




CNN-Based Character Recognition for License Plate Recognition System

Van Huy Pham^(✉) , Phong Quang Dinh, and Van Huan Nguyen

Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
phamvanhuy@tdt.edu.vn

Abstract. License Plate Recognition is a practical use of computer vision based application. With the increase in demand of automation transportation systems, this application plays a very big role in the system development. Also, the use of vehicles has been increasing because of population growth and human needs in recent years makes the application is more challenging. Moreover, license plates are available in diverse colors and style and that the presence of noise, blurring in the image, uneven illumination, and occlusion makes the task even more difficult for conventional recognition methods. We propose an approach of using a Convolutional Neural Networks (CNN) classifier for the recognition. Pre-processing techniques are firstly applied on input images, such as filtering, thresholding, and then segmentation. Then, we train a CNN classifier for character recognition. Although the performance of a CNN is very impressive, it costs much time to complete the character recognition step. In this study, a modified CNN is proposed to help the system run in real-time. Experimental results have done and analyzed with other methods.

Keywords: Convolution Neural Network · Character recognition
License Plate Recognition System

1 Introduction

License Plate Recognition (LPR) systems are very popular and studied all over the world. A LPR system is a combination of several modules and involves object detection, image processing, and pattern recognition. Beside image acquisition and preprocessing, the process of reading a license plate goes through 3 main phases. The first phase is plate localization or plate extraction. The second one is the character segmentation which each character is detected and separated from the others. The last one is the character recognition which the extracted characters are recognized. Because of quality of image and the variance of plate's shape, size, color, orientation, it is difficult to finding the plate location. The character segmentation may be in trouble due to quality of image, partially connected characters, noise, and rotation of plate. The last phase also has to deal with variation in fonts, similarity among characters and non-characters.

In the past two decades, many different methods have been proposed and tested. The purpose of this study is to provide a comparison for some of existing method and then propose an improvement in performance of our system which is based on different

methods of character recognition: Artificial Neural Network (ANN) and Convolutional Neural Networks (CNN).

The fact is that ANN-based OCR can work in real-time, but the performance of the system is not as good as CNN-based systems. Although the use of CNN yields better performance, it is very slow in both training and recognition phase. In our experiment using CNN Lenet-5, the license plate system recognized one character in about 11 ms, but the overall system needs to process in 30 ms to avoid false detection. Therefore, in this study, we proposed a CNN structure which processes 20× faster than ordinary CNN Lenet-5 that can help the system run in real-time.

2 Related Works

There are many difficulties that a LPR system must overcome. Due to the camera zoom factor, the extracted characters do not have the same size and thickness. Resizing the characters into one size before recognizing helps to overcome this problem. The character's font is not the same all the time since different states, different countries use different fonts. Extracted characters may have some noise or may be broken. In the following, we categorize some existing character recognition methods.

Ahmed et al. [1] have presented a template matching. Template matching is a simple method. The similarity between a character and the template is measure. The template that is the most similar to the character is recognized as the target. Most template matching methods use binary images because the gray-scale is changed due to any change in the lighting. Template matching is performed after resizing the extracted character into the same size. This method is useful for recognizing single font, non-rotated, non-broken. If a character is different from the template due to any font change, rotation, noise, the template matching produces incorrect recognition.

LeCun et al. [2] used HOG-feature for character recognition. At the training phase, training data was generated from a high-resolution image of each letter and distribution of each letter in the HOG-feature space was then obtained. At the recognition phase, each character is cut out from the image, calculated the HOG-feature vector, and recognized characters with the distribution in HOG-feature space obtained above. Siddharth et al. [3] used Support Vector Machine (SVM) classifier. SVM classifier is trained by a given set of training data and a model is prepared to classify test data based upon this model. For multiclass classification problem, we decompose multiclass problem into multiple binary class problems, and we design suitable combined multiple binary SVM classifiers. According to how all the samples can be classified in different classes with appropriate margin, different types of kernel in SVM classifier are used. Commonly used kernels are: Linear kernel, Polynomial kernel, Gaussian Radial Basis Function (RBF) and Sigmoid (hyperbolic tangent).

Sharma and Singh [4] has applied Artificial Neural Network for character recognition. This method simulates the way human neural system works to create intelligent behavior. The idea is to take a large number of characters, known as training set, and then develop a system which can learn from those training. In other words, the neural network uses the training to automatically infer rules for recognize character.

These methods can achieve impressive result on good dataset but still produce incorrect recognition in hard dataset. However, we can do much better if we introduce some concept from CNN theory. Bounchain [5] has applied Lenet-5 in character recognition task. This network was tested with a database containing more than 50,000 hand-written digits, all normalized in the input image. An error rate of about 0.95% was achieved. This is an impressive result on OCR task.

In this study, we will apply our CNN for OCR for the system running in real time by reducing some unnecessary convolutional layers.

3 The Proposed Method

3.1 Plate Detection and Segmentation

Plate detection. In order to localize license plates in a given image, different methods have been used. The simple and fast method is primarily based on identifying the vertical edges of each license plate in the input image [6]. After extracting vertical edges from the image, morphological filtering is applied to obtain candidate regions. Then spatial features such as area, aspect ratio and edge density are considered to discard wrong candidate regions.

In our study, we use the methods in [1, 7–9] for plate detection and segmentation. A combination of Hough Transform and a counter algorithm in [10] is used to detect license plate region. In the first step, the counter algorithm is applied to detect close boundaries of the objects. These counter lines are transformed to Hough coordinate to find interacted parallel lines that are considered as license plate candidates. To filter out the candidate plates, the aspect ratio and the horizontal cross cuts are used. In [7–9], Connected Component Labelling (CCL) is used license plate detection. CCL scans the image the labels the pixels according to the pixel connectivity. In [8], a feature extraction algorithm is used to count the similar labels to distinguish it as a region. The region corresponding with the maximum area is considered as a possible license plate region. Likewise, in [7], two detection methods are performed to detect white frame and to detect black characters. To determine the candidate frames, aspect ratio of license plate, height and width of characters have to be known. In [9], after the successful CCL on the binary image, measurements such as orientation, aspect ratio for every binary object in the image are calculated. Criteria such as orientation $< 35^\circ$, $2 < \text{aspect ratio} < 6$ are considered as candidate plate regions.

Character segmentation. In [11], adaptive thresholding is applied to binary the input image. Then connected component analysis is applied for character segmentation. The components obtained from the process may be character or non-character. Character's aspect ratio is considered to suppress non-character. The ratio is set based upon the observation from different images.

The method of row-column scan is chosen to segment characters in [10]. Firstly, the line scan method is used to scan the binary image and lower-upper bounds are located. Secondly, the column scan method is chosen to scan binary image and the left-right bounds are located. Based on these, each character can be accurately segmented.

Experimental results show that this method can even handle license plate images with fuzzy, adhering, or fractured characters with high efficiency.

In [1], the strategy is based on pixel count (i.e., vertically projecting and counting the number of pixel in each column). First, the image is converted into binary form and vertical projections are made. Then, the number of black pixels in each column is counted, and a histogram is plotted. The characters are segmented depending on the transition from a crest to its corresponding trough. To avoid unnecessary segmentation, some thresholds are taken properly.

3.2 Convolutional Neural Networks for Character Recognition

Convolutional neural networks have been successfully applied in the field of computer vision. Unlike artificial neural network, the layers of a CNN have neurons arranged in 3 dimensions: width, height, depth (Fig. 1). The neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected layer.

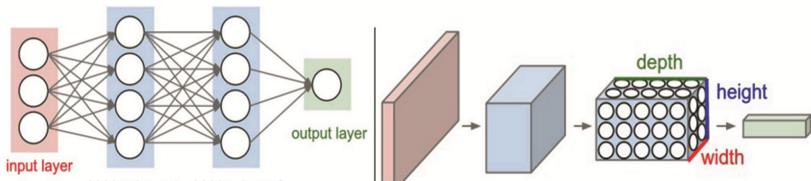


Fig. 1. Left: A regular 3-layer neural network. Right: A Convolutional Neural Networks arranges its neurons in 3 dimensions (width, height, depth) as visualized in one of the layers. Every layer of a CNN transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (red, green, blue). (Color figure online)

A general CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function. There are 3 main types to build CNN architecture: convolutional layer, pooling layer, and fully-connected layer (exactly as seen in artificial neural network). These layers will be stacked to form a full CNN architecture. In more detail, a simple CNN classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. INPUT holds the raw pixel values of the image with 3 color channels. CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. RELU layer applies an elementwise activation function, such as the $\max(0, x)$ threshold at zero. POOL layer performs a down-sampling operation along the spatial dimensions (width, height). FC layer computes the class scores. As with artificial neural networks, each neuron in this layer will be connected to all the numbers in the previous volume. In this way, CNN transforms the original image from original pixel values to the final class score.

In this study, we follow the work of using Lenet-5 in [2]. Lenet-5 comprises 7 layers, not counting the input, all of which contain trainable parameters (Fig. 2). In the

following, convolutional layers are labeled Cx, subsampling layers are labeled Sx, and fully-connected layers are labeled Fx, where x is the layer index. Layer C1 is a convolutional layer with 6 feature maps. Each unit in each feature is connected to a 5 × 5 neighborhood in the input. The size of the feature maps is 28 × 28. Layer S2 is a subsampling layer with 6 feature maps of size 14 × 14. Each unit in each feature map is connected to a 2 × 2 neighborhood in the corresponding feature map in C1. The 2 × 2 receptive fields are non-overlapping, therefore feature maps in S2 have half the number of rows and columns as feature maps C1. Layer C3 is a convolutional layer with 16 feature maps. Each unit in each feature map is connected to several 5 × 5 neighborhoods at identical locations in a subset of S2’s feature maps. Layer S4 is a subsampling layer with 16 feature maps of size 5 × 5. Each unit in each feature map is connected to a 2 × 2 neighborhood in the corresponding feature map in C3, in a similar way as C1 and S2. Layer C5 is a convolutional layer with 120 feature maps. Each unit is connected to a 5 × 5 neighborhood on all 16 of S4’s feature maps. Layer F6 contains 84 units and is fully connected to C5. As in artificial neural networks, units in layers up to F6 compute a dot product between their input vector and their weight vector, to which a bias is added. This weighted sum is then passed through a sigmoid function. Finally, the output layer is composed of Euclidean Radial Basis Function units (RBF), one for each class, with 84 input each. The output of each RBF unit y_i is computed as follow.

$$y_i = \sum_j (x_j - w_{ij})^2$$

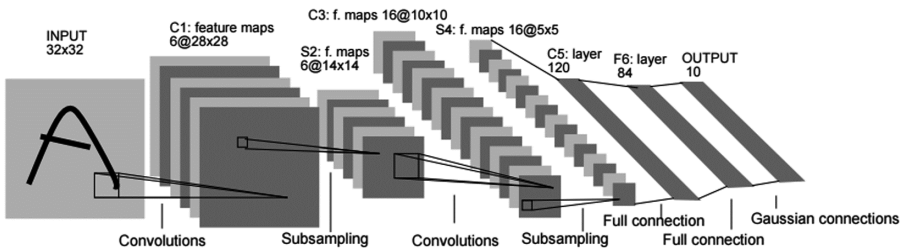


Fig. 2. Architecture of Lenet-5 [2]

In other words, each output RBF unit computes the Euclidean distance between its input vector and its parameter vector. The further away is the input from the parameter vector, the larger is the RBF output.

Lenet-5 is designed to extract local geometric feature from the input field in a way that preserves the approximate relative locations of these features. This is done by creating feature maps that are formed by convolving the image with local feature-extraction kernels. Lenet-5 has several advantages that make it attractive for recognizing characters when high variability is expected. First, Lenet-5 has state-of-the-art accuracy by its impressive performance. Second, Lenet-5 runs at high speeds on specialized hardware. Lenet-5 can also be readily trained to recognize new character styles and fonts. It works well for both handwritten and machine printed characters.

3.3 Our Modification of Lenet-5

The modified Lenet-5 comprises 6 layers, not counting the input. Layer C1 and C3 has 4 feature maps. Each unit in each feature map C1 is connected to a $4 \times 5 \times 5$ neighborhood in the input. Each unit in each feature map C3 is connected to $4 \times 3 \times 3$ neighborhoods at identical locations in a subset of S2's feature maps. Layer S2 and S4 is a subsampling layer with 4 feature maps. Layer F5 is a fully-connected layer with 1024 units. Each unit is connected to all units in S4's feature maps. Output layer contains 35 units and is fully connected to F5. As can be seen, we remove the last convolutional layer and we just use 4 feature maps in C1, C3 convolutional layer. The reason is there are only about 3 or 4 gray levels on each real character image. Therefore, we don't need many features as the original Lenet-5. The time-consuming tasks occur mostly in the convolutional layers, so reducing in the number of features makes our CNN faster. Moreover, each unit in each feature map C3 is only connected to 3×3 , instead of 5×5 neighborhoods, as the thickness of the characters is about 3 pixels. This can help to reduce the processing time (Fig. 3).

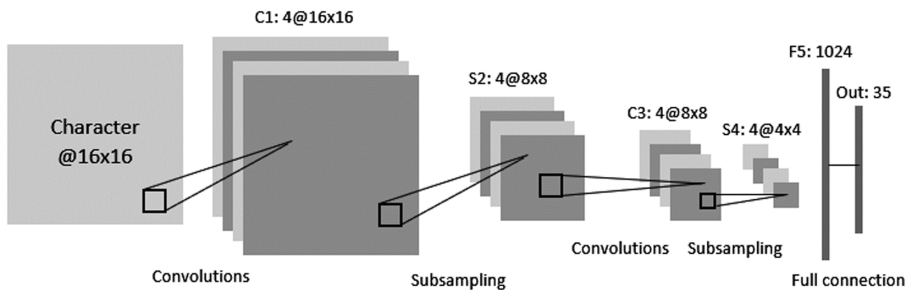


Fig. 3. The proposed Lenet-5 with less feature maps in convolutional layers

4 Experimental Results

For experiments, we used two datasets for evaluation: MNIST used in [2] and our collection. The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits commonly used for training various image processing system. Figure 4 shows some example characters in the database. The MNIST database contains 60,000 training images and 10,000 testing images.

For comparison, we built our own dataset from real-life traffic car images. On each of collected images, we extracted the regions which have a single character. The segmented characters are sorted into 35 categories: 0–9, A–Z. All the characters are resized to 16×16 before training. Figure 5 shows some examples of license plate characters in our dataset. The dataset was collected from US traffic images using a simple LPR commercial product in our company.

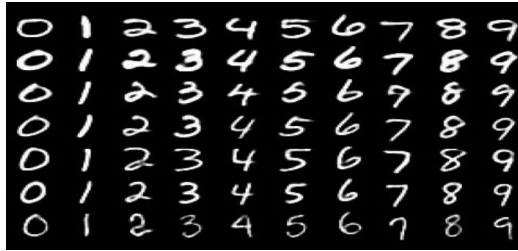


Fig. 4. Examples from MNIST dataset



Fig. 5. Car license plates from some traffic images

We use a dataset of 112,000 samples to train, and test on various states in US (Cali test set). The models used for comparison are ANN Lenet-5 and our proposed CNN, and we evaluate on the accuracy and processing time for each of the models.

Firstly, these models was trained with MNIST dataset and tested with our test set. Table 1 shows the accuracy and time process of ANN, CNN Lenet-5 and our proposed CNN in the MNIST dataset. The CNN-based models have best performance, and our proposed method is at a bit lower accuracy at 98.77% compared with 99.10% of Lenet-5 but the processing time is much lower at 0.38 ms compared with 10 ms of Lenet-5.

Table 1. The accuracy and time process of ANN, CNN Lenet-5 and our proposed CNN in the MNIST dataset.

Data test	ANN		CNN Lenet-5		Proposed CNN	
	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)
MNIST	96.820	0.050	99.190	10.158	98.770	0.386

For better and more practical evaluation, we divide the test set into 3 categories: normal and good dataset containing characters in common and good conditions; and hard dataset containing characters challenging the detection. Some examples are shown in Fig. 6.

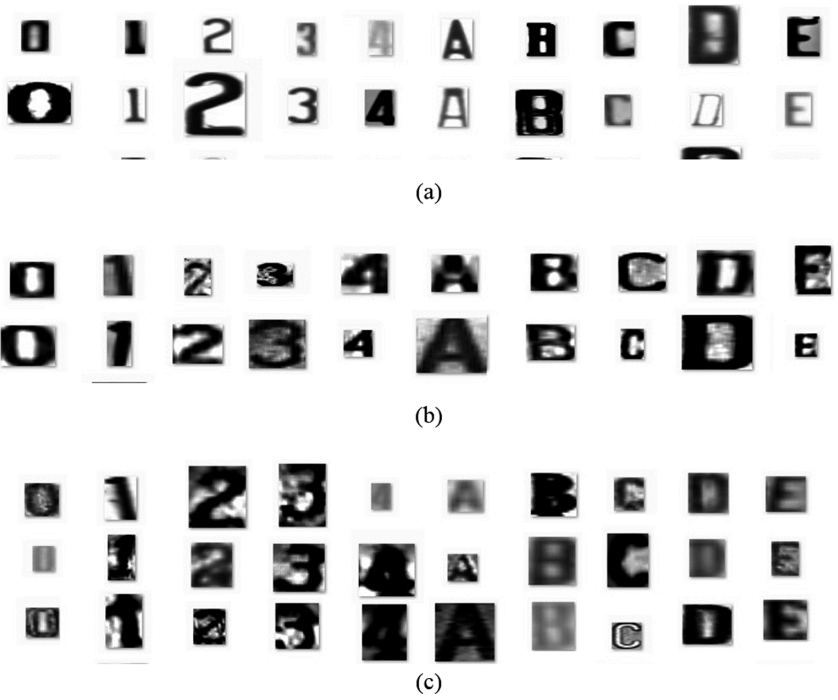


Fig. 6. Examples of good (a), normal (b) and hard (c) dataset

Tables 2, 3 and 4 show the accuracy and processing time of the 3 testing models on 3 levels of the testing datasets.

Table 2. The accuracy and processing time on ANN, CNN Lenet-5 and our proposed CNN with good dataset.

Data test	ANN		CNN Lenet-5		Proposed CNN		Number of sample
	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	
Arizona	96.160	0.055	99.745	11.239	98.162	0.507	1500
Idaho	94.900	0.054	99.565	11.502	97.819	0.503	1500
Missouri	93.378	0.055	99.454	11.309	97.640	0.547	1500
Texas	94.277	0.055	99.135	11.439	97.801	0.573	1500
Average	94.679	0.055	99.475	11.372	97.856	0.533	

Table 3. The accuracy and processing time on ANN, CNN Lenet-5 and our proposed CNN with normal dataset.

Data test	ANN		CNN Lenet-5		Proposed CNN		Number of sample
	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	
British Columbia	89.382	0.078	94.547	11.532	92.010	0.479	1500
WashingtonDC	92.117	0.054	95.343	11.396	93.885	0.509	1500
Georgia	90.429	0.054	95.149	11.372	93.804	0.595	1500
Vermont	88.147	0.058	94.987	11.276	91.653	0.667	1500
Average	90.019	0.061	95.007	11.394	92.838	0.562	

Table 4. The accuracy and processing time on ANN, CNN Lenet-5 and our proposed CNN with hard dataset.

Data test	ANN		CNN Lenet-5		Proposed CNN		Number of sample
	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	
Delaware	84.942	0.054	89.894	11.892	88.153	0.499	1500
Maryland	86.983	0.055	90.510	11.388	88.468	0.526	1500
Average	85.963	0.055	90.202	11.640	88.310	0.512	

Evaluation of the overall system is based on averaging the accuracy and the processing time of each models on the datasets. Table 5 shows the evaluation on our collected dataset.

Table 5. The average in accuracy and processing time of ANN, CNN Lenet-5 and our proposed CNN in our collected dataset

Data test	ANN		CNN Lenet-5		Proposed CNN	
	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)
Average	90.220	0.057	94.894	11.469	93.001	0.536

The accuracy of our proposed CNN is higher about 3% than ANN and less 2% than CNN Lenet-5. Although the time processing is still slower 10× than ANN, it is 20× faster than CNN Lenet-5. Therefore, our proposed CNN can still run in real time at a practically acceptable accuracy.

5 Conclusion

Starting from a basic classifier, various concepts for character recognition were introduced. In this study, we proposed a CNN model for character recognition task. The CNN lenet-5 is modified by reducing the number of convolution layer and the size of neighborhood to become our model. The model was tested with 36,500 images of character.

Although the accuracy is less 2% than Lenet-5, our proposed CNN gave satisfactory of performance and real-time processing.

References

1. Ahmed, M.J., Sarfraz, M., Zidouri, A., Al-Khatib, W.G.: License plate recognition system. In: Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems, 2003, ICECS 2003, pp. 898–901. IEEE (2003)
2. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998)
3. Siddharth, K.S., Jangid, M., Dhir, R., Rani, R.: Handwritten Gurmukhi character recognition using statistical and background directional distribution. *Int. J. Comput. Sci. Eng. (IJCSE)* **3**, 2332–2345 (2011)
4. Sharma, S., Singh, N.: Optical character recognition using artificial neural networks approach. *Int. J. Emerg. Technol. Adv. Eng.* **4**, 339–344 (2014)
5. Bouchain, D.: Character recognition using convolutional neural networks. *Inst. Neural Inf. Process.* 2007 (2006)
6. Zheng, D., Zhao, Y., Wang, J.: An efficient method of license plate location. *Pattern Recogn. Lett.* **26**, 2431–2438 (2005)
7. Wen, Y., Lu, Y., Yan, J., Zhou, Z., von Deneen, K.M., Shi, P.: An algorithm for license plate recognition applied to intelligent transportation system. *IEEE Trans. Intell. Transp. Syst.* **12**, 830–845 (2011)
8. Caner, H., Gecim, H.S., Alkar, A.Z.: Efficient embedded neural-network-based license plate recognition system. *IEEE Trans. Veh. Technol.* **57**, 2675–2683 (2008)
9. Anagnostopoulos, C.N.E., Anagnostopoulos, I.E., Loumos, V., Kayafas, E.: A license plate-recognition algorithm for intelligent transportation system applications. *IEEE Trans. Intell. Transp. Syst.* **7**, 377–392 (2006)
10. Jin, L., Xian, H., Bie, J., Sun, Y., Hou, H., Niu, Q.: License plate recognition algorithm for passenger cars in Chinese residential areas. *Sensors* **12**, 8355–8370 (2012)
11. Kumari, S., Gupta, D., Singh, R.M.: A robust method for vehicle license plate recognition based on Harries corner algorithm and artificial neural network. *Int. J. Comput. Appl.* **148**, 16–19 (2016)