# License Plate Recognition

by

Mohammed Abrar Ahasan Chowdhury
23141055
Soyelim Al Rozaik
23141056
Mahedi Hasan Shanto
18301185

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

Department of Computer Science and Engineering
Brac University
May 2023

# Declaration

It is hereby declared that

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

**Student's Full Name & Signature:**

[illegible]

Mohammed Abrar Ahasan Chowdhury
23141055

Soyelim Rozaik

Soyelim Al Rozaik
23141056

[illegible]

Mahedi Hasan Shanto
18301185

## Approval

The thesis/project titled “License Plate Recognition” submitted by

1. Mohammed Abrar Ahasan Chowdhury (23141055)
2. Soyelim Al Rozaik (23141056)
3. Mahedi Hasan Shanto (18301185)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 22, 2023.

**Examining Committee:**

Supervisor:
(Member)

Annajiat Alim Rasel

Digitally signed by Annajiat Alim Rasel
DN: cn=Annajiat Alim Rasel, o=Brac University, ou=CSE Department, email=annajiat@bracu.ac.bd, c=BD
Date: 2023.05.22 08:05:23 +06'00'

Mr Annajiat Alim Rasel
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Sifat

Sifat E Jahan
Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

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

# Abstract

In today's ever-growing technological society, Automatic License plate Recognition, ALPR, has many implications for solving traffic-related applications and transportation planning. Identifying cars in pursuit or stolen cars, controlling automatic parking access, registering missing vehicles from last found footage, and in many more hazardous or unpredictable situations, ALPR helps to identify and extract license plate information from surveillance footage. Thus in improving and making ALPR efficient, many techniques have been introduced with algorithms playing an essential part for vehicle surveillance systems, although many challenges are seen in correctly computing and recognizing license plates under different environmental conditions. In this research, we work with different algorithms for understanding Bangladeshi license plates, analyze the algorithms' efficiency in various environmental conditions or unlikely situations, and compare them with our model, which currently is giving 97% accuracy, to find the most suitable for recognizing them.

**Keywords:** License Plate Recognition, Tensorflow, OCR, OpenCV, EasyOCR.

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Bangladesh is a densely populated country with around 170 million people. Being on the road is a big part of daily life. And ensuring safety is a major concern. Before Automatic License plate recognition (ALPR), in various scenarios, detecting the license plate was a hand-held human job, with machines unable to pursue the task. Combatting vehicle crimes was quite hard due to image acquisition from surveillance videos being in low resolution. In Bangladesh, road accidents are one major problem that costs many lives every year. However, these problems are slowly being solved with the emergence of ALPR. Moreover, For many traffic-related applications, vehicle license plate identification and detecting systems are essential, including stolen vehicle recovery, traffic congestion monitoring, airport entrance tracking, speed controlling, and automatic parking lot access management [6]. In Bangladesh, the license plates are pretty different from other countries; they consist of Bengali letters and are ordered in two rows. Moreover, vehicles are increasing drastically, so traffic is quite hard to fight. ALPR technology can cope with the number of vehicle hijacks, kidnappings, or causing irregularities in traffic rules every year. Moreover, many more criminal or federal investigations can be sped up with the help of the emergence of ALPR. Since not many research is done to fine tune ALPR for Bangladeshi License Plates. Our goal is to create a model to be efficient in detecting and recognizing the characters in the license plate. To be able to achieve it, there are quite a few techniques for detecting License plates where usually four modules are used:

1. Pre-Processing
2. License Plate Detection
3. Character Segmentation
4. Character Recognition

## 1.2 Pre-Processing

Pre-Processing involves images acquired to be pre-processed for the license plate for easier detection. The pre-processing module usually involves Grayscale conversion, Median Filtering, and edge enhancement [19][13]. Many other researchers also use Geometric operations or binarization processes during pre-processing. In short, the image is processed for visual perception and has eased in computational processing. Pre-processing helps in removing useless information in the image and focuses on license plate optimization. The pre-processing takes care of a few tasks: removing the background noise, deblur the image, and removing reflection [20]. Several neural network approaches have been used to these pre-processing techniques, notably to generate more precise images and to accelerate image convergence. Once the pre-processing is done, the image is checked to locate the license plate.

## 1.3 License Plate Detection

License Plate Detection is the second module that helps to identify or localize where the license plates are situated in the image, there might be multiple license plates, and all should be localized and detected. A popular real-time object detection algorithm is You Look Only Once (YOLO), which can identify numerous items within a single frame shot. This algorithm is an essential feature of the system since it detects the portion of the plate that contains the license plate numbers [17]. Since the whole image has a vast number of surroundings data, edge detection is used for the localization of the number plate. Different techniques are used to extract the license plate from the image. Morphological technique [8] is used to extract the license plate by detecting the plate size, which is rectangular. Extraction is done on the image where the localized license plate is cropped and sent to the next step, character segmentation [6].

## 1.4 Character Segmentation

This module is where the characters from the received license plate from the previous module are extracted by segmentation process. Firstly, The license plate is converted into a binary picture, after which the letters are separated into segments to extract the characters individually [6]. After the characters are retrieved after segmentation then, they are individually recognized.

## 1.5 Character Recognition

Character Recognition involves various steps like character normalization, feature extraction, and character classification and recognition [12]. In summary, character recognition is the last module of license plate recognition. Character recognition is stated to be the most essential phase throughout identification phase since it determines the model's accuracy and recognition rate. This phase is dedicated to recognizing the information of license plate which are numbers and characters [6] [14]. The normalization process transforms the symbols into a block before the recognition phase. After that extra white spaces are removed from the cropped license plate. The license plate character pictures extracted from the cropped image are identified at this stage.

Each of these modules needs to be highly accurate. Many researchers use different algorithms to find the optimal solution using neural networks, complex image processing, and machine learning processes.

# Chapter 2

# Problem Statement

Researchers encounter several challenges regarding the automatic recognition and detection of license plates. We have looked at a few prominent issues here.

The very first issue is different variations in license plates. Bangladeshi license plates are pretty different from the rest of the world, with the license plate being different in size and Bangla text rather than English. Bangladesh has two license plates under the Bangladesh Road Transport Authority (BRTA) law. One is for private vehicles (white background), and the other is for trading vehicles (green background). Because of the language being different to the rest of the world, where many researches are done for recognition, a few factors must be considered when working with Bangladeshi license plate recognition.

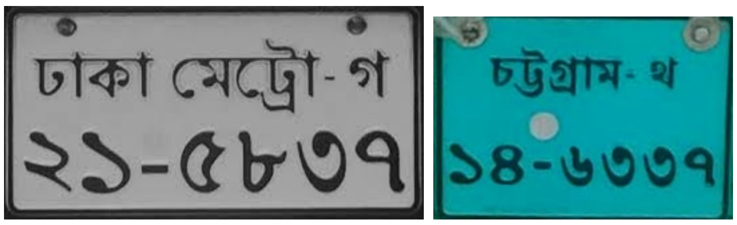


Figure 2.1: Bangladeshi License Plate Variation

Another issue has been the poor quality of vehicle license plates in the captured video frames from the surveillance footage. From factors like weather conditions and the time of the day, the video frame from the footage may vary quite drastically, and retrieving the characters and localizing the license plate from the image with various surrounding conditions may become quite difficult. Surveillance footage being in low resolution, the detection of the license plate and segmenting the texts to recognize them further becomes severely challenging, so different algorithms with the help of Deep Neural Networks have come forward to solve these.

Developing sequential coordination approaches based on image and video processing techniques is expected to solve these problems. Techniques such as computer vision object tracking and segmentation, finding the license plate region, recognizing and pinpointing the number and its color, and so on may be included in this processing sequence. When it comes to dealing with challenges and hurdles, many algorithms are pretty robust. Nonetheless, not much research has been done on Bangladeshi

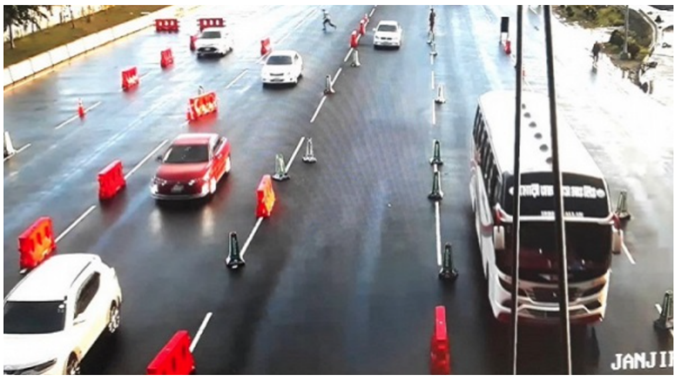


Figure 2.2: Surveillance Footage Quality Example

license plates, especially in conditions where some outside factors obstruct information in the license plates. As a result, these problems need research.

In figure 2.2 it can be seen that from low resolution images from CCTV footage the information on the license plate becomes pretty unclear to identify. As a result, including many other possible causes for license plate to be unclear or hidden, we are positive in training our model feeding as many images as possible for resolving this problem and finding the best algorithm to achieve that.

Therefore, the question that this research is trying to answer is:

Which algorithm and technique in ALPR are the most reliable and efficient under harsh and unpredictable circumstances?

# Chapter 3

# Research Objective

The first question we asked is which way is the most efficient for automatically recognizing the license plates of vehicles. However, this question has been asked several times and implemented throughout the years. Nevertheless, specifically, our objective is to find the best approach and create the most efficient and advanced model that would be able to detect license plates in conditions in which other models face issues in times when the license plate is quite not visible, supposedly obstructed due to environmental disturbance or ambiance. This paper will test the most popular algorithms used in the detection, segmentation, and recognition of license plates in scenarios where the pieces of information in the license plate are not clearly visible. Hence, we will be creating our own model to test and get the most accuracy.

This research would be diving deep into creating a whole annotated dataset containing the labels with license plates that are blurry and hazy similar to real life surveillance footage and try to find the best approach that would be able to identify even with the addition of such anomalies. This research will present a dataset containing 3,82,359 images of computer generated Bangladeshi license plates. The goal is to create such an enormous dataset that will help the model to learn from combinations and patterns of sequence of number through the labels provided so that the numbers are predicted correctly.

# Chapter 4

# Literature Review

The automatic license plate detection system has been a vital part of research for several years. Many researchers throughout the world have attempted in perfecting it using various efficiency and reliability methods. Almost the majority of these suggested methods are inapplicable to Bangladeshi license plates. Since the majority of the methods previously researched are area, language, and license plate specific. Prior studies on Bangladeshi car license plates have also been completed, but relative to other countries, they are pretty negligible.

In [21], the author heavily emphasized eliminating redundant noise from the photos localization process of the vehicle license plate. They employed a technique using a frequency domain mask to eliminate the rainfall drops in front of the license plate, a contrast enhancement technique, a Radon transformation to rectify the tilt, and an image entropy-based technique to filter out the license plate areas. They improved recognition accuracy by 94%.

In [5][11], Gisu Heo et al. propose a new license plate extraction algorithm named the double chance algorithm, Line grouping (LG), and Edge Density (ED). The first technique extracts and classifies line segments based on geometrical criteria. It correctly recognizes a rectangle at the plate's edge. The image is denied unless no rectangle group was created in the step of Line Grouping. However, The rejected picture is given a second opportunity using the Edge Density approach. The second method identifies plate areas with the densest vertical edges. The verification technique is used to evaluate the double chance framework. The character segmentation module is used in the verification process. Through the implementation of the double chance strategy with verification and a real-life database acquired from a surveillance camera, the accuracy of this paper is almost 99.5% which is quite reliable in a real-world application. Similarly, Yule Yuan et al. in [22] introduced a license plate recognition method based on a line density filter and a cascaded license plate classifier.

The authors in [32] suggested a license plate recognition system by merging a super-resolution approach with Alexie. The total accuracy of their suggested technique was 98.2%, but since the training and testing were done on comparatively less amount of data (training = 500, testing = 200), the model tends to be not reliable and suitable for the application in the real world.

Gee-Sean Hsu et al. [16] suggested a detection system based on the blob detection method called Maximally stable extremal regions, MSER for license plates. This detector is used for the character segmentation process. Farhad Faradji et al. [4] identified the Edges first in the input images. The vertical projections of the edge image are then chosen. the environmental ambiance in the backgrounds can be chosen as suitable zones mistakenly due to having vertical edges in structures. To avoid this issue, the researchers designed the compact factor. Comparing the structure the authors used the difference of those unimportant structures in the images having broad vertical edges, whereas a license plate has narrow vertical edges. This characteristic is applied to the compact factor. It is used to determine the brightness of pixels in rows and the local maximum that determines the viable candidate region. Like much other research, the use of Morphological based technique is used to remove plates.

In [12], the author worked to detect license plates for Bangladeshi vehicles using neural networks. They implemented a lossless image segmentation process called chain code. The license plate is extracted using the Sobel filter. The binary image is scanned horizontally and vertically for character segmentation, and connected components are analyzed. Different-sized colored images are used in the JPEG format as a dataset, where 300 images are taken for testing. The images were taken under various lighting and backdrop environments. Experiments demonstrate that the algorithm performs well regarding license plate extraction and character segmentation. This work was developed with MATLAB 7.0. The extraction of license plates yielded an 84% success rate, whereas character recognition yielded an 80% success rate. The extraction of license plates is influenced by dark areas, reflections, and shadows. Actual license plates could not be retrieved accurately due to poor illumination. When two characters are connected, the character segmentation phase fails. Character feature extraction is critical to the performance of the ALPR system.

In [26], the authors suggested license plate identification using the scale adaptive method. The research done in [26] builds on the work of Torralba et al. [7] and uses to calibrate the system for license plates using the foundation of [7]. In [17][19], the authors tested various techniques for detecting license plate detection. In [19], Varsha et al. implemented the histogram-based method which helped in the detection rate to be quite efficient with 97.75% accuracy, while the morphological method is better for recognition rate with 90.75% accuracy. The paper suggested that for efficiency regarding computational time, the morphology method to be better than other methods. They used a template-matching algorithm for the recognition of characters. Characters are matched with the ones available in the database, and the scores are matched. MATLAB2008a was applied to determine computational time. In [28], Sadique et al. review various works related to license plate recognition and show which paper stacks up to be efficient. It is concluded that [27], [25],[26],[23], and,[22] performed to be pretty efficient because of using multiple license plate datasets. Therefore, if an image contained multiple license plates, the system was efficient in detecting them. In [25], Hui Li et al. used three convolutional neural networks (CNN) models to identify text, remove false positives, and recognize text. Artificially created information is applied in [11] to make the model adaptable for many images. The technique in [26] focuses primarily on the scale-adaptive feature, keeping the license plate several distances from the camera. However, it fails to explicitly evaluate the issue of a license plate having a slanted orientation. The authors in [27] used OCR in detecting the License plate in unconstrained scenarios where the information on the license may be blocked or disrupted. The technique in [22] is similarly unsuitable for recognizing slanted license plates as it focuses solely on completion time, managing noisy images, and identifying numerous license plates in one image. Numerous tough datasets are utilized in [25],[ 18], and [23] to train the model to be less noise-sensitive. The algorithms in [25] and [23] enhance the data with image translation, rotation, and affine modification to make the system effective for identifying slanted license plates.

In [15], The author presented an innovative strategy for license plate identification. They applied text-line construction in this approach. Firstly, license plate placement will be selected by the result of text-line construction. Then, the authors used the locally best adaptive thresholding for the images to be converted to black and white for more precise license plate localization. Afterward, the technique of vertical projection is given for character segmentation and extracting the statistic characteristic with the information retrieved. The multilevel classification RBF neural network is used for the result to be accurate. Like Bangladesh, research on the Iranian license plate is also scarce. In [10], The author has proposed a new reliable, real-time solution for morphology and template matching based license plate identification system. The system's primary phase is separating the picture from the digital image captured under various conditions. Firstly, the image is assembled for identification, and after that, a morphology-based algorithm is used to locate the image for the license plate. The shape of the rectangular license plate provides the morphological operator's foundation. Character segmentation is done using a morphological process. It eliminates all minute, interconnected components. After that, the dilation operator separates each character from the others. Additionally, character segmentation is done utilizing partition scanning. The template matching technique is used along with the image correlation method to recognize the characters from the individual segments of characters received from the previous phase.

In [30], applying Connected Component Analysis (CCA) for segmentation, the authors suggested a system where they have also implemented the CNN model for recognizing the characters. This combined technique yielded an accuracy of 96.91%. Their detection method, however, is static and highly unsophisticated. Xiangjian He et al. [9] applied horizontal and vertical projection analysis for the character segmentation process. In order to recognize license plates, Nooruddin et al. [33] suggested combining color characteristics with MinPool and MaxPool features.
Applying the Hough Transform, which is used to detect lines, circles, or curves, Yuangang Zhang et al. [2] created character segmentation. The Hough Transform was used first to determine the plate region's horizontal borders, which aided in segmenting the characters with significant rotation. Combining vertical projection analysis and the previously known plate model character segmentation was achieved. In [18], Amin et al. suggested a method that combines OCR for the Bengali language. They used Edge Detection like used in [33]. The authors also applied Hough Transformation for license plate localization to detect text information similar to

that of [2]. They are not remarkably accurate, and the technique is not consistent although binary thresholding was implemented. Feng Yang et al. [3] devised a region-growing method for character segmentation. For segmentation, in [1], the authors applied connected component image processing.

In [24], the authors applied YOLOv3 and then a CNN model called ResNet-20 with around 1500 pictures in the dataset for the localization model. They have also included 6400 individual character images for the recognition model. Their reported accuracy was 92.7%. However, because their algorithm was solely for the Dhaka Area. As a result, it cannot be applied to other cities.

In [29], the YOLOv3 model was implemented for locating and detecting the license plate information by the authors. They used a limited dataset that only included 1050 pictures of private cars. They assert a 99.5% accuracy rate. Since trading vehicle license plates are not included in their dataset, their argument is utterly refuted. Additionally, they assessed accuracy using the license plate as a complete, binary form.

In [31], the authors have worked to search for the best technique for automatic license plate detection. The authors developed the most effective solution to this issue by combining seven distinct strategies in 2 phases. The most intriguing outcome thus far came from applying the YOLOv4 model to both the text recognition and license plate localization stages. This system has improved its text recognition and license plate localization accuracy using the YOLOv4 model by 99.37% and 96.31%, respectively. Although, the model is primarily trained on photos taken in bright conditions for the license plate localization step. Thus, this technology might not work as planned in nighttime conditions.

In [33], The authors suggested using color histograms in conjunction with the MinPool and MaxPool functions to locate and identify license plates. To study how various color spaces affected the recognition process, the detection algorithm was tested in a variety of color spaces. Based on numerous criteria, the suggested and constructed system obtained elevated amounts of accuracy in the identification phase and is extremely efficient.

# Chapter 5

# Dataset

The most significant contribution of this research would be the dataset. Currently, the number of datasets available for Bangladeshi License plates is scarce and for training a model to its full potential a dataset comprising of good amount of training and testing data is needed to get better accuracy. Generally we see images of license plate taken is captured of a vehicle from the front using mobile phone, which is highly unrealistic when compared to real life problem.

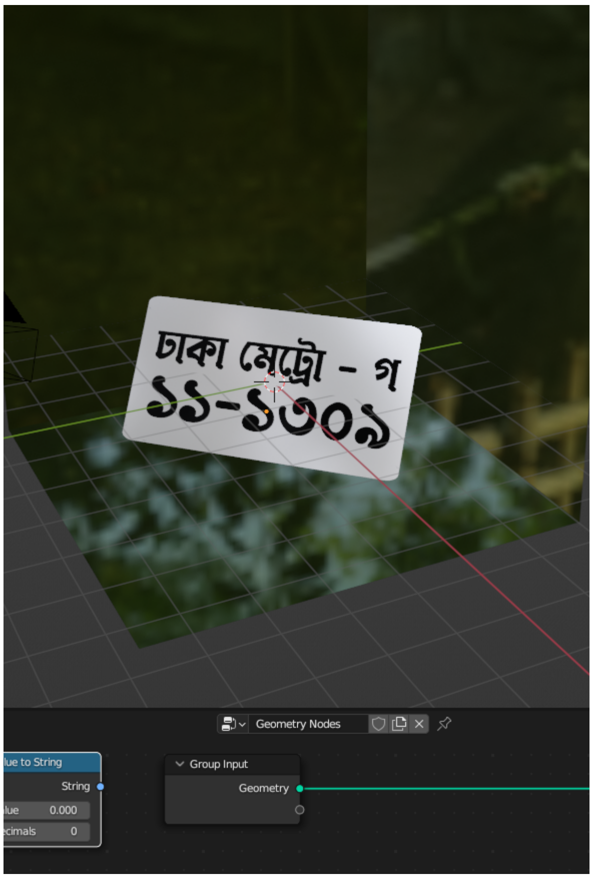


Figure 5.1: Using blender in recreating license plate

This research used blender to create a low-resolution model of the license plate. The idea of creating low-resolution and quite pixelated is due to real-life scenarios where image captured from surveillance footage causes a lot of quality degradation.

This research uses python scripting to generate 382,359 license plates for Dhaka Metropolitan starting from 01-0001 to 39-2358. The intention is to create a diverse dataset of blurry images for the model to train on.

The dataset consists of its own annotation files which contains the label and bounding box of the region of interest, which is the bottom row of the license plate, or the registered number for that vehicle. The annotations are used while training.

This dataset consists of a total of 382359 jpg images excluding their annotations. In total, our model has been trained with 267651 training images and tested on 114708 total images. The most important part of this dataset for making it relevant is its size and also the labels in the .xml file for the annotations which helps in the training.

Table 5.1: The Dataset features

| **Sets** | **Features** | **Sizes** | **Shapes** |
|---|---|---|---|
| TRAIN | Images | 267,651 | (250, 150, 1) |
| | Labels | 267,651 | (6, 10) |
| | ROI | 267,651 | (1, 4) |
| TEST | Images | 114,708 | (250, 150, 1) |
| | Labels | 114,708 | (6, 10) |
| | ROI | 114,708 | (1, 4) |

When generating each image file the annotations are also generated for each of the images using the blender python script.

## 5.1 Data Pre-Processing

As previously mentioned, the dataset that this research is contributing consists of blurry images, which are an example of real world situation. But before feeding the images for training. The images are needed to be pre-processed.
There are various problems when dealing with this kind of dataset, due to immense pixelation, the numbers cannot be detected easily. That is the reason before training and testing, the images in the dataset are required to be pre-processed. The pre-processing steps we used for every image are described below:

1. Apply bilateral filtering using `cv2.bilateralFilter` to remove noise while preserving edges.
2. Apply unsharp masking using `cv2.GaussianBlur` and `cv2.addWeighted` to enhance edges.
3. Convert the image to grayscale using `cv2.cvtColor`.
4. Apply adaptive histogram equalization to increase contrast in darker regions using `cv2.createCLAHE` and `clahe.apply`.
5. Apply a sharpening filter using `cv2.filter2D`.

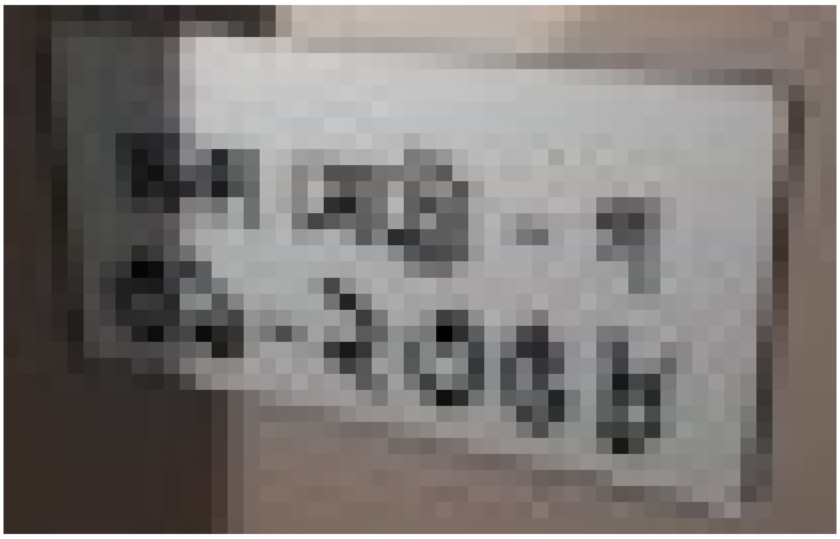


Figure 5.2: Image Before Pre-Processing

## 5.2 Explanation of Image Preprocessing Steps

### 5.2.1 Bilateral Filtering

Bilateral filtering is a non-linear filter that aims to reduce noise while preserving the edges in an image. It considers both the spatial distance and pixel intensity differences when applying the filter. The cv2.bilateralFilter function takes the input image and applies the bilateral filter with specified parameters, such as the filter size, color sigma, and space sigma. By using bilateral filtering, the code reduces noise in the image while maintaining the sharpness of the edges.

### 5.2.2 Unsharp Masking

Unsharp masking is an image sharpening technique that enhances the edges and details in an image. It involves creating a blurred version of the original image and then subtracting this blurred image from the original to obtain the high-frequency details. The cv2.GaussianBlur function applies Gaussian blurring to the image, effectively creating the blurred version. The cv2.addWeighted function combines the original image with the negative of the blurred image to obtain the enhanced edges and details. By applying unsharp masking, the code enhances the edges and details in the image, making them more prominent.

### 5.2.3 Conversion to Grayscale

Converting the image to grayscale simplifies the subsequent processing steps and reduces the computational complexity. Grayscale images have a single channel representing the intensity or brightness of each pixel. By converting the image to grayscale, the code reduces the color information while retaining the essential grayscale information required for further processing.

### 5.2.4 Adaptive Histogram Equalization

Adaptive histogram equalization is an image enhancement technique that improves the contrast in images, particularly in darker or low-contrast regions. It divides the image into small regions called tiles and applies histogram equalization separately to each tile. The cv2.createCLAHE function creates a CLAHE (Contrast Limited Adaptive Histogram Equalization) object with specified parameters, such as the clip limit and tile grid size. The clahe.apply method applies the adaptive histogram equalization to the grayscale image using the created CLAHE object. By applying adaptive histogram equalization, the code enhances the contrast in the darker regions of the image, making them more visually distinguishable.

### 5.2.5 Sharpening Filter

A sharpening filter enhances the edges and details in an image, making them more pronounced. The code applies a specific sharpening filter known as a kernel using `cv2.filter2D`. The kernel used in this case is a 3x3 matrix that enhances the edges by subtracting the surrounding pixel values from the central pixel value and then multiplying by 9. By applying the sharpening filter, the code further enhances the edges and details in the image, making them more noticeable.
These steps in the image pre-processing pipeline help improve the quality, clarity, and contrast of the images, making them more suitable for subsequent analysis or recognition tasks. Each step has a specific purpose and contributes to enhancing different aspects of the image, such as reducing noise, enhancing edges, improving contrast, and increasing the overall sharpness.
These steps in the image preprocessing pipeline help improve the quality, clarity, and contrast of the images, making them more suitable for subsequent analysis or recognition tasks. Each step has a specific purpose and contributes to enhancing different aspects of the image, such as reducing noise, enhancing edges, improving contrast, and increasing the overall sharpness. As a result, combing every one of these pre-processing step, we get our final image like this:

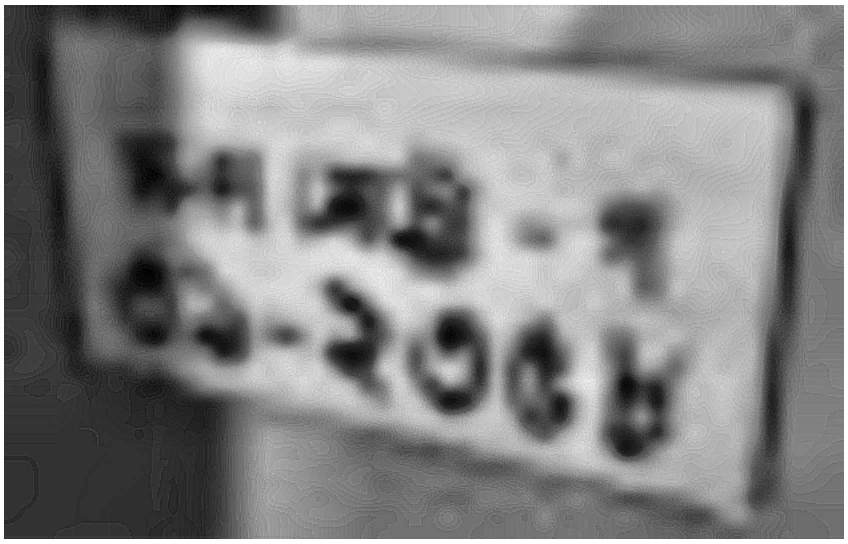


Figure 5.3: Image after Pre-Processing

As a result, now the dataset consist of better quality images which are ready to be trained, Moreover, for the model to not be biased over one single type of image, we are using two orientation of the image initially. Which is images captured from a different angle so that while detecting the characters, the model can learn from characters from different orientation. This research also implements data augmentation, which will be discussed in the models section.

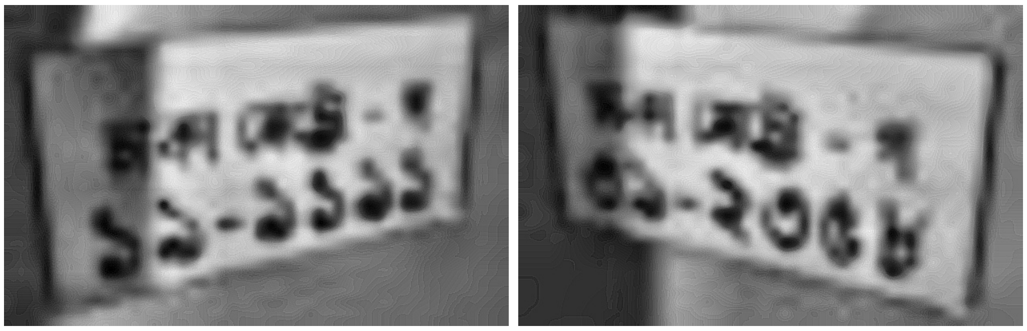


Figure 5.4: Data Variation

After the pre-processing the data are split into 4 folders; Train images, Train annotations, Test images and test annotations. For further computation it was necessary to split the data early on since the dataset is quite huge. The train-test split was done by 0.3, keeping 70% in train set and 30% in test set.

# Chapter 6

# Model

After the collection of datasets, the pre-processed images are used as input for the model along with their bounding box region and label which afterward are trained and tested on. Usually, in terms of the quality of the source, it should be taken with a pinch of salt, considering the fact that low-quality cameras are used for surveillance. That detection for the license plates is not that much to worry about, whereas the character recognition part becomes complicated.

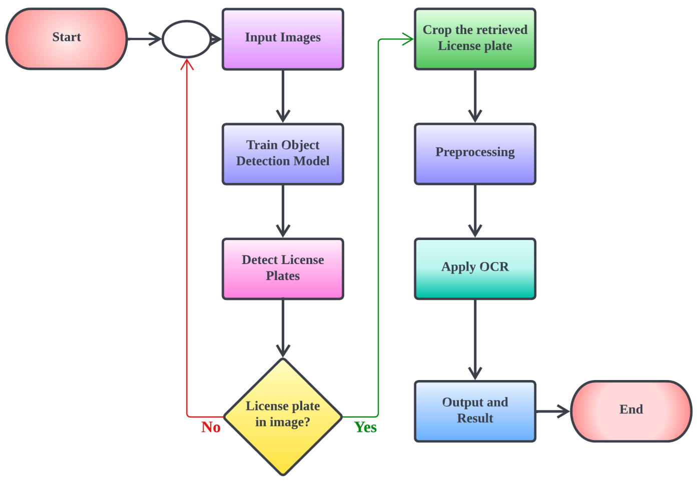


Figure 6.1: The flow chart of our proposed model for ALPR

The following steps have been followed in order to create our model:

1. Taking input from the datasets.
2. Training Object Detection Model on the dataset
3. Detecting or Localizing the license plate from the image
4. Processing the data according to its accuracy, quality, and sharpness.
5. Retrieving the text using OCR
6. Generate formatted output

This research used TensorFlow for object detection and training the model since it is the most reliable out of all. We are training our Bangladeshi license plate with their annotations, and on average while testing 100% of the time the license plate region was easily identified as the license plate from the image having no issues. However, this research focuses on the Bengali character recognition more than that of license plate localization. Tensorflow itself can localize license plates easily, whichever type of license play it may be. But for recognizing Bengali characters from a license plate is the main challenge.

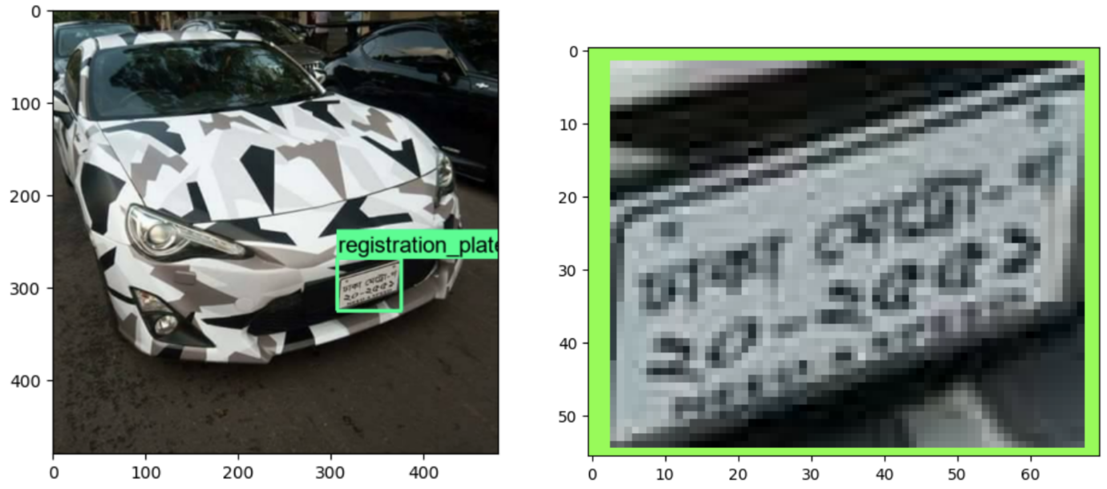


Figure 6.2: Object Detection and Cropping the Region

Pre-processing of the image has been done after cropping since we saw many examples of them becoming pixelated after cropping. Since in our model, we will be cropping to get the license plate region, we apply pre-processing after the cropping as implementing pre-processing after cropping the image would result in delivering a clearer output of the region. This research used different pre-processing algorithms and adjusted them according to our needs. As a result, the OCR has become considerably better than before. We used various OpenCV which is a library of programming functions for the computer vision for pre-processing. Moreover, the input images are resized to 250x150 pixels for faster computation.

## 6.1 Model Architecture

This research did not use any transfer learning, whilst creating a model from scratch to be able to recognize Bangladeshi Characters. The model uses 21 layers. We also worked with various pretrained models like ResNet50 and InceptionV3 but for more leverage on working with our dataset, for this research we intended on working using our own model. The layers in our model is described below:

1. **Input Layers:**

    - `input images`: This layer represents the input images with shape $(250, 150, 1)$. The image size is chosen to be resized to 250x150 pixels with a single channel (grayscale) because it captures enough details while keeping the computational complexity manageable.

- `input labels`: This layer represents the input labels with shape $(6, 10)$. From the annotations folder, the labels are extracted from the ¡text¿ region of the xml file. Since there are 6 digits in the license plate, each digit can have 10 possible classes (0-9). The labels are converted to their one-hot encoded representation is used to represent each digit's class.
- `input Region of interest`: This layer represents the input bounding boxes with shape $(1, 4)$. For every image, there is a bounding box region which is available in the xml file as well and this bounding box is used to determine where the label exists in the training image. Here, a single bounding box is used to specify the region of interest in the image where the license plate is located. The bounding box coordinates consist of four values (x_min, y_min, x_max, y_max).

2. **Preprocessing Layers for Images:**

   - `Rescaling`: This layer scales the pixel values of the input images between 0 and 1 by dividing them by 255. This rescaling is performed to normalize the pixel values and bring them into a consistent range, which can improve model convergence and performance.
   - `AveragePooling2D`: This layer performs average pooling with a pool size of $(2, 2)$. The purpose of this layer is to downsample the images and reduce their spatial dimensions. By taking the average value within each 2x2 patch, it captures the most important features and reduces the computational complexity of subsequent layers.

3. **Convolutional Layers for Images:**

   - `Conv2D`: This layer applies a 2D convolution operation with 32 filters of size $(3, 3)$ and ReLU activation function. Convolutional layers are commonly used in image processing tasks to extract relevant features from the input images. The choice of 32 filters helps the model learn different visual patterns and textures at various spatial scales.
   - `MaxPooling2D`: This layer performs max pooling with a pool size of $(2, 2)$. Max pooling is used to downsample the feature maps obtained from the convolutional layers. By selecting the maximum value within each 2x2 patch, it retains the most prominent features while reducing the spatial dimensions.

4. **Fully Connected Layers for Labels:**

   - `Flatten`: This layer flattens the input labels into a 1D vector. It transforms the structured label representation into a flat feature vector that can be connected to subsequent layers.
   - `Dense`: This layer is a fully connected layer with 64 units and relu activation function. It learns a representation of the label information by applying matrix multiplication to the flattened labels and learning the weights and biases associated with each unit.

5. **Concatenation of Outputs:**

    - The outputs from the convolutional layers and the fully connected layer for labels are concatenated using the `concatenate` layer. This step combines the information from both branches of the model, leveraging the features extracted from the images and the learned label representation.

6. **Additional Layers for Bounding Box:**

    - `Flatten`: This layer flattens the input bounding box coordinates into a 1D vector. Similar to the flatten layer for labels, it transforms the structured bounding box representation into a flat feature vector for further processing.
    - `Dense`: This layer is a fully connected layer with 64 units and ReLU activation function. It learns a representation of the bounding box information by applying matrix multiplication to the flattened bounding box coordinates and learning the associated weights and biases.

7. **Concatenation of Bounding Box Output:**

    - The output from the previous dense layer for bounding box and the concatenated output from the previous step are concatenated using the `concatenate` layer. This step combines the bounding box information with the concatenated output, allowing the model to incorporate spatial information about the location of the license plate.

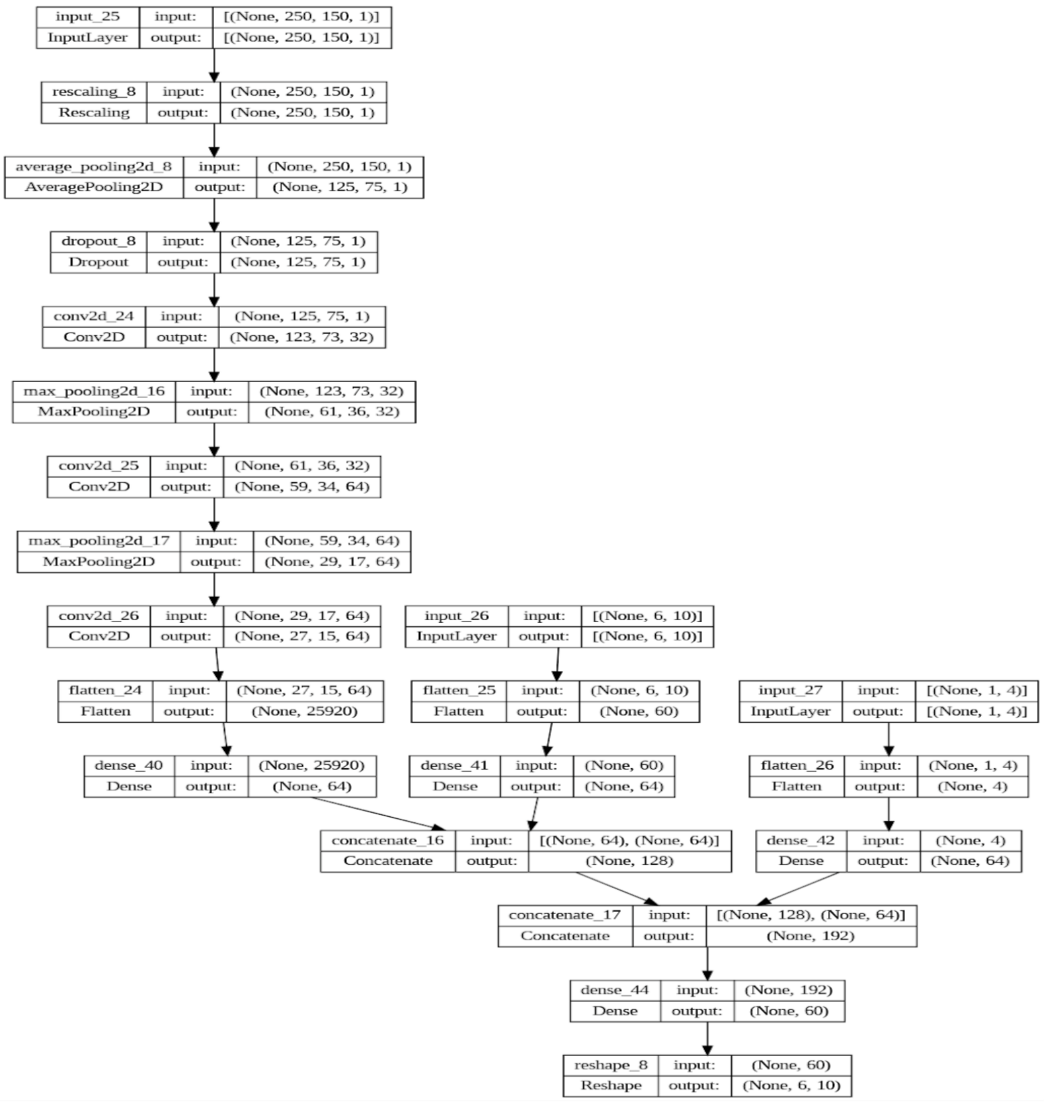
8. **Output Layer:**

    - `Dense`: This layer is the final output layer with 10 units and softmax activation function. It produces the predicted probabilities for each digit class (0-9) in the license plate. The output shape is '(6, 10)', corresponding to the 6 digits and 10 possible classes for each digit.
    - `Reshape`: This layer reshapes the output from (60) to $(6, 10)$. by applying a dense layer with 60 units and softmax activation. The purpose of this reshaping is to ensure that the output has the correct shape to match the label representation.

9. **Model Compilation:**

    - The model is compiled using the `adam` optimizer, which is an adaptive learning rate optimization algorithm. It adjusts the learning rate based on the gradients of the model parameters, enabling efficient training.
    - The loss function is set to `categorical_crossentropy`, which is suitable for multi-class classification tasks with one-hot encoded labels. It measures the dissimilarity between the predicted probabilities and the true labels.
    - The model is evaluated based on the accuracy metric, which calculates the percentage of correctly classified samples.

Convolutional and fully linked layers are combined in the model architecture to capture both picture features and label information. The model combine both spatial and label-based information thanks to concatenation. The input images are enhanced and normalized in the preprocessing layers, which helps the model train and produce precise predictions.



input_25 InputLayer input: [(None, 250, 150, 1)] output: [(None, 250, 150, 1)]
rescaling_8 Rescaling input: (None, 250, 150, 1) output: (None, 250, 150, 1)
average_pooling2d_8 AveragePooling2D input: (None, 250, 150, 1) output: (None, 125, 75, 1)
dropout_8 Dropout input: (None, 125, 75, 1) output: (None, 125, 75, 1)
conv2d_24 Conv2D input: (None, 125, 75, 1) output: (None, 123, 73, 32)
max_pooling2d_16 MaxPooling2D input: (None, 123, 73, 32) output: (None, 61, 36, 32)
conv2d_25 Conv2D input: (None, 61, 36, 32) output: (None, 59, 34, 64)
max_pooling2d_17 MaxPooling2D input: (None, 59, 34, 64) output: (None, 29, 17, 64)
conv2d_26 Conv2D input: (None, 29, 17, 64) output: (None, 27, 15, 64)
input_26 InputLayer input: [(None, 6, 10)] output: [(None, 6, 10)]
flatten_24 Flatten input: (None, 27, 15, 64) output: (None, 25920)
flatten_25 Flatten input: (None, 6, 10) output: (None, 60)
input_27 InputLayer input: [(None, 1, 4)] output: [(None, 1, 4)]
dense_40 Dense input: (None, 25920) output: (None, 64)
dense_41 Dense input: (None, 60) output: (None, 64)
flatten_26 Flatten input: (None, 1, 4) output: (None, 4)
concatenate_16 Concatenate input: [(None, 64), (None, 64)] output: (None, 128)
dense_42 Dense input: (None, 4) output: (None, 64)
concatenate_17 Concatenate input: [(None, 128), (None, 64)] output: (None, 192)
dense_44 Dense input: (None, 192) output: (None, 60)
reshape_8 Reshape input: (None, 60) output: (None, 6, 10)

Figure 6.3: Model Architecture

## 6.2 Training and validation of the Model

### 6.2.1 Train and Validation Split

The model training was a big challenge to deal with since the dataset is comprised of 267,651 images. As a result this research uses batches to train all the images. The batch size is set to 256. We used number of folds for cross-validation: num_folds = 5, which that we will be performing 5-fold cross-validation, dividing the dataset into 5 subsets for training and validation. It involves splitting the data into multiple subsets (folds) and training and evaluating the model on different combinations of these subsets. learning_rate_schedule(epoch) is a function that takes the epoch number as input and returns the learning rate for that epoch. The maximum number of epochs is set to 50, which clarifies that the model will be trained for up to 50 epochs. The learning rate schedule is used to adjust the learning rate during training. In this example, it starts with a learning rate of 0.001 for the first 10 epochs, then reduces it to 0.0001 for the next 10 epochs, and further reduces it to 0.00001 for the remaining epochs.

$$kf = KFold(n\_splits = num\_folds) \tag{6.1}$$

This line creates a KFold object with the specified number of folds. The loop that follows will iterate over each fold, splitting the dataset into training and validation sets accordingly. For the data to be more random we used

$$shuffled\_indices = np.random.permutation(len(train_images)) \tag{6.2}$$

Which generates a random permutation of indices for the training images. The purpose of shuffling is to ensure randomness in the training and validation sets for each fold. The x_train and y_train consists of the images and labels for the training set while the validation counterpart consists for the validation For steps per epoch we used

$$steps\_per\_epoch\_train = len(x\_train)//batch\_size \tag{6.3}$$

$$steps\_per\_epoch\_val = len(x\_val)//batch\_size \tag{6.4}$$

To calculate the number of steps that will be taken per epoch during training and validation. It is determined by dividing the total number of training/validation samples by the batch size. The training and validation datas are not directly fit in the model, but data augmentation is used to augment the data for uniqueness.

### 6.2.2 Data Augmentation

The training images are preprocessed and enhanced with data using the ImageDataGenerator class from Keras. It offers numerous possibilities for enhancing and modifying the photos, which aids in the generalization and performance of the model. Several data augmentation parameters are set to introduce variations in the training images, making the model more robust to different patterns and viewpoints. These parameters include:

**Training Data Generator:**

- `Rescaling`: Rescales the pixel values of the images by dividing them by 255, bringing them into the range of 0 to 1.
- `width_shift_range and height_shift_range`: Randomly shifts the images horizontally and vertically by a fraction of their total width and height, respectively (here, 0.1).
- `rotation_range`: Randomly rotates the images by a specified angle (here, 10 degrees).

- `sheer_range`: Applies random shearing transformations to the images within a certain range (here, 0.2).
- `zoom_range`: Randomly zooms into the images by a factor within a specified range (here, 0.2).
- `horizontal_flip`: Randomly flips the images horizontally.
- `vertical_flip`: Does not flip the images vertically.
- `fill_mode`: The strategy to fill in newly created pixels during augmentation. Here, 'reflect' is used, which reflects the image content at the boundaries.

**Passing Training data:**

- `train_image_dir`: The directory path containing the training images.
- `target_size`: The dimensions to which the images are resized during loading (here, (250, 150)).
- `color_mode`: The color mode of the loaded images. Here, 'grayscale' is used, indicating that the images are converted to grayscale.
- `class_mode`: The type of labels to generate for the images. Here, 'categorical' is used, indicating that the labels are one-hot encoded categorical values.

For resource limitations, not all images are fed with all the 8 types of data augmented changes, while every image is selected to randomly choose any 2 of the data augmentation process and thus can help to create a vast diverse dataset which will help to reduce the lack of generalization.

These data augmentation are done in a way keeping in mind that the bounding box region is updated for every single augmentation. The combination of the ImageDataGenerator and flow from directory allows for efficient loading, augmentation, and preprocessing of training images on-the-fly. The data generator provides a continuous stream of augmented images and their corresponding labels, which is then used for training our deep learning model.

### 6.2.3 Model Fitting

Before the model fitting there are few parameters this research paper used to ensure that, the model is getting the best accuracy with the minimum loss. We defined various callback functions that can be executed at various stages during training. Here, three callbacks are defined:

**Defining callbacks:**

- `ModelCheckpoint`: It saves the best model based on the validation loss during training.
- `EarlyStopping`: It stops the training if the validation loss does not improve for a certain number of epochs.

- `LearningRateScheduler`: It adjusts the learning rate based on the provided schedule.

Using these values the model is then trained keeping two lists, val losses and val accuracies. and After all the folds are complete the model gives us the best accuracies and minimum losses for each of the folds. Then the fold with the best output is saved as the best model.h5. The model is trained with the one hot encoded labels with the images and their corresponding bounding box. The desired output we intend are printing the Bengali characters in serial, the outputs are first generated as binary encoded which are decoded to English or Bengali depending on the preference.

# Chapter 7

# Result

## 7.1 Train and Validation Accuracy

After training using 267,651 images of license plates in multiple batches using 50 epochs as maximum, the model is giving the following result:

Table 7.1: Train and Validation Performance

| **Metrics** | **Train** | **Validation** |
|---|---|---|
| Accuracy | 96.73% | 94.70% |
| Loss | 0.1715 | 0.2010 |

We can see that the model is performing well by giving around 95% training and validation accuracy with loss being almost around 1.3 ranging from as low as 0.8. These accuracy values indicate how well the model is performing on the training and validation datasets, respectively. A higher accuracy indicates better performance, as the model is making more correct predictions. To ensure that the model is not overfitting, we used data augmentation to diversify the input train images.

## 7.2 Test Accuracy

After training, we load and save the best model and then the prediction step began, where the unprocessed license plates are fed to the prediction. Due to test set containing 114,708 images, the predictions were done in batches and for all images two lists had been created. One list comprises of the ground truth label which is retrieved from the annotations file and the another list contains the predicted labels.

$$accuracy = np.mean(ground\_truth\_labels == predicted\_labels) * 100 \tag{7.1}$$

As a result, we get the testing accuracy which checks the ground truth labels for its corresponding predicted labels and give us the following accuracy when tested on 10 batches with each batch having the size of 11,470.
Here we can see on average the model is about to predict with a score of 96.14%. The model successfully detect the license plate and retrieves the license plate number sequence.

Table 7.2: Batch-wise Performance

| Batch | Accuracy |
|---|---|
| 1 | 95.97% |
| 2 | 96.19% |
| 3 | 95.91% |
| 4 | 96.10% |
| 5 | 96.44% |
| 6 | 96.49% |
| 7 | 96.28% |
| 8 | 95.94% |
| 9 | 96.01% |
| 10 | 96.09% |

Finally to evaluate more in depth we check the precision and recall and the F1 score for our model and we saw that they are performing well. The performance on this model is not only tested on our test set generated by blender but also a dataset taken from kaggle which had around 2000 images to verify if our model is overfitting or not.

Table 7.3: Performance Metrics

| Metric | Value |
|---|---|
| Precision | 0.97 |
| Recall | 0.96 |
| F1-score | 0.93 |

## 7.3 Evaluation

After the training and evaluation, we have found the following results

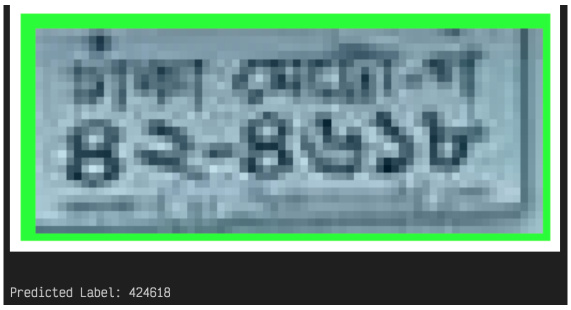
**Train Accuracy**: 0.9673
**Validation Accuracy**: 0.94.70
**Test Accuracy**: 0.9614
**Precision**: 0.97
**Recall**: 0.96
**F1-score**: 0.93

These evaluation metrics provide insights into the model's performance. The train accuracy indicates that the model correctly classifies 96% of the samples in the training set. The validation accuracy of 95% suggests that the model generalizes well to unseen data.The precision score of 0.97 signifies that when the model predicts a positive class, it is accurate 97% of the time, indicating a low rate of false positives. The recall score of 0.96 implies that the model identifies 96% of the positive samples correctly, indicating a low rate of false negatives.The F1-score of 0.93 combines precision and recall into a single metric, providing a balanced measure of the model's performance. It indicates that the model achieves a good trade-off between precision and recall, considering both false positives and false negatives. Overall, these evaluation metrics demonstrate that the model performs well with high accuracy, balanced precision and recall, and a good trade-off between these measures.

This research also used Easy OCR and Tesseract to see if in these harsh blurry conditions if the characters in the license plates are recognized successfully.

### 7.3.1 OUR MODEL

Our model has proven to be reliable on images taken from multiple sources. Predicting on the test set has given us the accuracy of 96.14%. But we also tested on different random blurry license plates to check the accuracy.

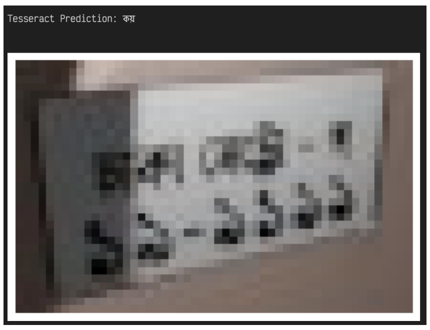


```
Predicted Label: 424618
```

Figure 7.1: Our Model Test Accuracy

### 7.3.2 TESSERACT

Tesseract is a well-known optical character recognition (OCR) engine that has become well-known for its capacity to transform pictures with printed or handwritten text into machine-readable text. It also supports Bengali Language. But when we tested Tesseract on our license plate to recognize the characters out of them, it was unable to recognize them.

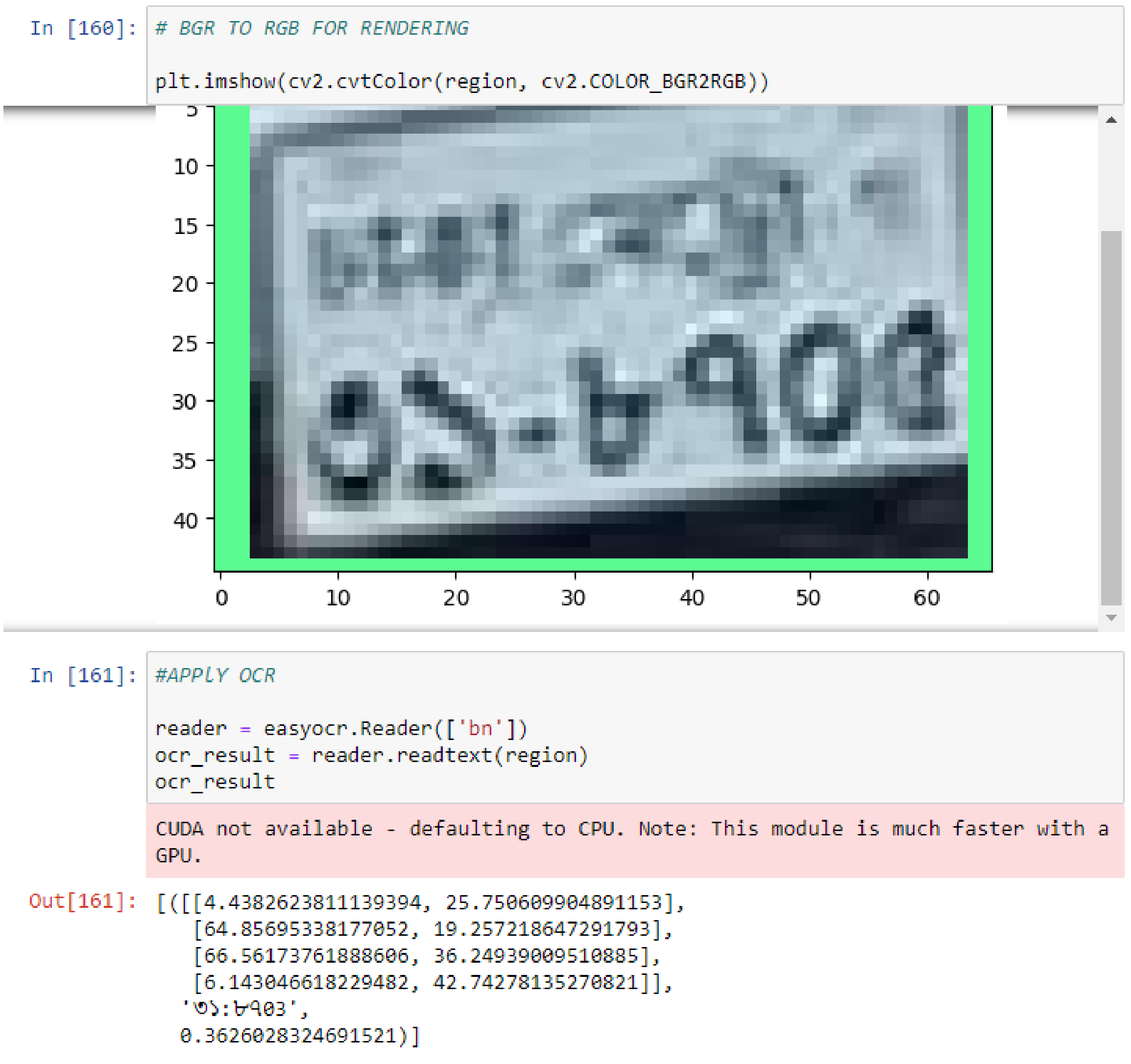


```
Tesseract Prediction: কয়
```

Figure 7.2: Tesseract Test Accuracy

### 7.3.3 EASYOCR

The process of extracting text from photos is made simpler by the optical character recognition (OCR) library EasyOCR, which is accessible and user-friendly. Users that want to integrate OCR capabilities into their apps or workflows will find it to be simple and intuitive to utilize. EasyOCR provides a user-friendly interface while still producing precise and trustworthy text recognition results. Bengali is one of the many languages supported by EasyOCR. This language flexibility enables users to process text in a variety of scripts and character sets, making it appropriate for use in multilingual contexts and international applications. EasyOcr was quite accurate on predicting the license plate but it struggled in few characters.



```
In [160]: # BGR TO RGB FOR RENDERING

          plt.imshow(cv2.cvtColor(region, cv2.COLOR_BGR2RGB))
```

```
In [161]: #APPLY OCR

          reader = easyocr.Reader(['bn'])
          ocr_result = reader.readtext(region)
          ocr_result

          CUDA not available - defaulting to CPU. Note: This module is much faster with a
          GPU.

Out[161]: [([[4.4382623811139394, 25.750609904891153],
             [64.85695338177052, 19.257218647291793],
             [66.56173761888606, 36.24939009510885],
             [6.143046618229482, 42.74278135270821]],
           '৩১:৮৭০3',
           0.3626028324691521)]
```

Figure 7.3: EasyOCR Test Accuracy

As a result, we can see our model is working better on blurry or hazy images of license plates.

## 7.4 Result Verification

We tested our training accuracies for our model and also ResNet50 on around 50 epochs for 5 folds and found the following results:

Table 7.4: Train Accuracies of our Model vs ResNet50

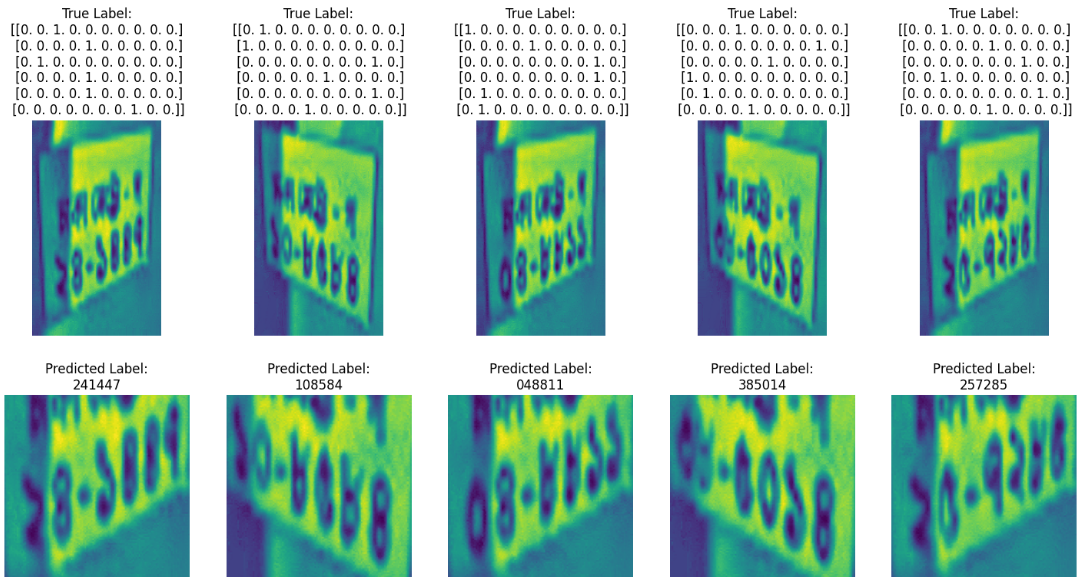
| Model | Train Accuracy |
|---|---|
| Our Model | 96.73% |
| ResNet50 | 98.21% |

For both training we used data augmentation and used Adam as the optimizer and loss function was Categorical Cross Entropy. Moreover we tested using EasyOCR and also Tesseract.

Table 7.5: Input Images and Classification Results

| Model | Input Images | Successful | Unsuccessful |
|---|---|---|---|
| Our Model | 114,708 | 110,323 | 4,385 |
| Tesseract | 114,708 | 11,652 | 103,056 |
| EasyOCR | 114,708 | 96,701 | 18,007 |

We took 5 random samples, displayed their real label from their corresponding annotation files, predicted using our model and then shown the predicted label.



```
True Label:
[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]
Predicted Label:
241447
```

```
True Label:
[[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]
Predicted Label:
108584
```

```
True Label:
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
Predicted Label:
048811
```

```
True Label:
[[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]
Predicted Label:
385014
```

```
True Label:
[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
Predicted Label:
257285
```

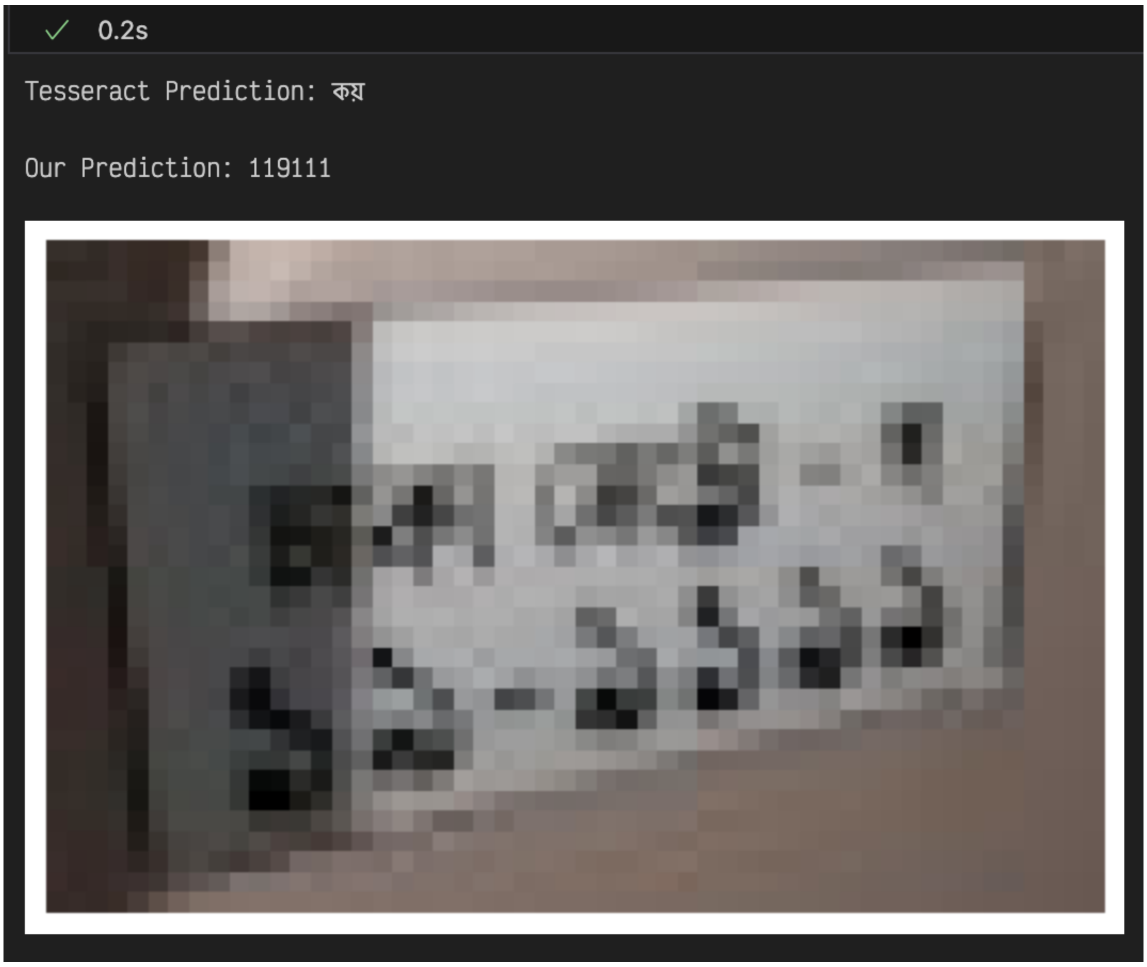
Figure 7.4: Predictions on 5 random Images

Here, the model is successfully predicting the output as what appear on the license plate, the output is in encoded format, which we use decode to translate to English, it can also be translated to Bengali as well.

As a result we see our model to be working well with various test images provided. To consider the images being unbiased we also used data generator to bring uniqueness to the train images, after that our model started perfroming well.

## 7.5 Challenges and Future Works

In the course of our research, we conducted tests on a dataset consisting of 114,708 images. Among these images, we encountered difficulties in accurately detecting a subset of around 4,385 images. As a result, our model achieved an overall accuracy of 96.14%. Going deeper to the cause and checking the images which were unsuccessful and the reason for those we checked the two lists we had, the ground truth label list and the predicted list, and when checking the indexes which are not similar we see majority of the indexes had 1 and 9 together. And if a single label being predicted has only one digit out of place we are considering it to be wrong.



```
✓ 0.2s

Tesseract Prediction: কয়

Our Prediction: 119111
```

Figure 7.5: Problem between 1 and 9

Here, although our model predicted few characters correctly unlike Tesseract, yet there are few more challenges to overcome. To investigate the reasons behind these detection failures, we delved deeper into the unsuccessful cases and scrutinized the corresponding images. Our analysis revealed a specific challenge in distinguishing between the number 9 and the number 1 in Bengali. These two digits exhibit significant visual similarities, making it challeng- ing for our model to correctly identify them.

The identification of the number 9 and 1 is crucial in various applications, and the model's struggle in differentiating them poses a notable obstacle. Addressing this challenge and improving the model's capability to accurately predict these digits in the future will be a valuable area of focus for our research and development efforts.

Also we want to work on the top row which is the metropolitan or city, due to limitations in resources we could not add more images containing various other divisions of Bangladesh or serial number and used license plate containing serial Ga () only of Dhaka Metropolitan. We want to expand our works with multiple metropolitan areas and also work on recognizing each character of a single metropolitan.

We also intend, on making even more robust dataset containing anomalies like dust, rainfall, light splash, moving vehicle swoosh effect and much more on each of the images when generating and target on making a dataset with over 10 combinations of effects on a single license plate without the inclusion of data augmentation.

As of now we are hoping our work will contribute in bringing a free publicly available dataset on which many other researchers can work and make the license plate detection better.

# Chapter 8

# Conclusion

Automatic License Plate Recognition is significant to research as we are always on the road every day of our life. It is used for both searching and security purposes, and in many crucial situations, ALPR can be a lifesaver. Despite extensive research, there is always a wide range of options that have not yet been addressed. However, our tests will be inaccurate and unreliable if our neural network models do not produce valid results. Therefore, we must carefully employ efficient techniques at the appropriate moment and place. Our genuine belief is that our paper will compare various ALPR algorithms on Bangladeshi License plates and contribute a valuable paper with a vast and information driven dataset that other researchers can use. Additionally, we want to advance and broaden the research on both Deep and Convolutional neural networks. The final goal of this research would be to inform the research community with our comprehensive analysis of these approaches as well as to extend the reach of comparative studies between the techniques used in license plate recognition for various purposes and how each of these algorithms and our model functions in recognizing the license plate efficiently in various environmental conditions.

# Bibliography

[1] S.-Z. Wang and H.-J. Lee, "Detection and recognition of license plate characters with different appearances," in *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, vol. 2, 2003, 979–984 vol.2. DOI: 10.1109/ITSC.2003.1252632.

[2] Y. Zhang and C. Zhang, "A new algorithm for character segmentation of license plate," *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*, pp. 106–109, 2003.

[3] F. Yang, Z. Ma, and M. Xie, "A novel approach for license plate character segmentation," Jun. 2006, pp. 1–6. DOI: 10.1109/ICIEA.2006.257234.

[4] F. Faradji, A. Rezaie, and M. Ziaratban, "A morphological-based license plate location," vol. 1, Sep. 2007, pp. I–57, ISBN: 978-1-4244-1437-6. DOI: 10.1109/ICIP.2007.4378890.

[5] G. Heo, M. Kim, I. Jung, D.-R. Lee, and I.-S. Oh, "Extraction of car license plate regions using line grouping and edge density methods," Dec. 2007, pp. 37–42, ISBN: 0-7695-3045-1. DOI: 10.1109/ISITC.2007.79.

[6] S. Ozbay and E. Erçelebi, "Automatic vehicle identification by plate recognition," 2007.

[7] A. Torralba, K. Murphy, and W. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, pp. 854–69, Jun. 2007. DOI: 10.1109/TPAMI.2007.1055.

[8] C.-N. Anagnostopoulos, I. Anagnostopoulos, I. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, pp. 377–391, Oct. 2008. DOI: 10.1109/TITS.2008.922938.

[9] X. He, L. Zheng, Q. Wu, W. Jia, B. Samali, and M. Palaniswami, "Segmentation of characters on car license plates," Oct. 2008, pp. 399–402. DOI: 10.1109/MMSP.2008.4665111.

[10] H. Kasaei, M. Kasaei, and S. Kasaei, "New morphology-based method for robustiranian car plate detection and recognition," *International Journal of Computer Theory and Engineering*, vol. 2, pp. 264–268, Jan. 2010. DOI: 10.7763/IJCTE.2010.V2.150.

[11] P. R. Sanap and S. P. Narote, "License plate recognition system-survey," *AIP Conference Proceedings*, vol. 1324, no. 1, pp. 255–260, 2010. DOI: 10.1063/1.3526208. [Online]. Available: https://doi.org/10.1063/1.3526208.

[12] A. Ghosh, S. Sharma, M. N. Islam, S. Biswas, and S. Akter, “Automatic license plate recognition (alpr) for bangladeshi vehicles,” *Global Journals Inc. (USA)*, vol. 11, pp. 69–73, Dec. 2011.

[13] H. Kolour, “An evaluationof license plate recognition algorithms,” *International Journal of Digital Information and Wireless Communications (IJDIWC)*, vol. 1, pp. 247–253, Jan. 2011.

[14] A. Nagare and S. Bhatia, “License plate character recognition system using neural network,” *International Journal of Computer Applications*, vol. 25, p. 4, Jul. 2011. DOI: 10.5120/3147-4345.

[15] B. Shan, “Vehicle license plate recognition based on text-line construction and multilevel rbf neural network,” *JCP*, vol. 6, pp. 246–253, Feb. 2011. DOI: 10.4304/jcp.6.2.246-253.

[16] G.-S. Hsu, J.-C. Chen, and Y.-Z. Chung, “Application-oriented license plate recognition,” *Vehicular Technology, IEEE Transactions on*, vol. 62, pp. 552–561, Feb. 2013. DOI: 10.1109/TVT.2012.2226218.

[17] D. A. Sinhal, “Comparative study of different techniques for license plate recognition,” *Journal of Advanced Computing and Communication Technologies*, vol. 1, pp. 1–5, Dec. 2013.

[18] M. R. Amin, N. Mohammad, and M. A. N. Bikas, “An automatic number plate recognition of bangladeshi vehicles,” *International Journal of Computer Applications*, vol. 93, pp. 24–27, 2014.

[19] M. V. K. Hadke and P. K. Ajmera, “Comparative study of license plate recognition,” *International journal of engineering research and technology*, vol. 3, 2014.

[20] N. Ibrahim, E. Kasmuri, N. Jalil, M. Adili, S. Salam, and R. Nawawi, “License plate recognition (lpr): A review with experiments for malaysia case study,” *Middle East Journal of Scientific Research*, vol. 14, Jan. 2014. DOI: 10.7321/jscse.v3.n3.15.

[21] S. Azam and M. M. Islam, “Automatic license plate detection in hazardous condition,” *J. Vis. Commun. Image Represent.*, vol. 36, pp. 172–186, 2016.

[22] Y. Yuan, W. Zou, Y. Zhao, X. Wang, X. Hu, and N. Komodakis, “A robust and efficient approach to license plate detection,” *IEEE Transactions on Image Processing*, vol. PP, pp. 1–1, Nov. 2016. DOI: 10.1109/TIP.2016.2631901.

[23] H. Li, P. Wang, and C. Shen, “Toward end-to-end car license plate detection and recognition with deep neural networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, Sep. 2017. DOI: 10.1109/tits.2018.2847291.

[24] S. Abdullah, M. M. Hasan, and S. M. S. Islam, “Yolo-based three-stage network for bangla license plate recognition in dhaka metropolitan city,” *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–6, 2018.

[25] H. Li, P. Wang, M. You, and C. Shen, “Reading car license plates using deep neural networks,” *Image and Vision Computing*, vol. 72, Mar. 2018. DOI: 10.1016/j.imavis.2018.02.002.

[26] M. Molina-Moreno, I. González Díaz, and F. Díaz-de-María, “Efficient scale-adaptive license plate detection system,” *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, pp. 1–13, Aug. 2018. DOI: 10.1109/TITS.2018.2859035.

[27] S. Montazzolli and C. Jung, “License plate detection and recognition in unconstrained scenarios,” Sep. 2018.

[28] M. F. Sadique and S. M. R. Haque, “A comparative study of license plate detection and recognition techniques,” Dec. 2019. DOI: 10.1109/ICCIT48885.2019.9038583.

[29] N. Saif, N. Ahmmed, S. Pasha, *et al.*, “Automatic license plate recognition system for bangla license plates using convolutional neural network,” *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pp. 925–930, 2019.

[30] M. Hossain, A. Suvo, A. Ray, M. Malik, and M. Mridha, “Number plate recognition system for vehicles using machine learning approach,” Feb. 2020.

[31] S. Hossain, M. Z. Hassan, and M. Masba, “Automatic license plate recognition system for bangladeshi vehicles using deep neural network,” Dec. 2021, ISBN: 978-981-16-6635-3. DOI: 10.1007/978-981-16-6636-0_8.

[32] N. Munna, M. Ahsan, M. Based, and J. Haider, “Intelligent system for vehicles number plate detection and recognition using convolutional neural networks,” *Technologies*, vol. 9, p. 9, Jan. 2021. DOI: 10.3390/technologies9010009.

[33] S. Nooruddin, F. A. Sharna, and S. M. M. Ahsan, “A bangladeshi license plate detection system based on extracted color features,” Apr. 2021. DOI: 10.1109/ICCIT51783.2020.9392672.