In [1]:
```
import tensorflow.compat.v1 as tf
from matplotlib import pyplot as plt;
import numpy as np;
tf.disable_v2_behavior()
```

```
WARNING:tensorflow:From C:\Users\kaush\anaconda3\lib\site-packages\tensorflow\python\compat\v2_compat.py:107: disable_resource_
variables (from tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [2]:
```
from tensorflow.keras.datasets import mnist
```

In [3]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [4]:
```
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

Out[4]:
```
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

In [5]:
```
new_x_train=[[0] for i in range(x_train.shape[0])];

for i in range(x_train.shape[0]):
    array=np.array(x_train[i]);
    array=array.reshape((784,));
    new_x_train[i]=array;

x_train=np.array(new_x_train);
```

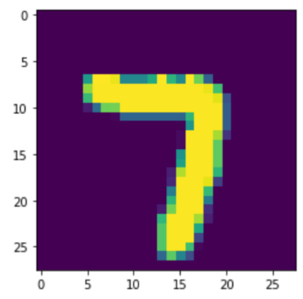
In [6]:
```
new_x_test=[[0] for i in range(x_test.shape[0])];

for i in range(x_test.shape[0]):
    array=np.array(x_test[i]);
    array=array.reshape((784,));
    new_x_test[i]=array;

x_test=np.array(new_x_test)
```

In [7]:
```
sess=tf.Session()
y_test=tf.one_hot(y_test,10);
y_train=tf.one_hot(y_train,10);
y_train=sess.run(y_train);
y_test=sess.run(y_test);
```

In [8]:
```
first_img=x_train[2090].reshape(28,28);
plt.imshow(first_img);
plt.show();
```



In [9]:
```
n_input=784;
hd1_layer=784;
hd2_layer=784;
n_output=10;
```

In [10]:
```
weights={
    'h1':tf.Variable(tf.random_normal([n_input,hd1_layer])),
    'h2':tf.Variable(tf.random_normal([hd1_layer,hd2_layer])),
    'out':tf.Variable(tf.random_normal([hd2_layer,n_output])),
}
```

In [11]:
```
biases={
    'h1':tf.Variable(tf.random_normal([hd1_layer])),
    'h2':tf.Variable(tf.random_normal([hd2_layer])),
    'out':tf.Variable(tf.random_normal([n_output])),
}
```

In [12]:
```
def forwardPropagation(x,weights,biases):
    input_layer1=tf.add(tf.matmul(x,weights['h1']),biases['h1']);
    output_layer1=tf.nn.relu(input_layer1);

    input_layer2=tf.add(tf.matmul(output_layer1,weights['h2']),biases['h2']);
    output_layer2=tf.nn.relu(input_layer2);
    output=tf.add(tf.matmul(output_layer2,weights['out']),biases['out']);
    return output;
```

In [13]:
```
x=tf.placeholder(tf.float32,[None,n_input]);
y=tf.placeholder(tf.int32,[None,n_output]);
```

In [14]:
```
pred=forwardPropagation(x,weights,biases);
predictions=tf.argmax(pred,1);
true_label=tf.argmax(y,1);
correct_pred=tf.equal(predictions,true_label);
```

In [15]:
```
cost=tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pred,labels=y));
```

```
WARNING:tensorflow:From C:\Users\kaush\anaconda3\lib\site-packages\tensorflow\python\util\dispatch.py:1176: softmax_cross_entro
py_with_logits (from tensorflow.python.ops.nn_ops) is deprecated and will be removed in a future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.
```

In [16]:
```
optimizer=tf.train.AdamOptimizer(learning_rate=0.001);
optimize=optimizer.minimize(cost)
```

In [17]:
```
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

In [18]:
```
predict,label,correct_pred,train_cost=sess.run([predictions,true_label,correct_pred,cost],feed_dict={x:x_train,y:y_train});
print("cost : ",train_cost);
```

```
cost :  1652074.4
```

In [19]:
```
from sklearn.metrics import f1_score
f1_score(label, predict, zero_division=1,average='micro')
```

Out[19]:
```
0.10151666666666669
```

In [20]:
```
batch_size=1000;
no_of_batch=int(x_train.shape[0]/batch_size);

for i in range(20):
    cost_val=0;
    start=1;
    end=1001;
    for j in range(no_of_batch):
        x_batch=x_train[start:end];
        y_batch=y_train[start:end];
        c,opt=sess.run([cost,optimize],feed_dict={x:x_batch,y:y_batch});
        cost_val +=c
        start+=1000;
        end+=1000;
    print(cost_val);
```

```
19707312.64453125
4070472.4375
2737336.666015625
2074439.619140625
1650091.1865234375
1348180.2763671875
1121426.2651367188
948127.8657226562
805133.0048828125
684115.2421875
587326.3430175781
507410.1652832031
431359.5720214844
372606.3125
327454.9130859375
286582.8623046875
244606.46875
217569.63049316406
187970.71142578125
172453.8116455078
```

In [22]:
```
predict,label,new_cost=sess.run([predictions,true_label,cost],feed_dict={x:x_train,y:y_train});
print("cost : ",new_cost);
```

```
cost :  3772.8376
```

In [23]:
```
from sklearn.metrics import f1_score
f1_score(label, predict, zero_division=1,average='micro')
```

Out[23]:
```
0.9680333333333333
```

In [24]:
```
predict_test,label_test,test_cost=sess.run([predictions,true_label,cost],feed_dict={x:x_test,y:y_test});
print("cost : ",test_cost);
```

```
cost :  14864.918
```

In [25]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(label_test, predict_test)
```

Out[25]:
```
array([[ 943,    0,   10,    0,    0,   16,    5,    2,    2,    2],
       [   0, 1102,   10,    1,    1,    1,    4,    6,   10,    0],
       [   8,    4,  978,    5,    4,    1,    5,   11,   15,    1],
       [   3,    2,   26,  888,    1,   51,    1,   12,   25,    1],
       [   1,    0,    4,    2,  939,    3,    7,    7,    7,   12],
       [   8,    1,    3,    3,    3,  848,    9,    1,   16,    0],
       [   5,    3,   10,    0,    9,   14,  909,    1,    6,    1],
       [   0,    5,   21,    6,    4,    4,    0,  974,    6,    8],
       [   4,    1,   20,    6,    7,   21,    6,    9,  895,    5],
       [   6,    4,    2,    5,   33,   10,    1,   44,   16,  888]],
      dtype=int64)
```

In [26]:
```
from sklearn.metrics import f1_score
f1_score(label_test, predict_test, zero_division=1,average='micro')
```

Out[26]:
```
0.9364
```