International Journal of Information Retrieval Research
Volume 9 • Issue 1 • January-March 2019

Real Time License Plate Recognition from Video Streams using Deep Learning

Saquib Nadeem Hashmi, Jaypee Institute of Information Technology, Noida, India
Kaushtubh Kumar, Jaypee Institute of Information Technology, Noida, India
Siddhant Khandelwal, Jaypee Institute of Information Technology, Noida, India
Dravit Lochan, Jaypee Institute of Information Technology, Noida, India
Sangeeta Mittal, Jaypee Institute of Information Technology, Noida, India

ABSTRACT

With ever increasing number of vehicles, vehicular management is one of the major challenges faced by urban areas. Automation in terms of detecting vehicle license plate using real time automatic license plate recognition (RT-ALPR) approach can have many use cases in automated defaulter detection, car parking and toll management. It is a computationally complex task that has been addressed in this work using a deep learning approach. As compared to previous approaches, license plates have been recognized from full camera stills as well as parking videos with noise. On a dataset of 4800 car images, the accuracy obtained is 91% on number plate extraction from images, 93% on character recognition. Proposed ALPR system has also been applied to vehicle videos shot at parking exits. Overall 85% accuracy was obtained in real-time license number recognition from these videos.

KEYWORDS

Automated License Plate Detection, Character Segmentation, Computer Vision, Convolutional Neural Network (CNN), Deep Learning, Image Processing, Morphological Operations, OCR, Template Matching

1. INTRODUCTION

Automatic License Plate Recognition(ALPR) is a technique to extract the license plate number from a still image or a video of a moving or stationary vehicle. It is a useful approach for vehicle surveillance. Robust ALPR has many use cases in combating thefts, illegal vehicles classification, customized electronic toll collection, cataloguing the movements of traffic in a premise, catching speed limit violators, determining what cars belong in a parking garage, expediting parking by eliminating the need for human confirmation of parking passes and many more (Chang, Chen, Chung & Chen, 2004; Du, Ibrahim, Shehata, & Badawy, 2013).

License plate detection and recognition is a challenging problem due to issues like noisy image inputs, occlusion, vehicle orientations, different license plate types, extra images on number plates, non -standard sizes, poor quality of camera among others (Du et al., 2013) Most of the existing solutions make simplistic assumptions as compared to real scenarios i.e. work for stationary cameras, with a specific viewing angle, at a specific resolution, for a specific type of license plate template (Chang et al., 2004; Du et al., 2013; Łubkowski & Laskowski, 2017; Björklund, Fiandrotti, Annarumma, Francini, & Magli, 2017; Khan, Shah, Wahid, Khan & Shahid, 2017). The proposed ALPR model addresses these challenges to a large extent. It works for cases like slightly blurred images, images

DOI: 10.4018/IJIRR.2019010105

at a distance, license plate oriented at different angles up to 15 degrees (approx.), multi-coloured license plates, text other than registration number on the license plate.

A typical ALPR system consists of four steps namely Vehicle image capture, Number plate extraction, Character segmentation and Character recognition (Chang et al., 2004; Du et al., 2013; Björklund et al., 2017; Panchal, Hetal, & Panchal, 2016; Azam & Islam, 2016; Yuan et al., 2017). But since our ALPR model is designed to work on videos, therefore, one more step is added at the beginning that is to split the video into frames. In general, the first step i.e. to capture an image of the vehicle is quite an exigent task as it is very difficult to capture image of a moving vehicle in real time, in such a manner that none of the components of vehicle especially the number plate is missed. But our implementation does not rely on single image, rather author's generated multiple frames (still images of which the video is composed) from the video and find possible license plate number corresponding to each frame. Then the registration number with maximum number of occurrences is chosen as the final output. The success of the fourth step of character recognition depends on how the second and the third steps are able to locate vehicle number plate and separate each character.

Template matching has been used here to find out a license plate from vehicle's image. The values of different parameters like minimum and maximum pixel width, aspect ratio, change in width and height, minimum and maximum contour area etc. have been hardcoded and a list of possible license plates is formed according to these parameters (Chang et al., 2004; Du et al., 2013; Łubkowski & Laskowski, 2017). Finally, a CNN classifier has been applied to every possible plate for Optical Character Recognition (OCR) (Chang et al., 2004; Björklund et al., 2017; Azam & Islam, 2016; Yuan et al., 2017; Khan et al., 2017) and obtain result corresponding to each possible plate. Then the plate with maximum number of characters recognized is selected as the final license plate. The two steps of license plate detection and OCR individually gave good accuracy (93% for CNN classifier and 91% for license plate detection). The accuracy of final combined model is 85%.

The steps involved, analysis of the proposed approach and motivation of this work has been given in detail in rest of the paper in 5 sections. Section 2 described the related work in this area and sets the motivation for this work. Architecture and algorithms for automated license plate recognition has been described in Section 3. Results of testing the algorithms on third party and self-created databases have been discussed in Section 4. The paper is concluded in Section 5.

2. RELATED WORK

Various approaches have been proposed in literature for detection, localization and classification of different objects in images. ALPR problem is about recognition of a vehicle's license plate number from images. The solution requires extensive image processing involving object detection and pattern recognition. Some of the challenges in correct number recognition are plate variations across countries, location of plate within image, different sizes, various colour combinations, disparate font sizes, occlusion, inclination and contamination of plate due to other frames, images or screws. Apart from image variations, amount of light and background may further make problem worse (Du et al., 2013). Łubkowski and Laskowski (2017) analysed the impact of these external factors on the quality and reliability of the license plate recognition process. Some works have specifically focused on addressing these problems.One the approaches to make ALPR robust to these issues is to simulate all these types of problems in a synthetic dataset and then test a classification technique on this dataset to achieve a promising solution. Bjorklundet et al. (2017) follow this approach for creating synthetic license plate images of varying illuminations and inclinations. A CNN trained model on these images gave 93% accuracy when tested on real license plate images. Problem of different illuminating conditions have been addressed by Panchal et. al. using feed forward segmentation step. Segmentation has been performed by connected component analysis using pixel frequency and image features of aspect ratio and height to correctly detect the plate (Panchal, 2016). This paper addressed the problem considering ideal case scenarios and didn't take into account different constraints in different surrounding conditions like shadow, sun shine, low brightness, low quality images, angle of the name plate and relative distance from the car. License plate localization in hazardous conditions was also addressed by Azam and Islam (2016). Fog, rain, blur and night time have been considered as hazardous conditions. Several technologies like frequency domain mask to filter rain streaks, statistical binarization for handling low contrast indoor, night, blur and foggy images, Radon transform for tilt correction and image entropy for identifying non-plate areas have been applied. The method has been shown to detect correct numbers within lesser time than two other state of art works.

License plate localization within an image showing vehicle is a natural first step. Therefore, many papers have worked towards increasing the accuracy of locating plate in a normal or noisy image. License plate detection from complex scenes in real time is a challenging problem. Yuan et al. (2017) first down-scaled the image and then applied a novel line density filter to extract candidate regions of plate robustly. A cascaded license plate classifier based on linear support vector machines has been used as classifier. Proposed method was evaluated on Caltech license plate dataset and authors' own recorded dataset of about 4000 images. They could achieve an accuracy of upto 96% with detection being typically done in 42 ms. Yingyong et. al. (2015) focused on the problem of license plate localization within an image. They utilized knowledge acquisition and reduction capabilities of rough sets along with neural networks to construct a plate positioning system. The system has been experimentally shown to improve accuracy of character positioning. Davis et. al. utilized the fact that license plates have more vertical edges than horizontal and proposed a fast plate localization mechanism (Davis, Arunvinodh, & Menon, 2015). Images were pre-processed to grey scale and binarized using adaptive thresholding followed by noise removal. They also proposed an algorithm to detect correct vertical edges of the plate. In all these works, license plate detection and segmentation was followed by detection of actual license number. Optical character recognition is a quite mature field of research and thus this step can be mostly done with near perfect accuracy.

With advent of deep learning technologies and their sophistication in image recognition, few holistic approaches to license number recognition based on this technique have also been proposed in literature (Nishani & Çiço, 2017). Xie et al. (2018) used a 37-class convolutional neural network (CNN) to detect all characters in an image. This resulted in a high recall performance, as compared with conventional approaches based on image processing and training a binary text/non-text classifier. In order to eliminate false positives, they used a second plate/non-plate CNN classifier. Bounding box refinement has been carried out using edge information of the license plates to improve the intersection-over-union (IoU) ratio. Authors also proposed a cascaded framework which effectively extracts license plates with both high recall and precision. After the license plate is detected, they demonstrated two methods for recognition. The first was with a CNN pipeline that included image binarization, character segmentation and character recognition. As analysed in the paper this method suffered difficulty in character segmentation steps. To overcome this difficulty, a segmentation independent method using deep Recurrent Neural Network (RNN) to recognize the license characters as a sequence labelling problem was proposed. RNN was trained with long short-term memory (LSTM) to recognize the sequential features extracted from the whole license plate via CNNs. The main advantage of this approach is that it is segmentation free. They observed that RNN method performs better than a baseline method of combining segmentation and deep CNN classification. This paper demonstrated applicability of deep learning in ALPR. Though the paper covered a wide range of methods from license plate recognition literature with rich comparisons, a performance on videos has also not been studied. Wang et al. (2015) used Scale-Invariant Features Transform (SIFT) method to solve ALPR problem in Chinese license plates without requiring any pre-processing. Proposed method was shown to be robust to complex background, scaling variation, tilts, contamination, illumination variation, partial occlusion and defective characters. Chinese character recognition could be achieved with 96% accuracy whereas the segmentation accuracy was near perfect. Apart from that, the average execution time was lower than 268 ms which is sufficient to claim real-time processing. However, it is not sure whether same performance can be achieved if the approach is adapted on other different number plates. Nquwi and Lim (2015) examined various methods for recognizing number plates. It was found that recognition rates are high in good quality images. They have proposed filters and morphological transformation based segmentation technique to detect number plates and back-propagation neural networks for recognition. Problem of recognition in presence of different design and font styles of car number plates has been discussed in Khan et al. (2017). Using multiple templates matching for recognition of characters and removal of noise by dynamically adjusting the pixel values was followed by testing on a synthetic challenging dataset comprising of different font style and design of number plates. Low rates of false positive and false negative values have been shown to occur. AmrBadr et al. (2011) in used image processing-based pipeline for automatic Number Plate Recognition System. They used morphological operations with histogram manipulation and Edge detection for plate localization and characters segmentation. For character classification and recognition, they used an artificial neural network. This work gave detailed method of number plate recognition section but character recognition and detection steps are naive and don't take into consideration constraints like angle, distance of the vehicle and bad quality of images. Apart from this, the approach is for offline detection and method of extending to real-time is not mentioned.

Masood et al. (2017) in detailed fully automated license plate detection and recognition system. It was built using a sequence of deep CNNs combined with accurate and efficient algorithms. The CNNs were trained and fine-tuned to make them robust under different conditions such as variations in pose, lighting, occlusion, etc. These CNNs can work across a variety of license plate templates with varying sizes, backgrounds, and fonts. This paper covered many aspects of number plate recognition and gave the solution of all the problems. However, the study is subjected to only images and not videos. Ian J. Goodfellow et al. (2014) demonstrated a unified approach to integrate localization, segmentation and recognition steps via the use of a deep convolutional neural network that operates directly on the image pixels for recognizing arbitrary multi-digit numbers from Street View imagery. They employed the DistBelief implementation of deep neural networks in order to train large and distributed neural networks on high quality images. They reached the best performance with architecture of deep convolutional network with eleven hidden layers. They evaluated their approach on a publicly available dataset and achieved over 96% accuracy in recognizing complete street numbers. They improved upon the state-of-the art on a per-digit recognition task, achieving 97.84% accuracy. This paper was not specifically for number plate recognition. It mostly outlined the approach for characters segmentation and recognition from street sign images. This paper, however, acted as motivator for the application of deep neural networks for the task after the identification of license plates in cars.

3. PROPOSED AUTOMATIC LICENSE PLATE RECOGNITION MODEL

In this section, different phases to automatically detect license plates from images and videos have been discussed. The whole process has been divided into several steps which is outlined in Algorithm 1.

The steps include loading the character recognising pre-trained character classifier in Line 1. The vehicle image on which ALPR is to be applied is then sent to the detectPlatesInScene() function to get a list of all the possible license plates that are present in the image. Then the images of possible license plates are passed to the detectCharsInPlate() function which gives the predicted number for each possible plate. The longest character sequence is then returned and accepted as final answer.

3.1. Split Input Video Into Frames

The first step of the algorithm is to load the input video as a list of images. If the input is an image, skip this step. The frame rate of splitting can be controlled. In the video, if the speed of vehicle is high, split the video into 50-80 frames to achieve good results. If the speed of the vehicle is quite slow (for example, on a speed breaker) then even 10 frames will be sufficient. The steps described now onwards are applied on each frame.

Algorithm 1. ALPRProcess()

```
Input: A image file path(Let's say img)
Output: A predicted License plate string and the cropped image of
the    plate.
// Algorithm applied on each image.
1: Model ←   Load the model saved in the char_reg.h5.
2:imageOriginalScene ← Resize 1.4 times(image)
3: listOfPossiblePlates←
DetectPlates.detectPlatesInScene(imgOriginalScene)
4: // Send the list of list of plates for character detection and
character segmentation and OCR.
5:listOfPossiblePlates ←
DetectChars.detectCharsInPlates(listOfPossiblePlates)
6: //Choose plate with maximum length
7:listOfPossiblePlates ← sort(possiblePlate:
len(possiblePlate.strChars))
8:licPlate←  listOfPossiblePlates[0]
```

3.2. License Plate Detection

The image is prepared for robust plate localization. First of all, noise is removed from the vehicle's image to get more accurate results. The image is converted to greyscale and different morphological operations were applied to get an image with enhanced edges and contrast. Second step is to blur the image to bring uniformity to it and finally, adaptive thresholding is applied to eliminate less important information (objects other than number plates). Techniques like converting to greyscale, histogram equalization, sharpening, OTSU masking (Du et al., 2013, Top Hat and Black Hat morphological methods, adaptive thresholding and blurring were applied (Björklund et al., 2017; Azam & Islam, 2016; Khan et al., 2017; Saleem, Tahir, & Farooq, 2016). All this is done to remove objects other than license plate from the image as much as possible. Figure 1 and Figure 2, shows these steps on a test case.

After removing noise from image, the next step is to detect the x and y coordinates of the license plate in the image. The steps involved to accomplish this have been described in Algorithm 2. Contours are detected from the pre-processed image. OpenCV documentation defines a contour as a curve joining all the continuous points having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. The coordinates of these contours were used to extract parts of image, store it as python class object of each individual contour (character in the plate) that store the boundaries, area, aspect ratio, width, height and angle. The list of these possible plates is then used as input in next phase. This procedure of plate detection from image has been summarized as Algorithm 2.

In Algorithm 2, various operations like detecting contours, putting geometric constraints, grouping etc are applied on the result of pre-processing. These operations extract all contours, eliminate the contours that do not appear to be characters by their geometry, group together the contours that belong to a single plate and extract the number plate from the original image. The algorithm was executed for all the possible plates and returned them in the form of a list. This list contains the object images which have the highest probability of being the real license plate. Figure 3 shows the license plate cut outs on some examples.

Algorithm 2. detectplatesInScene()

Input: image file name as imageOriginal
Output: list of lists where each list contains contours those together make a complete license plate.
// Algorithm to find license plate in the given images.

```
1: imageGrey,imgThresh ← Preprocess(imageOriginal)
2: contours ←  opencv.findContours(imageOriginal)
3: listofpossibleChars = [ ]
4: for c in contours:
5:    x,y,w,h ← opencv.boundingRect(c)
6:    if (h*w > 80 and w > 2 and h > 8 and 0.25 < (w/h)< 1.0):
7:          listofpossibleChars.append(c)
8:  listOfListsOfMatchingCharsInScene ←
GroupMatchingPlates(listofpossibleChars)
9:  listOfPossiblePlate = [ ]
10: for plate in listOfListsOfMatchingCharsInScene:
11:         possibleplate←  extractplate(plate)
12:         if possibleplate.imgplate not None:
13:               listOfPossiblePlate.append(possibleplate)
14:  return listOfPossiblePlate
```

Figure 1 shows the noise reduction of input image. Figure 2 shows thresholding steps to which try to remove objects other the license plate. Some test cases of license plate recognition have been shown in Figure 3. From examples in Figure 3, it can be seen that the program works quite well on images having license plate at some angle, images where car is quite far, vehicles other than car and some shadow on license plate.

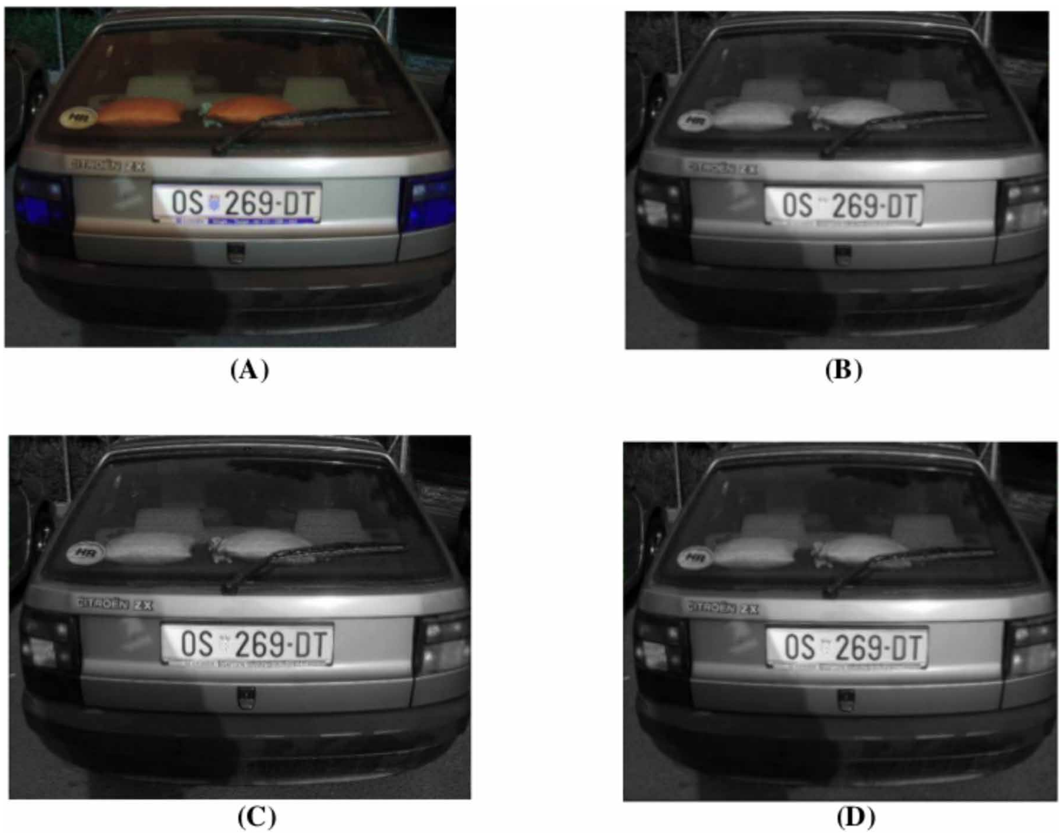
3.3. Character Segmentation

Cropped license plates from section 3.2 are now put for optical character recognition in this section. First requirement to apply OCR is to crop individual character images from license plates. Most of the time the license plate cut out is not suitable for direct character segmentation. Thus, unwanted symbols were to be removed from the license plate. Image processing steps described in step 2 were repeated to remove noise in the plate image. Authors found contours from the plate's image and kept only symbols which are possibly characters (A-Z, 0-9) (see Figure 4).

Next step was to enlarge the images and convert to greyscale as well increased the contrast. Subsequently, adaptive thresholding is applied to make the whole image black and white. Binary inverse (Björklund et al., 2017) can be done to get black characters on white background. The contrast helped in recognizing character boundaries. Characters other than the 36 characters (0-9 and A-Z) were removed from the plate and individual characters were cut out from the license plate. Height and width of each cropped character is increased by factor of 1.3 and 1.6 respectively and a white border is added (Chang et al., 2004; Łubkowski & Laskowski, 2017; Björklund et al., 2017; Azam & Islam, 2016; Khan et al., 2017; Saleem et al., 2016).

These steps were done to ensure that the input to CNN model for OCR is consistent with its training set. Algorithm 3 describes the process applied to detect individual character boundaries from

Figure 1. Image Pre-processing Steps: (A) Input Image (B) Grayscale Image (C) Image Contrast Enhanced (D) Image Blurred to Remove Noise



the detected plate. Here input is a list of all the plates in the image with their different geometric characteristics. Each plate image is then pre-processed to first find contours and then reject the contours that may not be characters followed by grouping the contours together based on their geometric parameters and then sending them to next module to recognize the number written in them. The list of all the plates with the corresponding number in them is returned as final output of this step.

Each character image is cropped and pre-processed before presenting to character recognition module. Input to this step is a list of contours from particular possible number plate. Each contour in the list is the image of a particular character present in that plate. Thresholding has been applied to plate image to crop particular character contour from the image. Bordering and resizing of the cropped image is done to aid better OCR. Processed and cropped character image is input to the character model to predict which character this contour is and append it to the license plate string. Details of character recognition module are given in next section.

3.4. Optical Character Recognition

After getting the individual character images from a license plate image, the next step is to apply OCR (Optical Character Recognition) (Du et al., 2013; Łubkowski & Laskowski, 2017; Björklund et al., 2017). To do this task two approaches have been tried namely, K nearest neighbour and convolutional neural networks.

Figure 2. Pre-processing steps for license plate extraction: (A)image after thresholding(B) Possible character Contours (C) Filtering Non-Character Contours (D) Eliminate Overlapping Contours

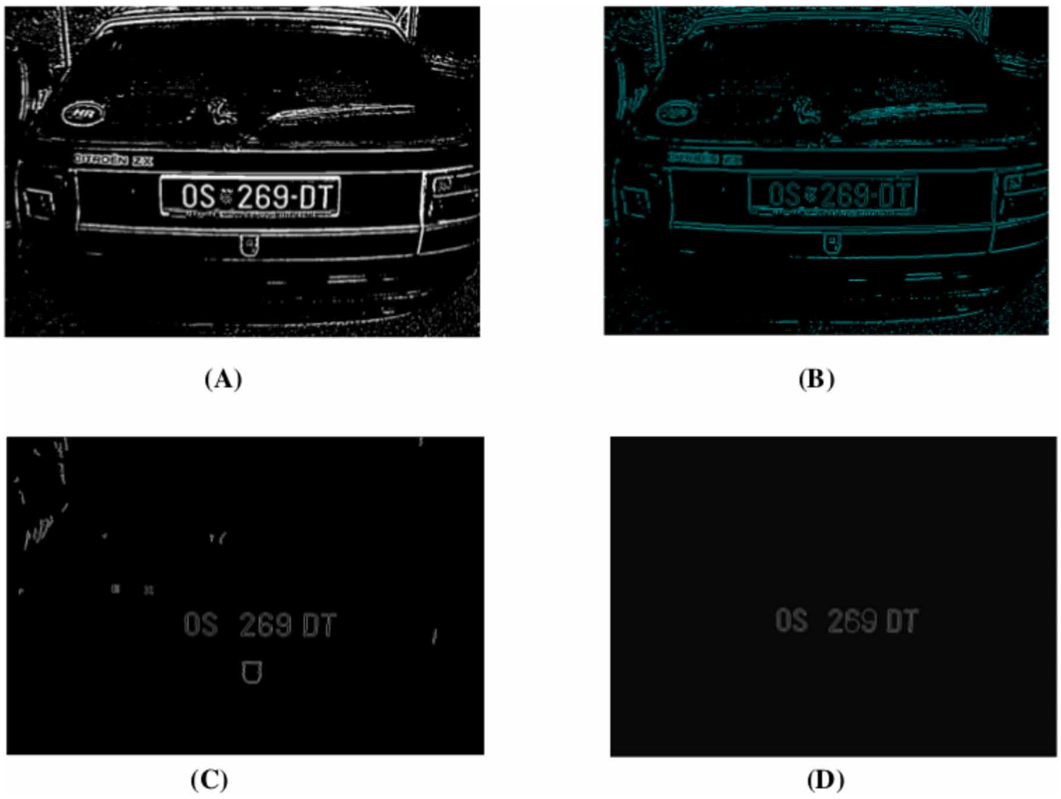


Figure 3. Some examples of license plate cut outs on test set

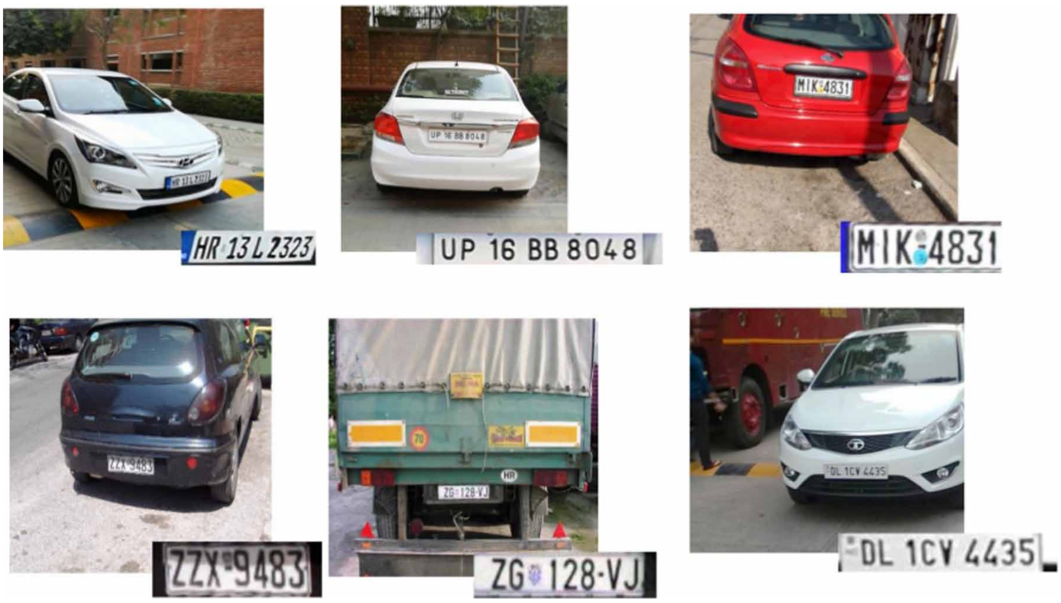
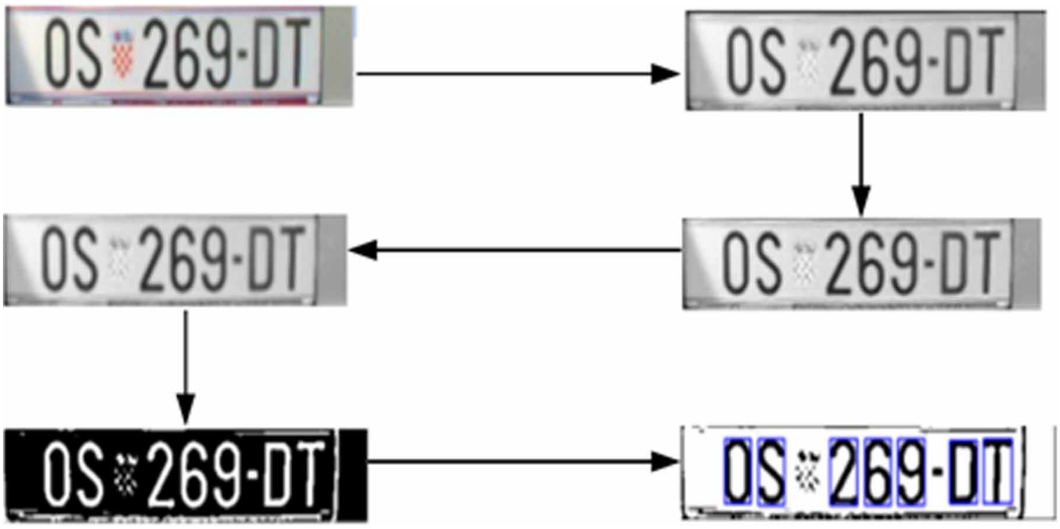


Figure 4. License Plate Noise Removal and OCR



a. **K Nearest Neighbours (KNN)** (Cao et al., 2018) algorithm is a well-known algorithm for regression and classification problems which is a simple algorithm that stores the spatial information of all available cases and classifies new cases based on its similarity with available examples. The similarity matrix for the algorithm depends on the problem. KNN has been implemented in Scikit learn library K Nearest Neighbor class with K as 3. As KNN implementation takes a single vector as input but input for OCR was image of a character. So, the image of shape (h,w,3) was converted to a vector of shape (h*w*3,1) and then input to the classifier for training and testing. For predicting the class for a new example, KNN finds K most similar examples to the test cases and outputs most dominant ones. Since here the cropped image is not strictly similar to the training images, KNN has difficulty in prediction.
b. **Convolutional Neural Networks (CNN)** (Khan et al., 2017; AmrBadr et al., 2011; Wang et al., 2015; Nishani & Çiço, 2017; Xie et al., 2018; Björklund et al., 2017) comprise of neurons that have learnable weights and biases Björklund et al., 2017; Panchal, Hetal, & Panchal, 2016; Azam & Islam, 2016; Yuan et al., 2017). Each neuron receives some inputs and performs a dot product

Algorithm 3. detectCharsInPlates()

```
Input: list of list of plates
Output: list of list of plates with predicted number on each plate.
// Algorithm to find the number written in the license plate.
1: for plates in listOfPossiblePlates:
2:          plates.grayscale,plates.thresh ←
preprocess(plates.img)
3:          plates.Thresh ← resize(plates.Thresh,1.6,1.6)
4:          Apply Binary plus otsuthresholding to Thresh
5:          listOfpossiblechars ← findpossiblechars(plates.Thresh)
6:          plates.strChar ← recognizeCharacterInPlate()
7:  return listOfPossiblePlates
```

Table 1. CNN Layers used in Character Model

Layer(type)	Output shape	Parameters
First Convolution layer	(None, 62, 62, 32)*	320
Max Pooling layer	(None, 31, 31, 32)	0
Second Convolution layer	(None, 29, 29, 32)	9248
Max Pooling layer	(None, 14, 14, 32)	0
Flatten Layer	(None, 6272)**	0
Dense layer	(None, 128)	802944
Dropout layer	(None, 128)	0
Dense Layer	(None, 36)***	4644
Total Parameters: 817, 156 Trainable Parameter: 817, 156 Non-Parameters: 0		

* The first dimension is the number of images intended to be sent, second and third dimension tell the height and width of the output matrix after that layer. The last number tells the number of channels.

** The number signifies the number of rows created after unwinding the previous layers output matrix.

*** 36 here signifies the number of classes.

with certain filters. Filters are 2D or 3D arrays which are used to modify an image in a desired way. Linear filtering, accomplished through an operation called convolution, is a technique in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighbourhood. Convolution is a neighbourhood operation in which each output pixel is the weighted sum of neighbouring input pixels. The matrix of weights is called the convolution kernel and is the filter actually. The convolutional neural network has been implemented in Keras (Davis et al., 2015) API which is a high-level API built on top of Tensorflow (Abad et al., 2015), an open source machine learning framework. It was trained using 32480 character images (de Campos, Babu, & Varma, 2009 classifying each image into one of 36 classes ranging from 0-9 and A-Z. The classifier was tested with 8032 images. It takes 64x64 sized input image in the first convolution layer.

Different layers of CNN used in building the required character model has been listed in Table 1.

- **Dropout:** For a neural network to perform better each of its neuron should learn and have its parameters updated in each epoch. But when the number of neurons in a layer is very high then it may happen that the contribution in the result of the whole network is more by some neurons, this will make the parameters of those neurons to be more updated. To avoid this, random shutting down of some k neurons is used while training. Here, 40% of the previous layers neurons are shut down.
- **Flatten:** As the artificial neural network layers can only process vectors as input so encodings were needed to be converted to vectors. Flatten layer does the same. It unwinds the encoding matrix to a vector.
- **Maxpool:** A pooling layer reduces the dimension of the input matrix with minimal loss retaining most of the information. Here max pooling with a kernel of 2x2 is used, means it will replace a 2x2 patch of the input with the highest pixel value of the patch. Also, a stride of 2 is taken to avoid mixing of features.
- **Dense:** This is the other name of artificial neural network layer. This layer does mapping of encoding vector to the right class.

The CNN classifier has been evaluated exhaustively over benchmark and self-shot test images. Details of analysis of results have been presented in Section 4.

3.5. Video Processing for ALPR from Frames

In order to show real time processing, proposed license plate detection and optical character recognition modules were implemented on video streams too. For processing, each video was split into frames at the rate of 26 Frames/sec. Each frame is then sent to the license number recognition pipeline and the count of each unique predicted number is updated. The license number with the highest frequency for current video is chosen as predicted number and is checked against actual license number for accuracy. The process is detailed step wise in Algorithm 4.

4. RESULTS AND ANALYSIS

This section gives information about the dataset used for training and testing the OCR model and license plate detector. All image pre-processing work has been done on Intel(R) Core(TM) i5 -6200 CPU @ 2.30 Ghz machine with 8GB DDR - 4 RAM. CNN Classifier is trained on Nvidia GeForce GTX 960 GPU with 4GB memory. In OCR, main metric has been accuracy of character recognition to its correct value.

4.1. Accuracy of Plate Detection from Image

Contours obtained from image were analysed for representing a possible license plate. Based on studied done in Khan et al. (2017) and Yuan et al. (2017), several boundary condition values for possible number plate contours are hardcoded. These are listed in Table 2.

The conditions on width and height help differentiate the regions where some characters or words are written from a license plate. It is assumed that in license plates number of characters should be at least 3. Apart from constraints listed in Table 2, a padding factor of 1.3 and 1.5 along the width and height of the plates is also considered. After these steps, a cleaner list of possible license plates in the given image is obtained.

After applying these boundary conditions an accuracy of 92% was achieved for license plate detection.

4.2. Datasets Used for OCR

Two types of dataset have been considered and used for evaluation. First dataset is of car images taken from Dlagnekov and Belongie (n.d.) and second is a character-digit dataset from de Campos et al. (2009), on which Optical character recognition model has been trained. Character and digit dataset originally had 60,000 total images. Some images of characters cropped from real number plates were added to images of all 36 classes (26 alphabets and 10 digits) making total size of dataset to 60,526 images. Few samples from each dataset have been shown in Figure 5 and Figure *6*.

From Figure 6, it can be deduced that data like fourth image of '0' and first image of 'X' don't occur frequently on number plates. Moreover, the number plate is generally written on white background in black font. Similarly, color and design like in image of 'H' and second image of 'D' are also not that frequent. So, to make our Optical character recognition problem simpler, such images were manually removed from dataset and all remaining images were used by converting to black and white. This made the problem easier for the OCR model. Some images from the new dataset are shown in Figure 7.

This reduced the dataset of to 40,512 images. Final number of images per class in the dataset is shown in Figure 8. It can be seen that the training dataset is a balanced one. This dataset was split into 75% and 25% samples for training and testing respectively. Besides the proposed algorithm is tested on the car dataset which comprised of 4800 images. The proposed algorithm is also tested on 50 short videos too. These videos were captured from exit points of a parking lot.

Algorithm 4. Process()

```
Input: A Video Path
Output :The cropped image of the license plate and storing the data
in the Mongodb database.
1: Image  ←  input('Enter the name of the Video: ')
2: data ←  videoSplit()
3: os.chdir('data')
4: For img in folder(data) :
5:    Image,pred ←  main(img)
6: If pred in result.keys() :
7:    result[pred] ←  result[pred] + 1
8: Else if (pred != ' '):
9:    result[pred] ←  1
10: Sort the dictionary by the frequency.
11:sort_by_value(result):
12:l ←  {x: y for y, x in result.items()}
13:r ←  list(sorted(l.keys()))
14:index ←  r[len(r) - 1]
15: //Store the result into a Mongodb database.
16: plate ←  l[index]
17:img←  result_imag[plate]
18:print('execution time is : ' + executionTime)
19:dict←  {}
20:dict['date and time'] ←  time.ctime()
21:dict['video'] ←  name
22:dict['video length'] ←  vdolength
23:dict['image'] ←  plate
24:dict['Total Frames in video'] ←  totalFrames
25:dict['execution_time'] ←  executionTime
26:dict['frequency ratio'] ←  "{0:.2f}".format(result[plate] / len(result))
27:col.insert(dict)
28: Print 'The name plate is :', plate, ' frequency is:'+result[plate]
29: Show → image(img)
```

4.3. Tuning CNN Classifier to Increase Accuracy OCR in Number Plate

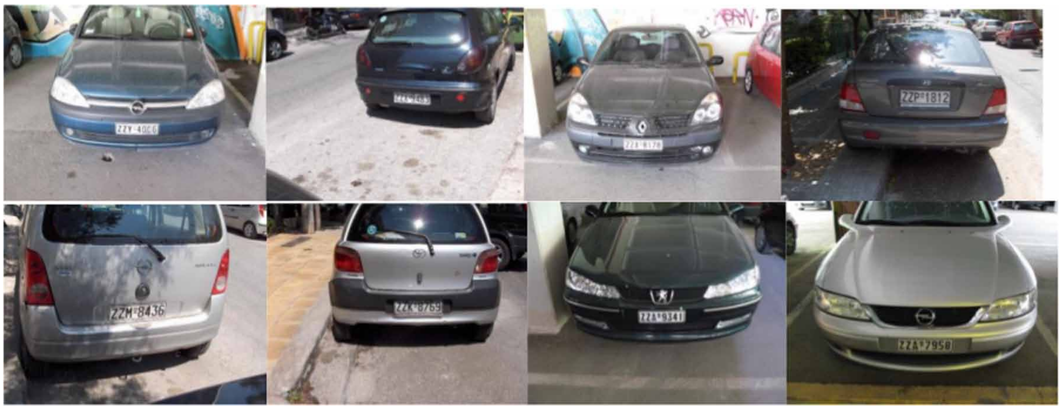
The KNN classifier gave an accuracy of 68.30% as compared to 85% by CNN classifier. The Convolutional Neural Network has been implemented with Keras (2015) API, built on top of Tensorflow (Abadi et al., 2015) deep learning framework. Accuracy was very low at 43.39% with raw data as input to Keras (2015) module. After applying steps proposed in Section 3, accuracy increased to 85%. Step by step increase has been shown in Table 3.

It can be seen from Table 3 that tuning the resizing parameters, using BINARY_INV, OTSU masking, Bicubic interpolation and Adadelta resulted in a significant jump in accuracy. Here, BINARY_INV is a method by which image are converted into a black-white image. Here it will make each pixel with value less than a threshold to be 255 and the ones with value greater than a threshold to be 0. OTSU is a method which looks for 2 peaks pixels in the image and enhances its contrast. kernel_1 is defined to be [[0,-1,0], [-1,5,-1], [0,-1,0]] and kernel_2 is [[-1,-1,-1], [-1,9,-1], [-1,-1,-1]]. Bicubic interpolation is an extension of cubic interpolation for interpolating data points on a two-dimensional regular grid. The interpolated surface is smoother than corresponding

Table 2. Boundary conditions to filter out license plate contours

Boundary Condition	Remarks
minimum width of the rectangle bounding the contour should be 2 pixels	maximum change in area should be less than ±0.5 units
minimum area of the rectangle bounding the contour should be greater than 80 units	maximum change in width should be ±0.8
minimum height of the rectangle bounding the contour should be greater than 8 pixel width	maximum change in height should be ±0.2
aspect ratio of the rectangle should be between 0.25 to 1.0	For differentiating other numbers or characters from license plate
Distance between centres of two contours should be more than 0.3 times the diagonal size of the smaller contour	For detecting overlapping contours
Distance between their centres should be less than 5 times the diagonal size of the smaller contour,	For deciding that two contours are part of a single plate
Angle of the line joining the centres of the two contours should be less than 12 radians	For deciding that two contours are part of a single plate

Figure 5. Instances from car image dataset



surfaces obtained by bilinear interpolation or nearest-neighbor interpolation. In this interpolation a 4x4 pixel neighbourhood is taken into account. Adadelta is a gradient descent optimization algorithm that ensures less aggressive reduction of the learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size.

4.4. Comparison of CNN and KNN Classifiers

Two classifiers, KNN and CNN have been used for OCR. The performances of these two classifiers on the test dataset have been compared based on various metrics. The initial results obtained were not up to the mark. So, parameter tuning was done to increase accuracy, decrease running time and improve the overall performance of the model. In classification problem, accuracy alone is not a sufficient parameter to judge the performance of the ALPR model. Sometimes accuracy can be a misleading parameter. For example, consider a skewed data set with 990 positive examples and 10 wrong examples. A model which classifies every example as positive will give an accuracy of 0.99 which is very high, but it is clearly evident that the classifier is fully biased. So, the authors considered more

Figure 6. Instances from the character-digit dataset

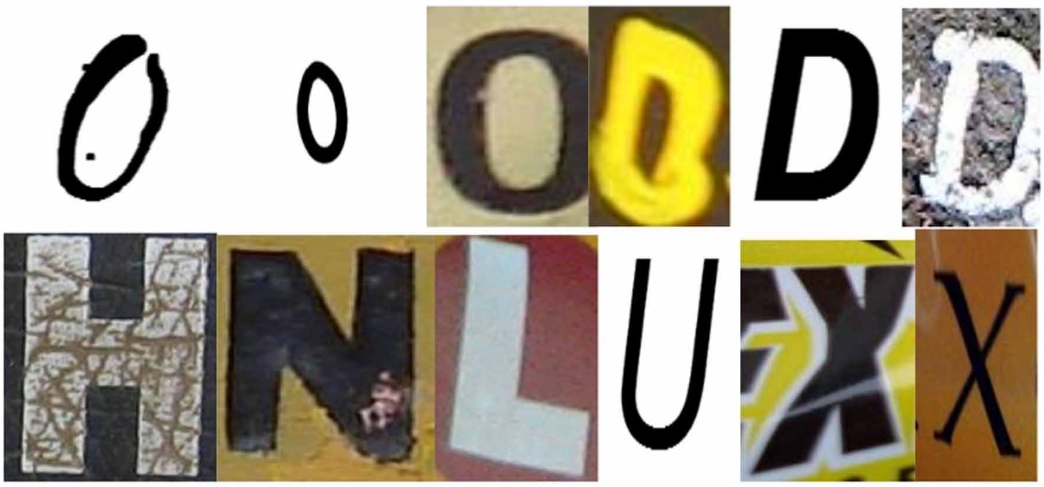
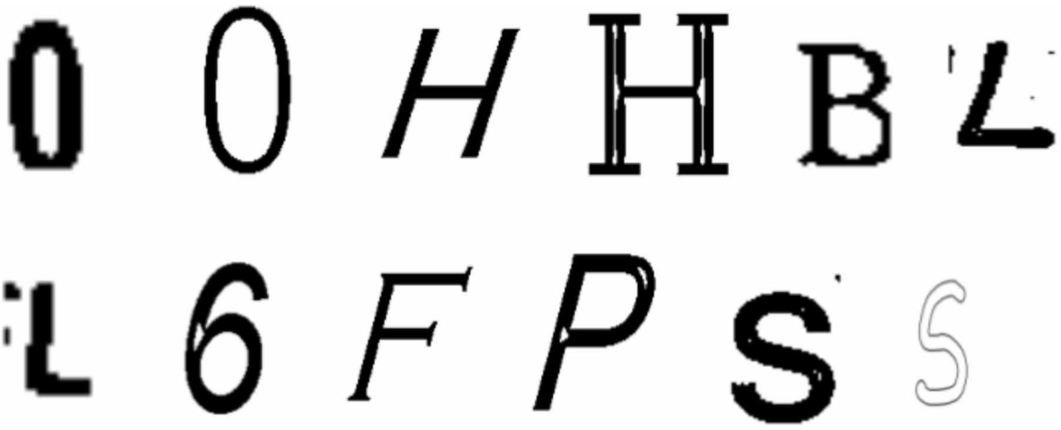


Figure 7. Some instances from the Character-digit dataset

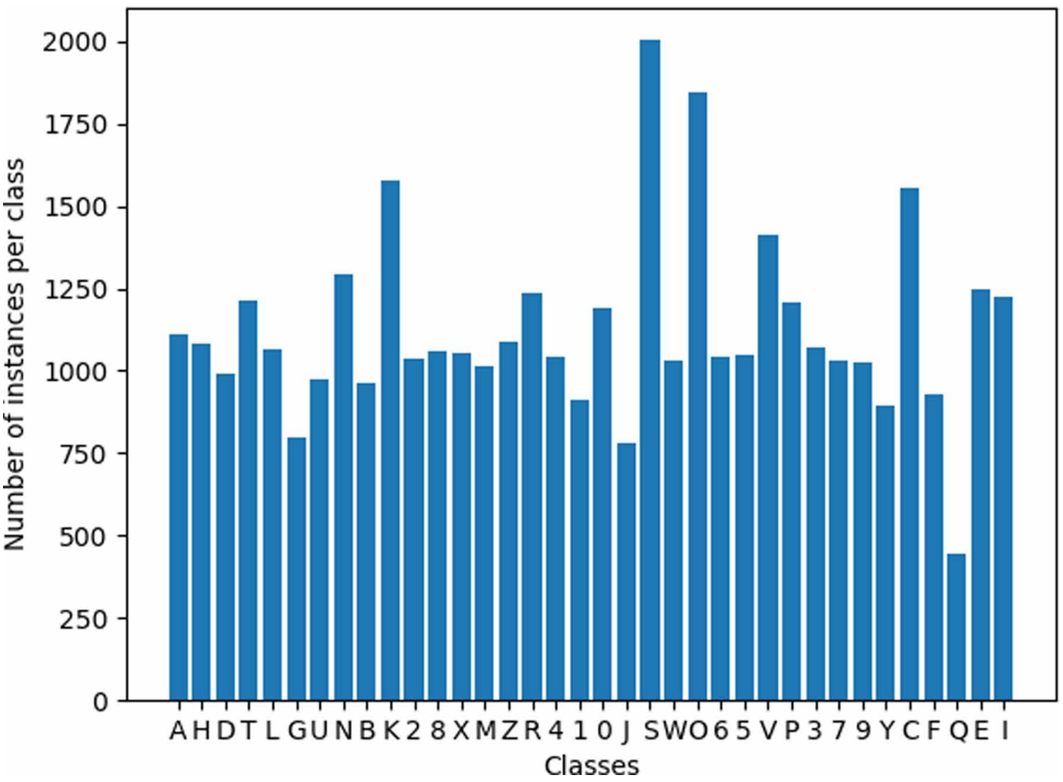


metrics like confusion matrix, precision, recall, F-1 score, Informedness and Matthews correlation coefficient to evaluate the performance.

Confusion matrix is a popular way to represent classification performance of a classifier. Confusion matrix for 36 symbols (26 characters and 10 digits) is shown in Figure 9. A scale on the right shows the colour and number relation. Dark colour means that more number of images were categorised correctly. A dark diagonal line represents that actual and predicted characters were same in most of the cases. It is noted that the confusion between '0'and 'O' and '1' and 'I' is higher than any other character pairs.

The KNN classifier showed a mixed performance with high false negative for 0 and O both. The KNN also has false positive and false negative for many other classes. Apart from confusion metrics, the outcome of classification has also been evaluated on following metrics:

Figure 8. Instances per classes



4.4.1. Precision or Positive Predictive Value (PPV)

Precision is defined as per equation (4). It is the fraction of relevant predictions among all the predictions done. The precision of different classes is shown in Figure 10. It can be seen that precision ranges from 80% for '0' to about 96% for '1,' 'A,' 'J,' etc.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

4.4.2. Recall or True Positive Rat e(TPR)

It is also known as sensitivity of classification. It is the fraction of relevant predictions that have been predicted over the total amount of relevant predictions. The recall of different classes is shown in Figure 11. It is important to have good recall results to estimate the actual overall performance of classification. Recall has reached near perfect in labelling '2', 'N' and '8' symbols. '0' has a low recall due to confusion with symbol 'O'.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

Table 3. Steps to increase accuracy of CNN classifier

S.NO	CHANGE	ACCURACY	RUNNING TIME (sec)
1)	CNN classifier only	43.39%	494.60
2)	Hist. equalization and de-noising added.	**40%**	840.0
3)	Hist. equalization and BINARY_INV masking on plate	55%	1243.59
4)	Only BINARY_INV masking of plate	54.89%	970.70
5)	Hist. equalization, resizing (1000 x 600) original image and BINARY_ INV masking of plate.	53.61%	698.0
6)	Hist. equalization and (BINARY_INV + OTSU) masking	56.14%	859.59
7)	Hist. equalization, (BINARY_INV+ OTSU) masking and sharpening with kernel_1	46.13%	956.67
8)	Hist. equalization, (BINARY_INV+ OTSU) masking and sharpening with kernel_2	49.14%	1023.67
9)	(BINARY_INV + OTSU) masking	60.00%	494.67
10)	(BINARY_INV + OTSU) masking and resize (1000,600) the original image	60.13%	441.67
11)	(BINARY_INV + OTSU) masking, resize (1000,600) the original image and de-noising.	58.13%	1641.67
12)	(BINARY_INV + OTSU) masking and add border(8px) to the individual character's image.	71.90%	241.80
13)	(BINARY_INV + OTSU) masking, add border(6px) to the individual character's image and resizing (width 1.6 times, height 1.3 times)	74.90%	285.50
14)	(BINARY_INV + OTSU) masking, add border(8px) to the individual character's image and resizing (width 1.4 times, height 1.4 times)	78.70%	368.60
15)	Cleaned the dataset, Bicubic interpolation, used Adadelta	**85%**	379.45

Figure 9. Confusion matrix of OCR for (a) CNN (b) KNN based classification

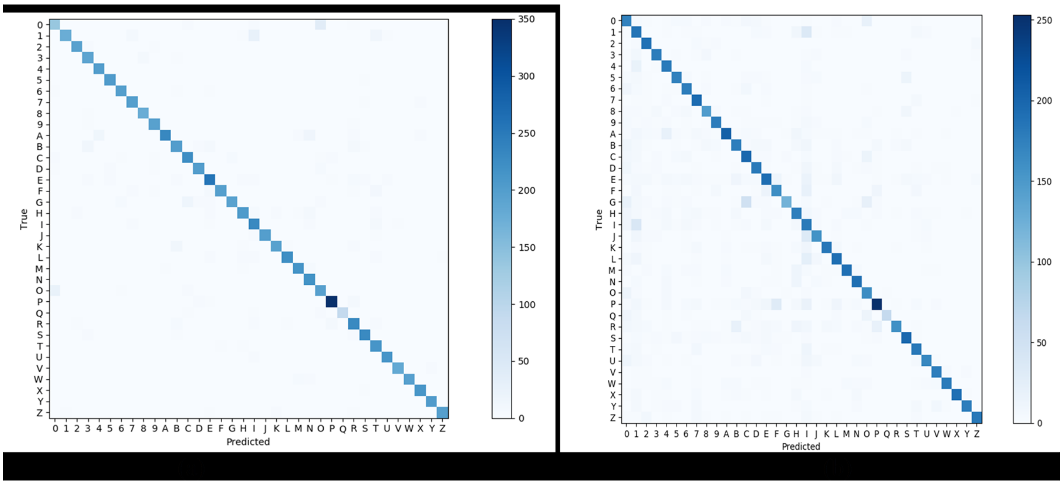
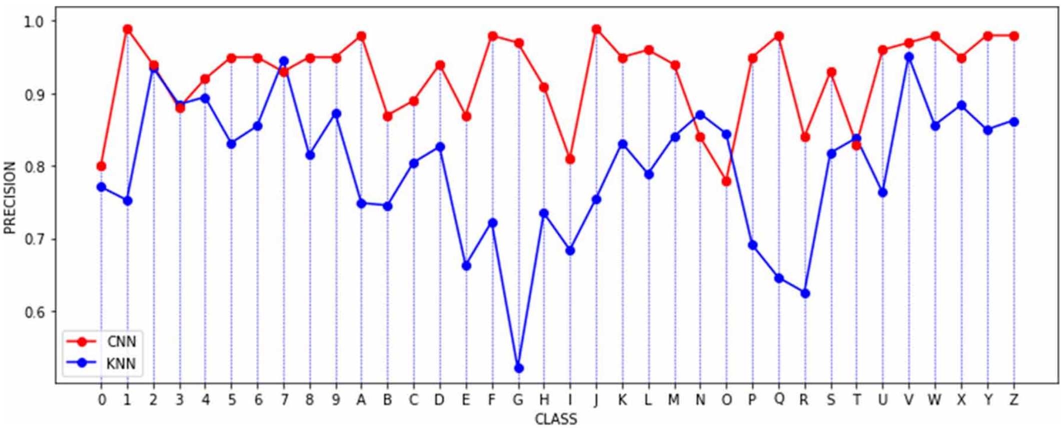


Figure 10. Precision per class



4.4.3. F1-Score

As only high precision and high recall is also not a perfect indicator for the quality for our model. In real world problems the classifier should perform well on both the metrics. The combined effect of Precision (P) and Recall(R) can be analysed by F1- score measure.The F1score can be calculated as per equation (3). Results of F1 score are shown in Figure 12.

$$\text{F1-Score} = \frac{2 \times P \times R}{P + R} \tag{3}$$

Value of F1 score is highest at 0.98 and lowest at 0.75. Thus, it can be seen that ‘0’ had high precision of recognition, but low F1 score.

4.4.4. Informedness

It is a generalization applied to evaluate multiclass classifications and estimates the probability of an informed decision. Calculation is done as per equation (4).

Figure 11. Recall per class

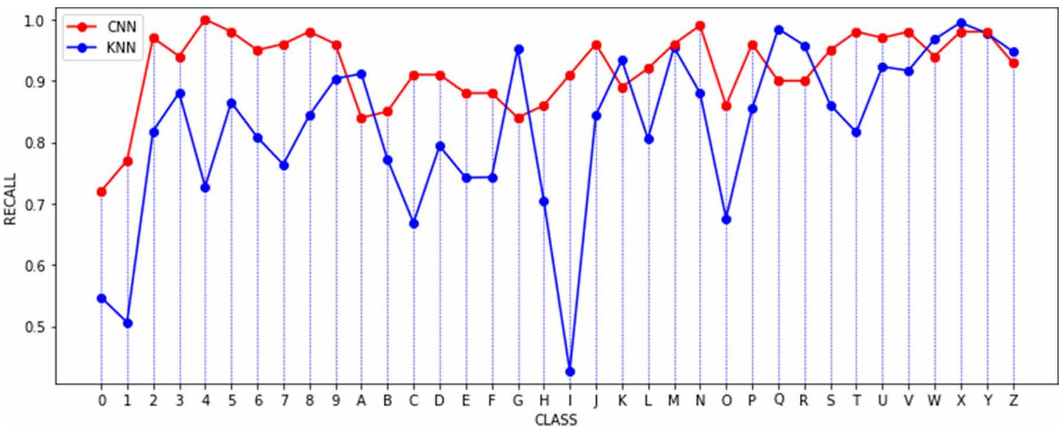


Figure 12. F1 score of all classes

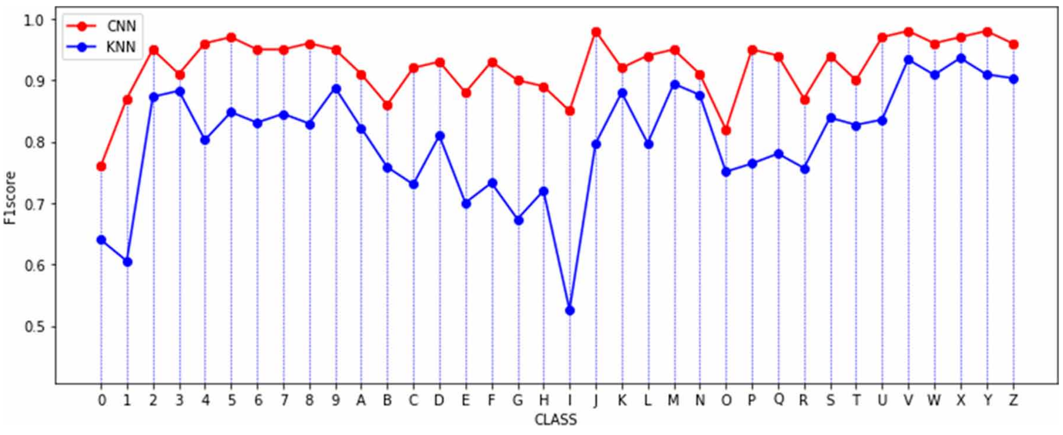
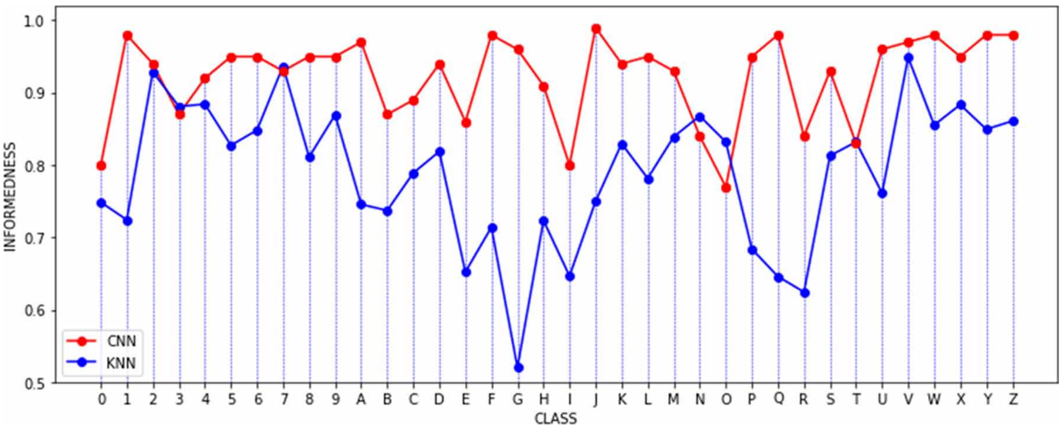


Figure 13. Informedness per class



$$\text{Informedness} = \frac{TP}{TP+FN} + \frac{TN}{TN+FP} - 1 \tag{4}$$

The Informedness of different classes is shown in Figure 13. Informedness clearly represents the confusion in classification of *‘0’* and *‘O’*. Other class with low value of Informedness is *‘I.’*

4.4.5. Matthews Correlation Coefficient

Matthews correlation coefficient is generally regarded as being one of the best metric to summarize the confusion matrix. It is calculated as per equation (5), defined in terms of a KxK confusion matrix C.

$$\text{Matthews Correlation Coefficient} = \frac{\sum k \sum l \sum m C\left[k,k\right] C\left[l,m\right] - C\left[k,l\right] C\left[m,k\right]}{\sqrt{\sum k \left(\sum l C\left[k,l\right]\right)\left(\sum k' \mid k'! = k \sum l' C\left[k'\right]\left[l'\right]\right)}\sqrt{\sum k \left(\sum l C\left[l,k\right]\right)\left(\sum k' \mid k'! = k \sum l' C\left[l'\right]\left[k'\right]\right)}} \tag{5}$$

Figure 14. Matthews correlation coefficient per class

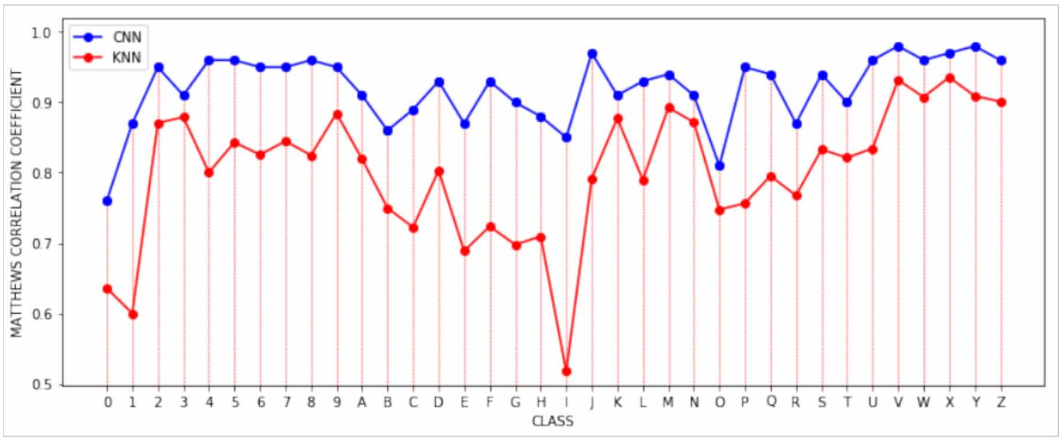
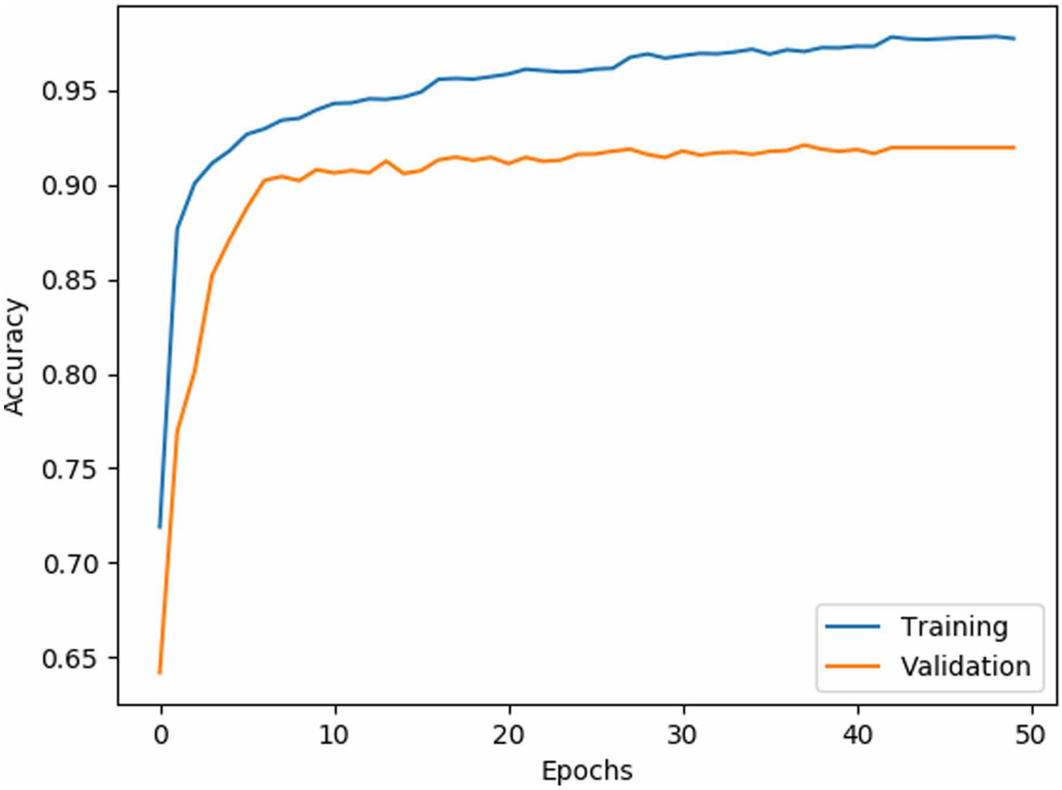


Figure 15. Accuracy vs number of epochs



The Matthews correlation coefficient of different classes is shown in Figure 14.

The correlation coefficient is discretized version of Pearson correlation coefficient. Results of correlation coefficient also show that best performance is obtained for *'J'*. To check against over-fitting, validation of performance of the classifier on a separate dataset which is not included in the training dataset was done for 50 epochs. This set has been called the validation dataset. Accuracy on the validation set versus training set after each epoch has been shown in Figure 15.

An epoch is full training cycle of one forward pass and one backward pass on the training dataset. It can be seenfrom the figure that the model converges after 20 epochs(approx.). At 20 epochs the training accuracy is about 95% and validation accuracy is about 93%. It can be seen that for training set, maximum accuracy is 0.98 against 0.92 obtained for validation set.

4.5. Performance Evaluation on Videos

Proposed ALPR was tested on some videos of vehicles shot at parking exits of our home institution. Table 4 shows analysis of different test car videos in following fields:

- **Image** - License number detected by ALPR
- **Video** – Name of video
- **Video length** – Length of video (in seconds) considered for ALPR
- **Execution time** – Time (in seconds) to run the program on a video
- **Frequency ratio** – No. of frames which contains correct final output divided by total number of frames in the video. It is the division of count of correct plate number divided by the total count of all the different plate numbers.

From the execution time of processing of each video for detection of license plate, it can be stated that the proposed algorithms can detect license plates in real time. This would be very useful in automated car entry/exit logging in parking lots and owner detection in case of traffic violations.

4.6. Discussion

Multiple metrics were used to establish the accuracy and robustness of the OCR methods. Precision measures the ability of grouping samples and consistency. High precision and low accuracy, as obtained with recognition of symbol ‘0’may have occurred due to systematic bias. Getting all the positives right is of equal value to getting all the negatives right, so you need to work equally hard on each class. Informedness captured normalization for Recall and Precision, giving equal weightage to both. Matthews correlation coefficient also depicts cohesiveness of the predictions. Results of these metrics on OCR in input image dataset of normal and noisy images were consistent to each other. Custom parameterized pre-processing techniques have ensured high accuracy of detection of correct numbers in even bad quality images. In all metrics, there is a big dip in characters ‘0’ and ‘I’ due to their low recognition rate. The classifier is wrongly predicting 0’s as O’s, ‘I’ as ‘1’ and vice versa. For all the other classes the classifier is performing nearly perfect. Minimum value for Recall, F1-Score and Matthew’s coefficient are at ‘0’ and for Precision and Informedness it is at ‘O’. A low value of precision for ‘O’ and recall for ‘0’ is due to high false positives in ‘O’ and high false negatives in recognizing ‘0’. Similarly, ‘1’ had high precision high but low recall. Also, ‘I’ has its precision very near to the minima of the precision. This is unavoidable due to geometric similarity of these two pairs to one another in English language.

Apart from archival dataset, proposed ALPR method was also tested on live images that were not part of training. The classifier achieved more than 90% accuracy in detection of license plate on these unknown images as well. Apart from that, ALPR was applied on real videos and due to redundancy of information; accuracy of plate detection in videos went up to 100% in the tested cases. Maximum execution time to capture number plates from outgoing cars videos was found to be 30 seconds which can be considered near real time performance.

Thus, it can be concluded that the proposed method will robustly work for ALPR on still snapshots or videos of vehicles taken at different capturing points, giving correct license plate numbers in real time. The obtained number can be logged into databases to learn long term driving patterns of particular cars with specific numbers.

Table 4. Performance evaluation of ALPR on videos of cars

S.NO	Video Name	Video Length(sec)	Execution Time(sec)	Frequency Ratio	License Plate Number
1	car1.mp4	4.48	22.30	0.41	LMN703
2	car6.mp4	4.2	20.56	0.63	DL10CT5806
3	car3.mp4	4.0	33.93	0.56	DL3CBZ438
4	car7.mp4	2.19	15.77	0.50	6BX0270
5	car9.mp4	3.82	23.65	0.53	DL4CS1797
6	car11.mp4	2.6	26.25	0.40	16XV2435
7	car12.mp4	3.6	21.28	0.57	HT13L2323
8	car13.mp4	2.11	19.81	0.61	R51AU293
9	car15.mp4	8.08	31.08	0.96	HR26CB2604
10	car16.mp4	9.62	30.32	0.67	DL1CP1256
11	car17.mp4	4.00	14.00	0.6	LBCS55B7
12	car19.mp4	2.68	15.52	0.43	PX6AX7Z97
13	car20.mp4	4.72	15.01	0.50	X4Z4S36
14	car21.mp4	4.53	14.13	0.41	DL1CY4435

5. CONCLUSION AND FUTURE WORK

In this paper, steps for Automatic License Plate Recognition for vehicle management have been proposed. It takes a vehicle video as input and gives the license plate number of the vehicle in text form. The license plate has been recognised using template matching technique which generated a list of possible plates in an image by matching various features like aspect ratio, area, height, width of license plate and also distance between characters, angle between them and change in area. The comparison between K Nearest Neighbours and Convolutional Neural Network showed that Convolutional Neural Network is a better alternative. Finally, 8 layered Convolutional Neural Network was used for OCR. An overall accuracy of 85% on the test set is achieved. The algorithm used in this paper not only accelerates the process but also increases the probability of detecting the license plate and extraction of characters, under certain set of constraints. Low Accuracy of classifier is mainly due to confusion between '0', 'O' and '1', 'I' in case of blurred or low-quality images. This can be improved by training the CNN classifier on larger set of various types of character images. The template matching technique to detect license can be surely improved by changing the parameters according to the license plate structure of that country/region or using more robust methods those generalize well on new problems. This will decrease the list of possible plates and thus CNN classifier will have to run on lesser number of characters. The processing speeds of given ALPR system can be further improved by using the parallelization approach for various stages of development.

REFERENCES

Chang, S.-L., Chen, L.-S., Chung, Y.-C., & Chen, S.-W. (2004, March). Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems*, *5*(1), 42–53. doi:10.1109/TITS.2004.825086

Du, S., Ibrahim, M., Shehata, M., & Badawy, W. (2013, February). Automatic License Plate Recognition (ALPR): A State-of-the-Art Review. *IEEE Transactions on Circuits and Systems for Video Technology*, *23*(2), 311–325. doi:10.1109/TCSVT.2012.2203741

Łubkowski, P., & Laskowski, D. (2017). Assessment of Quality of Identification of Data in Systems of Automatic Licence Plate Recognition. In J. Mikulski (Ed.), *Smart Solutions in Today's Transport. TST 2017. Communications in Computer and Information Science* (Vol. 715). Cham: Springer. doi:10.1007/978-3-319-66251-0_39

Björklund, T., Fiandrotti, A., Annarumma, M., Francini, G., & Magli, E. (2017). Automatic license plate recognition with convolutional neural networks trained on synthetic data. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Luton (pp. 1-6). doi:10.1109/MMSP.2017.8122260

Panchal, T., Patel, H., & Panchal, A. (2016). License plate detection using Harris corner and character segmentation by integrated approach from an image. *Procedia Computer Science*, *79*, 419–425. doi:10.1016/j.procs.2016.03.054

Azam, S., & Islam, M. M. (2016). Automatic license plate detection in hazardous condition. *Journal of Visual Communication and Image Representation*, *36*, 172–186. doi:10.1016/j.jvcir.2016.01.015

Yuan, Y., Zou, W., Zhao, Y., Wang, X., Hu, X., & Komodakis, N. (2017, March). A robust and efficient approach to license plate detection. *IEEE Transactions on Image Processing*, *26*(3), 1102–1114. doi:10.1109/TIP.2016.2631901 PMID:27893394

Yingyong, Z., Jian, Z., Yongde, Z., Xinyan, C., Guangbin, Y., & Juhui, C. (2015). Research on algorithm for automatic license plate recognition system. *International Journal of Multimedia Ubiquitous Engineering*, *10*(1), 101–108. doi:10.14257/ijmue.2015.10.1.9

Davis, A. M., & Arunvinodh, C. (2015, March). Automatic license plate detection using vertical edge detection method. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (pp. 1-6). IEEE. doi:10.1109/ICIIECS.2015.7193073

Nishani, E., & Çiço, B. (2017). Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation. In *2017 6th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1-4). doi:10.1109/MECO.2017.7977207

Xie, L., Ahmad, T., Jin, L., Liu, Y., & Zhang, S. (2018, February). A New CNN-Based Method for Multi-Directional Car License Plate Detection. *IEEE Transactions on Intelligent Transportation Systems*, *19*(2), 507–517. doi:10.1109/TITS.2017.2784093

Wang, Yu., Ban, X., Chen, J., Hu, B., & Yang, X. (2015). License plate recognition based on SIFT feature. *Optik-International Journal for Light and Electron Optics*, *126*(21), 2895–2901. doi:10.1016/j.ijleo.2015.07.040

Nguwi, Y. Y., & Lim, W. J. (2015). Number plate recognition in noisy image. In *2015 8th International Congress on Image and Signal Processing (CISP)*, Shenyang (pp. 476-480). doi:10.1109/CISP.2015.7407927

Khan, J. A., Shah, M. A., Wahid, A., Khan, M. H., & Shahid, M. B. (2017). Enhanced car number plate recognition (ECNPR) system by improving efficiency in preprocessing steps. In *2017 International Conference on Communication Technologies (ComTech)*, Rawalpindi (pp. 156-161). doi:10.1109/COMTECH.2017.8065766

Badr, A., Abdelwahab, M. M., Thabet, A. M., & Abdelsadek, A. M. (2011). Automatic number plate recognition system. *Annals of the University of Craiova-Mathematics and Computer Science Series*, *38*(1), 62–71.

Masood, S. Z., Shu, G., Dehghan, A., & Ortiz, E. G. (2017). License plate detection and recognition using deeply learned convolutional neural networks. arXiv:1703.07330

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2013). Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv:1312.6082

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Kudlur, M. et al. (2016, November). TensorFlow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation* (pp. 265-283). USENIX Association.

CholletF. (2015). Keras.

Saleem, N., Muazzam, H., Tahir, H. M., & Farooq, U. (2016). Automatic license plate recognition using extracted features. In *2016 4th International Symposium on Computational and Business Intelligence (ISCBI)*, Olten (pp. 221-225). doi:10.1109/ISCBI.2016.7743288

Cao, Y., Qi, H., Zhou, W., Kato, J., Li, K., Liu, X., & Gui, J. (2018). Binary Hashing for Approximate Nearest Neighbor Search on Big Data: A Survey. *IEEE Access: Practical Innovations, Open Solutions*, *6*, 2039–2054. doi:10.1109/ACCESS.2017.2781360

de Campos, T. E., Babu, B. R., & Varma, M. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal.

Dlagnekov, L., & Belongie, S. (n.d.). UCSD/Calit2 Car License Plate, Make and Model Database. Retrieved from http://vision.ucsd.edu/car_data.html

Saquib Nadeem Hashmi is currently a 3rd-year BTech Computer Science Engineering Student of Jaypee Institute of Information Technology, Noida. His core area of interest and research are Computer Vision and machine learning.

Kaushtubh Kumar is currently pursuing his BTech in Computer Science and Engineering from Jaypee Institute of Information Technology. His core interest is machine learning, deep learning and artificial intelligence.

Siddhant Khandelwal is currently pursuing his BTech in Computer Science Engineering from Jaypee Institute of Information Technology, Noida and he has completed the 3rd year in June 2018. His current area of interest and research are Machine Learning, Computer Vision and Data Science. He has successfully completed notable Machine Learning training course from Coursera and Udemy. He also has a keen interest in Data Structures and Algorithms and has participated in many competitive coding competitions.

Dravit Lochan is a Btech 3rd year undergraduate student from Jaypee Institute of Information Technology. Android developer, Open Source enthusiast, with a good hand at Python for scripting work and C/C++ for implementing Data Structures. He is a senior developer at JDEV, readily takes any opportunity which caters his vision of helping people with their work. Did GSoC'17 as a student with FOSSASIA. GCI mentor'18, GSoC mentor'18. Student coordinator at GDG-JIIT Noida.

Dr. Sangeeta Mittal is Assistant Professor (Senior Grade) in Jaypee Institute of Information Technology, Noida. Her areas of research interest include Software Defined Networks, Computer Networks, Network Security, Cryptography, Sensor Based Smart Environments and Wireless Sensor Networks. She has published several papers in international conferences and journals of repute. She earned her PhD from Jaypee Institute of Information Technology, M.E. in Computer Engineering from Punjab University Chandigarh India and BE from Maharishi Dayanand University, Rohtak, India. She has 15+ years of experience in teaching UG and PG computer science courses. She is a member of ACM and life member of CSI.