Year 2 Project

Handwritten Digit Recognition Implementation Using Convolutional Neural Network

---

By: Ziqi Sun (201522306), Xiaowei Shi (201522273), Wasif Ali (201361177).

Supervised by: Dr Angel Garcia-Fernandez

Department of Electrical Engineering and Electronics

19 April 2021

Abstract

In this report, the concept of neural networks will be investigated in order to create a handwritten digit recognition software. The purpose of the software is to allow a user to write a handwritten digit onto a screen, the commercial purpose would be a touch screen on a small smart watch, and have the software recognise the handwritten digit and convert into hexadecimal form on the device. The main purpose of this software is to allow the user to input data into their watch for specific activities such as searching the internet or sending a friend a message where a keyboard would otherwise be insufficient. The software will be built using a convolutional neural network (cnn) where labelled data is put through the cnn for a certain number of epochs and the cnn is then trained. The cnn has validation and test data put through it to ensure it is working correctly. Once it is done, the cnn will be connected to a Graphical User Interface (GUI) which will allow the user to input their data and have the result pop up on the screen in real time.

Declaration

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The main aim of this project is to create a software that recognises handwritten digits and converts them into hexadecimal computerised form. Handwritten digit recognition has a variety of uses in the modern world. In fact, the feature is so emended into modern systems it can sometimes be difficult to notice that it's there.

It will do this using a neural network trained for this specific purpose. The software works by taking a labelled dataset of handwritten digits and putting them through a Neural Network (NN), the NN, at this point, knows the number being drawn in each image. The system is them trained and then another set of data, that is unlabelled, is put through the neural network again for testing and validation. Neural Networks are essentially based off the behaviour of the human brain and how it learns. The idea is, as Hebb investigates in 1949, is that every time a neural pathway is used, it is strengthened. This is where the similarity lies between human and machine learning [1].

The use for Neural Networks themselves extends way beyond those that have been previously mentioned for digit recognition. Neural networks are used for all sorts of application ranging from digital receptionists to autonomous driving. The neural

network, however, in an autonomous vehicle may prove to be a lot more difficult since should anything go wrong with such application, lives can be at risk since in this application, the input would be a set of cameras, susceptible to obscurity, and the output would be the steering wheel and therefore the movement of the vehicle. [2]. The concept of neural networks will be explored in this report and the project will be coded in python using a range of platforms from Jupyter to PyCharm along with the use of external libraries such as TensorFlow and Keras.

1.2 Motivation

Devices that utilise this feature the most are that of smaller nature, devices that are not big enough to house keyboards on their screen and therefore require another method for digit input. This is where handwritten digit recognition proves to be imperative. In devices such as smart watches, this feature allows users to input their desired digits quickly and easily by simply writing it on the touchscreen of their device.

1.3 Objectives

The objective of this project is to create the Neural Network such that is has an appropriate accuracy of above at least 95%. It should do this where the NN is trained in a certain number of epochs to preserve computing power and time. It should also be connected to a user-friendly GUI where the user is able to draw their own digit in real time or extract a random handwritten digit from the MNIST dataset. Once this data is put through the NN, the result should be displayed on the GUI along with the accuracy.

Chapter 2

Materials and Methods

2.1 Material List (Software Only)

- PyCharm Community 2020.3.3
- IDLE (Python 3.7 64-bit)
- Jupyter Notebook (Python 3)

2.2 Method/Procedure

1. Import Tensorflow and Keras

   Some necessary modules were imported in the first step. Tensorflow has been installed and imported. Similarly, Keras was imported from Tensorflow, which can provide some convenient and compact Application Programming Interface (API) [3]. It also supports convolutional neural network and some common layers [4], and standard neural network absolutely. Thus, the base of building and training of the network was constructed.

2. Load MNIST Dataset from Keras

   In this step, the MNIST file was imported from keras.dataset. MNIST of handwritten digits is a standard and commonly used dataset in pattern

recognition of deep learning. It consists of 60,000 training examples and 10,000 testing examples, which all centred in 28*28 grayscale images [5].

3. Achieve and Process Data

The 28*28 images were converted to a single dimensional array, which is about 784 elements. It was done by the function called reshape from numpy. The main reason to do that is the single dimensional array can be used as the input layer and the original ones cannot. In addition, the grayscale value of the given pixel is 0-255. In order to make the training effect of the model better, the numerical normalization was mapped to 0-1.

4. Build Convolutional Neural Network

Convolutional neural network was built in this part. The primary reason to use it is that CNN is powerful in analysing visual images and can be a promising way to solve relevant problems [6]. Several layers were used.

- Convolutional layers

  Three convolutional layers were used. The main function is to convolve the input values and pass the result to the next layer [6]. It had 6, 16, 120 filters in turn and the filters were used as feature detectors which helped the network recognise digit feature [7]. In addition, the activation function was "relu" function, which can be simply considered as ReLU(z)=max (0, x) [8]. Increasing nonlinear effect was the main functionality [8].

- Pooling layers

  Two average pooling layers were used, which takes the average value of each local cluster of neurons in the feature map [9]. Most importantly, it can reduce the computation and overfitting [9].

- Fully connected layers

  The layers were used to connect neurons of the convolutional layer to other layers [10]. It had 84 neurons. Moreover, the "softmax" function was applied to the output layer (10 output values). The values were converted to a probability distribution [11], which were the results we desired.

