Pattern recognition Report

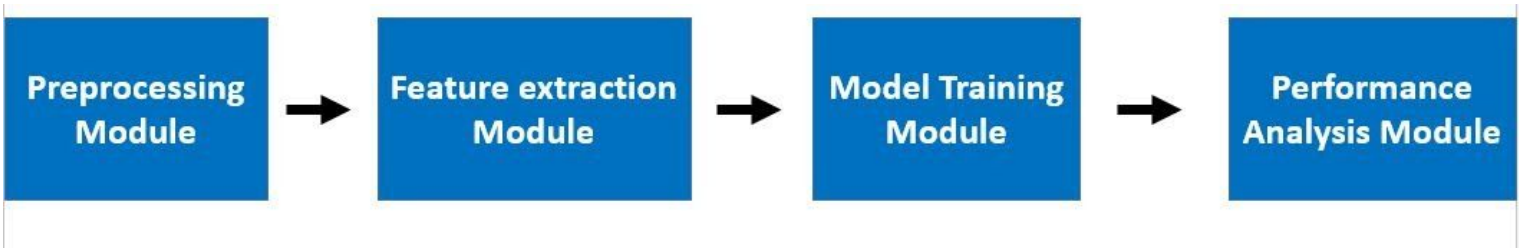
Writer Identification System

Team Members:

Sarah Mohamed Ahmed Lotfy	Sec. 1 Bn.23
Salma Mohamed Ali	Sec. 1 Bn.27
Kareem Omar	Sec. 2 Bn.6
Nourhan Gamal	Sec. 2 Bn.31

Date: 19 January, 2021

(a) Project Pipeline



The pilpline of our system is that we take the image then pass it to preprocessing module then take the preprocessed image and pass it the feature extraction module to get the features by using local binary patterns then pass these features to k-Nearest Neighbors KNN classifier to tell us who is the writer of the image.

(b) Preprocessing Module

In the preprocessing module we crop the handwritten region in image and split the cropped image to separated written lines images.

First: Crop the handwritten region

The IAM images composed of three parts separated by three black lines, the handwritten region is between line 2 and line 3,

Detect the coordinates of three black lines, crop and erode:

1. Convert img to binary image by OpenCV Otsu's thresholding.
2. Get the contours of the image by OpenCv findContours() function.
3. Loop on these contours and get the bounding rectangle of each contour by OpenCv boundingRect(contour).
4. Check if the width of the bounding rect is bigger than 1000 then check of height less than 500, so it is a line.
5. Get the coordinates x and y of the bounding rect of the detected line and put them in an array.
6. Sort the array containing y's coordinates in ascending order.
7. Get the x,y coordinates of the second and third lines then crop the image by "bin_img[newCooriate_y1+4:newCooriate_y2 , :]"
8. Make erosion for binary cropped image to reduce noise by cv2.erode(img, kernel, iterations=2) with ones kernel size 5*5 .

Second: Split cropped image to separated written lines images

We made this part because we tried to pass the cropped image to the feature extraction module but we got low accuracy but when we passed the split lines images we got higher accuracy.

Detect the written lines in cropped image then split it :

1. Get the sum of black pixels of each row in binary cropped image by “np.sum(img < 255, axis=1)”
2. Loop on these rows
3. Check if number of black pixels of row > 15
4. We want to get the up and down coordinates of each written line so if row - 6 < 0 so we put up coordinate =0 else up = row - 6
5. Then loop again with row number as the number of black pixels >15 and increment the row number. We made this to know the down coordinate of the written line.
6. Then crop the written line and put it in an array and continue the loop.

Note all these threshold numbers we get by experiment.

(c) Feature extraction module

In this part, we apply local binary pattern (LBP) to extract features from the images. After the image is passed through the preprocessing stage, we take the cropped gray image that resulted from the preprocessing module and apply LBP on it.

We use this formula:

$$LBP = \Sigma_i^{p-1} s(n_i - G_c)2^i$$

$$s(x) = 1,\ if\ x > 0,\ s(x) = 0,\ otherwise$$

We tried first the radius to be 1 (R = 1) and P = 8. Then we found it better to use R = 3 and P = 8. It gives us more accurate results.

After that, we get the result of applying LBP on the gray scale cropped image, and then we get the histogram for each intensity of the 256 of the gray.

Then we normalize the vector of the histogram, by dividing each value in the histogram over the mean of the histogram.

$$normalizedHistogram = histogram / mean(histogram)$$

That we have a vector of features of size (256x1).

(d) Model Training Module

For training model first we separated the image into written lines images then extracted the feature for each line and used the k-Nearest Neighbors Knn with n=5 to train the model using the lines features, as for testing separate each image to lines also and predicted each line then the result would be the most frequent class.

(e) Performance Analysis Module

In the Performance Analysis module extract the output into a list and print it out in a txt file for comparison with the correct answer. We also calculate the time for each test.png from retrieval until prediction, add all times to a list and print it out in a txt file.

Both txt files are located in the same path as the test cases and are named results.txt and time.txt respectively.

(f) Test generation

All tests are generated from the IAM data set using a folder containing all images from the dataset and another folder containing all the xml files corresponding to each image.

The generator separates the images by author and removes all unidentified authors (using id = ‘000’) and authors with less forms than 3 (2 for the training set and 1 for the test).

The generator then selects 3 random distinct authors and creates a test case as specified in the example attached to the project document.

The test is chosen randomly from the 3 authors and reserved for an answer list which is printed in a txt file located in the same folder as the test cases named answers.txt

Trials

We tried to pass cropped handwritten region image only to local binary patterns but it made accuracy very low so we pass a split separated written line images to local binary patterns that made accuracy higher.

We tested 17 test cases with KNN n=1 it gives same results as KNN n=5 so we make it use KNN n=5.

We tried to increase accuracy by making erosion to the binary cropped handwritten region image to remove noise and it gives higher accuracy than before. We tested this on **17 test case** and the result:

Without erosion -> accuracy = 82.3%

With erosion -> accuracy = 88.2 %.

We tried SVM as a classifier but it gives lower accuracy than KNN classifier.We tested this on **40 test cases** and the result:

SVM -> accuracy = 82.3 %

KNN with n =5 -> accuracy = 90%

We tried to resize the input image to reduce run time but it failed with us.

Last Successful trial

Make erosion for cropped handwritten images then split it to written lines images then pass it to local binary patterns to get features then pass features to KNN classifier with n=5.

We tested this on 40 test cases and we got accuracy = 90%

Link of 40 test cases:

https://drive.google.com/file/d/1ay27biVlsuU9kb7ZstaLQDBzfqbS6Nbn/view?usp=sharing

Enhancements and Future work

Will work on reducing the run time. We can try to resize the image.

We could consider a way to enhance the performance in our code is to use another feature in addition to the one we use. It would give us more accurate results.

We also want to enhance the time for our code to get the results. We could make a better use for Python libraries, or we could use OOP that would give us a better time and more organized code.

Or instead, we could change our approach completely. We could use neural networks. It would give us a much better accuracy.

Work load

Name	Work
Sarah Mohamed Ahmed Lotfy	Preprocessing module
Salma Mohamed Ali	Feature extraction module
Nourhan Gamal	Model Training Module
Kareem Omar	Performance Analysis Module / Test generation