Implementing CNN Architecture in Mnist Dataset

SECTION- B

Submitted By: Talha, Mehedi Hasan || 19-39597-1

Implementing CNN Architecture in Mnist Dataset

Mehedi Hasan Talha
CSE Department, American International University-Bangladesh
iamtalha.mht@gmail.com

***Abstract* – The sole purpose of this report is to present the implementation report of the CNN architecture in Mnist dataset. As a solution, the CNN architecture was implemented getting accuracy over 98%. Different types of optimizers such as Adam, SGD and RMSProp were used in modeling. This report will help find out the way to get maximum accuracy in Mnist dataset.**

Index Terms* – *CNN, Mnist, Python, Jupyter Notebook

I. INTRODUCTION:

A. Background information:
CNN or the convolutional neural network is a model to analyze image contents. It is a class of artificial neural network. It employs convolutional mathematical operation. The CNN has three layers. The input layer inputs a tensor with a shape. The hidden layers performs convolution. The output layer flattens the input from other layers and sent. The Mnist dataset was used on this project. It consists of handwritten digits. It has a training set of 60,000 examples and a test set of 10,000 examples. All the coding parts were done using the Jupyter Notebook from the Anaconda.

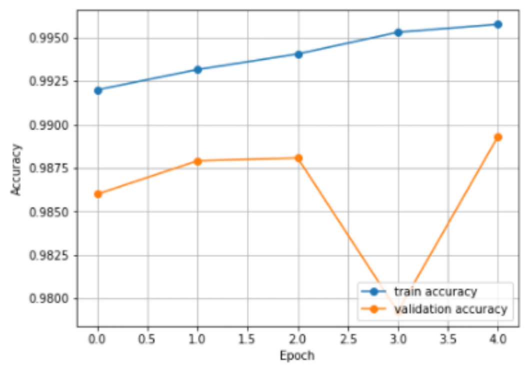
B. Overview of this report:
This report is intending to improve the accuracy of the CNN architecture. To improve the accuracy, it is necessary to implement the model with different types of optimizers. These optimizers will find the maximum accuracy.

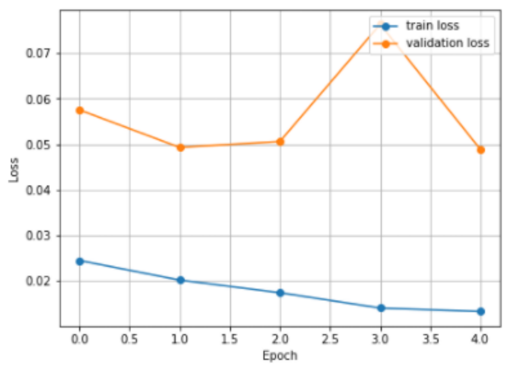
II. RESULT:

The Mnist dataset has 60,000 training images. So, images were trained at first. For the training, five epochs were conducted. And the training result was quite impressive. The images were reshaped into (28, 28, 1). That is why the accuracy went over 98%.
Using the different optimizers was also fruitful. These three optimizers gave three results. Each of them gave the accuracy over 98%.

A. Adam:

```
Epoch 1/5
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0245 - accuracy: 0.9920 - val_loss: 0.0575 - val_accuracy:
0.9860
Epoch 2/5
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0202 - accuracy: 0.9932 - val_loss: 0.0493 - val_accuracy:
0.9879
Epoch 3/5
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0174 - accuracy: 0.9941 - val_loss: 0.0506 - val_accuracy:
0.9881
Epoch 4/5
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0141 - accuracy: 0.9953 - val_loss: 0.0763 - val_accuracy:
0.9793
Epoch 5/5
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0133 - accuracy: 0.9958 - val_loss: 0.0489 - val_accuracy:
0.9893
```

