

```
In [1]: import tensorflow as tf
        from tensorflow import keras
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import random
        %matplotlib inline

In [2]: mnist = tf.keras.datasets.mnist
        (x_train, y_train), (x_test, y_test) = mnist.load_data()

In [3]: x_train = x_train / 255
        x_test = x_test / 255

In [4]: model = keras.Sequential([
        keras.layers.Flatten(input_shape=(28, 28)),
        keras.layers.Dense(128, activation="relu"),
        keras.layers.Dense(10, activation="softmax")
        ])

        model.summary()

C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an `input_shape`/'input_dim' argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 128)	100,480
dense_1 (Dense)	(None, 10)	1,290

Total params: 101,770 (397.54 KB)  
Trainable params: 101,770 (397.54 KB)  
Non-trainable params: 0 (0.00 B)

```
In [5]: model.compile(optimizer="sgd",loss="sparse_categorical_crossentropy",metrics=['accuracy'])

In [6]: history=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=10)

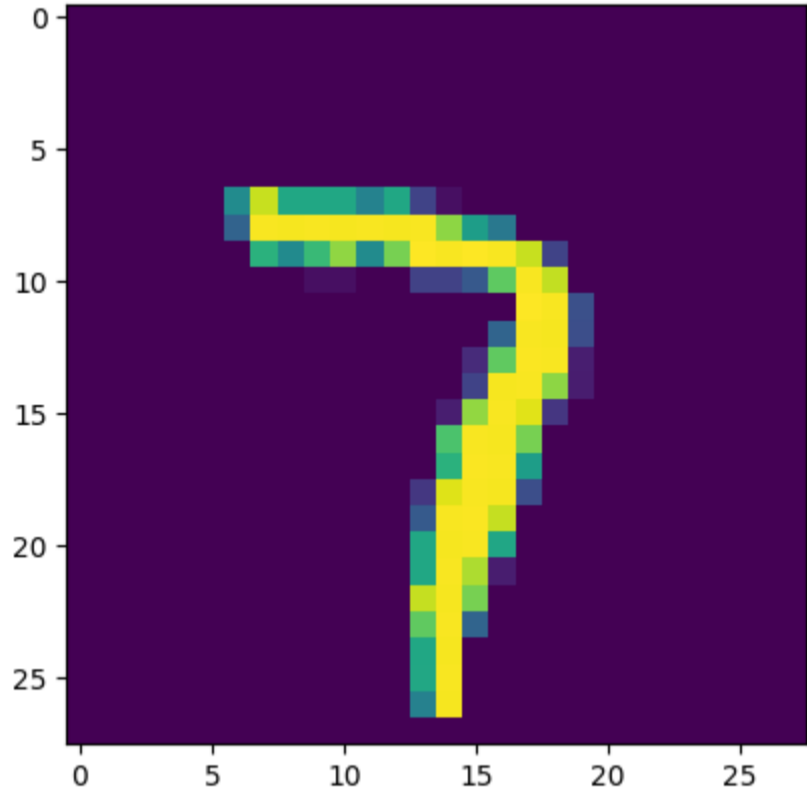
Epoch 1/10
1875/1875 ----- 26s 12ms/step - accuracy: 0.8404 - loss: 0.6364 - val_accuracy: 0.9017 - val_loss: 0.3569
Epoch 2/10
1875/1875 ----- 41s 12ms/step - accuracy: 0.9063 - loss: 0.3362 - val_accuracy: 0.9180 - val_loss: 0.2936
Epoch 3/10
1875/1875 ----- 41s 12ms/step - accuracy: 0.9187 - loss: 0.2882 - val_accuracy: 0.9286 - val_loss: 0.2592
Epoch 4/10
1875/1875 ----- 23s 12ms/step - accuracy: 0.9275 - loss: 0.2580 - val_accuracy: 0.9324 - val_loss: 0.2388
Epoch 5/10
1875/1875 ----- 24s 12ms/step - accuracy: 0.9342 - loss: 0.2354 - val_accuracy: 0.9390 - val_loss: 0.2199
Epoch 6/10
1875/1875 ----- 23s 12ms/step - accuracy: 0.9391 - loss: 0.2171 - val_accuracy: 0.9403 - val_loss: 0.2062
Epoch 7/10
1875/1875 ----- 22s 12ms/step - accuracy: 0.9434 - loss: 0.2016 - val_accuracy: 0.9459 - val_loss: 0.1926
Epoch 8/10
1875/1875 ----- 22s 12ms/step - accuracy: 0.9471 - loss: 0.1882 - val_accuracy: 0.9482 - val_loss: 0.1788
Epoch 9/10
1875/1875 ----- 22s 12ms/step - accuracy: 0.9509 - loss: 0.1765 - val_accuracy: 0.9511 - val_loss: 0.1729
Epoch 10/10
1875/1875 ----- 22s 12ms/step - accuracy: 0.9536 - loss: 0.1662 - val_accuracy: 0.9534 - val_loss: 0.1624