The neural network needs to be compiled after it has been built successfully. Some parameters were used. The optimizer was RMSprop, which maintained average of gradients [12]. "Categorical crossentropy" was chosen as the loss function, which evaluated the degree of difference between the predicted value and the true value of the model. Accuracy was the metrics when the neural network was compiling and the value should be high as the project aim.

5. Train Convolutional Neural Network

In this part, the "network. fit" was called to train the network. The number of iteration (Epoches) was set to 15, in order to run it 15 rounds. Verbose was set to 2 so that the program can output one line of records for each epoch [13].

6. Test Training Result

Test loss and test accuracy were set to be the evaluated results and were ready to be printed. Finally, all the code was run in this step.

7. Create Graphical User Interface

There were three functionalities that were needed for the system: a drawing board to collect handwritten content and to display pictures selected from the database, and a text box to display answers, also few buttons to interact with. The result is shown in the figure 1 below.

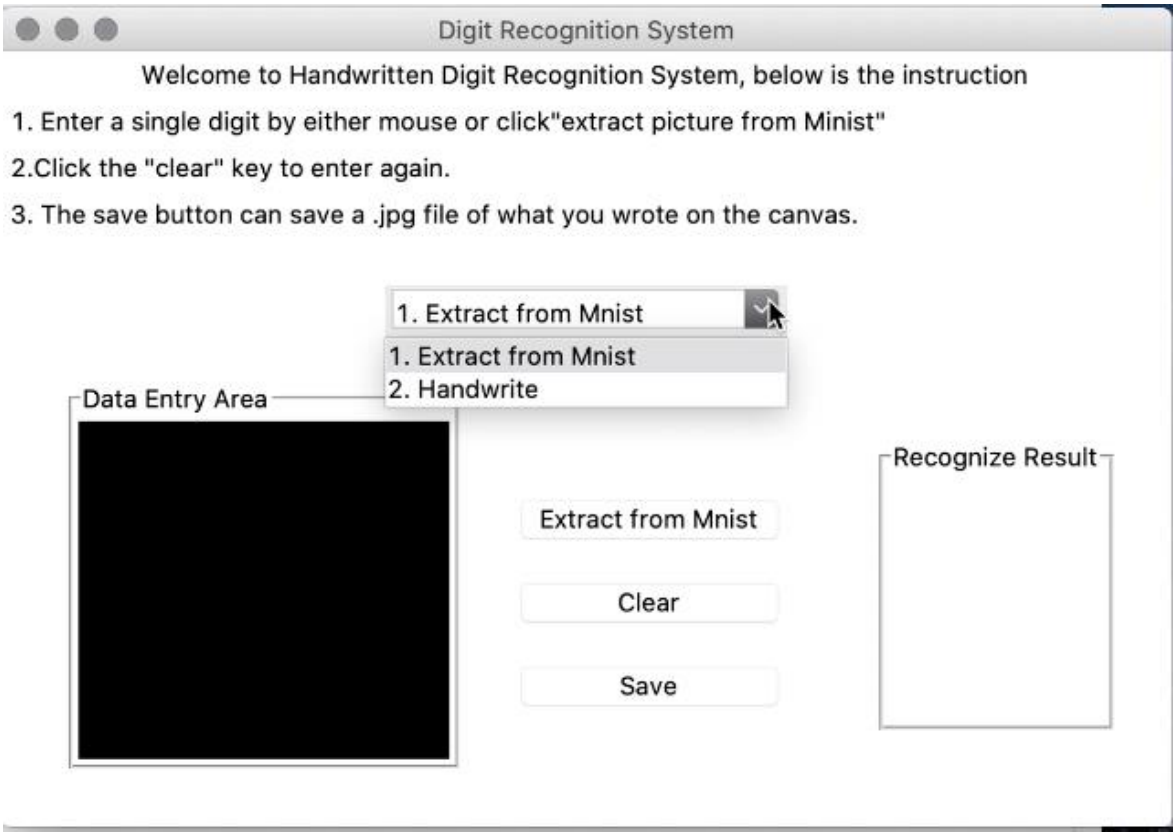


Figure 1. Graphical user interface for handwriting.

8. Format Handwritten Image

In order to fit the format of neural network input, OpenCV's API was used to convert RGB images to grayscale and binary images, and then reduced the pixels of the picture retrieved from the canvas component of the graphical user interface.

Chapter 3

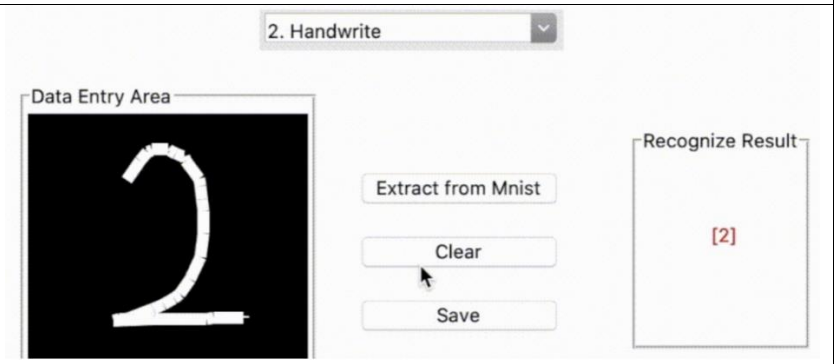
Results and Analysis

3.1 Results Presentation

After the program was run, the GUI was shown in figure 1. Several groups of tests were completed, which can be seen in the tables below.

Table 1. Testing result in handwritten mode

Handwritten digit	Practical value	Graphical User Interface
1	1	

4	4	
2	2	
8	0	
5	5	

As presented in Table 1, 5 digits have been handwritten in the entry area. Four digits have been predicted and recognised correctly, which are 1, 4, 2, and 5 in

sequence. However, wrong prediction still happened in the fourth testing. The number 8 was written, while the result was 0. It is not the value as we expected.

Table 2. Testing result in extract mode

Extract digit	Practical value	Graphical User Interface
4	4	
3	3	
8	8	

In the extract mode of the program. Three value groups were presented in Table 2. As shown in the GUI, it can be easily observed that all the digits were predicted

successfully. Not just the three values, many other tests have been done additionally. In this mode, the prediction accuracy is almost 100%.

3.2 Analysis

Unlike printed fonts, different people have different handwriting styles and different sizes, which makes the computer hard for handwriting recognition tasks. It is principal reason that 100% accuracy is complicated to be achieved.

3.1 part showed that the accuracy of extract mode was much higher than the handwritten mode. The main reason is the difference of handwritten style. As for the extract mode, even it was random, the datasets were still extracted from MNIST file. The distinction was not that obvious to some extent. However, as for the second mode, it was quite different. The sizes and places of the number cannot be fixed. For example, a tiny digit which was put in the lower left corner was difficult to be recognised well. In addition, people in different regions have different writing styles, probably do not match the writing style of the MNIST database.

Moreover, as mentioned in the sixth step of Materials and Methods section. Test loss and test accuracy were set to be the evaluated results and were printed in the following pictures.

```
Epoch 1/15
469/469 - 6s - loss: 0.3687 - accuracy: 0.8877
Epoch 2/15
469/469 - 6s - loss: 0.0961 - accuracy: 0.9704
Epoch 3/15
469/469 - 6s - loss: 0.0632 - accuracy: 0.9799
Epoch 4/15
469/469 - 6s - loss: 0.0478 - accuracy: 0.9850
Epoch 5/15
469/469 - 6s - loss: 0.0391 - accuracy: 0.9878
Epoch 6/15
469/469 - 6s - loss: 0.0322 - accuracy: 0.9898
Epoch 7/15
469/469 - 6s - loss: 0.0275 - accuracy: 0.9915
Epoch 8/15
469/469 - 7s - loss: 0.0236 - accuracy: 0.9927
Epoch 9/15
469/469 - 7s - loss: 0.0206 - accuracy: 0.9935
Epoch 10/15
469/469 - 7s - loss: 0.0172 - accuracy: 0.9945
Epoch 11/15
469/469 - 7s - loss: 0.0159 - accuracy: 0.9952
Epoch 12/15
469/469 - 7s - loss: 0.0146 - accuracy: 0.9955
Epoch 13/15
469/469 - 7s - loss: 0.0131 - accuracy: 0.9959
Epoch 14/15
469/469 - 7s - loss: 0.0112 - accuracy: 0.9962
Epoch 15/15
469/469 - 7s - loss: 0.0097 - accuracy: 0.9970
313/313 [==============================] - 1s 2ms/step - loss: 0.0347 - accuracy: 0.9906
test_loss :  0.03467671200633049
test_accuracy :  0.9905999898910522
```

Figure 2. The printed results of loss and accuracy.

It can be observed in Figure 2 that the accuracy was changed from 88.8% in the first epoch to 99.7% in the end, which was a significative improvement. In addition, the loss value was also changed from 0.369 to 0.0097. The final testing result was 0.0347 of the test loss value and 99.06% accuracy as shown in Figure 2.

The change of loss and accuracy was also presented in the Figure 3 below. Yellow line was for the accuracy and the blue one was for the test loss.

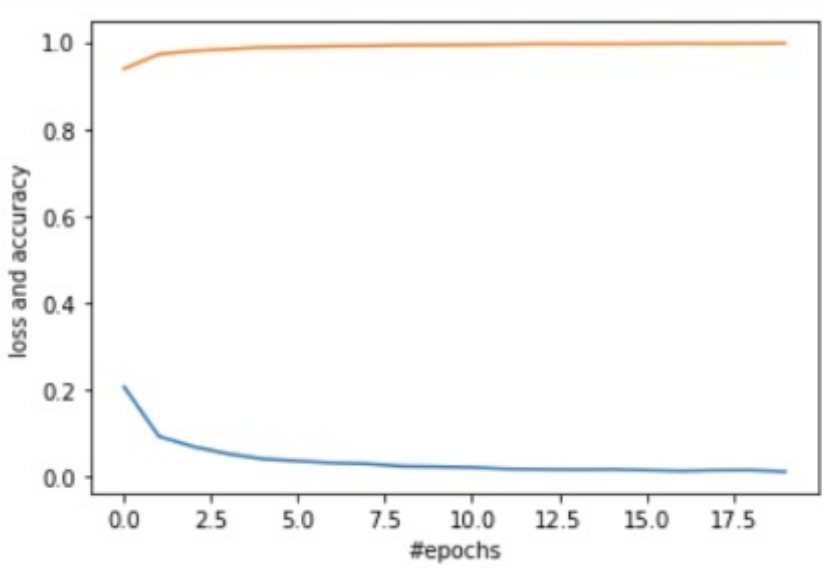


Figure 3. The loss and accuracy movement.

As shown in Figure 3, the loss value was on the decrease and the accuracy was rising all the time. It showed good performance. The ultimate value was appropriate and that is what we expected for this program. The detailed and individual pictures of loss and accuracy were saved in Appendix C.

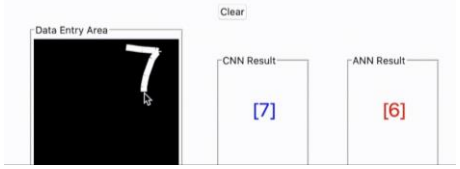
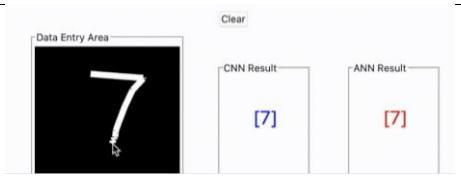
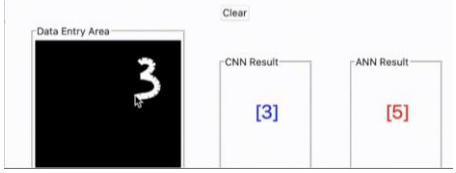
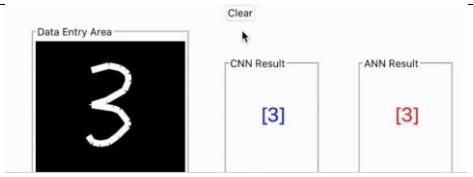
Chapter 4

Discussion and Conclusions

4.1 Comparison between ANN and CNN

The biggest difference between them is whether to use weights [16]. The working principle of ANN is to artificially reproduce the working principle of the simulated brain neural network. After the deviation from the answer appears, it will reflect and improve the way of thinking like a human being. The way ANN changes its thinking is achieved by changing the weights after each iteration (Epoch) [16].

Table 3. Comparison between practice accuracy between different networks

	At corner	In the middle
Example 1	Table 4. Comparison between practice accuracy between different networks 	
Example 2		

According to the test results of the MNIST test set, the correct rate that can be realized does not represent the actual performance of the two, because the pictures in the data set are accurately cropped, and the numbers will not be in the corners of the pictures and only occupy very small area, and the number of handwriting test trials we can provide is very limited. But in limited experiments, we found that cnn is indeed less sensitive to the position of the number in the picture in most cases, and the correct rate is higher when the number appears in the corner of the picture. There is no need to train dataset with scaled rotate and have digit shown at various positions.

In comparison, CNN is more inclined to provide multi-dimensional output for observing different features, and finally, connect the input to different features.

The biggest commonality between them is that they are constantly correcting the errors by comparing the answers to improve their learning and approach the answers.

4.2 The reason why the accuracy rate did not reach 100%

As mentioned in the 3.2-part, different people have different handwriting styles and different sizes, which makes the computer hard for handwriting recognition tasks. It is the principal reason that 100% accuracy is complicated to be achieved. In addition, some specific reason of the convolutional neural network, like not enough layers and dropout layers and so on. This can be improved with more time allocated.

4.3 Applications and Commercialisation

This project is not limited in using the image of the digits and recognizing the digit presented. Moreover, it can be applied to form data entry, sort postal mail, and bank check processing [14]. The developed system can even do the sexually explicit content detection and document categorization [14], which really benefits our future life. In detail, it can be made into a software of computer and mobile phones, instead of a lot of mechanized work, in order to reduce the use of manpower.

4.4 Ethical Considerations

It is worth considering that handwritten digit recognition may be used to falsify personal handwriting. Though people's handwriting style is not identical, the machine can falsify fonts with high similarity by recognizing handwriting. This may

cause personal property losses and other unnecessary troubles. Therefore, it is necessary to record the storage and use of data to prevent this from arising.

4.5 Future Work

If more time is allocated, the primary goal will be trying to improve the recognition accuracy of handwritten digits. It will be better if the rate can reach almost 100% like the top few programmers in Kaggle [15]. In addition, changing from handwritten digit recognition to letter recognition is also a possible direction, though it is more complicated. These can help beginners to learn machine learning in a deep way.

4.6 Conclusions

In conclusion, this paper has presented the use of a Convolutional Neural Network to create a software that recognises handwritten digits using a variety of pervious data sets. It has created a GUI that allows the user to either input their own handwritten digit in real time by drawing on the screen or extract data from the MNIST dataset. We see that using a Convolutional Neural Network (CNN) seems to be better for this software since it allows the user to input their handwritten digit anywhere on the screen and still have the model run efficiently. We decided to use TensorFlow since it's an approved package with more resources and capabilities, its made for deep learning and its more widely used. The original goals set out in terms of the GUI for this project were to have a model such that the user inputs their handwritten digit and presses a button to run the model and have their digit recognised, however we decided to take it one step further by having the model instantaneously run as the user draws their digit. This allows for a smoother process

and a more user-friendly experience. Along with this it allowed for extra space on the GUI design and another button to be added giving the user the feature to save their handwritten image as a JPEG file. The GUI however, ended up having a very basic design and had more time been available, a more user-friendly design may have been produced. The neural network came out very efficient at 97%. Although this is a relatively large number, it can still be improved as those NNs in the industrial market are sure to be touching higher than that, proving it is possible. Perhaps if a bigger dataset was used for more epochs, then maybe the NN would have achieved more accuracy. In summary, the goals outlined at the beginning of this paper have been achieved, a CNN of appropriate accuracy was created and an appropriate GUI was designed allowing the end user all the functions intended.

References

[1] Macukow, B., 2016. *Neural Networks – State of Art, Brief History, Basic Models and Architecture*. [online] SpringerLink. Available at: <https://link.springer.com/chapter/10.1007/978-3-319-45378-1_1> [Accessed 19 April 2021].

[2] Kocic, J., Jovicic, N. and Drndarević, V., 2021. *An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms*. [online] MDPI. Available at: <https://www.mdpi.com/1424-8220/19/9/2064> [Accessed 19 April 2021].

[3] K. Team, "Keras documentation: Keras API reference," *Keras*. [Online]. Available: https://keras.io/api/. [Accessed: 19-Apr-2021].

[4] K. Team, "Keras documentation: Core layers," *Keras*. [Online]. Available: https://keras.io/api/layers/core_layers/. [Accessed: 19-Apr-2021].

[5] "THE MNIST DATABASE," *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 19-Apr-2021].

[6] M. V. Valueva, N. N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020.

[7] R. Khandelwal, "Convolutional Neural Network: Feature Map and Filter Visualization," *Medium*, 18-May-2020. [Online]. Available: https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c#:~:text=CNN%20uses%20learned%20filters%20to%20convolve%20the

%20feature,bias%20values%20using%20get_weights%20%28%29%20for%20that%20layer. [Accessed: 19-Apr-2021].

[8] J. Brownlee, "A Gentle Introduction to the Rectified Linear Unit (ReLU)," *Machine Learning Mastery*, 20-Aug-2020. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/#:~:text=The%20rectified%20linear%20activation%20function%20or%20ReLU%20for,easier%20to%20train%20and%20often%20achieves%20better%20performance. [Accessed: 19-Apr-2021].

[9] S. Mittal, "A Survey of FPGA-based Accelerators for Convolutional Neural Networks," *Neural Computing and Applications*. [Online]. Available: https://www.academia.edu/37491583/A_Survey_of_FPGA_based_Accelerators_for_Convolutional_Neural_Networks. [Accessed: 19-Apr-2021].

[10] P. Mahajan, "Fully Connected vs Convolutional Neural Networks," *Medium*, 23-Oct-2020. [Online]. Available: https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5. [Accessed: 19-Apr-2021].

[11] "Softmax Function," *DeepAI*, 17-May-2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer. [Accessed: 19-Apr-2021].

[12] K. Team, "Keras documentation: RMSprop," *Keras*. [Online]. Available: https://keras.io/api/optimizers/rmsprop/. [Accessed: 19-Apr-2021].

[13] K. Team, "Keras documentation: Model training APIs," *Keras*. [Online]. Available: https://keras.io/api/models/model_training_apis/. [Accessed: 19-Apr-2021].

[14] S. M. Shamim, M. B. A. Miah, and A. Sarker, "Handwritten Digit Recognition using Machine Learning Algorithms." [Online]. Available: https://www.researchgate.net/publication/326408524_Handwritten_Digit_Recognition_Using_Machine_Learning_Algorithms#:~:text=The%20applications%20of%20digit%20recognition%20includes%20in%20postal,of%20a%20scanner%2C%20tablet%2C%20and%20other%20digital%20devices.

[15] "Digit Recognizer," *Kaggle*. [Online]. Available: https://www.kaggle.com/c/digit-recognizer/leaderboard. [Accessed: 19-Apr-2021].

[16] Guo Y., Bai L., Lao S., Wu S., Lew M.S. (2014) A Comparison between Artificial Neural Network and Cascade-Correlation Neural Network in Concept Classification. In: Ooi W.T., Snoek C.G.M., Tan H.K., Ho CK., Huet B., Ngo CW. (eds) Advances in Multimedia Information Processing – PCM 2014. PCM 2014. Lecture Notes in Computer Science, vol 8879. Springer, Cham. https://doi.org/10.1007/978-3-319-13168-9_26

Appendices

Appendix A

Project management forms

1. Attendance Record

Attendance record.docx

2. Role Allocation Form

Role allocation (responsibility ma

3. Contribution to project deliverables

Contribution to project deliverabl

4. Supervisor weekly meeting log-1

Supervisor weekly meeting l

Appendix B

Breakdown of individual contributions to the project

In addition to the third form in Appendix A. The contributions of this report were shown below.

Table 4 . Contributions of group report

Group Member	Contributions
Ziqi Sun	1. Materials and methods 2. Results and Analysis 3. Discussion
Xiaowei Shi	1. Materials and methods 2. Discussion
Wasif Ali	1. Abstract 2. Introduction 3. Conclusion