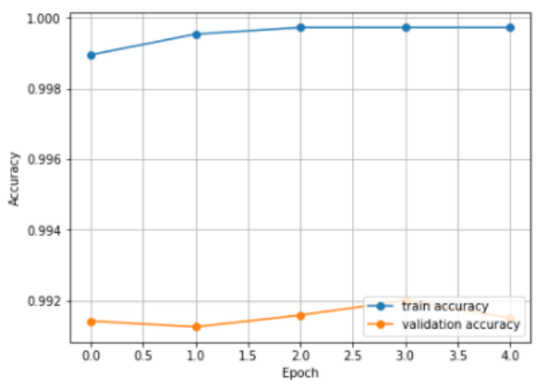


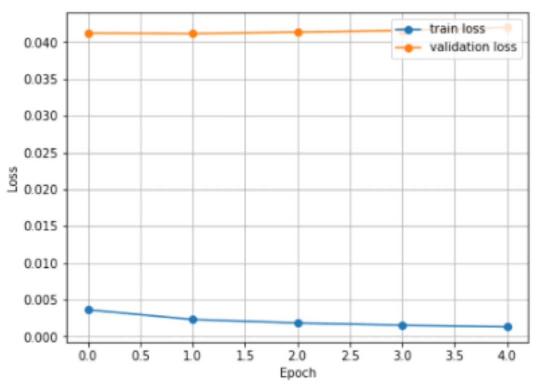


```
313/313 [==============================] - 1s 3ms/step - loss: 0.0419 - accuracy: 0.9896

Test Accuracy: 0.9896000027656555

Test Loss: 0.04190055653452873
```

B. SGD:

```
Epoch 1/5
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0036 - accuracy: 0.9990 - val_loss: 0.0412 - val_accuracy:
0.9914
Epoch 2/5
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0023 - accuracy: 0.9995 - val_loss: 0.0412 - val_accuracy:
0.9912
Epoch 3/5
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0018 - accuracy: 0.9997 - val_loss: 0.0414 - val_accuracy:
0.9916
Epoch 4/5
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0015 - accuracy: 0.9997 - val_loss: 0.0416 - val_accuracy:
0.9920
Epoch 5/5
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0013 - accuracy: 0.9997 - val_loss: 0.0420 - val_accuracy:
0.9915
```

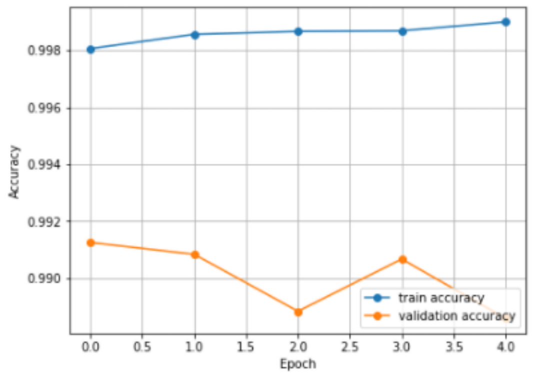


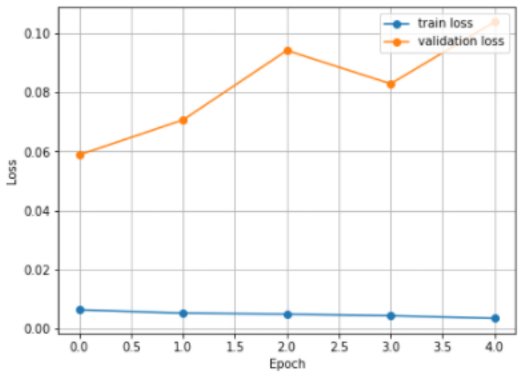


```
313/313 [==============================] - 1s 3ms/step - loss: 0.0385 - accuracy: 0.9929

Test Accuracy: 0.992900013923645

Test Loss: 0.03845653310418129
```

C. RMSProp:

```
Epoch 1/5
1500/1500 [==============================] - 9s 5ms/step - loss: 0.0065 - accuracy: 0.9981 - val_loss: 0.0589 - val_accuracy:
0.9912
Epoch 2/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0053 - accuracy: 0.9986 - val_loss: 0.0707 - val_accuracy:
0.9908
Epoch 3/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0050 - accuracy: 0.9987 - val_loss: 0.0942 - val_accuracy:
0.9888
Epoch 4/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0045 - accuracy: 0.9987 - val_loss: 0.0829 - val_accuracy:
0.9907
Epoch 5/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0036 - accuracy: 0.9990 - val_loss: 0.1039 - val_accuracy:
0.9886
```





```
313/313 [==============================] - 1s 3ms/step - loss: 0.0995 - accuracy: 0.9889

Test Accuracy: 0.9889000058174133

Test Loss: 0.09949786961078644
```

Among them, the RMSProp optimizer gave the best result.

III. DISCUSSION:

All the results and information given were correct and genuine. But, there are some limitations. The accuracy can be higher. If there was more epochs, there is possibility of getting less accuracy. To run the program, an NVidia GPU is required.