ANLY535 Final Project Report

CNN for Handwritten Recognition

Xiaowei Wan

The aim of this project is to implement a image classification algorithm to recoginize handwritten digits. This paper, I will present a simple 2D Convolutional Neural Network (CNN) to tackle the recognition of human handwritten digits. I want to tackle human handwriting recognition task for notes taken applications usage. It is useful for users who used to write on the paper, when you save paper notes into online notes app, it will be search friendly, and improve user experience. Hence, note taken apps can attract more users, and target different users.

Introduction

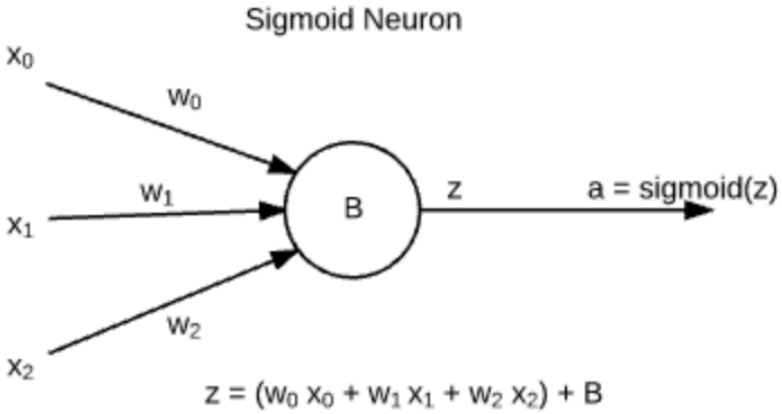
Nowadays, people are using notes taken app daily, and most of people still taking notes on paper, it will be very helpful if we can recognize human handwritten digits, and save it into note taken app. Thus it can improve search experience, the users can be potential user who can be converted into premium users for the applications. Different people have very different hand writing, even the same people written in different time are not identical. I am taking Machine Learning course, with Google TensorFlow came out, it is not hard to tackle this problem.

Give a set of labeled digits data, train a model which can be applied over to handwritten detection application (like notes, evernote, onenote…). Nowadays, Machine Learning and Deep learning is becoming more and more popular, as a note company, it will be good to smart detect handwriting, and translate handwriting notes into text, easy for search and improve user experience. This project

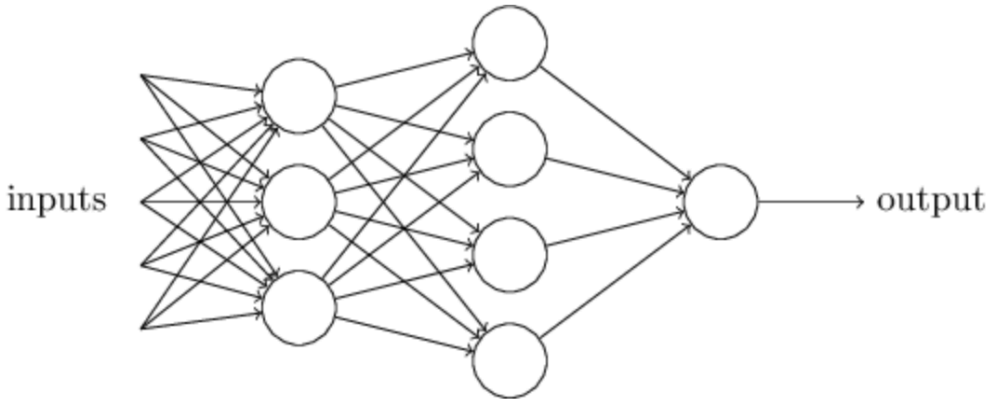
is trying to build a model to detect handwritten digits, and use the model into note app.

Method

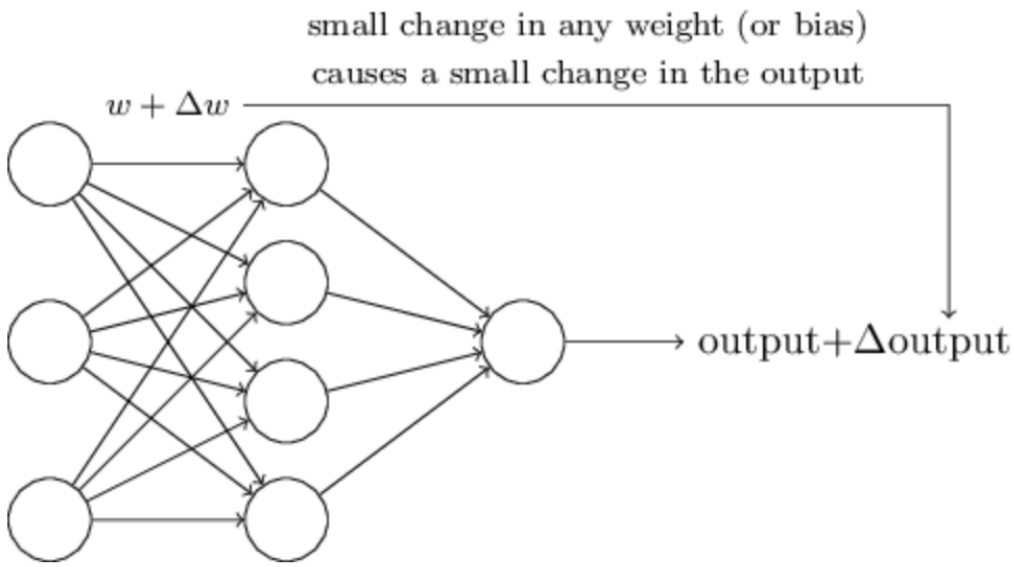
SNN (Simple Neural Network) – A neural network is made up by stacking layers of neurons, and is defined by the weights of connections and biases of neurons. Activations are a result dependent on a certain input, which made up building blocks known as Sigmoid Neurons. [9]



There are multiple layers between inputs and outputs.



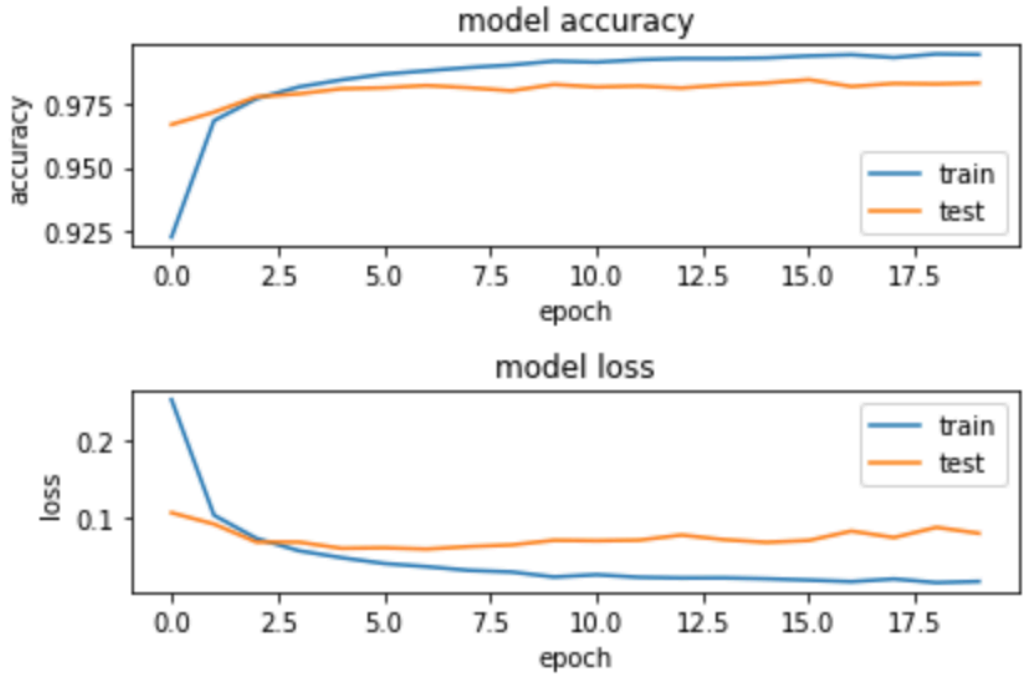
Each training, it will make small changes in some weight (or bias) in the network, it will cause a small corresponding change in the output from the network. So we can modify the weights and biases to get more accurate output. [10]



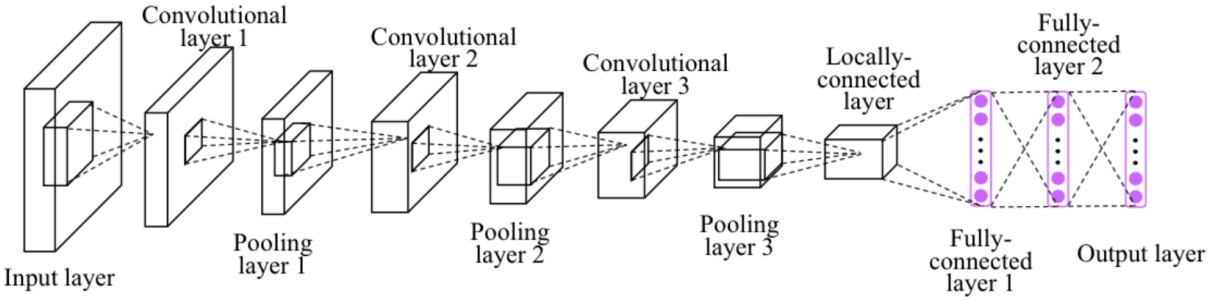
Simple Neural Network, train model with epoch 20, the result:

```
('Test Loss', 0.07854913353811863)
('Test Accuracy', 0.9836)
```

