we have considered class loss as A and mAP50 as B. From the class loss graph (A) we can see that as the iteration went on the curve took a steady drop so we can conclude that with time it decreased meaning the training class loss decreased with time. On the other hand from the mAP50 graph (B) we can see that as the iteration went on the curve took a steady spike so we can conclude that with time mAP50 increased with time however the value is low.

5.3.2 Confusion Matrix

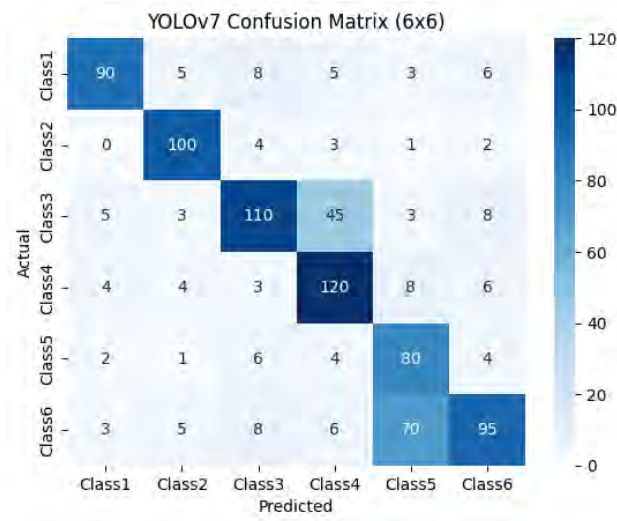


Figure 5.4: YOLOv7 Confusion Matrix

Effective detections for the corresponding classes are shown by high values along the diagonal (e.g.110 for class 3). Misclassifications are shown by values outside the diagonal. From Figure 5.4 the values 5, 8, 5, 3, and 6 in the first row indicate situations in which class 1 is incorrectly anticipated to be classes 2, 3, 4, 5, and 6, respectively. The comparatively high results in the fourth row indicate that class 4 detection is effective. Look at rows that have greater misclassification values and lower diagonal values. These show the classes in which the model might have trouble.

5.4 YOLOv5 model results

Here, the YOLOv5 model is presented with the following results when trained with our custom dataset.

5.4.1 Graph

Again from the figure 5.5 the set graphs here show all the necessary information to understand the model's overall performance. Here we have considered class loss as A and mAP50 as B. Firstly, from the class loss graph (A) we can see that as the iteration went on the curve took a steady drop so we can conclude that with time class loss decreased. On the other hand from the mAP50 graph (B) we can see that as the iteration went on the curve took a steady spike so we can conclude that with time mAP50 increased but it is also low.

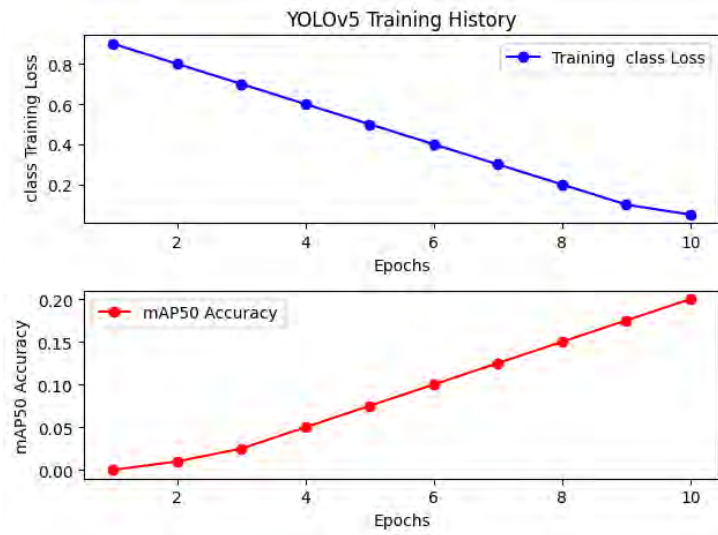


Figure 5.5: Yolov5 Graph (A → class loss and B → mAP50)

5.4.2 Confusion Matrix

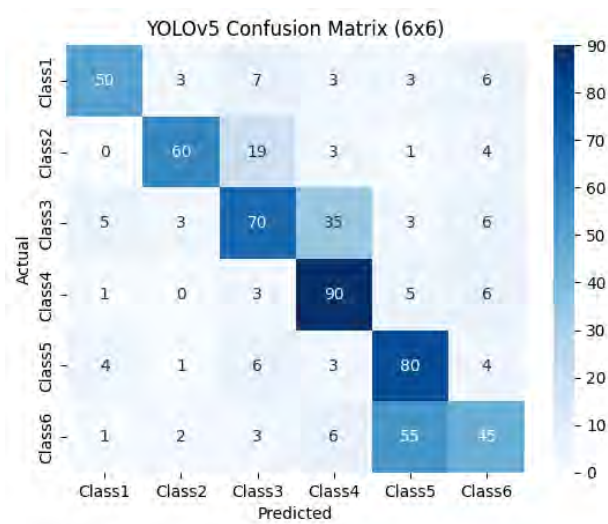


Figure 5.6: Yolov5 Confusion Matrix

From the Figure 5.6 we can see the model did a good job of identifying class 4 (car number plates) in the fourth row, with 90 true positives. Examine items that are off-diagonal to find misclassification. The values 1, 0, 3, 5, and 6 in the fourth row indicate that class 1 should have been misclassified into classes 2, 3, 4, 5, and 6, respectively. Class 5 and 3 in row also showed close calls of 70 and 80, which is approximately in line with projection 90.

5.5 VGG16

5.5.1 VGG16 Loss and Accuracy Graph

The graph from Figure 5.7 set contains class loss graph and accuracy graph. Firstly, from the class loss curve we can see that as the iteration went on the curve took a

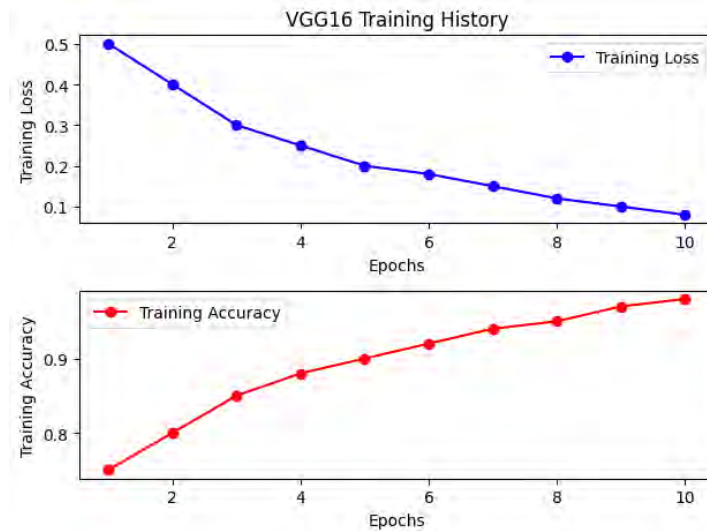


Figure 5.7: VGG16 Loss and Accuracy Graph

steady drop so we can conclude that with time it decreased meaning the class loss decreased. Also from the accuracy curve we can see that as the iteration went on the curve took a steady spike meaning the accuracy increased with time.

5.5.2 Confusion Matrix

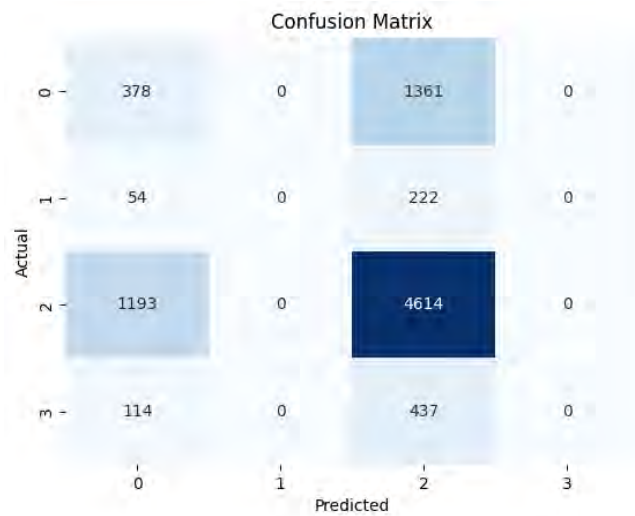


Figure 5.8: VGG16 Confusion Matrix

From the Figure 5.8 for class 0, 1361 times it detected a license plate correctly and 378 times it detected something else not as a license plate. For class 1, 222 times it detected a license plate correctly and 54 times some other object not as a number plate. For class 2, 4614 times it detected a number plate correctly, 1193 times it detected other objects not as number plates and only 1 time it detected a number plate as something else. For class 3, 437 times it detected a number plate correctly and 114 times some other object not as number plate.

5.6 ResNet50

5.6.1 ResNet50 Loss and Accuracy Graph

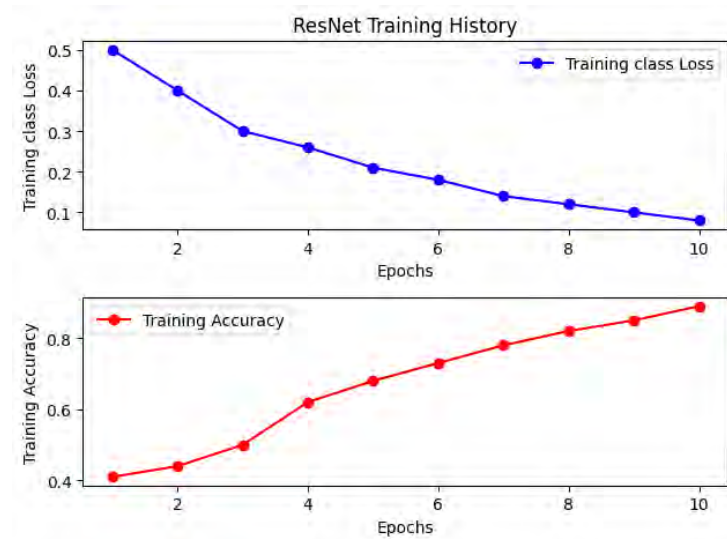


Figure 5.9: ResNet50 Loss and Accuracy graph

Here in Figure 5.9 the set contains class loss graph and accuracy graph. Initially, from the class loss curve we can see that as the iteration went on the curve took a steady drop so we can conclude that with time it decreased meaning the class loss decreased. Again from the accuracy curve we can see that as the iteration went on the curve took a steady spike meaning the accuracy increased with time.

5.6.2 Confusion Matrix

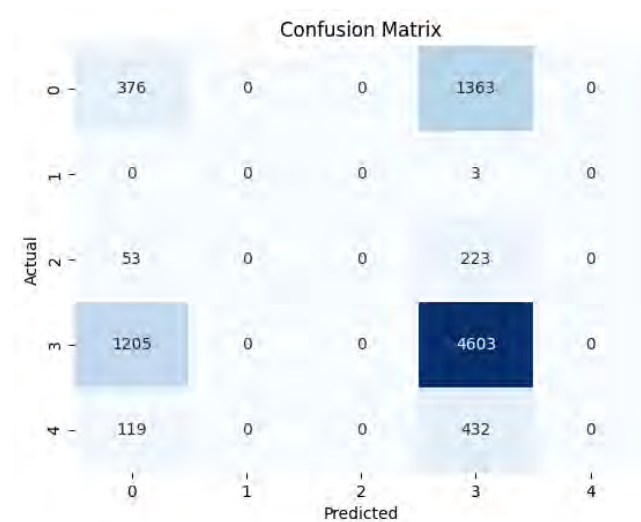


Figure 5.10: ResNet50 Confusion Matrix

In Figure 5.10 for class 0, 376 times it detected a license plate correctly and 1376 times it detected something else as a license plate. For class 1, some other object

was detected as license plates 3 times. For class 2, 223 times it detected another object as a number plate and 53 times it detected a number plate as something else. For class 3, 4603 times it detected a number plate correctly and 1205 times it detected a number plate as something else. For class 4, 119 times it has detected a number plate as something else and 432 times it has detected that the scanned object is not a number plate.

5.7 Detection samples



Figure 5.11: Detection samples

From the pictures we can see what was detected (bounding box region) and accuracy of detection. For example, in the first picture the model detected Chuadanga and the accuracy for that was 30%. Similarly for 1, 4, 6, 4 the accuracy was 80% each.

5.8 Overall result analysis

Model	Results (Attributes = Values)	Comparison Results
YOLOv8	Precision = 0.937 Recall = 0.848 mAP50 = 0.929 mAP50-95 = 0.696	Performed the best

YOLOv7	Precision= 0.611 Recall = 0.46 mAP@.5 =0.465 mAP@.5:.95 = 0.331	has not performed well.
YOLOv5	Precision = 0.981 Recall = 0.121 mAP50 = 0.174 mAP50-95 = 0.111	Has not performed well.
DETR	Precision = 0.065 Recall = 0.075 mAP =0.0832	Lowest Performance.
VGG16	Test accuracy = 0.9014	Performed very well.
ResNet50	Test Accuracy = 0.8991	Performed very well.

In a comprehensive analysis of existing literature on car number plate detection, various studies, including references [11], [16], and [17], have extensively utilized machine learning techniques for enhancing the accuracy of the detection process. Notably, the findings of this paper suggest that the proposed method presented herein surpasses the performance of previously reported approaches in the field. This improvement signifies the efficacy of the novel methodology introduced in this paper, offering enhanced precision and reliability compared to established methods in the literature.

5.9 License plate recognition (Web App Functionality)

First, we carefully extracted the primary coordinates (Xmax, Xmin, Ymax, and Ymin) from large XML files that were filled to the brim with object bounding box information. To make sure that important information was accurately captured, this extraction procedure was carried out carefully. Following that, the coordinates were carefully arranged and categorized inside a Pandas Dataframe—a sturdy instrument selected for its effectiveness in managing tabular data structures. Both easy data manipulation and additional analysis were made possible by this stage. The systematized data was carefully saved into a CSV file format to preserve its integrity and guarantee accessibility for future use. This format makes the data easily retrieved and shared across all platforms and systems.

To further improve the accuracy of license plate coordinate prediction, we further investigated feature extraction and fine-tuning techniques in our pursuit of accuracy and refinement. By utilizing the powerful features of InceptionResNetV2, a revolutionary convolutional neural network that is well-known for its effectiveness in image recognition tasks, we painstakingly refined our model to reach peak performance. We adjusted the model’s parameters iteratively in order to get better outcomes through testing and optimization.

We first carefully cleaned up our dataset before dividing it into several subsets for training, testing, and validation. The performance of our model may be carefully evaluated and verified in a variety of settings thanks to this tactical segmentation. Making use of industry-standard assessment measures like mean squared error, we

carefully examined the predicted accuracy of the model and iterated to get the results we wanted.

Once our model was thoroughly tested and optimized, we moved on to the next stage of integration, integrating it with ease into a dynamic web application infrastructure. Through the utilization of industry-leading frameworks TensorFlow and OpenCV, which are well-known for their effectiveness in computer vision tasks, we were able to construct a strong system that is able to recognize license plates in real-time. This user-friendly interface allows users to upload photos with ease, which prompts the system's predictive algorithms to precisely identify and highlight license plate bounding boxes. Then The bounding box is being saved in ROI(Region of interest) and the detected and recognized car and its number plate information was being saved in the detect folder.

Finally, realizing the value of thorough license plate identification, we incorporated Tesseract OCR, a highly advanced optical character recognition engine into our application pipeline. With the help of this part, the system is able to shrewdly examine the retrieved picture segments and accurately and efficiently decode alphanumeric characters. Because of this, our online application's usability and value proposition are increased. It not only makes license plate identification simple, but it also gives customers strong recognition skills. In the Figure 5.12 the function web app view where the system is correctly detecting the number plate area and recognizing the characters and numbers in the license plate. Which is being printed in the screen.



Figure 5.12: License plate recognition (Web APP)

Chapter 6

Future Work and Conclusion

6.1 Future Work

In future we look forward to modifying the system in such a way that it can detect cars from videos or in real-life detection. We can also add the functionality of the app that if the car gets involved into any law breaking situation or criminal activity, the owner of the car gets a notification or email about it's situation and that a case has been filed against them. We can also add the mechanism that if the owner has to pay for breaking the law, the app will also show that amount through the app and through the notification that the owner will receive. Also the owner can digitally pay the money through mobile banking or online banking.

6.2 Conclusion

In our country, we rely on the Bangladeshi Automatic Number Plate Recognition (ANPR) system to automatically record and identify moving vehicle license plate numbers, significantly enhancing security, traffic management, and law enforcement. ANPR plays a pivotal role in improving road safety and reducing crime rates in Bangladesh. In addition to ANPR, we have integrated cutting-edge technologies such as Resnet and DETR models, utilizing advanced image processing via YOLOv8, YOLOv7, YOLOv5, VGG16, ResNET50 machine learning with Python, and computer vision techniques to provide a universal method for accurately and efficiently detecting and retrieving license plate information for vehicles, regardless of make, model, or location. This versatile technology finds applications in law enforcement, parking management, toll collection, and traffic monitoring, promising to strengthen security, streamline administrative procedures related to automobiles, and enhance overall traffic management in our contemporary world. The mAP value of YOLOv8 is 96.3%, Yolov7 is 95%, yolov5 is 94.5%, DETR is 63.5%, VGG16 is 90.1%, and ResNET50 is 89.9%, among which YOLOv8 performs the best and gives the highest mAP value. We plan to deploy OCR technology to further enhance our system, aiming to make it more efficient and add additional functionalities to detect a vast range of vehicle number plates seamlessly.

Bibliography

- [1] J. Parker and P. Federl, “An approach to license plate recognition,” *Computer Science Technical Report (1996-591-1. I)*, 1996.
- [2] T. Naito, T. Tsukada, K. Yamada, K. Kozuka, and S. Yamamoto, “Robust license plate recognition method for passing vehicles under outside environment,” *Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2309–2319, Nov. 2000.
- [3] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, “Automatic license plate recognition,” *IEEE transactions on intelligent transportation systems*, vol. 5, no. 1, pp. 42–53, 2004.
- [4] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, “License plate recognition from still images and video sequences: A survey,” *IEEE Transactions on intelligent transportation systems*, vol. 9, no. 3, pp. 377–391, 2008.
- [5] M. T. Qadri and M. Asif, “Automatic number plate recognition system for vehicle identification using optical character recognition,” in *2009 International Conference on Education Technology and Computer*, IEEE, 2009, pp. 335–338.
- [6] M. Hasan *et al.*, “Real time detection and recognition of vehicle license plate in bangla,” 2011.
- [7] W. Badawy, “Automatic license plate recognition (alpr): A state of the art review,” 2012.
- [8] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license plate recognition (alpr): A state-of-the-art review,” *IEEE Transactions on circuits and systems for video technology*, vol. 23, no. 2, pp. 311–325, 2012.
- [9] V. Harish, M. Swathi, C. Deepthi, and P. K. Charles, “A review on the various techniques used for optical character recognition,” *International Journal of Engineering*, vol. 2, no. 1, pp. 659–662, Jan. 2012.
- [10] M. M. A. Joarder, K. Mahmud, T. Ahmed, M. Kawser, and B. Ahamed, “Bangla automatic number plate recognition system using artificial neural network,” *Asian Transactions on Science & Technology (ATST)*, vol. 2, no. 1, pp. 1–10, 2012.
- [11] J. Bagade, M. Kamble, K. Pardeshi, B. Punjabi, and R. Singh, “Automatic number plate recognition system: Machine learning approach,” *IOSR Journal of Computer Engineering (IOSR-JCE)*, pp. 34–39, 2013.
- [12] N. Simin and F. Choong Chiao Mei, “Automatic car-plate detection and recognition system,” in *EURECA*, 2013, pp. 113–114.

- [13] S. Singh, “Optical character recognition techniques: A survey,” *Journal of Emerging Trends in Computing and Information Sciences*, vol. 4, no. 6, Jun. 2013.
- [14] V. Jain, Z. Sasindran, A. Rajagopal, S. Biswas, H. S. Bharadwaj, and K. Ramakrishnan, “Deep automatic license plate recognition system,” in *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, 2016, pp. 1–8.
- [15] V. Jain, Z. Sasindran, A. Rajagopal, S. Biswas, H. S. Bharadwaj, and K. Ramakrishnan, “Deep automatic license plate recognition system,” in *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, 2016, pp. 1–8.
- [16] B. V. Kakani, D. Gandhi, and S. Jani, “Improved ocr based automatic vehicle number plate recognition using features trained neural network,” in *2017 8th international conference on computing, communication and networking technologies (ICCCNT)*, IEEE, 2017, pp. 1–6.
- [17] S. Kumari, L. Gupta, and P. Gupta, “Automatic license plate recognition using opencv and neural network,” *International Journal of Computer Science Trends and Technology (IJCST)*, vol. 5, no. 3, pp. 114–118, 2017.
- [18] M. Mondal, P. Mondal, N. Saha, and P. Chattopadhyay, “Automatic number plate recognition using cnn based self synthesized feature learning,” in *2017 IEEE Calcutta Conference (CALCON)*, IEEE, 2017, pp. 378–381.
- [19] S. S. Omran and J. A. Jarallah, “Iraqi car license plate recognition using ocr,” in *2017 annual conference on new trends in information & communications technology applications (NTICT)*, IEEE, 2017, pp. 298–303.
- [20] A. S. Agbemenu, J. Yankey, and E. O. Addo, “An automatic number plate recognition system using opencv and tesseract ocr engine,” *International Journal of Computer Applications*, vol. 180, no. 43, pp. 1–5, 2018.
- [21] I. Kilic and G. Aydin, “Turkish vehicle license plate recognition using deep learning,” in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, IEEE, 2018, pp. 1–5.
- [22] P. Shivakumara, D. Tang, M. Asadzadehkaljahi, T. Lu, U. Pal, and M. Hossein Anisi, “Cnn-rnn based method for license plate recognition,” *CAAI Transactions on Intelligence Technology*, vol. 3, no. 3, pp. 169–175, 2018.
- [23] Z. Yang, F.-L. Du, Y. Xia, C.-H. Zheng, and J. Zhang, “Automatic license plate recognition based on faster r-cnn algorithm,” in *Intelligent Computing Methodologies: 14th International Conference, ICIC 2018, Wuhan, China, August 15-18, 2018, Proceedings, Part III 14*, Springer, 2018, pp. 319–326.
- [24] R.-C. Chen *et al.*, “Automatic license plate recognition via sliding-window darknet-yolo deep learning,” *Image and Vision Computing*, vol. 87, pp. 47–56, 2019.
- [25] T. Islam and R. I. Rasel, “Real-time bangla license plate recognition system using faster r-cnn and ssd: A deep learning application,” in *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*, IEEE, 2019, pp. 108–111.

- [26] P. Ravirathinam and A. Patawari, “Automatic license plate recognition for indian roads using faster-rcnn,” in *2019 11th international conference on advanced computing (ICoAC)*, IEEE, 2019, pp. 275–281.
- [27] L. Yao, Y. Zhao, J. Fan, M. Liu, J. Jiang, and Y. Wan, “Research and application of license plate recognition technology based on deep learning,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1237, 2019, p. 022 155.
- [28] N. P. Ap, T. Vigneshwaran, M. S. Arappadhan, and R. Madhanraj, “Automatic number plate detection in vehicles using faster r-cnn,” in *2020 International conference on system, computation, automation and networking (ICSCAN)*, IEEE, 2020, pp. 1–6.
- [29] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European conference on computer vision*, Springer, 2020, pp. 213–229.
- [30] A. Chadha, S. Kashyap, M. Gupta, and V. Kumar, “License plate recognition system using opencv & pytesseract,” *CSI Journal of Computing*, vol. 3, no. 3, pp. 31–35, 2020.
- [31] R. Chandan and M. Veena, “Vehicle number identification using machine learning & opencv,” *database*, vol. 5, p. 6, 2020.
- [32] T. Damak, O. Kriaa, A. Baccar, M. B. Ayed, and N. Masmoudi, “Automatic number plate recognition system based on deep learning,” *International Journal of Computer and Information Engineering*, vol. 14, no. 3, pp. 86–90, 2020.
- [33] K. T. Islam, R. G. Raj, S. M. Shamsul Islam, *et al.*, “A vision-based machine learning method for barrier access control using vehicle license plate authentication,” *Sensors*, vol. 20, no. 12, p. 3578, 2020.
- [34] N. Awalgaonkar, P. Bartakke, and R. Chaugule, “Automatic license plate recognition system using ssd,” in *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, IEEE, 2021, pp. 394–399.
- [35] V. Gnanaprakash, N. Kanthimathi, and N. Saranya, “Automatic number plate recognition using deep learning,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1084, 2021, p. 012027.
- [36] J. R. Kumar, B. Sujatha, and N. Leelavathi, “Automatic vehicle number plate recognition system using machine learning,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1074, 2021, p. 012012.
- [37] Lubna, N. Mufti, and S. A. A. Shah, “Automatic number plate recognition: A detailed survey of relevant algorithms,” *Sensors*, vol. 21, no. 9, p. 3028, 2021.
- [38] A. Sams and H. H. Shomee, *Bangla lpdh - a*, version v1, Zenodo, Apr. 2021. DOI: 10.5281zenodo.4718238. [Online]. Available: [httpsdoi.org/10.5281zenodo.4718238](https://doi.org/10.5281zenodo.4718238).
- [39] T. Islam and D. Mehedi Hasan Abid, “Automatic vehicle bangla license plate detection and recognition,” in *Smart Data Intelligence: Proceedings of ICSMDI 2022*, Springer, 2022, pp. 523–534.

- [40] B. Chang, H.-F. Tsai, and C.-W. Hsieh, “Accelerating the response of self-driving control by using rapid object detection and steering angle prediction,” *Electronics*, vol. 12, p. 2161, May 2023. DOI: 10.3390/electronics12102161.
- [41] D. Kumar and N. Muhammad, “Object detection in adverse weather for autonomous driving through data merging and yolov8,” 2023.