

Final Report Submission

Pattern Project

Team ID:

Submitted to: Eng. Hussein Fadl

Submitted by:

Name: Ehsan Sayed Email: ehsansayed2019@gmail.com  
Name: Reham Abdellatif Email: reham.abdallatef47@gmail.com  
Name: Sara Mohamed Email: sarahasanien932@gmail.com  
Name: Omar Hashim Email: omarhashim94@gmail.com

Table of Contents

1. Project Overview
2. Project Pipeline
   1. Writers Identification Using Text Line based Features
   2. Local Binary Patterns (LBP)
3. Our Attempts
   1. Writers Identification Using Text Line based Features
      1. Preprocessing
      2. Feature Extraction for each Line
      3. Classifier
   2. Local Binary Patterns (LBP) – Final
      1. Preprocessing
      2. Feature Extraction for each Line
      3. Classifier
4. Performance Analysis Module
   1. Writers Identification Using Text Line based Features
   2. Local Binary Patterns (LBP)
5. Enhancements and Future Works
6. References
7. Workload
8. Project Overview

We start implementing this project using a paper called “Writer Identification Using Text Line Based Features”. The system presented in the paper consists of three parts: preprocessing, feature extraction and classification. This paper does not achieve our expected accuracy as we test 63 test cases and only 40 cases passed successfully. Consequently, we turned into another algorithm which called Local Binary Patterns (LBP) and this algorithm achieve 96% accuracy on 100 test cases.

1. Project Pipeline

2.1 Writers Identification Using Text Line based Feature

The implementation was divided into three files:

1. Cropping.py file which contains one function “cropping(image)”
2. OldFeatures.py file which contains how we extract the features:

* SplitLines(image):
* ExtractF1toF6(image):
* ExtractF7AndF8(image, f2):
* ExtractF9AndF10(image):
* ExtractF11(image):

getScore(image):

getSlant(image):

3- Main.py file which contains training and testing modules.

2.2 Local Binary Patterns (LBP)

The implementation was divided into three files:

1. Cropping.py file which contains one function “cropping(image)”
2. LBP.py file which contains our extracted features(255)“lbp(image)”
3. Main.py which contains training and testing modules.
4. Used Modules
   1. Writers Identification Using Text Line Based Features

As we mentioned before we start using this paper and the work was divided into three parts:

* + 1. Preprocessing

In this phase, we split the image into lines as follows: first, the image is binarized, second dilation is used to improve the segmented result and finally, contours are used to split the lines.

* + 1. Feature Extraction from Each Line

This paper works depending on twelve features but we implemented only 11 features which are:

* Group One (6 Features): The first features are based on the height of the three main writing zones, which are determined by the topline, the upper baseline, the lower baseline, and the bottom-line.

F1 = |Ytop line – Ybaseline |

F2 = |Yupper baseline – Ylower baseline |

F3 = |Ylower baseline – Ybottom line |

But One disadvantage of these quantities is that they represent absolute values for the height of the writing. So, the next three features represented the ratios between them.

F4 = F1 / F2

F5 = F1 / F3

F6 = F2 / F3

* Second Group: this group relates to the width of the writing.

F7 = median (d1, d2,.,dn) where d represents the distance between black pixels

F8 = F2 / F7 (to avoid the same problem mentioned above)

* The next feature describes the characteristics of the slant of the writing but this feature was very slow and increase our time significantly.
* The last two features are based on fractal geometry. In a first step the writing is dilated by various disks of radius n. Then the area A(Xn) of the dilated writing X, is measured. The fractal dimension D (Xn) is defined and then we plot the behavior of y = In A(Xn) - In(n) over x = ln (n).

Then we approximate a graph by three straight lines. The slope of the second- and third-line segment give the two fractal numbers which are used as features in the system:

F10 = slope (line 2)

F11 = slope (line 3)

* + 1. Classifier

We started using SVM model and the results was horrible, none of the test cases was passed successfully. Second, we used KNN and the results were improved a little bit but not our expected output (40 passed out of 60).

* 1. Local Binary Patterns (LBP)

Then we turned into this algorithm to improve the accuracy, three main parts also:

* + 1. Preprocessing

In this phase, only crop the first part in the paper which contains printed text.

* + 1. Feature Extraction from Each Line

The LBP feature vector, in its simplest form, is created in the following manner:

1. Divide the examined window into cells (e.g. 16x16 pixels for each cell).
2. For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
3. Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
4. Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.
5. Normalize the histogram.
6. Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.
   * 1. Classifier

We used SVM (built in function) and NN (we implemented this classifier from scratch) and the results were the same approximately.

1. Performance Analysis Module

|  |  |  |  |
| --- | --- | --- | --- |
| Module Name | Number of Test | Accuracy | Time |
| Writers Identification Using Text Line Based Features | 65 | 75% | Time per Image: 25 sec  Total Time : |
| Local Binary Patterns (LBP) | 100 | 98.5% | Time per Image:16.5 sec  Total Time: 27.5 min |

1. Enhancement and Future Work

In this section we will talk about LBP only.

We started implementing LBP from scratch but we found that the time was very large ,90 seconds per image to output the result, approximately. Then we used LBP which implemented in Python itself but we added some enhancements. Generally, LBP takes the image as an input (all pixels) but that will output wrong results. Our enhancement is passing to LBP important pixels from the image not the whole image (text pixels). This enhancement improves the accuracy greatly and reduce the time as well compare to the previous implemented LBP.

1. References

* Local Binary Patterns: <https://en.wikipedia.org/wiki/Local_binary_patterns>
* Writer Identification Text Line using based features Paper:

<https://drive.google.com/file/d/1qql4Zz6Gxb8QEa8Bn8_H5iLJfvgechQm/view?usp=sharing>

1. Workload

|  |  |
| --- | --- |
| Name | Activities |
| Ehsan Sayed | Writers Identification Using Text Line Based Features   * Feature 7 and 8 * Classifier KNN |
| Reham Abdallatef | Writers Identification Using Text Line Based Features   * Feature 10 and 11 * Preprocessing (Split lines, Cropping image)   LBP   * Encasement * SVM model |
| Sara Mohamed | Writers Identification Using Text Line Based Features   * Feature 1 to 6 |
| Omar Hashim | Writers Identification Using Text Line Based Features   * Feature 9   LBP   * Implement LBP algorithm * NN model implementation * Encasement |