

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 :

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. PI – Performance Index

The Accuracy or the Performance index is calculated simply by the following formula:

$$PI = 100 - (FAR + FRR)$$

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). The system will take as input Author's ID and signature image and tell us whether the signature image should be 'Accepted' or 'Rejected'.

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 %.

Features extracted and used for comparison include total Euclidian distance D of the pen travelled, Speed V_x and V_y express the functions of time, Acceleration A_x and A_y , Total time taken T_k , Length-to-width ratio, Amount of zero speed in direction x and y directions N_{vx} and N_{vy} , Amount of zero acceleration in direction x and y directions N_{ax} and N_{ay}

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%, it can be said 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

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.

In paper [3], the system 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). Table 4.1 shows the 16 features in a tabular form. The number of neurons in the output layer is 5 which are equal to the number of pattern classes.

Table 4.1: 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 give us 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.

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 described 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 ($39 \times 2 + 4$) negative (forgery) examples are taken.

In paper [7], the described system 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.

In paper [8], the described system 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%.

In paper [9] researchers gives away 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, researchers 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], the researchers put forward and implement 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.

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 described 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.

In paper [17], author 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.

SIFT and a SURF algorithm which is used for enhanced offline signature recognition is used in paper [18]. This process, Bag-of-words 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 train 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 shown 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, Kurtosis, 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 used 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.

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 described 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, equi-spacing 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 described 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 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. The system's flow can also be seen in figure 10.

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

In order to solve the shortcomings of manual identification in technical accuracy and subjectivity, this paper [33] gave us 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] gave 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.

In paper [35], author 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 used. 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 shown 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

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 find outs and is also described in the given figure 5.1.

1. The flowchart block diagram would start with image acquisition where the images would be taken in as input. This would include training and testing images.
2. Then the raw images are given to the pre-processing where they would go through series of pre-processing stages such as RGB to Grey, thresholding, Boundary Box cropping and noise removal filters. We use Otsu's thresholding method which will be explained in later stages of how and why we use it.
3. In feature extraction stage different set of shape and size based are to be extracted.
4. The images are then converted to LBP images in the LBP conversion stage.
5. The texture based LBP features are extracted again but this time from the LBP images.
6. The generated feature set of all the images along with their respective classes would be given to a classification algorithm.
7. The classification stage has K-nearest neighbors algorithm using Euclidian distance based approach
8. Finally this stage would give out the class to which the input test image belongs to as output decision class.

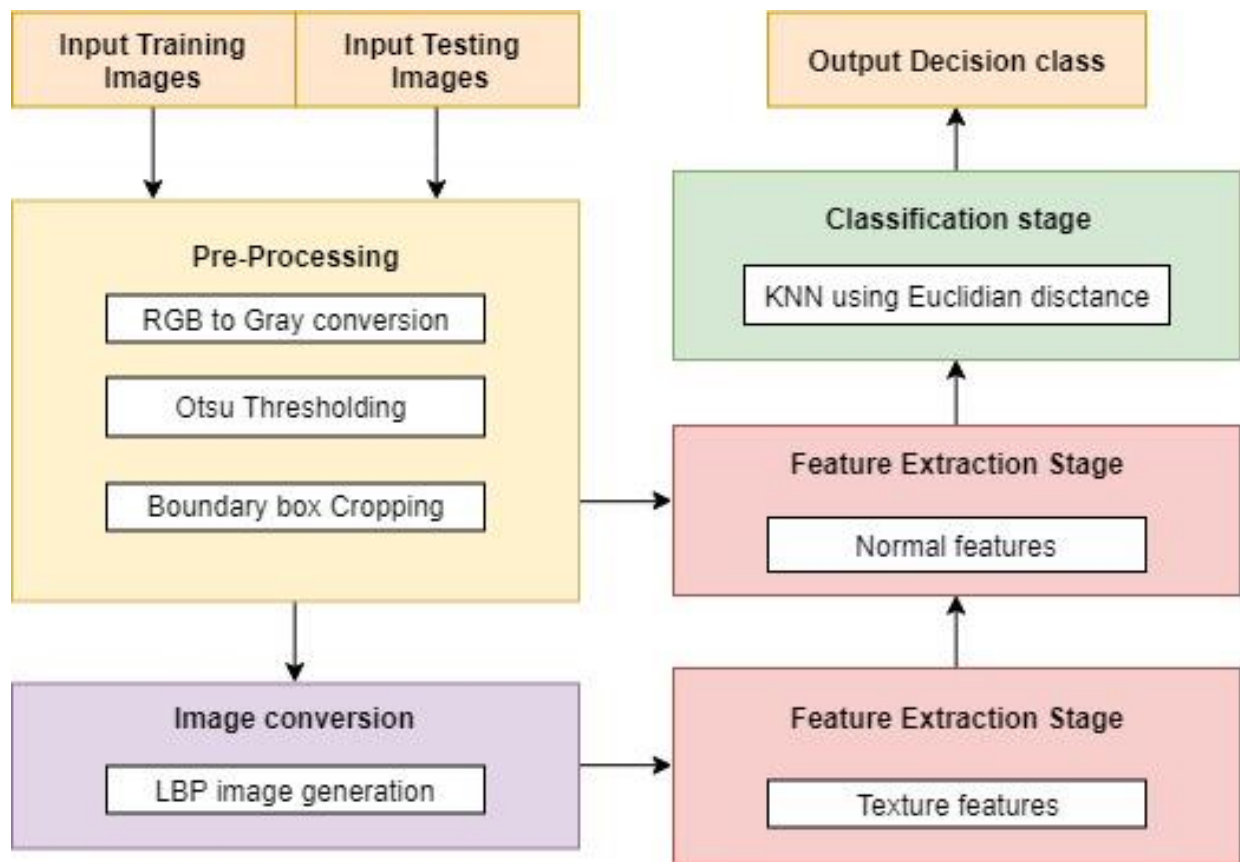


Figure 5.1: Block diagram of the proposed approach

5.2 Otsu's thresholding

5.2.1 Why use Otsu's thresholding?

In global thresholding, we used an arbitrary value for threshold value. 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.) Working of different thresholding methods is shown in figure 5.2

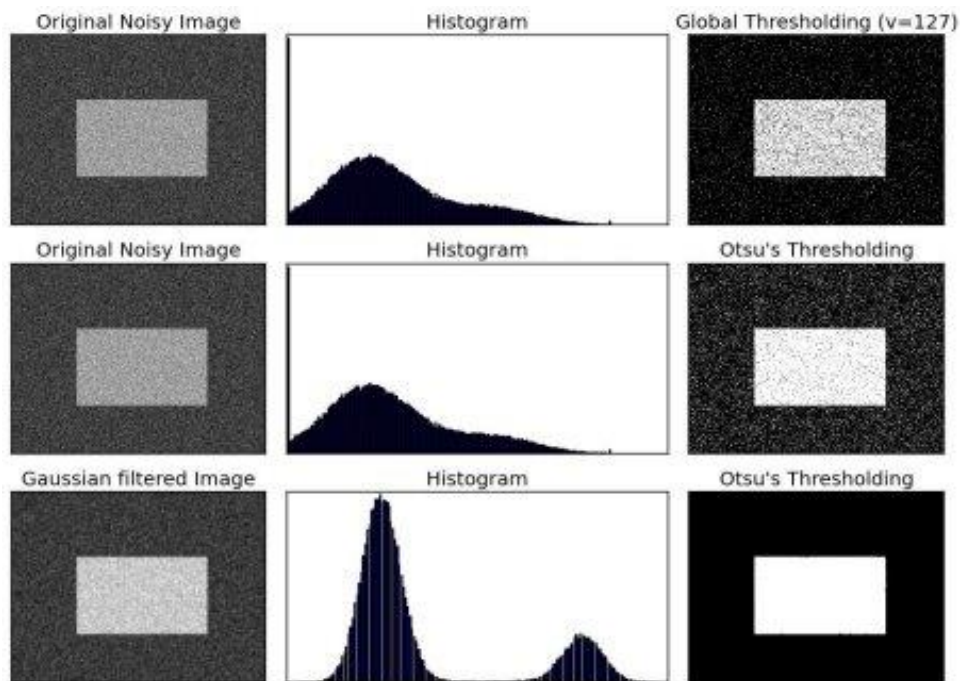


Figure 5.2: Different thresholding methods.

5.2.2 What is Otsu's thresholding?

Otsu's thresholding method corresponds to the linear discriminant criteria that assumes that the image consists of only object (foreground) and background, and the heterogeneity and diversity of the background is ignored [6]. Otsu set the threshold so as to try to minimize the overlapping of the class distributions [6]. Given this definition, the Otsu's method segments the image into two light and dark regions T_0 and T_1 , where region T_0 is a set of intensity level from 0 to t or in set notation $T_0 = \{0, 1, \dots, t\}$ and region $T_1 = \{t+1, \dots, l-1, l\}$ where t is the threshold value, l is the image maximum gray level (for instance 256). T_0 and T_1 can be assigned to object and background or vice versa (object not necessarily always occupies the light region).

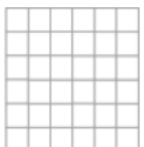
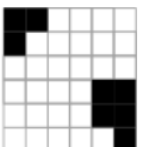
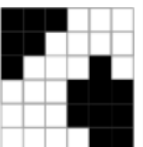

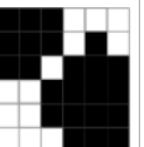

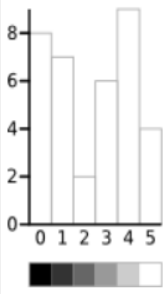
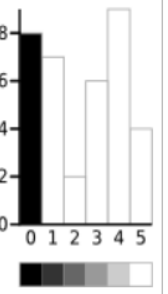
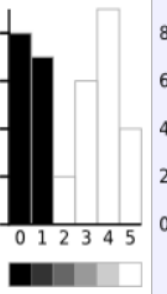
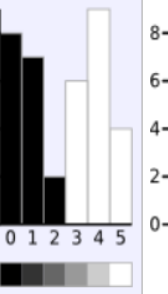
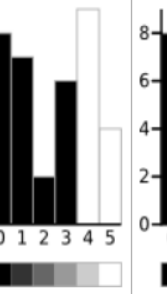

5.2.3 How Otsu's thresholding works?

Otsu's thresholding method scans all the possible thresholding values and calculates the minimum value for the pixel levels each side of the threshold. The goal is to find the threshold value with the minimum entropy for sum of foreground and background. Otsu's method determines the threshold value based on the statistical information of the image where for a chosen threshold value t the variance of clusters T_0 and T_1 can be computed. The optimal threshold value is calculated by minimizing the sum of the weighted group variances, where the weights are the probability of the respective groups. Algorithm of Otsu is given below with an example shown in table 5.1.

Algorithms

1. Compute histogram and probabilities of each intensity level
2. Set up initial weight $\omega_i(0)$ and mean $\mu_i(0)$
3. Step through all possible thresholds $t = 1, \dots$ maximum intensity
4. Update weights ω_i and means μ_i
5. Compute variances $\sigma_b^2(t)$
6. Desired threshold corresponds to the minimum within class variance σ_W^2

Table 5.1: An example of Otsu thresholding approach

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
						
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$	$W_b = 0.4722$	$W_b = 0.6389$	$W_b = 0.8889$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$	$M_b = 0.6471$	$M_b = 1.2609$	$M_b = 2.0313$
Variance, Background	$\sigma_b^2 = 0$	$\sigma_b^2 = 0$	$\sigma_b^2 = 0.2489$	$\sigma_b^2 = 0.4637$	$\sigma_b^2 = 1.4102$	$\sigma_b^2 = 2.5303$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$	$W_f = 0.5278$	$W_f = 0.3611$	$W_f = 0.1111$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$	$M_f = 3.8947$	$M_f = 4.3077$	$M_f = 5.000$
Variance, Foreground	$\sigma_f^2 = 3.1196$	$\sigma_f^2 = 1.9639$	$\sigma_f^2 = 0.7755$	$\sigma_f^2 = 0.5152$	$\sigma_f^2 = 0.2130$	$\sigma_f^2 = 0$
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$

5.3 Local Binary Patterns

5.3.1 What is LBP?

Local Binary Patterns, or LBPs for short, are a texture descriptor made popular by the work of Ojala et al. in their 2002 paper [43], 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.

5.3.2 How LBP works?

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 3×3 neighborhood of pixels just like shown below in figure 5.3:

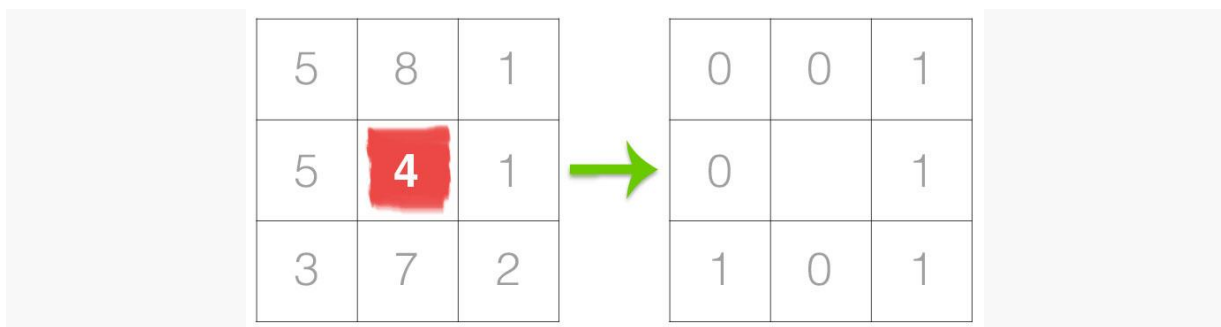


Figure 5.3: The first step in constructing a LBP.

In the above figure 14 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 3×3 neighborhood, 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 show in below figure 5.4.

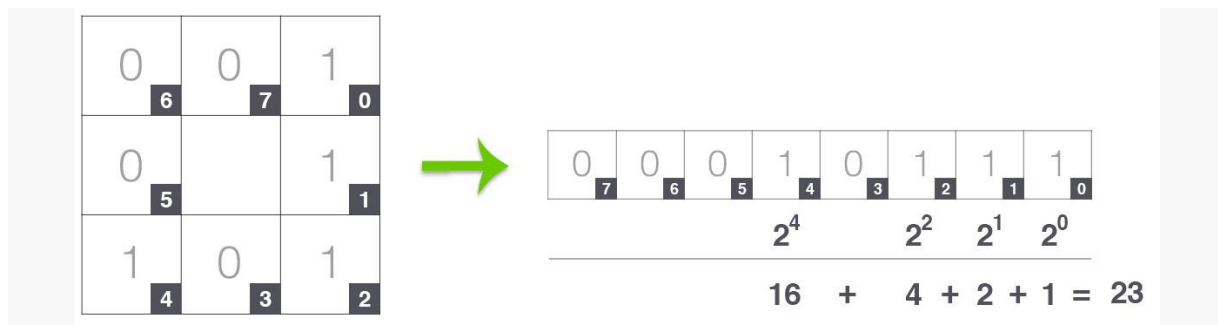


Figure 5.4: The second step in constructing a LBP.

