Mid Project Report

Md. Zahidul Haque

ID: 18-36130-1

Subject : Computer Vision and Pattern Recognition
Section-B

Abstract:

In this report I have implemented a CNN architecture to classify the MNIST handwritten dataset. Here al different optimizer like Adam, SGD, RSMProp has been used to check which one gives best accuracy.

Introduction:

Optimizers can be explained as a mathematical function to modify the weights of the network given the gradients and additional information, depending on the formulation of the optimizer. Optimizers are built upon the idea of gradient descent, the greedy approach of iteratively decreasing the loss function by following the gradient.

Such functions can be as simple as subtracting the gradients from the weights, or can also be very complex.

Better optimizers are mainly focused on being faster and efficient but are also often known to generalize well(less overfitting) compared to others

Adam: Adam is an optimization technique that is used to update network weights iteratively based on training data instead of the traditional stochastic gradient descentprocedure.

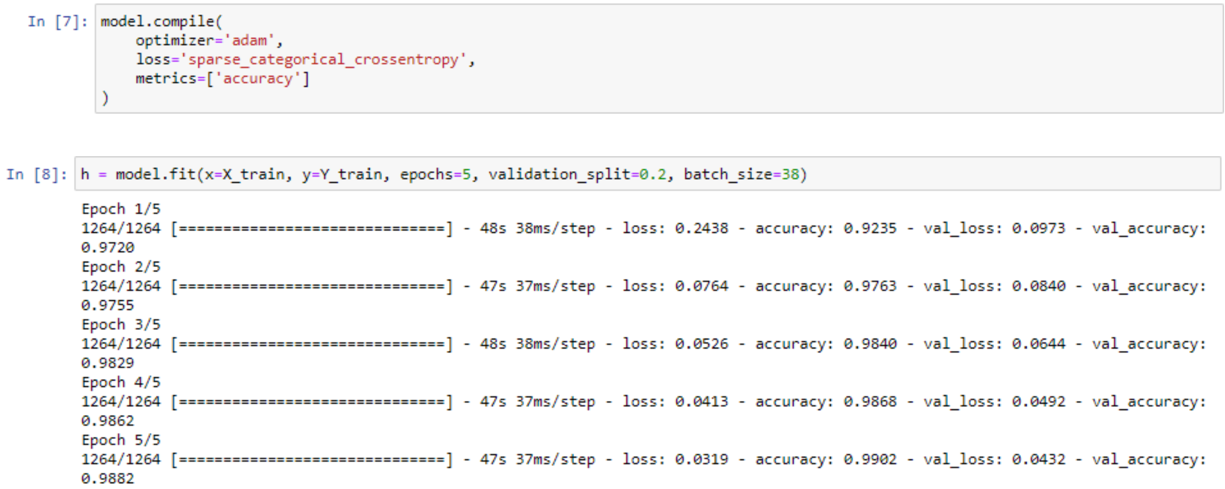
for noisy issues with sparse gradients.

SGD: SGD is a variant of gradient descent. Instead of performing computations on the whole dataset — which is redundant and inefficient — SGD only computes on a small subset or random selection of data examples. SGD produces the same performance as regular gradient descent when the learning rate is low..

RMSProp: RSMProp is a gradient-based optimization strategy. Gradients in particularly complicated functions, such as neural networks, have a propensity to evaporate or explode as input passes through them. RSMProp was created as a stochastic mini-batch learning algorithm.

Result :

<u>Here is the result :</u>

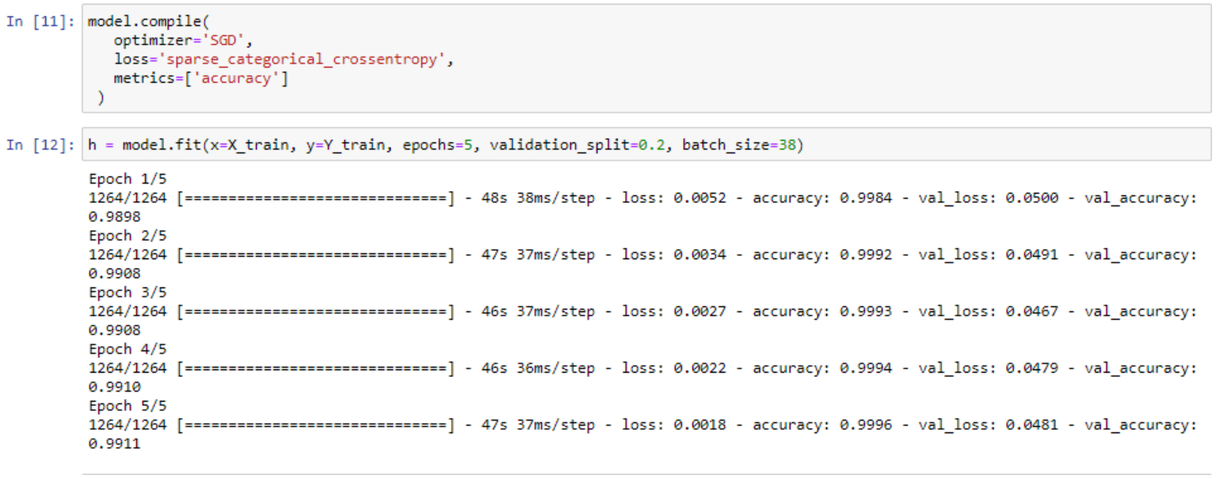


```
In [7]: model.compile(
            optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy']
        )
```

```
In [8]: h = model.fit(x=X_train, y=Y_train, epochs=5, validation_split=0.2, batch_size=38)

        Epoch 1/5
        1264/1264 [==============================] - 48s 38ms/step - loss: 0.2438 - accuracy: 0.9235 - val_loss: 0.0973 - val_accuracy:
        0.9720
        Epoch 2/5
        1264/1264 [==============================] - 47s 37ms/step - loss: 0.0764 - accuracy: 0.9763 - val_loss: 0.0840 - val_accuracy:
        0.9755
        Epoch 3/5
        1264/1264 [==============================] - 48s 38ms/step - loss: 0.0526 - accuracy: 0.9840 - val_loss: 0.0644 - val_accuracy:
        0.9829
        Epoch 4/5
        1264/1264 [==============================] - 47s 37ms/step - loss: 0.0413 - accuracy: 0.9868 - val_loss: 0.0492 - val_accuracy:
        0.9862
        Epoch 5/5
        1264/1264 [==============================] - 47s 37ms/step - loss: 0.0319 - accuracy: 0.9902 - val_loss: 0.0432 - val_accuracy:
        0.9882
```

With an accuracy of 98.79% and a loss of 4%.

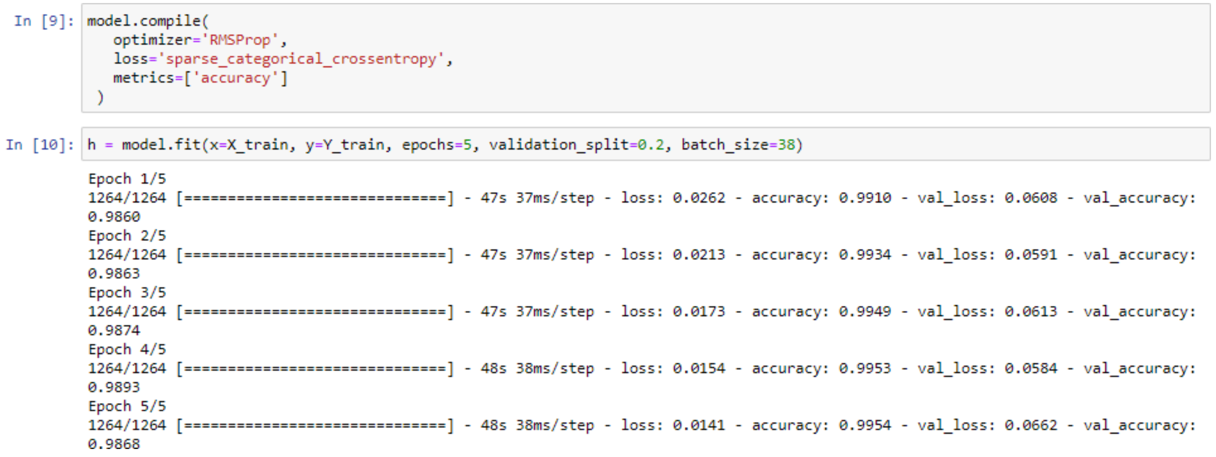


```
In [11]: model.compile(
            optimizer='SGD',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy']
          )
```

```
In [12]: h = model.fit(x=X_train, y=Y_train, epochs=5, validation_split=0.2, batch_size=38)
```

```
Epoch 1/5
1264/1264 [==============================] - 48s 38ms/step - loss: 0.0052 - accuracy: 0.9984 - val_loss: 0.0500 - val_accuracy:
0.9898
Epoch 2/5
1264/1264 [==============================] - 47s 37ms/step - loss: 0.0034 - accuracy: 0.9992 - val_loss: 0.0491 - val_accuracy:
0.9908
Epoch 3/5
1264/1264 [==============================] - 46s 37ms/step - loss: 0.0027 - accuracy: 0.9993 - val_loss: 0.0467 - val_accuracy:
0.9908
Epoch 4/5
1264/1264 [==============================] - 46s 36ms/step - loss: 0.0022 - accuracy: 0.9994 - val_loss: 0.0479 - val_accuracy:
0.9910
Epoch 5/5
1264/1264 [==============================] - 47s 37ms/step - loss: 0.0018 - accuracy: 0.9996 - val_loss: 0.0481 - val_accuracy:
0.9911
```

With an accuracy of 99.89% and a loss of 4%.



```
In [9]: model.compile(
           optimizer='RMSProp',
           loss='sparse_categorical_crossentropy',
           metrics=['accuracy']
         )
```

```
In [10]: h = model.fit(x=X_train, y=Y_train, epochs=5, validation_split=0.2, batch_size=38)
```

```
Epoch 1/5
1264/1264 [==============================] - 47s 37ms/step - loss: 0.0262 - accuracy: 0.9910 - val_loss: 0.0608 - val_accuracy:
0.9860
Epoch 2/5
1264/1264 [==============================] - 47s 37ms/step - loss: 0.0213 - accuracy: 0.9934 - val_loss: 0.0591 - val_accuracy:
0.9863
Epoch 3/5
1264/1264 [==============================] - 47s 37ms/step - loss: 0.0173 - accuracy: 0.9949 - val_loss: 0.0613 - val_accuracy:
0.9874
Epoch 4/5
1264/1264 [==============================] - 48s 38ms/step - loss: 0.0154 - accuracy: 0.9953 - val_loss: 0.0584 - val_accuracy:
0.9893
Epoch 5/5
1264/1264 [==============================] - 48s 38ms/step - loss: 0.0141 - accuracy: 0.9954 - val_loss: 0.0662 - val_accuracy:
0.9868
```

With an accuracy of 98.79% and a loss of 4%.

Discussion: Firstly Adam was utilized, with an accuracy of 98.79 percent and a loss of 4%. In SGD accuracy was 99.29 percent and the loss were 4%. percent Finally, after applying RSMProp, the accuracy was 98.79% and the loss was 4%.SGD was the best among all.