AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Course: COMPUTER VISION AND PATTERN RECOGNITION

Course Teacher: - DEBAJYOTI KARMAKER

Section: A

Mid Project Report

Name	ID
Md. Arman Joarder	18-37882-2

Abstract:

In this report I am going to discuss about implementing a CNN architecture to classify the MNIST handwritten dataset. As handwriting differ from person to person. The goal of this project is to propose a CNN (Convolutional Neural Network) model for MNIST dataset which will produce an accuracy over 98%. Three different optimizers (Adam, SGD, RMSprop) will be used to get the best result possible.

Introduction:

The Convolutional Neural Network (CNN) is a type of artificial neural network. Which is used in image processing and recognition. 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']
)
```

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

```
Epoch 1/5
1264/1264 [==============================] - 20s 16ms/step - loss: 0.2306 - accuracy: 0.9275 - val_loss: 0.1040 - val_accuracy:
0.9682
Epoch 2/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0709 - accuracy: 0.9778 - val_loss: 0.0563 - val_accuracy:
0.9837
Epoch 3/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0509 - accuracy: 0.9839 - val_loss: 0.0605 - val_accuracy:
0.9820
Epoch 4/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0385 - accuracy: 0.9881 - val_loss: 0.0610 - val_accuracy:
0.9817
Epoch 5/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0298 - accuracy: 0.9909 - val_loss: 0.0504 - val_accuracy:
0.9866
```

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

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0458 - accuracy: 0.9859

Test Accuracy: 0.9858999848365784

Test Loss: 0.04584516957402229
```

In this picture you can see I used optimizer as Adam and my accuracy is about 0.985 And loss is about 0.045

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

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

```
Epoch 1/5
1264/1264 [==============================] - 20s 16ms/step - loss: 0.0109 - accuracy: 0.9970 - val_loss: 0.0634 - val_accuracy:
0.9883
Epoch 2/5
1264/1264 [==============================] - 21s 17ms/step - loss: 0.0114 - accuracy: 0.9969 - val_loss: 0.0654 - val_accuracy:
0.9883
Epoch 3/5
1264/1264 [==============================] - 21s 17ms/step - loss: 0.0104 - accuracy: 0.9972 - val_loss: 0.0835 - val_accuracy:
0.9874
Epoch 4/5
1264/1264 [==============================] - 21s 17ms/step - loss: 0.0099 - accuracy: 0.9976 - val_loss: 0.0739 - val_accuracy:
0.9914
Epoch 5/5
1264/1264 [==============================] - 21s 17ms/step - loss: 0.0090 - accuracy: 0.9975 - val_loss: 0.0708 - val_accuracy:
0.9906
```

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

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0599 - accuracy: 0.9905

Test Accuracy: 0.9904999732971191

Test Loss: 0.059853386133909225
```

In this picture you can see I used RMSProp as the optimizer and the accuracy was also like Adam which is 0.990 but the loss is more than Adam which is 0.05.

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

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

```
Epoch 1/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0039 - accuracy: 0.9990 - val_loss: 0.0494 - val_accuracy:
0.9909
Epoch 2/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0025 - accuracy: 0.9994 - val_loss: 0.0485 - val_accuracy:
0.9912
Epoch 3/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0021 - accuracy: 0.9996 - val_loss: 0.0479 - val_accuracy:
0.9911
Epoch 4/5
1264/1264 [==============================] - 23s 18ms/step - loss: 0.0018 - accuracy: 0.9997 - val_loss: 0.0479 - val_accuracy:
0.9912
Epoch 5/5
1264/1264 [==============================] - 22s 18ms/step - loss: 0.0015 - accuracy: 0.9997 - val_loss: 0.0480 - val_accuracy:
0.9911
```

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

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0411 - accuracy: 0.9916

Test Accuracy: 0.991599977016449

Test Loss: 0.04110991954803467
```

In this output you can see I use SGD as the optimizer and its gives the best accuracy which is 0.991 and loss is 0.041. This the best accuracy which you can see in below.

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 0.991 and loss about 0.041. So For me in this project SGD was the best solution. And SGD is faster than Adam and RMSProp. Considering all the data, it can be concluded that we can get the best output from this model by using SGD optimizer. It gives the highest training and testing accuracy with the lowest validation loss.