In [5]:
```
import tensorflow as tf
from scipy.io import loadmat
import numpy as np
import random
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Reshape, Flatten, Input, GaussianNoise
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import pandas as pd
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint
```

In [6]:
```
train = loadmat('C://Personal//Datasets//Getting started with tensorflow 2//train_32x32
test = loadmat('C://Personal//Datasets//Getting started with tensorflow 2//test_32x32.m
```

In [7]:
```
train_X=train['X']
train_y=train['y']

test_X=test['X']
test_y=test['y']
```

In [8]:
```
train_X
```

Out[8]:
```
array([[[[ 33,  84,  19, ...,  92, 190, 216],
         [ 30,  76,  54, ...,  78, 188, 217],
         [ 38,  59, 110, ..., 101, 191, 212]],

        [[ 15,  86,  20, ...,  94, 205, 221],
         [ 23,  73,  52, ...,  82, 203, 222],
         [ 19,  66, 111, ..., 105, 206, 217]],

        [[ 15,  77,  25, ..., 114, 220, 226],
         [ 17,  78,  57, ..., 101, 218, 227],
         [ 19,  56, 116, ..., 125, 220, 221]],

        ...,

        [[ 72,  90,  65, ..., 200, 229, 200],
         [ 65,  78, 144, ..., 201, 231, 199],
         [ 56,  69, 223, ..., 203, 224, 191]],

        [[ 82,  88,  78, ..., 192, 229, 193],
         [ 77,  77, 148, ..., 193, 229, 188],
         [ 57,  67, 218, ..., 195, 224, 182]],

        [[ 89,  88,  98, ..., 190, 229, 197],
         [ 79,  78, 158, ..., 191, 228, 189],
         [ 59,  66, 220, ..., 193, 223, 186]]],


       [[[ 28,  85,  21, ...,  92, 183, 204],
         [ 39,  77,  53, ...,  78, 182, 205],
         [ 35,  61, 110, ..., 103, 186, 202]],

        [[ 14,  83,  19, ...,  93, 200, 210],
         [ 25,  73,  52, ...,  80, 199, 211],
         [ 22,  64, 106, ..., 106, 201, 208]],

        [[ 14,  74,  25, ..., 111, 218, 220],
         [ 20,  69,  56, ...,  98, 217, 221],
         [ 17,  59, 111, ..., 124, 218, 217]],
```

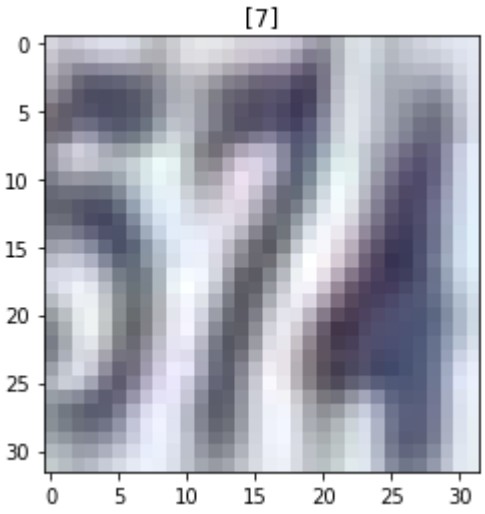
```
 ...,

 [[ 40,  89,  63, ..., 181, 227, 201],
  [ 39,  82, 137, ..., 180, 228, 199],
  [ 50,  64, 208, ..., 184, 223, 193]],

 [[ 67,  88,  91, ..., 177, 227, 195],
  [ 58,  79, 153, ..., 176, 226, 191],
  [ 52,  70, 214, ..., 180, 222, 186]],

 [[ 83,  88, 130, ..., 183, 228, 196],
  [ 78,  81, 180, ..., 182, 224, 190],
  [ 60,  67, 229, ..., 187, 221, 186]]],


[[[ 40,  83,  21, ...,  99, 171, 198],
  [ 41,  76,  53, ...,  84, 170, 198],
  [ 38,  60, 110, ..., 112, 175, 197]],

 [[ 18,  78,  20, ...,  94, 189, 202],
  [ 21,  77,  51, ...,  81, 189, 202],
  [ 26,  58, 106, ..., 110, 193, 201]],

 [[ 16,  61,  22, ..., 107, 213, 212],
  [ 17,  50,  52, ...,  94, 213, 211],
  [ 23,  54, 106, ..., 123, 215, 210]],

 ...,

 [[ 23,  90,  79, ..., 167, 231, 203],
  [ 29,  85, 147, ..., 166, 230, 200],
  [ 45,  63, 210, ..., 171, 226, 196]],

 [[ 35,  88, 125, ..., 172, 229, 198],
  [ 42,  83, 181, ..., 171, 226, 194],
  [ 44,  66, 230, ..., 176, 223, 191]],

 [[ 72,  85, 178, ..., 185, 227, 195],
  [ 69,  82, 218, ..., 184, 223, 190],
  [ 53,  70, 254, ..., 189, 220, 187]]],


...,


[[[ 86, 100,  88, ...,  99, 187, 233],
  [ 81,  98, 162, ...,  94, 185, 226],
  [ 75,  72, 237, ..., 110, 186, 228]],

 [[ 87,  98,  89, ...,  96, 204, 230],
  [ 82,  94, 163, ...,  91, 202, 224],
  [ 71,  76, 238, ..., 109, 199, 225]],

 [[ 82,  95,  84, ..., 108, 217, 228],
  [ 79,  93, 156, ..., 103, 217, 223],
  [ 65,  73, 230, ..., 124, 210, 221]],

 ...,

 [[104, 104,  62, ..., 210, 204, 198],
  [104, 104, 142, ..., 207, 200, 196],
  [ 87,  86, 227, ..., 204, 195, 190]],

 [[104, 102,  67, ..., 206, 196, 184],
```

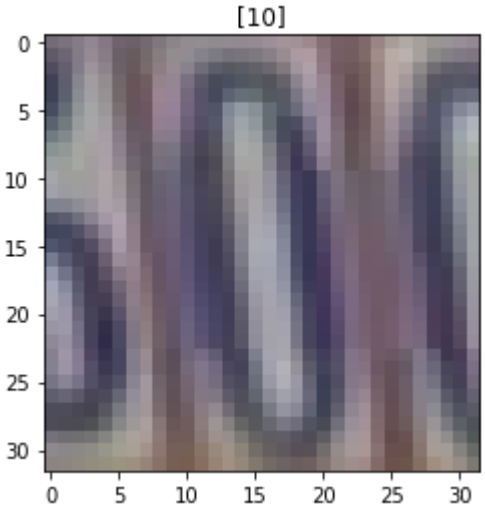
```
        [105, 102, 144, ..., 202, 193, 183],
        [ 81,  87, 226, ..., 200, 189, 177]],

       [[103, 100,  74, ..., 203, 196, 189],
        [105, 101, 145, ..., 197, 193, 187],
        [ 78,  78, 225, ..., 199, 189, 182]]],


      [[[ 84, 103,  88, ...,  94, 186, 231],
        [ 86, 104, 164, ...,  91, 184, 226],
        [ 64,  79, 240, ..., 103, 185, 228]],

       [[ 86, 106,  87, ...,  94, 198, 229],
        [ 79, 104, 160, ...,  91, 197, 224],
        [ 72,  79, 237, ..., 104, 194, 225]],

       [[ 82, 103,  88, ..., 110, 211, 227],
        [ 76, 103, 159, ..., 107, 211, 223],
        [ 72,  87, 237, ..., 121, 204, 222]],

       ...,

       [[110, 103,  60, ..., 219, 222, 195],
        [103, 104, 141, ..., 218, 216, 194],
        [ 84,  86, 230, ..., 215, 212, 186]],

       [[106, 103,  61, ..., 218, 214, 181],
        [105, 103, 141, ..., 215, 209, 181],
        [ 85,  87, 228, ..., 212, 205, 173]],

       [[106, 105,  65, ..., 212, 208, 186],
        [104,  99, 143, ..., 209, 205, 183],
        [ 86,  81, 226, ..., 209, 200, 177]]],


      [[[ 85, 103,  84, ...,  88, 190, 230],
        [ 88, 106, 160, ...,  87, 188, 226],
        [ 68,  82, 238, ...,  94, 190, 227]],

       [[ 89, 103,  81, ...,  85, 199, 230],
        [ 82, 105, 154, ...,  84, 197, 226],
        [ 72,  87, 233, ...,  93, 194, 227]],

       [[ 85, 104,  87, ..., 105, 208, 229],
        [ 79, 106, 158, ..., 103, 208, 225],
        [ 67,  91, 238, ..., 114, 201, 226]],

       ...,

       [[111, 113,  63, ..., 217, 232, 190],
        [104, 103, 144, ..., 217, 227, 190],
        [ 87,  88, 235, ..., 214, 223, 181]],

       [[109, 104,  62, ..., 221, 226, 178],
        [105, 104, 143, ..., 220, 221, 177],
        [ 86,  88, 232, ..., 219, 216, 169]],

       [[103, 103,  63, ..., 218, 218, 181],
        [106,  98, 145, ..., 217, 213, 178],
        [ 79,  80, 231, ..., 218, 209, 171]]]], dtype=uint8)
```

