TERUO MATOS MARUYAMA

AUTOMATIC INTRAPERSONAL VARIABILITY MODELING FOR OFFLINE SIGNATURE AUGMENTATION

(*versão pré-defesa, compilada em 17 de junho de 2021*)

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Luiz Eduardo Soares de Oliveira.

Coorientador: Robert Sabourin e Alceu de Souza Britto Jr..

CURITIBA PR

2021

RESUMO

Normalmente, em um cenário do mundo real, poucas assinaturas estão disponíveis para treinar um sistema de verificação automática de assinaturas (SVAA). Para resolver esse problema, diversas abordagens para a duplicação de assinaturas estáticas foram propostas ao longo dos anos. Essas abordagens geram novas amostras de assinaturas sintéticas aplicando algumas transformações na imagem original da assinatura. Algumas delas geram amostras realistas, especialmente o duplicator. Este método utiliza um conjunto de parâmetros para modelar o comportamento do escritor (variabilidade do escritor) ao assinar. No entanro, esses parâmetros só empiricamente definidos. Este tipo de abordagem pode ser demorado e pode selecionar parâmetros que não descrevem a real variabilidade do escritor. A principal hipótese desse trabalho é que a variabilidade do escritor observada no domínio da imagem também pode ser transferido para o domínio de características. Portanto, este trabalho propõe um novo método para modelar automaticamente a variabilidade do escritor para a posterior duplicação de assinaturas no domínio de imagem (duplicator) e domínio de característias (filtro Gaussiano e variação do método de Knop). Este trabalho também propõe um novo método de duplicação de assinaturas estáticas, que gera as amostras sintéticas diretamente no domínio de características usando um filtro Gaussiano. Além disso, uma nova abordagem para avaliar a qualidade de amostras sintéticas no domínio de características é apresentada. As limitações e vantagens de ambas as abordagens de duplicação de assinaturas também são exploradas. Além the usar a nova abordagem para avaliar a qualidade das amostras, o desempenho de um SVAA é avaliado usando as amostras e três bases the assinaturas estáticas bem conhecidas: a GPDS-300, a MCYT-75 e a CEDAR. Para a GPDS-300, quando o classificador SVM foi treinando com somente uma assinatura genuína por escritor, ele obteve um Equal Error Rate (EER) de 5,71%. Quando o classificador também utilizou as amostras sintéticas geradas no domínio de imagem, o EER caiu para 1,08%. Quando o classificador foi treinado com as amostras geradas pelo filtro Gaussiano, o EER caiu para 1,04%.

Palavras-chave: Biometria, Variabilidade do Escritor, Verificação de Assinaturas Manuscritas Estáticas

ABSTRACT

Normally, in a real-world scenario, there are few signatures available to train an automatic signature verification system (ASVS). To address this issue, several offline signature duplication approaches have been proposed along the years. These approaches generate a new synthetic signature sample applying some transformations in the original signature image. Some of them generate realistic samples, specially the duplicator. This method uses a set of parameters to model the writer's behavior (writer variability) during the signing act. However, these parameters are empirically defined. This kind of approach can be time consuming and can select parameters that do not describe the real writer variability. The main hypothesis of this work is that the writer variability observed in the image space can be transferred to the feature space as well. Therefore, this work proposes a new method to automatically model the writer variability for further signature duplication in the image (duplicator) and the feature space (Gaussian filter and a variation of the Knop's method). This work also proposes a new offline signature duplication method, which directly generates the synthetic samples in the feature space using a Gaussian filter. Furthermore, a new approach to assess the quality of the synthetic samples in the feature space is introduced. The limitations and advantages of both signature augmentation approaches are also explored. Despite using the new approach to assess the quality of the samples, the performance of an ASVS was assessed using them and three well-known offline signature datasets: GPDS-300, MCYT-75, and CEDAR. For the GPDS-300, when the SVM classifier was trained with only one genuine signature per writer, it achieved an Equal Error Rate (EER) of 5.71%. When the classifier also was trained with the synthetic samples generated in the image space, the EER dropped to 1.08%. When the classifier was trained using the synthetic samples generated by the Gaussian filter, the EER dropped to 1.04%.

Keywords: Biometrics, Writer Variability, Offline Handwritten Signature Verification

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ACC	Accuracy
AER	Average Error Rate
AIRS	Artifical Immune Recognition System
AUC	Area Under the Curve
BFS	Boosting Feature Selection
BMP	Bitmap
CEDAR	Center of Excellence for Document Analysis and Recognition
CSD1	Corpus Signature Database One
CNN	Convolutional Neural Network
DCGAN	Deep Convolutional Generative Adversarial Network
DPDF	Directional Probability Density Functions
dpi	Dots per Inch
DRT	Discrete Radon Transform
DSC-BFS	Decision Stump Commitree Boosting Feature Selection
EER	Equal Error Rate
ESC	Extended Shadow Code
FAR	False Acceptance Rate
FAR_{random}	False Acceptance Rate for Random Forgeries
$FAR_{skilled}$	False Acceptance Rate for Skilled Forgeries
FDE	Forensic Document Examiner
FLD	Fisher Linear Discriminant
FMR	False Match Rate
FNMR	False Non-match Rate
FRR	False Rejection Rate
FV	Fisher Vector
GAN	Generative Adversarial Network
GPDS	Grupo de Procesado Digital de Señales
GLBP	Gradient Local Binary Patterns
GLCM	Gray Level Co-ocurrence Matrix
GLM	Generalized Linear Model
HMM	Hidden Markov Models
HOG	Histogram of Oriented Gradients
HOT	Histogram of Templates
JPEG	Joint Photographic Experts Group

KNN	K-Nearest Neighbors
LBP	Local Binary Patterns
LDerivP	Local Derivative Patterns
LDP	Local Directional Patterns
LRF	Longest Run Feature
MCYT	Ministerio de Ciencia y Tecnología
MLP	Multi-Layer Perceptron
MSDN	Mutual Signature DenseNet
OC-LBP	Orthogonal Combination of Local Binary Patterns
OC-PCA	One-Class Principal Component Analysis
OC-SVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
PHOG	Pyramid Histogram of Oriented Gradients
PNG	Portable Network Graphics
PNN	Probabilistic Neural Network
Posets	Partially Ordered Sets
PSO	Particle Swarm Optimization
PUC-PR	Pontifícia Universidade Católica do Paraná
ReLU	Rectified Linear Units
RBF	Radial Basis Function
RPF	Ring-Peripheral Features
RLD	Run Length Distribution
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
SVS	Signature Verification System
TIFF	Tagged Image File Format
VLAD	Vector of Locally Aggregated Descriptors

LISTA DE SÍMBOLOS

$\|\Delta\|_{avg}$	Average Absolute Silhouette Index
π_{avg}	Average Parameter Vector
π_{dup}	Average Parameter Vector for the Duplicator
π_{gauss}	Average Parameter Vector for the Gaussian Filter
π_{knop}	Average Parameter Vector for the Variation of the Knop's Method
$\|\Delta\|$	Absolute Silhouette Index
π_{def}	Default Parameter Vector
Δ	Silhouette Index
ψ	Skew
η	Threshold

SUMÁRIO

1 INTRODUCTION

The traditional identity recognition methods use some previous knowledge or an object to confirm the identity of a person. The identity representations used by these methods can be lost, shared, or stolen, compromising the safety of the user. With the increasing need for safer security systems, biometrics have been occupying an academic and commercial prominent position. The biometric verification methods use the physical, behavioral, or chemical traits of a person to recognize him/her [73, p. 1-3] [68]. One of the behavioral biometric traits is the handwritten signature of the individual [88].

1.1 CHALLENGES

The handwritten signature can represent the name of an individual ranging from a simple abbreviation to a full name. Furthermore, it can present a legible or a flourished form [93, p. 309]. According to Allen (2016) [3], the signature is the most used type of handwriting by some writers. Each writer has a particular behavior when signing. Several aspects, such as age [50, 109], physical [124] and emotional state [69], health [69], culture [23, 38], and others can contribute to this behavior [93, p. 6]. Due to this particular behavior, signatures written by the same writer never are exactly alike. This is referred to as intrapersonal variability or writer variability. Furthermore, the signatures of different writers does not have the exact same visual aspect. This is called interpersonal variability [92]. Due to the interpersonal variability, the signatures are widely accepted and used to verify a person's identity [21] [140].

Considering the signature acquisition process, the signatures can be classified into online and offline signatures. While the online signature is acquired storing the dynamic information about it, the offline signature is acquired storing the static information about the signature. The dynamic information is acquired using a special device that can register the trajectory, speed, and pressure of the signature along the time. The static information is acquired using a camera or scanner digitizing the image of the signature, written in a surface like a piece of paper. Due to the lack of the dynamic information, it is harder to verify an individual's identity with offline signatures than with online signatures [22].

Because of the importance attached to signatures, they have been the target of falsification. There are three main types of forgeries. A random forgery is created when a forger uses his/her own name and does not know the target's name. A simple forgery is created when a forger only knows the target's name and uses it to create the forgery. It is called skilled forgery when a forger knows the target's name, and trains to reproduce the target's signature. For skilled forgeries, the forger tries to mimic the target's behavior imitating the visual aspect, trajectory, pressure, and speed of the signature. With this in mind, the automatic signature verification systems

were designed to avoid and to identify forgeries. These systems try to automatically divide the signatures into genuine signatures and forgeries [92].

1.2 MOTIVATION

Despite the promising results presented by the automatic offline signature verification systems, several of them require a great number of signatures per writer to achieve such results [56, 58, 123, 136, 146, 147]. Since the signature acquisition is costly and a time-consuming process, normally, this number of signatures is not available for use. Moreover, it requires several privacy, bureaucratic, legal, and security measures [122, 22]. To solve this issue some solutions like the search for the combination of the best features and classifiers [145, 60, 17, 56, 136, 146], the use of one-class classifiers [53, 60], and the use of offline signature augmentation approaches [65, 35, 48, 41, 42, 19, 23, 21, 39, 38, 109, 135] were proposed in the literature. Among these techniques, the offline signature augmentation approaches can be highlighted.

Offline signature augmentation techniques are used to synthetically generate signature samples to train a signature verification system. While duplication distorts the real signatures to generate synthetic signatures or duplicates [65, 35, 48, 41, 42, 19, 23, 21, 39, 38, 135], the composition uses a set of geometric shapes to create synthetic signatures [109]. The duplication techniques based on human behavior have presented more realistic duplicates [41, 42, 19, 23, 39, 21, 38], specially the method proposed by Diaz et al. (2017a) [21].

Diaz et al. (2017a) [21] argue that the writer variability can be modeled by a global set of parameters or a global parameter vector. Despite of different writers have their own behavior when they write, some of these behaviors can be shared by them during the writing process. Consequently, these common variability traits can be modeled by a global parameter vector. The authors tested their hypothesis with two signature datasets and using a predefined set of parameters. However, they defined the parameters empirically and manually. They considered two factors: the visual aspect of the duplicates and the performance of an automatic signature verification system. They selected the parameters that generated the most human-like duplicates and achieved the best performance using the automatic signature verification system [92].

1.3 OBJECTIVES

This work proposes a method to automatically model the writer variability. It is modeled using a set of offline signatures to optimize a global parameter vector. Like Diaz et al. (2017a) [21], the parameter vector describes the most common behavioral traits shared by a group of writers. With the writer variability model, it is possible to generate more realistic duplicates. Moreover, this work proposes a validation method to assess the quality of duplicates in the feature space using their writer variability. It also presents two methods to generate offline synthetic samples directly in the feature space using a Gaussian filter [85] and the variation of the Knop's method [77]. Therefore, the objectives of this work are:

- To automatically model the writer variability;
- To present an alternative to increase the number of signatures that are used in automatic offline signature systems;
- To generate more realistic synthetic signature samples (duplicates) using the writer variability model;
- To improve the performance of a verification system with the duplicates generated using the proposed method;
- To explore the effect of the duplicates on a verification system;

1.4 HYPOTHESIS

Considering a sufficient discriminant feature descriptor, the main hypothesis of this work is that the writer variability observed on the image space can be transferred to the feature space as well. Therefore, the proposed method tries to automatically model the writer variability considering only the feature space. Consequently, this work does not assess the visual aspect of the synthetic samples directly.

1.5 CONTRIBUTIONS

The main contributions of this work are listed as follows:

- It is presented a method to automatically model the writer variability using offline signatures, which was published in [92];
- For the first time, it is presented a method to generate duplicates of offline signature samples directly in the feature space with a Gaussian filter, which was also published in [92];
- It is presented a new approach to assess the quality of duplicates in the feature space considering the writer variability, which was also published in [92];
- The experiments show how the writer variability model can be used to generate more realistic duplicates in the image and the feature domain;
- The experiments details the effect of duplicates on an offline signature verification system;
- The advantages and limitations of both signature augmentation approaches are explored in this work.

1.6 DOCUMENT STRUCTURE

The next chapters are organized as follows: Chapter 2 presents the biometrics, writing, and signature concepts. Chapter 3 shows the state of the art of the offline handwritten signature verification. Chapter 4 describes the proposed method for modeling the intrapersonal writer variability and associated concepts. Moreover, it shows how the proposed method can be used to generate more realistic synthetic signature samples in the image and the feature space using three offline signature augmentation approaches. Chapter 5 shows how these synthetic samples can be assessed, using their features and the performance of an automatic signature verification system. Furthermore, it explores the advantages and limitations of the signature augmentation approaches. Finally, Chapter 6 presents the conclusions and the references.

2 THEORETICAL FOUNDATION

To better understand this work, the concepts of biometrics, writing, and signature are presented in the following sections. The related concepts are also presented.

2.1 BIOMETRICS

The word biometrics was derived from the Greek words *bio* (life) and *metric* (to measure) [119, p. 1]. Biometrics consists of the recognition of an individual using his/her chemical, physical, or behavioral traits. Unlike the traditional methods that need some knowledge or the possession of an object, biometrics uses the individual traits to recognize him/her. The identity representation used by the traditional methods can be lost, shared, handled, or stolen, compromising the security of a system. Due to the constant need for more secure systems, biometrics has gained a prominent position commercially and academically [72] [68].

2.1.1 Verification and Identification

The verification compares the biometric pattern of an individual with a pattern stored in a database. The individual who will be recognized claims that he/she has a determined identity, and the system performs a one-to-one comparison to determine if the statement is true or false. This procedure avoids different people use the same identity [73, p. 6]. The verification can be formally expressed as a bi-class problem (Equation 2.1). For a set of input features X_Q and an identity statement I, determine if (I, X_Q) belongs to ω_1 or ω_2. In which, ω_1 is the true statement, and ω_2 is the false statement. Therefore, X_Q is compared with X_I (set of features of the user stored in the database) to determine which class the user belongs to. The function $S(X_Q, X_I)$ measures the similarity between the input (X_Q) and the feature vectors of the user (X_I) stored in the database, using a threshold η [107, p. 8-9].

$$(I, X_Q) \in \begin{cases} \omega_1, & \text{if } S(X_Q, X_I) \geq \eta \\ \omega_2, & \text{otherwise} \end{cases} \tag{2.1}$$

The identification compares the biometric pattern of the individual with the patterns stored in the database. The system performs a one-to-many comparison to determine if the identity belongs to the individual. The identity of the individual will not be determined in case of the biometric pattern is not registered in the database. This procedure avoids that a person uses multiple identities [73, p. 6]. The identification can be formally expressed as a problem of multiple classes (Equation 2.2). For a set of input features X_Q, determine the identity $I_k, k \in \{1, 2, ...M, M + 1\}$. In which, $I_1, I_2, ...I_M$ are the M identities in the system, and I_{M+1} indicates the rejection case when none of the identities can be assigned to the input. Furthermore,

X_{I_k} is the biometric pattern corresponding to the identity I_k, and η is a predefined threshold [107, p. 9-10]

$$X_Q \in \begin{cases} I_K, & \text{if } K = \arg\max_k \{S(X_Q, X_{I_k})\} \text{ and } S(X_Q, X_{I_k}) > \eta \\ I_{M+1}, & \text{otherwise} \end{cases} \tag{2.2}$$

In the both formulations, 2.1 and 2.2, the function can represent a similarity, dissimilarity, or a distance metric. Greater is the dissimilarity or distance between X_Q and X_I, smaller is the correspondence between them. On the other hand, the greater is the similarity between X_Q and X_I, the greater is the correspondence between them [107, p. 10].

2.1.2 Performance

Several factors can affect a biometric system. The adverse conditions of the sensor, the alteration of the user biometric traits, changes in the environmental conditions, and the interaction between sensor and user can affect the system. Therefore, it is difficult for a biometric system to provide a perfect match between two biometric samples of the same user. Consequently, the distance between the sets of features of these samples generally is different from zero [107, p. 10].

Another factor that can affect a system is the variability of the biometric traits. The variability observed in a set of biometric traits of the same individual is named as intrapersonal variability. The variability observed in a set of biometric traits of different individuals is named as interpersonal variability. During the development of the biometric system, one of the objectives is to minimize the intrapersonal and maximize the interpersonal variability [73, p. 7].

A similarity metric indicates how similar two sets of biometric traits are. When the same biometric trait belongs to the same individual, it is called true, authentic, or genuine. Otherwise, it is called false. A false rejection or error type I happens when a genuine sample is considered as a false sample. In terms of a threshold, it happens when a value is lower than the threshold η. A false acceptance or error type II happens when a false sample is considered as a genuine sample. In terms of a threshold, it occurs when the value is greater than or equals to the threshold η (Figure 2.1) [107, p. 10-12].

The *False Acceptance Rate* (FAR) or *False Match Rate* (FMR) is the percentage of false values that exceeds the threshold η (Equation 2.3) [73, p. 8]. In which, *FA* represents the number of false acceptances and *N* represents the total number of samples [132, p. 35].

$$FAR = \frac{FA \times 100}{N} \tag{2.3}$$

The *False Rejection Rate* (FRR) or *False Non-match Rate* (FNMR) can be defined as a percentage of true values that is lower than the threshold η (Equação 2.4) [107]. In which, *FR* is the number of false rejections and *N* is the total number of samples [132, p. 35].

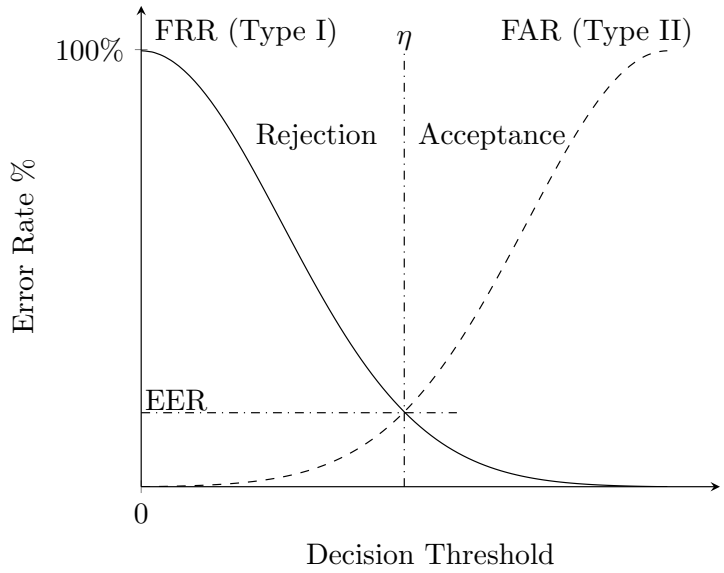


Figura 2.1: Equal Error Rate representation.
Source: Adapted from Sayeed et al. (2010) [113].

$$FRR = \frac{FR \times 100}{N} \tag{2.4}$$

When the threshold η changes, the values of FRR and FAR changes as well. However, when the threshold changes in a biometric system, it is not possible to decrease both errors at the same time [107, p. 12]. In this case, the Average Error Rate (AER) and the Equal Error Rate (EER) are used to have an overview of FAR and FRR. The AER is calculated using the mean of FAR and FRR (Equação 2.5) [49].

$$AER = \frac{FAR + FRR}{2} \tag{2.5}$$

The EER represents the intersection between FAR and FRR (Figure 2.1). The EER is widely used due to its simplicity and its capacity to summarize the performance of a biometric system [49]. Lower is the EER, better is the performance of the system [132, p. 35]. However, in practice the distributions of FRR and FAR are not continuous, and the intersection point between both errors may not exist. Therefore, the interval $[EER_{min}, EER_{max}]$ or simply the EER can be reported. Considering that the threshold η ranges from 0 to 100 and the EER is in an interval, the threshold η can be divided into η_1 (Equation 2.6) and η_2 (Equation 2.7). After determining the interval of the EER, it is possible to calculate it using the mean of EER_{min} and EER_{max}(Equation 2.9) [74, p. 149-151].

$$\eta_1 = \max\{\eta | FRR(\eta) \leq FAR(\eta)\} \tag{2.6}$$

$$\eta_2 = \min\{\eta | FRR(\eta) \geqslant FAR(\eta)\} \tag{2.7}$$

$$[EER_{min}, EER_{max}] \in \begin{cases} [FRR(\eta_1), FAR(\eta_1)], \text{if } FRR(\eta_1) + FAR(\eta_1) \leq FAR(\eta_2) + FRR(\eta_2) \\ [FAR(\eta_2), FRR(\eta_2)], \text{otherwise} \end{cases} \tag{2.8}$$

$$EER = \frac{EER_{min} + EER_{max}}{2} \tag{2.9}$$

2.2 WRITING

Writing consists of the transmission of ideas in a permanent or semi-permanent form using figures, drawings, or manuscripts. It began to develop due to the need for human beings to transmit ideas to each other. In the early years of humanity, humans have already registered their ideas in cavern walls with figures and paintings. It is believed that this type of register began around 20,000 to 10,000 BC. Lately, these figures were associated with words and became the ideograms. The ideograms were adopted by Sumerians, Chinese, Aztecs, Mayas, and Egyptians [78, p. 1].

Afterward, the figures became symbols, and the symbols started to represent syllables and sounds. The symbols that are used to represent syllables or sounds are called phonographs. Over time, these phonetic symbols were simplified and became the phonetic alphabet [78, p. 1]. The first known phonetic alphabet was created by the Phoenicians [91, p. 30]. This alphabet is composed of just 22 consonants, which represented the phonemes of the spoken language. Due to the Phoenician economy was based on fishing and trading, their alphabet was disseminated through the Eastern Mediterranean [51, p. 53].

The Phoenician alphabet has given rise to the Greek alphabet. In about 800 B. C., another alphabet called Aramaic was developed in the cities of modern Syria (then called Aram) [51, p. 51, 53]. The Hebrew, Arabic, and Indian alphabets arose due to the evolution of Aramaic alphabet [14, p. 10]. The Greek alphabet is composed of 24 letters, in which some of them are vowels [78, p. 2]. The word alphabet has origin of the two first letters of the Greek alphabet, alpha and beta [87, p. 7]. Subsequently, the Greek alphabet gave origin to the Cyrillic alphabet, that is used by Russia and the countries of Eastern Europe [78, p. 3]. It is believed that the Greek alphabet also gave origin to the angular Runic alphabet that was used by nations that lived in North Europe, Great-Britain, Scandinavia, and Iceland [7, p. 58].

The Greek alphabet has given rise to the Etruscan alphabet, which is the precursor of the Latin alphabet used by the Romans. This alphabet started with 21 letters [7, p. 58-60]. Due to the expansion of the Roman empire, the Latin alphabet disseminated and developed itself to use 26 letters [87, p. 9]. Each letter of the Latin alphabet is named according to the alphabet used in nowadays [78, p. 2].

In the 5th century, during the Roman empire weakening, the Teutons started to invade the Roman territory. Since Teutons did not have their own alphabet and the Latin alphabet covered the diversity of sounds in the Teuton language, they adopted the Latin alphabet. Some of Teutons were settled in the Grain-Britain and in the north Europe giving rise to the German and English. Others were settled in the region of France and Gaul giving rise to the French. Some of the them were settled in the region of Spain giving rise to the Spanish [7, p. 66-67]. Others were settled in the region of Portugal giving rise to the Portuguese [27, p. 34-42] [7, p. 67-68].

The words were written without space until the end of the 5th century. The people realized that the words should be separated when the silence reading developed. When Emperor Charlemagne unified Eastern Europe, between the centuries 8th and 9th, the lowercase and uppercase letters began to be used in the Latin manuscripts [87, p. 10-11]. Subsequently, the written became slightly inclined to the right side, the lowercase case letters became more rounded, and some of them began to connect each other [46, p. 13]. The simplification of the letter form turned the copy of manuscripts simpler and faster [78, p. 2].

Between the centuries 7th and 8th, the Gothic handwritten style emerged in the Northwest Europe [87, p. 11-12] [26, p. 475]. Due to its clear and compact aspect, it was widely used by the book copyists between the centuries 7th and 15th. This style spread through several European countries such as Germany, France, England, Spain, and Switzerland [46, p. 14] [91, p. 128-297] [45, p. 151-152, 251, 274, 279]. The first font styles of the western printed books have been inspired in this letter style [46, p. 14].

Little by little the Gothic style was losing popularity in Europe and a new style began to replace it [45, p. 279]. Around 1400, the Humanist writing style [64] appeared in Italy. France, Spain, and part of Switzerland quickly adopted this new style. England only adopted the new style at the end of the 17th century. The Gothic style remained strong in Germany until the 20th century, with the fall of the Nazi regime [46][45, p. 281-282].

Between the 14th and 15th centuries, a variation of the Humanist writing style, the Italic style, was created. The creation of this style is attributed to the Italian scholar, Niccolo Niccoli [87, p. 12]. However, it was only in 1522 that the Vatican clerk Ludovico Arrighi popularized this style of writing [78, p. 2]. In the 16th century, Italic writing already had many characteristics similar to the current cursive writing [87, p. 12]. Due to this fact, the Italic style is considered the precursor of the cursive writing [78, p. 2].

Between the 15th and 17th centuries, in the great navigations time, the European explorers have disseminated their writing systems in the Americas. The Germanic and Latin writing systems influenced the Americas. While the English system influenced the United States, most of Canada, and Guyana, the French system influenced the province of Quebec in Canada and French Guyana. The Dutch system influenced Suriname. While the Spanish influenced most countries in Latin America, the Portuguese writing system influenced only Brazil [78, p. 3].

The handwriting has been studied for more than 400 years. However, Camilo Baldi (1622) is considered the first person to perform methodological observations about handwriting.

In 1897, the French Abb Jean-Hippolyte Michon created the term graphology to refer to the study of handwriting. He also founded the Graphology Society, and he is considered the first person to provide a scientific basis for handwriting analysis. Continuing the work of Michon, J. Crépieus-Jamin divided the handwriting into seven fundamental elements: speed, pressure, shape, dimension, continuity, direction, and order [96].

Graphology can be used to identify some diseases and chemicals in the body of an individual [69]. It can also be used for educational purposes, to study historical documents, and to analyze handwritten forensic documents [96]. The graphology concepts can be used to identify the writer using handwritten passages [61] [9] [1] or handwritten signatures. However, they are generally used to attest the authenticity of a document [96] [21] [56] [136] [22].

2.3 SIGNATURE

The term signature is derived from the Latin word *signare* which means to sign or put a mark [93, p. 309]. The signature is considered a special type of handwriting. Normally, the signature is the handwritten representation of a person's name. It can be presented in several forms, from full names to abbreviations. It can also be presented from a legible to a flourished form [3, p. 87] [92]. Moreover, the signature shows the approval and agreement with the content of a document. For some writers, the signature is the most used form of writing along their lives. Due to the importance related to the signatures, they have been the main form of writing to be falsified [3] [93].

The signatures can be classified into genuine signatures, simulated signatures, and forgeries. The authentic or genuine signatures are naturally written by the individual. When the individual tries to imitate their signatures to make it looks like a forgery, it is named as a simulated signature. Despite the individual wrote their signatures, it is a result of an unnatural behavior. The falsification is made when a forger tries to mimic the signature of an individual to gain something in exchange. It is necessary to determine what is the natural variability of the individual who provided the signature to classify it in one of these kinds [103].

2.3.1 Intrapersonal and Interpersonal Variability

Since the handwritten signature is resulting of an individual behavior when signing, the signatures that were written by the same individual never have exactly the same visual appearance [78, p. 7]. This is known as intrapersonal variability or writer variability (Figure 2.2). Moreover, two individuals do not have the same signature. This is called interpersonal variability [63, p. 4-6]. It is very hard to determine the writer variability using only one or few signature samples. Due to this natural variability, several samples are necessary to analyze offline signatures. Several factors, such as handwriting skill, age [50], posture, dominant hand [127], environment, health [69], and the physical [124] and psychological state [69] can influence the writer behavior.

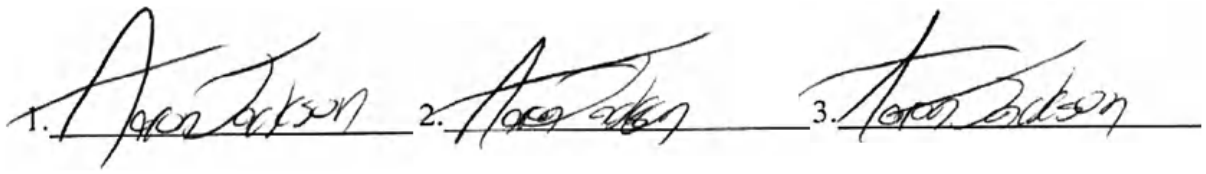


Figura 2.2: Offline Signatures of the Same Writer (Source: Adapted from Harralson, 2013 [63])

Cultural factors can also influence the writer variability [67]. One of these factors is the handwriting system used to write the signatures. The handwriting systems based on ideograms present a smaller variability than the systems based on the Latin alphabet. For example, the signatures of the Japanese citizens tend to have a small variability among them. Since their childhood Japanese people learn to sign using printed form characters. Consequently, it decreases the variability between signatures of different writers. Due to that, the Japanese government decreed in 1883 the use of red stamps as an official form of authentication [88].

Due to the small signature variability and security, the Indonesian government allows the periodically changing of individual signature register. However, the small variability is not restrict to only handwriting systems based on ideograms. Besides that, some writers can have more than one signature, since each signature belongs to a different alphabet system. This factor increases the difficult in analyzing the signatures and determining the individual variability [88].

2.3.2 Forgery

A forgery is made when a forger tries to mimic the signature of an individual to gain something in exchange [63, p. 6-8]. Forgeries can be divided into random, simple, and skilled. When the forger does not know the victim's name and they use their own signature, it is considered a random forgery. When the forger only knows the victim's name, it is considered a simple forgery. A skilled forgery is created when the forger has access to the victim's signature and trains to reproduce it. To reproduce a signature, the forgers try to replicate its speed, pressure, trajectory, and visual aspect. Basically, they try to mimic the victim's behavior. Depending the forger's skill, the skilled forgery can be hard to identify [140, p. 204] [92].

Due to the falsification possibility, the handwriting signature verification was created. Some factors such as line discontinuity, irregular line quality, blunt ending, lack of smooth pressure change patterns, pencil traces, and carbon tracings can contribute to detect forgeries. If the signature sample that will be analyzed is exactly alike the writer signature, it can be a forgery or a copy of the original one. On the other hand, if the signature sample that will be analyzed is quite different from a set of the writer signatures, it can be considered as a forgery as well [63, p. 7-8]. Afterwards the automatic signature verification systems were developed to automatize the verification process considering these and another factors [92]. These systems try to determine where the intrapersonal variability ends and the interpersonal variability begins [72]. Despite of the advances of the last decades, it remains a challenging issue [104] [68] [103] [72] [57] [22].

3 OFFLINE HANDWRITTEN SIGNATURE VERIFICATION

Since the signature is a behavioral biometric trait, it cannot be lost, stolen, or forgotten [72]. Furthermore, the signature is widely used and accepted in daily tasks such as closing of deals, wills, bank transactions, credit card receipt authentication, and others [21] [58]. Despite that, the signatures can be falsified [78, p. 55]. Due to increased concerns about the security of systems based on signatures [59] [36], several types of research about handwritten signature verification have been conducted [22].

The signature verification systems (SVS) can be divide into online (dynamic) and offline (static) signature verification systems [63]. In the online SVSs, generally, the writer signs using a digitizing tablet, a touch screen, or a special pen. These devices provide the signature over time. They also get data on the traced path, pressure, and speed. Using this set of data, the system determines if an input signature belongs to a writer or not [88]. In the offline SVSs, the writer signs a piece of paper that is digitized using a camera or scanner. Subsequently, the system determines if the signature belongs to the writer using its image features [140]. Unlike online SVSs, the offline SVSs only have the signature image, increasing the difficulty of this task [88]. In this work, I will only focus on offline signature verification systems.

An offline signature verification system is composed of several steps: signature acquisition (Section 3.1), preprocessing (Section 3.2), feature extraction (Section 3.3), classifier training (Section 3.4), and signature verification (Section 3.5) [68]. Due to its advances and challenges, each step has been attracting the interest of the scientific community [57]. The following subsections detail and show how each one of these steps is treated in the literature.

3.1 SIGNATURE ACQUISITION

Generally, the writer signs a piece of paper using a pen, pencil, or another tool to write. Then, the signature sample is digitized using a sensor such as a scanner or a camera [57]. Some perturbations can affect the quality of the signature sample. These perturbations can be caused by the sensor limitations, the tool used to write, quality of the paper, and posture of the writer [22].

Several researches used private offline signature databases [25] [65] [48] [97] [8] due to the lack of public ones. The use of private databases makes it difficult to compare the performance of different SVSs. Several factors such as complexity and size of a database can affect the performance of a SVS [57]. In the last decades, these constraints motivated the scientific community to create public offline signature databases [47] [44] [75] [127] [99]. However, there is a small number offline signature databases available for use. It may be due to the fact that signature acquisition process is time consuming and expensive, and it requires several legal, bureaucratic, and security measures [122]. Furthermore, due to the availability of some handwritten signature databases, the signature acquisition process has been disregarded by

several works [82] [138] [128] [4] [40] [84] [83] [53] [145] [137] [17] [23] [54] [55] [21] [56] [11] [136] [58] [146].

The most of public offline signature databases use a similar acquisition protocol. Each writer receives a form with several cells that are filled with the corresponding signatures. Generally, these cells have the same size as the real applications like bank checks or credit card receipts. Regarding the forgeries acquisition, each individual receive genuine signature samples and they are asked to imitate the signatures once or more times. Subsequently, the forms are collected, scanned with a resolution of 300 dpi (dots per inch) or 600 dpi, and preprocessed [57].

Table 3.1 summarizes the offline handwritten signature databases presenting the database name, number of writers, number of genuine signatures per writer, and the number of forgeries per writer. The following subsections describe the western databases: Brazilian PUC-PR, CEDAR, GPDS-960, and MCYT-75.

Tabela 3.1: Offline Handwritten Signature Databases Where #W, #G, #F, dpi, IF, C, CS Stand for the Number of Writers, Number of Genuine Signatures per Writer, Number of Forgeries per Writer, Resolution in dpi, Image Format, Color, and Cell Size in $cm \times cm$, Respectively

Database	**#W**	**#G**	**#F**	**dpi**	**IF**	**C**	**CS**
Brazilian PUCPR	60	40	10 simple, 10 skilled	300	BMP	Gray	3.00 x 10.00
	108	40	0				
CEDAR	55	24	24 skilled	300	PNG	Gray	5.08 x 5.08
MCYT-75	75	15	15 skilled	600	-	Gray	1.75 x 3.75
GPDS-960	881	24	30 skilled	300	BMP	Gray	5.00 x 3.50
							5.50 x 2.50

3.1.1 Brazilian PUC-PR

Brazilian PUCPR consists of 168 Brazilian writers. For each writer, there are 40 genuine signature samples. For the first 60 writers, they also acquired 10 simple forgery samples and 10 skilled forgery samples per writer. The signatures were acquired in cells of 3.00 cm x 10.00 cm, in 8-bit gray scale, at 300 dpi and stored in Bitmap (BMP) format [145].

3.1.2 CEDAR

CEDAR dataset consists of 55 writers. For each writer, there are 24 genuine signature samples and 24 skilled forgery samples. The signature images were acquired in cells of 5.08 cm x 5.08 cm, in 8-bit gray scale, at 300 dpi and stored in Portable Network Graphics (PNG) format [75].

3.1.3 MCYT-75

MCYT-75 consists of 75 writers of four different Spanish universities [20]. For each writer, there are 15 genuine signature samples and 15 skilled forgery samples. The signatures were acquired in cells of 1.75 cm x 3.75 cm, in 8-bit gray scale, at 600 dpi.

3.1.4 GPDS-960

GPDS-960 consists of 881 writers. For each writer, there are 24 genuine signature samples and 30 skilled forgery samples. Forgeries were produced based on the image of the genuine signatures. Each forger was able to practice for as long as they want. The signatures were acquired in cells of 5.00 cm x 3.50 cm and 5.50 cm x 2.50 cm, in 8-bit gray scale, at 300 dpi and stored in BMP format [127].

3.2 PREPROCESSING

Signature images can present several types of variability such as background, inclination, angle, size, stroke width, noise, etc [57]. Therefore, it is necessary to use some kind of preprocessing to prepare the images to next steps [52, p. 41]. In this step, generally, segmentation, noise removal, normalization, and representation algorithms are used [57] [68].

3.2.1 Signature Segmentation

Segmentation is responsible by the search and extraction of the signature from a document [57]. Like the most of pattern recognition problems, the signature segmentation is a complex and challenging process [52, p. 42-43]. Mainly, when it is necessary to extract signatures of bank checks. Bank checks have colorful backgrounds with several logos and preprinted lines increasing the complexity of the segmentation process [68].

Otsu's algorithm is one of the most used algorithms for signature segmentation [40] [84] [102] [83] [56]. Otsu's algorithm try to find best threshold that segment the signature of the background using statistical properties of the image. However, this algorithm does not present a good performance when the background or the illumination are not uniform [52, p. 744,747-751].

Posterization algorithm is another algorithm used for signature segmentation. It reduces the number of colors in the image fo a fixed number. After that, a thresholding algorithm is used to create a binary (black and white) image. Finally, the binary image is used to split the signature from background [128]. Several works consider the signatures already were segmented before the verification process, disregarding this step. Due to that, signature datasets containing the previously extracted signatures are normally used [57].

3.2.2 Noise Reduction

During the acquisition of offline signature images there are some noise. The noise can be produced by bad sensor working, irregular surface where the sensor is located, paper or ink reflectance, absorption capability of the paper, roughness of the paper, etc [72]. Therefore, noise reduction filters such as median [83] [70] [102] and mean [21] filters can be used. Despite these filters remove the noise, they also can hamper the contour of the signatures. Therefore, it is

important to determine the filter intensity to remove the noise and keep the contour information of the signatures [52, p. 18, 174-177].

Besides that, the morphological operations are also used to correct unconnected components in the signatures [40] [84] [21]. Generally, the operation of dilation is used to group poor defined strokes and the operation of erosion is used to remove small unconnected components [21].

3.2.3 Normalization

During the signature acquisition or the signature writing, the signature can be inclined and further affect the feature extraction process. Therefore, the inclination correction of the signatures is widely adopted by several signature verification system [138] [84] [83] [136]. The size also can affect the signature verification systems. Therefore, the signatures can be resized to a fixed size [102] [83] [136], or they can be cropped and put in the center of a window of fixed size [21] [56]. Another factor that can affect the performance of SVS is the position of the signature. Like the size normalization, the signature images are previously cropped and centralized in window of fixed size. This process tries to guarantee that the center of the signature is in the same region [138] [102] [136].

Several researches use the normalization as just an step or a prerequisite of the verification system, and they do not show the real influence of this process in the performance [138] [84] [102] [21] [136]. On the other hand, other researches show the real importance in determining the normalization effects in the offline signature verification. For example, Hafemann et al. (2016b) [55] shown that the normalization affect the performance of the SVSs. When they used the signature normalization, they improved the performance of their SVS decreasing the EER in about 4%.

3.2.4 Representation

Signature image in gray scale has being used for the extraction of texture features [128] [138] [40] [17] [23] [21] and for the extraction of Convolutional Neural Networks (CNNs) features [56] [58]. Besides the gray scale, the signature can be represented by a binary image [80] [70] [102] [136]. The signature image can have several representations, it depends on the binarization algorithm used to binarize it. Like the signature segmentation, the Otsu's algorithm is the most used to binarize signature images [70] [102].

The signature also can be represent by a skeleton [80] [84] [145] [21]. A skeleton is the structure that represents the approximation of the strokes' medial axis [52]. The skeleton is use represent the signatures when the features are sensible to the strokes width. However, the use of this kind representation can lead to the lost of important signature information [137]. The binary and skeleton images are normally used when it is necessary to extract some kind of structural or

geometric feature [80] [128] [4] [70] [102] [53] [145] [17] [11]. Besides that, the binary images are also used to extract CNNs features [136].

Another form to represent the signature is using its contour. Kumar and Puhan (2014) [83] used the upper and lower envelope to represent the signatures. Basically, the envelope represents the contour that surround all the writer signature. Like the skeleton, this representation is associated to the extraction of some structural or geometric feature.

3.3 FEATURE EXTRACTION

The feature extraction is one of the fundamental steps in a signature verification system [83]. The features can be classified into global and local features. While the global features describe the whole signature image [8] [80] [84] [83], the local features describe some parts of the signature image. The local features can be extracted using image segments [90] [137] [11] or using a grid overlapping the image [8] [138] [70] [40] [84] [57].

There is a constant search for the best features that better generalize and describe the handwritten signatures. Several kinds of features have been used with this purpose, from structural to the features extracted using CNNs [136]. Due to this diversity, the features can be divided into handcrafted features and representation learning.

3.3.1 Handcrafted Features

Geometric features are used to the describe the signature shapes [96] [8] [80]. Among them, there are height, width, caliber (ratio between height and width), and the signature area. Despite these features are generally used as global descriptors, they can also be used as local descriptors dividing the signature in regions, and extracting the features in each region [57].

Graphometric features have been the object of significant handwritten signature analysis specialists interest. Bertolini et al. (2010) [8] selected the stroke width, the stroke distribution, inclination, and the signature curvature to describe signatures. Oliveira et al. (2005) [96] select other graphometric features such as caliber, proportion, inclination, and the signature spacing during the signature verification. Like the geometric features, the graphometric features also need that the signatures are normalized following some criteria.

Features based on texture are also used for offline signature verification [128] [138] [40] [84] [137] [117] [17] [23] [21]. Among the texture features, the Local Binary Patterns (LBP) descriptor [128] [40] [137] [17] [23] and its variants [40] [115] [116] [117] [23] [21] are widely used. The LBP histogram is used to describe the local patterns of a image. Another texture descriptor is the Gray Level Co-occurrence Matrix (GLCM) [128]. This descriptor uses the relative frequency of the neighborhood pixels [57]. One of the main problems in using texture descriptors is the complexity, or heterogeneity of the background image. When the image has a complex or heterogeneous background, it can change the gray levels of the image. Therefore, the

image background is removed using some segmentation algorithm before the feature extraction [40].

Ferrer et al. (2012) [40] compared the performance of three texture descriptors for signature verification: LBP, Local Directional Patterns (LDP), and Local Derivative Patterns (LDerivP). Among these texture descriptors, the LDerivP achieved the best performance during the verification. However, this descriptor took 15 times more time to extract the features than the another descriptors.

Yilmaz et al. (2011) [138] used the combination of the Histogram of Oriented Gradients (HOG) [15] and LBP for signature verification. They considered Cartesian and polar coordinates systems to dived the signature images into region and to extract the HOG features. When the features were individually compared, the HOG with polar coordinates presented the best results. However, when the combinations were included, the combination between the LBP, and HOG with polar and Cartesian coordinates presented the best results. This shows that the different descriptors extracted complementary features from signatures.

The combination of geometric and texture features was also explored. Kumar et al. (2012) [84] proposed a new descriptor combining the structural and textural properties of the image. The descriptor is represented by an histogram of distribution of black pixels in the neighborhood, according to several distances between the central pixel and their neighbors. It presents a vector with 24 elements and has lower computational complexity than other descriptors. However, this descriptor is sensible to rotation. Therefore, before use it, the inclination of the signature images must be normalized.

The directional features try to describe the image using the orientation of the signature strokes [57]. Eskander et al. (2013) [33], Rivard et al. (2013) [106] e Diaz et al. (2017a) [21] extracted the Directional Probability Density Function (DPDF) of the signature gradients using grids of several sizes. This kind of features is translation and scale invariant. However, it is sensible to rotation and noise [30]. Zhang (2010) [139] investigated the Pyramid Histogram of Oriented Gradients (PHOG) to describe signatures. Like HOG, this descriptor represents the signature using an histogram of contour orientations. Moreover, it also represents the histogram in several sizes [57].

Beside the directional features, the features based on the signature stroke distribution are also used. Eskander et al. (2013) [33], Rivard et al. (2013) [106] e Diaz et al. (2017a) [21] used the descriptor Extended Shadow Code (ESC) to verify signatures. For a binary signature, the descriptor uses a grid overlaid to the image to compute the horizontal, vertical, and diagonal histogram projection of the pixels in the signature. Since the descriptor uses binary images to extract the features, it depends on binarization algorithm. Furthermore, it is sensible to scale, translation, and rotation [111].

The Run Length Distribution (RLD) descriptor is also used for signature verification. For a binary image, the descriptor computes the number of connected pixels with the same intensity and determines their orientation. This descriptor is used to compute the number of

black and white pixels horizontally, vertically, and right and left diagonally. Like ESC descriptor, this descriptor depends on binarization algorithm [11].

Another descriptor based on signature stroke distribution is the partially ordered sets or posets. Zois et al. (2016) [145], Diaz et al. (2017) [21], and Zois et al. (2019) [146] investigated the application of this descriptor in signature verification systems. First, the image is binarized and thinned. The thinning process provides the signature skeleton or the medial axis approximation of the signature strokes. This descriptor needs specifically an one-width skeleton. The posets features describe the skeleton transitions between binary patterns considering a window of 5x5 pixels [145].

The researchers have been using mathematical transformations to extract signature features [148] [53] [60] [98]. Zouari et al. (2014) [148] investigated the use of Fractal transformations. Soleimani et al. (2016) [121] and Ooi et al. (2016) [98] used Discrete Radon Transform (DRT) to represent signatures. While Guerbai et al. (2015) [53] used Curvelet transformations, Hamadene et al. (2016) [60] used the Contourlet transformations to extract features from the signature contours.

Features based on corresponding points of interest are also used to verify offline signatures [90] [137]. Among them, there are Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) [57]. Ruiz-del-Solar et al. (2008) [110], and Yilmaz and Yanikoğlu (2016) [137] used the SIFT descriptor to extract the frequency of the points of interest to compare the query signatures with reference signatures.

According to studies using eye-gaze tracking technology, the Forensic Document Examiners (FDEs) do not typically look at a signature as a whole [31] [32] [101] [95]. FDEs use a bottom-up search strategy to find areas in the signature that are resulting of pen movement changes. These areas of interest can present features that help to distinguish more easily between genuine signatures and forgeries [31] [95]. Okawa (2018a) [95] proposed the incorporation of these FDEs' behavior in automatic offline signature verification systems, using a feature extraction approach based on a Fisher vector (FV) with KAZE features from both the foreground and background signature images. FV can provide a more precise spatial distribution of the characteristics for each writer. Despite of that, it encodes the KAZE features in high-dimensional feature vectors. Therefore, Okawa used Principal Component Analysis (PCA) to generate more compact vectors without a significant performance loss [95].

KAZE is a Japanese word that means wind. The wind is natural phenomena that is defined by the flow of air on large scale and normally this flow can be described by a set of nonlinear processes. Based on the concept that the multiscale features of the images are ruled by nonlinear processes as well, Alcantarilla et al. (2012) proposed the KAZE features. These features provide a descriptor that is scale- and rotation-invariant in nonlinear scale spaces. Despite these features have a higher computational complexity than SIFT and SURF, Alcantarilla et al. showed that the KAZE features presented a better performance than them in detection and description tasks [2].

Besides to describe the signatures, the features also can be used to determine the writer variability [80] [102] [70] [83] [90]. Malik et al. (2014) [90] used the SURF descriptor to extract the points of interest to evaluate the variability of the signatures. They used the stability to evaluate it. When the several signature samples are compared, the stability measures the number of alterations between these samples. Just the points of interest that have a great stability were considered as matching points during the verification. The number of points of interest in the input signature and number of matching points were used to compute the probability of a signature be genuine [57].

Kovari and Charaf (2010) [80] modeled the writer variability using the statistical properties of local features. They used the position features of signatures such as: start position, end position, length, and the inclination angle. They also used geometrical features such as: perimeter, area, maximum diameter, maximum diameter angle, inscribed diameter, roundness, centroid, bounding circle, bounding box, extend, modification ratio, compactness, formfactor, moment axis angle convexity, solidity, and the aspect ratio.

Impedovo et al. (2012) [70] modeled the writer variability measuring the stability of signatures. They used binary images with regions of black pixels with five parallel segments equally spaced in four directions (vertical, horizontal, main diagonal, and secondary diagonal). Each intersection between the black pixels of the image and the segments of each direction were computed. For each writer, they computed the average cosine similarity in each region. Lower is the similarity, greater is the writer variability. Greater is the similarity, lower is the writer variability.

Subsequently, Pirlo and Impedovo (2013) [102] used the stability to verify signatures. The signature images were divided into regions. Each region of the test signature were compared with the region of the train signature. Like the stability, the comparison was performed computing the cosine similarity for each region. If the similarity was lower than or equal to the region stability, the region of the test signature was considered similar to the region of the train signature. The decision of all the signature regions were combined using the majority vote. If the writer has signature regions with a small variability, and it is presented a forgery with slightest different regions, the system is capable to identify the forgeries. However, for writers with a great variability in several regions, this approach cannot be enough to solve the problem. Moreover, this approach is sensitive to the signature rotation, inclination, and position. Therefore, it is necessary to apply a normalization before the comparison of the signature regions.

Kumar and Puhan (2014) [83] modeled the writer variability using geometrical features. For that purpose, they extracted the chord moments Each chord is a segment that connect two points of envelope of a signature. When it was compared with other structural features and the black-pixel distribution, it achieved the lowest FRRs and FARs, and the great accuracy. Besides this approach is stroke width and translation invariant, it is sensitive to rotation and signature scale. Therefore, a scale and inclination normalization needs to be applied before the feature extraction.

3.3.2 Representation Learning

The Convolutional Neural Networks (CNNs) have stood out due to their outstanding performance in several image classification tasks. Inspired by the behavior of the brain, this kind of network is composed of layers of artificial neurons. These layers learn to recognize patterns such as contours, textures, and the relation between image pixels [37]. Due to the capability of learning patterns, the CNNs can be used to automatically extract features of signature images [58]. However, the CNN training process is computationally costly and needs a great number of signature samples [136]. Besides that, several CNNs need that the input image has a fixed size. Consequently, the images need to be resized before forward them to the CNN. Therefore, some important features can be lost during this resizing process [58].

Hafemann et al. (2016b) [55] proposed a method to extract features of offline handwritten signatures using CNNs. Furthermore, they evaluated the impact of the normalization in the feature extraction. The authors shown that CNN features described better the signature images that were normalized. Hafemann et al. (2016a) [54] evaluated four different CNN architectures, and the respective feature extraction layers. Among the evaluated architectures, the simplified AlexNet shown promising results.

Hafemann et al. (2017a) [56] proposed a new CNN architecture for offline signature feature extraction. The architecture was trained considering two different scenarios. One, when the CNN was trained using only samples of genuine signatures, and it was named as SigNet. And the another one, when the CNN was trained using samples of genuine signatures and skilled forgeries, and it was named as SigNet-F. The features learned by the CNNs showed to be robust enough to generalize the features of different signatures based on the Latin alphabet. However, the authors did not performed experiments to show the generalization capability of the representation learning for different writing systems.

Later Hafemann et al. (2018) [58] proposed alterations in the SigNet architecture to enable the use of images with different sizes as input. Moreover, they evaluated the new architecture using signatures with resolutions of 100 dpi, 300 dpi, and 600 dpi. Using forgeries during the training of the CNN, for larger resolutions like 600 dpi the CNN achieved better performance than lower resolutions. However, when just genuine signatures were used for training, lower resolutions like 100 dpi showed better performance than larger resolutions. Furthermore, the authors showed the generalization capability of the representation learning using two more writing systems: Bengali and Devanagari.

Yilmaz et al. (2018) [136] proposed a new architecture of CNN with two input channels for feature extraction. According to Yilmaz et al. (2018) [136], this kind of architecture is less costly than other CNNs architectures during the training process. However, it is more costly than other architectures during the testing phase. The authors compared the performance of a signature verification system using the features extracted using their architecture with the SigNet-F features. The SigNet-F features showed a better performance than their CNN features. When the both kinds

of features were combined, they achieved better results than the features individually. This may indicate that both architectures provide complementary features to describe signatures. Besides that, the authors did not evaluate their architecture in different databases based on the Latin alphabet or another writing systems. Therefore, the generalization capability of their architecture was not proven.

The spatial information can help to identify small differences between the same strokes of the signatures and some common habits of different writers. Consequently, it can be used to distinguish genuine signatures from skilled forgeries. However, during the feature extraction the traditional CNN loses the spatial information about the signature strokes [144]. Exploring this limitation, Zheng et al. (2021) [144] proposed a method to keep some spatial information during the feature extraction. The authors combined two CNNs to perform the verification. While one of the CNNs discriminates different writers, the another one was used to distinguish the genuine signatures from skilled forgeries. Despite keeping the spatial information of the signatures, training two CNNs is computationally costly.

The Generative Adversarial Networks (GANs) are composed by two networks. One of them is the generator *G(z)* and another one is the discriminator *D(y)*. During the training, both networks are trained simultaneously. For an input image, the network *D(y)* returns a value indicating if the signature is a genuine signature or a forgery. For a forgery, *D(y)* returns a value close to zero. Otherwise, *D(y)* returns a positive value. For an input vector z, the network *G(z)* tries to generate a signature image that is used to train the network *D(y)*. Therefore, *D(y)* tries to maximize the returned value, while *G(z)* tries to generate signatures that are similar to the genuine ones. Consequently, *G(z)* tries to deceive *D(y)* and minimize the returned value [142].

Zhang et al. (2016) [142] proposed the use of GANs for feature learning of signatures. After the training, the authors used the features of all convolutional layers present in discriminator network. As CNNs, the GANs require a great number of signature samples for training to achieve an acceptable performance. However, the GANs do not need the labeled samples for training.

Another strategy is the use of metric learning. The methods based on metric learning try to automatically find a good distance metric. A good distance metric considers that two similar samples have a distance close to zero, while two different samples have a great distance between them [121]. Rantzsch et al. (2016) [105] proposed a signature verification method based on metric learning. Three kinds of signatures were used during the training phase: the reference signatures, genuine signatures, and forgeries. This method tries to find the metric that minimize the distance between the reference and the genuine signature, and maximize the distance between the reference and the forgery. Like a real scenario, the authors used the random forgeries for training. Besides learning the distance metric, the CNN VGG-16 was used to learn the signature features.

3.4 CLASSIFIER TRAINING

A classifier is trained using a learning or a training subset $\mathcal{L}$ containing genuine signatures of the writers. Each classifier is used to verify the signatures. When it is necessary to verify a signature X_Q belongs to a writer, a classifier is used to classify it in a genuine signature or a forgery. To assess the performance of a verification system, it is used a testing subset $\mathcal{T}$ with genuine signatures and forgeries. If one classifier is trained for each writer of the system, it is called a writer-dependent system (WD). On the other hand, if only one classifier is trained to verify the signatures of all writers of the system, it is called a writer-independent system (WI). For WI systems, it is common to train and test the system using different sets of writers. Therefore, a development subset $\mathcal{D}$ is used to train the WI classifier, and a exploiting subset $\mathcal{E}$ is subdivided into the previous subsets ($\mathcal{L}$ and $\mathcal{T}$) [57].

A writer-independent system learns to compare the signature that will be verified X_Q with a reference signature X_R [57]. During the test phase, the system compares the signature X_Q with the references X_R of the supposed writer to determine with it is a genuine signature or a forgery. One of the most used approaches to compare the signatures is using the dissimilarity representation [8] [138] [84] [33] [106] [60] [137] [146] [123]. Dissimilarity Z is calculated through difference between the reference signature feature vector V_R and the feature vector of the signature that will be verified V_Q [8].

Unlike WD systems, WI systems are scalable and can manage new writers without retraining or updating the model. Since the dissimilarity representation uses a pairwise combination of signatures, it increase the number of samples available for training. On the other hand, many of these samples have redundant information that do not help to improve the model during the training [123]. Furthermore, the WD systems generally present a better performance than WI systems [22] [89].

Hidden Markov Model (HMM) already have been widely used to verify handwritten signatures. It reproduces the writing sequence of the handwritten signature. Due to that, this classifier is ideal for online signature verification [56]. Despite that, HMMs are also used in offline signature verification [96] [4] [17] [21]. When they are used with offline signatures, each writing sequence is represented by a segment of the signature, and each segment represents a state. The transition between these states indicates the signature change along the time. During the verification, HMMs calculate the probability of each signature segment belong to some writer [16].

Since HMMs depend on the state sequence, they also depend on the writing system used to generate de signatures. The writing system based on Latin are the most used ones [96] [131] [4] [17] [21], occurring from left to right. Therefore, the most used state transitions is from left to right [57]. Due the nature of it, this classifier is sensible to the writing system used to sign. Consequently, this classifier is not suited to signatures that follow different writing systems.

Furthermore, like other classifier, the HMMs need a great number of signature samples to achieve an acceptable performance [10].

Support Vector Machine (SVM) is one of the most used classifiers for offline signature verification [138] [128] [4] [40] [83] [53] [145] [137] [17] [23] [21] [56] [11] [95] [147] [123] [144]. The popularity of this kind of classifier may be due to its performance during the verification. On the other hand, the SVM training may be computationally costly when there are several signature examples available [10].

When the SVM classifier is used, it is possible to use a kernel to improve the performance of it. A kernel is a function that transform the feature vector in a feature vector of a higher dimensionality. For classification problems that are not linearly separable, the kernel can help to solve this problem [10]. Among the kernels available, the most used ones are: linear [40] [83] [56], Radial Basis Function (RBF) or Gaussian [138] [128] [4] [40] [53] [145] [137] [17] [56] [11], and the χ^2 [40] [23] [21] kernels.

Depending on the set of features, one kernel can present better or worse performance than other kernels. Hafemman et al. (2017a) [56] achieved their best results using the RBF kernel and the features extracted using CNNs, while Diaz et al. (2017a) [21] achieved their best results using the χ^2 kernel and the LDerivP descriptor. The use of a kernel also implies in a increase of the computational cost according to the complexity of the chosen function [10].

A simple distance calculus between the feature vectors of signatures that will be verified and the reference signatures can also be used. Some distances such as Euclidean [117] [17], Mahalanobis [131], and Canberra [60] distance are used to verify offline signatures. Wen et al. (2009) [131], and Hamadene and Chibani (2016) [60] defined a threshold to determine if a signature belongs to a writer or it is a forgery. If the smallest distance is smaller than the threshold, the signature is considered as a genuine signature. Otherwise, it is considered as a forgery.

Serdouk et al. (2016) [117] used the K-NN classifier to calculate the distance between the feature vectors of the signature X_Q and the reference signatures. It also was used to calculate the distance between the vectors of X_Q and the forgery references. If X_Q is closer to a genuine signature, it is considered as a genuine signature. Otherwise, it was considered as a forgery.

Liu et al. (2021) [89] proposed the Mutual Signature DenseNet (MSDN) to extract the signature features and to learn a similarity metric from regions of the signature images. Liu et al. showed that the MSDN can be used as a WI and a WD classifier. Despite this architecture can be used as a WI classifier, it showed a better performance as a WD classifier to verify offline signatures.

Unlike Rantzsch et al. (2016) [105] and Liu et al. (2021) [89], Soleimani et al. (2016) [121] do not use a CNN to learn the signature features with the dissimilarity metric. Soleimani et al. (2016) [121] extract the features using HOG and DRT descriptors to train, and test their method. Besides the distance metric, their method depends on the representation capability of

the chosen features. If the chosen features do not describe the signatures well, the verification system will present unsatisfying results.

