OCR based Automatic Number Plate Recognition (ANPR) using Artificial Neural Network

Abhash Jain, Abhishek Kumar Srivastava, Ajay Venkatakrishnan, Darshan Bhandari, Deepak Gupta, Kamal Sharma

Department of Computer Science
North Carolina State University, Raleigh, North Carolina, USA
Email: {ajain28, asrivas3, avenka12, dbhanda, dgupta22, ksharma5}@ncsu.edu

***Abstract*—** Increasing amount of traffic has become a major problem in every country because enforcing traffic rules and monitoring them has become more difficult with the increasing volume of traffic. An automated, fast, accurate and robust vehicle plate recognition system has become need for the traffic control and enforcement of traffic regulations and the solution to this problem is ALPR. This project is dedicated to implement an OCR based license plate recognition using artificial neural network trained on the dataset of object features. The whole system can be categorized under three major modules, namely Number Plate Localization, Plate Character Segmentation and Plate Character Recognition. The system is simulated on 50 national and international motor vehicle number plate images and results obtained justifies the its intended requirement.

Keywords - Artificial Neural Network(ANN), Convolutional Neural Network(CNN), Optical Character Recognition(OCR), Plate Localization, Character Segmentation, Morphological Operations.

I. INTRODUCTION

In an age where everything is automated, the need for automating the detection of number plates in traffic signals, toll plazas and for security purposes becomes an essential tool that helps to solve a problem. This can be achieved with the help of Artificial Neural Networks and OCR (Optical Character Recognition).

This project is divided into 3 segments:
1) Number plate localization
2) Character Segmentation
3) Character Detection (using Artificial Neural Network)

In a real scenario image sensor capture a vehicle and this will be fed to our first stage, where we localize only the number plate from the entire image. Then we segment the localized number plate to individual characters and these characters are used as input to Neural Network (3rd stage). This final stage detects each character and outputs the final license plate number as a string.

This report has the following sections. After the introduction we have:
1) Related work which summarizes all the papers that we have read and are relevant to our project.
2) Methodology: This section describes the method used to implement the project.
3) Implementation: This consists of descriptive details of our implementation
4) Results: This section summarizes the evaluation performed using our model
5) Conclusion: This concludes the learning from the project including the objective achieved.

II. RELATED WORK

Number Plate Localization is a difficult task as various factors such as plate condition, light condition and weather conditions hamper efficiency of the Plate Localization Algorithm. Various Algorithms are proposed overtime to counter such factors and give efficiently localize plate. The Algorithm proposed in [1][2] use clipping of the signals generated from the intensity distribution if the image and use Pearson's correlation coefficient to correctly identify the plate. There are other Algorithms that use Morphological Operations and Connected Components Analysis to efficiently localize the plate [3] from the vehicle image which also gives high accuracy of finding number plate from the images. Apart from these two algorithms some new algorithms are also proposed that use Sobel and Threshold Algorithms [4] to localize the plate which also gives high accuracy under certain given conditions. Some Other Algorithms also exist that use Sliding Window along with Machine Learning Models on the Input Image this method is by far the most accurate algorithm but is very computationally expensive.

Character Segmentation on the Number Plate is also an important task as the output from this is used to predict the characters. Several Algorithms are proposed overtime to efficiently perform it. The Algorithm proposed in [1] uses Dilation Method to extract the Characters from the image. The Algorithm proposed in [4] uses Morphological Operations and Contours to Extract the segmented characters.

OCR is the also very tricky task as appropriate Method and Model must be chosen to accurately predict the Characters. [1] uses Artificial Neural Network trained with character features to predict the characters. While [4] uses SVM as the learning algorithm to predict the characters from the image.

III. METHODOLOGY

First step in automatically recognizing number plate is to preprocess the image and localize the license plate. The localized plate is then segmented into individual characters which is then fed into feed forward Artificial Neural Network (ANN) [1].

A. Preprocessing and Localization

Image is converted into gray scale and blurred to remove noise points. Then vertical edges are detected from the blurred image using Sobel Edge Detection Algorithm. After this, vertical band clipping is done to detect potential vertical bands with license plates and then correlation between these bands and ideal license plate is calculated to identify the band with license plate. Now for horizontal band clipping is done by binarizing the image using Otsu method and connected component analysis is used to detect the license plate area from the band. Thus, retrieving license plate from the image. After this, characters are segmented from the plate by identifying contours of the characters in the plate and extracting them.

B. Character Prediction

The output of the first step, i.e. (Images of individual characters) is used as an input this step. The built ANN model is trained with images of individual characters.

1) Each image is converted to 1D array and provided as input to the neural network for the feed forward phase
2) Error is calculated at the end of the feed forward phase.
3) Error is backpropogated through the layers using the backpropogation algorithm.
4) Weights are updated using Stochastic Gradient Descent/ Adam Optimization algorithm.

IV. IMPLEMENTATION

Project is implemented in Python2.7 using libraries – OpenCV, NumPy, SciPy, tqdm, scikit, Hickle and Pandas. The architecture has two basic modules – preprocessing and character recognition using ANN.

A. Preprocessing

First step in preprocessing is to localize the number plate. For this the image is first converted to gray scale to reduce the impact of shadows and reflections in the image.

Fig 1: RGB and Gray Scale image

The gray image is then transformed to binary image using Sobel edge detection technique for better processing. The pixel represented by the cell y in the destination image (Fig. 2) is affected by the pixels $x_0 ... x_8$ according to the formula

$$y = x_0 * m_0 + x_1 * m_1 + x_2 * m_2 + x_3 * m_3 + x_4 * m_4 + x_5 * m_5 + x_6 * m_6 + x_7 * m_7 + x_8 * m_8$$

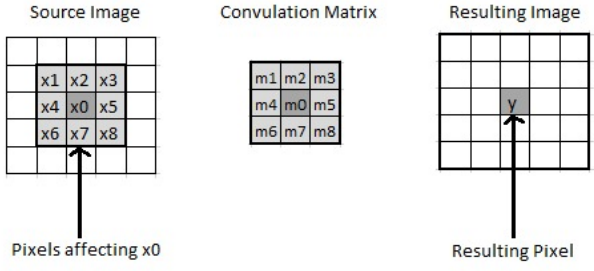


Fig 2: pixel affected by neighbors according to convolution matrix

Convolution matrix used in sobel edge detection is

$$Gx = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Fig 3: Image after Sobel Edge detection

The image is then projected along the y-axis and a plot is further generated. The generated plot is then blurred to get smooth peaks.

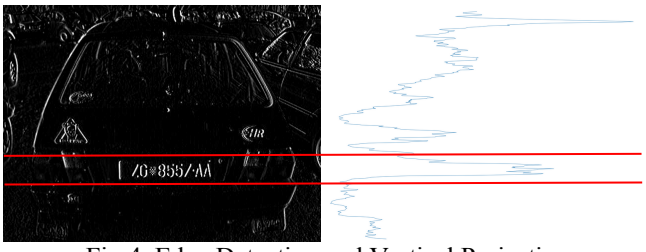


Fig 4: Edge Detection and Vertical Projection

These projections are calculated by using the mathematical formula:

$$p_y(y) = \sum_{i=0}^{w-1} f(i,y)$$

Here, f(i,y) gives the intensity of image at the i^{th} row and y^{th} column, and w is the width of the image.

The peaks in vertical projection correspond to the bands with possible candidates for number plates. According to paper, most of the images containing complex background has number plate in top three variation curves in the vertical projection. But according to our analysis, selecting top four increased the probability of finding vertical band with number plate. So top four variation curves were selected by using local maxima [2]:

$$y_{bm} = \arg\max_{y_0 \le y \le y_1} \{p_y(y)\}$$

$$y_{b0} = \max_{y_0 \le y \le y_{bm}} \{y | p_y(y) \le 0.55 * p_y(y_{bm})\}$$
$$y_{b1} = \min_{y_{bm} \le y \le y_1} \{y | p_y(y) \le 0.55 * p_y(y_{bm})\}$$

where $y_0 = 0$ and y_1 = height of the image

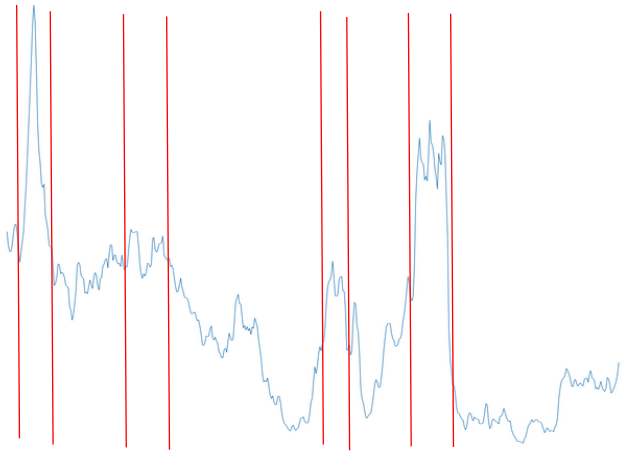


Fig 5: Top four variation curves in vertical projection

The correlation of ideal image is calculated with these top four variation curves by using Pearson correlation. The variation curve with maximum absolute value of correlation has the highest probability of containing number plate.

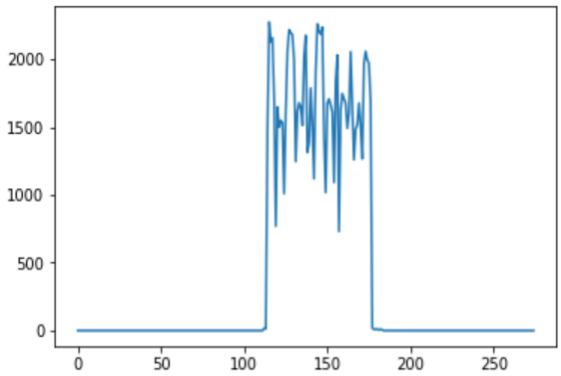


Fig 6: Plot of ideal Plate

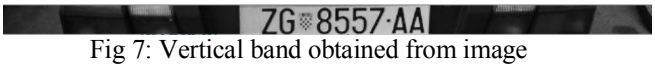


Fig 7: Vertical band obtained from image

For horizontal segmentation the process used for vertical segmentation will not work as the characters in the plate produce more number of fluctuating peaks in the projection on x-axis. Hence, we used Morphological Operations to segregate plate image from the complete band [3].

First, we used adaptive histogram equalization to enhance the contrast of the image then we performed Morphological opening using square shaped structuring element. Then subtracted this new image from the enhanced image. This removed the noise points from the image.

Next, we performed Image Binarization by using Otsu's method. After binarization we used Sobel Algorithm to find vertical edges and then to fill the holes in the image we used dilation then we used morphological opening and erode operations for exact detection of candidate plate area. After this, we may get more than one candidate areas. To remove the unnecessary areas and identify the correct area we used Connected Component Analysis and identify the correct blob with license plate. And then extracted the image by using the row and column of the extracted blob.

Fig 8: Number plate after horizontal clipping of vertical band

This number plate image is then segmented to give images of individual characters. This is done by first resizing the image to a standard size of 140 * 33 pixels. Then the contours are found in this resized plate image [4]. These contours are then validated. Those contours which have size greater than height width ratio of 0.2 and area 60 are selected. The selected contours are cropped from the plate image and converted to individual images of size 40 * 40 with background as white and characters written in black. These segmented images are then fed into ANN for optical character recognition.

B. Character Recognition using ANN

PREPROCESSING

We initially pre-process all the training images and store them in a CSV file. This pre-processing step converts all images to black and white and then resize them to 25*25 pixel images using OpenCV. The 625 pixel values are then written to a CSV file. The size of the CSV file for the 48000 images comes around to 173 MB.

TRAINING

For training the images we used Artificial Neural networks (ANN). In our ANN model, we have provided the user with 3 different activation functions:

a) Sigmoid
b) Relu
c) Tanh

Hyper Parameters

a) Batch Size: 16 – This is the size of the images processed in a single feed to the neural network.

b) Learning rate = 0.01 – This is the learning rate that controls the weight updates in the gradient descent algorithm.
c) Number of layers = 2
d) Hidden Nodes = Configured by user (default: 400 and 300)
e) Epochs: Configured by user (default:100)
f) Optimization algorithm: SGD, Adam
g) Activation function: User Configured (Default: sigmoid)

All default parameters were chosen after multiple training iterations on training data to maximize accuracy.

Architecture

We have 2 hidden layers which have a N hidden nodes in 1st layer and M hidden nodes in the 2nd layer. (Default: N = 400, M = 300)

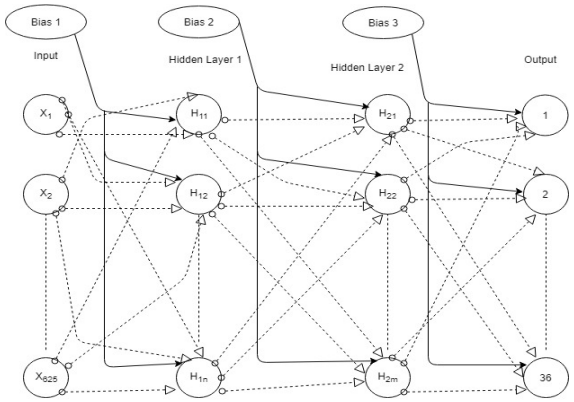


Fig 9: Architecture of ANN

Back-propagation

We use the back-propagation algorithm to update the weights. The error at the final layer is back-propagated through the layers using back-propagation. The error at the final layers is measured using:

error = Y – Y'

where Y - Output from neural network

Y' - Expected output

This error is back-propagated through the layers using the back-propagation algorithm.

