Writer Identification System

Submitted to:

Eng. Hussein Fadl

Submitted by:

TEAM 11

Ahmed Essam	SEC 1	BN	2
Philopateer Nabil	SEC 2	BN	4
Mazen Amr	SEC 2	BN	8
Mahmoud Ahmed Sebak	SEC 2	BN	18

Table of content

Introduction

Writer identification system proves very useful in a wide range of applications, such as the fields of security, biometrics, and forensics. Not limited to that, but it also raised interest in the analysis of historical texts, historical musical scores with unknown composers, forged signatures in official documents, and many other applications.

Project Pipeline

Our project is composed of three main modules: Preprocessing module, Feature extraction module and Training and Prediction module.

Briefly for each test case, we load the images of the three writers and do some preprocessing on them to extract the region of interest (handwritten text region) from each image, then we do line segmentation on the images to extract separated text lines.

Then we extract texture descriptor features from each line and train a support vector machine classifier on the extracted features.

To classify the test image, lines are extracted and a feature vector is calculated for each line. Then, we classify each line separately and take the majority voting.

Preprocessing Module

Paragraph Extraction

In order to Extract the paragraph, an intuitive way is to remove noise then recursive dilation but this algorithm would consume a lot of time and computations. But, we noticed horizontal bounding lines in the image. So, in order to detect these lines the following algorithm was used:

- Sobel edge detection is used to detect horizontal lines which gave better results than canny results which gave discontinuous horizontal lines.
- Followed by opening to remove noises in order to keep the lines the kernel used was a rectangle with width = 5 and height = 3.

- Hough algorithm was used to generate the lines from the image with width = 600 pixel to remove any misclassification of horizontal lines.
- The returned lines are sorted and two lines within a certain range are chosen as upper and lower bound.
- Upper line = max(lines within 400, 1100)
- Lower line = min(lines within 2500, 2850)

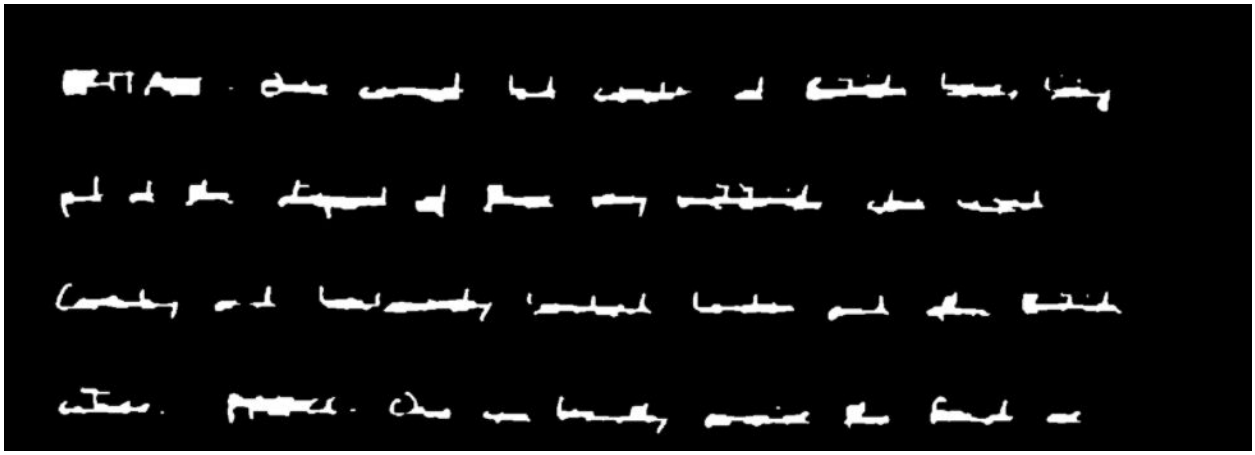
Line Segmentation

To segment the image into text lines we followed [1], steps as follows:

1) Binarization

   Our target in this step is to do some morphological operators on the image to highly detect each line.

