Handwritten equation solver using CNN

LETS GROW MORE - Virtual Internship 2023

Name : Shiva Dagdu Mehenge

Data Science Intern

#LGMVIP

In [1]:

```
import numpy as np
import cv2
from PIL import Image
from matplotlib import pyplot as plt
%matplotlib inline
import os
from os import listdir
from os.path import isfile, join
import pandas as pd
import pickle
```

In [2]:

```
def load_images_from_folder(folder):
    train_data=[]
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder,filename),cv2.IMREAD_GRAYSCALE)
        img=~img
        if img is not None:
            ret,thresh=cv2.threshold(img,127,255,cv2.THRESH_BINARY)

            ctrs,ret=cv2.findContours(thresh,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
            cnt=sorted(ctrs, key=lambda ctr: cv2.boundingRect(ctr)[0])
            w=int(28)
            h=int(28)
            maxi=0
            for c in cnt:
                x,y,w,h=cv2.boundingRect(c)
                maxi=max(w*h,maxi)
                if maxi==w*h:
                    x_max=x
                    y_max=y
                    w_max=w
                    h_max=h
            im_crop= thresh[y_max:y_max+h_max+10, x_max:x_max+w_max+10]
            im_resize = cv2.resize(im_crop,(28,28))
            im_resize=np.reshape(im_resize,(784,1))
            train_data.append(im_resize)
    return train_data
```

In [10]:

```
data=[]
data=load_images_from_folder('extracted_images\-')
len(data)
for i in range(0,len(data)):
    data[i]=np.append(data[i],['10'])

print(len(data))
```

```
33997
```

In [11]:
```
data0=load_images_from_folder('extracted_images\+')
for i in range(0,len(data0)):
    data0[i]=np.append(data0[i],['0'])
data=np.concatenate((data,data0))
print(len(data))
```

```
59109
```

In [12]:
```
data1=load_images_from_folder('extracted_images\!')
for i in range(0,len(data1)):
    data1[i]=np.append(data1[i],['1'])
data=np.concatenate((data,data1))
print(len(data))
```

```
60409
```

In [13]:
```
data2=load_images_from_folder('extracted_images\(')
for i in range(0,len(data2)):
    data2[i]=np.append(data2[i],['2'])
data=np.concatenate((data,data2))
print(len(data))
```

```
74703
```

In [14]:
```
data3=load_images_from_folder('extracted_images\)')
for i in range(0,len(data3)):
    data3[i]=np.append(data3[i],['3'])
data=np.concatenate((data,data3))
print(len(data))
```

```
89058
```

In [15]:
```
data4=load_images_from_folder('extracted_images\,')
for i in range(0,len(data4)):
    data4[i]=np.append(data4[i],['4'])
data=np.concatenate((data,data4))
print(len(data))
```

```
90964
```

In [16]:
```
df=pd.DataFrame(data,index=None)
df.to_csv('train_handwritten.csv',index=False)
```

In [17]:
```
data = pd.read_csv('train_handwritten.csv',index_col=False)
labels = data[['78']]
```

In [18]:
```
data.drop(data.columns[[78]],axis=1,inplace=True)
data.head()
```

Out[18]:

	0	1	2	3	4	5	6	7	8	9	...	775	776	777	778	779	780	781	782	783	784
0	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
1	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
2	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
3	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
4	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10

5 rows × 784 columns

In [19]:

```python
np.random.seed(1212)
import keras
from keras.models import Model
from keras.layers import *
from keras import optimizers
from keras.layers import Input, Dense
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras import backend as K
K.image_data_format()
```

Out[19]:

```
'channels_last'
```

In [20]:

```python
labels=np.array(labels)
```

In [21]:

```python
from keras.utils import to_categorical

cat=to_categorical(labels,num_classes=0)

```

In [22]:

```python
print(cat[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. 1. 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.]
```

In [23]:
```
1 data.head()
```

Out[23]:

	0	1	2	3	4	5	6	7	8	9	...	775	776	777	778	779	780	781	782	783	784
0	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
1	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
2	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
3	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10
4	255	255	255	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	10

