AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

(AIUB)

Course: COMPUTER VISION AND PATTERN RECOGNITION

Course Teacher: - DEBAJYOTI KARMAKER

Section: A

CVPR Mid Project Report

Name	**ID**
Maliha Tasnim Ananna	18-37895-2

Abstract:

The Convolutional Neural Network gained popularity through its use with image data and is currently the state of the art for detecting what an image is. In this report I am going to describe about implementing a CNN architecture to classify the MNIST handwritten dataset. 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. A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data. CNNs have been used for understanding in Natural Language Processing (NLP) and speech recognition, although often for NLP Recurrent Neural Nets (RNNs) are used. 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 [==============================] - 22s 17ms/step - loss: 0.2234 - accuracy: 0.9315 - val_loss: 0.1491 - val_accuracy:
0.9533
Epoch 2/5
1264/1264 [==============================] - 22s 18ms/step - loss: 0.0743 - accuracy: 0.9774 - val_loss: 0.0545 - val_accuracy:
0.9843
Epoch 3/5
1264/1264 [==============================] - 23s 18ms/step - loss: 0.0517 - accuracy: 0.9839 - val_loss: 0.0595 - val_accuracy:
0.9827
Epoch 4/5
1264/1264 [==============================] - 24s 19ms/step - loss: 0.0392 - accuracy: 0.9880 - val_loss: 0.0440 - val_accuracy:
0.9882
Epoch 5/5
1264/1264 [==============================] - 24s 19ms/step - loss: 0.0318 - accuracy: 0.9900 - val_loss: 0.0487 - val_accuracy:
0.9857
```

```
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: 0.0439 - accuracy: 0.9867

Test Accuracy: 0.9866999983787537

Test Loss: 0.04394633322954178
```

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

0.986 And loss is about 0.043

```
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 [==============================] - 22s 17ms/step - loss: 0.0238 - accuracy: 0.9924 - val_loss: 0.0432 - val_accuracy:
0.9887
Epoch 2/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0208 - accuracy: 0.9935 - val_loss: 0.0482 - val_accuracy:
0.9873
Epoch 3/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0170 - accuracy: 0.9949 - val_loss: 0.0457 - val_accuracy:
0.9896
Epoch 4/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0148 - accuracy: 0.9958 - val_loss: 0.0577 - val_accuracy:
0.9868
Epoch 5/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0131 - accuracy: 0.9962 - val_loss: 0.0566 - val_accuracy:
0.9895
```

```
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: 0.0590 - accuracy: 0.9888

Test Accuracy: 0.9887999892234802

Test Loss: 0.05900608375668526
```

In this picture you can see I used RMSProp as the optimizer and the accuracy was also like Adam which is 0.988 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: 0.0051 - accuracy: 0.9985 - val_loss: 0.0484 - val_accuracy:
0.9913
Epoch 2/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0030 - accuracy: 0.9991 - val_loss: 0.0468 - val_accuracy:
0.9914
Epoch 3/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0023 - accuracy: 0.9994 - val_loss: 0.0460 - val_accuracy:
0.9921
Epoch 4/5
1264/1264 [==============================] - 22s 17ms/step - loss: 0.0019 - accuracy: 0.9995 - val_loss: 0.0461 - val_accuracy:
0.9924
Epoch 5/5
1264/1264 [==============================] - 23s 18ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 0.0464 - val_accuracy:
0.9920
```

```
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: 0.0413 - accuracy: 0.9911

Test Accuracy: 0.991100013256073

Test Loss: 0.04133503884077072
```

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.