Handwritten Signature Verifier

A Project Synopsis

Submitted by

Tejas Jadhav

Under The Guidance Of

Prof. Abhay Kolhe

in partial fulfillment for the award
of the degree of

M. TECH.
in
COMPUTER ENGINEERING
at

SVKM's NMIMS,
Mukesh Patel School of Technology Management & Engineering,
Mumbai

2018 - 19

CERTIFICATE

This is to certify that the project synopsis entitled “Handwritten Signature Verifier” is the proposed work by Tejas Jadhav of M. Tech. (Computer Engineering), MPSTME (NMIMS), Mumbai, during the III/IV semester of the academic year 2018 - 2019 is verified by me.

The presentation for the same is also verified by me.

Prof. Abhay Kolhe	Tejas Jadhav
Internal Mentor	Project designer

Table of Contents

1. Introduction

Biometrics is currently perceived as a vital technology for setting up secure access control. It utilizes physiological qualities of humans for recognizing individual, and signature is one of the characteristics usable for biometrics. Singular recognizable proof technology utilizing human countenances, more often than not called confront acknowledgment technology, has been concentrated primarily in western nations since 1990s. It has been the most widespread tool for paper documents verification for many decades. In real life the human-expert can verify handwritten signatures easily, but it is a complicated task for doing by machines today.

1.1 Real life applications

This application of Image processing has number of uses in the real world and are seen to be very critical in many industries. Some of them are as follows:

1. Finance: IT-Processing firms of German savings banks are offering their customers solutions to embed dynamic signatures securely into electronic documents in an Adobe Live Cycle environment
2. Insurance: Signing an insurance contract and documenting the consulting process that is required by EU legislation have caused several insurance companies to go paperless with either signature capturing tablets connected to a notebook or a tablet PC.
3. Real Estate: Increasingly popular among real estate agents in USA are options of paperless contracting through signing on Tablet PCs.
4. Health: The hospital of Ingolstadt is capturing and verifying the signatures of their doctors that fill electronic patient records on tablet PCs. The "National Health Service" organization in the UK has started such an implementation.
5. Telecom: Signing phone and DSL contracts in the telecom shops is another emerging market.

1.2 Types & methods of Signature Verification

There are two types of signature verification methods:

1. Offline Signature Verification Methods:

In this method, system makes use of a simple scanner or a camera that can take in image having signature & process the image further with whatever features it gets. This method can be seen useful in many applications such as banking cheques, medical certificates & prescriptions etc.

2. Online Signature Verification Methods:

Here the system makes use of special acquisition devices like stylus pens and tablets that can take in signature features in real time and may give better accuracy

1.3 Performance evaluation parameters

1. FAR – False Acceptance Rate :

The false acceptance rate in simple words is rate of falsely accepted signatures. This is given by the following formula:

$$FAR = \frac{\text{Total Number Imposter Signatures Accepted as Genuine}}{\text{Total Number of Forgery Tests Performed}}$$

2. FRR – False Recognition Rate : Rate of falsely rejected signatures

The false acceptance rate in simple words is rate of falsely rejected signatures. This is given by the following formula:

$$FRR = \frac{\text{Total Number Genuine Signatures Rejected as Imposter}}{\text{Total Number of Genuine Matching Tests Performed}}$$

3. EER – Equal Error Rate

The Equal Error Rate (EER) corresponds to the error value for which false acceptance rate is equal to false rejection rate. These rates determine the quality of an authentication system, but the acceptable values depend on the level of security desired for a specific application.

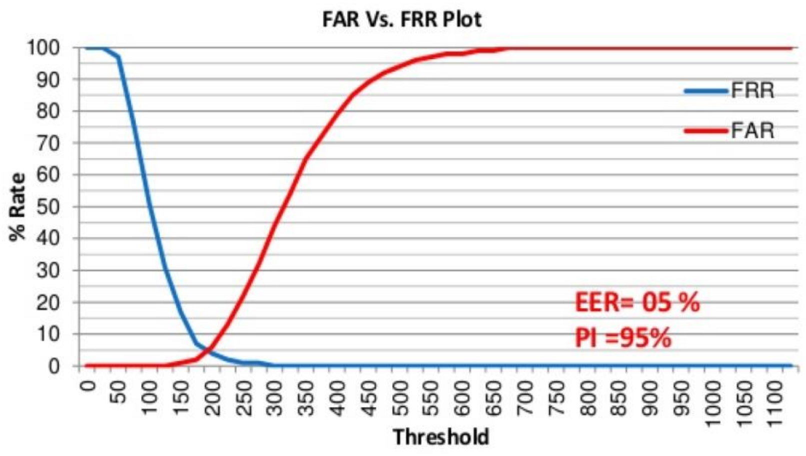


Figure 1: Performance evaluation graph

4. PI – Performance Index

Performance index is nothing but the opposite of the Equal error rate. This is calculated easily by the following formula:

PI = 100 – EER

2. Motivation

Today the human signature of a person is used as an identification of person because we are all know that the each person has distinct signature and every signature has its own physiology or behavioral characteristics. So the human signature used as an identification of person in various work like bank checks etc. The fraud person can easily generated the signature instead of unique signer in fraud way so we need a signature identification system

Nowadays, person identification is very important in security and resource access control. A reliable signature recognition system can be seen as an important part of law enforcement, finance & banking and many business processes. In such applications, an accurate recognition of person through his handwritten signature is critical.

Thus the motivation behind this project is the growing need for a full proof signature verification scheme which can guarantee maximum possible security from fake signatures. The idea behind the project is also to ensure that the proposed scheme can provide comparable and if possible better performance than already established offline signature verification schemes. The prospect of minimizing the memory space for storing signature image by the preprocessing extracted feature and the training is completed in acquiring less time and provide better accuracy. The need to make sure that only the right people are authorized to access high-security systems has paved the way for the development of systems for automatic personal authentication

3. Problem definition

Signature Recognition is the procedure of determining to whom a particular signature belongs to, as said earlier. This problem is a huge research topic under the Image Processing domain. But this problem is also sometimes crossed with Machine Learning domain along with the Image Processing domain. This is not as simple as said it seems. It is just easier said than done.

Thus the system to be designed must have an algorithm that will extract features and recognize and verify the signer’s authentication. System would take as input signature images and tell us following things: If the signature is forged or genuine (Signature Verification)

4. Literature review

This section lists down the various algorithms that we are to study. Most of them work in 3 stages, which are pre-processing, main processing i.e. feature extraction and then post-processing stage which includes decision making or classification. There are a high number of algorithms that can be implemented for recognition of handwritten signatures these days, but following are the algorithms which give other newly formulated algorithms a backbone.

In paper [1] the researchers are considering some simple parameters like speed, acceleration, pen down time, distance, etc. and based on the literature studies they discuss how these features can be used in the Image processing environment to improve the performance of handwritten signature. The approach provides a PI of 97.5 %.

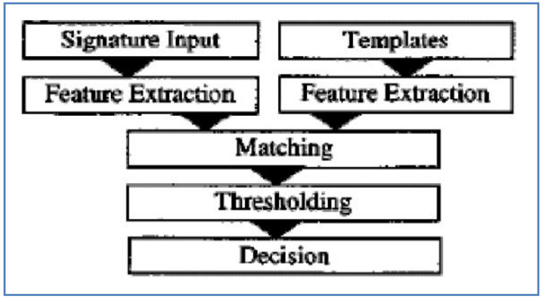


Figure 2: Flowchart of paper [1]

Features extracted and used for comparison include total Euclidian distance D of the pen travelled, Speed Vx and Vy express the functions of time, Acceleration Ax and Ay, Total time taken Tk, Length-to-width ratio, Amount of zero speed in direction *x* and *y* directions *Nvx* and *Nvy* , Amount of zero acceleration in direction *x* and *y* directions *Nax* and *Nay*

The paper [2] presents a crowdsourcing experiment to establish the human baseline performance for signature recognition tasks and a novel attribute-based semi-automatic signature verification system inspired in FDE analysis. It combines the DTW algorithm with Attribute based algorithm to obtain better accuracy and increase the recognition rate. With EER of 5%, we can say that recognition rate is 95%. Attribute based recognition features include Shape, Proportionality, Text-loops, Order, Punctuation, Flourish-characteristics, Hesitation, Alignment to the baseline, Slant of the strokes, Strokes-length, Character spacing

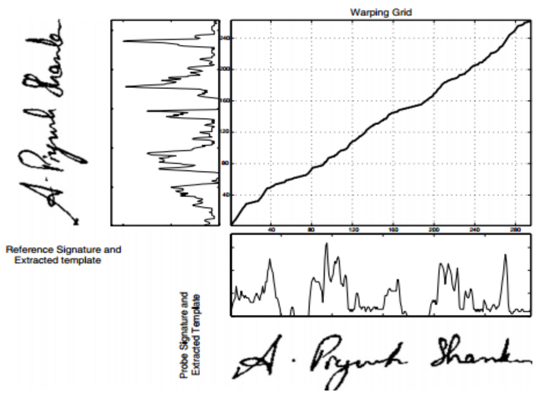


Figure 3: Dynamic Time warping

The dynamic time warping algorithm finds an optimal match between two sequences of feature vectors which allows for stretched and compressed sections of the sequence.

The proposed system in paper [3] functions in three stages. Pre-processing stage; this consists of grey scale conversion, binarisation and fitting boundary box. Feature extraction stage where total 16 radon transform based projection features are extracted which are used to distinguish the different signatures. Finally in the classification stage; an efficient BPNN, Back Propagation Neural Network is prepared and trained with 16 extracted features. The average recognition accuracy of this system ranges from 87% - 97% with the training set of 10–40 persons. Global features include_Height of the signature, Width of the signature, and Centroid along both X axis, Centroid along both Y axis

The radon transform derives projections of the image matrix along predefined directions. A projection of a two dimensional function f(x,y) is a group of line integrals. The radon transform calculates the line integrals from multiple sources along parallel paths, or beams, in a predefined direction. The beams are spaced one pixel unit apart. To represent an image, the radon transform takes multiple, parallel-beam projections of the image from different angles by rotating the source around the centre of the image. Radon transform of a signature image I for the angles specified in the vector 'theta' is computed using the Matlab function radon().

It returns a vector 'r' for each angle in theta containing the Radon Transform. Mean and standard deviation of all the vectors 'r' are calculated and taken as a local features.

The number of neurons in the first layer is n (n=16 in this work) which is equal to the dimensionality of the input pattern vectors (Number of input nodes equals number of input features used). The number of neurons in the output layer is 5 which are equal to the number of pattern classes.

Table 1: Local radon transform features

1	Height	9	Mean90
2	Width	10	Std90
3	Centroid of X axis	11	Mean120
4	Centroid of Y axis	12	Std120
5	Mean30	13	Mean150
6	Std30	14	Std150
7	Mean60	15	Mean180
8	Std60	16	Std180

The developed BPNN is trained with signature from different persons. Large numbers of images are required to ensure proper training of the NN.

In paper [4], researchers propose a new on-line writer authentication system using the pen altitude, pen azimuth, shape of signature, and writing pressure in real time. It is well expected that altitude and the azimuth of gripped pen under signing depends on the shape of writer's hand and the habit of writing. After individuality in the altitude and the azimuth of gripped pen under signing is explained, the experimental result with writing information by 24 writers is shown. It is found that the authentication rate of 98.2% is obtained.

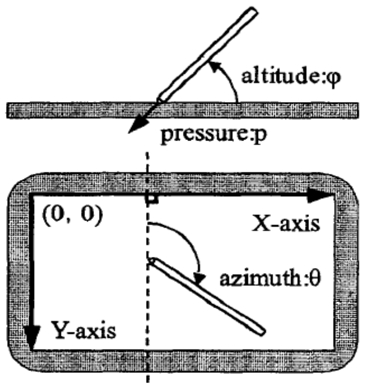


Figure 4: Data from tablet and pen.

The data set prepared is based on the following Features to be calculated dynamically at every instance of time: X-coordinate: *x(t)*, Y-coordinate: *y(t)*, Pressure: *p(t)*, Azimuth: $\theta(t)$, Altitude: $\phi(t)$

In paper [5], researchers propose a new on-line writer authentication system that uses the pen azimuth, point positions of signature, and writing pressure in real time for creating 7 fuzzy characteristics. The stability of the altitude and the azimuth under signing is also compared with the bit-mapped data and the pen pressure along time for one writer.

Researchers used collection of signatures for testing this system. Signature verification experiment has been conducted with 100 users across 25 original and 25 fake signatures for each user. The recognition rate shown was 99.8%. A 28-dimensional feature vector *for* the signature is formed.

In this paper [6] researchers present an off-line signature verification and recognition system using the global, directional and grid features of signatures. Support Vector Machine (SVM) was used to classify & verify the signatures and a classification ratio of 0.95 was obtained. Thus 95% accuracy is obtained as the recognition of signatures represents a multiclass problem SVM's one-against-all method was used.

The system researchers introduced is divided into two major sections: (i) Training signatures, (ii) Verification or recognition of given signature.
Features used were Signature area, Signature height-to-width ratio, Maximum horizontal histogram and maximum horizontal histogram, Maximum vertical histogram and maximum vertical histogram, Horizontal and vertical center of the signature are calculated using the formulas

In this system, a multi class system is constructed by combining two class SVMs, radial basis function is used which gave the best results. In the training phase, for each person 8 positive (genuine) and 82 (39 x 2 + 4) negative (forgery) examples are taken.

The proposed system of paper [7] functions in three stages. Pre-processing stage consists of four steps namely gray scale conversion, binarization, thinning and fitting boundary box process in order to make signatures ready for the important feature extraction. Feature extraction stage consists total 59 global and local wavelet based energy features to be extracted which are used to classify the different signatures. Finally in classification, a simple Euclidean distance classifier is used as decision tool. The average recognition accuracy obtained using this model ranges from 90% to 100% with the training set of images of 10 to 30 persons.

The proposed method in paper [8] is based on the hypothesis; reducing the variability of signatures leads to boost up the recognition rate. Therefore, the variance reduction technique is applied to normalize offline handwritten signatures by means of an adaptive dilation operator. Then the variability of signatures is analyzed in terms of coefficient of variation (CV). The optimal CV is obtained and used as a threshold limit value to be acceptable variance reduction. The average recognition rate after experimental analysis is found to be 94.87%.

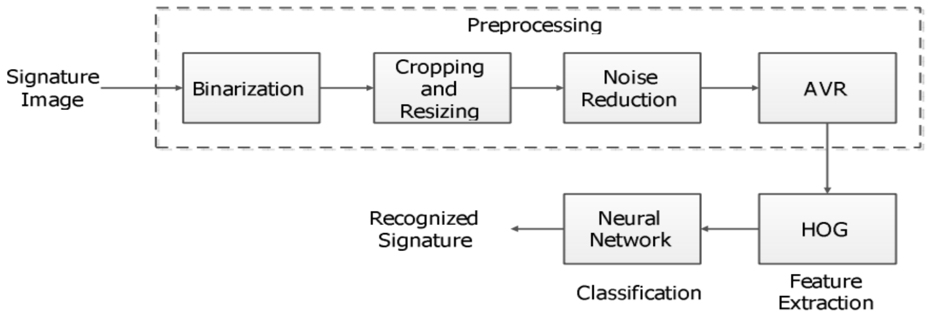


Figure 5: Flowchart for paper [8]

The paper [9] proposed a SIFT and a SURF algorithm which is used for enhanced offline signature recognition. This process, Bag-of-word features, was operated by making vector quantization technique, which outlined the key points for each training image inside a unified dimensional histogram. They put features of bag-of-word inside multiclass Support Vector Machine (SVM) classifier established upon the radial basis function (RBF) for a training and testing. They used Open CV C++ as an image processing tool and tool for feature extraction. In this paper, we compare the performance of SIFT on SVM based RBF kernel with SURF on SVM based RBF kernel. It was found out that the use of SIFT with SVM-RBF kernel system, it has an accuracy of 98.75% compared that of SURF with SVM-RBF kernel it has an accuracy of 96.25%. The SURF algorithm (speeded up robust transform) is composed mainly two parts. In the first part, locate the interest point in the image. The surf features are calculated and points are matched between the input and out signatures. The surf features are computed and points are matched across the input and out signatures. SURF detectors are looked in the significant points in the image, and descriptors are used to get the feature from vectors at each interest point just as in the SIFT. Hessian-matrix approximation consists of SURF to put through and locate the key points rather than variant aspects of the Gaussians (DOG) filter prepared in SIFT. This algorithm is very much similar to SIFT algorithm. But, it is actually three times faster than SIFT. About 64 dimensions can be used in SURF to save the time consumption for the two traits which are the matching and the computation.
Scale Invariant Feature Transform is one of several computer vision algorithms which is clearly aimed at extracting distinctive and invariant features from images. Features are extracted by making the SIFT algorithm invariant to image scale, rotation, and partially robust to modifying viewpoints and changes in illumination.

In paper [10], we proposed and implemented an innovative approach based on upper and lower envelope and Eigen values techniques. Envelope represents the shape of the signature. The feature set consists of features such as large and small Eigen values computed from upper envelope and lower envelope and its union values. Both the envelopes are combined by performing union operation and their covariance is calculated. The ratios and the differences of high and low points of both these envelopes are calculated. Lastly average values of both the envelopes are obtained. These features set are coupled with support vector machine classifier that lead to 98.5% of accuracy.

In the paper [11] researchers, use this daily based biometric characteristic for identification and classification of students' papers and various exam documents used at University of Mostar. Paper uses a number of Global features, Grid Features, SIFT features. For classification, Support Vector Machine is used and the accuracy obtained is of 88.97%. Global features includes Aspect ratio, number of pixels that belong to the signature, Baseline shift, Global Slant Angle, Number of Edge Points, Number of Cross-Points and Spatial Symbols, Horizontal and vertical Center of gravity, Length name, Length surname, Maximal horizontal and vertical histogram, the length and ratio of Adjacency Columns, Heaviness of signature, Ratio of first vertical pixel to height, Number of Closed Loops.

In paper [12], a bank cheque is taken and the signature is collected from the bank cheque by taking it out by cropping the area of interest. The image is then trained and stored into the trained database using an efficient Feed forward artificial neural network. Then signatures to be tested are compared with the signatures that are stored into the test database. The accuracy of the system is tested out to be 85.00%. In pre-processing, Otsu's thresholding or binarization algorithm is used which is one of the finest one, allows image object to be separated from its background and later Gaussian low pass filter is applied to remove unwanted noise.

Fuzzy Logic and Artificial Neural Network Based Off-line Signature Verification and Forgery Detection System is presented in paper [13]. As there are distinct and necessary variations in the feature set of each signature, so in order to match a particular signature image with the database image, the structural features of the signatures and also the local feature variations in the signature characteristics are used. The paper suggests that, variation in personality of signatures, because of age, sickness, geographic location and emotional state of the person actuates the problem

A new approach to document image retrieval based on signature is described in paper [14]. The database consists of document images with English text combined with headlines, logo, ruling lines, trade mark and signature. In searching a particular repository of business documents, the actual work to be done is using a query signature image to retrieve from a database. DT-RCWF and DT-CWT are used for extraction features and recognition rate of images is of 79.32%.

This paper [15] focuses on the pre-processing phase, which is an alternative way to improve the accuracy and to make such factors stable. This research is basically based on the hypothesis that, a table signature size is able to boost up the efficiency which means the recognition rate. Polar Scale Normalization, Adaptive Variance Reduction, Histogram of Oriented Gradients are the techniques used in the system and 98.39% of accuracy is shown.

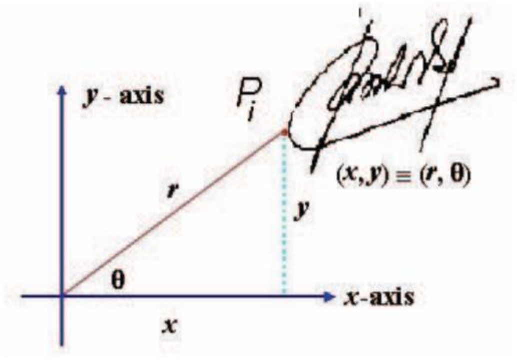


Figure 6: Polar scale normalization

The pre-processing techniques include Binarization, Complementation, Cropping, and Resizing. The classification of the test data is done using an efficient artificial neural network.

A signature verification system based on Dynamic Time Warping (DTW) is proposed in paper [16]. Pre-processing techniques have Maximum Length Vertical Projection (MLVP) method, Minimum Length Horizontal Projection (MLHP) method. The technique basically works by extracting the vertical projection based features from signature images and by comparing probe and reference feature templates using elastic matching classification. A 98% accuracy is gained and show promising results.

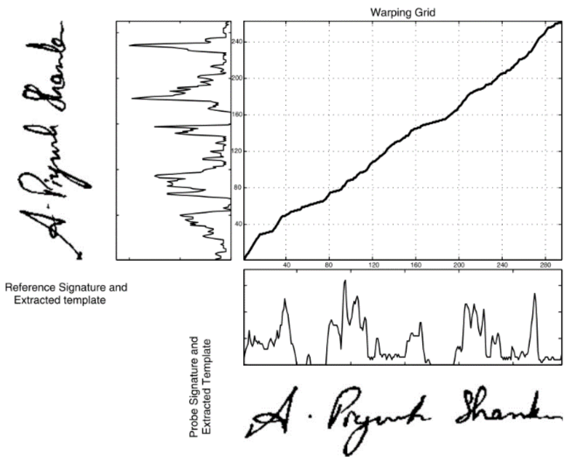


Figure 7: DTW algorithm of paper [16]

Present paper [17] focuses on different steps including browsing a bank cheque, pre-processing, feature extraction, recognition. The paper extracts and uses features such as Contrast, Homogeneity, Energy and Entropy for comparing two different signature images.

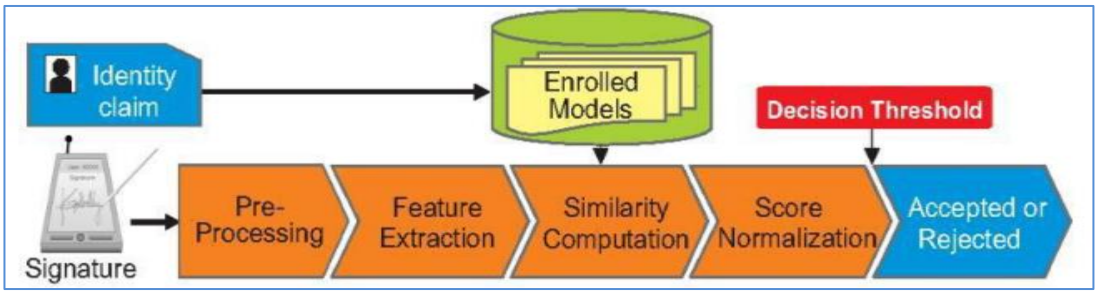


Figure 8: Flowchart of paper [17]

SIFT and a SURF algorithm which is used for enhanced offline signature recognition is used in paper [18]. This process, Bag-of-word features, was operated by making vector quantization technique, which outlined the key points for each training image inside a unified dimensional histogram. Accuracy obtained is 98.75%. SVM classification has limitations in speed and size while both phase of try and test of the algorithm and the chosen of the kernel function parameters.

A writer independent offline handwritten signature verification model, also known as global model, for signature verification is proposed in the paper [19]. Otsu's thresholding is used for pre-processing the image. System uses Local Binary Pattern based feature vector extraction along with its variants and classifies the images using SVM. An accuracy of 95.75% is received and the results shown are promising.

An offline signature verification using neural network is projected in paper number [20], where the signature is written on a paper are obtained using a scanner or a camera captured and presented in an image format. In pre-processing, color to grayscale, and finally to black and white, Resizing the image, thinning. Features extracted include Eccentricity, Skewness, Kurtois, Orientation Entropy, Euler number, Solidity, Mean, Standard deviation. The classification is done using a Cascaded feed-forward back-propagation networks and recognition rate of 92.5% is obtained.

System converts a scanned signature to a shape form and Eigen-signature construction is proposed for extracting the feature vector from a shape formed signature. The test feature vector is said to belong to i^{th} class, if it possess minimum distance with i^{th} class sample when compared to other class samples Thus paper [21] shows accuracy of 91.40%. The pre-processing techniques include binarizing or thresholding. Noise is eliminated using a simple morphological filter, thinned. The test feature vector is said to belong to i^{th} class, if it possess minimum distance with i^{th} class sample when compared to other class samples.

The purpose in paper number [22] is to select relevant features from those features set. For doing that, researchers compute the importance score of each features using two methods: Information Gain Ratio and Correlation. Over 440 Histogram features, 550 Fresh features, 220 DCT features were extracted an accuracy obtained was 95.5%. Limitations seen are rotation normalization and time normalization. Once all the global features are extracted well, the next trick is ranking these features. Ranking score is obtained from Information Gain Ratio and Correlation. So the result obtained are two list of 1210 features set ranked in order of their Gain. From each list, the features set are then divided into sub features set having 10 first ranked features, 20 first ranked features, 30 first ranked features, and up to 1210 features. So each ranked list will produce sub features set of 121 features.

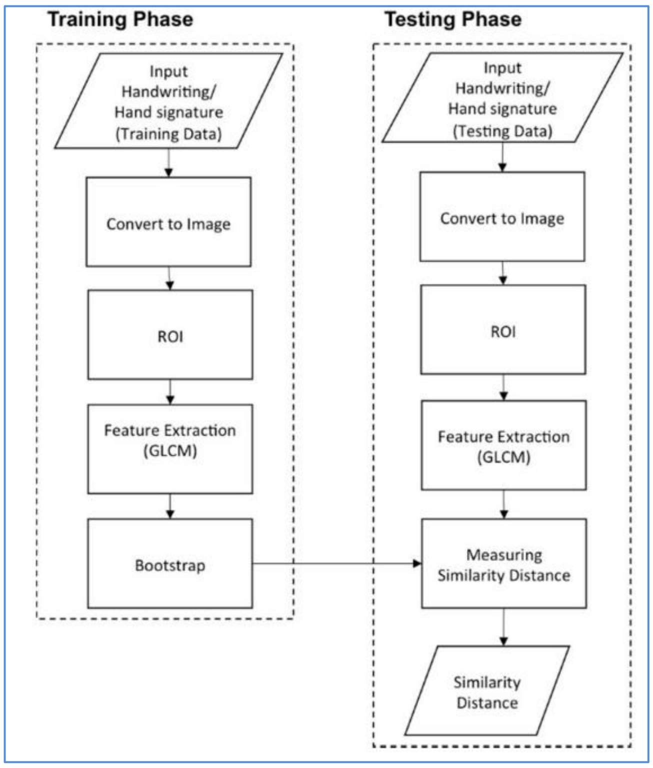

Training Phase
Testing Phase
Input Handwriting/ Hand signature (Training Data)
Input Handwriting/ Hand signature (Testing Data)
Convert to Image
Convert to Image
ROI
ROI
Feature Extraction (GLCM)
Feature Extraction (GLCM)
Bootstrap
Measuring Similarity Distance
Similarity Distance

Figure 9: Flow Diagram for Paper [23]

Signature and handwriting recognition on a mobile device using the Gray Level Co-occurrence Matrix (GLCM) for texture-based feature extraction and the bootstrap for performing single classifier model is proposed in the paper [23]. The accuracy obtained is 88.46%.

The paper [24] examines authentication systems based on handwritten signature and the main informative parameters of signature such as size, shape, velocity, pressure, etc. along with DCT, DFT. System using K-Nearest neighbors' algorithm and Random forest algorithm for classification and the accuracy of 95% is shown.

An offline signature recognition system which uses histogram of oriented gradients is presented. Pre-processing techniques used are color normalization, median filtering, angle normalization and exact bounding box, resized into 256×512 pixels. Feature extraction is done with Gradient Computation, Gradient Vote, and Normalization Computation. A three layered feedforward backpropagation neural network is used for classification and the recognition rate of paper [25] is 96.87%

In this paper [26], the trace transform based affine invariant features are applied for signature verification. The diametric and trace functions are appropriately chosen to compute a set of circus functions from each signature image. Recognition rate of 76% is obtained. Low accuracy, more invariant functional will be designed for usefulness in signature verification.

This paper [27], deals with the analysis of discriminative powers of the features that can be extracted from an on-line signature, how it's possible to increase those discriminative powers by Dynamic Time Warping as a step in the pre-processing of the signal coming from the tablet. The accuracy obtained is 99.7%. Pre-processing is done using Filtering, equispacing by Linear Interpolation, Normalization, DTW Alignment, Derived Signals: Speed and Acceleration The main processing includes init minus min End minus min Average Root Square Average Time over 0 Crossing 0 Mean over 0 Standard deviation Number of traces, Time of signature, Writing Time of signature / Time of Signature, Height / Width, Area, Length.

Paper [28] uses a texture base approach called the Local binary Pattern algorithm along with SVM, The signature models are trained with genuine signatures on white background and tested with other genuine and forgeries mixed with different backgrounds. Results depict that a basic version of local binary patterns (LBP) or local directional and derivative patterns are more robust than others like GLCM features to the grey level distortion with SVM with histogram oriented kernels as a classifier or rotation invariant uniform LBP. The proposed configurations are checked and evaluated under different situations and conditions: changing the number of training signature images, database with different inks, multiple signing sessions, increasing the number of signers and combining different features Results have also been provided when looking for the signature in the check and segmenting it automatically. In all these cases, the best results were obtained with the LDerivP feature set, which improve the results obtained in the predefined baseline, showing quite significant improvements with fictitious signature images.

Paper [29] uses Conic section function neural network (CSFNN) circuitry was designed for offline signature recognition. CSFNN is a unified framework for multilayer perceptron (MLP) and radial basis function (RBF) networks and the size of feature vectors was not suitable for designed CSFNN chip structure owing to the input limitations of the circuit. Pre-processing is done using noise reduction algorithm, skeletonization. The few main disadvantages of analog systems include its sensitivity to ambient noise and to temperature. Accuracy shown is 95.5%.

This paper [30] explores the usefulness of local binary pattern (LBP) and local directional pattern (LDP) texture measures to discriminate off-line signatures. Comparison between these several texture normalizations is generated in order to look for reducing pen dependence. The recognition rate is 91.92%. The pre-processing techniques include binarized, cropped, or-exclusive operation, Texture histogram normalization. Least Squares Support Vector Machine (LS-SVM) is applied for classification.

The goal of this study [31] is to investigate the effect of combining transform features to authenticate signatures. Due to genuine human error and lack of consistency, comparing signatures requires pre-processing to assist with standardization. An EER of

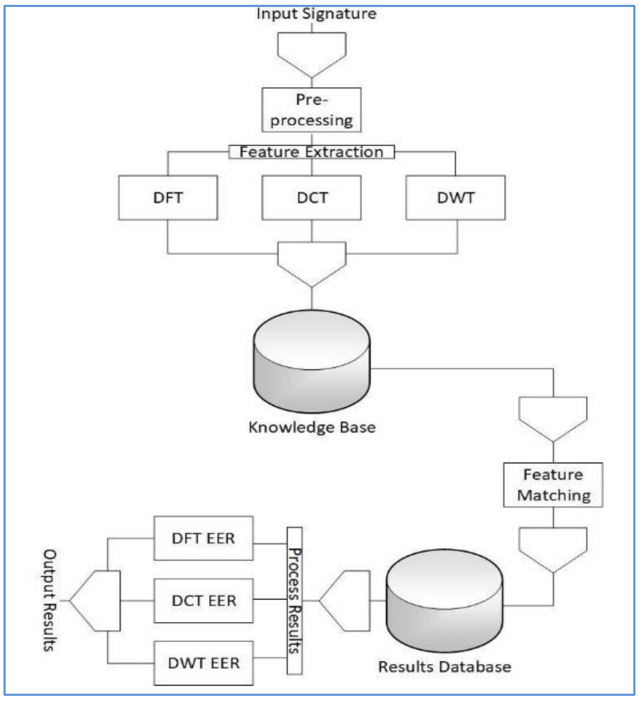


Figure 10: Flowchart of system in paper [31]

2.46% which means a recognition rate of 99.40% is shown by the system. The features extracted include x coordinate, y coordinate, pen pressure, azimuth, altitude, DFT, DCT and DWT. Classification is done with Fast Dynamic Time Warping (FastDTW) algorithm

In this paper [32], gender discrimination has been proposed by feature extraction method. The proposed framework considers handwritten Hindi signature of each individuals as an input for gender detection. Recognition rate obtained is 92.90%.

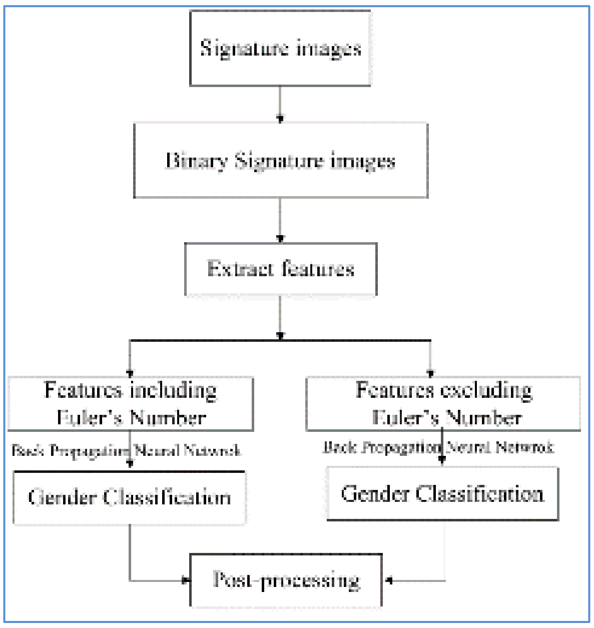

Signature images
Binary Signature images
Extract features
Features including Euler's Number
Features excluding Euler's Number
Back Propagation Neural Network
Back Propagation Neural Network
Gender Classification
Gender Classification
Post-processing

Figure 11: Flow chart of paper [32]

In order to solve the shortcomings of manual identification in technical accuracy and subjectivity, this paper [33] proposed an off-line signature identification method based on Support Vector Machine (SVM). Accuracy shown here is 85.00%. Dataset are stained by useless border. Other pre-processing techniques used in the system include Image binarization, denoising, removing blank margins. Features include global feature like height, width, area and slant angle and sum of black pixels. Identification rates within and between writing systems is prone to swing in some degree under different partitioning strategy.

This paper [34] titled presents a set of geometric signature features for offline automatic Signature verification based on the description of the signature envelope and the interior stroke distribution in polar and Cartesian coordinates. Feature extraction method includes Outline Detection and Representation, Feature Vector Based on Polar Coordinates, Feature Vector Based on Cartesian Coordinates. Classification is performed using The HMM Signature Model, Support Vector Machine Signature Model and Euclidean Distance-Based Signature Model.

The proposed system [35] segments each signature curve based on pen's velocity value. The signature curve, would be decomposed in low or high partition according to velocity's value. For each partition, hand movement direction between two consequent point Extracted. 95.10%. Pre-processing technique here are removing noise, translation invariance, rotation 7 scale invariance. Signature features extracted and used shape of signature would be partitioned based on dynamic feature such as velocity or pressure. Classification is done using Hidden Markov Model.

Optical flow is used to define a stability model of the genuine signatures for each signer in paper [36]. Stability between the unknown signature and reference signatures is estimated and consistency with the stability model of the signer is evaluated. Accuracy obtained is 96.00%. In pre-processing, signature image size was adjusted to a fixed area. Optical flow vectors were used for the analysis of the local stability of a signer to detect the stable regions in the signature. Optical flow vectors are for the analysis of the local stability of a signer to detect the stable regions in the signature.

In this paper [37] an efficient off-line signature verification method based on an interval symbolic representation and a fuzzy similarity measure is proposed. 88.26% Local Binary Pattern, interval-valued symbolic model. A histogram-based threshold technique, mean filter for noise removal, minimum bounding boxes. Classification of the signature image is done based on the similarity value, an adaptive writer-dependent acceptance threshold

A new approach of showing online signatures by interval-valued symbolic features. The online signature also gives global features that are to be extracted and used to form an interval-valued feature vectors. Methods for signature verification and recognition based on the symbolic representation are also proposed in paper [38] Recognition rate is 95.40%.

This paper [39] explores the utility of information derived from the dynamic time warping (DTW) cost matrix for the problem of online signature verification. The prior research and experimentations in literature primarily utilize only the DTW scores to authenticate a test signature. Accuracy measured is 97.24%. As the paper itself suggests local features & histogram representation shall be an interesting extension.

This paper [40] presents a new approach for online signature verification that exploits the potential of local stability information in handwritten signatures. Different from previous models, this approach classifies a signature using a multi domain strategy. Accuracy obtained is 97.90%. Both variable and stable regions of a signature image object could be considered for supporting the advanced personalized techniques in signature verification and recognition, since probably, from a behavioural point of view, a variability model of a signer/author could also be very complementary and informative to a stability model.

5. Proposed work (algorithm/ techniques/ experimental setup)

5.1 Algorithm to be used

The proposed work intends to stretch and extend one of the papers discussed in the above section of literature review which includes the features extraction of Local Binary Pattern from the signature images and to generate an efficient offline signature recognition and verification system. This algorithm is also to be combined with some simple features of the signature image that can have global features that define features of a part or parts of the signature as well as the global features which requires signature as a whole.

At architectural level the basic structure and flow of the system is very much similar to how most of the research finds out. The flowchart block diagram would start with image acquisition where the image would be taken in as input. Then the images would go through series of pre-processing stages such as RGB to Grey, thresholding, Boundary Box cropping and noise removal filters. In feature extraction stage different set of shape and size based are to be extracted. The images are then converted to LBP images and all the texture based LBP features are extracted again but this time from the LBP images. The generated feature set of all the images along with their respective classes would be given to a classification algorithm. This currently has K-nearest neighbors using Euclidian distance based approach and can be combined with another to increase accuracy. Finally this stage would give out the class to which the input test image belongs to as output decision class.

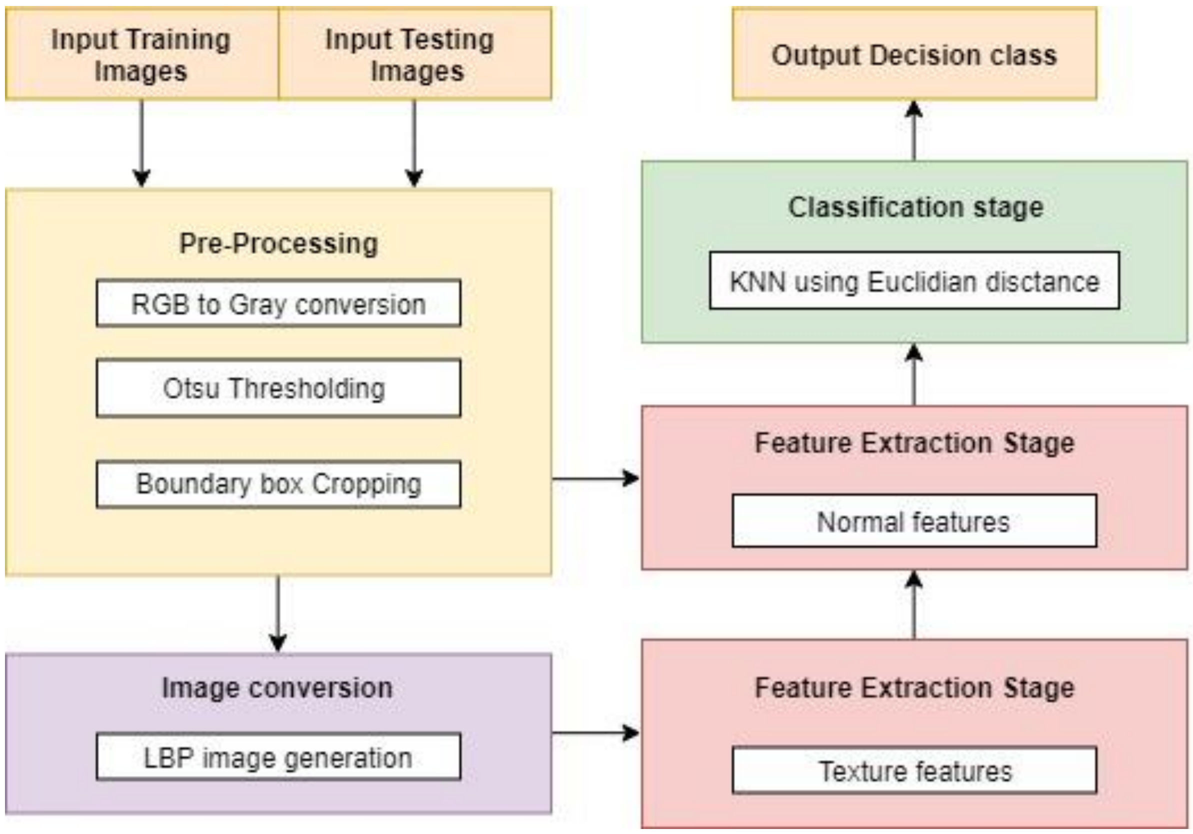


Figure 12: Block diagram of the proposed approach

5.2 Otsu thresholding

In global thresholding, we used an arbitrary value for threshold value, right? So, how can we know a value we selected is good or not? Answer is, trial and error method. But consider a **bimodal image** (*In simple words, bimodal image is an image whose histogram has two peaks*). For that image, we can approximately take a value in the middle of those peaks as threshold value, right? That is what Otsu binarization does. So in simple words, it automatically calculates a threshold value from image histogram for a bimodal image. (For images which are not bimodal, binarization won't be accurate.)

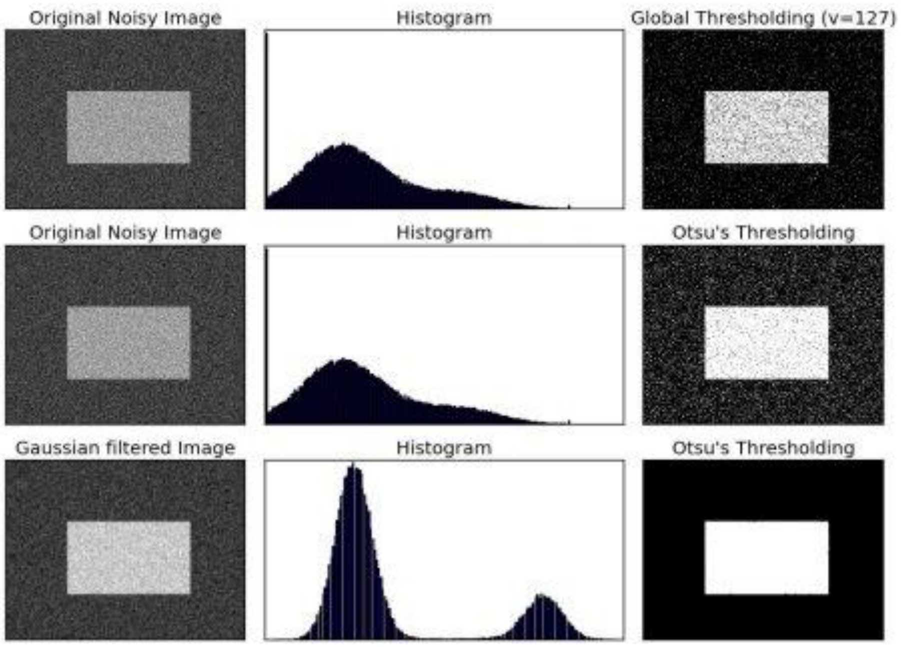
For this, our **cv.threshold()** function is used, but pass an extra flag, **cv.THRESH_OTSU**. **For threshold value, simply pass zero**. Then the algorithm finds the optimal threshold value and returns you as the second output, retVal. If Otsu thresholding is not used, retVal is same as the threshold value you used.

Check out below example. Input image is a noisy image. In first case, I applied global thresholding for a value of 127. In second case, I applied Otsu's thresholding directly. In third case, I filtered image with a 5x5 gaussian kernel to remove the noise, then applied Otsu thresholding. See how noise filtering improves the result.

Algorithm

1. Compute histogram and probabilities of each intensity level
2. Set up initial \omega _{i}(0) and \mu _{i}(0)
3. Step through all possible thresholds {\displaystyle t=1,\ldots } maximum intensity
4. Update \omega _{i} and \mu _{i}
5. Compute \sigma _{b}^{2}(t)
6. Desired threshold corresponds to the maximum \sigma _{b}^{2}(t)

Otsu's method exhibits the relatively good performance if the histogram can be assumed to have bimodal distribution and assumed to possess a deep and sharp valley between two peaks. But if the object area is small compared with the background area, the histogram no longer exhibits bimodality.[4] And if the variances of the object and the background intensities are large compared to the mean difference, or the image is severely corrupted by additive noise, the sharp valley of the gray level histogram is degraded. Then the possibly incorrect threshold determined by Otsu's method results in the segmentation error.



5.2 Local Binary Patterns

Local Binary Patterns, or LBPs for short, are a texture descriptor made popular by the work of Ojala et al. in their 2002 paper, Multiresolution Grayscale and Rotation Invariant Texture Classification with Local Binary Patterns (although the concept of LBPs were introduced as early as 1993).

Unlike Haralick texture features that compute a global representation of texture based on the Gray Level Co-occurrence Matrix, LBPs instead compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighborhood of pixels.

The first step in constructing the LBP texture descriptor is to convert the image to grayscale. For each pixel in the grayscale image, we select a neighborhood of size r surrounding the center pixel. A LBP value is then calculated for this center pixel and stored in the output 2D array with the same width and height as the input image.

For example, let’s take a look at the original LBP descriptor which operates on a fixed 3 x 3neighborhood of pixels just like this:

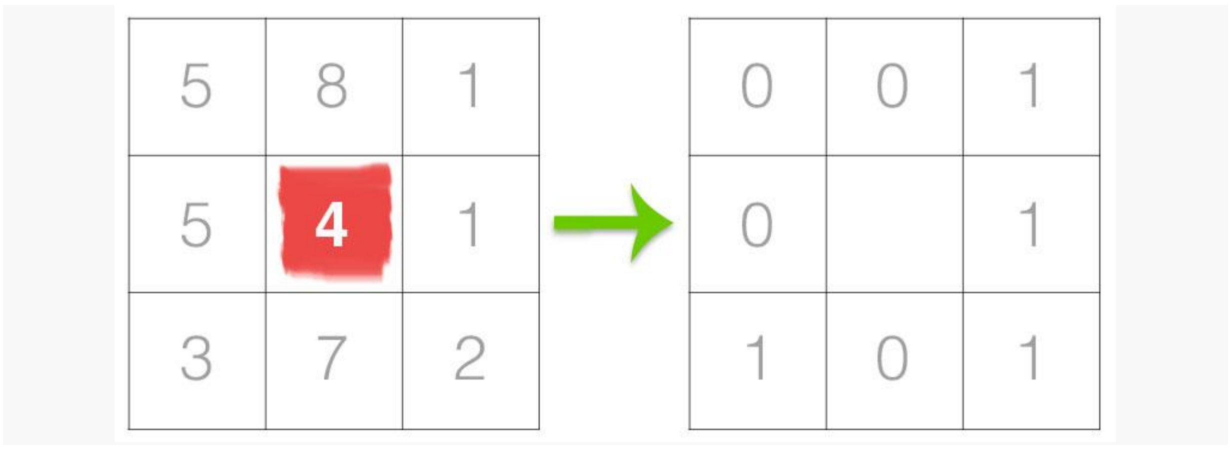


Figure 13: The first step in constructing a LBP is to take the 8 pixel neighborhood surrounding a center pixel and threshold it to construct a set of 8 binary digits.

In the above figure we take the center pixel (highlighted in red) and threshold it against its neighborhood of 8 pixels. If the intensity of the center pixel is greater-than-or-equal to its neighbor, then we set the value to 1; otherwise, we set it to 0. With 8 surrounding pixels, we have a total of 2 ^ 8 = 256 possible combinations of LBP codes.

From there, we need to calculate the LBP value for the center pixel. We can start from any neighboring pixel and work our way clockwise or counter-clockwise, but our ordering must be kept consistent for all pixels in our image and all images in our dataset. Given a 3 x 3neighborhood, we thus have 8 neighbors that we must perform a binary test on. The results of this binary test are stored in an 8-bit array, which we then convert to decimal, like this:

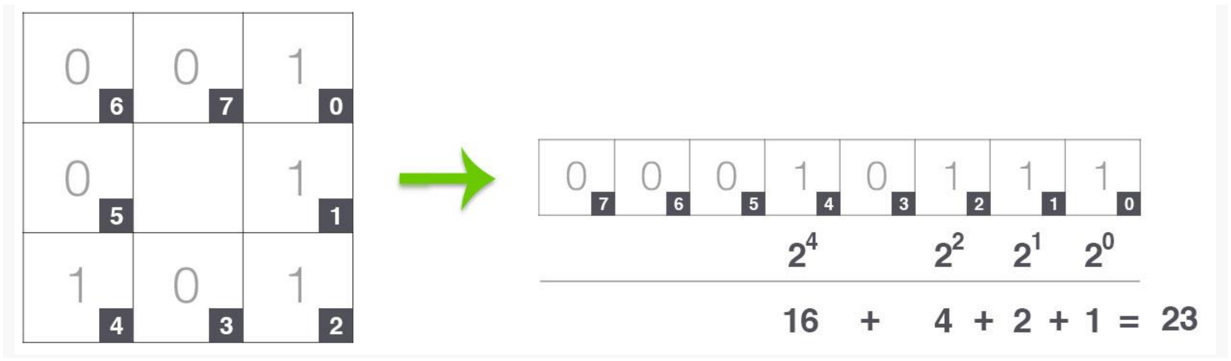


Figure 14: Taking the 8-bit binary neighborhood of the center pixel and converting it into a decimal representation. (Thanks to Bikramjot of Hanzra Tech for the inspiration on this visualization!)

In this example we start at the top-right point and work our way **clockwise** accumulating the binary string as we go along. We can then convert this binary string to decimal, yielding a value of 23. This value is stored in the output LBP 2D array, which we can then visualize below:

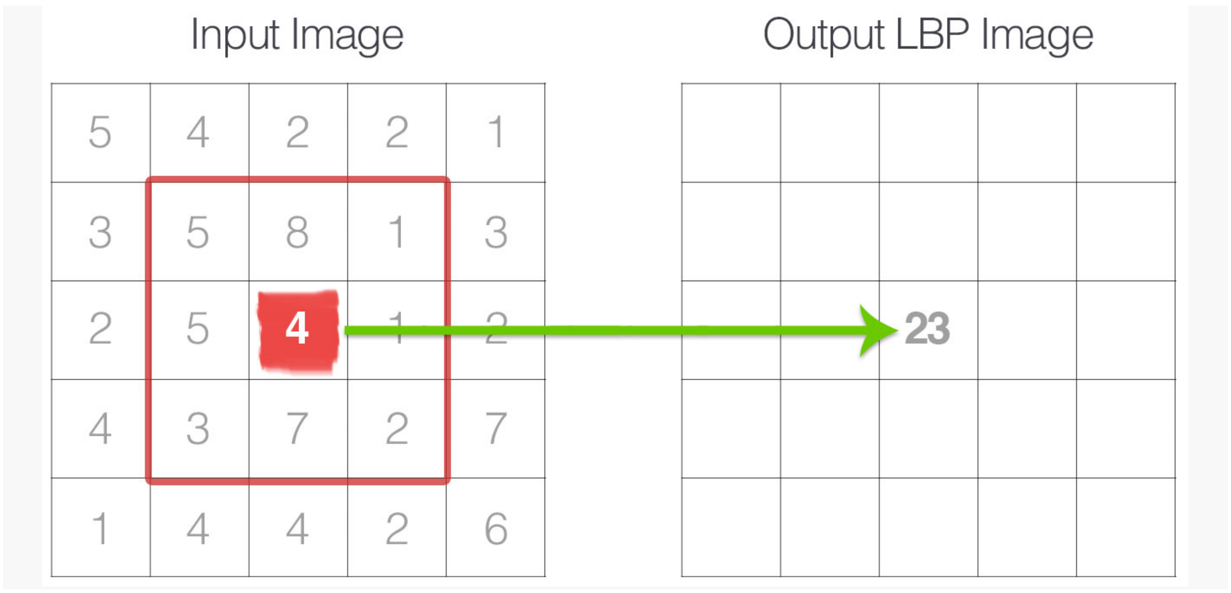


Figure 15: The calculated LBP value is then stored in an output array with the same width and height as the original image.

This process of thresholding, accumulating binary strings, and storing the output decimal value in the LBP array is then repeated for each pixel in the input image.Here is an example of computing and visualizing a full LBP 2D array:

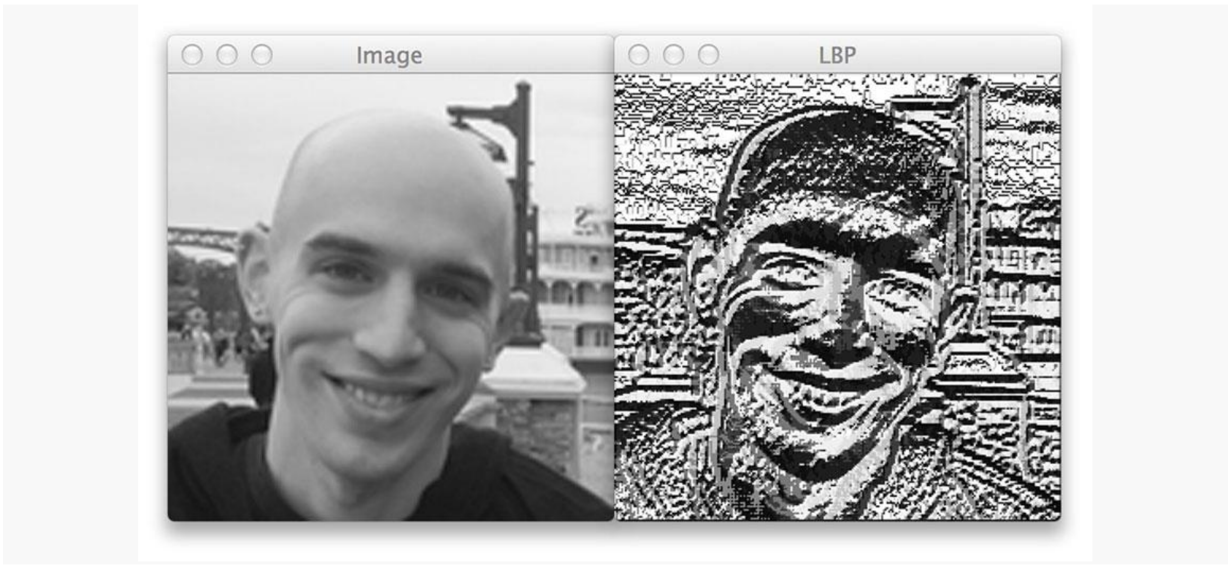


Figure 16: Computing the LBP representation (right) from the original input image (left).

The last step is to compute a histogram over the output LBP array. Since a 3 x 3 neighborhood has 2 ^ 8 = 256 possible patterns, our LBP 2D array thus has a minimum value of 0 and a maximum value of 255, allowing us to construct a 256-bin histogram of LBP codes as our final feature vector:

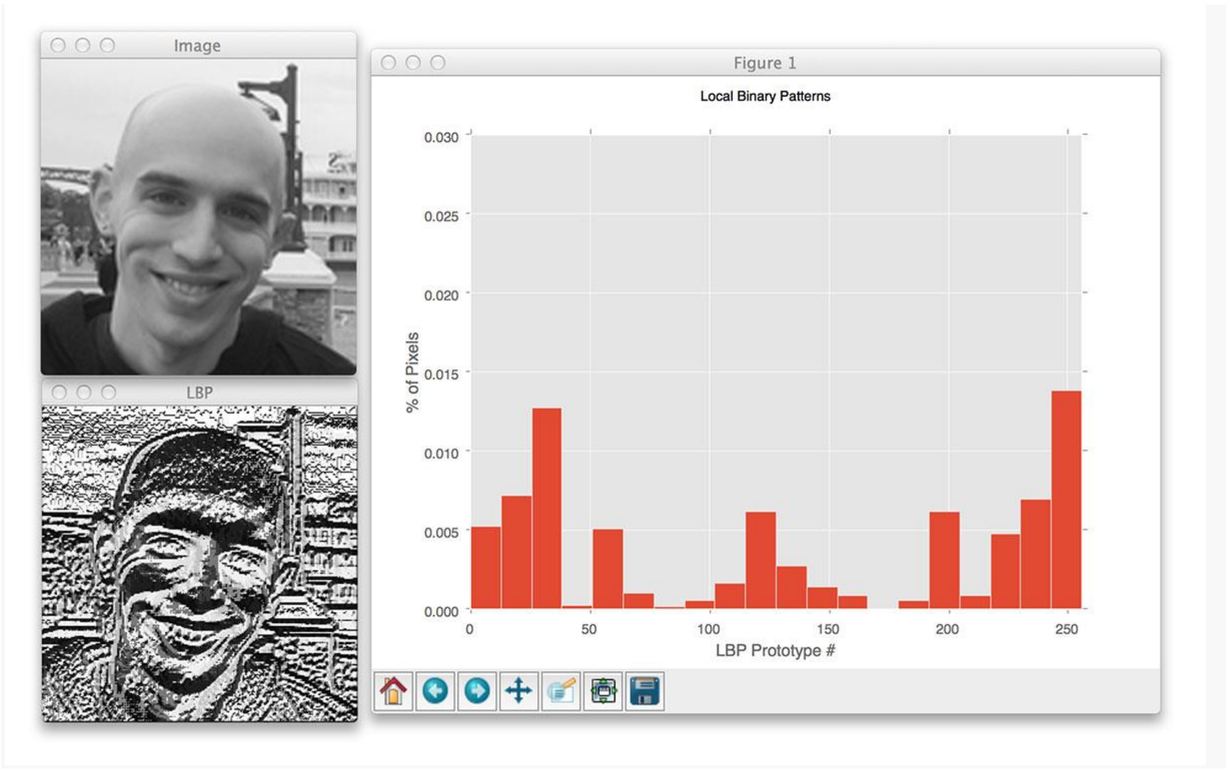


Figure 17: Finally, we can compute a histogram that tabulates the number of times each LBP pattern occurs. We can treat this histogram as our feature vector.

A primary benefit of this original LBP implementation is that we can capture extremely fine-grained details in the image. However, being able to capture details at such a small scale is also the biggest drawback to the algorithm — we cannot capture details at varying scales, only the fixed 3 x 3 scale!

To handle this, an extension to the original LBP implementation was proposed by Ojala et al. to handle variable neighborhood sizes. To account for variable neighborhood sizes, two parameters were introduced:

1. The number of points p in a circularly symmetric neighborhood to consider (thus removing relying on a square neighborhood).
2. The radius of the circle r, which allows us to account for different scales.

Below follows a visualization of these parameters:

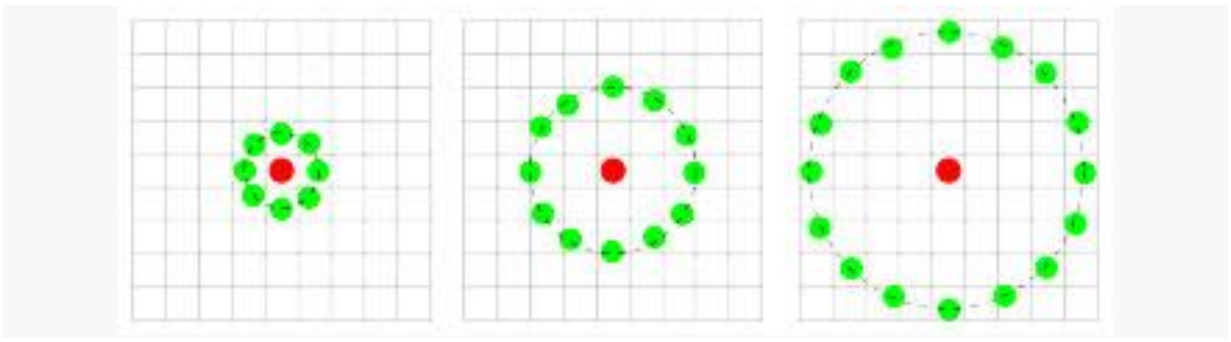


Figure 18: Three neighborhood examples with varying p and r used to construct Local Binary Patterns.

Lastly, it's important that we consider the concept of LBP **uniformity**. A LBP is considered to be uniform if it has **at most** two 0-1 or 1-0 transitions. For example, the pattern 00001000 (2 transitions) and 10000000 (1 transition) are both considered to be **uniform patterns** since they contain at most two 0-1 and 1-0 transitions. The pattern 01010010) on the other hand is not considered a uniform pattern since it has six 0-1 or 1-0 transitions.

The number of uniform prototypes in a Local Binary Pattern is completely dependent on the number of points p. As the value of p increases, so will the dimensionality of your resulting histogram. Please refer to the original Ojala et al. paper for the full explanation on deriving the number of patterns and uniform patterns based on this value. However, for the time being simply keep in mind that given the number of points p in the LBP there are p + 1 **uniform patterns**. The final dimensionality of the histogram is thus p + 2, where the added entry tabulates all patterns that are **not uniform**.

So why are uniform LBP patterns so interesting? Simply put: they add an extra level of rotation and grayscale invariance, hence they are commonly used when extracting LBP feature vectors from images.

5.2 K-Nearest Neighbours

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).
We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:

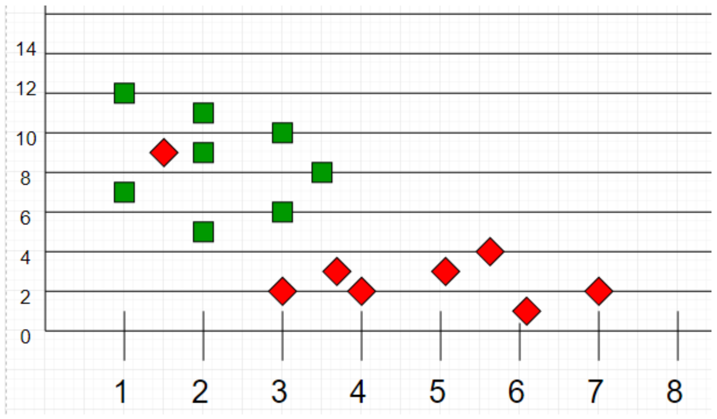


Figure 19: data points

Now, given another set of data points (also called testing data), allocate these points a group by analyzing the training set. Note that the unclassified points are marked as 'White'.

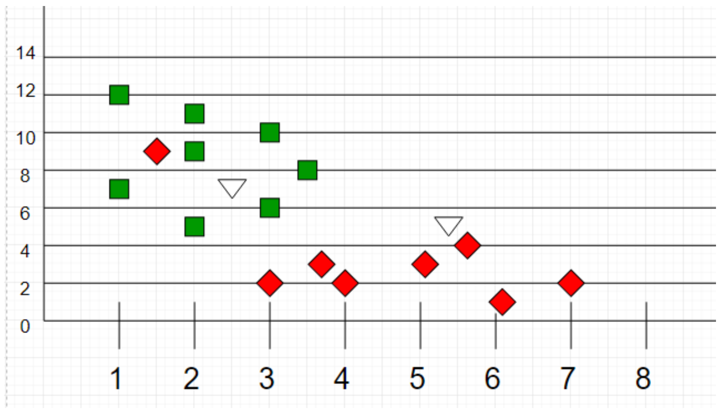


Figure 20: data points

Intuition

If we plot these points on a graph, we may be able to locate some clusters, or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbours belong to. This means, a point close to a cluster of points classified as 'Red' has a higher probability of getting classified as 'Red'.

Intuitively, we can see that the first point (2.5, 7) should be classified as 'Green' and the second point (5.5, 4.5) should be classified as 'Red'.

Algorithm

Let 'm' be the number of training data samples. Let p be an unknown point.

1. Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y).

```
2. for i=0 to m:
3.   Calculate Euclidean distance d(arr[i], p).
```

4. Make set S of K smallest distances obtained. Each of these distances correspond to an already classified data point.
5. Return the majority label among S.

5.2 Implementation tools end setup

There are be many implementation programming tools that can be used for image Processing and Machine learning projects like MATLAB. This project will also require a database and an interface between the programming tool and the database.

5.2.1 Python using PyCharm

Implementation programming tool to be used here is Python. Python is powerful... and fast; plays well with others, runs everywhere, is friendly & easy to learn and is Open. It is a popular programming language used in web development (server-side), software development, mathematics, system scripting.

- Python can be easy to pick up whether you're a first time programmer or you're experienced with other languages. The following pages are a useful first step to get on your way writing programs with Python!
- The Python Package Index (PyPI) hosts thousands of third-party modules for Python. Both Python's standard library and the community-contributed modules allow for endless possibilities.

Figure 213: Python logo

Contains number of libraries, which are easy to install & import...

- OpenCV : Computer vision and machine learning software library.
- NumPy : Scientific computing & array-processing
- Imutils : Functions to make basic image processing functions easier
- Math : Provides access to the mathematical functions
- MatplotLib : Python 2D plotting library
- Pymysql : A simple database interface for Python
- OS : allows easy file handling
- Scipy : Provides many user-friendly and efficient numerical routines
- PySimpleGUI : User interface renderrer

PyCharm is a python editor and compiler. Allow to be more productive by saving time while PyCharm takes care of the routine. Allows to focus on the bigger things and embrace the keyboard-centric approach to get the most of PyCharm's many productivity features.

Get Smart Assistance with PyCharm that knows everything about our code. We can always rely on it for intelligent code completion, on-the-fly error checking and quick-fixes, easy project navigation, and much more.

Figure 22: PyCharm logo

5.2.2 Database using MySQL

A **database** is a collection of information that is organized so that it can be easily accessed, managed and updated. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

So in this project we are going to deal with a lot of data such as training features, testing features and also classes etc. So to store and reuse the data from somewhere, we are going to need a database.

SQL is a standard language for storing, manipulating and retrieving data in databases. It is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language. SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Figure 23: SQL logo

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

Figure 24: MySQL logo

6 Work done

6.1 Research

The work done includes further research of the related research papers to explore more and more about in depth knowledge of the existing algorithms that can be used for signature recognition and verification. The research allowed me to read more research papers on one of the algorithm that I have finalized to use, Local Binary Pattern based approach. Along with finalization of the algorithm, I have read and also studied more number of research papers based on Signature Recognition and Verification and an excel sheet of all the Literature Review has been prepared The literature review contains total of 40 research papers based on the topic Signature Recognition. Most of the papers make use of 3 stages: Preprocessing stage, Feature extraction stage, Classification stage

6.2 Dataset preparation

The work also includes dataset preparation include downloading more and more of the online available datasets that people around the world generate and upload for others to use. I have thus downloaded the dataset from the Web.

The data set I have taken from one of the websites with the uploaded by researchers online on the website mentioned in References *[42]*. Since the dataset downloaded is vast and contains a lot of images I have prepared a subset of the vast dataset and kept aside directory of folders of images:

Total 25 Authors, all having genuine as well as forged signatures and thus 50 classes. The dataset is divided in an approximate ratio of 4:1 for preparing training and testing data respectively. Therefore 637 training images (77.12%) and 189 testing images (22.88%) now combine to a total of 566 images

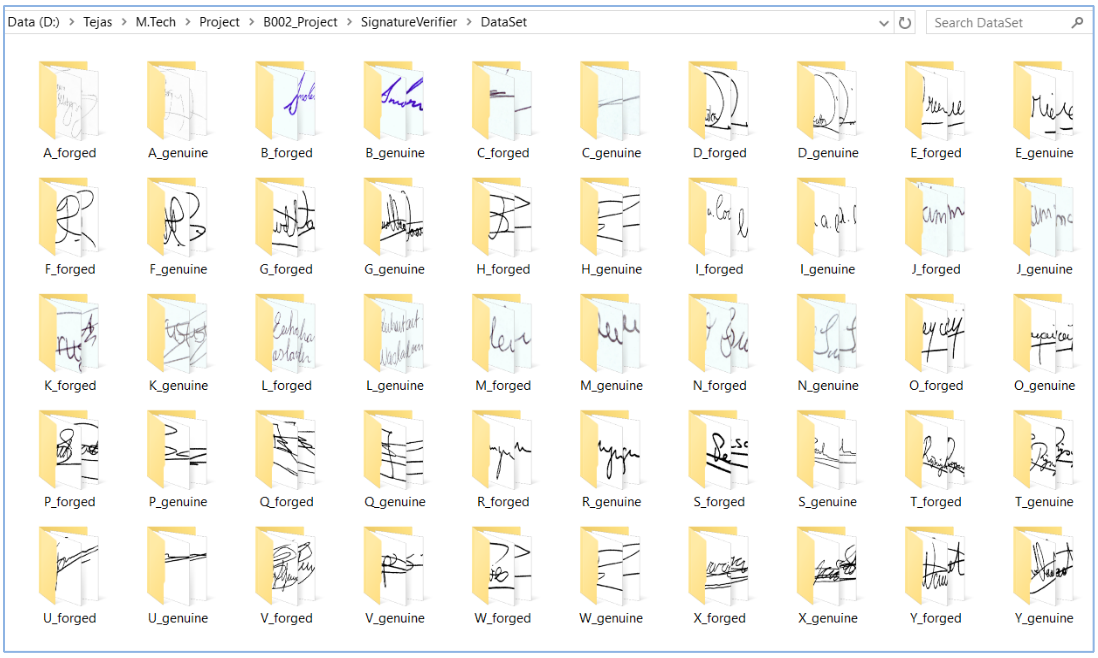
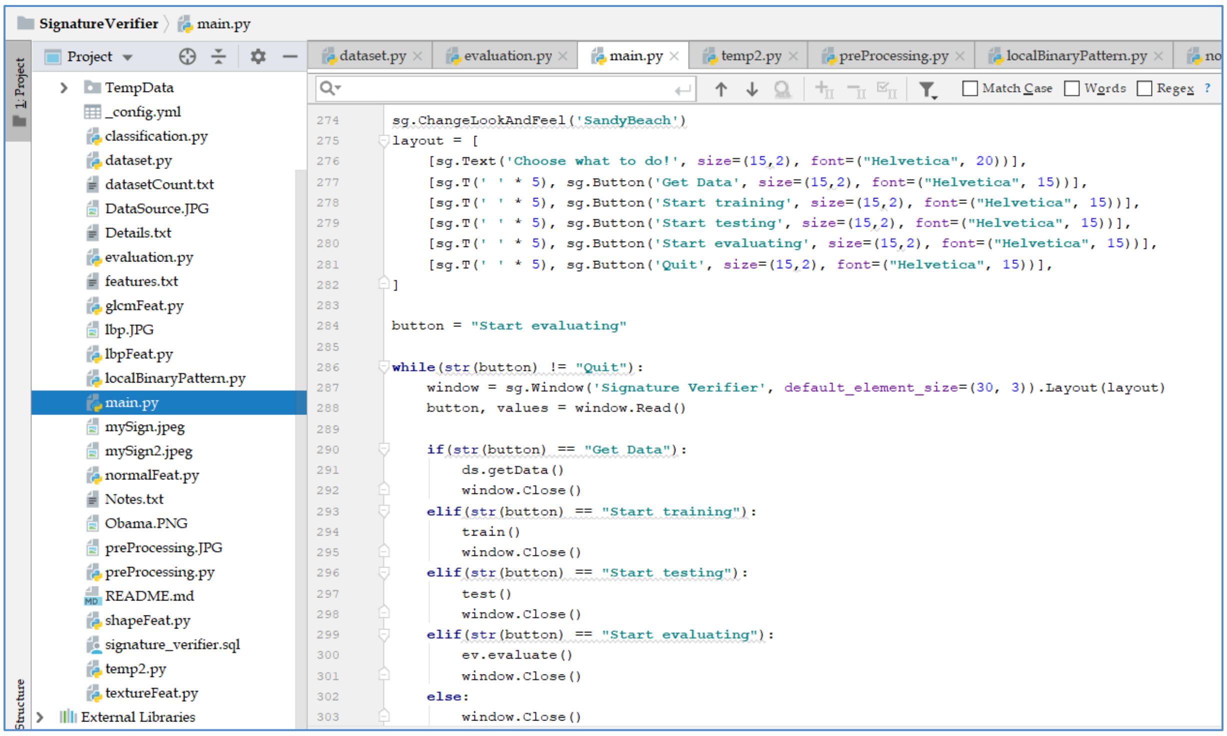


Figure 25: Dataset prepared

6.3 Implementation

Implementing with python includes total of 8 python.py files and an SQL file created in PyCharm that can be viewed as below Preprocessing stage that have been implemented include reading the image resizing, grayscale, denoising, threshold based segmentation, boundary box generation displaying the image



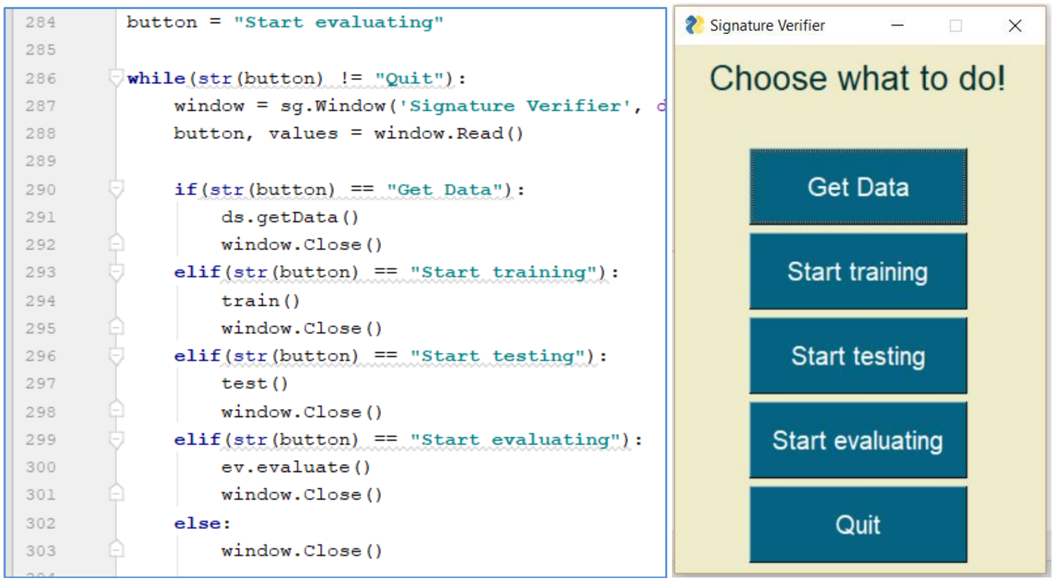
```
SignatureVerifier 〉 main.py

Project
  > TempData
    _config.yml
    classification.py
    dataset.py
    datasetCount.txt
    DataSource.JPG
    Details.txt
    evaluation.py
    features.txt
    glcmFeat.py
    lbp.JPG
    lbpFeat.py
    localBinaryPattern.py
    main.py
    mySign.jpeg
    mySign2.jpeg
    normalFeat.py
    Notes.txt
    Obama.PNG
    preProcessing.JPG
    preProcessing.py
    README.md
    shapeFeat.py
    signature_verifier.sql
    temp2.py
    textureFeat.py
  > External Libraries
```

```
sg.ChangeLookAndFeel('SandyBeach')
layout = [
    [sg.Text('Choose what to do!', size=(15,2), font=("Helvetica", 20))],
    [sg.T(' ' * 5), sg.Button('Get Data', size=(15,2), font=("Helvetica", 15))],
    [sg.T(' ' * 5), sg.Button('Start training', size=(15,2), font=("Helvetica", 15))],
    [sg.T(' ' * 5), sg.Button('Start testing', size=(15,2), font=("Helvetica", 15))],
    [sg.T(' ' * 5), sg.Button('Start evaluating', size=(15,2), font=("Helvetica", 15))],
    [sg.T(' ' * 5), sg.Button('Quit', size=(15,2), font=("Helvetica", 15))],
]

button = "Start evaluating"

while(str(button) != "Quit"):
    window = sg.Window('Signature Verifier', default_element_size=(30, 3)).Layout(layout)
    button, values = window.Read()

    if(str(button) == "Get Data"):
        ds.getData()
        window.Close()
    elif(str(button) == "Start training"):
        train()
        window.Close()
    elif(str(button) == "Start testing"):
        test()
        window.Close()
    elif(str(button) == "Start evaluating"):
        ev.evaluate()
        window.Close()
    else:
        window.Close()
```

Figure 26: Python program files prepared in PyCharm

6.3.1 Main.py

The entire system is controlled from the main function which is inside the main.py. This is the main python file where the system working starts, calls the other functions, gives the appropriate results and ends.

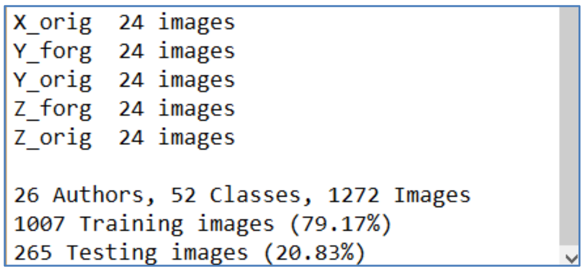


```
button = "Start evaluating"

while(str(button) != "Quit"):
    window = sg.Window('Signature Verifier', d
    button, values = window.Read()

    if(str(button) == "Get Data"):
        ds.getData()
        window.Close()
    elif(str(button) == "Start training"):
        train()
        window.Close()
    elif(str(button) == "Start testing"):
        test()
        window.Close()
    elif(str(button) == "Start evaluating"):
        ev.evaluate()
        window.Close()
    else:
        window.Close()
```

Figure 27: Main.py and GUI

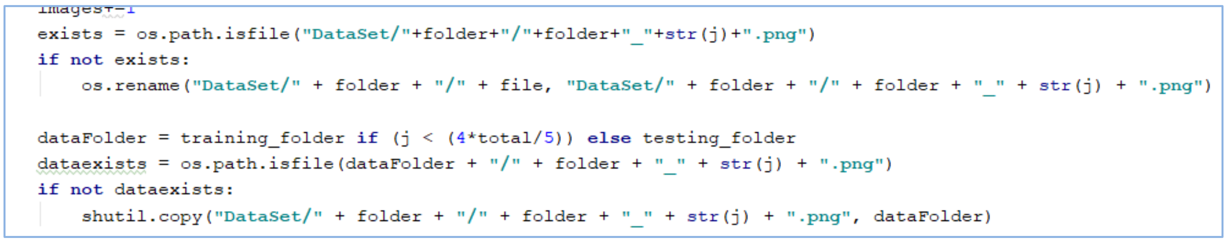
6.3.2 Dataset.py

Our dataset is read, renamed, copied and organized in the correct naming convention to a different folder, from where our system will use. For eg : xyz.png → A_orig_7.png. The function also writes a dataAnalysis.txt where it gives an analysis of all the data and the count of all the dataset



```
X_orig  24 images
Y_forg  24 images
Y_orig  24 images
Z_forg  24 images
Z_orig  24 images

26 Authors, 52 Classes, 1272 Images
1007 Training images (79.17%)
265 Testing images (20.83%)
```

Figure 28: dataAnalysis.txt



```
exists = os.path.isfile("DataSet/"+folder+"/"+folder+"_"+str(j)+".png")
if not exists:
    os.rename("DataSet/" + folder + "/" + file, "DataSet/" + folder + "/" + folder + "_" + str(j) + ".png")

dataFolder = training_folder if (j < (4*total/5)) else testing_folder
dataexists = os.path.isfile(dataFolder + "/" + folder + "_" + str(j) + ".png")
if not dataexists:
    shutil.copy("DataSet/" + folder + "/" + folder + "_" + str(j) + ".png", dataFolder)
```

Figure 29: Dataset.py

6.3.3 PreProcessing.py

This function takes the image in the raw format and converts it into a pre-processed format. This will make the image ready for processing. Preprocessing stage that have been implemented include reading the image resizing, RGB to Grey conversion, de-noising, threshold based segmentation, boundary box generation.

The binarization or also called as thresholding is method which converts a grey image into a black and white image. This is done using a threshold and hence the name Thresholding. But a problem in thresholding is finding the correct threshold and thus we use an advanced thresholding technique called the Otsu's thresholding. The Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. We can see from the adjoining image how the threshold must vary for different images having different histograms.

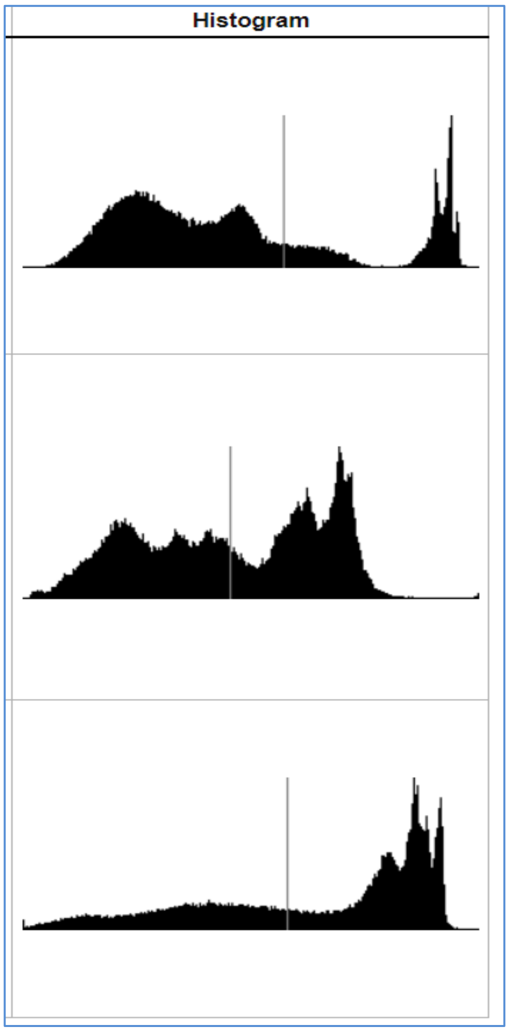


Figure 30: Varying thresholds

The boundary box cropping is one of the preprocessing stages that was not a predefined function so I got the liberty to create it myself. Its general idea is to remove the white spaces from all four sides of the image and make the problem a little smaller. The idea here was to simply keep cropping out all the rows until we find the first black pixel from top and bottom and do the same for columns from both the sides.

After few of the preprocessing stages the signature image would become a lot cleaner without noise and ready to be used for further processing. Following image shows the same.

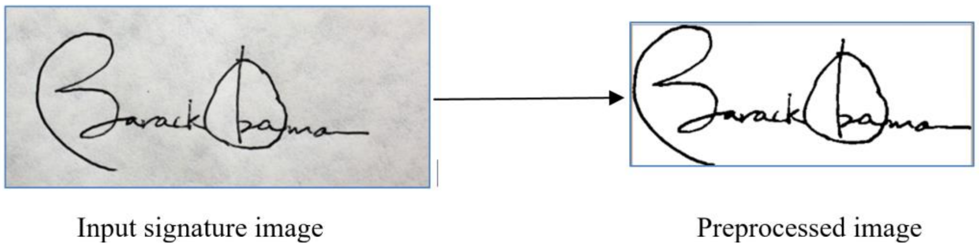
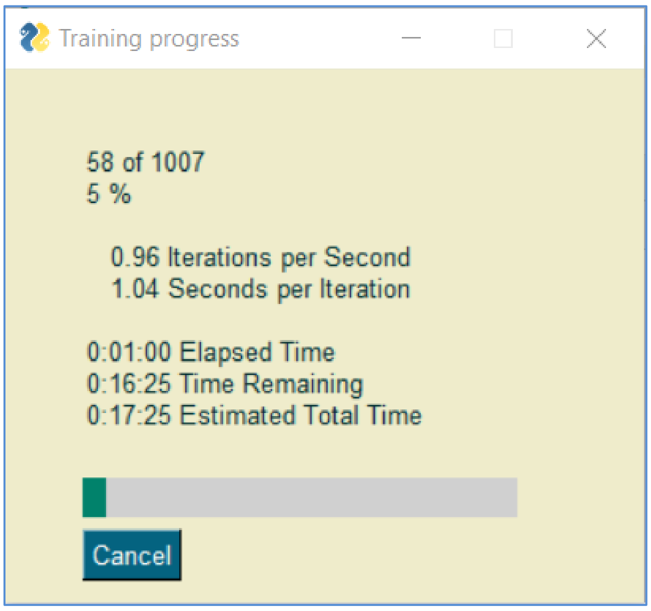


Figure 31: Preprocessing signature image

6.3.4 NormalFeat.py

Feature extraction stage means to take out numeric values out of the image so that an algorithmic process can compare them and in turn compare the image. We can use a feature set having a huge number of features but that will not do any good if their impact is not much over their image. Thus we narrow down to a feature vector set having 16 features which includes 8 normal features and 8 texture features. The pre-processed image is used for taking out normal features and LBP image is given to extract texture features.

The following image shows the PySimpleGui progress bar for our training period for over 1000 images.

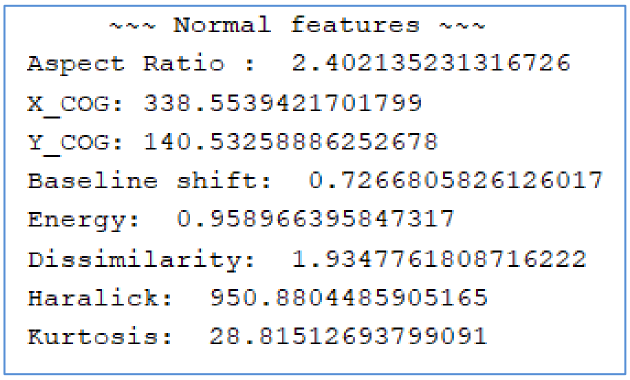


```
58 of 1007
5 %

  0.96 Iterations per Second
  1.04 Seconds per Iteration

0:01:00 Elapsed Time
0:16:25 Time Remaining
0:17:25 Estimated Total Time
```

Figure 32: Progress meter bar GUI while training and testing

The normal shape based features that are extracted from the pre-processed image are given below and are also shown on the above image:

1. Aspect Ratio: The ratio of the width to the height of an image or screen.
2. Center of Gravity - X: The Center of Gravity or Center of Mass statistic calculates where the COG of the image lies. The COG of X is calculated by:
   COG_X = COG_X + (I*x)
3. Center of Gravity - Y: The Y- Center of Gravity :
   COG_Y = COG_Y + (I*y)
4. Baseline Shift: This depicts which side (left or right) is the signature waited more and by how much. Calculated difference of left and right COG_Y.
5. Energy: the sum of square elements in GLCM
6. Dissimilarity: the weights with which GLCM probabilities are multiplied increase linearly away from the diagonal
7. Haralick: describe the overall texture of the image using measures
8. Kurtosis: Kurtosis is a measure of the combined weight of a distribution's tails relative to the center of the distribution



```
      ~~~ Normal features ~~~
Aspect Ratio :  2.402135231316726
X_COG: 338.5539421701799
Y_COG: 140.53258886252678
Baseline shift:  0.7266805826126017
Energy:  0.958966395847317
Dissimilarity:  1.9347761808716222
Haralick:  950.8804485905165
Kurtosis:  28.81512693799091
```

Figure 33: Normal features printed on console

6.3.5 LocalBinaryPattern.py

This python file contains function that converts the image to an LBP image which stands for Local Binary Pattern. The code in the image convert the image into LBP image which is darker and shows off the textures of the image to help us extract them. LBP stands for Local Binary Pattern. The grey image is converted to a LBP image in this stage. This LBP image is dark in colour and is very much useful to extract texture based features out of the image.

The first step in constructing a LBP is to take the 8 pixel neighbourhood surrounding a center pixel and threshold it to construct a set of 8 binary digits. Taking the 8 bit binary neighborhood of the center pixel and converting it into a decimal representation. The calculated LBP value is then stored in an output array with the same width and height as the original image. The mask is then moved ahead in the image and the process repeats. The following image shows the steps with an example.

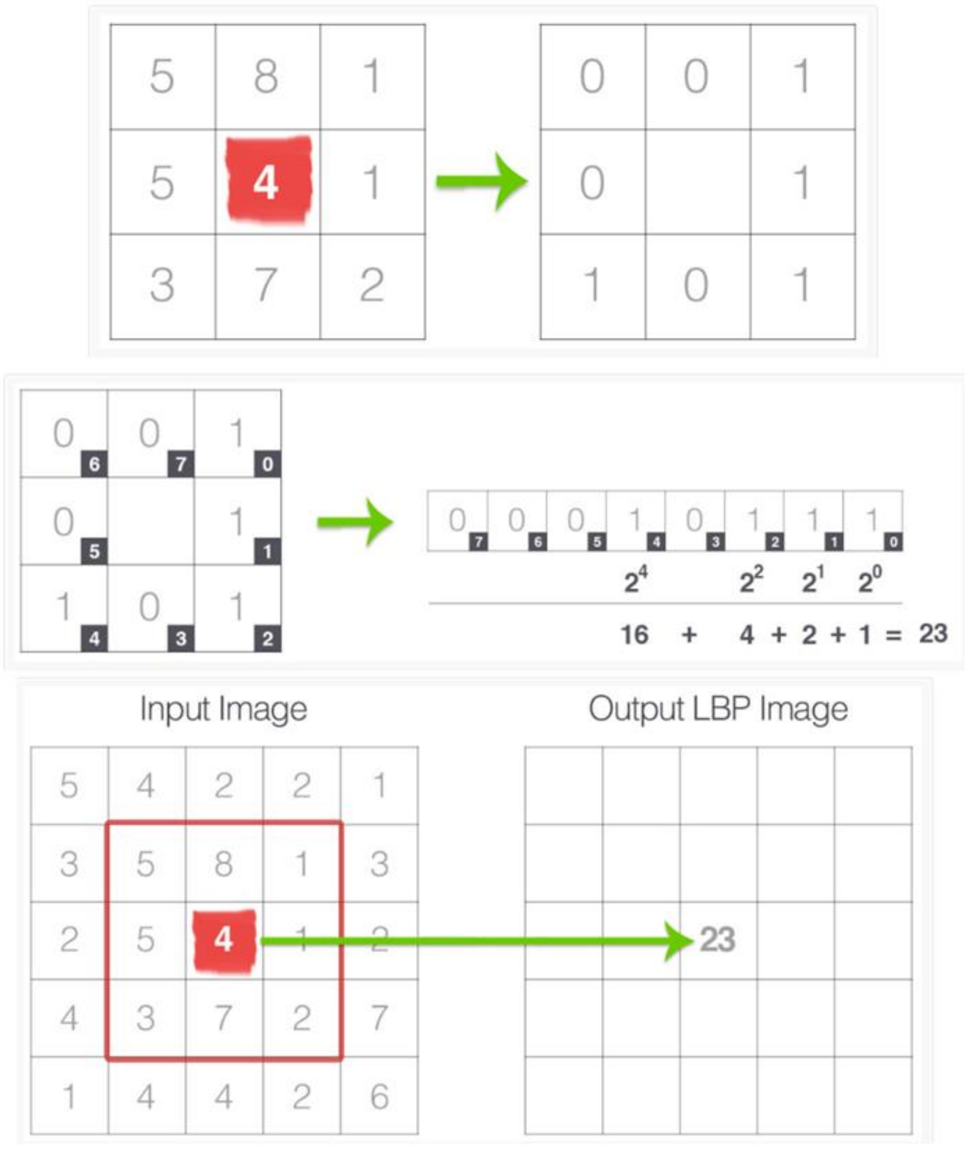


Figure 34: LBP image conversion algorithm

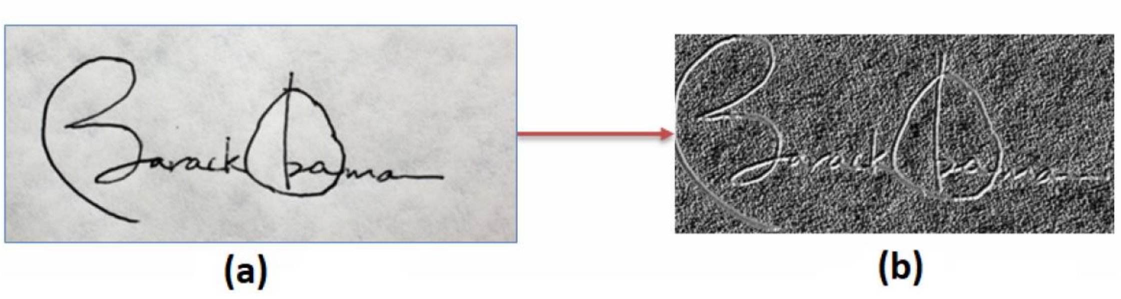
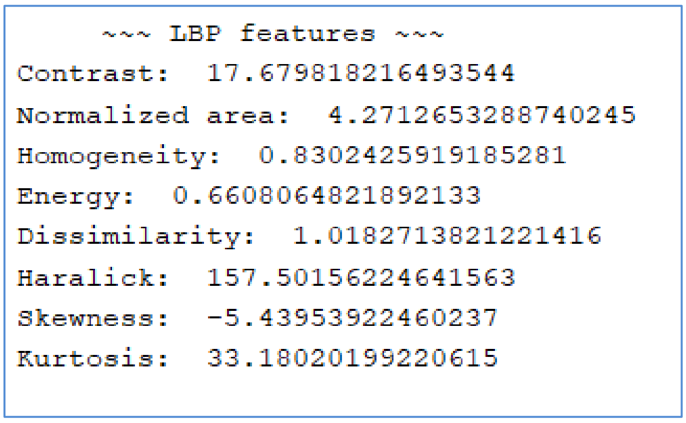


Figure 35: LBP image conversion:
(a) Grey image; (b) LBP image

6.3.6 LbpFeat.py

The LBP texture based features that are extracted from the LBP image are given below:

9. Contrast: measure of the intensity contrast between a pixel and its neighbor over the whole image
10. Normalized Area: The amount of darker pixels that constitute the object in the image divided by the total size (no of pixels) in the image.
11. Homogeneity: value that calculates the tightness of distribution of the elements in the GLCM to the GLCM diagonal
12. Energy: the sum of square elements in GLCM
13. Dissimilarity: Dissimilarity measure of LBP image
14. Haralick: Haralick feature of LBP image
15. Skewness: asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right
16. Kurtosis: Kurtosis measure of LBP image



```
     ~~~ LBP features ~~~
Contrast:  17.679818216493544
Normalized area:  4.2712653288740245
Homogeneity:  0.8302425919185281
Energy:  0.6608064821892133
Dissimilarity:  1.0182713821221416
Haralick:  157.50156224641563
Skewness:  -5.43953922460237
Kurtosis:  33.18020199220615
```

Figure 36: Normal features printed on console

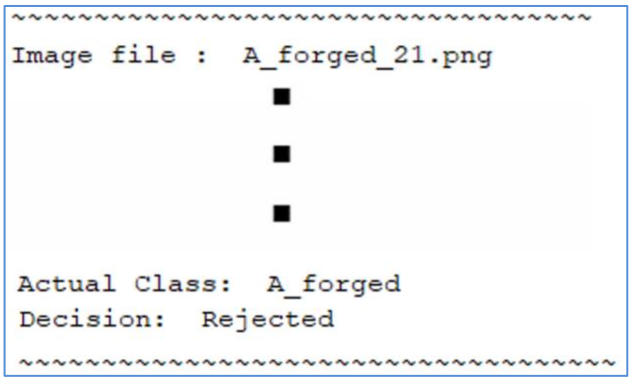
6.3.7 Classification.py

The learning of the system would be dependent heavily on a supervised base learning, which means that the training set images must have the class information in some form. Here I have renamed the training set images in such a way that the name will define the class it belongs to. The python code will first read all the images in the given directory and get the class of each based on its name. Since we have 25 authors each having forged and genuine signatures, we currently have a total of 50 classes.

For example, an image file has a name A_forged_21.png. Its actual class would be ‘A_forg’. This means that the signature image belongs to author A and is a forged one.

The classifier we have used is KNN which stands for k-nearest neighbours. It is basically a classification algorithm that means it assigns a class to a test image based on its feature values. The k-nearest neighbours’ algorithm uses Euclidian distance method to find the distance between two training points. Thus using Euclidian distance we find k nearest neighbouring training points of our test point based on its features and the class with maximum number of occurrences is taken as the decision class for that test image and is assigned to that image. If the decision class is genuine the image is ‘Accepted’ and otherwise ‘Rejected’.

Our designed KNN classification algorithm code also situated in this file. KNN stands for K-nearest neighbors. This classification gives us the decision class that our projects intended output would be.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Image file :  A_forged_21.png
                  ■

                  ■

                  ■

Actual Class:  A_forged
Decision:  Rejected
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 37: Normal features printed on console

6.3.8 Evaluation.py

For system to be useful for the society it needs to be accurate and has to work well. For our signature recognition and verification system, accuracy of the system is basically how correctly does our system recognizes a particular signature by giving us whom does it belongs to and whether it is forged or genuine.

The evaluation parameter used for measuring accuracy of the system is recognition rate. We find the FAR and FRR which stand for False Acceptance Rate and False Rejection Rate. The number of falsely accepted images over the total images is FAR and the number of falsely rejected images over the total images is FRR

In our experimentation, we find our accuracy to be highest with 85.28% at K = 22 in our KNN classifier. The table below shows the different recognition rates for different values for K.

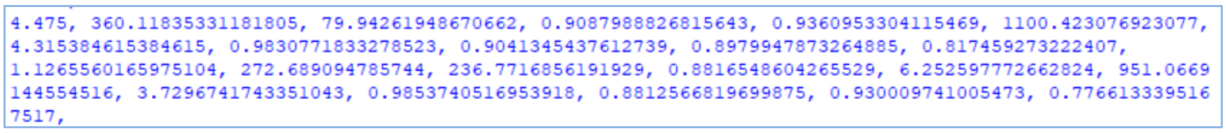
Table 2: Expermintatal Analysis

K	15	20	22	25	30
FRR	16.5%	15.2%	14.72%	16.1	18.5
FAR	6.2%	2.2%	0%	0%	0%
Accuracy	77.3%	82.6%	**85.28%**	83.9%	81.5%

We can also compare our system with an existing system which we retrieved from the web. This system uses a Convolutional Siamese network along with the constrastive loss function. They chose Euclidian distance as the distance metric for comparing the output feature vectors. The model achieved an accuracy of 73.34%. Deviations of 1-2% are possible as accuracy depends on the threshold. The threshold for the siamese network was computed by taking the average of True positive rate and True negative rate using ROC. The data used is same as the one used in our system described in this paper.

6.3.9 Signature_verifier.sql

The features extracted are to be stored in 2 dimensional variable here in the system named as ‘trainingFeatures’ and ‘testingFeatures’ where ea1ch row has features for each image. Now all the features are then collected in this 2 dimensional array to be later used during classification of the test images. The Feature Vector output is generated for some images and is shown in the figure 25.



```
4.475, 360.11835331181805, 79.94261948670662, 0.9087988826815643, 0.9360953304115469, 1100.423076923077,
4.315384615384615, 0.9830771833278523, 0.9041345437612739, 0.8979947873264885, 0.817459273222407,
1.1265560165975104, 272.689094785744, 236.7716856191929, 0.8816548604265529, 6.252597772662824, 951.0669
144554516, 3.7296741743351043, 0.9853740516953918, 0.8812566819699875, 0.930009741005473, 0.776613339516
7517,
```

Figure 38: Feature vectors in python console

This data doesn’t really help us understand which is what and also cannot be reused after the execution of code. They might want to be recalculated when the system is run again. To avoid that and to take the system directly to the testing stage, we make use of database. Here we use MySql database to be run using Apache through the XAMPP control panel. Following image shows how the database

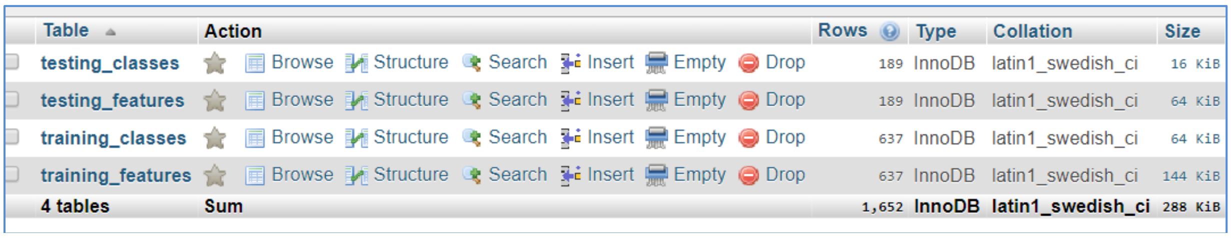
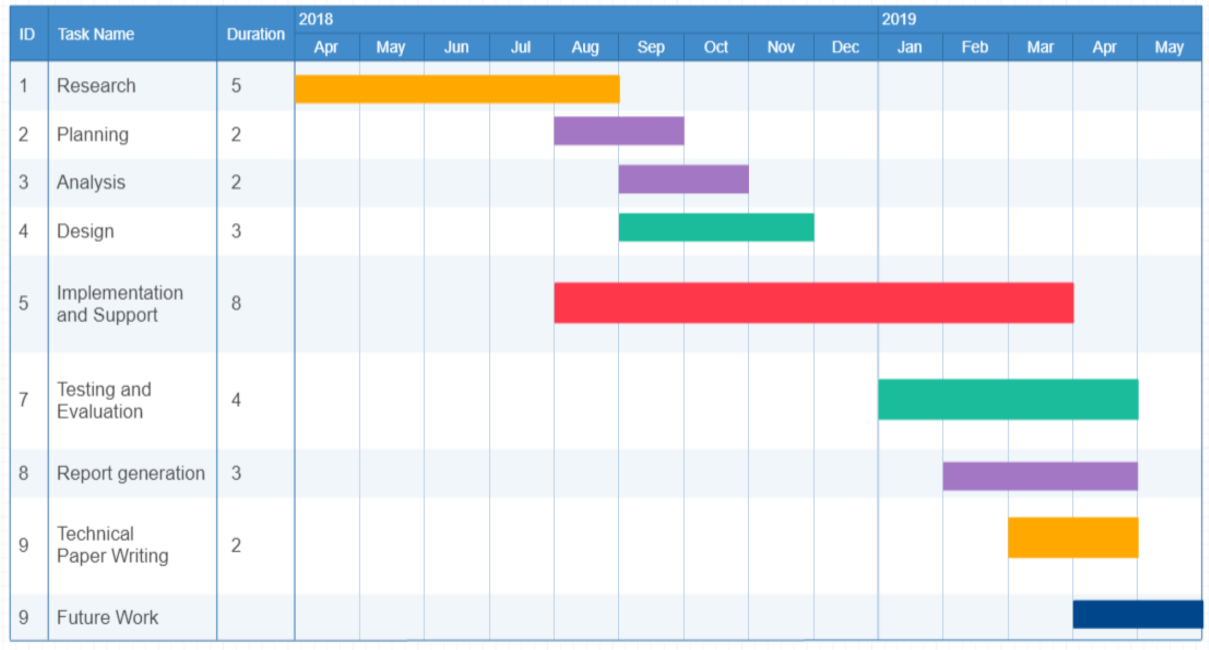


Table	Action							Rows	Type	Collation	Size
testing_classes	☆	Browse	Structure	Search	Insert	Empty	Drop	189	InnoDB	latin1_swedish_ci	16 KiB
testing_features	☆	Browse	Structure	Search	Insert	Empty	Drop	189	InnoDB	latin1_swedish_ci	64 KiB
training_classes	☆	Browse	Structure	Search	Insert	Empty	Drop	637	InnoDB	latin1_swedish_ci	64 KiB
training_features	☆	Browse	Structure	Search	Insert	Empty	Drop	637	InnoDB	latin1_swedish_ci	144 KiB
4 tables	Sum							1,652	InnoDB	latin1_swedish_ci	288 KiB

Figure 39: Tables in MySQL database

7 Matching of implementation with proposed plan

Following Gantt chart shows the action plan for the project that is done or is to be done in a monthly basis for different modules. For the amount of work done till the end of March, Technical paper writing and Implementation of the Project has finally has been completed. Testing and evaluation is under way and can be said that it is in a way done as well, as mentioned in the following Gantt chart. As shown in the above sections, we can say that the project work is around 99% done.



ID	Task Name	Duration	2018									2019				
			Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May
1	Research	5	🟧	🟧	🟧	🟧	🟧									
2	Planning	2					🟪	🟪								
3	Analysis	2						🟪	🟪							
4	Design	3						🟩	🟩	🟩						
5	Implementation and Support	8					🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥		
7	Testing and Evaluation	4										🟩	🟩	🟩	🟩	
8	Report generation	3											🟪	🟪	🟪	
9	Technical Paper Writing	2												🟧	🟧	
9	Future Work														🟦	🟦

Figure 40: Gantt Chart

8 References

[1] S. F. A. Zaidi and S. Mohammed, “Biometric Handwritten Signature Recognition,” 2007.

[2] D. Morocho, A. Morales, J. Fierrez, and R. Vera-Rodriguez, “Towards human-assisted signature recognition: Improving biometric systems through attribute-based recognition,” *ISBA 2016 - IEEE Int. Conf. Identity, Secur. Behav. Anal.*, 2016.

[3] S. A. Angadi, S. Gour, and G. Bhajantri, “Offline Signature Recognition System Using Radon Transform,” *2014 Fifth Int. Conf. Signal Image Process.*, pp. 56–61, 2014.

[4] S. Hangai, S. Yamanaka, and T. Hammamoto, “ON-LINE SIGNATURE VERIFICATION BASED ON ALTITUDE AND DIRECTION OF PEN MOVEMENT,” *Proc. 15th Int. Conf. Pattern Recognition. ICPR-2000*, vol. 3, no. l, pp. 479–482, 2000.

[5] I. V Anikin and E. S. Anisimova, “Handwritten signature recognition method based on fuzzy logic,” *2016 Dyn. Syst. Mech. Mach.*, 2016.

[6] E. Ozgunduz, T. Senturk, and M. E. Karsligil, “Off-line signature verification and recognition by support vector machine,” *13th Eur. Signal Process. Conf.*, no. 90, pp. 1–4, 2005.

[7] S. A. Angadi and S. Gour, “Euclidean Distance Based Offline Signature Recognition System Using Global and Local Wavelet Features,” *2014 Fifth Int. Conf. Signal Image Process.*, pp. 87–91, 2014.

[8] Ruangroj Sa-Ardship and K. Woraratpanya, “Offline Handwritten Signature Recognition Using Adaptive Variance Reduction,” *7th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE), Chiang Mai, Thail. Offline*, pp. 258–262, 2015.

[9] M. A. Djoudjai, Y. Chibani, and N. Abbas, “Offline signature identification using the histogram of symbolic representation,” *5th Int. Conf. Electr. Eng. - Boumerdes*, pp. 1–5, 2017.

[10] A. B. Jagtap and R. S. Hegadi, “Offline Handwritten Signature Recognition Based on Upper and Lower Envelope Using Eigen Values,” *Proc. - 2nd World Congr. Comput. Commun. Technol. WCCCT 2017*, pp. 223–226, 2017.

[11] T. Marušić, Ž. Marušić, and Ž. Šeremet, “Identification of authors of documents based on offline signature recognition,” *MIPRO*, no. May, pp. 25–29, 2015.

[12] S. L. Karanjkar and P. N. Vasambekar, “Signature Recognition on Bank cheques using ANN,” *IEEE Int. WIE Conf. Electr. Comput. Eng.*, no. December, 2016.

[13] G. S. Prakash and S. Sharma, “Computer Vision & Fuzzy Logic based Offline Signature Verification and Forgery Detection,” *IEEE Int. Conf. Comput. Intell. Comput. Res.*, 2014.

[14] M. S. Shirdhonkar and M. B. Kokare, “Document image retrieval using signature as query,” *2011 2nd Int. Conf. Comput. Commun. Technol. ICCCT-2011*, pp. 66–70, 2011.

[15] R. Sa-Ardship and K. Woraratpanya, “Offline Handwritten Signature Recognition Using Polar-Scale Normalization,” *8th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE), Yogyakarta, Indones.*, pp. 3–7, 2016.

[16] A. Piyush Shanker and A. N. Rajagopalan, “Off-line signature verification using DTW,” *Pattern Recognit. Lett.*, vol. 28, no. 12, pp. 1407–1414, 2007.

[17] Nancy and P. G. Goyal, “Signature Processing in Handwritten Bank Cheque Images,” *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 2, no. 5, pp. 1239–1243, 2014.

[18] A. T. Nasser and N. Dogru, “Signature recognition by using SIFT and SURF with SVM basic on RBF for voting online,” *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018–Janua, pp. 1–5, 2017.

[19] A. Kumar and K. Bhatia, “A Robust Offline Handwritten Signature Verification System Using Writer Independent Approach,” *3rd Int. Conf. Adv. Comput. Autom.*, 2017.

[20] H. Anand, “Enhanced Signature Verification and RECOGNITION USING MATLAB,” *Int. J. Innov. Res. Adv. Eng.*, vol. 1, no. 4, pp. 88–94, 2014.

[21] B. H. Shekar and R. K. Bharathi, “Eigen-signature: A robust and an efficient offline signature verification algorithm,” *Int. Conf. Recent Trends Inf. Technol. ICRTIT 2011*, pp. 134–138, 2011.

[22] A. R. Rahardika, “Global Features Selection for Dynamic Signature Verification,” *Int. Conf. Inf. Commun. Technol. Glob.*, pp. 348–354, 2018.

[23] T. Handhayani, A. R. Yohannis, and Lely Hiryanto, “Hand Signature and Handwriting Recognition as Identification of the Writer using Gray Level Co- Occurrence Matrix and Bootstrap,” *Intell. Syst. Conf.*, no. September, pp. 1103–1110, 2017.

[24] A. Beresneva, A. Epishkina, and D. Shingalova, “Handwritten Signature Attributes for its Verification,” pp. 1477–1480, 2018.

[25] M. P. Patil, Bryan Almeida, Niketa Chettiar, and Joyal Babu, “Offline Signature Recognition System using Histogram of Oriented Gradients,” *Int. Conf. Adv. Comput. Commun. Control*, 2017.

[26] M. M. Kumar and N. B. Puhan, “Offline signature verification using the trace transform,” *2014 IEEE Int. Adv. Comput. Conf.*, pp. 1066–1070, 2014.

[27] O. Miguel-Hurtado, L. Mengibar-Pozo, M. G. Lorenz, and J. Liu-Jimenez, “On-Line Signature Verification by Dynamic Time Warping and Gaussian Mixture Models,” *2007 41st Annu. IEEE Int. Carnahan Conf. Secur. Technol.*, pp. 23–29, 2007.

[28] M. Ferrer and J. Vargas, “Robustness of offline signature verification based on gray level features,” *IEEE Trans. Inf. FORENSICS Secur.*, vol. 7, no. 3, pp. 966–977, 2012.

[29] B. Erkmen, N. Kahraman, R. A. Vural, and T. Yildirim, “Conic section function neural network circuitry for offline signature recognition,” *IEEE Trans. Neural Networks*, vol. 21, no. 4, pp. 667–672, 2010.

[30] M. A. Ferrer, A. Morales, and J. F. Vargas, “Off-line signature verification using local patterns,” *2nd Natl. Conf. Telecommun.*, pp. 1–6, 2011.

[31] M. Tahir and M. U. Akram, “Online Signature Verification using Hybrid Features,” *Conf. Inf. Commun. Technol. Soc. Online*, pp. 11–16, 2018.

[32] M. Pal, S. Bhattacharyya, and T. Sarkar, “Euler number based feature extraction technique for Gender Discrimination from offline Hindi signature using SVM & BPNN classifier,” *2018 Emerg. Trends Electron. Devices Comput. Tech.*, pp. 1–6, 2004.

[33] W. Pan and G. Chen, “A Method of Off-line Signature Verification for Digital Forensics,” *12th Int. Conf. Nat. Comput. Fuzzy Syst. Knowl. Discov.*, pp. 488–493, 2016.

[34] M. A. Ferrer, J. B. Alonso, and C. M. Travieso, “Offline geometric parameters for automatic signature verification using fixed-point arithmetic,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 993–997, 2005.

[35] S. A. Farimani and M. V. Jahan, “An HMM for Online Signature Verification Based on Velocity and Hand Movement Directions,” *Iran. Jt. Congr. Fuzzy Intell. Syst.*, pp. 205–209, 2018.

[36] G. Pirlo and D. Impedovo, “Verification of static signatures by optical flow analysis,” *IEEE Trans. Human-Machine Syst.*, vol. 43, no. 5, pp. 499–505, 2013.

[37] A. Alaei, S. Pal, U. Pal, and M. Blumenstein, “An Efficient Signature Verification Method Based on an Interval Symbolic Representation and a Fuzzy Similarity Measure,” *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 10, pp. 2360–2372, 2017.

[38] D. S. Guru and H. N. Prakash, “Online Signature Verification and Recognition: An Approach based on Symbolic Representation.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1059–73, 2009.

[39] A. Sharma and S. Sundaram, “On the Exploration of Information from the DTW Cost Matrix for Online Signature Verification,” *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 611–624, 2018.

[40] G. Pirlo, V. Cuccovillo, M. Diaz-Cabrera, D. Impedovo, and P. Mignone, “Multidomain Verification of Dynamic Signatures Using Local Stability Analysis,” *IEEE Trans. Human-Machine Syst.*, vol. 45, no. 6, pp. 805–810, 2015.

[41] Python : https://www.python.org/

[42] Dataset source :- http://www.iapr-tc11.org/mediawiki/index.php?title=Datasets_List#Handwritten%20Documents