LAB 6

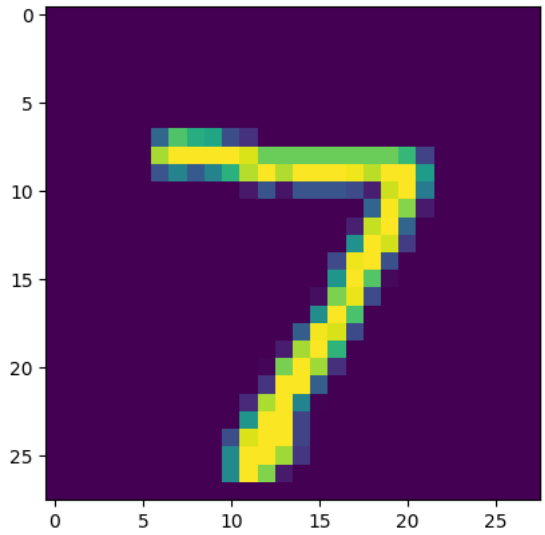
```
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
(trainX,trainY),(testX,testY)=tf.keras.datasets.mnist.load_data()
```

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

```
plt.imshow(testX[0])
```

```
<matplotlib.image.AxesImage at 0x7fabb2959a30>
```




```
trainY = tf.keras.utils.to_categorical(trainY, num_classes=10)
testY=tf.keras.utils.to_categorical(testY,num_classes=10)
testY[0]
```

```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.], dtype=float32)
```

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten  ,MaxPool2D
#create model
model = Sequential()
#add model layers
#model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28,28,1)))
#model.add(Conv2D(32, kernel_size=3, activation='relu'))
#model.add(Flatten())

model.add(Conv2D(64,kernel_size=3,padding="same", activation="relu", input_shape=(28,28,1)))
model.add(MaxPool2D())

model.add(Conv2D(32, kernel_size=3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Conv2D(16, kernel_size=3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 64)        640

 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 32)        18464

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 32)         0
 2D)
```

```
conv2d_2 (Conv2D)           (None, 7, 7, 16)          4624

max_pooling2d_2 (MaxPooling  (None, 3, 3, 16)         0
2D)

flatten (Flatten)           (None, 144)               0

dense (Dense)               (None, 10)                1450

=================================================================
Total params: 25,178
Trainable params: 25,178
Non-trainable params: 0
_________________________________________________________________
```

```
#compile model using accuracy to measure model performance
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
#train the model
model.fit(trainX, trainY, validation_data=(testX, testY), epochs=3)
```

```
Epoch 1/3
1875/1875 [==============================] - 128s 68ms/step - loss: 0.6310 - accuracy: 0.8611 - val_loss: 0.1249 - val_accuracy: 0.
Epoch 2/3
1875/1875 [==============================] - 126s 67ms/step - loss: 0.1220 - accuracy: 0.9618 - val_loss: 0.1193 - val_accuracy: 0.
Epoch 3/3
1875/1875 [==============================] - 125s 67ms/step - loss: 0.0925 - accuracy: 0.9713 - val_loss: 0.0796 - val_accuracy: 0.
<keras.callbacks.History at 0x7faba9a3bf40>
```

```
#predict first four images in the test set
model.predict(testX[:4])
```

```
1/1 [==============================] - 0s 120ms/step
array([[4.26867416e-07, 9.10553717e-06, 5.94076759e-04, 1.95196571e-06,
        3.87814140e-07, 2.31255926e-09, 1.11952515e-12, 9.99381781e-01,
        3.12352199e-06, 9.22185427e-06],
       [1.67164826e-09, 3.89006800e-06, 9.99995947e-01, 1.74245807e-09,
        2.15993695e-10, 2.82072708e-15, 7.91194055e-10, 1.13949800e-12,
        1.06531708e-07, 9.26034314e-13],
       [3.04029905e-04, 9.96822953e-01, 5.99274470e-04, 3.95510369e-06,
        1.74452295e-03, 1.85437216e-06, 5.01875184e-06, 2.97715538e-04,
        2.59263215e-05, 1.94573731e-04],
       [8.81296754e-01, 4.79501409e-07, 5.71903482e-04, 1.08808645e-05,
        4.29366537e-06, 2.08232726e-04, 1.17108524e-01, 1.92473770e-08,
        3.40935076e-04, 4.58002993e-04]], dtype=float32)
```

```
#actual results for first four images in test set
testY[:4]
```

```
array([[0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

```
plt.imshow(testX[3])
```

```
<matplotlib.image.AxesImage at 0x7fabaa38c250>
```