1. Introduction

1.1 Problem Definition

The scientific world is deploying research in intelligent transportation systems which have a significant impact on people's lives. License Plate Recognition is a computer vision technology to extract the license number of vehicles from images. It is an embedded system which has numerous applications and challenges. Typical LPR systems are implemented using proprietary technologies and hence are costly. This closed approach also prevents further research and development of the system. With the rise of free and open source technologies the computing world is lifted to new heights. People from different communities interact in a multicultural environment to develop solutions for never ending problems. One of the notable contributions of the open source community to the scientific world is Python. Intel's research in Computer Vision bore the fruit called Open Computer Vision (OpenCV) library, which can support computer vision development.

License plate recognition is a Computer Vision technique which is able to recognize a license plate number. There are many applications ranging from complex security systems to common areas and from parking admission to urban traffic control. License plate recognition has complex characteristics due to diverse effects such as light. This mainly focuses on CNN and proprietary tools OpenCV in which LPR systems implement using Free Software Open Computer Vision Library including python. First we capture the image from the camera then load into the system after that we use OpenCV library tools. Then we make the training set of different characters of different sizes. On the basis of these training sets we extracted the character from images. When the license plate is detected, its digits are recognized and displayed in the GUI.

1.2 Motivation

In this era, license plate recognition system have various applications and can be implemented in various fields and these includes:

1.2.1 Parking -The plate number is used to automatically check staff members and calculate parking fee for non-staff members by comparing the exit and entry times.

1.2.2 Access Control - a gate automatically opens for authorized members in a secured area, thus replacing or assisting the security guard. The events are logged on a database and could be used to search the history of events

1.2.3. Law enforcement - The manual process of preparing a violation fine is replaced by an automated process which reduces the overhead and turnaround time. Using a list of stolen cars or unpaid fines, an ALPR system deployed on the roadside, may perform a real-time match between the passing cars and those on the list, hence raising alerts for passing cars on the list.

1.2.4. Marketing Tool - the car plates may be used to compile a list of frequent visitors in a particular shop for marketing purposes, or to build traffic profiles, such as the frequency of entry verses the hour or day.
1.2.5. Tolling - the car number is used to calculate the travel fee on a toll-road, or used to double-check the ticket.

Such routine tasks, possibly handling large volumes of traffic, may be more efficiently done if automated than if done by humans. This would reduce associated costs of operation, increase processing speed and minimize errors that may result due to fatigue or monotony if a human operator were involved.

1.3 Objectives

The main goal of our project is to develop a Nepali license plate recognition software which can recognize license plates from images and then convert the license plates images into text.

2. Related Works

We researched and collected data from various projects similar to ours. In these projects ALPR systems have been developed using different techniques. Some of the existing applications are listed below:

2.1 Kaushik Deba, Md. Ibrahim Khana, Anik Sahaa, and KangHyun Job, (2012) [1]

There is segmentation technique used to sliding concentric windows (scw). In this we extract license plates from natural properties by finding vertical and horizontal edges from vehicle region. On the Basis of a novel adaptive image segmentation technique is for detecting candidate region and Color verification for candidate region by using HSI color model on the basis of using hue and intensity in HSI colour model verifying green and yellow LP and white LP, respectively. Basically they focus on artificial neural network (AAN) new algorithms which are based on Korean number plate system.

2.2 M.I.Khalil, (2010) [2]

This LPR system consists of 4 modules: Image acquisition, licensed plate extraction, segmentation & recognition of individual character. After the license plate extraction phase, INFORMATION RECOGNITION PHASE (IPR) is applied. For this phase "moving window technique" is used. For recognizing the image license plate, the country name is loaded as the

source image. Then the first image entry of the country image set is loaded as an object. The moving window technique is applied to detect that object within the image.

2.3 Kumar Parasuraman, (2010) [3]

There is SVM i.e. Supervised Learning Technique is used which takes Statistical Learning Theory and for best the structural risk minimization as its optimal object to realize. Two main approaches have been suggested for applying SVMs for multiclass classification. They are "one against all" and "one against one". A number plate region is located by using mean shift method and extracted; the histogram projection method in horizontal direction is applied for a simple segmentation only

2.4 Mukesh Singh Chandel, Satya Verma, (2018) [4]

In India automatic Indian number plate detection and recognition systems have an important role in modern travel and traffic systems. It helps to monitor the urban traffic and to recognize automobile thefts. In this digital modern world a large number of vehicles, cars around us in our daily life creates many disturbances such as heavy road traffic, the stealing of vehicles at the places like parking areas, toll plazas, heavy traffic roads, Shops. Every year many vehicles, car are stolen. No any vehicle can be tracked. This automatic number plate recognition is the solution of this problem. There are many algorithms and methodologies are be used to tracking of the vehicle number, but it is more challenging task in some of factors like as different non-uniform font style and letter on plate which is effects in recognition process. . In India the car number plate has a different shape and size. To verify the number plate of Indian cars, there it goes through the following ways: To find the vehicles number plate which is done by capturing images of vehicles with the help of good quality CCTVand then extracting is done through the help of image segmentation of the captured image and recognition of characters is done with the help of OCR. After these processes the number of vehicle license plates is displayed on a graphical user interface in Matlab platform and it will verify with the stolen vehicle database with stolen date for further use of alarm will ring if stolen vehicle is detected. This system can be used for security purposes in heavy traffic areas where it is difficult for the normal traffic police to verify the vehicle is stolen or not. IndexTerms—Number plate Extraction, MATLAB, Recognition, Digital Camera, luminance condition, Arduino.

2.5 Nwe Ni Kyaw, G. R. Sinha, Khin Lay Mon, (2018) [5]

License plate recognition (LPR) plays an important role in few major applications such as automatic parking system, surveillance applications, toll system etc. This paper studies major research contributions on LPR for various license plates of different countries. An emphasis is made to find out the current status of the work for Myanmar vehicle number plates. An extensive survey of related research suggests few important observations which are highlighted along with few recommendations that could overcome the major challenges of existing research in context with Myanmar number plates. Robustness and varying conditions are few of major points that

need attention in this area; moreover the research on LPR for Myanmar number plates needs to be vibrantly done.

3. Datasets

Dataset includes a number of photos for every letter(i.e. 0-9 ,BAA, PA) which are extracted from real plates. Each category of letter contains 125 or more photos totalling 2033 photos. Dataset is preprocessed and converted to 1-D array and divided into two parts:

Training set : more than 1500 samples

Testing set : 500 samples

preprocessing includes RGB to grayscale conversion, resizing(50x50) and conversion to image array.

4. Methods and algorithms used

4.1 Preprocessing of image

A series of operations are performed on the input image like color conversion, thresholding of image and morphological transformations to filter noise. In this project the following preprocessing operations are followed using openCV library:

color conversion : BGR to HSV, BGR to gray

thresholding : BINARY + OTSU

morphological transformation : gaussian blur, erosion and dilation, canny edge detection

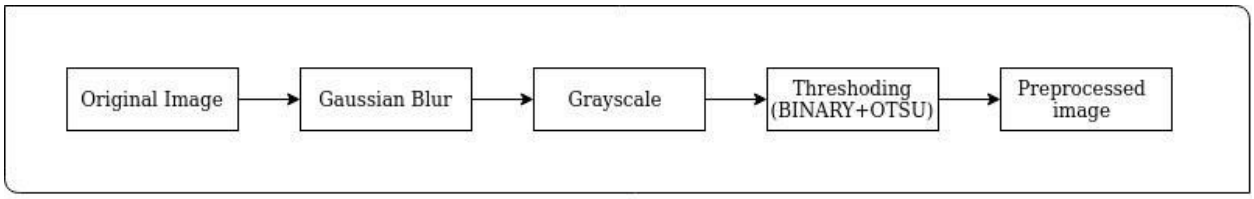


fig schematic diagram of image preprocessing

4.2 localization of plate

For localization of license plate from the input image, the following steps were followed:

- HSV mask: Input image is converted from BGR to HSV format to mask red color from the image as the private number plates have red background.
- preprocessing of HSV masked image with morphological transformations
- FInding closed contours in the image and bounding rectangles on those contours having four sides
- Applying aspect ratio test and area filter on the bounded rectangle to choose the correct rectangle which includes the license plate
- segmenting the region of interest on the image

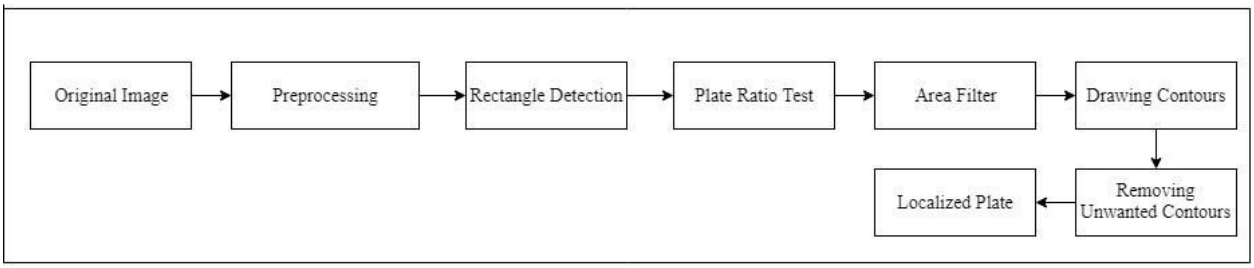


fig : schematic diagram of plate localization

4.3 character segmentation

The detected plate undergoes a series of operations to extract the region of interest(characters) from it. These operations include a series of tests and filters.

-preprocessing of the detected license plate image

-Border cleaning: border noise is removed by eliminating certain part of the image border which is of no use

- Average area filter: the characters inside the license plate have relatively similar areas. average area filter filters the bounded rectangles which are below a threshold value. This is useful as many small noises are present in the image.

- segmenting bounded rectangles satisfying the constraints

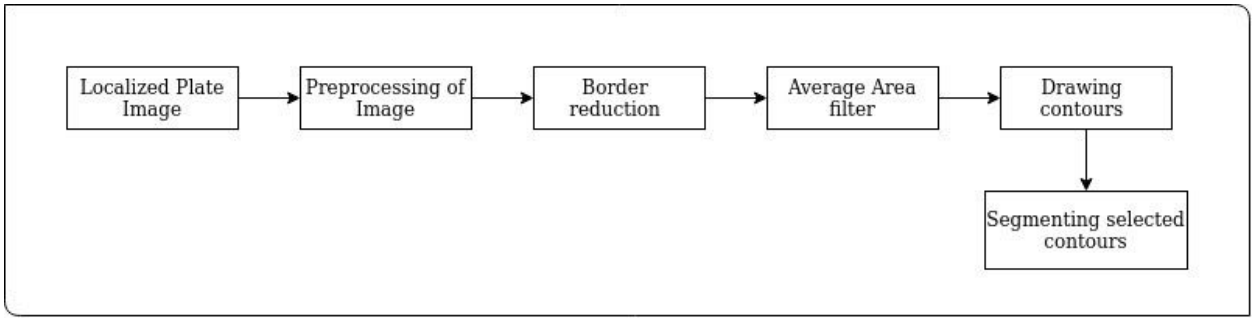


fig : schematic diagram of character segmentation

4.4 convolutional neural network(CNN)

A classic use case of CNNs is to perform image classification.They're basically just neural networks that use Convolutional layers, which are based on the mathematical operation of convolution. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

4.4.1 convolution layer:

CNNs include conv layers that use a set of filters to turn input images into output images. A conv layer's primary parameter is the number of filters it has.When programming a CNN, the input is a tensor with shape (number of images) x (image width) x (image height) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map width) x (feature map height) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

- Convolutional kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has

10,000 weights for *each* neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

In our case (3x3) filter is used. To use convolution operation and create convolution layers Tensorflow keras sequential model is used.

4.4.2 Pooling

Neighboring pixels in images tend to have similar values, so conv layers will typically also produce similar values for neighboring pixels in outputs. As a result, much of the information contained in a conv layer's output is redundant.Pooling layers solve this problem. All they do is reduce the size of the input it's given by (you guessed it) *pooling* values together in the input. The pooling is usually done by a simple operation like max, min, or average. *Max pooling* uses the maximum value from each of a cluster of neurons at the prior layer.*Average pooling* uses the average value from each of a cluster of neurons at the prior layer.

4.4.3 Fully connected layer

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

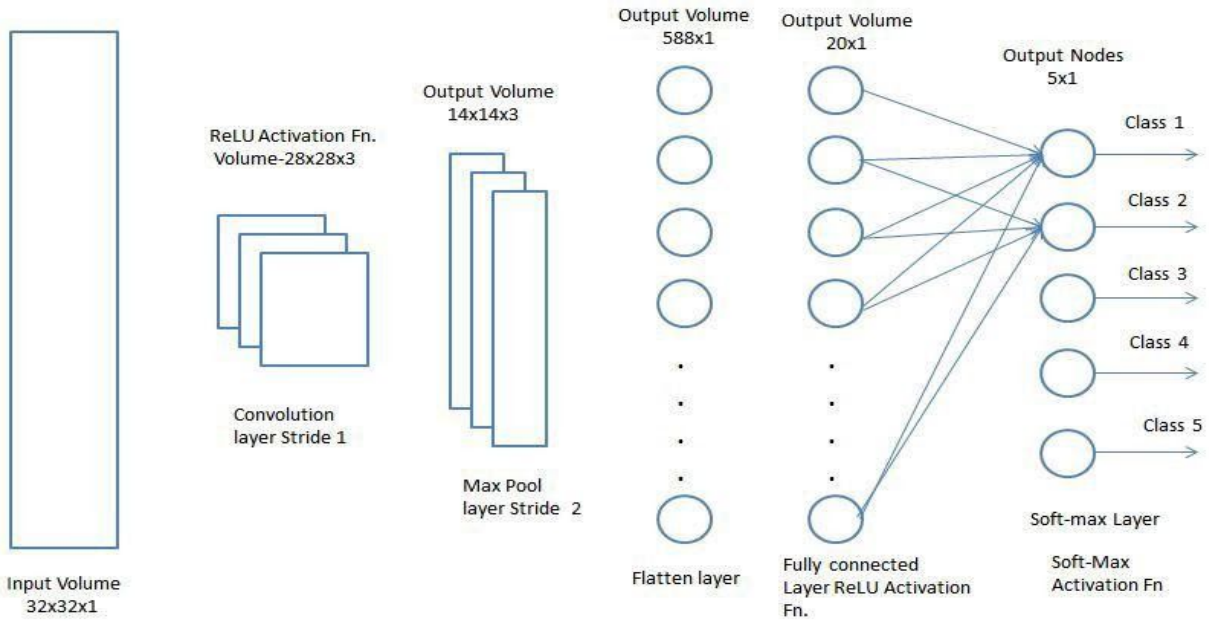


fig: diagram showing convolution operation and fully connected layer

After the convolution and pooling layers, our classification part consists of a few fully connected layers. However, these fully connected layers can only accept 1 Dimensional data. To convert our 3D data to 1D, we flatten the 3D feature map that we previously obtained from the convolution and pooling layers. This arranges our 3D volume into a 1D vector.

The last layers of a CNN are fully connected layers. Neurons in a fully connected layer have full connections to all the activations in the previous layer. These fully connected layers act as a regular ANN(Artificial Neural Network). These layers also use the RelU activation function for non-linearity.

The last fully connected layer is a SoftMax classifier that produces a probability distribution for the classes of images. These probabilities indicate how well the model has classified the image. The softmax function is given by:

$$f_i(\vec{a}) = \frac{e^{a_i}}{\sum_k e^{a_k}}$$

4.5. Model chosen for this project

Model overview:

model type = sequential

Number of convolution layers = 3

Number of nodes in each convolution layer = 128

Filter size = (3x3)

Number of pooling layers = 3

Max Pooling size = (2x2)

Fully connected layer = 1 (12 nodes for 12 classes)

loss function = categorical cross entropy

Activation function = relu

Classifier = softmax

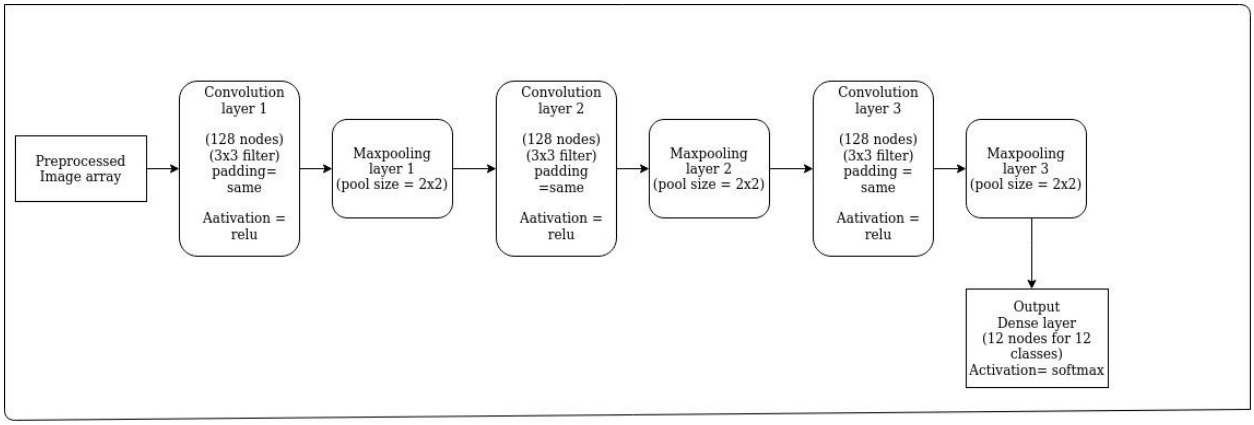


fig : schematic diagram of CNN model

4.6 libraries and functions used:

4.6.1. TensorFlow - Used for CNN model training and testing

libraries	**functions**
tensorflow.keras.models	**Sequential()** - to create sequential cnn model
tensorflow.keras.callbacks	**TensorBoard()** - to visualize metrics
tensorflow.keras.layers	**Conv2D()** - to create convolution layers **Activation()** - to create non-linear data **MaxPooling()** - to use the maximum value from each of a cluster of neurons at the prior layer **Flatten()** - converts to 1-D array **Dense()** - to create dense layers **Dropout()** - to randomly trigger a neuron
tensorflow.keras.utils	**to_categorical()** - used for multi-class classification

4.6.2. OpenCV and imutils - Used for image processing

libraries	**functions**
cv2	**GaussianBlur()** - to apply gaussian filter to

	image **cvtColor()** - convert BGR to gray , BGR to HSV **threshold()** - to create binary image **findContours()** - to find contour points in image **bitwise_and()** - to mask the image **arcLength()** - to calculate perimeter of contours **approxPolyDP()** - to approximate the polynomial type **boundingRect()** - to bound rectangles from contour points **morphologyEx()** - to apply morphological transformations like erosion, dilation, chain approximation
imutils	**resize()** - to resize image proportionally

4.6.3. numpy , pickle

libraries	**functions**
numpy	**array()** - for array manipulation **average()** - to calculate average **reshape()** - to convert data to 1-D array
pickle	**open()** - to open pickle file **close()**, **dump()** **load()** - to load saved data

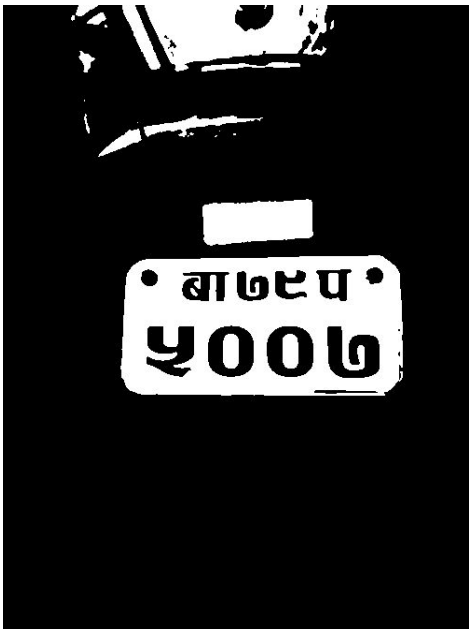
5. Experiments

Experiments were conducted on rear images of bikes and scooters. For experimental data various images were captured of the local vehicles. Only private vehicles of the bagmati zone are taken into consideration due to lack of enough datasets. Such an image is given as an input to the system and the system performs a series of procedures to segment the image and feed it to the convolutional neural network(CNN). CNN then fits the input to the pretrained model and predicts the characters on the license plate.

5.1 Preprocessing of input image

original image

original to grayscale conversion



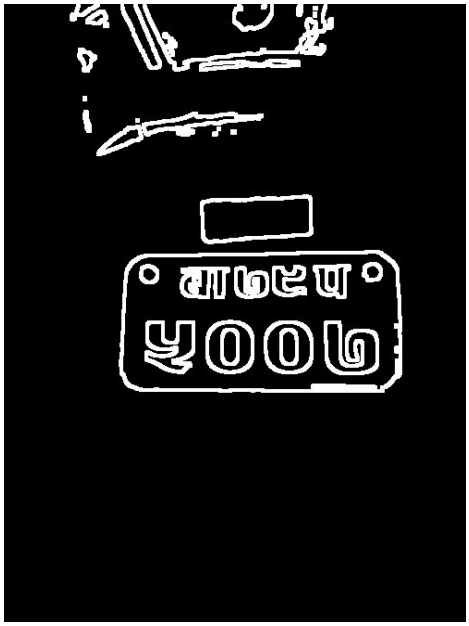
thresholding of image using BINARY + OTSU

5.2 Localization of license plate

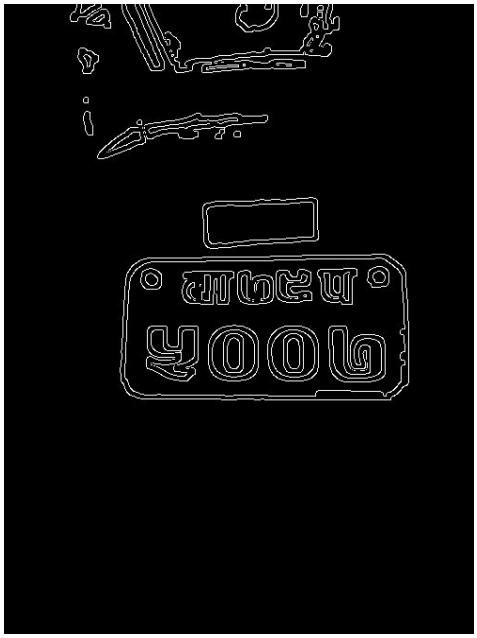
original image

BGR to HSV conversion of image with red masking

thresholding of image



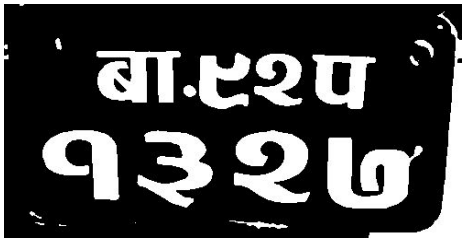
gradient transformation on image



canny edge detection

localized plate from input image

5.3 Character segmentation



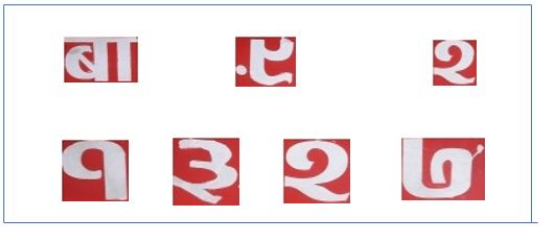
preprocessing of plate image

border reduction of preprocessed image

applying average area filter

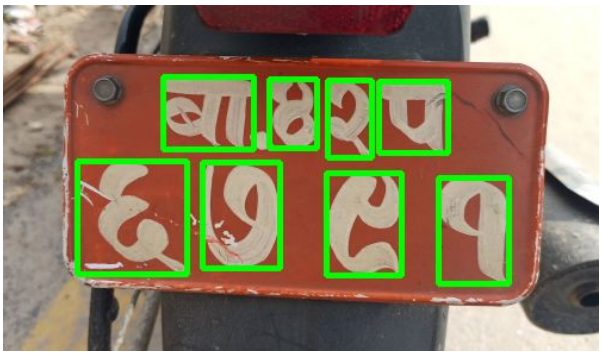
erosion and dilation morphological transformation

drawing contours on region of interest



segmenting selected bounded region

5.4.CNN prediction

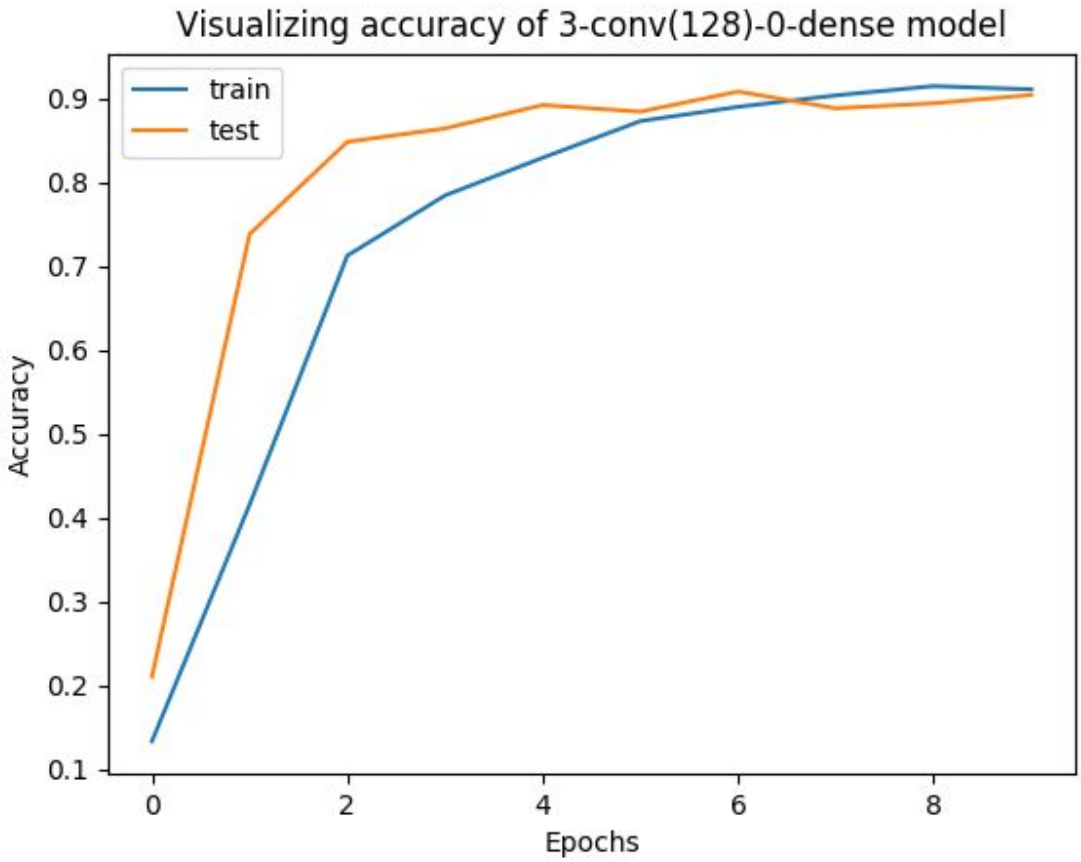
Segmented plate	model output	actual output
	BAA 92 PA 1327	BAA 92 PA 1327
	BAA 79 PA 5007	BAA 79 PA 5007
	0 82 PA 6791	BAA 42 PA 6791
	BAA 94 PA 1036	BAA 94 PA 1036

	x 78 PAA 9133	BAA 78 PA 9133

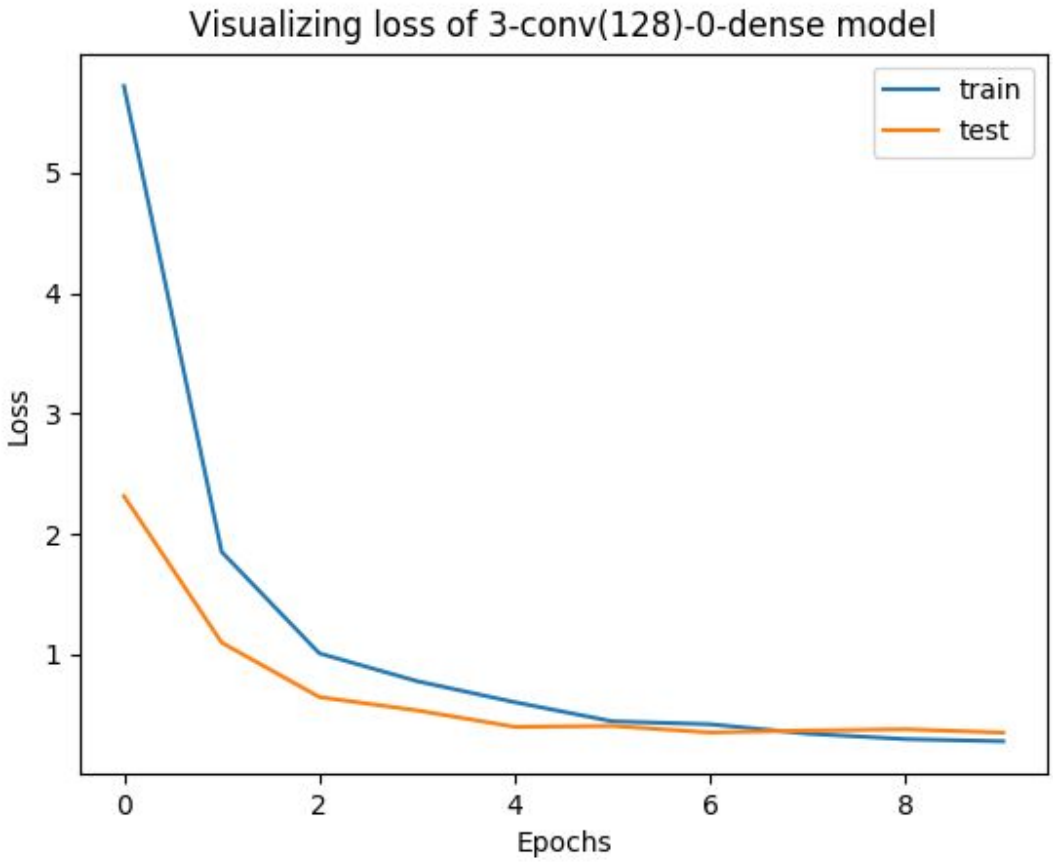
6. Evaluation of results

6.1 Accuracy

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Accuracy is a great measure but only when there is symmetric datasets where values of false positive and false negatives are almost the same. Therefore, other parameters should be considered to evaluate the performance of the model. For our model, we have got **0.904** which means our model is approx. **90%** accurate.



accuracy plot for training and testing data



loss plot for training and testing data

6.2 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.High precision relates to the low false positive rate. We have got **0.933** precision which is a good result.

6.3 Recall(Sensitivity)

Recall is the ratio of correctly predicted positive observations to the all observations in actual class.We have got recall of **0.885** which is good for this model as it's above **0.5**.

6.4 F1 score

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if there are uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. In our case, F1 score is **0.908**. **F1 Score = 2*(Recall * Precision) / (Recall + Precision)**

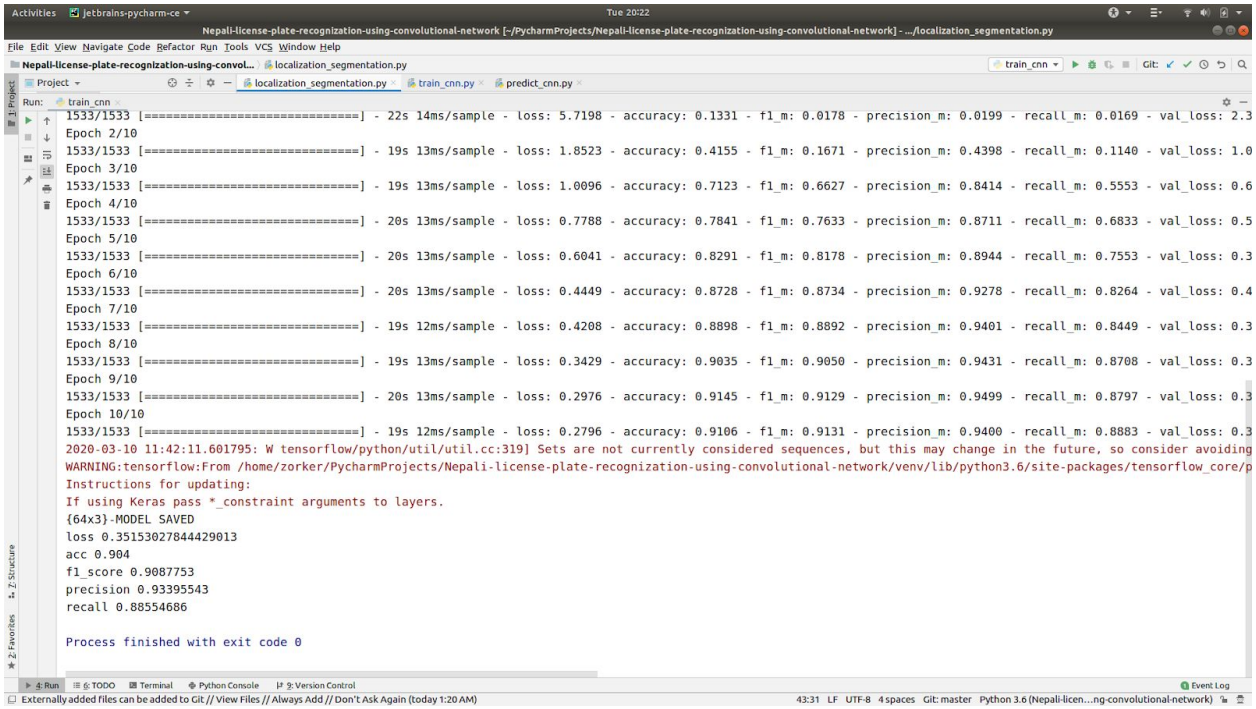
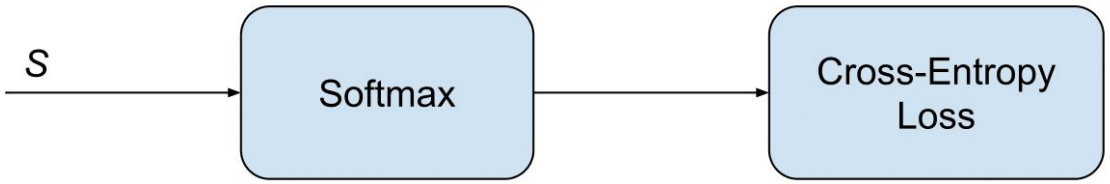


fig training, evaluation metrics

6.5 Loss function

Also called **Softmax** Loss. It is a **Softmax activation** plus a **Cross-Entropy loss**. If we use this loss, we will train a CNN to output a probability over the C classes for each image. It is used for multi-class classification.



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

In the specific (and usual) case of Multi-Class classification the labels are one-hot, so only the positive class Cp

keeps its term in the loss. There is only one element of the Target vector t which is not zero ti=tp

. So discarding the elements of the summation which are zero due to target labels, we can write:

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

Where **Sp** is the CNN score for the positive class.

Defined the loss, now we'll have to compute its **gradient respect to the output neurons** of the CNN in order to backpropagate it through the net and optimize the defined loss function tuning the net parameters. So we need to compute the gradient of CE Loss respect each CNN class score in s. The loss terms coming from the negative classes are zero. However, the loss gradient respecting those negative classes is not cancelled, since the Softmax of the positive class also depends on the negative classes scores.

The gradient expression will be the same for all C except for the ground truth class Cp, because the score of Cp (sp) is in the numerator.

After some calculus, the derivative respect to the positive class is:

$$\frac{\partial}{\partial s_p}\left(-log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)\right) = \left(\frac{e^{s_p}}{\sum_j^C e^{s_j}} - 1\right)$$

And the derivative respect to the other (negative) classes is:

$$\frac{\partial}{\partial s_n}\left(-log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)\right) = \left(\frac{e^{s_n}}{\sum_j^C e^{s_j}}\right)$$

Where sn is the score of any negative class in C different from Cp.

7. Discussion on results

Evaluation metrics show the efficiency of the model. The obtained metrics shows that our model is good enough with 89 -91% validation accuracy. The validation loss obtained for this model is approx. 0.34 which is a good result for our CNN model. The accuracy can be altered by performing a series of changes in the model. Our model classifies characters correctly only in case of rear images of private bikes of bagmati zone. with this accuracy a more generalized model can be made. we opted to apply such constraints due to lack of enough datasets.

8. Contributions of each group members

Bipesh Subedi

research and study

logical design

character segmentation

Modelling, training and testing the model

Milan Thapa:

research and study

preprocessing of image

datasets collection and manipulation

localization of license plate

9. Code

Link to the GitHub repo of the code

https://github.com/bipeshrajsubedi/Nepali-License-Plate-Recognition-Using-CNN.git

10. Conclusion and Future Extensions to the Project

License plate recognition is a wide field which can be implemented using many different algorithms and techniques. Every method has its own advantages and disadvantages. Our proposed methodology initially does the pre-processing steps which includes RGB to grayscale conversion, noise removal, and binarization of the image. After which the license plate is extracted using Sobel's edge detection algorithms. Then the characters are segmented using horizontal scanning which is given as input to the CNN in order to recognize the character correctly. Training our system with the help of ANN made our system more reliable and efficient in order to recognize the characters correctly.

Although we can see that so many algorithms have been implemented in various previous projects, in order to make a robust system for automatic license plate recognition, there are still many loop holes left in the system which can be filled in order to make the system more future-proof and reliable.

Our project however works on the simple font styles which is being used normally on license plates of the cars as per the rules made by the governing bodies of traffic department. But in order to handle the cases where people don't follow these rules, it can be handled in future projects being implemented in this field of license plate recognition.

Some of the various fields which can be explored in this project are as follows:

- Car model recognition
- Multilingual character recognition
- Fancy character recognition etc.

Reference

[1] Kaushik Deba, Md. "Ibrahim Khana, Anik Sahaa, and Kang-Hyun Job,“An Efficient Method of Vehicle License Plate Recognition Based on Sliding Concentric Windows and Artificial Neural Network”." Science Direct, CSIT-2012.

[2] Khalil, M. I. "Car plate recognition using the template matching method." International Journal of Computer Theory and Engineering 2.5 (2010).

[3] Parasuraman, Kumar, and P. Vasantha Kumar. "An efficient method for Indian vehicle license plate extraction and character segmentation." IEEE International conference on computational intelligence and computing research. Vol. 18. 2010.

[4] Mukesh Singh Chandel, Satya Verma “An Implementation of Fast and Efficient Real Time Indian Standard High Security Number Plates Recognition and Detection System of Stolen Cars using RTOS” (2018).

[5] Nwe Ni Kyaw, G. R. Sinha, Khin Lay Mon” License Plate Recognition of Myanmar Vehicle Number Plates A Critical Review” (2018).