

# Text Independent Writer Recognition Using LBPH & SVM

January 17, 2021

## Team #1

Mahmoud Othman Adas SEC:2, BN:19<sup>\*</sup> and Yosry Mohamed Yosry SEC:2, BN:33<sup>†</sup> and  
Ahmad Mahmoud AbdElMen'em Afifi SEC:1, BN:5<sup>‡</sup> and Abdulrahman Khalid Hassan SEC:1, BN:30<sup>§</sup>

Department of Computer Engineering, Cairo University

Email: <sup>\*</sup>mahmoud.ibrahim97@eng-st.cu.edu.eg, <sup>†</sup>yosry.mohammad99@eng-st.cu.edu.eg,

<sup>‡</sup>Ahmed.Afifi98@eng-st.cu.edu.eg, <sup>§</sup>abdulrahman.elshafie98@eng-st.cu.edu.eg

### *Abstract—*

#### I. INTRODUCTION

#### II. PIPELINE

#### III. PREPROCESSING MODULE

#### IV. FEATURE EXTRACTION MODULE

#### V. CLASSIFICATION MODULE

#### VI. PERFORMANCE ANALYSIS

#### VII. SPEED ENHANCEMENTS

We put alot of effort on speeding up the training process. The most effective optimization was parallelizing the feature extraction by extracting each image's features in a seperate process and then collecting all the features before training.

Processes are quite heavy, but python threads are totally useless, thanks to GIL's locking mechanism. We believe that if we port the code to another language, the execution time would be much lower using threads and manual memory allocation.

Python lists are known to be very slow, so we replaced them all with numpy arrays, and allocated most of the needed memory ahead

before all training. A quite speed gain came from fine tuning `skimage` and `OpenCV` parameters.

We tried to use `Numba` and `Cython` to optimize the execution time but they didn't have an effect. It was probably becaue most of the code calls `numpy`, `skimage` and `OpenCV`, which are all written in C and well optimized for memory and `cpu`.

#### VIII. UNSUCCESSFUL TRIALS

We started with and settled on using LBPH for feature extraction and SVM for classification. They both gave around 80% accuracy at the beginning, and with tuning for preprocessing the accuracy reached 99% over large tests. During that, part of the team were testing other feature extraction methods and classifiers.

We tried to extract *Histogram of Oriented Gradients* (HOG) features. Using HOG gave very low accuracy (46%) on 15 test cases. We extracted HOG from the binary image and then gray image, with no visible changes.

Then we tried to extract the *Hu Moments* that are used to describe the shapes. We extracted *Hu Moments* from each binary line in the image.

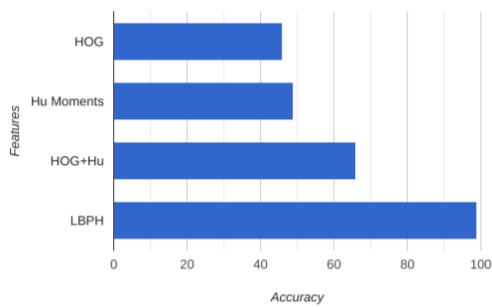


Figure (1) Feature Extraction Methods Accuracy for 15 tests.

Using *Hu Moments* with SVM gave accuracies lower than that of HOG on the same number of test cases.

Being very low made sense, because we tried to describe the shape of the whole line. So we tried to extract *Hu Moments* from a sliding window of size  $13 \times 13$ . This gave accuracy of  $\tilde{48}\%$  on 15 tests. On some lucky iterations, it gave  $\tilde{80}\%$ .

Then we tried to mix both HOG features and *Hu Moments*. This gave us accuracy of  $\tilde{66}\%$  on 15 tests. It wasn't slower than only HOG, because we used subset of both features.

By this time, LBPH reached  $\tilde{99}\%$  accuracy on 200 tests. So we abandoned optimizing the feature extraction anymore.

Figure 1 shows the accuracies for the different features.

We tried another classification method beside SVM. We used *K-Nearest Neighbours* (KNN), and it gave the same accuracies of LBPH but was noticeably slower. It made sense that KNN is slower, because it iterates over the features multiple times to find the mean and cluster them.

## IX. FUTURE WORK

### X. WORKLOAD DISTRIBUTION

A. *Mahmoud Othman Adas*

- TODO

B. *Yosry Mohamed Yosry*

- TODO

C. *Ahmad Mahmoud AbdElMen'em Afifi*

- TODO

D. *Abdulrahman Khalid Hassan*

- TODO

## XI. CONCLUSION