Name - Devasy Patel

Roll No - 20BCE057

Practical-3B

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-pytho
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all file

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserv
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside o
```

```
/kaggle/input/digit-recognizer/sample_submission.csv
/kaggle/input/digit-recognizer/train.csv
/kaggle/input/digit-recognizer/test.csv
```

In [2]:

```
import pandas as pd
import tensorflow as tf
```

In [3]:

```
train = pd.read_csv('/kaggle/input/digit-recognizer/train.csv')
train.head(1)
```

Out[3]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pix
0	1	0	0	0	0	0	0	0	0	0	...	0	

1 rows × 785 columns

In [4]:

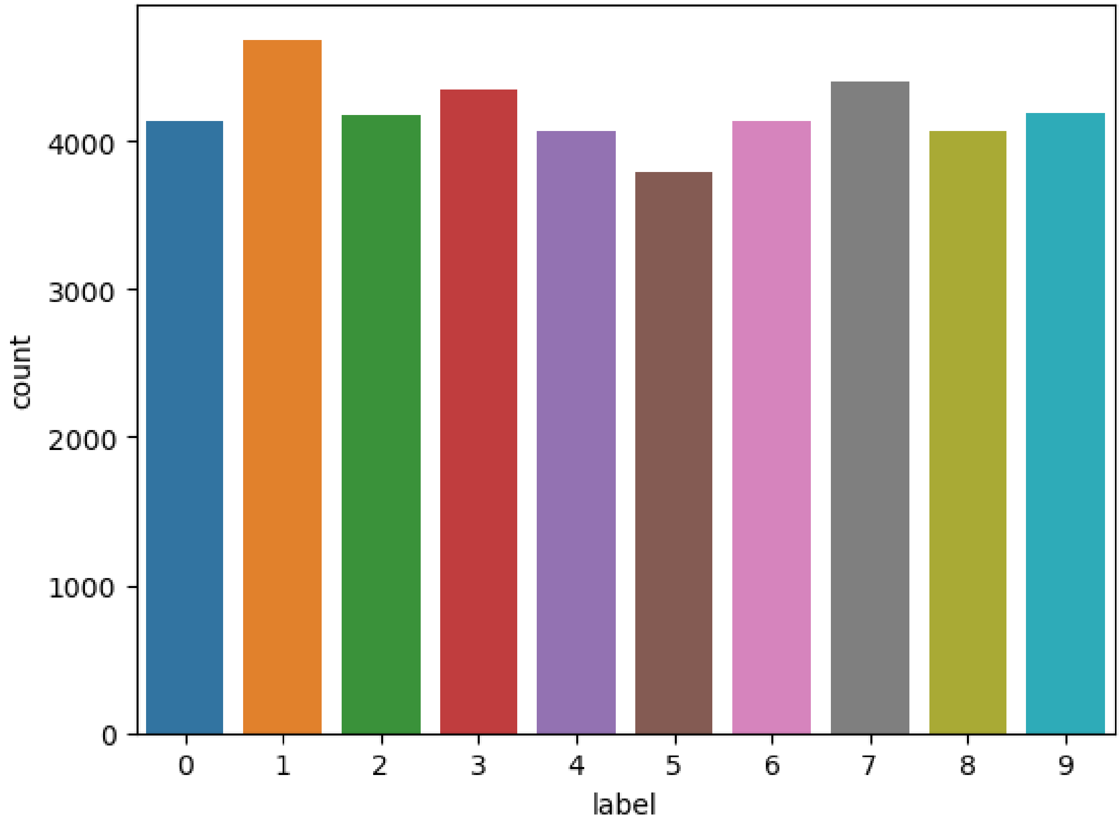
```
y = train.iloc[::,0]
X = train.iloc[::,1:]
```

In [5]:

```
import seaborn as sns
sns.countplot(x='label', data=train)
```

Out[5]:

```
<AxesSubplot:xlabel='label', ylabel='count'>
```



In [6]:

```
test = pd.read_csv('/kaggle/input/digit-recognizer/test.csv')
test.head()
```

Out[6]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pi
0	0	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 784 columns

In [7]:

```
X = X/255
Xtest = test/255
```

In [8]:

```
X = X.values.reshape([-1, 28, 28, 1])
Xtest = Xtest.values.reshape([-1, 28, 28, 1])
```

In [9]:

```
from keras.utils.np_utils import to_categorical
y = to_categorical(y, num_classes = 10)
y.shape
```

Out[9]:

```
(42000, 10)
```

In [10]:

```
from sklearn.model_selection import train_test_split
Xtrain, Xval, ytrain, yval = train_test_split(X, y, test_size=0.1, random_state=42)
Xtrain.shape, Xval.shape, ytrain.shape, yval.shape, Xtest.shape
```

Out[10]:

```
((37800, 28, 28, 1),
 (4200, 28, 28, 1),
 (37800, 10),
 (4200, 10),
 (28000, 28, 28, 1))
```

In [11]:

```
from tensorflow.keras import models,layers
```

In [12]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=([28,28,1])))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
#model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

In [13]:

```
model.compile(optimizer='adam', loss=['categorical_crossentropy'], metrics=['accuracy'])
```

In [14]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 1600)              0         
                                                                 
 dense (Dense)               (None, 64)                102464    
                                                                 
 dense_1 (Dense)             (None, 32)                2080      
                                                                 
 dense_2 (Dense)             (None, 10)                330       
                                                                 
=================================================================
Total params: 123,690
Trainable params: 123,690
Non-trainable params: 0
_________________________________________________________________
```

In [15]:

```
es = tf.keras.callbacks.EarlyStopping(monitor='loss', mode='min',verbose=1, patience=5)
```

In [16]:

```
model.fit(Xtrain, ytrain, batch_size=100, epochs=20, callbacks=[es])
```

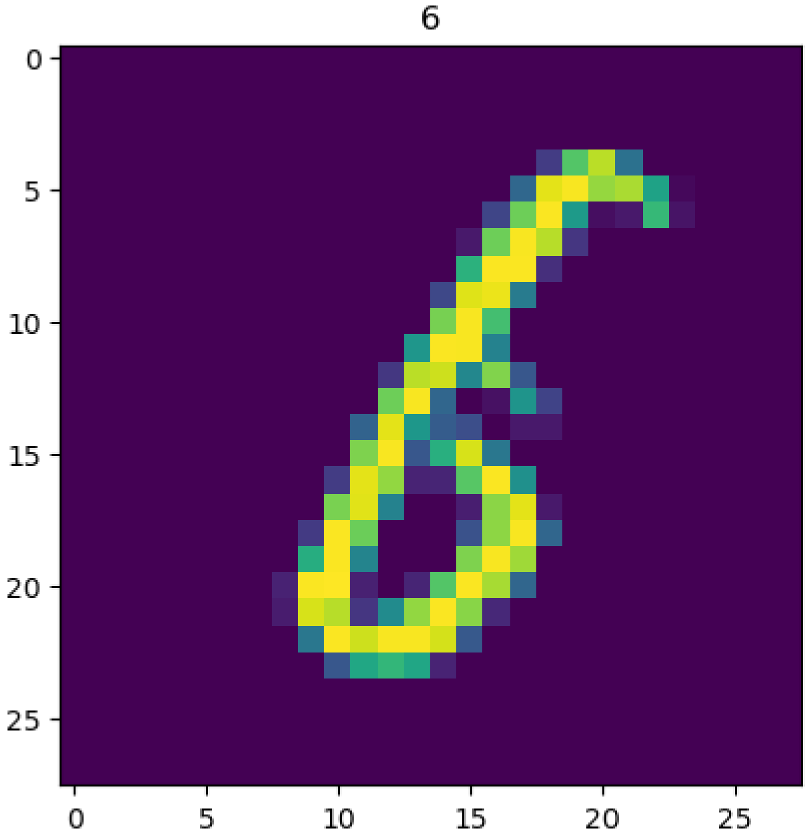
```
Epoch 1/20
378/378 [==============================] - 17s 43ms/step - loss: 0.3265 -
accuracy: 0.9014
Epoch 2/20
378/378 [==============================] - 16s 44ms/step - loss: 0.0833 -
accuracy: 0.9742
Epoch 3/20
378/378 [==============================] - 16s 43ms/step - loss: 0.0587 -
accuracy: 0.9817
Epoch 4/20
378/378 [==============================] - 16s 41ms/step - loss: 0.0441 -
accuracy: 0.9862
Epoch 5/20
378/378 [==============================] - 16s 41ms/step - loss: 0.0361 -
accuracy: 0.9886
Epoch 6/20
378/378 [==============================] - 15s 40ms/step - loss: 0.0290 -
accuracy: 0.9913
Epoch 7/20
378/378 [==============================] - 16s 41ms/step - loss: 0.0265 -
accuracy: 0.9914
Epoch 8/20
378/378 [==============================] - 15s 40ms/step - loss: 0.0205 -
accuracy: 0.9934
Epoch 9/20
378/378 [==============================] - 15s 41ms/step - loss: 0.0167 -
accuracy: 0.9948
Epoch 10/20
378/378 [==============================] - 15s 40ms/step - loss: 0.0161 -
accuracy: 0.9946
Epoch 11/20
378/378 [==============================] - 15s 41ms/step - loss: 0.0116 -
accuracy: 0.9963
Epoch 12/20
378/378 [==============================] - 15s 40ms/step - loss: 0.0130 -
accuracy: 0.9958
Epoch 13/20
378/378 [==============================] - 15s 41ms/step - loss: 0.0089 -
accuracy: 0.9970
Epoch 14/20
378/378 [==============================] - 15s 40ms/step - loss: 0.0091 -
accuracy: 0.9968
Epoch 15/20
378/378 [==============================] - 15s 41ms/step - loss: 0.0084 -
accuracy: 0.9971
Epoch 16/20
378/378 [==============================] - 15s 39ms/step - loss: 0.0076 -
accuracy: 0.9976
Epoch 17/20
378/378 [==============================] - 15s 39ms/step - loss: 0.0056 -
accuracy: 0.9983
Epoch 18/20
378/378 [==============================] - 15s 41ms/step - loss: 0.0060 -
accuracy: 0.9982
Epoch 19/20
378/378 [==============================] - 15s 40ms/step - loss: 0.0055 -
accuracy: 0.9979
Epoch 20/20
378/378 [==============================] - 15s 41ms/step - loss: 0.0057 -
accuracy: 0.9980
```

Out[16]:

```
<keras.callbacks.History at 0x7fcab9ace2d0>
```

In [17]:

```
import matplotlib.pyplot as plt
plt.imshow(Xtrain[145][:,:,0])
plt.title(ytrain[145].argmax());
```



In [18]:

```
ypred = model.predict(Xtest)
ypred = np.argmax(ypred,axis=1)
```

```
875/875 [==============================] - 5s 5ms/step
```

In [19]:

```
ypred
```

Out[19]:

```
array([2, 0, 9, ..., 3, 9, 2])
```

In [20]:

```
df = pd.DataFrame({'ImageId': list(range(1, len(ypred)+1)), 'Label': ypred})
df.to_csv('submission.csv', index=False)
```