In [7]: print("Loss and accuracy of model are : ",model.evaluate(x_test,y_test) )

313/313 ----- 3s 10ms/step - accuracy: 0.9534 - loss: 0.1624
Loss and accuracy of model are : [0.16239498555660248, 0.9534000158309937]

In [8]: n=random.randint(0,9999)
        predicted_value=model.predict(x_test)
        plt.imshow(x_test[n])
        print(predicted_value[n])

313/313 ----- 3s 8ms/step
[1.1755784e-06 2.7273209e-06 5.4452830e-04 4.3449661e-04 3.4641369e-05
 5.2590945e-06 4.1484336e-07 9.9851257e-01 1.9509982e-05 4.4474230e-04]
```

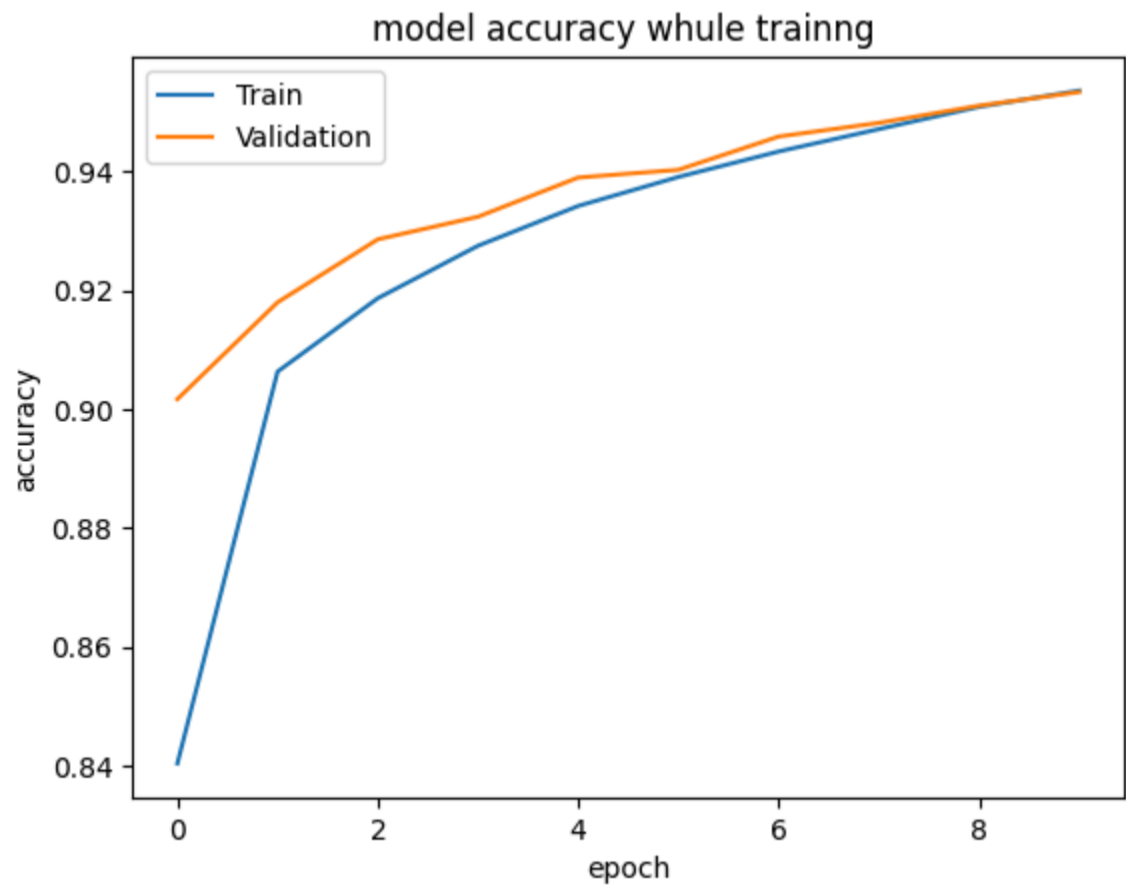


