



## Research Article

# Writer Independent Offline Signature Verification System using Global and Local Geometric Features

Iman M. Yaseen<sup>1,\*</sup> Computer Science  
University of Duhok

Duhok, Kurdistan Region - Iraq

[Iman.Yassen@UoD.ac](mailto:Iman.Yassen@UoD.ac)Hasan S.M. Al-Khaffaf<sup>2</sup> Computer Science  
University of Duhok

Duhok, Kurdistan Region - Iraq

[Hasan.Salim@UoD.ac](mailto:Hasan.Salim@UoD.ac)

## ARTICLE INFO

Article History

Received: 07/01/2024

Accepted: 31/03/2024

Published: 30/6/2024

This is an open-access article under the CC BY 4.0 license:

<http://creativecommons.org/licenses/by/4.0/>

## ABSTRACT

The objective of this paper is to propose an offline signature verification system (OSVS) designed for writer-independent applications. The system shall differentiate between original and forged signatures. The system encompasses key stages such as pre-processing, feature extraction, and model training and testing, employing the Support Vector Machine algorithm for classification. One challenge in creating a good OSVS is to find proper signature features to be used in the training/classification phases. In this research, global and local features are utilized. This includes signature area, mean, standard deviation, perimeter, number of connected components, number of vertical and horizontal edges, number of end points, number of branch points, and number of lines. The contributions of this paper are on several aspects of the offline signature verification process. Investigation in this study include data pre-processing techniques (normalization vs. standardization), kernel selection (Poly vs. RBF), dataset distribution for training and testing (80%-20% vs. 5-fold), and variations of the C and gamma parameters (C=1, 10, 100 and gamma=1, 10, 100). Improving the recognition rate involves removing of features with little or bad effect on the recognition rate. An algorithm for model-agnostic feature importance is executed, revealing that the most crucial features in the classification process are mean, standard deviation, perimeter, number of connected components, and number of end points. Signatures are classified as either original or forgery, and the model's performance is assessed on the CEDAR dataset. Experimental results shows a 95.65% accuracy of the proposed system when utilizing standardization with the RBF kernel.

**Keywords:** *offline signature verification; writer independent; geometric features; k-fold; SVM.*

## 1. INTRODUCTION

Signature is popular and one of the oldest ways for an individual's authentication and verification. Each person's signature is unique and different from others. Consequently, in terms of security and fraud prevention, signatures still one of the most widely used methods for authenticating legal and official documents such as contracts, bank checks, certificates, commercial transactions, and credit cards. Forging a signature is unlawful and unethical but can only be done in a controlled experiment for example when creating system solutions such as the system proposed in this paper. An automatic verification system, must have the ability to distinguish between original and forged signatures. Signature verification systems for authentication can be created using the signature features. For verification systems, previous knowledge of the original signers and their signature style is needed [1]. There are two main categories for automating signature verification based on how the signature was obtained: online and offline verification. In online verification, special devices such as smartphones, tablets, and optical pens are used to obtain a signature as it's produced. Online models collect additional information from individuals while writing to carry out the verification [2]. On the other hand, offline signatures are obtained after the writing procedure on paper is finished, then scanned as an input to the system [3].

The complexity of offline signature verification is higher than online one because in the latter model, dynamic features are available during the writing process such as pen acceleration, velocity, angle, time and pressure [4]. The

handwritten signature (offline) is produced via a complicated process that takes into consideration the signer's conditions and psychophysical state at the time of the writing process, leading to dynamic variations in the signature. Even with these difficulties, offline signature verification is more crucial for applications because of the unavailability of digital capturing devices. Both Writer-Dependent (WD) and Writer-Independent (WI) approaches are typically used to build the OSVS [5], [6], [7], [8]. With the WD approach, a personal model is created for each individual based on two distinct classes (original class and forgery class). The drawbacks of this technique are that the classifier needs to be retrained every time a new individual is added to the system. In order to create a reliable system using the WD approach, a large number of signatures are required, which is not practically possible in some real-world applications. On the contrary, only one model is created in a WI approach (also known as a global model) to address every individual without having to be retrained. This model is capable of classifying samples of unknown individual signatures. Hence it was chosen in this work.

The rest of this paper is organized as follows. The literature review is presented in section 2. The proposed methodology is proposed in section 3. Finding the important features is presented in section 4. While section 5 presents the experimental results and discussions. Finally, section 6 presents the conclusions.

## 2. LITERATURE REVIEW

Reference [6] presented the R-SigNet architecture to reduce feature space learning for WI signature verification. This network extracts generic features automatically. They trained and tested the verification system using Support Vector Machine (SVM) by taking advantage of small generic feature space. The system's result proved that the training time is reduced while the performance improved. Reference [9] introduced a novel technique based on best feature selection. Global features are extracted from signature image such as aspect ratio, signature area, pure height, pure width and normalized actual signature height. Local features are also extracted like signature centroid, angle, slope, and distance. Genetic Algorithm is applied on extracted features to select best features. Feature verification is performed by using SVM. Reference [10] presented an automatic technique with a multi-level features fusion and optimal features selection for offline signature verification. Eight geometric features and 22 Grey Level Co-occurrences Matrix (GLCM) are calculated from pre-processing signature images. A skewness-kurtosis controlled PCA (SKcPCA) based feature selection to select the best features for the classification into original and forged signatures. Reference [5] investigated the range of geometric features to create an effective OSVS using a writer-independent technique and multiple classifiers. For classification, they used SVM with Gaussian radial basis function (RBF) and polynomial kernel (Poly). Reference [11] introduced a morphological-based approach for offline signature verification. In this method, the signature image is partitioned into eight grids of equal size. Morphological pattern spectra are then computed for each grid to address scaling challenges. The resulting vertical morphological spectra are concatenated to form an eighth-dimensional feature vector, represented by normalized histograms. The SVM and MLP are employed for verification in this system. Reference [12] introduced the Partial Invariant Chord Oriented Gap (PICO) feature, a novel approach. By examining the directional variations in the gaps (sequences of white pixels) between strokes in a signature, they devised this feature through heuristic evolution. In the context of a writer-dependent system, a collection of genuine and forged signatures was employed to identify a distinct set of partial invariant chords. A threshold ( $d_{inv}$ ) is used to select different sets of PICO chords for each writer, then another training step is performed by using PICO chords and the similarity threshold ( $d_{th}$ ) is calculated. To determine whether the testing signature is legitimate or forgery, a majority score based system is used.

