Hand Written Digits(MNIST) ML Project

-Pandey Shourya Prasad

Overview of project:

I used the Handwritten digits dataset which had 60,000 training cases and 10,000 testing cases.

```
In [3]: (X_train, y_train),(X_test, y_test)= keras.datasets.mnist.load_data()

In [4]: len(X_train)
Out[4]: 60000

In [28]: len(X_test)
Out[28]: 10000
```

Tools used: Tensorflow, NumPy, matplotlib and Keras(in Tensorflow)

I basically made 5 models, each having a different activation function and losses function. Through this, I tried to figure out the model which had the highest accuracy during learning the training test.

I have used adaptive movement estimation(adam) in all the models.(Initial rate was set to be remain as default)

I have used the Dense layer in all the models.(Dense layer takes in use of all the neurons coming from previous layer/input)

Test Dataset had 60,000 numbers which basically had an array of the intensity of each block of 28x28 grid. So, I had to flatten that data set into (28*28,1)= (784,1) for the first layer.

Model1 :

```
In [12]: model1=keras.Sequential([
             keras.layers.Dense(15,activation="sigmoid"),
         keras.layers.Dense(10,activation="sigmoid")

])
model1.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=['accuracy']
)
model1.fit(X_train_flattened,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 1s 721us/step - loss: 0.8030 - accuracy: 0.8315
Epoch 2/5
1875/1875 [==============================] - 1s 681us/step - loss: 0.3459 - accuracy: 0.9086
Epoch 3/5
1875/1875 [==============================] - 1s 684us/step - loss: 0.2790 - accuracy: 0.9228
Epoch 4/5
```

This model has two layers:-

i) First layer has 15units and activation function sigmoid

ii)Second layer has 10 units and activation function sigmoid

The loss function used in this model is sparse categorical cross entropy

Dataset was trained 5 times(epochs=5) and at the end accuracy was 93.59%

Model2:

```
In [13]: model2=keras.Sequential([
             keras.layers.Dense(15,activation="sigmoid"),
         keras.layers.Dense(10,activation="softmax")

         ])
         model2.compile(
             optimizer="adam",
             loss="sparse_categorical_crossentropy",
             metrics=['accuracy']
         )
         model2.fit(X_train_flattened,y_train, epochs=5)

         Epoch 1/5
         1875/1875 [==============================] - 1s 677us/step - loss: 0.8685 - accuracy: 0.8089
         Epoch 2/5
         1875/1875 [==============================] - 1s 683us/step - loss: 0.3565 - accuracy: 0.9085
         Epoch 3/5
         1875/1875 [==============================] - 1s 684us/step - loss: 0.2845 - accuracy: 0.9214
         Epoch 4/5
         1875/1875 [==============================] - 1s 685us/step - loss: 0.2535 - accuracy: 0.9293
         Epoch 5/5
         1875/1875 [==============================] - 1s 701us/step - loss: 0.2348 - accuracy: 0.9339

Out[13]: <keras.callbacks.History at 0x7fb93c7d0d60>
```

This model has two layers:-

i) First layer has 15units and activation function sigmoid

ii)Second layer has 10 units and activation function softmax

The loss function used in this model is sparse categorical cross entropy.

Dataset was trained 5 times(epochs=5) and at the end accuracy was 93.39%
[ACCURACY NOT MUCH DIFFERENT FROM Model1]

Model3:

```
In [18]: model3=keras.Sequential([
             keras.layers.Dense(15,activation="sigmoid"),
         keras.layers.Dense(12,activation="relu"),
         keras.layers.Dense(10,activation="linear")

])
model3.compile(
    optimizer="adam",
    loss="mean_squared_error",
    metrics=['accuracy']
)
model3.fit(X_train_flattened,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 2s 704us/step - loss: 3.9260 - accuracy: 0.0555
Epoch 2/5
1875/1875 [==============================] - 2s 848us/step - loss: 1.2465 - accuracy: 0.0457
Epoch 3/5
1875/1875 [==============================] - 1s 773us/step - loss: 1.0556 - accuracy: 0.0562
Epoch 4/5
1875/1875 [==============================] - 1s 714us/step - loss: 0.9594 - accuracy: 0.0594
Epoch 5/5
1875/1875 [==============================] - 1s 716us/step - loss: 0.8974 - accuracy: 0.0686
```

```
Out[18]: <keras.callbacks.History at 0x7fb93c2f4b80>
```

This model has three layers:-

i) First layer has 15units and activation function sigmoid
ii)Second layer has 12 units and activation function ReLU
iii)Third layer has 10 units with activation function linear

The loss function used in this model is mean squared error.

Dataset was trained 5 times(epochs=5) and at the end accuracy was 06.86%
[ACCURACY WAS MUCH POORER THAN Model1 and Model2]

Model4:

```
In [17]: model4=keras.Sequential([
             keras.layers.Dense(15,activation="sigmoid"),
         keras.layers.Dense(10,activation="softmax")

])
model4.compile(
    optimizer="adam",
    loss="mean_squared_error",
    metrics=['accuracy']
)
model4.fit(X_train_flattened,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 1s 688us/step - loss: 27.3046 - accuracy: 0.0992
Epoch 2/5
1875/1875 [==============================] - 1s 702us/step - loss: 27.3046 - accuracy: 0.0932
Epoch 3/5
1875/1875 [==============================] - 1s 702us/step - loss: 27.3046 - accuracy: 0.0930
Epoch 4/5
1875/1875 [==============================] - 1s 695us/step - loss: 27.3046 - accuracy: 0.0941
Epoch 5/5
1875/1875 [==============================] - 1s 782us/step - loss: 27.3046 - accuracy: 0.0989
```

```
Out[17]: <keras.callbacks.History at 0x7fb93c423a60>
```

This model has two layers:-

i) First layer has 15units and activation function sigmoid

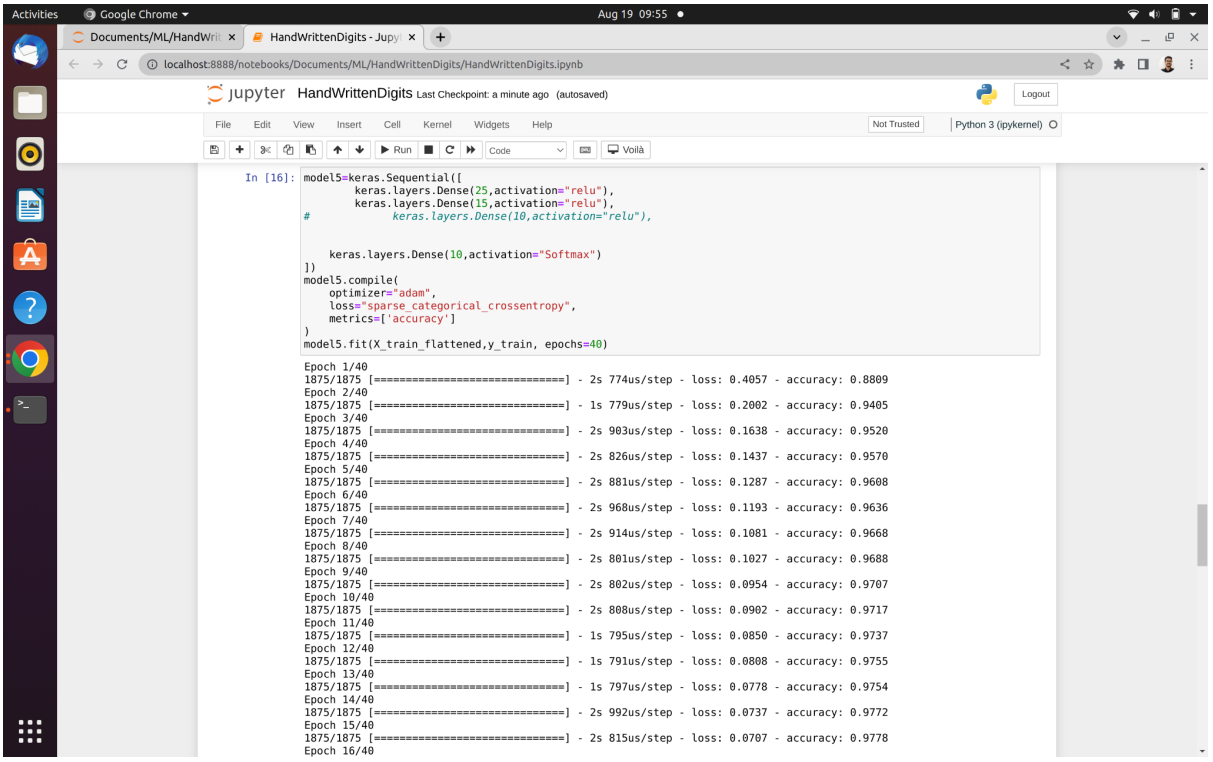
ii)Second layer has 12 units and activation function softmax

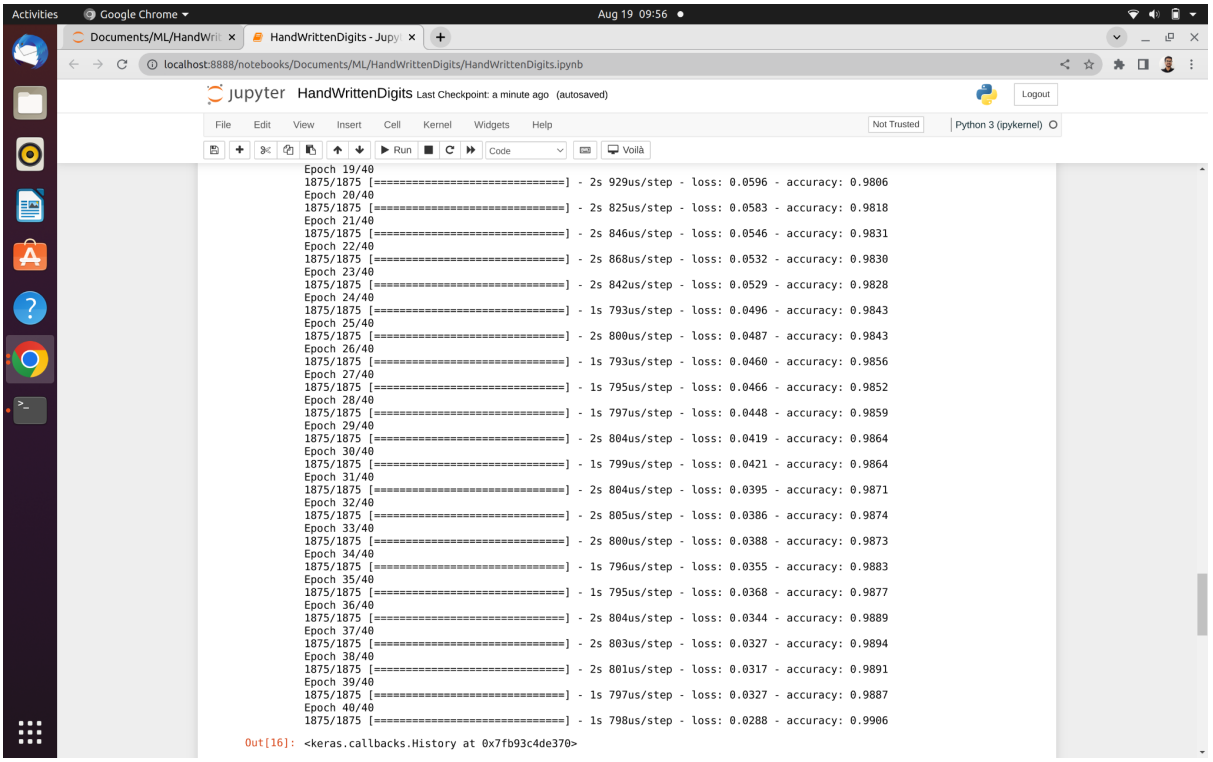
The loss function used in this model is mean squared error.

Dataset was trained 5 times(epochs=5) and at the end accuracy was 09.89%.

[ACCURACY WAS MUCH POORER THAN Model1 and Model2(slightly higher than Model3)]

Model5:


Activities
Google Chrome
Aug 19 09:55
Documents/ML/HandWrit
HandWrittenDigits - Jupy
localhost:8888/notebooks/Documents/ML/HandWrittenDigits/HandWrittenDigits.ipynb
jupyter HandWrittenDigits Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Voilà
In [16]:
model5=keras.Sequential([
keras.layers.Dense(25,activation="relu"),
keras.layers.Dense(15,activation="relu"),
# keras.layers.Dense(10,activation="relu"),
keras.layers.Dense(10,activation="Softmax")
])
model5.compile(
optimizer="adam",
loss="sparse_categorical_crossentropy",
metrics=['accuracy']
)
model5.fit(X_train_flattened,y_train, epochs=40)
Epoch 1/40
1875/1875 [==========] - 2s 774us/step - loss: 0.4057 - accuracy: 0.8809
Epoch 2/40
1875/1875 [==========] - 1s 779us/step - loss: 0.2002 - accuracy: 0.9405
Epoch 3/40
1875/1875 [==========] - 2s 903us/step - loss: 0.1638 - accuracy: 0.9520
Epoch 4/40
1875/1875 [==========] - 2s 826us/step - loss: 0.1437 - accuracy: 0.9570
Epoch 5/40
1875/1875 [==========] - 2s 881us/step - loss: 0.1287 - accuracy: 0.9608
Epoch 6/40
1875/1875 [==========] - 2s 968us/step - loss: 0.1193 - accuracy: 0.9636
Epoch 7/40
1875/1875 [==========] - 2s 914us/step - loss: 0.1081 - accuracy: 0.9668
Epoch 8/40
1875/1875 [==========] - 2s 801us/step - loss: 0.1027 - accuracy: 0.9688
Epoch 9/40
1875/1875 [==========] - 2s 802us/step - loss: 0.0954 - accuracy: 0.9707
Epoch 10/40
1875/1875 [==========] - 2s 808us/step - loss: 0.0902 - accuracy: 0.9717
Epoch 11/40
1875/1875 [==========] - 1s 795us/step - loss: 0.0850 - accuracy: 0.9737
Epoch 12/40
1875/1875 [==========] - 1s 791us/step - loss: 0.0808 - accuracy: 0.9755
Epoch 13/40
1875/1875 [==========] - 1s 797us/step - loss: 0.0778 - accuracy: 0.9754
Epoch 14/40
1875/1875 [==========] - 2s 992us/step - loss: 0.0737 - accuracy: 0.9772
Epoch 15/40
1875/1875 [==========] - 2s 815us/step - loss: 0.0707 - accuracy: 0.9778
Epoch 16/40


Activities
Google Chrome
Aug 19 09:56
Documents/ML/HandWrit
HandWrittenDigits - Jupy
localhost:8888/notebooks/Documents/ML/HandWrittenDigits/HandWrittenDigits.ipynb
jupyter HandWrittenDigits Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Voilà
Epoch 19/40
1875/1875 [==========] - 2s 929us/step - loss: 0.0596 - accuracy: 0.9806
Epoch 20/40
1875/1875 [==========] - 2s 825us/step - loss: 0.0583 - accuracy: 0.9818
Epoch 21/40
1875/1875 [==========] - 2s 846us/step - loss: 0.0546 - accuracy: 0.9831
Epoch 22/40
1875/1875 [==========] - 2s 868us/step - loss: 0.0532 - accuracy: 0.9830
Epoch 23/40
1875/1875 [==========] - 2s 842us/step - loss: 0.0529 - accuracy: 0.9828
Epoch 24/40
1875/1875 [==========] - 1s 793us/step - loss: 0.0496 - accuracy: 0.9843
Epoch 25/40
1875/1875 [==========] - 2s 800us/step - loss: 0.0487 - accuracy: 0.9843
Epoch 26/40
1875/1875 [==========] - 1s 793us/step - loss: 0.0460 - accuracy: 0.9856
Epoch 27/40
1875/1875 [==========] - 1s 795us/step - loss: 0.0466 - accuracy: 0.9852
Epoch 28/40
1875/1875 [==========] - 1s 797us/step - loss: 0.0448 - accuracy: 0.9859
Epoch 29/40
1875/1875 [==========] - 2s 804us/step - loss: 0.0419 - accuracy: 0.9864
Epoch 30/40
1875/1875 [==========] - 1s 799us/step - loss: 0.0421 - accuracy: 0.9864
Epoch 31/40
1875/1875 [==========] - 2s 804us/step - loss: 0.0395 - accuracy: 0.9871
Epoch 32/40
1875/1875 [==========] - 2s 805us/step - loss: 0.0386 - accuracy: 0.9874
Epoch 33/40
1875/1875 [==========] - 2s 800us/step - loss: 0.0388 - accuracy: 0.9873
Epoch 34/40
1875/1875 [==========] - 1s 796us/step - loss: 0.0355 - accuracy: 0.9883
Epoch 35/40
1875/1875 [==========] - 1s 795us/step - loss: 0.0368 - accuracy: 0.9877
Epoch 36/40
1875/1875 [==========] - 2s 804us/step - loss: 0.0344 - accuracy: 0.9889
Epoch 37/40
1875/1875 [==========] - 2s 803us/step - loss: 0.0327 - accuracy: 0.9894
Epoch 38/40
1875/1875 [==========] - 2s 801us/step - loss: 0.0317 - accuracy: 0.9891
Epoch 39/40
1875/1875 [==========] - 1s 797us/step - loss: 0.0327 - accuracy: 0.9887
Epoch 40/40
1875/1875 [==========] - 1s 798us/step - loss: 0.0288 - accuracy: 0.9906
Out[16]: <keras.callbacks.History at 0x7fb93c4de370>

This model has three layers:-

i) First layer has 25units and activation function ReLU

ii)Second layer has 15 units and activation function ReLU

iii)Third layer has 10 units with activation function Softmax

The loss function used in this model is sparse categorical cross entropy

Dataset was trained 40 times(epochs=40) and at the end accuracy was 99.06%(As it was giving the higher accuracy for 5 epochs so I tried making it learn 40 times so as to increase its accuracy)

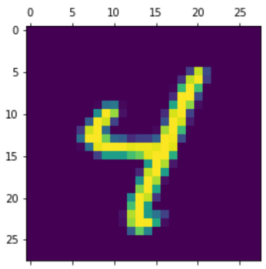
[THIS MODEL WAS THE BEST MODEL AND HAD HIGHEST PERCENTAGE OF ACCURACY AMONG ALL THE FIVE MODELS]

OBSERVATION:

I tried random examples and then predicted answer from all the models.

Eg:-

```
In [19]: plt.matshow(X_test[2306])
Out[19]: <matplotlib.image.AxesImage at 0x7fb93c1f56d0>
```




```
In [27]: y_test[2306]
Out[27]: 4
```

Testcase was 2306 and plt.matshow() helped me in showing the image of testcase

y_test() returns the answer associated with X_test()

```
In [21]: a=model1.predict(X_test_flattened)
         np.argmax(a[2306])
         313/313 [==============================] - 0s 424us/step
Out[21]: 4

In [22]: a=model2.predict(X_test_flattened)
         np.argmax(a[2306])
         313/313 [==============================] - 0s 455us/step
Out[22]: 3

In [23]: a=model3.predict(X_test_flattened)
         np.argmax(a[2306])
         313/313 [==============================] - 0s 464us/step
Out[23]: 4

In [24]: a=model4.predict(X_test_flattened)
         np.argmax(a[2306])
         313/313 [==============================] - 0s 426us/step
Out[24]: 6

In [25]: a=model5.predict(X_test_flattened)
         np.argmax(a[2306])
         313/313 [==============================] - 0s 469us/step
Out[25]: 4
```

We can notice that correct answer was predicted by model1, model3 and model5

Even though model2 had large accuracy but it predicted wrong answer and model3 had low accuracy but it predicted the correct answer.

np.argmax() returned the index of maximum value in the whole array.