5 rows × 784 columns

In [24]:
```
1 data.shape
```

Out[24]:
```
(90964, 784)
```

In [25]:
```
1 temp=data.to_numpy()
```

In [26]:
```
1 X_train = temp.reshape(temp.shape[0], 28, 28, 1)
```

In [27]:
```
1 temp.shape[0]
```

Out[27]:
```
90964
```

In [28]:
```
1 X_train.shape
```

Out[28]:
```
(90964, 28, 28, 1)
```

In [29]:
```
1 l=[]
2 for i in range(14326):
3     l.append(np.array(data[i:i+1]).reshape(1,28,28))
```

In [30]:
```
np.random.seed(7)
```

In [31]:
```
len(l[0])
```

Out[31]:
```
1
```

In [32]:
```
X_train.shape
```

Out[32]:
```
(90964, 28, 28, 1)
```

In [33]:
```
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape=(28, 28,1), activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(15, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(13, activation='softmax'))
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [34]:
```
1 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

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

 conv2d_1 (Conv2D)           (None, 12, 12, 15)        4335

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 15)         0
 2D)

 dropout (Dropout)           (None, 6, 6, 15)          0

 flatten (Flatten)           (None, 540)               0

 dense (Dense)               (None, 128)               69248

 dense_1 (Dense)             (None, 50)                6450

 dense_2 (Dense)             (None, 13)                663

=================================================================
Total params: 81,016
Trainable params: 81,016
Non-trainable params: 0
_________________________________________________________________
```

In [35]:

```
import pickle
pickle.dump(model, open('model.pkl', 'wb'))
model.save_weights("model_final.h5")
```

```
Keras weights file (<HDF5 file "variables.h5" (mode r+)>) saving:
...layers\conv2d
......vars
.........0
.........1
...layers\conv2d_1
......vars
.........0
.........1
...layers\dense
......vars
.........0
.........1
...layers\dense_1
......vars
.........0
.........1
...layers\dense_2
......vars
.........0
.........1
...layers\dropout
......vars
...layers\flatten
......vars
...layers\max_pooling2d
......vars
...layers\max_pooling2d_1
......vars
...optimizer
......vars
.........0
...vars
Keras model archive saving:
File Name                                             Modified             Size
config.json                                     2023-03-30 18:02:32         3646
metadata.json                                   2023-03-30 18:02:32           64
variables.h5                                    2023-03-30 18:02:32       351008
```

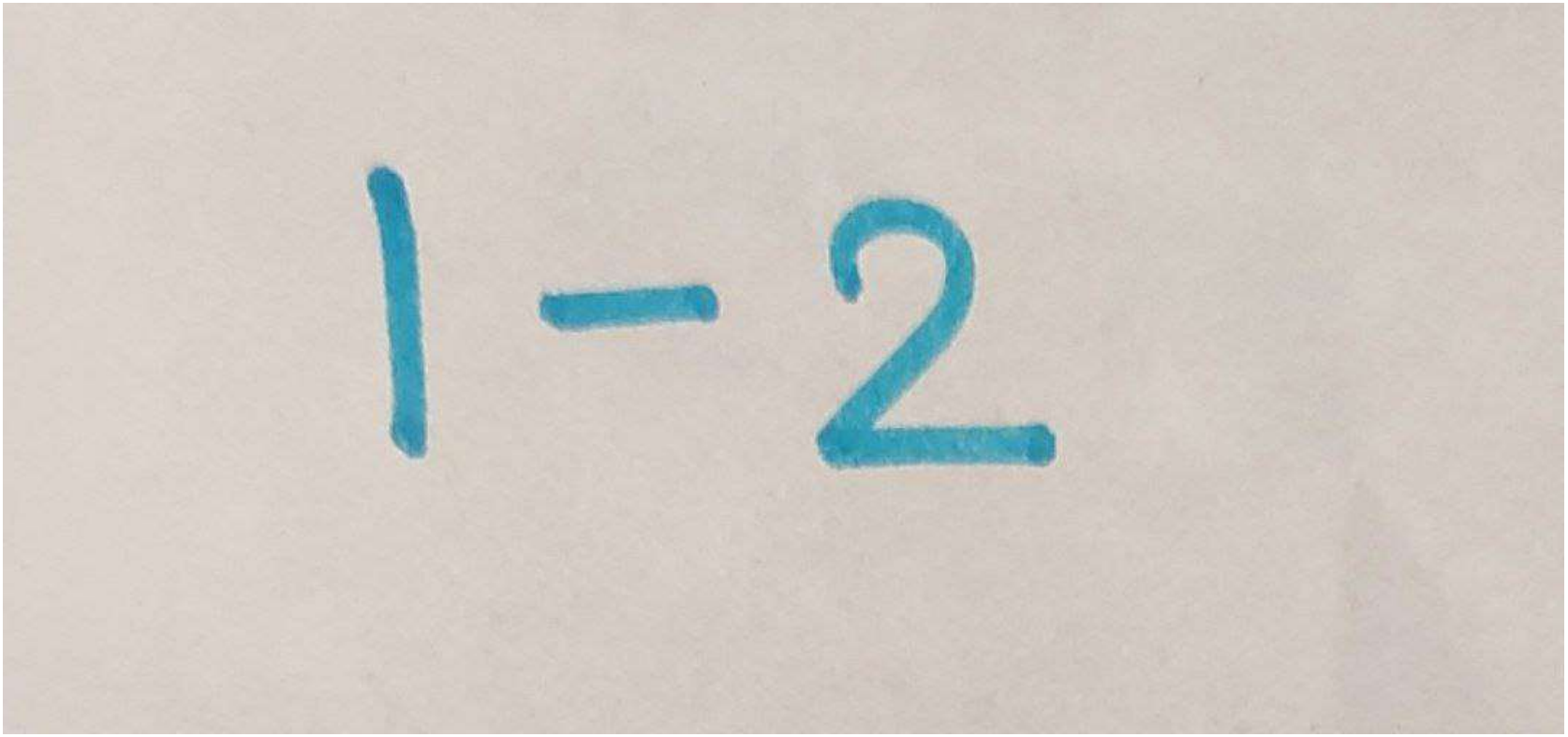
In [36]:

```
import cv2
import numpy as np
img = cv2.imread('Test 7.jpg',cv2.IMREAD_GRAYSCALE)
```

In [37]:

```
from IPython.display import Image
Image(filename='Test 7.jpg')
```

Out[37]:



In [38]:

```
if img is not None:
    img=~img
    ret,thresh=cv2.threshold(img,127,255,cv2.THRESH_BINARY)
    ctrs,ret=cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    cnt=sorted(ctrs, key=lambda ctr: cv2.boundingRect(ctr)[0])
    w=int(28)
    h=int(28)
    train_data=[]
    print(len(cnt))
    rects=[]
    for c in cnt :
        x,y,w,h= cv2.boundingRect(c)
        rect=[x,y,w,h]
        rects.append(rect)
    print(rects)
    bool_rect=[]
    for r in rects:
        l=[]
        for rec in rects:
            flag=0
            if rec!=r:
                if r[0]<(rec[0]+rec[2]+10) and rec[0]<(r[0]+r[2]+10) and r[1]<(rec[1]+rec[3]+10) and rec|
                    flag=1
                l.append(flag)
            if rec==r:
                l.append(0)
        bool_rect.append(l)
    print(bool_rect)
    dump_rect=[]
    for i in range(0,len(cnt)):
        for j in range(0,len(cnt)):
            if bool_rect[i][j]==1:
                area1=rects[i][2]*rects[i][3]
                area2=rects[j][2]*rects[j][3]
                if(area1==min(area1,area2)):
                    dump_rect.append(rects[i])
    print(len(dump_rect))
    final_rect=[i for i in rects if i not in dump_rect]
    print(final_rect)
    for r in final_rect:
        x=r[0]
        y=r[1]
        w=r[2]
```

```
        h=r[3]
        im_crop =thresh[y:y+h+10,x:x+w+10]
        im_resize = cv2.resize(im_crop,(28,28))
        im_resize=np.reshape(im_resize,(28,28,1))
        train_data.append(im_resize)
```

```
184
[[253, 114, 33, 94], [260, 137, 3, 3], [268, 223, 1, 1], [270, 282, 21, 27], [270, 248, 2, 1], [271, 300,
2, 1], [271, 251, 2, 4], [271, 232, 1, 1], [271, 178, 8, 8], [273, 211, 4, 12], [273, 157, 10, 9], [274,
231, 2, 1], [274, 224, 2, 1], [274, 129, 3, 6], [275, 235, 1, 2], [275, 224, 11, 29], [275, 150, 3, 3],
[276, 166, 3, 3], [277, 272, 2, 2], [277, 267, 1, 3], [279, 232, 6, 6], [279, 184, 3, 3], [279, 182, 3,
3], [280, 279, 1, 1], [280, 274, 3, 2], [280, 216, 1, 2], [280, 208, 5, 2], [280, 190, 3, 3], [280, 187,
3, 3], [281, 252, 1, 1], [281, 219, 3, 4], [283, 302, 3, 3], [283, 271, 1, 1], [284, 248, 6, 10], [284, 2
25, 2, 3], [285, 215, 1, 1], [288, 273, 1, 2], [288, 261, 2, 3], [373, 200, 58, 21], [387, 202, 3, 3], [3
91, 204, 12, 10], [392, 216, 3, 4], [397, 219, 2, 1], [397, 203, 3, 3], [401, 218, 2, 1], [406, 218, 2,
2], [406, 205, 3, 4], [407, 207, 7, 7], [407, 204, 7, 3], [409, 216, 6, 2], [416, 217, 2, 1], [419, 215,
7, 3], [421, 215, 1, 1], [422, 204, 6, 1], [426, 209, 3, 5], [431, 206, 1, 1], [432, 211, 1, 1], [434, 21
4, 5, 2], [444, 202, 5, 6], [445, 199, 1, 1], [449, 198, 5, 4], [455, 208, 1, 1], [455, 201, 1, 3], [458,
198, 1, 1], [464, 200, 1, 1], [471, 195, 22, 21], [486, 205, 5, 7], [488, 208, 1, 1], [564, 290, 49, 26],
[567, 294, 4, 3], [573, 287, 2, 2], [573, 170, 1, 4], [573, 168, 1, 1], [575, 305, 3, 4], [575, 177, 18,
26], [575, 168, 1, 2], [575, 163, 1, 1], [578, 154, 2, 2], [579, 160, 9, 12], [580, 181, 1, 1], [581, 28
4, 3, 8], [581, 280, 1, 1], [583, 275, 4, 3], [583, 173, 2, 6], [583, 155, 1, 2], [583, 148, 3, 2], [585,
152, 3, 7], [585, 145, 2, 2], [587, 280, 3, 1], [588, 153, 4, 15], [588, 145, 1, 1], [589, 313, 2, 5], [5
90, 273, 1, 1], [590, 150, 2, 1], [591, 287, 2, 2], [591, 144, 1, 1], [591, 142, 1, 1], [592, 146, 2, 2],
[593, 149, 4, 5], [594, 265, 3, 3], [595, 290, 1, 1], [595, 141, 1, 1], [597, 301, 7, 9], [597, 138, 5,
```

In [39]:

```
equation=''
for i in range(len(train_data)):
    train_data[i]=np.array(train_data[i])
    train_data[i]=train_data[i].reshape(1,28,28,1)
    result=np.argmax(model.predict(train_data[i]), axis=-1)
    if(result[0]==10):
        equation = equation +'-'
    if(result[0]==11):
        equation = equation +'+'
    if(result[0]==12):
        equation = equation +'*'
    if(result[0]==0):
        equation = equation +'0'
    if(result[0]==1):
        equation = equation +'1'
    if(result[0]==2):
        equation = equation +'2'
    if(result[0]==3):
        equation = equation +'3'
    if(result[0]==4):
        equation = equation +'4'
    if(result[0]==5):
        equation = equation +'5'
    if(result[0]==6):
        equation = equation +'6'
    if(result[0]==7):
        equation = equation +'7'
    if(result[0]==8):
        equation = equation +'8'
    if(result[0]==9):
        equation = equation +'9'

print(equation)
```

```
1/1 [==============================] - 0s 235ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 9ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 3ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 5ms/step
1/1 [==============================] - 0s 16ms/step
66617676667671666
```

In [40]:
```
1 eval(equation)
```

Out[40]: 66617676667671666

Thank You !

In [ ]:
```
1
```