```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import random
```

```
mnist = tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test) = mnist.load_data()
x_train = x_train/255
x_test = x_test/255
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),
    keras.layers.Dense(128 , activation = "relu"),
    keras.layers.Dense(10 , activation = "softmax")
])
```

```
model.summary
```

```
<bound method Model.summary of <keras.src.engine.sequential.Sequential object at 0x7bdfec669960>>
```

```
model.compile(loss = "sparse_categorical_crossentropy" , optimizer = "sgd", metrics = ['accuracy'])
H = model.fit(x_train, y_train , validation_data=(x_test,y_test),epochs = 3)
```

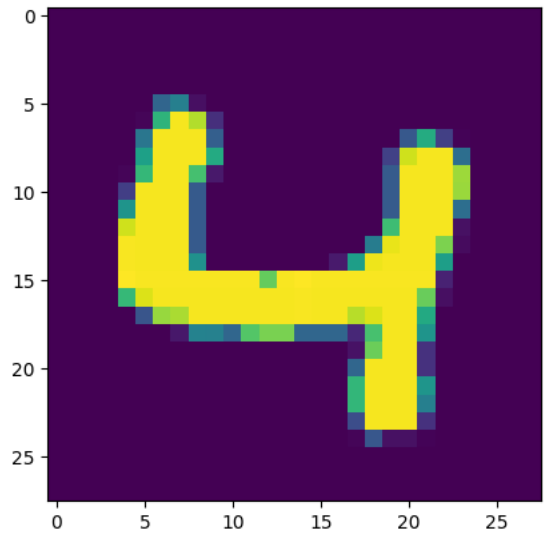
```
Epoch 1/3
1875/1875 [==============================] - 7s 3ms/step - loss: 0.6528 - accuracy: 0.8342 - val_loss: 0.3621 - val_accuracy: 0.9031
Epoch 2/3
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3407 - accuracy: 0.9057 - val_loss: 0.2974 - val_accuracy: 0.9184
Epoch 3/3
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2915 - accuracy: 0.9180 - val_loss: 0.2631 - val_accuracy: 0.9268
```

```
import numpy as np
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.1f" %test_loss)
print("Accuracy=%.1f" %test_acc)

n=random.randint(0,len(x_test)-1)
plt.imshow(x_test[n])
plt.show()
predicted_value = model.predict(x_test[n:n+1])
predicted_class = np.argmax(predicted_value)
plt.show()

print('Predicted Probability:', predicted_value[0][predicted_class])
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.2631 - accuracy: 0.9
Loss=0.3
Accuracy=0.9
```




```
1/1 [==============================] - 0s 21ms/step
Predicted Probability: 0.9870583
```

```
plt.plot(H.history['loss'])
```

```
plt.plot(H.history['val_loss'])
plt.title('model loss')
plt.xlabel('loss')
plt.ylabel('epoch')
plt.legend(['Train','Validation'],loc = 'upper left')
plt.show()
```