In [ ]:

```
#!pip install np_utils
```

In [ ]:

```
#import tensorflow as tf
#print(tf.__version__)
```

In [ ]:

```
#!pip uninstall tensorflow
#!pip install tensorflow==2.2.0
```

In [ ]:

```
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,MaxPooling2D,Activation,Dropout,Flatten,BatchNor
malization,MaxPool1D
from tensorflow.keras.optimizers import SGD,Adam
import numpy as np
```

Read the data

In [ ]:

```
train_data = pd.read_csv('train.csv')
train_data.head()
```

Out[ ]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 785 columns

In [ ]:

```
test_data = pd.read_csv('test.csv')
test_data.head()
```

Out[ ]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 784 columns

In [ ]:

```
train_data['label'].value_counts()
```

Out[ ]:

```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```

Finding null values

In [ ]:

```
def f1_null(train_data):
  count=train_data.isnull().sum().sum()
  return count
```

In [ ]:

```
count_null = f1_null(train_data)
print(count_null)
```

```
0
```

To check the duplicates values

In [ ]:

```
def f1_duplicates(train_data):
  count=train_data.duplicated().sum().sum()
  return count
```

In [ ]:

```
count_duplicates = f1_duplicates(train_data)
print(count_duplicates)
```

```
0
```

In [ ]:

```
print(train_data.shape)
print(test_data.shape)
```

```
(42000, 785)
(28000, 784)
```

Check the Column name for both train and test data set

In [ ]:

```
def to_check_column(train_data):
  column_name = train_data.columns
  return column_name
```

In [ ]:

```
train_column_name = to_check_column(train_data)
print("Column_name",train_column_name)
```

```
Column_name Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5',
       'pixel6', 'pixel7', 'pixel8',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
```

In [ ]:

```
test_column_name = to_check_column(test_data)
print("Test column_name",test_column_name)
```

```
Test column_name Index(['pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6',
       'pixel7', 'pixel8', 'pixel9',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=784)
```

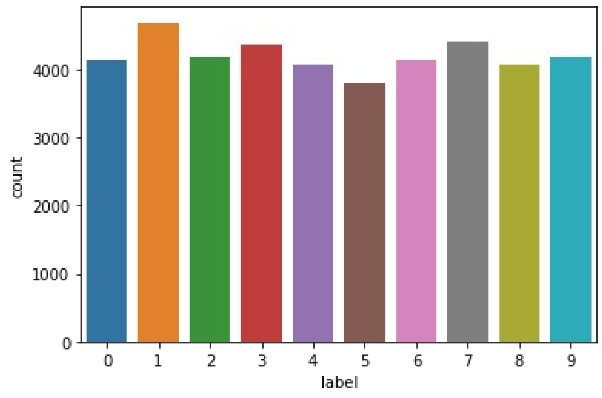
Count Plot

In [ ]:

```
### let's use count plot to know the count of each label
def count_plot(train_data):
  sns.countplot(train_data.label)
```

In [ ]:

```
count_plot(train_data)
```



Data is already clean. There is no duplicates and no missing values. Data is already balanced

Need to find range of pixels

In [ ]:

```
train_data['pixel106'].value_counts()
```

Out[ ]:

```
0      41624
255       19
254       15
253        9
191        8
       ...
11         1
176        1
46         1
198        1
193        1
Name: pixel106, Length: 183, dtype: int64
```

In [ ]:

```
train_data['pixel10'].value_counts()
```

Out[ ]:

```
0    42000
Name: pixel10, dtype: int64
```

As per observation,pixel intensities are currently between the range 0 and 255.we proceed to normalize the features between 0 and 1. Converted the numeric class vector to binary one hot encoding

In [ ]:

```
# Feature Normalization
y = train_data['label']
y.value_counts()
```

Out[ ]:

```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```

In [ ]:

```
print(type(y))
```

```
<class 'pandas.core.series.Series'>
```

In [ ]:

```
y_train = to_categorical(y, num_classes = 10)
```

In [ ]:

```
X = train_data.drop(labels=['label'],axis=1)
```

In [ ]:

```
X.head()
```

Out[ ]:

	pixel0	**pixel1**	**pixel2**	**pixel3**	**pixel4**	**pixel5**	**pixel6**	**pixel7**	**pixel8**	**pixel9**	**...**	**pixel774**	**pixel775**	**pixel776**	**pixel777**	**pixel778**
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 784 columns

Normalize the pixels values

In [ ]:

```
def normalize(data):
  data = data/255.0
  return data
```

In [ ]:

```
X = normalize(X)
print(X.shape)
```

```
(42000, 784)
```

In [ ]:

```
X['pixel106'].value_counts()
```

Out[ ]:

```
0.000000    41624
1.000000       19
0.996078       15
0.992157        9
0.749020        8
            ...
0.043137        1
0.690196        1
0.180392        1
0.776471        1
0.756863        1
Name: pixel106, Length: 183, dtype: int64
```

Reshape the image

In [ ]:

```
X_train = X
```

In [ ]:

```
def image_reshape(X_train):
  X_train = X_train.values.reshape(-1,28,28,1)
  return X_train
```

In [ ]:

```
X_train = image_reshape(X_train)
```

In [ ]:

```
test_data = image_reshape(test_data)
```

In [ ]:

```
print(X_train.shape)
print(test_data.shape)
```

```
(42000, 28, 28, 1)
(28000, 28, 28, 1)
```

Image has (28**28) pixels has been stock into Pandas. DataFrame as 1D Vectors of 784 values. We will reshape all data that indicates(height,width,channel).Channel 3 means RGB.

In [ ]:

```
test_data = normalize(test_data)
print(test_data.shape)
```

```
(28000, 28, 28, 1)
```

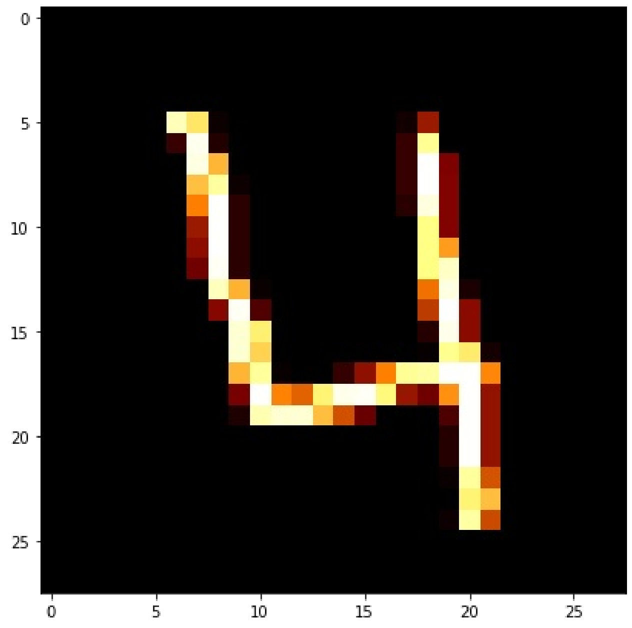
Let show the image

In [ ]:

```
def image_show(X,idx):
  plt.figure(figsize=(7,7))
  grid_data = X.iloc[idx].to_numpy().reshape(28,28)
  plt.imshow(grid_data,interpolation=None,cmap='afmhot')
  plt.show()
```

In [ ]:

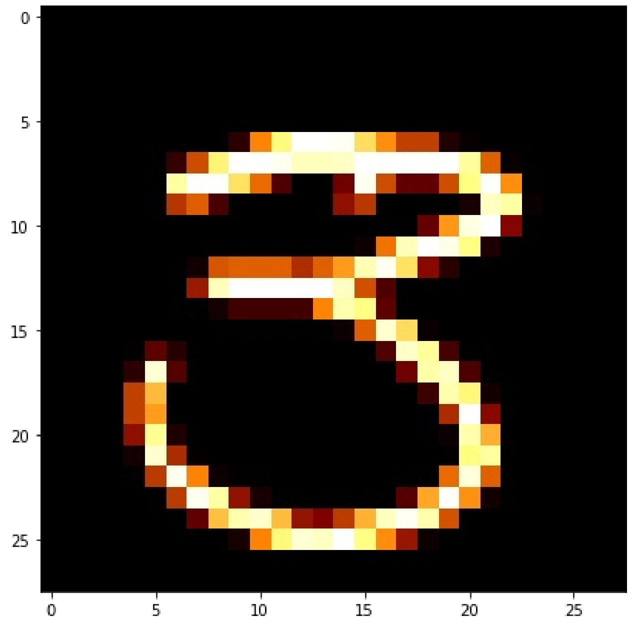
```
a1=image_show(X,3)
print(a1)
```



```
None
```

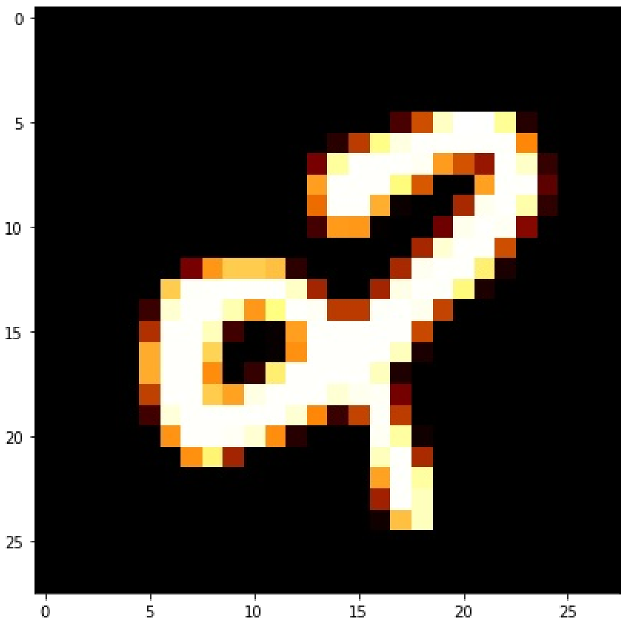
In [ ]:

```
b1 = image_show(X,7)
b1
```



In [ ]:

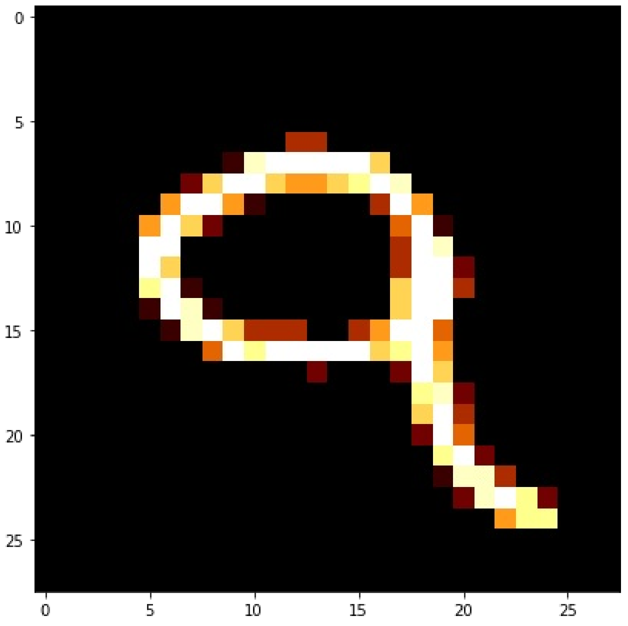
```
c1= image_show(X,104)
print(c1)
```



```
None
```

In [ ]:

```
a2 = image_show(X,779)
print(a2)
```



```
None
```

Splitting train test image dataset

In [ ]:

```
x_train,x_val,y_train,y_val = train_test_split(X_train,y_train,stratify=y_train,test_size=0.1,random_state=42)
print(x_train.shape)
print(x_val.shape)
print(y_train.shape)
print(y_val.shape)
```

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

Architecture of CNN -- [[(CNN2D-relu)*2]->Maxpooling->Dropout]*2]->Flatten->Dense->output

In [ ]:

```
model = Sequential()
model.add(Conv2D(filters=32,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(filters=32,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters=64,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(filters=64,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(10,activation='softmax'))
```

In [ ]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_5 (Conv2D)            (None, 28, 28, 32)        832       
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 28, 28, 32)        25632     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 14, 14, 64)        51264     
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 14, 14, 64)        102464    
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 7, 7, 64)          0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 3136)              0         
_________________________________________________________________
dense_5 (Dense)              (None, 512)               1606144   
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_6 (Dense)              (None, 10)                5130      
=================================================================
Total params: 1,791,466
Trainable params: 1,791,466
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
# Define the optimizer
optimizer = RMSprop(learning_rate=0.001,rho=0.9,momentum=0.1)
```

In [ ]:

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

In [ ]:

```
print(y_train.shape)
```

```
(37800, 10)
```

In [ ]:

```
model.fit(x_train,y_train,epochs=2,batch_size=50,validation_data=(x_val,y_val))
```

```
Epoch 1/2
756/756 [==============================] - 426s 563ms/step - loss: 0.1790 - accuracy: 0.9430 - val_l
oss: 0.0745 - val_accuracy: 0.9793
Epoch 2/2
756/756 [==============================] - 426s 564ms/step - loss: 0.0554 - accuracy: 0.9840 - val_l
oss: 0.0443 - val_accuracy: 0.9881
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f91018a8d50>
```

Saved and load the Model

In [ ]:

```
!mkdir -p saved_model
```

In [ ]:

```
model.save(' ')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow/python/ops/resource_variab
le_ops.py:1817: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_
ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

In [ ]:

```
model.save('saved_model/my_model')
```

In [ ]:

```
# my_model directory
!ls saved_model

# Contains an assets folder, saved_model.pb, and variables folder.
!ls saved_model/my_model
```

```
my_model
assets  saved_model.pb  variables
```

Reload a fresh keras model from saved model

In [ ]:

```
#new_model = tf.keras.models.load_model('saved_model/my_model')

# Check its architecture
#new_model.summary()
```

Let's Do Confusion Matrix

In [ ]:

```
#predict the values from validation data set
y_pred = model.predict(x_val)
#convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_pred,axis=1)
#convert validation observation to one hot vector
y_true = np.argmax(y_val,axis=1)
#calculate the confusion matrix
confusion_matrix(y_true,y_pred_classes)
```

Out[ ]:

```
array([[406,   0,   0,   0,   0,   0,   5,   0,   2,   0],
       [  0, 463,   0,   0,   3,   0,   0,   0,   1,   1],
       [  0,   2, 415,   0,   0,   0,   0,   0,   1,   0],
       [  0,   0,   1, 428,   0,   1,   0,   3,   0,   2],
       [  0,   0,   0,   0, 402,   0,   1,   0,   1,   3],
       [  0,   0,   0,   2,   0, 374,   2,   0,   1,   1],
       [  0,   0,   0,   0,   0,   0, 413,   0,   1,   0],
       [  0,   0,   3,   0,   0,   0,   0, 435,   1,   1],
       [  0,   0,   1,   0,   1,   0,   2,   0, 402,   0],
       [  0,   0,   0,   0,   4,   0,   0,   0,   3, 412]])
```

In [ ]:

```
classification_report(y_true,y_pred_classes)
```

Out[ ]:

```
'              precision    recall  f1-score   support\n\n           0       1.00      0.98      0.9
9       413\n           1       1.00      0.99      0.99       468\n           2       0.99      0.9
9      0.99       418\n           3       1.00      0.98      0.99       435\n           4       0.9
8      0.99      0.98       407\n           5       1.00      0.98      0.99       380\n           6
0.98      1.00      0.99       414\n           7       0.99      0.99      0.99       440\n
8       0.97      0.99      0.98       406\n           9       0.98      0.98      0.98       419\n\
n    accuracy                           0.99      4200\n   macro avg       0.99      0.99      0.99
4200\nweighted avg       0.99      0.99      0.99      4200\n'
```

In [ ]:

```
y_predict = model.predict(test_data)

#select the index with maximum probability
results = np.argmax(y_predict,axis=1)
```

In [ ]:

```
print(results)
```

```
[2 0 9 ... 3 9 2]
```

In [ ]:

```
df_test = pd.read_csv("sample_submission.csv")
df_test.head()
```

Out[ ]:

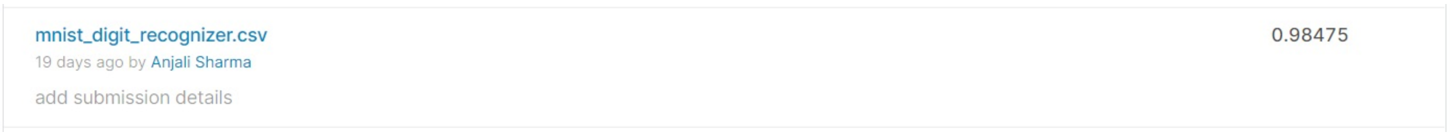
	ImageId	Label
0	1	0
1	2	0
2	3	0
3	4	0
4	5	0

In [ ]:

```
y_predict_lr_test = pd.DataFrame({"ImageId":df_test["ImageId"],"Label":results})
y_predict_lr_test.to_csv('mnist_digit_recognizer.csv', index=False)
y_predict_lr_test.head(5)
```

Out[ ]:

	ImageId	Label
0	1	2
1	2	0
2	3	9
3	4	9
4	5	3


mnist_digit_recognizer.csv
19 days ago by Anjali Sharma
add submission details
0.98475

Got the kaggle score 0.98

Apply image Augmented Technique

In [ ]:

```
def image_augmentation(img,transform):
  'helper function to show the data augmentation'
  img = PIL.Image.open(img)
  fig,ax = plt.subplot(1,2,figsize=(2,4))
  ax[0].set_title(f'original image {img.size}')
  ax[0].image_show(img)
  ax[1].set_title(f'transformed image {img.size}')
  ax[1].tansform(img)
  ax[1].image_show(img)
```

In [ ]:

```
#image_resize = transforms.Resize((23,45))
#image_augmentation(a1,image_resize)
```

In [ ]:

```
print(x_train.shape)
print(y_val.shape)
```

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

Adding BatchNormalization which normalize the hidden layer

In [ ]:

```python
from tensorflow.python.keras.backend import batch_normalization
# Architecture of CNN --  [[(CNN2D-relu)*2]->Maxpooling->Dropout]*2]->Flatten->Dense->output

model = Sequential()
model.add(Conv2D(filters=64,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(BatchNormalization(axis=1))
model.add(Conv2D(filters=64,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(BatchNormalization(axis=1))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(BatchNormalization(axis=1))
model.add(Conv2D(filters=64,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(BatchNormalization(axis=1))
model.add(Conv2D(filters=64,kernel_size=(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
model.add(BatchNormalization(axis=1))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())
model.add(BatchNormalization(axis=1))
model.add(Dense(512,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(10,activation='softmax'))
```

In [ ]:

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

In [ ]:

```python
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_9 (Conv2D)            (None, 28, 28, 64)        1664      
_________________________________________________________________
batch_normalization (BatchNo (None, 28, 28, 64)        112       
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 28, 64)        102464    
_________________________________________________________________
batch_normalization_1 (Batch (None, 28, 28, 64)        112       
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 64)        0         
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 64)        56        
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 64)        102464    
_________________________________________________________________
batch_normalization_3 (Batch (None, 14, 14, 64)        56        
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 64)        102464    
_________________________________________________________________
batch_normalization_4 (Batch (None, 14, 14, 64)        56        
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 64)          0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 3136)              0         
_________________________________________________________________
batch_normalization_5 (Batch (None, 3136)              12544     
_________________________________________________________________
dense_7 (Dense)              (None, 512)               1606144   
_________________________________________________________________
batch_normalization_6 (Batch (None, 512)               2048      
_________________________________________________________________
dense_8 (Dense)              (None, 10)                5130      
=================================================================
Total params: 1,935,314
Trainable params: 1,927,822
Non-trainable params: 7,492
_________________________________________________________________
```

In [ ]:

```
model.fit(x_train,y_train,epochs=10,batch_size=50,validation_data=(x_val,y_val))
```

```
Epoch 1/10
756/756 [==============================] - 969s 1s/step - loss: 0.1237 - accuracy: 0.9620 - val_loss
: 0.0612 - val_accuracy: 0.9840
Epoch 2/10
756/756 [==============================] - 964s 1s/step - loss: 0.0457 - accuracy: 0.9857 - val_loss
: 0.0439 - val_accuracy: 0.9890
Epoch 3/10
756/756 [==============================] - 963s 1s/step - loss: 0.0341 - accuracy: 0.9891 - val_loss
: 0.0507 - val_accuracy: 0.9857
Epoch 4/10
756/756 [==============================] - 960s 1s/step - loss: 0.0284 - accuracy: 0.9908 - val_loss
: 0.0367 - val_accuracy: 0.9900
Epoch 5/10
756/756 [==============================] - 953s 1s/step - loss: 0.0237 - accuracy: 0.9927 - val_loss
: 0.0421 - val_accuracy: 0.9895
Epoch 6/10
756/756 [==============================] - 959s 1s/step - loss: 0.0220 - accuracy: 0.9928 - val_loss
: 0.0343 - val_accuracy: 0.9924
Epoch 7/10
756/756 [==============================] - 969s 1s/step - loss: 0.0209 - accuracy: 0.9930 - val_loss
: 0.0427 - val_accuracy: 0.9900
Epoch 8/10
756/756 [==============================] - 963s 1s/step - loss: 0.0167 - accuracy: 0.9946 - val_loss
: 0.0540 - val_accuracy: 0.9876
Epoch 9/10
756/756 [==============================] - 949s 1s/step - loss: 0.0174 - accuracy: 0.9947 - val_loss
: 0.0415 - val_accuracy: 0.9919
Epoch 10/10
391/756 [==============>...............] - ETA: 7:28 - loss: 0.0088 - accuracy: 0.9972
```

In [ ]:

```
!mkdir -p saved_model_v2
```

In [ ]:

```
model.save('saved_model_v2/my_model_v2')
```

In [ ]:

```
#predict the values from validation set
y_pred = model.predict(x_val)
# convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_pred,axis=1)
#convert validation obserbation to one hot vecors
y_true = np.argmax(y_val,axis=1)
#let's call confusion matrix
confusion_matrix(y_true,y_pred_classes)
```

Out[ ]:

```
array([[412,   0,   0,   0,   0,   0,   1,   0,   0,   0],
       [  0, 467,   1,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0, 416,   0,   0,   0,   0,   2,   0,   0],
       [  0,   0,   0, 433,   0,   1,   0,   0,   0,   1],
       [  0,   1,   0,   0, 402,   0,   0,   0,   0,   4],
       [  0,   0,   0,   5,   0, 366,   1,   0,   3,   5],
       [  0,   0,   0,   0,   0,   0, 412,   0,   2,   0],
       [  0,   0,   4,   1,   0,   0,   0, 435,   0,   0],
       [  3,   2,   0,   0,   0,   0,   2,   0, 399,   0],
       [  0,   1,   0,   0,   4,   0,   0,   1,   3, 410]])
```

In [ ]:

```
y_predict = model.predict(test_data)
#let's take index with maximum probabilty
result = np.argmax(y_predict,axis=1)
print(result)
```

```
[2 0 9 ... 3 9 2]
```

In [ ]:

```
df_test_batch = pd.read_csv("sample_submission.csv")
df_test_batch.head()
```

Out[ ]:

	ImageId	Label
0	1	0
1	2	0
2	3	0
3	4	0
4	5	0

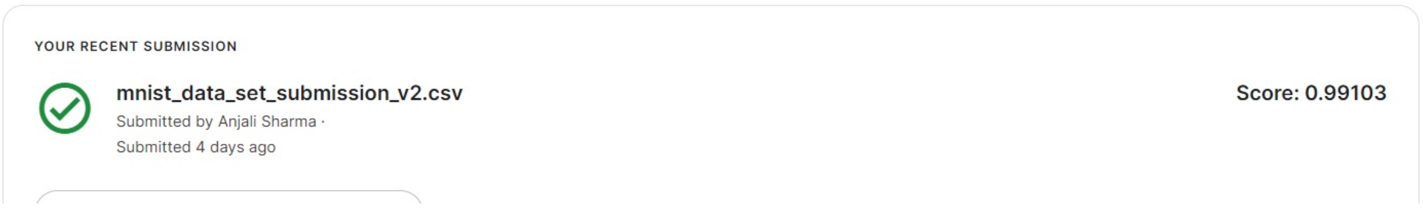
In [ ]:

```
y_predict_cnn_batch_test = pd.DataFrame({"ImageId": df_test_batch["ImageId"],"Label":result})
y_predict_cnn_batch_test.to_csv("mnist_data_set_submission_v2.csv",index=False)
y_predict_cnn_batch_test
```

Out[ ]:

	ImageId	Label
0	1	2
1	2	0
2	3	9
3	4	0
4	5	3
...	...	...
27995	27996	9
27996	27997	7
27997	27998	3
27998	27999	9
27999	28000	2

28000 rows × 2 columns


YOUR RECENT SUBMISSION

mnist_data_set_submission_v2.csv

Submitted by Anjali Sharma ·

Submitted 4 days ago

Score: 0.99103

Yaahh, Kaggle score is increased than previous one

saved the both Mode

In [ ]:

```
!tar -czvf mnist.tar.gz ./saved_model
```

```
./saved_model/
./saved_model/my_model/
./saved_model/my_model/saved_model.pb
./saved_model/my_model/assets/
./saved_model/my_model/variables/
./saved_model/my_model/variables/variables.index
./saved_model/my_model/variables/variables.data-00000-of-00001
```

In [ ]:

```
!tar -czvf mnist_v1.tar.gz ./saved_model_v2
```

```
./saved_model_v2/
./saved_model_v2/my_model_v2/
./saved_model_v2/my_model_v2/saved_model.pb
./saved_model_v2/my_model_v2/assets/
./saved_model_v2/my_model_v2/variables/
./saved_model_v2/my_model_v2/variables/variables.index
./saved_model_v2/my_model_v2/variables/variables.data-00000-of-00001
```

In [ ]:

```
from keras.models import load_model
import tensorflow as tf
```

Load the MNIST Model -1

In [ ]:

```
import tarfile
my_tar = tarfile.open('mnist.tar.gz')
my_tar.extractall()
my_tar.close()
```

Load MNIST MODEL -2

In [ ]:

```
import tarfile
my_tar = tarfile.open('mnist_v1.tar.gz')
my_tar.extractall()
my_tar.close()
```

In [ ]:

```
!ls saved_model/my_model
```

```
assets  saved_model.pb  variables
```

In [ ]:

```
model_v1 = tf.keras.models.load_model('saved_model/my_model')
```

```
WARNING:tensorflow:SavedModel saved prior to TF 2.5 detected when loading Keras model. Please ensure
that you are saving the model with model.save() or tf.keras.models.save_model(), *NOT* tf.saved_mode
l.save(). To confirm, there should be a file named "keras_metadata.pb" in the SavedModel directory.
```

In [ ]:

```
model_v1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        832

 conv2d_1 (Conv2D)           (None, 28, 28, 32)        25632

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 dropout (Dropout)           (None, 14, 14, 32)        0

 conv2d_2 (Conv2D)           (None, 14, 14, 64)        51264

 conv2d_3 (Conv2D)           (None, 14, 14, 64)        102464

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 64)          0

 flatten (Flatten)           (None, 3136)              0

 dense (Dense)               (None, 512)               1606144

 dropout_2 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 1,791,466
Trainable params: 1,791,466
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
!ls saved_model_v2/
```

```
my_model_v2
```

In [ ]:

```
!ls saved_model_v2/my_model_v2/
```

```
assets  saved_model.pb  variables
```

In [ ]:

```
import warnings
warnings.filterwarnings("ignore")
model_v2 = tf.keras.models.load_model('saved_model_v2/my_model_v2')
```

```
WARNING:tensorflow:SavedModel saved prior to TF 2.5 detected when loading Keras model. Please ensure
that you are saving the model with model.save() or tf.keras.models.save_model(), *NOT* tf.saved_mode
l.save(). To confirm, there should be a file named "keras_metadata.pb" in the SavedModel directory.
```

In [ ]:

```
model_v2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_9 (Conv2D)           (None, 28, 28, 64)        1664      
                                                                 
 batch_normalization (BatchN  (None, 28, 28, 64)       112       
 ormalization)                                                   
                                                                 
 conv2d_10 (Conv2D)          (None, 28, 28, 64)        102464    
                                                                 
 batch_normalization_1 (Batc  (None, 28, 28, 64)       112       
 hNormalization)                                                 
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 14, 14, 64)       0         
 2D)                                                             
                                                                 
 batch_normalization_2 (Batc  (None, 14, 14, 64)       56        
 hNormalization)                                                 
                                                                 
 conv2d_11 (Conv2D)          (None, 14, 14, 64)        102464    
                                                                 
 batch_normalization_3 (Batc  (None, 14, 14, 64)       56        
 hNormalization)                                                 
                                                                 
 conv2d_12 (Conv2D)          (None, 14, 14, 64)        102464    
                                                                 
 batch_normalization_4 (Batc  (None, 14, 14, 64)       56        
 hNormalization)                                                 
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 7, 7, 64)         0         
 2D)                                                             
                                                                 
 flatten_3 (Flatten)         (None, 3136)              0         
                                                                 
 batch_normalization_5 (Batc  (None, 3136)             12544     
 hNormalization)                                                 
                                                                 
 dense_7 (Dense)             (None, 512)               1606144   
                                                                 
 batch_normalization_6 (Batc  (None, 512)              2048      
 hNormalization)                                                 
                                                                 
 dense_8 (Dense)             (None, 10)                5130      
                                                                 
=================================================================
Total params: 1,935,314
Trainable params: 1,927,822
Non-trainable params: 7,492
_________________________________________________________________
```

In [ ]:

```
!pip install -q streamlit
```

In [ ]:

```
!pip install -q pyngrok
```

In [ ]:

```
!pip install -q streamlit_ace
```

REFERENCE: https://www.analyticsvidhya.com/blog/2020/12/deploying-machine-learning-models-using-streamlit-an-introductory-guide-to-model-deployment/ (https://www.analyticsvidhya.com/blog/2020/12/deploying-machine-learning-models-using-streamlit-an-introductory-guide-to-model-deployment/)

In [ ]:

```
import pandas as pd
import streamlit as st
```

In [ ]:

```
test_data = pd.read_csv("test.csv")
```

In [ ]:

```
test_data.head()
```

Out[ ]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows x 784 columns

In [ ]:

```
test_data.shape
```

Out[ ]:

```
(28000, 784)
```

In [ ]:

```
test_data['pixel1'].value_counts()
```

Out[ ]:

```
0    28000
Name: pixel1, dtype: int64
```

In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
```

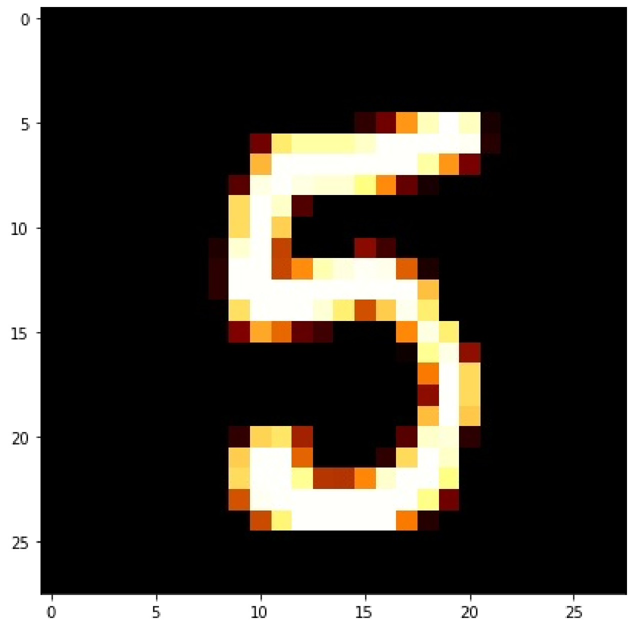
In [ ]:

```
def image_show(X,idx):
  plt.figure(figsize=(7,7))
  grid_data = X.iloc[idx].to_numpy().reshape(28,28)
  plt.imshow(grid_data,interpolation=None,cmap='afmhot')
  plt.show()
```

In [ ]:

```
test_data = test_data/255.0

image_show(test_data,10)
```



In [ ]:

```
def prediction(idx):
  plt.figure(figsize=(7,7))
  image= test_data.iloc[idx].to_numpy().reshape(1,28,28)
  p = model_v1.predict(image)
  return p
```

In [ ]:

```
p = prediction(9)
print("prediction {}".format(np.argmax(p)))
```

```
prediction 3
```

```
<Figure size 504x504 with 0 Axes>
```

Model Deployment of MNIST DATA set using streamlit library

In [ ]:

```
%%writefile app.py
import streamlit as st
import matplotlib.pyplot as plt
from keras.models import load_model
import tensorflow as tf
import numpy as np
import pandas as pd
model_v1 = tf.keras.models.load_model('saved_model/my_model')
test_data = pd.read_csv("test.csv")
def prediction(idx):
  idx = int(idx)
  plt.figure(figsize=(7,7))
  image= test_data.iloc[idx].to_numpy().reshape(1,28,28)
  p = model_v1.predict(image)
  return p
# this is main function which defines the web pages
def main():
   #front end element on web pages
   html_temp = """
   <div style ="background-color:yellow;padding:13px">
   <h1 style ="color:black;text-align:center;">Digit Recogintion</h1>
   </div>
   """
   # display the front end aspect
   st.markdown(html_temp, unsafe_allow_html = True)
   idx = st.number_input("Enter the MNIST image index")
   result= " "
   if st.button("Predict"):
     result = prediction(idx)
     st.success("MNIST digit Recogintion based on entered image's index =  {}".format(np.argmax(result)))
if __name__=='__main__':
    main()
```

```
Overwriting app.py
```

In [ ]:

```
!streamlit run app.py &>/dev/null&
```

In [ ]:

```
!ngrok authtoken 24QK9tsxhgVhAIssPQhPrFNQSWg_3j6PtXWiKHCUnCtXn6BbG
```

```
Authtoken saved to configuration file: /root/.ngrok2/ngrok.yml
```

In [ ]:

```
from pyngrok import ngrok
public_url = ngrok.connect('8501')
public_url
```

```
INFO:pyngrok.ngrok:Opening tunnel named: http-8501-6a1477c9-ea55-4714-b1d7-68b961dd0ea7
2022-09-12 19:03:11.716 INFO    pyngrok.ngrok: Opening tunnel named: http-8501-6a1477c9-ea55-4714-b1
d7-68b961dd0ea7
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:11+0000 lvl=info msg="no configuration paths supplied"
2022-09-12 19:03:11.860 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:11+0000 lvl=info msg="no c
onfiguration paths supplied"
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:11+0000 lvl=info msg="using configuration at default c
onfig path" path=/root/.ngrok2/ngrok.yml
2022-09-12 19:03:11.868 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:11+0000 lvl=info msg="usin
g configuration at default config path" path=/root/.ngrok2/ngrok.yml
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:11+0000 lvl=info msg="open config file" path=/root/.ng
rok2/ngrok.yml err=nil
2022-09-12 19:03:11.876 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:11+0000 lvl=info msg="open
config file" path=/root/.ngrok2/ngrok.yml err=nil
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:11+0000 lvl=info msg="starting web service" obj=web ad
dr=127.0.0.1:4040
2022-09-12 19:03:11.884 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:11+0000 lvl=info msg="star
ting web service" obj=web addr=127.0.0.1:4040
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg="tunnel session started" obj=tunn
els.session
2022-09-12 19:03:12.031 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg="tunn
el session started" obj=tunnels.session
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg="client session established" obj=
csess id=ba6443affdb0
2022-09-12 19:03:12.041 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg="clie
nt session established" obj=csess id=ba6443affdb0
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg=start pg=/api/tunnels id=25da1acf
266fdeae
2022-09-12 19:03:12.069 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg=start
pg=/api/tunnels id=25da1acf266fdeae
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg=end pg=/api/tunnels id=25da1acf26
6fdeae status=200 dur=460.431µs
2022-09-12 19:03:12.082 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg=end p
g=/api/tunnels id=25da1acf266fdeae status=200 dur=460.431µs
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg=start pg=/api/tunnels id=8c31481f
2f4089b1
2022-09-12 19:03:12.097 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg=start
pg=/api/tunnels id=8c31481f2f4089b1
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg=end pg=/api/tunnels id=8c31481f2f
4089b1 status=200 dur=132.412µs
2022-09-12 19:03:12.103 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg=end p
g=/api/tunnels id=8c31481f2f4089b1 status=200 dur=132.412µs
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg=start pg=/api/tunnels id=c4977021
6db854d3
2022-09-12 19:03:12.109 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg=start
pg=/api/tunnels id=c49770216db854d3
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg="started tunnel" obj=tunnels name
="http-8501-6a1477c9-ea55-4714-b1d7-68b961dd0ea7 (http)" addr=http://localhost:8501 url=http://58b0-
34-138-217-149.ngrok.io
2022-09-12 19:03:12.160 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg="star
ted tunnel" obj=tunnels name="http-8501-6a1477c9-ea55-4714-b1d7-68b961dd0ea7 (http)" addr=http://loc
alhost:8501 url=http://58b0-34-138-217-149.ngrok.io
```

```
Out[ ]:
```

```
<NgrokTunnel: "http://58b0-34-138-217-149.ngrok.io" -> "http://localhost:8501">
```

```
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg="started tunnel" obj=tunnels name
=http-8501-6a1477c9-ea55-4714-b1d7-68b961dd0ea7 addr=http://localhost:8501 url=https://58b0-34-138-2
17-149.ngrok.io
2022-09-12 19:03:12.167 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg="star
ted tunnel" obj=tunnels name=http-8501-6a1477c9-ea55-4714-b1d7-68b961dd0ea7 addr=http://localhost:85
01 url=https://58b0-34-138-217-149.ngrok.io
INFO:pyngrok.process.ngrok:t=2022-09-12T19:03:12+0000 lvl=info msg=end pg=/api/tunnels id=c49770216d
b854d3 status=201 dur=90.402484ms
2022-09-12 19:03:12.176 INFO    pyngrok.process.ngrok: t=2022-09-12T19:03:12+0000 lvl=info msg=end p
g=/api/tunnels id=c49770216db854d3 status=201 dur=90.402484ms
```