In [2]:
```
import numpy as np
import tensorflow as tf    #https://www.youtube.com/watch?v=_c_x8A3mNDk
```

In [3]:
```
from tensorflow import keras
import matplotlib.pyplot as plt
```

In [5]:
```
data=tf.keras.datasets.mnist
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.datasets import mnist
```

In [6]:
```
(x_train,y_train),(x_test,y_test)=data.load_data()   # x is attribute and y
```

In [7]:
```
x_test.shape        # show the data set of test 28 pixel , 28 pixel   pictu
```

Out[7]: (10000, 28, 28)

In [23]:
```
x_train.shape    # show the data set of training   28 pixel , 28 pixel   pi
print(x_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.
```

In [7]:
```
# To perform Machine Learning, it is important to convert all the values fr
# The simplest way is to divide the value of every pixel by 255 to get the
```

In [8]:
```
x_train,x_test =x_train/255,x_test/255      # Covert all values between 0 t
```

In [9]:
```
model=tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=(28,28)),      # covert 2D into  1D
tf.keras.layers.Dense(150,activation='relu'),      #  Dense Means Fully cor
tf.keras.layers.Dense(10,activation='softmax')     # classifying into 10 cl
])
```

In [10]:
```
# The default learning rate is 0.01 and no momentum is used by default.
sgd=SGD(0.02)    # is learning rate  0.02
#adam
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metr

#
```

In [11]:
```
history=model.fit(x_train, y_train,validation_data=(x_test,y_test),epochs=5
```

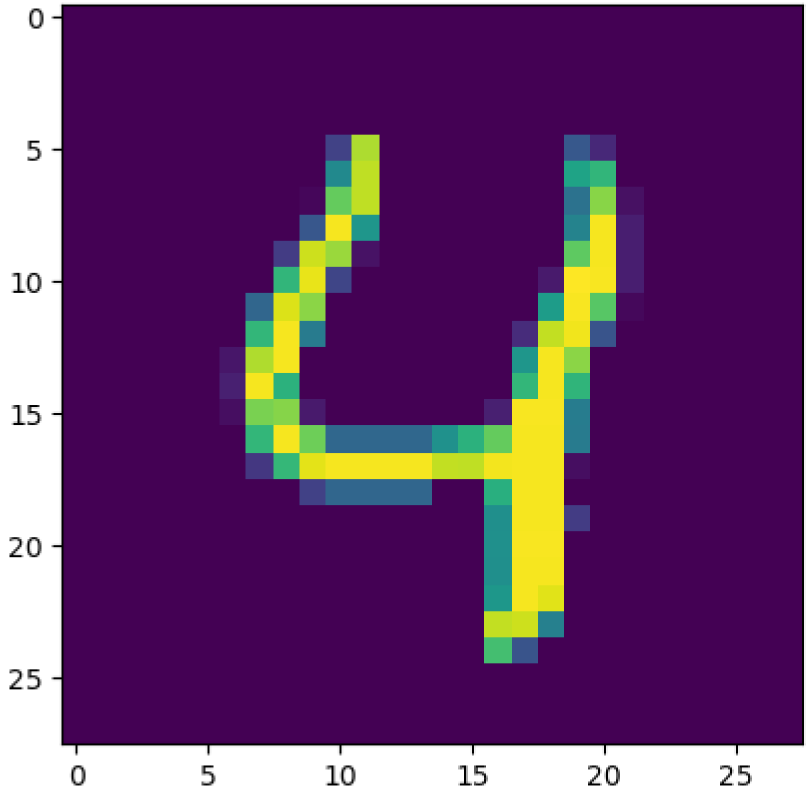
```
Epoch 1/5
1875/1875 [==============================] - 10s 4ms/step - loss: 0.6422
- accuracy: 0.8350 - val_loss: 0.3570 - val_accuracy: 0.9026
Epoch 2/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3351 -
accuracy: 0.9075 - val_loss: 0.2902 - val_accuracy: 0.9184
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2849 -
accuracy: 0.9209 - val_loss: 0.2585 - val_accuracy: 0.9279
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2544 -
accuracy: 0.9289 - val_loss: 0.2368 - val_accuracy: 0.9347
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2314 -
accuracy: 0.9358 - val_loss: 0.2165 - val_accuracy: 0.9400
```

In [14]:
```
# Evaluate the model
test_loss,test_acc=model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.2179 - a
ccuracy: 0.9383
```

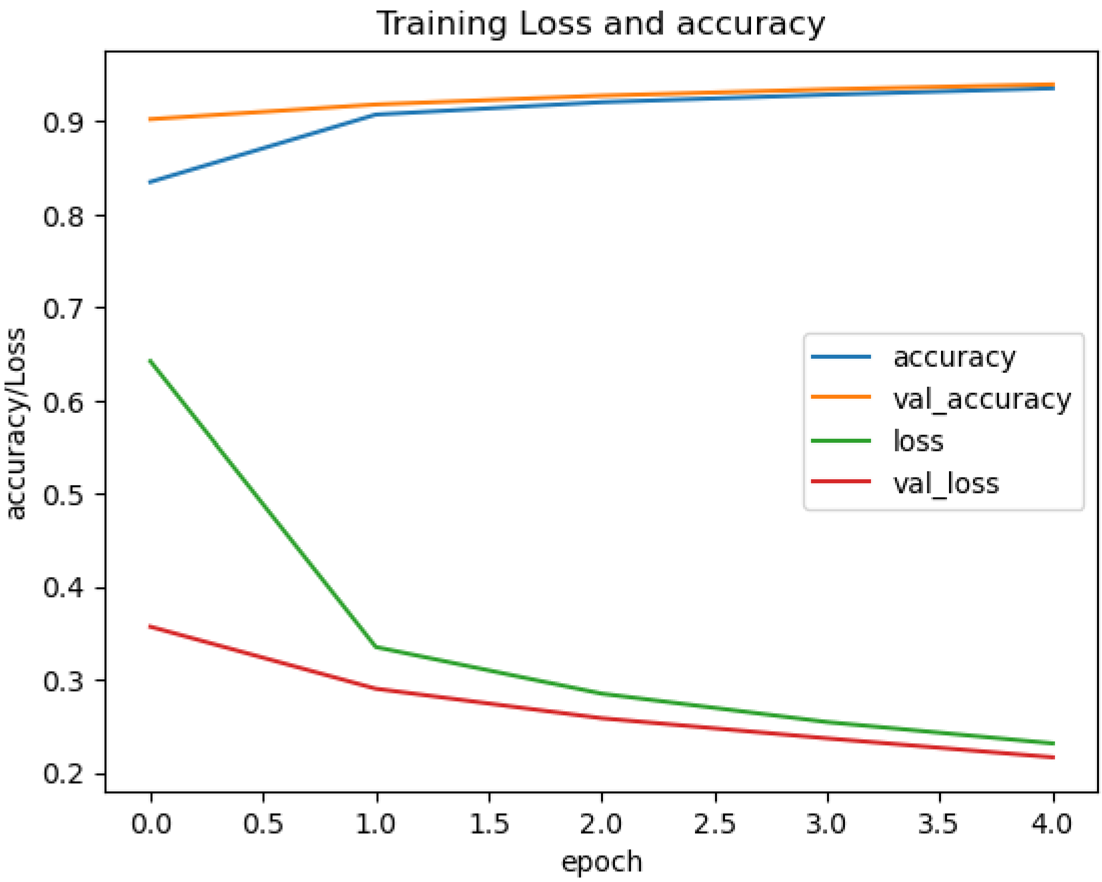
In [12]:
```
plt.imshow(x_test[4])
prediction=model.predict(x_test)   #predict the data
print(np.argmax(prediction[4]))   # print data depend on max probaiblites
#plt.show()
```

```
313/313 [==============================] - 1s 2ms/step
4
```



In [14]:
```
# graph represents the model's loss
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('Training Loss and accuracy')
plt.ylabel('accuracy/Loss')
plt.xlabel('epoch')
plt.legend(['accuracy', 'val_accuracy','loss','val_loss'])
plt.show()
```



In [ ]: