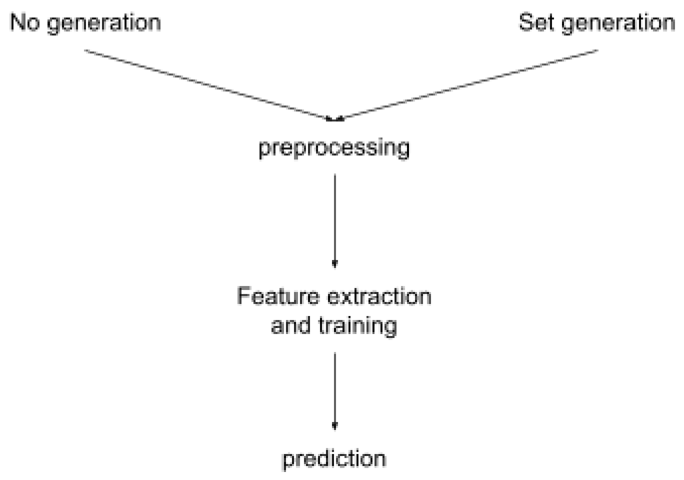
Pattern Recognition
Writer Identification System

Project Pipeline

The project has three main functionalities (preprocessing, feature extraction and training, testing).
The pipeline starts with the preprocessing where the photos get cropped to get rid of the text above and extract only the hand written text, and divided into lines, each line will be in separate photo and will be processed individually.
Then we will go through feature extraction where we will apply LBP and get the histogram of each line, then we will label the data for each writer then train the classifier using SVM.
Finally we get the test photo and apply the preprocessing in it and predict the writer for this photo using the trained classifier.
Also there is data generation from a data set for handwritten photos. We downloaded it and made code that generates sets in the same form of the provided test data for better testing for the project, however, this part is optional to run but it is in the project.



How To Run

Repo link https://github.com/shadyfahmy/writer-identification-system
Install requirements.txt using pip install. And run main.py with the following arguments:

Argument	Type	Default value	Explanation
-n (Number of tests)	Integer	-1 (means that the input will be the number of test in data folder)	Number of test cases in test folder needed to be testet
-g (Generate N tests)	Bool (true, false)	false	Whether to generate new test cases in test folder
-a (Accuracy	Integer 1-3 (1,2,3)	3	To determine level of accuracy used

Example:

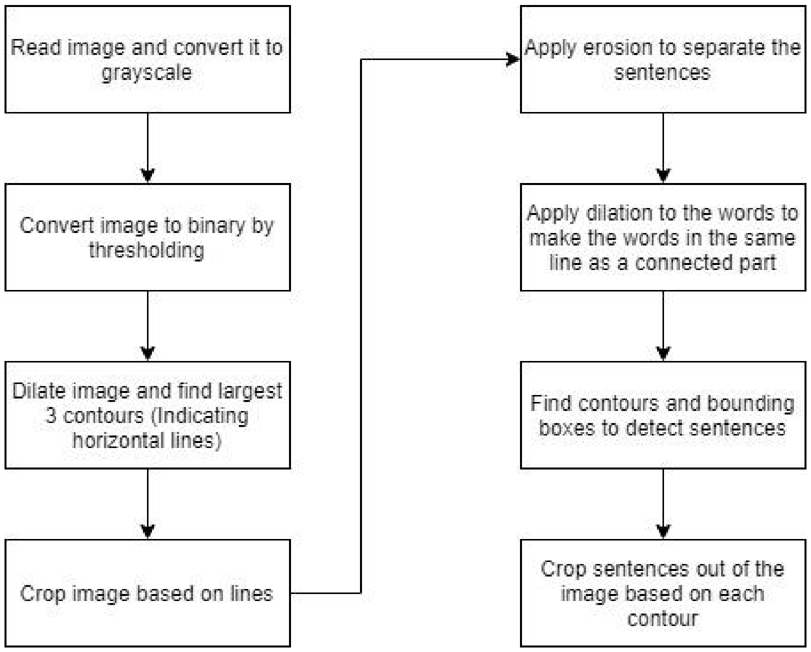
Generate 100 test
```
python3 main.py -n 100 -g true
```