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 5.5:

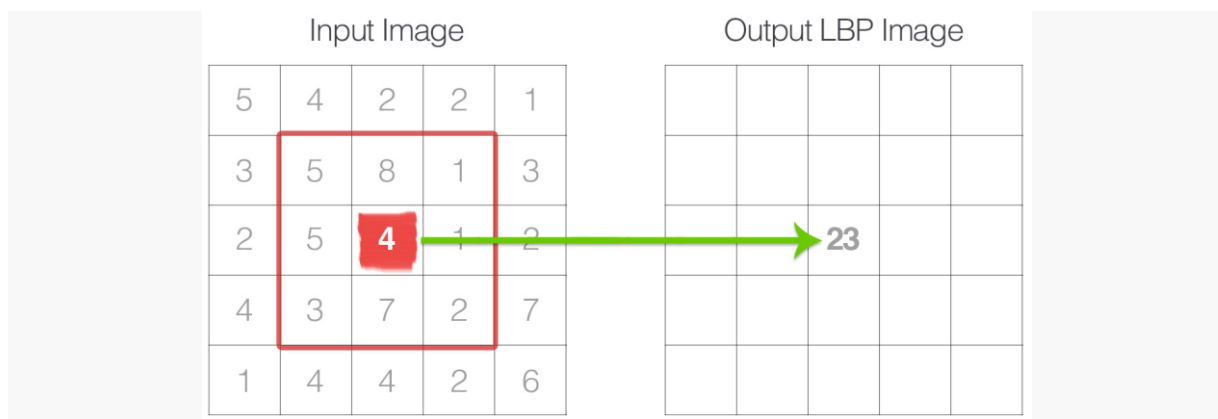


Figure 5.5: The third step in constructing a LBP.

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.

5.3.2 Why use LBP?

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!

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 [41] 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.

The main idea is that for every pixel of an image the LBP-code is calculated. The occurrence of each possible pattern in the image is kept up. Simply put: they add an extra level of rotation and grayscale invariance, hence we are to use when extracting texture feature vectors from images. . Another striking thing is the fact that, by taking only the pixels with uniform patterns, the background is also preserved. This is because the background pixels all have the same color (same gray value) and thus their patterns contain zero transitions.

The LBP images are darker in color and expose off their textures on a higher scale. This would thus allow our system to extract out our LBP based texture features from the image more easily. The features will have lesser amount of variance within the classes.

5.4 K-Nearest Neighbours

5.4.1 What is KNN?

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, show in figure 5.9:

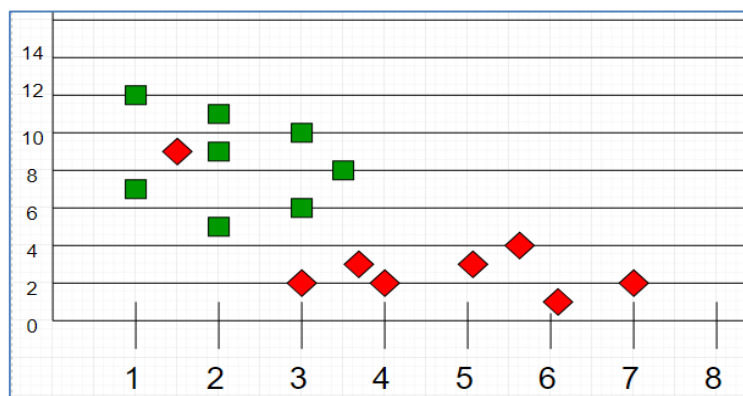


Figure 5.6: data points set 1

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' (Refer figure 5.10).

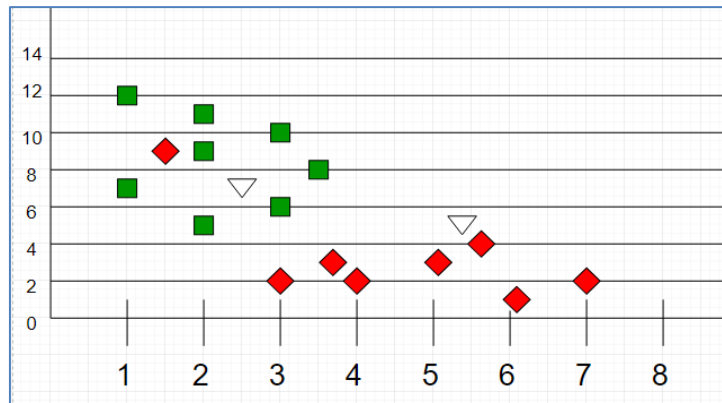


Figure 5.7: data points set 2

5.4.2 How KNN works?

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 :
 - Calculate Euclidean distance $d(arr[i], p)$.
 - Make set S of K smallest distances obtained. Each of these distances correspond to an already classified data point.
3. Return the majority label among S.

5.4.3 Why use KNN?

The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions. Therefore, you can use the KNN algorithm for applications that require high accuracy but that do not require a human-readable model.

The quality of the predictions depends on the distance measure. Therefore, the KNN algorithm is suitable for applications for which sufficient domain knowledge is available. This knowledge supports the selection of an appropriate measure. The KNN algorithm is a type of lazy learning, where the computation for the generation of the predictions is deferred until classification. Although this method increases the costs of computation compared to other algorithms, KNN is still the better choice for applications where predictions are not requested frequently but where accuracy is important.

5.5 Implementation tools and 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.

5.5.1 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.

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 rendererr

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.

5.5.2 Database using MySQL

In our proposed project we are going to deal with a lot of data, so to store and reuse the data from somewhere, we are going to need a database. We use here MySQL, which is the most popular Open Source Relational SQL Database Management System.

The features extracted are to be stored in 2 dimensional variable here in the system named as ‘trainingFeatures’ and ‘testingFeatures’ where each 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 5.11.

```
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 5.8: 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 the MySQL database. Here we use MySql database to be run using Apache through the XAMPP control panel. Following figure 5.12 shows the tables in the proposed system’s 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 5.9: Tables in MySQL database

5.5.3 Hardware

The following are the minimum hardware requirements for the system to run including the above software based requirements.

- OS: Windows 8 or above
- Processor: x64 processor of Any Intel or AMD or NVidia
- Disk:
 - 2 GB for Python 3.6 including required packages
 - 3 MB for the proposed system files
 - 115 MB for signature image dataset
- RAM: 4 GB

6 Implementation work

6.1 Dataset

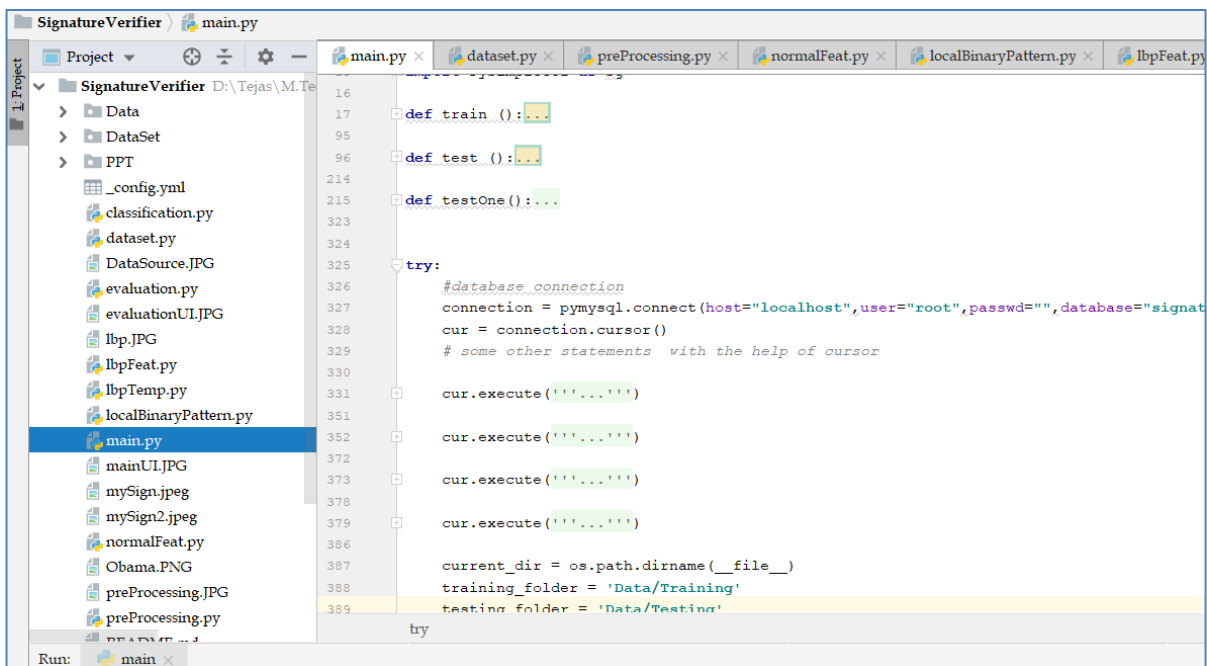
The work also includes dataset preparation include downloading more and more of the online available datasets that people around the world generate and upload for others to use. We have thus downloaded the dataset from the Web. CEDAR signature database [44] <https://cedar.buffalo.edu/Databases/index.html> contains signatures of 55 signers belonging to various cultural and professions. Each of these signers signed 24 genuine signatures 20 minutes apart. Each of the forgers tried to emulate the signatures of 3 persons, 8 times each, to produce 24 forged signatures for each of the genuine signers. Hence the dataset comprise $55 \times 24 = 1,320$ genuine signatures as well as 1,320 forged signatures. The signature images in this dataset are available in gray scale mode.

The signature verification system would need a set of over 1000 image dataset having handwritten signatures from different authors, which will test our system enough and thus we have prepared a subset of the vast dataset and kept aside directory of folders of images which contains total 26 Authors, all having genuine as well as forged signatures and thus 52 different classes.

The dataset is divided in an approximate ratio of 4:1 for preparing training and testing data respectively. Therefore 988 training images and 260 testing images kept in two different folders combine to a total of 1248 images. Folder structure is shown below in figure 6.1

6.2 Development

Implementing with python in done which includes total of 8 python.py files and an SQL file created in PyCharm that can be seen in below given figure 6.1.



6.2.1 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. Refer following figure 6.3 for the menu buttons and figure 6.4 for corresponding functions that they call

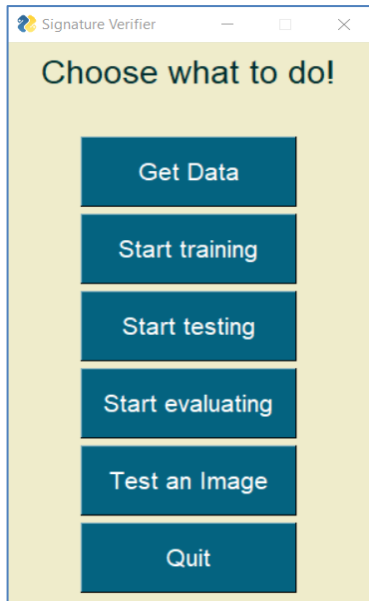


Figure 6.2: Main menu GUI

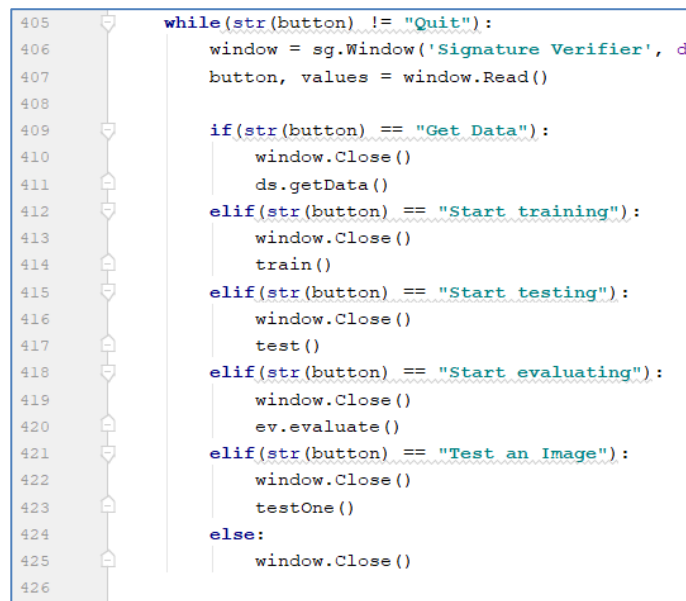


Figure 6.3: Menu functions

Following buttons are displayed on the main menu screen:

- Get data: This button will call the 'getData' function from Dataset.py which is used for analyzing and organizing our image dataset.
- Start Training: This function goes through all other functions of pre-processing, feature extraction of all the training images and also inserts the feature vector and class data in to our database.
- Start Testing: This function goes through all other functions of pre-processing, feature extraction and also classification of all the testing images and also inserts the feature vector and class data in to our database.
- Start Evaluation: This button will evaluate our system and find the accuracy.
- Test an Image: Clicking on this button will display a popup form (shown in below figure 6.5) and allow user to browse through his computer and perform testing over a single image.

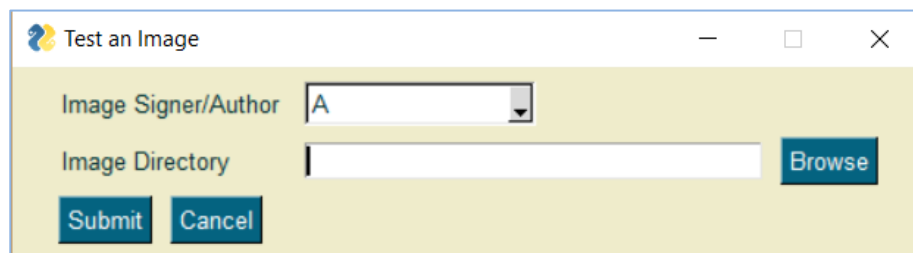


Figure 6.4: Test An image UI

- Quit: This will exit the system and end execution

6.2.2 Dataset.py

This file has a 'getData()' function that writes a 'dataAnalysis.txt' where it gives an analysis of all the data and the count of all the dataset. This is printed in a separate file called as shown in figure 6.6 and also on UI popup as shown in figure 6.7

The images of our dataset are renamed in the correct naming convention, copied and organized to a different folder, from where our system will use. For instance: 'xyz.png' is renamed to 'A_orig_7.png'. Here 'A' is the Author, 'orig' means that the image is genuine signed by author, 'A' and 7 is just a number appended to keep the image name unique within the folder. This name is also used as a primary key of all the tables in our database.

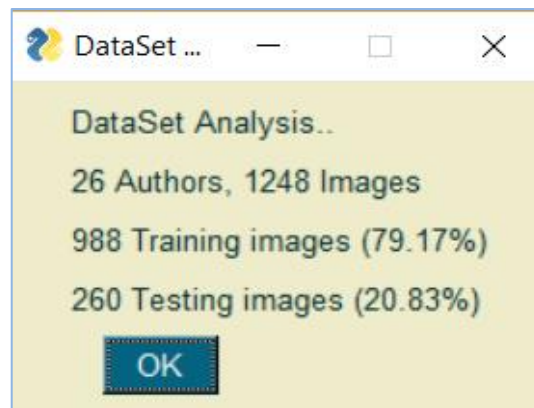


Figure 6.5: DataSet analysis popup UI

6.2.3 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 figure 6.9 how the threshold must vary for different images having different histograms.

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 figure 6.10 shows the same.

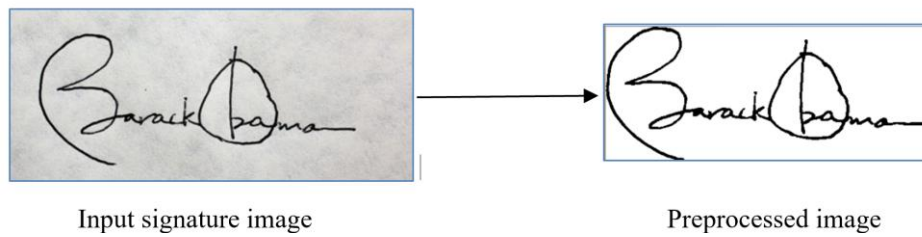


Figure 6.6: Preprocessing signature image

6.2.4 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 figure 6.11 shows the PySimpleGui progress bar for our training period for over 1000 images.

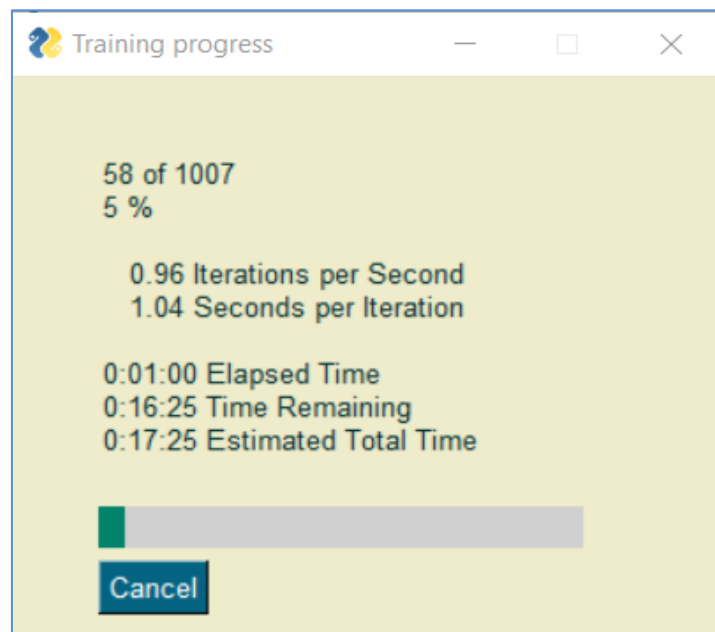


Figure 6.7: Progress meter bar GUI while training and testing

The normal shape based features that are extracted from the pre-processed image are given below and are also shown on the below figure 6.12:

