Handwritten Chinese Characters Recognition Using Two-Stage Hierarchical Convolutional Neural Network

Nina Aleskerova
Moscow Institute
of Physics and Technology
Computer Vision and Natural Language
Processing Laboratory (ABBYY Lab)
Email: nina.antonova@phystech.edu

Aleksei Zhuravlev
Moscow Institute
of Physics and Technology
Computer Vision and Natural Language
Processing Laboratory (ABBYY Lab)
Email: zhuravlev@phystech.edu

Abstract—**Convolutional Neural Networks (CNNs) are widely used for handwritten character recognition tasks. The task of handwritten Chinese character recognition is especially outlined by a large number of target classes. The total number of Chinese characters exceeds 40,000, the average Chinese uses several thousand characters in his speech. In this article, we will work with about 4,000 Chinese characters. Dealing with a large number of classes might be problematic for a single-network approach in terms of both speed and accuracy, especially when running on a CPU. In this paper, we suggest an approach to meet this challenge to do separate handwritten Chinese characters recognition. We propose that a hierarchy of multiple neural networks should be used wherein a first-level network chooses a second-level network that performs the final recognition. Further, as every second-level network is only trained for a subset of classes, it can be simpler and more accurate than the whole classification network. In our approach, we combine several classes of Chinese characters with similar external features in one group (cluster) and work with these clusters further. Experimental results on CASIA 200-class and 3755-class handwritten datasets are presented, which compare the proposed hierarchical approach with the classical single-network approach.**

I. Keywords

handwritten Chinese characters recognition, Hierarchical CNN, clustering, CASIA Offline dataset.

II. Introduction

In the present time, handwritten character recognition problem becomes more popular because of its various applications. Deep CNNs are known to solve well this problem but they can be ineffective in terms of the number of required resources and classification quality. Therefore, new approaches are required to solve typical for the group of tasks problems related to resources and required accuracy. The number of common Chinese characters is too high for the problem of handwritten symbols recognition to be successfully solved by conventional classification methods [1]. We need a new method aimed at training on a sample with a large number of classes. Our approach, representing hierarchical deep CNN, could become well-suited tools for such problems.

The approach that we used is more suitable for increasing the network accuracy when dealing with a classification problem with a large number of classes.

Hierarchical deep neural networks are not widely used so far, but there are several previous works that describe using hierarchical CNNs in different tasks of visual recognition, for example, [2] and [3], in part, in handwritten Chinese characters recognition [4]. In the last work, experiments are conducted on the same dataset as ours and their approach with identifying groups of similar elements in the first stage and searching for differences between these similar characters within each group in the second stage is generally similar to our approach.

There are several works on solving the problem of recognizing handwritten Chinese characters working on the same dataset and using methods that are somehow similar to the one used by us. Besides [1] that is based on Residual CNN and [4], we can also mark the works [5], [6], [7] and [8] since all of them solve the problem of handwritten Chinese characters recognition. In [5] author represents a method using classical CNNs of various depths, with experiments on the same datasets as in our work. In [6] a method for constructing classical CNN with the selection of the optimal loss function is described. In both works, a network has a traditional one-stage structure. In [7] the authors propose a method to recognize Chinese characters by multitype attributes, which are based on radicals, pronunciation, and structure of characters. This approach is significantly different from ours and from the rest of the methods we mentioned since it uses knowledge of the very structure of Chinese characters and it can be used to recognize the characters which are not in the training set. [8] proposes a specific for CASIA dataset radical-based approach to handwritten Chinese characters recognition, namely a radical analysis network with densely connected architecture (DenseRAN). Compared to all of the above, our method is a hierarchical classification and it is aimed at improving quality compared to the traditional CNN. It also does not use internal knowledge about the structure of the character.

In this paper we will propose the construction of our two-stage hierarchical neural network, then we will describe the experiments that we conducted and the datasets we used for them. At the end of the paper, we will present the results of our experiments and plans and suggestions on how to improve them.

III. Model description

In this section, we will describe a general model for constructing a hierarchical classifier. This method can be applied to a neural network with any architecture.

As it was already mentioned, the method consists of two stages, or levels. In the first stage, all training samples are divided into a given number of clusters, each of which contains all elements of some classes. Then we train the first-level classifier to identify the number of cluster to which the element is assigned according to its class label. Since it is practically impossible to attribute all samples of the same class to one cluster, some of the data will be classified incorrectly, which leads to errors of the first-level classifier. Therefore, in the second stage, we will cover the clusters with sets of classes that are more extended compared to the result obtained in the first stage to reduce the number of classification errors. In the end, we train second-level classifiers to finally identify original class labels. The amount of these classifiers is equal to the number of clusters.

The first-level classifier is capable of identifying and distinguishing the different subsets of the data. But first, we have to identify these subsets based on the results of a neural network with a simple architecture that can distinguish all target classes.

Such a partition of the set into subsets is equivalent to constructing a map from the set of all classes of size N to the set of clusters of size M:

$$f : Y \to C$$

$$Y = \{0, 1, ...N-1\},\ C = \{0, 1, ...M-1\}, M \leq N$$

In order to make the training of the first level classifier possible, first, the data should be subjected to the following processing:

1) let us have a training set: $D = \{(x_i, y_i) | x_i \in X, y_i \in Y\}$
2) on set D we train a classifier that is capable of distinguishing all the N classes
3) we shuffle training set D and take a part of it, the size of which should not exceed 20 % of the size of D.

$$\hat{D} \subset D$$

$$\hat{D} = \{(x_i, y_i) | x_i \in \hat{X}, y_i \in \hat{Y}\}$$

4) we form an extractor of features from original data by deleting the last fully connected layer from our trained network and, if necessary, adding another fully connected layer with an output shape equal to the number of components in the feature vector. This number depends on a dataset and on a number of classes and it is better to choose it empirically.

$$ext : X \to F,\ F \in R^k,$$

where k is the size of feature vector.

5) we pass the subset through the extractor to obtain a set of feature vectors for each element $x \in \hat{X}$:

$$\hat{F} = ext(\hat{X})$$

6) we cluster the set of feature vectors by some clustering algorithm. For example, in our experiment, K-Means algorithm was used. As a result, M clusters are formed:

$$g : \hat{F} \to C,$$

7) after that, we assign each of the original classes to the cluster that contains the largest number of train samples of this class, that is:

$$f(y) = \underset{c \in C}{argmax} |\{(x, y) \in \hat{D} \mid g(x) = c\}|$$

After the mapping f is created, we can mark all the data from training set D by their cluster indices, i.e., we form a new training sample:

$$\overline{D} = \{(x_i, f(y_i)) : (x_i, y_i) \in D\}$$

Then we train the first-stage classifier on this training sample: for each training element it predicts the number of cluster in which the element lies according to the class which it belongs to.

On figure 1 examples of ten randomly chosen clusters are represented. One can notice that elements that have been placed into one cluster have some common external features.

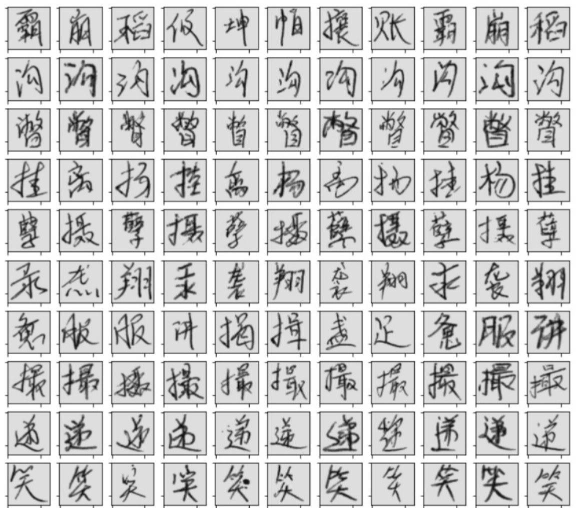

Fig. 1. Examples of ten clusters built on the first stage: Chinese characters lying on the same row belong to the same cluster

In order to build second-level classifiers based on clustering that we got on the first stage, we have to define again sets of classes on which these classifiers will be trained. We could use the same partition as at the previous step of our two-stage construction, but it may not be optimal, due to the fact that the first-level classifier makes mistakes one way or another while it assigns samples to clusters.

Therefore, it is proposed to compensate for the frequency errors of the first-level classifier by constructing extended sets of classes for second-level classifiers. In order to do this, we will use the results of the first-level classifier on some pending samples.

So, the sets of classes for the second-level classifiers are constructed as follows:

1) we shuffle again training set D and take another part of it as S, the size of S should be about 15 - 20 % of the size of D.
2) for each element $(x, y) \in S$ we define the result of first-stage clustering as $s(x) \in C$, where s is a composition of two functions:
$$s(x) = g(ext(x))$$
3) for each original class $y \in Y$ and for each cluster $c \in C$ we build a confusion matrix: we count the amount of samples from S that have been assigned to class y wherein they have been placed to cluster c by first-stage classifier:
$$d(y, c) = |\{(x, y) \in S : s(x) = c\}|$$
4) for each original class $y \in Y$ we count the probability distribution on the set consisting of all the clusters: that is, we count the proportion of elements belonging to class y that have been assigned to cluster c in respect to the amount of all elements belonging to class y:
$$r_y(c) = \frac{d(y, c)}{\sum_{i=0}^{M} d(y, i)}$$
5) for each of the original classes $y \in Y$ we form a list of pairs:
$$l_y = [(r_y(c), c)|\ c \in C]$$
6) we sort every one of these lists by decreasing of first argument
7) for each $y \in Y$ we process the list l_y. First, we initialize $T = 0$ - share of elements of this class covered by the current set of sets of classes. Then we process pairs (r, c) from the list:
   - we add class number y to the set number c
   - we update current coverage $T \to T + r$
   - we stop processing after we get $T \geq T_{acc}$

   Here, T_{acc} is a hyper-parameter of algorithm that shows what share of elements of every class has to remain to be correctly classified in S. By setting different values of T_{acc}, one can get various results of network speed and quality. Higher values of T_{acc} will lead to larger sizes of the resulting sets of classes. Consequently, that will lead to a longer network training time and may cause controversial changes in accuracy. It means that for each experiment, the hyperparameter value should be specially selected.

We should note that the sets obtained as a result of this replenishment operation already can intersect and they will do so, in contrast to the sets built in the first-level classifier, because now some class may lie in the replenished set of every cluster that contains a sufficiently large number of elements belonging to this class.

After we construct these target sets of classes for each of the original clusters, every second-level network is trained on classification problems within the corresponding set. It can have the same architecture as a first-level network, or another. Practically, these sets (even after the process of adding classes with the results of the first-level classifier) are still much smaller than the original dataset. This means that trained classifiers can be arranged quite simpler and, at the same time, achieve high-quality values.

It is important to note that the total error of the hierarchical CNN practically consists of two types of errors - the error of the first-level classifier and the error of the second-level classifiers. Errors of the first type are related to the fact that the first-level classifier can constitute an extended set of classes (that is, the set obtained by replenishing the clusters with errors of the first-level classifier), into which the target class is simply not included. In this case, the second-level classifier a priori cannot make the right decision in choosing the target label. At the same time, errors of the second type occur when the first-level classifier worked correctly, that is, the target class is present in the extended set of classes, but the second-classifier made an error in choosing the target class.

IV. Experiments

A. Datasets

For our work we used one of CASIA Offline Chinese Handwriting Databases (CASIA-HWDB1.1)[9].

This dataset was created by 300 different authors in order to provide the various spelling of characters. The database has 1,172,907 characters samples in total which contain 51,158 symbols (that is 171 alphanumeric and symbols) and 1,121,749 Chinese character samples. All these samples are united into 3,755 separate classes, each of them representing one Chinese character, but variously handwritten by several authors.

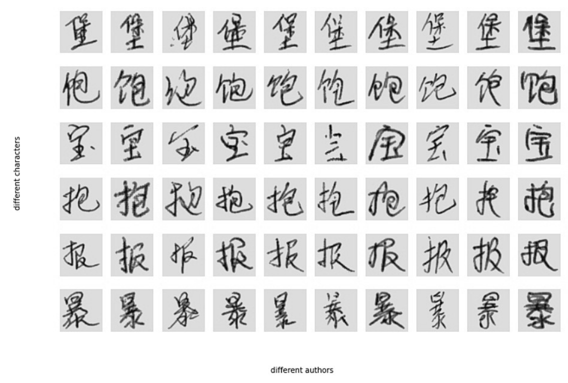

Fig. 2. Various versions of six random handwritten Chinese characters written by different authors

On figure 2 several examples of various spelling of the same character by different authors can be observed.

In the Competition dataset there are only 224,419 Chinese characters but it also contains 3,755 classes representing Chinese characters.
We also created a small dataset that was derived from CASIA-HWDB1.1 by the selection of samples belonging to the 200 classes representing the most common Chinese characters in this dataset and by limitation of the number of elements in the dataset. This small dataset only contains 40,000 train samples. Nevertheless, the results that are obtained by using a two-stage classification method on this dataset, can be significant because this experiment combines recognition of the most popular Chinese characters with much faster recognition than on the main dataset.
For fair comparison with previous works we also used Competition Test Dataset [10] that was used for Chinese Handwriting Recognition competition in 2013 and was based on CASIA-HWDB/OLHWDB databases.

B. Network parameters

As was already mentioned, we conducted two experiments with different numbers of classes, therefore, different parameters were selected for them.
We used a popular framework Pytorch for both networks training since it met the necessary requirements for the convenience of working with our networks and datasets.
Data preprocessing, besides converting the dataset into HDF5 format, only included the selection of 200 most common classes of Chinese characters for the first experiment and getting rid of extra symbols (not hieroglyphs) in the second one, without any data augmentation.
Network architecture is different for experiments with a 200-classes dataset and a 3755-classes dataset. As was already mentioned, we used quite simple architecture in order to just evaluate the effect of a hierarchical approach to the problem. One can combine a more complex architecture with this method to solve problems of handwriting character recognition with any other datasets.
The structure of the CNN for the experiment with 3755 classes can be seen in table 1.
The architecture of network for the 200-classes dataset is similar to the one provided in table 1, but is a little simpler, with four convolutional layers and the extractor as an extra linear layer due to a much smaller output size required (equal to the number of components we took) than in architecture given in table 1.
For the extractor, we took such sizes as 51 components and 1024 components for datasets with 200 classes and 3755 classes, correspondingly. In both cases, the extractor is used instead of the last fully connected layer for the feature extraction.
The first-level classifier has the same architecture as corresponding conventional classifier whose architecture is provided in one of the tables, except that it has the output shape equal to the number of clusters into which we divided all the elements of the dataset.
Second-level classifiers also have the same architecture as the corresponding conventional classifier, except that each of the second-level classifiers has the output shape actually equal to the number of classes containing in the corresponding cluster after replenishing it with first-level classifier errors.
The complete identity of the architectures of these networks was made in order to test the pure effectiveness of our hierarchical approach.
For our experiments we chose such values for the number of clusters at the first level: 20, 40 and 80 clusters for the dataset with 200 classes and 400 and 800 clusters for the dataset with 3755 classes in order to evaluate the dependence of network performance on the number of clusters.
In the first series of experiments, we took the hyperparameter value $T_{acc} = 1$ to find the maximum replenished sets of clusters for two datasets and corresponding network architectures. In the second series of experiment, we varied the values of T_{acc} on the same dataset and network architecture to find out how this hyperparameter affects the resulting accuracy.
What concerns other parameters of our network, we used Adam as optimization method [11] with learning rate = 0.001, 5 epochs for the dataset with 200 classes and 7 epochs for the one with 3755 classes in both conventional CNNs and hierarchical CNNs.

TABLE I
NETWORK ARCHITECTURE FOR 3755-CLASSES EXPERIMENT

Layer	Description	Output
$Input$	grayscale image with size 64 x 64	1 x 64 x 64
$Conv_1$	kernel size = 3, stride = 1, BN, ReLU	64 x 64 x 64
$Max\ pool$	2 x 2 max pooling stride = 2	64 x 32 x 32
$Conv_2$	kernel size = 3, stride = 1, BN, ReLU	128 x 32 x 32
$Max\ pool$	2 x 2 max pooling stride = 2	128 x 16 x 16
$Dropout$	dropout ratio = 0.25	128 x 16 x 16
$Conv_3$	kernel size = 3, stride = 1, BN, ReLU	256 x 16 x 16
$Max\ pool$	2 x 2 max pooling stride = 2	256 x 8 x 8
$Conv_4$	kernel size = 3, stride = 1, BN, ReLU	512 x 8 x 8
$Max\ pool$	2 x 2 max pooling stride = 2	512 x 4 x 4
$Dropout$	dropout ratio = 0.25	512 x 4 x 4
$Conv_5$	kernel size = 3, stride = 1, BN, ReLU	512 x 4 x 4
$Max\ pool$	2 x 2 max pooling stride = 2	512 x 2 x 2
$Dropout$	dropout ratio = 0.25	512 x 2 x 2
$Linear$	fully connected layer, BN, ReLU	1 x 1024
$Extractor$	the results of $Linear$	1 x 1024
$Dropout$	dropout ratio = 0.25	1 x 1024
$Classifier$	fully connected layer	1 x 3755

