# test_images

August 22, 2019

[1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os

from sklearn.cluster import KMeans
```

## 1 Script below binarizes images and demonstrates with a batch of test images. The before and after are displayed sequentially.

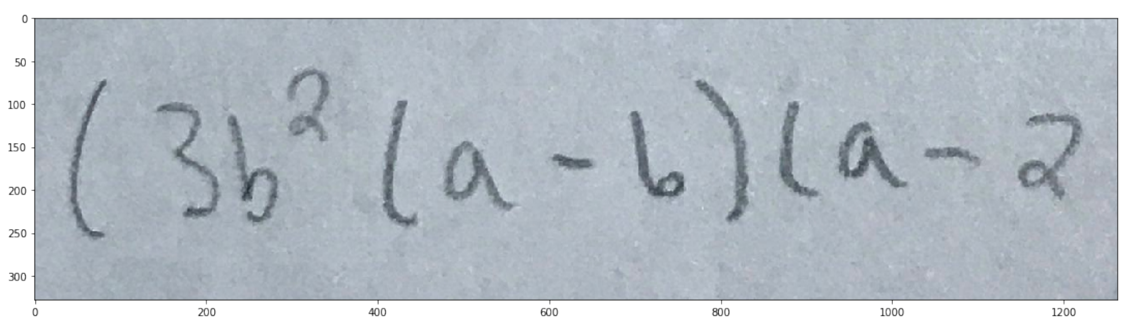
First test performed on 08-21-2019 Last test performed on 08-21-2019

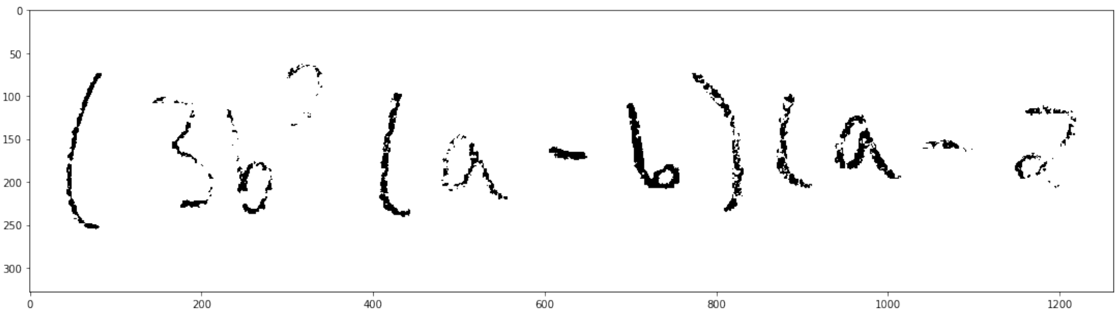
Once the script is done processing, save as .pdf so that it can be compared to previous scripts and attempts at the problem.

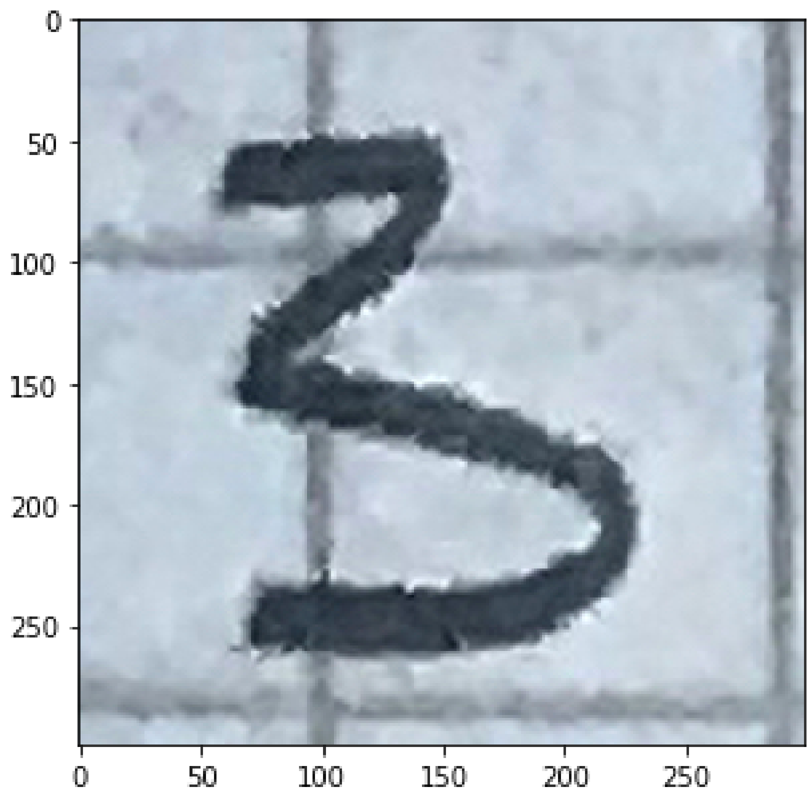
[13]:
```
test_dir = 'Test_Images/'

for filename in os.listdir(test_dir):

    # read in image
    img = cv2.imread(test_dir + filename)
    #img = cv2.resize(img, (600,150))

    plt.figure(figsize=(20,5))
    plt.imshow(img);

    # convert to grayscale
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # dilate the image in order to get ride of the math
    dilated_img = cv2.dilate(gray_img, np.ones((10,10), np.uint8))

    # show dilated image
    # plt.figure(figsize=(20,10))
    # plt.imshow(dilated_img,cmap='gray');
```

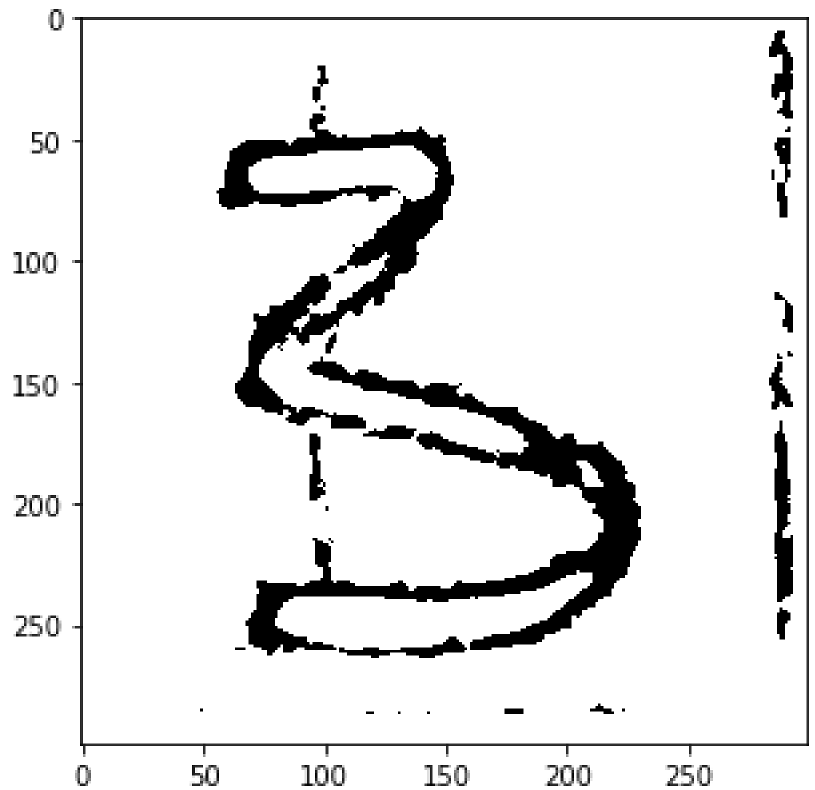
```python
    ### Step 2: Blur ###
    # suppress anything else with a blur function

    blur_img = cv2.medianBlur(dilated_img, 21)

    ### Step 3: Calculate the difference between the original and background␣
↪just made ###
    # identical bits will be black (close to zero difference), text will be␣
↪white (large difference)

    diff_img = 255 - cv2.absdiff(gray_img, blur_img)

    ### Step 4: Apply Simple Threshold ###
    thresh = 180
    maxValue = 255
    ret, thresh2 = cv2.threshold(diff_img, thresh, maxValue, cv2.THRESH_BINARY)

    ### Other Option - Step 4: Apply Otsu Binarization ###
    thresh = 0
    maxValue = 255
    ret, thresh3 = cv2.threshold(thresh2, thresh, maxValue, cv2.
