Intrapersonal Parameter Optimization for Offline Handwritten Signature Augmentation

Teruo M. Maruyama, Luiz S. Oliveira, Alceu S. Britto Jr., and Robert Sabourin, *Member, IEEE*

***Abstract*—Usually, in a real-world scenario, few signature samples are available to train an automatic signature verification system (ASVS). However, such systems do indeed need a lot of signatures to achieve an acceptable performance. Neuromotor signature duplication methods and feature space augmentation methods may be used to meet the need for an increase in the number of samples. Such techniques manually or empirically define a set of parameters to introduce a degree of writer variability. Therefore, in the present study, a method to automatically model the most common writer variability traits is proposed. The method is used to generate offline signatures in the image and the feature space and train an ASVS. We also introduce an alternative approach to evaluate the quality of samples considering their feature vectors. We evaluated the performance of an ASVS with the generated samples using three well-known offline signature datasets: GPDS, MCYT-75, and CEDAR. In GPDS-300, when the SVM classifier was trained using one genuine signature per writer and the duplicates generated in the image space, the Equal Error Rate (EER) decreased from 5.71% to 1.08%. Under the same conditions, the EER decreased to 1.04% using the feature space augmentation technique. We also verified that the model that generates duplicates in the image space reproduces the most common writer variability traits in the three different datasets.**

***Index Terms*—Handwriting signature verification, data augmentation, cold start, parameter optimization, biometric.**

I. Introduction

THE term *signature* is derived from the Latin word *signare* which means to put a mark, and it may be considered a special type of handwriting for several reasons. Generally, the signature is a handwritten representation of a person's name, which may be presented in several formats, ranging from a simple abbreviation to the complete name. It can also have a legible or a flourished format [1, p. 309] [2, p. 87]. Furthermore, each writer has an individual behavior when signing. Several factors, such as culture [3, p. 1368], handwriting skill, age [4], health [5], and the physical and emotional state [6, p. 26-27] can affect this behavior. Due to this individual behavior, signatures produced by the same writer never have the exact same visual appearance. This is called intra-personal variability or writer variability. Since the handwritten signatures of two writers are never exactly alike either, their signatures can be used to distinguish individuals [6, p. 7]. This is known as interpersonal variability [7].

Thanks to interpersonal variability, handwritten signatures are widely used and accepted in verifying a person's identity in legal, administrative, and financial endeavors [8]. Even with the technological advancements that have occurred in the last few decades, handwritten signatures are still the most frequently used type of handwriting among certain writers [2, p. 64] [9].

Like other behavioral biometric traits, the handwritten signature cannot be lost, stolen or forgotten [10]. However, signatures can be forged [6, p. 55]. Forgeries can be classified as random, simple, or skilled. A forgery is considered *random* when the forger does not know the target's name and they use their own signature instead. When the forger knows only the target's name, the forgery is *simple*. The forgery is considered *skilled* when the forger has access to the target's signature and trains to reproduce it. When a forger is trying to reproduce a signature, they try to copy the speed, pressure, and other visual features of the genuine signature. Consequently, the forger tries to reproduce the writer variability of their target [11, p. 204] [12], [13]. Given these elements, modeling the writer variability is a complex and challenging task.

When discussing automatic handwritten signature verification [7], [14]–[16] two different forms of acquisition are considered: online (dynamic) and offline (static). Online signatures are acquired using a digital device, such as a digitizing table, and are represented as a sequence of values over time. These values can represent the pressure, speed, coordinate of the pen, etc., [7]. Offline signatures are acquired after signatures have been written on a piece of paper. The image of the signature is digitized using a scanner or a digital camera. Unlike online signatures, offline signatures are represented by color or gray level data [16]. This lack of complementary features makes it even more difficult to identify offline signature falsifications [8].

Contributing to increase the difficulty of offline signature verification, in the real-world scenario, the number of genuine signature samples per writer is usually too small to train a machine learning model for automatic handwritten signature verification. One alternative is to use data augmentation

Manuscript received March 27, 2020; revised June 19, 2020 and September 24, 2020; accepted October 12, 2020. Date of publication October 23, 2020; date of current version December 1, 2020. This work was supported in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brazil (CAPES)-Finance Code 001 and in part by the Department of Foreign Affairs, Trade and Development (DFATD)-Canada. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Emanuele Maiorana. *(Corresponding author: Teruo M. Maruyama.)*

Teruo M. Maruyama and Luiz S. Oliveira are with the Department of Informatics, Federal University of Paraná, Curitiba 81531-990, Brazil (e-mail: shinigam8@gmail.com; luiz.oliveira@ufpr.br).

Alceu S. Britto Jr., is with the Programa de Pós-Graduação em Informática (PPGIa), Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba 80215-901, Brazil (e-mail: alceu@ppgia.pucpr.br).

Robert Sabourin is with the Laboratoire d'imagerie, de vision et d'intelligence artificielle, École de technologie supérieure, Université du Québec, Montreal, QC H3C 1K3, Canada (e-mail: robert.sabourin@etsmtl.ca).

Digital Object Identifier 10.1109/TIFS.2020.3033442

techniques to increase the number of signatures in the feature and image space. When this process is performed in the image space, it is known as signature duplication [17]. Several signature duplication techniques have been proposed in the literature [17]–[24]. In particular, techniques based on human behavior present more realistic duplicates than other approaches [18], [20], [22]. Among these techniques, the method proposed by Diaz *et al.* [18] must be highlighted.

In their work, Diaz *et al.* [18] argue that writer variability can be described by a global set of parameters. Even though each writer has a different behavior when they sign, some common behaviors can be shared by different writers during the writing process. Therefore, these common variability traits can be modeled by a global set of parameters. To support their hypothesis, the authors performed experiments on two different datasets and alphabetical systems using a set of predefined parameters. However, the parameters were defined empirically, and were thus not based on real writer variability. The parameters were manually optimized based on two factors: human-like aspect and the performance of an automatic signature verification system. The parameters that produced the most human-like duplicates and the best performance with the automatic signature verification system were selected.

Most data augmentation techniques in the feature space have low computational complexity and are simple to implement. Some of them require at least two or three samples to increase the number of training samples [25]. Among the simple and low computational complexity techniques, the application of a Gaussian filter does not require the use of more than one sample. Nevertheless, it still needs to define a parameter to be applied, despite its simplicity. Generally, this parameter is determined manually [25]–[27].

In this work, we advocate that writer variability can be better modeled based on real data. With that in mind, the main contributions of this work are: 1) a method to automatically model writer variability based on offline signatures; 2) for the first time, we present a method to generate synthetic offline signature samples in the feature space using a Gaussian filter; 3) we show how our model can be used to generate more realistic offline signature samples in the image and in the feature spaces; and 4) we propose a new approach to validate the writer variability of synthetic signature samples. Instead of selecting parameters manually, we adopt a real parameter black-box optimization strategy. As a consequence, we can build more robust signature verification systems using very few genuine signatures. Considering a discriminant feature descriptor, we hypothesize that the writer variability observed on the image space can be reflected in the feature space. Our method thus tries to model the writer variability, while considering just the feature space. Therefore, the visual appearance of our duplicates is not directly assessed in this work.

To support our claims, we conducted experiments based on three well-known benchmarks, GPDS-960, CEDAR, and MCYT-75. Considering a real-world scenario, no more than three genuine signatures per writer were used to train the verification system. Duplicates generated using the default parameters proposed by Diaz *et al.* [18] and synthetic samples with our model were also compared. When the ASVS used the duplicates generated using the default parameters, it achieved EERs of 0.83, 0.70, and 3.04, for MCYT, GPDS, and CEDAR, respectively. When it used the duplicates generated by our method, it achieved EERs of 0.07, 0.24, and 2.16, for MCYT, GPDS, and CEDAR, respectively. When it used the synthetic feature vectors generated by our method, it achieved the lowest EERs of 0.01, 0.20, and 0.82, for MCYT, GPDS, and CEDAR, respectively. Thus, despite using fewer genuine signatures per writer than other state-of-the-art signature verification systems, the proposed method outperformed them. The model proved capable of generalizing the most common writer variability traits that are modeled using a GPDS subset, on different datasets. With the proposed method, automatic handwritten signature verification systems can even more closely approximate the real-world scenario.

The remainder of this paper is organized as follows: Section II reviews related works on handwritten signature duplication and feature space augmentation. Section III describes the proposed method for modeling writer variability and associated concepts. In addition, it shows how the proposed method can be used by an automatic signature verification system. Section IV shows how writer variability can be evaluated, considering signature features and the performance of a verification system. Lastly, Section V presents the conclusions.

II. Related Works

Over the last few decades, many advances have been presented in signature verification literature, which have been covered by some key survey papers [14], [15]. More recently, Hafemann *et al.* [16] and Diaz *et al.* [7] have discussed new trends, such as the use of deep learning techniques applied to handwritten signatures. Such methods have achieved superior results in multiple benchmarks. Readers interested in the state of the art of signature verification systems, should please refer to these works and to [28]–[31]. In this section, we cover the core of our work, i.e., signature duplication methods and data augmentation methods in the feature space.

A. Signature Duplication Methods

Signature duplication methods consist of algorithms used to generate new artificial signatures, with one or more signatures used as seeds [18]. Duplication methods can be divided into: i) the creation of online (dynamic) signatures using real online samples [17], [22], [32]–[38]; ii) the creation of offline (static) signatures using real online samples [20], [24], [39]–[42]; iii) the creation of offline signatures using real offline samples [18], [19], [21], [23], [43]–[47]; and iv) the creation of online signatures by using real offline samples. Despite recent advances in the recovery of dynamic signatures from static signatures [48]–[51], the last type of method still an open issue [7].

Duplication methods can still be classified according to the approach used to create new signatures, which in turn can be based on geometrical transformations or be bio-inspired. Figure 1 depicts the taxonomy of duplication methods.

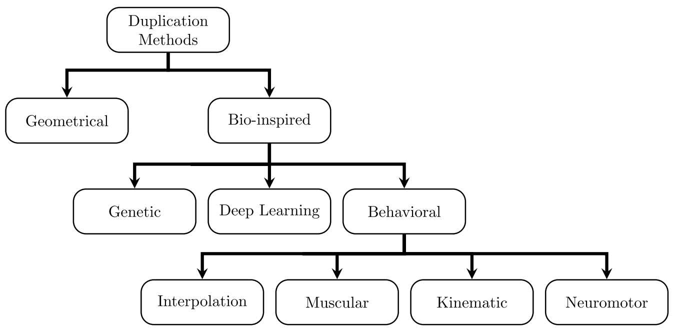


Fig. 1. Taxonomy of the duplication methods.

Methods based on geometrical transformations usually employ rotation [21], [23], [47], scaling [21], [23], [38], a perspective view [23], displacement [19], [21], [38], and warping [24] to increase the number of signatures. They can add some natural and unnatural distortions, which may be used to create artificial genuine signatures [21]–[24], [38] and artificial signature forgeries [23], [47], respectively. As pointed out by Ruiz *et al.* [47], signatures with unnatural distortions can be used as genuine or random forgeries to improve the performance of verification systems. One drawback of geometrical transformations is that they can create duplicates that are not necessarily visually similar to genuine signatures [18]. Even though duplication methods based on geometrical transformations can improve the performance of signature verification systems, they neglect an important aspect of handwriting, namely, the writer's behavior. To bridge this gap, several bio-inspired methods have been proposed in the literature, and can be divided into three main categories, namely, Genetic Algorithms, Deep Learning and Behavioral approaches.

Song and Sun [17] proposed a method to duplicate dynamic signatures based on genetic algorithms. Firstly, a set of duplicates with successive stroke sections are generated. These duplicates must be similar to some sections of the input signature. Then, the input signatures are used to generate a set of duplicates with some different strokes. Both sets are then combined to generate new duplicates. The resulting population is iteratively updated by a cloning-mutation-selection operation. Although it does consider the variability of input signatures, this bio-inspired method does not provide the signature variability that needs to be maintained during the generation of duplicates.

With regard to deep learning, Melo *et al.* [42] proposed a model to generate offline signatures based on online signatures that learn to map online signatures to offline signatures. It requires both types of signature to be trained. However, most datasets do not have both types of signatures available to perform this task. The acquisition of an online signature and the corresponding offline signatures requires several precautions and resources. Despite promising results, deep learning models require a great amount of data to be trained. As this requirement could not possibly be met for their specific case, the authors used a dataset of 23,000 mapped online and offline words. However, all the words had been written by the same writer. Therefore, the trained model did not consider the individual variabilities of different writers.

The third category of the bio-inspired methods tries to mimic the behavior of a person when he/she is writing. In this context, different approaches, such as interpolation, muscular, kinematic, and neuromotor approaches have been used to underpin the proposed methods.

Since dynamic signature data is related to writer behavior, some works use this type of data to generate static signatures [40], [41]. When dynamic information about the signature trajectory is available, it can be interpolated to generate new static signature duplicates. In order to enhance the reliability of this kind of duplicate, information relating to pressure and speed is also used. The reliability of this kind of approach depends strongly on the choice of the interpolation algorithm [40].

Although the handwriting process is still not fully understood, some duplication methods try to model human behavior during a signing act. As mentioned above, some of them are based on muscular models, and others on kinematic theory. The methods based on muscular models try to reproduce the trajectory of signature strokes when the writer is moving the muscles [52]. For their part, those based on kinematic theory try to reproduce the writer's muscular speed effects on the handwritten signature [20], [32], [33], [36], [37], which thus necessitates the presence of online signatures with velocity and pressure information. Methods based on muscular models and kinematic theory are mainly designed to duplicate flourished signatures. Therefore, for legible signatures, the methods can present signature duplicates that are not human-like [36], [45].

The fourth behavioral class category is neuromotor theory [18], [34], [35], [39], [43], [44], [46], which considers a set of muscles, skeletal parts, eyes, and the central nervous system in creating signatures. In this case, the brain stores the signature generation plan and sends electric impulses to the eyes and to specific muscles, allowing the signature plan to be executed by a series of specific muscular contractions and articulatory movements [46].

Most neuromotor methods are not concerned with stroke sequences. However, some of them are mainly focused on the signature trajectory plan [34], [43]. This plan determines the distribution and sequence of strokes which are used to create a new signature. Generally, the sequence information is only present in dynamic signatures. However, Djioua and Plamondon (2009) [53], introduce an algorithm to extract the speed information from an offline signature. Such an algorithm was successfully used by Ferrer *et al.* [46] to create a trajectory plan. Furthermore, Ferrer *et al.* [34] showed that the trajectory plan and complexity of a method depend on the alphabet used to sign.

To simulate the desired motor effects, most neuromotor methods use a grid to map the distribution of strokes or characters on the written surface. Different architectures (hexagonal [34], [43], quadrangular [46], and sinusoidal [18]) have been investigated for generating the desired deformations. In [34], the authors showed that the grid density is directly related to the alphabet used to sign. While dense grids are indicated for eastern alphabets such as Bengali and Devanagari, sparse grids are more suitable for western alphabets such as Latin. Diaz *et al.* [18], on the other hand, showed that the sinusoidal grid produces good duplicates for signatures in

TABLE I

SIGNATURE DUPLICATION METHODS (ON-2-OFF: FROM REAL ONLINE TO DUPLICATED OFFLINE SIGNATURE; ON-2-ON: FROM REAL ONLINE TO DUPLICATED ONLINE SIGNATURE; OFF-2-OFF: FROM REAL OFFLINE TO DUPLICATED OFFLINE SIGNATURE; OFF-2-ON: FROM REAL OFFLINE TO DUPLICATED OFFLINE SIGNATURE; IF THE METHOD CONSIDERS THE WRITER VARIABILITY, IT IS MARKED WITH A X)

Reference	**Conversion**	**Method**	**Writer Variability**	**Alphabet**
Rabasse et al., 2008 [24]	On-2-Off	Affine\Geometrical Transformations	X	Latin\Chinese
Ferrer et al., 2013b [20]	On-2-Off	Spectral Analysis\Kinematic Theory\Paper and Ink Model	-	Latin
Diaz et al., 2014a [39]	On-2-Off	Neuromotor Inspired Model\Ink Model	-	Latin
Diaz et al., 2014b [40]	On-2-Off	Interpolation\Ink Model	-	Latin
Galbally et al., 2015 [41]	On-2-Off	Interpolation\Ink Model	X	Latin
Melo et al., 2019 [42]	On-2-Off	Deep Learning Model	-	Latin
Munich and Perona, 2003 [38]	On-2-On	Time Origin Translation\Affine-scale	-	Latin
Galbally et al., 2009 [22]	On-2-On	Affine\Geometrical Transformations	X	Latin
Galbally et al., 2012 [37] [36]	On-2-On	Spectral Analysis\Kinematic Theory	X	Latin
Song and Sun, 2014 [17]	On-2-On	Clonal Selection Algorithm	X	Latin\Chinese
Diaz et al., 2015 [33]	On-2-On	Kinematic Theory	X	Latin
Diaz et al., 2018 [32]	On-2-On	Kinematic Theory	X	Latin
Ferrer et al., 2017 [35]	On-2-On\Off-2-Off	Neuromotor Inspired Model	X	Latin\Chinese
Ferrer et al., 2018 [34]	On-2-On\Off-2-Off	Neuromotor Inspired Model\Ink Model	X	Bengali\Devanagari
Huang and Yan, 1997 [23]	Off-2-Off	Affine\Geometrical Transformations	-	Latin\Chinese
Fang et al, 2002 [19]	Off-2-Off	Elastic Matching	-	Latin
Frias-Martinez et al., 2006 [21]	Off-2-Off	Affine\Geometrical Transformations	X	Latin
Ferrer et al., 2013a [45]	Off-2-Off	Active Shape Model\Muscular Model\Ink Model	X	Latin
Ferrer et al., 2015 [46]	Off-2-Off	Neuromotor Inspired Model\Ink Model	X	Latin
Diaz et al., 2016a [43]	Off-2-Off	Neuromotor Inspired Model\Ink Model	X	Bengali
Diaz et al., 2016b [44]	Off-2-Off	Neuromotor Inspired Model\Ink Model	X	Bengali\Devanagari
Diaz et al., 2017 [18]	Off-2-Off	Neuromotor Inspired Model\Ink Model	X	Latin
Ruiz et al., 2019 [47]	Off-2-Off	Geometrical\Morphological Transformations\ Noise Addition	-	Latin
Open Issue	Off-2-On	-	-	-

Devanagari, Bengali, and Latin. Two other aspects that play an important role when duplicating a handwritten signature are the paper and ink used. In that regard, some methods try to reproduce successive ink depositions on the paper and the roughness of the paper used [6, p. 195-197] [20], [41], [44].

Similar to methods based on geometrical transformations, behavior-based methods also use several parameters to control writer variability, which is generally defined empirically. Defining such parameters is time-consuming, and very often produces values that do not describe the real writer variability. Table I summarizes the duplication methods showing the reference, type of input and output of the method, the strategy used by each method, whether or not the method considers the writer variability (V), and the signature alphabet.

The literature shows that neuromotor-based methods produce better duplicates than models based on geometrical transformations [18], [34], [35], [39]. Among the neuromotor-based methods, we highlight the one proposed by Diaz *et al.* [18], which uses an ink model to produce high quality duplicates. The duplicates created with this method were used to train a signature verification system and outperformed the system trained with the duplicates created by the geometrical transformation proposed in [21] by a fair margin. Furthermore, it proved to be robust using signatures of different alphabet systems such as Latin, Bengali, and Devanagari. Notwithstanding the good results presented by the authors in [18], the problem of properly defining the parameters that control the generation of duplicates remains unsolved. In this paper, we aim to bridge this gap by proposing a real parameter black-box optimization strategy.

B. Data Augmentation Methods in the Feature Space

Instead of increasing the number of image samples, some techniques such as linear delta [25], interpolation [27], [54], [55], extrapolation [27], delta-encoder [25], and application of random noise [25]–[27], [56], [57] increase the number of samples in the feature space. Unlike image space augmentation, it is very difficult to interpret the synthetic feature vectors [58]. Despite this, most feature space augmentation techniques have low computational complexity and are easy to implement [25].

The linear delta technique finds the difference between the same class samples and uses it to generate a new feature vector. First, the difference between two feature vectors is computed, and is added to a third feature vector of the same class. The delta-encoder extends the linear delta concept using an autoencoder-based model to learn the differences between pairs of samples. First, it computes the intra-class deformations between pairs of training examples. Subsequently, it generates synthetic feature vectors applying the learned transformations to the original feature vectors [25].

The interpolation strategy uses two feature vectors with the same label to generate a new one. For each feature vector, the K nearest neighbors with the same label in feature space are computed. For each pair of neighboring vectors V_k and V_j, a new vector V' can be generated using Equation 1. The degree of interpolation λ is controlled using an interval from 0 to 1 [27].

$$V' = \left(V_k - V_j\right)\lambda + V_j \tag{1}$$

Similar to interpolation, extrapolation can be applied to a vector V_j using Equation 2. In this case, the degree of extrapolation λ is controlled using an interval from 0 to ∞ [25], [27].

$$V'_j = \left(V_j - V_k\right)\lambda + V_j \tag{2}$$

Despite the simplicity of interpolation, extrapolation, and linear delta techniques, these strategies need a least

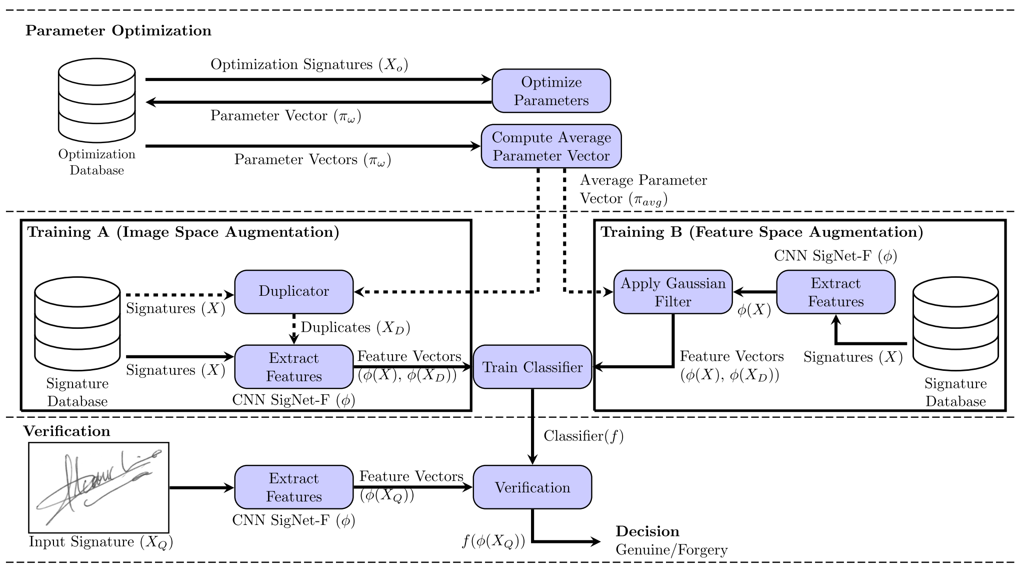


Fig. 2. Offline signature verification system using the proposed method. Training A is used only for data augmentation in the image space, while Training B is used only for data augmentation in the feature space.

two or three feature vectors to generate a new one [25], [27], [54]. In some cases, only one sample is available [18]. Therefore, in this specific case, these strategies cannot be used.

Another simple way the number of training feature vectors are increased is by applying random noise to them [26]. The one-dimensional Gaussian filter is widely used to apply random noise in the feature space [25]–[27]. Unlike the extrapolation, interpolation, and linear delta strategies, random noise can be used with only one sample [27]. Furthermore, it has low computational complexity, and is easier to implement than the delta-encoders [25].

To the best of our knowledge, none of these techniques have previously been used to increase the number of offline signatures in the feature space. Therefore, we also propose a method to increase the number of offline signatures in the feature space based on writer variability.

III. The Proposed Method

To develop our method, we considered two different data augmentation techniques, one in the image space using Duplicator, and the other in the feature space using a Gaussian filter. Figure 2 depicts the overall framework of the proposed method, from parameter optimization through to the final decision, i.e., assign genuine or forgery to a given query signature. The Convolutional Neural Network SigNet-F (ϕ) is used to extract a representation $\phi(X)$ from each signature image X. For a given set of writers in the optimization database, $\phi(X_o)$ is then used to optimize the parameters which describe the signature variability of each writer ω. The result of this optimization is a parameter vector π_ω for each writer that describes his/her variability. Then, the average parameter vector π_{avg} for all writers available in the optimization database is computed.

If we consider only the image space augmentation, the average parameter vector π_{avg} and the signatures X of the training set are used by Duplicator to generate duplicates respecting the writer variability. The model ϕ is used to extract the feature vectors $\phi(X)$ from signatures X, and the feature vectors $\phi(X_D)$ from duplicates X_D. If only the feature space augmentation is considered, the average parameter vector π_{avg} and the signature feature vectors $\phi(X)$ of the training set are used by the Gaussian filter to generate new feature vectors $\phi(X_D)$ respecting the writer variability.

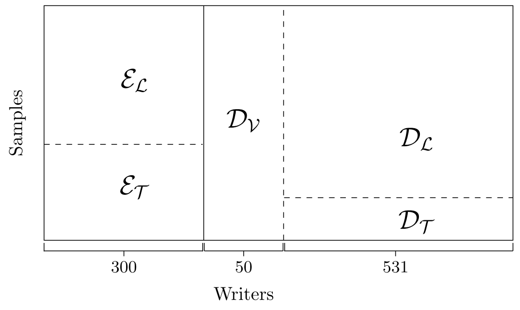
The feature vectors $\phi(X)$ and $\phi(X_D)$ are used to train a classifier f for each writer of the verification system. For an input signature X_Q, the model ϕ extracts the feature vector $\phi(X_Q)$ and sends it to the pre-trained classifier f. Using the feature vector $\phi(X_Q)$, the classifier f makes a decision $f(\phi(X_Q))$ as to whether the signature X_Q is genuine or a forgery.

A. Datasets

The experimental procedure in the present study was performed using the handwritten signature datasets GPDS-960 [59], CEDAR [60], and MCYT-75 [61]. The datasets are summarized in Table II considering the number of writers, number of genuine samples, number of skilled forgeries, and the window size (height $\times$ width) used to normalize the signature images in each dataset. The GPDS dataset consists of 881 different writers, and has 24 genuine samples and 30 skilled forgeries per writer. To compare the results with Hafemann *et al.* [62], we used the same GPDS partitioning (Figure 3) as them. The last 581 writers compose the development dataset $\mathcal{D}$, which is subdivided into $\mathcal{D}_\mathcal{L}$, $\mathcal{D}_\mathcal{T}$ and $\mathcal{D}_\mathcal{V}$ subsets. Subset $\mathcal{D}_\mathcal{L}$ is used to train the Convolutional Neural Network (CNN), while $\mathcal{D}_\mathcal{T}$ is used to monitor the evolution of the CNN training. These two subsets contain the same 531 writers, but different signature samples from each writer. Subset $\mathcal{D}_\mathcal{L}$ contains 90% of the signature samples, while $\mathcal{D}_\mathcal{T}$ contains 10% of them. Some writers of subset $\mathcal{D}_\mathcal{L}$ are also used to optimize the parameters of our method.

TABLE II
THE OFFLINE SIGNATURE DATASETS USED IN THIS WORK

Dataset	Writers	Genuine Signatures	Forgeries	Window Size
GPDS-960	881	24	30	952 × 1360
CEDAR	55	24	24	730 × 1042
MCYT-75	75	15	15	600 × 850



$\mathcal{E}_{\mathcal{L}}$
$\mathcal{E}_{\mathcal{T}}$
$\mathcal{D}_{\mathcal{V}}$
$\mathcal{D}_{\mathcal{L}}$
$\mathcal{D}_{\mathcal{T}}$
Samples
300
50
531
Writers

Fig. 3. GPDS Dataset partitioning.

Subset $\mathcal{D}_{\mathcal{V}}$ contains 50 writers, and is used to make all the choices regarding the CNN model, hyperparameters of the SVM classifiers, and the initial range used to optimize the parameter vectors.

Finally, the first 300 writers (GPDS-300) are used to train and test the SVMs used in this work. They belong to exploitation subset $\mathcal{E}$. The samples that are used to train the classifiers are called $\mathcal{E}_{\mathcal{L}}$, while those used to test them are called $\mathcal{E}_{\mathcal{T}}$.

The CEDAR consists of 55 different writers, and has 24 genuine samples and 24 skilled forgeries per writer [60]. The MCYT-75 consists of 75 different writers, with 15 genuine samples and 15 skilled forgeries [61]. These datasets were used to show the generalization capability of the proposed method.

B. Normalization Process

Hafemann *et al.* [62] showed that feature extraction can be influenced by the normalization process. Furthermore, the CNN expects signatures with the same size. Therefore, signature images are normalized using the procedure proposed in [62]. Firstly, the signature images are segmented using the Otsu algorithm [63]. The signature pixels remain in grayscale, while the background pixels are converted to white (255). The center of mass of the signature is computed and placed into the center of a window of height × width pixels (Table II). Due to the difference between the acquisition protocols of the signature datasets [59], [60], [61], their signatures have different sizes. Therefore, each dataset has its window size to normalize them. This process attempts to maintain the proportion of the different signature sizes.

The color of all the pixels is inverted using Equation 3. The resulting image is resized to 170 × 242 pixels. Finally, the central portion of the image with 150 × 220 pixels is cropped.

$$I(x, y) = 255 - I(x, y) \tag{3}$$

C. Convolutional Neural Network SigNet-F

We used one of the methods proposed by Hafemann *et al.* [62] to extract signature features because of the outstanding results it provided on several benchmarks. The choice was also made thanks to the discriminant nature of the descriptor. It uses a writer-independent feature learning method, in which a development set $\mathcal{D}_{\mathcal{L}}$ is used to learn a feature representation $\phi(X)$. This representation is learned using a Convolutional Neural Network (CNN) to discriminate among writers in $\mathcal{D}_{\mathcal{L}}$. In this context, the CNN (called Signet-F) is trained with both genuine signatures and skilled forgeries, optimizing to jointly discriminate between writers, and between genuine signatures and forgeries. In experiments performed in [62], the SigNet-F achieved the best results in the GPDS-300 and CEDAR datasets, and therefore, we use it in our work.

The subset $\mathcal{D}_{\mathcal{L}}$ was used to learn signature features, while the process was monitored using the subset $\mathcal{D}_{\mathcal{T}}$. The training was performed for 60 epochs, with an initial learning rate of 0.001. After every 20 epochs, the learning rate was divided by 10. Considering the need for a high volume of data to train the CNN, random patches of 150 × 220 pixels were extracted from the normalized 170 × 242 pixel signatures. During the feature extraction, the CNN layer FC7 was used to extract vectors with 2048 elements.

D. Duplicator

Diaz *et al.* [18] proposed a neuromotor method combined with an ink model for signature duplication called Duplicator, which uses a set of 30 parameters that control the signature variability. The first 6 parameters (α_A^{min}, α_A^{max}, α_P^{min}, α_P^{max}, α_S^{min}, α_S^{max}) are mainly responsible for describing the writer variability. To create the writer variability, a sinusoidal transformation is applied. The sine amplitude is determined by α_A^{min} and α_A^{max}, while the sine period is determined by α_P^{min} and α_P^{max}. Finally, the sine phase is delimited by α_S^{min} and α_S^{max}. Considering a flexible surface where the signature is written, these six parameters control how this surface will be deformed. As a consequence, the signature will also be distorted. The next 20 parameters (ξ_x^1, σ_x^1, μ_x^1, ξ_x^2, σ_x^2, μ_x^2, ξ_x^3, σ_x^3, μ_x^3, ξ_y^1, σ_y^1, μ_y^1, ξ_y^2, σ_y^2, μ_y^2, ξ_y^3, σ_y^3, μ_y^3, k_1, and k_2) describe the distribution of unconnected strokes in the image. These strokes are displaced taking into account three different kinds of ratio intervals, which for their part are determined by k_1 and k_2. To choose an interval, the ratio between the number of stroke pixels and the number of signature pixels is calculated. If a stroke sits within one of these intervals r, 6 parameters (σ_x^r, μ_x^r, ξ_x^r, σ_y^r, μ_y^r, and ξ_y^r) are used to displace the unconnected stroke. The ink deposition effect is determined by ψ and the last 3 parameters (ξ_S, σ_S, μ_S) are used to control the signature inclination.

In this work, we focused on the optimization of the first six parameters which are mainly responsible for defining the writer variability [18], and the others were kept at their default values. Table III shows the default values of all 31 parameters.

TABLE III
DEFAULT PARAMETER VECTOR PROPOSED IN [18] TO DUPLICATE OFFLINE HANDWRITTEN SIGNATURES

Description	Parameter	Default Values
Intra-class Variability	α_A^{min}	5
	α_A^{max}	30
	α_P^{min}	0.5
	α_P^{max}	1
	α_S^{min}	0
	α_S^{max}	1
Inter-Component Variability	$\{\xi_x^1, \sigma_x^1, \mu_x^1\}$	$\{-0.5, 20, 2*\sigma_x^1\}$
	$\{\xi_x^2, \sigma_x^2, \mu_x^2\}$	$\{-0.5, 1.4*\sigma_x^1, 2*1.4*\sigma_x^1\}$
	$\{\xi_x^3, \sigma_x^3, \mu_x^3\}$	$\{-0.5, 1.8*\sigma_x^1, 2*1.8*\sigma_x^1\}$
	$\{\xi_y^1, \sigma_y^1, \mu_y^1\}$	$\{-0.5, 8, \sigma_y^1\}$
	$\{\xi_y^2, \sigma_y^2, \mu_y^2\}$	$\{-0.5, 1.2*\sigma_y^1, 1.2*\sigma_y^1\}$
	$\{\xi_y^3, \sigma_y^3, \mu_y^3\}$	$\{-0.5, 1.5*\sigma_y^1, 1.5*\sigma_y^1\}$
	k_1	0.33
	k_2	0.67
	ψ	0.8
Inclination	ξ_S	-0.19
	σ_S	3.28
	μ_S	-1.30

E. Gaussian Filter

As previously shown, the one-dimensional low-pass Gaussian filter is widely used to generate synthetic samples in the feature space (Equation 4). Despite the simplicity of the filter [25], a parameter σ is still needed to control its intensity. The standard deviation σ is randomly selected, considering a uniform distribution from σ_{min} to σ_{max}. This interval introduces some variability to the synthetic samples. Based on the same idea as in Duplicator, we used this interval to represent the writer variability, and thus optimized the parameters that determine this interval using the optimization process described in the next section.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{4}$$

F. Parameter Optimization

Parameter optimization is performed in a bid to find a set of parameters for data augmentation methods, which allow the methods to generate synthetic samples respecting the distribution of a given writer. In a real-world scenario, two or more writers have different intra-class variabilities. This difference notwithstanding, some common behaviors can be shared by these writers along the writing process, and these common writer variability traits can be described by a global set of parameters. Since the first six parameters of the duplicator are mainly responsible for the intra-personal variability of writer signatures [18], and optimization is a time-consuming task [64], these parameters are chosen to represent the writer variability traits. Regarding the Gaussian filter, we used the parameters σ_{min} and σ_{max} to represent the writer variability traits.

