+ Code + Text Copy to Drive Connect

```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Flatten
```

```
[ ] (X_train,y_train),(X_test,y_test) = keras.datasets.mnist.load_data()
```

```
[ ] X_test.shape
```

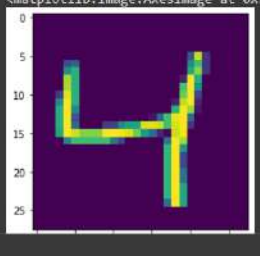
```
(10000, 28, 28)
```

```
[ ] y_train
```

```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

```
[ ] import matplotlib.pyplot as plt
    plt.imshow(X_train[2])
```

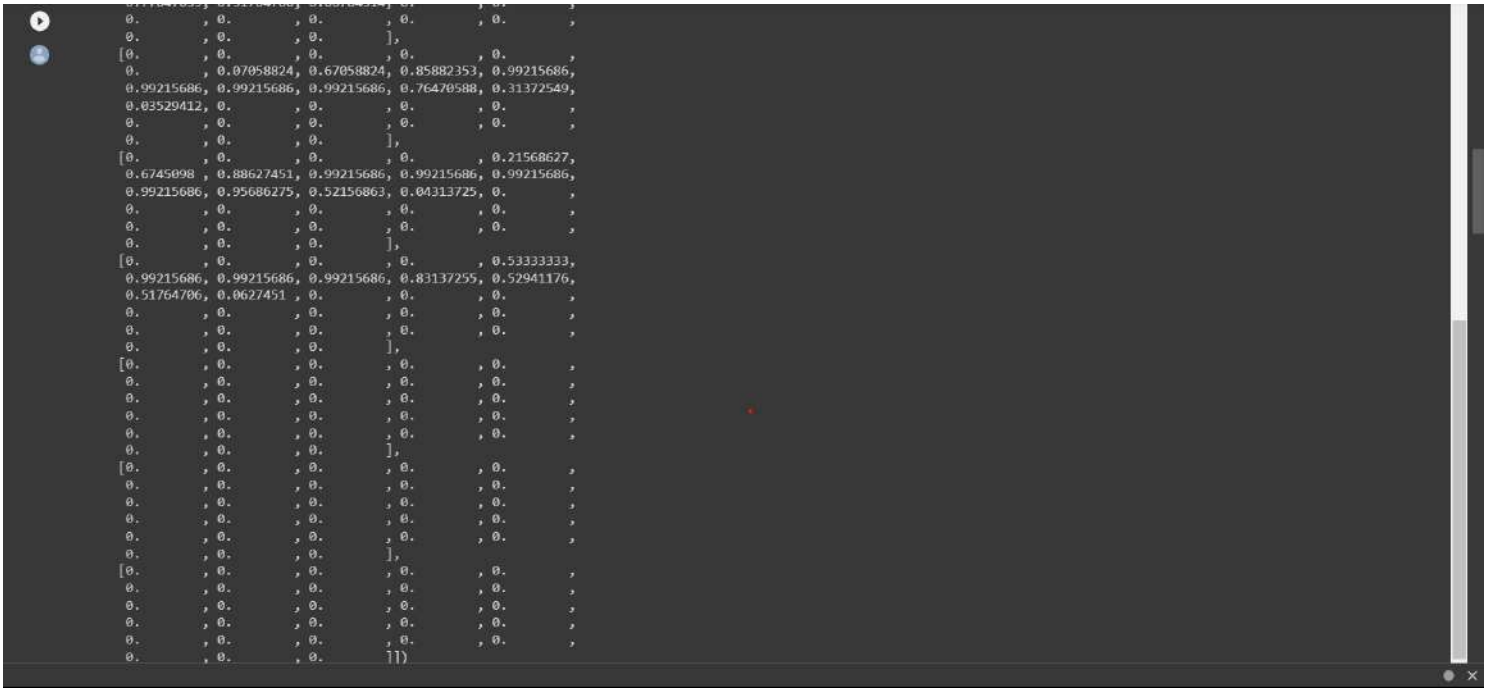
```
<matplotlib.image.AxesImage at 0x7f573ed9f790>
```



```
[ ] X_train = X_train/255
    X_test = X_test/255
```

```
x_train[0]
```

```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
```


0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   ],
[0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.07058824, 0.67058824, 0.85882353, 0.99215686,
0.99215686, 0.99215686, 0.99215686, 0.76470588, 0.31372549,
0.03529412, 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   ],
[0.   , 0.   , 0.   , 0.   , 0.21568627,
0.6745098 , 0.88627451, 0.99215686, 0.99215686, 0.99215686,
0.99215686, 0.95686275, 0.52156863, 0.04313725, 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   ],
[0.   , 0.   , 0.   , 0.   , 0.53333333,
0.99215686, 0.99215686, 0.99215686, 0.83137255, 0.52941176,
0.51764706, 0.0627451 , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   ],
[0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   ],
[0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   , 0.   , 0.   ,
0.   , 0.   , 0.   ]])

```
model = Sequential()

model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

```
[ ] model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_4 (Flatten)         (None, 784)               0

 dense_11 (Dense)            (None, 128)               100480

 dense_12 (Dense)            (None, 32)                4128

 dense_13 (Dense)            (None, 10)                330

=================================================================
Total params: 104,938
Trainable params: 104,938
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] model.compile(loss='sparse_categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
```

```
[ ] history = model.fit(X_train,y_train,epochs=25,validation_split=0.2)

    Epoch 1/25
    1500/1500 [==============================] - 5s 3ms/step - loss: 0.2845 - accuracy: 0.9168 - val_loss: 0.1425 - val_accuracy: 0.9570
    Epoch 2/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.1164 - accuracy: 0.9638 - val_loss: 0.1258 - val_accuracy: 0.9605
    Epoch 3/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0800 - accuracy: 0.9754 - val_loss: 0.0912 - val_accuracy: 0.9728
    Epoch 4/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0594 - accuracy: 0.9817 - val_loss: 0.1014 - val_accuracy: 0.9697
    Epoch 5/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0462 - accuracy: 0.9851 - val_loss: 0.1024 - val_accuracy: 0.9687
    Epoch 6/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0377 - accuracy: 0.9881 - val_loss: 0.1060 - val_accuracy: 0.9712
    Epoch 7/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0313 - accuracy: 0.9898 - val_loss: 0.0960 - val_accuracy: 0.9758
    Epoch 8/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0283 - accuracy: 0.9902 - val_loss: 0.1061 - val_accuracy: 0.9732
    Epoch 9/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0230 - accuracy: 0.9923 - val_loss: 0.1160 - val_accuracy: 0.9713
    Epoch 10/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0179 - accuracy: 0.9939 - val_loss: 0.1134 - val_accuracy: 0.9761
    Epoch 11/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0177 - accuracy: 0.9937 - val_loss: 0.1204 - val_accuracy: 0.9742
    Epoch 12/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0156 - accuracy: 0.9947 - val_loss: 0.1103 - val_accuracy: 0.9761
    Epoch 13/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0156 - accuracy: 0.9949 - val_loss: 0.1221 - val_accuracy: 0.9743
    Epoch 14/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0121 - accuracy: 0.9960 - val_loss: 0.1246 - val_accuracy: 0.9764
    Epoch 15/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0122 - accuracy: 0.9958 - val_loss: 0.1362 - val_accuracy: 0.9753
    Epoch 16/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0100 - accuracy: 0.9964 - val_loss: 0.1340 - val_accuracy: 0.9772
    Epoch 17/25
    1500/1500 [==============================] - 4s 3ms/step - loss: 0.0131 - accuracy: 0.9955 - val_loss: 0.1371 - val_accuracy: 0.9752
    Epoch 18/25
```

```
[ ] y_prob = model.predict(X_test)
```

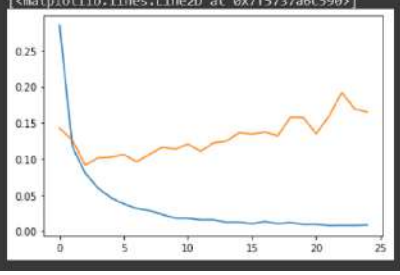
```
[ ] y_pred = y_prob.argmax(axis=1)
```

```
[ ] from sklearn.metrics import accuracy_score
    accuracy_score(y_test,y_pred)
```

```
0.9755
```

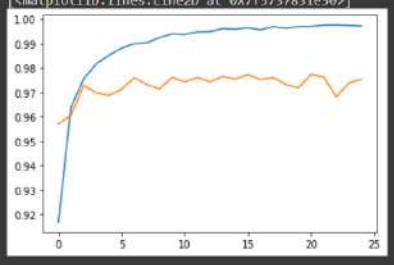
```
[ ] plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7f5737a6c590>]
```



```
[ ] plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
```

```
[<matplotlib.lines.Line2D at 0x7f5737831e50>]
```




```
[ ] plt.imshow(X_test[1])
```

```
<matplotlib.image.AxesImage at 0x7f5736844c50>
```