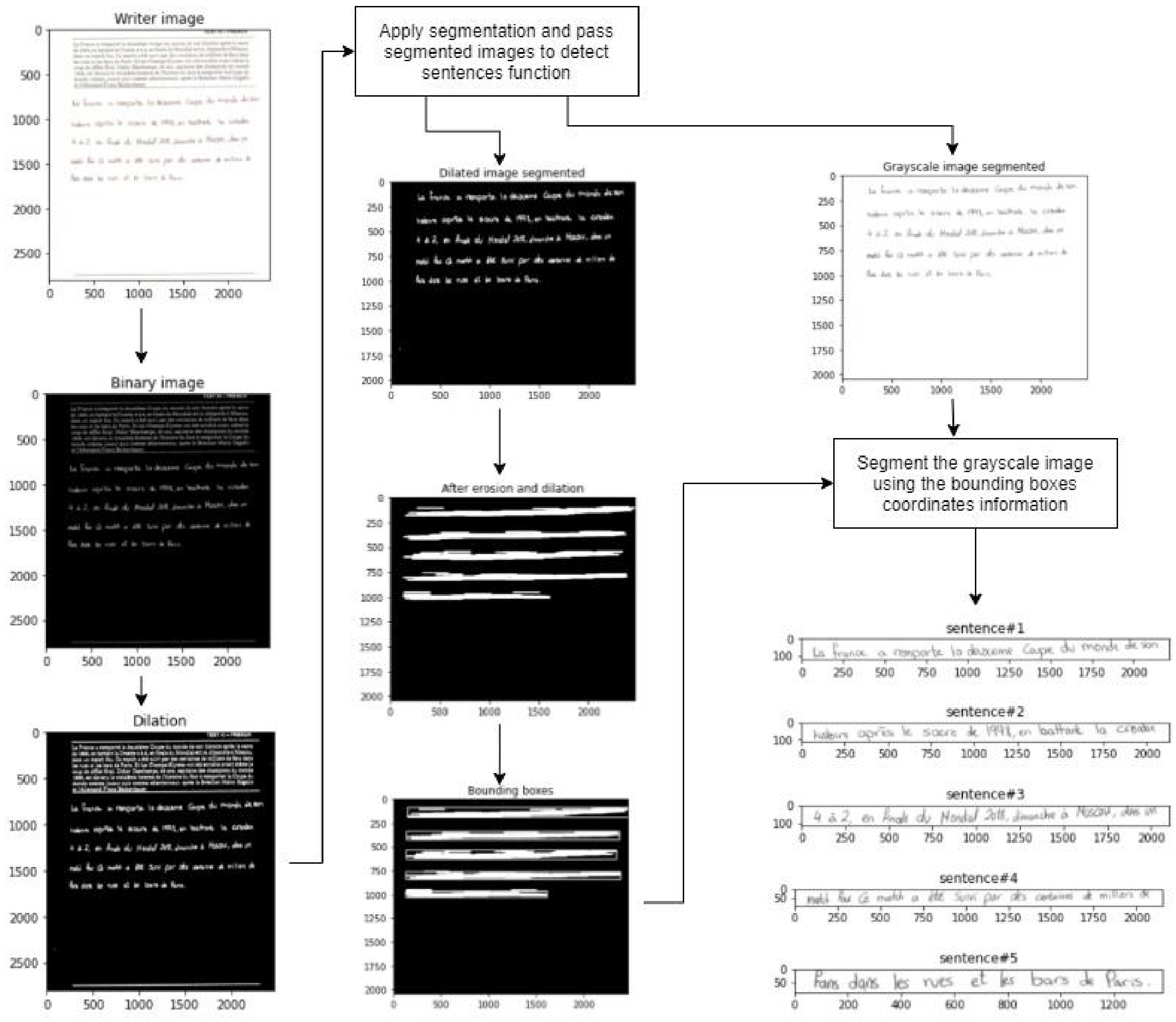
Run 100 test
```
python3 main.py -n 100
```

Run default
```
python3 main.py
```

Preprocessing Module

We divided this module into 2 parts, firstly we extracted the handwritten paragraph by detecting the longest 3 contours which indicates 3 lines found in the dataset images, Then cropping the image from the second line to the third.
Then we pass the cropped image to the detect sentences function which segments the paragraph into sentences each sentence in a different image, in the following flow chart the detect sentences function in details.
in the following flow charts we explain the extract handwritten function followed by detect sentences function in more details:


Read image and convert it to grayscale
Convert image to binary by thresholding
Dilate image and find largest 3 contours (Indicating horizontal lines)
Crop image based on lines
Apply erosion to separate the sentences
Apply dilation to the words to make the words in the same line as a connected part
Find contours and bounding boxes to detect sentences
Crop sentences out of the image based on each contour


Writer image
Binary image
Dilation
Apply segmentation and pass segmented images to detect sentences function
Dilated image segmented
After erosion and dilation
Bounding boxes
Grayscale image segmented
Segment the grayscale image using the bounding boxes coordinates information
sentence#1
sentence#2
sentence#3
sentence#4
sentence#5

Feature Extraction/Selection Module

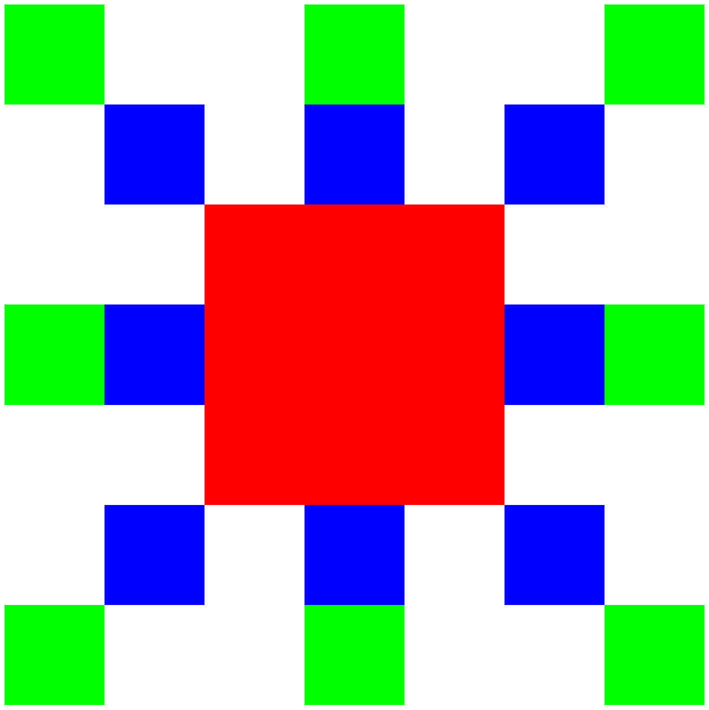
In feature extraction we use LBP we apply it on the lines photos to extract the histogram from the lines photos.
For applying the LBP we should specify the number of points (P) and the radius (R) we used P = 4 and R = 3, but for simplicity we will say P = 8 and R = 1, in this case the LPB will consider the neighbour cells that one pixel away (R) and will take eight of them that is the P, and look for each neighbour if that neighbour greater than the center cell will be assigned to 1 other than that will be assigned to 0. Then we calculate the binary number we have and assign its value to the center cell.
After that we get the histogram of the image and label that histogram with the number of the writer.
We used 3 implementations to calculate LBP. Skimage library, a for loop, and array indexing. Here’s a comparison between the three

	Skimage LBP	For Loop LBP	Array Indexing
Avg total test time	1.8 secs	74 secs	0.34 seconds

Indexing is much faster as we could see, the idea to use array indexing was inspired by OmarBazaraa's writer identification.