Another strategy is the use of the one-class models [53] [11]. The one-class models use the genuine signatures of the writer as examples. Guerbai et al. (2015) [53] and Bouamra et al. (2018) [11] used the one-class support vector machine (OC-SVM) classifier to try to solve the signature verification problem. According to Bouamra et al. (2018) [11], the main advantage of using one-class classifiers is that these classifiers can differentiate the genuine signatures of forgeries without using forgery examples. Besides that, this kind of classifier has shown good performance on unbalanced problems [6].

Despite the strong points of the one-class classifier, it cannot determine the writer variability when there is a unique sample or few signature samples per writer [57]. Therefore, more samples are needed for training of this type of classifier. Moreover, the ideal number of samples for training of one-class classifiers was not widely investigated [53] [11].

The CNNs can be used to extract features and to classify as classifiers. As previously exposed, the CNN training is computationally costly and require a great number of signature samples. Besides that, the great number of layers and the size of the architecture also influence in the computational cost of the CNN [58] [136]. According to Domany et al. (1996) [29] the CNNs are recommended to extract features, but they are not necessarily recommended to classify. The traditional classifiers may be more suited to verify offline signatures than the last layer of the CNNs. Hafemann et al. (2017a) [56] showed experimentally this behavior. Despite the architecture proposed by them achieved a good performance during the signature feature extraction, the same architecture did not achieve a good performance as a classifier. Therefore, it is important to identify the weak and the strong points of each architecture when a CNN is being chosen or developed.

Another approach is the combination of several classifiers. Bertolini et al. (2010) [8] trained several SVM classifiers with different graphometric features. A genetic algorithm was used to select and combine a subset of classifiers. Besides combining classifiers of the same type, it is possible to combine different types of classifier to verify signatures [4].

Yilmaz and Yanikoğlu (2016) [137] showed that is possible to combine a writer-independent with a writer-dependent system. In the WI system, they used a SVM classifier with dissimilarity vectors. In the WD system, they trained a SVM for each writer using the HOG and LBP feature vectors. Each system provided a probability of the signature be genuine. The probabilities were combined using the mean rule.

3.5 SIGNATURE VERIFICATION

To show the performance of the verification systems, the researches that used at least one of the datasets were selected: GPDS-960 (Table 3.2), MCYT (Table 3.3) e CEDAR (Table 3.4). It is important to highlight that each research uses a different experimental protocol. One

of these experimental protocol factors is the number of writers used for training of the SVS. Therefore, the researches were grouped considering the databases used without restricting the number of writers used in the experiments.

Some experimental details such as the type of the system (Writer-Dependent or Writer-Independent), feature descriptors, classifier algorithms, and the number of reference signatures per writer used for training are summarized in the following Tables 3.2, 3.3, and 3.4. Regarding the performance of the system, some metrics such as $FAR_{skilled}$ (percentage of skilled forgeries that were accept as genuine signatures), FRR (percentage of genuine signatures that were reject by the system), AER (average between $FAR_{skilled}$ and FRR), and the EER (Equal Error Rate) [57] are also summarized. Since the researches have different experimental protocols, a direct comparison among them is not possible [92].

Tabela 3.2: Performance of the signature verification methods using GPDS database

Type	**Features and Classifier**	**#R**	**$FAR_{skilled}$** (%)	**FRR** (%)	**AER** (%)	**EER** (%)
WD [118]	HOT (AIRSV)	10	5.91	12.80	8.76	9.30
WD [139]	PHOG (GLM)	19	17.77	29.03	-	-
	PHOG (FLD)	19	3.90	6.06	-	-
	PHOG (KNN)	19	2.64	8.57	-	-
	PHOG (MLP)	19	4.09	7.26	-	-
	PHOG (SVM)	19	3.25	4.50	-	-
WD [138]	HOG, LBP (SVM)	5	-	-	-	17.53
		12	-	-	-	15.03
WD [137]	HOG, LBP (SVM)	5	-	-	-	7.98
		12	-	-	-	6.97
WD [40]	LBP (SVM)	10	-	-	-	23.03
WD [43]	LBP (SVM)	2	-	-	-	3.14
		5	-	-	-	1.46
		10	-	-	-	0.76
WD [17]	LBP (SVM)	5	-	-	-	18.80
WD [116]	LBP (SVM)	16	9.61	8.29	9.14	-
WD [116]	LBP_{riu} (SVM)	16	10.19	7.83	9.34	-
WD [116]	OC-LBP (SVM)	16	9.76	8.29	9.23	-
WD [116]	OC-LBP, LBP_{riu} (SVM)	16	9.11	6.70	8.25	-
WD [128]	LBP, GLCM,	5	4.79	23.09	-	12.88
	Contour (SVM)	10	13.13	7.46	-	11.04
WD [42]	LBP, LDP (SVM)	10	-	-	-	15.90
WD [40]	LDP (SVM)	10	-	-	-	21.52
WD [117]	GLBP, LRF (AIRS)	16	13.16	11.38	12.52	-
WD [115]	OC-LBP, LRF (SVM)	16	9.11	7.70	8.61	-
	OC-LBP (SVM)	16	9.76	8.29	9.23	-
WD [21]	LDerivP (SVM)	2	-	-	-	21.63
		5	-	-	-	17.19
		8	-	-	-	14.58
WD [40]	LDerivP (SVM)	10	-	-	-	15.35

Tabela 3.2: Performance of the signature verification methods using GPDS database (continuation)

Type	**Features and Classifier**	**#R**	**$FAR_{skilled}$** (%)	**FRR** (%)	**AER** (%)	**EER** (%)
WD [115]	LRF (SVM)	16	11.80	15.41	13.13	-
WD [11]	RLD (SVM)	1	8.13	5.73	-	-
		4	9.66	3.88	-	-
		8	7.77	3.65	-	-
		12	6.64	3.63	-	-
WD [21]	ESC, DPDF (BFS)	2	-	-	-	28.55
		5	-	-	-	24.04
		8	-	-	-	20.39
WD [33]	ESC, DPDF (BFS)	12	18.17	27.25	15.24	-
		14	22.71	18.06	13.96	-
WD [21]	Posets (SVM)	2	-	-	-	25.01
		5	-	-	-	21.68
		8	-	-	-	18.66
WD [145]	Posets (SVM)	5	18.79	4.65	4.86	5.48
		10	9.31	2.91	2.94	3.53
		12	8.68	6.72	2.59	3.24
WD [4]	Segmentation	4	48.69	19.44	26.92	-
	Grid (SVM, HMM)	8	49.00	14.88	25.10	-
		12	47.25	19.19	25.42	-
WD [53]	Curvelet	4	-	-	16.92	-
	Transform (SVM)	8	-	-	15.95	-
		12	-	-	15.07	-
WD [17]	Geometrical (HMM)	5	-	-	-	22.50
WD [21]	Geometrical (HMM)	2	-	-	-	32.01
		5	-	-	-	27.86
		8	-	-	-	26.60
WD [102]	Stability (Cosine Similarity)	12	-	-	-	7.2
WD [17]	Zernike Moments (Distance)	5	-	-	-	35.16
WD [54]	AlexNet (SVM)	5	1.87	31.48	-	5.41
		12	4.73	10.42	-	4.17
	Reduced AlexNet	5	1.74	26.33	-	4.53
	(SVM)	12	5.13	6.55	-	3.47
WD [56]	SigNet (SVM)	5	-	-	-	3.92
		12	-	-	-	3.15
	SigNet-F (SVM)	5	4.68	6.03	-	2.42
		12	3.53	3.94	-	1.69
WD [136]	SigNet-F (SVM)	5	-	-	-	2.66
		12	-	-	-	2.08
WD [58]	SigNet-SPP-300dpi (SVM)	12	-	-	-	3.15
	SigNet-SPP-300dpi-F (SVM)	12	-	-	-	0.41
WD+WI	CNN with 2 channels,	5	-	-	-	1.16

Tabela 3.2: Performance of the signature verification methods using GPDS database (continuation)

Type	**Features and Classifier**	**#R**	**FAR$_{skilled}$** (%)	**FRR** (%)	**AER** (%)	**EER** (%)
[136]	SigNet-F (SVM)	12	-	-	-	0.88
WD+WI [142]	DCGANs (Adaboost)	14	-	-	16.08	-
WI [84]	Surroundness (MLP)	24	13.76	13.76	-	-
WI [33]	ESC, DPDF (BFS)	14	27.04	26.42	17.82	-
WI [60]	Contourlet	3	18.92	24.56	21.74	-
	Transformation	4	20.67	17.00	18.83	-
	(Distance)	5	21.10	15.73	18.42	-
WI [136]	CNN with 2 channels	5	-	-	-	4.72
	(SVM)	12	-	-	-	2.88
WI [146]	Asymmetric P_{2AD} (DSC-BFS)	5	-	-	-	3.06
WI [147]	K-SVD/OMP F_3 (SVM)	12	-	-	-	0.70
WI [123]	SigNet (SVM)	12	-	-	-	3.69
	Condensed Nearest Neighbors SigNet (SVM)	12	-	-	-	3.47

Through the Tables 3.2, 3.3, and 3.4, it is evident that the writer-dependent systems are the most used. This may be related to the performance of this kind of system and its implementation simplicity. Furthermore, several researches use the GPDS-960 database. It is may be associated to the robustness of the databases. Among the GPDS, MCYT, and CEDAR databases, the GPDS has the greatest number of writers and signatures.

Tabela 3.3: Performance of the signature verification methods using MCYT database

Type	**Features and Classifier**	**#R**	**FAR$_{skilled}$** (%)	**FRR** (%)	**AER** (%)	**EER** (%)
WD [17]	Zernike Moments (Distance)	5	-	-	-	35.51
WD [131]	RPF (Distance)	5	-	-	-	15.30
WD [131]	RPF (HMM)	5	-	-	-	15.02
WD [17]	LBP (SVM)	5	-	-	-	16.07
WD [42]	LBP, LDP (SVM)	10	-	-	-	23.42
WD [40]	LBP (SVM)	10	-	-	-	11.28
WD [43]	LBP (SVM)	2	-	-	-	2.28
		5	-	-	-	0.35
		10	-	-	-	0.26
WD [128]	LBP, GLCM,	5	2.71	24.13	-	11.28
	Contour (SVM)	10	6.77	8.59	-	7.23
WD [21]	LDerivP (SVM)	2	-	-	-	16.06
		5	-	-	-	11.90
		8	-	-	-	9.12

Tabela 3.3: Performance of the signature verification methods using MCYT database (continuation)

Type	**Features and Classifier**	**#R**	**$FAR_{skilled}$** (%)	**FRR** (%)	**AER** (%)	**EER** (%)
WD [40]	LDerivP (SVM)	10	-	-	-	18.16
WD [40]	LDP (SVM)	10	-	-	-	10.15
WD [23]	Texture (SVM)	2	-	-	-	16.59
		5	-	-	-	11.67
WD [21]	Geometrical	2	-	-	-	19.03
	(HMM)	5	-	-	-	15.27
		8	-	-	-	12.02
WD [17]	Geometrical (HMM)	5	-	-	-	19.98
WD [21]	ESC, DPDF (BFS)	2	-	-	-	23.67
		5	-	-	-	16.58
		8	-	-	-	15.26
WD [21]	Posets (SVM)	2	-	-	-	16.50
		5	-	-	-	14.02
		8	-	-	-	11.57
WD [145]	Posets (SVM)	5	25.19	4.48	5.62	6.02
		10	17.21	4.96	3.45	4.01
WD [98]	DRT (PNN)	5	-	-	-	13.86
		10	-	-	-	9.87
WD [118]	HOT (AIRSV)	10	2.40	12.80	7.60	10.60
WD [56]	SigNet (SVM)	5	-	-	-	3.58
		10	-	-	-	2.87
	SigNet-F (SVM)	5	-	-	-	3.70
		10	-	-	-	3.00
WD [58]	SigNet-SPP-600dpi (SVM)	10	-	-	-	3.64
	SigNet-SPP-600dpi Fine-tuned (SVM)	10	-	-	-	3.40
WD [95]	KAZE/FV (SVM)	10	-	-	-	5.47
	KAZE/BoVW (SVM)	10	-	-	-	8.40
	KAZE/VLAD (SVM)	10	-	-	-	6.90
WD+WI [135]	CapsNet	5	5.33	7.32	6.32	8.95
		10	2.66	1.33	1.99	2.58
WI [146]	P_{2AD} Asymmetric (DSC-BFS)	5	-	-	-	3.50
WI [147]	K-SVD/OMP F_3 (SVM)	10	-	-	-	1.37
WI [123]	Condensed Nearest Neighbors SigNet (SVM)	10	-	-	-	2.89

The features based on texture were the most used ones to describe offline signatures. In addition, it is possible to observe the crescent interest of the scientific community in using CNNs to learn features. Among the classifiers used in SVS, the most used one was the Support Vector Machine. Despite this classifier is computationally costly, it also presents a better performance

Tabela 3.4: Performance of the signature verification methods using CEDAR database

Type	**Features and Classifier**	**#R**	**$FAR_{skilled}$** (%)	**FRR** (%)	**AER** (%)	**EER** (%)
WD [83]	Envelope (SVM)	4	21.72	10.54	-	-
		8	13.98	9.32	-	-
		12	7.42	13.18	-	-
		18	5.68	6.36	-	-
WD [53]	Curvelet	4	-	-	8.70	-
	Transformation	8	-	-	7.83	-
	(SVM)	12	-	-	5.60	-
WD [116]	LBP (SVM)	16	1.36	2.04	1.71	-
WD [116]	LBP_{riu} (SVM)	16	3.18	0.68	1.94	-
WD [116]	OC-LBP (SVM)	16	2.27	1.81	2.05	-
WD [116]	OC-LBP, LBP_{riu} (SVM)	16	2.04	1.13	1.60	-
WD [117]	GLBP, LRF (AIRS)	16	2.12	4.93	3.54	-
WD [115]	OC-LBP, LRF (SVM)	16	2.04	0.45	1.25	-
WD [115]	OC-LBP (SVM)	16	3.18	1.36	2.28	-
WD [115]	LRF (SVM)	16	9.77	9.09	9.44	-
WD [56]	SigNet (SVM)	4	-	-	-	5.87
		8	-	-	-	5.03
		12	-	-	-	4.76
WD [56]	SigNet-F (SVM)	4	-	-	-	5.92
		8	-	-	-	4.77
		12	-	-	-	4.63
WD [58]	SigNet-SPP-600dpi (SVM)	10	-	-	-	3.60
WD [58]	SigNet-SPP-600dpi Fine-tuned (SVM)	10	-	-	-	2.33
WD [144]	Micro Deformation	5	-	-	-	3.89
	CNN (SVM)	10	-	-	-	2.95
		12	-	-	-	2.76
WD [89]	MSDN	12	-	-	-	1.67
WI [82]	Morphological Features (SVM)	24	11.59	11.59	-	-
WI [84]	Surroundness (MLP)	24	8.33	8.33	-	-
WI [60]	Contourlet	3	0.0	6.24	3.12	-
	Transformation	4	0.0	5.11	2.55	-
	(Distance)	5	0.0	4.21	2.10	-
WI [146]	Asymmetric P_{2AD} (DSC-BFS)	5	-	-	-	2.90
WI [147]	K-SVD/OMP F_3 (SVM)	10	-	-	-	0.79
WI [123]	Condensed Nearest Neighbors SigNet (SVM)	12	-	-	-	3.32
WI [89]	MSDN	1	-	-	-	6.74

than other classifiers [139] [57]. Another interesting factor is the number of references per writer to train the SVS. Several researches use a lot of signature references to achieve a better performance. However, in a real-world scenario there are few references per writer available to train such systems.

3.6 FEW SIGNATURE EXAMPLES FOR TRAINING

One of the main problems of offline signature verification is the amount of signature samples available for training of the signature verification system [11]. Most of the offline signature verification systems need a great amount of signature examples to achieve an acceptable performance. However, in a real scenario, only a few signature samples per individual are provided to train the system [122]. Some applications use only a single sample per individual, it increases the difficult in determine the writer variability [48].

There are some well established commercial signature databases, but the sharing and using of this data is very restrict due to legal, privacy and bureaucratic concerns. One solution is the creation of own robust database. However, the writers can be concerned about the sharing of their personal information. Moreover, this process is costly and takes time. Besides that, it is a repetitive and exhaustive process to the writers, compromising the quality of the provided signatures [122].

Table 3.5 summarizes the main works that used few signature samples per individual during the training of offline signature verification systems. For each work, Table 3.5 presents the bibliographic reference, kind of features, classifier, database, number of writers, number of signature references per writer for training, and the performance achieved. Due to the widely diversity of adopted protocols to evaluate the SVSs, a direct comparison is not possible. Despite this widely diversity of SVSs, few of them consider up to three examples of signatures per writer to train their systems [43] [60] [19] [21] [11] [89]. Therefore, the works that also use up to five examples of signature per writer during the training were considered [128] [138] [4] [83] [53] [98] [145] [137] [17] [56] [136] [146] [144].

Tabela 3.5: Offline Signature Verification with Few Examples per Writer for Training

Reference	**Features and Classifier**	**Database**	**#W**	**#R**	**Performance (%)**			
					FAR	**FRR**	**AER**	**EER**
Wen et al., 2009 [131]	RPF (Mahalanobis Distance)	MCYT	100	5	-	-	-	15.30
	RPF (HMM)			5	-	-	-	15.02
Vargas et al., 2011 [128]	Contour, GLCM, LBP (SVM)	GPDS	100	5	4.79	23.09	-	12.88
				10	13.13	7.46	-	11.04
		MCYT	75	5	2.71	24.13	-	11.28
				10	6.77	8.59	-	7.23
Yilmaz et al., 2011 [138]	HOG, LBP (SVM)	GPDS	160	5	-	-	-	17.53
				12	-	-	-	15.03
Batista et al.,	Segmentation Grid	Brazilian	60	4	35.17	26.17	20.38	-

Tabela 3.5: Offline Signature Verification with Few Examples per Writer for Training (continuation)

Reference	Features and Classifier	Database	#W	#R	Performance (%)			
					FAR	FRR	AER	EER
2012 [4]	(SVM, HMM)			8	38.83	16.00	18.71	-
				12	34.33	13.50	15.08	-
				20	30.00	4.67	9.71	-
		GPDS	160	4	48.69	19.44	26.92	-
				8	49.00	14.88	25.10	-
				12	47.25	19.19	25.42	-
Ferrer et al., 2013b [43]	LBP (SVM)	GPDS	75	2	-	-	-	2.20
				5	-	-	-	1.00
				10	-	-	-	0.47
		GPDS	350	2	-	-	-	3.14
				5	-	-	-	1.46
				10	-	-	-	0.76
		MCYT	75	2	-	-	-	2.28
				5	-	-	-	0.35
				10	-	-	-	0.26
Kumar & Puhan, 2014 [83]	Envelope (SVM)	CEDAR	55	4	21.72	10.54	-	-
				8	13.98	9.32	-	-
				12	7.42	13.18	-	-
				18	5.68	6.36	-	-
Guerbai et al., 2015 [53]	Curvelet Transformation (OC-SVM)	CEDAR	30	4	-	-	8.70	-
				8	-	-	7.83	-
				12	-	-	5.60	-
		GPDS	160	4	-	-	16.92	-
				8	-	-	15.95	-
				12	-	-	15.07	-
Ooi et al., 2016 [98]	DRT (PNN)	MCYT		5	-	-	-	13.86
				10	-	-	-	9.87
Zois et al., 2016 [145]	Posets (SVM)	CEDAR	55	5	15.91	4.44	3.64	4.12
				10	11.52	5.83	2.74	3.02
		GPDS	300	5	18.79	4.65	4.86	5.48
				10	9.31	2.91	2.94	3.53
				12	8.68	6.72	2.59	3.24
		MCYT	75	5	25.19	4.48	5.62	6.02
				10	17.21	4.96	3.45	4.01
Yilmaz & Yanikoğlu, 2016 [137]	HOG, LBP (SVM)	GPDS	140	5	-	-	-	7.98
				12	-	-	-	6.97
Hamadene & Chibani, 2016 [60]	Contourlet Transformation (Canberra Distance)	CEDAR	45	3	0	6.24	3.12	-
				4	0	5.11	2.55	-
				5	0	4.21	2.10	-
		GPDS	290	3	18.92	24.56	21.74	-
				4	20.67	17.00	18.83	-
				5	21.10	15.73	18.42	-
		GPDS	462	3	16.60	31.06	23.83	-

Tabela 3.5: Offline Signature Verification with Few Examples per Writer for Training (continuation)

Reference	Features and Classifier	Database	#W	#R	Performance (%)			
					FAR	FRR	AER	EER
				4	18.41	18.86	18.63	-
				5	18.39	18.07	18.23	-
Das et al., 2016 [17]	Zernike Moments (Euclidean Distance)	GPDS	100	5	-	-	-	35.16
	Geometrical (HMM)			5	-	-	-	22.50
	LBP (SVM)			5	-	-	-	18.80
	Zernike Moments (Euclidean Distance)	MCYT	100	5	-	-	-	35.51
	Geometrical (HMM)			5	-	-	-	19.98
	LBP (SVM)			5	-	-	-	16.07
Diaz et al., 2016b [23]	Texture	MCYT	75	2	-	-	-	16.59
				5	-	-	-	11.67
		Bengali	100	2	-	-	-	10.67
				5	-	-	-	6.06
		Devanagari	100	2	-	-	-	11.88
				5	-	-	-	9.01
Hafemann et al., 2016a [54]	AlexNet (SVM)	GPDS	160	5	0.89	32.12	-	4.25
				14	2.77	10.41	-	3.37
	Reduced AlexNet (SVM)			5	1.03	26.56	-	3.83
				14	2.77	6.75	-	2.74
	AlexNet (SVM)	GPDS	300	5	1.87	31.48	-	5.41
				14	4.73	10.42	-	4.17
	Reduced AlexNet (SVM)			5	1.74	26.33	-	4.53
				14	5.13	6.55	-	3.47
Diaz et al., 2017a [21]	Geometrical (HMM)	GPDS	300	2	-	-	-	32.01
				5	-	-	-	27.86
				8	-	-	-	26.60
	ESC, DPDF (BFS)			2	-	-	-	28.55
				5	-	-	-	24.04
				8	-	-	-	20.39
	LDerivP (SVM)			2	-	-	-	21.63
				5	-	-	-	17.19
				8	-	-	-	14.58
	Posets (SVM)			2	-	-	-	25.01
				5	-	-	-	21.68
				8	-	-	-	18.66
	Geometrical (HMM)	MCYT	75	2	-	-	-	19.03
				5	-	-	-	15.27
				8	-	-	-	12.02
	ESC, DPDF (BFS)			2	-	-	-	23.67
				5	-	-	-	16.58
				8	-	-	-	15.26
	LDerivP (SVM)			2	-	-	-	16.06
				5	-	-	-	11.90
				8	-	-	-	9.12

Tabela 3.5: Offline Signature Verification with Few Examples per Writer for Training (continuation)

Reference	Features and Classifier	Database	#W	#R	Performance (%)			
					FAR	FRR	AER	EER
	Posets (SVM)			2	-	-	-	16.50
				5	-	-	-	14.02
				8	-	-	-	11.57
Hafemann et al., 2017a [56]	SigNet (SVM)	Brazilian	60	5	7.17	4.63	5.90	2.92
				10	10.70	1.22	5.96	2.07
				15	12.62	0.23	6.42	2.01
	SigNet-F (SVM)			5	2.72	17.17	9.94	5.11
				10	6.55	9.25	7.90	4.03
				15	8.80	5.47	7.13	3.44
	SigNet (SVM)	CEDAR	55	4	-	-	-	5.87
				8	-	-	-	5.03
				12	-	-	-	4.76
	SigNet-F (SVM)			4	-	-	-	5.92
				8	-	-	-	4.77
				12	-	-	-	4.63
	SigNet (SVM)	GPDS	160	5	-	-	-	3.23
				12	-	-	-	2.63
	SigNet-F (SVM)			5	5.17	5.16	-	2.41
				12	3.66	3.59	-	1.72
	SigNet (SVM)	GPDS	300	5	-	-	-	3.92
				12	-	-	-	3.15
	SigNet-F (SVM)			5	4.68	6.03	-	2.42
				12	3.53	3.94	-	1.69
	SigNet (SVM)	MCYT	75	5	-	-	-	3.58
				10	-	-	-	2.87
	SigNet-F (SVM)			5	-	-	-	3.70
				10	-	-	-	3.00
Bouamra et al., 2018 [11]	RLD (OC-SVM)	GPDS	281	1	8.13	5.73	-	-
				4	9.66	3.88	-	-
				8	7.77	3.65	-	-
				12	6.64	3.63	-	-
Yilmaz et al., 2018 [136]	SigNet-F (SVM)	GPDS	160	5	-	-	-	2.66
				12	-	-	-	2.08
	CNN with 2 channels (SVM)			5	-	-	-	4.72
				12	-	-	-	2.88
	CNN with 2 channels, SigNet-F (SVM)			5	-	-	-	1.16
				12	-	-	-	0.88
Zois et al., 2019a [146]	Asymmetric P_{2AD} (DSC-BFS)	CEDAR	35	5	-	-	-	2.90
		GPDS	195	5	-	-	-	3.06
		MCYT	48	5	-	-	-	3.50
Yapıcı et al., 2020 [135]	CapsNet	MCYT	-	5	5.33	7.32	6.32	8.95
				10	2.66	1.33	1.99	2.58
Zheng et al., 2021 [144]	Micro Deformation CNN (SVM)	CEDAR	55	5	-	-	-	3.89
				10	-	-	-	2.95
				12	-	-	-	2.76

Tabela 3.5: Offline Signature Verification with Few Examples per Writer for Training (continuation)

Reference	Features and Classifier	Database	#W	#R	Performance (%)			
					FAR	FRR	AER	EER
Liu et al., 2021 [89]	MSDN	CEDAR	55	1	-	-	-	6.74

Among the used databases, GPDS is used in almost all of the works with few signature examples per writers. It may be due the GPDS is one of the most robust signature datasets present in the literature [47] [44] [75] [127] [99]. Features based on texture are the most used to describe signatures in works with few examples per writer [128] [138] [43] [137] [17] [23] [21]. Since the texture descriptors are easy to implement and showed promising results in describing offline handwritten signatures, it may explain this behavior [57] [22]. Despite the features learned by deep learning models are not widely used to describe few signature samples, they showed promising results [54] [56] [136] [135].

Like the other works, SVM is the most used classifier [128] [138] [4] [43] [83] [145] [17] [23] [54] [21] [56] [136], its variations [53] [11], and combinations [137]. Despite its computational cost, this kind of classifier has been showing promising results with few signature samples per writer [137]. Regarding the approaches used to solve this kind of problem, the use of one-class classifiers [53] [11], the search for the best features to describe signatures [54] [56] [136] [135], and the offline signature augmentation techniques [43] [23] [21] can be highlighted.

3.7 OFFLINE SIGNATURE AUGMENTATION

Offline signature augmentation techniques are used to create synthetic signature samples to train a signature verification system [65] [35] [41] [42] [19] [23] [39] [21] [38] [109] [135]. It can be done modifying real signatures [65] [35] [41] [42] [19] [23] [39] [21] [38] [135] or creating pure synthetic signatures [109]. When the real signatures are used to create new signature samples, it can be called signature duplication [21]. On the other hand, if the real signatures are not used to create new samples, it can be called signature composition [73]. Despite the compositional techniques do not use real data to create new signatures, they can provide new information to improve the performance of SVSs [109]. Unlike compositional techniques, the duplication techniques apply some transformation on the real signatures that introduces some kind of writer variability. Furthermore, the duplication techniques modeling the writer variability proved to create realistic samples that resemble the real signatures [41] [42] [19] [23] [39] [21] [38]. Duplication methods can be classified according to the approach used by them, that can be based on geometrical transformations or be bio-inspired. Figure 3.1 depicts the taxonomy of the offline signature augmentation techniques [92].

Normally, compositional methods use a set of elements with a particular order to compose a new synthetic text sample [73]. Ruiz et al. (2020) [109] proposed a compositional method combining some geometrical figures such as lines, triangles, crosses, semi-ellipses,

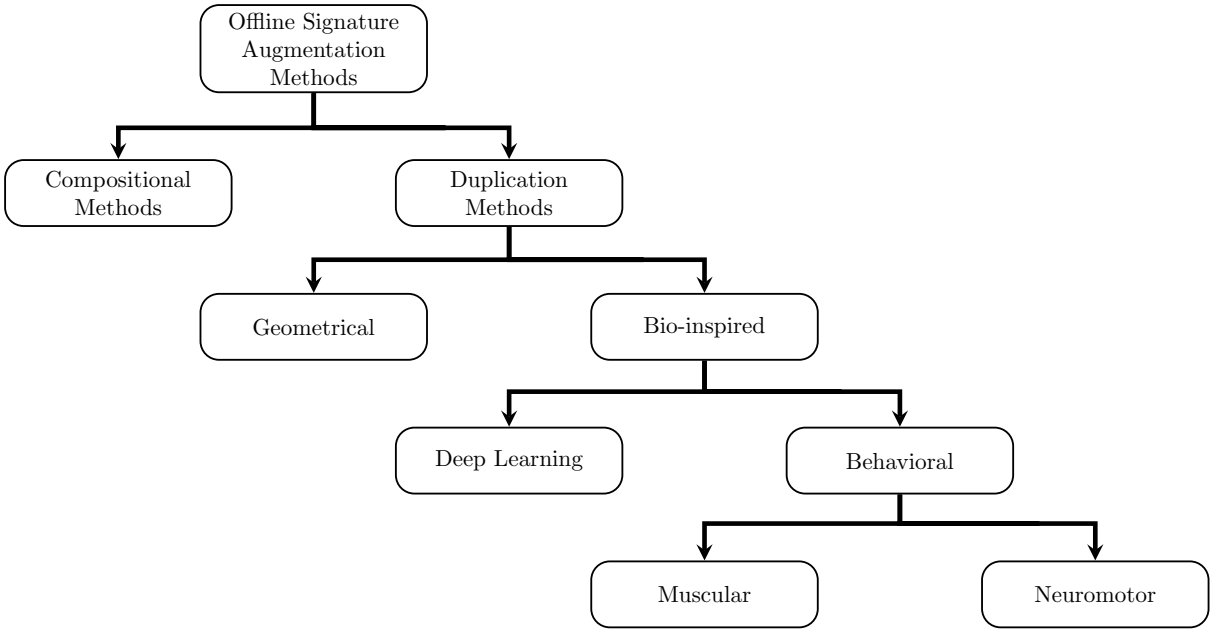


Figura 3.1: Taxonomy of the Offline Signature Augmentation Techniques

ellipses, circles, semi-circles, and cycloids. Despite the proposed composition method did not produced realistic signatures, the synthetic signatures helped a Siamese Neural Network to learn some basic geometrical features from signatures.

In the last decades, duplication techniques have evolved from simple geometrical transformations [65] [35] [48] to behavioral models [41] [42] [19] [23] [39] [21] [38]. Geometrical transformations such as rotation [65] [48], scale [65] [48], perspective [65], and displacement [35] [48] are applied to increase the number of signature samples. These kind of transformations can add some natural and unnatural distortions. Normally, the natural distortions may be used to generate synthetic genuine signatures [65] [35] [48]. Huang and Yang (1997) [65] also used unnatural distortions to generate synthetic signature forgeries. Despite of that, skilled forgeries may present natural distortions that resemble the genuine signatures [140] [39]. Therefore, forgery duplicates with unnatural distortions can decrease the performance of a signature verification system.

Most methods based on geometrical transformations use several parameters to represent the duplicate variability. However, the most of researches using these methods do not detail how each parameter was determined. Besides that, the geometrical transformations can generate duplicates that are not necessarily similar to the genuine signatures [21]. Despite these methods increase the performance of SVSs, they do not consider the behavior of each writer to generate the duplicates [65] [35] [48]. Since the most of methods based on geometrical transformations do not consider the writer's behavior, several bio-inspired methods were developed. The bio-inspired methods can be divided into Deep Learning and Behavioral approaches [92].

Besides feature learning and classification, deep learning can also be used in data augmentation tasks like offline signature augmentation. Yapıcı et al. (2020) [135] proposed the

use of Cycle-Generative Adversarial Network (Cycle-GAN) to duplicate signatures. Cycle-GAN learns how the offline signatures are generated using pairs of signatures. The model tries to learn the transformations that convert a signature A in a signature B and vice-versa. Yapıcı et al. (2020) compared their duplication method with traditional data augmentation techniques such as rotation, flipping and mirroring. They showed that their method achieved better results than the traditional techniques. Despite the network generated unrealistic duplicates, the performance of authors' signature verification system was improved when the duplicates were used to train the system.

The handwriting is a complex process that develops since the childhood until the adulthood. First, the shape and sequence of the strokes are learned. Later, the movements to reproduce the shape and the sequence are trained [42]. As this process improves, the writing becomes a repetitive and customary task [78]. Even the handwriting process is still not fully understood, some methods try to model the human's behavior when they are signing [92] [41] [42] [19] [23] [39] [21] [38]. These behavioral techniques can take in account different strategies such as muscular and neuromotor approaches. The approaches based on muscular models [41] try to recreate the trajectory of signature strokes and the effect of the speed during the movement of the writer's muscles. Since these methods are designed to duplicate flourished signatures, they can present not human-like duplicates when legible signatures are used [92].

Some behavioral methods are based on neuromotor theory [42] [19] [23] [39] [21] [38]. Instead of just considering the muscles, these methods also consider a set of skeleton parts, eyes, and the central nervous system to generate signatures. First, the brain creates and stores the signature generation plan. The central nervous system sends electric impulses to the eyes and to the muscles responsible by the writing. Consequently, the plan is executed by a series of muscular contractions and joint movements producing the desired signature [42].

Normally, the neuromotor inspired methods do not consider the sequence of the signature strokes. On the other hand, there are some of them that are mainly concerned about the signature trajectory plan [42] [39] [38]. This plan specifies the position and the sequence of strokes used to generate the signature. Despite the sequence information is only present in online signatures, it can also be extracted from offline signatures. Ferrer et al. (2015) [42] used the model proposed by Djioua and Plamondon (2009) [28] to extract the speed information from offline signatures, and created a signature trajectory plan. Moreover, Ferrer et al. (2018) [38] showed that the trajectory plan and the method complexity is related to the alphabet used to sign [92].

Like the muscular methods, the neuromotor-based methods try to model the desired motor effects presented in the signatures. These methods use a deformable grid which maps the distribution of the signature strokes and characters on the written surface [21]. To map such surface, some grid architectures such as hexagonal [39] [38], quadrangular [42], and sinusoidal [23] [21] are used. Ferrer et al. (2018) [38] showed that the grid density is associated to the character spacing and size of the different alphabets used to sign. While more sparse grids are indicated for western alphabets like Latin, dense grids are indicated for eastern alphabets like

Bengali and Devanagari [38]. However, Diaz et al. ([23, 21]) showed that the sinusoidal grid can be used for both kind of alphabets like Latin, Devanagari, and Bengali [92].

Besides the signatures depend on the human's behavior, they also depends on paper [41], ink [41] [24] [42] [23] [21], and the instrument used to write [78, p. 7-21, 27-29]. The paper and ink models are used to duplicate offline signatures and generate more realistic synthetic samples. While the paper models try to mimic the roughness of the written surface, the ink models try to mimic the deposition process of the ink on the paper surface [43].

Similar to geometrical methods, the behavioral methods use several parameters to control the writer variability. These kind of methods determine these parameters empirically or manually. These approaches may be time-consuming, complex, and may select parameters that does not describe the real writer variability [92]. Besides that, the reproduction of these approaches may be unfeasible. As consequence of the wrong definition of parameters, these approaches may present unnatural signatures, reducing the performance of signature verification systems [35] [21]. Table 3.6 summarizes the offline signature augmentation methods presented in the literature with their reference, strategy used by the method, if the methods considers the writer variability, and the writing alphabet, respectively.

Tabela 3.6: Offline signature augmentation methods (If the method considers the writer variability, it is marked with a ✓.)

Reference	**Method**	**Writer Variability**	**Alphabet**
Huang and Yan, 1997 [65]	Affine\ Geometrical Transformations	-	Latin\Chinese
Fang et al, 2002 [35]	Elastic Matching	-	Latin
Frias et al., 2006 [48]	Affine\ Geometrical Transformations	✓	Latin
Ferrer et al., 2013a [41]	Active Shape Model\ Muscular Model\ Paper Model\Ink Model	✓	Latin
Ferrer et al., 2015 [42]	Neuromotor Inspired Model\ Ink Model	✓	Latin
Diaz et al., 2016a [19]	Neuromotor Inspired Model\ Ink Model	✓	Bengali
Diaz et al., 2016b [23]	Neuromotor Inspired Model\ Ink Model	✓	Bengali\Devanagari
Diaz et al., 2017a [21]	Neuromotor Inspired Model\ Ink Model	✓	Latin
Ferrer et al., 2016 [39]	Neuromotor Inspired Model	✓	Latin\Chinese
Ferrer et al., 2018 [38]	Neuromotor Inspired Model\ Ink Model	✓	Bengali\Devanagari
Ruiz et al., 2020 [109]	Morphological\Geometrical Transformations\ Noise Addition	-	Latin
Yapıcı et al., 2020 [135]	Deep Learning Model	✓	Latin

According to the results presented in the literature, the neuromotor inspired theory methods generate more realistic duplicates than compositional methods and geometrical transformations [24] [21] [39] [38]. Among the neuromotor-inspired methods, the method proposed by Diaz et al. (2017) [21] can be highlighted. This method uses a neuromotor inspired model combined with a ink model to produce realistic duplicates. Diaz et al. trained a SVS with duplicates generated by their method and compared with the same SVS trained with the ones generated by geometrical transformations proposed in [48]. The SVS trained with duplicates generated by their method outperformed the other one trained with geometrical duplicates. Furthermore, they also showed that their method can be used to create synthetic signatures of different writing systems such as Latin [21], Bengali, and Devanagari [19] [23]. Despite of the advantages of the method, the parameters used to generate the signature duplicates are empirically defined. Therefore, a method to automatically optimize the parameters based on the writer variability is proposed in this work.

4 THE PROPOSED METHOD

As previously exposed, one of the main problems of offline signature verification is the number of signature samples available to train a signature verification system [22]. Despite the duplication techniques based on neuromotor theories provide realistic signature samples, they assume the writer variability, empirically determining the parameters that model it [21]. Therefore, a method to automatically optimize the writer variability parameters based on real data is proposed in this work. Figure 4.1 illustrates a signature verification system using the proposed method, from the parameter optimization to the final verification.

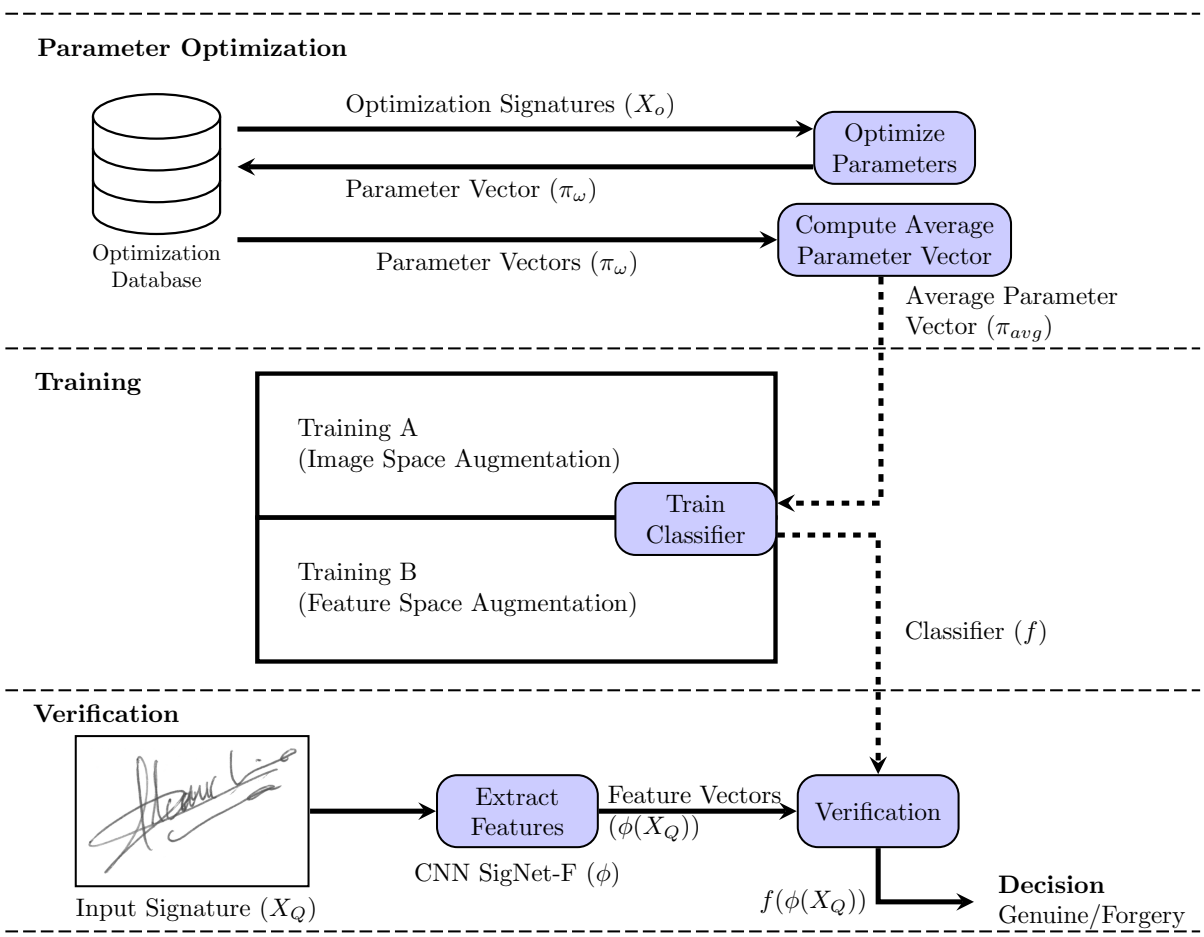


Figura 4.1: The proposed method being used in an offline signature verification system. While the Training A is only used for the signature augmentation in the image domain, the Training B is only used for the signature augmentation in the feature domain.

Taking into account a discriminant feature descriptor, the main hypothesis of this work is that the writer variability observed in the image space can be reflected in the feature space. Therefore, the method tries to model the writer variability only using the feature space. The visual aspect of the signatures is not directly assessed. The Convolutional Neural Network Signet-F (ϕ) is used to extract the feature vector $\phi(X)$ from each signature image X. Considering a set of

writers, $\phi(X_O)$ is used to optimize the parameters which describe the variability of a writer ω. As result of the optimization, there is a parameter vector π_ω for each writer ω that represents their variability. Subsequently, the average parameter vector π_{avg} is calculated using all the available parameter vectors in the optimization database [92].

The proposed method consider three kinds of signature augmentation techniques, one in the image space using Duplicator [21], and other two in the feature space using a Gaussian filter [85] and a variation of the Knop's method [77]. For the image space augmentation, Duplicator uses the average parameter vector π_{avg} and the signature X of the training set to generate duplicates X_D, respecting the writer variability. With the signatures X and duplicates X_D, the CNN ϕ is used to extract the feature vectors $\phi(X)$ and $\phi(X_D)$, respectively (Figure 4.2). For the feature space augmentation, the Gaussian filter (Figure 4.3) or the variation of the Knop's method (Figure 4.4) uses the average parameter vector π_{avg} and the signature feature vectors $\phi(X)$ to generate new feature vectors $\phi(X_D)$ respecting the writer variability [92].

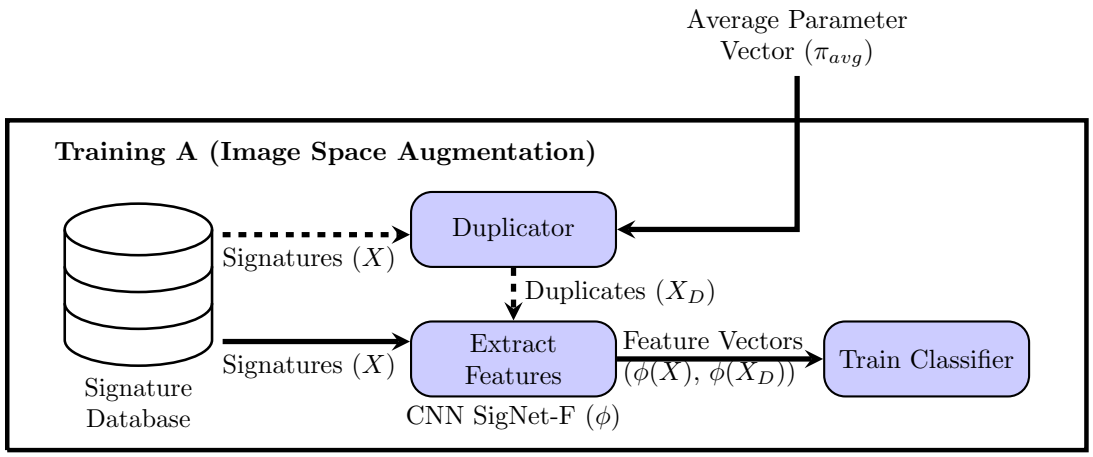


Figura 4.2: Training A is only used when the duplicates are generated in the image domain to train the classifiers.

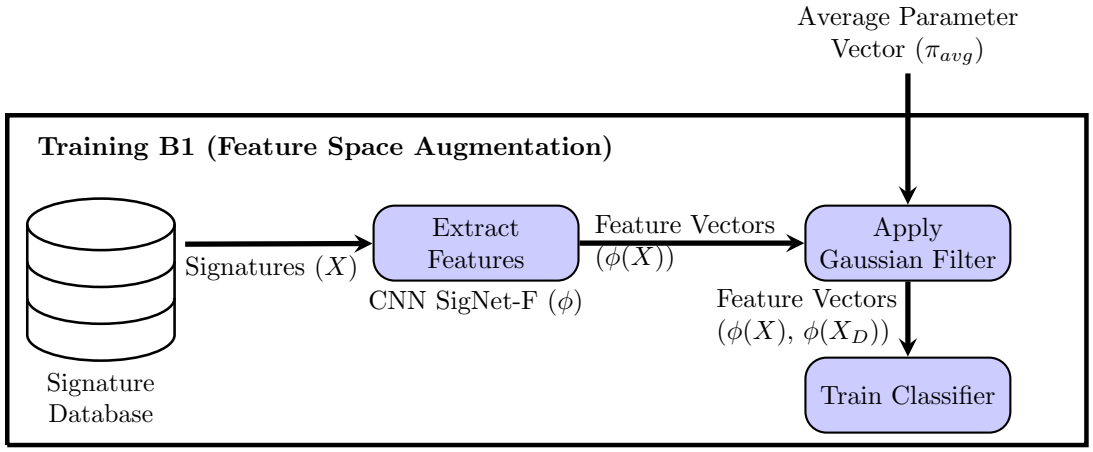


Figura 4.3: Training B1 is only used when the duplicates are generated using the Gaussian filter to train the classifiers.

For each writer of the system, the feature vectors $\phi(X)$ and $\phi(X_D)$ are used to train a classifier f. For a signature that will be verified X_Q, the CNN ϕ extracts feature vector $\phi(X_Q)$

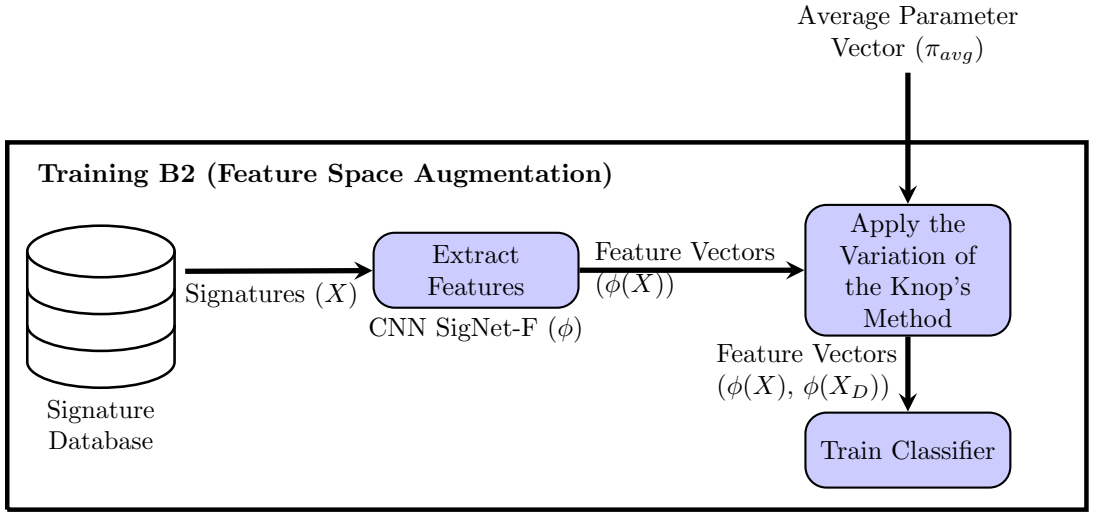


Figura 4.4: Training B2 is only used when the duplicates are generated using the variation of the Knop's method to train the classifiers.

and submits it to the previously trained classifier f. The classifier f use the feature vector $\phi(X_Q)$ to decide if the signature X_Q is a genuine signature or a forgery [92]. Each component of the signature verification system depicted in Figure 4.1 is detailed in the next subsections.

4.1 OFFLINE SIGNATURE DATASETS

To perform the experiments, the offline handwritten signature datasets GPDS-960, MCYT-75, and CEDAR were used. To compare the results with Hafemann et al. (2017a) [56], a similar experimental protocol was adopted, including for the datasets partitioning. Development dataset $\mathcal{D}$ is composed of the last 581 writers, which is subdivided into $\mathcal{D}_{\mathcal{L}}$, $\mathcal{D}_{\mathcal{T}}$, and $\mathcal{D}_{\mathcal{V}}$ subsets. While subset $\mathcal{D}_{\mathcal{L}}$ is used to a Convolutional Neural Network learn the features of the offline signatures, $\mathcal{D}_{\mathcal{T}}$ is used to monitor the evolution of the CNN learning. Both subsets contain the same 531 writers with different proportions of signature samples from each writer. While $\mathcal{D}_{\mathcal{L}}$ has 90% of the samples, $\mathcal{D}_{\mathcal{T}}$ has 10% of the samples. The optimization method uses some of the writers present in subset $\mathcal{D}_{\mathcal{L}}$. The remaining 50 writers compose subset $\mathcal{D}_{\mathcal{V}}$, which is used to make all the choices regarding the CNN model, hyperparameters of the classifiers, and the initial range used to optimize the parameter vectors [92].

The first 300 writers (GPDS-300) belong to exploitation subset $\mathcal{E}$. They are use to train and test the SVM classifiers used in the signature verification system. While subset $\mathcal{E}_{\mathcal{L}}$ contains samples that are used to train the classifiers, subset $\mathcal{E}_{\mathcal{T}}$ contains those that are used to test them. The MCYT-75 and CEDAR datasets were used to show generalization capability of the proposed method.

4.2 NORMALIZATION

Since a CNN of a fixed input size is being used in this work, signature images are normalized following the protocol adopted by [56]. Otsu's algorithm is used to remove the background of the original signature images and convert it to white (255). Subsequently, the center of mass of the signature is placed into the center of a window of height x width pixels. To keep the same proportion of the signature images, each dataset has its own window size (Table 4.1). Due to the different signature image sizes for different datasets, this process is needed. The color of all pixels is inverted and resized to 170×242 pixels. Afterward, the central region of the image with 150×220 pixels is cropped [92].

Tabela 4.1: Window Sizes Used in the Signature Normalization.

Dataset	**Window Size**
GPDS-960	952×1360
MCYT-75	600×850
CEDAR	730×1042

4.3 CONVOLUTIONAL NEURAL NETWORKS SIGNET AND SIGNET-F

As previously exposed, the CNN SigNet (Table 4.2) was proposed by Hafemann et al. (2017a) [56] to learn offline handwritten signature features. This architecture was based on CNN AlexNet that achieved outstanding results in classifying high-resolution images into 1,000 classes [81]. SigNet is composed of five convolutional (C1, C2, C3, C4, and C5), three max-pooling (P1, P2, and P3), and two fully connected layers (($P(y|X)$) and ($P(f|X)$)). The main advantage of using a CNN is it can learn translation invariant features and hierarchies of features [13, p. 123].

Tabela 4.2: Architecture of CNN SigNet-F

Layer	**Size**	**Parameters**
Input	$1 \times 150 \times 220$	
Convolution (C1)	$96 \times 11 \times 11$	*Stride*=4, pad=0
Pooling (P1)	$96 \times 3 \times 3$	Stride=2
Convolution (C2)	$256 \times 5 \times 5$	*Stride*=1, pad=2
Pooling (P2)	$256 \times 3 \times 3$	Stride=2
Convolution (C3)	$384 \times 3 \times 3$	*Stride*=1, pad=1
Convolution (C4)	$384 \times 3 \times 3$	*Stride*=1, pad=1
Convolution (C5)	$256 \times 3 \times 3$	*Stride*=1, pad=1
Pooling (P3)	$256 \times 3 \times 3$	*Stride*=2
Fully Connected (FC6)	2048	
Fully Connected (FC7)	2048	
Fully Connected + *Softmax* $(\boldsymbol{P(y|X)})$	M	
Fully Connected + *Sigmoid* $(\boldsymbol{P(f|X)})$	1	

Among the SigNet layers, the convolutional and fully connected layers have learnable parameters that are optimized during the CNN training [56]. Except for the last layer, for each one of the learnable layers a batch normalization followed by the Rectified Linear Units (ReLU) non-linearity operations are applied. Both operations are used to speed up the CNN training process. While the batch normalization stabilizes the distribution of the learnable parameters and reduces the influence of the parameter initialization [71], ReLU models the neuron's output allowing it to represent more color intensities than other kind of models [94] [81].

The convolutional layers are used to learn a set of local signature features, while the max-pooling layers are used to merge the most similar ones [86]. Subsequently, the fully connected layers establish the relationship between the features learned in the previous layers [133] [56]. The fully connected layers represent this relationship in a 2048-dimensional space. When a fully connected layer is used after one layer of the same kind, it enables the CNN to learn more complex hierarchies of features [13, p. 123]. The last two layers are used to classify the signatures. While the *Fully Connected + Softmax* ($P(y|X)$) layer is used to identify the owner of the signature, the *Fully Connected + Sigmoid* ($P(f|X)$) layer is used to verify if the signature is genuine or a forgery [56].

Due to the outstanding results provided on well known benchmarks and high discrimination capability of this approach, SigNet-F is used as a signature feature descriptor model in this work. It adopts a writer-independent feature learning strategy to learn a representation $\phi(X)$ from a development subset $\mathcal{D}_{\mathcal{L}}$. As the CNN learns, the discrimination of writers present in $\mathcal{D}_{\mathcal{L}}$ increases. When SigNet is trained using genuine and skilled forgeries, it is called SigNet-F. Since the CNN Signet-F use both kinds of signatures, it also learns to discriminate between genuine signatures and forgeries [92] [56].

While GPDS subset $\mathcal{D}_{\mathcal{L}}$ was used to a Convolutional Neural Network learn the features of the offline signatures, $\mathcal{D}_{\mathcal{T}}$ was used to monitor the evolution of the CNN learning. The CNN was trained during 60 epochs, with a initial learning rate of 0.001. After every 20 epochs, the learning rate was divided by 10. Since the deep learning model needs a great amount of data, random patches of 150×220 pixel were extracted from the normalized 170×242 pixel signatures. In the feature extraction phase, the CNN layer FC7 was used to extract the feature vectors with 2048 elements [92].

Each feature vector $\phi(X)$ was normalized using the function described in the Equation 4.1. The mean of the feature vectors is denoted by u, while the standard deviation vector is denoted by s [126]. The normalization tries to reduce the influence of features with great values over the features with small values [134]. Since this kind of normalization tries to keep the same Gaussian distribution for every feature vector, it helps to speed up the RBF SVM classifier training [126]. It also tries to improve the generalization capability of the features [130].

$$\phi(X)_{norm} = \frac{\phi(X) - u}{s} \tag{4.1}$$

4.4 DUPLICATOR

Diaz et al. (2017a) [21] proposed a method called Duplicator to generate synthetic offline signatures based on neuromotor theory and an ink model. It defines a set of 30 parameters to control the signature variability. The first 6 parameters (a_A^{min}, a_A^{max}, a_P^{min}, a_P^{max}, a_S^{min}, and a_S^{max}) are mainly responsible for describing the writer variability. To represent this writer variability, a sinusoidal transformation is adopted. To apply the sinusoidal transformation, these parameters are used to determine three values: α_A, α_P, and α_S. While the sine amplitude is determined by α_A, the α_P determines the sine period. Finally, the sine phase is delimited by α_S [92]. Each one of these values is determined randomly selecting a value of a uniform distribution from the minimum value to the maximum value. Equations 4.2, 4.3, and 4.4 show how each value is determined.

$$\alpha_A = \mathcal{U}_{\text{rand}}\left(\alpha_A^{min}, \alpha_A^{max}\right) \tag{4.2}$$

$$\alpha_P = \mathcal{U}_{\text{rand}}\left(\alpha_P^{min}, \alpha_P^{max}\right) \tag{4.3}$$

$$\alpha_S = \mathcal{U}_{\text{rand}}\left(\alpha_S^{min}, \alpha_S^{max}\right) \tag{4.4}$$

The Equation 4.5 controls the distortion of the image in the x axis, while the Equation 4.6 controls the same effect in the the y axis. It is possible to see how each parameter affect the signature image. M is determined by the image width, while N is determined by the image height.

$$x^S = x + \frac{M}{\alpha_A} \sin\left[2\pi\left(\frac{\alpha_P}{M}x + \alpha_S\right)\right] \tag{4.5}$$

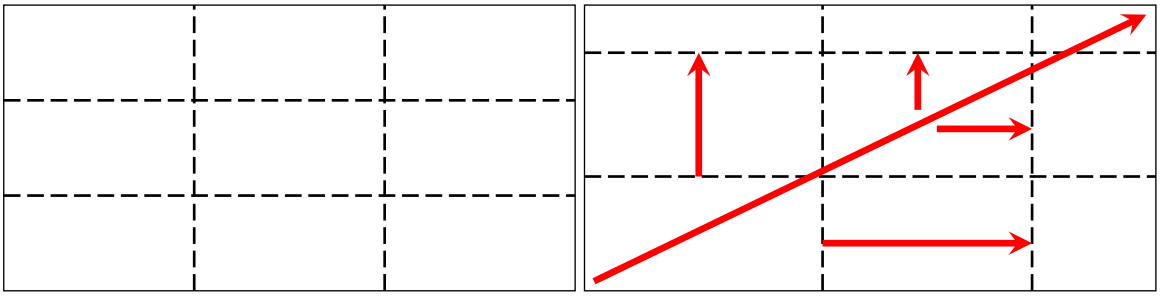
$$y^S = y + \frac{N}{\alpha_A} \sin\left[2\pi\left(\frac{\alpha_P}{N}y + \alpha_S\right)\right] \tag{4.6}$$

As shown in the Equations 4.5 and 4.6, the α_A is inversely proportional to the distortion applied to the signature image. Lower is the parameter α_A, higher is the distortion applied to the signature image. Therefore, the magnitude of α_A is related to the signature image dimensions. If we have signatures with large dimensions, we need large α_A values to reduce the distortion effect. To see the distortion effects caused by the parameters α_A^{min} and α_A^{max}, just the values of these parameters were changed. The other parameters were kept with the values presented in Table 4.3. The arrows show the distortion effect of α_A on Figure 4.5. When the α_A decreases, the upper and right border of the signature are compressed. In addition to the upper and right border compression, the lower and left border of the duplicate are stretched out. Figure 4.6 shows this behavior on real duplicates.

As shown in the Equations 4.5 and 4.6, the α_P is directly proportional to the distortion applied to the signature image. Higher is the parameter α_P, higher is the distortion applied to

Tabela 4.3: Default parameter values proposed by Diaz et al. (2017a) [21] 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
Displacement	$\{\xi_x^1, \sigma_x^1, \mu_x^1\}$	$\{-0.5, 20, 2 * \sigma_x^1\}$
	$\{\xi_x^2, \sigma_x^2, \mu_x^2\}$	$\{-0.5, 1.4 * \sigma_x^1, 2 * 1.4 * \sigma_x^1\}$
	$\{\xi_x^3, \sigma_x^3, \mu_x^3\}$	$\{-0.5, 1.8 * \sigma_x^1, 2 * 1.8 * \sigma_x^1\}$
	$\{\xi_y^1, \sigma_y^1, \mu_y^1\}$	$\{-0.5, 8, \sigma_y^1\}$
	$\{\xi_y^2, \sigma_y^2, \mu_y^2\}$	$\{-0.5, 1.2 * \sigma_y^1, 1.2 * \sigma_y^1\}$
	$\{\xi_y^3, \sigma_y^3, \mu_y^3\}$	$\{-0.5, 1.5 * \sigma_y^1, 1.5 * \sigma_y^1\}$
	k_1	0.33
	k_2	0.67
Ink Deposition	ψ	0.8
Inclination	ξ_S	-0.19
	σ_S	3.28
	μ_S	-1.30



(a) Normal surface (b) Distortion effect of α_A

Figura 4.5: Scheme of the distortion effect with α_A on the duplicate image.

the signature image. When the α_P decreases, the lower and upper border of the signature are stretched. The center of the signature is stretched as well. When the α_P increases, the lower and upper border of the signature are compressed. The center of the signature is compressed as well. The arrows show the distortion effect of α_P on Figure 4.7. Figure 4.8 shows the distortion effects of α_P on real duplicates.

After applying the intra-component distortions, a binary mask is created using the distorted image. First, a dilation operation with a rectangular kernel of 5×5 is applied in a copy of the image. Subsequently, the Equation 4.7 is applied in the resulting image. The pixels greater than 0 are considered as signature, the remaining pixels are considered as background. The unconnected strokes of the mask are labeled using the Efficient Run-Length Algorithm

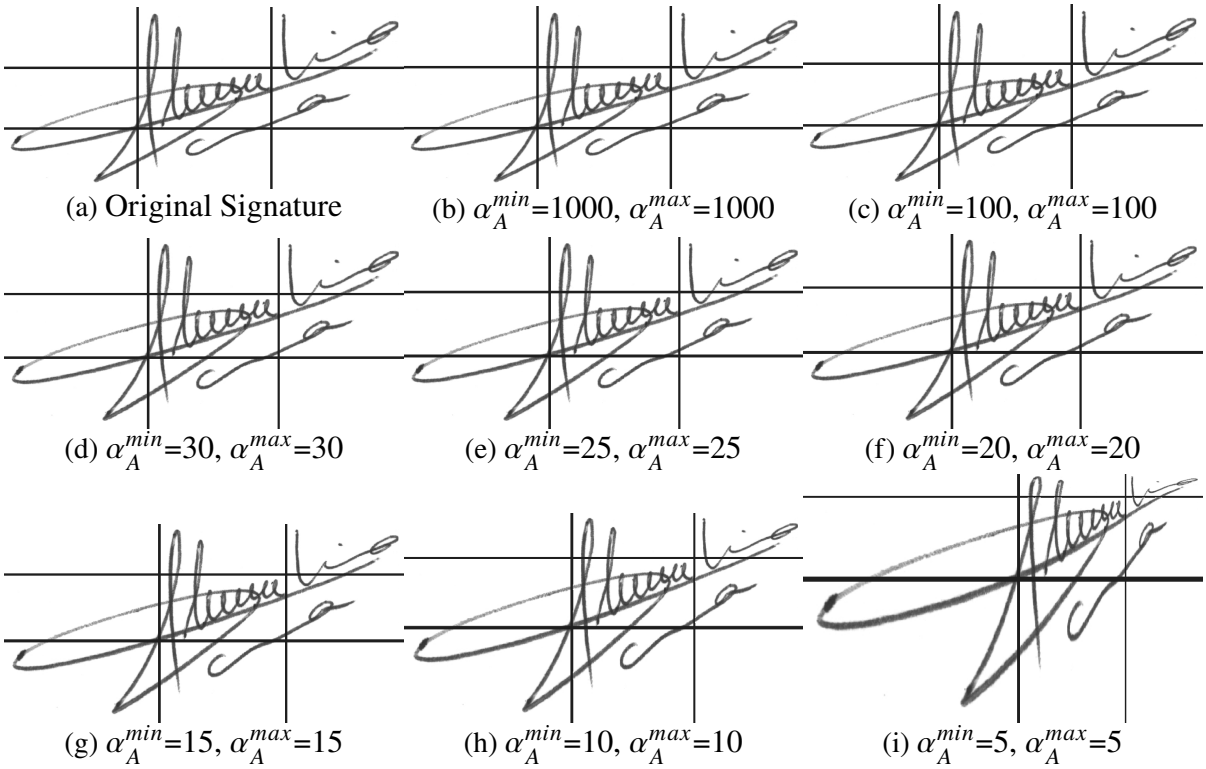


Figura 4.6: Distortion effect of α_A on real duplicate images.

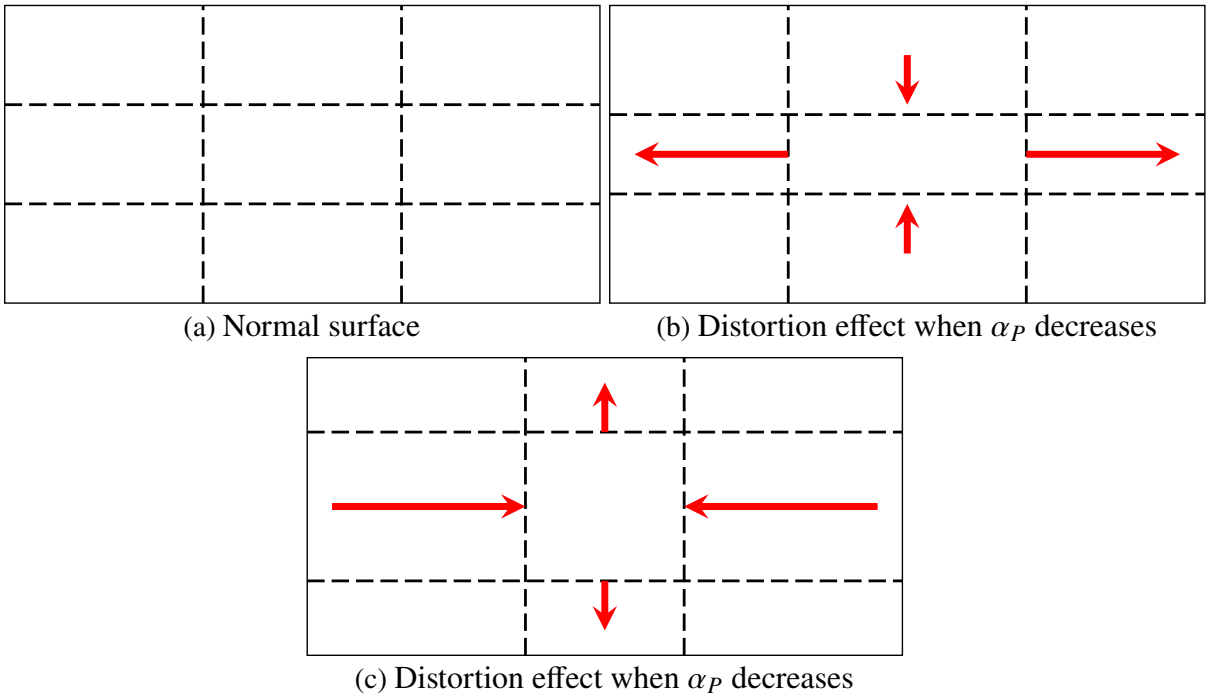
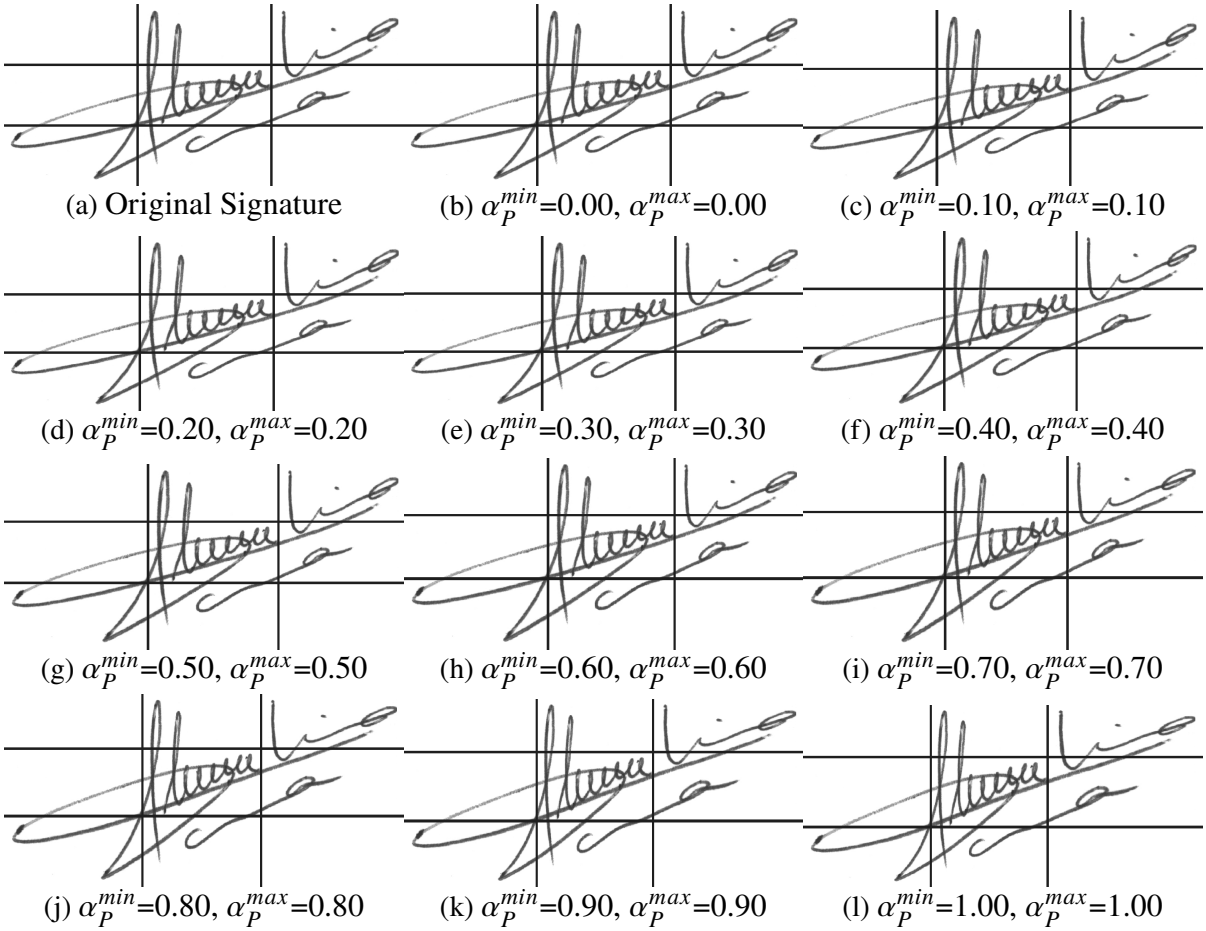


Figura 4.7: Scheme of the distortion effect with α_P on the duplicate image.

[62, p. 37-48]. The image background is numbered with zeros, while the signature strokes are numbered with one or greater numbers.

$$I_p(x,y) = \text{round}\left(\text{round}\left(\frac{I(x,y)\times n_L}{255}\right)\times\frac{255}{n_L}\right) \tag{4.7}$$



(a) Original Signature (b) α_P^{min}=0.00, α_P^{max}=0.00 (c) α_P^{min}=0.10, α_P^{max}=0.10

(d) α_P^{min}=0.20, α_P^{max}=0.20 (e) α_P^{min}=0.30, α_P^{max}=0.30 (f) α_P^{min}=0.40, α_P^{max}=0.40

(g) α_P^{min}=0.50, α_P^{max}=0.50 (h) α_P^{min}=0.60, α_P^{max}=0.60 (i) α_P^{min}=0.70, α_P^{max}=0.70

(j) α_P^{min}=0.80, α_P^{max}=0.80 (k) α_P^{min}=0.90, α_P^{max}=0.90 (l) α_P^{min}=1.00, α_P^{max}=1.00

Figura 4.8: Distortion effect for different α_P values.

The position of unconnected strokes are changed using a Generalized Extreme Value (GEV) distribution [79]. This kind of distribution is usually used to model the extreme behaviors of the natural phenomena such as temperatures, winds, floods, waves, earthquakes, etc [21]. For a random variable x, the GEV density function can be computed using the Equation 4.8.

$$f(x;\xi,\sigma,\mu) = \frac{1}{\sigma}\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{-\frac{1}{\xi}-1} e^{\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{-\frac{1}{\xi}}} \tag{4.8}$$

The unconnected strokes are displaced taking into account how big they are. First, the total number of signature pixels γ_T and the number of the unconnected stroke pixels γ_i are computed. Finally the proportion of the unconnected stroke pixels Γ that belongs to the signature is computed (Equation 4.9). The proportion Γ can be classified in three groups: the group that Γ is lower than k_1; the group that Γ is equals to or greater than k_1, and lower than k_2; and the group that Γ is equals to or greater than k_2. If the stroke is in one of these groups 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 change the position of the unconnected stroke. Therefore, Duplicator has 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) that control the distribution of unconnected strokes in the image.

$$\Gamma = \frac{\gamma_i}{\gamma_T} \tag{4.9}$$

The coordinates of each unconnected component is changed using the random values generated by the GEV distribution (Equations 4.10 and 4.11). The signal of the operation (+ or −) is randomly determined. For different writers, the three groups of proportion r can change. However, when Diaz et al. (2017a) [21] fixed the values of $k1$ and $k2$, they considered that these groups of proportions are the same for all writers.

$$x_{new} = x \pm \delta_x \tag{4.10}$$

$$y_{new} = y \pm \delta_y \tag{4.11}$$

The ink deposition effect is determined by ψ. When Duplicator superpose the strokes, this parameter regulates how dark they are. The last 3 parameters (ξ_S, σ_S, μ_S) are used to control the signature inclination. As the strokes displacement, the signature inclination uses a Generalized Extreme Value (GEV) distribution to choose a range of values. These values represent the inclination angles in degrees used to rotate the signature images [21].

4.5 GAUSSIAN FILTER

The One-dimensioal Gaussian filter is widely used to feature space augmentation (Equation 4.12). The Gaussian filter is defined by the mean μ and the standard deviation σ. Replacing the mean μ by zero in Equation 4.12, it results in Equation 4.13. Due to the simplicity of the Gaussian filter with $\mu = 0$, it is used to generate duplicates in the feature space [18] [114] [85].

$$G(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.12}$$

$$G(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{4.13}$$

Despite the low complexity of this filter, a parameter σ is still needed to control its intensity. The same σ is used to apply the transformation in all the elements x of the same feature vector $\phi(X)$. The standard deviation σ is randomly selected, considering a uniform distribution from σ_{min} to σ_{max} (Equation 4.14). Based on the same idea as in Duplicator, this interval is used to represent the writer variability, and thus optimized the parameters that determine this interval using the optimization process described in Section 4.7 [92].

$$\sigma = \mathcal{U}_{\text{rand}}\left(\sigma_{min}, \sigma_{max}\right) \tag{4.14}$$

4.6 VARIATION OF THE KNOP'S METHOD

Paley and Wiener (1934) [100, p. 146-153], and Box and Muller (1958) [12] proposed a method to generate random variables considering a Gaussian distribution and polar coordinates. Instead of using the sine and cosine operations of the Box-Muller method, Bell (1968) [5] used an approximation of these functions with a von Neumann rejection [129]. It reduced the computational complexity of the generation process. Later, Knop (1968) [77] simplified some equations of the algorithm to speed up the generation process. A variation of the Knop's algorithm is implemented in the numpy library by the method numpy.random.normal(), which is used in the experiments. Algorithm 1 presents the pseudo-code of this Knop's algorithm variation.

Algorithm 1 normal

Input:
loc: location
$scale$: scale
N: number of new samples
Output:
$samples$: new samples following a Gaussian distribution

$n \leftarrow 0$
$samples \leftarrow \emptyset$
while $n < N$ **do**
    $u \leftarrow 0$
    $v \leftarrow 0$
    $R2 \leftarrow 0$
    $L \leftarrow 0$
    ▷ Chooses a random angle θ with von Neumann rejection
    do
        $u \leftarrow 2 \times \mathcal{U}_{rand}(0,1) - 1$
        $v \leftarrow 2 \times \mathcal{U}_{rand}(0,1) - 1$
        $R2 \leftarrow u^2 + v^2$
    while $R2 >= 1$ **or** $R2 = 0$
    ▷ Chooses L randomly from a chi-squared distribution and normalizes it using $R2$
    $L \leftarrow \sqrt{\frac{-2 \times \ln R2}{R2}}$
    ▷ Generates a new sample
    if $n\%2 = 0$ **then**
        $s \leftarrow loc + u \times L \times scale$
    else
        $s \leftarrow loc + v \times L \times scale$
    end if
    $samples \leftarrow samples + s$
    $n \leftarrow n + 1$
end while

Since the algorithm works with polar coordinates, it defines the position of the new sample in terms of an angle θ and a radius L. To define the angle θ, the algorithm considers

the mean μ of the Gaussian distribution as the center of a circle with unitary radius. It also considers that the angles of the circle from 0 to 2π are uniformly distributed. Thus, the generated samples have the same probability to assume any angle θ in the circle [12]. Instead of computing θ using sine and cosine operations, it computes θ indirectly with von Neumann rejection [129]. It basically uses the trigonometric equation of Pythagoras. The square of the radius R^2 is equivalent to the sum of the squares of two random uniformly distributed variables (u and v) (Equation 4.15 and Figure 4.9). These two independent variables are uniformly distributed in the range (0, 1). To assure that u and v are inside the circle with unitary radius, the algorithm considers that the squared radius must be greater than 0 and smaller than 1. If this condition is not met, the von Neumann rejection is computed again for new values of u and v [5].

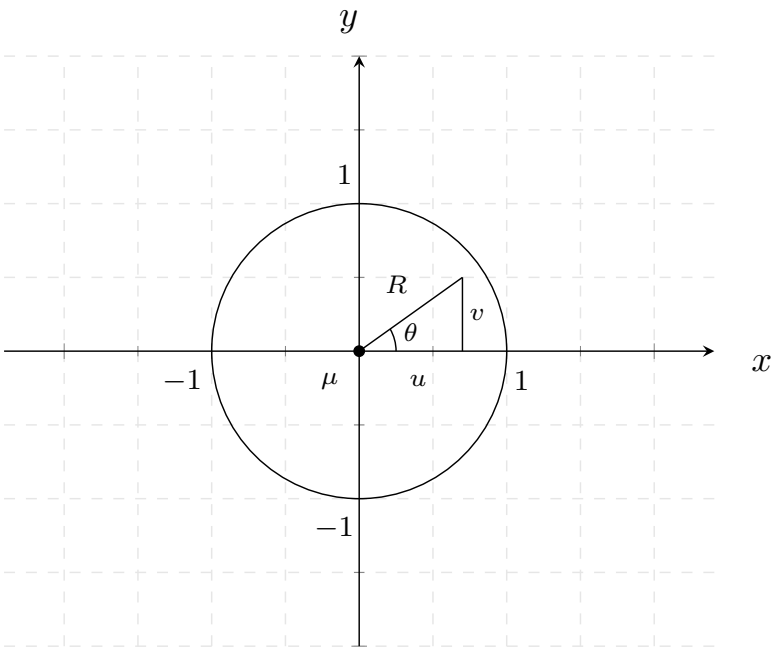
$$R^2 = u^2 + v^2 \tag{4.15}$$

After computing the angle θ, the algorithm computes L. It determines the distance between the center of distribution μ and the sample that will be generated. Due to the Gaussian distribution, the probability of the samples being generated close to the center is higher than the probability of they being generated in the borders of the same circle. L can be determined selecting a random number of a chi-squared distribution [5]. It also can be expressed as a negative logarithmic distribution in terms of R^2 [77] [5] (Equation 4.16).

