In [272]:

```
from keras.models import load_model       # importing the required libraries
import cv2
import scipy.ndimage
import tensorflow as tf
import math
from PIL import Image
import sys
import os
import csv
import numpy as np
import matplotlib.pyplot as plt
import imutils
```

In [273]:

```
model = load_model('gg.h5')          #loading the pretrained weights
model.summary()                    #summary of model
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 32)        25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               803072
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 13)                3341
=================================================================
Total params: 888,301
Trainable params: 888,301
Non-trainable params: 0
_________________________________________________________________
```

In [265]:

```
def getBestShift(img):
        cy,cx = scipy.ndimage.measurements.center_of_mass(img) #finds centre_of_mass of
img
        rows,cols = img.shape
        shiftx = np.round(cols/2.0-cx).astype(int)
        shifty = np.round(rows/2.0-cy).astype(int)
        return shiftx,shifty
def shift(img,sx,sy):                  #shifts the image in given direction
    rows,cols = img.shape
    M = np.float32([[1,0,sx],[0,1,sy]])
    shifted = cv2.warpAffine(img,M,(cols,rows))
    return shifted
```

In [266]:

```
def manipulate_image(img): #function converting image in a format that is of mnist datase
t images on which model is trained to give accurate results
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)    # converting image into grayscale
    gray = cv2.resize(img, (28, 28),interpolation=cv2.INTER_AREA)
    (thresh, gray) = cv2.threshold(255-gray, 160, 255, cv2.THRESH_BINARY | cv2.THRESH_OTS
U) # thresolding is done to get fine clear image
    while np.sum(gray[0]) == 0:         #
        gray = gray[1:]                 #NEXT 4 while loop deletes all the extra pixels f
rom the sides
    while np.sum(gray[:,0]) == 0:       #
        gray = np.delete(gray,0,1)
    while np.sum(gray[-1]) == 0:
        gray = gray[:-1]
    while np.sum(gray[:,-1]) == 0:
        gray = np.delete(gray,-1,1)
    rows,cols = gray.shape
    if rows > cols:                    #if-else resizes the outer images so that both dim
ensions become equal to 20x20
        factor = 20.0/rows
        rows = 20
        cols = int(round(cols*factor))
        gray = cv2.resize(gray, (cols,rows))
    else:
        factor = 20.0/cols
        cols = 20
        rows = int(round(rows*factor))
        gray = cv2.resize(gray, (cols, rows))
    colsPadding = (int(math.ceil((28-cols)/2.0)),int(math.floor((28-cols)/2.0)))
    rowsPadding = (int(math.ceil((28-rows)/2.0)),int(math.floor((28-rows)/2.0)))
    gray = np.lib.pad(gray,(rowsPadding,colsPadding),'constant')                # paddin
g the image to convert it into required shape
    shiftx,shifty = getBestShift(gray)                                          #which
is 28x28
    shifted = shift(gray,shiftx,shifty)        #getBestShift finds centre_of_mass and sh
ift function shifts the image in given direction
    gray = shifted
    gray = gray/255                    # normalising the image to be send for prediction
s
    return gray
```

In [267]:

```
def return_output(img1,ratio): #function returning output as numbers that can be further
processed
    gray = manipulate_image(img1)       #changing the image in a format in which mnist i
s trained
    gray = gray.reshape((1,28,28,1))
    predicted_11 = model.predict(gray)
    predicted_1 = np.argmax(predicted_11, axis = 1)   # taking the max of all the predic
tions
    if ratio<0.3:                    # for 1 using w/h ratio  of contours
        predicted_1[0] = 1
    if ratio>1.5:                    # for '-' also using w/h ratio of contours
        predicted_1[0] = 11
    return predicted_1[0]
```

In [268]:

```
def precedence(op):
    if op == '+' or op == '-':
        return 1
    if op == '*' or op == '/':
        return 2
    return 0
def applyOp(a, b, op):
    if op == '+': return a + b
    if op == '-': return a - b
    if op == '*': return a * b
    if op == '/': return a // b
def evaluate(tokens):
    values = []
```

```
    ops = []
    i = 0
    while i < len(tokens):
        if tokens[i] == ' ':
            i += 1
            continue
        elif tokens[i] == '(':
            ops.append(tokens[i])
        elif tokens[i].isdigit():
            val = 0
            while (i < len(tokens) and
                tokens[i].isdigit()):
                val = (val * 10) + int(tokens[i])
                i += 1
            values.append(val)
        elif tokens[i] == ')':

            while len(ops) != 0 and ops[-1] != '(':
                val2 = values.pop()
                val1 = values.pop()
                op = ops.pop()
                values.append(applyOp(val1, val2, op))
            ops.pop()
        else:
            while (len(ops) != 0 and
                precedence(ops[-1]) >= precedence(tokens[i])):
                val2 = values.pop()
                val1 = values.pop()
                op = ops.pop()
                values.append(applyOp(val1, val2, op))
            ops.append(tokens[i])
        i += 1
    while len(ops) != 0:
        val2 = values.pop()
        val1 = values.pop()
        op = ops.pop()
        values.append(applyOp(val1, val2, op))
    return values[-1]
```

In [270]:

```
def evaluate_string(img1):                          # return string
    img = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)        #converting images to grayscale
    ret, img = cv2.threshold(255-img, 160, 255, 0)
    r = cv2.selectROI(img)                                # r selects region of given im
ages and passes it for evaluation
    img = img[int(r[1]):int(r[1]+r[3]), int(r[0]):int(r[0]+r[2])]         # cropping the
given image as per roi
    img1 = img1[int(r[1]):int(r[1]+r[3]), int(r[0]):int(r[0]+r[2])]
    _,contours, hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIM
PLE) #detecting contours of all numbers and operators
    a = []
    mm = 0
    mn = 10000000
    for cnt in contours:        #finding min and max area of all contours
        xx = cv2.contourArea(cnt)
        mm = max(mm,xx)
        mn = min(mn,xx)
#     print(mm)
#     print(mn)
#     print(0.05*mm)
    for cnt in contours:
        x, y, w, h = cv2.boundingRect(cnt)
        M = cv2.moments(cnt)              # finds moments of the all contours
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        imCrop = img1[int(y):int(y+h), int(x):int(x+w)]
        img = cv2.cvtColor(imCrop, cv2.COLOR_BGR2GRAY)
#         plt.imshow(img,cmap="gray")
#         plt.axis("off")
#         plt.show()
```

```
        xx = cv2.contourArea(cnt)
        print(xx)
        if xx < mn*1.1 and w/h>0.8 and w/h<1.2:   #detecting '.' directly using contour
area and w/h ratio
            cla = 12
            a.append([cla, cx])
            print(cla)
            continue
        else:                                     #using trained model for rest of numbe
rs and operator detection
            kernel = np.ones((2,2),np.uint8)
            imCrop = cv2.erode(255-imCrop,kernel,iterations = 2)
            imCrop = 255-imCrop
            plt.imshow(imCrop,cmap="gray")
            plt.axis("off")
            plt.show()
            ratio = w/h
            cla = return_output(imCrop,ratio)
            print(cla)
            a.append([cla, cx])
    sub_li = a
    kk = len(sub_li)
    for i in range(0, kk):
        for j in range(0, kk-i-1):
            if (sub_li[j][1] > sub_li[j + 1][1]):
                tempo = sub_li[j]
                sub_li[j] = sub_li[j + 1]
                sub_li[j + 1] = tempo
    s = ""                                        # empty string in which the required det
ections are to be filled
    for i in range(len(sub_li)):
        if sub_li[i][0] == 10:
            s = s+"+ "
        elif sub_li[i][0] == 11:
            s = s+"- "
        elif sub_li[i][0] == 12:
            s = s+"* "
        elif (i+1)<len(sub_li) and (sub_li[i+1][0]==10 or sub_li[i+1][0]==11 or sub_li[
i+1][0]==12):
            s = s+str(sub_li[i][0])+" "
        else:
            s = s+str(sub_li[i][0])
    return s
```

In [275]:

```
img1 = cv2.imread('ssa.png')    # input the required image for evaluation
ss = evaluate_string(img1)      # final string detected
print(ss)
ans = evaluate(ss)              # evalute function evaluates the given string to give req
uired output
print(ans)                       # final answer
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-275-c7085c1e2f03> in <module>
      1 img1 = cv2.imread('ssa.png')    # input the required image for evaluation
----> 2 ss = evaluate_string(img1)      # final string detected
      3 print(ss)
      4 ans = evaluate(ss)              # evalute function evaluates the given string to
give required output
      5 print(ans)                       # final answer

<ipython-input-270-564892b6ff99> in evaluate_string(img1)
      1 def evaluate_string(img1):
----> 2     img = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
      3     ret, img = cv2.threshold(255-img, 160, 255, 0)
      4     r = cv2.selectROI(img)
      5     img = img[int(r[1]):int(r[1]+r[3]), int(r[0]):int(r[0]+r[2])]

error: OpenCV(3.4.2) c:\miniconda3\conda-bld\opencv-suite_1534379934306\work\modules\imgp
roc\src\color.hpp:253: error: (-215:Assertion failed) VScn::contains(scn) && VDcn::contai
```

```
ns(dcn) && VDepth::contains(depth) in function 'cv::CvtHelper<struct cv::Set<3,4,-1>,stru
ct cv::Set<1,-1,-1>,struct cv::Set<0,2,5>,2>::CvtHelper'
```

In [ ]:

```
```