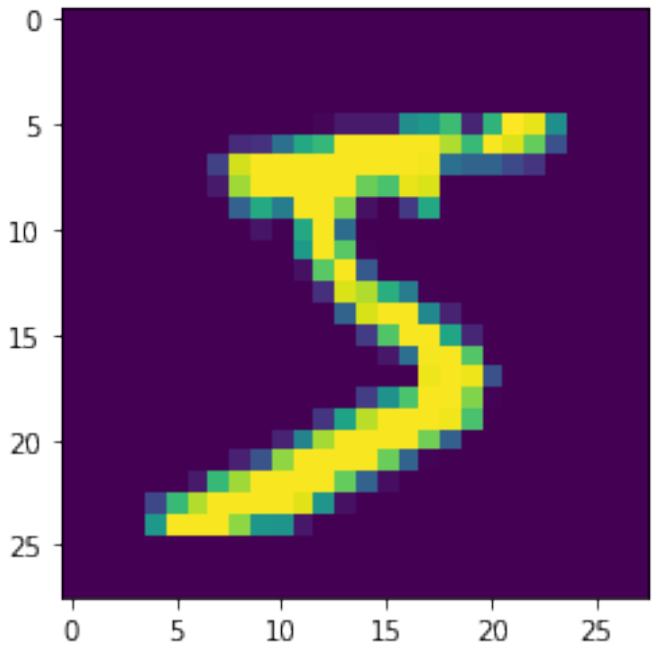
Hand Written Digit Recognition.

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
mnist=tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test) = mnist.load_data()
x_train.shape

plt.imshow(x_train[0])
plt.show()
plt.imshow(x_train[0], cmap = plt.cm.binary)

print(x_train[0]) # before normalizing

x_train=tf.keras.utils.normalize(x_train, axis = 1)
x_test=tf.keras.utils.normalize(x_test, axis = 1)
plt.imshow(x_train[0], cmap = plt.cm.binary)

print(x_train[0])# after normalizing

print(y_train[0])
```



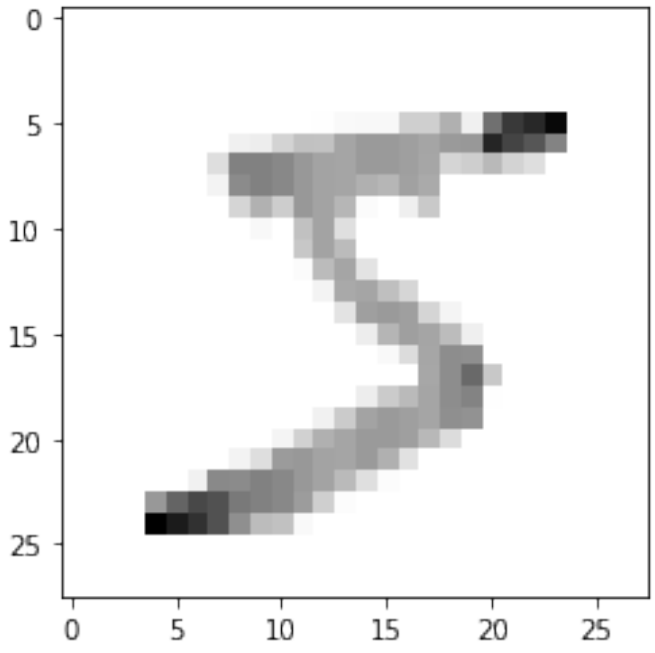
```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   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   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126
136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253
253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253
251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247
241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43
154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0
0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0
0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108
1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253
119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253
253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93
252
```

