Classify Handwritten Digit Images by Convolutional Neural Network

Sifan Xu
a1816684@adelaide.edu.au

Abstract

In recent years, deep learning has developed rapidly in the computer field. Due to Convolutional Neural Networks(CNN), great success has been achieved in image recognition, object recognition and many other aspects. This report aims at learning CNN and using CNN to design an image classification to correctly identify numbers based on MNIST data. This CNN model consists of 2 convolutional layers with max-pooling layer and 2 fully-connected (Dense) layers. The accuracy of the model on the validation set is approximately 99%.

1. Introduction

With the development of deep learning, artificial neural network(ANN). Neural networks are used to solve problems like computer vision and speech recognition, which are difficult to solve with traditional programming. However, using full connection neural network to process large size images has the following drawbacks. First, expanding the image into vectors will lose spatial information. Second, too many parameters will make it difficult to train the model. Third, a large number of parameters will lead to network overfitting. Therefore, Convolutional Neural Network was designed to handle the problems. LeCun et al. (1998) proposed Lenet-5 which is one of the earliest pre-trained model architecture to recognize the handwritten and machine-printed characters. [4]

In this report, a simple CNN model with 2 convolutional layers and 2 fully-connected layers is built to verify the handwritten numeral images.

2. Data

The data used in this project were provided in https://www.kaggle.com/competitions/digit-recognizer/data. There are 3 files in the website which are “train.csv”, “test.csv”, “sample_submission.csv”. The “train.csv” and “test.csv” files contain data of grayscale images from 0 to 9. Table 1 shows parts of “train.csv” file. It could be found that there are 42000 lines and 785 columns in “train.csv”. Each line in the table represents an image. The image is 28 pixels in height and 28 pixels in width, and each pixel has a pixel value associated with it, indicating the brightness or darkness of the pixel. The larger the number, the darker it will be. Therefore, the first column shows the digit drawn by the user and the rest 784 columns are the pixel-values of the associated image.

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7
0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
41995	0	0	0	0	0	0	0	0	0
41996	1	0	0	0	0	0	0	0	0
41997	7	0	0	0	0	0	0	0	0
41998	6	0	0	0	0	0	0	0	0
41999	9	0	0	0	0	0	0	0	0

42000 rows × 785 columns

Table 1: part of data in train.csv

Figure 1 shows the number of the labels in “train.csv”, the “1” label has the largest number and “5” label has the smallest number. However, the number of these 10 figures is not much different, and the distribution is relatively balanced in general.

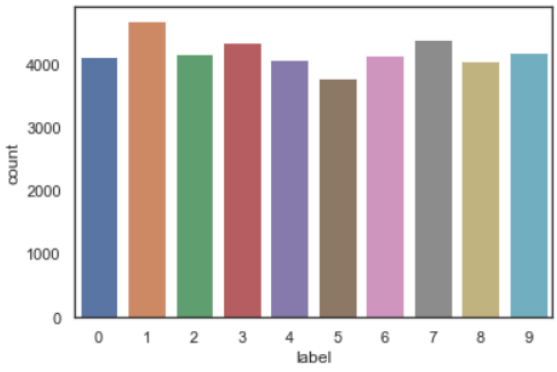

Figure 1: part of data in train.csv

The “test.csv” is similar to “train.csv” while it lacks the label column. Therefore, in this project the file will not be used and the “train.csv” will be divided into train set and validation set. 80% of the “train.csv” will be used to train the model and the rest will be used to test the accuracy of

the model and check if the model is overfitting.

3. Method

In this project, Convolutional Neural Network (CNN) is used to classify the digit pictures. O'Shea and Nash (2015) made an introduction to the CNN that convolution neural network is a special deep neural network model. [2] Its particularity is reflected in two aspects: on the one hand, the connection between its neurons is not fully connected, and on the other hand, the weight of the connection between some neurons in the same layer is shared. This reduces the complexity of the network model and the number of weights.

The overall architecture is shown in Figure 2. The CNN is mainly composed of the following layers: input layer, convolutional layer, ReLU layer, pooling layer and full connection layer. The input layer usually loads the pixel value of the image and the data is processed by other layers and finally output as the predicted value.