Batch sizes were equal to 100 and 500 for these two datasets

in case of conventional CNN and in first-level classifier. However, on the second stage the sizes of datasets used in each second-level classifier vary widely. The size of the dataset depends on the number of classes contained in the replenished set based on the corresponding cluster. Still, this size is in any case much smaller than the size of a full dataset. Therefore, the batch size for these datasets has to be smaller. For example, we took batch size = 20 in experiment with 200 classes and 40 clusters and batch size = 40 in experiment with 3755 classes and 800 clusters. Apparently, the more number of clusters we set in an experiment with the same dataset, the less we should take this parameter.
Figure 3 shows the distribution of the number of classes in clusters replenished in the second stage for an experiment with a dataset of 3755 classes and 800 clusters. The sizes of these replenished clusters range from 0 to 44 with a mean value of 10.6 and a standard deviation of about 6.

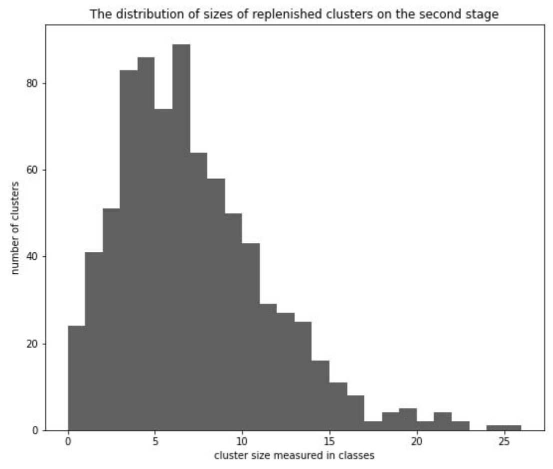

Fig. 3. The distribution of sizes of replenished clusters on the second stage

Practice shows that the number of cluster affects the quality rate in that way: the more cluster number we choose, the better result we get, but at the same time, the more time we spend on second-level classifier training. Therefore, one should choose values according to its priorities: better speed or accuracy.

V. Results

We measured the accuracy separately on each model, therefore, we considered test samples that were assigned to a particular cluster, that included one or more classes. These accuracy results could vary greatly depending on the number of classes in the cluster. Therefore, we needed to average them across all clusters. After we obtained the accuracy of each model, the number of which is equal to the number of clusters, we calculated the final accuracy using the following formula:

$$Acc_{res} = \frac{\sum_{i=1}^{M} Acc[i] * num[i]}{\sum_{i=1}^{M} num[i]}$$

where:
Acc_{res} - resulting accuracy on whole test dataset,
M - number of clusters,
$Acc[i]$ - accuracy on cluster i,
$num[i]$ - amount of classes in cluster i.
That is, we take the weighted sum of the accuracy of all second-level classifiers, where their size acts as weights. Thus, we contribute to the accuracy of a separate classifier trained on some set of classes to the resulting accuracy, proportional to the size of this set, that is, the number of classes included in it. This is an important thing to do because these sets can be very uneven in size.
For CASIA-HWDB dataset and the small dataset based on it the results of measurements of accuracy for both experiments with different cluster numbers and hyperparameter $T_{acc} = 1$ can be seen in table 2.

TABLE II
Comparison table between traditional one-stage deep CNN and two-stage hierarchical CNN for CASIA-HWDB: the small dataset with 200 target classes and the original dataset with 3755 target classes

	Traditional one-stage CNN	Hierarchical CNN
200-classes model accuracy for 20 clusters in percent	96.40	**97.33**
200-classes model accuracy for 40 clusters in percent	96.40	**97.64**
200-classes model accuracy for 80 clusters in percent	96.40	**97.77**
3755-classes model accuracy for 400 clusters in percent	93.36	**94.11**
3755-classes model accuracy for 800 clusters in percent	93.36	**94.62**

TABLE III
Comparison table for hierarchical CNNs for dataset with 200 target classes and 40 clusters with different T_{acc} values

T_{acc}	Network accuracy
1	**97.64**
0.75	**98.85**
0.66	**98.39**
0.5	**98.76**

As we can see from table 2, accuracy is directly proportional to the number of clusters into which we split the set of classes in the dataset. This conclusion may seem obvious, but it should be noted that with an increase in the number of clusters, the number of classes contained in the coverage of the cluster and the number of elements in the dataset for the corresponding second-level classifier can vary at different rates.
However, with the hyperparameter $T_{acc} = 1$ we do not always

obtain the highest values of accuracy, which can be seen from table 3. In this table, one can observe the results of the experiments with the dataset containing 200 target classes divided into 40 clusters and with different hyperparameter T_{acc} values.

For the Competition dataset we also compared the quality results of conventional classifier and hierarchical model with 800 clusters and hyperparameter $T_{acc} = 1$. This comparison is shown in table 4. On this dataset we also got significant improvement in quality for our new approach. We also add results from previous works for the reference. These works use more complex network architectures than ours, which explains the gap in quality.

TABLE IV
COMPARISON TABLE FOR TRADITIONAL AND HIERARCHICAL CNN APPROACH ON ICDAR 2013 COMPETITION DATASET (INCLUDING PREVIOUS WORKS)

Traditional one-stage CNN	87.85
Hierarchical CNN	**92.10**
ICDAR-2013 Winner [10]	94.77
DenseRAN [8]	96.66
STN-Residual-34 [12]	97.37

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a model of two-stage hierarchical deep CNN that can cope with a large number of classes better than conventional deep classification CNNs.

It is worth noticing that the architecture of the two-stage hierarchical CNN can be varied widely and give different results for this particular problem and other ones, similar to ours. This approach is well-suited for any recognition task with a large number of classes.

Since we got a successful result of comparison between conventional and hierarchical CNN obtaining a significantly higher accuracy indicator for the hierarchical CNN, we can apply this approach in the future. In our experiments, the architecture of the neural network was quite simple, so the overall accuracy is not very high compared to other results achieved in this area. But in future, we plan to take the architecture that is more advanced and suitable for this task and apply it in this method, so that we would be able to achieve much more impressive results. We also believe that proposed architectures can be further optimized to obtain more efficient model in terms of speed.

We also plan to work on the structure of the first-level classifier in order to minimize the number of errors at the first stage. That can also significantly increase the accuracy of the classification in general.

REFERENCES

[1] Linhui Chen, Liangrui Peng, Gang Yao, Changsong Liu and Xudong Zhang. “A Modified Inception-ResNet Network with Discriminant Weighting Loss for Handwritten Chinese Character Recognition”, in Proc. ICDAR 2019, pp. 1220–1225

[2] Cong Kha Nguyen, Cuong Tuan Nguyen, Nakagawa Masaki. “Tens of Thousands of Nom Character Recognition by Deep Convolution Neural Networks”, in Proc. ICDARW 2017 (HIP), pp.37–41

[3] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, Yizhou Yu. “HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition”, in Proc. ICCV 2015, pp.2740–2748

[4] Qingqing Wang, Yue Lu. “Similar Handwritten Chinese Character Recognition Using Hierarchical CNN Model”, in Proc. ICDAR 2017 pp.603–608

[5] Yuhao Zhang. “Deep Convolutional Network for Handwritten Chinese Character Recognition”, 2015

[6] Junyi Zou. “CASIA Handwritten Chinese Character Recognition Using Convolutional Neural Network and Similarity Ranking (with TensorFlow Source Code)”, Nov 19, 2018

[7] Sheng He, Lambert Schomaker. “Open Set Chinese Character Recognition using Multi-typed Attributes”, arXiv preprint 1808.08993

[8] Wenchao Wang, Jianshu Zhang, Jun Du, Zi-Rui Wang and Yixing Zhu “DenseRAN for Offline Handwritten Chinese Character Recognition”, in Proc. ICFHR 2018, pp. 104–109

[9] Cheng-Lin Liu, Fei Yin, Da-Han Wang, Qiu-Feng Wang. “CASIA Online and Offline Chinese Handwriting Databases”, in Proc. ICDAR 2011, pp. 37–41

[10] Fei Yin ; Qiu-Feng Wang ; Xu-Yao Zhang ; Cheng-Lin Liu. “ICDAR 2013 Chinese Handwriting Recognition Competition”, in Proc. ICDAR 2013, pp. 1464–1470

[11] Diederik P. Kingma, Jimmy Ba “Adam: A Method for Stochastic Optimization”, in Proc. ICLR 2015

[12] Zhao Zhong, Xu-Yao Zhang, Fei Yin, Cheng-Lin Liu ”Handwritten Chinese Character Recognition with Spatial Transformer and Deep Residual Networks”, 23rd International Conference on Pattern Recognition (ICPR) 2016, pp. 3429-3434