↪THRESH_BINARY+cv2.THRESH_OTSU)


    plt.figure(figsize=(20,5))
    plt.imshow(thresh3,cmap='Greys_r');
```

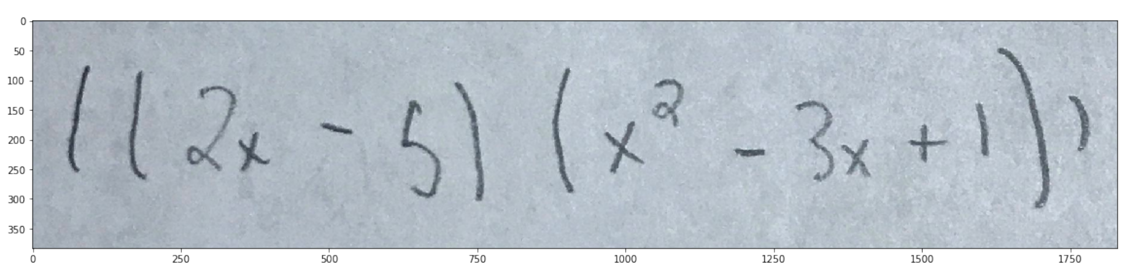
```
/Users/utoarca/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9:
RuntimeWarning: More than 20 figures have been opened. Figures created through
the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly
closed and may consume too much memory. (To control this warning, see the
rcParam `figure.max_open_warning`).
  if __name__ == '__main__':
/Users/utoarca/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:43:
RuntimeWarning: More than 20 figures have been opened. Figures created through
the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly
closed and may consume too much memory. (To control this warning, see the
rcParam `figure.max_open_warning`).
```

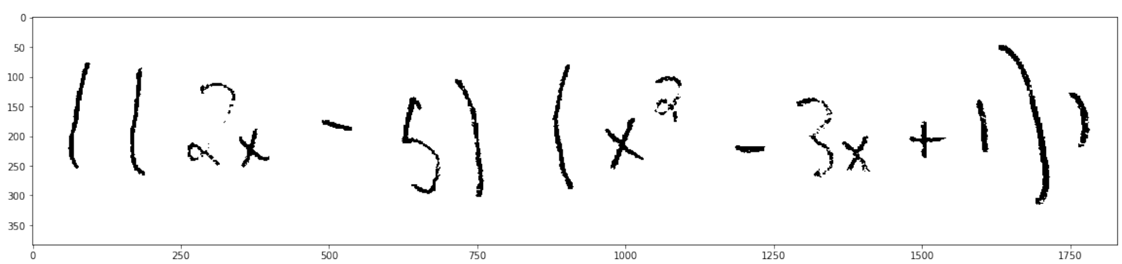


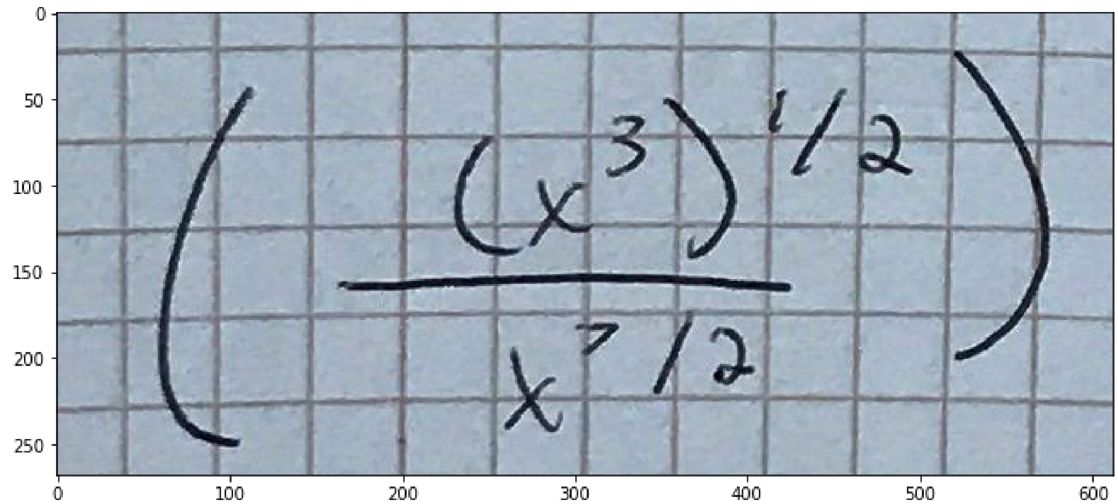


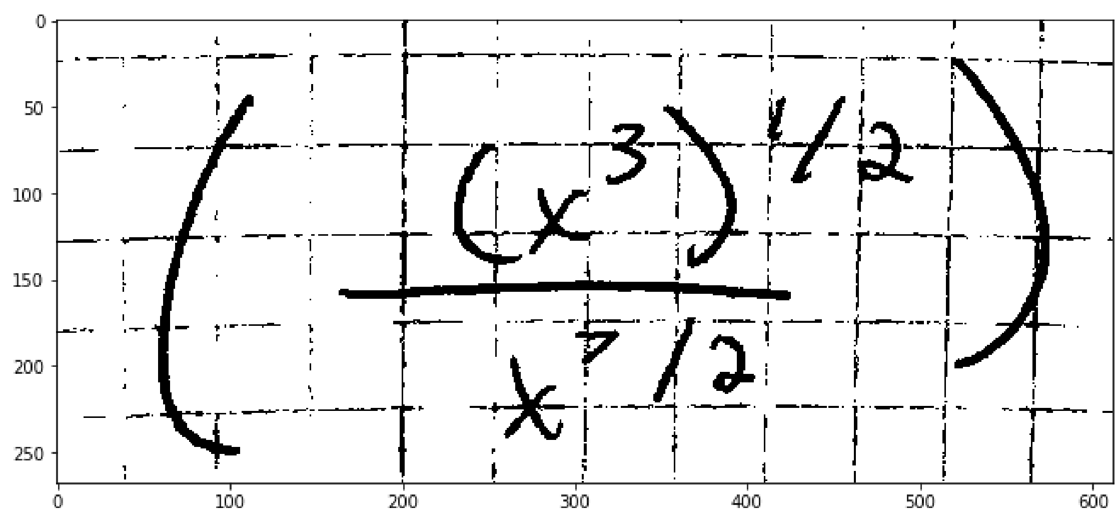


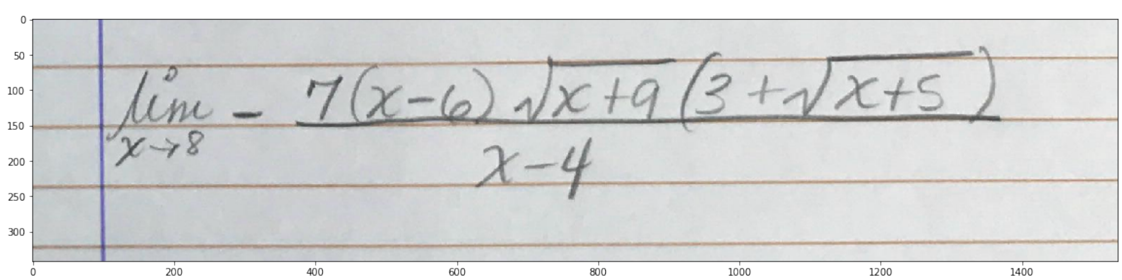


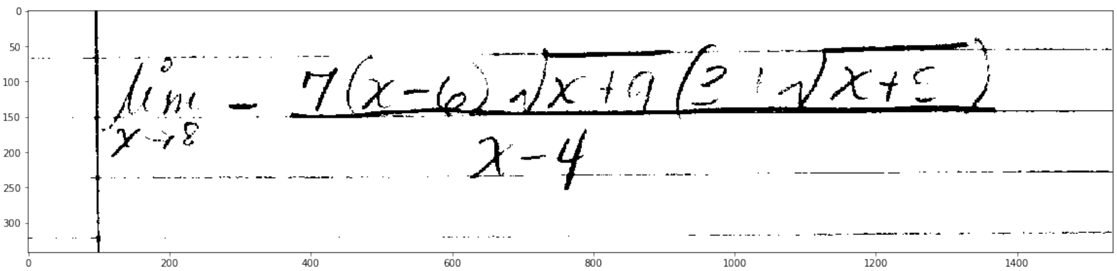


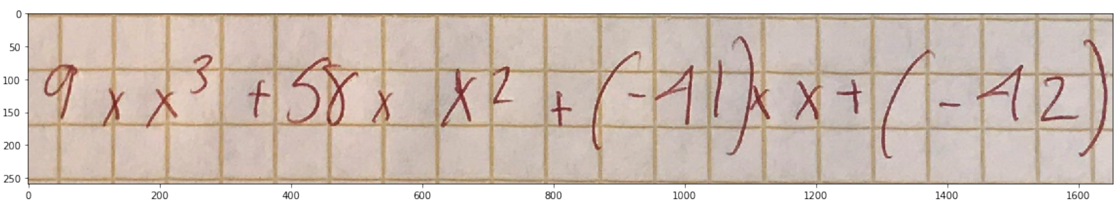


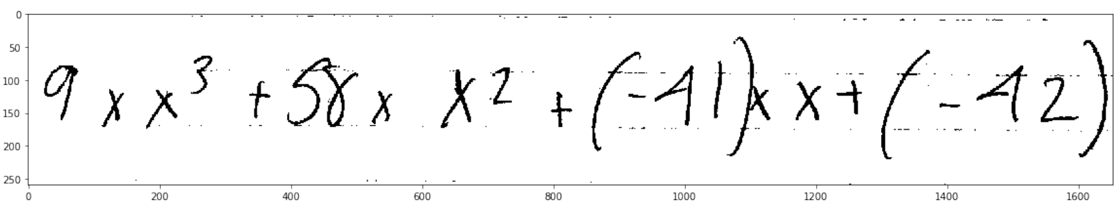


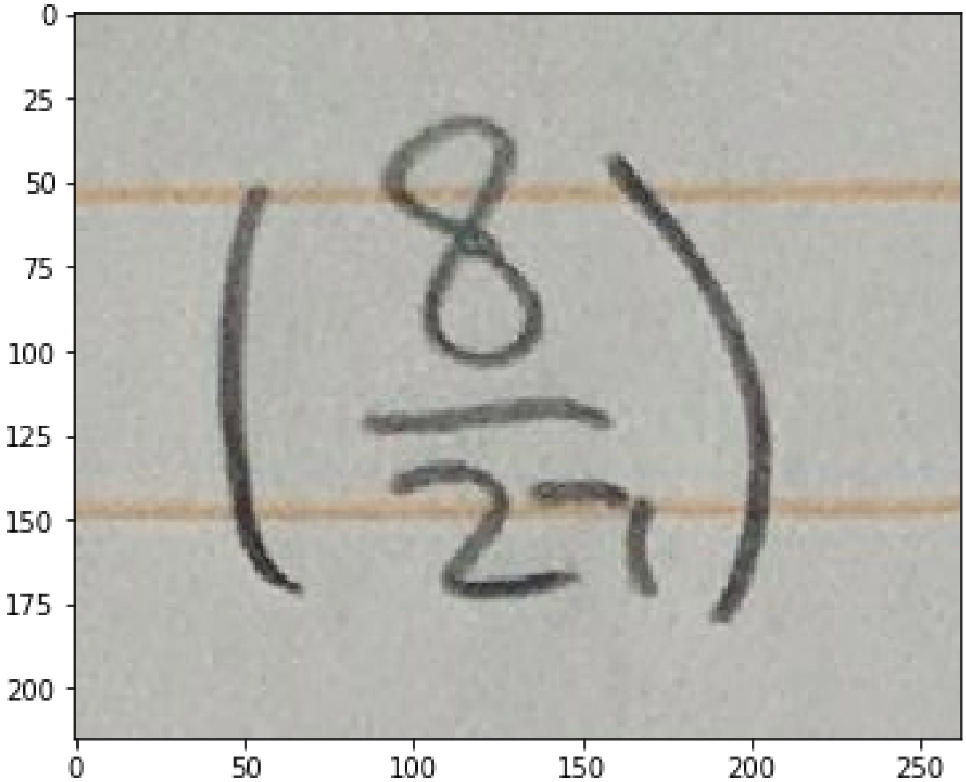


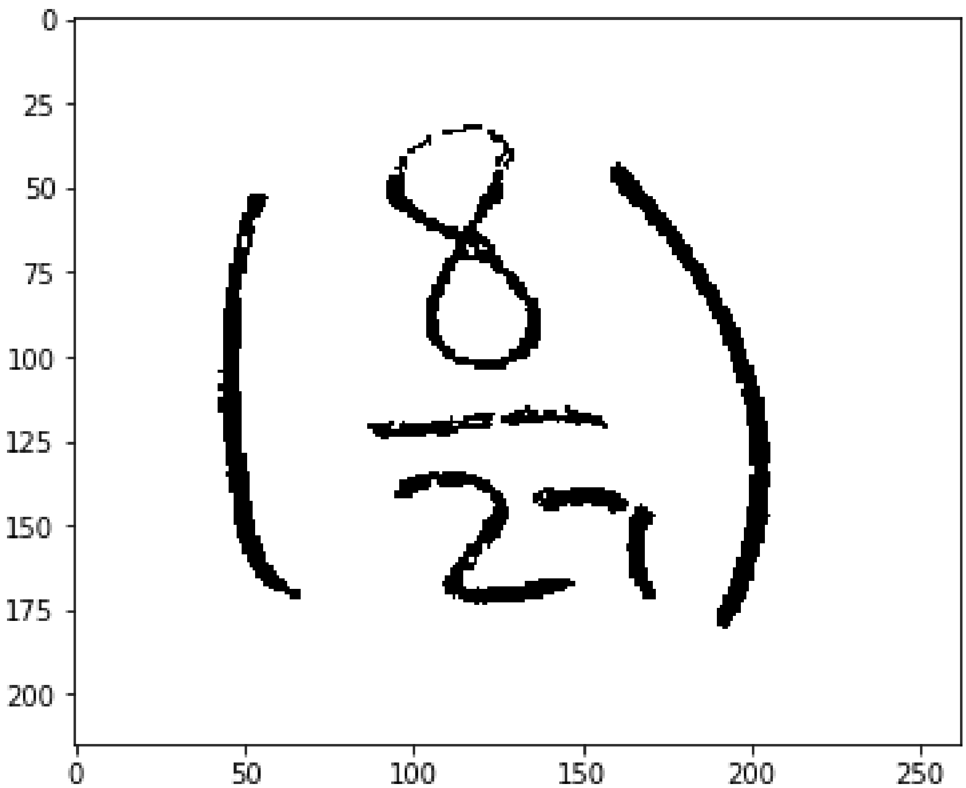


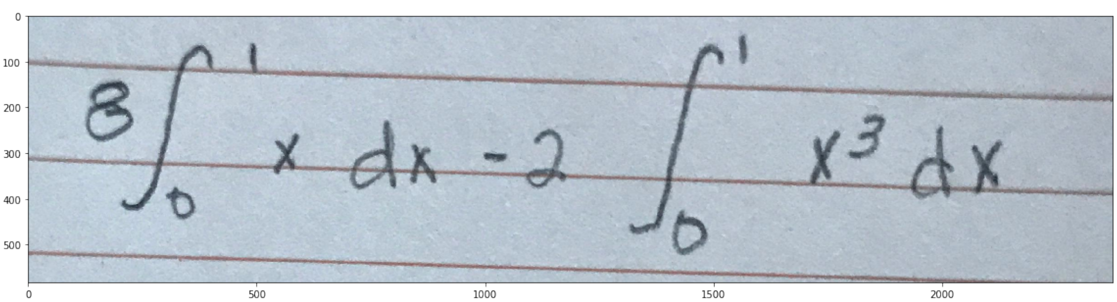


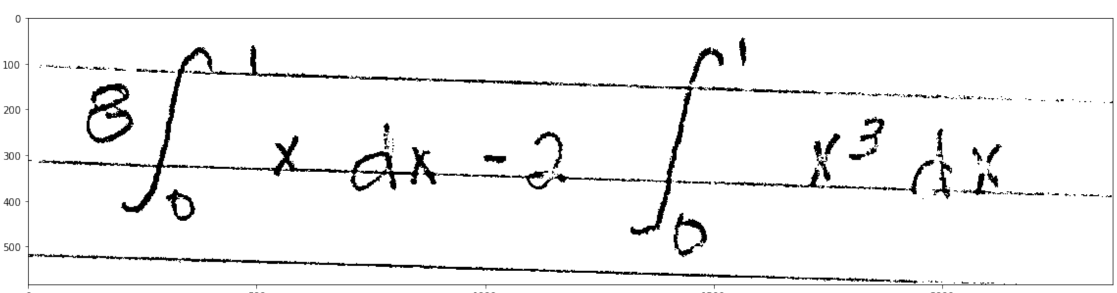

9 x x3 + 58 x x2 + (-41) x x + (-42)

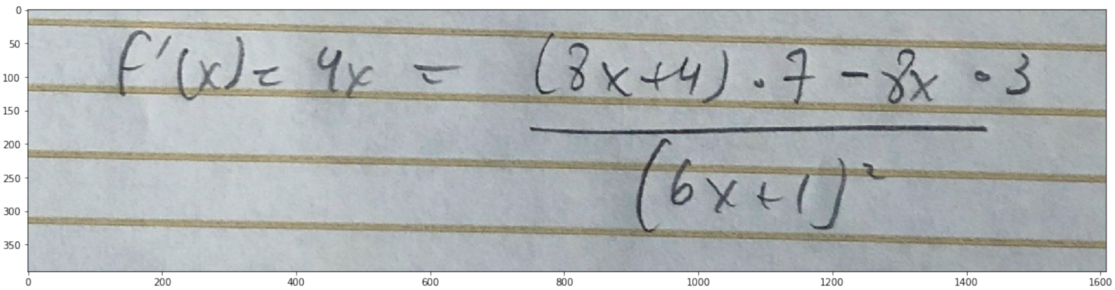

9 x x3 + 58 x x2 + (-41) x x + (-42)

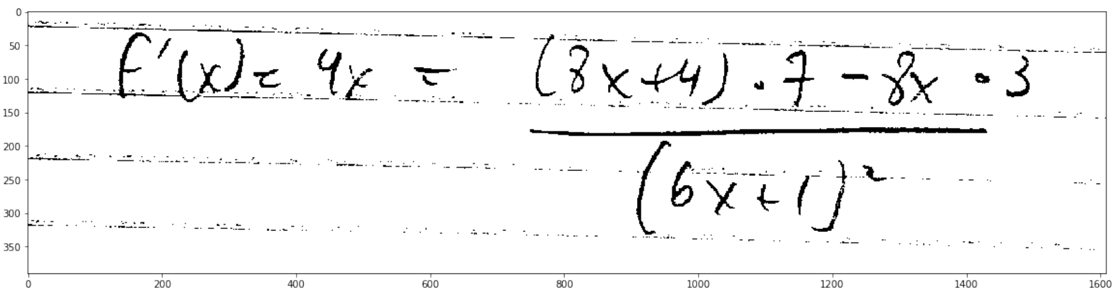

$\left(\frac{8}{27}\right)$
0
25
50
75
100
125
150
175
200
0
50
100
150
200
250

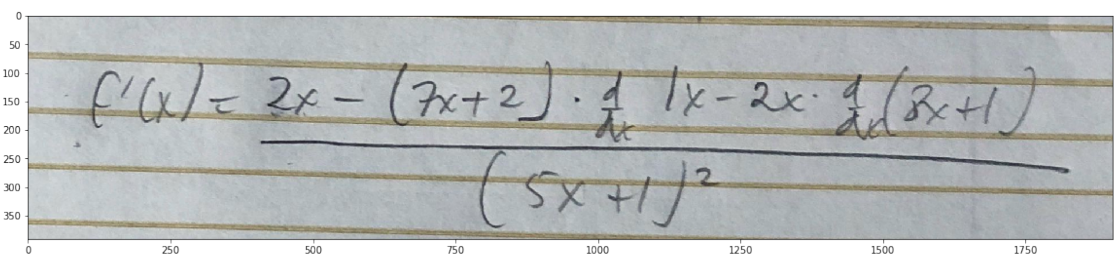


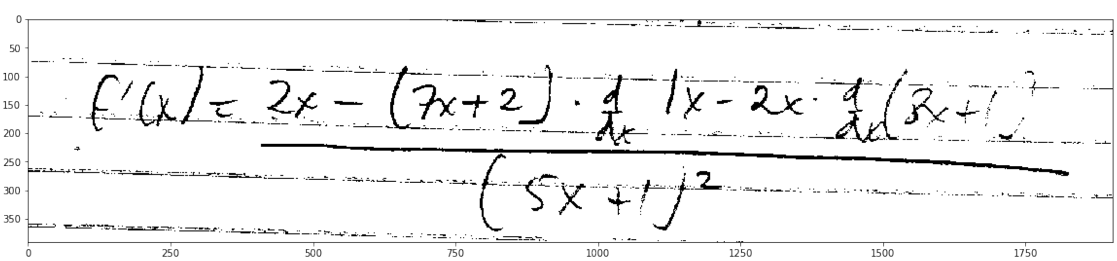


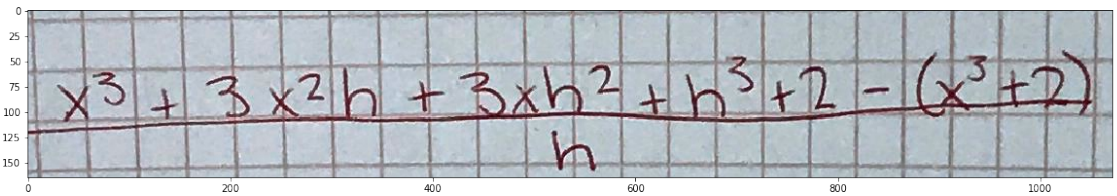


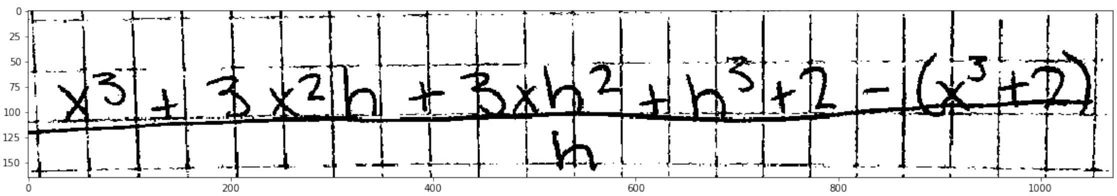

f'(x) = 4x = (8x+4) · 7 − 8x · 3
(6x+1)^2

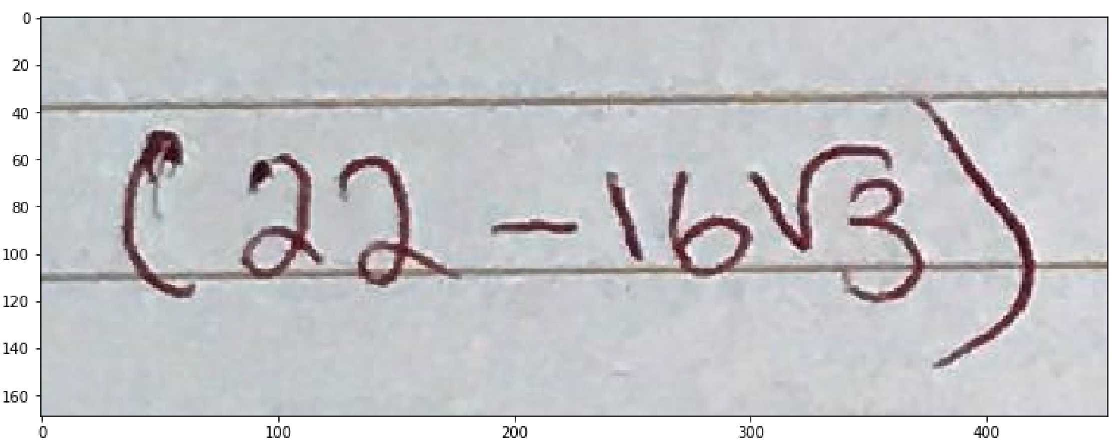

f'(x) = 4x = (8x+4) · 7 − 8x · 3
(6x+1)^2

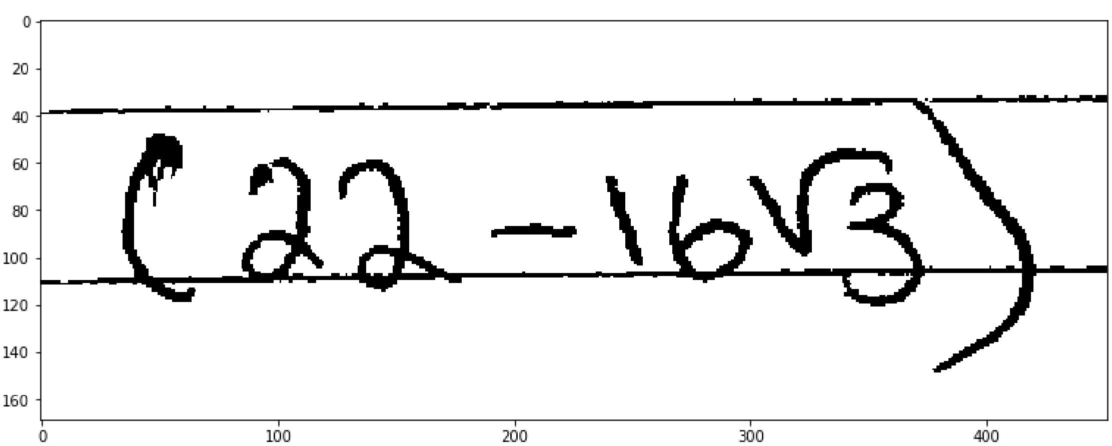

f'(x) = 2x − (7x+2) · d/dx 1x − 2x · d/dx (8x+1)
(5x+1)^2

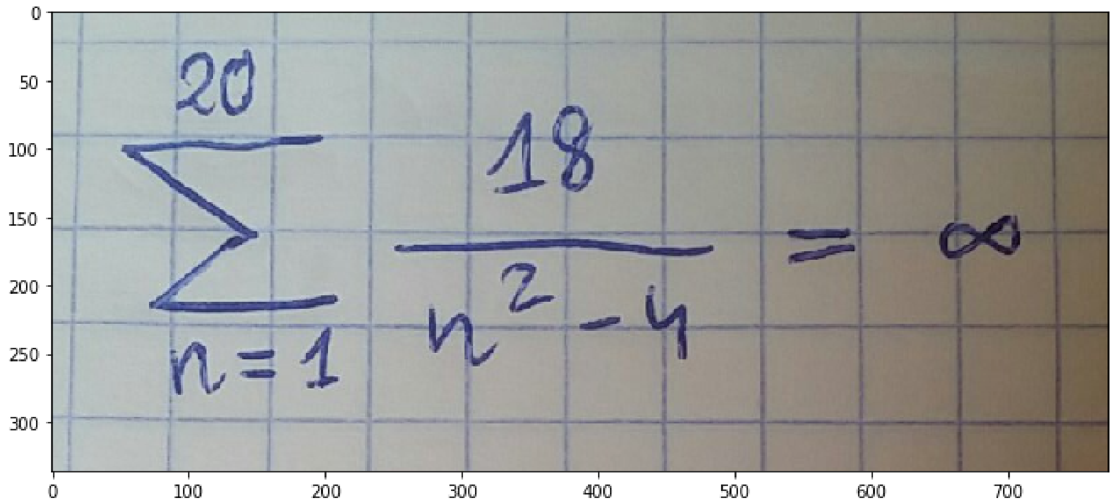

f'(x) = 2x − (7x+2) · d/dx 1x − 2x · d/dx (8x+1)
(5x+1)^2

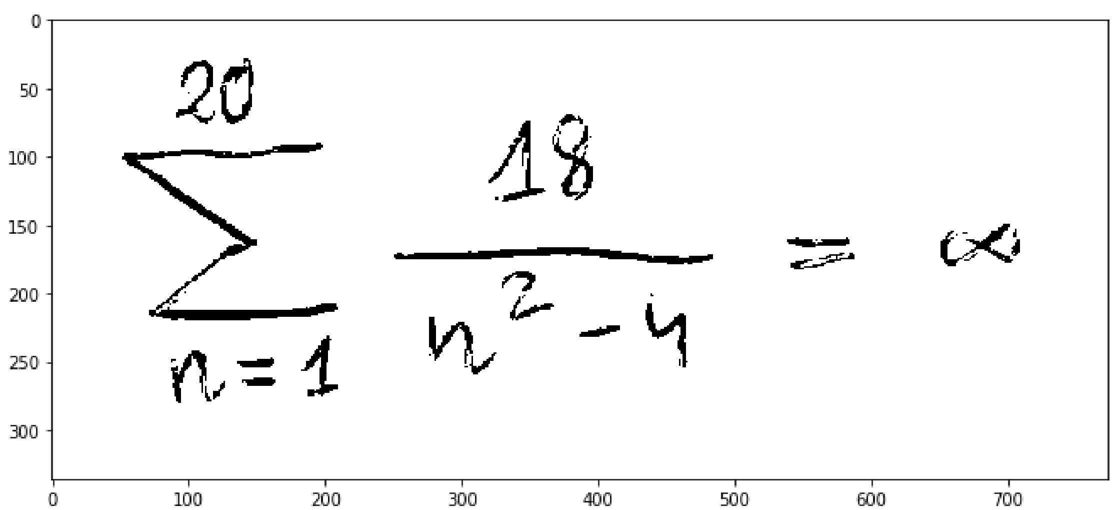

$\frac{x^3 + 3x^2h + 3xh^2 + h^3 + 2 - (x^3+2)}{h}$

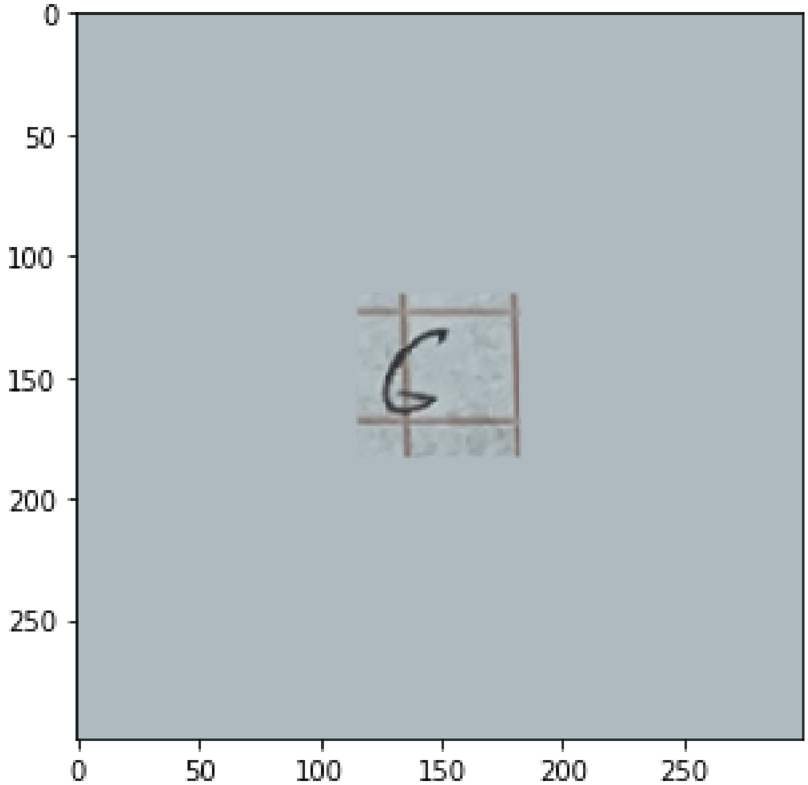

$\frac{x^3 + 3x^2h + 3xh^2 + h^3 + 2 - (x^3+2)}{h}$

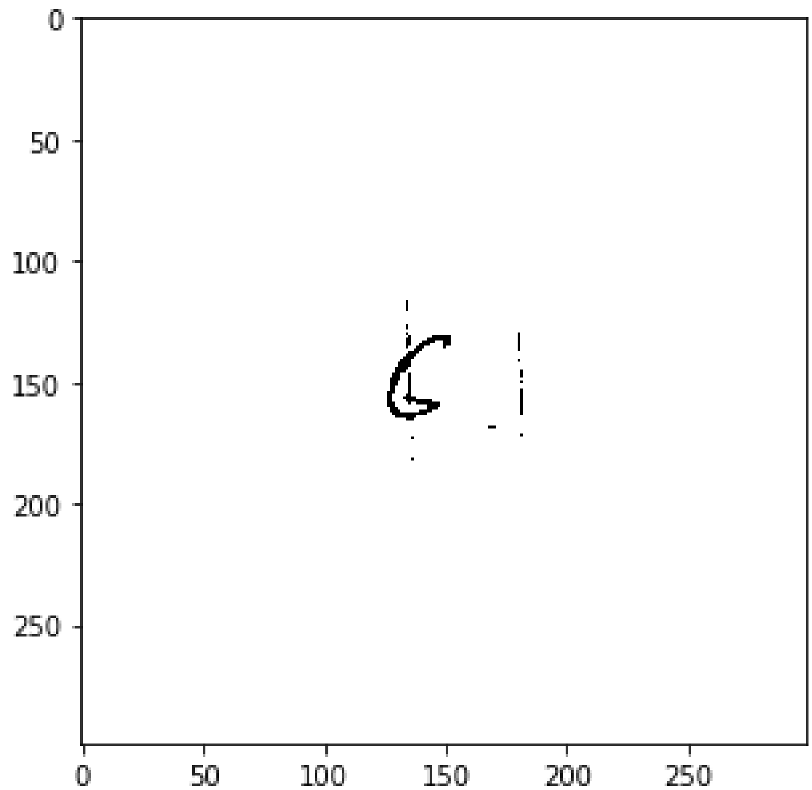

$(22 - 16\sqrt{3})$

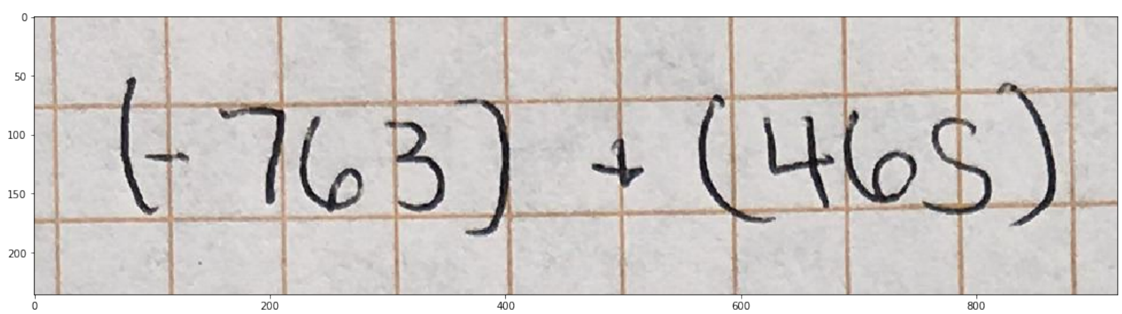

$(22 - 16\sqrt{3})$

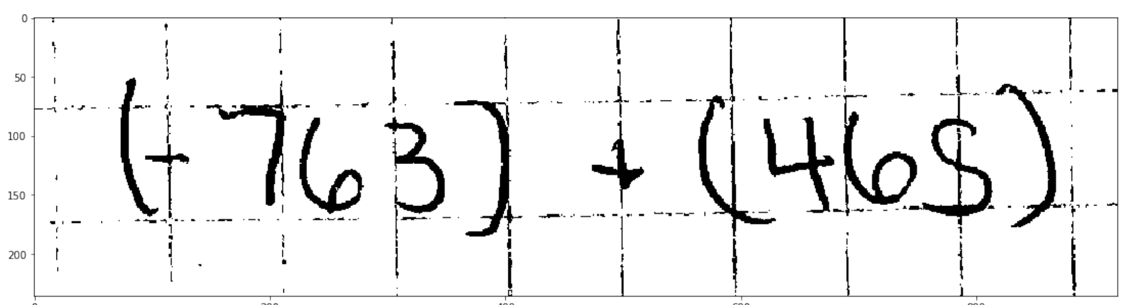

0 50 100 150 200 250 300
0 100 200 300 400 500 600 700

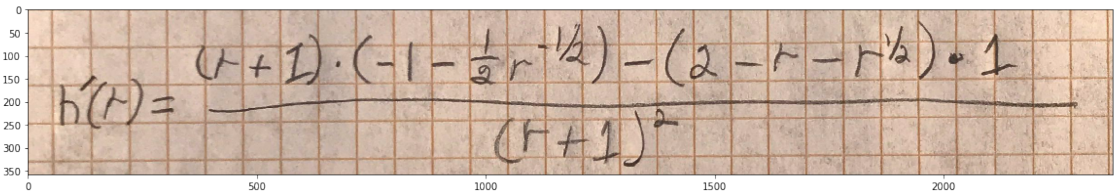

0 50 100 150 200 250 300
0 100 200 300 400 500 600 700

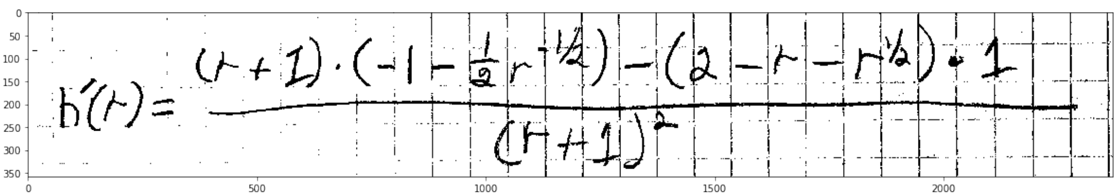

0
50
100
150
200
250
0
50
100
150
200
250

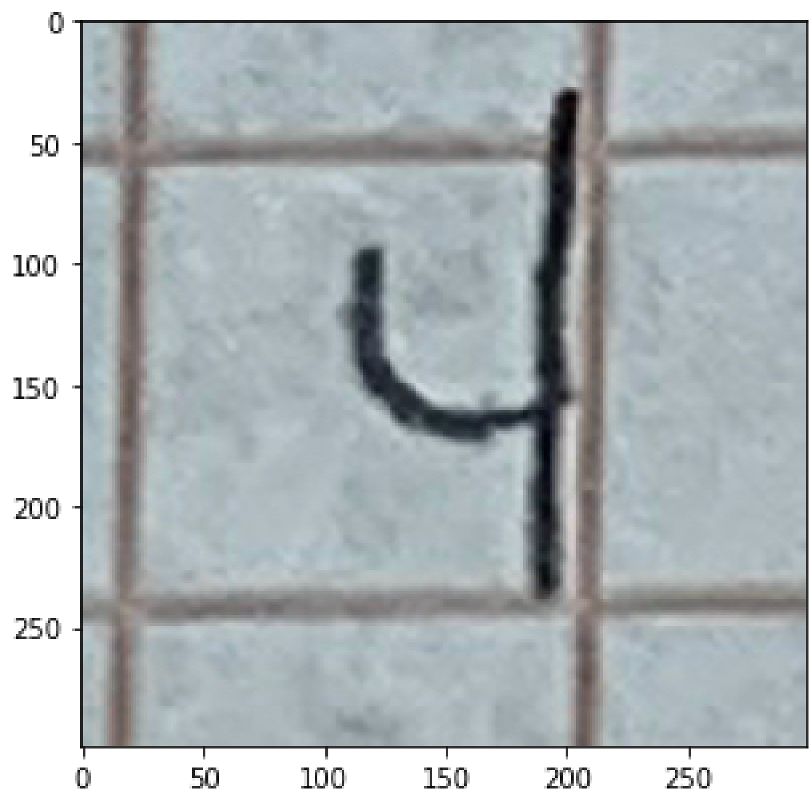


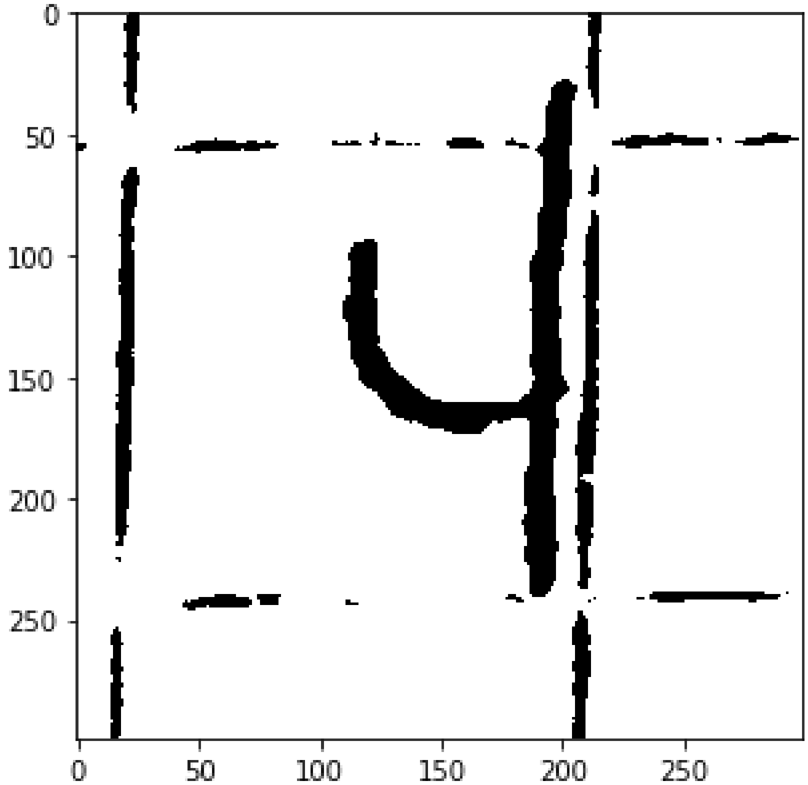





$h'(r) = \frac{(r+1)\cdot(-1-\frac{1}{2}r^{-1/2}) - (2-r-r^{1/2})\cdot 1}{(r+1)^2}$


$h'(r) = \frac{(r+1)\cdot(-1-\frac{1}{2}r^{-1/2}) - (2-r-r^{1/2})\cdot 1}{(r+1)^2}$




0
50
100
150
200
250
0
50
100
150
200
250

[ ]:

[ ]: