The Algorithm of Handwritten Character Recognition Based on Autoencoder Network and KNN Classifier

Yixing Chen, Kang Yang, Mingquan Zhou, Yue Zhuo

***Abstract*—In this article, a method for addressing handwritten character recognition is developed. This method is made up of Autoencoder Network and KNN classifier. Autoencoder Network is used to compress the redundant information and extract features of character pictures. KNN classifier is used to recognize written characters based on features extracted from the latent layer of Autoencoder Network. Based on the results of the experiment part in this article, the average correctness of recognizing 'a' and 'b' is up to 97.5% and the average correctness of recognizing 'hard' dataset is up to 92.5%. This article also compares the results of Autoencoder and KNN Classifier with those from Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP), which demonstrates the better performance of the proposed method.**

***Index Terms*— Autoencoder Network, KNN, Handwritten Character Recognition**

I. Introduction

Handwritten character recognition is a promising research area in pattern recognition. This technology has a variety of applications in the real world, such as handwritten inputting, security system, office automation and human-machine interaction [1]. Thus, many kinds of methods of developed to realize handwritten character recognition.

Recently, applying a neural network to solve handwritten character recognition is a hotspot. Vijay Patil[2] applied MLP network trained by the error back-propagation algorithm directly to realize a more than 70%. Binu[3] solved this problem by extracting features from wavelet energy and classifying characters through extreme learning machine. However, one of the most common frameworks is applying Convolutional Neural Networks (CNN) to process this problem, since CNN can realize information compression and feature extraction by convolution and pooling operation. Simard[4] reported an error rate of 0.4% on the MNIST handwritten character recognition dataset, using a fairly simple CNN. Ciresan[5] focused on applying deep convolutional neural networks (DNN) to process a huge of the dataset and got a good performance.

However, the parameter setting of CNN is not an easy job. The whole conjunction operation of CNN and classification learning based on MLP is easy to make CNN be trapped into overfitting when the training dataset is not large. Thus, a simpler classification framework needs to be introduced when the training data is not large. Wang[6] introduce a simple method to compress the redundant information of character pictures and thus to reduce the possibility of overfitting, which can be considered as a reasonable method to process recognition of handwritten characters when the training dataset is not large.

The whole training dataset in this project has less than 8000 images, which is not a large training dataset. Applying MLP or CNN to solve this problem is not easy to set suitable parameters and has a high possibility of overfitting when increasing layers and neurons to get a higher accuracy for training dataset.

Thus, in order to decrease the possibilities of overfitting, find suitable parameters easily, and increase the predicting accuracy, this article uses Autoencoder Network to compress the redundant information of character image and applies KNN classify to handwritten characters. This article first briefly introduces the Autoencoder Network and KNN classifier in Section 2 Implementation. In the next section, different training dataset and test dataset will be used to test the performance of the proposed method, CNN and MLP. Finally, some conclusions are drawn in Section 4.

II. Implementation

Handwritten character images in the provided dataset have many kinds of sizes. However, the input of Autoencoder Network is constant. Thus, these images need to be preprocessed before be training in Autoencoder Network. Preprocessing contains two steps – reshape images into images with the same size, and convert images with the same size into one-dimension vectors.

After preprocessing dataset, all images are put into Autoencoder Network to train parameters in this network. The objective function is to minimize the difference between input and output. In other words, training network is to make the output of this network is as possible as similar to the input of the network. In this case, the parameters in each layer will represent the information of input well.

Thus, the latent layer can be considered as features to train KNN classifier. The whole implementation framework of handwritten character recognition is shown in Fig 1.

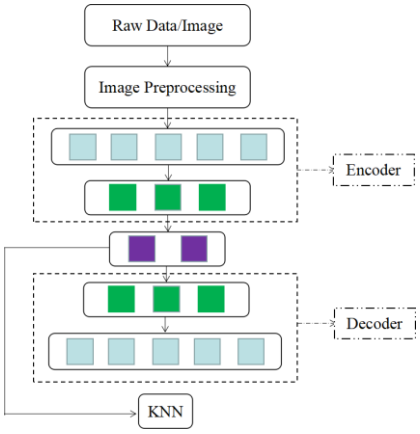

Fig 1. The Flow Chart of the Recognition Framework

A. *Image Preprocess*

1) Reshape Images

The input size of Autoencoder Network is constant. Thus, all images in provided dataset should be shaped into the same size. In this article, all images are reshaped into 32×32 pixels by PIL library.

2) Convert 2-D Images into 1-D Vectors

After reshaping operations, all images are shaped into 32 × 32 pixels. However, the input of Autoencoder Network should be a 1-D vector. Thus, the 2-D image data are converted into a 1-D vector with a length of 1024. This process is shown in Fig. 2.

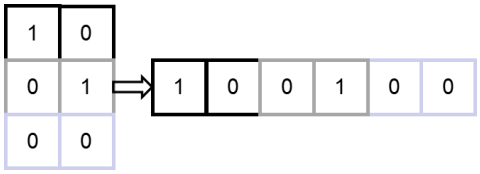

Fig 2. The transformation from a matrix to a vector

B. *Autoencoder Network*

Autoencoder Network is a kind of unsupervised learning method. The objective function is to minimize the difference between input and output. In other words, training network is to make the output of this network is as possible as similar to the input of the network. Thus, parameters in each layer will represent the information of input data well.

Autoencoder Network is made up with two parts -- encoder and decoder. In the encoder network, the number of neurons in each layer decreases, which means the input layer has the largest neuron number and the last latent layer has the smallest neuron number in the encoder network. In the decoder network, the number of neurons in each layer increases, which means the latent layer has the smallest neuron numbers and the output layer has the largest neuron numbers. Thus, as the data is processed in the encoder network, the information of input data is compressed and as the data is processed in the encoder network, the information of input data is restored. Normally, the number of neurons in the input layer is the same with the number in the output layer. The whole structure of Autoencoder Network is shown in Fig 3.

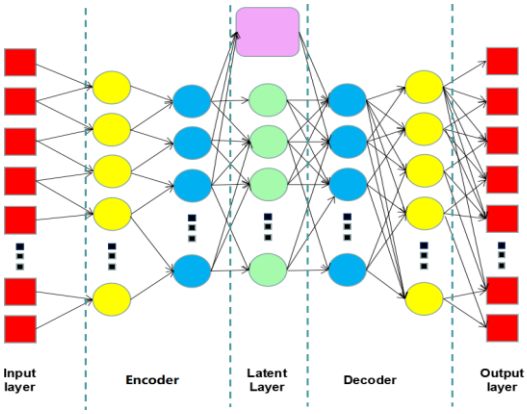

Fig 3. The structure diagram of autoencoder

The detailed work mechanism is introduced hereby. For the encoder, the tanh function is used to calculate the hidden layer representation $\boldsymbol{y}$ after the linear transformation of the input layer with the weight $\boldsymbol{W_e}$ and bias vector $\boldsymbol{b_e}$. This process is illustrated as:

$$y = \sigma(W_e x + b_e) \tag{1}$$

where σ is the tanh activation function.

For the decoder, the computation process is similar but adverse. The tanh function is used to calculate the output layer representation $\boldsymbol{z}$ after the linear transformation of the hidden layer with the weight $\boldsymbol{W_d}$ and bias vector $\boldsymbol{b_d}$.

$$z = \sigma(W_d y + b_d) \tag{2}$$

where σ is the tanh activation function.

The loss function is defined with squared error:

$$L = \sum_{i=1}^{n} (z_i - x_i)^2 \tag{3}$$

where L is the reconstruction loss, z_i is the ith node of the output layer, x_i is the ith node of the input layer.

C. *KNN Classifier*

After extracting features by Autoencoder Network, the feature obtained would be used to classify characters. Due to the high performance of KNN in multimodal classification and the property of not being overfitted, a classifier based on KNN algorithm is used in this project.

The core principle of the KNN algorithm can be described as that the sample belongs to a class if most of the K closest samples in the feature space belong to this class. Preprocessed data, labels and the value of K is used as input. The Euclidean distance between data and its neighbors can be expressed as

$$distance\left[\left((x,y),(a,b)\right)\right] = \sqrt{(x-a)^2} + \sqrt{(y-b)^2} \tag{4}$$

where (x, y) is the position of the train data and (a, b) is the position of the closest neighbor. Then the labels of the K nearest neighbors will be obtained and the majority vote will be performed to return the class with the highest occurrence frequency in the first K neighbors as the predicted label of test data. The pseudo-code of the KNN algorithm is shown in table 1.

K − Nearest Neighbor Classifier$(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{x})$:

$\backslash\backslash\ \boldsymbol{X}$: *train data*; $\boldsymbol{Y}$: *train label*; $\boldsymbol{x}$: *neighbors*

for $i = 1$ ***to*** n ***do***:

define the Euclidean distance $d(\boldsymbol{X_i}, x)$

end for

Compute set I containing indices for the K smallest distances $(\boldsymbol{X_i}, x)$

return *the majority label for* $\boldsymbol{Y_i}$

Table 1: Pseudo-code of KNN algorithm

D. Methods of Identifying Unknown Characters

Fig 4 shows the decision boundary trained by dataset provided by the project. The three kinds of points represent sample data in train dataset. The two curves represent the decision boundary. Fig? shows the decision boundary trained by the dataset where additional data (unknown characters) are added. The 'fork', 'square', and 'rhombus' points come from the dataset provided by the project. The black 'circle' points represent unknown characters in the dataset provided by the project.

