

TEAM ID PNT2022TMID14851

```
import numpy #for numerical analysis
import tensorflow #open source ml tool by google

from tensorflow.keras.datasets import mnist #mnist dataset
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers

from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow import keras

from tensorflow.keras.optimizers import Adam
from keras.utils import np_utils
```

## LOADING DATASET

```
(x_train,y_train),(x_test,y_test)=mnist.load_data()

print(x_train.shape)
print(y_train.shape)

(60000, 28, 28)
(60000,)

print(x_test.shape)
print(y_test.shape)

(10000, 28, 28)
(10000,)
```

## ANALYZE THE DATA

```
x_train[3]

0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 60, 228, 251, 251, 94, 0, 0, 0, 0, 0, 0,
 0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 155, 253, 253, 189, 0, 0, 0, 0, 0, 0, 0,
 0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 20, 253, 251, 235, 66, 0, 0, 0, 0, 0, 0, 0,
 0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```

32, 205, 253, 251, 126, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
104, 251, 253, 184, 15, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 80,
240, 251, 193, 23, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 32, 253,
253, 253, 159, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 151, 251,
251, 251, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 48, 221, 251,
251, 172, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 234, 251, 251,
196, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 253, 251, 251,
89, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 159, 255, 253, 253,
31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 48, 228, 253, 247, 140,
8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 64, 251, 253, 220, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 64, 251, 253, 220, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 24, 193, 253, 220, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0]], dtype=uint8)

```

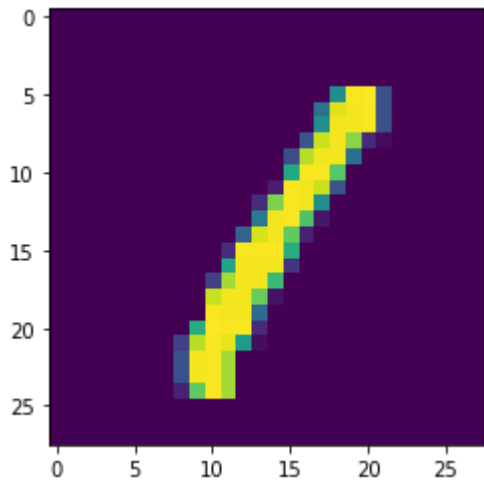
```
y_train[3]
```

```
1
```

```
import matplotlib.pyplot as plt
```

```
plt.imshow(x_train[3])
```

<matplotlib.image.AxesImage at 0x7efc78d14b10>



## RESHAPING THE DATA.

```
x_train=x_train.reshape(60000,28,28,1).astype('float32')
x_test=x_test.reshape(10000,28,28,1).astype('float32')
```

## APPLY ONE HOT ENCODING

```
no_of_classes=10
y_train=np_utils.to_categorical(y_train,no_of_classes)
y_test=np_utils.to_categorical(y_test,no_of_classes)
```

```
y_test[3]
```

```
array([1., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

## CREATE THE MODEL

```
model=Sequential()

model.add(Conv2D(64,(3,3),input_shape=(28,28,1),activation='relu'))
model.add(Conv2D(32,(3,3),activation='relu'))

model.add(Flatten())
model.add(Dense(no_of_classes,activation='softmax'))
```

## COMPILING THE MODEL

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

## TRAIN THE MODEL

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

```
Epoch 1/5
1875/1875 [=====] - 128s 68ms/step - loss: 0.2302 - accurac
Epoch 2/5
1875/1875 [=====] - 127s 68ms/step - loss: 0.0649 - accurac
Epoch 3/5
1875/1875 [=====] - 123s 66ms/step - loss: 0.0454 - accurac
Epoch 4/5
1875/1875 [=====] - 126s 67ms/step - loss: 0.0364 - accurac
Epoch 5/5
1875/1875 [=====] - 124s 66ms/step - loss: 0.0271 - accurac
<keras.callbacks.History at 0x7efc7452f890>
```



## METRICS ARE NOTED