Accuracy is 98.36%. Below plot accuracy model and loss model:



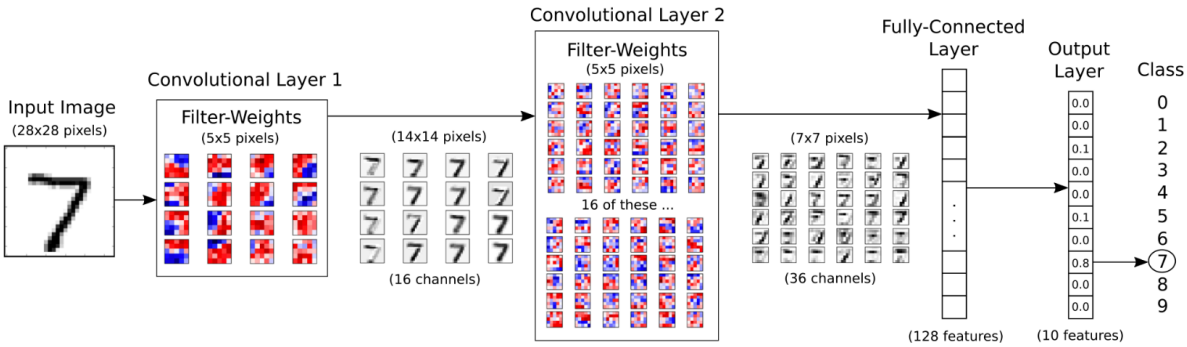
CNN is widely used for facial recognition, self-driving cars, object detection.
CNN Overview [2]:



It is been approved that CNN is very good on text classification. So in this project, I will focus on using CNN. The input of my project is a gray level image, the intensity level of which varies from 0 to 255. For simplicity, input image will only contain certain fixed size, and contain only center digit. We can preprocess this images into required images using computer vision techniques [3], which will not discuss here, you can check out if you are interested.
In this project, will be using MNIST handwritten digits. The dataset contains 60,000 examples for training and 10,000 examples for testing. The digits have been size-normalized and centered in 28x28 pixels image.
There are 10 classes (one for each of the 10 digits). I will build and compile CNN model to train 60,000 images and subsequently test its classification accuracy using 10,000 test images.
The training dataset is structured as a 3-dimensional array of instance, image width and image height. For a multi-layer pmodel, we will reduce the image down into a vector of pixels. In this case, it will be 784 (28*28) pixels.
We will do reshape to transform the dataset. We will do multiple CNN layers neural network. Including Convolutional layers, Pooling layers, Dropout layers, Flatten layers and Dense layers. A softmax activation function is used on the output layer to return the outputs

into probability like values and allow one class of the 10 to be selected as the model's output prediction.
I run the model of fitting over 5 and 20 epochs and a batch size of 128.

Flow chart shoing how CNN implemented:[11]



MNIST Dataset overview [2]

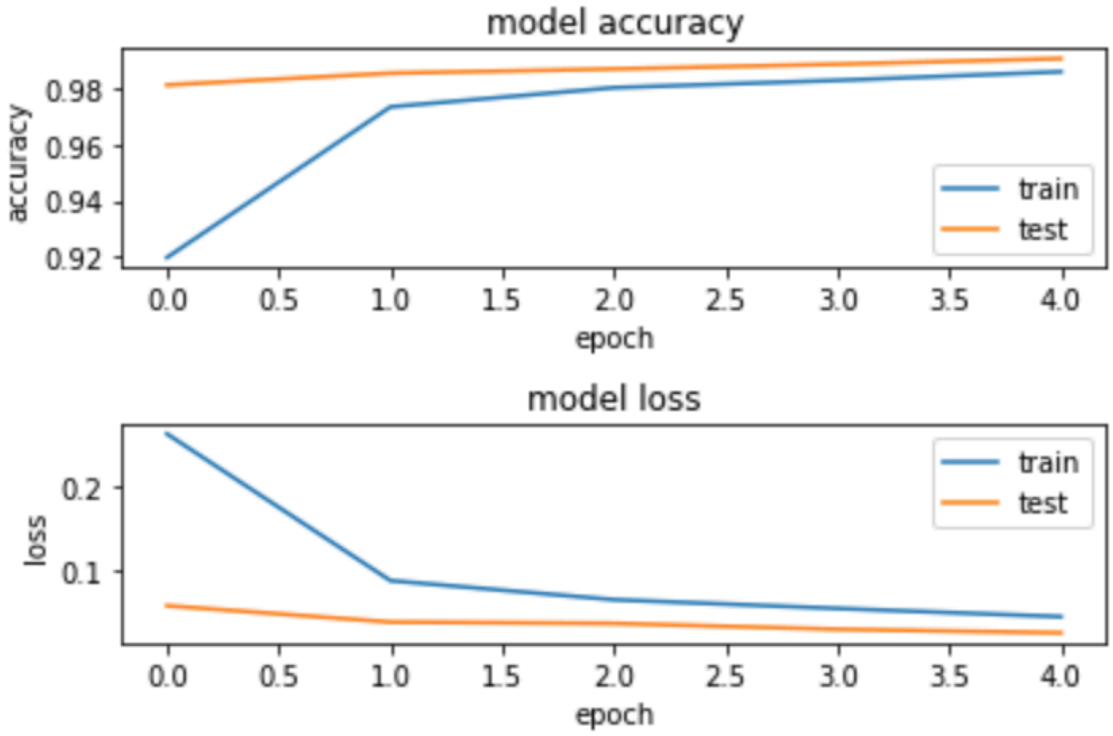
Results

Run the CNN model with the training images, from fig 1, we can tell that the accuracy is 99%, very high, loss is 2%, low loss, and from the plot of model accuracy and model loss, it shows that the result is not overfitting.
Train epoch 5 result:

Fig 1:

```
Test loss: 0.026911706153815614
Test accuracy: 0.9908
```

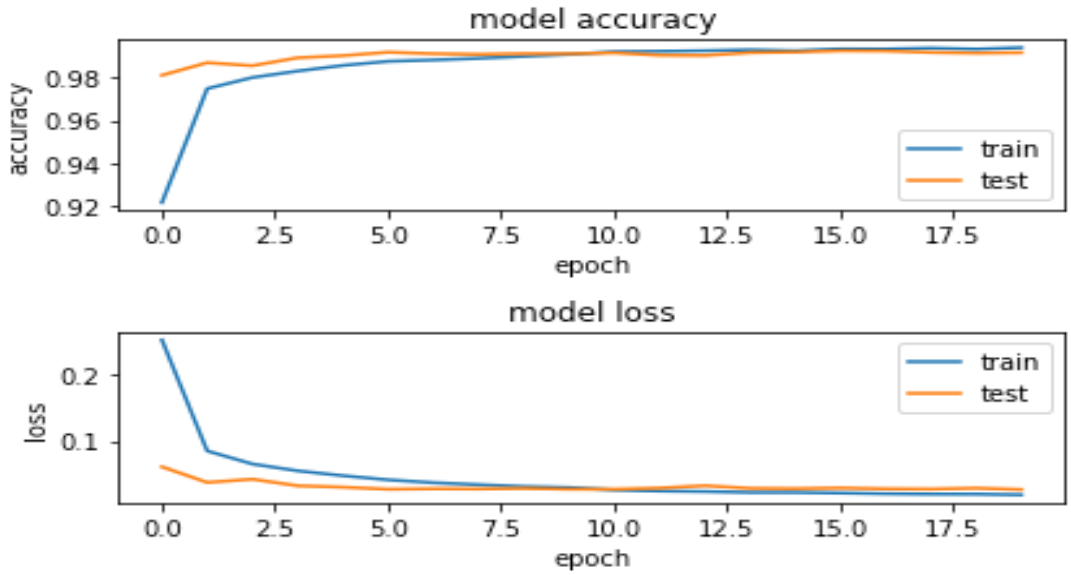
Fig 2:



Train epoch 20 result:

```
In [11]: # evaluate model
         score = model.evaluate(x_test, y_test, verbose=0)
         print('Test loss:', score[0])
         print('Test accuracy:', score[1])
```

```
Test loss: 0.026310968459603283
Test accuracy: 0.9918
```



The more epoch you train, the more accuracy you will get, with a real work datasets, meaning it will be much bigger, you can train with higher epochs, and get better output.

Discussion

There are lots of different algorithms, such as K-NN, SVM, etc. I found some comparison from other research paper, below is the result of the accuracy and the running time for different model on the test dataset:

Table 1: Comparison of different model before PCA

Model	Accuracy	Time
Cosine Similarity	82.21%	0.05s
Softmax Regression	91.38%	30.53s
Linear SVM	91.72%	132.62s
K-NN	97%	1035.77s
CNN	99.17%	164.33s

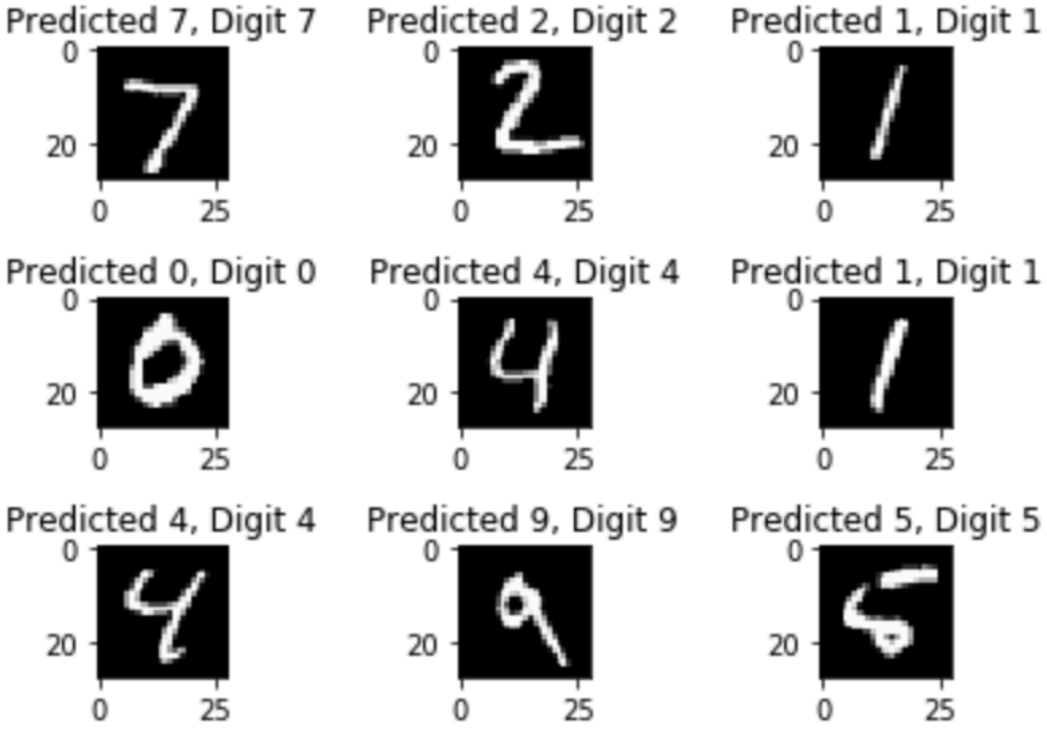
[4]

From Fig 1 and Fig 2, the result of CNN model is very good, have 99% accuracy.

In my project, the prediction images are graysacle, and looked the images are pretty high quality, and in real world, the images' quality may be very bad, thus, for further testing, I will use computer vision techniques to preprocess the images by blurring the images, and cropping the images, flipping images, and resizing the images. Then building and compiling the model to train the images. Then calculate the accuracy and loss, compare the results with non-preprocess dataset to evaluate the conclusion.

Fig 3:

```
Found correct labels:  9901
```



Conclusion

By research of different models performance on MNIST dataset, my implementation here using CNN seems the best solution to using different layers building a model to train the dataset, and predict the results.
There must be better solution, and we can investigate on other algorithm and models.
With higher than 99% accuracy, I can say that we can use this model into note taken app, and build a feature of it.

References

1. http://individual.utoronto.ca/gauravjain/ECE462-HandwritingRecognition.pdf
2. https://github.com/aymericdamien/TensorFlow-Examples/blob/master/notebooks/3_NeuralNetworks/convolutional_network.ipynb
3. https://docs.opencv.org/3.4/da/d6e/tutorial_py_geometric_transformations.html
4. https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a059.pdf
5. https://pdfs.semanticscholar.org/8a3d/c02a337f6a07d0f6da254992defa007a8322.pdf
6. MNIST Database of Handwritten digits: http://yann.lecun.com/exdb/mnist/
7. https://mxnet.incubator.apache.org/tutorials/python/mnist.html
8. https://towardsdatascience.com/a-simple-2d-cnn-for-mnist-digit-recognition-a998dbc1e79a
9. https://github.com/kdexd/digit-classifier
10. http://neuralnetworksanddeeplearning.com/chap1.html
11. https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb