Implementing CNN architecture to classify the MNIST dataset with over 98% accuracy

Abstract:

Convolutional neural network (CNN) architecture commonly applied to analyze visual imagery. It can be used to classify a certain image through a big dataset. In this case MNIST dataset were used which contain handwritten number from one to nine. The goal of this project is to implement CNN to classify the MNIST dataset with over 98% accuracy. In order to peruse that some modification in the sequential model were being made. In the sequential model some functions like 'Convo2d', 'Maxpooling2D', 'Flatten', 'Padding', 'Dropout' was used to achieve 98% accuracy. Different optimizer like 'Adam','SGD','RSMprop' was tested to bring out the best accurate percentage and loss percentage. After much modification and testing of 3 different optimizers in the code the target of 98% accuracy was attained.

Introduction:

In deep learning, a convolutional neural network **(**CNN**)** is a class of deep neural networks, most commonly applied to analyze visual imagery. The primary purpose for a convolutional layer is to detect features such as edges, lines, blobs of color, and other visual elements. The filters can detect these features. The more filters that we give to a convolutional layer, the more features it can detect. The hidden layer consists of these filters. And these layers can be manipulated bring out the best results.

my task was to modify an existing code of a different dataset. For this particular task I have used MNIST dataset. This contains handwritten numbers from zero to nine of sixty thousand data. Among them fifty thousand data were training data which I have used. Then come the sequential model which is of most important. Here I did some modification to the existing code to get the highest accuracy possible. After giving the input with shape I have used the Conv2D function for filter process which have the kernel function of (3,3). I added padding function for that extra layer of filter and activation function was 'relu'. Then Maxpooling2D, Dropout and flatten function were used. After that output layer were inserted which had activation 'softmax' and dense of unit 10. On the model compile we tested some optimizer for the best accurate result. For that I have tested optimizers 'Adam', 'SGD', 'RMSprop' which gave different results. Then I trained all the data to the see their accuracy and plotted a figure to picture the graph. Finally, we checked the accuracy of the test data set in order to differentiate between the trin and test data set.

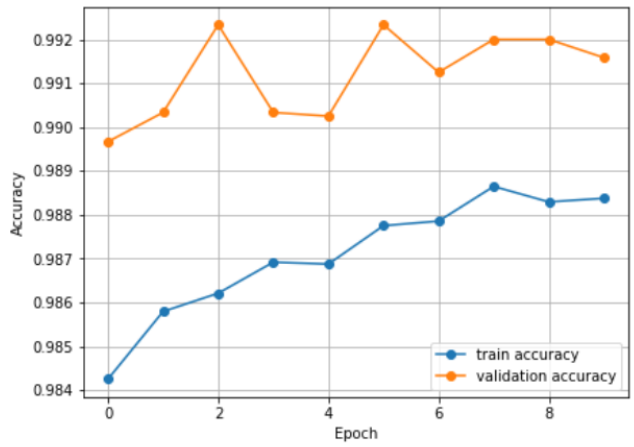
Result:

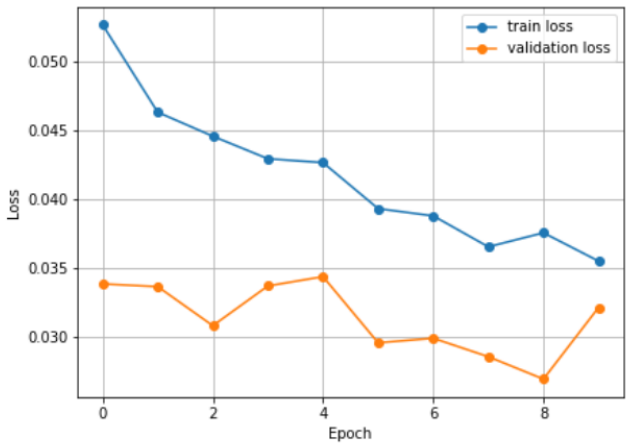
Adam: At first, I tested the 'Adam' optimizer with epoch number 10 and validation split of 0.2:

Process of training and accuracy of training data set:

```
Epoch 1/10
1500/1500 [==============================] - 51s 34ms/step - loss: 0.0527 - accuracy: 0.9843 - val_loss: 0.0339 - val_accuracy: 0.9897
Epoch 2/10
1500/1500 [==============================] - 50s 34ms/step - loss: 0.0463 - accuracy: 0.9858 - val_loss: 0.0337 - val_accuracy: 0.9903
Epoch 3/10
1500/1500 [==============================] - 50s 33ms/step - loss: 0.0446 - accuracy: 0.9862 - val_loss: 0.0308 - val_accuracy: 0.9923
Epoch 4/10
1500/1500 [==============================] - 50s 34ms/step - loss: 0.0429 - accuracy: 0.9869 - val_loss: 0.0337 - val_accuracy: 0.9903
Epoch 5/10
1500/1500 [==============================] - 50s 34ms/step - loss: 0.0427 - accuracy: 0.9869 - val_loss: 0.0344 - val_accuracy: 0.9902
Epoch 6/10
1500/1500 [==============================] - 50s 34ms/step - loss: 0.0393 - accuracy: 0.9877 - val_loss: 0.0296 - val_accuracy: 0.9923
Epoch 7/10
1500/1500 [==============================] - 50s 33ms/step - loss: 0.0388 - accuracy: 0.9879 - val_loss: 0.0299 - val_accuracy: 0.9912
Epoch 8/10
1500/1500 [==============================] - 51s 34ms/step - loss: 0.0366 - accuracy: 0.9886 - val_loss: 0.0286 - val_accuracy: 0.9920
Epoch 9/10
1500/1500 [==============================] - 50s 34ms/step - loss: 0.0376 - accuracy: 0.9883 - val_loss: 0.0270 - val_accuracy: 0.9920
Epoch 10/10
1500/1500 [==============================] - 51s 34ms/step - loss: 0.0355 - accuracy: 0.9884 - val_loss: 0.0321 - val_accuracy: 0.9916
```

Graph:





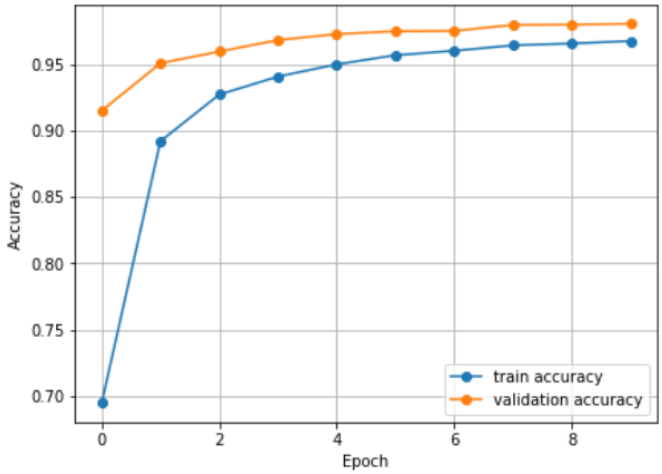
Accuracy and loss of test data set using 'Adam':

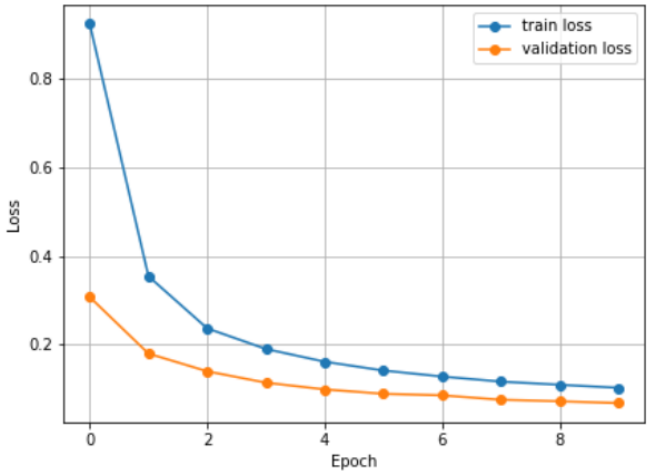
```
313/313 [==============================] - 2s 5ms/step - loss: 0.0315 - accuracy: 0.9906

Test Accuracy: 0.9905999898910522

Test Loss: 0.0314968042075634
```

SGD: then, I tested the 'SGD' optimizer with epoch number 10 and validation split of 0.2:





Accuracy of training dataset:

```
Epoch 9/10
1500/1500 [==============================] - 53s 35ms/step - loss: 0.1147 - accuracy: 0.9634 - val_loss: 0.0689 - val_accuracy: 0.9810
Epoch 10/10
1500/1500 [==============================] - 53s 35ms/step - loss: 0.1060 - accuracy: 0.9668 - val_loss: 0.0661 - val_accuracy: 0.9808
```

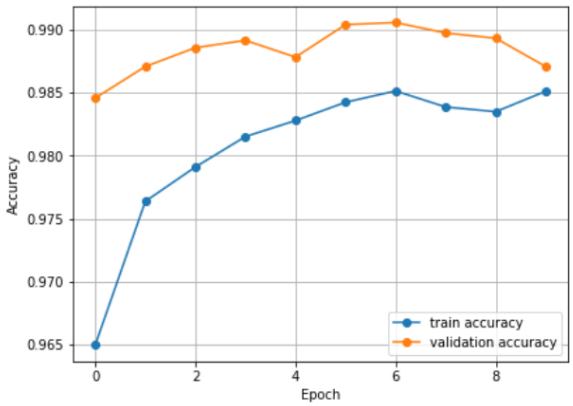
Accuracy and loss of test data set using 'SGD':

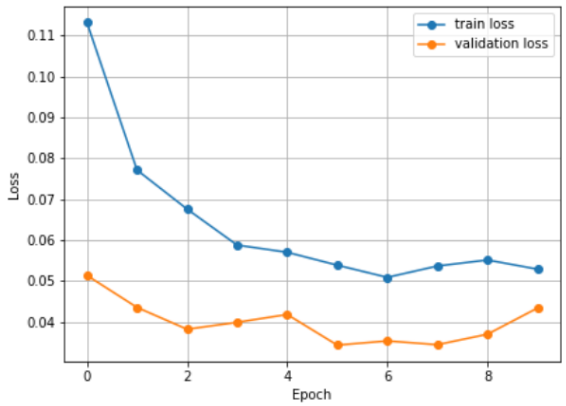
```
313/313 [==============================] - 2s 5ms/step - loss: 0.0266 - accuracy: 0.9915

Test Accuracy: 0.9915000200271606

Test Loss: 0.02661772072315216
```

RMSprop: lastly, I tested the 'RMSprop' optimizer with epoch number 10 and validation split of 0.2:





Accuracy and loss of training data set using ‘RMSprop’:

```
Epoch 9/10
1500/1500 [==============================] - 52s 34ms/step - loss: 0.0551 - accuracy: 0.9835 - val_loss: 0.0370 - val_accuracy: 0.9893
Epoch 10/10
1500/1500 [==============================] - 52s 34ms/step - loss: 0.0529 - accuracy: 0.9851 - val_loss: 0.0434 - val_accuracy: 0.9871
```

Accuracy and loss of test data set using ‘RMSprop’:

```
313/313 [==============================] - 2s 5ms/step - loss: 0.0252 - accuracy: 0.9925

Test Accuracy: 0.9925000071525574

Test Loss: 0.02522377297282219
```

Discussion:

In this project I have implanted CNN architecture to classify MNIST dataset. Since our instructor had thought the code before I just had to do some modification to the code to get the preferred result. The goal was to implement and get the 98% accuracy result. I have also tested three different optimizers to check which have better result.

We see from the result section the training process, the graph and the percentage of accuracy. We also see the val_accuracy which is important because we use to measure the main accuracy weather its correct or not. If the main accuracy results close to val_accuracy then the result is correct. In all three-optimizer testing the validation split was set to 0.2. According to the result section first optimizer ‘Adam’ have accuracy of 98.84%, loss of 3.5% and val_accuracy of 99.16%. since the main result and the val results are very close we can say it’s a correct result. In second optimizer ‘SGD’ we get the accuracy of 96.68% and 98% val_accuracy. The last optimizer ‘RMSprop’ was the lowest one with 98.51 accuracy and loss of 5%. So, we can say from the results that our goal of 98% accuracy was achieved rather spectacularly since all three optimizers had over 98% accuracy. But as we can see from the graphs that three optimizers have different graphs and values. Comparing the three optimizer we can see that optimizer ‘Adam’ have best accuracy which is 98.84. So, we can say that the optimizer ‘Adam” is best.

The other two optimizer were very good too for this particular task since they had over 98% accuracy. During this task I had to introduce myself to some new function like dropout and padding which made the project a success. At first, I tried using batches and epoch=5 to get the result later I used different epoch number to get the best results. Overall, after much try and research the project became a success.