COMPUTER VISION AND PATTERN RECOGNITION

Project Report

Mid Term

Title: Implement a CNN architecture to classify the MNIST handwritten dataset and Test with different optimizer

Submitted By:

Shorif Ahmed Afridi Mozumder

ID: 18-38048-2

CSE

Section: B

Course Instructor:

Dr. Debajyoti Karmaker

Associate Professor, Computer Science

Implement a CNN architecture to classify the MNIST handwritten dataset and Test with different optimizer

Abstract: In terms of recognition of hand written text or number for a computer maybe quite challenging if there is not enough amount of prior knowledge. In the field of handwriting recognition, we have come to a very matured time because of training Optical character recognition (OCR). But still, we need to do further research and come up with some really fast and reliable algorithm to make our systems much smoother and more accurate result. In this project I am trying to train and classify the MNIST handwritten dataset implementing Convolutional neural networks (CNN) architecture and aim to achieve at least 98% accuracy. I tried to test with different algorithm like Adam, Adadelta.

Introduction: For multilayer perception and recognition of image, speech, or audio signal inputs the Convolutional neural networks (CNN) is superior among all the neural networks. In this project the MNIST dataset is being train and analyze for the best accuracy in terms of recognize the handwritten information of an Optical character recognition (OCR) system. There are 60000 images set of handwritten digits and 10000 images of train image in the MNIST database. Each of the image is 28*28 pixels with the pixel value of 255. Using these amounts of data with the CNN model and Adadelta and Adam algorithm I tried to execute for recognition of the handwritten number.

Here is the Sequential model which is used to execute algorithm or Optimizer:

mid_project.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
activation_13 (Activation)   (None, 26, 26, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 24, 24, 32)        9248
_________________________________________________________________
activation_14 (Activation)   (None, 24, 24, 32)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 10, 10, 64)        18496
_________________________________________________________________
activation_15 (Activation)   (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
activation_16 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_10 (Dense)             (None, 512)               524800
_________________________________________________________________
activation_17 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_11 (Dense)             (None, 10)                5130
_________________________________________________________________
activation_18 (Activation)   (None, 10)                0
=================================================================
Total params: 594,922
Trainable params: 594,922
Non-trainable params: 0
_________________________________________________________________
```

Fig: Model 1

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 24, 24, 16)        416
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 12, 12, 16)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 8, 8, 32)          12832
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 4, 4, 32)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                32832
_________________________________________________________________
dense_13 (Dense)             (None, 10)                650
=================================================================
Total params: 46,730
Trainable params: 46,730
Non-trainable params: 0
_________________________________________________________________
```

Fig: Model 2

Results:

mid_project.ipynb

Edit View Insert Runtime Tools Help All changes saved

de + Text

```
Epoch 1/12
120/120 [==============================] - 134s 1s/step - loss: 2.2175 - accuracy: 0.3051
Epoch 2/12
120/120 [==============================] - 134s 1s/step - loss: 2.2077 - accuracy: 0.3434
Epoch 3/12
120/120 [==============================] - 134s 1s/step - loss: 2.1975 - accuracy: 0.3785
Epoch 4/12
120/120 [==============================] - 135s 1s/step - loss: 2.1876 - accuracy: 0.4098
Epoch 5/12
120/120 [==============================] - 135s 1s/step - loss: 2.1776 - accuracy: 0.4343
Epoch 6/12
120/120 [==============================] - 134s 1s/step - loss: 2.1678 - accuracy: 0.4550
Epoch 7/12
120/120 [==============================] - 134s 1s/step - loss: 2.1578 - accuracy: 0.4683
Epoch 8/12
120/120 [==============================] - 134s 1s/step - loss: 2.1478 - accuracy: 0.4785
Epoch 9/12
120/120 [==============================] - 134s 1s/step - loss: 2.1373 - accuracy: 0.4882
Epoch 10/12
120/120 [==============================] - 135s 1s/step - loss: 2.1268 - accuracy: 0.4957
Epoch 11/12
120/120 [==============================] - 136s 1s/step - loss: 2.1155 - accuracy: 0.5001
Epoch 12/12
120/120 [==============================] - 138s 1s/step - loss: 2.1040 - accuracy: 0.5046
```

Graph:

```
plt.show()
```

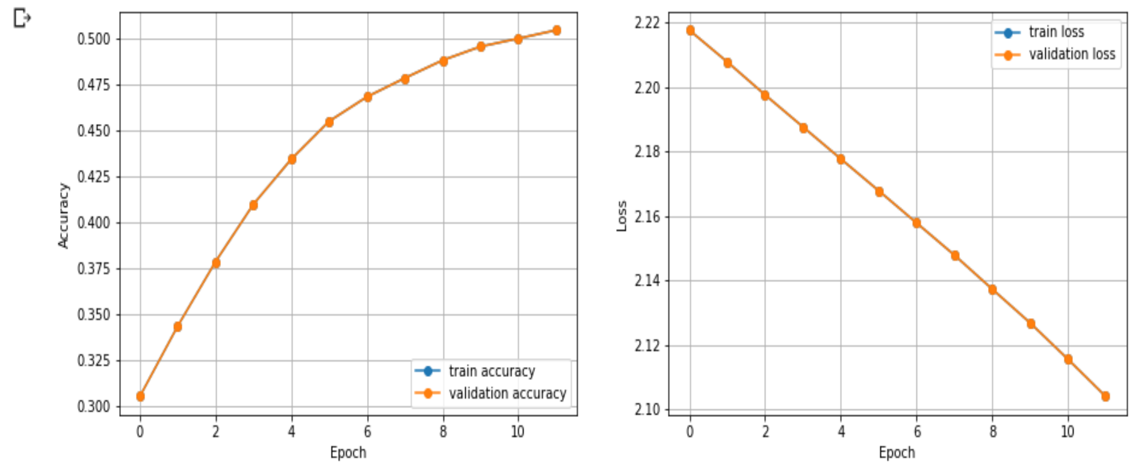

Fig: Plot for Adadelta (Model - 1)

Adam:

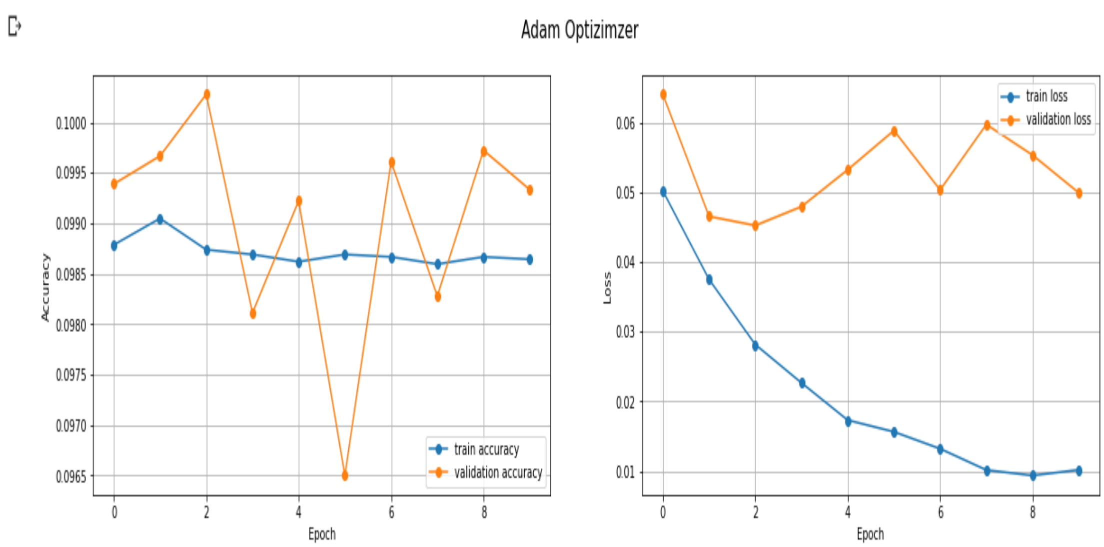

Fig: Plot for Adam (Model - 2)

SGD:

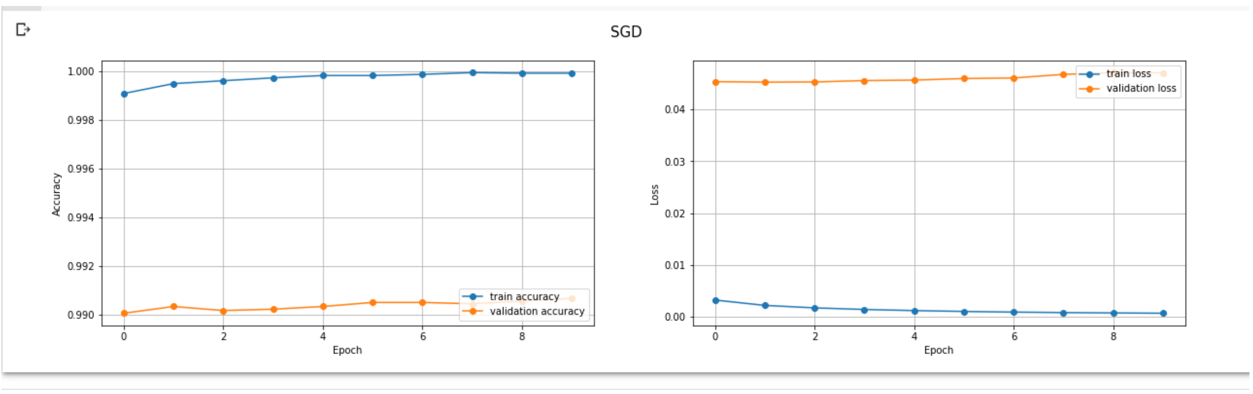

Fig: Plot for SGD (Model - 2)

Optimizer	Accuracy (Train)	Accuracy (Test)
Adadelta	51.03%	50.46%
Adam	98.06%	99.12%
SGD	99.99%	99.07%

Discussion: For Model -1 with the Adadelta Optimizer gives 50.46% accuracy which did not get expected accuracy 98%+. But for the proposed model 2 provides accuracy above 98%. For Adam it gives 99.12% Test accuracy and for SGD 99.07%. Could not test RMSprop optimized but instead I used Adadelta.