In this work, we used a Particle Swarm Optimization (PSO) algorithm [65] to find the first six variability parameters used by the Neuromotor-based duplicator, and the two parameters used by Gaussian filter. The PSO was originally proposed by Kennedy and Eberhart [66] to optimize continuous nonlinear functions. The algorithm is based on the behavior of a flock of birds or school of fish looking for places with abundant food. Due to its efficiency in solving several kinds of optimization problems [65], [67]–[69] and its simple implementation [64], the PSO is used in our work.

Considering the search space of d dimensions, each particle π with index i represents the position and a possible solution to an optimization problem. During optimization, a velocity υ with index i is associated with each particle in the search space. For each iteration n of the algorithm, the velocity υ_{id}^{n+1} of the next iteration (Equation 5) is updated according to the values of the local minima particle p_{id}^n and the global minima particle $\pi_{\omega d}^n$ [65].

The υ_{id}^{n+1} is also computed considering two uniformly distributed variables within [0,1], r_{1id}^n and r_{2id}^n. These variables introduce some diversity to the particles during the search. This diversification is regulated by the constant $1-\chi_o$. The constant χ_o itself controls the intensity of the search to find the best solution [65]. While the constant c_{o1} regulates the tendency of the particles to approach their local minima particle, the constant c_{o2} regulates their tendency to approach their global minima particle [69]. The perturbation constant γ_o controls the stability of the algorithm regulating the effect of both constants c_{o1} and c_{o2} concurrently.

$$\begin{aligned} \upsilon_{id}^{n+1} &= (1-\chi_o)\upsilon_{id}^n + \chi_o c_{o1}\gamma_o r_{1id}^n (p_{id}^n - \pi_{id}^n) \\ &\quad + \chi_o c_{o2}\gamma_o r_{2id}^n (\pi_{\omega d}^n - \pi_{id}^n) \end{aligned} \tag{5}$$

According to the optimization experiments performed by Zhao [65], when the PSO uses the constants $(1-\chi_o) = (3-\sqrt{5})/2$, $\chi_o c_{o1}\gamma_o = (1+\sqrt{5})/2$, and $\chi_o c_{o2}\gamma_o = 1$ it is more accurate, efficient, and stable than the traditional PSO [66]. Therefore, we used these constants in Equation 5, which resulted in Equation 6. Considering the computed velocity υ_{id}^{n+1}, the position of the particle π_{id}^{n+1} is updated using Equation 7.

$$\begin{aligned} \upsilon_{id}^{n+1} &= \frac{(3-\sqrt{5})}{2}\upsilon_{id}^n + \frac{(1+\sqrt{5})}{2} r_{1id}^n (p_{id}^n - \pi_{id}^n) \\ &\quad + r_{2id}^n (\pi_{\omega d}^n - \pi_{id}^n) \end{aligned} \tag{6}$$

$$\pi_{id}^{n+1} = \pi_{id}^n + \upsilon_{id}^{n+1} \tag{7}$$

The parameter vector is encoded into a 6-dimensional particle π for image space augmentation, while for feature space augmentation, it is encoded into a 2-dimensional particle π. The parameters are randomly initialized considering a uniform distribution with low and high limits. Table IV shows the low and high limits used during the optimization process. The low limit of the max parameter is defined using the value of the min parameter. For example, if the parameter α_A^{min} is initialized with the value 20, the low limit of the parameter α_A^{max} is 20.

For the image space augmentation, the limits of the parameters were defined considering the extreme values of the default parameters proposed by Diaz *et al.* [18]. If we use α_A^{min} and α_A^{max} equal to or greater than 10, the duplicator generates duplicates with fewer distortions than when values lower than 10 are used. However, if we use large values of α_A, the duplicates will be equal to the signature used as seed.

TABLE IV
PARAMETER INITIALIZATION RANGE USED TO OPTIMIZE THE PARAMETER VECTORS

Limit	Parameter Initialization Range							
	α_A^{min}	α_A^{max}	α_P^{min}	α_P^{max}	α_S^{min}	α_S^{max}	σ_{min}	σ_{max}
Low	10.0	α_A^{min}	0.0	α_P^{min}	0.0	α_S^{min}	0.01	σ_{min}
High	100.0	100.0	1.0	1.0	1.0	1.0	1.00	1.00

In order to find the best values of α_A^{min} and α_A^{max}, we set the limits of α_A^{min} from 10 to 100. According to the authors in [18] the values of α_P^{min}, α_P^{max}, α_S^{min}, and α_S^{max} should be between 0 and 1. However, when these parameters assume extremely small or extremely large values, the duplicator generates unnatural signature duplicates.

For the feature space augmentation, the limits of the parameters were defined, considering the mathematical constraints of the Gaussian filter. When σ is 0, it leads to a mathematical indetermination. We also thought about the degree of perturbation applied by the Gaussian filter in the original feature vectors. If we perturb the original feature vector too much, it can generate a feature vector that does not resemble the same class as the original [27]. Therefore, we considered the limits ranging from 0.01 to 1.00.

To guide the optimization process, we used the silhouette index [70]. Usually, this index is used to measure how good two or more clusters are. If the silhouette index is equal to 1, the clusters have a small intraclass and a large interclass variability; if it is equal to -1, it means the data were assigned to the wrong clusters. Finally, if the silhouette index is equal to 0, then the clusters are overlapped. This is exactly what we are looking for. Therefore, during the optimization process, our goal is to find a set of parameters such that the silhouette index is close to 0.

The silhouette index evaluates the sparsity and the distance between the clusters concurrently. Several functions can be used to compute the dissimilarity between the elements of the clusters. As recommended by Rousseeuw [70], we used the Euclidean distance as a dissimilarity function $d(.)$. The average dissimilarity between the i^{th} element and the elements of the same cluster Cs can be computed using Equation 8. Basically, $a(.)$ measures the sparsity inside each cluster.

$$a(\phi(X_i)) = \frac{\sum_{j=1}^{n_{Cs}} d(\phi(X_i); \phi(X_j))}{n_{Cs} - 1} \tag{8}$$

The average dissimilarity between the i^{th} element and the elements of the other cluster Cr is computed using Equation 9. Subsequently, Equation 9 is used in Equation 10 to compute the minimum distance between the border of the clusters Cs and Cr. This indicates how far the clusters are from each other. It is important to highlight that the clusters Cs and Cr are different.

$$d(\phi(X_i); Cr) = \frac{\sum_{j=1}^{n_{Cr}} d(\phi(X_i); \phi(X_j))}{n_{Cr}} \tag{9}$$

$$b(\phi(X_i)) = \min\{d(\phi(X_i); Cr)\} \tag{10}$$

Equations 8 and 10 are combined in Equation 11. For an arbitrary feature vector $\phi(X_i)$ as reference, Equation 11 evaluates the intraclass and the interclass variability concurrently. Following that, the value is normalized using the maximum value between the intraclass and the interclass variability.

$$\delta(\phi(X_i)) = \frac{b(\phi(X_i)) - a(\phi(X_i))}{\max\{b(\phi(X_i)); a(\phi(X_i))\}} \tag{11}$$

Equation 12 computes the $\delta(\phi(X_i))$ for all elements of the clusters. The $\delta(\phi(X_i))$s are summed and divided by the number of elements n_C in all clusters. To simplify the optimization process, we used only the absolute value of the silhouette index ($|\Delta|$).

$$|\Delta| = \left| \frac{\sum_{i=1}^{n_C} \delta(\phi(X_i))}{n_C} \right|, \quad |\Delta| \in [0, 1] \tag{12}$$

Details of the proposed optimization algorithm are presented in Algorithm 1. It receives as input N (the number of samples that will be generated for each genuine signature), W (list of writers with their respective genuine signatures), and I (the number of iterations for which the algorithm will run). The output is the average parameter vector π_{avg}. While Algorithm 2 presents the evaluation process of parameters to generate duplicates, Algorithm 3 details the evaluation of parameters to generate samples in the feature space.

Then, the optimization process starts with the PSO generating a set of parameter vectors that are used to create the synthetic samples. In the image space augmentation, for each signature of the writer X_o, a duplicate X_D is generated. The duplicates are normalized and the feature vectors $\phi(X_o)$, and $\phi(X_D)$ are extracted. In the feature space augmentation, the signatures of the writer are normalized and the feature vectors $\phi(X_o)$ are extracted. For each feature vector $\phi(X_o)$, a synthetic feature vector $\phi(X_D)$ is generated using a Gaussian filter. The feature vectors of the genuine signatures $\phi(X_o)$ and synthetic samples $\phi(X_D)$ are used to compute the absolute value of the silhouette index $|\Delta|$. If the cluster of genuine signatures and the cluster of synthetic samples have an equal or similar variability, $|\Delta| \to 0$. Therefore, the parameter vectors π_ω with the lowest absolute silhouette indices are selected and saved for each writer. The parameter vectors are updated for the next iteration using Equations 6 and 7. The process is repeated until the stop condition is satisfied. In the end, the average parameter vector (π_{avg}) is computed, and describes the common behavioral biometric traits shared by the writers in the optimization database. It is important to highlight that we hypothesized that the writer variability observed on the image space can be reflected in the feature space. Based on this hypothesis, the parameter optimization is performed, considering only the feature vectors. Therefore, we assume that the duplicates will have a human-like appearance if $|\Delta| \to 0$. In other words, the interaction between the first 6 parameters is taken into account considering the minimization of the absolute value of Δ (Equation 12).

Algorithm 1 The Parameter Optimization Algorithm (Sigvar)

```
Input:
    $N$: number of duplicates per signature
    $W$: list of writers
    $I$: number of iterations
Output:
    $\pi_{avg}$: average parameter vector
1: $\pi_{avg} \leftarrow \emptyset$
2: foreach $\omega \in W$ do
3:     $X \leftarrow$ loadSignatures($\omega$)
4:     $\triangleright$ Initialize the particles
5:     $particles \leftarrow$ initializeParticles()
6:     $\Delta_{localmin} \leftarrow 9999$
7:     $\Delta_{min} \leftarrow 9999$
8:     $\pi_\omega \leftarrow \emptyset$
9:     foreach $iteration \in I$ do
10:        $p \leftarrow \emptyset$
11:        foreach $\pi \in particles$ do
12:            $\triangleright$ Evaluate parameter vector
13:            $\Delta \leftarrow$evalParameters($\pi$,$X$,$N$)
14:            if $|\Delta| < |\Delta_{min}|$ then
15:                $|\Delta_{min}| \leftarrow |\Delta|$
16:                $\pi_\omega \leftarrow \pi$
17:            end if
18:            if $|\Delta| < |\Delta_{localmin}|$ then
19:                $|\Delta_{localmin}| \leftarrow |\Delta|$
20:                $p \leftarrow \pi$
21:            end if
22:        end for
23:        $\triangleright$ Update the value of each particle using Equations 6 and 7
24:        $particles \leftarrow$ updateParticles(particles, p, $\pi_\omega$)
25:    end for
26:    $\triangleright$ Save the best parameters for each writer
27:    saveParameters($\omega$,$\pi_\omega$)
28:    $\triangleright$ Sum all the parameter vectors
29:    $\pi_{avg} \leftarrow \pi_{avg} + \pi_\omega$
30: end for
31: $\triangleright$ Count the number of writers in the list $W$
32: $n_W \leftarrow$getNumberOfWriters(W)
33: $\triangleright$ Compute the average parameter vector
34: $\pi_{avg} \leftarrow \pi_{avg}/n_W$
```

Algorithm 2 evalParameters (Duplicator)

```
Input:
    $\pi$: parameter vector
    $X$: signatures
    $N$: number of duplicates per signature
Output:
    $\Delta$: Silhouette Index
1: $\pi_{avg} \leftarrow \emptyset$
2: $X_D \leftarrow \emptyset$
3: foreach $X_i \in X$ do
4:     aux$X_D \leftarrow$ Duplicator($\pi$, N,$X_i$)
5:     $\triangleright$ Concatenate $X_D$ and aux$X_D$
6:     $X_D \leftarrow X_D \frown$aux$X_D$
7: end for
8: $\triangleright$ Normalize the signatures and duplicates
9: norm$X$ $\leftarrow$normalize($X$)
10: norm$X_D$ $\leftarrow$normalize($X_D$)
11: $\triangleright$ Extract the features from the normalized signatures and duplicates using the SigNet-F
12: $\phi(X)$ $\leftarrow$extractFeatures(norm$X$)
13: $\phi(X_D) \leftarrow$ extractFeatures(norm$X_D$)
14: $\triangleright$ Calculate the value of the silhouette index
15: $\Delta$ $\leftarrow$silhouetteIndex($\phi(X)$,$\phi(X_D)$)
```

Algorithm 3 evalParameters (Gaussian Filter)

```
Input:
    $\pi$: parameter vector
    $X$: signatures
    $N$: number of new samples per signature
Output:
    $\Delta$: Silhouette Index
1: $\triangleright$ Normalize the signatures and duplicates
2: norm$X$ $\leftarrow$normalize($X$)
3: $\triangleright$ Extract the features from the normalized signatures using the SigNet-F
4: $\phi(X)$ $\leftarrow$extractFeatures(norm$X$)
5: $\phi(X_D) \leftarrow \emptyset$
6: foreach $\phi(X_i) \in \phi(X)$ do
7:     aux$\phi(X_D) \leftarrow$ applyGaussianfilter($\pi$, N, $\phi(X_i)$)
8:     $\triangleright$ Concatenate $\phi(X_D)$ and aux$\phi(X_D)$
9:     $\phi(X_D) \leftarrow \phi(X_D) \frown$aux$\phi(X_D)$
10: end for
11: $\triangleright$ Calculate the value of the silhouette index
12: $\Delta$ $\leftarrow$silhouetteIndex($\phi(X)$,$\phi(X_D)$)
```

G. Training

Once the search process is complete, we can use the optimized parameters to create synthetic samples to train the Writer-Dependent classifiers used by the signature verification system. Four different scenarios were considered to train these classifiers. The first scenario considers that there are no duplicates at all to train the SVMs, and is used as a baseline. The second one uses duplicates that were created by using the default parameters reported in Table III. The third creates duplicates using the parameters found by the proposed algorithm described in the previous section. Finally, the last scenario creates synthetic feature vectors using a Gaussian filter and the parameters found by the optimization process.

To validate the impacts of the proposed optimization method and to fairly compare the results with those of Hafemann *et al.* [62], we adopted the same signature verification system proposed by them. The SigNet-F is used to extract the feature vectors of the normalized genuine and random forgery signatures. For each writer, an SVM (Support Vector Machine) classifier with an RBF kernel is trained using these feature vectors. The genuine signatures were considered as a positive class and the random forgeries were considered as a negative class. The genuine signatures of other writers are used as random forgeries. The difference in the numbers of positive and negative examples may lead to classifiers that tend to select one class more frequently than others. Therefore, different C weights were adopted to positive and negative classes. For the negative class, C^- is equal to 1. Before determining the C^+, it is necessary to compute the skew ψ. The skew ψ was computed using the number of positive examples P (genuine signatures) used for training and the number of negative examples N (random forgeries) used for training (Equation 13). Then, C^+ was computed using C^- and ψ (Equation 14).

$$\psi = \frac{N}{P} \tag{13}$$

$$C^+ = \psi C^- \tag{14}$$

H. Verification

After training the Writer-Dependent classifiers the system is ready to be deployed. During the verification phase, each query signature image X_Q is normalized and the feature vector $\phi(X_Q)$ is extracted using the SigNet-F. The feature vector $\phi(X_Q)$ is submitted to an SVM classifier f, and it makes a decision $f(\phi(X_Q))$. As a result of the decision $f(\phi(X_Q))$, the signature X_Q is classified as a genuine or a forgery sample. To assess the performance of the proposed method, the mean Equal Error Rate (EER) of the verification system was computed. The verification system was assessed considering the three previously exposed scenarios.

IV. Experimental Results

To validate the proposed method, two different approaches were considered: feature-level (Section IV-A) and performance-level (Sections IV-B and IV-C). All experiments were carried out on the three datasets described in Section III-A.

It is well known that finding a solution for an optimization problem using a meta-heuristic algorithm is a time-consuming task, and it was no different in our case. For this reason, instead of using all writers available in the $\mathcal{D}_\mathcal{L}$, we selected a subset of 20 writers representing the entire population to compose our optimization dataset. The 20 writers that covered the distribution of $\mathcal{D}_\mathcal{L}$ were randomly selected (writers 431, 490, 503, 525, 588, 611, 631, 641, 643, 654, 673, 676, 701, 716, 797, 825, 897, 912, 935, and 945) and used as input to the optimization algorithm.

As discussed in Section III-F, at the end of the optimization process, for each writer from the optimization dataset, there is one optimized parameter vector, which is used to calculate the average parameter vector (π_{avg}) that is used by the data augmentation method. While the duplicator uses the average parameter vector π_{dup}, the Gaussian filter uses the average parameter vector π_{gauss}. Table V shows the silhouette index, the average parameter vectors π_{dup} and π_{gauss}, and the default parameter vector. As we can see, π_{dup} is quite distant from the default vector. Mainly, the first two parameters (α_A^{min} and α_A^{max}) have greater average parameter vector values than those specified for the default parameter vector. Consequently, the silhouette index ($|\Delta|$) of the π_{dup} is lower than that of

TABLE V
Average Parameter Vectors π_{dup} and π_{gauss}, Default Parameters (π_{def}) Proposed in [18], and Silhouette Indices

Parameter	Parameter Vectors		
	π_{def}	π_{dup}	π_{gauss}
α_A^{min}	5.000	69.300	-
α_A^{max}	30.000	88.70	-
α_P^{min}	0.500	0.320	-
α_P^{max}	1.000	0.530	-
α_S^{min}	0.000	0.470	-
α_S^{max}	1.000	0.740	-
σ_{min}	-	-	0.290
σ_{max}	-	-	0.720
$\|\Delta\|$	0.153	0.047	0.040

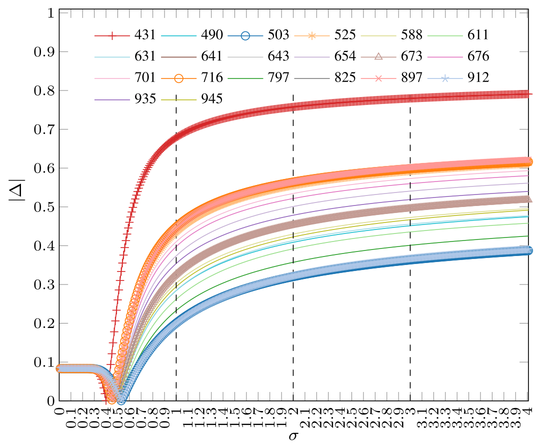


Fig. 4. Effect of Gaussian filter in the $|\Delta|$ for each writer of $20\mathcal{D}_\mathcal{L}$.

the π_{def}. Therefore, π_{dup} better represents the variability of these writers than π_{def} does.

Figure 4 shows the effect of the Gaussian filter on the silhouette index for each writer of $20\mathcal{D}_\mathcal{L}$. It can be seen that each writer has their own ideal sigma interval denoting their specific writer variability. Furthermore, even a simple noise addition technique needs some kind of optimization to generate more realistic synthetic samples.

A. Validation at Feature Level

Before discussing the impacts of the proposed optimization method in terms of performance, i.e., reduction of the EER in the signature verification system, we will present an analysis of the quality of the synthetic samples that are created by the duplicator and Gaussian filter using the average parameter vectors.

Since we are only using the feature vectors of handwritten signatures, it was necessary to perform the validation of the method at the feature level. For the same writer, it is expected that genuine signatures and synthetic samples will have similar aspects, and as a result, the signature and synthetic feature vectors should be similar as well. Besides the similarity, the synthetic samples are also expected to keep the original writer variability. It is important to highlight that the feature descriptor must be sufficiently discriminant to measure the dissimilarity between the signatures and synthetic samples [62].

TABLE VI
AVERAGE ABSOLUTE VALUES OF THE SILHOUETTE INDICES ($|\Delta|_{avg}$), AVERAGE SPARSITY OF GENUINE CLUSTERS (co_{avg}), AND STANDARD DEVIATIONS FOR GPDS-300, CEDAR, AND MCYT-75

Dataset	$\|\Delta\|_{avg}$			co_{avg}
	π_{def}	π_{dup}	π_{gauss}	
GPDS-300	0.14 ± 0.10	0.04 ± 0.05	0.04 ± 0.04	18860.60 ± 1854.13
CEDAR	0.70 ± 0.14	0.56 ± 0.18	0.28 ± 0.13	13788.87 ± 804.96
MCYT-75	0.37 ± 0.12	0.15 ± 0.10	0.10 ± 0.06	15900.48 ± 945.49

If the feature descriptor is poor, then the distinction between them will be poor as well [30].

In this experiment, for each writer, we used the first 12 genuine signatures to build the genuine cluster in the 2048 dimensional feature space created by the SigNet-F. Therefore, 4 clusters with 12 feature vectors were created: one with genuine signatures, one with duplicates created with the optimized parameter vector π_{dup}, one with the duplicates created with the default parameter vector π_{def}, and one with the feature vectors generated with the optimized parameter vector π_{gauss}.

To measure how close the synthetic samples and genuine signatures are, we used the absolute value of the silhouette index $|\Delta|$. As earlier explained, if both clusters have a similar variability, then $|\Delta| \to 0$. For each writer, three $|\Delta|$ values were computed using Equation 12: the $|\Delta|$ between the genuine cluster and the cluster of duplicates using the default parameter vector π_{def}, the $|\Delta|$ between the genuine cluster and the duplicates using the average parameter vector π_{dup}, and the $|\Delta|$ between the genuine cluster and the synthetic feature vectors using the average parameter vector π_{gauss}. Then, the average silhouette indices $|\Delta|_{avg}$ and the standard deviations were computed. To measure the sparsity of each dataset in the feature space, we used the average cohesion of all writers in each dataset. The cohesion co is computed using the sum of the squared differences between all elements $\phi(X_i)$ of the cluster and the cluster centroid μ (Equation 15) [71, p. 578]. For each dataset, we computed the cohesions of all genuine clusters and their average, co_{avg}. Table VI shows the $|\Delta|_{avg}$, the average sparsities co_{avg}, and standard deviations of the three datasets described in Section III-A.

$$co = \sum_{i=1}^{n} (\phi(X_i) - \mu)^2 \qquad (15)$$

Although the average parameter vector π_{dup} does not represent the individual variability of each writer, it does represent the variability better than the π_{def}. Besides, duplicates created using π_{def} may introduce some distortions that may lead to unnatural signature duplicates (Figure 5). As can be seen in Figure 5 and Table VI, minimizing the silhouette index helps improve the quality of duplicates. Furthermore, the average parameter vector π_{gauss} helps generate synthetic samples in the feature space that resemble the original signatures.

Since the average parameter vectors π_{dup} and π_{gauss} were optimized using a subset of GPDS-960, the lowest $|\Delta|_{avg}$ was achieved using the GPDS-300 dataset. Even using the signatures of just 20 writers, the method was able to represent the writer variability of 300 different writers. As well, using just the parameters optimized in the GPDS dataset, the proposed method proved able to better represent the writer variability in the CEDAR and MCYT-75 datasets.

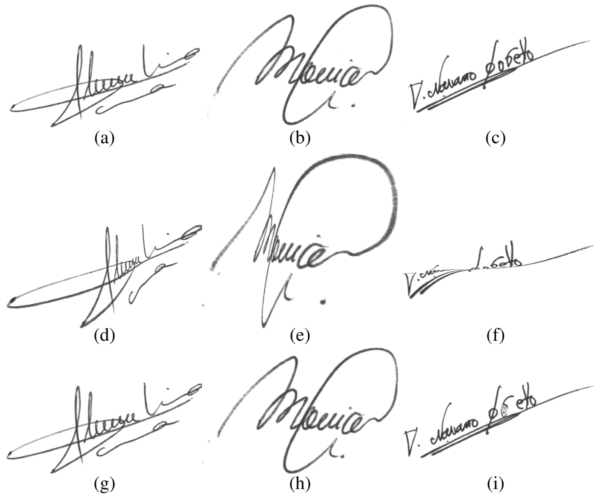


Fig. 5. Genuine signatures (a-c), duplicates (d-f) generated by Duplicator with default parameters, and duplicates (g-i) generated by Duplicator with the average parameter vector π_{dup}.

As can be seen, π_{def}, π_{dup}, and π_{gauss} have difficulty representing the variability of the writers present in the CEDAR dataset. In the case of π_{dup} and π_{gauss}, this may be due to the kind of signatures present in the GPDS dataset, where the parameters have been optimized. Since the GPDS dataset has highly sparse signatures in the feature space, it is expected that the parameter vector will be able to reproduce the sparsity of signatures with the same nature. According to Table VI, the writers in the CEDAR dataset present a smaller sparsity (co_{avg}) in the feature space than those writers in the other two datasets. Since we applied two indirect transformations and one direct one in the feature space, it can be seen that the nature of the transformations applied in the feature space also impacts the quality of synthetic samples. The results presented in Table VI suggest that it is more difficult to generate synthetic samples that are close to the low-sparsity clusters than it is to generate them close to high-sparsity ones.

B. Validation at Performance Level Using the Duplicator

To respect the constraints imposed by problems in the real world, where few genuine signatures per writer are available, no more than three genuine signatures per writer were used in these experiments to train the Writer-Dependent SVMs of the signature verification system described in Section III-H. The genuine signatures of other writers of the dataset were used as random forgeries to train the classifiers. Each genuine signature and forgery was randomly selected for training. During the testing, the genuine signatures and forgeries were randomly selected as well.

To better assess the impacts of the number of duplicates in reducing the EER, for each genuine signature, we created up to 22 duplicates. Furthermore, to compare the performance using the duplicates with that achieved by Hafemann *et al.* [62],

TABLE VII

DATASET SEPARATION INTO TRAINING AND TESTING. THE NUMBER OF GENUINE SIGNATURES G, RANDOM FORGERIES R, AND SKILLED FORGERIES S ARE SPECIFIED. THE NUMBER OF DUPLICATES USED FOR TRAINING ALSO IS SPECIFIED

Dataset	**Training set**		**Testing set**		
	G	R	G	R	S
GPDS-300	r ∈ {1, ..., 3} + r × (d ∈ {0, ..., 22})	(14 × 581) + (14 × 581 × d)	10	10	10
MCYT-75	r ∈ {1, ..., 3} + r × (d ∈ {0, ..., 22})	(10 × 74) + (10 × 74 × d)	5	-	15
CEDAR	r ∈ {1, ..., 3} + r × (d ∈ {0, ..., 22})	(12 × 54) + (12 × 54 × d)	10	-	10

we used the same experimental protocol. In our case, we used fewer original genuine signatures for training and included the duplicates during the training process. Table VII summarizes the separation of the dataset into training and testing. For each number of duplicates, the experiment was repeated 10 times, and the average EER and standard deviation were reported for the three datasets described in Section III-A. As in the previous experiments, duplicates were created using π_{def} and π_{dup}.

For each random forgery used to train the SVMs, from 0 to 22 duplicates were also used, meaning that the random forgeries and the corresponding duplicates were used for training. In GPDS, 14 genuine signatures of 581 other writers and the corresponding duplicates are used as random forgeries. For example, if we use 14 genuine signatures with 22 duplicates each for 581 writers, we have the original random forgeries, plus the corresponding duplicates ($(14 \times 581) + (14 \times 581 \times 22) = 187,082$). The classifiers were tested using 10 genuine signatures, 10 random forgeries and 10 skilled forgeries per writer from subset $\mathcal{E}_{\mathcal{T}}$.

For the MCYT-75 dataset, 10 genuine signatures of 74 other writers and the corresponding number of duplicates were used as random forgeries. The classifiers were tested using 5 genuine signatures and 15 skilled forgeries. For the CEDAR dataset, 12 genuine signatures of 54 other writers were used as random forgeries. The classifiers were tested using 10 genuine signatures and 10 skilled forgeries.

Figure 6 shows the average EER of each number of duplicates in the GPDS dataset. As expected, the greater the number of duplicates, the smaller the EER. It should, however, be noted that the system trained with the duplicates created with π_{dup} outperforms the one trained with π_{def} in all scenarios. This is somewhat attributable to the quality of the duplicates created using the optimized parameters. As shown in the previous section, the default parameters may sometimes lead to unnatural duplicates that negatively affect the performance of the verification system. Like the real signatures, the duplicates can also provide complementary information about the signatures of a writer.

Figure 7 shows the average EER of each number of duplicates in the MCYT-75 dataset. As observed earlier, when the number of duplicates increases, the EER drops simultaneously. This experiment also showed the generalization capability of the proposed method. Even though π_{dup} was optimized on a subset of GPDS-960, it was able to produce high quality duplicates for MCYT-75 as well, since this dataset contains highly variable signatures, similar to those found in the GPDS dataset.

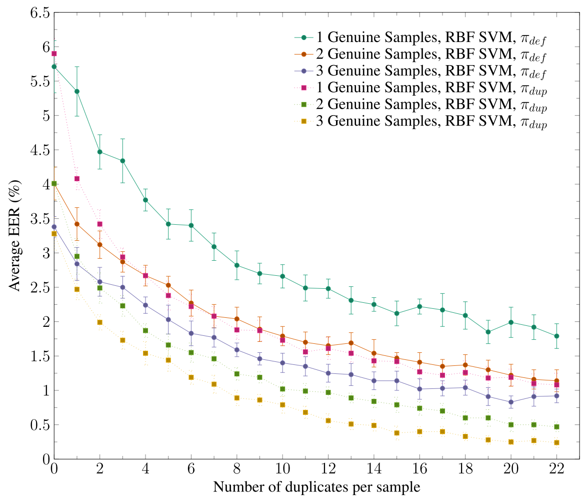


Fig. 6. Average EER achieved using GPDS-300 dataset, Signet-F and the Proposed Method with a Large Range of parameters.

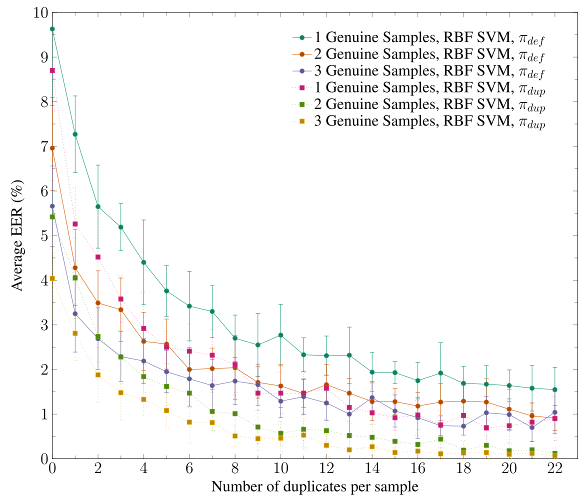


Fig. 7. Average EER achieved using MCYT-75 dataset, Signet-F and the Proposed Method with a Large Range of parameters.

Figure 8 shows the performance of each number of duplicates in the CEDAR dataset. As observed in the previous datasets, while the EER follows the same trend, the drop in the EER is, however, more subtle. This corroborates the hypothesis that the distribution of the CEDAR signatures is hard to represent due to the difference between the writer variability of the optimization dataset and the CEDAR dataset. Nevertheless, the performance achieved using duplicates generated with π_{dup} is better than that obtained using π_{def}.

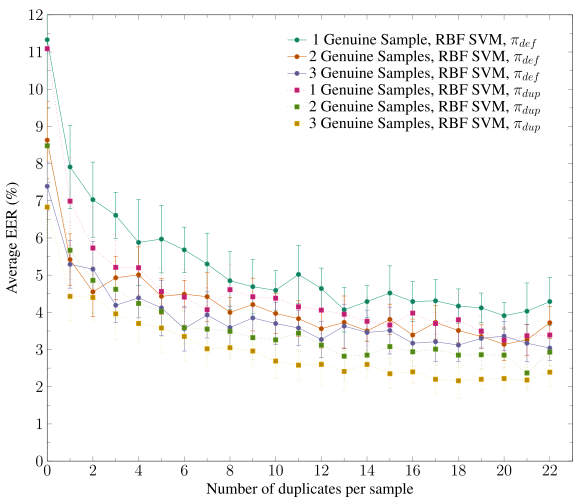


Fig. 8. Average EER achieved using CEDAR dataset, Signet-F and the Proposed Method with a Large Range of parameters.

Table VIII summarizes the best results achieved in each experiment and compares them with the signature verification system proposed in [62], which uses up to 12 genuine signatures; the one proposed in [28], which uses 10 and 12 genuine signatures, and the original duplicator proposed in [18]. It should be noted that the results published in [18] and [28] use other representations, and therefore, a direct comparison is not possible. As can be seen, the proposed method achieves outstanding results. Using the optimized parameter vector, it achieves state-of-the art results using no more than three genuine signatures.

Even duplicates generated using an π_{dup} provide additional data that enable the classifiers to learn to distinguish between genuine signatures and skilled forgeries. Similar to Frias-Martinez *et al.* [21] and Diaz *et al.* [18], the best results were achieved using between 15 and 22 duplicates (Table VIII). This may be an indication of the ideal number of duplicates that can be used to improve the performance of a verification system. Notwithstanding the fact that parameters were optimized using the GPDS dataset, the proposed method was able to generate real duplicates when it was tested using CEDAR and MCYT-75.

C. Validation at Performance Level Using the Gaussian Filter

As mentioned before, we hypothesized that adding some intraclass variability in the image space would induce some intraclass variability in the feature space as well, which allows measuring the fitness function in the feature space. To prove that this a valid strategy, we also present a method in which the feature data points increase when the duplicate feature points are placed on the feature domain. This was implemented by perturbing the genuine feature vector with correlated noise, which was added using a low-pass Gaussian filter (Equation 4).

We used the same protocol adopted in Section IV-B to evaluate the performance in the GPDS-300, MCYT-75, and CEDAR datasets. Instead of using the duplicates, we used the feature vectors with noise to train the SVM classifiers. To generate the feature vectors with noise, the standard deviation σ was defined considering the average parameter vector π_{gauss}.

Figures 9, 10, and 11 respectively show the performance on the GPDS-300, MCYT-75, and CEDAR datasets, respectively. The performance is summarized in Table VIII. As can be observed, most of the results achieved by the classifiers trained with the synthetic feature vectors achieved results similar to those of the proposed method reported in Section IV-B. For the CEDAR dataset, the synthetic feature vectors achieved

TABLE VIII

Summary of the Experimental Results Where #W, #S, and #D Stand for the Number of Writers Used for Training, the Number of Genuine Samples Used for Training, and the Number of Duplicates Per Sample Used for Training, Respectively

Reference	Feature	Classifier	Dataset	#W	#S	#D	EER (%)
Diaz et al., 2017 [18]	LDerivP	SVM	GPDS	300	2	20	21.63
					5	20	17.19
					8	20	14.58
			MCYT	75	2	20	16.06
					5	20	11.90
					8	20	9.12
Hafemann et al., 2017 [62]	SigNet-F	SVM	GPDS	300	5	0	2.42
					12	0	1.69
			CEDAR	55	4	0	5.92
					8	0	4.77
					12	0	4.63
			MCYT	75	5	0	3.70
					10	0	3.00
Zois et al. 2019 [28]	KSVD/OMP (F_3)	SVM	GPDS	300	12	0	0.70
			CEDAR	55	10	0	0.79
			MCYT	75	10	0	1.37
Baseline (Without Duplicates)	SigNet-F	SVM	GPDS	300	1	0	5.71
					2	0	4.01
					3	0	3.38
			CEDAR	55	1	0	11.33
					2	0	8.63
					3	0	7.39
			MCYT	75	1	0	9.63
					2	0	6.96
					3	0	5.66
Duplicator π_{def}	SigNet-F	SVM	GPDS	300	1	22	1.79
					2	22	1.14
					3	22	0.92
			CEDAR	55	1	22	4.29
					2	22	3.72
					3	22	3.04
			MCYT	75	1	22	1.55
					2	22	0.91
					3	22	1.04
Proposed Method $20\mathcal{D}_{\mathcal{L}}$ π_{dup}	SigNet-F	SVM	GPDS	300	1	22	1.08
					2	22	0.47
					3	22	**0.24**
			CEDAR	55	1	22	3.39
					2	22	2.93
					3	22	**2.39**
			MCYT	75	1	22	0.90
					2	22	0.12
					3	22	**0.07**
Proposed Method $20\mathcal{D}_{\mathcal{L}}$ π_{gauss}	SigNet-F	SVM	GPDS	300	1	22	1.04
					2	22	0.48
					3	22	**0.20**
			CEDAR	55	1	22	2.47
					2	22	1.31
					3	22	**0.82**
			MCYT	75	1	22	0.72
					2	22	0.12
					3	22	**0.01**

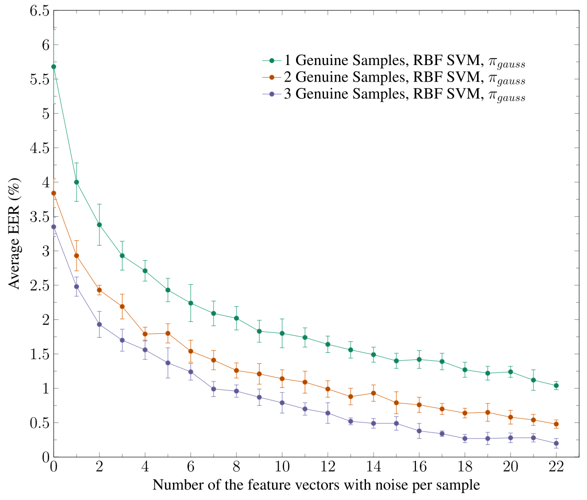


Fig. 9. Average EER achieved using GPDS-300 dataset, Signet-F and the feature vectors with noise.

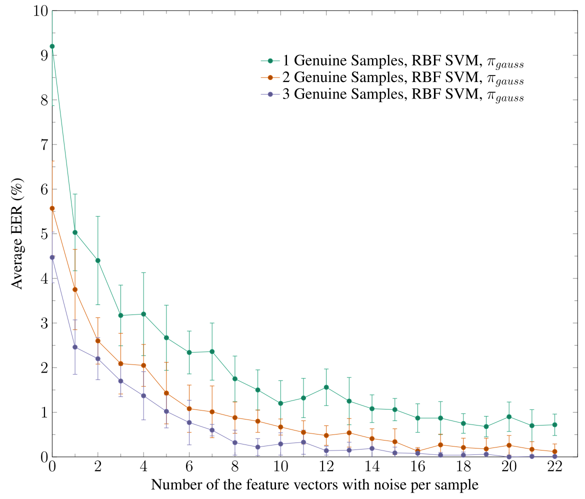


Fig. 10. Average EER achieved using MCYT-75 dataset, Signet-F and the feature vectors with noise.

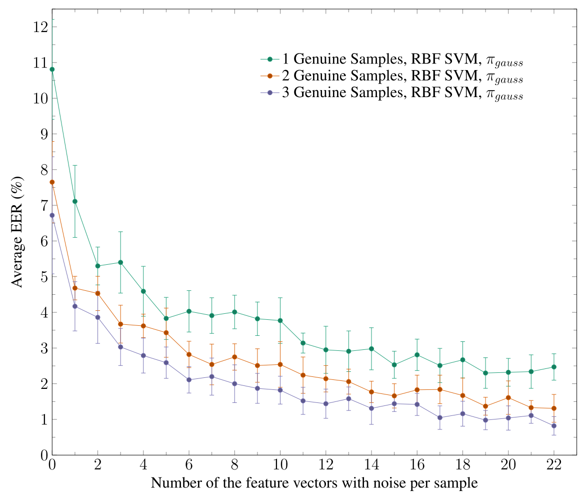


Fig. 11. Average EER achieved using CEDAR dataset, Signet-F and the feature vectors with noise.

better results than did the duplicates. According to Figure 4, the sigma interval regulates the noise intensity applied in the feature vectors. If the sigma interval is high, it will generate synthetic feature vectors with more noise. Therefore, these synthetic feature vectors will be far from the original feature vectors. Consequently, the EER will be higher than when a small interval is used.

The feature space augmentation does not provide the signature image that can be used by different feature descriptors. It is worth to mention that generating duplicated signatures with realistic appearances helps provide a better understanding of signature execution from several neuroscientific perspectives. Moreover, such synthetic specimens can be used with any feature extraction method [18].

V. Conclusion

In this work, we proposed a method to automatically optimize and select the parameters used to generate synthetic samples of offline handwritten signatures in the image and feature spaces. We showed that the proposed method can be used to increase the number of signatures used to train an automatic handwritten signature verification system. In addition to improving the performance of the system, the proposed method was able to represent writers' variability better than the parameters proposed by Diaz *et al.* (2017) [18]. Furthermore, a new approach to validate the writer variability of synthetic signature samples using their features was proposed. The experimental results support the hypothesis that the writer variability observed on the image space is reflected in the feature space as well.

The proposed method achieved an EER almost equal to zero in MCYT-75. This dataset is characterized by writers with a great variability. Since the writers from the GPDS dataset used to optimize the parameters also showed great variability, this behavior was expected. The proposed method achieved low EERs in the CEDAR dataset. However, the optimization of the six parameters here may not be enough to generate more compact clusters in a feature space such as in the CEDAR dataset. Therefore, more parameters can be optimized to solve this issue. The different distributions of the three datasets suggest that other transformations can be investigated to improve the performance in the CEDAR dataset. The proposed method showed interesting results for three different signature datasets based on the Latin alphabet. To verify its generalization capability for different writing systems, the method can also be evaluated using signatures based on other alphabetical systems.

Notwithstanding the fact that the feature space augmentation in our proposed method has a low computational complexity [25] and presented promising results, it nonetheless needs to have optimized parameters in order to generate more realistic samples. As well, it does not provide a signature image that can be used by different feature descriptors. Since the method can represent the variability of signatures during data augmentation in the image space, it can therefore be used to create more robust offline handwritten signature datasets. In addition, the generated signatures can be used to train more robust CNN models.

Acknowledgment

The authors would like to thank Prof. Diego Bertolini of the Federal University of Technology - Paraná, Prof. Daniel Weingaertner of the Federal University of Paraná; Luiz Gustavo Hafemann and Rafael Menelau Oliveira e Cruz of École de Technologie Supérieure.

REFERENCES

[1] S. Mitra and M. Gofman, *Biometrics in a Data Driven World: Trends, Technologies, and Challenges*. Boca Raton, FL, USA: CRC Press, 2017.

[2] M. J. Allen, *Foundations of Forensic Document Analysis: Theory and Practice*. Hoboken, NJ, USA: Wiley, 2016.

[3] S. Z. Li and A. Jain, *Encyclopedia of Biometrics*, 2nd ed. New York, NY, USA: Springer, 2015.

[4] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, and J. Ortega-Garcia, "Reducing the template ageing effect in on-line signature biometrics," *IET Biometrics*, vol. 8, no. 6, pp. 422–430, Nov. 2019.

[5] D. Impedovo and G. Pirlo, "Dynamic handwriting analysis for the assessment of neurodegenerative diseases: A pattern recognition perspective," *IEEE Rev. Biomed. Eng.*, vol. 12, pp. 209–220, 2019.

[6] K. M. Koppenhaver, *Forensic Document Examination: Principles and Practice*. Totowa, NJ, USA: Humana Press, 2007.

[7] M. Diaz, M. A. Ferrer, D. Impedovo, M. I. Malik, G. Pirlo, and R. Plamondon, "A perspective analysis of handwritten signature technology," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 117:1–117:39, 2019.

[8] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Fixed-sized representation learning from offline handwritten signatures of different sizes," *Int. J. Document Anal. Recognit.*, vol. 21, no. 3, pp. 1–14, 2018.

[9] J. Linden, R. Marquis, S. Bozza, and F. Taroni, "Dynamic signatures: A review of dynamic feature variation and forensic methodology," *Forensic Sci. Int.*, vol. 291, pp. 216–229, Oct. 2018.

[10] A. K. Jain, K. Nandakumar, and A. Ross, "50 years of biometric research: Accomplishments, challenges, and opportunities," *Pattern Recognit. Lett.*, vol. 79, pp. 80–105, Aug. 2016.

[11] D. D. Zhang, *Automated Biometrics: Technologies and Systems*. Norwell, MA, USA: Kluwer, 2000.

[12] M. I. Malik and M. Liwicki, "From terminology to evaluation: Performance assessment of automatic signature verification systems," in *Proc. Int. Conf. Frontiers Handwriting Recognit.*, Sep. 2012, pp. 614–618.

[13] E. N. Zois, A. Alexandridis, and G. Economou, "Writer independent offline signature verification based on asymmetric pixel relations and unrelated training-testing datasets," *Expert Syst. Appl.*, vol. 125, pp. 14–32, Jul. 2019.

[14] R. Plamondon and G. Lorette, "Automatic signature verification and writer identification—The state of the art," *Pattern Recognit.*, vol. 22, no. 2, pp. 107–131, Jan. 1989.

[15] D. Impedovo and G. Pirlo, "Automatic signature verification: The state of the art," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 609–635, Sep. 2008.

[16] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Offline handwritten signature verification—Literature review," in *Proc. 7th Int. Conf. Image Process. Theory, Tools Appl. (IPTA)*, Nov. 2017, pp. 1–8.

[17] M. Song and Z. Sun, "An immune clonal selection algorithm for synthetic signature generation," *Math. Problems Eng.*, vol. 2014, no. 15, pp. 1–12, 2014.

[18] M. Diaz, M. A. Ferrer, G. S. Eskander, and R. Sabourin, "Generation of duplicated off-line signature images for verification systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 5, pp. 961–964, Apr. 2017.

[19] B. Fang, C. H. Leung, Y. Y. Tang, P. C. K. Kwok, K. W. Tse, and Y. K. Wong, "Offline signature verification with generated training samples," *IEE Proc. Vis., Image Signal Process.*, vol. 149, no. 2, pp. 85–90, Apr. 2002.

[20] M. A. Ferrer, M. Diaz-Cabrera, A. Morales, J. Galbally, and M. Gomez-Barrero, "Realistic synthetic off-line signature generation based on synthetic on-line data," in *Proc. 47th Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2013, pp. 1–6.

[21] E. Frias-Martinez, A. Sanchez, and J. Velez, "Support vector machines versus multi-layer perceptrons for efficient off-line signature recognition," *Eng. Appl. Artif. Intell.*, vol. 19, no. 6, pp. 693–704, Sep. 2006.

[22] J. Galbally, J. Fierrez, M. Martinez-Diaz, and J. Ortega-Garcia, "Improving the enrollment in dynamic signature verfication with synthetic samples," in *Proc. 10th Int. Conf. Document Anal. Recognit.*, 2009, pp. 1295–1299.

[23] K. Huang and H. Yan, "Off-line signature verification based on geometric feature extraction and neural network classification," *Pattern Recognit.*, vol. 30, no. 1, pp. 9–17, Jan. 1997.

[24] C. Rabasse, R. M. Guest, and M. C. Fairhurst, "A new method for the synthesis of signature data with natural variability," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 3, pp. 691–699, Jun. 2008.

[25] V. Kumar, H. Glaude, C. de Lichy, and W. Campbell, "A closer look at feature space data augmentation for few-shot intent classification," in *Proc. 2nd Workshop Deep Learn. Approaches Low-Resource NLP (DeepLo)*, 2019, pp. 1–10.

[26] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in *Proc. Int. Symp. Med. Robot.*, 2015, pp. 121–126.

[27] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," in *Proc. 5th Int. Conf. Learn. Repr.*, 2017, pp. 1–12.

[28] E. N. Zois, D. Tsourounis, I. Theodorakopoulos, A. L. Kesidis, and G. Economou, "A comprehensive study of sparse representation techniques for offline signature verification," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 1, no. 1, pp. 68–81, Jan. 2019.

[29] M. B. Yilmaz and K. Öztürk, "Recurrent binary patterns and CNNs for offline signature verificatio," in *Proc. Future Technol. Conf.*, 2019, pp. 417–434.

[30] V. L. F. Souza, A. L. I. Oliveira, R. M. O. Cruz, and R. Sabourin, "A white-box analysis on the writer-independent dichotomy transformation applied to offline handwritten signature verification," *Expert Syst. Appl.*, vol. 154, Sep. 2020, Art. no. 113397.

[31] M. Diaz, M. A. Ferrer, S. Ramalingam, and R. Guest, "Investigating the common authorship of signatures by off-line automatic signature verification without the use of reference signatures," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 487–499, 2020.

[32] M. Diaz, A. Fischer, M. A. Ferrer, and R. Plamondon, "Dynamic signature verification system based on one real signature," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 228–239, Jan. 2018.

[33] M. Diaz, A. Fischer, R. Plamondon, and M. A. Ferrer, "Towards an automatic on-line signature verifier using only one reference per signer," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2015, pp. 631–635.

[34] M. A. Ferrer *et al.*, "Static and dynamic synthesis of Bengali and Devanagari signatures," *IEEE Trans. Cybern.*, vol. 48, no. 10, pp. 2896–2907, Oct. 2018.

[35] M. A. Ferrer, M. Diaz, C. Carmona-Duarte, and A. Morales, "A behavioral handwriting model for static and dynamic signature synthesis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1041–1053, Jun. 2017.

[36] J. Galbally, J. Fierrez, J. Ortega-Garcia, and R. Plamondon, "Synthetic on-line signature generation. Part II: Experimental validation," *Pattern Recognit.*, vol. 45, no. 7, pp. 2622–2632, Jul. 2012.

[37] J. Galbally, R. Plamondon, J. Fierrez, and J. Ortega-Garcia, "Synthetic on-line signature generation. Part I: Methodology and algorithms," *Pattern Recognit.*, vol. 45, no. 7, pp. 2610–2621, Jul. 2012.

[38] M. E. Munich and P. Perona, "Visual identification by signature tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 200–217, Feb. 2003.

[39] M. Diaz-Cabrera, M. A. Ferrer, and A. Morales, "Cognitive inspired model to generate duplicated static signature images," in *Proc. 14th Int. Conf. Frontiers Handwriting Recognit.*, Sep. 2014, pp. 61–66.

[40] M. Diaz-Cabrera, M. Gomez-Barrero, A. Morales, M. A. Ferrer, and J. Galbally, "Generation of enhanced synthetic off-line signatures based on real on-line data," in *Proc. 14th Int. Conf. Frontiers Handwriting Recognit.*, Sep. 2014, pp. 482–487.

[41] J. Galbally, M. Diaz-Cabrera, M. A. Ferrer, M. Gomez-Barrero, A. Morales, and J. Fierrez, "On-line signature recognition through the combination of real dynamic data and synthetically generated static data," *Pattern Recognit.*, vol. 48, no. 9, pp. 2921–2934, Sep. 2015.

[42] V. K. S. L. Melo, B. L. D. Bezerra, D. Impedovo, G. Pirlo, and A. Lundgren, "Deep learning approach to generate offline handwritten signatures based on online samples," *IET Biometrics*, vol. 8, no. 3, pp. 215–220, May 2019.

[43] M. Diaz *et al.*, "Multiple generation of bengali static signatures," in *Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Oct. 2016, pp. 42–47.

[44] M. Diaz, M. A. Ferrer, and R. Sabourin, "Approaching the intra-class variability in multi-script static signature evaluation," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 1147–1152.

[45] M. A. Ferrer, M. Diaz-Cabrera, and A. Morales, "Synthetic off-line signature image generation," in *Proc. Int. Conf. Biometrics (ICB)*, Jun. 2013, pp. 1–7.

[46] M. A. Ferrer, M. Diaz-Cabrera, and A. Morales, "Static signature synthesis: A neuromotor inspired approach for biometrics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 667–680, Mar. 2015.

[47] V. Ruiz, I. Linares, A. Sanchez, and J. F. Velez, "Off-line handwritten signature verification using compositional synthetic generation of signatures and siamese neural networks," *Neurocomputing*, vol. 374, pp. 30–41, Jan. 2020.

[48] M. Diaz, M. A. Ferrer, A. Parziale, and A. Marcelli, "Recovering western on-line signatures from image-based specimens," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 1204–1209.

[49] K. K. Lau, P. C. Yuen, and Y. Y. Tang, "Universal writing model for recovery of writing sequence of static handwriting images," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 05, pp. 603–630, Aug. 2005.

[50] E.-M. Nel, J. A. du Preez, and B. M. Herbst, "Estimating the pen trajectories of static signatures using hidden Markov models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1733–1746, Nov. 2005.

[51] V. Nguyen and M. Blumenstein, "Techniques for static handwriting trajectory recovery: A survey," in *Proc. 8th IAPR Int. Workshop Document Anal. Syst. (DAS)*, 2010, pp. 463–470.

[52] P. Morasso and F. A. Mussa Ivaldi, "Trajectory formation and handwriting: A computational model," *Biol. Cybern.*, vol. 45, no. 2, pp. 131–142, Sep. 1982.

[53] M. Djioua and R. Plamondon, "Studying the variability of handwriting patterns using the kinematic theory," *Human Movement Sci.*, vol. 28, no. 5, pp. 588–601, Oct. 2009.

[54] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[55] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "DBSMOTE: Density-based synthetic minority over-sampling TEchnique," *Int. J. Speech Technol.*, vol. 36, no. 3, pp. 664–684, Apr. 2012.

[56] G. Kurata, B. Xiang, and B. Zhou, "Labeled data generation with encoder-decoder LSTM for semantic slot filling," in *Proc. Interspeech*, Sep. 2016, pp. 725–729.

[57] X. Teng, T. Wang, X. Zhang, L. Lan, and Z. Luo, "Enhancing stock price trend prediction via a time-sensitive data augmentation method," *Complexity*, vol. 2020, pp. 1–8, Feb. 2020.

[58] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019.

[59] F. Vargas, M. Ferrer, C. Travieso, and J. Alonso, "Off-line handwritten signature GPDS-960 corpus," in *Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2007, pp. 764–768.

[60] M. K. Kalera, S. Srihari, and A. Xu, "Offline signature verification and identification using distance statistics," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 18, no. 07, pp. 1339–1360, Nov. 2004.

[61] L. Hu and Y.-H. Wang, "On-line signature verification based on fusion of global and local information," in *Proc. Int. Conf. Wavelet Anal. Pattern Recognit.*, Nov. 2007, pp. 295–306.

[62] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks," *Pattern Recognit.*, vol. 70, pp. 163–176, Oct. 2017.

[63] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.

[64] Y. Zhang, X. Liu, F. Bao, J. Chi, C. Zhang, and P. Liu, "Particle swarm optimization with adaptative learning strategy," *Knowl. Based Syst.*, vol. 196, Jun. 2020, Art. no. 105789.

[65] F. Zhao, "Optimized algorithm for particle swarm optimization," *Int. J. Math. Comput. Phys. Elect. Comput. Eng.*, vol. 10, pp. 96–100, Feb. 2016.

[66] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[67] K. Y. Huang, "A hybrid particle swarm optimization approach for clustering and classification of datasets," *Knowl.-Based Syst.*, vol. 24, no. 3, pp. 420–426, Apr. 2011.

[68] S. Salehi, A. Selamat, M. Reza Mashinchi, and H. Fujita, "The synergistic combination of particle swarm optimization and fuzzy sets to design granular classifier," *Knowl.-Based Syst.*, vol. 76, pp. 200–218, Mar. 2015.

[69] C. Fan, B. Hou, J. Zheng, L. Xiao, and L. Yi, "A surrogate-assisted particle swarm optimization using ensemble learning for expensive problems with small sample datasets," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106242.

[70] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[71] P. N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. London, U.K.: Pearson, 2018.

Teruo M. Maruyama received the B.S. degree in computer engineering and the M.Sc. degree in applied computing from the State University of Ponta Grossa, Ponta Grossa, Brazil, in 2013 and 2017, respectively. He is currently pursuing the Ph.D. degree in informatics with the Federal University of Paraná, Curitiba, Brazil. His current interests include pattern recognition, machine learning, and image processing.

Luiz S. Oliveira received the B.S. degree in computer science from Unicenp, Curitiba, Brazil, the M.Sc. degree in electrical engineering and industrial informatics from the Centro Federal de Educação Tecnológica do Paraná (CEFET-PR), Curitiba, and the Ph.D. degree in computer science from the École de Technologie Supérieure, Université du Québec, in 1995, 1998, and 2003, respectively. From 2004 to 2009, he was a Professor with the Computer Science Department, Pontifical Catholic University of Paraná, Curitiba. In 2009, he joined the Federal University of Paraná, Curitiba, where he is currently a Professor with the Department of Informatics and the Head of the Graduate Program in computer science. His current interests include pattern recognition, machine learning, image analysis, and evolutionary computation.

Alceu S. Britto Jr., received the M.Sc. degree in industrial informatics from the Centro Federal de Educação Tecnológica do Paraná (CEFET-PR), Brazil, in 1996, and the Ph.D. degree in computer science from the Pontifícia Universidade Católica do Paraná (PUCPR), Brazil, in 2001. In 1989, he joined the Informatics Department, Universidade Estadual de Ponta Grossa (UEPG), Brazil. In 1995, he also joined the Computer Science Department, PUCPR, and in 2001, the Master's Program in Informatics (PPGIa). His current interests include pattern recognition, machine learning, image analysis, and evolutionary computation.

Robert Sabourin (Member, IEEE) joined the Physics Department, Montreal University, in 1977, where his main contribution was the design and implementation of a microprocessor-based fine tracking system combined with a low-light level CCD detector. In 1983, he joined the staff of the École de Technologie Supérieure, Université du Québec, Montreal, QC, Canada, where he co-founded the Department of Automated Manufacturing Engineering. He is currently a Full Professor with the Université du Québec, and teaches pattern recognition, evolutionary algorithms, neural networks, and fuzzy systems. In 1992, he joined the Computer Science Department, Pontificia Universidade Católica do Paraná, Curitiba, Brazil. Since 1996, he has been a Senior Member of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University. Since 2012, he has been the Research Chair specializing in adaptive surveillance systems in dynamic environments. He is the author or coauthor of more than 450 scientific publications, including journals and conference proceedings. His research interests include adaptive biometric systems, adaptive classification systems in dynamic environments, dynamic classifier selection, and evolutionary computation.