# CS771 Assignment 3

ALLAN ROBEY, SUBHAJIT PANDAY, DIVYESH DEVANGKUMAR TRIPATHI, ARCHIT GUPTA, AVNISH TRIPATHI

TOTAL POINTS

**90 / 100**

QUESTION 1

## *1* Method Description **40 / 40**

✓ **+ 40 pts** *Method description*

**- 15 pts** Insufficient method description

**+ 0 pts** No submission or else very sparse description

QUESTION 2

## *2* Experimental Results **50 / 60**

**+ 0 pts** Correct

**+ 50** *Point adjustment*

GROUP NO: 43

Grading scheme for code:
Submission size s (in KB): s < 128 (10 marks), 128 <= s < 512 (8 marks), 512 <= s < 1024 (6 marks), s > 1024 (4 marks)
Inference time t (in sec): t < 5 (10 marks), 5 <= t < 20 (8 marks), 20 <= t < 40 (6 marks), t > 40 (4 marks)
Code match c: floor( c * 40 ) marks

s = 9431.7 KB: 4 marks
s = 13.2 sec: 8 marks
score = 0.9733 : 38 marks
TOTAL: 50 marks

## 1 Method Description 40 / 40

✓ **+ 40 pts** *Method description*

**- 15 pts** Insufficient method description

**+ 0 pts** No submission or else very sparse description

## 2 Experimental Results 50 / 60

**+ 0 pts** Correct

**+ 50** *Point adjustment*

GROUP NO: 43

Grading scheme for code:

Submission size s (in KB): s < 128 (10 marks), 128 <= s < 512 (8 marks), 512 <= s < 1024 (6 marks), s > 1024 (4 marks)

Inference time t (in sec): t < 5 (10 marks), 5 <= t < 20 (8 marks), 20 <= t < 40 (6 marks), t > 40 (4 marks)

Code match c: floor( c * 40 ) marks

s = 9431.7 KB: 4 marks

s = 13.2 sec: 8 marks

score = 0.9733 : 38 marks

TOTAL: 50 marks

# CS771 Introduction to Machine Learning Assignment 3

**Group Name: Beginners2**

**Archit Gupta** 21111015

**Avnish Tripathi** 22111014

**Allan Robey** 22111007

**Divyesh Tripathi** 22111020

**Subhajit Panday** 22111058

## 1 Algorithm Description:

In order to classify the image, we first pre-processed it by separating the characters in the background from those in the foreground. After segmenting the original image into individual characters, we next fed the characters into a Linear SVM for the purpose of classification. A detailed description of the entire flow of the work has been prescribed below:

### 1.1 Pre-Processing Phase:

Character separation is a challenging task due to the RGB format and obfuscation lines in the provided images. The following steps were taken in order to prepare an image for segmentation (by removing obfuscation lines):

- The background pixels were successfully eliminated by learning the colour of the backdrop from corner pixels and subtracting that colour from the image.
- We utilised a conventional 3x3 kernel and eroded the image for 8 iterations using the cv2.erode() function. For iterations 4,5,6,7, and 8, we randomly selected 20 photos to serve as the test set. We discovered from the dry runs that 8 iterations were ideal for removing the obfuscation lines while yet maintaining the unique qualities of the individual letters.
- To create a binary image with only black and white pixels, the image was converted to grayscale and then thresholded.

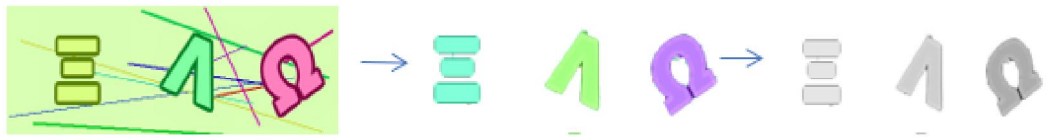

Figure 1: Pre-Processing of image

## 1.2 Segmentation Phase:

Below is a description of the heuristic we have used to separate the image into distinct characters. The individual character segments found by our system after segmentation are seen in Figures 1 and 2.

- The provided CAPTCHA training examples were carefully examined, and we discovered that even though the images were tilted around a pivot, the letter boundaries did not overlap.

- We can clearly distinguish between the characters in the image, which will help us distinguish between nearby characters by scanning along the vertical axis.

- It is possible to segment out the characters from the provided CAPTCHAs by sweeping the entire image with a vertical line from the left end to the right end, or a column vector.

- When we found a bright pixel in the sweeping column vector, we inferred that the pixel was the beginning of a contour. This was only possible because, as was already said, a vertical line could be used to divide the characters. We further thought that once every pixel in the sweeping vector had expired, we had completed sweeping over a significant character. In order to correctly scan the final character of the CAPTCHA, either all of the pixels on the sweeping vector turned black or we reached the end of the image.

- With the method prescribed above, we can thus determine the total number of characters in an image as well as the specific segments that each contain a character.

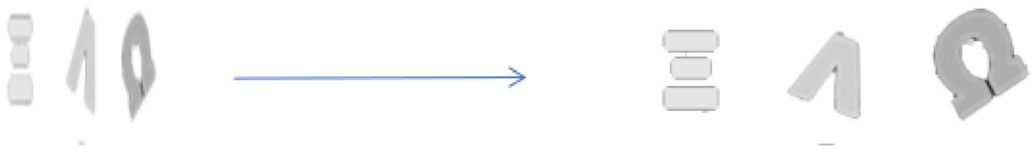
Figure 2: Segmentation of image

## 1.3 Classification Phase:

- We input the separated characters to Linear SVM to determine which characters are present in the segmented image after getting the separated characters. First, all of the photos were scaled to 30x30 pixels for input, and output labels were mapped to corresponding input images using dictionary. To be employed in the prediction process, these encodings were dumped to a file.

- In order to learn the model, we have used scikit learn train test split of 90-10.

- We need to tune the hyperparameter 'C' in order to obtain the optimal performance. The scikit-learn GridSearch function has been used to tune the hyperparameter 'C'.

- For finding value of Hyperparameter 'C' there are two famous techniques random search and grid search. We have used both these techniques. First, grid search is used to get an idea of the required range and then random search is used to get a good value of the hyperparameter within that range. In grid search, we pass predefined values for hyperparameters to the GridSearchCV function. We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take.

- GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method.

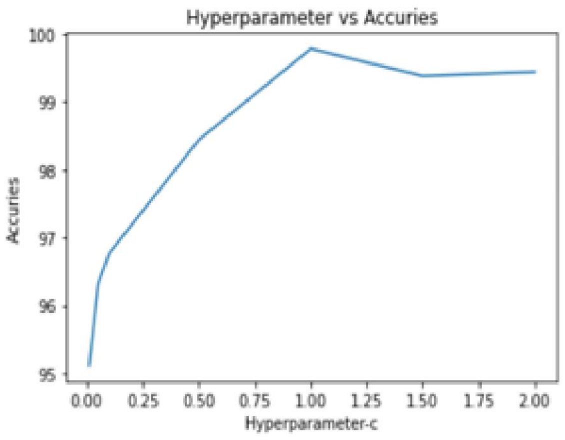

Figure 3: Hyperparameter 'C' Tuning

## Reasons for choosing Linear SVM as classifier:

- For the purpose of image classification various methodologies had been employed such as Random Forest Classifier, Logistic Regression, Linear SVM and Kernelized SVM.
- All the classifiers had given considerable performance when employed for the task of image classification. However, it could be observed that Random Forest Classifier and Linear SVM gave a small marginal better performance than Logistic Regression and Kernelized SVM.
- Now, Random Forest Classifier had a minute performance advantage over Linear SVM however the model size of Random Forest Classifier is quite high as compared to Linear SVM.
- Thus, due to the above-mentioned factors, we have determined that Linear SVM would be the ideal classifier for the concerned problem.

| **Model** | **Size** | **Train Time** | **Accuracy** | **Test Time** |
|---|---|---|---|---|
| Linear SVM | 4 MB | 4 sec | 99.78 | 84 sec |
| Kernel SVM | 8 MB | 9 sec | 97.85 | 171 sec |
| Logistic Regression | 174 KB | 274 sec | 96.15 | 82 sec |
| Random Forest | 12 MB | 544 sec | 99.80 | 80 sec |

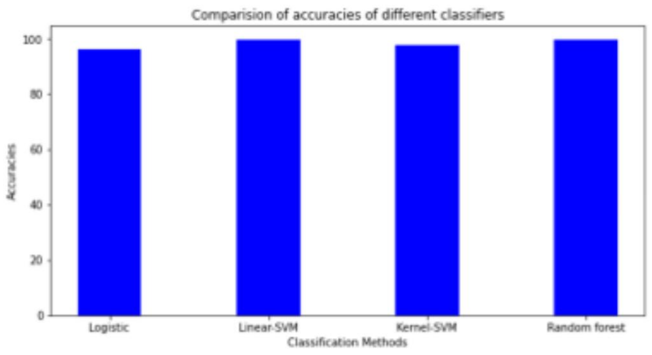

Figure 4: Comparison of Accuracy of Different Classifiers