The sparks foundation-Data Science and Business Analytics-AUG21

Task 1: Optical Character Recognition by deep learning(Level-Beginner)

Task by: Deepti Taram

Domain: Computer Vision and IoT

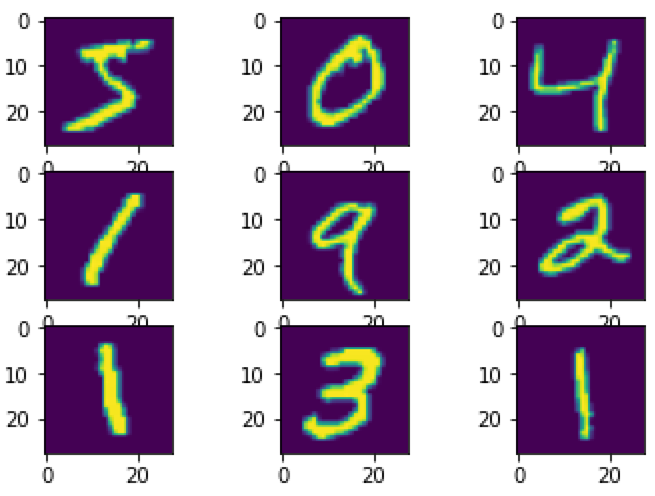
Problem Statement: To recognize the hand written digit classification

In [1]:
```
!pip install tensorflow --quiet
```

In [2]:
```
import tensorflow as tf
from matplotlib import pyplot as plt
import numpy as np
```

In [3]:
```
objects = tf.keras.datasets.mnist
(training_images,training_labels),(test_images, test_labels)= objects.load_data()
```

In [4]:
```
for i in range(9):
    #define subplot
    plt.subplot(330+1+i)
    # plot raw pixel data
    plt.imshow(training_images[i])
```



In [5]:
```
print(training_images.shape)
print(training_images[0])
```

```
(60000, 28, 28)
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   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 [6]:
```
training_images = training_images/255.0
test_images = test_images/255.0
```

In [7]:
```
import numpy as np
import keras.models
from keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [8]:
```
from keras.models import Sequential, Model
from keras.layers import Dense, Flatten, BatchNormalization, Activation, Dropout
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from keras import optimizers

model = Sequential()
model.add(Conv2D(filters=32, kernel_size=2, padding='same', input_shape=(28,28,1)))
model.add(Activation('relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

In [9]:
```
model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28,28)),
                                    tf.keras.layers.Dense(128,activation='relu'),
                                    tf.keras.layers.Dense(10,activation=tf.nn.softmax)])
```

In [10]:
```
model.compile(optimizer = tf.keras.optimizers.Adam(),
              loss = 'sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [11]:
```
model.fit(training_images,training_labels, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 2s 913us/step - loss: 0.2622 - accuracy: 0.9257
Epoch 2/5
1875/1875 [==============================] - 2s 884us/step - loss: 0.1165 - accuracy: 0.9656
Epoch 3/5
1875/1875 [==============================] - 2s 937us/step - loss: 0.0791 - accuracy: 0.9758
Epoch 4/5
1875/1875 [==============================] - 2s 918us/step - loss: 0.0600 - accuracy: 0.9815
Epoch 5/5
1875/1875 [==============================] - 2s 908us/step - loss: 0.0467 - accuracy: 0.9855
```

Out[11]:
```
<keras.callbacks.History at 0x1d4ec66c7f0>
```

After 5 iteration we have achieved accuracy of 98.55%. But thsi is accuracy og training session.

In [12]:
```
print(model.evaluate(test_images,test_labels))
```

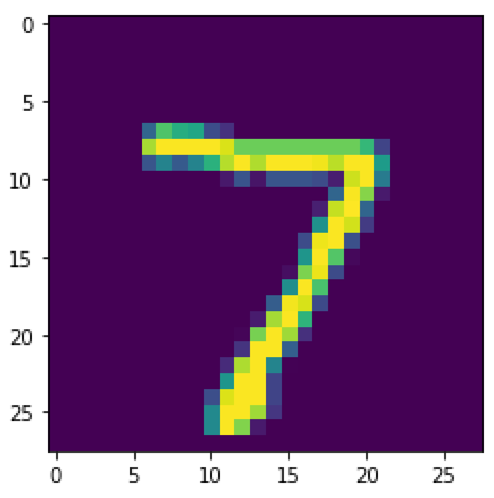
```
313/313 [==============================] - 0s 771us/step - loss: 0.0731 - accuracy: 0.9759
[0.07313447445631027, 0.9758999943733215]
```

Now here after testing the model we get the accuracy of 97.59% slightly lower than that of training secsion

Other methods to check the accuracy of how its performing

In [14]:
```
plt.imshow(test_images[0])
prediction= model.predict(test_images)
print(np.argmax(prediction[0]))
```

```
7
```



In [ ]: