Assignment - 2

Adrija Bera 210071
Amay Raj 210116
Amandeep Upadhyay 210114
Aniket Sen 210134
Jefrin Jo W 210471

Tanuja Kaleli 211099

Abstract

To design an algorithm that can take a set of images and return the parity of the hexadecimal number in that image.

1 Methodology

Given:

- Size of training images : 500×100 pixels
- Size of reference images : 100×100 pixels

This problem can be broadly classified as multi-class classification problem where we aim to recognize the character by training our model on a set of reference images, and hence find its parity.

Step 1: Pre-processing of train and reference images

- **Monochrome** : All images are converted to gray-scale and a further conversion to monochrome by a threshold offered by *Otsu's Method*.
- **Inversion** : Images are inverted so that the background black matches with extra black pixels introduced around the corners due to rotation.
- **Erosion** : This is used to reduce the thickness of lines present in the image. As a result, the obstructing lines are almost removed. The thickness of letters also reduced but they are still good enough to be further used.
- **Rotation** : 7 copies of each reference character are made with rotation angles of $-30, -20, -10, 0, 10, 20, 30$ degrees and stored as new reference images in an array. These are used for comparison on the training/testing images.

Step 2: Cropping and Comparison with Reference Set

- **Cropping** : Obtained test image is cropped using a tuned parameter in order to extract last character.
- Dimensionality of images are made consistent through appropriate resizing.
- Each image is then compared with each of 16×7 reference images through the process of **'Subtraction'**.

Subtraction: Two images with same dimensionality are superimposed and corresponding pixels are subtracted in the vector matrix. In the resulting matrix, lesser the sum of elements in matrix, lesser the difference and hence, more the resemblance.

Preprint. Under review.

Step 3: Correct classification based on Subtraction matrix

- Each test image is compared with 16×7 reference images and sum of elements of difference matrix is stored in a new matrix.
- Least element of above gives us the most resembled character and algorithm outputs the same.

Step 4: Parity generation

After successful identification of last character, parity of the last character gives us the parity of the hexadecimal number itself.

2 Hyperparamater Tuning

"Selecting an optimal value for a hyperparameter is often considered an art rather than a science."

Hence, there exists no fixed algorithm to find out the best hyperparameters. Instead, it depends on its utility in the Machine Learning model .

The hyperparameters used in our attempt were as follows:

1. Threshold
2. Eroding Thickness
3. Cropping Last Character
4. Horizontal Cropping of Reference Images
5. Vertical Cropping of Reference Images
6. Resizing parameter

2.1 Threshold

Firstly, the images are converted to gray-scale to remove the unnecessary information of colors. Then, the images are converted to monochrome using a thresholding parameter, which is determined using the *Otsu's Method*.

2.2 Eroding Thickness

The erosion parameter is the kernel size. Kernel is the structuring element that determines the precise effect of the erosion on the input image. A too low size would not affect at all while a too large kernel size erodes the boundaries of the characters too. A kernel size of 2 gave optimum results as to eroding obstructing lines as well as maintaining legible thickness of characters in image.

2.3 Cropping Last Character

For obtaining the last character which is crucial to parity generation, we crop the test image with a boundary of $10 : 100 \times 360 : 450$ pixels. This was found manually. We used matplotlib.pyplot to display the image in an XY plot. We found the right X and Y axes limits which bounded a straight image closely (refer Fig.1). Then, cropped it with an offset of 18 horizontally on both sides and an offset of 8 vertically above and below so that the boundary could hold the character even if it was rotated to ± 30 degrees (refer Fig.2). This results in a 90×90 image, with boundaries $10 : 100 \times 360 : 450$ pixels.

2.4 Horizontal Cropping of Reference Images

First limitation is that the reference image should also be 90×90 for subtraction. We cropped the image instead of resizing it from 100×100 to 90×90, so that the resolution of image doesn't change. The possible horizontal cropping limits are from $0 : 90$ to $10 : 100$. We have tuned this as a hyper-parameter and found that $5 : 95$ gave 100% accuracy in parity detection.

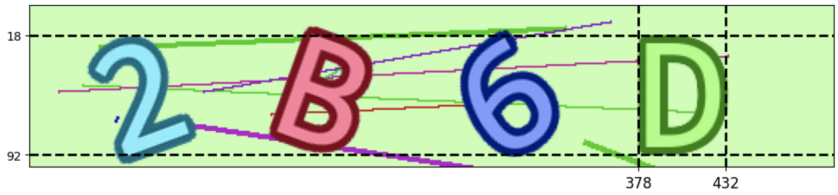


Figure 1: Straight Image Boundary

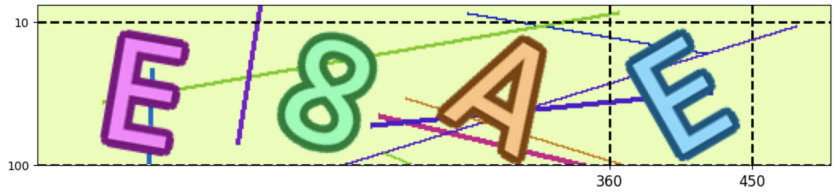


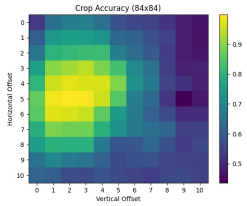
Figure 2: Boundary chosen to Enclose $\pm 30^{\circ}$ Rotated ones

2.5 Vertical Cropping of Reference Images

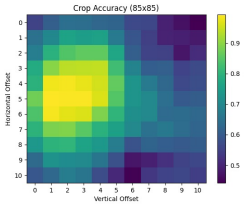
The possible horizontal cropping limits are from $0 : 90$ to $10 : 100$. We have tuned this as a hyper-parameter and found that $3 : 93$ gave 100% accuracy in parity detection.

2.6 Resizing Parameter

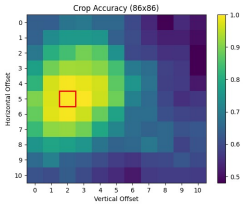
We tried to resize the images initially, to a smaller value say, 50×50 pixels, to reduce computing time and found a significant drop in accuracy. As a result, size was slowly decreased from 90×90 and the below heat maps show our optimum choice of resizing parameter with respect to different croppings of the reference image.



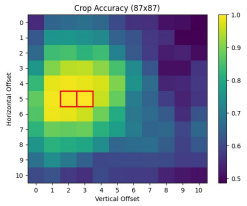
(a) 84×84 pixels



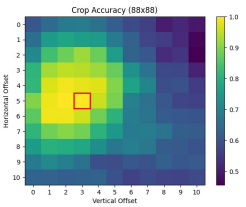
(b) 85×85 pixels



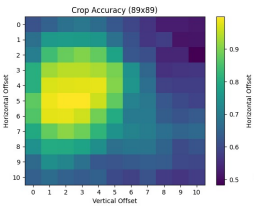
(c) 86×86 pixels



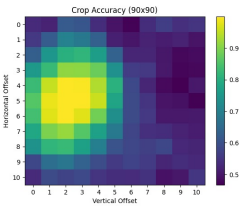
(d) 87×87 pixels



(e) 88×88 pixels



(f) 89×89 pixels



(g) 90×90 pixels

Figure 3: Images depicting the heat maps for vertical cropping of images in the range 84×84 pixels to 90×90 pixels

3 Conclusion

Based on the above images and fine-tuning of various hyper-parameters, we have reached the following conclusions:

- Using pictures in gray-scale and turning them into monochrome resulted in better identification and reduced the execution time around $3 - 4$ ms.
- Optimized sizes of respective images were found through manually looking at the image and fine-tuning the hyper-parameters.
- Subtraction method provided an appreciable parity accuracy.

Through the optimization process outlined above, our model achieved a **parity match score of** 1.0 (accuracy of 100%) within a training and execution time of 7 **milliseconds** per image when 2000 images were used. (subject to change because the first iteration takes around 60 milliseconds due to reference image preprocessing).

References

[1] OpenCV Tutorial for Erosion and Image Morphology

[2] Documentation for *Otsu's method*

[3]Algorithm for Subtraction of Image Matrix