Appendix C

1. Loss Function

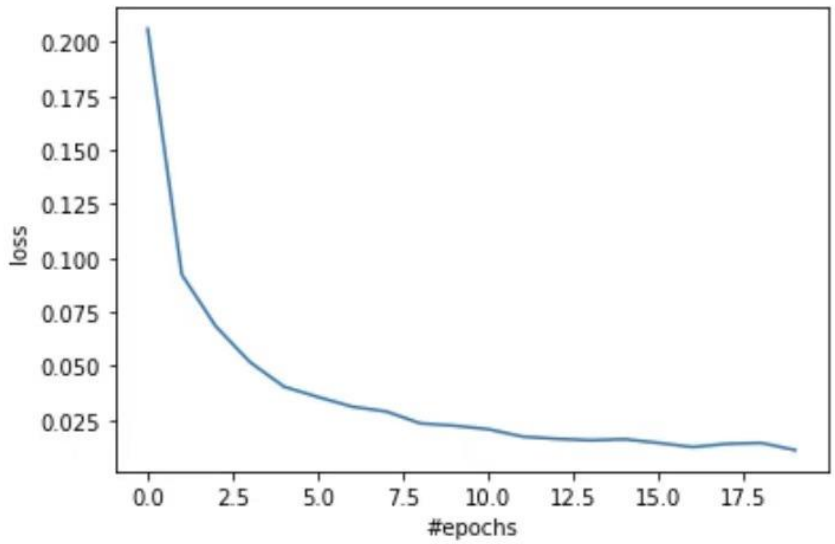


Figure 4. The variation of loss function

2. Accuracy

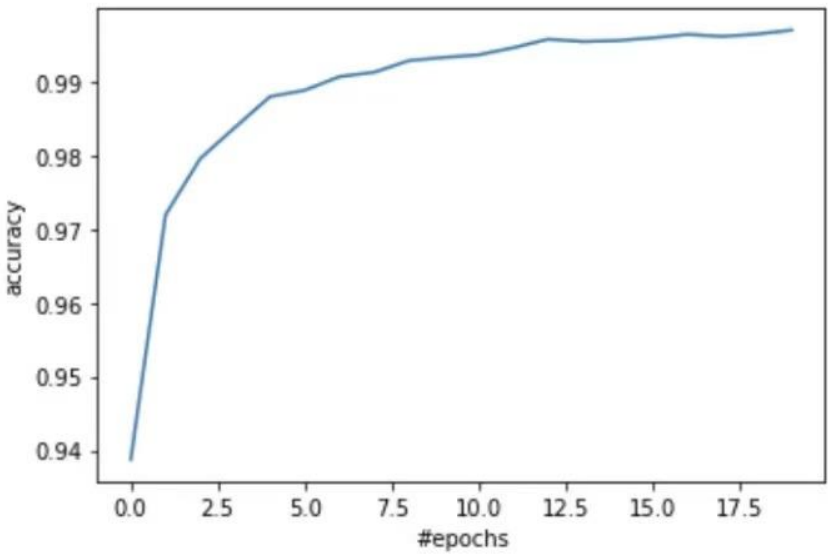


Figure 5. The variation of Accuracy

Appendix D

Source Code

Convolutional neural network

```
from keras import models, layers
from keras.utils import to_categorical
from keras.datasets import mnist
from keras.optimizers import RMSprop
import numpy

(train_image, train_label), (test_image, test_label) = mnist.load_data()

# building convolutional neural network
def LeNet():
    net = models.Sequential()
    net.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
    net.add(layers.AveragePooling2D((2, 2)))
    net.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
    net.add(layers.AveragePooling2D((2, 2)))
    net.add(layers.Conv2D(filters=120, kernel_size=(3, 3), activation='relu'))
    net.add(layers.Flatten())
    net.add(layers.Dense(84, activation='relu'))
    net.add(layers.Dense(10, activation='softmax'))
    return net
# compile it
net = LeNet()
net.compile(optimizer=RMSprop(lr=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])

train_images = train_image.reshape((60000, 28, 28, 1)).astype('float') / 255
test_images = test_image.reshape((10000, 28, 28, 1)).astype('float') / 255
```

```
train_labels = to_categorical(train_label)
test_labels = to_categorical(test_label)

# testing the results
net.fit(train_images, train_labels, epochs=15, batch_size=128, verbose=2)
test_loss, test_accuracy = net.evaluate(test_images, test_labels)
print('test_loss : ', test_loss)
print('test_accuracy : ', test_accuracy)
```

GUI

```
""" for user to control"""

import sys
import pickle
import gzip

from tkinter import *
from tkinter import ttk
import tensorflow as tf
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras import models, layers
from keras.optimizers import RMSprop
from PIL import Image, ImageDraw,ImageQt,ImageTk
import numpy as np
import cv2 as cv

f = gzip.open('mnist.pkl.gz', 'rb')
if sys.version_info < (3,):
    data = pickle.load(f)
else:
    data = pickle.load(f, encoding='bytes')
```

```
f.close()
(x_train, y_train), (x_test, y_test) = data


#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam',
loss='categorical_crossentropy',metrics=['accuracy'])
#model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',metrics=['accuracy'])
#model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=6)
#model.summary()
#model.save_weights("model.h5")
model.load_weights("anotherModel.h5")


network = Sequential()
network.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
network.add(layers.AveragePooling2D((2, 2)))
network.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
network.add(layers.AveragePooling2D((2, 2)))
network.add(layers.Conv2D(filters=120, kernel_size=(3, 3), activation='relu'))
network.add(layers.Flatten())
network.add(layers.Dense(84, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer=RMSprop(lr=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
network.load_weights("convolution.h5")
```

```
count = 0
class Controller(Frame):



    def __init__(self,master=None):
        super().__init__(master)
        self.master = master
        self.pack()
        self.createWidgets()

        self.startFlag = False
        self.handWriting = Image.new("RGB",(200,200),(0,0,0))
        self.imgDraw = ImageDraw.Draw(self.handWriting)
    def createWidgets(self):
        label1 = Label(window, text ="Welcome to Handwritten Digit Recognition
System, below is the instruction").pack(side = TOP)
        label2 = Label(window, text ="1. Enter a single digit by either mouse or
click\"extract picture from Minist\"").pack(side = TOP,anchor = W)
        label3 = Label(window, text = "2.Click the \"clear\" key to enter
again.").pack(side = TOP,anchor = W)
        label4 = Label(window, text ="3. The save button can save a .jpg file of what
you wrote on the canvas.").pack(side = TOP,anchor = W)
        label5 = Label(window, text ="").pack(side = TOP,anchor = W)

        self.modeCombobox = ttk.Combobox(window, width = 20)
        self.modeCombobox["values"] = ("1. Extract from Mnist","2. Handwrite")
        self.modeCombobox.current(0)
        self.modeCombobox.pack()

        writeFrame = LabelFrame(window,text="Data Entry Area")
        buttonFrame = Frame(window)
        resultFrame = LabelFrame(window,text="Recognize Result")
        writeFrame.place(x=35,y=170,width=200,height=200)
        buttonFrame.place(x=265,y=210,width=135,height=150)
        resultFrame.place(x=450,y=200,width=120,height=150)
```

```
        self.canvas = Canvas(writeFrame, bg="black",width=200,height=200 )
        self.canvas.bind("<B1-Motion>",self.writing)
        self.canvas.bind("<ButtonRelease>",self.stop)
        self.canvas.pack(fill=BOTH,expand=YES)

        clearButton = Button(buttonFrame, text = "Clear", command = self.clear)
        saveButton = Button(buttonFrame, text = "Save", command = self.save)
        mnisButton = Button(buttonFrame, text = "Extract from Mnist", command
= self.extract)
        mnisButton.pack(side = TOP,anchor = W,expand=YES,fill=X )
        clearButton.pack(side = TOP,anchor = W,fill=X )
        saveButton.pack(side = TOP,anchor = W,expand=YES,fill=X)

        self.resultCanvas = Canvas(resultFrame)
        self.resultCanvas.pack(fill=BOTH,expand=YES)

    def writing(self,event):
        self.modeCombobox.current(1)
        self.resultCanvas.delete("all")
        if not self.startFlag:
            self.startFlag = True
            self.x=event.x
            self.y=event.y
        self.canvas.create_line((self.x,self.y,event.x,event.y),width = 8,fill="white")
        self.imgDraw.line((self.x,self.y,event.x,event.y),fill="white",width = 19)
        self.x=event.x
        self.y=event.y
        self.imgArrOrigin = np.array(self.handWriting)
        self.imgArr = cv.resize(self.imgArrOrigin,(28,28))#interpolation?
        self.imgArr = cv.cvtColor(self.imgArr,cv.COLOR_BGR2GRAY)
        self.imgArr = self.imgArr.reshape((1,28,28,1)).astype('float')/255
        #self.imgArr = self.imgArr/255.0
        #self.imgArr = self.imgArr.reshape(1,28,28)
        result = network.predict(self.imgArr)
        label = np.argmax(result,axis =1)
        self.resultCanvas.create_text(60,55,text=str(label),fill="red")
```

```
    def stop(self,event):
        self.startFlag = False
    def clear(self):
        self.canvas.delete("all")
        self.handWriting = Image.new("RGB",(200,200),(0,0,0))
        self.imgDraw = ImageDraw.Draw(self.handWriting)
        self.resultCanvas.delete("all")
    def extract(self):
        self.canvas.delete("all")
        self.resultCanvas.delete("all")
        self.modeCombobox.current(0)
        randomInt = np.random.randint(0,9999)
        self.mnistArray = x_test[randomInt]
        mnistArrayBig = cv.resize(self.mnistArray,(200,200),interpolation =
cv.INTER_LINEAR)
        self.mnistImage = ImageTk.PhotoImage(Image.fromarray(mnistArrayBig))
        self.canvas.create_image(100,100,image=self.mnistImage)
        self.mnistArray = self.mnistArray/255.0
        self.mnistArray = self.mnistArray.reshape(1,28,28)
        result = model.predict(self.mnistArray)
        label = np.argmax(result,axis =1)
        self.resultCanvas.create_text(60,55,text = str(label),fill="blue")
    def save(self):
        self.imgArr = np.array(self.handWriting)
        self.imgArr = cv.resize(self.imgArr,(28,28))
        global count
        cv.imwrite(str(count)+".jpg",self.imgArr)
        count = count+1
        cv.imshow(str(count)+".jpg",self.imgArr)
        print("file saved")


window = Tk()
window.title("Digit Recognition System")
window.geometry("600x400")
```

```
controller = Controller(master=window)
window.mainloop()
```

GUI(Comparison)

```
""" for user to control"""

import sys
import pickle
import gzip

from tkinter import *
from tkinter import ttk
import tensorflow as tf
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras import models, layers
from keras.optimizers import RMSprop
from PIL import Image, ImageDraw,ImageQt,ImageTk
import numpy as np
import cv2 as cv

f = gzip.open('mnist.pkl.gz', 'rb')
if sys.version_info < (3,):
    data = pickle.load(f)
else:
    data = pickle.load(f, encoding='bytes')
f.close()
(x_train, y_train), (x_test, y_test) = data



#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(512, activation='relu'))
```

```
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam',
loss='categorical_crossentropy',metrics=['accuracy'])
#model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',metrics=['accuracy'])
#model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=6)
#model.summary()
#model.save_weights("model.h5")
model.load_weights("anotherModel.h5")


network = Sequential()
network.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
network.add(layers.AveragePooling2D((2, 2)))
network.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
network.add(layers.AveragePooling2D((2, 2)))
network.add(layers.Conv2D(filters=120, kernel_size=(3, 3), activation='relu'))
network.add(layers.Flatten())
network.add(layers.Dense(84, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer=RMSprop(lr=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
network.load_weights("convolution.h5")


class Controller(Frame):


    def __init__(self,master=None):
        super().__init__(master)
        self.master = master
        self.pack()
```

```
        self.createWidgets()

        self.startFlag = False
        self.handWriting = Image.new("RGB",(200,200),(0,0,0))
        self.imgDraw = ImageDraw.Draw(self.handWriting)
    def createWidgets(self):
        label1 = Label(window, text ="Welcome to Handwritten Digit Recognition
System").pack(side = TOP)
        label2 = Label(window, text ="below is the instruction:").pack(side =
TOP,anchor = W)
        label3 = Label(window, text ="1. Enter a single digit using mouse").pack(side
= TOP,anchor = W)
        label4 = Label(window, text ="2.Click the \"clear\" key to enter
again.").pack(side = TOP,anchor = W)
        label5 = Label(window, text ="***This System Is Designed for Comparing
CNN and ANN Result***").pack()
        label6 = Label(window, text ="").pack(side = TOP,anchor = W)

        clearButton = Button(window, text = "Clear", command = self.clear)
        clearButton.pack()

        writeFrame = LabelFrame(window,text="Data Entry Area")
        convoResultFrame = LabelFrame(window,text="CNN Result")
        artiResultFrame = LabelFrame(window,text="ANN Result")
        writeFrame.place(x=35,y=170,width=200,height=200)
        convoResultFrame.place(x=280,y=210,width=120,height=150)
        artiResultFrame.place(x=450,y=210,width=120,height=150)

        self.canvas = Canvas(writeFrame, bg="black",width=200,height=200 )
        self.canvas.bind("<B1-Motion>",self.writing)
        self.canvas.bind("<ButtonRelease>",self.stop)
        self.canvas.pack(fill=BOTH,expand=YES)

        self.resultCanvas = Canvas(artiResultFrame)
        self.resultCanvas.pack(fill=BOTH,expand=YES)
        self.resultCanvas1 = Canvas(convoResultFrame)
```

```
        self.resultCanvas1.pack(fill=BOTH,expand=YES)

    def writing(self,event):
        #self.modeCombobox.current(1)
        self.resultCanvas.delete("all")
        self.resultCanvas1.delete("all")
        if not self.startFlag:
            self.startFlag = True
            self.x=event.x
            self.y=event.y
        self.canvas.create_line((self.x,self.y,event.x,event.y),width = 8,fill="white")
        self.imgDraw.line((self.x,self.y,event.x,event.y),fill="white",width = 19)
        self.x=event.x
        self.y=event.y
        self.imgArrOrigin = np.array(self.handWriting)
        self.imgArr = cv.resize(self.imgArrOrigin,(28,28))#interpolation?
        self.imgArr = cv.cvtColor(self.imgArr,cv.COLOR_BGR2GRAY)
        self.imgArr = self.imgArr.reshape((1,28,28,1)).astype('float')/255
        #self.imgArr = self.imgArr/255.0
        #self.imgArr = self.imgArr.reshape(1,28,28)
        artiResult = model.predict(self.imgArr)
        label = np.argmax(artiResult,axis =1)
        self.resultCanvas.create_text(60,55,text=str(label),fill="red",font=("Purisa",
25))
        convoResult = network.predict(self.imgArr)
        label1 = np.argmax(convoResult,axis =1)

self.resultCanvas1.create_text(60,55,text=str(label1),fill="blue",font=("Purisa",25))
    def stop(self,event):
        self.startFlag = False

    def clear(self):
        self.canvas.delete("all")
        self.canvas.delete("all")
        self.handWriting = Image.new("RGB",(200,200),(0,0,0))
        self.imgDraw = ImageDraw.Draw(self.handWriting)
```

```
        self.resultCanvas.delete("all")
        self.resultCanvas1.delete("all")

window = Tk()
window.title("Digit Recognition System")
window.geometry("600x400")
controller = Controller(master=window)
window.mainloop()
```

Testing in Jupyter

```
from tkinter import *

from tkinter import ttk

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Flatten

from PIL import Image, ImageDraw,ImageQt,ImageTk

import numpy as np

import cv2 as cv

from keras.utils import np_utils


(x_train, y_train), (x_test, y_test) = mnist.load_data()


model = Sequential()

model.add(Flatten(input_shape=(28,28)))

model.add(Dense(512, activation='relu'))
```

```
model.add(Dense(256, activation='relu'))

model.add(Dense(128, activation='relu'))

model.add(Dense(64, activation='relu'))

model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam',
loss='categorical_crossentropy',metrics=['accuracy'])

model.summary()

newytrain = np_utils.to_categorical(y_train)

newytest = np_utils.to_categorical(y_test)

newxtrain = x_train/255

newxtest = x_test/255

model.fit(newxtrain, newytrain, epochs=20)


model.save_weights("anotherModel.h5")


from matplotlib import pyplot as plt


plt.plot(model.history.history['accuracy'])

plt.plot(model.history.history['loss'])

plt.xlabel('#epochs')

plt.ylabel('accuracy and loss')

plt.show()


from matplotlib import pyplot as plt
```

```
plt.plot(model.history.history['accuracy'])

plt.xlabel('#epochs')

plt.ylabel('accuracy and loss')

plt.show()


from matplotlib import pyplot as plt


plt.plot(model.history.history['loss'])

plt.xlabel('#epochs')

plt.ylabel('accuracy and loss')

plt.show()
```