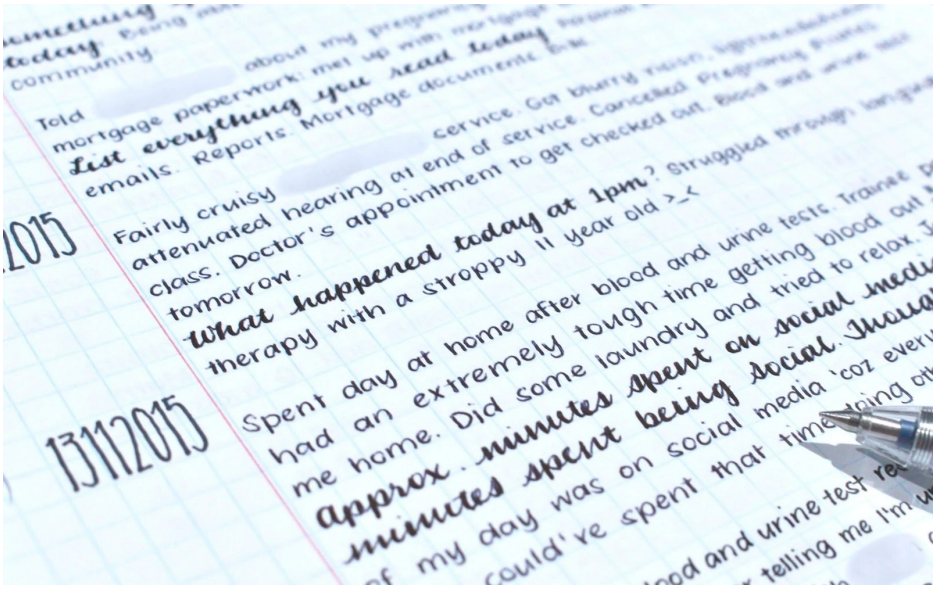
Cairo University
Faculty of Engineering
Credit Hours System



Writers Pattern Recognition

CMPN450: Pattern Recognition and Neural Networks

Submitted To:

Dr. AbdulMoneim Bayoumi

Eng Hussein Fadl

Eng Ahmed Salem

Group 13 Members

- Kamel Mohsen Kamel - 1162325
- Ahmad Nader Adel - 1162296
- Samy Saeed - 1165136
- Hazem Ayman - 1162144

Table of Contents

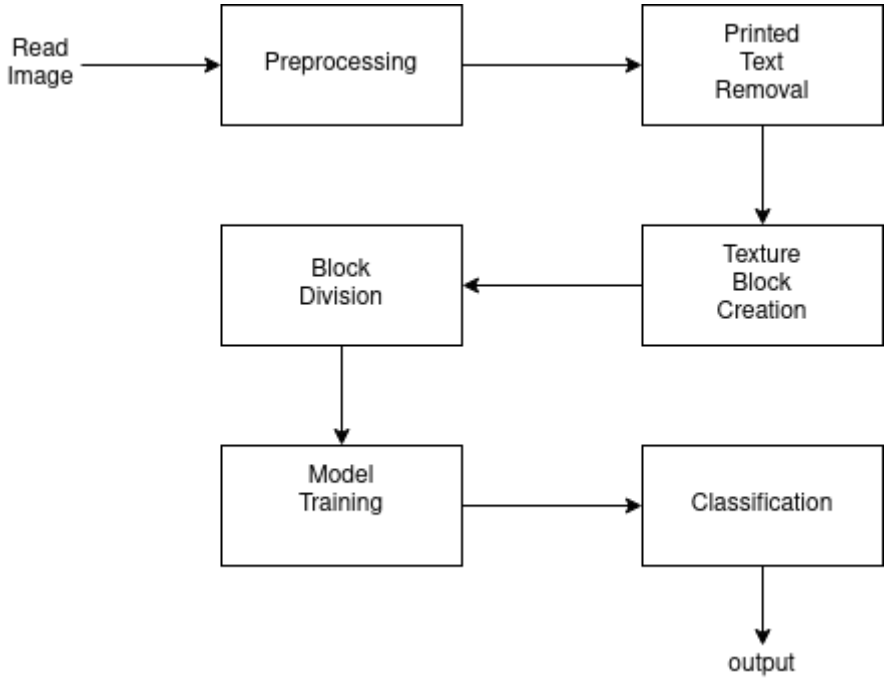
Introduction

The problem of identifying writers based on their handwriting is an interesting challenge, as it is crucial in forensic science, and important to prove unidentified documents that belong to famous writers, or to prove wills of deceased persons. This problem has been researched by many scientists and engineers and many approaches have been implemented and tested.