```
In [9]: history.history.keys()

Out[9]: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])

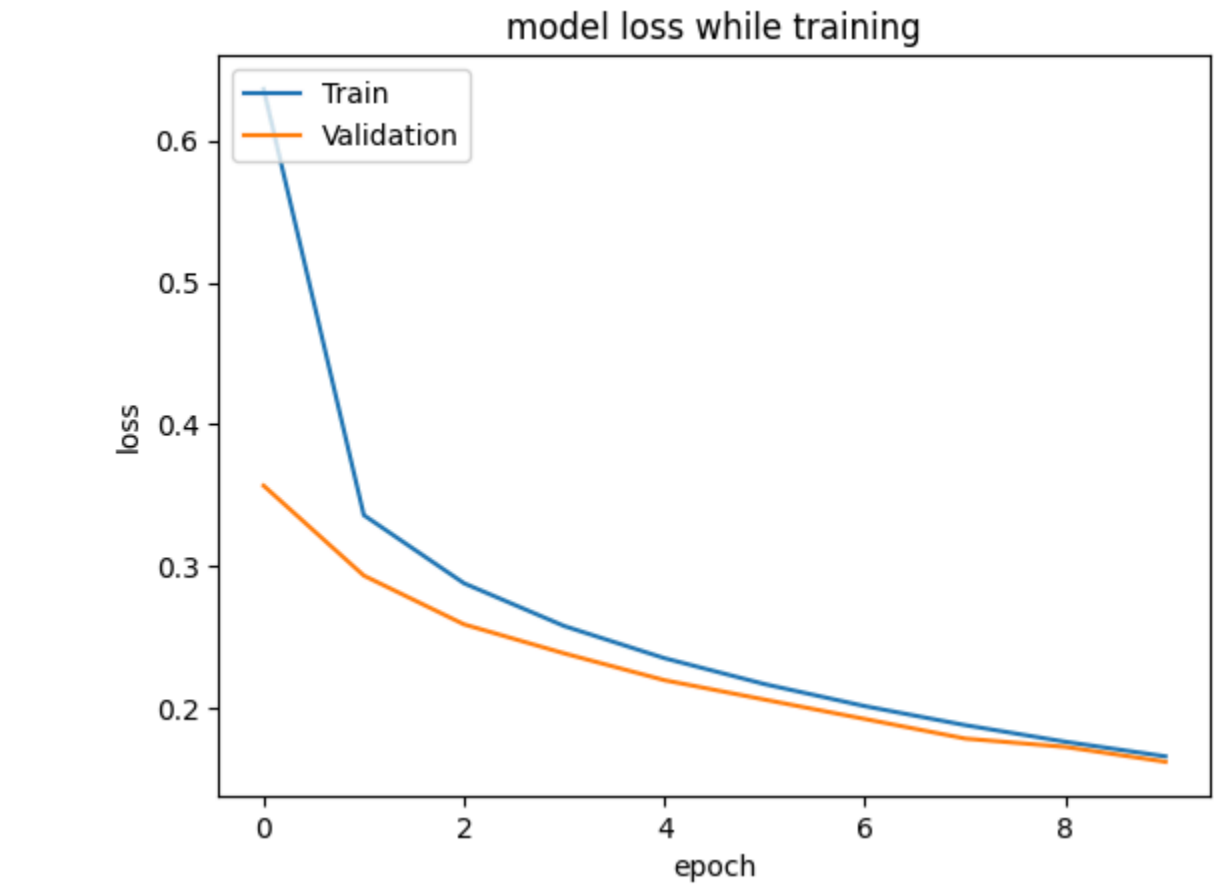
In [10]: plt.plot(history.history['accuracy'])
         plt.plot(history.history['val_accuracy'])
         plt.title('model accuracy while trainng ')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['Train', 'Validation'], loc='upper left')

Out[10]: <matplotlib.legend.Legend at 0x23b36869fa0>
```



```
In [11]: plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('model loss while training')
         plt.ylabel('loss')
         plt.xlabel('epoch')
         plt.legend(['Train', 'Validation'], loc='upper left')

Out[11]: <matplotlib.legend.Legend at 0x23b1a8000b0>
```



In [ ]: