License Plate Recognition using Convolutional Neural Network

Ahmed Afify, Amine Boussada, and Mahmoud Nasr

Abstract— **The demand of smart transportation systems has drastically risen in the last recent years due to the ever-increase of the car industry. License plate recognition (LPR) systems are massively used for traffic and law enforcement. The system is mainly composed of three stages. The first stage receives the cars image then finds the license plate. After that, the plate characters are segmented before being fed to the Convolutional Neural Network (CNN) for recognition. The CNN extracts the main features of the characters, which will be then employed for the detection of characters. In this paper, we introduce a LPR framework that works efficiently for both titled and blurry images and yields a high testing accuracy of 93% with the use of a small number of labelled license plate characters.**

I. INTRODUCTION

License plate recognition (LPR) system is one of the technologies that has been widely employed in many current transportation systems [1]. LPR systems are used for traffic and security systems such as traffic rule violation and custom security. License plate recognition systems have had a huge contribution in surveillance and traffic systems. However, there are still some problems that reduce their performance. Most of today's LPR systems are incapable of detecting plates when there are a large number of vehicles especially in a traffic jam. Most of the LPR systems are designed to work under normal conditions. Any deviations from these conditions can result in a system with poor performance. For example, the system does not perform properly under low lighting or poor weather conditions [2]. In addition, the orientation of the license plate can be another challenge of LPR.

License plate recognition system is usually divided into three stages; the first extracts the license plate from the image, and the second segments the image into separate characters. These characters are then recognized using the third stage of the system. License plate extraction and character segmentation steps are usually done using image processing techniques. When it comes to character recognition, there are two famous approaches.

One is based on template matching, and the second is based on Convolutional Neural Networks (CNN). Template matching utilizes a sliding window approach, where a window slides across the entire image and outputs a percentage match. If the window has a percent match that is above a predefined limit, it will recognize the desired object. However, when using CNN, the system is trained on a dataset and then is able learn the important features and generalize pretty well. For example, if we want to check if there is a car in an image, we do not need to check for all cars in the world as in template matching. In CNN, we probably need to train our network over thousands of car images to be able to find the car in an image. The ability to generalize is one of the unique features that make the CNN stand out over template matching and other detection methods. In addition, template matching is time consuming and impractical in real-time applications. Therefore, many recent LPR systems are CNN based. The only problem with using CNNs is that they require a large labelled dataset for training.

In this project, our main contribution is the design of a robust LPR system that can detect license plates that are blurry and tilted. These two issues have not been well investigated in the papers we reviewed earlier. We also followed a different approach for license plate extraction. The review papers used edge detection as a main step for the plate extraction, but this can affect the border of the license plate and makes the detection harder. Moreover, we developed a full system that can locate the plate, segment its characters, and then recognize them. This was not the case for most of the papers we studied. Labelled license plate characters are not readily available, and training on another dataset such as handwritten characters and then testing on LP characters proved to obtain very low accuracy. In this paper, this problem is remedied by augmenting a ready dataset with a small set labelled LP characters, thus saving a lot of time and effort that would have been spent in creating the large dataset required for this task.

The proposed LPR system consists of three main stages. The first is to locate the license plate, and the next stage is to segment the characters. The last is character recognition. Several image processing techniques were applied before feeding the input image to the CNN. This includes gray-scale conversion [3], binarization [4], sharpening, and contrast enhancement. For character segmentation, we used both horizontal projection [2], [4], [5] and vertical projection [4], [5], [6], [7]. Furthermore, we implemented LeNet-5 CNN for the character recognition. LeNet-5 CNN has better character recognition efficiency than other CNN architectures as it is usually used as a benchmark in LPR systems [3], [5], [6]. However, it is slower and more complex. By using this CNN, we were able to achieve a high prediction accuracy of 93%. This paper first examines image pre-processing techniques, then briefly introduces the methods utilized for plate localization and character segmentation. After that, detailed explanation for LeNet-5 CNN architecture and its implementation are given. The paper then concludes with a discussion of the advantages of our system as opposed to others followed with suggested future research investigation.

II. License Place Extraction

A. Contrast Enhancement

It is one of the most important techniques that is employed in a large number of applications in image processing. It is used to make different parts of the image clear through utilizing differences in luminance. Basically, it alters the histogram of the original image such that the pixels are rearranged between the brightest and darkest pixels of the image. As a result, different objects in the image are distinguished from each other and from the background. In our case, we first convert the RGB image to *CIELAB* color space. This color space consists of three components: L, a, and b. L refers to luminosity, a represents the green-red axis, and b represents blue-yellow color component. After the brightness of the pixels is changed, they are converted back to the original RGB space. It is worthwhile to mention that, by doing this, only the luminosity of the pixels is adjusted while the original colors remain unchanged. In the current implementation, the pixels' intensities are adjusted such that only 1% of them are among the brightest and darkest. Fig. 1 shows the effect of contrast enhancement on a license's plate image.

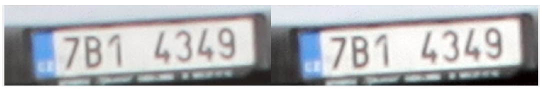

Fig.1. Original (left) vs Enhanced Image (right)

B. Image Sharpening

Image sharpening is utilized to enhance the contrast between bright and dark parts of the image to make certain details of the image more visible. Since such an approach would help fight blurry license plates, Unsharp masking is applied after contrast enhancement. Unsharp masking is one of the commonly used sharpening filters, which creates a blurred (unsharp) version of the original image then subtracts it from original image. As a result, this leaves out the more prominent details resulting in a clearer, less blurry image. Moreover, this process makes the edges sharper which is crucial for license plate extraction. After sharpening, the image is converted into grayscale then contrast enhancement is applied followed by a repeated application of sharpening to make sure that different features of the gray image are visible.

C. Binarization

After several enhancements are performed on the gray image, it is converted to black and white using binarization. It works by defining a threshold intensity above which all pixels are set to a value of 1 (white), while the rest are changed to 0 (black).

D. Closed Component Analysis

Finding closed contours is the last stage in license plate extraction. After the image is converted into black and white, the closed contour corresponding to the license plate has to be found. There are several methods for contour extraction, and one of the widely used methods is Closed Component Analysis (CCA). As demonstrated in [3] and [7], the algorithm tries to look for closed contours and arrange them according to their shapes. Since the plate is a rectangle, some constraints are assigned to find the correct closed contour as some of them do not represent the license plate. The area, height, and width of each found closed contours are calculated and then the aspect ratio between the height and width is used to determine if the shape is a square or a rectangle. The focus is set towards finding square and rectangular shapes in the image and not any other shapes. For the square, aspect ratio is almost equals to one. Since we may have several rectangles in the image, a minimum area is set on the rectangle. After that, only the rectangle that satisfies all these constraints is taken to be the license plate and therefore that area is extracted.

E. Tilt Correction

So far, we illustrated how the system works under normal environmental conditions and for poor lighting. However, there persists another problem in LPR systems which is studied in this work as well. The license plate may be rotated within the image and therefore it has to be rotated to make the plate in the horizontal position since the character segmentation which follows this stage can only work for images in normal positioning. Therefore, it is important to make sure that the image is well oriented before doing any segmentation for the characters. Almost all the papers reviewed did not address this issue, while those that addressed it, used the Hough transform to detect parallel lines in an image. However, Hough transform is not guaranteed to provide the right orientation angle as we may have several lines, and this makes it hard to decide which angles we should use. In the current system, after the license plate is found, the orientation angle of its region is found and then rotated to be horizontal. That is, Hough transform was not used to find the lines in the image, but instead, the orientation angle of the entire license plate rectangle is calculated then it is rotated accordingly. Fig. 2 demonstrates tilted correction.

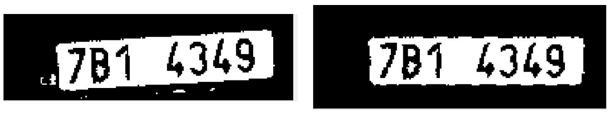

Fig.2. Tilt Correction

III. Character Segmentation

Character segmentation is a crucial step after preprocessing. It is used to divide license plate characters into individual characters which can then be classified (recognized) by the CNN. There are two main stages for character segmentation; horizontal projection and vertical projection.

A. Horizontal Projection

Horizontal projection serves to find the upper and lower bound of the plate characters. It helps in removing undesired regions of the plate [2], [4], [5]. Horizontal projection is a simple summation of the image pixels in horizontal direction along all vertical position lines. All projections above a certain threshold are found and then the height between these points (expected to be the lower and upper bounds) is used to check for the expected size of characters. Afterwards, the height that matches the expected size is used to extract the characters using the upper and lower bounds identified.

B. Vertical Projection

After the upper and lower regions of the characters are identified, vertical projection is applied to divide the image into individual characters. Vertical projection is obtained by adding the pixels in a vertical direction along the horizontal axis. The discontinuity in the projection histogram refers to the start and the end of the character. Fig. 3 shows the result of character segmentation on the LP shown in Fig. 1

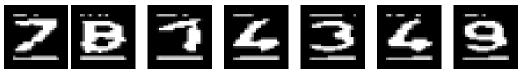
Fig.3. Segmented Characters

In some cases, vertical projection fails to detect the characters correctly. This may happen due to the presence of noise in the image or improper orientation of the license plate. For instance, if the image is tilted, a single character can be detected as more than one character. Moreover, license plates may have screws, and this can be detected as a character. To overcome this, aspect ratio constraint, which is the ratio between the width and the height of the character was used. A threshold was set for the width, height, and the ratio to precisely detect the desired characters. It is important to note that all images were resized to the same size before both horizontal and vertical projection to avoid problems that would arise in setting different thresholds for plates of different sizes. This is proven to be effective and robust against different conditions that affect the performance of a vertical projection given that the tilting was fixed previously..

IV. Character Recognition using CNN

The final step in the LPR system is recognizing the characters that were segmented. This is done using a convolutional network which is able to classify the segmented characters into 36 classes corresponding to 10 (0-9) digits and 26 English letters. For the current implementation, LeNet-5 CNN was utilized. LeNet-5 is a traditional CNN that was invented in 1998 by Yann LeCun, Leon Bottou, Yosuha Bengio, and Patrick Haffner [8]. It was able to learn the MNIST (70,000) raw hand-written digits images features and then automatically classify them into 10 classes (0-9). It achieved 1% error rate, which was close to the state of the art results at the time the network was introduced.

A. Network Architecture

Nowadays, LeNet-5 is considered a simple CNN architecture as it is composed of only 7 layers including 3 convolutional layers, 2 pooling layers, 1 fully connected layer, and an output layer. Yet, its accuracy in character recognition is very high and is very often used as a benchmark which makes it the perfect candidate for this work. Its architecture is shown in the following table:

TABLE I. LeNet-5 Layers Architecture

Layer	Type	Feature Map	Input Size	Filter Size	Stride
Input	Image	1	32×32	-	-
1	Convolution	6	28×28	5×5	1
2	Average Pooling	6	14×14	2×2	2
3	Convolution	16	10×10	5×5	1
4	Average Pooling	16	5×5	2×2	2
5	Convolution	120	1×1	5×5	1
6	Fully Connected	-	84	-	-
Output	Fully Connected	-	36	-	-

The last layer originally had 10 outputs (as the network is designed to identify 10 digits (0-9)). However, in our case, the network has 36 units to accommodate for the English alphabets besides the digits.

LeNet-5 is suitable for digit classification tasks. However, with the increasing complexity of the datasets, number of image samples, and more classes, it is not efficient for other image classification tasks such as the famous ImageNet or CIFAR-10 datasets.

On the other hand, LeNet-5 is the perfect neural network configuration for our task as we need to classify only 36 character classes, and the dataset that we used consist of about 38,000 images that is comparable in dataset size and image dimensions with the MNSIT dataset. Using a more complicated Deep neural network would need more training time, and can result in overfitting with our data.

B. Implementation

In our system, we implemented the LeNet-5 CNN, as explained above, but had to make an adjustment to it as it had designed 10 output nodes for 10 digits (0-9) as described previously. Moreover, the LeNet-5 CNN was originally used with the MNIST dataset, we attempted to train the network with different datasets, and the results are all mentioned below. In all of our implementations, the dataset was divided into 80% for training and 20% for validation. Testing was done on the characters extracted from the license plates using our own framework described above.

First, the network was trained on the EMNIST dataset [9], which is an extension of the original MNIST dataset but includes English letters in addition to the digits. This dataset included $425,600$ characters, and it obtained a training and validation accuracy of 94.1% and 91.2% respectively. However, when the trained model was tested on our license plate characters output, it achieved a mere 33.3% accuracy. This is explainable since the handwritten characters of EMNIST are very different in style from those of the license plates. Therefore, another experiment was conducted on the Char74k [10], which is composed of $36,000$ characters that are evenly distributed among each class (1000 for each). These characters were taken from street views and signs, and therefore include all kinds of fonts and shapes, which is assumed to generalize better. This dataset achieved higher training and validation accuracy (98.6% and 93.8%), but performed slightly worse on the test set with an accuracy of 29.5%.

The challenge was that the network was tested against data it has never encountered during training. As a result, the only solution was to provide it with a dataset of labelled characters extracted from license plates. Unfortunately though, no dataset was found online with these requirements, and therefore it was very hard to provide that due to the long time and effort it would require to generate such labelled data. In this work, a system that requires much less effort is presented. We augmented the already existent dataset with some labelled characters extracted from license plates. The Char74k was chosen as it, although achieves lower accuracy than EMNIST, is believed to be better in this case since it is much smaller and adding a small number of samples would not be ignored during training. 1500 samples were extracted and labelled which were different from the final testing data. The addition of a mere 4% made the accuracy sky rocketed from 29.5% to **93**%. The training accuracy was 98.9% and that achieved a validation accuracy of 94.1%. This shows that, with the addition of a small number of LP characters, the CNN learned to generalize much better and accuracy increased significantly. There were attempts with smaller LP samples. Adding 500 samples got the testing accuracy to 90.1%, and adding 1000 samples increased that to 92.0%. This supports the idea that as we add more training data similar to the testing data, we expect to get a much better accuracy. However, our approach eliminates the need of creating a large dataset by making use of an already existing dataset (Char74k) which is not directly related to our application thus saving a lot of time and effort whilst achieving a great performance.

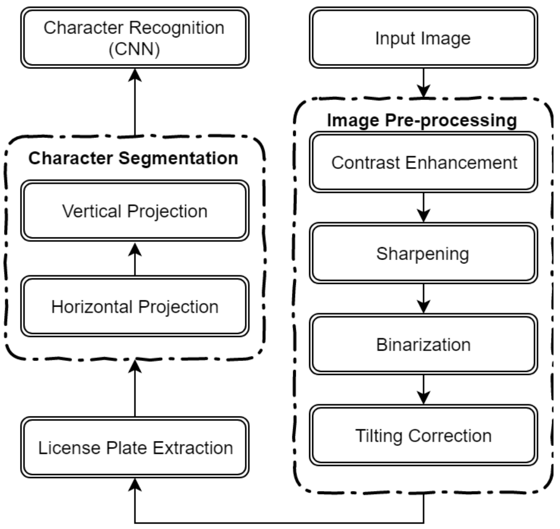

Fig.4. Full Proposed License Plate Recognition System

V. Discussion

The proposed LPR system can recognize the plate characters even if the image is blurry or titled; something that previous works did not consider. If the image is not properly positioned, a system that does not consider tilting will segment one character as two or more characters. Thus, it is very crucial to set the image to a normal orientation for correct segmentation. To orientate the image back to its original position, the angle of the license plate rectangle was used instead of using Hough transform for parallel lines detection, and we have explained this earlier in the paper. Additionally, the current implementation does not use Sobel operator for edge detection as other papers did. The operator is known for its simplicity and easy implementation on hardware. However, it is very sensitive to noise making license plate extraction harder. Instead, several image preprocessing techniques were introduced to enhance the content of the image and preserve its edges. First, contrast enhancement for the RGB image was performed to make different parts of the image clearer. After that, the image was sharpened to enhance the contrast between the bright and dark pixels. This made certain details of the image more visible including the edges. These two steps do not only enhance the image edges but also fix a bad lighting problem and thus, make the recognition possible under such conditions. Moreover, a full system was created that is able to detect a license plate and extracts its characters and then recognize (classify) them using CNN. On the other hand, to our knowledge, no other works were made in developing such a complete system that combines both image processing techniques and CNN. Other papers developed a system that can either extract the plate and segment its characters or a CNN that is able to classify the characters. Fig. 4 illustrates the full proposed system. To obtain high accuracy, a large dataset of license plate characters was required but such a dataset is not available online. The proposed system was initially trained on the Char74k dataset but achieved very low accuracy of 29%. By augmenting our generated characters with the dataset, the accuracy increased to 93%, which provides an approach similar to *Transfer Learning* in tackling cases which do not have enough labelled data.

VI. Conclusion

In this work, we introduced a robust LPR system that performs well in harsh environmental conditions, namely blurriness, low lighting, and tilting. Previous research work did not design a full LPR system that tackles all these issues. For future work, we are considering optimizing the LeNet-5 architecture to make it simpler and boost its recognition accuracy further. Moreover, we are aiming to implement our system on a real-time hardware. Additionally, Generative adversarial networks (GANs) can be utilized to map LP characters to handwritten characters, then test the resulting images using a CNN trained on handwritten characters. This will make the testing data more similar to the training data and therefore result in a higher accuracy and reduce the time taken to generate and label characters for the training set.

References

[1] H. Li, R. Yang, and X. Chen, "License plate detection using convolutional neural network," in *Computer and Communications (ICCC), 2017 3rd IEEE International Conference on*. IEEE, 2017, pp. 1736–1740.

[2] C.-H. Lin, Y.-S. Lin, and W.-C. Liu, "An efficient license plate recognition system using convolution neural networks," in *2018 IEEE International Conference on Applied System Invention (ICASI)*. IEEE, 2018, pp. 224–227.

[3] P. Rajendra, K. S. Kumar, and R. Boadh, "Design of a recognition system automatic vehicle license plate through a convolution neural network," *International Journal of Computer Applications*, 2017.

[4] S. Saunshi, V. Sahani, J. Patil, A. Yadav, and S. Rathi, "License plate recognition using convolutional neural network," *IOSR Journal of Computer Engineering (IOSR-JCE)*, 2017.

[5] V. H. Pham, P. Q. Dinh, and V. H. Nguyen, "Cnn-based character recognition for license plate recognition system," in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2018, pp. 594–603.

[6] Q. Wang, "License plate recognition via convolutional neural networks," in *Software Engineering and Service Science (ICSESS), 2017 8th IEEE International Conference on*. IEEE, 2017, pp. 926–929.

[7] G. Kosala, A. Harjoko, and S. Hartati, "License plate detection based on convolutional neural network: Support vector machine (cnn-svm)," in *Proceedings of the International Conference on Video and Image Processing*. ACM, 2017, pp. 1–5.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[10] T. E. De Campos, B. R. Babu, M. Varma *et al.*, "Character recognition in natural images." *VISAPP (2)*, vol. 7, 2009.