One of those approaches has been implemented by Priyanka Singh et al, here. The approach uses many different features like LBP, LPQ, DWT+LEP, and many others. Also, various classifiers were used like KNN, SVM, and others.

The highest accuracy detected was returned using LBP feature and SVM classifier. As such, we used this approach to implement our classifier.

Pipeline



Preprocessing Module

1. **Gaussian Blur:** a Gaussian filter is applied to remove any unwanted noise that can affect finding contours later.
2. **Binary Thresholding:** then we apply binary thresholding to get a binary image.
3. **Bitwise Not:** inverting the image to get a black background and white text.
4. **Morphology**: then we apply morphology to connect elements to improve accuracy of contours finding.

Top Removal

Further processing is applied to remove the printed text at the top of the page, since this is text-independent handwriting classification, by using findContours function of opencv library and sorting the contours that satisfy a certain condition which indicate that is an handwritten part. There is some value was obtained by try and error til it succeeds.

Texture Block Creation

The paper suggested creating a “texture block”, which are essentially blocks of sampled condensed text from the original document. So, we rearrange the images by removing space between words (horizontal space) and space between lines (vertical lines). This is done by finding sentences and connected components and copying them into a new image frame and dismissing the space between them.

Block Division

In this last bit of preprocessing, the full texture block that was created in the last step is sampled randomly into 9 smaller, fixed size texture blocks. (Size is 128*256) based on the paper’s recommendation.

Local Binary Pattern (feature extraction)

Then local binary patterns are calculated for each of the texture blocks, and a histogram is computed.

Training and Classification

The chosen classifier is Support Vector Machine

Support Vector Machines are very powerful in classifying cases with extreme features, this means that these classes can be very similar and we aim to focus on the extreme features that draw the line for our classification. It is also very good for classes with multidimensional features and helps fit the hyperplane that represents the decision boundary. For these properties, SVMs have proven great results compared to KNNs and other classification algorithms in the case of writer recognition that we have in hand. Our SVM has a linear kernel and a C parameter of 300 based on an optimization function that tried the model with different values then we chose the C that got the best possible accuracy.

Performance Analysis:

Classifications and timings for each test case are output in a classifications.txt and time.txt. We also output errors in a file called error.log . The code will not stop if

an error occurs as a try-except block was added, if an exception occurs a random choice is output instead, and the error.log shows the exception happened at which test case.

As optimizations are still being made to the project at the time of writing this report, the accuracy might be higher in the given code (inshallah), but for now these are the results we obtained:

Number of test cases	Accuracy	Average test case classification time	C (regularization parameter)
50	96%	30 seconds	300
100	98%	30 seconds	300
200	97 %	30.96 seconds	300

- The Test cases for 200 were different than for 50, and 100 , since we executed the code on two different pcs.

Other Dismissed Approaches

- The paper suggests ordering lines by using the following equation:

$$Y_{next} = Y_{previous} + \frac{h}{2}$$

  Where h is the average height of connected components in a sentence. We discovered after changing those values that 1.85 yielded optimal results.
- We also tried various values of the regularization parameter C. values less than 300 yielded an accuracy of 80%.

Additional Modules

Test Case Generator

A python script was written that traverses a folder called “forms”, (which has the combined data of formsA-D, formsE-H, and formsI-Z), and uses the information of forms.txt in the “ascii” folder to create folders with unique writers, so folder ‘000’ has only the writings of writer ‘000’ and so on. The functions read_forms_data() and order_data_set() do that.

Then after the folders are created according to the writers, the function filter() is run to filter which writers have at least 3 documents.

Finally, the function generate_random_test_cases(number) is used to create test cases in a folder where the input path is specified. This function also generates a text file with the correct value of the writer.

Disclaimer: The input paths are not optimized due to the time constraint. However, if this project is to be the same next year for the next class, or a similar one, we would like this python script to be included to make things easier.

Workload Distribution

Member	Tasks
Kamel Mohsen	Preprocessing, Image Rearrangement, classification, LBP calculation
Ahmad Nader	Image Rearrangement, Test Case generator
Samy Saeed	Classification, Test Case generator
Hazem Ayman	Preprocessing, Image Rearrangement, classification

The following is the github repo:

https://github.com/AhmadNaderAdel/Writer-Recognition

Please remember that the number of commits is not an accurate representation of contribution, especially because a lot of the code is the result of collaborations on online meetings where only one member is writing the code and the rest are brainstorming, planning and guiding.

Future Work and Enhancements

Due to time constraints, other features in the paper were not tried. Also, other approaches from other papers were not implemented. So, perhaps a combination of classifiers would yield better results, even though it might affect the time.