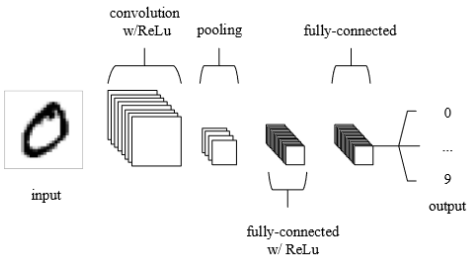

Figure 2: overall architecture of CNN (https://arxiv.org/pdf/1511.08458.pdf)

3.1. Convolutional layer

Convolutional layer is the core layer of constructing convolutional neural network, which generates most of the computation in the network. The convolutional layer will determine the output of neurons connected to the local area of the input by calculating the scalar product between its weight and the area connected to the input volume. The function of the convolutional layer is shown in Figure 3, a 8 ×8 size data becomes a 6×6 data after being processed by a 3×3 filter.

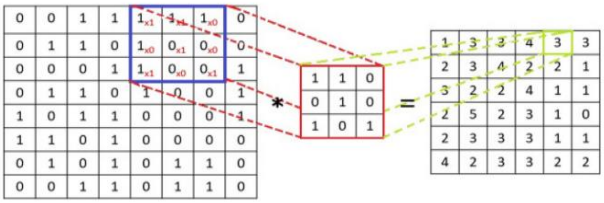

Figure 3: convolutional layer (https://www.sciencedirect.com/topics/engineering/convolutional-layer)

3.1.1 Filter

The filter is the core of the convolutional layer, and the image data is reduced in dimension after being processed by the filter. The filters are usually odd, as if filter is even, it usually needs some asymmetric padding. Besides, an odd filter has a central position which is necessary in processing the image data. The depth of the filter is the same as that of the pictures. Color pictures have a depth of 3 as they have R, G, B channels and gray image have a depth of 1.

When the filter parameters are different, the extracted feature values are also different. There can be multiple filters like horizontal and vertical. The height and width of each layer in the filter can be also different.

3.1.2 Stride

The filter slides over the width and height of the input, and calculates the dot product between the input and the filter at each spatial location. The F×F filter moves left and right and up and down with stride S.

3.1.3 Padding

A N×N image data will become a smaller data with ((N-F)/S+1) ×((N-F)/S+1) size. Therefore, it is convenient to fill the input data body with 0 at the edge, which allows the filtered data to be the same size as the original data, and the edge data of the original data can be calculated multiple times when extracting features. If zero filling is not carried out, the size of the data volume will be reduced, and the information on the image edge will be lost too quickly. After padding, the size of the filtered data is ((N+2P-F)/S+1) ×((N+2P-F)/S+1). If it cannot be divided, it will be rounded down. The filter must be within the range of original image or original image plus padding.

3.1.4 Function of Convolutional layer

The purpose of convolution operation is to extract different features of input, and the function of the convolutional layer is to determine which useful convolution filters can preliminarily characterize the image features by constantly changing the convolution kernels, and then obtain the output matrix multiplied by the corresponding convolution filters. In another word, the main function of the convolutional layer is to retain the characteristics of the picture, the feature of "incomplete connection and parameter sharing" greatly reduces the network parameters, ensures the sparsity of the network and prevents over fitting. [10]

3.2. ReLU layer

Rectified Linear Units layer (ReLU layer) uses Rectified Linear Units as the activation function of this layer of nerves.

$$f(x) = \max(0, x)$$

It can enhance the nonlinear characteristics of the decision function and the whole neural network without changing the convolution layer itself. The reasons of using Relu are as follows. First, using the Relu activation function can save computation. If sigmoid or other

functions are used, the activation function is calculated exponentially, which requires a large amount of calculation. When backpropagation is used to calculate the error gradient, the derivation involves division, which also requires a large amount of calculation. Second, when sigmoid function is back-propagated in the deep network, it is common that the gradient disappear happens, which makes it impossible to train the deep network. Third, Relu will make the output of some neurons zero, which will result in the sparsity of the network, reduce the interdependence of parameters, and ease the occurrence of over fitting.

3.3. Pooling layer

Pooling is another important concept in convolutional neural networks. In fact, it is a nonlinear form of downsampling. Its role is to gradually reduce the spatial size of the data volume. In this way, the number of parameters in the network can be reduced, which reduces the cost of computing resources, and can also effectively control over fitting. A pooling layer is usually inserted periodically between successive convolution layers. There are many different kinds of pooling functions, among which Max pooling shown in Figure 4 is the most common. It divides the input image into several rectangular regions, and outputs the maximum value for each sub region.

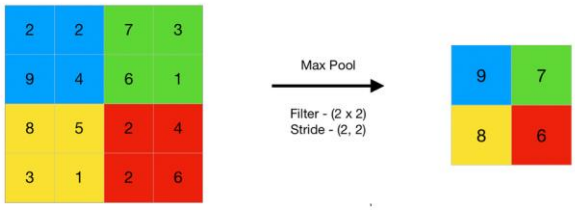

Figure 4: max pooling (https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/)

In addition to max pooling, pooling layer can also use other pooling functions, such as "average pooling". In the past, average pooling was used widely, but max pooling has replaced it due to the better performance. Graham (2014) conveyed that as the pooled layer reduces the size of data too quickly, the trend in the current literature is to use smaller pooled filters, or even no longer use the pooled layer. [6]

3.4. Fully-connected (Dense) layer

The fully-connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. [2] The fully-connected layer plays the role of "classifier" in the whole convolutional neural network, it gets the final output through the softmax function.

$$S_i = \frac{e^i}{\sum_j e^j}$$

The softmax function is used in multi classification process, it maps the output of multiple neurons to the interval (0,1) and the sum of the outputs equals to 1 which can be understood as probability.

3.5. CNN architecture

With the development of deep learning, more and more CNN architectures have been proposed. The following are some typical architectures.

3.5.1 LeNet-5

The first successful convolutional neural network application of LeNet-5 was implemented by Yann LeCun in the 1990s. It has 2 convolutional layers, average-pooling layer and 3 fully-connected layers. This network architecture has about 60000 parameters.

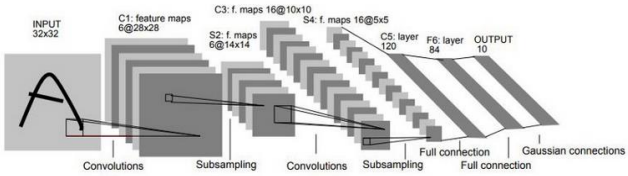

Figure 5: LeNet-5 (https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/)

3.5.2 VGG-16

VCG-16 architecture has 13 convolutional layers and 3 fully-connected layers, which are composed of 138 M variables. It shows that depth is a key part to make the algorithm perform excellently. [7]

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: VCG-16(https://arxiv.org/pdf/1409.1556.pdf)

3.5.3 *Inception-v1*

The 22 layer network architecture has 5M parameters, and the network is constructed by tight modules. The method of stacking convolution layers is not adopted, but the method of stacking modules composed of convolution layers. Each module consists different kinds of filters. [8]

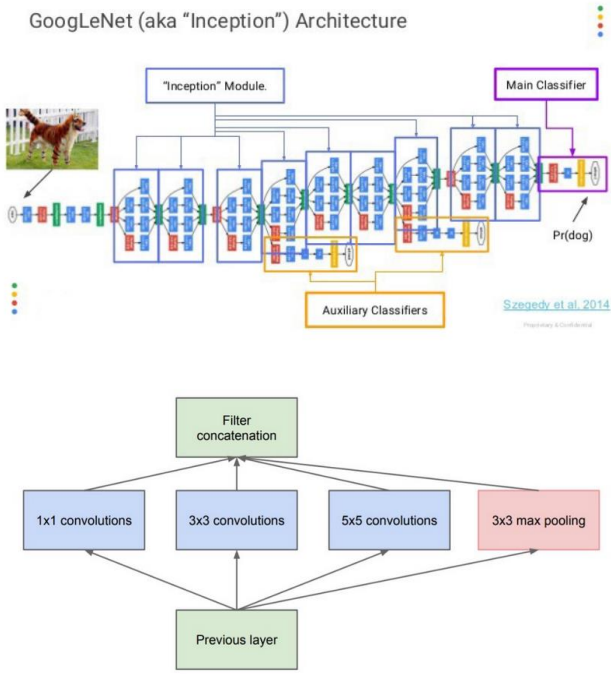

Figure 6: Inception-v1 architecture and Inception Module (https://medium.com/@abheerchrome/inception-v1-architecture-explained-454b2eb66baf)

3.5.4 *ResNet-50*

In the above several CNNs, better performance is achieved by increasing the number of layers. However, with the increasing depth of the network, the accuracy has reached saturation. ResNet no longer adopts the full connection mode. It uses a special skip link and uses a lot of batch normalization. [9]

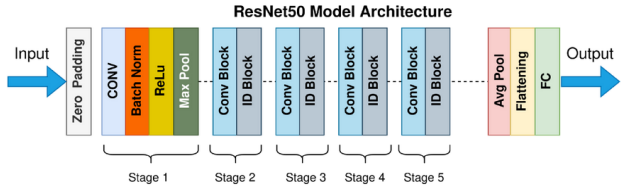

Figure 7 : ResNet-50 (https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758)

4. Experiment

This project aims at learning CNN and building a CNN model to classify the number corresponding to each image, and a predicted label value will be obtained when the image data is processed by CNN model. There are 10 numbers from 0 to 9 in the labels, and each image represents a number.

4.1.1 *Load and preprocess data*

The "train.csv" and "test.csv" files are loaded to obtain the data set. The data in "train.csv" have 42000 lines and 785 columns, each line represents a picture with 28 pixels in height and 28 pixels in width. The first column is the label of the picture and the rest columns are the pixel value. The "test.csv" file has 28000 rows and 784 columns, the label is lacked when compared with "train.csv". Therefore, only the "train.csv" file will be used in this project.

As the training image data is stored in the data frame as a one-dimensional vector containing 784 values, the next step is to process the data to make its format meet the input requirements of the CNN model. The "label" column was extracted as the labels, and the rest columns were used as features. The feature data was reshaped as 28 in height, 28 in width and 1 in depth. Figure 8 shows the examples of the digits plotted by matplotlib.pyplot.

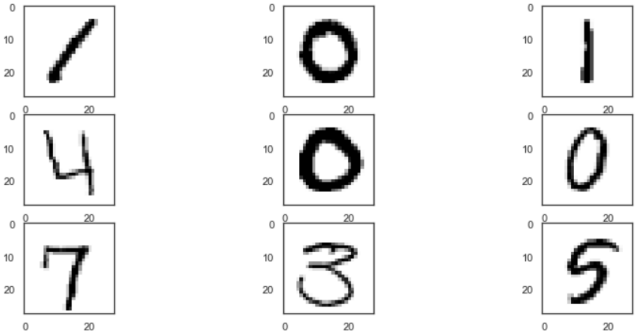

Figure 8: Examples of the digits

After the feature data have been processed, the label data should be also converted into one-hot vectors. As there are 10 kinds of labels in the data, the one-hot vectors have 10 in width, and the number of the label will make the vector value 1 at the same index and 0 elsewhere. For example, if the label is "1", the one-hot vector will be [0,1,0,0,0,0,0,0,0,0].

For reasons that the label distribution of the 42000 training images is relatively balanced, random splitting of the training set will not cause some labels to be over represented in the verification set. The training set is randomly divided into two parts: 20% is the verification set of the evaluation model, and the rest 80% is used for the training model.

4.1.2 *Set CNN model*

The Keras Sequential API was used when building the CNN model in this report, it could add one layer at a time. Table 3 shows the architecture of the CNN model in this project.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
dropout (Dropout)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0
dropout_1 (Dropout)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
dense (Dense)	(None, 512)	1606144
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 10)	5130

Total params: 1,630,090
Trainable params: 1,630,090
Non-trainable params: 0

Table 3: CNN architecture

The first layer is a convolutional layer, it has 32 convolution filters with the size of 3×3. The padding was set as "same" to make the image data remain the same size after filters. The filtered data introduces the linear computing system of the convolution layer into the nonlinear characteristics through the activation function ReLU. A 2×2 Maxpooling layer was used to make the original 28 × 28 pixel image reduced to 14 × 14 pixel image. A regularization method called "Dropout" was used to prevent overfitting, it randomly discard some nodes in the layer for each training sample.

The second layer is also a convolutional layer, it has 64 convolution filters with the size of 3 × 3 and ReLU activation function. A 2×2 Maxpooling layer was used to make the 14 × 14 pixel image reduced to 7 × 7 pixel image.

Next, a flatten layer was added to convert the final feature map into a one-dimensional vector. The flatten layer is need as it could add fully-connected layers after some convolutional/maxpool layers.

Finally, 2 fully-connected layers were added in the model. The first fully-connected layer contains 512 nodes and data is transferred to the next layer through ReLU activation function. The last fully-connected layer uses the Softmax activation function to output 10 nodes which is the probability distribution of 10 categories of digital labels.

4.1.3 Train the model

After the model is defined, the optimizer, loss function, evaluation function and learning rate annealer need to be set. The optimizer can iteratively improve parameters such as filter value and weight to minimize losses. RMSprop is used to reduce its radical and monotonically decreasing learning rate by weighted averaging the cumulative gradient. The loss function is used to measure how poorly the model performs on images with known tags, specifically the error rate between observed tags and predicted tags. In this project, "categorical_crossentropy" was used as it is a multi classification problem. Next, the "accuracy" was chose as evaluation function. Finally, ReduceLROnPlateau function in keras.callbacks was used to set learning rate annealer, it will automatically adjust the learning rate during training.

To avoid over fitting problems, ImageDataGenerator function was used to expand the data set. The training image is randomly rotated by 15 degrees, randomly scaled by 15%, randomly moved horizontally by 15% of its width, and vertically by 15% of its height. The vertical_flip and horizontal_ flip were not used, because flipping horizontally or vertically will result in incorrect classification of symmetric numbers such as 6 and 9.

Considering the time spent in training, the epochs were set as 20 when training the model.

4.1.4 Evaluate the model

Loss and accuracy curves of training set and validation set were plotted to evaluate the model. Figure 9 show the curves, the x axis is the number of the epoch and y axis is the value of loss and accuracy.

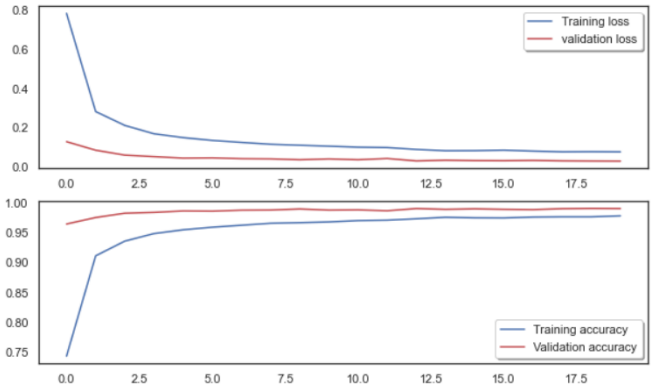

Figure 9: Loss and accuracy curves

According to the loss curve, the loss of the train set decreases rapidly before 2 epochs. The loss of the validation set is very small and is always smaller than the training set. According to accuracy curve, the training accuracy increases rapidly before 2 epochs, the validation accuracy is always very high and stables at approximately 99% after 13 epochs. Therefore, the model does not over fit the training set and has high accuracy in correctly classifying digital images.

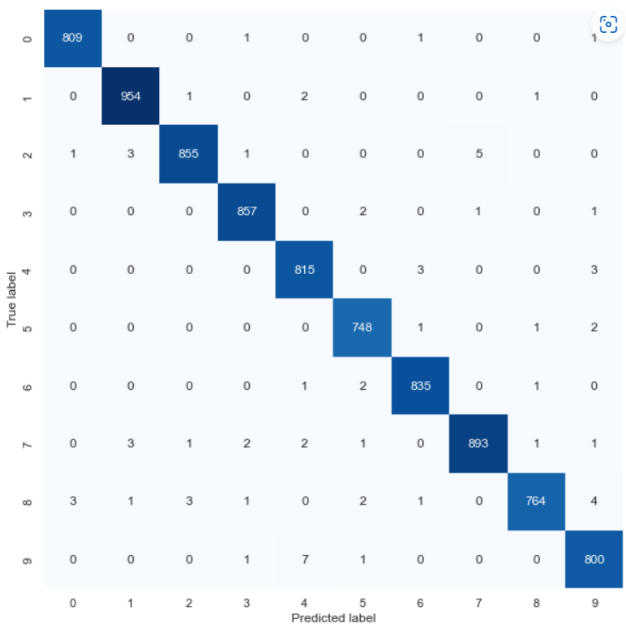

Figure 10: confusion matrix

Figure 10 is a confusion matrix to evaluate the ability of the model in classifying each number, the x axis is the predicted label and y axis is the true label. According to the picture, it could be found that the model performs very well in all numbers while it may mistake "2" for "7", it may also mistake "9" for "4".

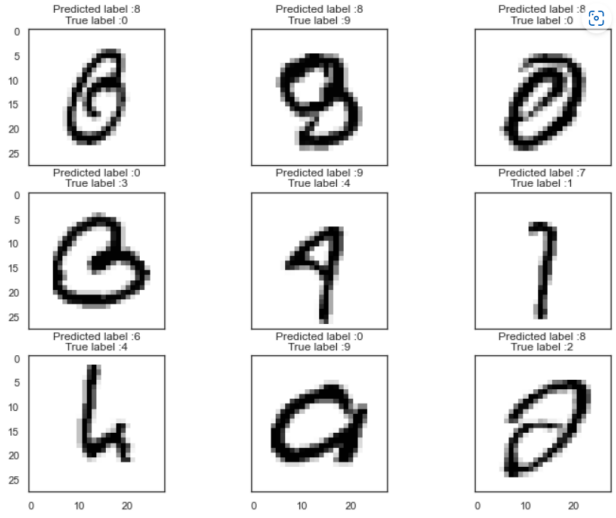

Figure 11: error examples

Some error predicted examples are shown in Figure 11. According to the examples, it is possible that the model made wrong predictions as some of these mistakes may also be man-made. Even for humans, it is difficult to recognize these numbers correctly.

In conclusion, the CNN model has a strong ability to recognize the handwritten digits and the accuracy of the model identification results is approximately 99%.

5. Code

The code could be found on github at https://github.com/a1816684/Deep-learning-assignment2/blob/main/assignment2.ipynb

6. Reflection

In this project, a simple CNN model was built for image classification. I have a preliminary understanding of CNN model such as the principle and architecture of it. CNN has the following advantages compared with full connection neural network when processing large size images like avoid losing spatial information, fast training speed, prevent overfitting.

MNIST data are chosen in this project as this handwritten image dataset is suitable for beginners to learn benchmarking classification algorithms. This project helps me understand and use CNN to handle image data for the first time. There are many other knowledges about CNN worth learning.

In the future, I plan to implement VCG-16 architecture to learn the influence of network depth on CNN. Besides, I plan to implement other CNN architectures to have a better understanding of it.

References

[1] Gu, J, Wang, Z, Kuen, J, Ma, L, Shahroudy, A, Shuai, B, … Chen, T 2018, 'Recent advances in convolutional neural networks', Pattern Recognition, vol. 77, pp. 354–377.

[2] O'Shea, K., & Nash, R. 2015. An introduction to convolutional neural networks. *arXiv preprint* .

[3] LeCun, Y., & Bengio, Y. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10).

[4] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[5] Dash, S, Acharya, BR, Mittal, M, Abraham, A & Kelemen, A 2020, Deep Learning Techniques for Biomedical and Health Informatics, S Dash, BR Acharya, M Mittal, A Abraham & A Kelemen (eds), 1st ed. 2020., Springer International Publishing, Cham.

[6] Graham, Benjamin. Fractional Max-Pooling. 2014-12-18.

[7] Simonyan, K & Zisserman, A 2014, 'Very Deep Convolutional Networks for Large-Scale Image Recognition'.

[8] Szegedy, C, Liu, W, Jia, Y, Sermanet, P, Reed, S, Anguelov, D, … Rabinovich, A 2014, 'Going Deeper with Convolutions'.

[9] He, K, Zhang, X, Ren, S & Sun, J 2016, 'Deep Residual Learning for Image Recognition', in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2016-, IEEE, pp. 770–778.

[10] Wu, C, Wang, L, Yang, Z, Wang, J, Han, L & Li, F 2022, 'Convolution Neural Network for Renal Function Assessment Based on Glomerular Filtration Rate', Journal of Physics: Conference Series, vol. 2185, no. 1, p. 12033–.