The two figures show that for a well-trained model, the decision boundary will change when the training dataset is changed.

If we just train the KNN classifier model based on the dataset provided by the project, the decision boundary may be loose. In this case, some unknown characters will be misjudged into known characters.

However, if we add more characters to train dataset, especially add more characters, which are unknown in dataset provided by the project. The trained decision boundary will become tighter. In this case, these characters unknown in dataset provided by the project have fewer possibilities of being misjudged into known characters.

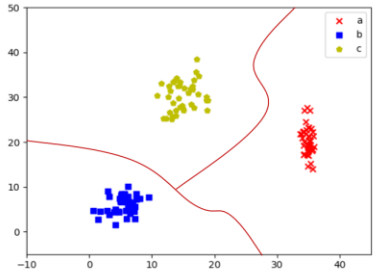

Fig 4. data classification without unknown data

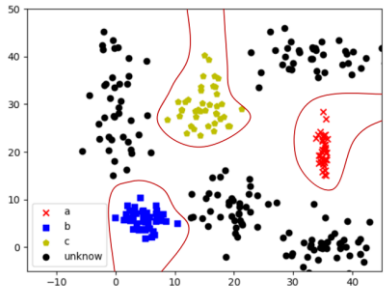

Fig 5. data classification with unknown data

Thus, in order to increase the accuracy of predicting known characters and unknown characters. We search more character dataset and combine them with the data provided by the project to train our model.

The added dataset contains all handwritten characters other than 'a', 'b', 'c', 'd', 'h', 'i', 'j', 'k' characters. Each character has 55 images. Thus, we totally add 55× 46 images into original dataset. Labels for these unknown characters are set to -1. The added handwritten character image dataset is downloaded from http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/.

III. EXPERIMENTS

The results in this section come from two datasets. One dataset is provided by the project, which contains 'a', 'b', 'c', 'd', 'h', 'i', 'j', 'k' characters and has totally 7847 images. Another dataset combines the dataset provided by the project and the dataset which contains 2530 unknown character images. This dataset totally contains 10377 handwritten character images.

These goals of these experiments are to demonstrate the good performance of the proposed method (Autoencoder Network + KNN classifier)

These performances include precision, stability and the generalization of the method.

In order to show the effectiveness of the proposed method, the best results obtained from CNN and MLP are provided to be compared.

The two datasets are separated into train dataset and test dataset, which has a ratio of 8 to 2. 0.8 times of dataset are used to train model, and the rest of dataset is to test the effectiveness of the model.

A. The precision of Autoencoder Network

The method to test the precision of Autoencoder Network is to compare the output result with the input data. The smaller is the difference, the better is the precision of Autoencoder Network.

The Fig 6 shows the precision of Autoencoder Network, the first images of 'a' and 'b' are the original images of 'a' and 'b'.The following 5 images of 'a' and 'b' are the reconstructed images with the number of neurons in the latent layer increasing from 5 to 50. From these images, we can find that as the number of neurons in the latent layer increases, the more precise images are constructed. However, when the number of neurons in the latent layer increase from 30 to 50, the precision of the image does not have obvious improvement.

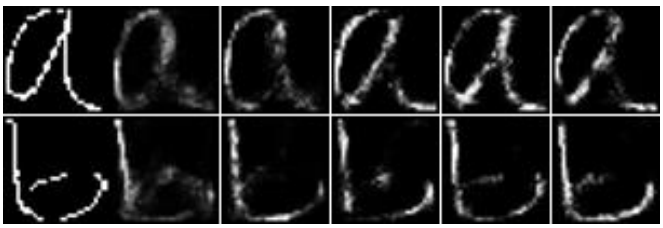

Fig. 6. The original images of 'a' and 'b' and the reconstructed images of 'a' and 'b' when the latent layer with 5 nodes, 10 nodes, 20 nodes, 30nodes, 50 nodes

In this experiment, the training dataset with a number of 6277 ‘a’, ‘b’, ‘c’, ‘d’, ‘h’, ‘i’, ‘j’, ’k’ images come from the data provided by the project are used to train Autoencoder Network. Then, the parameters in the latent layer are applied to train KNN classifier.

Using parameters in the latent layer of the rest 1569 ‘a’, ‘b’, ‘c’, ‘d’, ‘h’, ‘i’, ‘j’, ’k’ images to test the precision of the classification of KNN. The precision is affected by the size of the latent layer of Autoencoder Network. The results are shown in TABLE I.

TABLE I
TABLE OF PRECISION WITH DIFFERENT NODE NUMBER IN LATENT LAYER

Node number	5	10	20	30	50
Loss	0.0625	0.0488	0.0472	0.0443	0.0449
precision	82.67%	90.67%	92.00%	93.67%	93.33%

From the results of TABLE 1, we could find if the number of nodes in the latent layer increases from 5 to 30, the precision increases as well. However, the precision reaches the climax when the number of nodes is around 30. If the number still increases from 30, the precision does not apparently increase.

B. Stability and Precision of Autoencoder + KNN Method

Models in TABLE II. are trained by dataset provided by the project, which contains ‘a’, ‘b’, ‘c’, ‘d’, ‘h’, ‘i’, ‘j’, ’k’ characters and has totally 6277 images. The three models are tested by dataset (easy dataset) provided by the project, which contains ‘a’, ‘b’ characters and has totally 392 images.

TABLE II
PRECISION TABLE OF “EASY” DATA SET

times	1	2	3	4	5
Autoencoder	96.05%	93.79%	95.48%	96.04%	96.61%
CNN	97.02%	96.32%	98.46%	97.28%	98.54%
MLP	96.32%	97.67%	96.42%	96.32%	97.47%

Models in TABLE III. are trained by dataset provided by the project, which contains ‘a’, ‘b’, ‘c’, ‘d’, ‘h’, ‘i’, ‘j’, ’k’ characters and has totally 6277 images. The three models are tested by dataset provided by the project, which contains a’, ‘b’, ‘c’, ‘d’, ‘h’, ‘i’, ‘j’, ’k’ characters and has totally 1592 images.

TABLE III
PRECISION TABLE OF DATA SET WITH 8 TARGET CHARACTERS

times	1	2	3	4	5
autoencoder	93.67%	93.17%	93.67%	94.33%	93.83%
CNN	89.17%	88.13%	88.45%	88.98%	90.03%
MLP	86.24%	87.36%	85.19%	86.63%	86.32%

Models in TABLE IV. are trained by dataset provided by the project, which contains ‘a’, ‘b’, ‘c’, ‘d’, ‘h’, ‘i’, ‘j’, ’k’ characters and has totally 8301 images. The three models are tested by dataset (‘hard dataset’) provided by the project, which contains a’, ‘b’, ‘c’, ‘d’, ‘h’, ‘i’, ‘j’, ’k’ characters and has totally 2077 images.

TABLE IV
PRECISION TABLE OF DATA SET WITH “HARD” DATA SET

times	1	2	3	4	5
autoencoder	96.67%	95.83%	96.33%	96.17%	96.50%
CNN	86.43%	86.72%	88.59%	90.06%	86.43%
MLP	86.74%	86.92%	86.74%	86.98%	87.23%

We could conclude from the results above that the proposed method (Autoencoder+KNN) has a better precision and stability.

IV. CONCLUSIONS

In the project reported in this paper, a handwritten character recognition based on Autoencoder and KNN is proposed. By comparing CNN and MLP, the proposed method has a better precision and generalization ability.

When adjusting the parameters of CNN and MLP, in order to get a better training result, we need to set a relatively large network to fit the training data. For CNN, the network uses 400×100×9 neurons to fit data. For MLP, the network applies 1024×1024×9 to fit data. The training dataset is relatively smaller to the parameters, which may lead to the worse generalization ability of CNN and MLP.

When classifying high dimensional data with a small dataset, it’s better to figure out a method to reduce the redundancy of features and not use a complex model to fit the data. This will help us obtain a model with a more generalized ability.

V. REFERENCES

[1] M. Wang, Y. Chen and X. Wang, "Recognition of Handwritten Characters in Chinese Legal Amounts by Stacked Autoencoders," 2014 22nd International Conference on Pattern Recognition, Stockholm, 2014, pp. 3002-3007. doi: 10.1109/ICPR.2014.518

[2] *Patil V, Shimpi S. Handwritten English character recognition using neural network[J]. Elixir Comput Sci Eng, 2011, 41: 5587-5591.*

[3] Chacko B P, Krishnan V R V, Raju G, et al. Handwritten character recognition using wavelet energy and extreme learning machine[J]. International Journal of Machine Learning and Cybernetics, 2012, 3(2): 149-161.

[4] Simard P Y, Steinkraus D, Platt J C. Best practices for convolutional neural networks applied to visual document analysis[C]//null. IEEE, 2003: 958.

[5] Cireşan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification[J]. arXiv preprint arXiv:1202.2745, 2012.

[6] Wang M, Chen Y, Wang X. Recognition of handwritten characters in chinese legal amounts by stacked autoencoders[C]//Pattern Recognition (ICPR), 2014 22nd International Conference on. IEEE, 2014: 3002-300.