```

    ~~~ 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 6.8: Normal features printed on console

1. Aspect Ratio:

The ratio of the width to the height of an image or screen. This gives us information of how long or broad the person's signature would be. For example, if a graphic has an aspect ratio of 2:1, it means that the width of the signature is twice as large as the height. When resizing graphics, maintaining the aspect ratio is important to avoid stretching.

AR = width of the image / height of the image

2. Center of Gravity - X:

The Center of Gravity or Center of Mass statistic calculates where the Center of Mass of the image lies.

The COG of X is calculated by:

COG_X = Sum of all x-coord of black pixels / total number of black pixels

3. Center of Gravity - Y:

The signature object's concentration within the image can be found out by Center of Gravity.

The COG of Y is calculated by:

COG_Y = Sum of all y-coord of black pixels / total number of black pixels

4. Baseline Shift:

This depicts which side (left or right) is the signature weighted more and by how much. For eg: person who signs from left bottom corner to right upper corner would have a slant shape of the signature and would give us a high positive shift

Baseline shift = Right half's COG_Y - Left half's COG_Y.

5. Energy:

Energy is used to describe a measure of "information" when formulating an operation under a probability framework such as MAP (maximum a priori) estimation in conjunction with Markov Random Fields. Sometimes the energy can be a negative measure to be minimised and sometimes it is a positive measure to be maximized.

$$Energy = \sum_{i,j=0}^{N-1} (P_{ij})^2$$

6. Dissimilarity:

The weights with which GLCM probabilities are multiplied increase linearly away from the diagonal. Instead of weights increasing exponentially (0, 1, 4, 9, etc.) as one moves away from the diagonal as Contrast did, the dissimilarity weights increase linearly (0, 1, 2, 3 etc.).

$$\text{Dissimilarity} = \sum_{i,j=0}^{N-1} P_{i,j} |i - j|$$

7. Haralick:

Whether considering the intensity or grayscale values of the image or various dimensions of color, the co-occurrence matrix can measure the texture of the image. Because co-occurrence matrices are typically large and sparse, various metrics of the matrix are often taken to get a more useful set of features.

These are 13 texture features, based on the adjacency matrix (the adjacency matrix stores in position (i,j) the number of times that a pixel takes the value i next to a pixel with the value j . Given different ways to define next to, you obtain slightly different variations of the features. Standard practice is to average them out across the directions to get some rotational invariance. They can be computed for 2-D or 3-D images and are available in the `mahotas.features.haralick` module.

```
textures = mt.features.haralick(img)
ht_mean = textures.mean(axis=0)
```

8. Kurtosis:

Kurtosis is a measure of the combined weight of a distribution's tails relative to the center of the distribution. Kurtosis somehow detects if a distribution is flat or peaky, and later was associated to perceptual aspects of sparse coding. It is often considered as a measure a sparsity, and used in early deconvolution methods.

It is recommended to interpret the kurtosis values in combination with noise and resolution measurement. High kurtosis values should go hand in hand with low noise and low resolution. In this case the denoising algorithms are very relentless to low frequencies, but at the cost of losing the ability to resolve fine structures and textures. If the kurtosis values are low, there are two possibilities:

- clearly visible noise and a good result in resolution could be a indicator for sharpening algorithms
- if there is low noise and a good result in resolution this would be the best case

$$kurtosis = \frac{m_4}{m_2^2} = \frac{\left(\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4\right)}{\left(\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2\right)^2}$$

6.2.5 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 a 3x3 mask the 8 pixel neighbourhood surrounding a centre pixel. We start from upper left corner, and threshold the surrounding 8 pixel with the centre pixel. The one pixel having higher intensity than the centre pixel is set to 0 and one lesser than the centre pixel is set to 1. This mask is then read in a clockwise manner to construct a set of 8 binary digits. Taking the 8 bit binary number, we convert it into a decimal representation. This decimal number is nothing but our LBP value which is set in our LBP image with the same width and height as the original image for the same (x,y) pixel. The mask is then moved ahead in the image and the process repeats. We can refer the above image which shows the steps with an example. The below figure 6.13 shows an original image and its LBP image form.

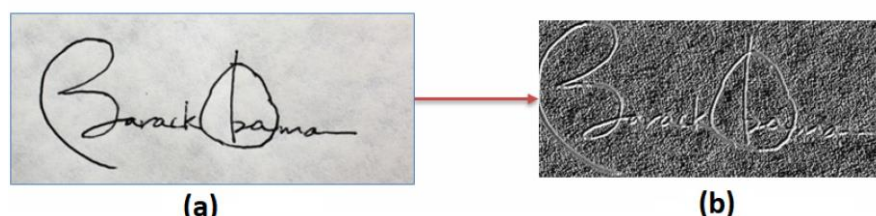


Figure 6.9: LBP image conversion:
(a) Grey image; (b) LBP image

To obtain better results in this, we perform variations in the LBP algorithm. We can keep the starting pixel as any of the 8 neighbor pixels for both clockwise and anti-clockwise reading of the binarized number. This gives us 16 different variants of LBP conversion that can be tested in our system later for which the accuracy is highest.

Thus we will have following 16 different variants for our LBP implementation based on the starting point or the leftmost pixel as starting for reading

- TL – Top Left : $x-1, y-1$
- T – Top : $x, y-1$
- TR – Top Right : $x+1, y-1$
- R – Right : $x+1, y$
- BR – Bottom Right : $x+1, y+1$
- B – Bottom : $x, y+1$
- BL – Bottom Left : $x-1, y+1$
- L – Left : $x-1, y$

$x-1, y-1$	$x, y-1$	$x+1, y-1$
$x-1, y$	x, y	$x+1, y$
$x-1, y+1$	$x, y+1$	$x+1, y+1$

Figure 6.10: 8-neighbour pixels

6.2.6 LbpFeat.py

The LBP texture based features that are extracted from the LBP image are printed on the PyCharm console as shown in figure 6.16 and are given below:

```
~~~ 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 6.11: LBP texture features printed on console

1. Contrast:

The measure of the intensity contrast between a pixel and its neighbor over the whole image. To emphasize a large amount of contrast, create weights so that the calculation results in a larger figure when there is great contrast between adjacent pixels. Values on the GLCM diagonal show no contrast, and contrast increases away from the diagonal. So, create a weight that increases as distance from the diagonal increases.

$$\text{Contrast} = \sum_{i,j=0}^{N-1} P_{i,j} (i-j)^2$$

2. 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. Gives the global weight of the image object within the image.

For instance a person using more of the pen's ink to sign will cover more of the white background to be colored black and will have a higher Normalized area.

- pixel_area = img.sum()
- normalised_area = (pixel_area)/(img.size)

3. Homogeneity:

Dissimilarity and Contrast result in larger numbers for more windows showing more contrast. If weights decrease away from the diagonal, the calculated texture measure will be larger for windows with little contrast. Homogeneity weights values by the inverse of the Contrast weight, with weights decreasing exponentially away from the diagonal.

$$\text{Homogeneity} = \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1 + (i-j)^2}$$

4. **Energy:** the Energy measure of LBP image as described in section 6.2.4
5. **Dissimilarity:** Dissimilarity measure of LBP image as described in section 6.2.4
6. **Haralick:** Mean of 13 Haralick feature of LBP image measure of LBP image as described in section 6.2.4
7. **Skewness:** Asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. It is a measure of (lack of) symmetry [1]. For instance, if the skewness is negative, the histogram is negatively skewed. That means its left tail is longer or fatter than its right one. Therefore, the frequency over the darker intensities (closer to zero) is wider spread (less concentrated, but not necessarily less frequent than the right tail!). The positive skewness is the opposite.
8. **Kurtosis:** Kurtosis measure of LBP image measure of LBP image as described in section 6.2.4

6.2.7 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 and is also printed on console as shown in figure 6.17.


```

('F_orig_23', 'F_orig', 'A_forg', 'Rejected')
('F_orig_24', 'F_orig', 'F_orig', 'Accepted')
('G_forg_20', 'G_forg', 'G_forg', 'Rejected')
('G_forg_21', 'G_forg', 'G_forg', 'Rejected')
('G_forg_22', 'G_forg', 'G_forg', 'Rejected')
('G_forg_23', 'G_forg', 'G_orig', 'Accepted')
('G_forg_24', 'G_forg', 'G_orig', 'Accepted')
('G_orig_20', 'G_orig', 'Z_forg', 'Rejected')
('G_orig_21', 'G_orig', 'G_orig', 'Accepted')
('G_orig_22', 'G_orig', 'G_orig', 'Accepted')
('G_orig_23', 'G_orig', 'G_orig', 'Accepted')

```

Figure 6.12: Classification printed on console

6.2.8 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

We also will have to perform testing and evaluations for the following 16 different variants for our LBP implementation based on the starting point or the leftmost pixel as starting for reading and the results are as shown in below figure 6.1.

Table 6.1: Expermintatal Analysis of different LBP Variants

Variant	Clockwise	Anti-Clockwise
TL – Top Left	80.46	80.46
T – Top	83.07	80.46
TR – Top Right	81.29	83.67
R – Right	82.76	81.92
BR – Bottom Right	81.23	82.31
B – Bottom	80.08	81.92
BL – Bottom Left	84.67	81.54
L – Left	80.46	80.84

Thus based on our above data we can make a line graph which is given below in figure 6.18. The line graph depicts the variants on horizontal axis and accuracy on the Y-axis, the variants are labelled as above table shows and two different lines are put in the same graph for comparative study of both. We can infer that the anti-clockwise reading shows better results and highest accuracy

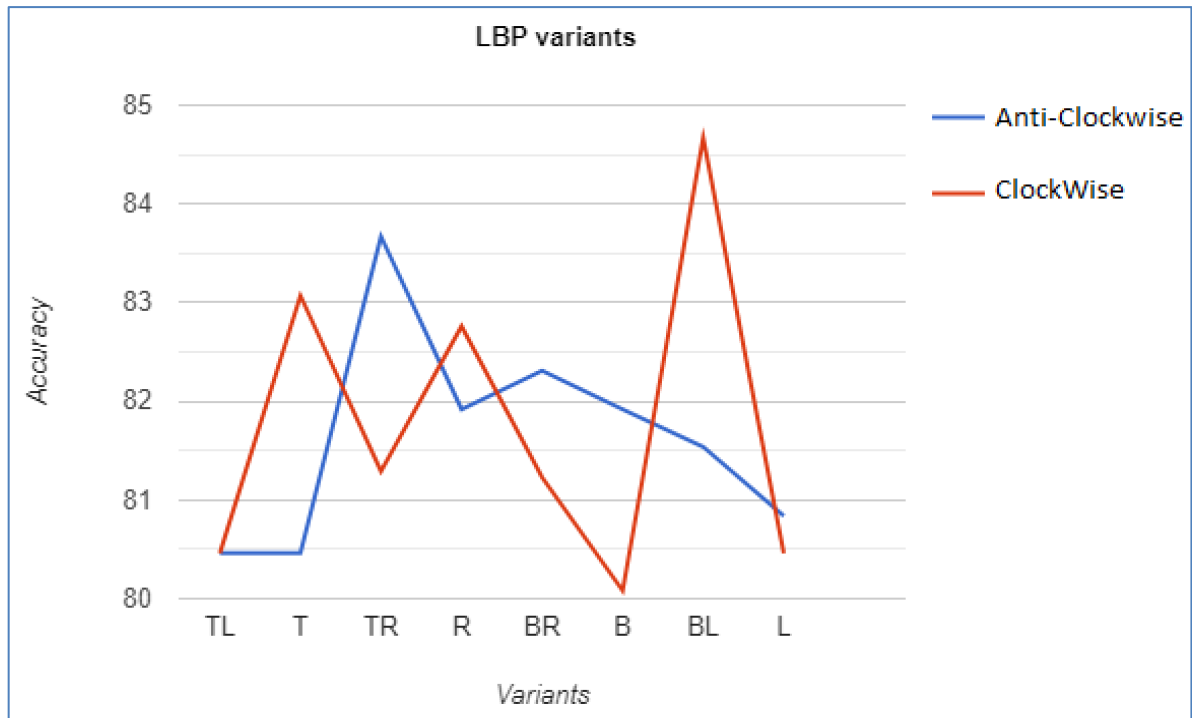


Figure 6.13: Line Graph of all the LBP Variants shown in table 6.1

In our experimentation, we find our accuracy to be highest with **84.67 %** at $K = 22$ in our KNN classifier and can be seen on the popup in figure 6.19. The table 6.2 below shows the different recognition rates for different values for K . The accuracy of the system is measured by the following formula.

$$\text{Accuracy} = 100 - (\text{FAR} + \text{FRR})$$

Table 6.2: Expermintatal Analysis with different values of K

K	10	20	22	30	40
FRR	20.77 %	15.2 %	15.33 %	15.38 %	17.69 %
FAR	1.92 %	2.2 %	0.0 %	2.69 %	2.69 %
Accuracy	77.31 %	83.46 %	84.67 %	81.92 %	79.62 %

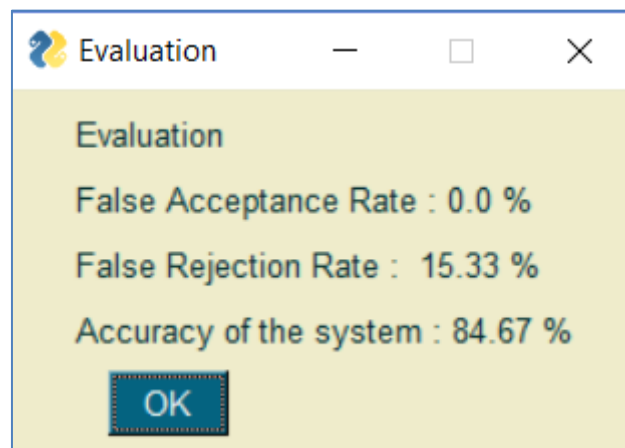


Figure 6.14: Evaluation popup UI

7 Comparative study

We can also compare our system with an existing system which we retrieved from the github [43] which is the design of the system described in paper [42]. Github, a software hosting website which has brought millions of developers together to discover, share, and build better software

7.1 Dataset

The data, designer have used is a subset of the same dataset as the one used in this system described in this report and thus we could compare our systems. The CEDAR signature dataset is one of the benchmark datasets for signature verification. The naming conventions of this system is obviously different from our system. This dataset is read and used for training and testing within this system.

7.2 Preprocessing

For preprocessing, the images from the dataset were converted to grayscale first, which means the 3 dimensional image ae first converted to a dimensional image. The image is then inverted and scaled down to 0 or up to 255 depending on whether the pixel value was below or above 50(this was done to remove any background specks and proved to simple yet effective technique for this task). Image tensor sizes of 225x155 were fed into the model. Images were grouped in pairs of genuine and forged images, where the label was 1 if both were genuine and of the same writer and 0 otherwise. 13500 image pairs of each label where chosen, 15% of which were used for testing.

7.3 Model used

The model used by the designer is a Convolutional Siamese network. Siamese neural network is a class of neural network architectures that contain two or more *identical* subnetworks. Identical here means they have the same configuration with the same parameters and weights. Parameter updating is mirrored across both subnetworks.

Along with the Siamese neural network, the system makes use of a Constrastive loss function This network uses Euclidian distance as the distance metric for comparing the output feature vectors, similar to how we used it in our K nearest neighbor classification algorithm. The block diagram of the same can be seen in figure 7.1

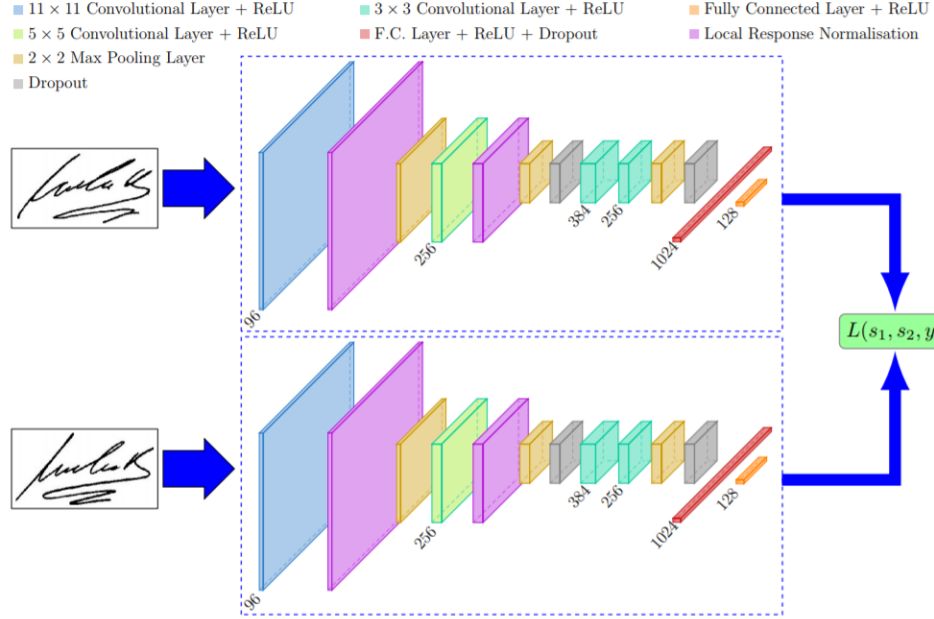


Figure 7.1: Architecture of existing project

7.4 Comparison & Discussion

For comparing the results of this system with our proposed system, we make use of the same dataset of images for running this system. The execution of this existing system is done in python and in the PyCharm too as we used for our proposed project and the same can be seen in the given figure below 7.2. The model achieved an accuracy of 77.28% on our signature. 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.

As shown in table 7.1, the proposed system takes more amount of time than existing system because of the LBP conversion which is time consuming. Our system makes us of feature vector given to KNN classifier on the other hand this system has a Siamese network with threshold and the feature distinguishing takes place with Euclidian distance very similar to our KNN. The Accuracy is lower than our system, thus our system can be said to be slightly better when considering accuracy. But choosing a system does not just depends on the systems efficiency, there are many other aspects that must be considered while designing a system such as application of the system, time and space requirements of the system, complexity of the algorithms and cost of setting up the application.

Table 7.1: Comparison table based on testing time and average accuracy

System	Algorithm	Dataset	Testing Time	Avg Accuracy
Exisiting Approach	Convolutional Siamese network	CEDAR 1248 images	10.83 minutes 2.5 sec per image	77.28 %
Proposed Approach	LBP features with KNN	CEDAR 1248 images	21.66 minutes 5 sec per image	81.69 %

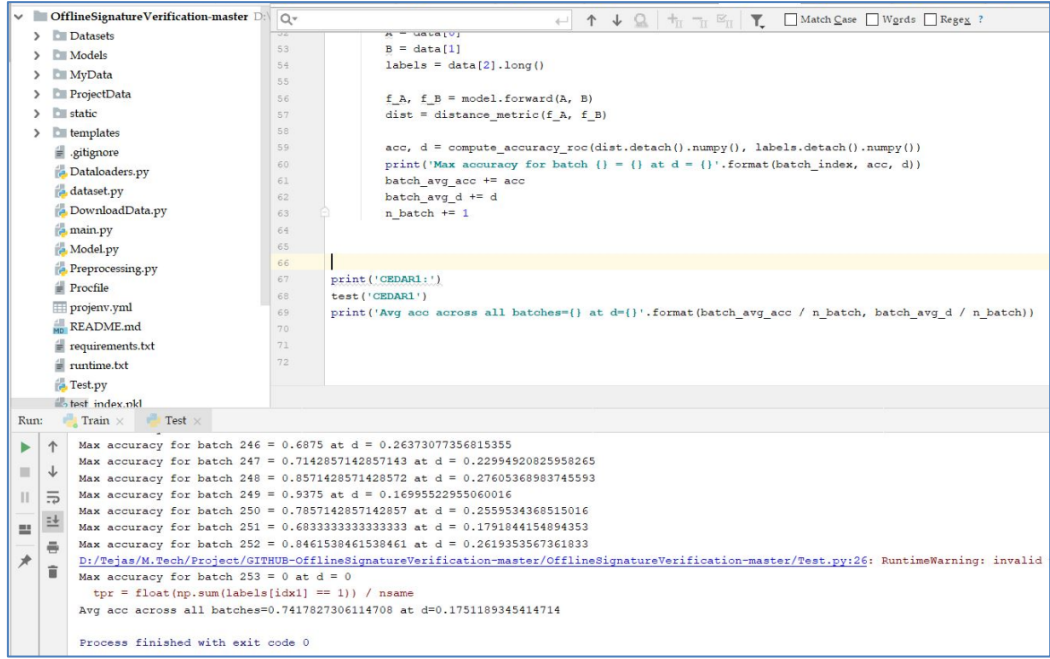


Figure 7.2: Running the Existing project

Now we compare the two systems based on the 16 feature set of our proposed system. We will make certain changes in the existing project to make the system extract the 12 distinct features from the image shown in figure 7.3 and the data produced is shown in the table 7.2

