```
!pip install -q keras
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import models,datasets,layers
import matplotlib.pyplot as plt
import matplotlib.image as mp
```

```
(train_images,train_labels), (test_images,test_labels)=datasets.mnist.load_data()
```

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

```
print('x_train:', train_images.shape)
print('y_train:', train_labels.shape)
print('x_test:', test_images.shape)
print('y_test:', test_labels.shape)
```

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

```
pd.DataFrame(train_images[0])
```

	0	1	2	3	4	5	6	7	8	9	...	18	19	20	21	22
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	175	26	166	255	247
6	0	0	0	0	0	0	0	0	30	36	...	225	172	253	242	195
7	0	0	0	0	0	0	0	49	238	253	...	93	82	82	56	39
8	0	0	0	0	0	0	0	18	219	253	...	0	0	0	0	0
9	0	0	0	0	0	0	0	0	80	156	...	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	14	...	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	...	25	0	0	0	0

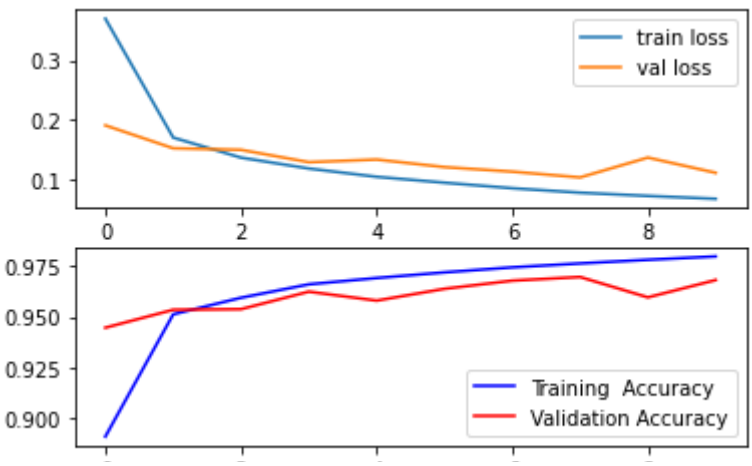
```
train_images=train_images/255
test_images= test_images/255


model=models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28, 1)))
model.add(layers.Dense(32,activation="relu"))
model.add(layers.Dense(16,activation="relu"))
model.add(layers.Dense(10,activation="softmax"))
model.compile(optimizer='adam',loss="sparse_categorical_crossentropy",metrics=['accuracy']
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_3 (Flatten)         (None, 784)               0

 dense_9 (Dense)             (None, 32)                25120

 dense_10 (Dense)            (None, 16)                528

 dense_11 (Dense)            (None, 10)                170

=================================================================
Total params: 25,818
Trainable params: 25,818
Non-trainable params: 0
_________________________________________________________________
```

```
h1 = model.fit(train_images, train_labels, epochs=10,validation_data=(test_images,test_lab
```

```
Epoch 1/10
1875/1875 [==============================] - 7s 3ms/step - loss: 0.3697 - accuracy: 0
Epoch 2/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.1702 - accuracy: 0
Epoch 3/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.1365 - accuracy: 0
Epoch 4/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.1182 - accuracy: 0
Epoch 5/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.1043 - accuracy: 0
Epoch 6/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.0944 - accuracy: 0
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0851 - accuracy: 0
Epoch 8/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.0776 - accuracy: 0
Epoch 9/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.0722 - accuracy: 0
Epoch 10/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.0673 - accuracy: 0
```

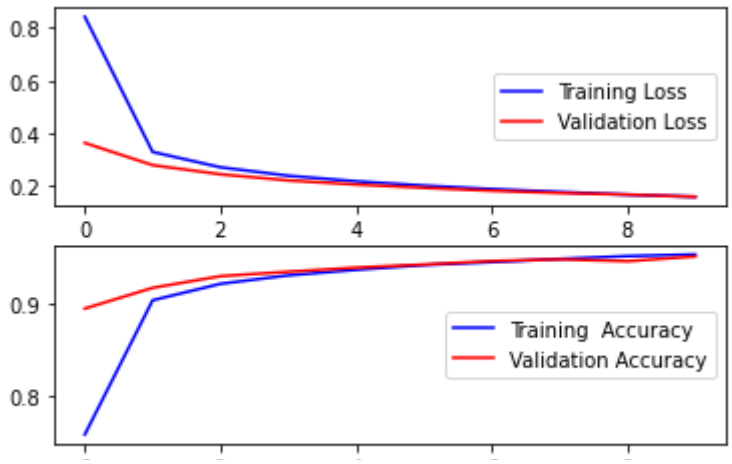
```
import matplotlib.pyplot as plt
#Plotting the training and validation loss and accuracy
f,ax=plt.subplots(2,1)

#Loss
ax[0].plot(h1.history['loss'], label='train loss')
ax[0].plot(h1.history['val_loss'], label='val loss')
ax[0].legend(loc='upper right')
#Accuracy
ax[1].plot(h1.history['accuracy'],color='b',label='Training  Accuracy')
ax[1].plot(h1.history['val_accuracy'],color='r',label='Validation Accuracy')
ax[1].legend(loc='lower right');
```




```
model.compile(optimizer='sgd',loss="sparse_categorical_crossentropy",metrics=['accuracy'])
h2 = model.fit(train_images, train_labels, epochs=10,validation_data=(test_images,test_lab
```

```
Epoch 1/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.8424 - accuracy: 0
Epoch 2/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.3294 - accuracy: 0
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2712 - accuracy: 0
Epoch 4/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2392 - accuracy: 0
Epoch 5/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2180 - accuracy: 0
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2018 - accuracy: 0
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1885 - accuracy: 0
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1778 - accuracy: 0
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1673 - accuracy: 0
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1593 - accuracy: 0
```

```
#Plotting the training and validation loss and accuracy
f,ax=plt.subplots(2,1)

#Loss
ax[0].plot(h2.history['loss'],color='b',label='Training Loss')
ax[0].plot(h2.history['val_loss'],color='r',label='Validation Loss')
ax[0].legend(loc='center right');
#Accuracy
ax[1].plot(h2.history['accuracy'],color='b',label='Training  Accuracy')
ax[1].plot(h2.history['val_accuracy'],color='r',label='Validation Accuracy')
ax[1].legend(loc='center right');
```

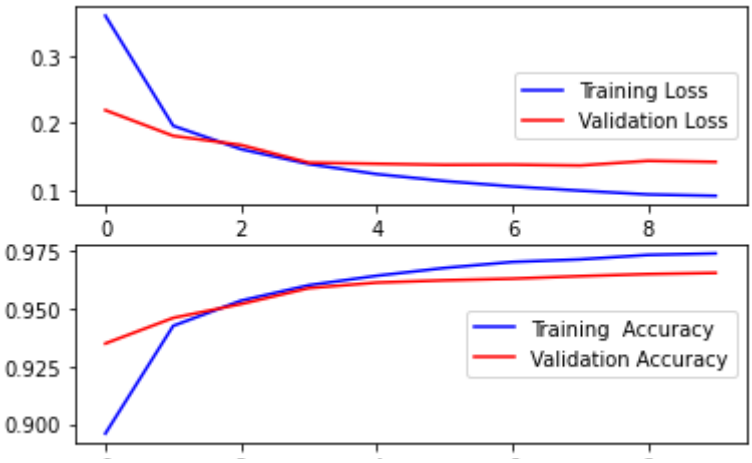



```
model.compile(optimizer='rmsprop',loss="sparse_categorical_crossentropy",metrics=['accurac
h3 = model.fit(train_images, train_labels, epochs=10,validation_data=(test_images,test_lab
```

```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3601 - accuracy: 0
Epoch 2/10
```

```
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1956 - accuracy: 0
Epoch 3/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1607 - accuracy: 0
Epoch 4/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1384 - accuracy: 0
Epoch 5/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1234 - accuracy: 0
Epoch 6/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1130 - accuracy: 0
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1050 - accuracy: 0
Epoch 8/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0986 - accuracy: 0
Epoch 9/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0930 - accuracy: 0
Epoch 10/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0909 - accuracy: 0
```

```
#Plotting the training and validation loss and accuracy
f,ax=plt.subplots(2,1)

#Loss
ax[0].plot(h3.history['loss'],color='b',label='Training Loss')
ax[0].plot(h3.history['val_loss'],color='r',label='Validation Loss')
ax[0].legend(loc='center right');
#Accuracy
ax[1].plot(h3.history['accuracy'],color='b',label='Training  Accuracy')
ax[1].plot(h3.history['val_accuracy'],color='r',label='Validation Accuracy')
ax[1].legend(loc='center right');
```

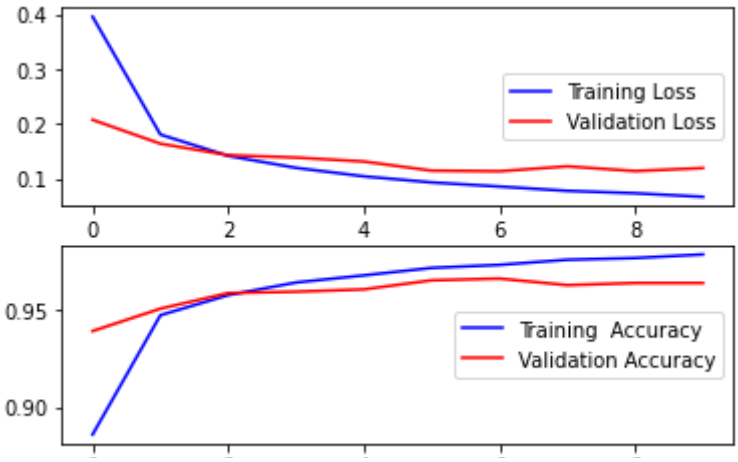



```
model.compile(optimizer='nadam',loss="sparse_categorical_crossentropy",metrics=['accuracy'
h4 = model.fit(train_images, train_labels, epochs=10,validation_data=(test_images,test_lab
```

```
Epoch 1/10
1875/1875 [==============================] - 7s 3ms/step - loss: 0.3952 - accuracy: 0
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1808 - accuracy: 0
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1420 - accuracy: 0
```

```
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1202 - accuracy: 0
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1047 - accuracy: 0
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0937 - accuracy: 0
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0862 - accuracy: 0
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0783 - accuracy: 0
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0740 - accuracy: 0
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0675 - accuracy: 0
```

```
#Plotting the training and validation loss and accuracy
f,ax=plt.subplots(2,1)

#Loss
ax[0].plot(h4.history['loss'],color='b',label='Training Loss')
ax[0].plot(h4.history['val_loss'],color='r',label='Validation Loss');
ax[0].legend(loc='center right');
#Accuracy
ax[1].plot(h4.history['accuracy'],color='b',label='Training  Accuracy')
ax[1].plot(h4.history['val_accuracy'],color='r',label='Validation Accuracy')
ax[1].legend(loc='center right');
```




```
score = model.evaluate(test_images,test_labels)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.1198 - accuracy: 0.9
Test score: 0.1198384165763855
Test accuracy: 0.9642999768257141
```

```
model_name = 'digits_recognition.h5'
model.save(model_name, save_format='h5')
```

```
loaded_model = tf.keras.models.load_model(model_name)
predictions_one_hot = loaded_model.predict([test_images])
print('predictions_one_hot:', predictions_one_hot.shape)
```

```
313/313 [==============================] - 1s 1ms/step
predictions_one_hot: (10000, 10)
```

```
# Let's extract predictions with highest probabilites and detect what digits have been act
predictions = np.argmax(predictions_one_hot, axis=1)
pd.DataFrame(predictions)
plt.imshow(test_images[0].reshape((28,28)), cmap=plt.cm.binary)
plt.show()
```