The British College
KATHMANDU

Coursework Submission Coversheet
(individual coursework only)

Faculty of Arts, Environment and Technology

LBU Student Id: 77202315

For checking by the student:

Please ensure all information is complete and correct and attach this form securely to the front of your work before posting it in a coursework collection box.

Award name: BSc (Hons) Computing

Module code: COMP607

Module name: Production Project

Module run: April 2021 – July 2021

Coursework title: Automatic Nepali Number Plate Recognition from vehicle image using AI

Due Date: 12th July 2021

Module leader: (In LBU) Patrick Ingham

Module Supervisor: (In TBC) Rohit Raj Pandey

TURNITIN Checked: YES

Submission date & time: Date: 12th July 2021 Time: 7:30 PM

Total Word Count: 12252 **Total Number of Pages (including this front sheet):** 77

In submitting this form with your assignment, you make the following declaration:
I declare, that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification either at LBU or elsewhere. I have fully attributed/referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.

Signed: Pravesh Date: 12th July 2021

For completion by the faculty:

☐ This mark is provisional and subject to moderation and approval by the relevant examining board

Teacher's Feedback

Teacher's Signature: _____ **Date:** _____

Automatic Nepali Number Plate Recognition from vehicle image using AI

Student Name: Pravesh Pansari

ID: 77202315

Course: B.Sc. Computing (Hons)

I. ABSTRACT

The task of identifying a vehicle by extracting the vehicle license plate from an image or video is known as automatic number plate recognition. It reads symbols on license plates using optical character recognition. License plate localization on the image, character segmentation, character recognition of the segmented characters are the processes involved in a number plate recognition system. This research study aims to present an automatic Nepali number plate recognition system based on machine learning. Before the characters from a localized plate can be segmented and recognized, image artifacts must be removed from the image and the characters must be enhanced. This is done by applying various image pre-processing techniques to the localized plate image. The system then uses Convolutional Neural Network (CNN) for predicting each segmented character. The CNN is trained on a mixed Nepali character dataset comprising of characters generated from different computer fonts and handwritten characters. The accuracy of a such system depends significantly on many factors like image quality, illumination, skewness, distortions, and the presence of image artifacts.

II. CONTENTS

III. LIST OF ABBREVIATIONS

i. **ANPR –** Automatic Number Plate Recognition
ii. **SVM –** Support Vector Machine
iii. **CNN –** Convolutional Neural Network
iv. **ANN –** Artificial Neural Network
v. **OCR –** Optical Character Recognition
vi. **GPU –** Graphical Processing Unit
vii. **HSV –** Hue, Saturation, Value
viii. **RGB –** Red Green Blue
ix. **YOLO –** You Only Look Once
x. **ASD –** Adaptive Software Development
xi. **ReLU –** Rectified Linear Unit

1. INTRODUCTION

Automatic Number Plate Recognition (ANPR) systems and devices are crucial in transportation management and surveillance. They can identify vehicles by detecting the number plate on the vehicle and recognizing the unique identification number provided to each vehicle. Automatic traffic control, electronic toll collection, vehicle tracking and monitoring, border crossing, robbery cases and a variety of other application areas can benefit from or be possible with ANPR systems.

The development of ANPR systems can be categorized into three major steps: locating the number plate, segmenting the characters on the plate, and recognizing the individual characters from the plate. To develop such an ANPR system, computer vision algorithms must be implemented with imaging hardware. Some computer vision techniques required for such systems include number plate localization, plate resizing and binarization, character segmentation, color mapping and optical character recognition (OCR) for character recognition. These techniques are combined with machine learning algorithms to learn to recognize useful features from input data. The accuracy of such system depends on variety of factors: low resolution image, poor lighting conditions, fuzzy images, variation in font style and size and in plate structures.

Automatic number plate recognition is a deeply researched problem and has been implemented successfully in many countries for a long time. There are very few research papers published on this domain for Nepali number plates. Some systems are only able to recognize half of the number plate characters. Nepali number plates are made up of characters from a pool of 29 characters. Furthermore, Nepali number plates use 6 different type of color combination for vehicle classification. Through this paper, a machine learning based Nepali number plate recognition system capable of automatically identifying and labeling a number plate, is proposed.

Due to the variation and recent changes in license plate structures and layouts in Nepal, expecting the system to accurately extract all number is an unreasonable assumption. There are many Nepali license plates that are very old and characters on them are roughened up. As the vehicles in Nepal are categorized based on ownership and service type by different colorings of the license plates, the color can be extracted to provide further information of the vehicle. This can be particularly useful in a scenario where some of the characters are missing or misclassified by the system, and thus can be confirmed or checked in the database with color as an extra parameter. Therefore, this project will output two main information about the vehicle in the image: the identification number, and the vehicle color or type (by ownership).

2. LITERATURE REVIEW

In the last few decades vehicle identification has been an active research. Much research has been conducted to identify the vehicle type, the model, and the number plate. Automatic Number Plate Recognition (ANPR) system is a widely researched field in machine learning and computer vision. Most current practical implementations of such system include use of object detection to localize the number plate on the vehicle, then preprocess the cropped number plate and use edge detection to detect characters in the plate. For character recognition Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs) have been widely used.

An ANPR system is an image processing method that recognizes cars based on their license plates. It comprises of acquiring a picture, detecting the placement of a number in the image, and extracting characters for interpretation (Mutholib, Gunawan, Chebil and Kartiwi, 2013). ANPR systems typically consist of two components: a camera that captures images of car license plates and a software that extracts the plates from the collected images and then uses a character recognition tool that allows pixels to be converted into numerically readable characters (Friedrich, Jehlicka & Schlaich, 2008).

A paper proposed an Automatic Number Plate Recognition System for Nepali Number plates (Pant, Gyawali & Acharya, 2015). They detect the number plate on the image by making use of an HSV mask which primarily recognizes private vehicles – red number plates. SVM is for classifying segmented characters. The character dataset was made up of 12 different characters.

To find a Chinese license plate an approach devised for calculating horizontal and vertical differences to locate a precise rectangle with a vehicle number (Wu & Li, 2011). In this method, the vehicle image is converted to grayscale before being binarized automatically. The time taken to locate a plate with such a method was claimed to be 0.8s. For locating a number plate on an image various method have been proposed and implemented. One such method consists of using morphological operations on the image and using vertical and horizontal projections for plate localization (Badr et al., 2011). Afterwards ANN is used to classify and recognize characters segmented after edge detection.

Pant et al. (2015) had used actual cropped images of 12 different characters of license plates as the dataset for the training a model to recognize plate characters. For this project, computer generated character images from over 29 different Nepali fonts have been used in combination with a handwritten Devanagari character dataset for 30 characters. The primary motivation for this approach is that acquiring large number of images of characters used in number plates is an inefficient and not a feasible task. Furthermore, the data must

contain reasonable number of images of each character used in the license plate, which is impossible to gather from one city.

Number plate extraction is a critical part of an ANPR system. Generally, a single camera is installed on a lane which captures the vehicles. There are number of challenges faced when these cameras capture vehicles plates; the plates might be skewed, broken or dirty (Lubna, Mufti and Shah, 2021). Lubna et al. (2021) reviewed multiple algorithms for both number plate detection and character segmentation for ANPR systems. They examined number plate segmentation using connected components, using horizontal/vertical projection, using character features, and using boundary information. For recognition methods, they reviewed character recognition methods using template matching and using extracted features.

3. REVIEW OF TECHNOLOGIES

With rapid development of artificial intelligence, various technologies and methods have been established for its applications. Powerful image processing libraries have been created which allows manipulation of images. Development of state-of-the-art object detection models such as YOLO, LSNet, ResNeSt-200, etc. have made locating objects in images and videos without the need of much tinkering with the architecture of the algorithm.

YOLOv3 is one of the best object detection system out there. You only look once (YOLO) is a state-of-the-art, real-time object detection system (Redmon & Farhadi, 2018). They claim that YOLOv3 is an extremely fast and accurate detecting system. The key section in this project where this algorithm was used is localizing the license plate on the vehicle image. The data collected was 153 images of different vehicles with different color Nepali number plates annotated. Even with this small amount of data, the trained model had 80%+ accuracy in detecting the plates in the test dataset.

Various image processing and machine learning libraries exist in Python; thus, it was chosen as the main programming language for the whole project. For processing the license plate image, OpenCV library was chosen (OpenCV, 2021). An alternative is, the Scikit-Image but it is slower compared to the OpenCV as its written in C/C++. A better performance was also observed in median filter which was used as the blur to remove noise from the plate image.

Character recognition was a crucial part of the project. The two machine learning algorithms chosen for this task was Convolutional Neural Network (CNN) and Support Vector Machine (SVM). They are classification algorithms which are great for classifying characters from image. CNNs uses filters to extract features from the image like edges and corners (Ghosh et al., 2019). Then the classes are predicted using an activation function, generally sigmoid for binary classification and *softmax* for multi-class classification. Whereas SVMs, first plot each data item in *n-dimensional* space where *n* is the number of features of the item, then a hyperplane that differentiates the classes accurately is located. After training and testing both algorithms, with different configurations and hyperparameters, CNN ultimately showed higher performance than SVM. While the accuracy differed by a small margin, the time to train the model was significantly lower in CNN. Thus, CNN was used as the character recognition algorithm for detecting segmented characters from a license plate.

TensorFlow is an open source platform for machine learning (TensorFlow, 2021). It consists of many state-of-the-art models and allows easy development and deployment of machine learning applications. It allows building and training of models through a high-level API known as Keras. Here the sequential approach is used from the Keras API to

build and train a CNN model for character recognition. Scikit-learn is a library which offers simple and efficient tools for predictive data analysis. It is open source and is built on NumPy, SciPy and Matplotlib. It provides many classification, regression and clustering algorithms like Linear Regression, Logistic Regression, Support Vector Machines etc. The *k*-means clustering algorithm is used from Scikit-learn to extract dominant colors from a license plate.

As the whole system was to be deployed with a web interface, for seamless transition of languages Flask, a framework written in Python, was decided to be used. Flask is a microframework which aims to keep the core extensible and simple (Flask Documentation (2.0.x), 2021). Google Colaboratory allows writing and execution of Python in the browser with free GPU access. Precisely, it is a free Jupyter notebook environment the runs on the cloud. They require little to no configurations for use and have many machine learning libraries like TensorFlow already installed. It was used extensively throughout the project for training and testing machine learning models due to hardware limitations in the local environment.

4. METHODOLOGY & DESIGN

1. System Architecture

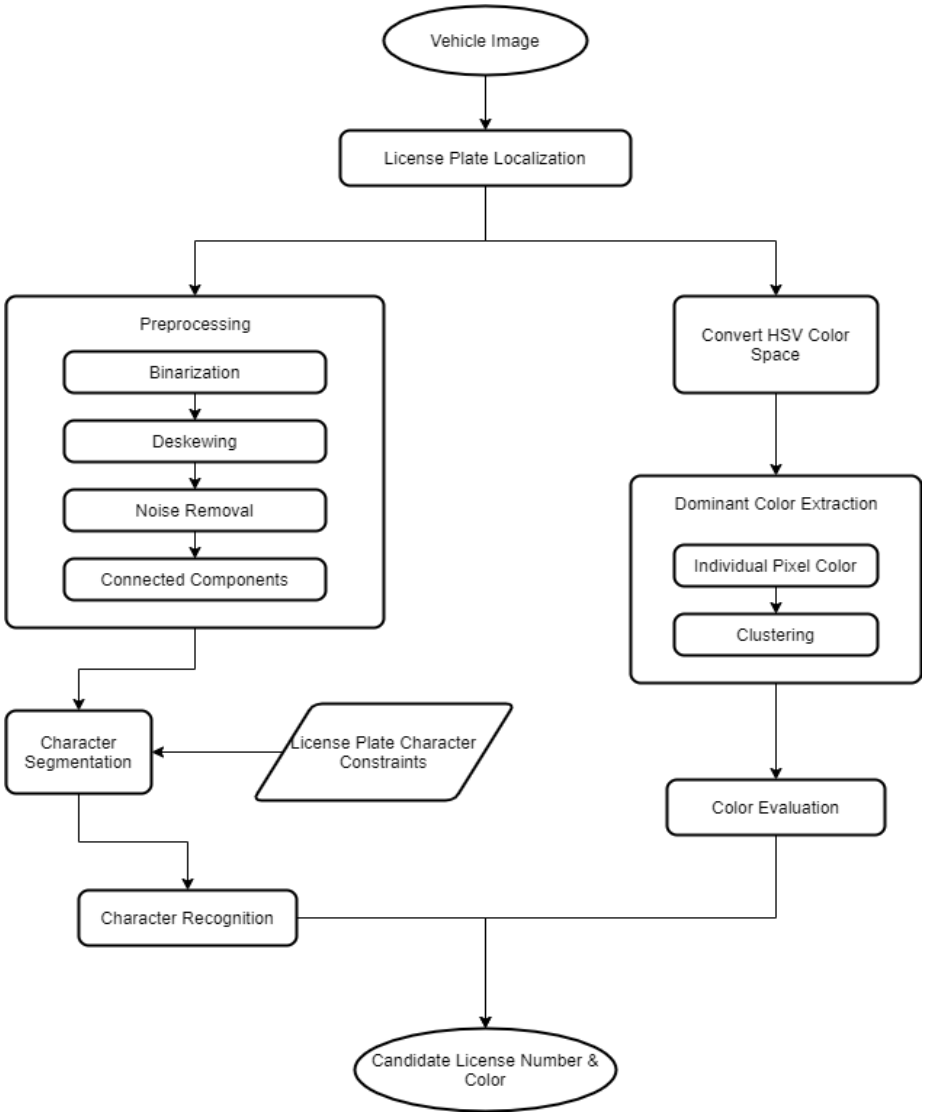


Figure 4: System Architecture for License Plate Recognition

Fig. 4 depicts the system architecture implemented for the whole license plate detection and recognition system. The whole process can be divided into three major steps: License plate localization, Image preprocessing and character segmentation and character training and prediction. The first part deals with detecting and locating the license plate on the input image frame. After the localizing of the plate, it is cropped, and is used for two purposes: character recognition & color detection. Further preprocessing is done of the number plate image to fine tune important details for better segmentation of the characters. Each character is then segmented from the plate and the extracted character is stored as a separate

image for Optical Character Recognition (OCR). The extracted characters are passed one by one to a trained OCR engine, which predicts the character in the image. For color detection of the plate, the unprocessed plate image is converted to the HSV color space. Then three dominant colors from the HSV image are extracted. Finally, the detected colors are evaluated against a range of color values to output the color of the plate.

2. Development Methodology

Agile refers to a set of software development approaches that encourage flexible planning, incremental development, and continuous delivery. It is an iterative and incremental based development software process, where the project needs, and requirements are changeable according to customer needs (Sharma, Sarkar & Gupta, 2012). The overall goal is to improve customer satisfaction by collaborating directly with consumers and incorporating their comments throughout the development process. One of the major principles of Agile methodology is that changes in the project requirements are permitted, regardless if they occur late in the development process. Agile procedures take advantage of change to help customers gain a competitive advantage. Waterfall methodology involves identifying requirements in advance and delivering projects in stages. It is useful for projects with a set scope, but it is prone to failure if demands alter later in the development process. This project follows the Agile method of software development.