## 3. THE PROPOSED METHODOLOGY

The main steps in the proposed method for the proposed OSVS include:

- Step 1: using a third-party dataset of signature.
- Step 2: pre-processing of signature image.
- Step 3: extracting features from signature image.
- Step 4: normalization/standardization of extracted features (i.e. pre-processing of data prior to training and testing).
- Step 5: training a model.
- Step 6: testing the model.

Figure 1 shows a diagram of the proposed methodology.

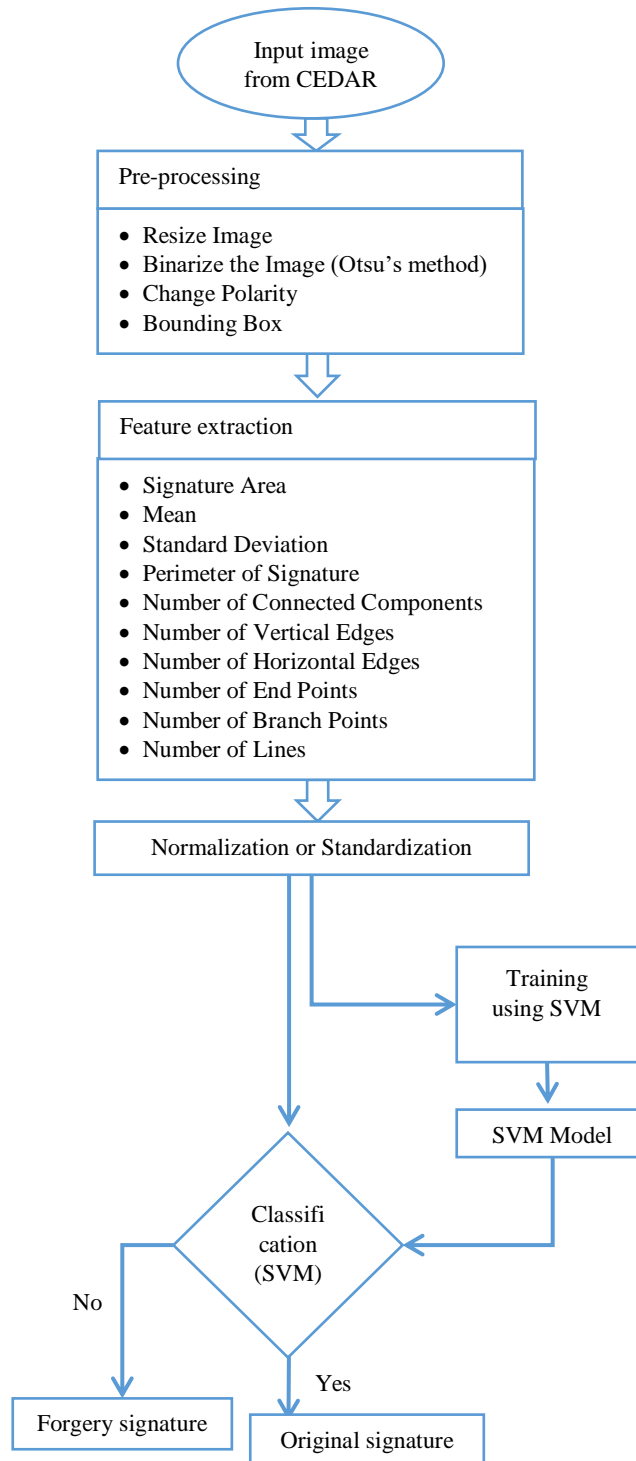


Fig. 1. Diagram of the proposed system.

### 3.1 CEDAR Dataset

In our proposed method, The Centre of Excellence for Document Analysis and Recognition (CEDAR) dataset is used [13]. The dataset is freely available and were widely used by researchers. This dataset includes the signature of 55 individuals (signers) from different backgrounds and cultures. A 24 original signatures were provided by each of them. For each of the initial signatories, a total of 24 forged signatures were generated by mimicking the signatures of three individuals eight times. This process yielded a combined set of 1320 authentic signatures and an equal number of forged signatures. It is worth noting that the signature images included in the CEDAR dataset are presented in grayscale mode. Figure 2 shows two samples, original on the left and forged on the right.

