CS771A Assignment 3

Unsupervised Learners
Aman Raw (190108), `amnraw@iitk.ac.in`
Akash Biswas (200074), `abiswas20@iitk.ac.in`
Akshat Gupta (200085), `akshatg20@iitk.ac.in`
Kembasaram Nitin (200505) `knitin20@iitk.ac.in`
Modem Sai Charan (200586), `modensc20@iitk.ac.in`

Task 1

Describe the method you used to find out what characters are present in the image. Give all details such as algorithm used including hyperparameter search procedures, validation procedures.

Approach

1. Background Pixel Identification

As, the background is lighter than the foreground, so, we took some pointers from this article[1]. So first of all we convert our image to HSV format using $cv2.cvtColor(< BGR\ image >, cv2.COLOR\_BGR2HSV)$. So, for an input image $0.png$ we have the image in HSV format as follows

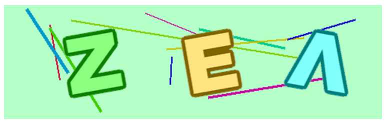
Figure 1: 0.png in BGR

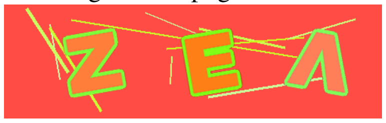
Figure 2: 0.png in HSV

Now, it is quite easy to determine the background pixel of image as $(0, 0)$ is always a background pixel. We can create a mask using $cv2.inRange()$ as follows

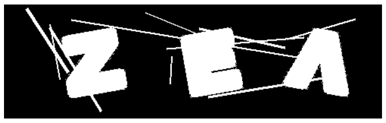
Figure 3: Mask for 0.png

Now, we can just take Bit-wise AND of mask and the HSV image so that we just have the foreground as follows

So, now we are left with only the letters and the obfuscating lines.

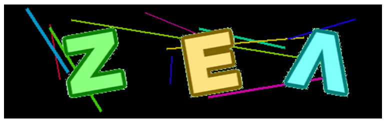
Figure 4: Foreground of 0.png

2. Obfuscating line identification

The obfuscating lines were removed by using $erosion$ from OpenCV library.

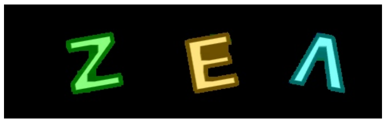
Figure 5: 0.png after erosion

After learning a few parameters, we optimised the hyperparameters so, that there are no obfuscating lines but also the features remain intact.

3. Image segmentation

As, each of the image contain three characters and they are "almost" equidistant along X axis. So, we simply segment the image into three equal images of same width. We first convert the image to numpy array and then do slicing at appropriate places.

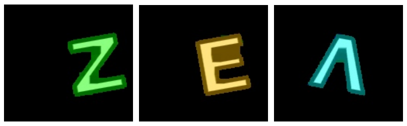
Figure 6: Left, middle & right segments of 0.png

4. Action

Now, we need to switch to PIL for the function $getbbox()$ it offers. The function gives a $4-tuple$ such that we get a box which bounds the characters. So, by cropping and re-sizing, we can always just-fit the character in the image. Also, we remove the color component of the image as it is of little help but definitely would make the model quite large. We did not include rotations of given images in our training set as they did not improve classification greatly, and also increased the model size by 7-fold. Also, the distortion caused by the rotations is somewhat mitigated by the cropping of the images due to the $getbbox()$ function.

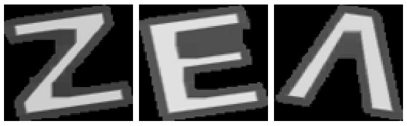
Figure 7: Left, middle & right final segments of 0.png

5. Classification

We employed a OvA classifier using Logistic Regression model using the $scikit-learn$ library for classification of the images.

Task 2

Answered in the zip file submission.

References

[1] How to remove the background of an image using OpenCV?
https://stackoverflow.com/questions/58576698/how-to-remove-the-background-of-an-image-using-opencv

[2] Erosion and Dilation in OpenCV
https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

[3] Python PIL | ImagePath.Path.getbbox() method
https://www.geeksforgeeks.org/python-pil-imagepath-path-getbbox-method/