Mid Project Report

Computer Vision & Pattern Recognition

Name: Osmani, Sabbir Ahmmed

ID: 18-37313-1

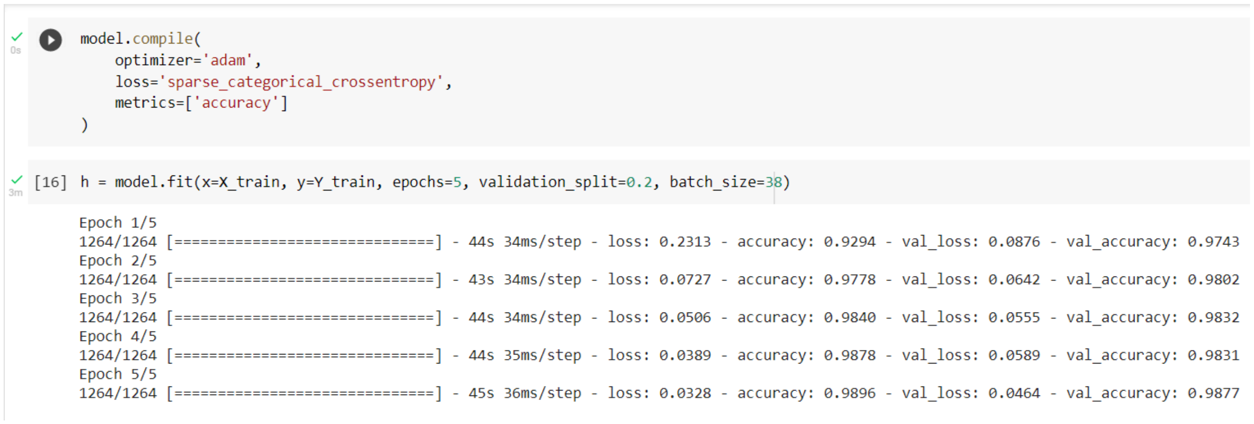
Sec: B

Fall 2021-2022

Abstract: In this report I am going to discuss about implementing a CNN architecture to classify the MNIST handwritten dataset. Also I used different optimizer like Adam, SGD, RSMProp to check which one gives best accuracy.

Introduction: Optimizers are techniques or approaches that adjust the characteristics of your neural network, such as weights and learning rate, to decrease losses. Optimization algorithms or methods are in charge of lowering losses and delivering the most accurate outcomes. Optimizers are algorithms or techniques for changing the properties of your neural network, such as weights and learning rate, in order to decrease losses.... Optimization algorithms or strategies are in charge of decreasing losses and providing the most accurate results feasible. As earlier I said I will use 3 type of optimizer and now I will discuss about them: Adam: Adam is a deep learning model training technique that replaces stochastic gradient descent. Adam combines the finest features of the AdaGrad and RMSProp methods to provide an optimization technique for noisy issues with sparse gradients. SGD: SGD is an iterative approach for finding the best smoothness qualities for an objective function. One popular and persuasive argument for optimizers is that SGD generalizes better than Adam. RMSProp: Root Mean Square Propagation is abbreviated as RMSprop. In neural network training, RMSprop is a gradient-based optimization strategy.

Result: In this portion I will discuss about my result/output of my project:

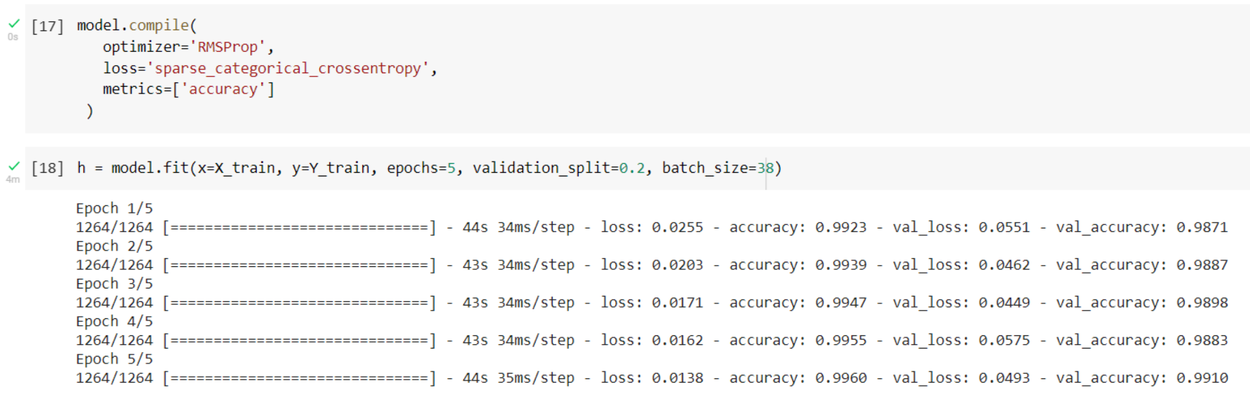


```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

```
[16] h = model.fit(x=X_train, y=Y_train, epochs=5, validation_split=0.2, batch_size=38)
```

```
Epoch 1/5
1264/1264 [==============================] - 44s 34ms/step - loss: 0.2313 - accuracy: 0.9294 - val_loss: 0.0876 - val_accuracy: 0.9743
Epoch 2/5
1264/1264 [==============================] - 43s 34ms/step - loss: 0.0727 - accuracy: 0.9778 - val_loss: 0.0642 - val_accuracy: 0.9802
Epoch 3/5
1264/1264 [==============================] - 44s 34ms/step - loss: 0.0506 - accuracy: 0.9840 - val_loss: 0.0555 - val_accuracy: 0.9832
Epoch 4/5
1264/1264 [==============================] - 44s 35ms/step - loss: 0.0389 - accuracy: 0.9878 - val_loss: 0.0589 - val_accuracy: 0.9831
Epoch 5/5
1264/1264 [==============================] - 45s 36ms/step - loss: 0.0328 - accuracy: 0.9896 - val_loss: 0.0464 - val_accuracy: 0.9877
```

In this picture you can see I used optimizer as Adam and my accuracy is about 0.98+ and loss is about 0.05+

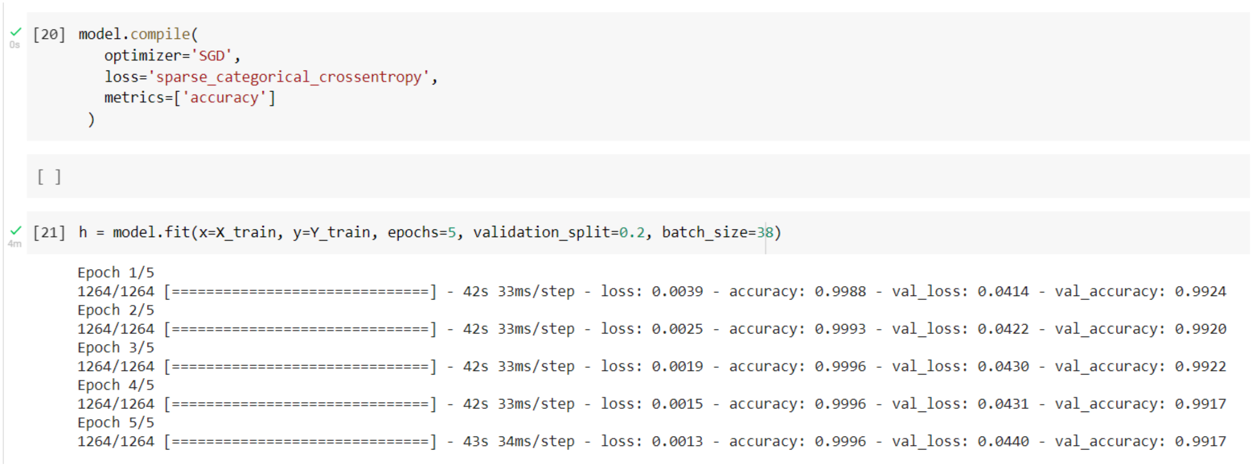


```
[17] model.compile(
       optimizer='RMSProp',
       loss='sparse_categorical_crossentropy',
       metrics=['accuracy']
     )
```

```
[18] h = model.fit(x=X_train, y=Y_train, epochs=5, validation_split=0.2, batch_size=38)

     Epoch 1/5
     1264/1264 [==============================] - 44s 34ms/step - loss: 0.0255 - accuracy: 0.9923 - val_loss: 0.0551 - val_accuracy: 0.9871
     Epoch 2/5
     1264/1264 [==============================] - 43s 34ms/step - loss: 0.0203 - accuracy: 0.9939 - val_loss: 0.0462 - val_accuracy: 0.9887
     Epoch 3/5
     1264/1264 [==============================] - 43s 34ms/step - loss: 0.0171 - accuracy: 0.9947 - val_loss: 0.0449 - val_accuracy: 0.9898
     Epoch 4/5
     1264/1264 [==============================] - 43s 34ms/step - loss: 0.0162 - accuracy: 0.9955 - val_loss: 0.0575 - val_accuracy: 0.9883
     Epoch 5/5
     1264/1264 [==============================] - 44s 35ms/step - loss: 0.0138 - accuracy: 0.9960 - val_loss: 0.0493 - val_accuracy: 0.9910
```

In this picture you can see I used RMSProp as the optimizer and the accuracy was also like Adam which is 0.98+ but the loss is more than Adam which is (0.04- 0.07)+



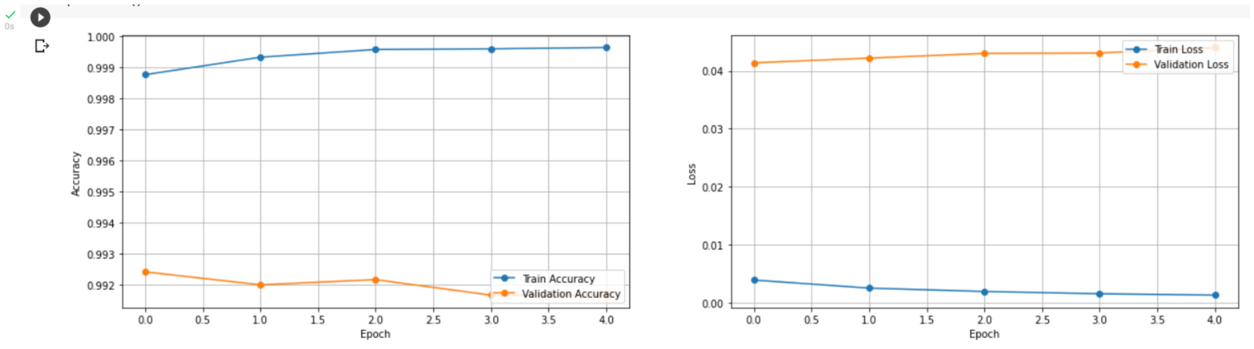
```
[20] model.compile(
       optimizer='SGD',
       loss='sparse_categorical_crossentropy',
       metrics=['accuracy']
     )
```

```
[ ]
```

```
[21] h = model.fit(x=X_train, y=Y_train, epochs=5, validation_split=0.2, batch_size=38)

     Epoch 1/5
     1264/1264 [==============================] - 42s 33ms/step - loss: 0.0039 - accuracy: 0.9988 - val_loss: 0.0414 - val_accuracy: 0.9924
     Epoch 2/5
     1264/1264 [==============================] - 42s 33ms/step - loss: 0.0025 - accuracy: 0.9993 - val_loss: 0.0422 - val_accuracy: 0.9920
     Epoch 3/5
     1264/1264 [==============================] - 42s 33ms/step - loss: 0.0019 - accuracy: 0.9996 - val_loss: 0.0430 - val_accuracy: 0.9922
     Epoch 4/5
     1264/1264 [==============================] - 42s 33ms/step - loss: 0.0015 - accuracy: 0.9996 - val_loss: 0.0431 - val_accuracy: 0.9917
     Epoch 5/5
     1264/1264 [==============================] - 43s 34ms/step - loss: 0.0013 - accuracy: 0.9996 - val_loss: 0.0440 - val_accuracy: 0.9917
```

In this output you can see I use SGD as the optimizer and its gives the best accuracy which is 0.991100013256073 and loss is 0.04486643522977829. This the best accuracy which you can see in below.



```
[23] test_loss, test_acc = model.evaluate(X_test, Y_test)
     print('\nTest Accuracy:', test_acc)
     print('\nTest Loss:', test_loss)

     313/313 [==============================] - 3s 8ms/step - loss: 0.0324 - accuracy: 0.9922

     Test Accuracy: 0.9922000169754028

     Test Loss: 0.0323898121714592
```

So after using all the optimizer SGD gives the best output.

Discussion: So after using all these, I can say according to my test Adam gives a good result and RMSProp also gives a better result but the loss was more than Adam. But, at the last I used SGD which gives me a better result which have accuracy of 0.99+ and loss about 0.04+. So For me in this project SGD was the best solution. And SGD is faster than Adam and RMSProp.