We have also provided the following optimization algorithms:

a) Stochastic Gradient Descent: This was calculated using the formula below to update the weights

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

The above formula takes the gradient as the input and updates the weights over multiple iterations until the weights converge to global optimum value.

b) Adam: The Adam optimization algorithm is an improvement over the simple SGD algorithm. It helps update the weights and converges much faster than SGD. The following formulas can be used to update the weights:

$$m_t = \beta_1 m_{(t-1)} + (1 - \beta_1) g_t$$

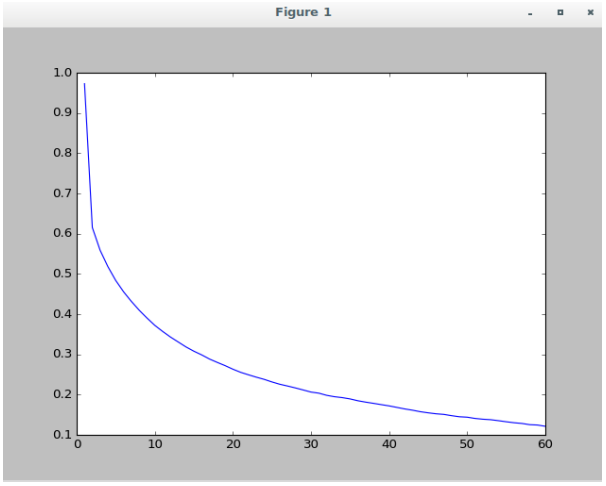
$$v_t = \beta_2 v_{(t-1)} + (1 - \beta_2) g_t^2$$

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon} \widehat{m_t}$$

We can observe that the loss on the training data decreases through the following graph:



X-axis – Epochs, Y-axis -- Loss

Fig 10: Graph between epoch and loss after each epoch

FINAL PROCESSING

After training the data, we store the weights (model) in a Hickle file in HDF5 (Hierarchical Data format). This allows the user to use a previously trained model on new test images instead of training the model for every new test input.

At the end of the training our model will be ready to predict the individual alphanumeric characters (0-9, A-Z). The final output will be a list of probabilities of all the characters and the character with maximum probability is chosen as the label.

COMPARISON WITH LIBRARY

To ensure that our model performs well, we have compared it with scikit's MLP Classifier and we have observed that our model achieves accuracy similar to that of

the library. We have also explored Convolutional Neural Networks using Keras. As ANNs performed well enough for this task we decided to integrate it with our project.

INTEGRATION OF NEURAL NETWORK WITH PREPROCESSED TEST IMAGES

The pre-processed step segments each character as explained before in separate image file. Each character is picked up by the algorithm and fed to the trained network model to predict the final label. In this way for all characters in a single license plate and return it to the user. In our implementation we provide two options to the user:

a) Predict license plate on already optimized model
b) Train a new model based on default values or user configured parameters.

V. RESULTS

The dataset includes images of cars with number plates clicked in different conditions – images clicked with some angles, night images, images clicked from large/small distance, clean/smudged plates. Following are the results observed after testing:

	Accuracy (in %)
Plate Localization	78.9
Character segmentation	60
Character recognition from segmented characters	55.66

Plate Localization accuracy is reduced due to reasons such as reflection around number plate, if image is clicked from side view, stickers are present near the number plate, appropriate light is absent around number plate, if image is taken at night time, and if large logo are present around license plate.

Accuracy of character segmentation highly depends upon the design of number plate. If number plate has large distance between characters or contains any symbols between characters, then the segmentation accuracy decreases.

Accuracy of character recognition by ANN highly depends upon the quality of segmented characters. If the segmented characters are highly deformed or not properly segmented, or there is similarity created in characters due to used fonts (like '0' or 'O', 'B' or '8') then the ANN is unable to predict the characters accurately.

VI. CONCLUSION

Image Characterization is a highly used in Image processing. In the first stage our project aimed to take a vehicle image from the image sensor and perform localization followed by segmentation of image into individual characters. The second stage involved OCR using ANN which was the performed on the segmented characters to predict the Number Plate. In our evaluation we found out that ANN with 2 layers outperforms other simple classification algorithms.

REPOSITORY

Link to project code: https://github.ncsu.edu/dgupta22/ALDA-Project.
Data used to train ANN:

a) http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/EnglishFnt.tgz
b) http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/EnglishHnd.tgz
c) http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/EnglishImg.tgz
d) https://drive.google.com/file/d/1dSHrk0BsC06fNIgI2PhMf02jyhgDqVlf/view?usp=sharing

Data used for teting: https://github.ncsu.edu/dgupta22/ALDA-Project/tree/master/testImages

REFERENCES

[1] Bhavin V. Kakani, Divyang Gandhi, Sagar Jani, "Improved OCR based Automatic Vehicle Number Plate Recognition using features trained Neural Network." IEEE-40222, ICCCNT 2017

[2] Ondrej Martinsky, Algorithmic and mathematical Principles of automatic number plate Recognition systems, Brno University of technology, 2007.

[3] Sarbjit Kaur, Sukhvir Kaur , "An Efficient Approach for Number Plate Extraction from Vehicles Image under Image Processing." IJARCS, 2014

[4] "Mastering OpenCV with Practical Computer Vision Projects" by Daniel Lelis Baggio, Packt Publishing Ltd.

MEMBER CONTRIBUTIONS

Abhash Jain: Designed and Implemented ANN with library to compare the accuracy and fine tuning the hyper parameters of the ann. In the middle of ANN implementation studied about CNN to explore the possibility of increasing the accuracy of OCR. My starting few days was invested in reading and learning about neural network prior to implementation. For this I have done lot of reading as I was new to this field. Worked with the Deepak and Abhishek to integrate their code with our ANN model. I have also written integration interface for ANN module. From the start of project topic finding to writing the report, it had taken me 11 weeks. I have also helped in integration testing to get results and analyzed it for report.

Abhishek Kumar Srivastava: Played part in choosing the Research Paper for this project and designing the project architecture. Then I contributed and coded the part of vertical clipping of plate area in the Data Preprocessing and designed and coded the horizontal clipping of number plate from the vertical band. For the vertical plate clipping I read several papers and implemented them, as the original paper did not have enough explanation for the implementation and choose the best one and implemented it in project. I also contributed in defining the process of setting up the environment for the execution of the project. I also coded and resolved the issues that were faced during Integration. I also contributed in writing various parts of report like Related Works, Methodology, Implementation etc. I also performed Unit and Integration Testing of the project and compiled the results.

Ajay Venkatakrishnan: Started for looking out whether to implement ANN or CNN. After discussing with other team members, we decided to go with ann. Implemented ANN, accompanied with Abhash and Darshan. Explored the possibilities of different optimization algorithms, and other hyper parameters and implemented them. To evaluate the performance, I collected the data and generated the loss vs epoch graph. I had completed the ANN section in the report along with Abhash and Darshan. The entire project was discussed and implemented within 3 months.

Darshan Bhandari: Identified the different ways to preprocess the image to feed our neural network and helped Abhash find the perfect tuning of hyper parameters. Made the test dataset for our model. Initial few days I was learning about ANN and CNN, which is suitable for our implementation. To test this project, I created an environment form scratch which consisted all the required libraries and made the corresponding READ_ME file with Abhash and Ajay. The entire duration of the project which included searching, development, enhancement and testing took me and the team 3 months.

Deepak Gupta: I contributed in zeroing in on research paper for the project and in making the design for project architecture. As part of pre-processing module team, I contributed in writing code for performing initial pre-processing on input image, vertical band clipping and segmentation of clipped number plate. I also wrote the code for integration interface. During integration, I contributed in fixing integration issues. I also contributed in testing by writing script for automated testing and by performing unit testing of pre-processing and localization code. Further, I contributed in aggregating the results of testing and analyzing them. I contributed in writing results along with methodology, implementation section related to pre-processing and localization module in report, and aggregating and formatting of sections of the report.

Kamal Sharma: Started with finding research paper for our project. I contributed in implementation of image pre-processing module as me, Abhishek and Deepak worked on this phase of project. I contributed in writing code for the segmentation of clipped number plate. I also helped Abhishek and Deepak in vertical band clipping of given image. Apart from that I made the dataset for our part. I contributed in testing of our project and in analyzing the results from testing to evaluate the model performance. In addition to this I helped in writing the implementation part of the project report and in making the poster.