

▼ Sprint 2

Team ID - PNT2022TMID07039

▼ 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 [=====] - 120s 63ms/step - loss: 0.2808 - accuracy
Epoch 2/5
1875/1875 [=====] - 95s 50ms/step - loss: 0.0734 - accuracy
Epoch 3/5
1875/1875 [=====] - 88s 47ms/step - loss: 0.0524 - accuracy
Epoch 4/5
1875/1875 [=====] - 97s 52ms/step - loss: 0.0375 - accuracy
Epoch 5/5
1875/1875 [=====] - 96s 51ms/step - loss: 0.0275 - accuracy
<keras.callbacks.History at 0x237357554b0>
```

▼ 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)
```

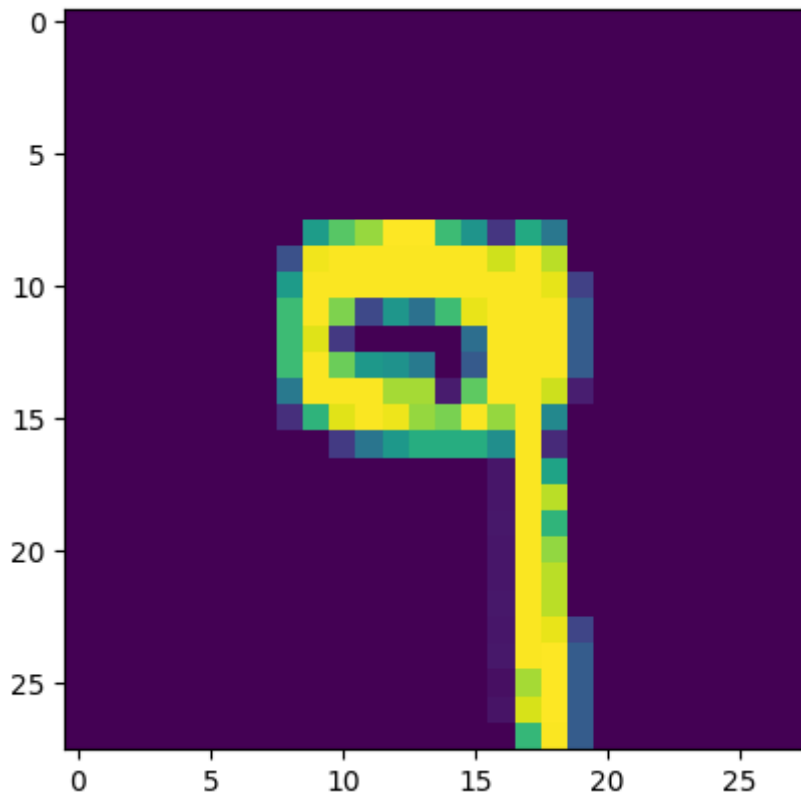
```
Metrics (Test loss &Test Accuracy) :
[0.12333168834447861, 0.9781000018119812]
```

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

```
1/1 [=====] - 0s 93ms/step
[[1.1050688e-15  9.2188809e-14  5.0320639e-15  5.0875104e-10  2.6267043e-01
  1.0428948e-08  5.2634429e-17  2.9271017e-09  6.4425700e-07  7.3732901e-01]]
```

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

 <matplotlib.image.AxesImage at 0x2373479ccd0>



```
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

▼ Save The model

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

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