```
  253 187   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
249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183
253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253
253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253
201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81
2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0
0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0
0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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   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.         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.         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.
 0.00393124 0.02332955 0.02620568 0.02625207 0.17420356 0.17566281
 0.28629534 0.05664824 0.51877786 0.71632322 0.77892406 0.89301644
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.05780486 0.06524513 0.16128198 0.22713296
 0.22277047 0.32790981 0.36833534 0.3689874  0.34978968 0.32678448
 0.368094   0.3747499  0.79066747 0.67980478 0.61494005 0.45002403
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.12250613 0.45858525 0.45852825 0.43408872 0.37314701
 0.33153488 0.32790981 0.36833534 0.3689874  0.34978968 0.32420121
 0.15214552 0.17865984 0.25626376 0.1573102  0.12298801 0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.04500225 0.4219755  0.45852825 0.43408872 0.37314701
 0.33153488 0.32790981 0.28826244 0.26543758 0.34149427 0.31128482
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.1541463  0.28272888 0.18358693 0.37314701
 0.33153488 0.26569767 0.01601458 0.         0.05945042 0.19891229
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.0253731  0.00171577 0.22713296
 0.33153488 0.11664776 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.         0.20500962
 0.33153488 0.24625638 0.00291174 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.01622378
 0.24897876 0.32790981 0.10191096 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.
 0.04586451 0.31235677 0.32757096 0.23335172 0.14931733 0.00129164
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.10498298 0.34940902 0.3689874  0.34978968 0.15370495
 0.04089933 0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.06551419 0.27127137 0.34978968 0.32678448
 0.245396   0.05882702 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.02333517 0.12857881 0.32549285
 0.41390126 0.40743158 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.         0.32161793
 0.41390126 0.54251585 0.20001074 0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.06697006 0.18959827 0.25300993 0.32678448
 0.41390126 0.45100715 0.00625034 0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.05110617 0.19182076 0.33339444 0.3689874  0.34978968 0.32678448
 0.40899334 0.39653769 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.04117838 0.16813739
 0.28960162 0.32790981 0.36833534 0.3689874  0.34978968 0.25961929
 0.12760592 0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.04431706 0.11961607 0.36545809 0.37314701
 0.33153488 0.32790981 0.36833534 0.28877275 0.111988   0.00258328
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.05298497 0.42752138 0.4219755  0.45852825 0.43408872 0.37314701
 0.33153488 0.25273681 0.11646967 0.01312603 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.37491383 0.56222061
```

```
  0.66525569 0.63253163 0.48748768 0.45852825 0.43408872 0.359873
  0.17428513 0.01425695 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.92705966 0.82698729
  0.74473314 0.63253163 0.4084877  0.24466922 0.22648107 0.02359823
  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.         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.        ]]
5
```




```
import numpy as np

IMG_SIZE=28
x_trainr=np.array(x_train).reshape(-1, IMG_SIZE,IMG_SIZE,1)
```

```
x_testr=np.array(x_test).reshape(-1, IMG_SIZE,IMG_SIZE,1)
print("Training Samples dimension",x_trainr.shape)
print("Testing Samples dimension",x_testr.shape)
```

```
Training Samples dimension (60000, 28, 28, 1)
Testing Samples dimension (10000, 28, 28, 1)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation,
Flatten, Conv2D, MaxPooling2D

###  First Convolution Layer
model = Sequential()
model.add(Conv2D(64,(3,3),input_shape=x_trainr.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

###  Second Convolution Layer
model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

###  3rd Convolution Layer

model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

### Fully Connected Layer
model.add(Flatten())
model.add(Dense(64))
model.add(Activation("relu"))


### Fully Connected Layer #2
model.add(Dense(32))
model.add(Activation("relu"))

### Final Fully Connected Layer
model.add(Dense(10))
model.add(Activation("softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 activation (Activation)     (None, 26, 26, 64)        0

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        36928

 activation_1 (Activation)   (None, 11, 11, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 activation_2 (Activation)   (None, 3, 3, 64)          0

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 activation_3 (Activation)   (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 activation_4 (Activation)   (None, 32)                0

 dense_2 (Dense)             (None, 10)                330

 activation_5 (Activation)   (None, 10)                0

=================================================================
Total params: 81,066
Trainable params: 81,066
Non-trainable params: 0
_________________________________________________________________
```

```
print("Total Training Samples =",len(x_trainr))
model.compile(loss ="sparse_categorical_crossentropy", optimizer
```

```
="adam", metrics=['accuracy'])
h1=model.fit(x_trainr,y_train,epochs=10, validation_split= 0.3)
```

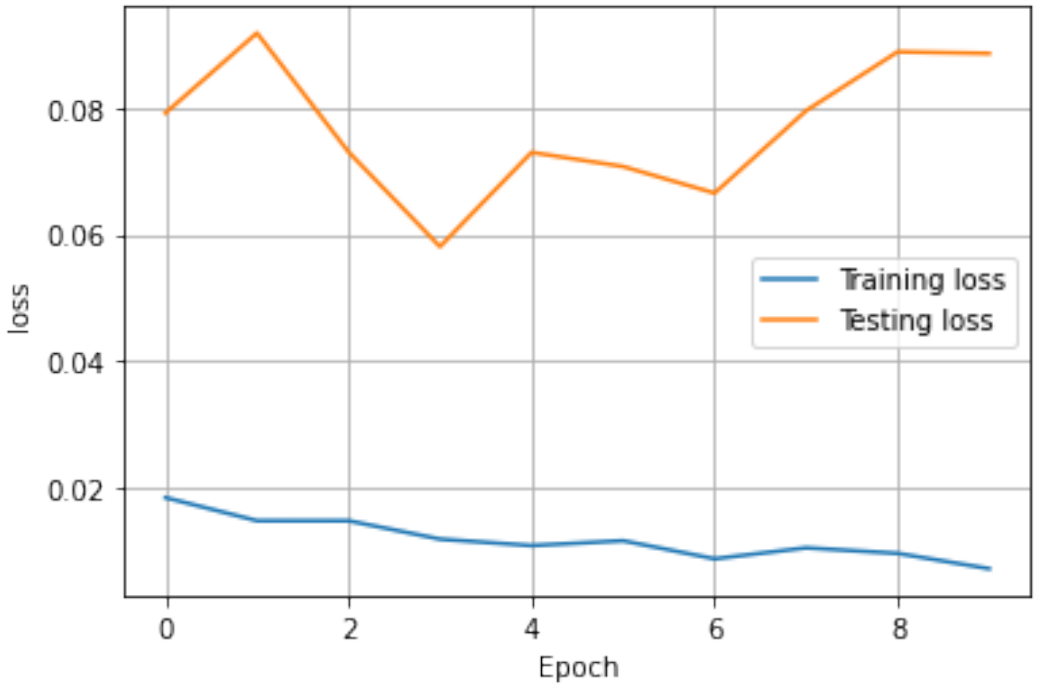
```
Total Training Samples = 60000
Epoch 1/10
1313/1313 [==============================] - 69s 52ms/step - loss:
0.0185 - accuracy: 0.9940 - val_loss: 0.0791 - val_accuracy: 0.9812
Epoch 2/10
1313/1313 [==============================] - 72s 55ms/step - loss:
0.0149 - accuracy: 0.9950 - val_loss: 0.0918 - val_accuracy: 0.9820
Epoch 3/10
1313/1313 [==============================] - 71s 54ms/step - loss:
0.0149 - accuracy: 0.9950 - val_loss: 0.0730 - val_accuracy: 0.9822
Epoch 4/10
1313/1313 [==============================] - 68s 52ms/step - loss:
0.0120 - accuracy: 0.9959 - val_loss: 0.0581 - val_accuracy: 0.9858
Epoch 5/10
1313/1313 [==============================] - 67s 51ms/step - loss:
0.0109 - accuracy: 0.9964 - val_loss: 0.0729 - val_accuracy: 0.9857
Epoch 6/10
1313/1313 [==============================] - 67s 51ms/step - loss:
0.0117 - accuracy: 0.9958 - val_loss: 0.0707 - val_accuracy: 0.9828
Epoch 7/10
1313/1313 [==============================] - 67s 51ms/step - loss:
0.0089 - accuracy: 0.9970 - val_loss: 0.0665 - val_accuracy: 0.9851
Epoch 8/10
1313/1313 [==============================] - 67s 51ms/step - loss:
0.0106 - accuracy: 0.9968 - val_loss: 0.0795 - val_accuracy: 0.9836
Epoch 9/10
1313/1313 [==============================] - 67s 51ms/step - loss:
0.0097 - accuracy: 0.9970 - val_loss: 0.0888 - val_accuracy: 0.9817
Epoch 10/10
1313/1313 [==============================] - 68s 51ms/step - loss:
0.0073 - accuracy: 0.9975 - val_loss: 0.0885 - val_accuracy: 0.9837
```

```
r1=pd.DataFrame(h1.history)
r1['Epoch']=h1.epoch
r1.tail
```

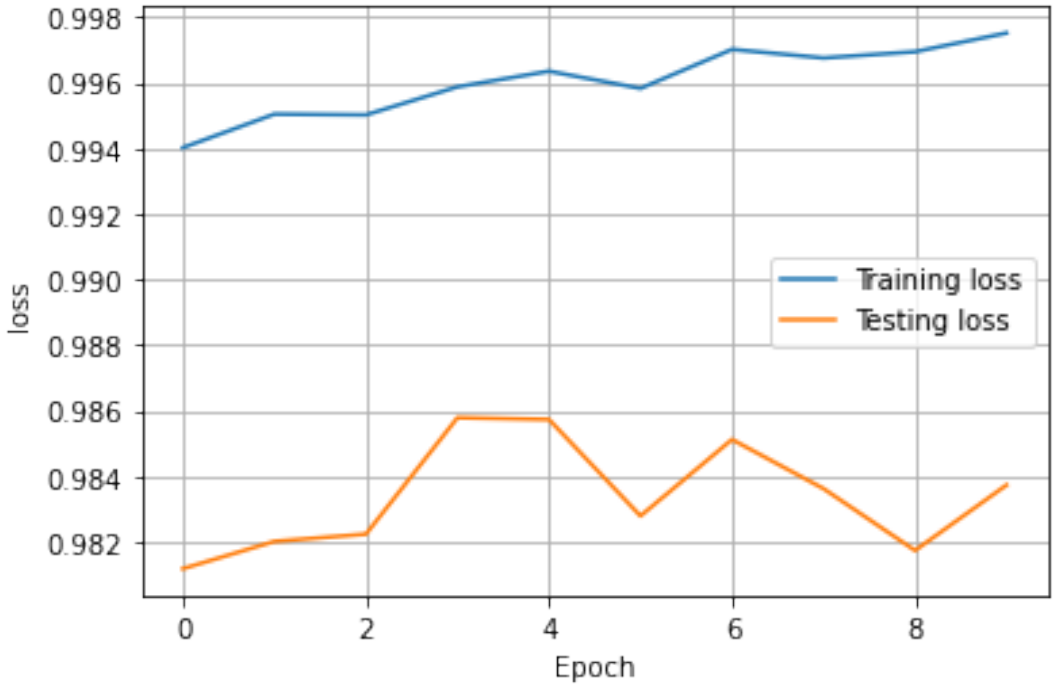
```
<bound method NDFrame.tail of        loss  accuracy  val_loss
val_accuracy  Epoch
0  0.018503  0.994024  0.079133      0.981167      0
1  0.014865  0.995048  0.091753      0.982000      1
2  0.014853  0.995024  0.073018      0.982222      2
3  0.011956  0.995881  0.058068      0.985778      3
4  0.010936  0.996357  0.072932      0.985722      4
5  0.011696  0.995833  0.070705      0.982778      5
```

```
6  0.008851  0.997024  0.066512      0.985111      6
7  0.010582  0.996762  0.079505      0.983611      7
8  0.009707  0.996952  0.088790      0.981722      8
9  0.007292  0.997524  0.088533      0.983722      9>
```

```
plt.plot(r1['Epoch'],r1['loss'],label='Training loss')
plt.plot(r1['Epoch'],r1['val_loss'],label='Testing loss')
plt.xlabel('Epoch')
plt.ylabel('loss')
plt.legend()
plt.grid()
plt.show()
```




```
plt.plot(r1['Epoch'],r1['accuracy'],label='Training loss')
plt.plot(r1['Epoch'],r1['val_accuracy'],label='Testing loss')
plt.xlabel('Epoch')
plt.ylabel('loss')
plt.legend()
plt.grid()
plt.show()
```




```
test_loss, test_acc= model.evaluate(x_testr,y_test)
print("Test Loss on 10,000 test Samples",test_loss)
print("Validation Accuracy on 10,000 on test Samples",test_acc)
```

```
313/313 [==============================] - 4s 13ms/step - loss: 0.0742
- accuracy: 0.9854
Test Loss on 10,000 test Samples 0.07422824949026108
Validation Accuracy on 10,000 on test Samples 0.9854000210762024
```

```
y_predict = model.predict([x_testr])
```

```
y_predict
```

```
array([[4.72564765e-10, 1.56244029e-09, 1.34179436e-06, ...,
        9.99998689e-01, 1.09729344e-11, 2.89590329e-09],
       [1.16848263e-12, 3.03815376e-11, 1.00000000e+00, ...,
        2.43454996e-13, 2.36088926e-13, 7.79841265e-16],
       [2.95212232e-13, 1.00000000e+00, 3.16915483e-12, ...,
        2.57694040e-13, 1.70665468e-10, 2.61081152e-12],
       ...,
       [1.50958756e-19, 1.63019541e-13, 2.00130870e-16, ...,
        2.46993103e-11, 1.71084650e-13, 5.23074917e-11],
       [1.23729056e-17, 1.26750246e-24, 1.03572546e-24, ...,
        2.48853405e-22, 4.12881371e-18, 2.45252000e-18],
       [2.31633734e-04, 1.83890048e-09, 6.56145289e-07, ...,
        3.03585579e-09, 2.25770646e-06, 2.17906972e-07]],
dtype=float32)
```

```
print(np.argmax(y_predict[0]))
```

```
7
```

```
print(np.argmax(y_predict[128]))
plt.imshow(x_test[128])
```

```
8
```

```
<matplotlib.image.AxesImage at 0x7f7a97a070d0>
```