Car Plate Character Recognition Using a Convolutional Neural Network with Shared Hidden Layers

Yujie Liu
School of Electronics and Information Engineering,
Soochow University,
Suzhou 215006, P. R. China
Email: lancelod.liu@gmail.com

He Huang
School of Electronics and Information Engineering,
Soochow University,
Suzhou 215006, P. R. China
Email: cshhuang@gmail.com

***Abstract*—In character recognition, convolutional neural network (CNN) outperforms most of the other models. However, to guarantee a satisfactory performance, CNNs usually need a great number of samples. Due to the differences between the Chinese and the alphanumeric characters, the most common way to recognize the two classes is to use two independent CNNs respectively. In this paper, to solve the problem of the Chinese character shortage, we implement a CNN model which has shared hidden layers and two distinct softmax layers for the Chinese and the alphanumeric character respectively. To avoid over-fitting problem in the training process, the early stopping rule is employed. By training and testing on our two small-scale car plate character databases, the model gets a 9.289% and 9.632% relative reduction in test errors for the Chinese and the alphanumeric characters respectively over conventional CNN models.**

Keywords*—car plate, character recognition, convolutional neural network, shared hidden layer.

I. Introduction

Unlike other models, convolutional neural networks have the ability of preserving the neighborhood relations and the spatial locality of the input in their hidden higher-level feature representations [1]. Inspired by the concept of simple and complex cells in the visual cortex, CNNs consist of alternating layers of feature extraction (simple cells) and sub-sampling (complex cells). In contrast to fully connected neural networks, the extensive use of shared weights reduces the number of degrees of freedom without the loss of expressive power, which makes CNNs easy to train by gradient descent algorithm [2]. Hence, convolutional neural networks have been applied widely in visual recognition.

A CNN model with layer-skipping is introduced in [3] to deal with the vehicle type classification problem and excels. A novel deep CNN is proposed for wild animal monitoring [4]. Many models and training strategies are proposed to solve handwritten character recognition problem [5] [6] [7] [8] [9] which exhibit the viability of CNN models. As a promising pre-training method, convolutional auto-encoder (CAE) performs well in getting better initial kernels of CNN and reduces the final recognition error [1] [10].

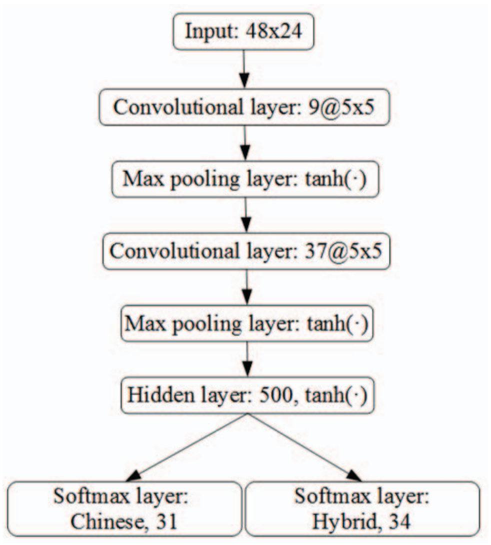
For car plate character recognition task, artificial neural network is the mostly used model [11]. Multi-layer perceptron (MLP) is utilized in [12] [13]. Local binary pattern (LBP) and radial basis function (RBF) network is introduced in [14]. Character thinning and fuzzy classifier is proposed in [15]. Fourier descriptors of the fuzzy outline is utilized to distinguish characters in [16]. Besides, based on LBP features, a bilayer classifier is proposed in [17]. However, most of these methods need a pre-defined feature extractor to construct the input of the trained classifier while CNN models use trainable feature extraction layers instead.

Experiment in [1] shows that with more training samples, CNNs could achieve a more satisfactory recognition result. However, in some specific applications, the number of training samples is often restricted which leads to a problem of sample shortage. To achieve a better performance without collecting more samples, various solutions have been proposed. A usual practice is to introduce random degree of distortion and gaussian noise into training samples. As suggested in [1], it is feasible and practical. Trained by patches extracted from the input images, the model outperformed traditional methods using hand-crafted features [10]. In addition to the two practices, for cross tasks learning, a resource-limited system and a resource-rich system can benefit from each other under the MLP model [18] [19]. A shared hidden layer, cross-language architecture of deep neural network is proposed in [20] which proves that the feature transformations can be shared across different tasks. Recently, a novel architecture of deep CNN is introduced in [21] and demonstrates that the convolutional kernel, as a common feature extractor, can be shared as well as the hidden layers even the characters from different tasks are not similar to each other.

As for the study of Chinese car plate character recognition, the first Chinese character is the most difficult to collect for two reasons. Firstly, the Chinese character in the plate represents the province issuing the car plate and therefore every Chinese car plate only contains one Chinese character. Secondly, since the land area of China is so large that in any province of China, we barely see cars registered in other provinces and as a result, it is difficult to get some certain characters of the Chinese character set such as the characters representing Tibet and Hainan province. In contrast, the alphanumeric characters are much easier for us to collect. Thus, the sample number of the alphanumeric is often much larger than the number of the Chinese characters. Due to the large number of samples, the alphanumeric classifier could achieve a satisfactory result while the Chinese classifier suffers from the sample shortage problem. Inspired by [21] and taking the similarity between the Chinese and the alphanumeric characters such as strokes and font styles into consideration, we implement the architecture and simplify it to a five layers structure.

To obtain character samples, an efficient color–edge location algorithm is applied. Then a modified connected component analysis method is utilized to segment the plates. Finally, we get 2092 Chinese and 12552 alphanumeric characters to train our model. Experiment result shows that due to the full utilization of samples, the model gets a better performance than the conventional CNN model which demonstrates the advantage of the shared hidden layers structure.

II. Shared Hidden Layer CNN



Input: 48x24
Convolutional layer: 9@5x5
Max pooling layer: tanh(·)
Convolutional layer: 37@5x5
Max pooling layer: tanh(·)
Hidden layer: 500, tanh(·)
Softmax layer: Chinese, 31
Softmax layer: Hybrid, 34

Fig. 1. The architecture of the shared hidden layer CNN

An overview of the shared hidden layer CNN (SHL-CNN) model we used is given in Fig. 1. In this model, all the convolutional-maxpooling layers and hidden layers are shared across both Chinese and alphanumeric task. These hidden layers work as a common feature extraction and transformation module. The bottom softmax layers are not shared and work as two task-oriented classifier modules [21]. Unlike the model of [21], there is no contrast normalization layer in our model.

A. Architecture of the model

In our experiment, all the input are reshaped into 48×24, converted to grayscale and normalized. We set the size of each convolution kernel to 5×5 and the kernel set number of the first and the second convolution layer to 9 and 37 respectively. Taking both accuracy and convergence speed into consideration, we set the neuron number of the hidden layer to 500. As for the bottom layer, the number 31 and 34 stand for the number of Chinese provinces and alphanumeric characters ($0 \sim 9, A \sim Z$ without I and O) respectively. Except for the softmax layer, all the activation functions are hyperbolic tangent function to implement the nonlinear transformation.

1) Convolution layer: Each convolutional layer owns several sets of kernels $\{KernelSet_i\}$ and the number of kernels owned by every $KernelSet_i$ is the same as the number of input feature images. The i^{th} output of one convolutional layer, denoted by $ConvOut_i$, is got by convolving all the input feature images with the corresponding kernel in the $KernelSet_i$ and summing all the convolution outputs as (1).

$$ConvOut_i = \sum_{j=1}^{N} input_j * KernelSet_i[j] \tag{1}$$

where $*$ means convolution operation and N stands for the number of kernels owned by $KernelSet_i$. In this paper, we use grayscale image as input. Thus, the setting of the first convolution layer is 9×1 where 9 stands for the number of the kernel set and 1 stands for the size of each kernel set. As the output images of the first convolution-maxpooling layer are the input to the second convolution layer and hence, the setting of the second convolution layer is 37×9. The demonstration of the convolution process is shown as Fig. 2.

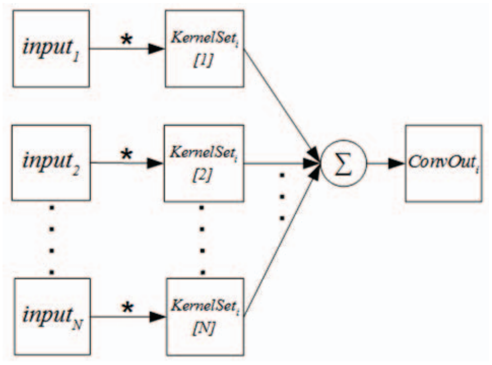


Fig. 2. Convolution operation

2) Maxpooling layer: Taking the maximum value over non overlapping sub-regions, maxpooling downsamples the latent representation from the convolution layer and enhances the feature extraction ability of kernels by introducing sparsity over the hidden representation. With such a sparse latent code, kernels are forced to be more general [1]. The maxpooling is defined as (2).

$$\begin{aligned} PoolOut_i(r,c) = Max\,\{ConvOut_i(m,n)\} \\ r \in [0, H/ps), c \in [0, W/ps) \\ m \in [r \times ps, (r+1) \times ps), \\ n \in [c \times ps, (c+1) \times ps) \end{aligned} \tag{2}$$

where H and W are the height and width of $ConvOut_i$ and ps is the pool size of maxpooling. The implementation of maxpooling takes two steps. Step one is to partition the input matrix $M_{H \times W}$ into $\frac{H}{ps} \times \frac{W}{ps}$ non overlapping sub-regions whose size is $ps \times ps$. Step two is to take the maximal value of every sub-region as the corresponding value in the output matrix.

After maxpooling the output of the convolution layer, we add a bias b_i to the maxpooling output and get the output of one convolution-maxpooling layer as (3):

$$output_i(x,y) = \tanh(PoolOut_i(x,y) + b_i) \tag{3}$$

From (3) we can see that it is every kernel set shares the same bias across all the kernels it owns.

3) Hidden and Softmax layer: The two convolution-maxpooling layers are trained as feature extractors. Thus, the input of the hidden layer can be regarded as latent representation extracted from the input images. Because the output of the convolution-maxpooling layer is a set of feature images whose shape is $R \times C$, we need to reshape all of them into $RC \times 1$ and then link them one by one to fit the input shape of hidden layer. After that, we can get the output of every hidden neurons as:

$$y_i^{hidden} = \tanh(\sum_{k=1}^{K} W_{ik} \times x_k^{input} + b_i^{hidden}) \tag{4}$$

Then, we can get the input of each output neuron according to (5):

$$x_j = \sum_{i=1}^{I} W_{ji} \times y_i^{hidden} + b_j^{output} \tag{5}$$

Based on the softmax definition shown as (6), we can get the probability that the input image is a member of class j:

$$p_j = \frac{e^{x_j}}{\sum_{q=1}^{Q} e^{x_q}} \tag{6}$$

where Q is the number of classes.

Finally, with maximal probability selected out, a definition of the prediction of the model is demonstrated as follows:

$$Y = argmax\{p_i | i = 1, 2, ..., Q\} \tag{7}$$

where $argmax\{\cdot\}$ returns the index of the maximal member in the input set.

4) Cost and Error: For a given batch of input-output pairs $\{(x^i, d^i) | i \in [0, D)\}$, the log-likelihood is defined as:

$$L = \frac{1}{D} \sum_{i=1}^{D} ln(p_{d^i}) \tag{8}$$

where D is the batch size of input samples and p_{d^i} represents for the probability that the input x^i could be classified into class d^i.

Let $D = 1$, from (8) we can see that L varies from from $-\infty$ to 0.0 when p_{d^i} varies from 0.0 to 1.0. From the intuition we know that the cost usually should increase when the likelihood decreases. Therefore, the negative likelihood is introduced to define the *cost* as (9):

$$cost = -\frac{1}{D} \sum_{i=1}^{D} ln(p_{d^i}) \tag{9}$$

The cost function is used during training procedure. Nevertheless, to evaluate the performance of a model, the error rate is defined as (10):

$$E = \frac{1}{D} \sum_{i=1}^{D} 1\{Y^i != d^i\} \tag{10}$$

where Y_i is the corresponding index of the maximal probability of the softmax neurons and $1\{\cdot\}$ is defined as (11).

$$1\{condition\} = \begin{cases} 1 & condition = True, \\ 0 & condition = False. \end{cases} \tag{11}$$

B. Training of the model

In general, the gradient descent method is used to update all the model parameters. The learning rates of all four layers are $0.2000, 0.1600, 0.1280$ and 0.1024 correspondingly in order to balance the gradient decay. To get a fast convergence speed and avoid local minimal problem, all the initial weights are randomly generated from a uniform distribution. The distribution interval is determined by the input and the output numbers [22]. For $tanh(\cdot)$ activation function, results obtained in [22] show that the initial weight interval should be $[-\sqrt{\frac{6}{fan_{in}+fan_{out}}}, \sqrt{\frac{6}{fan_{in}+fan_{out}}}]$, where fan_{in} is the number of neurons in the $(i-1)^{th}$ layer, and fan_{out} is the number of neurons in the i^{th} layer.

As a common problem of complex models, over fitting happens if the training could not stop at a proper time. Over fitting models usually show great fitness for the training set while perform badly in the testing set. To get rid of it, we implement an early stopping rule described as Algorithm 1.

```
$patience$ = $N$
$test\_freq$ = $N/4$
$iter = 0$, $epoch = 0$, $n\_epoch = 50$
$best\_error = \infty$, $stopping = False$
$train\_batch\_number = N/10$
while $epoch < n\_epoch$ and not $stopping$ do
    for $mini\_batch$ in $[0, train\_batch\_number)$ do
        $cost$ = train($mini\_batch$, $CNN$)
        update($cost$, $CNN$)
        $iter = iter + 1$
        if $iter = patience$ then
            $stopping = True$
            break
        end
        if $iter = test\_freq$ then
            $E$ = test($mini\_batch$, $CNN$)
            if $E < best\_error$ then
                $patience = patience * 1.5$
                $best\_error = E$
            end
        end
    end
end
```

Algorithm 1: Early stopping algorithm

During the training phase, the parameters of the shared hidden layers are updated all the time. On the contrary, the parameters of the Chinese and the alphanumeric softmax layer are updated by the Chinese and the alphanumeric samples respectively. When the training procedure is done, the model can be used to recognize both Chinese and alphanumeric characters by means of using the corresponding softmax layer. Utilizing the joint training strategy, we can make improvement in the recognition error of the two classes of characters.

III. Experiment and Results

We have collected 2092 Chinese characters and 12552 alphanumeric characters and labeled them manually. To compare with the conventional CNN model, we construct an identical CNN model which owns the same structure of convolution-maxpooling layers and the same number of hidden neurons as the SHL-CNN except for that it only has one softmax layer while the SHL-CNN model owns two. In addition, all the train settings keep invariant to both models.

A. Databases

There are two databases used in our experiment. Table I gives an overview of the two databases.

TABLE I. Database Structure

Database	Train	Test	All
Chinese	1051	1041	2092
Alphanumeric	6285	6267	12552

The first database consists of 2092 Chinese characters which are labeled according to the province or district they represent for. To construct the train and the test set, we have partitioned these characters fifty-fifty by province respectively and the specific distribution is shown as Fig. 3.

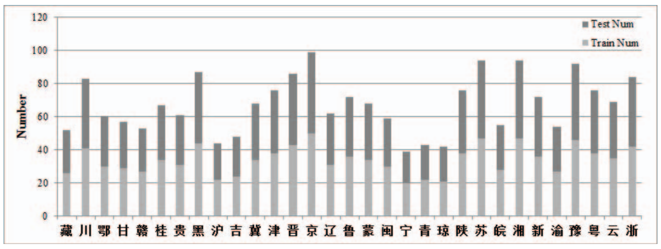


Fig. 3. Chinese Database

The second database consists of 12552 alphanumeric characters and considering that neither character “I” nor “O” exists in Chinese car plate, they are labeled as 34 classes in all and are partitioned fifty-fifty by class as the first database for the same purpose and the specific distribution is shown as Fig. 4. We must point out that the two partition operations on the two databases are performed randomly to avoid subjective effects.

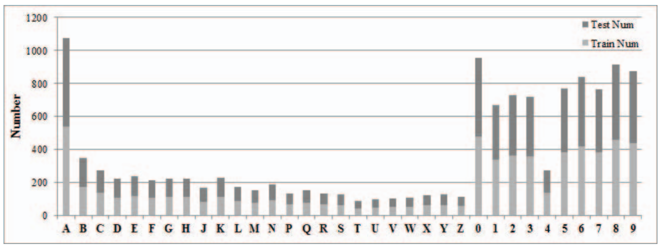


Fig. 4. Alphanumeric Database

Fig. 5 shows a small part of the two databases from which we can conclude that both databases own a variety of illumination conditions and distortion degrees. To build the two databases, we processed 2189 real life images based on the following three-step procedure.

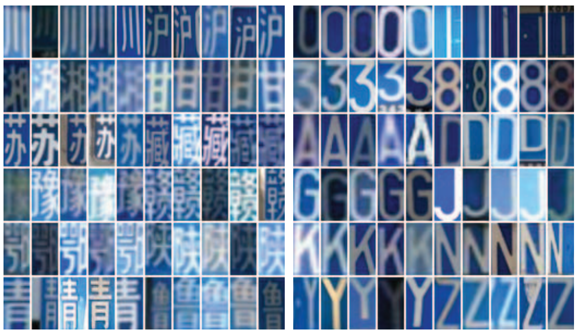


Fig. 5. Training samples

1) Location step: First of all, the input RGB image is separated into R map, G map and B map. Then, three corresponding edge maps, E_R, E_G and E_B, are obtained using Sobel operator. Based on the three edge maps, we construct the blue–white edge image E_{BW}. Then, the edge density image D is got by convolving E_{BW} with a mean matrix with size 13×25, and D is binarized by Otsu method. After that, all the connected components of D are found to calculate morphology information. Finally, the connected components satisfying morphology restrictions are reserved as the candidates.

In this step, 2166 out of 2189 plates are obtained and the location success rate is 98.95%.

2) Segmentation step: At first, all the plates got from the first step are converted to gray images according to (12), (13).

$$I(i,j) = (0.7R(i,j) + 0.2G(i,j) + 0.1B(i,j)) \times f(R(i,j), G(i,j), B(i,j)) \tag{12}$$

where

$$f(x,y,z) = e^{-std(x,y,z)/150} \tag{13}$$

To eliminate horizontal frames, according to [23], a row relocation algorithm is applied based on projection analysis. After that, all the plates are binarized and the connected component (CC) analysis is applied to segment them and finally, 14644 character samples (2092 Chinese and 12552 alphanumeric) are acquired and the segment success rate is 96.58%.

3) Labeling and storage step: The last step is the labeling and storage where all the characters are labeled manually and reshaped to 48×24 RGB image files. However, before being sent to the CNN, all the character samples are converted to grayscale and normalized to $0.0 \sim 1.0$. Except for the grayscale conversion, no further pre-processing is applied to the samples in consideration of the generalization performance of the model.

B. Comparison with conventional CNN and MLP

We use the two aforementioned databases to test the SHL-CNN model and the two conventional CNN models. We train all the models until the early stopping rule takes effect or the maximal training epoch is reached. Results shown in Table II reveal the reduction in error compared with the conventional

CNN model. For both Chinese and alphanumeric databases, SHL-CNN shows a better performance than the conventional CNN. The absolute reductions (AR) in error are 0.384% and 0.131% and the relative reductions (RR) in error are 9.289% and 9.632% for the Chinese and the alphanumeric respectively. Comparison with the common model MLP which has 200 hidden neurons is also given in Table III. As a more complex model, convolutional neural networks excels much.

TABLE II. RECOGNITION ERROR COMPARISON WITH CON-CNN

Database	Model		AR	RR
	Con-CNN	SHLCNN		
Chinese	4.134%	3.750%	0.384%	9.289%
Alphanumeric	1.360%	1.229%	0.131%	9.632%

TABLE III. RECOGNITION ERROR COMPARISON WITH MLP

Database	Model		AR	RR
	MLP	SHLCNN		
Chinese	7.780%	3.750%	4.030%	51.799%
Alphanumeric	1.704%	1.229%	0.475%	27.876%

Fig. 6 and 7 describes the correlation between the test error and the number of train samples in the two databases. In consideration of simplicity, the classes with zero error rate have not been included in the figures.

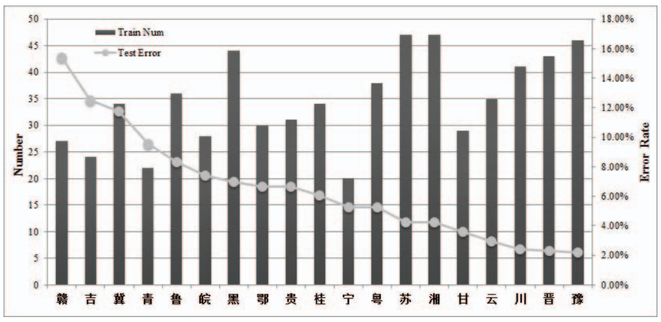


Fig. 6. Test error and sample number on the Chinese Database

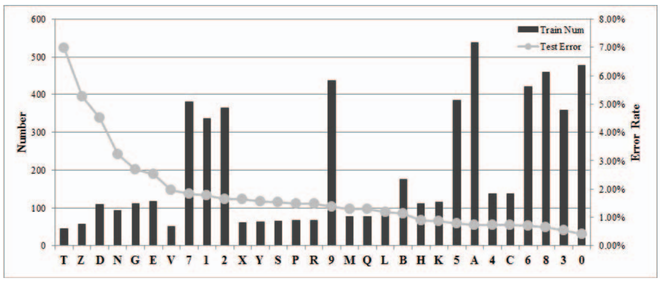


Fig. 7. Test error and sample number on the Alphanumeric Database

The result indicates that the imbalance of sample distribution in database does affect the performance on different classes. For an instance, from Fig. 6 we can see that the first character, "Gan", has the worst error rate because it only has 27 training samples. Meanwhile, the second character, "Ji", has less training samples than "Gan" but a better error rate because it has less strokes than "Gan" which makes the pattern easier for the model to learn. In addition, in Fig. 7, "T" has a worse error rate of 6.98% than the "1" because "T" shares part of stroke features with "1" and we use 336 "1"s but only 44 "T"s to train the model. In short, the more strokes one class of character has or the more similar one class is to other classes, the more samples it will need to improve the performance on the class.

As an important metric of the license plate recognition system, the average consuming time of recognizing one single character is $12.7\ ms$ which leads to the result that the total consuming time of recognizing 7 characters is $88.9\ ms$.

IV. CONCLUSIONS

In this paper, the SHL-CNN model is implemented for car plate character recognition task. The idea of sharing the hidden layers and using different final softmax layers for different tasks proves to be beneficial for the representation extraction ability of the model. Experiment on the Chinese and alphanumeric characters demonstrates that all the car plate characters, no matter which class they belong to, can share the same front feature extractors, namely the convolution-maxpooling and the hidden layers. Under the SHL-CNN framework, error reductions are verified in both databases.

The performance could be improved furthermore. With a proper learning rate setting strategy and an appropriate sample re-arrangement in training phase, we expect to train the model faster and to get a better error reduction. Besides, a solution to the imbalanced sample distribution is under construction. These are the focus of our future work.

REFERENCES

[1] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *International Conference on Artificial Neural Networks*, DOI. 10.1007/978-3-642-21735-7_7, pp. 52-59, 2011.

[2] R. Wagner, M. Thom, R. Schweiger, G. Palm, and A. Rothermel, "Learning Convolutional Neural Networks From Few Samples," *International Joint Conference on Neural Networks*, DOI. 10.1109/IJCNN.2013.6706969, pp. 1-7, 2013.

[3] Z. Dong, M. Pei, Y. He, T. Liu, Y. Dong, and Y. Jia, "Vehicle Type Classification Using Unsupervised Convolutional Neural Network," *International Conference on Pattern Recognition*, DOI. 10.1109/ICPR.2014.39, pp. 172-177, 2014.

[4] G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," *International Conference on Image Processing*, DOI. 10.1109/ICIP.2014.7025172, pp. 858-862, 2014.

[5] C. Wu, W. Fan, Y. He, J. Sun, and S. Naoi, "Handwritten Character Recognition by Alternately Trained Relaxation Convolutional Neural Network," *International Conference on Frontiers in Handwriting Recognition*, DOI. 10.1109/ICFHR.2014.56, pp. 291-296, 2014.

[6] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "Flexible, High performance convolutional neural networks for image classification," *International Joint Conference on Artificial Intelligence*, pp. 1237-1242, 2011.

[7] 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.

[8] P. Simard, D. Steinkraus, and J. Platt, "Best practices for convolutional neural networks applied to visual document analysis," *International Conference on Document Analysis and Recognition*, DOI. 10.1109/ICDAR.2003.1227801, pp. 958-963, 2003.

[9] M. L. Yu, P. C. K. Kwok, and C. H. Leung, "Segmentation and recognition of Chinese bank check amounts," *International Journal on Document Analysis and Recognition*, vol.3, no.4, pp. 207-217, 2001.

[10] M. Wang, Y. Chen, and X. Wang, "Recognition of Handwritten Characters in Chinese Legal Amounts by Stacked Autoencoders," *International Conference on Pattern Recognition*, DOI. 10.1109/ICPR.2014.518, pp. 3002-3007, 2014.

[11] H. K. Sulehria, Y. Zhang, D. Irfan, and A. K. Sulehria, "Vehicle number plate recognition using mathematical morphology and neural networks," *WSEAS Transactions on Computers*, vol. 7, no. 6, pp. 781-790, 2008.

[12] L. Angeline, W. Y. Kow, W. L. Khong, M. Y. Choong, and K. T. K. Teo, "License Plate Character Recognition via Signature Analysis and Features Extraction," *International Conference on Computational Intelligence*, *Modelling and Simulation*, DOI: 10.1109/CIMSim.2012.66, pp. 1-6, 2012.

[13] M. Nejati, H. Pourghassem, and A. Majidi, "Iranian License Plate Character Recognition Using Mixture of MLP Experts," *International Conference on Communication Systems and Network Technologies*, DOI: 10.1109/CSNT.2013.55, pp. 219-223, 2013.

[14] X. X. Chen, and C. Qi, "A super-resolution method for recognition of license plate character using LBP and RBF," *International Workshop on Machine Learning for Signal Processing*, DOI: 10.1109/MLSP.2011.6064550, pp. 1-5, 2011.

[15] L. Jin, H. Xian, and J. Bie, "License Plate Recognition Algorithm for Passenger Cars in Chinese Residential Areas," *SENSORS*, vol. 12, no. 6, 2012.

[16] J. Zhao, S. Ma, W. Han, Y. Yang, and X. Wang, "Research and implementation of license plate recognition technology," *Chinese Control and Decision Conference*, DOI: 10.1109/CCDC.2012.6244605, pp. 3768-3773, 2012.

[17] G. S. Hsu, J. C. Chen, and Y. Z. Chung, "Application-Oriented License Plate Recognition," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 2, pp. 552-561, 2013.

[18] A. Stolcke, F. Grezl, M. Hwang, X. Lei, N. Morgan, and D. Vergyri, "Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons," *International Conference on Acoustics*, *Speech and Signal Processing*, vol. 1, pp. I-I, 2006.

[19] C. Plahl, R. Schluter, and H. Ney, "Crosslingual portability of chinese and english neural network features for french and german lvcsr," *Workshop on Automatic Speech Recognition and Understanding*, DOI. 10.1109/ASRU.2011.6163960, pp. 371-376, 2011.

[20] J. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," *International Conference on Acoustics*, *Speech and Signal Processing*, DOI. 10.1109/ICASSP.2013.6639081, pp. 7304-7308, 2013.

[21] J. Bai, Z. Chen, B. Feng, and B. Xu, "Image character recognition using deep convolutional neural network learned from different languages," *International Conference on Image Processing*, DOI. 10.1109/ICIP.2014.7025518, pp. 2560-2564, 2014.

[22] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *International Conference on Artificial Intelligence and Statistics*, vol. 9, JMLR W&CP 9, pp. 249-256, 2010.

[23] W. Li, D. Liang, X. Wang, and D. Yu, "Character segmentation for degraded license plate," *Journal of Computer-aided Design & Computer Graphics*, vol. 16, no. 5, pp. 698-703, 2004.