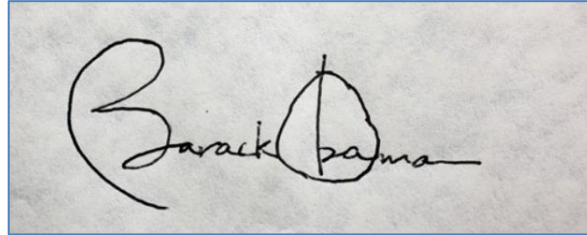


Figure 7.3: Sample signature image for comparison

Table 7.2: Comparison table based on 12 features

Features	Proposed	Existing
Aspect Ratio	2.435406699	1.419354839
X_COG	257.0992043	119.1404977
Y_COG	103.0727925	108.7890905
Baseline shift	0.986136187	5.276752452
Contrast	7373.77901	10099.03668
Normalized area	36.26455852	0.269882698
Homogeneity	0.169261182	0.844692328
Energy	0.917838035	0.689751126
Dissimilarity	56.92684512	39.6040654
Haralick	3225.801864	4928.362667
Skewness	0.252952231	1.183689407
Kurtosis	-1.364966573	-0.062512875

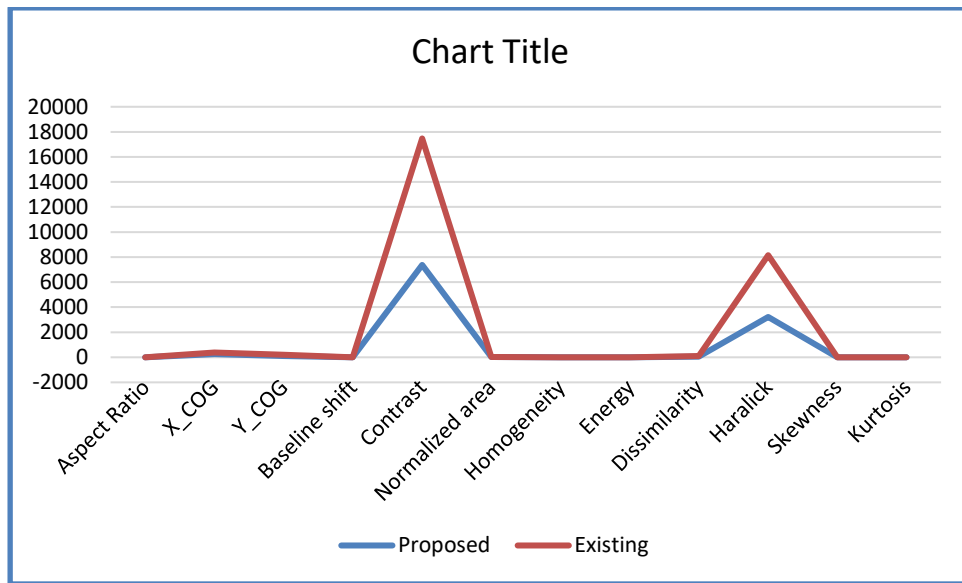


Figure 7.5: Simple line graph of comparing features

The above graph in figure 7.5 shows differences in the features of the two systems based on their absolute values. This does not tell us much about the scale and the difference as the numbers are scattered in a range from -4 all the way up to 7000. So we make use of a 100% stacked columns bar graph which will help us understand the differences in the numbers. By making one parameter 100 and scaling down the other based on the same ratio. This intelligent graph is shown below in figure 7.3.

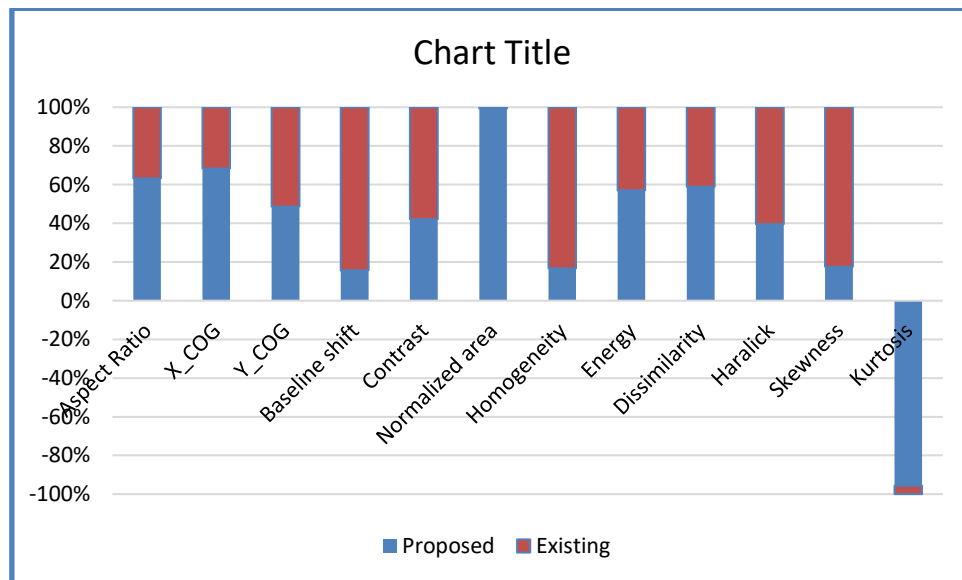


Figure 7.3: 100% stacked columns bar graph of comparing features

8 Conclusion & Future Work

We can conclude by stating that the remarkable work done by the researchers in over 40 different papers have led and motivated us to design a system which is clearly described in this report. We provide full credits to all the authors of these papers for sharing their knowledge and their astonishing work they have put forward that led us to our study. By looking at them and comparing, we can say which one is good by merely looking at their recognition rates. But choosing a system does not just depends on the systems efficiency, there are many other aspects that must be considered while designing a system such as application of the system, time requirements, space requirements of the system, complexity of the algorithms and cost of setting up the application and so on.

The research allowed us to learn more and more about the domain and the problem of signature recognition and verification. The research work of over 40 different research work on the problem of signature recognition/verification was done for this paper which allowed to prepare and plan for a signature verification system that was designed and presented in this paper.

The system's accuracy which comes out can still be brought up by adding more amount of research and combining the existing techniques with our work. The system's does take time to train which can also be improvised but testing phase is much quicker. The Local binary Pattern algorithm allows to extract texture based features which has shown great results in similar systems such as face recognition. There are still many approaches out there that may be better than what are listed here and thus the work does not end here.

9 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. 1, 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. Puan, “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] T. Ojala, M. Pietikainen and D. Harwood, "A comparative study of texture measures with classification based on feature distributions" *Pattern Recognition* vol. 29, 1996.
- [42] Sounak Deya, Anjan Duttaa J. Ignacio Toledo, Suman K.Ghosha, Josep Lladós, Umapada Palb, "SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification" *Corr* (2017). arXiv:1707.02131
- [43] Aftaab Zia, Github Existing Project :
<https://github.com/Aftaab99/OfflineSignatureVerification>
- [44] CEDAR image Dataset Source :
<https://cedar.buffalo.edu/Databases/index.html>

10. Appendix

Paper Published

Tejas Jadhav, “Handwritten Signature Verification using Local Binary Pattern Features and KNN”, International Research Journal of Engineering and Technology (IRJET), Vol.06, Issue.04, pp.579-586, April 2019

Indexing

- An ISO 9001-2008 Certified Journal
- Certificate of indexing 2018
- A UGC Recognized Journal
- Google Scholar
- Academia Database
- INNO SPACE
- Cite Factor
- Research Gate
- SJournals Index
- Electronic Journals Library
- Computer Science Directory

Impact Factor: 7211 as of 2018

E - ISSN: 2395-0056

P - ISSN: 2395-0072

Published Paper: <https://www.irjet.net/archives/V6/i4/IRJET-V6I4130.pdf>