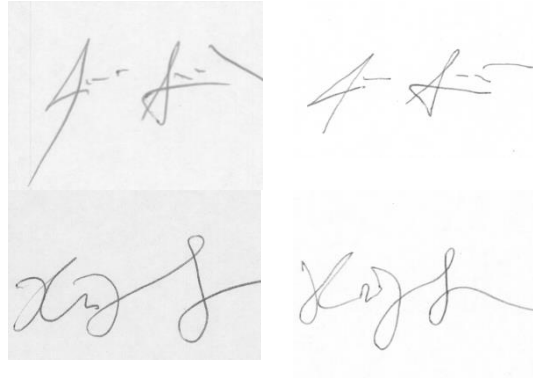


Fig. 2. Sample images from CEDAR dataset.

### 3.2 Pre-processing

During this phase, various image enhancement operations are executed on the signature image with the aim of eliminating superfluous data from the image. The pre-processing stage includes the following steps:

1. **Resize Image:** Signature images within the CEDAR dataset vary in size. To standardize their dimensions, the images are centered and resized to a resolution of 200 x 200.
2. **Binarize the Image:** The signature image undergoes conversion into a binary format (black and white) through the application of the Otsu binarization algorithm [14]. The algorithm aims to identify the threshold value with the minimal total entropy for both the foreground and background. In accordance with Otsu's approach, the threshold is determined based on statistical data from the image. For a given threshold value, denoted as 't', the variance of clusters T0 and T1 can be computed. The optimal threshold value is ascertained by minimizing the sum of the weighted class variances, with weights representing the probabilities of the respective groups. Initially, the histogram and probability (p) for each intensity level of the grayscale image are computed. The initial weights (w), mean ( $\mu$ ), and variance parameter ( $\sigma$ ) for each intensity level are all set to zero, as outlined by [9] and [15]. The formula for determining the within-class variance at any given threshold 't' is expressed as follows:

$$\sigma_w^2 = w_{bg}(t) * \sigma_{bg}^2(t) + w_{fg}(t) * \sigma_{fg}^2(t) \quad (1)$$

where

$$w_{bg}(t) = \sum_{i=1}^t p(i) \quad (2)$$

$$w_{fg}(t) = \sum_{i=t+1}^l p(i) \quad (3)$$

$$\mu_{bg}(t) = \frac{\sum_{i=1}^t i * p(i)}{w_{bg}(t)} \quad (4)$$

$$\mu_{fg}(t) = \frac{\sum_{i=t+1}^l i * p(i)}{w_{fg}(t)} \quad (5)$$

$$\sigma_{bg}^2(t) = \frac{\sum_{i=1}^t (i - \mu_{bg}(t))^2 * p(i)}{w_{bg}(t)} \quad (6)$$

$$\sigma_{fg}^2(t) = \frac{\sum_{i=t+1}^l (i - \mu_{fg}(t))^2 * p(i)}{wfg(t)} \quad (7)$$

3. **Changing Polarity (if needed):** In this step of pre-processing the image pixels value are reversed (flipped). All background pixel inverted to zero and foreground pixels (signature) to 255.
4. **Bounding Box:** Create a rectangle box enclosing the signature in the signature image (convex hull). The unnecessary parts of the signature image are cropped (removed). This method further speeds up processing by removing empty space around the signature. Figure 3 shows a sample image passing through pre-processing stage.

### 3.3 Feature Extraction

Following the pre-processing phase, ten geometric features are derived from the pre-processed signature image. This set of features encompasses signature area, mean, standard deviation, perimeter, the number of connected components, vertical edges, horizontal edges, end points, branch points, and lines. These individual features are then consolidated to create a vector representing the geometric characteristics of the signature.

During the feature extraction stage, the following procedures are undertaken:

- Initially, ten geometric features, referred to as global features, are extracted from the entire image.
- Subsequently, the image is partitioned into equal 4x4 areas, as illustrated in Figure 4, and the identical ten features are extracted individually from each partition (zone) of the signature image, now referred to as local features.

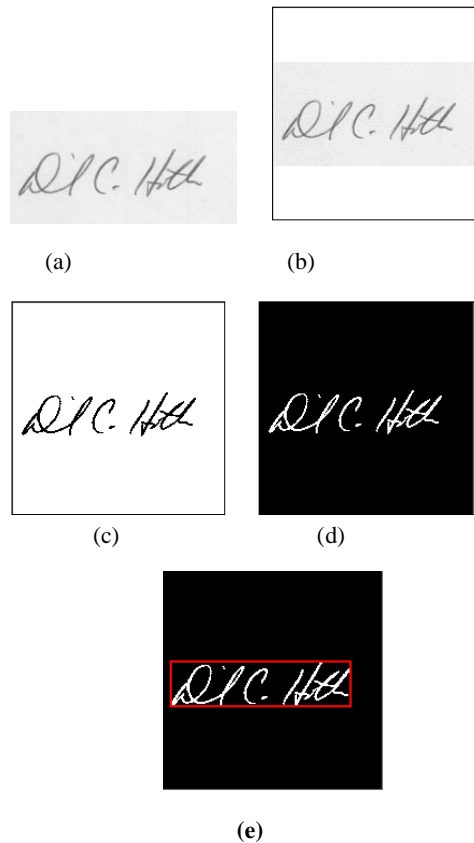


Fig. 3. Pre-processing of a sample image. (a) Input image from CEDAR dataset (b) Resized image (c) Binarized image using Otsu's method (d) Change polarity applied (e) Bounding box around signature.



Fig. 4. Image partitioning (a) Input image (b) Image divided into 4x4 parts (zones).

The features that were extracted from signature image are described in the following subsections.

1. **Signature Area:** The computation of the signature image pixel area involves multiplying the height of the signature image within the bounding box by its corresponding width.

$$\text{Area} = \text{Height} * \text{Width} \quad (8)$$

2. **Mean:** To determine the mean feature, the total number of foreground pixels is divided by the total number of pixels in a signature image.

$$\text{Mean} = \frac{1}{\text{total}} \sum_{i=0}^{\text{total}} p_{fg}(i) \quad (9)$$

3. **Standard Deviation:** To calculate the standard deviation feature, the deviation of the foreground from its mean is assessed.

$$\text{SD} = \sqrt{\frac{\sum (\text{Mean} - p_{fg}(i))^2}{\text{total}}} \quad (10)$$

4. **Perimeter of Signature:** The complete distance encircling a signature image is referred to as its perimeter. It represents the length of the outline or boundary of any two-dimensional geometric shape. The computation of the signature's perimeter involves multiplying 2 by the sum of its length and width.

$$\text{Perimeter} = 2 (l_x + l_y) \quad (11)$$

where

$l_x$ : length of the side

$l_y$ : length of the breadth (width)

5. **Number of Connected Components:** Connected components refer to groups of pixels sharing the same intensity value and connected to each other through 8-connectivity. The Connected Component Labeling (CCL) algorithm, developed by [16], is a two-scan algorithm. In the initial scan, all object pixels in the image are examined and assigned temporary labels. Following the determination of all temporary labels, the second scan consolidates labels belonging to the same connected component (CC) into a single class label. Each pixel is then labeled with a grey-level or color label, depending on the component it was assigned to. Figure 5 illustrates the mask employed by the CCL algorithm.

p (i-1,j-1)	q (i-1,j)	r (i-1,j+1)
s (i,j-1)	x (i,j)	

Fig. 5. Mask used in the CCL algorithm (8-connectivity window).

6. **Number of Vertical Edge:** Sobel gradient kernel is used in the vertical direction (Figure 6) to get the number of vertical edges feature.

**Sobel Edge Detector:** In our proposed method, Sobel edge detector is used to find the number of both vertical and horizontal edges. A Sobel edge detector has made up of two 3x3 convolution kernels, the second kernel is the 90 degree rotation of the first kernel [17].

- 7. Number of Horizontal Edge:** The Sobel gradient kernel is employed in the horizontal direction, as depicted in Figure 6, to obtain the feature representing the number of horizontal edges.

$$\begin{matrix} \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} & \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \\ \text{x-direction kernel} & \text{y-direction kernel} \\ \text{(vertical)} & \text{(horizontal)} \end{matrix}$$

Fig. 6. Sobel filter. (left) x-direction. (right) y-direction.

- 8. Number of End Points:** An endpoint is a foreground pixel in its neighbourhood that has only one other foreground pixel. To determine the number of endpoints feature, the Zhang-Wang thinning algorithm is applied to obtain the one-pixel-wide skeleton of the signature image. Thinning algorithms are commonly employed to extract the skeleton of a pattern by eliminating redundant data at the shape boundary, resulting in a slender representation. The Zhang-Wang algorithm serves as an off-the-shelf parallel thinning algorithm, executing thinning operations on the binary image through repetitive sub-iterations until no more points can be removed without breaking or shortening lines. One sub-iteration removes south and east boundary pixels, while the other removes north and west boundary pixels. The algorithm includes additional rules to preserve the connectivity of the skeleton [18].
- 9. Number of Branch Points:** A branch point is a foreground pixel surrounded by two to four other foreground pixels in its neighbourhood.
- 10. Number of Lines:** To determine the number of lines, the Hough Transform (HT) is applied. The HT is a robust method for detecting straight lines in an image, even in the presence of noise and missing data. In line detection using HT, each point in the image votes for every possible line that passes through it. These votes are then accumulated in a Hough Space, represented by an accumulator array as shown in Figure 7. In the Hough space, each line in the signature image corresponds to a point. Each line is represented as a point in the  $(\rho, \theta)$  parameter space, where 'p' denotes the distance from the origin to a line perpendicular to it, and 'θ' is the angle formed by the positive x-axis and the line [19]. The relevant equation is as follows:

$$\rho = x \cos \theta + y \sin \theta \quad (12)$$

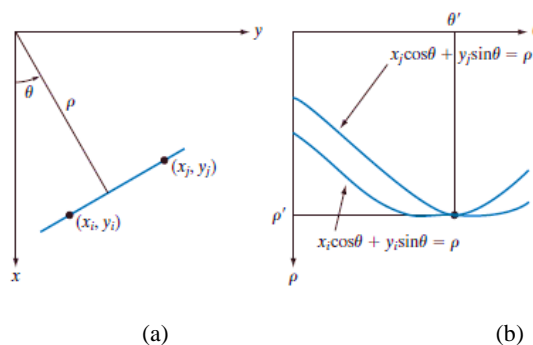


Fig. 7. Hough Transform (a) Line representation in xy-plane. (b) Hough space. Taken from [19].

### 3.4 Normalization and Standardization

Normalization is a technique of data pre-processing which is used to modify feature values in a dataset to a standard scale. This serves to ease data analysis and modelling as well as reduce the effect of different sizes on the accuracy of

the models in machine learning. Normalization is one of the scaling techniques, where values are shifted and rescaled until they fall in the range of 0 and 1. It is also referred to as Min-Max scaling [20].

For normalization the following formula is used:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (13)$$

where

$X'$ : is the new value of feature value ( $X$ ) which is between 0 and 1.

$X_{min}$ : is the minimum value of each feature.

$X_{max}$ : is the maximum value of each feature.

Another method of scaling is standardization, which involves centering the numbers around the mean and utilizing a unit standard deviation. This process aims to set the mean of the attribute to 0, resulting in a distribution with a unit standard deviation [20].

For standardization the following formula is used:

$$x' = \frac{x - \mu}{\sigma} \quad (14)$$

where

$\mu$ : is the mean of feature values.

$\sigma$ : is the standard deviation of feature values.

### 3.5 Training Data

The data undergo training through the SVM algorithm. The primary objective of SVM is to employ a maximal margin classifier, represented by an optimal hyper-plane. The process involves mapping the data domain to a feature space using a kernel function, enabling the classification of classes. Figure 8 illustrates two class samples separated by a margin using the SVM approach.

A separating hyper-plane can be written as:

$$w \cdot u + b \geq 0 \quad (15)$$

The hyper-planes defining the sides (class) of the margin can be written as:

$$w \cdot u + b \geq 1 \quad \text{for positive class} \quad (16)$$

$$w \cdot u + b \leq -1 \quad \text{for negative class} \quad (17)$$

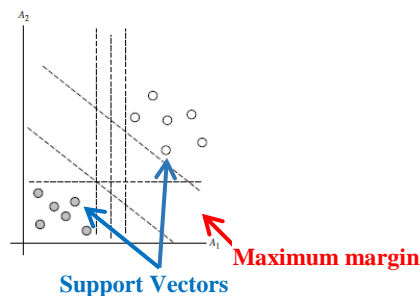


Fig. 8. Training of linearly divided data by SVM. Taken from [21].



#### 4. FINDING FEATURE IMPORTANCE THROUGH PERMUTATION

Feature importance is a concept that transcends any particular machine learning model or technique; instead, it is a strategy applicable to any model, irrespective of its fundamental design or complexity. Permutation feature importance stands out as a widely-used model-agnostic method for assessing the significance of features. This method involves evaluating how the model's performance changes when the values of a specific feature are randomly shuffled. As this technique severs the link between the feature and the target, the reduction in the model score indicates the extent to which the model relies on that feature. Comparing the model's performance before and after shuffling allows for the determination of feature importance, with the feature causing the most substantial decline in performance being deemed the most important. The process initiates with fitting a model to the dataset, followed by making estimates while shuffling the values of each column (feature) in the dataset. This process is repeated multiple times (e.g., 3, 5, 10), resulting in a mean importance score for each input feature, along with a distribution of scores based on the repetitions.

Permutation importance algorithm overview:

1. Inputs: Fitted predictive model ( $m$ ) and tabular dataset ( $D$ ) for either training or validation.
2. Compute the reference score ( $s$ ) of model  $m$  on dataset  $D$ , representing the accuracy for the classifier.
3. For each feature ( $j$ ) in the columns of  $D$ :
  - A. For each repetition ( $k$ ) in the range 1 to  $K$ :
    - i. Randomly shuffle column  $j$  of dataset  $D$  to create a corrupted version named  $D_{k,j}$ .
    - ii. Compute the score ( $s_{k,j}$ ) of model  $m$  on the corrupted data  $D_{k,j}$ .
  - B. Compute the importance ( $i_j$ ) for feature  $f_j$ , defined as:

$$i_j = s - \frac{1}{K} \sum_{k=1}^K s_{k,j} \quad (18)$$

#### 5. EXPERIMENTAL RESULTS AND DISCUSSION

The feature extraction process is executed through Visual Basic, while the training and testing phases are implemented using Python. The experiment is conducted on a laptop equipped with a Core i5 CPU and 8 GB of RAM. In this study, both global and local features are extracted and utilized for each signature image. Each signature image is characterized by 170 features, and the dataset comprises a total of 44,880 records stored in a Comma Separated Values (CSV) database file. Various performance metrics, including Accuracy, Precision, Recall, F1-score, False Acceptance Rate (FAR), and False Rejection Rate (FRR), are employed to assess the model's performance.

Accuracy computed as in (19) below.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (19)$$

**True Positive (TP):** The model correctly predicted the signature as original, and the actual value was indeed original.

**True Negative (TN):** The model accurately predicted the signature as forgery, and the actual value was indeed a forgery.

**False Positive (FP):** The model incorrectly predicted the signature as original, but the actual value was a forgery. This is also known as a Type-I error.

**False Negative (FN):** The model erroneously predicted the signature as forgery, but the actual value was original. This is also referred to as a Type-II error.

$$Precision = \frac{TP}{TP+FP} \quad (20)$$

$$Recall = \frac{TP}{TP+FN} \quad (21)$$

$$F1\text{-score} = \frac{2*(Precision+Recall)}{Precision+Recall} \quad (22)$$

$$FAR = \frac{FP}{FP+TN} \quad (23)$$

$$FRR = \frac{FN}{FN+TP} \quad (24)$$

In this paper, we studied many aspects of the offline signature verification including: pre-processing of data (normalization vs. standardization); kernel (Poly vs. RBF); dataset distribution between training and testing (80%-20% vs. 5-fold); different values of the C and gamma parameters (C=1, 10, 100 and gamma=1, 10, 100). The results are reported in Tables I-V. Note that, the best result in each table is shown in bold.

The first experiment studied the following parameters: Kernel (Poly and RBF); C (1, 10, 100); gamma (1, 10, 100); pre-processing of features (normalization vs. standardization); and the dataset is divided into training set (80%) and testing (20%) vs. 5-fold. The comparison between poly and RBF kernels with (normalization and standardization) is shown in Tables I and II. In the normalization case (Table I), the highest accuracy value obtained for Poly kernel is 77.29%, with C parameter equals 100. However, the accuracy is raised to 89.24% when RBF kernel is used, with the value of C=100 and gamma=10. For the case of standardization (Table II) the accuracy was 68.02% with Poly kernel and value of C=100, and 95.42% for RBF kernel, with the value of C=100 and gamma=1.

TABLE I. COMPARISON BETWEEN POLY AND RBF KERNEL WITH NORMALIZATION FEATURES.

kernel	C	gamma	Accuracy (%)	FAR (%)	FRR (%)
poly	1	-	72.89	8.00	45.94
poly	10	-	75.89	8.00	39.99
poly	100	-	77.29	8.07	37.14
RBF	1	1	86.12	26.22	1.68
RBF	1	10	87.88	21.91	2.43
RBF	1	100	87.78	22.25	2.30
RBF	10	1	88.05	22.02	1.99
RBF	10	10	88.74	20.81	1.81
RBF	10	100	88.5	22.47	0.66
RBF	100	1	88.91	20.79	1.5
<b>RBF<sup>a</sup></b>	<b>100</b>	<b>10</b>	<b>89.24</b>	<b>20.3</b>	<b>1.32</b>
RBF	100	100	88.93	21.71	0.55

<sup>a</sup>. Precision= 83.1%, Recall= 98.7%, F1-score= 90.2%.

TABLE II. COMPARISON BETWEEN POLY AND RBF KERNEL WITH STANDARDIZATION FEATURES.

kernel	C	gamma	Accuracy (%)	FAR (%)	FRR (%)
poly	1	-	52.42	7.22	87.38
poly	10	-	63.14	34.58	39.08
poly	100	-	68.02	28.46	35.43
RBF	1	1	85.32	14.8	14.54
RBF	1	10	80.13	23.44	16.33
RBF	1	100	72.48	42.73	12.5
RBF	10	1	92.94	7.98	6.13
RBF	10	10	86.59	17.27	9.58
RBF	10	100	73.97	40.48	11.75
<b>RBF<sup>b</sup></b>	<b>100</b>	<b>1</b>	<b>95.42</b>	<b>6.39</b>	<b>2.78</b>
RBF	100	10	86.85	17.11	9.22
RBF	100	100	73.97	40.51	11.73

<sup>b</sup> Precision= 93.9%, Recall= 97.2%, F1-score= 95.5%.

In the second experiment we studied same parameters as in the first experiment with one difference, the dataset is divided into training and testing using 5-fold cross-validation technique. The k-fold technique is used to prevent bias in the selection of both training and testing data, to estimate the model performance on unseen data, also due to computational cost. The comparison between poly and RBF kernels with (normalization and standardization) is shown in Tables III and IV. In the normalization case, the highest accuracy value obtained for Poly kernel is 75.47%, with C parameter equals 10. However, the accuracy is raised to 88.91% when RBF kernel is used, with the value of C=100 and gamma=10. For the case of standardization (Table IV) the accuracy was 61.04% with Poly kernel with value of C=10, and 94.07% for RBF kernel, with the value of C=100 and gamma=1.

TABLE III. RESULTS OF MODEL WITH 5-FOLD CROSS VALIDATION AND NORMALIZATION FEATURES.

kernel	C	gamma	Accuracy (%)	FAR (%)	FRR (%)
poly	1	-	72.64	8.32	46.39
poly	10	-	75.47	8.33	40.73
RBF	1	1	85.55	26.96	1.92
RBF	1	10	87.24	22.67	2.84
RBF	1	100	86.71	23.49	3.08
RBF	10	1	87.68	22.7	1.93
RBF	10	10	88.37	21.27	1.98
RBF	10	100	87.56	23.2	1.68
RBF	100	1	88.74	21.09	1.42
<b>RBF<sup>c</sup></b>	<b>100</b>	<b>10</b>	<b>88.91</b>	<b>20.81</b>	<b>1.36</b>
RBF	100	100	88.05	22.6	1.3

<sup>c</sup> Precision= 82.6%, Recall= 98.6%, F1-score= 89.9%.

TABLE IV. RESULTS OF MODEL WITH 5-FOLD CROSS VALIDATION AND STANDARDIZATION FEATURES.

kernel	C	gamma	Accuracy (%)	FAR (%)	FRR (%)
poly	1	-	51.22	11.2	86.35
poly	10	-	61.04	40.69	37.22
RBF	1	1	81.57	18.24	18.61
RBF	1	10	78.33	25.05	18.28
RBF	1	100	71.64	41.95	14.75
RBF	10	1	90.93	9.91	8.23
RBF	10	10	86.01	17.53	10.43
RBF	10	100	73.09	39.97	13.83
<b>RBF<sup>d</sup></b>	<b>100</b>	<b>1</b>	<b>94.07</b>	<b>7.45</b>	<b>4.41</b>
RBF	100	10	86.38	17.08	10.15
RBF	100	100	73.09	39.99	13.82

<sup>d</sup> Precision= 92.8%, Recall= 95.6%, F1-score= 94.2%.

From Tables I-IV, it can be seen that standardization is superior to normalization (also confirmed by the literature). The RBF kernel provided better results than Poly kernel which is due to its complexity and efficiency. Using standardization with the values of 100, 10 for C and gamma parameters, respectively provided the best results among other values. The 5-fold is more accurate than 80%-20% division ratio, although the former gets lower accuracy than the latter.

In the first and second experiments (Tables I-IV), all the ten features are used. However, some features are more important than others. Hence, in the next section, a third experiment is performed to find the most important features out of the ten features used in this study.

In experiment 3, the permutation feature importance algorithm is used to reveal the most important feature among the ten used features. Table V summarizes the results of using 10, 8, 7, 6, 5, and 4 features (best results shown in bold type face). It was shown from Table V that five out of the ten featured are the most important in signature classification of images into original and forged. These features are the mean, standard deviation, perimeter, the number of connected components, and the number of end points.

Table VI shows a comparison between different works on offline signature verification. The experiments are quite different in the sense of method, dataset, and classification approach whether WD or WI. Our method showed better FRR than that of Sharif et al. [9] and Batool et al. [10], although our algorithm has lower performance of 5.82% for FAR compared to 4.67% of Sharif et al. and 3.34% for Batool et al. Our algorithm shows better results than that of Shekar et al. [11] and Kumar and Puhan [12] in both FRR and FAR.

TABLE V. A 5-FOLD EXPERIMENT ON STANDARDIZED FEATURES AFTER REVEALING IMPORTANCE FEATURES.

Accuracy <sup>e</sup> (%)	Precision (%)	Recall (%)	F1-score (%)	FAR (%)	FRR (%)
all 10 features used: area, mean, standard deviation, perimeter, number of connected components, number of vertical edges, number of horizontal edges, number of end points, number of branch points, and number of lines					
94.07	92.78	95.59	94.16	7.45	4.41
8 features used, mean, standard deviation, perimeter, number of connected components, number of vertical edges, number of end points, number of branch points, and number of lines					
93.98	93.06	95.05	94.04	7.09	4.95
7 features used, standard deviation, perimeter, number of connected components number of end points, number of branch points, and number of lines					
94.12	93.43	94.92	94.17	6.68	5.07
6 features used, mean, standard deviation, perimeter, number of connected components, number of end points, and number of lines					
94.15	93.81	94.53	94.17	6.23	5.46
5 features used, mean, standard deviation, perimeter, number of connected components, number of end points					
<b>95.65</b>	<b>94.37</b>	<b>97.12</b>	<b>95.72</b>	<b>5.82</b>	<b>2.88</b>
4 features used, end, mean, standard deviation, perimeter, number of connected components					
95.96	93.96	98.28	96.06	6.36	1.72

<sup>e</sup> RBF kernel, C=100, gamma= 1.

TABLE VI. COMPARISON OF PROPOSED METHOD WITH THE RESULTS OF CEDAR DATASET OF EXISTING METHODS.

Author(s)	WD/WI	Dataset	Features extracted / Method	FAR <sup>f</sup> (%)	FRR <sup>f</sup> (%)	Accuracy <sup>f</sup> (%)
Sharif et al. [9]	WD	CEDAR	Global features contain aspect ratio, area of signature, pure width, pure height and normalized actual signature height, area of black pixels. The local features comprise of the centroid, slope, angle, and distance of signature image	5 samples <sup>g</sup> 8.33, 10 samples 4.17, 12 samples 4.67	5 samples <sup>g</sup> 12.5, 10 samples 8.33, 12 samples 4.67	-
Batool et al. [10]	-	CEDAR	GLCM and Geometric features (area, orientation, solidity, perimeter)	3.34	3.75	-
Shekar et al. [11]	WD	CEDAR	The pre-processed signature image is vertically partitioned into 8 grids and corresponding grid structured morphological pattern spectrum is obtained for each grid	7.9	8.33	91.67
Kumar and Puhan [12]	WD	CEDAR	Partial Invariant Chord Oriented Gap (PICOG) feature which captures both intra-class and inter-class variations among the signatures with a writer	17.50	17.95	82.27
Proposed method	WI	CEDAR, 5-fold	Mean, standard deviation, perimeter, number of connected components, number of end points	5.82	2.88	95.65

<sup>f</sup>. When multiple datasets are being used, only CEDAR results are reported in this work

<sup>g</sup>. Number of genuine signature samples for training.

Additionally, our study is more practical because it is a WI based that does not need retraining when a new data is added to the systems. While all the other similar studies in Table VI are WD based that needs retraining after adding new user to the system. This work faced many challenges such as fine tuning the parameters of SVM and finding the best features among many available features used in the literature. The first issue is solved empirically by using many values for the C and gamma parameters as shown in Tables I-IV. Finding the best features for training and classification is solved through a permutation importance algorithm explained in Section 4.

## 6. CONCLUSIONS

In conclusion, our study introduced an offline signature verification approach employing geometric global and local features. Through experimental findings, it was established that standardization outperforms normalization in the context of offline signature verification. Additionally, when employed with the SVM classifier, the RBF kernel demonstrated superior performance compared to polynomial kernels. The highest accuracy achieved in this study was 95.65% based on a 5-fold validation. Importantly, our investigation highlighted the significance of five features—mean, standard deviation, perimeter, number of connected components, and number of end points—in influencing classification accuracy, underscoring their pivotal role in the offline signature verification process.

## Conflicts of Interest

The authors declare no conflicts of interest.

## References

- [1] Md. Ajij, S. Pratihar, S. R. Nayak, T. Hanne, and D. S. Roy, "Off-line signature verification using elementary combinations of directional codes from boundary pixels," *Neural Comput. Appl.*, March 2021, doi: 10.1007/s00521-021-05854-6.

- [2] V. L. F. Souza, A. L. I. Oliveira, and R. Sabourin, "A writer-independent approach for offline signature verification using deep convolutional neural networks features," in 7th Brazilian Conference on Intelligent Systems (BRACIS), 22-25 October 2018, Sao Paulo, Brazil, pp. 212–217, doi: 10.1109/BRACIS.2018.00044.
- [3] S. Pal, A. Alaei, U. Pal, and M. Blumenstein, "Performance of an off-line signature verification method based on texture features on a large indic-script signature dataset," in 12th IAPR Workshop on Document Analysis Systems (DAS), 11-14 April 2016, Santorini, Greece, pp. 72–77, doi: 10.1109/DAS.2016.48.
- [4] P. N. Narwade, R. R. Sawant, and S. V. Bonde, "Offline handwritten signature verification using cylindrical shape context," 3D Res., vol. 9, no. 4, p. 48, October 2018, doi: 10.1007/s13319-018-0200-0.
- [5] A. Kumar and K. Bhatia, "Development of writer independent offline signature verification system through multiple classifier and geometric features," in International Conference on System Modeling & Advancement in Research Trends (SMART), 23-24 November 2018, Moradabad, India, pp. 255–260, doi: 10.1109/SYSMART.2018.8746943.
- [6] D. Avola, M. J. Bigdello, L. Cinque, A. Fagioli, and M. R. Marini, "R-SigNet: Reduced space writer-independent feature learning for offline writer-dependent signature verification," Pattern Recognit. Lett., vol. 150, pp. 189–196, October 2021, doi: <https://doi.org/10.1016/j.patrec.2021.06.033>.
- [7] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Writer-independent feature learning for offline signature verification using deep convolutional neural networks," in International Joint Conference on Neural Networks (IJCNN), 24-29 July 2016, Vancouver, BC, Canada, pp. 2576–2583, doi: 10.1109/IJCNN.2016.7727521.
- [8] R. Kumar, J. D. Sharma, and B. Chanda, "Writer-independent off-line signature verification using surroundedness feature," Pattern Recognit. Lett., vol. 33, no. 3, pp. 301–308, February 2012, doi: 10.1016/j.patrec.2011.10.009.
- [9] M. Sharif, M. A. Khan, M. Faisal, M. Yasmin, and S. L. Fernandes, "A framework for offline signature verification system: Best features selection approach," Pattern Recognit. Lett., vol. 139, pp. 50–59, November 2020, doi: <https://doi.org/10.1016/j.patrec.2018.01.021>.
- [10] F. E. Batool et al., "Offline signature verification system: a novel technique of fusion of GLCM and geometric features using SVM," Multimed. Tools Appl., April 2020, doi: 10.1007/s11042-020-08851-4.
- [11] B. H. Shekar, R. K. Bharathi, J. Kittler, Y. V. Vizilter, and L. Mestestskiy, "Grid structured morphological pattern spectrum for off-line signature verification," in International Conference on Biometrics (ICB), 19-22 May 2015, Phuket, Thailand, pp. 430–435, doi: 10.1109/ICB.2015.7139106.
- [12] M. M. Kumar and N. B. Puan, "Chord oriented gap feature for offline signature verification," in 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), 10-12 December 2014, Singapore, pp. 799–803, doi: 10.1109/ICARCV.2014.7064406.
- [13] H. Srinivasan, S. N. Srihari, and M. J. Beal, "Machine learning for signature verification," in BT - Computer Vision, Graphics and Image Processing, 31-16 December 2006, Madurai, India, pp. 761–775.
- [14] N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. Syst. Man. Cybern., vol. 9, no. 1, pp. 62–66, January 1979, doi: 10.1109/TSMC.1979.4310076.
- [15] J. Yousefi, "Image binarization using Otsu thresholding algorithm," Ontario, Canada Univ. Guelph, vol. 10, April 2011.

- [16] L. Di Stefano and A. Bulgarelli, "A simple and efficient connected components labeling algorithm," in Proceedings 10th International Conference on Image Analysis and Processing, 27-29 September 1999, Venice, Italy, pp. 322–327, doi: 10.1109/ICIAP.1999.797615.
- [17] X. J. Jiang and P. J. Scott, "Chapter 11 - Characterization of free-form structured surfaces," X. J. Jiang and P. J. B. T.-A. M. Scott, Eds. Academic Press, 2020, pp. 281–317.
- [18] Y. Y. Zhang and P. S. P. Wang, "A parallel thinning algorithm with two-subiteration that generates one-pixel-wide skeletons," in Proceedings of 13th International Conference on Pattern Recognition, 25-29 August 1996, Vienna, Austria, vol. 4, pp. 457–461 vol.4, doi: 10.1109/ICPR.1996.547608.
- [19] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed., Elsevier, 2018.
- [20] P. Trebuňa, J. Halčinová, M. Fil'o, and J. Markovič, "The importance of normalization and standardization in the process of clustering," in IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMII), 23-25 January 2014, Herl'any, Slovakia, pp. 381–385, doi: 10.1109/SAMI.2014.6822444.
- [21] J. Han, J. Pei, and M. Kamber, Data Mining: Concepts and Techniques, 3rd ed., Elsevier, 2011.