Sprint 2 Team ID - PNT2022TMID25854

Importing the required libraries

```
import numpy as np
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataset
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A Layer consists of a tensor- in tensor-out computat
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the regular deepl
#faltten -used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #onvoLutiona l Layer
from keras.optimizers import Adam #opt imizer
from keras. utils import np_utils #used for one-hot encoding
import matplotlib.pyplot as plt   #used for data visualization
```

```
(x_train, y_train), (x_test, y_test)=mnist.load_data ()
x_train=x_train.reshape (60000, 28, 28, 1).astype('float32')
x_test=x_test.reshape (10000, 28, 28, 1).astype ('float32')
number_of_classes = 10  #storing the no of classes in a variable
y_train = np_utils.to_categorical (y_train, number_of_classes) #converts the output in bin
y_test = np_utils.to_categorical (y_test, number_of_classes)
```

Add CNN Layers

```
#create model
model=Sequential ()
```

```
#adding modeL Layer
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(32, (3, 3), activation = 'relu'))
```

```
#flatten the dimension of the image
model.add(Flatten())
```

```
#output layer with 10 neurons
model.add(Dense(number_of_classes,activation = 'softmax'))
```

Compiling the model

```
#Compile model
model.compile(loss= 'categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])
```

```
x_train = np.asarray(x_train)
y_train = np.asarray(y_train)
```

Train the model

```
#fit the model
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5, batch_size=32)
```

```
Epoch 1/5
1875/1875 [==============================] - 196s 104ms/step - loss: 0.2392 - accura
Epoch 2/5
1875/1875 [==============================] - 193s 103ms/step - loss: 0.0675 - accura
Epoch 3/5
1875/1875 [==============================] - 194s 104ms/step - loss: 0.0493 - accura
Epoch 4/5
1875/1875 [==============================] - 194s 103ms/step - loss: 0.0372 - accura
Epoch 5/5
1875/1875 [==============================] - 199s 106ms/step - loss: 0.0314 - accura
<keras.callbacks.History at 0x7f2675e2f650>
```

Observing the metrics

```
# Final evaluation of the model
metrics = model.evaluate(x_test, y_test, verbose=0)
print("Metrics (Test loss &Test Accuracy) : ")
print(metrics)
```

```
Metrics (Test loss &Test Accuracy) :
[0.09881837666034698, 0.9771000146865845]
```

```
prediction=model.predict(x_test[6000:6001])
print(prediction)
```

```
1/1 [==============================] - 0s 90ms/step
[[4.4744990e-14 6.9022756e-17 8.0684600e-12 3.0985490e-07 3.8223479e-06
  3.0827724e-08 1.4755837e-14 1.3811174e-06 1.8938267e-07 9.9999416e-01]]
```

```
plt.imshow(x_test[6000])
```

```
import numpy as np
print(np.argmax(prediction, axis=1)) #printing our Labels from first 4 images
```

```
[9]
```

```
np.argmax(y_test[6000:6001]) #printing the actual labels
9
```

```
9
```

Save The model

```
# Save the model
model.save('models/mnistCNN.h5')
```