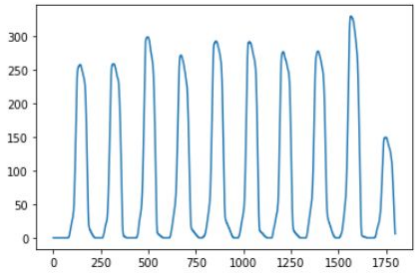
   The output of this step is like the image below:



2) Y Histogram Projection

   The idea is to use a simple and fast method to correctly distinguish possible line segments in the handwritten text.

So we get the Y histogram which describes the number of white pixels in each row, we also smooth the histogram to get rid of the peaks resulting from the upper and lower areas of text lines and make the peaks focused upon the middle of each line.



As shown in the figure above, each line corresponds to a peak in the histogram.

3) Text Line Separation

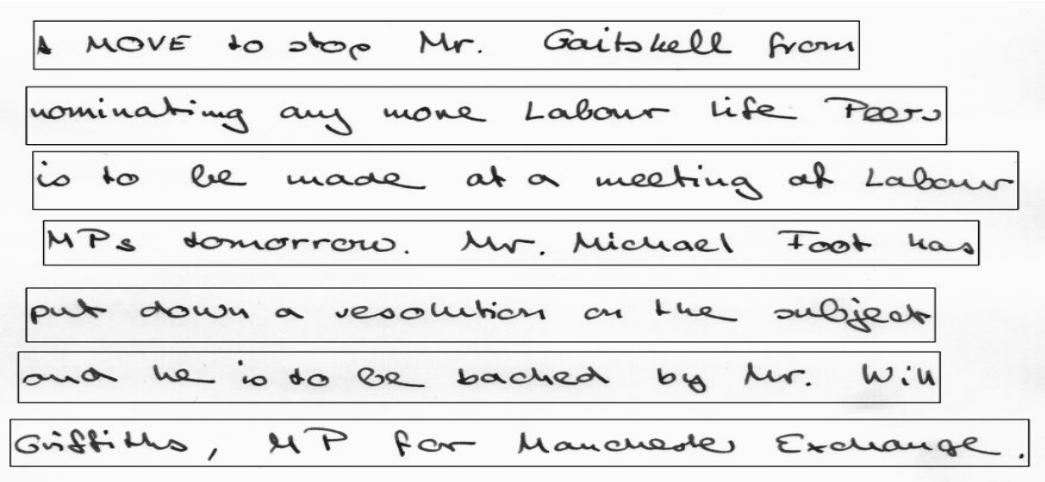
   From the Y histogram, we get the peaks and get the valleys between these peaks, then separate each two lines at the position of the valley (local minima) between them.

4) Removing false lines

   Due to some thresholding errors, false lines may be detected so we loop over the segmented lines and check if the number of white pixels is below a predefined threshold and discard this false line.

We evaluated this segmentation method on the IAM dataset and got an accuracy above 99.5%.

A sample segmented image is shown below,



Feature Extraction Module

We used LBP (Local Binary Pattern) as our feature extractor to differentiate between writers' handwriting. The LBP is done on every line in the given image to ensure that we capture the handwriting style. We noticed that each line contains a large white area (non text areas) which makes the histograms of different writers very close. To solve this we made a binary mask to remove the non text areas and focus only on the handwritten text.

Then for each masked image line we calculate the LBP for every pixel in that image as follows :

$$\mathrm{LBP} = \sum_{i=0}^{P-1} s(n_i - G_c)2^i$$

$$s(x) = \begin{cases} 1, & if\ x > 0 \\ 0, & otherwise \end{cases}$$

For every pixel, we take a circle around it with radius **R** and take a number of neighbouring pixels **N** along the circle border. Then, the value of each pixel is calculated as a sum of **s(x)** weighted by 2^i where **s(x)** takes the difference between the gray level values of the pixel and the neighbouring pixel and returns 1 if the difference is greater than 0 and 0 otherwise.

After constructing the LBP matrix for the line, Histogram is calculated and used as our feature vector for this line.
We used different combinations of the radius of the circle and the number of samples inside it in order to reach the best values. We found that increasing radius and samples will lead to good accuracy to a certain limit and is also inefficient in time. While decreasing radius and samples will lead to decrease in the accuracy as the feature won't be able to describe the writer style. So after, many trials we found that **Radius = 6** and **Number of samples = 8** gives us the best results.

Model Selection & Training Module

After we preprocessed the image and segmented it into lines and then extracted the LBP histogram for each line we train our model on each line individually.

But first we should select our model.

Model selection

We found that most research papers that used the LBP works on either SVM or KNN models. So, we ran many experiments to decide which is more relevant in our case.

To evaluate both of them we divided our data into training set and validation set, we trained each model on the training data and then evaluated it on the validation set

First we worked on a portion of the IAM dataset (159 writers), the following was the results:

Model	Accuracy
SVM, C = 0.5	93.67 %
SVM, C = 1	98.2 %
SVM, C = 2	98.98 %
SVM, C = 5	99.24 %
KNN, N = 3	97.2 %

As shown in the table above, SVM with regularization parameter C = 5 yields the best accuracy (99.24 %).

But to make sure more that SVM with C = 5 is better, **we used the whole dataset with 657 writers**, Training data contains 1002 images and validation data contains 537 images

Results are shown in the following table:

Model	Accuracy
SVM, C = 1	86 %
SVM, C = 3	92.2 %
SVM, C = 5	94.59 %
SVM, C = 10	94.78 %
KNN, N = 3	91.8 %

From the above results we selected our model to be SVM with regularization parameter = 5, and kernel function = RBF.

Training

We tried two different techniques,

1) Training on the LBP histogram of each line individually.
2) Training on the accumulated lbp histogram of all text lines.

We found that the first technique yields a much better accuracy.

Classification Module

Given the image we preprocess it and then extract the LBP histogram for each text line in the image.

We tried two different techniques,

1) Pass the LBP histogram of each line in the image to the model and predict the writer of each line individually. Then a majority voting is used to determine the most probable writer of the image.
2) Pass the accumulated histogram of all the text lines in the image to the model and predict its writer as a whole.

We found that predicting with majority line voting yields a much more accuracy. The second technique is only used if there is a tie in the lines voting.

Performance Analysis

We made many test cases with different combinations of writers. The results of these random test cases are shown in the following table.

Number of Test Cases	Accuracy		Average time per test case
100	100%	100%	5 sec
500	99.8%	100%	5 sec
1000	99.9%		5 sec

Other Trials

Based on the fact that descriptors gives high accuracy in the problem, we have tried using wavelet transform to describe the handwritten paragraph in form of frequency coefficients and get the histogram of the wavelets and use it as a feature but it failed to give higher performance than the local binary pattern.

We have also tried adding simple features as line height to add 1% accuracy but didn't work as planned and we didn't prefer to add an extra 1% accuracy in the cost of doubling the execution time by adding another computational exhaustive feature.

Future Work

We believe that we reached the maximum accuracy from texture based features and augmenting it with other features may result in a decrease in the accuracy or increase in the execution time as stated before. However, We can extend our work by trying advanced features as codebooks that focus on extracting features from words rather than lines . We can also use deep learning techniques as Convolutional Neural Networks (CNN) which gave state of the art accuracy in similar applications.

Distributed Workload

Ahmed Essam	- Line segmentation - Model selection
Philopateer Nabil	- Paragraph Extraction. - Wavelet attempt.
Mazen Amr	- Feature Extraction - Wavelet attempt
Mahmoud Ahmed	- Feature Extraction - Model training and prediction