Different Accuracies

We implemented different accuracies by changing the pixels that we decide the LBP based on.

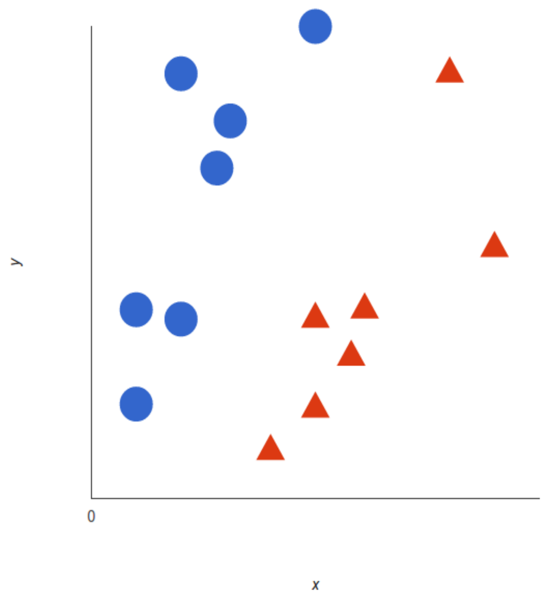


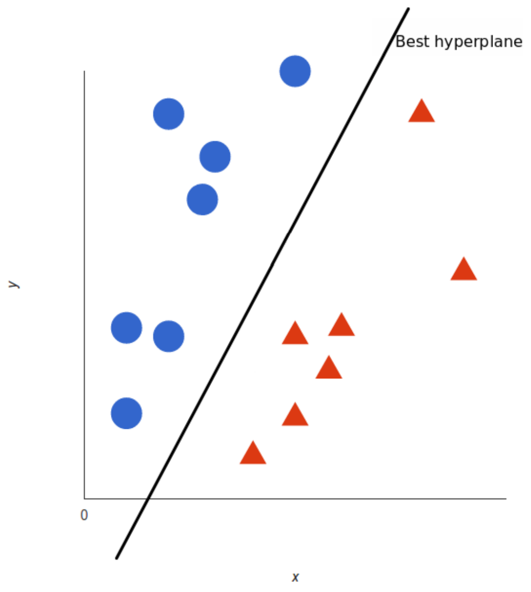
Accuracy = 1 calculates the LBP from the green pixels only, (R = 3, P = 8)
Accuracy = 2 calculates the LBP from the green and blue pixels only, (R = 3,2, P = 16)
Accuracy = 3 calculates the LBP from the green, blue and red pixels (R = 3,2,1, P = 24)

Accuracy Level	Average Total Runtime	Accuracy
1	0.34 seconds	99%
2	0.37 seconds	99.4%
3	0.42 seconds	99.75%

Model Selection/Training Module

We use SVM (support vector machine) as our training model, using the histogram data and the labels the SVM trains the data and classifies it.
SVM tries to get the best hyperplanes that separate all three data labels so it can predict a new photo based on that hyperplane.
After that we have our model trained and ready for prediction based on the training.





Data and labels

hyperplane separate the data

Performance Analysis Module

We tested our model on 500 random test cases generated by the test cases generation module. To measure accuracy we compare the classifier's output with the writers stored by the test cases generation module. We also measure the execution time for each test case and calculate the average time for the 500 test cases.

Test cases generation module

We developed a module to generate N test cases where N is a given parameter. Each test case has 3 different writers where we have 2 different samples for each writer and a test image by one of the three writers and it can't be one of the 6 chosen samples. The test cases are generated randomly by picking writers from the IAM dataset and randomly picking images written by each of the chosen writers. We store the writer’s number of the test image in each test case to be used in the performance analysis module.

Enhancements and Future work

This project could be enhanced by adding more optimization to the implementation (e.g. implementing SVM in a more efficient and fast way, faster implementation for the LBP). Also more features could be added to classify the photos along the LBP so we can get more accuracy, however, this feature should increase the performance and/or the accuracy of the project and do not have an opposite effect to one of them.