```
metrics=model.evaluate(x_test,y_test,verbose=0)
print("metrics-score=>test loss & accuracy")
print(metrics)
```

```
metrics-score=>test loss & accuracy
[0.11036540567874908, 0.9764000177383423]
```

## TEST THE MODEL

```
prediction=model.predict(x_test[:5])
print(prediction)
```

```
1/1 [=====] - 0s 84ms/step
[[6.25657795e-15 1.05156142e-18 1.22086008e-09 2.45196552e-09
 1.33981165e-17 9.07641993e-17 4.98111414e-19 1.00000000e+00
 2.75971468e-11 2.33391622e-11]
 [1.02854422e-12 5.58150123e-11 1.00000000e+00 9.26562091e-11
 2.58257417e-17 1.22140988e-20 3.76503646e-12 2.03179154e-18
 2.17259214e-11 2.70688090e-21]
 [2.85233637e-09 9.99993920e-01 5.40673739e-07 3.44808820e-10
 2.74280274e-06 1.12679146e-07 4.11499196e-10 7.90978660e-09
 2.64735422e-06 2.92728147e-10]
 [9.99999881e-01 5.13201010e-16 9.24923071e-08 8.89283981e-13
 1.56655305e-14 1.21902911e-12 6.39609754e-11 1.28959387e-12
 8.11355072e-09 2.94187679e-08]
 [8.81784663e-12 1.38155817e-13 5.78738706e-12 1.68293005e-10
 9.99999285e-01 4.03126352e-16 3.91080943e-18 3.06052591e-15
 4.98500893e-11 7.03791216e-07]]
```

```
import numpy as np
```

```
print(np.argmax(prediction,axis=1))
```

```
[7 2 1 0 4]
```

```
print(y_test[:5])
```

```
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]  
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]  
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]  
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]  
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]
```

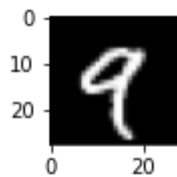
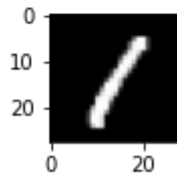
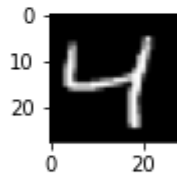
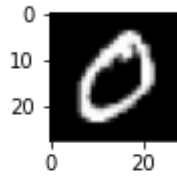
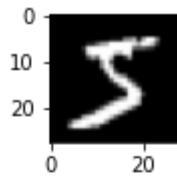
## SAVING THE MODEL

```
model.save('models/mnistcnn.h5')
```

## TEST THE SAVED MODEL

```
print('x_train:' +str(x_train.shape))  
print('y_train:' +str(y_train.shape))  
print('x_test:' +str(x_test.shape))  
print('y_test:' +str(y_test.shape))  
from matplotlib import pyplot  
for i in range(9):  
    pyplot.subplot(330+1+i)  
    pyplot.imshow(x_train[i],cmap=pyplot.get_cmap('gray'))  
    pyplot.show()
```

```
x_train:(60000, 28, 28)
y_train:(60000,)
x_test:(10000, 28, 28)
y_test:(10000,)
```



```
from tensorflow.keras.models import load_model
model=load_model('models/mnistcnn.h5')
from PIL import Image
for index in range(9):
    img=x_train[index].reshape((28,28))
    imgarray=np.array(img)
    imgarray=imgarray.reshape(1,28,28,1)
    y_pred=model.predict(imgarray)
    print(np.argmax(y_pred))
```

```
1/1 [=====] - 0s 58ms/step
5
1/1 [=====] - 0s 19ms/step
0
1/1 [=====] - 0s 20ms/step
4
1/1 [=====] - 0s 16ms/step
1
1/1 [=====] - 0s 21ms/step
9
1/1 [=====] - 0s 22ms/step
2
```

```
1/1 [=====] - 0s 19ms/step
1
1/1 [=====] - 0s 17ms/step
3
1/1 [=====] - 0s 18ms/step
1
```

[Colab paid products](#) - [Cancel contracts here](#)0s

✓

completed at 10:16 PM

×