Four Stage Automatic License Plate Recognition System for Bangla Low-Resolution License Plate images

Rabeya Afrose Tithy	**1847CSE00719**
Mirajul Islam	**1847CSE00724**
Samira Islam	**1847CSE00725**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MANARAT INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

August, 2022

Four Stage Automatic License Plate Recognition System for Bangla Low-Resolution License Plate images

A thesis

Submitted in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Submitted by

Rabeya Afrose Tithy	**1847CSE00719**
Mirajul Islam	**1847CSE00724**
Samira Islam	**1847CSE00725**

Supervised by

Naimul Haque

Lecturer

Department of Computer Science and Engineering
Manarat International University

Dhaka, Bangladesh

August, 2022

DECLARATION

We, hereby, declare that the thesis presented in this report is the outcome of the research performed by us under the supervision of Naimul Haque, Lecturer, Department of Computer Science and Engineering, Manarat International University, Dhaka, Bangladesh. The work was spread over one final year course, CSE400: Thesis, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Rabeya Afrose Tithy
1847CSE00719
Department of Computer Science and Engineering
Manarat International University

Mirajul Islam
1847CSE00724
Department of Computer Science and Engineering
Manarat International University

Samira Islam
1847CSE00725
Department of Computer Science and Engineering
Manarat International University

APPROVAL

This thesis titled, **"Four Stage Automatic License Plate Recognition System for Bangla Low-Resolution License Plate images"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering.

Examination held on: August 20, 2022

Group Members:

Rabeya Afrose Tithy	**1847CSE00719**
Mirajul Islam	**1847CSE00724**
Samira Islam	**1847CSE00725**

Naimul Haque
Lecturer & Supervisor
Department of Computer Science and Engineering
Manarat International University

BOARD OF EXAMINERS

Naimul Haque
Lecturer
Department of Computer Science and Engineering
Manarat International University

Supervisor

Mohammad Rafiqul Islam
Associate Professor & Head
Department of Computer Science and Engineering
Manarat International University

Member (Ex-Official)

Md. Ali Hossain
Assistant Professor
Department of Computer Science and Engineering
Manarat International University

Member

Sohaib Abdullah
Assistant Professor
Department of Computer Science and Engineering
Manarat International University

Member

Zahurul Haque
Lecturer
Department of Computer Science and Engineering
Manarat International University

Member

zahirul islam babor
Lecturer
Department of Computer Science and Engineering
Manarat International University

Member

Dr. Md. Haider Ali
Professor
Department of Computer Science and Engineering
Manarat International University

Member (External)

ACKNOWLEDGEMENT

At first we want to thank The Most Glorified, The Most High Almighty Allah for enabling us to complete the thesis on time and for keeping us in good health which was much-needed for successful completion of this work.

It is an honor for us to thank those who made this thesis work possible. We owe our deepest gratitude to our supervisor, Naimul Haque, Lecturer, Department of Computer Science and Engineering, Manarat International University, whose encouragement, guidance and support from the initial level to the end enabled us to develop an understanding of the subject and helped us a lot to finish our research.

We would like to thank Mohammad Rafiqul Islam, Associate Professor and Head, Department of Computer Science and Engineering, Manarat International Universityand all our teachers and friends for their constant encouragement and help in different processes during data preparation and testing phases.

Dhaka
August, 2022

Rabeya Afrose Tithy

Mirajul Islam

Samira Islam

ABSTRACT

Automatic License Plate Recognition (ALPR) Sys- tem is a crucial task that is used in numerous critical situations. Although there are several ways to accomplish license plate recognition, our method is intended to be effective not just on license plates with high resolution, but even when the license plate (LP) is of very low resolution (LR). Previous works on ALPR systems emphasized achieving recognition benchmarks on LP images that are close to the sensor. However, if the camera is far from the vehicle, the generated detected frame yields an LP image of LR, which is hard for the recognition model to classify. Our method employed Enhanced Super Resolution Generative Adversarial Networks (ESRGANs) to upscale the $[32 \times 24]$ LR image to high resolution of dimension $[256 \times 192]$ with a scale factor ratio of 8. For training purposes, we also collected and trained the models on a dataset of 2211 images. The Peak signal-to-noise ratio (PSNR) for the reconstruction on 200 testing samples is 12.587.The optical character recognition accuracy for the reconstructed and ground truth images are 78respectively.

Keywords: **Computer Vision, Deep Learning, ALPR, Bangla, License Plate, Super-Resolution, Detection, Recognition**

Contents

List of Figures

List of Tables

List of used Abbreviations

1. Adam : Adaptive Moment Estimation
2. ANN : Artificial Neural Network
3. CNNs : Convolutional Neural Networks
4. DNN : Deep Neural Network
5. FC : Fully Connected
6. MaxPool : Max Pooling
7. MTCNN : Multi-task Cascaded Convolutional Networks
8. ReLU : Rectified Linear Units
9. Resnets : Residual neural networks
10. RGB : Red Green Blue
11. VGG : Visual Geometry Group

Chapter 1

Introduction

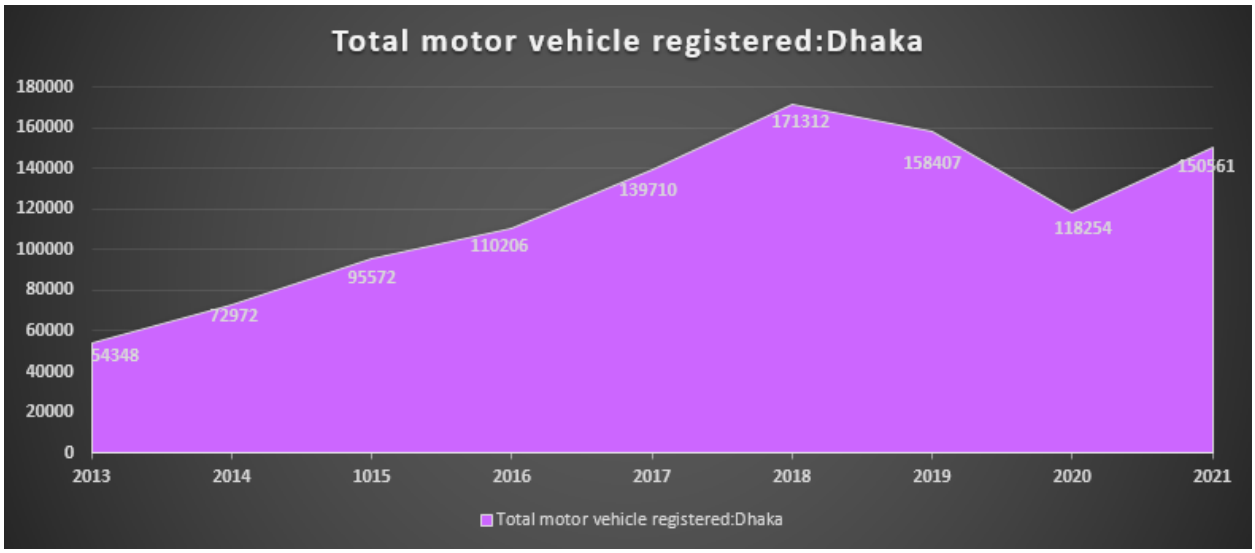
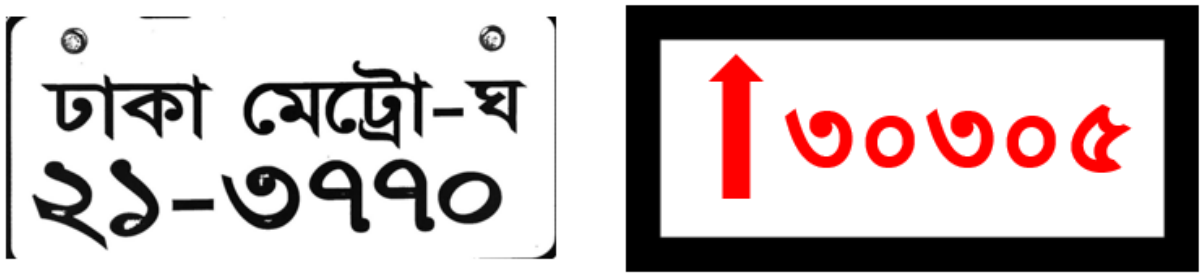


Figure 1.1: . Registrations of vehicle number plate in Dhaka for the past nine years

1.1 Problem Definition

Automatic License Plate Recognition System(ALPRS) is a significant research modality in the field of computer science and Engineering. It has been very efficient approach for the vehicle surveillance that captured the image of a vehicle and automatically increase the number plate resolution then recognizes their license number. ALPRS applied for the purpose like traffic safety enforcement, traffic toll collection, automatic parking system, border control, road traffic monitoring, law enforcement, in security system and so on. This system is very important to all over the world specially in Bangladesh because we are densely populated country and the capital of Bangladesh, Dhaka City is unbelievable traffic jam area. According to BRTA (Bangladesh Road Transport Authority) Dhaka has around 4.5 million registered vehicles as of 2021 which is increasing rapidly every year. The annual report of BRTA total motor vehicle registered graph are shown in Figure 1.1. Crime, traffic jam, violation is most common thing in this city. This could be objective for the increasing road traffic accidents and related deaths. So Automatic License Plate Recognition System(ALPRS) is very important to our country than any other cities in the world.

In Bangladesh, Bangladeshi vehicle number plates are written in Bengali alphabet and Bengali numerals with a fixed two-line text format and color for different types of vehicles. The international vehicle registration code for Bangladesh is BD. The two types of vehicles are used in Bangladesh are civil vehicles and army vehicles shown in Figure 1.2.



a b

Figure 1.2: . Bangladeshi vehicle number plates:(a) civil vehicle number plate, (b) army vehicle number plate.(c) Private Vehicle.

According to BRTA. the first line of the license plate contains the vehicle registration area, if it has been registered in a metropolitan area and the word Metro has been written, followed by a hyphen a letter contains the category of the vehicle shown in figure1.3. The second contains six digits, the first two of which is for vehicle class number and the last four separated from first two by a hyphen constitute unique vehicle registration number. There are total Thirty-two letters have been assigned to indicate the type of the vehicle.

1.2 Overview of Bangladeshi license plate

The BRTA standard format of vehicle number plates in Bangladesh is showing in figure 1.3 as following “city name—class letter of a vehicle and its number—vehicle number”. For example, “DHAKA METRO—GHA 3770”. Here, “Dhaka” represents the city name, and “GHA: represents vehicle class in Bangla alphabets. The second line (number line) contains six digits, where the first 2 digits (21) denote the vehicle class number and the last 4 digits (3770) represent the vehicle registration number in Bangla numerals.There are a total of Thirty-two letters have been assigned to indicate the type of the vehicle. all the classes name of BRTA standard licence plate are mentioned in TABLE 1.1

1.3 Thesis Overview

This research work developed a system that is proficient to recognize vehicle number plates. The system builds with four major steps: **licence plate detection,image pre-processing, ,En-**

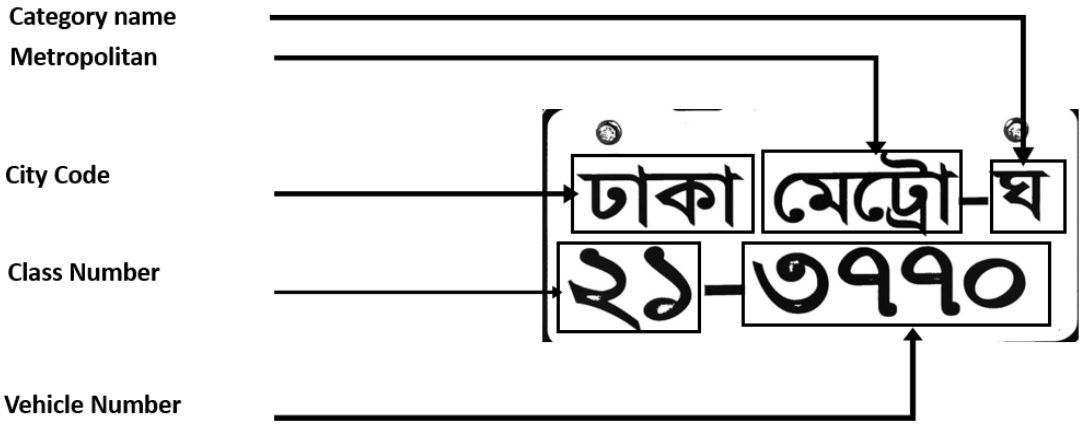


Figure 1.3: BRTA Standard License Plate format

Table 1.1: **Classes name of BRTA standard Bangla license plate**

Format type	**Class name**
Name of the city	Dhaka Metro
Letter	Ka, Kha, Ga, Gha, Ca, Cha, Ja, Jha, Ta, Tha, Da, Dha, Na, Pa, Pha, Ma, THA, E, Ha, La, A, Bha, Sha, U, Sa, Ba
Digit	0, 1, 2, 3, 4, 5, 6, 7,8,9

hanced Super Resolution Generative Adversarial Networks to get a high-resolution image, and recognition of each character by using easyOCR. super resolution is the most important task as it provides the result of the entire number plate analysis. recognition of of blurred license plates is more challenging, and this is overcome by using the super resolution method that transformed the blurred number plate image into a clear image. So, the most challenging part of this research is to clear image by using **ESRGAN** super resolution method.

1.4 Scope of Study

Our research is to use **YOLOV4** features to detect Bangla license plates. In many cases, license plate images have been captured in ideal conditions making it less useful in a busy city like Dhaka. One of the importance for this research is that the dataset containing Bangla license plates representing non-ideal conditions such as existence of different lighting conditions, viewing angles, transparency etc.

In this propose method we represent a real-time system for Bangla license plate detection and recognition system. In that case to get good result to enhance super resolution we used deep learning based super resolution method by using Generative Adversarial Networks**(GAN)**. The system is robust as we have tested it in a dataset containing images that were collected by us from different camera setup and environmental conditions. Our detection system is real-time as we have used the latest object detection algorithm name as **YOLOV4**.

In recognition part we have used **EasyOCR** Model.EasyOCR is an open-source, ready-to-use Optical Character Recognition(OCR) with 80+ supported languages and all popular writing scripts.

1.5 contributions

The Prime contribution of this paper is as follows as:

1) We have collected a dataset containing 2500 images which were taken in many different real-world conditions and from different viewing angles. The dataset will be made publicly available.

2) To the best of our knowledge, we have used YOLOv4 for the first time in the Bangla license plate detection and recognition problem. This technique achieves greatest efficiency with other well utilized models in the literature.

3) This crucial part of our ALPR system is the Super Resolution as we focus more on the low lp images. We have used **Enhanced Super Resolution Generative Adversarial Networks (ESRGAN)** where we used an up-scaling factor of 8 to upscale the low-resolution images. The architecture of the generator of the ESRGAN which consists of several complicated Residual-in-Residual Dense Block (RRDB) structures.

4) we used the frameworksof EasyOCR for the recognition part . The LR license plate is passed through CRNN model for the recognition of the scene text region.

Chapter 2

Literature review and problem outline

===

The authors in [1] proposed a deep learning model for a system that can identify license plates. They used some synthetic, custom-made data in their dataset, but it's challenging to get good results in real time scenario. Although the paper used a Generative Adversarial Network (GAN) to enhance super resolution, they did not accurately describe their results. Furthermore, the researcher employed YOLOV4 model for both the detection and the recognition of LP, which was a poor decision for this kind of works.
Raiyan et al. proposed a license plate recognition method with template matching in 2014 [2]. The paper focused on the character and digit recognition task only. In this paper they used a special feature of Bangla character called "matra" and image processing techniques. The paper showed some test cases but didn't avail to make any general conclusion which makes it image specific. However, in this paper clearly not mentioned about dataset. only used a small number of images to perform the experiment which lacks generalization.

In [3], To extract and recognize Thai automobile license plates, the researchers presented a technique based on Maximally Stable Extremal Regions (MSER) and Back-Propagation Neural Network (BPNN). Lin et al. [4] presented a hierarchical approach for recognizing license plates. They used YOLOv2 to detect vehicles in the first step. They used SVM to recognize license plates after detecting the license plates. Following that, they divided the character and created a CNN network to recognize it. A YOLO-based technique was presented for detecting and recognizing Brazilian license plates in [5]. However, the dataset they utilized isn't particularly difficult, given it simply includes a frontal view of a car. Like [6], the authors in [7] also employed YOLOv2 for vehicle detection exclusively, whereas MSER with BPNN was used for license plate detection and CNN was used for character recognition. Detection before detecting the license plate. The authors in [6] also did this and rationalized this approach by arguing that it reduced false positives on license plate detection. But our approach is significantly different from the above-mentioned approaches in the sense that we dropped vehicle detection step making the process faster without deteriorating performance.

In [8], the authors have preprocessed the raw dataset into the YOLOv3 format for the detection task. They have implemented a CNN model for the segment recognition which achieved 97.5% accuracy which is very well but they used only 2000 raw data. however, the researchers didn't point out approximately how they segmented the unclear blurry license plate because in real-scenario it is impossible to obtain great resolution quality images all the time.

Abdullah, et al [9] proposed a YOLO-based license plate recognition system for Bangladeshi Vehicles. They employed YOLOv3 to detect the license plate from a vehicle image in the first

step of this work. However, this system is simplest for the Dhaka Metropolitan area as they didn't use different metropolitan license plates. Their accuracy level is 92.7% in the digit and character recognition stage. They stated in their paper that they would publish their dataset if it was required, but they did not.

In [10], the authors proposed an end-to-end system for Bangladeshi license plate recognition. However, Their paper lacks particular information about their dataset. As a result, evaluating or comparing their work is extremely difficult. The researchers used YOLOv3 for both detection and recognition of license plates, which is a rather naive choice for this type of work. So, because of their data set was not diverse enough, their model reached a high level of accuracy, but it could not be used in real-life circumstances.

In [4], the researchers developed a system for detecting and recognizing of vehicle number plates using a convolutional neural network (CNN), a deep learning technique. This system comprises of two parts: number plate detection and number plate recognition. In the detection part, a vehicle's image is captured through a digital camera. They have collected the number plate from BRTA. In that case their pictures have been captured in very ideal conditions. However, in order to get good result dataset needs to containing non-ideal conditions such as presence of different lighting conditions, viewing angles, weathers and so on.

Chapter 3

Dataset

===

3.1 Online Dataset

We initially searched for data online to complete our research, but we did not find any useful data. We discovered a small amount of data on some websites, but these data have not yielded satisfactory results in our real-time system so far.in order to collect our dataset we chose to collect data from city street and parking area.

3.2 Real-time dataset

We have collected more than 3000 images of various kinds of vehicles that are frequently seen in Dhaka Metropolitan City.Some sample Images are showing in figure 3.1. Images are snapped with cameras and different mobile phones from various locations around Dhaka.Images of still and moving vehicles were taken at different times of day and night in various weather conditions with varying distances, tilt, and angles between November 2021 and March 2022. The vehicle image dataset consists of almost all types of vehicles present in Bangladesh, excluding army, police, and government vehicles. The Images were in color and used the Joint Photographic Experts Group **(JPEG)**, **(PNG)** and **(JPG)** format as a standard. Our data-set contains 2300 images for training and 200 more for testing.

The data-set has the following features, among others:
• We have gathered license plate images throughout the day, at sunset, and at night in order to guarantee that there is variety in lighting situations. Additionally, we have gathered images from locations with a lot of lighting diversity.
• For the purpose of developing and testing a real-time detection system, our data-set contains multiple license plates in a single image.
• The quality of the captured images may be low due to a variety of factors, and the detection system needs to be smart enough to detect those up as well. Images captured with mobile cameras that are blurry and poor quality are included in our data-set. Several of the images also contain foggy.

Figure 3.1: Sample Images of Data-set

Chapter 4

State of Arts Model

===

4.1 YOLOv4

YOLOv4 is an effective object detection model that improves on YOLOv3 [16] by incorporating many literature based methods and modules. The architecture consists of backbone, neck, and head (dense prediction and sparse prediction). CSPDarknet53 is used as the backbone of this model, which improves the learning capacity of a Convolutional Neural Network (CNN) and acts as a feature extractor. As necessary, they employ path aggregation network with spatial pyramid pooling as neck which composes information from different layers of backbone. Anchor based YOLOv3 acts as head of YOLOv4 architecture. This architecture uses bag of freebies (BoF) and bag of specials (BoS) for both backbone and detector. BoF methods only alter the training strategy or raise the cost of training without increasing the inference cost. BoS consists of plugin modules and post-processing techniques that add a small amount to the inference cost yet enhance accuracy considerably. The object detector model of YOLOv4 is shown in 4.1.

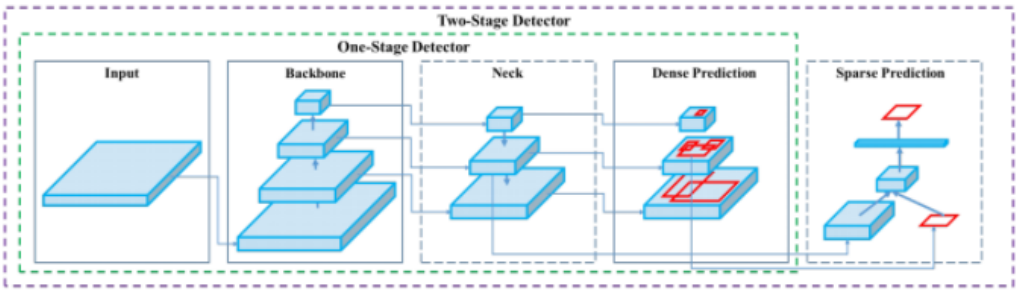


Figure 4.1: YOLOv4 object detector model

As we've seen, there are numerous options when it comes to architecture. So, what precisely are we looking for while selecting architecture? The best balance of input network resolution (input image size), number of convolution layers, number of parameters, and number of output layers is used as the selection criteria (filters). Additionally, we have an additional block for enlarging the receptive field (bag of specials) as well as the ideal approach for transitioning from different backbone levels to various detector levels (Necks). The final architecture presented in the paper is CSPDarkNet53 (Head) => SSP + PANet (Neck) => YOLOv3 (head) Other close competitors was CSPResNext50 and EfficientNet-B3 but CSPDarkNet gave higher FPS than the rest.

4.2 Enhanced Super Resolution Generative Adversarial Networks

Enhanced Super Resolution Generative Adversarial Networks (ESRGAN) [15] is an enhanced version of SRGAN [17]. Through this GAN network, a low-resolution image can be upsampled to a high-resolution image using a configurable up-scaling factor. This GAN model eliminates all batch normalization layers from SRGAN and uses a more complicated Residual-in-Residual Dense Block (RRDB) structure, which decreases computational cost, improves image perceptual quality, and increases image resolution. The network architecture and RRDB structure is shown in Fig. 5. The generator of this model uses different loss functions to improve the quality of image and to remove artifacts. The total loss of the generator is represented by:

$$L_{Generator} = L_{percep} + \lambda L_G^{Ra} + \eta L_c$$

where, L_percep is perceptual loss, $L_G^{Ra}$is adversarial loss, and Lc is content loss. A relativistic discriminator is used in ESRGAN, which replaces the binary standard discriminator of usual GAN. The discriminator loss is defined as:

$$L_D^{Ra} = -E_{x_t}[\log(D_Ra(x_t, x_g))] - E_{x_g}[\log(1 - D_Ra(x_g, x_t))]$$

In this work, L_D^{Ra} is the relativistic average discriminator, x_t and x_g are target and generated high-resolution license plate image, and $E_x$represents the operation of taking images in mini-batches.
Some of the results for the predicted reconstructed images of the low LP images are shown in the Figure 4.2. The images on the left hand side of the figure are the input low resolution LP and the reconstructed SR images are shown in the right side while the ground truths are shown in the middle.

4.3 EasyOCR Model

The optical character recognition is a recognition method in which the input is an image and the output is string of character. OCR is a process which separates the different characters from each other taken from an image. Template matching is one of the approaches of OCR. The cropped image is compared with the template data stored in database. OCR automatically identifies and recognizes the characters without any indirect input. The characters on the number plate have uniform fonts then the OCR for number plate recognition is less complex as compared to other methods.

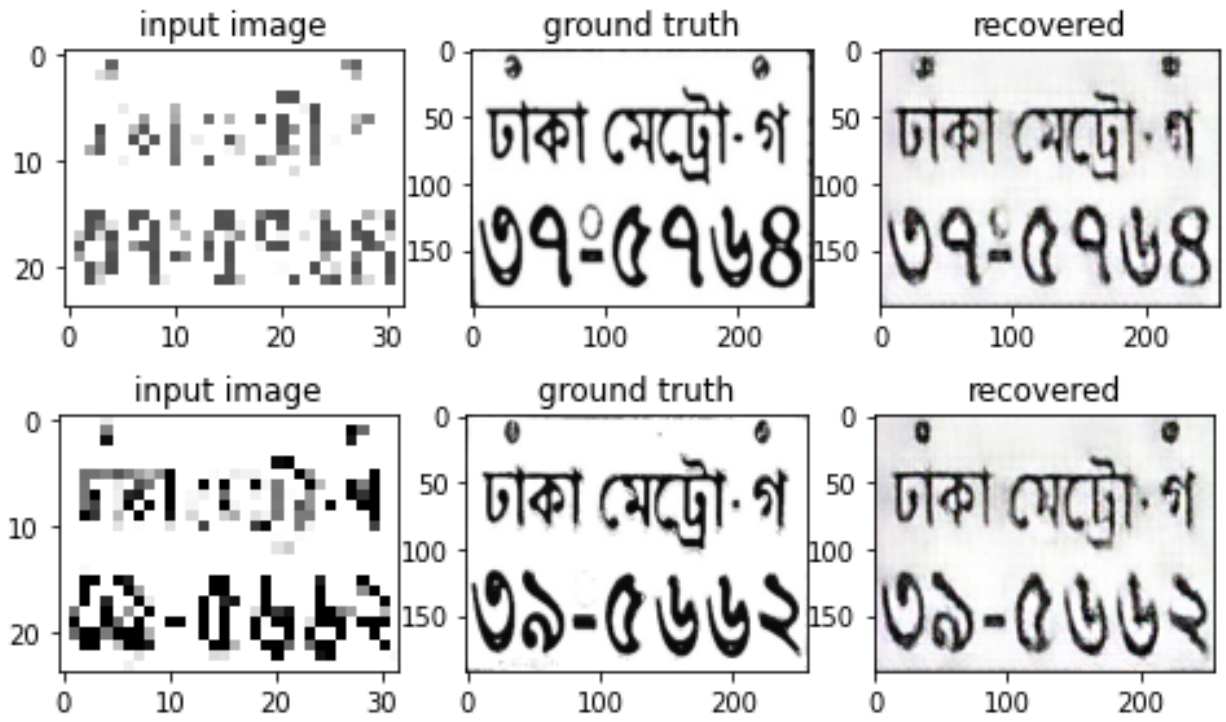


Figure 4.2

EasyOCR is an open-source, ready-to-use Optical Character Recognition(OCR) with 80+ supported languages and all popular writing scripts including Bangla, Latin, Chinese, Arabic, Devanagari, Cyrillic, etc.

The inter-workings of the EasyOCR are presented in Figure 4.3 where we can see the framework uses two Neural Network models. First, the the image has to go through CRAFT algorithm for Scene text Identification(STI) [11]. CRAFT is STI technique that accurately identifies text areas by examining each character and character affinity. The method ensures high flexibility in detecting complex scene text images, such as texts that are randomly orientated, curved, or deformed.
Next, the recognition is done on the text areas that were detected in the input image in the previous step. The EasyOCR uses CRNN model [12] for the recognition. It consists of three basic parts: feature extraction and VGG, sequence labeling (LSTM)[], and decoding (CTC)[13]. For feature extraction, currently using Resnet [14]. A modified version of the deep-text-recognition-benchmark framework serves as the training pipeline for recognition execution.

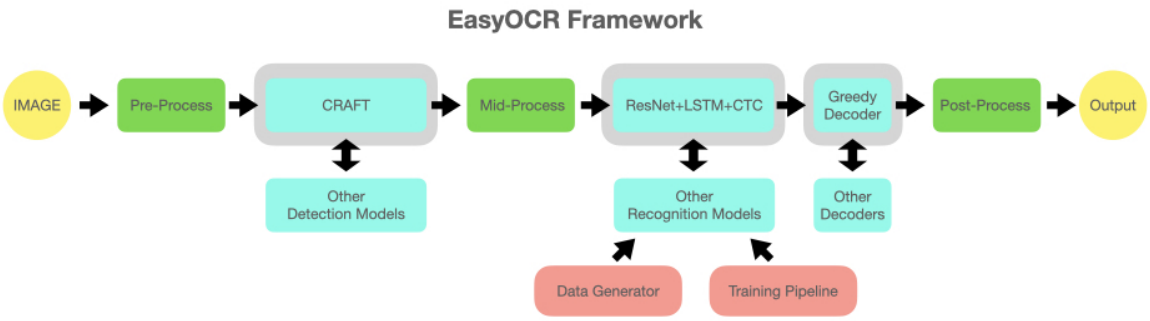


Figure 4.3: Framework for the EasyOCR model.

Chapter 5

Proposed methodology

===

Our end-to-end ALPR system have four sub-processes in order to recognise license plate images. The steps of our suggested technique are as follows:

1. **License Plate Detection**
2. **Prepossessing**
3. **Super-resolution**
4. **Character Recognition**

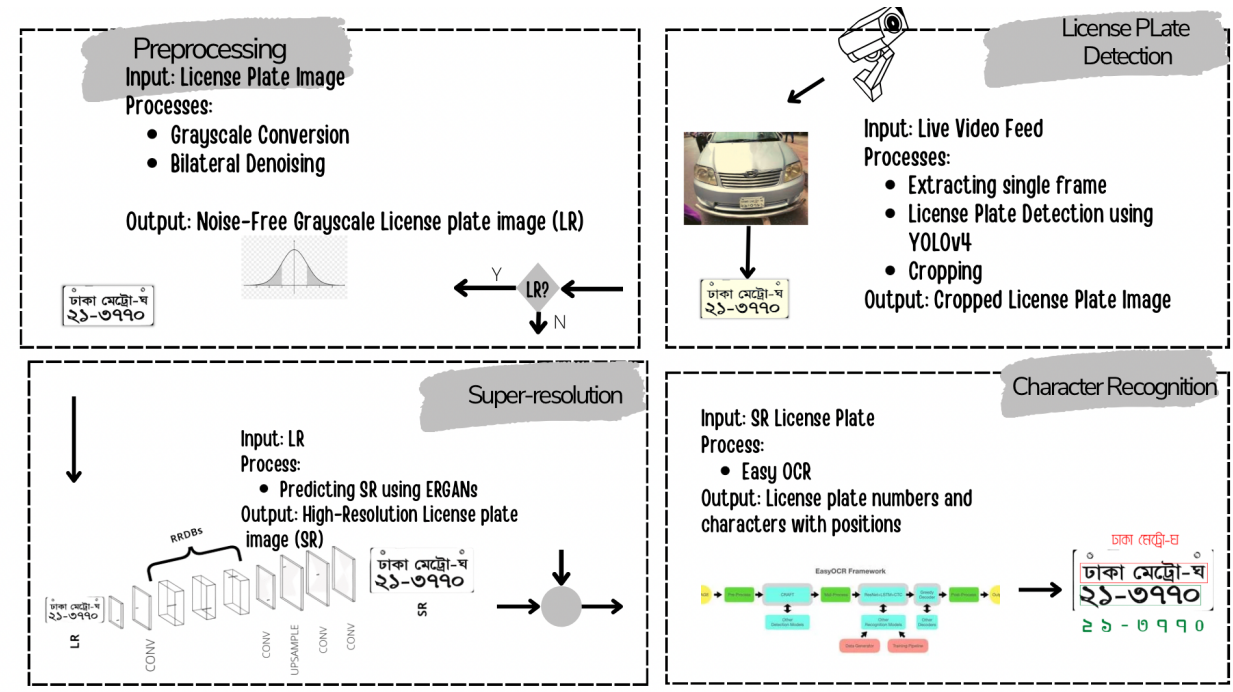


Figure 5.1: Four stage process to detect, localise and recognise low resolution license plate images

In the beginning, we capture solitary frames from live videos. We used the **YOLOv4** model to detect the license plate in order to extract the region of interests (ROI) from the license plate(lp) images. In order to obtain a license plate image without noise, we then applied some prepossessing to that image. We increased the resolution of our low resolution image using ERGANs. Finally, we used **EasyOCR** to localize and detect scene texts in Bangla.

5.1 License Plate Detection

Firstly we need to detect license plate portion from the vehicle image. But detecting license plate from real life scenario is not so easy because of high speed cars, low light environment

conditions, and various type of vehicles. To overcome this adverse situation we need to use some advanced machine learning procedure. That's why we used **YOLOv4** deep learning object detector method to detect LP. We used pretrained model of YOLOv4 and applied this model to our dataset to detect license plate.

The YoLOv4 ,first, divides the image into an S x S grid. (1) Each of these grid cells predicts B bounding boxes and confidence scores for these boxes. The confidence score indicates how sure the model is that the box contains an object and also how accurate it thinks the box is that predicts. The confidence score can be calculated using the formula: $C = P_r(object) * IoU$ IoU: Intersection over Union between the predicted box and the ground truth. If no object exists in a cell, its confidence score should be zero.

Each bounding box consists of five predictions: $x, y, w, h, , confidence$. Additionally, C conditional class probabilities $P_r(Class_i|Object)$ are predicted for each grid cell. Regardless of the number of boxes B, it only predicts one set of class probabilities per grid cell. Individual box confidence forecasts are compounded by these conditional class probabilities during testing to provide class-specific confidence scores for each box. These ratings display the likelihood of that class as well as how well the box matches the object.

$$P_r(Class_i|Object) * P_r(Object) * IoU = P_r(Class_i) * IoU$$

The final predictions are encoded as an $SxSx(B * 5 + C)$ tensor. Multiple detection of the same object may be discovered by the method. Through the use of a method called non-max suppression, the algorithm only identifies the item once. Think about a scenario in which the algorithm discovered three bounding boxes for the same object. The graphic below displays the boxes with the corresponding probability.

Figure 5.2: Multiple Bounding Boxes Of the Same Object

The boxes have probability of 0.70, 0.97 and 0.65 respectively. We will first choose the box with the highest probability and output that as a prediction before removing the duplicates. Then, given the expected output, remove any bounding box with $IoU > 0.5$ (or any threshold value). The outcome is 0.97.

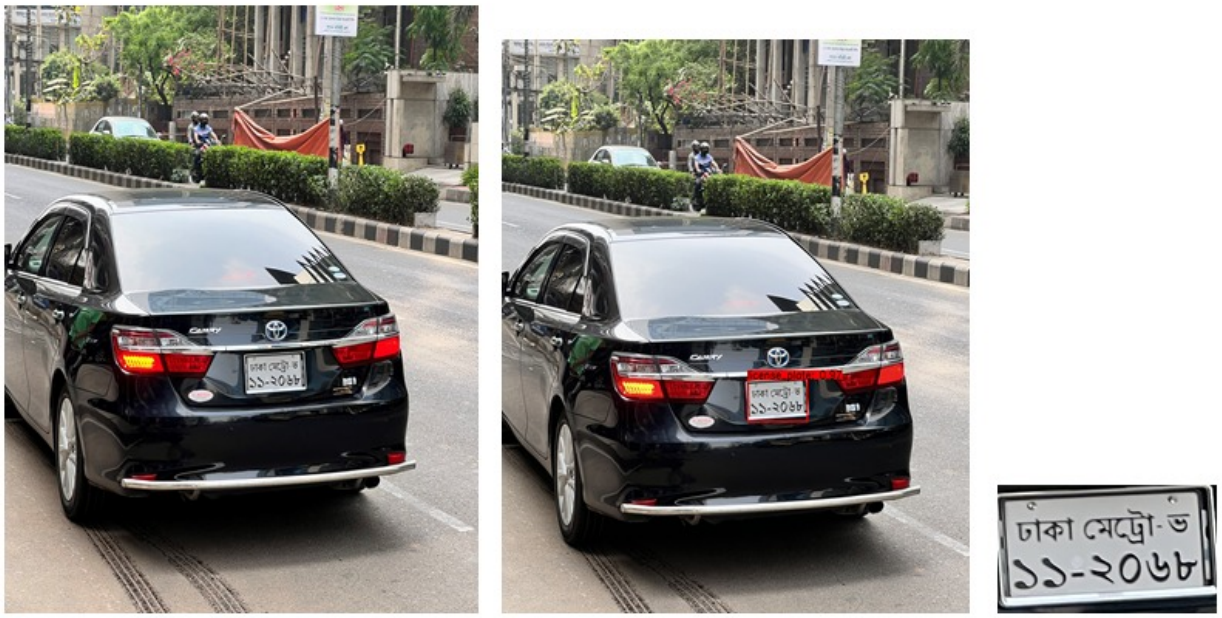


Figure 5.3: After detecting the license plate we cropped the detected portion from the image

5.2 Pre-processing

From the detection phase, we get the output of the bounding box consists of four predictions$(x, y, w, h,)$ with maximum $confidence$ score for the image frame containing car with license plate. Using these predictions, we crop-out the license plate regions from the images. For further scene text recognition using **EasyOCR**, we pre-process our cropped-out lp. The colored lp obtained in the detection layer converted first into grayscale image. Next, the grayscale image is bilaterally filtered to reduce the noise.

The cropped-out gray lp images in resolution depending on the distance of the camera from the lp of the car. But, in order to train the Super Resolution model, we need to have the training images of fixed length. We resized our $2,211$ training images of variable dimension to $[256 \times 192]$ such that when we want a pass low resolution image of $[32 \times 24]$ size, the image resolution increase to $[256 \times 192]$ with an up-scaling factor of 8.

5.3 Super Resolution(×8)

This is the crucial part of the our ALPR system is the Super Resolution as we focussed more on the low lp images. We used Enhanced Super Resolution Generative Adversarial Networks (ESRGAN) where we used up-scaling factor of 8 to upscale the low resolution images. The architecture of the generator of the ESRGAN is shown in the Figure 5.4 which consists of several complicated Residual-in-Residual Dense Block (RRDB) structures.

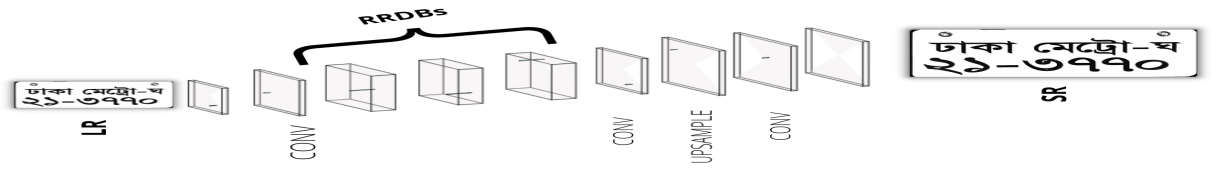


Figure 5.4: Overview of the **ESRGAN's generator model**

The parameters to train the Generator and the Discriminator networks for the super-resolution have be shown in the Table 5.1:

Table 5.1: The parameters for the training ESRGAN

Generator Architecture Parameters	
No. RRDB blocks	10
Loss Functions	MAE+VGG Loss
Optimizer	Adam
Learning Rate	1×10^{-3}
Epochs	5
Steps per Epoch	300
GANs training Parameters	
lrGenerator	1×10^{-4}
lrDiscriminator	$1x10^{-4}$
epochs	2
stepsPerEpoch	500
valSteps	100

5.4 Optical Character recognition

After, the Super Resolution Prediction from the previous stage, we obtain the reconstructed high resolution image of a low resolution LP image. The SR image is then passed through the frameworks of EasyOCR 4.3. The LR license plate is passed through CRNN (cite) model for the recognition of the scene text region. After that sequence labeling is done using **CTC**

(Labelling Unsegmented Sequence Data with Recurrent Neural Networks) and is done to extract the bangla texts in the detected text area. The output of the framework is the bounding box of the predicted text area with class prediction and confidence score associated with each bounding box.

Chapter 6

Results

===

We have calculated the confusion matrix of license plate detection. It was drawn table 6.1. The confusion matrix was generated: **2725** dataset images

Table 6.1: Confusion Matrix of license plate detection

	Positive	**Negative**
Predicted Positive	2568	5
Predicted Negative	98	54

And the accuracy rate was 96.2%. Where the precision was 99.8% and recall was 96.3%. The FPR rate was 8.5% and finally, the F-1 Score was 98%.

Table 6.2: Calculation of Confusion Matrix for license plate

Measure	**Value**
Precision	0.998
Recall	0.963
FPR	0.085
F-Measure	0.980
Accuracy	0.962

We trained our **ESRGAN** [15] model using **2211** LP images which are carefully selected in order to obtain maximum reconstructed resolution. After training the model for 5 epochs, the **Mean Squared Error (MSE)** and **Peak Signal to Noise Ratio (PSNR)** for the reconstructed LP images are **0.055** and **12.587** which obtained for **200** testing low resolution **LP** images of size $[32 \times 28]$. The Figure 6.1 indicates that **training PSNR (psnr)** and **validation PSNR (val_psnr)** trend while training. The diagram shows the upward trend for both measure for 5 epochs.

the **Mean Squared Error (MSE)** and **Peak Signal to Noise Ratio (PSNR)** for the reconstructed lp images are **0.055** and **12.587** which obtained for **200** testing low resolution lp images of size $[32 \times 28]$. The Figure 6.1 indicates that **training psnr(psnr)** and **validation psnr (val**$_p$$snr)trendwhiletraining.Thediagramshowstheupwardtrendfortheboth measurefor5epochs.$
Some of the results for the predicted reconstructed images of the low LP images are shown in the Figure 4.2. The images on the left hand side of the figure are the input low resolution LP and the reconstructed SR images are shown in the right side while the ground truths are shown in the middle.
The reconstructed Super Resolutions images of size $[256 \times 224]$ **are then passed through the framework of EasyOCR to obtain scene text area localization bounding box and the**

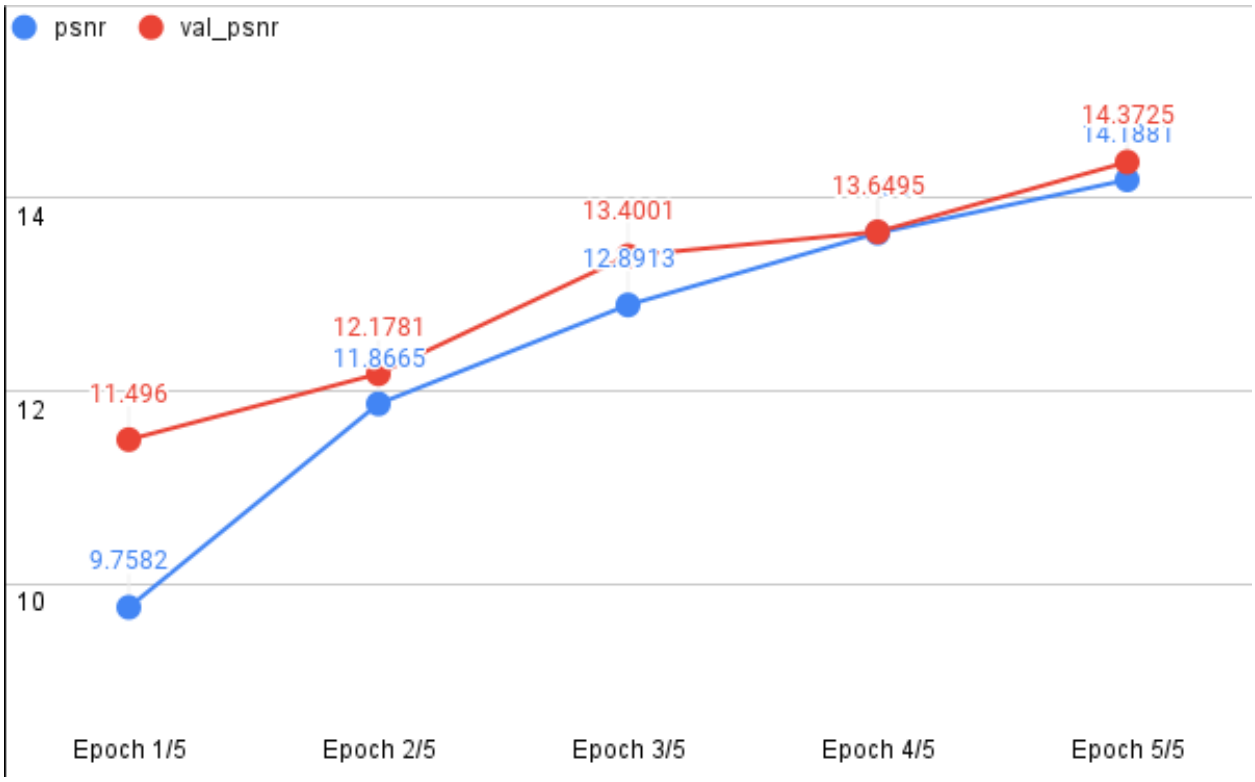


Figure 6.1: bounding box and confidence score for some of the ground truth sample

prediction text with the confidence score for each text areas. The recognition for some of the ground truth lp images has been shown in the Figure: 6.2.

Table 6.3: **Accuracies based on class prediction**

Class name	**Accuracy(LR->SR)**	**Accuracy(GT)**
Dhaka Metro	1.5	1.5
Ka, Kha, Ga, Gha, Ca, Cha, Ja, Jha, Ta, Tha, Da, Dha, Na, Pa, Pha, Ma, THA, E, Ha, La, A, Bha, Sha, U, Sa, Ba	0.5	0.91
0, 1, 2, 3, 4, 5, 6, 7,8,9	0.714	0.898

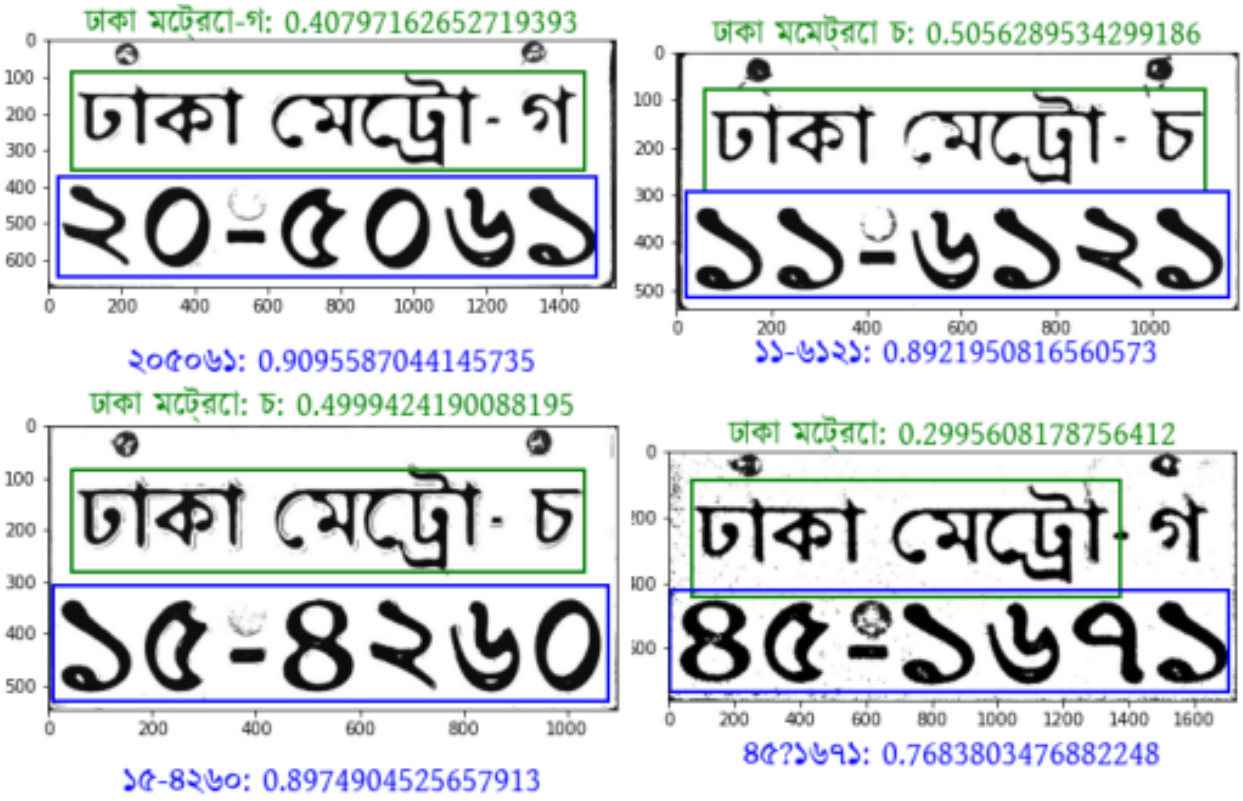


Figure 6.2: Bar graph for the Accuracies of the trained models on the testing set

Chapter 7

Conclusion Future Work

===

Using YOLOv4 and EasyOCR [16], we have constructed an end-to-end license plate recognition system consisting of four stages. However, our system was specially designed such that it can handle low-resolution images. Since it is difficult for the recognition model to extract features from low-resolution images. We trained ESRGAN model to upscale our images by a factor of 8. The system evaluated the recognition accuracy of 78% and 91% on the reconstructed images, upscaled by the generator model low resolution of size 32×24, and on their ground truth images.
On the face of it, it looks as though the recognition accuracy is a bit less than previous approaches. Nevertheless, the system's performance should be considered in the context of low resolution. None of the previous works focused on low resolution; not as low as 32×24 dimension. The resized low-resolution images are not even recognizable by the human eye. In the future, we are looking forward to comparing our model with human. The recognition model is pretrained; if only the model is fine-tuned using training data, the system can potentially excel over every other existing ALPR system.

The proposed system tested on Bangladeshi vehicles shows a complete approach that achieves a mean average precision of 99.8% for LP detection and 89.8% for LR license Plate recognition recognition separately, which is the highest ever accuracy on Bangla LP to the best of our knowledge. The proposed method works well with blurry low-resolution LP.The dataset will be more balanced with additional data gathering from particular areas and vehicles.but the results are more precise. Despite the accuracy of our results for image input, this study can be expanded to real-time Bangla LP data identification. More research should be done to identify distorted and skewed LP in order to boost receptivity in real-world situations.

we created a dataset consisting of license plates for multi-class vehicles. For detecting the license plate,we used YOLOV4 method and extract the license plate for recognition. The extracted image from the first network is the input for the second ESRGAN"s model which is responsible for Super resolution and then for recognition the license plate number we have used Easy OCR recognizing . Our model was tested with 200 images and correctly produced the license plate number.

Chapter 8

Appendices

===

Detecting License plate:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if len(physical_devices) > 0:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)
from absl import app, flags, logging
from absl.flags import FLAGS
import core.utils as utils
from core.yolov4 import filter_boxes
from core.functions import *
from tensorflow.python.saved_model import tag_constants
from PIL import Image
import cv2
import numpy as np
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

flags.DEFINE_string('framework', 'tf', '(tf, tflite, trt')
flags.DEFINE_string('weights', './checkpoints/yolov4-416',
                    'path to weights file')
flags.DEFINE_integer('size', 416, 'resize images to')
flags.DEFINE_boolean('tiny', False, 'yolo or yolo-tiny')
flags.DEFINE_string('model', 'yolov4', 'yolov3 or yolov4')
flags.DEFINE_list('images', './data/images/kite.jpg', 'path to input image')
flags.DEFINE_string('output', './detections/', 'path to output folder')
flags.DEFINE_float('iou', 0.45, 'iou threshold')
flags.DEFINE_float('score', 0.50, 'score threshold')
flags.DEFINE_boolean('count', False, 'count objects within images')
flags.DEFINE_boolean('dont_show', False, 'dont show image output')
flags.DEFINE_boolean('info', False, 'print info on detections')
flags.DEFINE_boolean('crop', True, 'crop detections from images')
flags.DEFINE_boolean('ocr', False, 'perform generic OCR on detection regions')
flags.DEFINE_boolean('plate', False, 'perform license plate recognition')

def main(_argv):
    config = ConfigProto()
    config.gpu_options.allow_growth = True
    session = InteractiveSession(config=config)
    STRIDES, ANCHORS, NUM_CLASS, XYSCALE = utils.load_config(FLAGS)
    input_size = FLAGS.size
```

```
    images = FLAGS.images
    folder = './data/images/'
    count=0

    # load model
    if FLAGS.framework == 'tflite':
            interpreter = tf.lite.Interpreter(model_path=FLAGS.weights)
    else:
            saved_model_loaded = tf.saved_model.load(FLAGS.weights, tags=[
tag_constants.SERVING])
    entries = sorted(os.listdir(folder), key=len)
    # loop through images in list and run Yolov4 model on each
    for image_path in entries:
        count = count+1
        original_image = cv2.imread(os.path.join(folder,image_path))
        original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
        print(os.path.join(folder,image_path))
        image_data = cv2.resize(original_image, (input_size, input_size))
        image_data = image_data / 255.

        # get image name by using split method
        image_name = image_path.split('/')[-1]
        image_name = image_name.split('.')[0]

        images_data = []
        for i in range(1):
            images_data.append(image_data)
        images_data = np.asarray(images_data).astype(np.float32)

        if FLAGS.framework == 'tflite':
            interpreter.allocate_tensors()
            input_details = interpreter.get_input_details()
            output_details = interpreter.get_output_details()
            interpreter.set_tensor(input_details[0]['index'], images_data)
            interpreter.invoke()
            pred = [interpreter.get_tensor(output_details[i]['index']) for i in
range(len(output_details))]
            if FLAGS.model == 'yolov3' and FLAGS.tiny == True:
                boxes, pred_conf = filter_boxes(pred[1], pred[0], score_threshold
=0.25, input_shape=tf.constant([input_size, input_size]))
            else:
                boxes, pred_conf = filter_boxes(pred[0], pred[1], score_threshold
=0.25, input_shape=tf.constant([input_size, input_size]))
        else:
            infer = saved_model_loaded.signatures['serving_default']
            batch_data = tf.constant(images_data)
            pred_bbox = infer(batch_data)
```

```
            for key, value in pred_bbox.items():
                boxes = value[:, :, 0:4]
                pred_conf = value[:, :, 4:]

        # run non max suppression on detections
        boxes, scores, classes, valid_detections = tf.image.
    combined_non_max_suppression(
            boxes=tf.reshape(boxes, (tf.shape(boxes)[0], -1, 1, 4)),
            scores=tf.reshape(
                pred_conf, (tf.shape(pred_conf)[0], -1, tf.shape(pred_conf)[-1]))
    ,
            max_output_size_per_class=50,
            max_total_size=50,
            iou_threshold=FLAGS.iou,
            score_threshold=FLAGS.score
        )

        # format bounding boxes from normalized ymin, xmin, ymax, xmax ---> xmin,
     ymin, xmax, ymax
        original_h, original_w, _ = original_image.shape
        bboxes = utils.format_boxes(boxes.numpy()[0], original_h, original_w)

        # hold all detection data in one variable
        pred_bbox = [bboxes, scores.numpy()[0], classes.numpy()[0],
    valid_detections.numpy()[0]]

        # read in all class names from config
        class_names = utils.read_class_names(cfg.YOLO.CLASSES)

        allowed_classes = list(class_names.values())


        if pred_bbox is not None:
            # if crop flag is enabled, crop each detection and save it as new
    image
            if FLAGS.crop:
                crop_path = os.path.join(os.getcwd(), 'detections', 'crop')

                crop_objects(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB),
    pred_bbox, crop_path, allowed_classes, image_name)

            # if ocr flag is enabled, perform general text extraction using
    Tesseract OCR on object detection bounding box
            if FLAGS.ocr:
                ocr(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB), pred_bbox)

            # if count flag is enabled, perform counting of objects
```

```
            if FLAGS.count:
                counted_classes = count_objects(pred_bbox, by_class = False, allowed_classes=allowed_classes)
                for key, value in counted_classes.items():
                    print("Number of {}s: {}".format(key, value))
                image = utils.draw_bbox(original_image, pred_bbox, FLAGS.info, counted_classes, allowed_classes=allowed_classes, read_plate = FLAGS.plate)
            else:
                image = utils.draw_bbox(original_image, pred_bbox, FLAGS.info, allowed_classes=allowed_classes, read_plate = FLAGS.plate)

            image = Image.fromarray(image.astype(np.uint8))
            #if not FLAGS.dont_show:
                #image.show()
            image = cv2.cvtColor(np.array(image), cv2.COLOR_BGR2RGB)
            cv2.imwrite(FLAGS.output + 'detection_iii_' + image_name + '.jpg', image)

if __name__ == '__main__':
    try:
        app.run(main)
    except SystemExit:
        pass
```

Listing 8.1: Detecting License plate

Training of Super resolution

```
from google.colab import drive
drive.mount('/content/drive')
!git clone https://github.com/naimul011/BanglaLicencePlateSuperResolutionGAN
!git clone https://github.com/zzxvictor/License-super-resolution
!mv License-super-resolution/Models ./Models
!rm -r License-super-resolution
!mv BanglaLicencePlateSuperResolutionGAN/Utilities/ ./Utilities
!rm -r BanglaLicencePlateSuperResolutionGAN
from Utilities.io import DataLoader
from Utilities.lossMetric import *
from Utilities.trainVal import MinMaxGame
from Models.RRDBNet import RRDBNet
from Models.GAN import Discriminator
! ls /content/drive/MyDrive/Processed_data/training_data | wc -l
import numpy as np
import glob
#PATH = 'PATH_TO_OUTPUT_DIR/192_96' # onlyuseimageswithshape192by96fortraining
PATH = '/content/drive/MyDrive/Processed_data/training_data'
files = glob.glob(PATH + '/*.png') * 3 # dataaugmentation , sameimagewithdifferentbrightnessandcontrast
np.random.shuffle(files)
train, val = files[:int(len(files)*0.8)], files[int(len(files)*0.8):]
loader = DataLoader()
trainData = DataLoader().load(train, batchSize=16)
valData = DataLoader().load(val, batchSize=64)
discriminator = Discriminator()
extractor = buildExtractor()
generator = RRDBNet(blockNum=10)
# asimplecustomlossfunctionthatcombinesMAElosswithVGGloss , asdefinedintheSRGANpaper
def contentLoss(y_true, y_pred):
    featurePred = extractor(y_pred)
    feature = extractor(y_true)
    mae = tf.reduce_mean(tfk.losses.mae(y_true, y_pred))
    return0 .1*tf.reduce_mean(tfk.losses.mse(featurePred, feature)) + mae

optimizer = tfk.optimizers.Adam(learning_rate=1e-3)
generator.compile(loss=contentLoss, optimizer=optimizer, metrics=[psnr, ssim])
# epochissetto1fordemonstrationpurpose . InpracticeIfound20isagoodnumber
# WhenthemodelreachesPSNR=20/ssim=0.65, wecanstartthemin-max game
history = generator.fit(x=trainData, validation_data=valData, epochs=5,
```

```
        steps_per_epoch =300, validation_steps =100)
# trainingparameter . epochissetto1fordemonstration
# pleasetrainthenetworkutillitreachessnRatio ~= 22
PARAMS = dict (lrGenerator = 1e -4,
                lrDiscriminator = 1e -4,
               epochs = 2 ,
                  stepsPerEpoch = 500 ,
                  valSteps = 100 )
game = MinMaxGame (generator, discriminator , extractor )

# ideallypeaksignalnoiseratio (snRation or psnr) shouldreach ~22

log, valLog = game .train(trainData, valData , PARAMS )

#generator.save_weights('/content/drive/MyDrive/Dataset/rrdb/', save_format ='tf
    ')
generator.save_weights( save_format ='tf')
#generator.save_weights('generator.h5')
!ls
!cp rrdb.data-00000-of-00001 /content/drive/MyDrive/trained_model2
!cp chechpoint /content/drive/MyDrive/trained_model2
!cp rrdb.index /content/drive/MyDrive/trained_model2
```

Listing 8.2: Super resolution

Testing of Super resolution

```
i = 1
sr = []
gt = []
for downSample, original in data.take(200):
    yPred = model.predict(downSample)
    #downSample = np.squeeze(np.clip(downSample, a_min=0, a_max=1))
    #yPred = np.squeeze(np.clip(yPred, a_min=0, a_max=1))
    #original = np.squeeze(np.clip(original, a_min=0, a_max=1))

    sr.append(yPred)
    gt.append(original)
    #plt.imshow(downSample)
    #plt.imsave(f'/gdrive/MyDrive/downsample/image_new{i}',downSample,format='png
    ')
    #plt.imsave(f'/gdrive/MyDrive/recovered/image_new{i}',yPred,format='png')
    #cv2.imwrite(f'/gdrive/MyDrive/recovered/image_{i}.png',yPred)
    i += 1
    #painter.plot(downSample, original, yPred)
```

Listing 8.3: Super resolutio

Character Recognition

```
# -*- coding: utf-8 -*-
"""Untitled0.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1hHAol0ygZAZSP4lHN8Tn60tKn-k5kOH3
"""

!pip install easyocr
!pip install imutil

!pip uninstall opencv-python-headless==4.5.5.62
!pip install opencv-python-headless==4.1.2.30

import cv2
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
from PIL import ImageFont, ImageDraw, Image
import numpy as np
import imutils
import easyocr
from matplotlib.patches import Rectangle
from google.colab import files

from google.colab import drive
drive.mount('/gdrive')

! ls -1 /gdrive/MyDrive/Processed_data/training_data | wc -l

! ls -1 /gdrive/MyDrive/Dataset/preprocessed  | wc -l

# create figure
fig = plt.figure(figsize=(10, 7))

# setting values to rows and column variables
rows = 4
columns = 10


for i in range(40):
  name = '/gdrive/MyDrive/downsample/image_new'+str(i+1)
  img = cv2.imread(name)

```

```
  fig.add_subplot(rows, columns, i+1)

  plt.imshow(img)
  plt.axis('off')




#plt.savefig("recog.png")
#files.download("recog.png")

img = cv2.imread('2.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))
#plt.savefig("1.png")
#files.download("1.png")

bfilter = cv2.bilateralFilter(gray, 11, 17, 17) #Noise reduction
edged = cv2.Canny(bfilter, 30, 200) #Edge detection
plt.imshow(cv2.cvtColor(bfilter, cv2.COLOR_BGR2RGB))
#plt.savefig("1.png")
#files.download("1.png")

reader = easyocr.Reader(['bn'])

result = reader.readtext(bfilter)

print(result[0][1])
print(result[0][0])

print(result[1][1])
print(result[1][0])

#(x_1,y_1) and then width=x_4-x_1, height=y_2-y_1
print(result[0][0][1][1])
x1 = result[0][0][0][0]
y1 = result[0][0][0][1]

x4 = result[0][0][3][0]
y2 = result[0][0][3][1]

width=x4-x1
height=y2-y1

result

```

```
prop = fm.FontProperties(fname='/gdrive/MyDrive/kalpurush.ttf')
font = ImageFont.truetype('/gdrive/MyDrive/kalpurush.ttf', 200)

width, height = Image.fromarray(bfilter).size

plt.text(width/2, -20, result[1][1]+": 0.5056289534299186", fontproperties=prop, fontsize=20, horizontalalignment='center', color='GREEN')
plt.text(width/2, 650, result[2][1]+": 0.8921950816560573", fontproperties=prop, fontsize=20, horizontalalignment='center', color='RED')



plt.imshow(cv2.cvtColor(bfilter, cv2.COLOR_BGR2RGB))
plt.gca().add_patch(Rectangle((60, 76),1114-60,292-76,linewidth=2,edgecolor='g',facecolor='none'))
plt.gca().add_patch(Rectangle((19, 294),1164-19,516-294,linewidth=2,edgecolor='r',facecolor='none'))
#[44, 84], [1034, 84], [1034, 280], [44, 280]
#[9, 308], [1078, 308], [1078, 531], [9, 531]
plt.savefig("2.png")
files.download("2.png")

#!/usr/bin/env python
# -*- coding: utf-8 -*-
text = result[0][-2] +' ' + result[1][-2] + ' ' + result[3][-2] + result[2][-2]
#font = cv2.FONT_HERSHEY_SIMPLEX
font = ImageFont.truetype('/gdrive/MyDrive/kalpurush.ttf', 200)
img=Image.fromarray(bfilter)
draw = ImageDraw.Draw(img)
HEIGHT = 236
WIDTH = 236
w, h = draw.textsize(text, font=font)
draw.text((0,40), text, font=font)
#res = cv2.putText(bfilter, text=text, org=(00,30), fontFace=font, fontScale=1, color=(0,255,0), thickness=2, lineType=cv2.LINE_AA)
#res = cv2.rectangle(bfilter, tuple(approx[0][0]), tuple(approx[2][0]), (0,255,0),3)
plt.imshow(img)

import glob
from skimage import io

prop = fm.FontProperties(fname='/gdrive/MyDrive/kalpurush.ttf')
font = ImageFont.truetype('/gdrive/MyDrive/kalpurush.ttf', 50)

images = []
results = []
```

```
i  = 0
reader = easyocr.Reader(['bn'])

for img in glob.glob("/gdrive/MyDrive/Processed_data/testing/*"):
    images.append(img)
    img = cv2.imread(img)
    result = reader.readtext(img)

    width, height = Image.fromarray(img).size

    #text = result[0][-2] +' ' + result[1][-2] + ' ' + result[3][-2] + result[2][-2]
    #text = str(result[0]) +'/n'+str(result[1])
    plt.text(width/2, -20, result[0], bbox=dict(fill=False, edgecolor='red', linewidth=2), fontproperties=prop, fontsize=15, horizontalalignment='center', color='green')
    plt.text(width/2, 20, result[-1], bbox=dict(fill=False, edgecolor='red', linewidth=2), fontproperties=prop, fontsize=15, horizontalalignment='center', color='green')

    plt.imshow(img)
    #plt.title(result)
    plt.savefig("/gdrive/MyDrive/recognized_200tootoo/"+str(i)+".png")
    plt.close()

    results.append(result)
    i += 1

images

import pandas as pd

# intialise data of lists.
data = {'Reecovered Images':images,
        'Recogtiion': results}

# Create DataFrame
df = pd.DataFrame(data)

df.to_csv('/gdrive/MyDrive/result_200tootoo.csv', index=False)
df

text = result[0][-2] +' ' + result[1][-2] + ' ' + result[3][-2] + result[2][-2]
prop = fm.FontProperties(fname='/gdrive/MyDrive/kalpurush.ttf')
font = ImageFont.truetype('/gdrive/MyDrive/kalpurush.ttf', 200)

width, height = Image.fromarray(bfilter).size
```

```
175
176 plt.text(width/2, -10, text, bbox=dict(fill=False, edgecolor='red', linewidth=2),
        fontproperties=prop, fontsize=20, horizontalalignment='center', color='green'
    )
177 plt.imshow(cv2.cvtColor(bfilter, cv2.COLOR_BGR2RGB))
178 plt.savefig("1.png")
179 files.download("1.png")
180
181 !ls /gdrive/MyDrive/Processed_data/testing_data | wc -l
182
183 import glob
184 i = 0
185
186 for img in glob.glob("/gdrive/MyDrive/Processed_data/testing/*.jpg"):
187     n = cv2.imread(img, cv2.IMREAD_UNCHANGED)
188     resized = cv2.resize(n, (256, 192))
189     #print(img)
190     cv2.imwrite(f'/gdrive/MyDrive/Processed_data/testing_data/image_{i}.png',
    resized)
191     i += 1
192
193 import glob
194 i = 0
195
196 for img in glob.glob("/gdrive/MyDrive/Processed_data/training/*.jpg"):
197     n = cv2.imread(img, cv2.IMREAD_UNCHANGED)
198     resized = cv2.resize(n, (256, 192))
199     #print(img)
200     cv2.imwrite(f'/gdrive/MyDrive/Processed_data/training_data/image_{i}.png',
    resized)
201     i += 1
202
203 """## Batch Processing"""
204
205 import glob
206 from skimage import io
207
208
209 i = 0
210
211 for img in glob.glob("/gdrive/MyDrive/yolov4-updated/crop/*.jpg"):
212     print(img)
213     print(i)
214
215     n = cv2.imread(img)
216     #n = io.imread(img)
217     if(n is not None):
```

```
        gray = cv2.cvtColor(n, cv2.COLOR_BGR2GRAY)
    bfilter = cv2.bilateralFilter(gray, 11, 17, 17) #Noise reduction
    #print(img)
    cv2.imwrite(f'/gdrive/MyDrive/Dataset/preprocessed/image_{i}.png',bfilter)
    i += 1
```

Listing 8.4: Character Recognition

References

[1] Homaira Shomee and Ataher Sams. "License Plate Detection and Recognition System for All Types of Bangladeshi Vehicles Using Multi-step Deep Learning Model". In: Nov. 2021, pp. 01–07. DOI: 10.1109/DICTA52665.2021.9647284.

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

[3] Tao Hong and Anilkumar Kothalil Gopalakrishnam. "License plate extraction and recognition of a Thai vehicle based on MSER and BPNN". In: *2015 7th International Conference on Knowledge and Smart Technology (KST)* (2015), pp. 48–53.

[4] Nur-A Alam et al. "Intelligent system for vehicles number plate detection and recognition using convolutional neural networks". In: *Technologies* 9.1 (2021), p. 9.

[5] Sergio Montazzolli and Cláudio Rosito Jung. "Real-Time Brazilian License Plate Detection and Recognition Using Deep Convolutional Neural Networks". In: *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (2017), pp. 55–62.

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

[7] Mingsong Chen et al. "Video vehicle detection and recognition based on mapreduce and convolutional neural network". In: *International Conference on Swarm Intelligence*. Springer. 2018, pp. 552–562.

[8] Md Mesbah Sarif et al. "Deep learning-based Bangladeshi license plate recognition system". In: *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE. 2020, pp. 1–6.

[9] Sohaib Abdullah, Md Mahedi Hasan and Sheikh Muhammad Saiful Islam. "YOLO-based three-stage network for Bangla license plate recognition in Dhaka metropolitan city". In: *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*. IEEE. 2018, pp. 1–6.

[10] **Nazmus Saif et al. "Automatic license plate recognition system for bangla license plates using convolutional neural network". In: *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE. 2019, pp. 925–930.**

[11] **Youngmin Baek et al. "Character region awareness for text detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9365–9374.**

[12] **Baoguang Shi, Xiang Bai and Cong Yao. "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 39.11 (2016), pp. 2298–2304.**

[13] **Alex Graves et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 369–376.**

[14] **Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.**

[15] **Xintao Wang et al. "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks". In: *CoRR* abs/1809.00219 (2018). arXiv: 1809.00219. URL: http://arxiv.org/abs/1809.00219.**

[16] **JaidedAI. *EasyOCR*. https://github.com/JaidedAI/EasyOCR. 2021.**

bibliography.bib