Implement a CNN architecture to classify the MNIST handwritten dataset.

NAME: HURAIYA AFROSE

ID:19-39465-1

SEC:A

Abstract: The aim of this paper is developed an application of convolution neural network for image classification problem. Here I used MNIST dataset to test the performance of CNN model. There are many algorithms developed in the area of computer vision to recognize images. Here the goal of our work will be to create a model that will be able to determine and identify the handwritten digit from the MNIST dataset with better accuracy. To conduct this experiment, I use here activation function, optimizer, learning rate etc. Here I use three types of optimizers Adam, SGD, RSMProp. Optimizer rate or epochs has impact on accuracy. In activation function results are shown. We get more than 98% accuracy using these three optimizers.

Introduction: 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, or what is contained in the image. Hand written recognition domain is an active area of research hand written digit recognition. We solve this identification using Convolution Neural Network. Convolution neural network is similar to artificial neural network said by O'Shea & Nash (2015). There are many kinds of layers in CNN like convolution layer, pooling layer, fully connected layer. 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. I use here three types of optimizer Adam, SGD, RSMProp. The details of these three-optimizer given below:

ADAM Optimizer

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. We can get best accuracy from the ADAM optimizer [1]. It can handle sparse gradients on noisy problems.

SGD Optimizer

Stochastic gradient descent is an iterative method for optimizing an objective function with suitable smoothness properties [2].

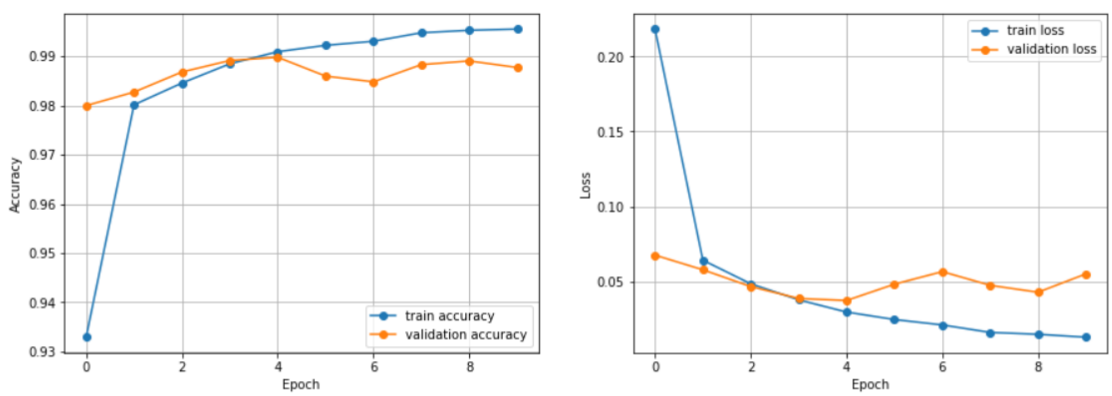
RSMProp Optimizer

RSMProp full meaning Root Mean Squared Propagation. It is an extension of gradient descent and the AdaGrad version of gradient descent that uses a decaying average of partial gradients in the adaptation of the step size for each parameter [3].

Result

Using ADAM Optimizer

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




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

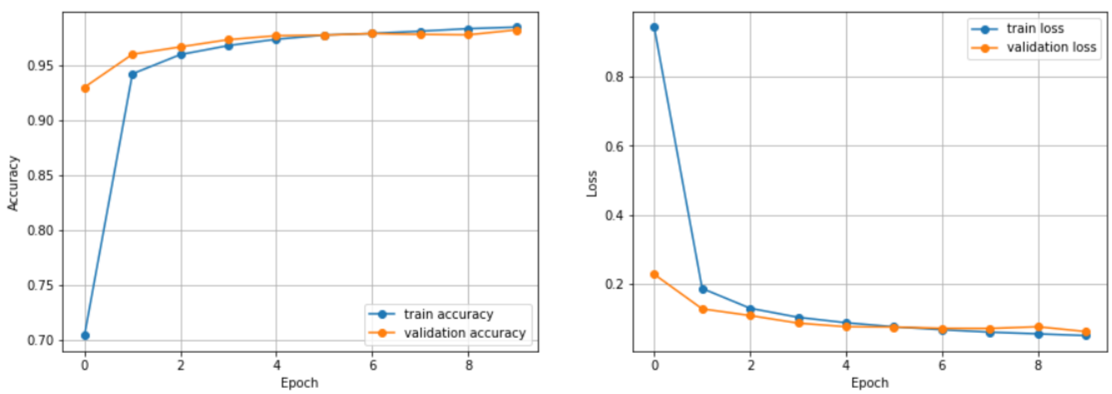
```
313/313 [==============================] - 1s 4ms/step - loss: 0.0543 - accuracy: 0.9874

Test Accuracy: 0.9873999953269958

Test Loss: 0.05432631075382233
```

This is the result of ADAM optimizer. Here I get test Accuracy 98.73% and Test loss 5.43%.

Using SGD Optimizer

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

```
[26]  plt.show()
```



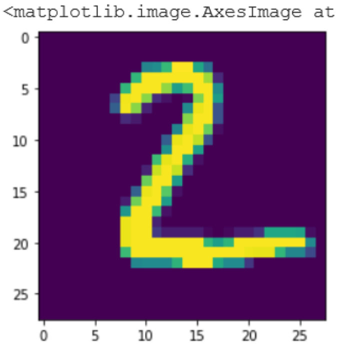

```python
test_loss, test_acc = model.evaluate(x_test, y_test)
print('\nTest Accuracy:', test_acc)
print('\nTest Loss:', test_loss)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0507 - accuracy: 0.9852

Test Accuracy: 0.9851999878883362

Test Loss: 0.05066604167222977
```

```python
[48] x_train=x_train=x_train.reshape(-1,28,28)
     x_test=x_test=x_test.reshape(-1,28,28)
```

```python
plt.imshow(x_test[1])
```

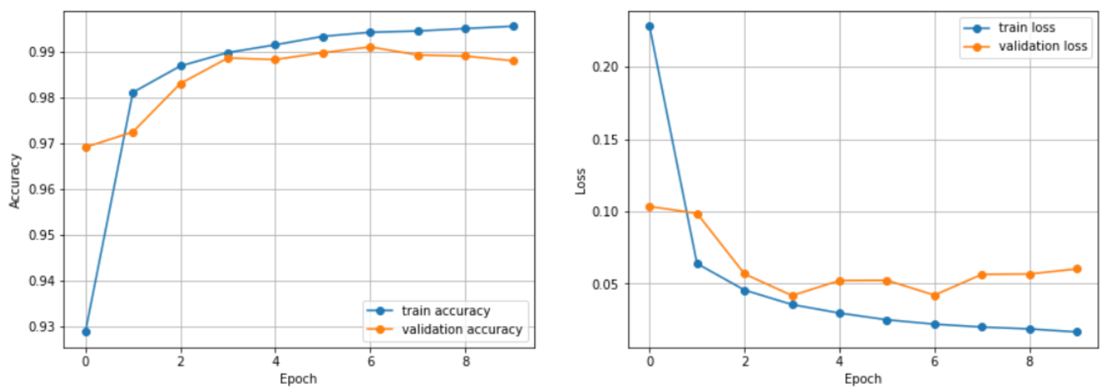
```
<matplotlib.image.AxesImage at 0x7f4b8a07e9d0>
```



This is the result of ADAM optimizer. Here I get test Accuracy 98.52% and Test loss 5.06%. prediction and test image is same in the result section.

Using RSMProp Optimizer

[62]




```python
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: 0.0579 - accuracy: 0.9884

Test Accuracy: 0.9883999824523926

Test Loss: 0.05794122815132141
```

```python
h = model.fit(x=x_train, y=y_train, epochs=10, validation_split=0.2, batch_size=32)
```

```
Epoch 1/10
1500/1500 [==============================] - 13s 8ms/step - loss: 0.2282 - accuracy: 0.9287 - val_loss: 0.1034 - val_accuracy: 0.9691
Epoch 2/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0638 - accuracy: 0.9811 - val_loss: 0.0986 - val_accuracy: 0.9724
Epoch 3/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0455 - accuracy: 0.9868 - val_loss: 0.0567 - val_accuracy: 0.9830
Epoch 4/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0355 - accuracy: 0.9898 - val_loss: 0.0417 - val_accuracy: 0.9886
Epoch 5/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0296 - accuracy: 0.9915 - val_loss: 0.0521 - val_accuracy: 0.9883
Epoch 6/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0249 - accuracy: 0.9933 - val_loss: 0.0523 - val_accuracy: 0.9898
Epoch 7/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0219 - accuracy: 0.9942 - val_loss: 0.0420 - val_accuracy: 0.9910
Epoch 8/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0199 - accuracy: 0.9945 - val_loss: 0.0564 - val_accuracy: 0.9893
Epoch 9/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0186 - accuracy: 0.9950 - val_loss: 0.0566 - val_accuracy: 0.9890
Epoch 10/10
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0165 - accuracy: 0.9955 - val_loss: 0.0601 - val_accuracy: 0.9880
```

This is the result of RSMProp optimizer. Here I get test Accuracy 98.84% and Test loss 5.79%.

Discussion: paper introduced three types of optimizers to solve image classification problem on MNISY hand written dataset. At first, I used ADAM optimizer and it give 98.73% accuracy for MNIST dataset. Using SGD, I get Accuracy 98.52% and Test loss 5.06% and from RSMProp optimizer I get test Accuracy 98.84% and Test loss 5.79%. Comparing three optimizer I get maximum accuracy from RMSProp optimizer then ADAM and minimum accuracy from SGD optimizer. So now we can say that from three optimizer RSMProp is give us faster and highly accuracy.

REFERENCE:

[1]. https://keras.io/api/optimizers/adam/

[2]. https://en.wikipedia.org/wiki/Stochastic_gradient_descent

[3]. https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/