E23CSEU0055_Lab02_CSET301(P)

Name - Arihant Gupta - E23CSEU0055

Batch - EB02

Date - Aug 7, 2025

Notebook: cset301-aiml-sem5/aug05/ai_ml_02.ipynb at main · GuptajiRocks/cset301-aiml-sem5

1. Upload Image to Colab Environment
    a.
```
from google.colab import files
uploaded = files.upload()

# Output
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving IMG-20250715-WA0023.jpg to IMG-20250715-WA0023.jpg

#FileName
filen = list(uploaded.keys())[0]
print(filen)
```

2. Load and Display the Image and Convert from BGR to RGB
    a. I will be using the OpenCV library to open and load the image. And display it using the ‘imshow’ function of Matplotlib.
    b.
```
import pandas as pd
import matplotlib.pyplot as plt
import cv2
testi = "test.jpg"

img = cv2.imread(testi)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # BGR to RGB conversion
plt.imshow(img)
plt.axis("off")
```
    c. Output

i. 

3. Resize the image to a standard shape (e.g., 128×128 pixels).
   a.
```
img_resize = cv2.resize(img, (300,300))
plt.imshow(img_resize)
plt.axis("off")
```
   b. Output
      i. 

4. Convert the image to grayscale.
   a. cvtColor function of cv2 has been used.
   b. dispIg is a function created by me for faster plotting.
   c.
```
gray_image = cv2.cvtColor(img_resize, cv2.COLOR_RGB2GRAY)
dispIg(gray_image)
```
   d. Output

i. 

5. Convert the grayscale image to binary using thresholding.
   a. Chose threshold value at 100. (When choosing 200, most image was turning black).
   b.
```
thresh = 100
max_val = 255
_, bin_img = cv2.threshold(gray_image, thresh, max_val, cv2.THRESH_BINARY)
plt.subplot(1, 2, 1)
dispIg(bin_img)
plt.subplot(1,2, 2)
dispFunc(bin_img)
```
   c. Output

   i.
   ii. The differences in the images occur because of the cmap being different.

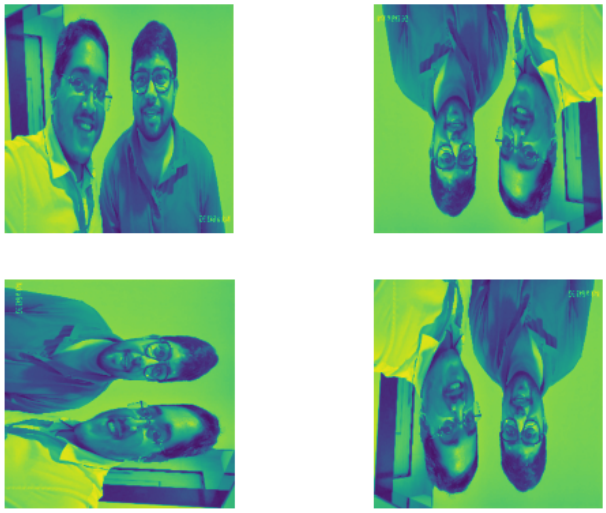
6. Normalize the image pixel values.
   a.
```
normalized = gray_image / 255.0
print("Normalized image shape:", normalized.shape)
print("Min pixel value:", normalized.min())
print("Max pixel value:", normalized.max())
print("Min pixel value:", gray_image.min())
print("Max pixel value:", gray_image.max())
```
   b. Output
      i.
```
Normalized image shape: (300, 300)
Min pixel value: 0.054901960784313725
Max pixel value: 0.9333333333333333
Min pixel value: 14
```

```
Max pixel value: 238
```

7. Apply basic augmentations like flipping and rotating the image.
   a.
```
fh = cv2.flip(gray_image, 1)
fp = cv2.flip(gray_image, 0)
r9 = cv2.rotate(gray_image, cv2.ROTATE_90_CLOCKWISE)
r180 = cv2.rotate(gray_image, cv2.ROTATE_180)

plt.subplot(2,2,1)
dispFunc(fh)
plt.subplot(2,2,2)
dispFunc(fp)
plt.subplot(2,2,3)
dispFunc(r9)
plt.subplot(2,2,4)
dispFunc(r180)

plt.show()
```

   b. Output
      i.


8. Apply filters to remove noise such as Gaussian or median blur.
   a.
```
gauss_bl = cv2.GaussianBlur(gray_image, (3,3), 0)
dispIg(gauss_bl)
```

   b. Output

i. 

9. Flatten the image and display the new shape.
    a.
    ```
    flat_image = gray_image.flatten()
    print(flat_image)

    print(f"Original Shape -> {img.shape}")
    print(f"New Shape -> {flat_image.shape}")
    ```
    b. Output
        i.
        ```
        [182 184 183 ... 186 189 196]
        Original Shape -> (1836, 3264, 3)
        New Shape -> (90000,)
        ```