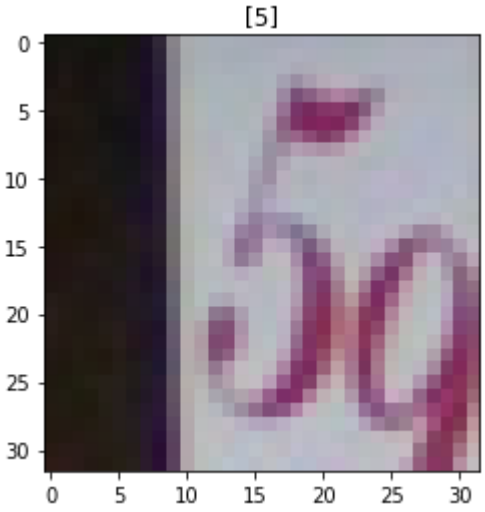
In [9]:
```
n=10
image_num=[]
for i in range(n):
```

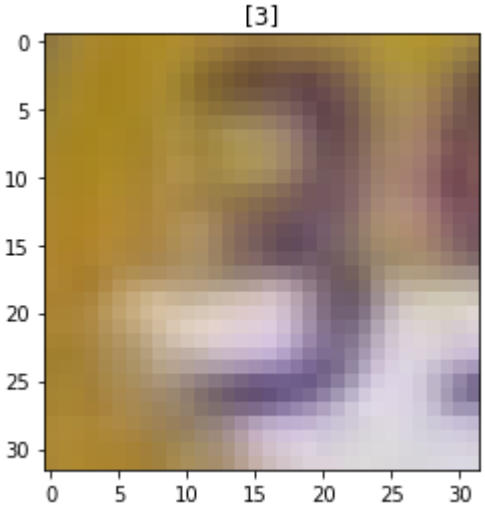
```
    image_num.append(random.randint(0, train_X.shape[3]))

for i in range(len(image_num)):
    plt.imshow(train_X[:,:,:,image_num[i]])
    plt.title(train_y[image_num[i],:])
    plt.show()
```

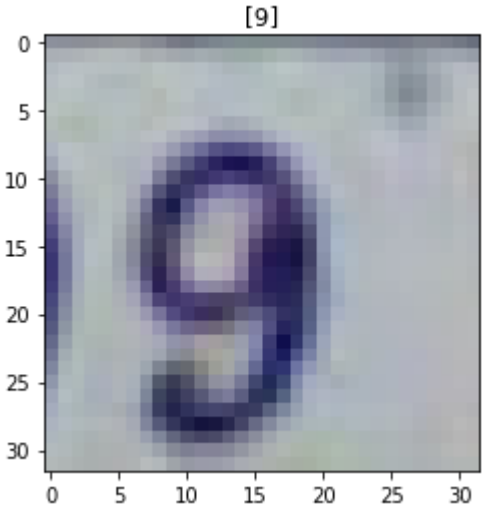
```
C:\Users\Salehin\anaconda3\lib\site-packages\matplotlib\text.py:1165: FutureWarning: ele
mentwise comparison failed; returning scalar instead, but in the future will perform ele
mentwise comparison
  if s != self._text:
```

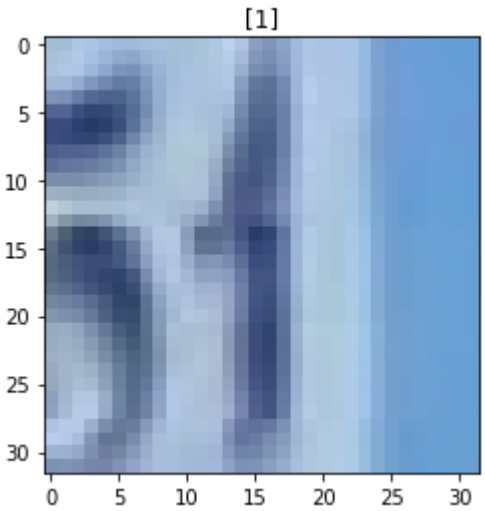


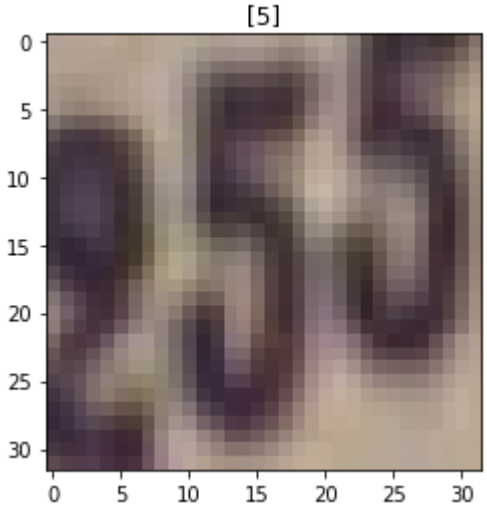


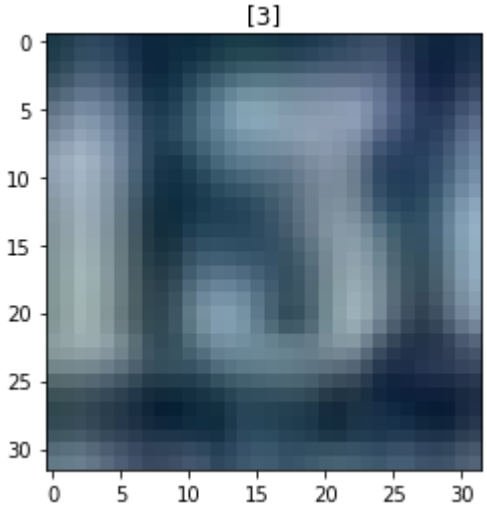


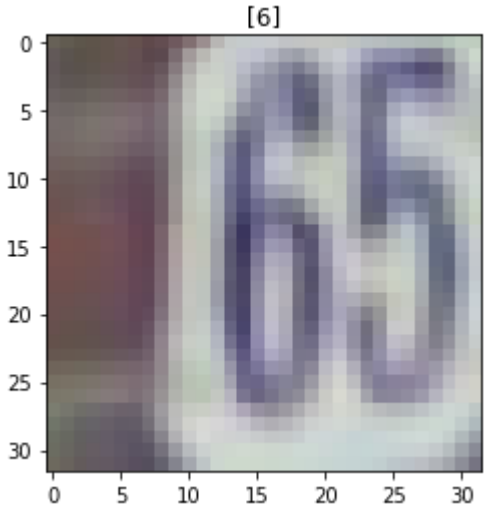

[3]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

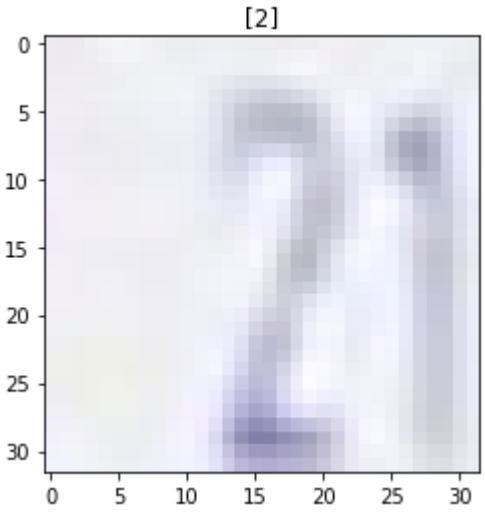

[9]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


[1]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


[5]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


[3]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


[6]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

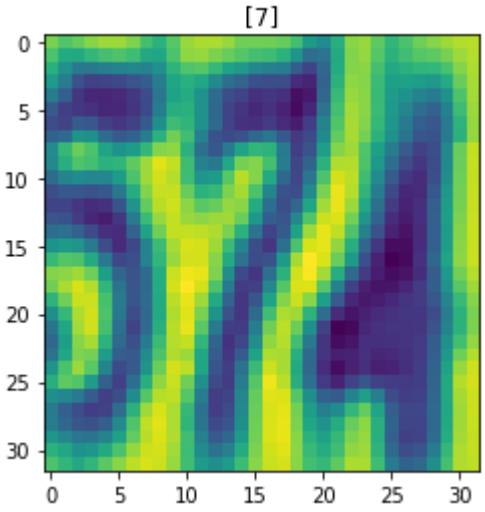


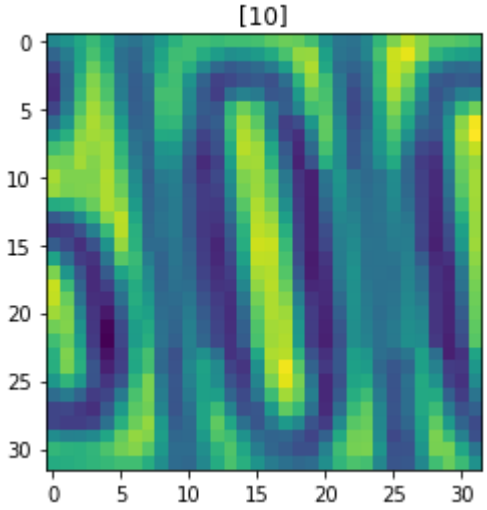
In [10]:
```
train_X.shape
```

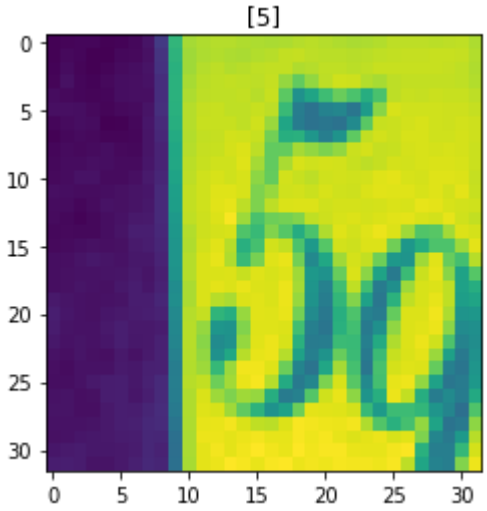
Out[10]: (32, 32, 3, 73257)

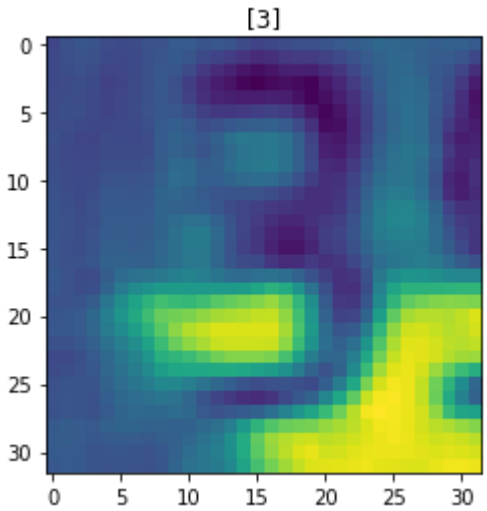
In [11]:
```
#COnvert to grayscale image
train_X=np.array(train['X'])
train_y=np.array(train['y'])

test_X=np.array(test['X'])
test_y=np.array(test['y'])

train_X_gray=np.mean(train_X,axis=2)
test_X_gray=np.mean(test_X,axis=2)
```

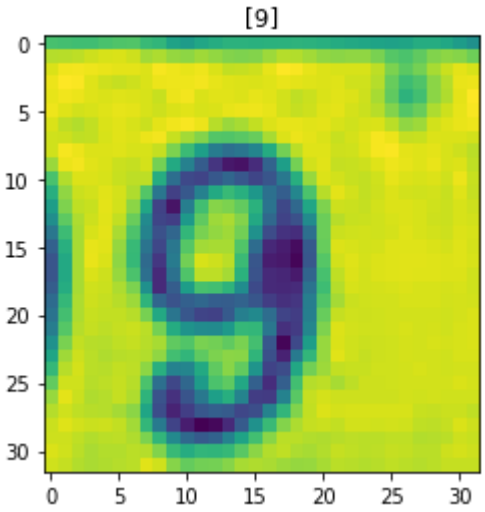
In [12]:
```
for i in range(len(image_num)):
    plt.imshow(train_X_gray[:,:,image_num[i]])
    plt.title(train_y[image_num[i],:])
    plt.show()
```

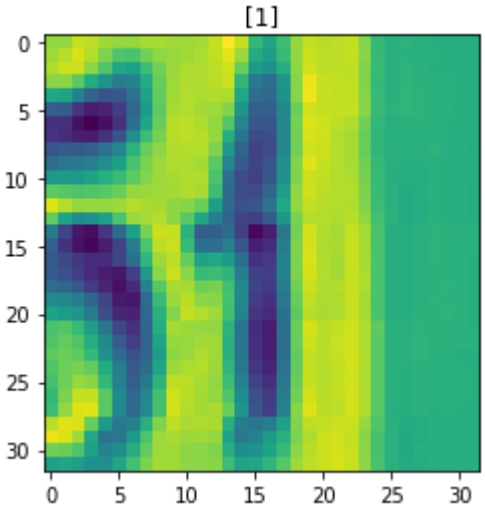


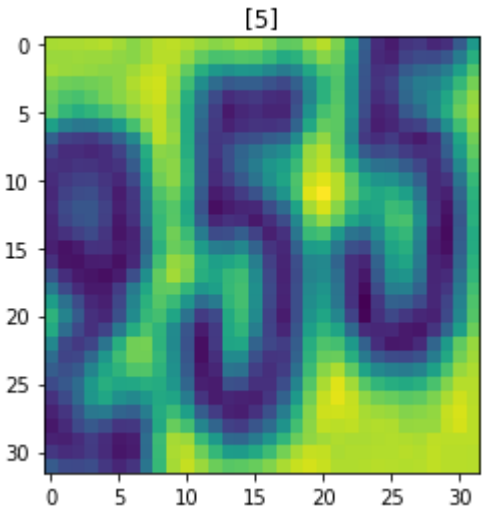

[10]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

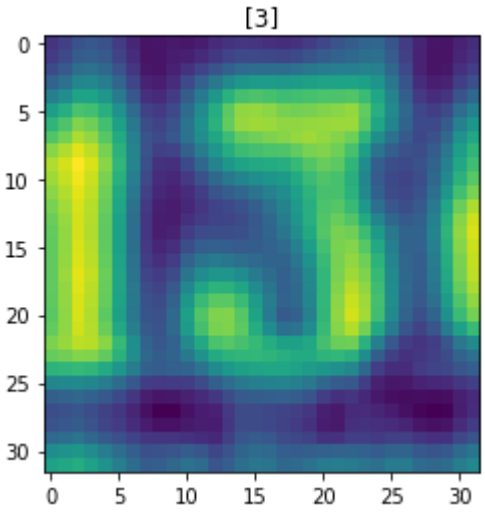

[5]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

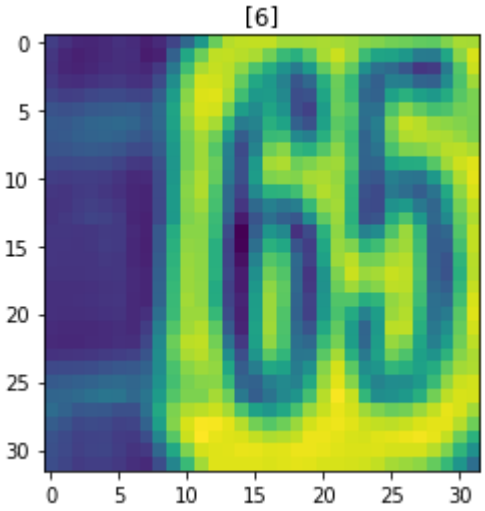

[3]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

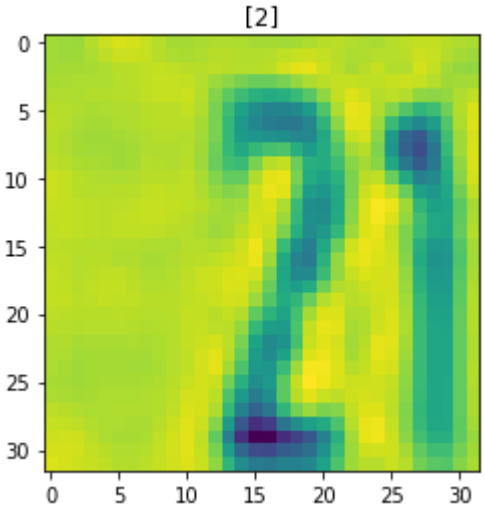

[9]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


[1]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


[5]
0
5
10
15
20
25
30
0
5
10
15
20
25
30







In [13]:
```
#check_shape
train_X_gray.shape
```

Out[13]: (32, 32, 73257)

In [14]:
```
#The number of examples column should be the 1st one, that's what the neural network ex
train_X_changed=np.zeros((73257,32,32))
for i in range(train_X_gray.shape[2]):
```

```
    train_X_changed[i,:,:]=train_X_gray[:,:,i]
```

In [15]:
```
#Now the dimension
train_X_changed.shape
```

Out[15]:
```
(73257, 32, 32)
```

In [16]:
```
test_X_changed=np.zeros((26032,32,32))
for i in range(test_X_gray.shape[2]):
    test_X_changed[i,:,:]=test_X_gray[:,:,i]
```

In [17]:
```
test_X_changed.shape
```

Out[17]:
```
(26032, 32, 32)
```

In [18]:
```
#cnn expects 3 dimensional input. so making an extra dimension for image
train_X_cnn=train_X_changed[...,np.newaxis]
test_X_cnn=test_X_changed[...,np.newaxis]
```

In [19]:
```
train_X_cnn.shape
```

Out[19]:
```
(73257, 32, 32, 1)
```

In [20]:
```
#convert labels to categorical variables
y_example=to_categorical(train_y)
y_example
```

Out[20]:
```
array([[0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.]], dtype=float32)
```

In [21]:
```
y_test_example=to_categorical(test_y)
```

In [22]:
```
#Normalize
train_X_changed=train_X_changed/255
test_X_changed=test_X_changed/255
```

In [23]:
```
train_X_cnn=train_X_cnn/255
test_X_cnn=test_X_cnn/255
```

In [24]:
```
#BUild a Neural Network classfier
def get_model():
    """
    This function should build a Sequential model according to the above specification.
    weights are initialised by providing the input_shape argument in the first layer, g
    function argument.
    Your function should return the model.
    """
    model=Sequential([Flatten(input_shape=(32,32)),
                      Dense(512,activation='relu'),
                      Dense(256,activation='relu'),
                      Dense(11,activation='softmax')
```

```
                    ])
    return model
```

In [25]:
```
model=get_model()
```

In [26]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 512)               524800
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 11)                2827
=================================================================
Total params: 658,955
Trainable params: 658,955
Non-trainable params: 0
_________________________________________________________________
```

In [27]:
```
es=EarlyStopping(monitor="val_loss",patience=3)
checkpoint=ModelCheckpoint('nncheckpoint/checkpoint_model.{epoch}',save_best_only=True)
```

In [28]:
```
model.compile(optimizer='Adam',loss="categorical_crossentropy",metrics=['accuracy'])
```

In [29]:
```
history=model.fit(train_X_changed,y_example,validation_split=0.15,epochs=20,batch_size=
```

```
Epoch 1/20
973/973 [==============================] - 9s 7ms/step - loss: 2.0619 - accuracy: 0.2626
- val_loss: 1.6781 - val_accuracy: 0.4060
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.1\assets
Epoch 2/20
973/973 [==============================] - 8s 8ms/step - loss: 1.4435 - accuracy: 0.5170
- val_loss: 1.2609 - val_accuracy: 0.6090
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.2\assets
Epoch 3/20
973/973 [==============================] - 8s 8ms/step - loss: 1.2088 - accuracy: 0.6162
- val_loss: 1.1298 - val_accuracy: 0.6476
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.3\assets
Epoch 4/20
973/973 [==============================] - 9s 9ms/step - loss: 1.0937 - accuracy: 0.6603
- val_loss: 1.0948 - val_accuracy: 0.6609
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.4\assets
Epoch 5/20
973/973 [==============================] - 8s 8ms/step - loss: 1.0225 - accuracy: 0.6847
- val_loss: 0.9691 - val_accuracy: 0.7031
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.5\assets
Epoch 6/20
973/973 [==============================] - 8s 8ms/step - loss: 0.9792 - accuracy: 0.6995
- val_loss: 1.0159 - val_accuracy: 0.6800
Epoch 7/20
973/973 [==============================] - 10s 10ms/step - loss: 0.9430 - accuracy: 0.71
14 - val_loss: 0.9313 - val_accuracy: 0.7106
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.7\assets
Epoch 8/20
973/973 [==============================] - 8s 8ms/step - loss: 0.9095 - accuracy: 0.7240
- val_loss: 0.9046 - val_accuracy: 0.7214
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.8\assets
Epoch 9/20
```

```
973/973 [==============================] - 7s 7ms/step - loss: 0.8874 - accuracy: 0.7317
- val_loss: 0.8829 - val_accuracy: 0.7317
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.9\assets
Epoch 10/20
973/973 [==============================] - 7s 7ms/step - loss: 0.8663 - accuracy: 0.7374
- val_loss: 0.8834 - val_accuracy: 0.7329
Epoch 11/20
973/973 [==============================] - 7s 7ms/step - loss: 0.8514 - accuracy: 0.7413
- val_loss: 0.8605 - val_accuracy: 0.7360
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.11\assets
Epoch 12/20
973/973 [==============================] - 8s 8ms/step - loss: 0.8333 - accuracy: 0.7474
- val_loss: 0.8602 - val_accuracy: 0.7369
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.12\assets
Epoch 13/20
973/973 [==============================] - 7s 7ms/step - loss: 0.8200 - accuracy: 0.7517
- val_loss: 0.8191 - val_accuracy: 0.7513
INFO:tensorflow:Assets written to: nncheckpoint\checkpoint_model.13\assets
Epoch 14/20
973/973 [==============================] - 9s 9ms/step - loss: 0.8083 - accuracy: 0.7549
- val_loss: 0.8511 - val_accuracy: 0.7433
Epoch 15/20
973/973 [==============================] - 7s 7ms/step - loss: 0.7968 - accuracy: 0.7581
- val_loss: 0.8511 - val_accuracy: 0.7453
Epoch 16/20
973/973 [==============================] - 7s 7ms/step - loss: 0.7850 - accuracy: 0.7627
- val_loss: 0.8385 - val_accuracy: 0.7460
```

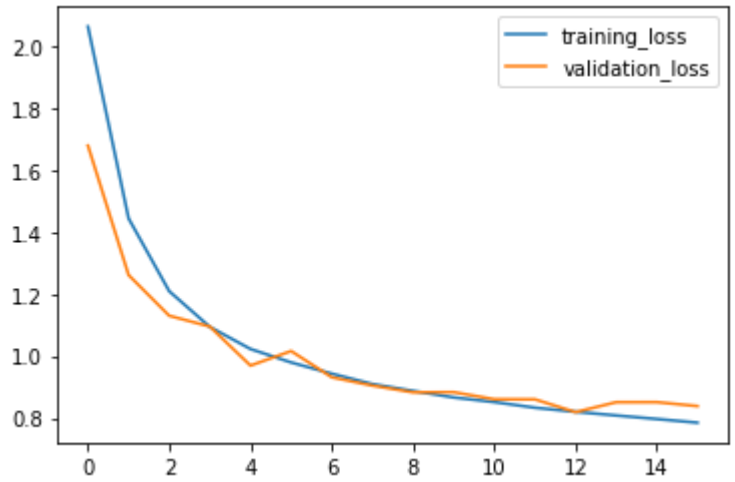
In [30]:
```
model.evaluate(test_X_changed,y_test_example)
```

```
814/814 [==============================] - 2s 2ms/step - loss: 0.9182 - accuracy: 0.7306
```

Out[30]:
```
[0.9181862473487854, 0.7305623888969421]
```

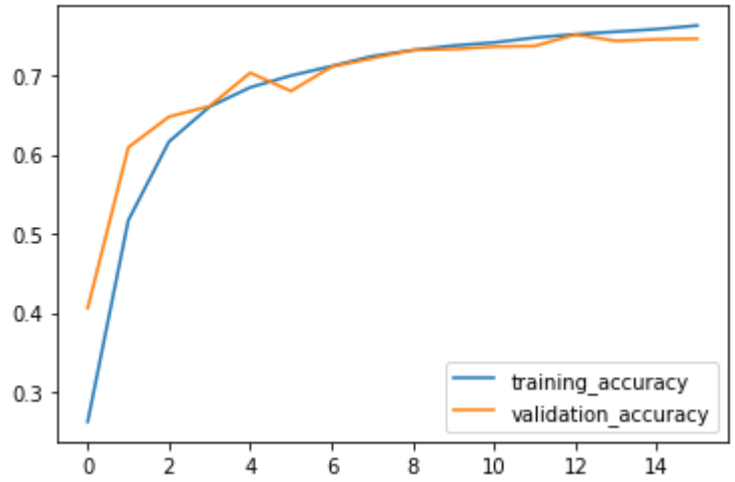
In [31]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training_loss','validation_loss'])
```

Out[31]:
```
<matplotlib.legend.Legend at 0x19abf4f4fd0>
```



In [32]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training_accuracy','validation_accuracy'])
```

Out[32]:
```
<matplotlib.legend.Legend at 0x19a499ecfd0>
```



In [33]:
```
from tensorflow.keras.layers import Dense, Flatten, Softmax, Conv2D, MaxPooling2D,Dropo

def get_model(rate):
    """
    This function should build a Sequential model according to the above specification.
    weights are initialised by providing the input_shape argument in the first layer, g
    function argument.
    Your function should return the model.
    """
    model=Sequential([Conv2D(8, kernel_size=(3,3),padding='same', activation='relu',inp
                      Dropout(rate),Conv2D(8, kernel_size=(3,3),padding='same', activat
                  Dense(64,activation='relu'), BatchNormalization(),Dropout(rate),Dense
    return model
```

In [34]:
```
train_X.shape
```

Out[34]:
```
(32, 32, 3, 73257)
```

In [35]:
```
get_conv_model=get_model(0.05)
```

In [36]:
```
get_conv_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 8)         80
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 8)         32
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 8)         0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 8)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 8)         584
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 8)         32
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 8)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
```

```
dense_3 (Dense)              (None, 64)                131136
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                4160
_________________________________________________________________
batch_normalization_3 (Batch (None, 64)                256
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 11)                715
=================================================================
Total params: 137,251
Trainable params: 136,963
Non-trainable params: 288
_________________________________________________________________
```

In [37]:
```
es=EarlyStopping(monitor="val_loss",patience=3)
checkpoint=ModelCheckpoint('cnncheckpoint/checkpoint_model.{epoch}',save_weights_only=F
```

In [38]:
```
get_conv_model.compile(optimizer='Adam',loss="categorical_crossentropy",metrics=['accur
```

In [ ]:
```

```

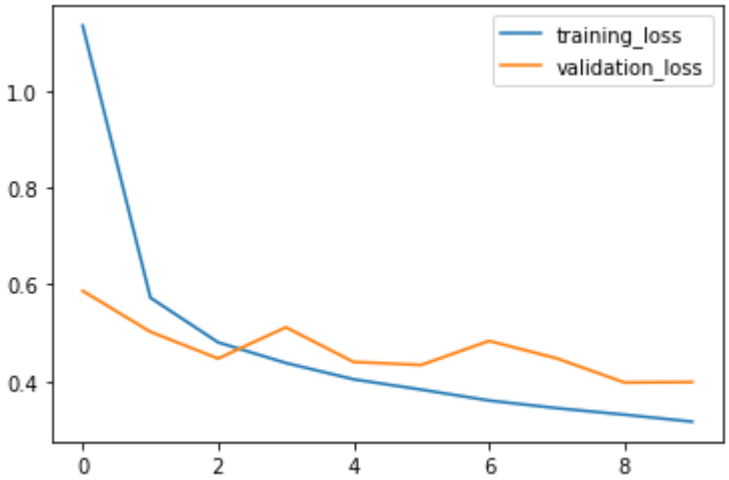
In [39]:
```
history2=get_conv_model.fit(train_X_cnn,y_example,validation_split=0.15,epochs=10,batch
```

```
Epoch 1/10
973/973 [==============================] - 46s 46ms/step - loss: 1.1365 - accuracy: 0.63
49 - val_loss: 0.5869 - val_accuracy: 0.8237
INFO:tensorflow:Assets written to: cnncheckpoint\checkpoint_model.1\assets
Epoch 2/10
973/973 [==============================] - 50s 51ms/step - loss: 0.5726 - accuracy: 0.82
49 - val_loss: 0.5021 - val_accuracy: 0.8470
INFO:tensorflow:Assets written to: cnncheckpoint\checkpoint_model.2\assets
Epoch 3/10
973/973 [==============================] - 54s 56ms/step - loss: 0.4802 - accuracy: 0.85
25 - val_loss: 0.4467 - val_accuracy: 0.8625
INFO:tensorflow:Assets written to: cnncheckpoint\checkpoint_model.3\assets
Epoch 4/10
973/973 [==============================] - 54s 55ms/step - loss: 0.4375 - accuracy: 0.86
56 - val_loss: 0.5115 - val_accuracy: 0.8393
Epoch 5/10
973/973 [==============================] - 53s 54ms/step - loss: 0.4037 - accuracy: 0.87
48 - val_loss: 0.4396 - val_accuracy: 0.8672
INFO:tensorflow:Assets written to: cnncheckpoint\checkpoint_model.5\assets
Epoch 6/10
973/973 [==============================] - 55s 56ms/step - loss: 0.3823 - accuracy: 0.88
11 - val_loss: 0.4334 - val_accuracy: 0.8691
INFO:tensorflow:Assets written to: cnncheckpoint\checkpoint_model.6\assets
Epoch 7/10
973/973 [==============================] - 51s 52ms/step - loss: 0.3597 - accuracy: 0.88
90 - val_loss: 0.4833 - val_accuracy: 0.8559
Epoch 8/10
973/973 [==============================] - 48s 49ms/step - loss: 0.3439 - accuracy: 0.89
40 - val_loss: 0.4471 - val_accuracy: 0.8597
Epoch 9/10
973/973 [==============================] - 43s 44ms/step - loss: 0.3305 - accuracy: 0.89
70 - val_loss: 0.3970 - val_accuracy: 0.8808
INFO:tensorflow:Assets written to: cnncheckpoint\checkpoint_model.9\assets
Epoch 10/10
```

```
973/973 [==============================] - 44s 45ms/step - loss: 0.3159 - accuracy: 0.90
02 - val_loss: 0.3980 - val_accuracy: 0.8819
```

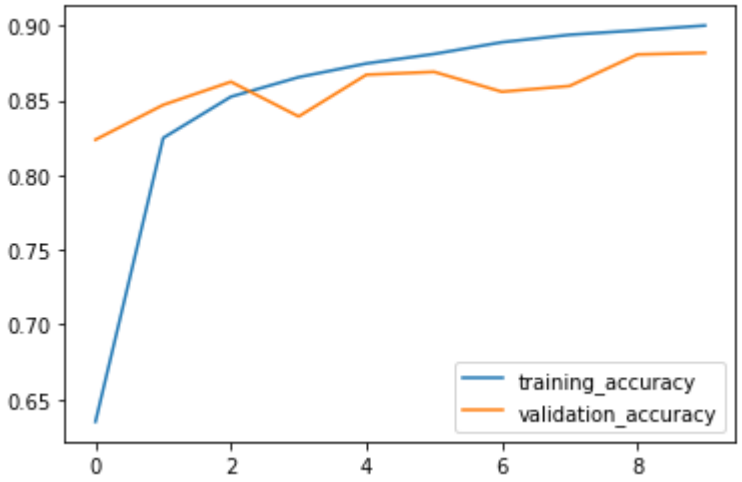
In [40]:
```
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.legend(['training_loss','validation_loss'])
```

Out[40]:
```
<matplotlib.legend.Legend at 0x19aad881940>
```



In [41]:
```
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.legend(['training_accuracy','validation_accuracy'])
```

Out[41]:
```
<matplotlib.legend.Legend at 0x19aaf168220>
```



In [42]:
```
get_conv_model.evaluate(test_X_cnn,y_test_example)
```

```
814/814 [==============================] - 5s 6ms/step - loss: 0.4423 - accuracy: 0.8678
```

Out[42]:
```
[0.4423384368419647, 0.8677781224250793]
```

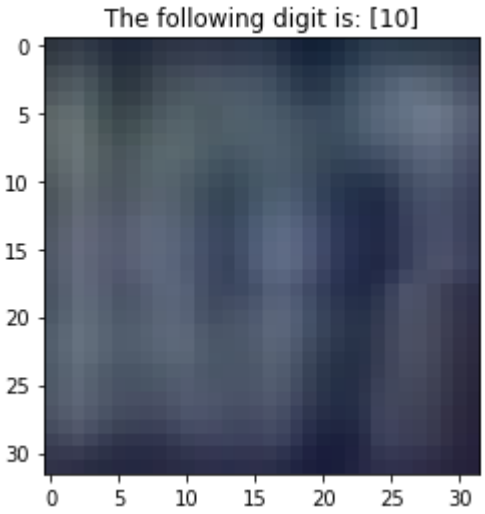
In [44]:
```
get_conv_model.load_weights(tf.train.latest_checkpoint('cnncheckpoint'))
```

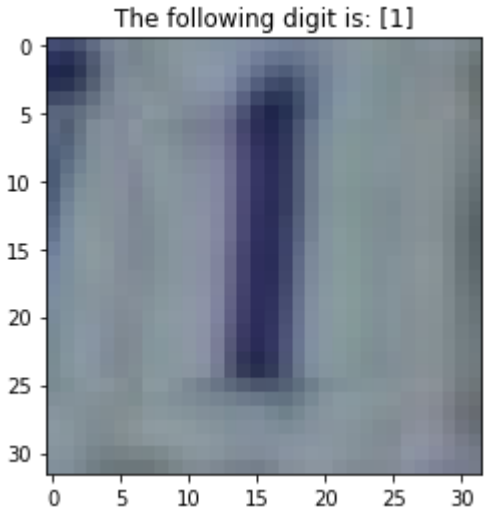
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-44-66f6155d6529> in <module>
----> 1 get_conv_model.load_weights(tf.train.latest_checkpoint('cnncheckpoint'))

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in load_weights
```

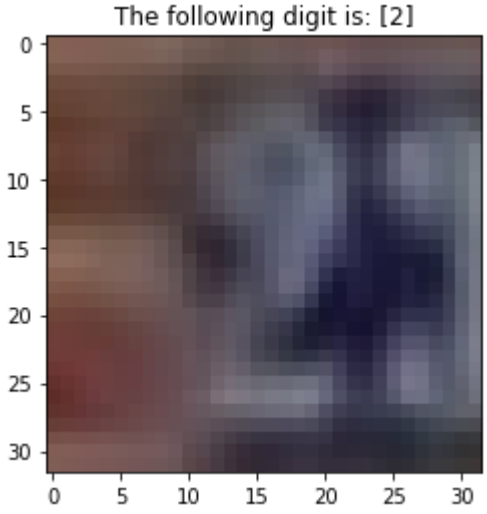
```
(self, filepath, by_name, skip_mismatch, options)
   2293           'True when by_name is True.')
   2294
-> 2295     filepath, save_format = _detect_save_format(filepath)
   2296     if save_format == 'tf':
   2297       status = self._trackable_saver.restore(filepath, options)

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in _detect_save
_format(filepath)
   2910
   2911   filepath = path_to_string(filepath)
-> 2912   if saving_utils.is_hdf5_filepath(filepath):
   2913     return filepath, 'h5'
   2914

~\anaconda3\lib\site-packages\tensorflow\python\keras\saving\saving_utils.py in is_hdf5_
filepath(filepath)
    323
    324 def is_hdf5_filepath(filepath):
--> 325   return (filepath.endswith('.h5') or filepath.endswith('.keras') or
    326           filepath.endswith('.hdf5'))

AttributeError: 'NoneType' object has no attribute 'endswith'
```

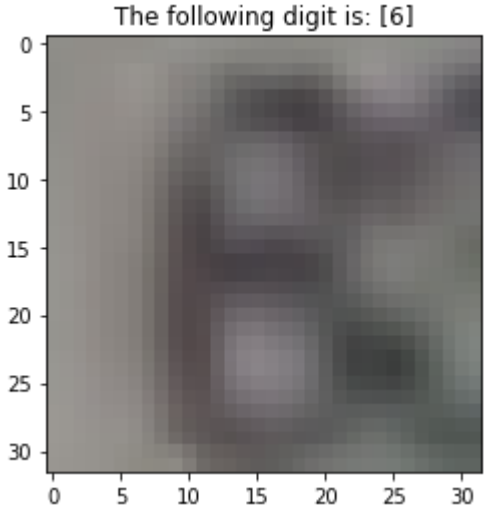
In [45]:

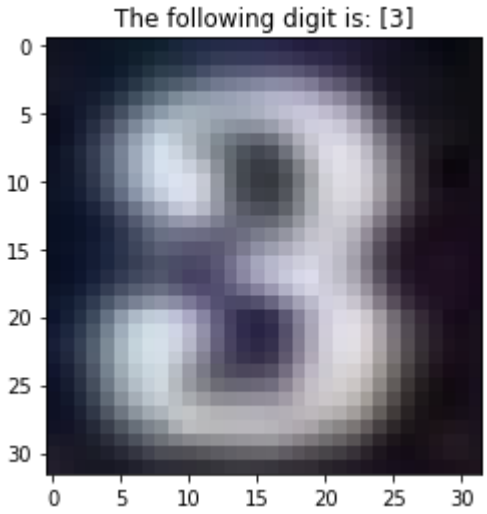
```
#randomly select 5 images and display their labels
n=5
image_num=[]
for i in range(n):
    image_num.append(random.randint(0, test_X.shape[3]))

for i in range(len(image_num)):
    plt.imshow(test_X[:,:,:,image_num[i]])
    plt.title('The following digit is: {}'.format(test_y[image_num[i],:]))
    plt.show()
```




The following digit is: [1]


The following digit is: [2]


The following digit is: [6]



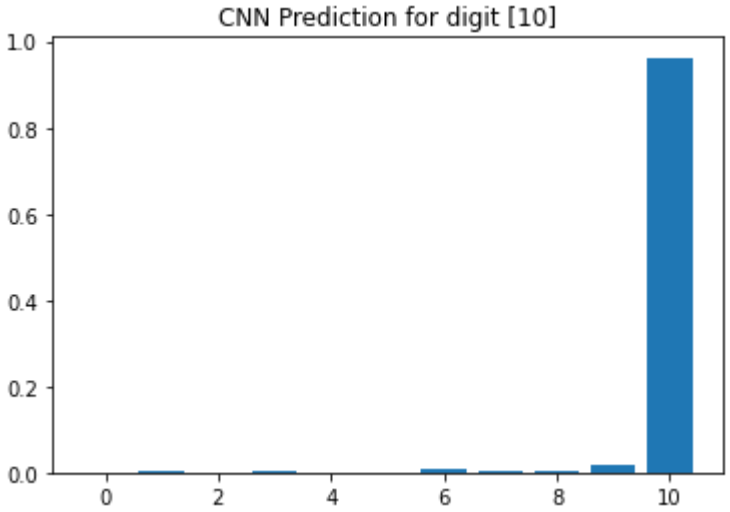
In [46]:
```
predictions=get_conv_model.predict(test_X_cnn[image_num[0:5],:,:,:])
```

In [48]:
```
xaxis=[i for i in range(0,11)]
xaxis
```

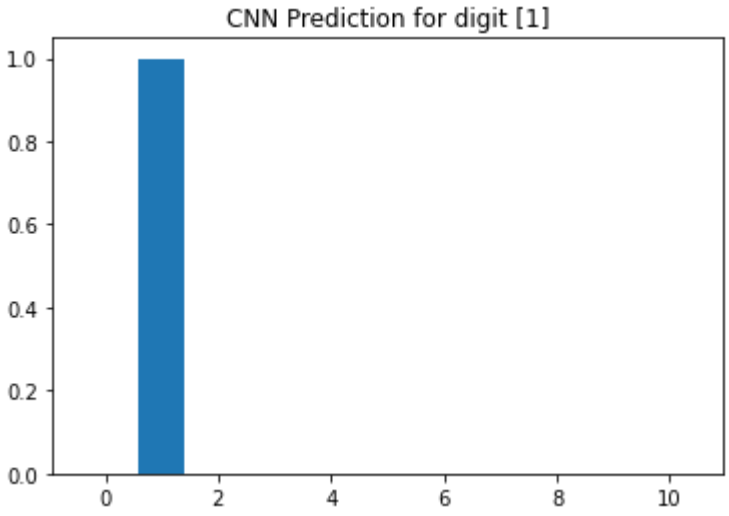
Out[48]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [49]:
```
for i in range(len(predictions)):
    plt.bar(xaxis,predictions[i])
    plt.title(label="CNN Prediction for digit {}".format(test_y[image_num[i],:]))
    print("cnn prediction for this image is {}".format(np.argmax(predictions[i])))
    plt.show()
```

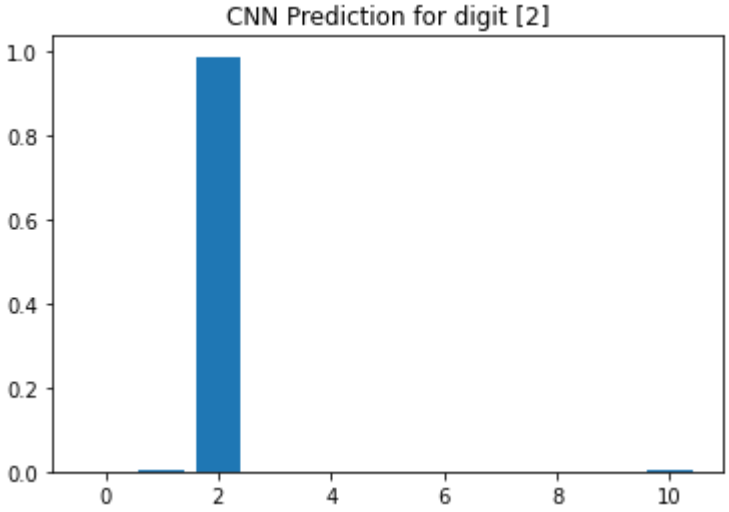
```
cnn prediction for this image is 10
```



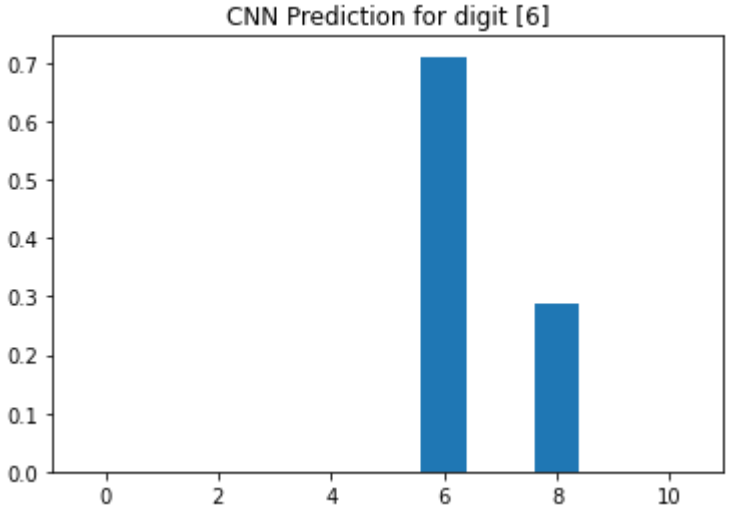

```
cnn prediction for this image is 1
```



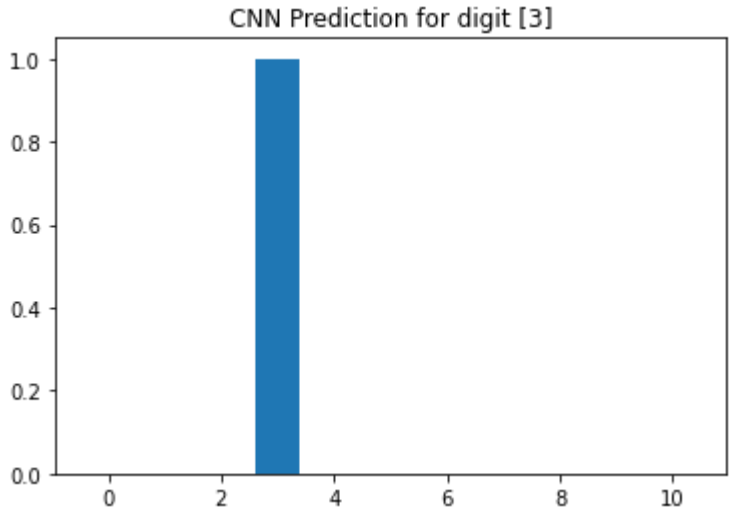

```
cnn prediction for this image is 2
```




```
cnn prediction for this image is 6
```



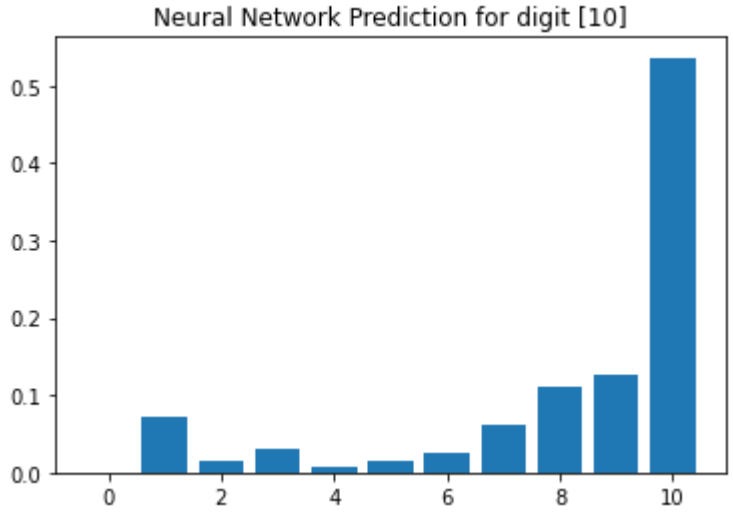

```
cnn prediction for this image is 3
```



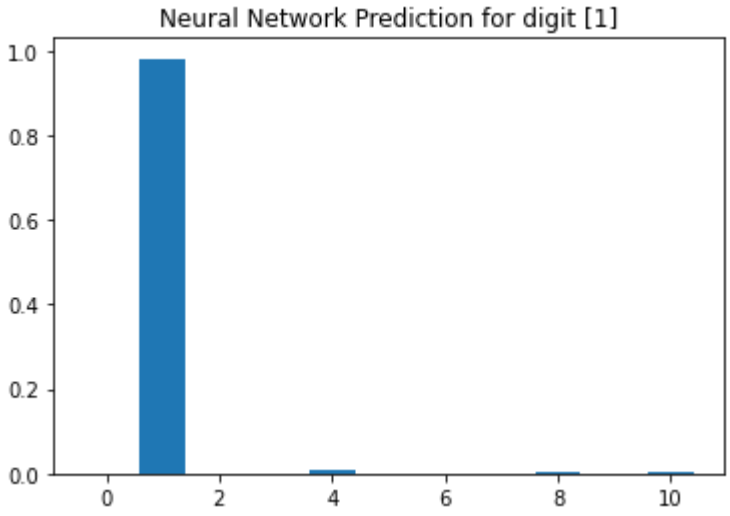
In [50]:
```
#Let's check the predictions with neural network

predictions=model.predict(test_X_changed[image_num[0:5],:,:])
```

In [51]:
```
for i in range(len(predictions)):
    plt.bar(xaxis,predictions[i])
    plt.title(label="Neural Network Prediction for digit {}".format(test_y[image_num[i]
    print("Neural Network prediction for this image is {}".format(np.argmax(predictions
    plt.show()
```

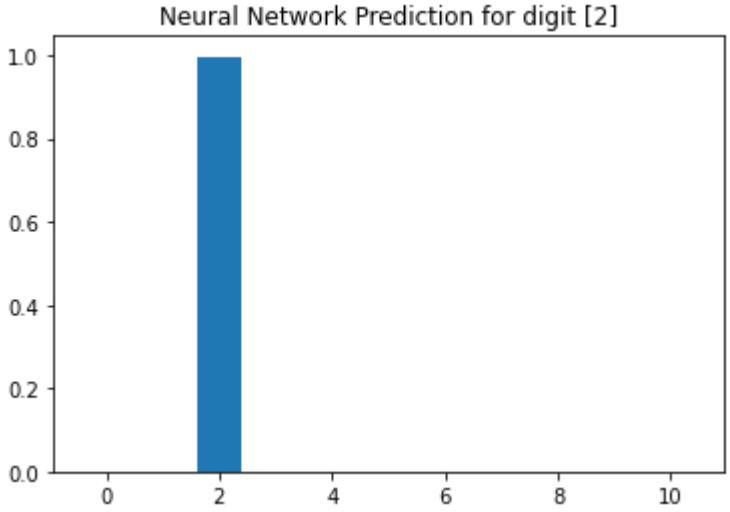
```
Neural Network prediction for this image is 10
```



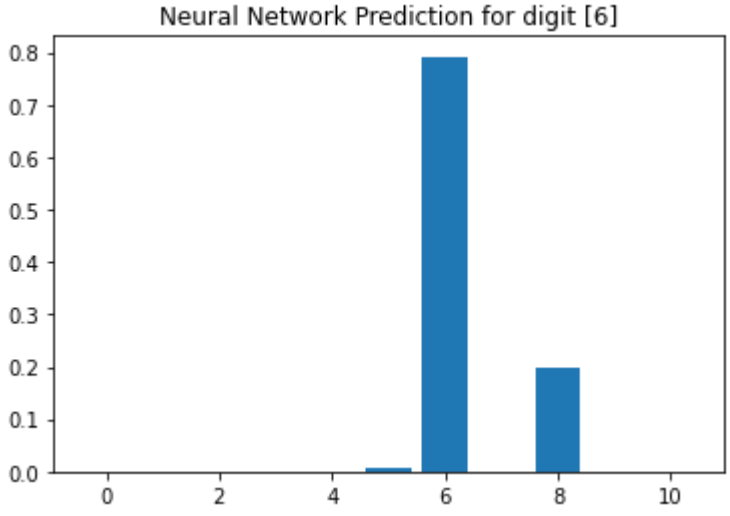

```
Neural Network prediction for this image is 1
```



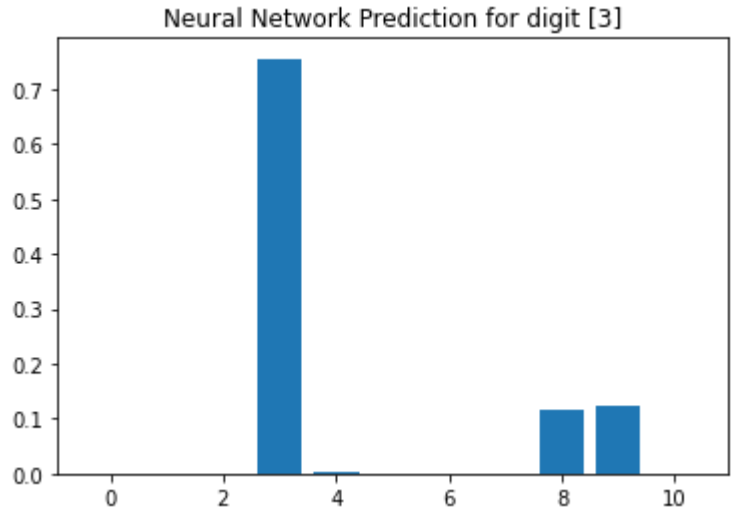

```
Neural Network prediction for this image is 2
```




```
Neural Network prediction for this image is 6
```




```
Neural Network prediction for this image is 3
```



In [52]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\salehin\anaconda3\lib\site-packages
(6.0.7)
Requirement already satisfied: nbformat>=4.4 in c:\users\salehin\anaconda3\lib\site-pack
ages (from nbconvert) (5.0.8)
Requirement already satisfied: bleach in c:\users\salehin\anaconda3\lib\site-packages (f
rom nbconvert) (3.2.1)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\salehin\anaconda3\lib\si
te-packages (from nbconvert) (1.4.3)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\salehin\anaconda3\lib\site
-packages (from nbconvert) (0.3)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\salehin\anaconda3\lib\site-
packages (from nbconvert) (0.8.4)
Requirement already satisfied: testpath in c:\users\salehin\anaconda3\lib\site-packages
(from nbconvert) (0.4.4)
Requirement already satisfied: pygments>=2.4.1 in c:\users\salehin\anaconda3\lib\site-pa
ckages (from nbconvert) (2.7.2)
Requirement already satisfied: jupyterlab-pygments in c:\users\salehin\anaconda3\lib\sit
e-packages (from nbconvert) (0.1.2)
Requirement already satisfied: jupyter-core in c:\users\salehin\anaconda3\lib\site-packa
ges (from nbconvert) (4.6.3)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\salehin\anaconda3\lib
\site-packages (from nbconvert) (0.5.1)
Requirement already satisfied: traitlets>=4.2 in c:\users\salehin\anaconda3\lib\site-pac
kages (from nbconvert) (5.0.5)
Requirement already satisfied: jinja2>=2.4 in c:\users\salehin\anaconda3\lib\site-packag
es (from nbconvert) (2.11.2)
Requirement already satisfied: defusedxml in c:\users\salehin\anaconda3\lib\site-package
s (from nbconvert) (0.6.0)
Requirement already satisfied: ipython-genutils in c:\users\salehin\anaconda3\lib\site-p
ackages (from nbformat>=4.4->nbconvert) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\salehin\anaconda3\lib
\site-packages (from nbformat>=4.4->nbconvert) (3.2.0)
Requirement already satisfied: webencodings in c:\users\salehin\anaconda3\lib\site-packa
ges (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: packaging in c:\users\salehin\anaconda3\lib\site-packages
(from bleach->nbconvert) (20.4)
Requirement already satisfied: six>=1.9.0 in c:\users\salehin\anaconda3\lib\site-package
s (from bleach->nbconvert) (1.15.0)
Requirement already satisfied: pywin32>=1.0; sys_platform == "win32" in c:\users\salehin
\anaconda3\lib\site-packages (from jupyter-core->nbconvert) (227)
Requirement already satisfied: nest-asyncio in c:\users\salehin\anaconda3\lib\site-packa
ges (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.4.2)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\salehin\anaconda3\lib\s
ite-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (6.1.7)
```

```
Requirement already satisfied: async-generator in c:\users\salehin\anaconda3\lib\site-pa
ckages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.10)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\salehin\anaconda3\lib\site-p
ackages (from jinja2>=2.4->nbconvert) (1.1.1)
Requirement already satisfied: attrs>=17.4.0 in c:\users\salehin\anaconda3\lib\site-pack
ages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (20.3.0)
Requirement already satisfied: setuptools in c:\users\salehin\anaconda3\lib\site-package
s (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (50.3.1.post20201107)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\salehin\anaconda3\lib\site
-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (0.17.3)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\salehin\anaconda3\lib\site-p
ackages (from packaging->bleach->nbconvert) (2.4.7)
Requirement already satisfied: pyzmq>=13 in c:\users\salehin\anaconda3\lib\site-packages
(from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (19.0.2)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\salehin\anaconda3\lib\si
te-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.1)
Requirement already satisfied: tornado>=4.1 in c:\users\salehin\anaconda3\lib\site-packa
ges (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.0.4)
Note: you may need to restart the kernel to use updated packages.
```