1. Kanban & Adaptive Software Development (ASD)

Kanban and ASD are two agile methodologies which provide unique set of workflow and practices (What is agile development, 2021). Kanban is a method for organizing product development that focuses on continuous delivery. To help teams collaborate more efficiently, Kanban employs a pull-based workflow paradigm. Kanban's three key ideas are to see what you are doing right now, limit the quantity of work in progress, and improve flow. ASD focuses on rapid software development and evolution. This method involves a series of cycles of speculation, collaboration, and learning. Mission-focused, feature-based, iterative, timeboxed, risk-driven, and change-tolerant are some of the key characteristics of an ASD methodology. The development of this project was done with a mixture of both the Kanban and the ASD approach. Using the iterative mission-focused approach of ASD along with the flexibility and the visualization of Kanban, a hybrid method was used.

5. IMPLEMENTATION & TESTING

The whole system architecture is shown in Fig. 4. Briefly, the major processes can be described as: First of all the license plate on the image is located and extracted, then various image preprocessing techniques are applied to enhance the regions of interest and remove unnecessary areas like noise, this is followed by the character segmentation process which extracts the characters from the image, finally the extracted characters are the fed to an OCR engine for character recognition. The license plate image before preprocessing techniques have been applied to it is also used for color recognition of the plate. Testing for models is done immediately after training so that parameters can be configured, or model architecture can be changed. After this internal testing and a decent accuracy of the two models, the system is deployed using Flask and is accessed as a REST API. All the steps and techniques for the number plate recognition system are described in detail in the following sections.

1. License Plate Localization

Figure 5.1.1: An image of a car with Nepali License Plate

There are various methods for localizing an object in an image. For locating a license plate in an image, the aspect ratio of the license plate can be used. Another technique that can be used is to use state-of-the-art object detection systems. They can be configured accordingly to detect custom objects in an image. YOLOv3, a state-of-the-art object detection system is used in this project to locate the license plates on the image. It is a single network that is trained to predict both the bounding box of an object as well as the object class. It is an extremely fast network, which processes images in real-time at 45 frames per second. Thus, a dataset was prepared that contains the Nepali license plate along with the annotations file.

The annotations were used on Darknet, a project used to retrain YOLO pretrained models. The steps for plate localization are detailed in the sections below.

1. Data Collection

To train the YOLO model, images of different colors of cars, motorcycles, etc. with various license plate of different colors, structure and layout were collected. 172 different valid images were collected finally. Images were collected online from car and vehicle resale websites as well as real images of vehicles were captured. After the collection, each image was annotated by drawing a bounding box around the license plate in each image. For yolo, the coordinates of the box are stored in a text file, with the same name as of the original image. For the annotation process, a program known as LabelImg created by Tzulatin in 2015 was used. It is a graphical image annotation tool which labels object bounding boxes in images. Using this desktop application, annotations can be saved in YOLO format (text format) which is compatible with Darknet config file input. The dataset was divided into 138 images for training and 34 images as the validation dataset, corresponding to an 85-15 split due to lower number of data.

2. Model Training

There were changes made to the configuration file used by the model. The number of batches was set to 64 and the number of classes to 1. Batches is the number of labeled images that are utilized to generate a gradient and update the weights using backpropagation during the forward pass. The batch is subdivided into blocks, whose number is given by subdivisions. Images of a block are computed parallel on the GPU. The number of maximum batches and steps were set according to the number of classes.

$$\max_{batches} = num_{classes} * 2000$$

Steps is a checkpoint (number of iterations) at which scales will be applied, and scales are coefficients that will be multiplied to the learning rate at this checkpoint. The scales and the steps value determine how the learning rate will change during training as the number of iterations increases.

$$steps = [80\%\ of\ \max_{batches}, 90\%\ of\ \max_{\text{batches}}]$$

The learning rate of 0.001 was used for the training. The training was done on the split training dataset on *Google Colaboratory* while using GPU. After the completing, the loss was reduced to less than 0.06. After testing, the trained model predicted approximately 90% images accurately. As this configuration resulted in a great accuracy, with no overfitting,

no further changes were made to the parameters as no significant performance gain was observed.

Figure 5.1.2: Cropped License Plate

Fig. 5.1.2 is the result of using the trained model to locate the license plate present in the Fig. 5.1.1 It can be observed that the model successfully localizes the Nepali Number Plate in an image.

2. Image Preprocessing

Pre-processing the resultant license plate image is the next step for this system. The goal of image pre-processing is to improve image data by eliminating unwanted distortions, and noise and enhance the image features that are important for the further steps (Sonka, Hlavac and Boyle, 1993). Pre-processing entails specific image operations to enhance the region of interest, so it must be tailored to each situation separately. Image pre-processing might have a significant positive impact on the quality of feature extraction and image analysis results (Krig, 2018). There may be artifacts like noise, skewness, etc. in the image during pre-processing that need to be rectified before feature measurement and analysis. Some image pre-processing methods for enhancements like sharpening, color balance and blurring filters. They are employed to optimize the specific features rather than resolving issues. Multiple techniques have been applied to the license plate image before character segmentation, some are essential like binarization whereas some like dilation enhance the letters by making them bolder. The details of the image preprocessing techniques used are in the sections below.

1. Image Binarization

The process of converting a picture to black and white is known as image binarization. A threshold is chosen in this method to classify certain pixels as black and others as white. However, the key issue is determining the proper image threshold value. Choosing the best threshold value can be difficult or impossible at times. This problem can be solved with

Adaptive Thresholding. However, even better thresholding can be achieved by using the Value attribute of an image converted to HSV.

$$R = no.of\ rows\ in\ image$$

$$C = no.of\ columns\ in\ image$$

$$threshold = \frac{\sum_{j=1}^{C} \frac{\sum_{i=1}^{R} V_{ij}}{C}}{R}$$

The mean of the Value column of the whole image is used as the threshold to binarize the license plate image. The equation above shows how the threshold was calculated. The sum of *Value* column of each row is calculated, then each row is divided by the number of *columns.* The values are then summed up row wise, which then results in scalar value. That scalar value is divided by the number of rows to get the threshold based on brightness of image i.e. Value.

Figure 5.2.1: Binarized License Plate

Pixels that represent useful information in a binary image are called foreground pixels, while pixels that do not represent helpful information are called background pixels. In this project, foreground pixels were set to be white and the background to be black. Fig. 5.2.1 is the binarized output of the license plate image in Fig. 5.1.2. The image is first converted to grayscale for better binarization. All the white pixels have value 1 whereas the black pixels have value 0.

2. Median Filtering

Figure 5.2.2: After applying Erosion and Median Blur

Median filtering is a simple and effective approach for removing noise. Its shot noise removal capabilities are extremely strong. Shot noise is made up of isolated numbers with a sharp spike. The median filter is effective for suppressing isolated noise without distorting sharp edges. The median filter replaces a pixel with the neighborhood's median. It is a non-linear filter that replaces a pixel's gray value with the median of its neighbors' gray values. It is effective in removing salt-and-pepper noise from the image. Fig. 5.2.2 is the result of applying a median blur to the binary image in Fig. 5.2.1.

3. Morphological Operations

Image morphological operations are used to eliminate flaws during segmentation, and they usually work with binary images. Kernel is a structuring element which moves over each pixel in the original image to create a pixel in the new processed image. Two major image morphological operations are erosion and dilation. Erosion erodes away the boundaries of the foreground object in a binary image, typically in white. As the kernel, moves across the pixels, if a single pixel under the kernel around a pixel is 0, it is eroded. Dilation is the opposite of erosion. In dilation, if a single pixel under the kernel around a pixel is 1, it is set to 1, thus dilated. Image erosion helps in removal of noise, but it shrinks the required objects in the image. To increase the size of the objects after erosion, dilation can be used.

Figure 5.2.3: Dilation and Deskewing applied to the image

Fig. 5.2.2 shows the effect of erosion applied to the plate image. It is observed that much of the noise has been removed. After erosion, dilation has been applied to close gaps in the letters, as shown in Fig. 5.2.3.

4. Deskewing

Because the pictures of the vehicles are captured while they are moving, there is a good chance that the image will be skewed. The skew in the image is also caused by a discrepancy in the camera location and the vehicle's license plate, as well as rough roads and the consequent vibrations of the vehicle (Jin et al., 2012). Deskewing is the technique of straightening a crookedly scanned or written image, such as one that is slanted too far in one direction or is misaligned. The amount of rotation required to align an image horizontally and vertically is known as skew. Skew is measured in

degrees. Deskewing is the process of removing skew from an image by rotating it in the opposite direction by the same amount as the skew (Deskewing | Document Imaging 2021). As a result, the text runs horizontally and vertically across the page rather than at an angle. It is done by calculating the initial skew of a rectangular area in the image and then rotating the image accordingly to fix the skew. After applying deskewing on the image in Fig. 5.2.2, the resultant image in Fig. 5.2.3 has been rotated by an angle of *2.45* degrees.

3. Character Segmentation

Character segmentation is the process of extracting a rectangular area enclosing each character from the preprocessed and localized license plate image. Histogram projection can be used to identify gaps between lines and characters. The characters are extracted from the image by analysis of their projection. Thus, a projection analysis is carried out, first to extract lines in the localized plate by horizontally projecting it, then to locate individual characters by applying vertical projection on each line. The details of the strategy applied, and their workings are explained the following sections.

1. Histogram Projection

The histogram of pixel projection consists of finding the upper and lower limits, of the characters in an image. A histogram of an image can be projected either vertically or horizontally. Horizontal histogram projection includes identifying the number of foreground pixels along the image's rows. The resulting array is size of the height of the image. After plotting the histogram, higher peaks can be related to high number of foreground pixels in a row, and conversely low peaks imply low number of foreground pixels in a row. Vertical histogram projections include counting of the number of foreground pixels along the image's columns. The resulting array is of size equal to the width of the image. After plotting the histogram, higher peaks can be related to high number of foreground pixels in a column, and conversely low peaks imply low number of foreground pixels in a column.

2. Line Level Segmentation

As Nepali number plates can include multiple rows of characters, each row must be extracted from the plate. After applying horizontal projection, the rows that represent the text in a line have high number of foreground pixels, thus they represent the peaks in the histogram. Whereas the lower peaks in the histogram are the rows of the image which

represent gaps in-between the lines of text. Thus, rows which correspond to lower peaks can be selected as the segmenting lines to separate the lines of text.

$$R = no.of\ rows\ in\ image$$

$$h_{projection} = \sum_{i=1}^{R} I_{ij}$$

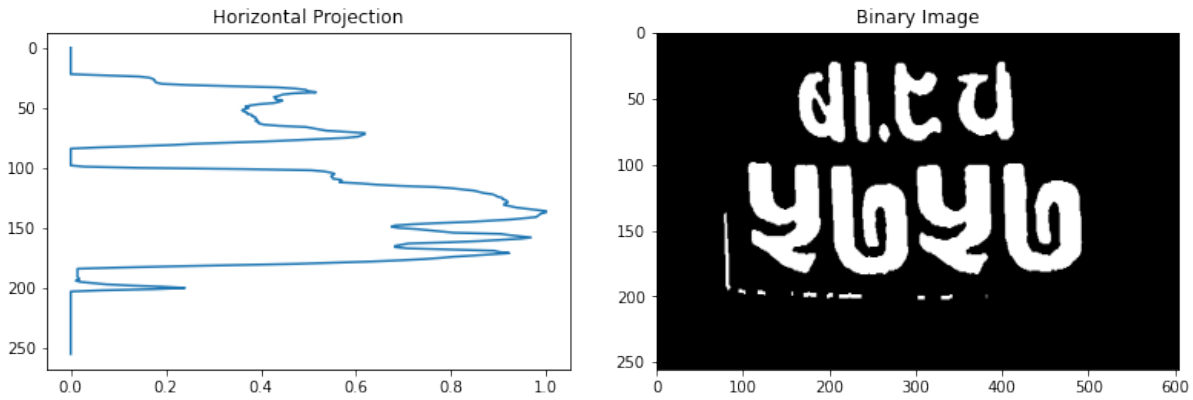


Figure 5.3.1: Horizontal Projection of Foreground Text

In Fig. 5.3.1, (left) shows the histogram of sum of white pixels in each row, (right) shows the image which is horizontally projected. The gaps between row can be clearly identified by the lower values in the histogram. By using, an optimum threshold of the horizontal projection values, all rows with sum of foreground pixels below the threshold are set to 0.

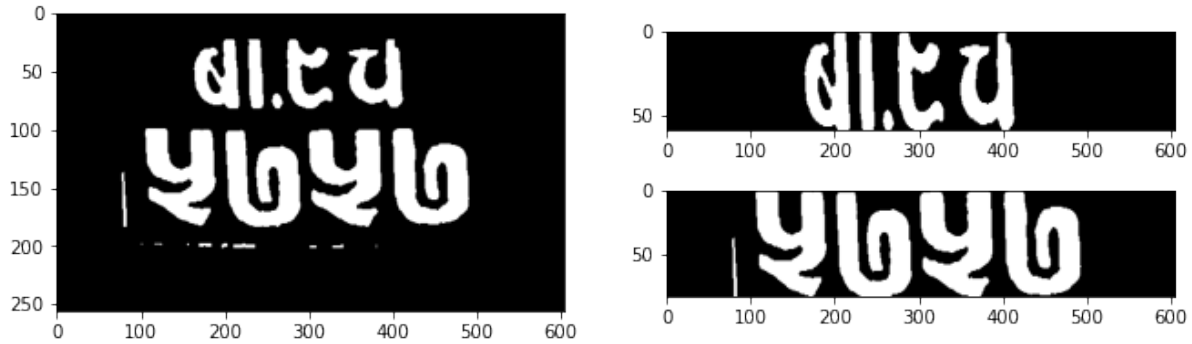


Figure 5.3.2: Line Level Segmentation using Horizontal Projection; (left) remove rows < threshold, (right) Segmented lines in two images

Fig. 5.3.2 (left) shows the result of such a process, where all rows are removed whose sum of all pixel values are less than the threshold value. The threshold value, that is calculated using a heuristic approach, is given by:

$$threshold = \frac{\max(h_{projection}) - \min\ (h_{projection > 0})}{10}$$

The horizontal projection for the new image is recalculated. The lines from the image are then extracted by looping over each row of $h_{projection}$, and comparing each row to the next. If the current row of $h_{projection}$ is equal to 0 (background pixels), and the next row in $h_{projection}$ is greater than 0 (foreground pixels), it indicates the start of a line. Conversely, if the next row of $h_{projection}$ is equal to 0 (background pixels) and the current row in $h_{projection}$ is greater than 0 (foreground pixels), it indicates the end of the line. Row number for each start and end of line are stored separately, thus resulting in the location of line in the original image.

The pseudocode in Appendix F describes the algorithm for segmenting lines from the image. The lines are stored in *lines* list by extracting all rows from image, where the row's starting and ending index are stored in the *line* list. A heuristic approach is applied to remove noise lines from the image. As there are maximum of 3 rows in a Nepali number plate, a constant value of 6 is calculated, which is then used to divide the number of rows.

$$num_{lines} = 3$$

$$\min_{height} = \frac{num_{rows}}{num_{lines} \times 3}$$

This value is the threshold height, if the height of a line is less than the threshold it is not extracted from the image.

3. Character Level Segmentation

For segmenting each character from a line of text, vertical histogram projection can now be used. After vertically projecting the image, columns that represent the characters have high number of foreground pixels thus are represented by the higher peaks in the histogram. Whereas the lower peaks in the histogram correspond to gaps between the characters. Columns which are represented by lower peaks in the histogram can then be used as the segmenting lines to separate the characters in the line of text.

$$C = no.of\ columns\ in\ image$$

$$v_{projection} = \sum_{j=1}^{C} I_{ij}$$

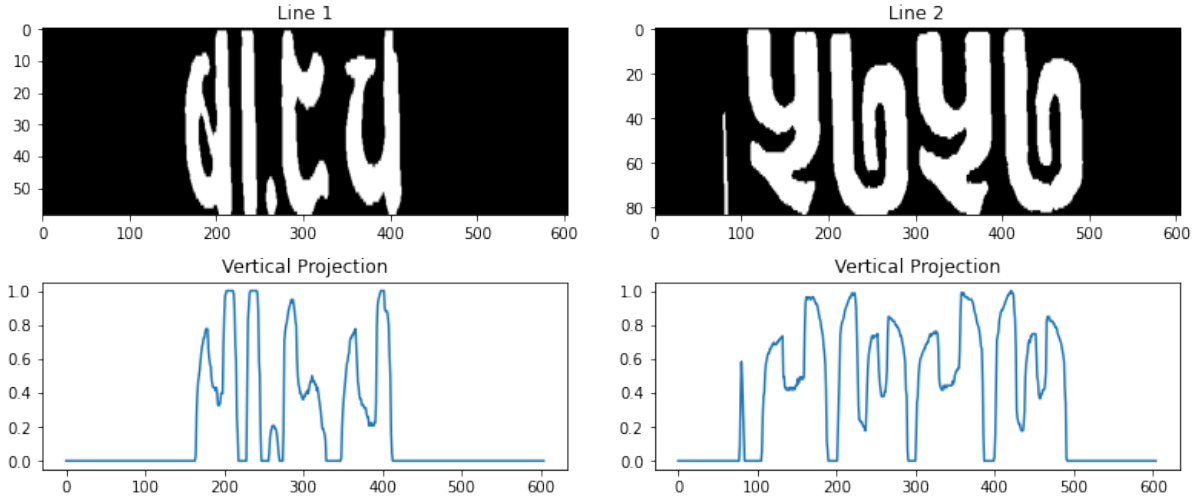


Figure 5.3.3: Vertical Projection of segmented lines

In Fig. 5.3.3, lower graph shows the histogram of sum of white pixels in each column of the image of a plate line above it. The gaps between columns can be clearly identified by the lower values in the histogram. These lower peaks correspond to the in-between gaps of characters. By using, an optimum threshold of the vertical projection values for each line, all columns with sum of foreground pixels below the threshold are set to 0.

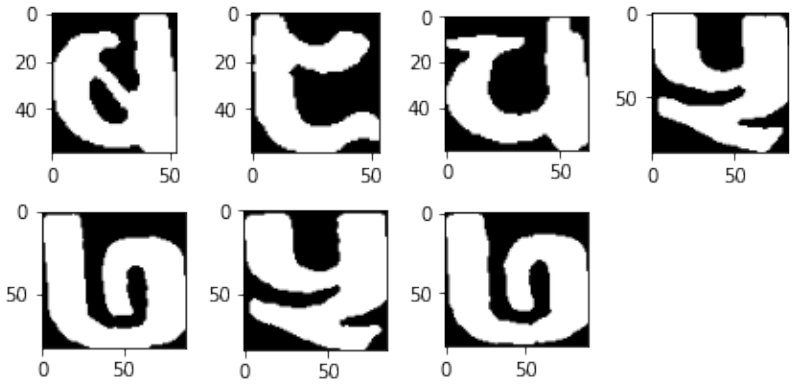


Figure 5.3.4: Character Segmentation after vertical projection

Fig. 5.3.4 shows the result of segmenting an image after applying vertical projection to it. Initially, all the columns are removed whose sum of foreground pixels is less than a threshold value. The threshold value is calculated separately for each line images and a heuristic approach is used to derive a suitable formula. The threshold value is given by:

$$threshold = \frac{\max(v_{projection}) - \min(v_{projection>0})}{10}$$

After this step, vertical projection values for the lines are recalculated. The individual letters from the image are then extracted by looping over each element of $v_{projection}$, and comparing each column to the next. If the current column in $v_{projection}$ equals to 0

(background pixels), and the next column of $v_{projection}$ is greater than 0 (foreground pixels), it indicates the start of a character. Conversely, if the next column of $v_{projection}$ is equal to 0 (background pixels) and the current column in $v_{projection}$ is greater than 0 (foreground pixels), it indicates the end of the character. Column number for each start and end of the character are stored separately, thus resulting in the location of a character in the original image. This process is repeated for each line in the image.

The pseudocode in Appendix F describes the algorithm for segmenting characters from a list of line images. The individual letters are stored in *letters* list by extracting all columns from image, where the column's starting and ending index are stored in the *letters* list. A heuristic approach is applied to remove noise from the image during segmentation. As there are typically 4 letters in a line in Nepali number plate, a constant value of for each line is calculated, which is then used to divide the number of columns.

$$num_{letters} = 4$$

$$\min_{width} = \frac{num_{columns}}{num_{letters} \times 3}$$

This value is the threshold width, if the width of a character is less than the threshold it is not extracted from the image.

4. Character Recognition

The final stage in the development of an automatic license plate recognition system is the recognition phase. As a result, it completes all the procedures that preceded the acquisition of the image, followed by the location of the plate, and preprocessing and the segmentation. The character recognition algorithm, also knowns as the Optical Character Recognition (OCR) engine, must predict characters from the images collected at the end of the segmentation phase. OCR is the conversion of text images into machine-encoded text. The machine learning model that will be utilized for this recognition must be capable of reading an image and rendering the appropriate character. Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) are two of the most popular machine learning algorithms for character recognition.

Neural Networks (NNs) comprise of neurons, typically multiple neurons, that have learnable weights and biases. Each neuron in the network receive some input and they perform a dot product. They transform an input through a sequence of hidden layers. Hidden layers consist of a collection of neurons that are fully connected to the neurons present in the previous layer. The output layer, which is the final fully connected layer, represents the class scores in classification problems (CS231n CNNs for Visual

Recognition, 2021). Regular neural networks do not scale well with image classification problem as the image has to be scaled down two a 1-dimensional vector before training the model. Due to this, the number of trainable parameters increase significantly. A special class of ANNs, known as the Convolutional Neural Network (CNN) are prevalent in image and video processing.

1. Convolutional Neural Networks (CNNs)

CNNs are made up of three main types of layers: convolutional layers, pooling layers, and fully connected layers (NNs). Typically, multiple layers of each are stacked to create a CNN architecture. Usually, a CNN takes an input of a 2-D image that has *m* rows, *n* columns and 3 channels. The input then goes through a series of layers for processing (Wu 2017). Features are first extracted from raw inputs which are then classified by the fully connected layers. The details for the layers in CNN are given below:

A. ***Convolution Layers***: They perform the core operation of the feature extraction process. The convolution operation's primary goal is to extract high-level characteristics from the input, such as edges. During the convolution operation, the *kernel/filter* shift over the input data and perform elementwise multiplication. For every location, that the kernel slides over, it performs the same operation and outputs a single value. The kernel shifts to the right with a value called *stride* until it reaches the width, after which it slides down *stride* number of rows. This operation is repeated until the kernel reaches the last row. This brings all the information into a matrix sized less than the input matrix. The dimensionality of the features can be configured to be either increased or to remain same. This is done by setting the *padding* of the layer, which adds blank or empty pixels, increasing the size of the image. These layers typically involve the *ReLU* activation function to convert all negative values to 0.

B. ***Pooling Layers***: They also reduces the size of the features. The aim is to reduce the number of parameters and computation in the network through dimensionality reduction. They also have *stride* and *kernel* values that can be configured. There are two types of pooling: average and max. Average pooling involves averaging the value of all pixels under the kernel, whereas max pooling involves extracting the maximum value from all the pixels under the kernel. They also perform noise suppression along with dimensionality reduction.

C. ***Fully Connected Layers***: Also known as dense layers, they are regular layers of neurons in a NN. As NNs, require a *1*-dimensional vector input, the extracted

features are first *flattened* into a 1-D vector. The flattened vector is then fed to the neural network. Activation functions like *softmax* and *sigmoid* are used to classify the outputs.

2. Dataset Collection

Nepali License Plate uses 30 different characters from the Devanagari script. They comprise of 10 numerical and 20 alphabetical characters. There is no standard for the font used in the creation of Nepali number plates. A wide variety of fonts are used in the plates, making it difficult for ANPR systems to recognize characters. To reduce error due to such circumstances, a mixed dataset comprising of both computer-generated characters and handwritten characters is used. All 30 characters from 29 different Nepali fonts are used to generate 580 different images of each character with dimensions *52 x 52*. The character images are differentiated by rotating the characters, dilating the characters, and increasing their font size. 20 different images of a single character are obtained by applying the above operations. And when repeated for each different font, the total length of resultant dataset equaled to 17,400 images. All images binary images with, black color as the background and white color as the foreground.

To further vary the dataset, a handwritten Devanagari dataset is used (Devanagari Character Dataset, 2018). It consists of 58 different characters from Devanagari. Only the required characters are used. They are binarized, then their color is inverted and are finally resized to dimensions *52 x* 52. A dilation is applied to all the images which are stored separately. The number of images in the whole dataset is now 30,855. The dataset is then split into training and testing sets with an 80%-20% split. As a result, the obtained data composes now of 30 classes and for each class there are 950-1050 images of *52 x 52* pixel dimension of JPG formats.

3. Model Architecture

Initially multiple configurations were applied to obtain a decent accuracy. ReLU activation is used in all convolutional and dense layers except the output layer, where *softmax* activation is used. The shape of the input image is 52 x 52 x 1. Each outer layer consists of 2 convolutions layers followed by a max-pooling layer. They are then followed by dense layers and finally a *softmax* output of 30 classes. After training and testing the model, steps were taken to improve the model by reducing overfitting and underfitting:

Underfitting (Bias): To reduce underfitting in the network, the network was made wider by adding more layers and deeper by increasing the number of filters in the

convolutional layers. This allows the model to learn more complex features from the characters.

Overfitting (Variance): To reduce the variance multiple strategies were applied:

- *Batch Normalization*: This involves normalizing the activations of the neurons reducing the change in the distribution of inputs. It was performed after every convolutional layer.
- *Kernel Regularization*: Regularization is used to penalize the loss function; this increases loss which results in smaller weights making the model simpler. A L2 regularization with hyperparameter 0.0005 was added to all convolutional layers of Layer 2-5.
- *Dropout Layers*: After each fully connected layer except the outer layer a dropout layer with hyperparameter of 40% was added. This reduces the dependence on specific neurons of the network to generate output by dropping 40% of the neurons.
- *Variable Learning Rate*: Variable learning rate was added to model which decreased the learning rate by a factor of 0.5. As the performance of the model stagnated, the learning rate was decreases. This resulted in a significant gain in performance of the model.

Table 5.4.1 shows the final architecture for the model. It includes 5 *outer layers* which each consist of two convolutional layers followed by a max-pooling layer. The padding for every convolutional layer is set to *same* and stride to 1. After all the outer layers, a *flatten* layer is included to flatten the input for the dense layers. There are three dense layers and a fully connected output layer with *softmax.*

Layer	Type	Kernel Size	Filters	Activation	Output Shape
Layer 1	Convolution	7	64	ReLU	52 × 52 × 64
	Convolution	7	64	ReLU	52 × 52 × 64
	Max Pooling	2			26 × 26 × 64
Layer 2	Convolution	5	128	ReLU	26 × 26 × 128
	Convolution	5	128	ReLU	26 × 26 × 128
	Max Pooling	2			13 × 13 × 128
Layer 3	Convolution	5	256	ReLU	13 × 13 × 256
	Convolution	3	256	ReLU	13 × 13 × 256
	Max Pooling	2			6 × 6 × 256
Layer 4	Convolution	3	512	ReLU	6 × 6 × 512
	Convolution	3	512	ReLU	6 × 6 × 512
	Max Pooling	2			3 × 3 × 512
Layer 5	Convolution	3	1024	ReLU	3 × 3 × 1024
	Convolution	3	1024	ReLU	3 × 3 × 1024
	Max Pooling	2			1 × 1 × 1024
Layer 6	Dense			ReLU	1 × 2048
Layer 7	Dense			ReLU	1 × 4096
Layer 8	Dense			ReLU	1 × 1024
Layer 9	Dense			Softmax	1 × 30

Table 5.4.1: CNN Architecture for character recognition

4. Performance & Results

After training the model for 29 epochs the model predicted letters with a training accuracy of 99.98% and a testing accuracy of 94.11%. There was a significant gain in performance by introducing variable learning rate. The average precision for all the classes was 94.44% and the average recall was 95.02%. The f1-score of 94.15% was measured. Table 5.4.2 shows all the metrics, on which the model has been evaluated, with their results.

The lowest f1-score was for the class ‘GA’, about 83% (see Appendix B). This might have been caused due to the similarity of the character to ‘BA’. It can be observed that the average f1-score of the model is 94.15%, thus the model predictions consist of low false positives and low false negatives.

Training Accuracy	99.98%
Validation Accuracy	94.11%
Precision	94.44%
Recall	95.02%
F1-Score	94.15%

Table 5.4.2: Performance of model. Detailed classification report for each class can accessed in the Appendix

5. Color Prediction

Nepali license plates are categorized into six types based on ownership and service type. This classification of vehicles is indicated by the color of the license plate of the vehicle. Detection of color plate might help if the system is failed to extract correct vehicle number. The color of a license plate might be altered by the image quality and illumination. Thus, a simple color comparison is not viable to detect the color of the license plate.

Plate Color	Classification
Red	Private
Black	Public
White	Government
Yellow	National Corporation
Green	Tourist
Blue	Diplomatic

Table 5.5.1: Classification of Vehicles by service type and their license plate colors

Table 5.5.1 shows the colors of the Nepali license plates classified based on vehicle service type and ownership. Three dominant colors from the localized plate image are extracted first, using a clustering algorithm. *k*-means (Pedregosa, 2011) is used as the clustering algorithm. Then, each of these dominant colors are checked against a range of predefined color values for each corresponding license plate color. The steps of this algorithm are explained in the sections below.

1. HSV Color Conversion

HSV (hue, saturation, and value) is a color model representation that is intended to be more closely aligned with how human vision perceive color-making properties. In this approach, colors of each hue are organized in a circle around a central axis of neutral colors that vary from black, at the base, to white, at the top. Unlike RGB (red, green, and blue) color space, HSV model separates the *luma* (image intensity), or the brightness in an image, from the *chroma* (color information). Due to this property, HSV model is superior to RGB models in many conditions where color extraction is necessary.

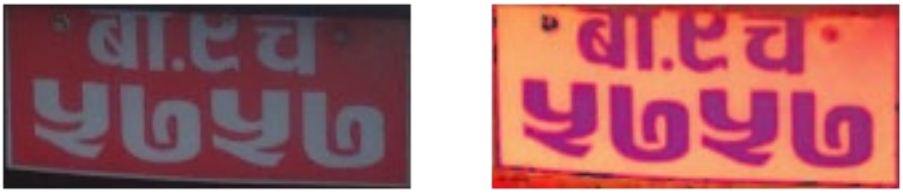


Figure 5.5.1: License plate image (left) converted to HSV (right)

Hue value in HSV ranges from 0 to 360, the saturation value ranges from 0 to 1 and Value attribute has a range of 0 and 1 also. An RGB image can be easily converted to the HSV space. Fig. 5.5.2 depicts a localized license plate converted to the HSV color space.

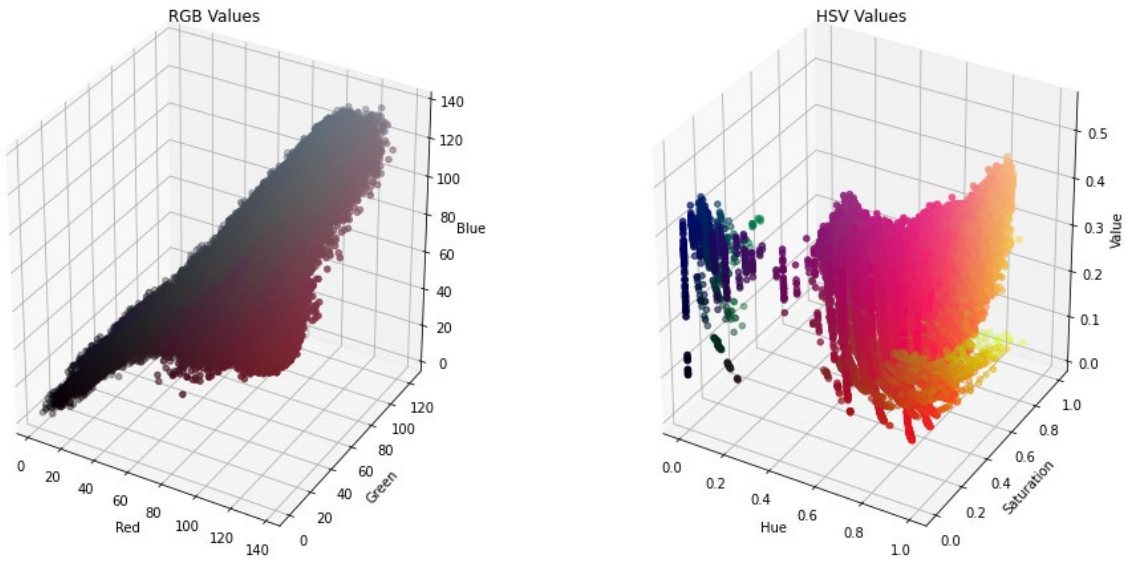


Figure 5.5.2: The color values of both images plotted in 3d

In Fig. 5.5.2 it can be observed that, the color information is better separated in the HSV plot (right). Also, the plot reveals that better clusters may form in the HSV model as the pixel color values have better distance among them over the RGB ones. This is useful as the next step is to extract dominant colors, which uses clustering of the pixel color values.

2. k-Means Clustering

k-means clustering is an unsupervised machine learning algorithm, which given a set of samples $(x_1, x_2, x_3, \dots, x_n)$, where each x is an m-dimensional vector, partitions (clusters) of the n samples into k sets, minimizing a criterion known as within-cluster sum-of-squares (MacQueen, 1967). The number of clusters, k, is required to be specified for this algorithm. It scales effectively to huge numbers of samples and has been employed in many different domains across a wide range of application areas (Pedregosa, 2011). The *k*-means algorithm divides the n samples into k disjoint partitions (clusters) C, which are each described by the mean μ_j of the samples in the cluster. These means are called as the *centroids*, that may or may not be present in the original sample set. The *k*-means algorithm aims to select centroids such that the within-cluster sum-of-squares criterion is minimized.

A *k*-means algorithm starts with choosing k samples from the whole dataset. After this initialization of the centroids, two other steps are iterated:

> **Assignment Step:** Assign each sample to the cluster with the nearest centroid. A sample is assigned to only one cluster even if it could be assigned to two or more of them.
>
> **Update Step:** Recalculate the centroids by taking the mean of all the samples assigned to each previous cluster.

The difference between the values of old centroid and the new centroid are calculated after each iteration. The algorithm stops looping over the two steps when this difference is less than a threshold. The algorithms have converged when the assignments no longer change significantly.

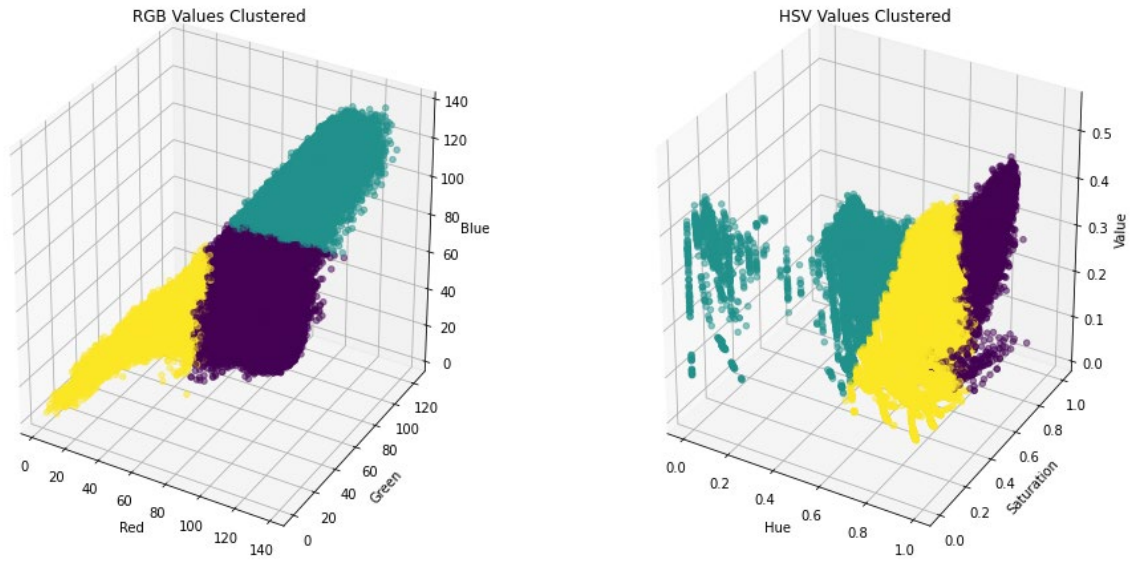


Figure 5.5.3: The color plots after clustering is applied. The different colors represent the clusters.

The dominant colors in an image can be extracted by finding the centroids of the clusters in a *k*-means clustering applied image. Fig. 5.5.3 shows the different clusters after a *k*-means clustering with *k* as 3 is used to cluster the pixel color values in both the RGB and the HSV color space. The centroids of these clusters, not visible in figure, are used as the dominant colors in the license plate image.

3. RGB Color Evaluation

After the three dominant colors from the image have been identified, a simple comparison is made to check where these values lie in a color range. A color range is defined in the RGB model for each of the six license plate colors (Table 5.5.1). The black color is checked for separately because of its presence in almost all the images.

Color	**RGB Color Space**					
	Red		Green		Blue	
	Min	Max	Min	Max	Min	Max
Red	100	255	0	100	0	100
Black	0	100	0	100	0	100
White	100	255	100	255	100	255
Blue	0	100	0	100	100	255
Yellow	100	255	100	255	0	100
Green	0	100	100	255	0	100

Table 5.5.2: Color Range values for the license plate colors

Each dominant color is checked for in the color ranges specified in Table 5.5.2. If any of the three-dominant color from the localized plate image falls in a range, the plate is assigned that color. As the centroids, or the dominant colors, have been located they are converted from the HSV color space to the RGB color model. Fig. 5.5.6 shows the dominant colors, captured from both the HSV model (converted to RGB) and the RGB model.

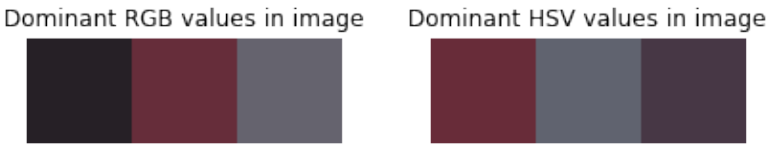


Figure 5.5.6: Dominant colors from both RGB and HSV clusters visualized in the RGB color space

The pseudocode in Appendix F, shows a simple implementation of the algorithm to evaluate a color. First all the dominant colors are converted to RGB from the HSV color space. They are then scaled to a range of 0 to 255. Then each dominant color is checked against the range in Fig. 5.5.5. As soon as a color is matched to a range, the color is set to the result and broken from the loop. Additionally, the black color check is only done when no previous colors are found in the other ranges. It is done because in most of cases, black is observed to be a dominant color in the localized images. The result for the colors in Fig. 5.5.6 after applying this algorithm is *red* which has been correctly identified.

6. Deployment

The whole project is deployed on a web application. Flask, a Python framework, is used here to embed the machine leaning model into a web application. The application is hosted on a local server. The app uses a single end point which takes in an input image and processes the image and returns the relevant output. The input is read from the *index.html* and the results are output on *predict.html*.

Initially the system took a long time to calculate process and predict a plate. Some optimization was performed to reduce the time taken like:

- The model weights for both character recognition and license plate localization are saved and loaded only once before the app execution. This reduce the time as before model was loaded on every request.
- During color prediction image is now resized to smaller dimensions to speed up the *k*-means clustering of the colors. This significantly decreased the color recognition time.
- Specific functions and classes were only loaded from all modules now instead of importing the whole module.

7. Testing

1. Application Testing (Black Box Testing)

The application testing was done after a model was improved or new requirements were added. It includes uploading different type of images both correct and incorrect to evaluate the behavior of the product. All the conducted black box tests for the product can be viewed in Appendix B.

Test Case: Detect License Plate on Image		**Test No:** 1	
Description: Application should successfully detect and locate license plate on image			
Prerequisites: The web application must be loaded in a browser			
Step	**Action**	**Expected Result**	**Evaluation**
1	User uploads an image using the Upload button	The image must be uploaded, and a preview must be shown	Pass
2	User clicks on the Predict button	An annotated image with a rectangle around the plate is shown	Pass

Figure 5.7.1: Test No. 1

Fig 5.7.1 shows the result of a test conducted to evaluate the product whether it predicts and locates the license plate on an image successfully.

Test Case: Recognize characters of a license plate		**Test No:** 2	
Description: Application should successfully detect and locate license plate on image			
Prerequisites: The web application must be loaded in a browser			
Step	**Action**	**Expected Result**	**Evaluation**
1	User uploads an image using the Upload button	The image must be uploaded, and a preview must be shown	Pass
2	User clicks on the Predict button	The predicted characters are shown correctly	Pass
Input Plate Number: **BA 3 CHA 8 0 8 4**		Output Number: **BA 3 CHA 8 0 8 4**	

Figure 5.7.2: Test 2

Fig 5.7.2 shows the result of a successful prediction of license plate characters from an image. Further tests for color prediction, incorrect images, incorrect predictions are provided in Appendix B.

2. Internal Testing

Models were tested using the relevant split data sets. API testing was done using Postman, a client which allows to send HTTP request. UI testing was done using Microsoft Edge browser.

Model Testing	**Test 01**	**Test 02**	**Test 03**
CNN Model locates license plate	Pass	Pass	Pass
Locates number plate accuracy (80%+)	Fail	Pass	Pass
CNN Model predicts characters	Pass	Pass	Pass
Predicts the plate color (80%+)	Fail	Fail	Pass
Predicts the plate number (80%+)	Fail	Fail	Pass
API Testing		**Test 01**	**Test 02**
API is connected to frontend		Fail	Pass
Can handle garbage images		Fail	Pass
Predicts and saves annotated images		Pass	Pass
Optimized prediction time		Fail	Pass
UI Testing	**Test 01**	**Test 02**	**Test 03**
Upload button and predict button are functional	Pass	Pass	Pass
Shows relevant error messages on incorrect input	Fail	Fail	Pass
Shows annotated image after prediction	Fail	Pass	Pass
Shows further information (time taken, confidence score)	Fail	Fail	Pass

3. Valid Image Testing

S. No.	Image	Expected Number	Expected Color	Predicted Number	Predicted Color
1		BA 1 CHA 3 9 5 2	Red	BA 1 CHA 3 9 5 2	Red
2		BA 1 4 CHA 7 4 8 0	Red	BA 1 4 CHA 7 4 8 0	Red
3		BA 1 2 CHA 6 9 4 4	Red	BA 1 2 CHA 6 9 4 4	Red
4		BA 1 JA 4 1 3 1	Black	NA 5 4 1 3 1	Black
5		BA 1 JA 6 3 3 3	Black	BA 1 JA 6 3 3 3	Black

6. PRODUCT EVALUATION

The whole project was evaluated using a set of correct and incorrect images. The system successfully recognized and reported incorrect images (images with no license plate). For the correct images, a set of 15 multiple different images having different illumination, orientation, low resolution, distortions and different types and sizes of vehicles with multiple colors of license plates was used. The model successfully recognized the number plates and delivered the color and number of the vehicle accurately during 90% of the tests. That accuracy surpasses the original requirement initially recognized during the planning of the project.

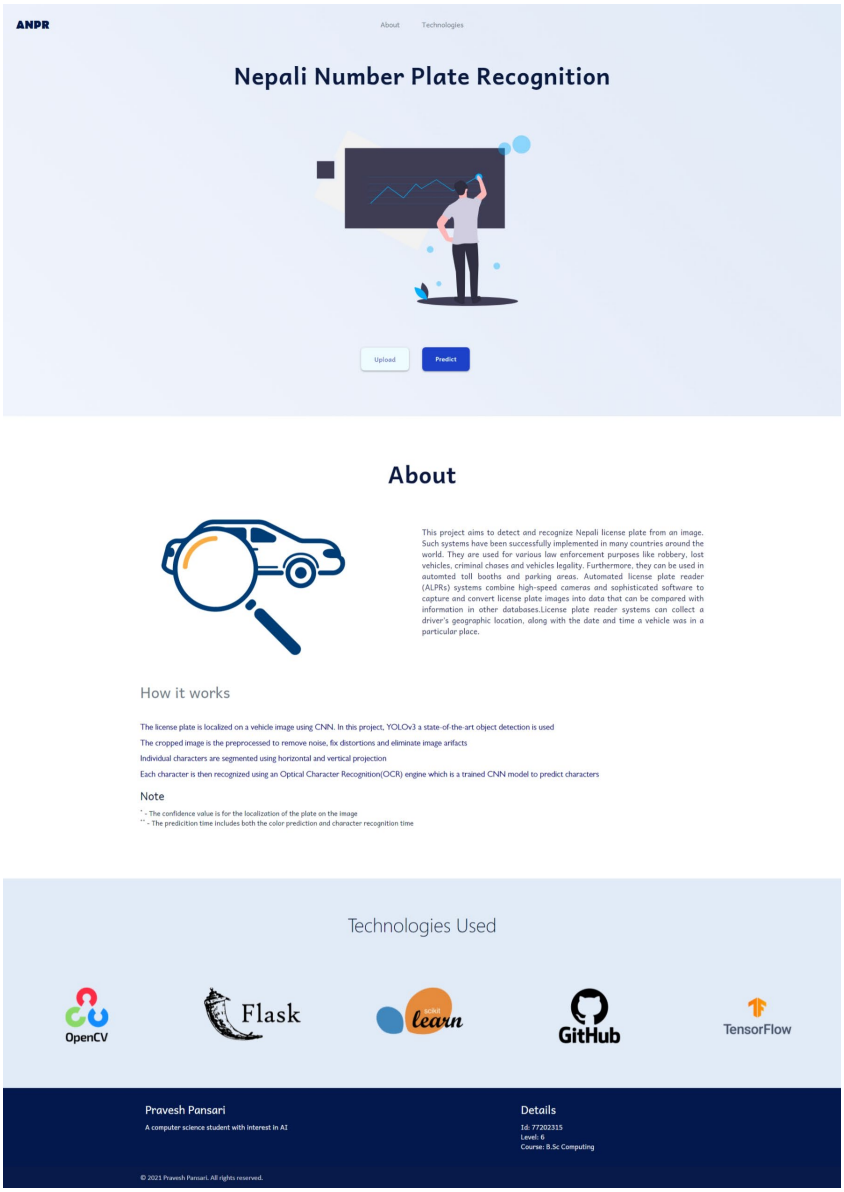


Figure 6: The web interface of the product

The average time of prediction for both the number and color recognition is less than 0.85 seconds, where the average time for number prediction is approx. 0.6s and color recognition is 0.25s. While the system is fast, it can benefit from better hardware and probably design. From the requirements point of view, all the points with *must* and *should* specification of both functional and non-functional requirements have been successfully completed. Majority of all the other requirements have been satisfied and addressed in the project.

The functional requirements were readily addressed and satisfied for this project. Enough data set was acquired for the training of the models. Two CNN models were successfully built and trained on the mentioned datasets. After training, the number plate was successfully extracted, and the characters were successfully recognized. The color of the number plate was also extracted using clustering algorithm and the vehicle number plate type was classified based on the color. The confidence score for locating the number plate is calculated and displayed along with the time taken for prediction. The model is deployed on a REST API and has a simple GUI access to run the product.

Non-functional requirements specified initially have also been addressed. The product is thoroughly tested and validated. Both black box and white box testing is done. All the software, technologies, platforms, and libraries used to develop and deploy the product are open source and free. The code is properly commented and formatted. The final model is deployed on a user-friendly interface. It can identify some details in low quality images like color. Two types of machine learning algorithms are used to build the system, CNN and *k*-means clustering.

7. PROJECT EVALUATION

This project shows the use of unsupervised learning (*k*-means clustering) and supervised learning (CNN) algorithms to create a detection system which can extract the color and number of a license plate.

- The system successfully detects license plate on vehicles in images.
- It localizes and outputs the vehicle registration number.
- The product provides a simple UI to access and use the model
- The accuracy of the model on the testing images was well over 80% (the initial required)

By establishing the above points, we can claim that all the objectives, initially specified in the project proposal, have been addressed and successfully implemented. The project followed the timeline indicated in the Gantt chart (see Appendix C) tentatively. Version control was used to keep track of the project changes and keep backup of the software. GitHub was used to create a remote repository and commits were made when relevant. During development clean coding and modular design was used. Comments were added in the code where necessary and unnecessary code removed. This allowed to making changes later easier due to detailed comments, also modular design meant changes would not break the whole code and can be applied easily to specific parts of code.

As a Kanban strategy was used Trello, a free online board was used to visualize the project phases and progress (see Appendix A). After each testing phase, further development was carried out if required or requirement analysis was done (see Appendix A). The advantage of this technique during this project development was that it allowed the visualization of the progress of the project.

While the development of the project tried to adhere to the dates in the Gantt Chart, some delay was caused in collecting the data during the current situations (lockdown due to pandemic). As going outdoors frequently was not possible, it slowed down the collection of vehicle images. Furthermore, an area of improvement which could be addressed is the web interface. A native desktop application would be much faster but require more resources and time. This project aims to research and deliver a method for a Nepali number plate recognition system, which after all the evaluation and testing, can be specified as a successful attempt.

8. SUMMARY & CONCLUSIONS

1. Summary

License plate number recognition systems have had benefited from recent advancements in artificial intelligence and machine learning technologies. The accuracy of these systems is affected by a variety of factors. In Nepal, due to the lack of standard font and layout, a system that can recognize all different number plates is not feasible. Embossed license plates have been introduced in Nepal, but they are very rarely used and will probably take a long time to spread in the whole country. Poorly illuminated images, low resolution images, highly skewed images and dirty or half visible number plates are some of them. For a practical implementation of such as system, these factors must be kept in mind before installing the cameras and the system.

This project attempts to provide one of the many methods to develop a working ANPR system for Nepali number plates using machine learning. It shows the implementation of both supervised and unsupervised machine learning algorithms. Some limitations for these systems have been recognized as well.

2. Conclusions

This project aims to provide a method to design and develop a robust automatic license plate recognition system for Nepal. In the future, where a standard font and better license plates (embossed) are used, it will become much easier and efficient to implement such systems. The product successfully utilizes CNN and clustering algorithms to localize a Nepali number plate, recognize Nepali characters from a plate and detect its color. The product is deployed using a Python framework providing a simple UI for access. The product has an accuracy of over 94%, even with the nonstandard font and layout of Nepali license plates. Thus, it can be concluded that technologies in artificial intelligence can be applied successfully and feasibly to dynamic scenarios like recognition of Nepali number plates.

3. Future

While gaining a decent accuracy, further improvements can be made to this system like a better character dataset for character recognition, using more efficient techniques to localize a plate and expanding the model to work on videos. Further improvements can be made in the speed of the recognition software, and instead of a web interface, ideally, a native desktop application can be made which is connected to the license plate number database and the cameras. Improvements required for a practical implementation of such as system in Nepal can be summarized as follows:

- Decent cameras around the city which capture vehicle images
- A standardized number plate for the country, with a standard font
- A native desktop application for more efficiency
- Expanding this system to process videos as well

BIBILOGRAPHY

Aha! Labs. 2021. *What is agile development?* [online] Available at: https://www.aha.io/roadmapping/guide/product-development-methodologies/what-is-agile-development [Accessed 21 June 2021].

Badr, A., Abd El-Wahab, M., Thabet, A., and Abdelsadek, A. (2011). CCTV—Automatic Number Plate Recognition System. *Mathematics and Computer Science Series Volume, 38*, p.62-71.

Cs231n.github.io. (2021). *CS231n Convolutional Neural Networks for Visual Recognition*. [online] Available at: https://cs231n.github.io/convolutional-networks/ [Accessed 24 June 2021].

Devanagari Character Dataset. (2018, May 7). [Dataset] [online] Available at: https://www.kaggle.com/ashokpant/devanagari-character-dataset

Friedrich, M., Jehlicka, P., and Schlaich, J. (2008). Automatic number plate recognition for the observance of travel behavior. *Proc., the 8th International Conference on Survey Methods in Transport*.

Flask.palletsprojects.com. (2021). *Foreword — Flask Documentation (2.0.x)*. [online] Available at: https://flask.palletsprojects.com/en/2.0.x/foreword/#what-does-micro-mean [Accessed 21 June 2021].

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A. and De, D. (2019). Fundamental Concepts of Convolutional Neural Network. *Intelligent Systems Reference Library*, pp.519-567.

Jin, L., Xian, H., Bie, J., Sun, Y., Hou, H. and Niu, Q. (2012). License Plate Recognition Algorithm for Passenger Cars in Chinese Residential Areas. *Sensors*, 12(6), pp.8355-8370.

Wu, J. (2017). Introduction to convolutional neural networks. National Key Lab for Novel Software Technology, Nanjing University, China.

Joseph Redmon and Ali Farhadi (2018). YOLOv3: An Incremental Improvement. *CoRR, abs/1804.02767*.

Krig, S. (2018). *Computer Vision Metrics*. 1st ed. Berkeley, CA: Apress, pp.39-43.

Leadtools.com. (2021). *Deskewing | Document Imaging | Raster Imaging C API Help*. [online] Available at: https://www.leadtools.com/help/sdk/v21/main/api/deskewing.html [Accessed 21 June 2021].

Lubna, Mufti, N. and Shah, S. (2021). Automatic Number Plate Recognition: A Detailed Survey of Relevant Algorithms. *Sensors*, 21(9), p.3028.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. Le Cam & J. Neyman (Eds.), *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). California: University of California Press.

Mutholib, A., Gunawan, T., Chebil, J. and Kartiwi, M. (2013). Development of Portable Automatic Number Plate Recognition System on Android Mobile Phone. *IOP Conference Series: Materials Science and Engineering*, 53, p.012066.

OpenCV. (2021). *Home - OpenCV*. [online] Available at: https://opencv.org/ [Accessed 20 June 2021].

Pant, A., Gyawali, P. and Acharya, S. (2015). Automatic Nepali Number Plate Recognition with Support Vector Machines. In: *9th International Conference on Software, Knowledge, Information Management and Applications*. Kathmandu: IEEE, pp.1-3.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, p.2825–2830.

Sharma, S., Sarkar, D., and Gupta, D. (2012). Agile Processes and Methodologies: A Conceptual Study. *International Journal on Computer Science and Engineering, 4*.

Sonka, M., Hlavac, V. and Boyle, R. (1993). Image pre-processing. *Image Processing, Analysis and Machine Vision*, pp.56-111.

TensorFlow. (2021). *TensorFlow*. [online] Available at: <https://www.tensorflow.org/> [Accessed 12 July 2021].

Tzutalin. LabelImg. Git code (2015). https://github.com/tzutalin/labelImg

Wu, H. and Li, B. (2011). License plate recognition system. *2011 International Conference on Multimedia Technology*, pp.5425-5427.

APPENDIX A – SDLC AND PRODUCT DESIGN

A1. Dataflow diagram

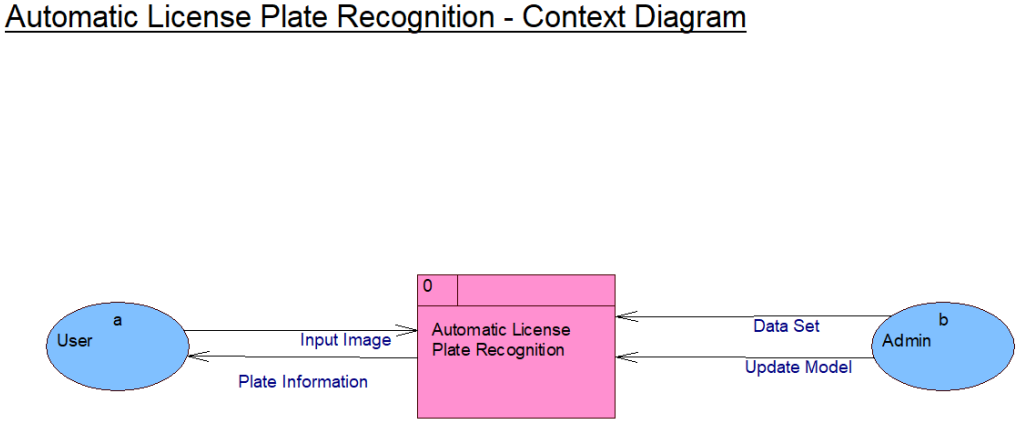


Figure A1: Data flow diagram: Context

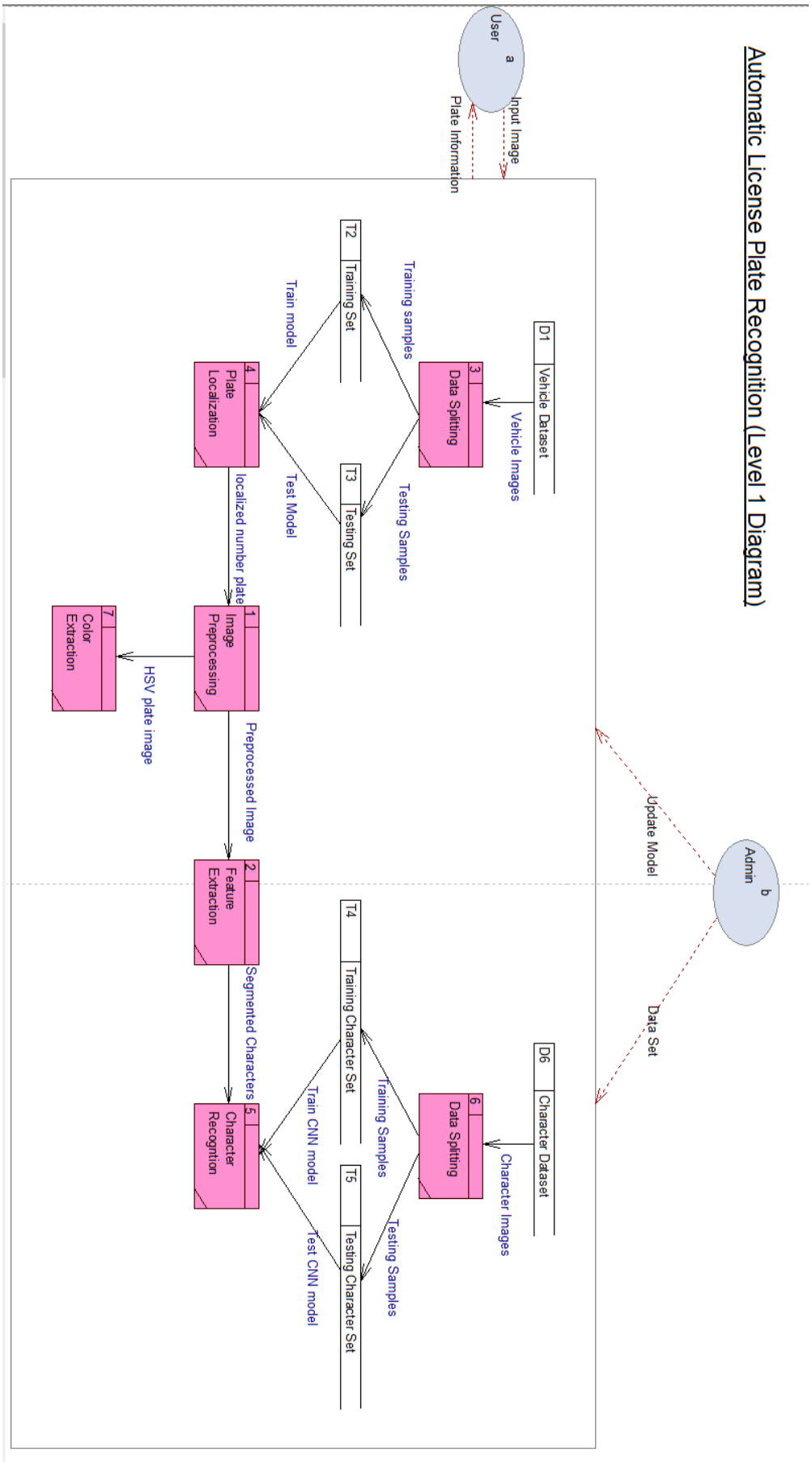


Figure A2: Data Flow Diagram Level 1

A2: System Design

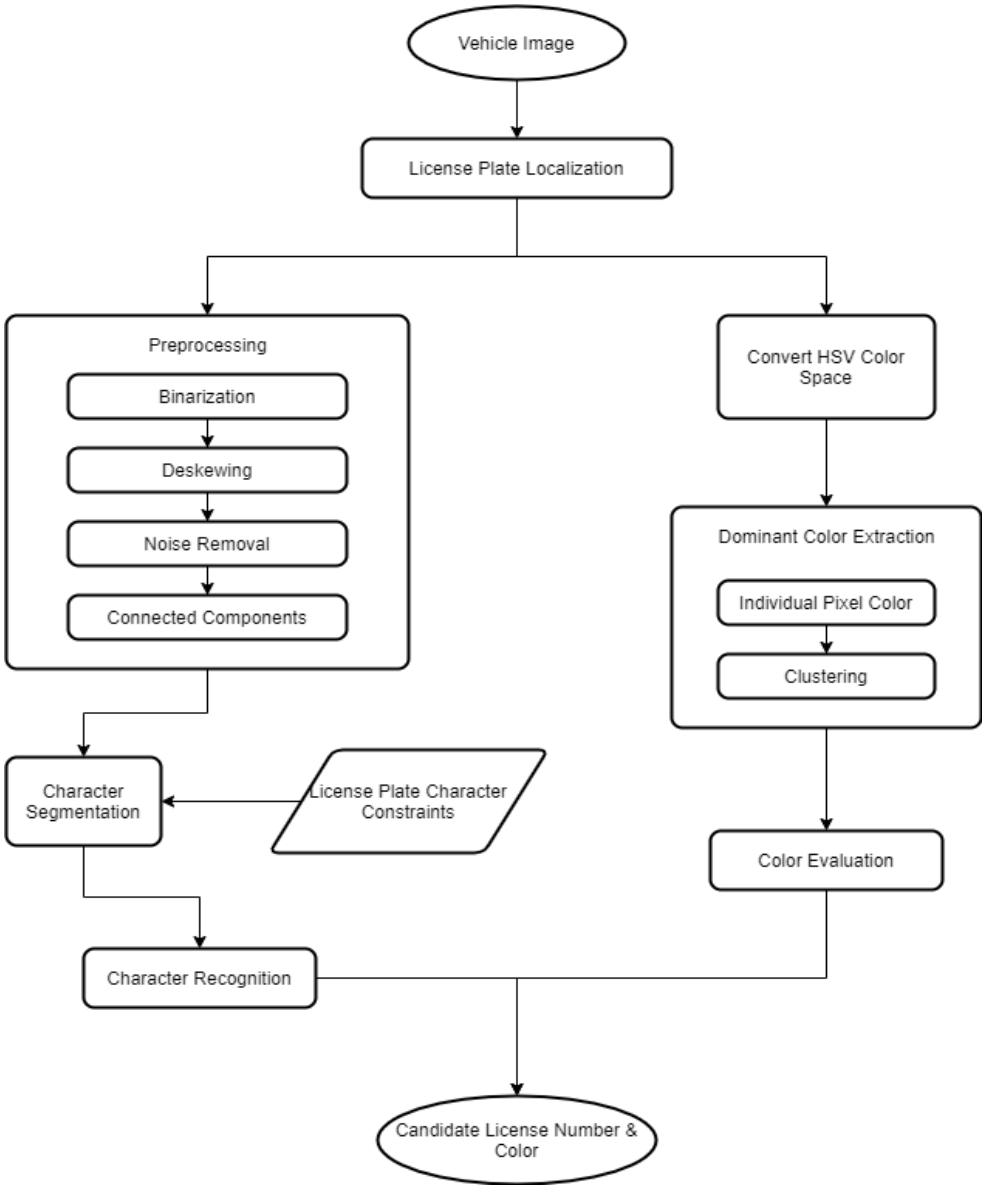


Figure A3: System Architecture

A3: Development Life Cycle for Machine Learning Model

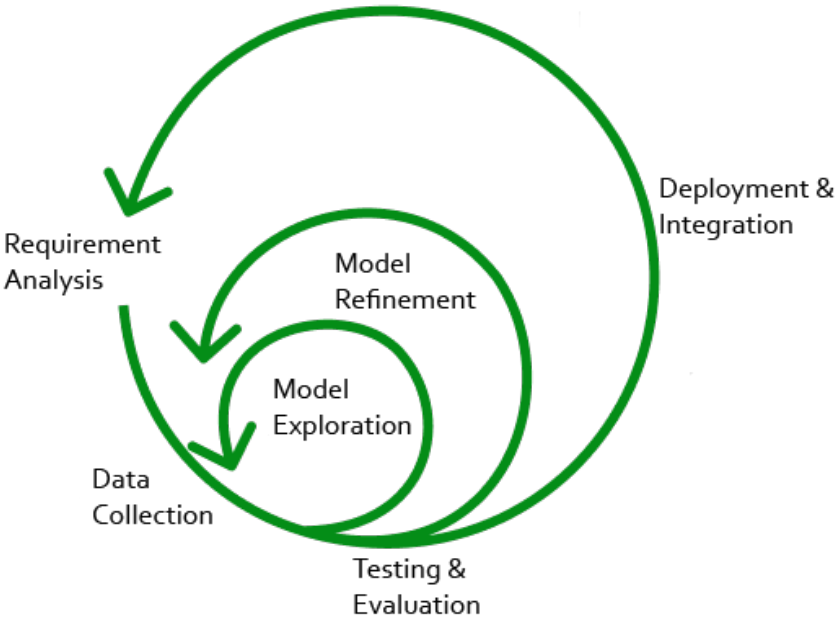


Figure A4: Machine Learning Development Cycle

A4. Kanban Board

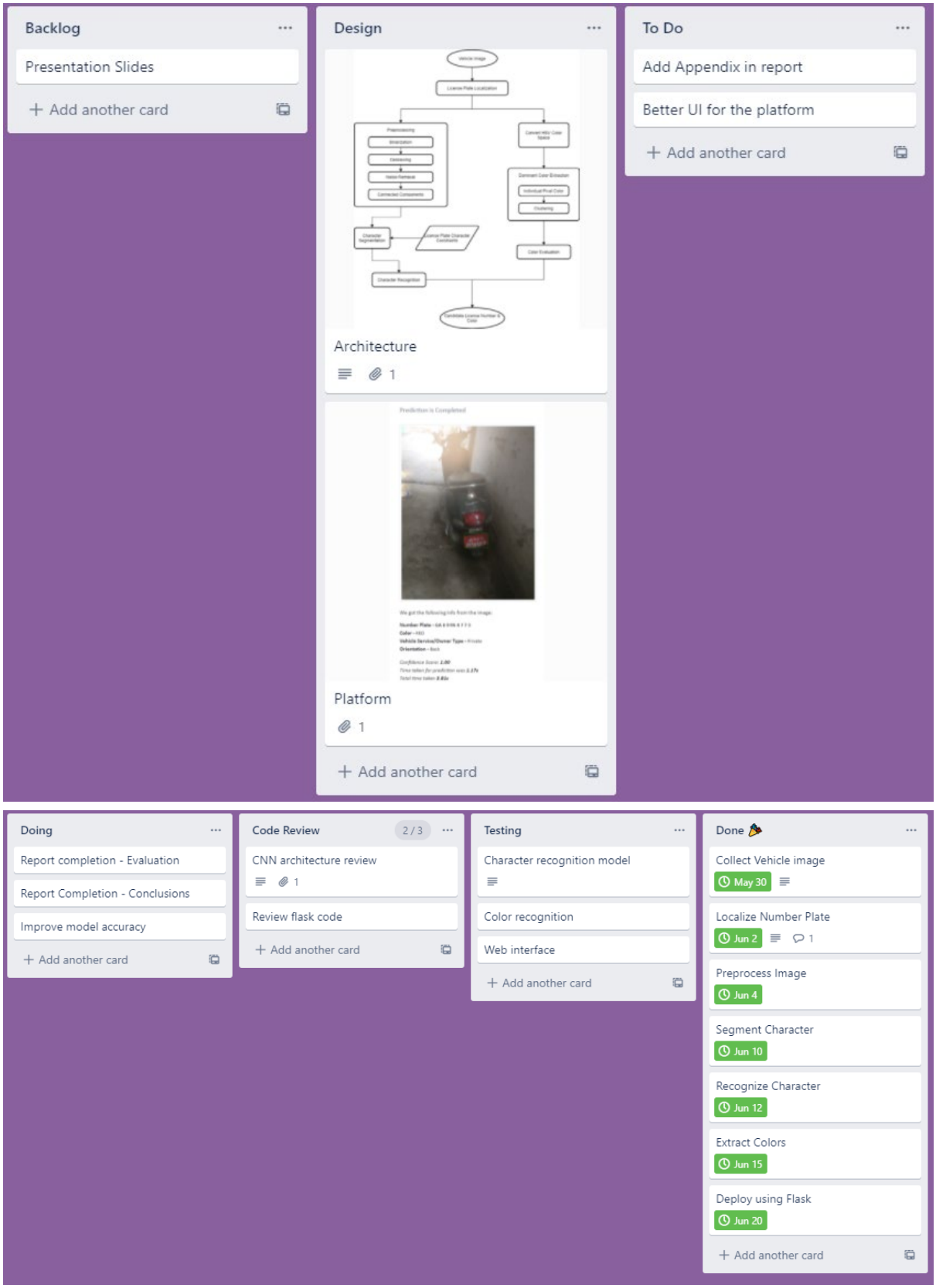


Figure A5: Kanban Board

APPENDIX B – TESTING AND EVALUATION

B1. Testing results of each class in character recognition

Class	Precision (%)	Recall (%)	F1-Score (%)
0	100.00	100.00	100.00
1	97.18	99.42	98.29
2	98.04	77.12	86.33
3	97.78	96.70	97.24
4	99.03	78.41	87.52
5	95.57	96.18	95.87
6	96.63	94.51	95.56
7	91.29	95.97	93.57
8	98.50	96.34	97.41
9	93.38	98.17	95.71
MA	90.18	93.63	91.87
KA	89.34	89.01	89.17
SA	98.30	100.00	99.14
JA	93.77	99.27	96.44
NA	89.60	98.73	93.94
BA	98.88	97.07	97.97
GA	78.74	87.26	82.78
LA	86.55	94.27	90.24
DHHA	97.50	99.36	98.42
BHA	86.96	88.24	87.59
RA	77.78	93.96	85.11
KHA	87.58	89.81	88.68
DA	80.31	98.73	88.57
YA	100.00	100.00	100.00
GHA	99.40	95.38	97.35
CHA	94.51	99.42	96.90
JHA	100.00	99.42	99.71
YNA	100.00	100.00	100.00
THHA	100.00	94.22	97.02
PA	92.51	100.00	96.11

Figure B1: Testing results for each class in character recognition

B2. Black Box Testing

a. Color Prediction Test

Test Case: Detect Color from Low Quality Image		**Test No:** 3	
Description: Application should successfully detect and predict color of plate			
Prerequisites: The web application must be loaded in a browser			
Step	**Action**	**Expected Result**	**Evaluation**
1	User uploads an image using the Upload button	The image must be uploaded, and a preview must be shown	Pass
2	User clicks on the Predict button	Green is shown as the result	Pass
Input Image Color: Green		Output: Green	

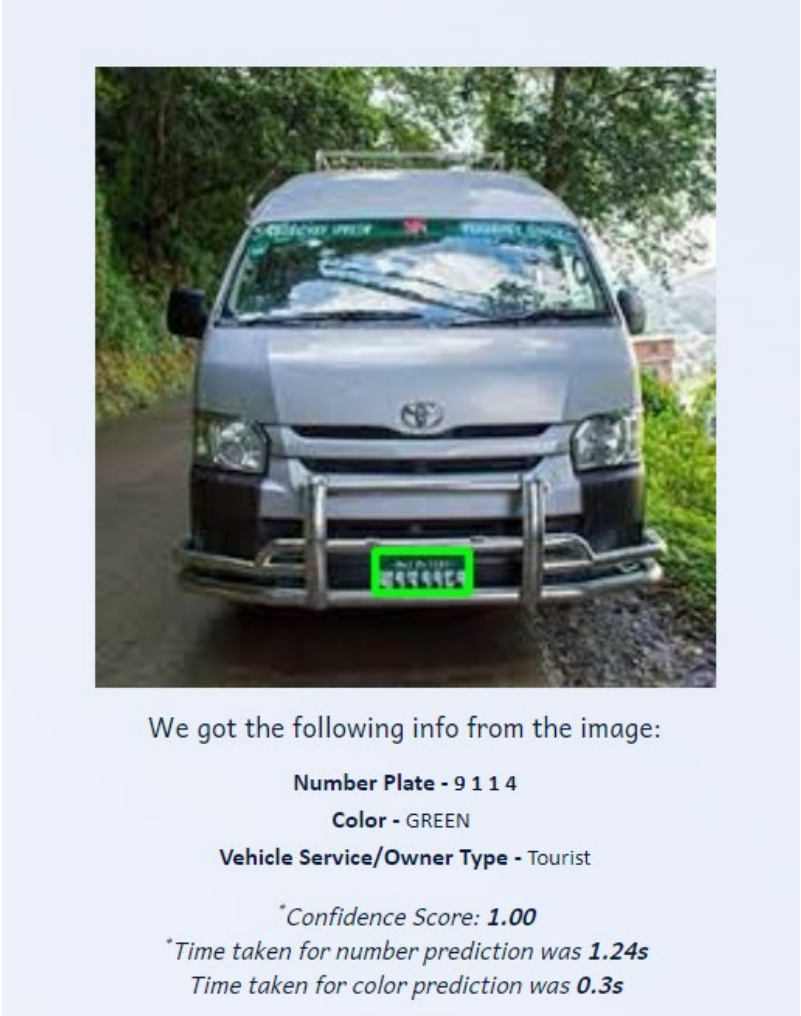


Figure B2: Green color testing

b. Incorrect image test

Test Case: Correctly identify incorrect image		**Test No:** 4	
Description: Application should successfully provide error message for incorrect image			
Prerequisites: The web application must be loaded in a browser			
Step	**Action**	**Expected Result**	**Evaluation**
1	User uploads an image using the Upload button	The image must be uploaded, and a preview must be shown	Pass
2	User clicks on the Predict button	An error message is shown	Pass

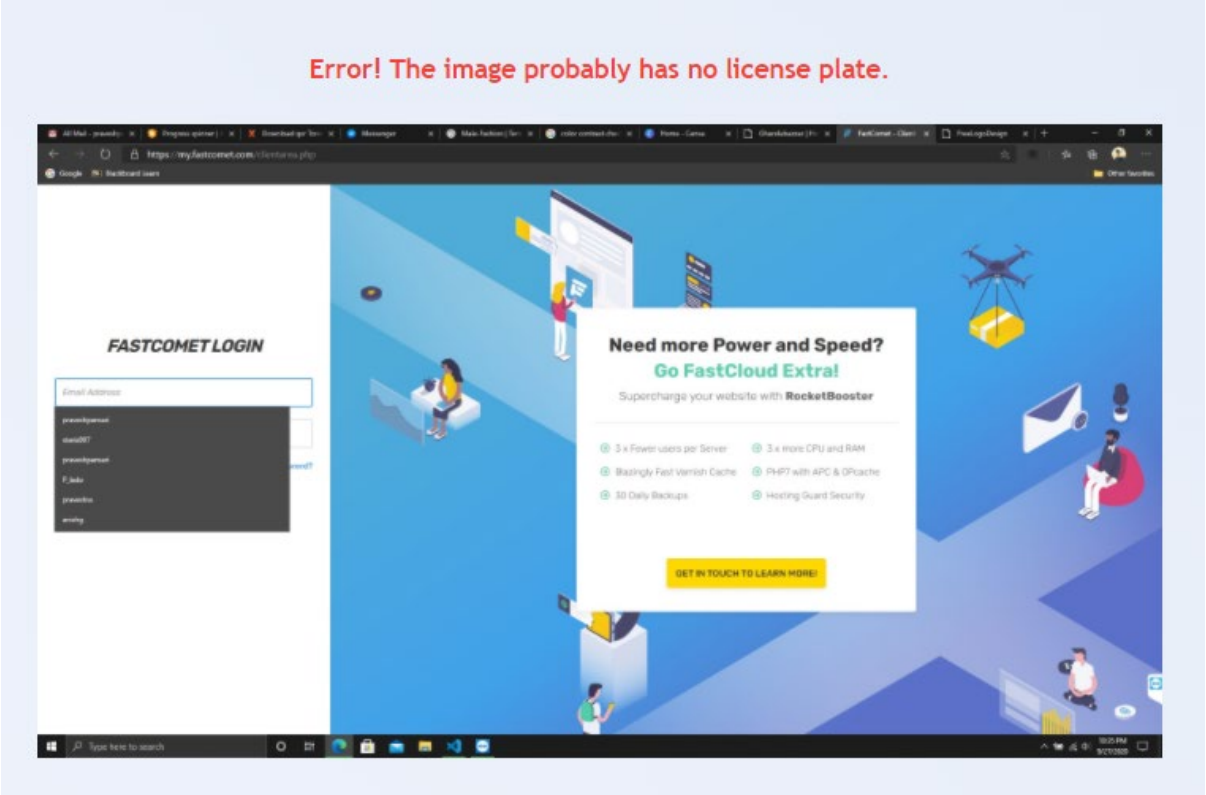


Figure B3: Incorrect image testing

c. Image testing

S. No.	Image	Expected Number	Expected Color	Predicted Number	Predicted Color
1		BA 1 CHA 3 9 5 2	Red	BA 1 CHA 3 9 5 2	Red
2		BA 1 4 CHA 7 4 8 0	Red	BA 1 4 CHA 7 4 8 0	Red
3		BA 1 2 CHA 6 9 4 4	Red	BA 1 2 CHA 6 9 4 4	Red
4		BA 1 JA 4 1 3 1	Black	NA 5 4 1 3 1	Black

5		BA 1 JA 6 3 3 3	Black	BA 1 JA 6 3 3 3	Black
6		BA 7 CHA 1 3 6	Red	BA 7 CHA 1 3 6	Red
7		BA 6 CHA 6 8 9 3	Red	YA 6 CHA 6 8 9 3	Red
8		BA 1 9 CHA 2 5 7 8	Red	THHA 1 9 CHA 2 5 7 8	Red

APPENDIX C – PROJECT MANAGEMENT TECHNIQUES

C1. Ethical Consent

Leeds Beckett University

School of Computing, Creating Technology & Engineering

Consent Form

To be completed by the participant. If the participant is under 18, to be completed by parent/ guardian or person acting in loco parentis.

Project title Automatic Nepali Number Plate Recognition from vehicle image using AI

Researcher's name Pravesh Pansari

Supervisor's name Resham Pun

This Consent Form has 2 parts:

- Participant Information Sheet.
- Confirmation of Consent Form (for signature if you agree to take part)

Please tick the boxes to confirm you have read and agree to each point

- I have read the Participant Information Sheet and the nature and purpose of the research project has been explained to me. ☑
- I have had the opportunity to ask questions and I have received satisfactory answers to all my questions ☑
- I understand the purpose of the research project, my involvement in it and I agree to take part ☑
- I understand that my participation is voluntary and that I may withdraw from the research project at any stage, without giving any reason ☑
- I understand that information gained during the study may be used to generate statistics and may be included in a published report ☑
- I understand that my personal details will remain confidential and that all data will be anonymised prior to publication ☑
- I understand that data will be stored only for training models and after a satisfactory accuracy has been obtained, the data will be removed. ☑
- *I understand that any data, images, video or audio recordings captured during this research project will be held securely and will not be used after completion of the research project, unless permission is explicitly given to do so;.* ☑
- I understand that I may contact the researcher Pravesh Pansari or supervisor Resham Pun if I require further information about the research [by emailing], and that I may contact the Research Ethics Coordinator Rohit Pandey of the School of Computing, Creative Technology and Engineering, Leeds Beckett University, [by emailing] if I wish to withdraw from the research project or to make a complaint relating to my involvement in the research. ☑

I have read the above information [or it has been read to me]. I have had the opportunity to ask questions about it and any questions I have asked, have been answered to my satisfaction. I consent voluntarily to participate as a participant in this research project.

Signed ..... (Research participant)

Print name ..... **Date** .....

Signature of Researcher.....

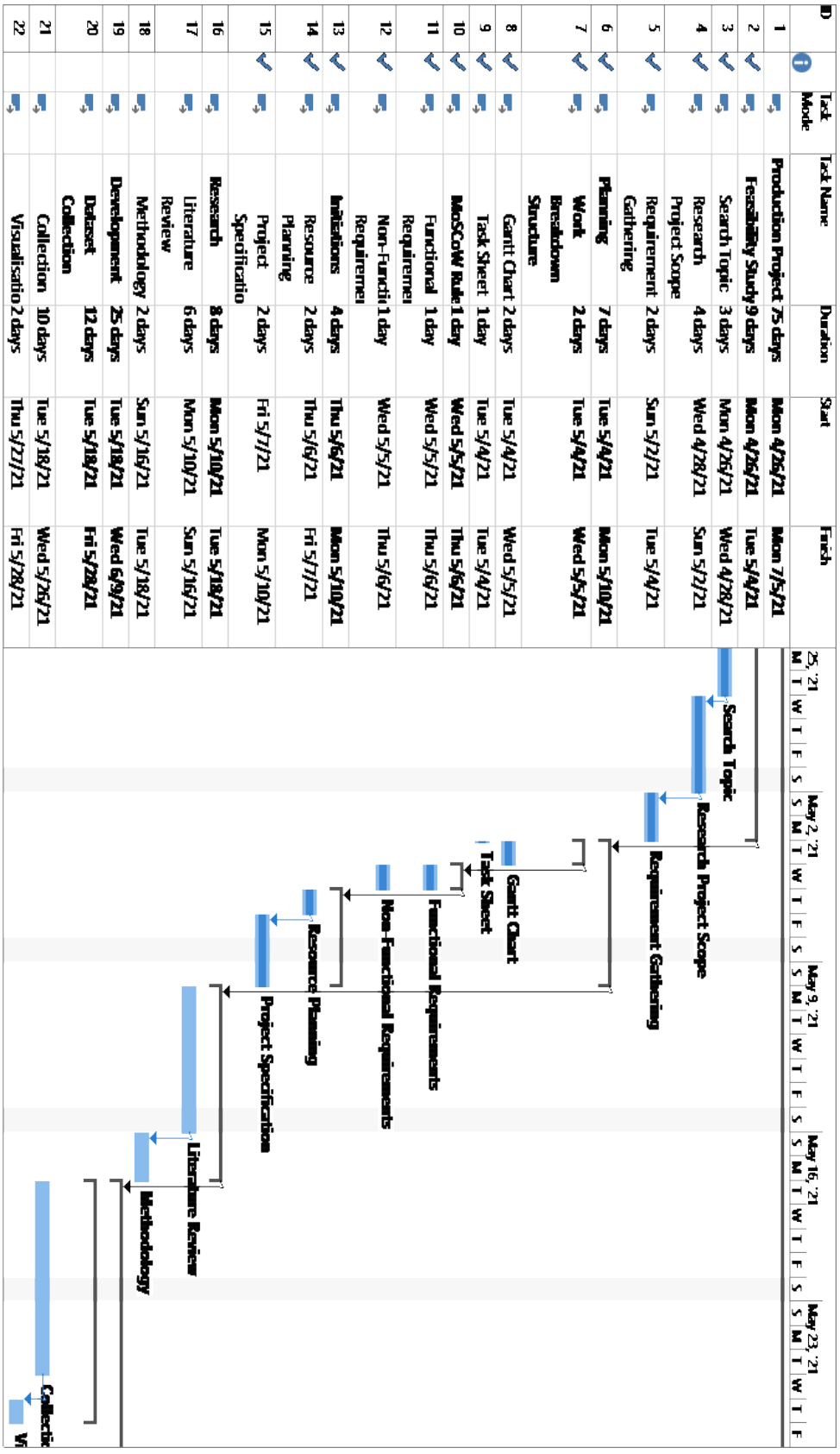
Contact details

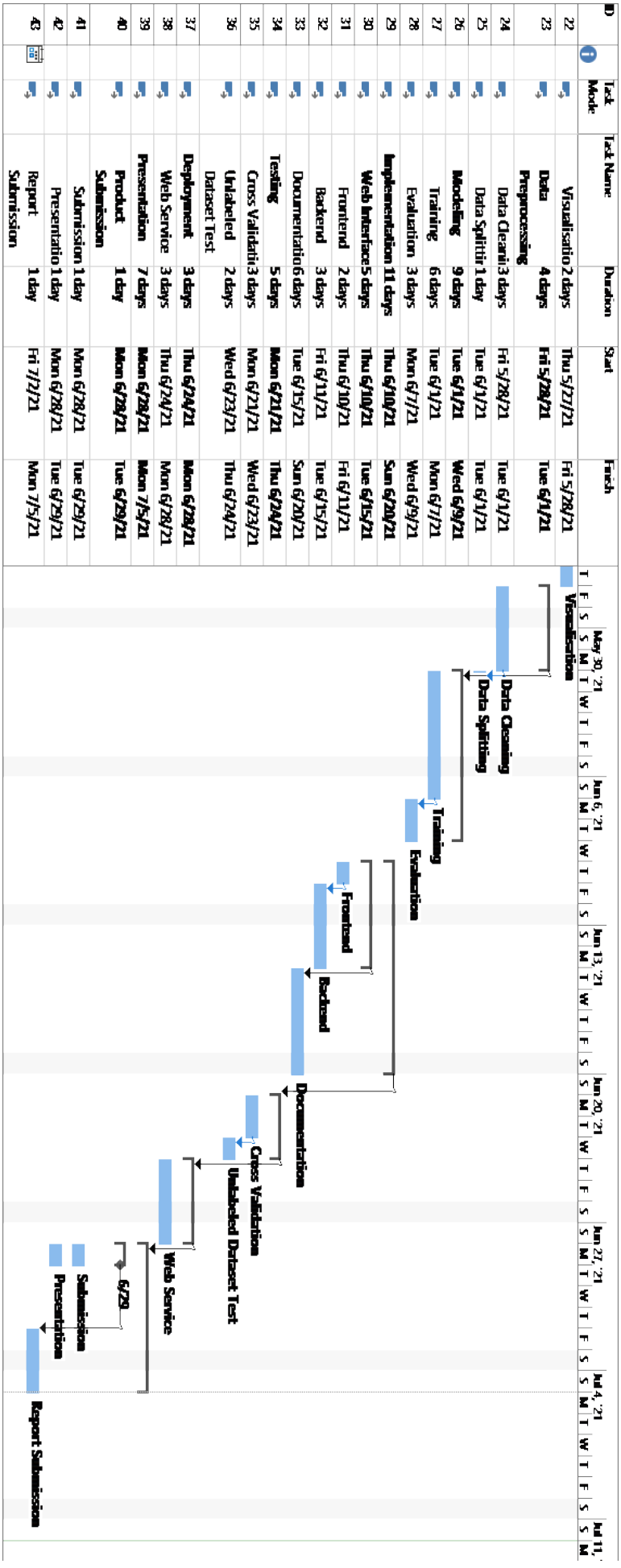
Researcher: email(praveshpansari@gmail.com)

Supervisor: email(rpun@thebritishcollege.edu.np)

School Ethics Coordinator: email(rpandey@thebritishcollege.edu.np)

C2. Gantt Chart




ID
Task Mode
Task Name
Duration
Start
Finish
22 Visualisatio 2 days Thu 5/27/21 Fri 5/28/21
23 Data Preprocessing 4 days Fri 5/28/21 Tue 6/1/21
24 Data Cleanin 3 days Fri 5/28/21 Tue 6/1/21
25 Data Splittin 1 day Tue 6/1/21 Tue 6/1/21
26 Modeling 9 days Tue 6/1/21 Wed 6/9/21
27 Training 6 days Tue 6/1/21 Mon 6/7/21
28 Evaluation 3 days Mon 6/7/21 Wed 6/9/21
29 Implementation 11 days Thu 6/10/21 Sun 6/20/21
30 Web Interface 5 days Thu 6/10/21 Tue 6/15/21
31 Frontend 2 days Thu 6/10/21 Fri 6/11/21
32 Backend 3 days Fri 6/11/21 Tue 6/15/21
33 Documentatio 6 days Tue 6/15/21 Sun 6/20/21
34 Testing 5 days Mon 6/21/21 Thu 6/24/21
35 Cross Validatio 3 days Mon 6/21/21 Wed 6/23/21
36 Unlabeled Dataset Test 2 days Wed 6/23/21 Thu 6/24/21
37 Deployment 3 days Thu 6/24/21 Mon 6/28/21
38 Web Service 3 days Thu 6/24/21 Mon 6/28/21
39 Presentation 7 days Mon 6/28/21 Mon 7/5/21
40 Product Submission 1 day Mon 6/28/21 Tue 6/29/21
41 Submission 1 day Mon 6/28/21 Tue 6/29/21
42 Presentatio 1 day Mon 6/28/21 Tue 6/29/21
43 Report Submission 1 day Fri 7/2/21 Mon 7/5/21
May 30, '21
Jun 6, '21
Jun 13, '21
Jun 20, '21
Jun 27, '21
Jul 4, '21
Jul 11, '
Visualisation
Data Cleaning
Data Splitting
Training
Evaluation
Frontend
Backend
Documentation
Cross Validation
Unlabeled Dataset Test
Web Service
6/29
Submission
Presentation
Report Submission

C3. Meeting Records

School of Computing, Creative Technologies, and Engineering 2020/21
Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 1
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 18th May 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	First Meeting
2	
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 19th May 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Present Initial Project Specification Plan
2	
3	
4	
5	
6	

Comments of supervisor (if any):

Make necessary edits and complete plan

……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 2
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 19th May 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Present Initial Project Specification Plan
2	
3	
4	
5	
6	

Comments of student (if any):

……

……

……

……

……

……

……

……

……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 20th May 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Complete Risk Register
2	Complete marking scheme according to requirements.
3	Submit Risk Register.
4	
5	
6	

Comments of supervisor (if any):

Change the project title to make it more specific and clearer

……

……

……

……

……

……

……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 3
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 20th May 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Complete Risk Register
2	Complete marking scheme according to requirements.
3	Submit Risk Register.
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 26th May 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Complete marking scheme for the project.
2	Submit the marking scheme to the supervisor before deadline.
3	
4	
5	
6	

Comments of supervisor (if any):

Risk register is fine can proceed with submission.

……
……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 4
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 26th May 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Complete marking scheme for the project.
2	Submit the marking scheme to the supervisor before deadline.
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 28th May 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Product Prototype Demo
2	
3	
4	
5	
6	

Comments of supervisor (if any):

Evaluation strategy all must be mentioned in marking scheme.
Add critical review of report in marking scheme
Marking scheme is fine can proceed with submission after correction.

……
……
……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 4
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 26th May 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Complete marking scheme for the project.
2	Submit the marking scheme to the supervisor before deadline.
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 28th May 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Product Prototype Demo
2	
3	
4	
5	
6	

Comments of supervisor (if any):

Evaluation strategy all must be mentioned in marking scheme.
Add critical review of report in marking scheme
Marking scheme is fine can proceed with submission after correction.

……
……
……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 5
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 28th May 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Product Prototype Demo
2	
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 7th June 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Product Prototype Presentation
2	
3	
4	
5	
6	

Comments of supervisor (if any):

……
……
……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 6
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 7th June 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Product Prototype Presentation
2	
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 18th June 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Report Demonstration (40%)
2	Product Progress
3	
4	
5	
6	

Comments of supervisor (if any):

……
……
……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:**7
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 11th June 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Product Progress
2	
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 14th June 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Report Demonstration (40%)
2	
3	
4	
5	
6	

Comments of supervisor (if any):

……
……
……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 8
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 14th June 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Report Presentation (40% complete)
2	
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 21st June 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Product Presentation
2	
3	
4	
5	
6	

Comments of supervisor (if any):

Complete remaining report

……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21

Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 9
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 21st June 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Final product draft present
2	
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:** 26th June 2021 12:30 – 13:30

Agreed Actions to complete before next meeting:

1	Final report draft
2	
3	
4	
5	
6	

Comments of supervisor (if any):

Need better UI for web

Add more information in interface

……
……
……
……
……
……
……

School of Computing, Creative Technologies, and Engineering 2020/21
Level 6 Production Project

MEETING RECORD SHEET	**Meeting Number:** 10
Student: Pravesh Pansari	**Student I.D.:** 77202315
Date of Meeting: 26th June 2021	**Supervisor:** Resham Pun

Actions agreed at previous meeting (completed or comment)

1	Show final product
2	Discuss about final report
3	
4	
5	
6	

Comments of student (if any):

……
……
……
……
……
……
……
……
……

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:**

Agreed Actions to complete before next meeting:

1	
2	
3	
4	
5	
6	

Comments of supervisor (if any):

Submit report for evaluation before final submission

……
……
……
……
……
……
……
……

APPENDIX D – BACKUP AND RISK ANALYSIS

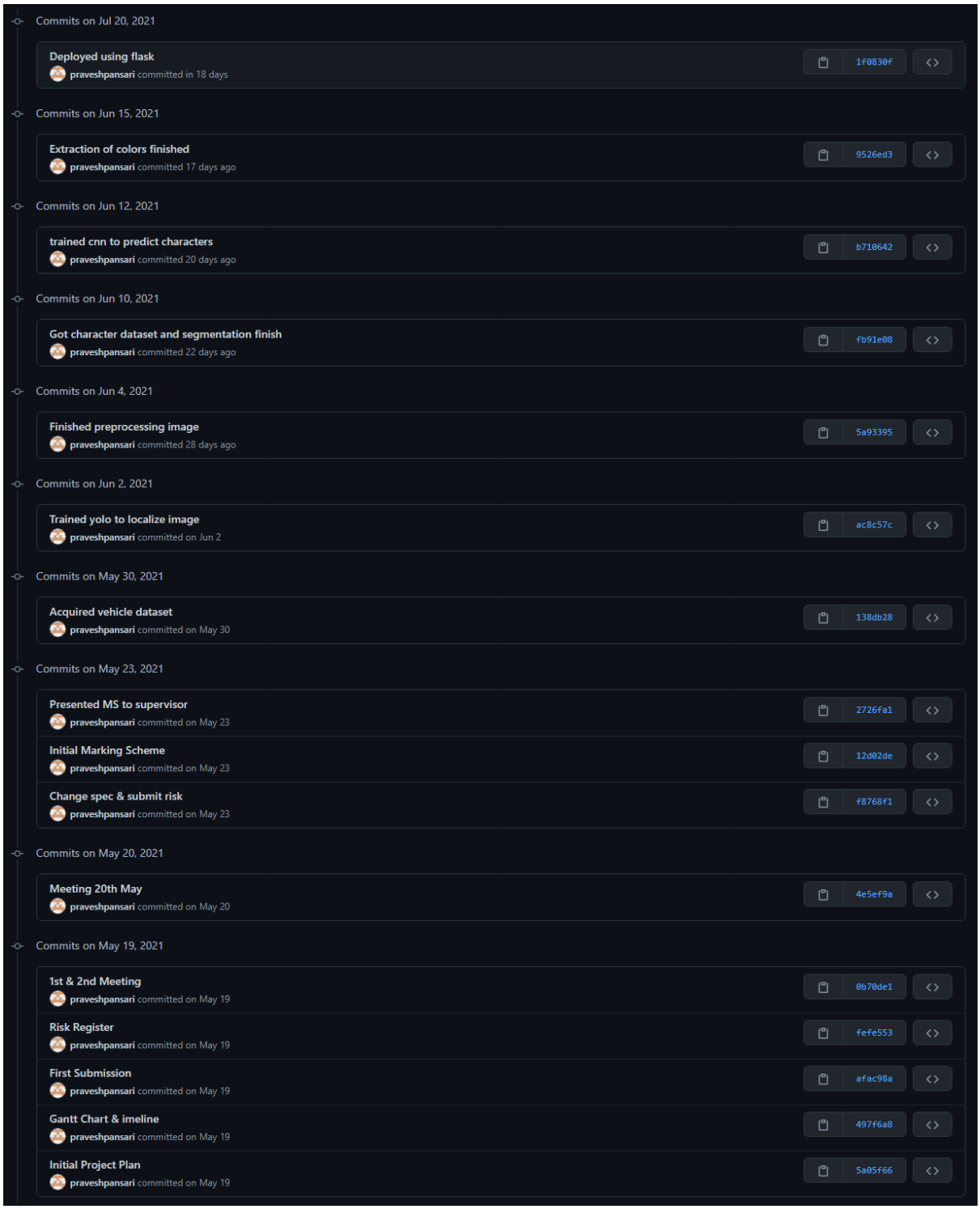
D1. Risk Register

ID	Risk Description	Likelihood	Impact	Severity	Owner	Mitigation	Status
1	Hardware Failure / Incapable Hardware	Low	Medium	Low	Pravesh Pansari	Deep Learning requires powerful hardware. Thus, can result in hardware failure too. Use platform such as Google Colab for performing heavy training tasks.	Open
2	Unable to collect data (Pandemic, Natural Disaster etc.)	High	Medium	High	Pravesh Pansari	Large amount of image data is required for the model training. In case of insufficient data, try to use data available online.	Open
3	Unable to access internet / Internet service disruption	Low	Low	Low	Pravesh Pansari	Due to unprecedented electricity outages internet service disruption can occur. Can use mobile data as hotspot for internet access.	Open
4	Ethics	Low	Medium	Low	Pravesh Pansari	Ethical approval might be required during collection of vehicle images for the dataset. Alternative would be to use open source dataset available online.	Open
5	Health Issues (Pandemic)	Medium	High	High	Pravesh Pansari	Practice specified safety measures. In case of health crisis, inform supervisor about the hazard and its consequences on the project timeline.	Open

Figure D1: Risk Analysis

D2. Code Repository

https://github.com/praveshpansari/anpr-nepal



APPENDIX E – PROJECT SPECIFICATION

BSc (Hons) Computing Course 2020/21

Level 6 Production Project

Name: Pravesh Pansari	**Student I.D.:** 77202315
Course: BSc (Hons) Computing	**Supervisor's Name:** Resham Pun

Final Project Individual Aim & Objectives

Title of my Project: Automatic Nepali Number Plate Recognition from vehicle image using AI

Aim of my Project: The main aim of this project is to identify and track vehicles for law enforcement purposes by recognizing vehicle registration number.

Objectives of my Project: The main objective is to produce a robust product which:

- Detects license/number plates on vehicles.
- Localizes and outputs the registration number.
- Provides simple interface for access and usability.
- High accuracy and f-score (above 80%) of the model.

Specification of my Product:

Functional Requirements:

Functional Requirements	MoSCoW
Acquire necessary dataset with enough data	M
Built and Train a model on the dataset	M
Extract and label the number plate	M
Deploy model in REST API	M
GUI to access the product	S
Train and Compare Models	S
Show Confidence Score	C
Classify type of Number Plate	W
Allow access for future updates	W
Predict Color of Number Plate	W

Non-Functional Requirements:

Non-Functional Requirements	MoSCoW
Testing and validation of the product	M
User open source & free software	M
Interface is user friendly	S
Proper documentation of the product	S
Use two types of ML algorithms	C
Detect in low quality images	W

Research:

The extraction of vehicle license plate details from an image or a series of images is known as automatic license plate recognition (ALPR) (Chang et al, 2004). The proposed ALPR system consists of two major steps: license plate location and license number identification. Tesseract (Smith, 2007), which is maintained by Google, is one of the open source OCR tools that supports multiple languages. It is used to recognize characters from text (Zheng et al, 2013). This study trains and compares two selected machine learning algorithms in various domains. The collected data can be used in a variety of applications, including electronic payment systems (toll payment, parking fee payment), traffic surveillance (including violations of traffic rules), and locating a stolen or missing vehicle. Road traffic accidents in Nepal are caused due to driver related rule violations like speeding, overloading, overworking as well has vehicle related factors like mechanical and old vehicles (Karkee, 2020). The passengers as well as the drivers violate the traffic rules but are not punished or penalized heavily or in some cases are not captured (Dhakal, 2018). This system of automatic number recognition can help in capturing of traffic law breakers as well as help in robbery cases.

Evaluation:

The evaluation of the product will be based on the fulfillment of the requirements of the project as well as how well the product captures the aim and completes the objectives. Evaluation will be done using some images from the data set as well as cross validation methods will be used. The accuracy, time complexity and f-score will be measured for the performance measure. Invalid datasets might be used to test the model's ability to differentiate between correct and incorrect images.

Project Planning & Methodology

Project Planning:

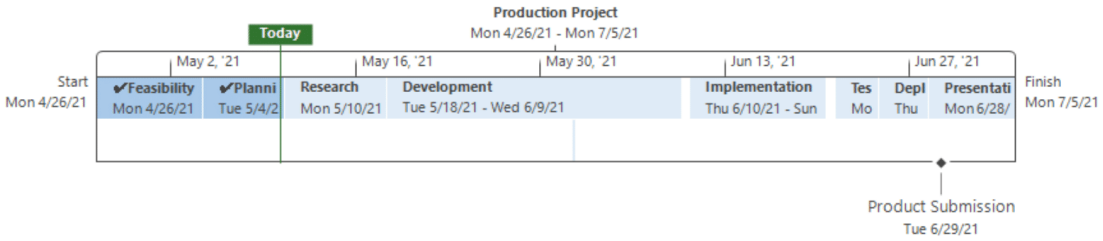
The final product will be accessible through a simple web interface which consumes an in planning is further clarified using the Project Timeline and Gantt chart.

Gantt Chart:

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Production Project**	**75 days**	**Mon 4/26/21**	**Mon 7/5/21**
2		**Feasibility Study**	**9 days**	**Mon 4/26/21**	**Tue 5/4/21**
3		Search Topic	3 days	Mon 4/26/21	Wed 4/28/21
4		Research Project Scope	4 days	Wed 4/28/21	Sun 5/2/21
5		Requirement Gathering	2 days	Sun 5/2/21	Tue 5/4/21
6		**Planning**	**7 days**	**Tue 5/4/21**	**Mon 5/10/21**
7		**Work Breakdown Structure**	**2 days**	**Tue 5/4/21**	**Wed 5/5/21**
8		Gantt Chart	2 days	Tue 5/4/21	Wed 5/5/21
9		Task Sheet	1 day	Tue 5/4/21	Tue 5/4/21
10		**MoSCoW Rule**	**1 day**	**Wed 5/5/21**	**Thu 5/6/21**
11		Functional Requirements	1 day	Wed 5/5/21	Thu 5/6/21
12		Non-Function Requirements	1 day	Wed 5/5/21	Thu 5/6/21
13		**Initiations**	**4 days**	**Thu 5/6/21**	**Mon 5/10/21**
14		Resource Planning	2 days	Thu 5/6/21	Fri 5/7/21
15		Project Specification	2 days	Fri 5/7/21	Mon 5/10/21
16		**Research**	**8 days**	**Mon 5/10/21**	**Tue 5/18/21**
17		Literature Review	6 days	Mon 5/10/21	Sun 5/16/21
18		Methodology	2 days	Sun 5/16/21	Tue 5/18/21
19		**Development**	**25 days**	**Tue 5/18/21**	**Wed 6/9/21**
20		**Dataset Collection**	**12 days**	**Tue 5/18/21**	**Fri 5/28/21**
21		Collection	10 days	Tue 5/18/21	Wed 5/26/21
22		Visualisation	2 days	Thu 5/27/21	Fri 5/28/21
23		**Data Preprocessing**	**4 days**	**Fri 5/28/21**	**Tue 6/1/21**

ID	Task Mode	Task Name	Duration	Start	Finish
24		Data Cleaning	3 days	Fri 5/28/21	Tue 6/1/21
25		Data Splitting	1 day	Tue 6/1/21	Tue 6/1/21
26		**Modeling**	**9 days**	**Tue 6/1/21**	**Wed 6/9/21**
27		Training	6 days	Tue 6/1/21	Mon 6/7/21
28		Evaluation	3 days	Mon 6/7/21	Wed 6/9/21
29		**Implementation**	**11 days**	**Thu 6/10/21**	**Sun 6/20/21**
30		**Web Interface**	**5 days**	**Thu 6/10/21**	**Tue 6/15/21**
31		Frontend	2 days	Thu 6/10/21	Fri 6/11/21
32		Backend	3 days	Fri 6/11/21	Tue 6/15/21
33		Documentation	6 days	Tue 6/15/21	Sun 6/20/21
34		**Testing**	**5 days**	**Mon 6/21/21**	**Thu 6/24/21**
35		Cross Validation	3 days	Mon 6/21/21	Wed 6/23/21
36		Unlabeled Dataset Test	2 days	Wed 6/23/21	Thu 6/24/21
37		**Deployment**	**3 days**	**Thu 6/24/21**	**Mon 6/28/21**
38		Web Service	3 days	Thu 6/24/21	Mon 6/28/21
39		**Presentation**	**7 days**	**Mon 6/28/21**	**Mon 7/5/21**
40		**Product Submission**	**1 day**	**Mon 6/28/21**	**Tue 6/29/21**
41		Submission	1 day	Mon 6/28/21	Tue 6/29/21
42		Presentation	1 day	Mon 6/28/21	Tue 6/29/21
43		Report Submission	1 day	Fri 7/2/21	Mon 7/5/21

Project Timeline:



Methodology:

The methodology I have planned to use for this project is Agile methodology, which will allow revision of requirements and features for the product.

Resources

The hardware and software I require to complete my Project successfully:

Software Used:

- Tesseract
- OpenCV
- Imutils
- Python
- Google Colaboratory
- Jupyter
- HTML/CSS/JS

Hardware Used:

- Laptop (Windows 10)
- Ethernet Connection

Human Resource

I am working on my Project with the following people

Name: Pravesh Pansari	Module Leader: Mahesh Maharjan
	Supervisor: Resham Pun

Initial Bibliography

- Dhakal, K., 2018. Road Traffic Accidents in Kathmandu Valley. *Journal of Health Promotion*, 6, pp.37-44.
- Rajendra Karkee 2020. Risk of road traffic injuries and their prevention in Nepal. *Journal of Public Health and Emergency, 4*(0).
- Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan Chen 2004. Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems, 5*(1), p.42-53.
- Smith, R., 2007. An Overview of the Tesseract OCR Engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*.
- Zheng, L., He, X., Samali, B. and Yang, L., 2013. An algorithm for accuracy enhancement of license plate recognition. *Journal of Computer and System Sciences*, 79(2), pp.245-255.

APPENDIX F – CODE SNIPPETS

F1. Algorithm for segmenting lines from the image

```
lines = []
line = []
FOR each row in h_projection – 1:
   IF h_projection[row] == 0 and h_projection[row + 1] > 0
      append row to line
   ENDIF
   size = len(line)
   IF size and h_projection[row] > 0 and h_projection[row + 1] == 0
      IF row – line[0] > min_height
         append row to line
         append image rows from line[0]:line[1] to lines
      reintialize line to []
      ENDIF
   ENDIF
ENDLOOP
```

F2. Algorithm for segmenting characters from a list of line images

```
letters = []
FOR each line in lines:
   letter = []
   recalculate v_projection for line
   FOR each col in v_projection - 1:
      IF v_projection[col] == 0 and v_projection[col + 1] > 0
         append col to letter
      ENDIF
      size = len(letter)
      IF size and v_projection[col] > 0 and v_projection[col + 1] ==
   0
         IF col - line[0] > min_width
            append col to letter
            append image columns from letter[0]:letter[1] to letters
         reintialize letter to []
         ENDIF
      ENDIF
   ENDLOOP
ENDLOOP
```

F3. Comparison of dominant colors to threshold ranges

```
dom_colors = hsv_to_rgb(dom_colors)
dom_colors = dom_colors * 255
result = ''
FOR color in dom_colors:
   IF color is in yellow color range
      set result to 'red'
      BREAK
   ENDIF
   IF color is in green color range
      set result to 'green'
      BREAK
   ENDIF
   .
   .
   .
   IF color is in white color range
      set color to 'white'
      BREAK
   ENDIF
ENDLOOP
IF result is empty:
   IF color is in range black
   set result to 'black'
   BREAK
ENDIF
```