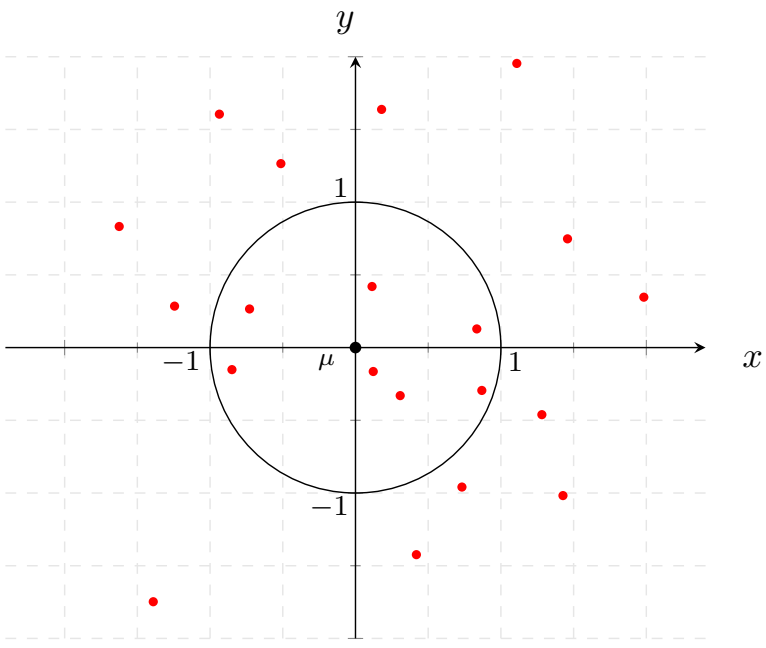
$$L = \sqrt{\frac{-2 \times \ln R^2}{R^2}} \tag{4.16}$$

Subsequently, the distribution is scaled using a variable $scale$. Then, the center of the distribution is translated to a predefined location loc. At the end of the process, the loc will be surrounded by the generated samples following a Gaussian distribution. If we want to generate the synthetic samples surrounding a genuine sample, we can use the location of the genuine sample as the predefined location loc. Figure 4.10a shows the scaling of the distribution using $scale = 0.375$, and Figure 4.10b shows the translation of the distribution to $loc = (1, 1)$.

The variation of the Knop's method [77] is used to generate duplicates in the feature space. Considering a genuine signature in the feature space as input, it generates the duplicates surrounding this signature following a Gaussian distribution. While the genuine sample represents the center of a hypersphere in the feature space, the duplicates represent the region covered by this hypersphere [12]. Since the variation of the Knop's method uses a Gaussian distribution to generate the duplicates, the probability of the duplicates being generated close to the center is higher than the probability of they being generated in the borders of the same hypersphere [5]. To regulates how sparse this hypersphere can be, it uses a $scale$ [77]. The $scale$ is defined randomly selecting a value of a uniform distribution ranging from $scale_{min}$ to $scale_{max}$. Both parameters also were optimized using the method proposed Section 4.7. They are used to represent the writer variability.

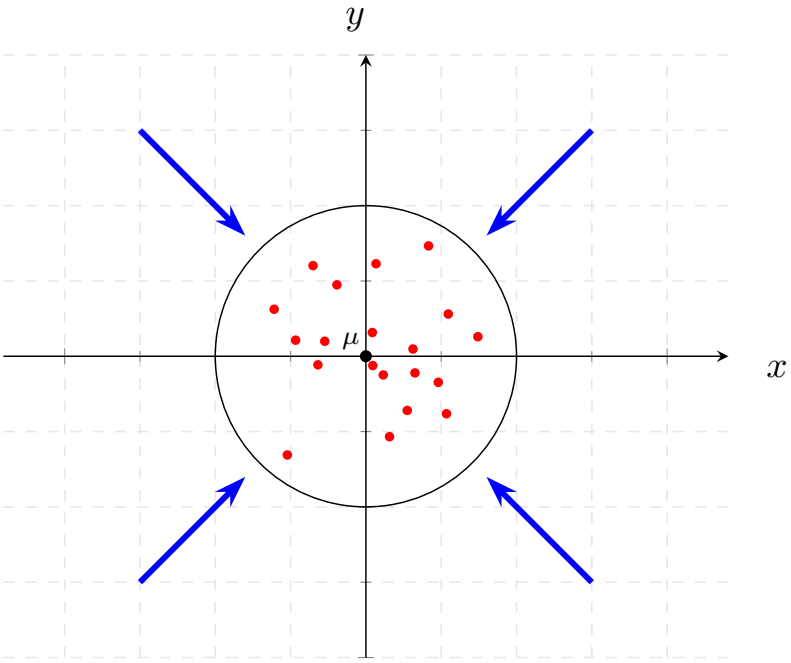


(a) Circle used to compute the angle θ

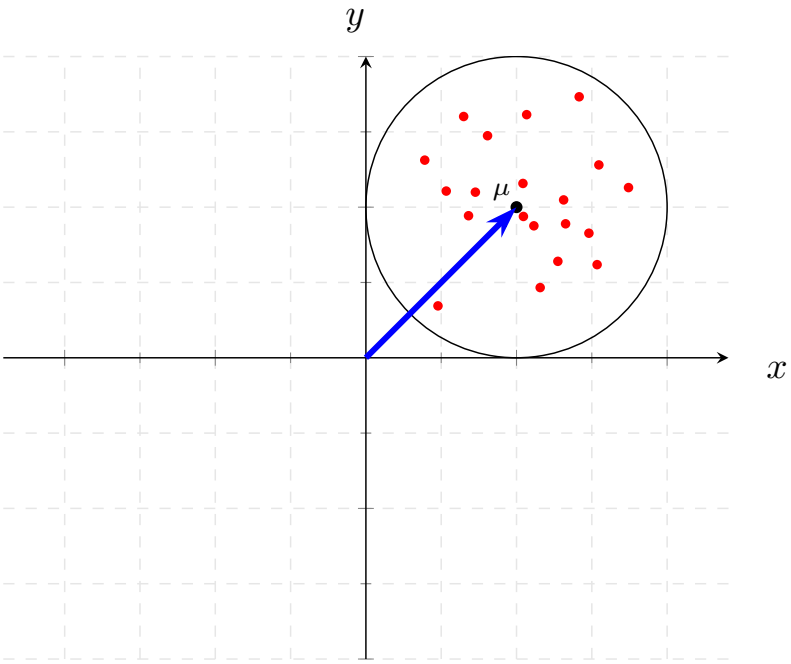


(b) 20 Samples generated following a Gaussian distribution

Figura 4.9: Generation process of the synthetic samples following a Gaussian distribution.



(a) Scaling the distribution



(b) Translation of the distribution

Figura 4.10: Scaling and Translation of the Gaussian Distribution

4.7 PARAMETER OPTIMIZATION

Two or more writers show a different behavior when they are signing. Consequently, they have different writer variabilities. At the same time, they can share some of these behaviors along the writing process [78, p. 7-30]. Therefore, a global set of parameters can be used to describe these common writer variability traits. The first six parameters of the duplicator were optimized because they are mainly responsible for describing the intra-personal variability of writer's signatures [21], and the optimization process is a time-consuming task [142]. For the Gaussian filter, the parameters σ_{min} and σ_{max} are used to represent the writer variability traits [92].

Parameter optimization tries to find the best set of parameters for signature augmentation methods, which allows them to generate synthetic samples respecting the writer variability. In this work, a variation of the Particle Swarm Optimization (PSO) [143] algorithm is used to find the first six variability parameters used by duplicator, and the two parameters used by Gaussian filter. Kennedy and Eberhart [76] proposed the original version of the PSO to optimize continuous nonlinear functions. This algorithm is inspired by the flock of birds or school of fish looking for places with plenty of food. The PSO [143] algorithm was chosen due to its implementation simplicity [141] and efficiency solving several kinds of problems [92] [66] [112] [34].

In a optimization problem, there is a search space of d dimensions in which the solution or solutions are located. Each particle π with index i represents a possible solution and its position in the search space. During the optimization process, each particle moves with a velocity v with index i in this space. For each iteration n of the algorithm, the velocity of the next iteration v_{id}^{n+1} (Equation 4.17) is updated considering the current particle velocity v_{id}^{n}, and the values of the local minima particle p_{id}^{n} and the global minima particle $\pi_{\omega d}^{n}$. While the local minima particle p_{id}^{n} represents the particle that is the nearest one to the desired solution only in the current iteration n, the global minima $\pi_{\omega d}^{n}$ represents the particle that is the nearest one to the desired solution among all particles. The velocity update process can increase or decrease the velocity of the particle in the search space. Despite of a particle with high velocity can cover a large area of the search space, it also can pass by a possible solution without considering it. On the other hand, a particle with low velocity cover a small area but if the solution is present in that area, it also has a higher chance to find it [92] [143].

$$\begin{aligned} v_{id}^{n+1} &= (1-\chi_o)v_{id}^{n} + \chi_o c_{o1}\gamma_o r_{1id}^{n}(p_{id}^{n} - \pi_{id}^{n}) \\ &\quad +\chi_o c_{o2}\gamma_o r_{2id}^{n}(\pi_{\omega d}^{n} - \pi_{id}^{n}) \end{aligned} \tag{4.17}$$

The v_{id}^{n+1} is also calculated using two uniformly distributed variables within [0,1], r_{1id}^{n} and r_{2id}^{n}. These variables help to diversify the places where the search will happen. The constant $1-\chi_o$ regulates this diversification. The constant χ_o itself regulates the intensity of the search to

find the best solution, in a specific place of the search space [143]. The constant c_{o1} controls the trend of the particles to approach from their local minima particle, while c_{o2} controls the trend of the particles to approach from their global minima particle [34]. The perturbation constant γ_o regulates the stability of the algorithm controlling the effect of both constants c_{o1} and c_{o2} at the same time [92].

In this work, the variation of the PSO proposed by Zhao (2016) [143] is used. It was implemented using the constants $(1-\chi_o) = (3-\sqrt{5})/2$, $\chi_o c_{o1}\gamma_o = (1+\sqrt{5})/2$, and $\chi_o c_{o2}\gamma_o = 1$. He showed that his variation of the PSO is more accurate, efficient, and stable than the traditional PSO. Replacing the constants of Equation 4.17 by the Zhao's constants results in Equation 4.18. Consequently, the position of the particle of the next iteration π_{id}^{n+1} in the search space is updated considering its velocity v_{id}^{n+1} (Equation 4.19) [92].

$$\begin{aligned} v_{id}^{n+1} &= \frac{(3-\sqrt{5})}{2} v_{id}^{n} + \frac{(1+\sqrt{5})}{2} r_{1id}^{n}(p_{id}^{n} - \pi_{id}^{n}) \\ &\quad + r_{2id}^{n}(\pi_{\omega d}^{n} - \pi_{id}^{n}) \end{aligned} \tag{4.18}$$

$$\pi_{id}^{n+1} = \pi_{id}^{n} + v_{id}^{n+1} \tag{4.19}$$

Each parameter of the parameter vector represent one of the d dimensions. For the image space augmentation, a particle π of 6 dimensions represents the parameter vector. For the feature space augmentation, a particle π of 2 dimensions the parameter vector. A random uniform distribution with low and high limits was used to initialize the parameter values. These low and high limits were defined using the $\mathcal{D}_V$ and considering the constraints of each data augmentation technique used in this work. The limits used during the optimization process are presented in Table 4.4. The low limits of the *max* parameters are determined using the value assigned to the *min* parameters. For example, when α_A^{min} is equals to 15, the low limit of the parameter α_A^{min} has the same value.

The limits of the duplicator parameters reported in Table 4.4 were chosen taking into account the extreme values of the default parameters proposed by Diaz et al. (2017a) [21]. As previously exposed in Section 4.4, if both α_A^{min} and α_A^{max} have values equal to 10 or lower than it, the duplicator provide duplicates with higher distortions than when these values are higher than 10. On the other hand, if α_A has large values, the duplicates will be equal to the signature used as input. Therefore, the limits of α_A^{min} were defined from 10 to 100. Diaz et al. (2017a) [21] argue that the values of α_P^{min}, α_P^{max}, α_S^{min}, and α_S^{max} should be between 0 and 1. When the duplicator uses these extreme values, it generates duplicates with an unnatural and unrealistic aspect [92].

For the Gaussian filter, the limits were chosen considering its mathematical constraints. For σ equals to 0, the Gaussian filter function presents a mathematical indetermination. Besides that, an important aspect is the degree of perturbation applied in feature space by the Gaussian

Tabela 4.4: Parameter Initialization Range During the Parameter Optimization

Parameter	Parameter Initialization Range	
	Low Limit	**High Limit**
α_A^{min}	10.00	100.00
α_A^{max}	α_A^{min}	100.00
α_P^{min}	0.00	1.00
α_P^{max}	α_P^{min}	1.00
α_S^{min}	0.00	1.00
α_S^{max}	α_S^{min}	1.00
σ_{min}	0.01	1.00
σ_{max}	σ_{min}	1.00
$scale_{min}$	0.00001	1.00
$scale_{max}$	$scale_{min}$	1.00

filter function. If a great perturbation is applied in the original signature feature vector, the Gaussian filter can generate a synthetic feature vector that does not resemble the original signature anymore. With that in mind, the limits were defined from 0.01 to 1.00. The Gaussian filter's behavior and its parameter intervals are explored in the experiments of the Chapter 5 [92].

For the variation of the Knop's method, the limits also were chosen considering its mathematical constraints. For $scale$ equals to 0, the variation of the Knop's method does not apply any transformation on the genuine signatures in the feature space. The $scale$ controls the sparsity of the duplicates considering a genuine sample as the center of the distribution. If $scale$ is great, the duplicates will be placed far from the genuine samples. Consequently, these duplicates will not resemble the genuine signatures in the feature space. With these constraints in mind, the limits were defined from 0.00001 to 1.

Following a similar idea to the parameter initialization, Table 4.5 presents the limits used to initialize the velocities during the optimization process. For each parameter or dimension d, there is a corresponding velocity v_d that is used during the optimization. The velocities of each particle are randomly initialized considering a uniform distribution from the low limit to the high limit. Since the parameters α_A^{min} and α_A^{max} are greater than other parameters, their velocities v_1 and v_2 also are greater than other ones. Furthermore, the α_A^{min} and α_A^{max} have a greater influence than other parameters. Therefore, it is important that the particle moves in the search space with a non-zero velocity. The negative and positive velocities allow the optimization process to move in different directions of the search space. Moreover, it can reduce the velocity of the particles to search for solutions in promising areas of the search space [143].

The parameter and velocity ranges were determined using $50\mathcal{D}_\mathcal{V}$. It was considered the number of iterations that the PSO took to converge. For the most writers, it took up to 10 iterations to converge. However, for some of them, it took up to 20 iterations to converge. Therefore, the velocity ranges that converged in 20 iterations were used.

Besides defining the values related to the particles, PSO needs an objective function that guides the optimization process. In this work, a variation of the silhouette index is used as an

Tabela 4.5: Velocity Initialization Range During the Parameter Optimization

Velocity	Velocity Initialization Range	
	Low Limit	High Limit
v_1 (α_A^{min})	-3.00	-1.00
v_2 (α_A^{max})	1.00	3.00
v_3 (α_P^{min})	-0.10	0.10
v_4 (α_P^{max})	v_3	0.10
v_5 (α_S^{min})	-0.10	0.10
v_6 (α_S^{max})	v_5	0.10
v_1 (σ_{min})	-0.10	0.10
v_2 (σ_{max})	v_1	0.10
v_1 ($scale_{min}$)	-0.10	0.10
v_2 ($scale_{max}$)	v_1	0.10

objective function. Normally, the traditional silhouette index [108] is used to show how good are two or more clusters. When the silhouette index assumes the 1 value, the clusters are compact and are far from each other. In terms of variability, they have as small intraclass variability and a large interclass variability. When the silhouette index assumes the -1 value, it means that the data does not belong to the current clusters. Finally, when the silhouette index assumes the 0 value, the cluster are perfectly overlaid. In terms of variability, the clusters have exactly the same intraclass and interclass variability. Therefore, this last case is the goal of the whole optimization process. Basically, the optimization process tries to find the parameter vector that brings the silhouette index value close to 0 [92].

As previously exposed, the silhouette index evaluates the sparsity of each clusters and the distance between them at the same time. To perform this task, the silhouette index uses a dissimilarity function $d(.)$. Despite several functions can be used to calculate the dissimilarity between the elements of the clusters, Rousseeuw (1987) [108] recommends the use of the Euclidean distance as a dissimilarity function. To measure the sparsity inside each cluster Equation 4.20 is used. It calculates the average dissimilarity between the i_{th} element and the elements of the same cluster C_s (Figure 4.11).

$$a(\phi(X_i)) = \frac{\sum_{j=1}^{n_{Cs}} d(\phi(X_i); \phi(X_j))}{n_{Cs} - 1} \tag{4.20}$$

To measure how far the clusters are from each other, the average dissimilarity between the i_{th} element and the elements of other cluster C_r is calculated using Equation 4.21. Afterward, Equation 4.21 is used in Equation 4.22 to calculate the minimum distance between the border of the clusters C_s and C_r. This indicates the interclass variability of the clusters. It is important to highlight that the clusters C_s and C_r are different from each other [92].

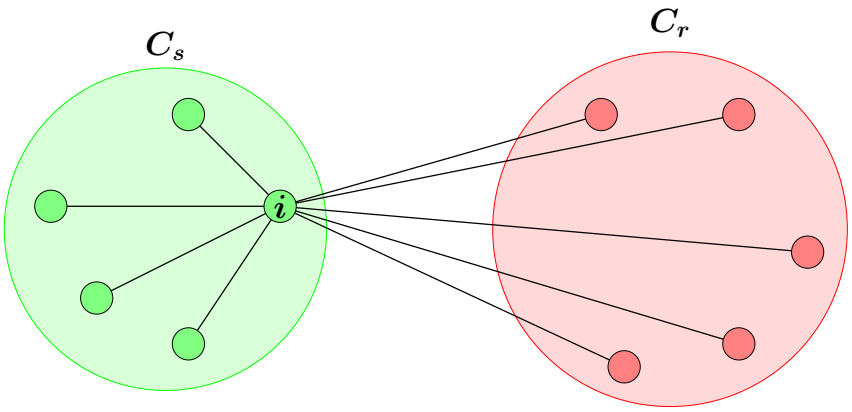


Figura 4.11: Distances between the i_{th} element of the cluster C_s and the other elements, including the elements of the cluster C_r.

$$d(\phi(X_i); Cr) = \frac{\sum_{j=1}^{n_{Cr}} d(\phi(X_i); \phi(X_j))}{n_{Cr}} \tag{4.21}$$

$$b(\phi(X_i)) = \min \{d(\phi(X_i); Cr)\} \tag{4.22}$$

Since Equation 4.20 evaluates the intraclass variability and Equation 4.22 evaluates the interclass variability, they are combined in Equation 4.23 to evaluate both variabilities concurrently. It is done using an arbitrary feature vector $\phi(X_i)$ as reference. To normalize the resulting value, it is divided by the maximum value between the intraclass and interclass variability [92].

$$\delta(\phi(X_i)) = \frac{b(\phi(X_i)) - a(\phi(X_i))}{\max \{b(\phi(X_i)); a(\phi(X_i))\}} \tag{4.23}$$

To evaluate all clusters at the same time, it is necessary to calculate $\delta(\phi(X_i))$ for all the cluster elements. The $\delta(\phi(X_i))$s are summed and divided by the total number of elements n_C in all clusters. As result, the silhouette index Δ (Equation 4.24) function ranges from -1 to 1 (Figure 4.12).

$$\Delta = \frac{\sum_{i=1}^{n_C} \delta(\phi(X_i))}{n_C}, \Delta \in [-1, 1] \tag{4.24}$$

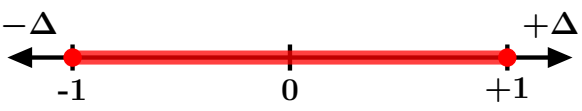


Figura 4.12: Range of the Silhouette Index function.

As previously exposed, the PSO needs to minimize the silhouette index function to find the parameter vector that best describe the writer variability. However, the minimum value of

this function is -1 and the objective value is 0. Since the silhouette index is a symmetric function and the objective value is located in the middle of the range, a modulus operation can be applied to this function to use only the positive side of it (Figure 4.13). Consequently, Equation 4.25 simplifies the minimization of the silhouette index function during the optimization process. The absolute silhouette index $|\Delta|$ is used as the objective function of the Particle Swarm Optimization algorithm.

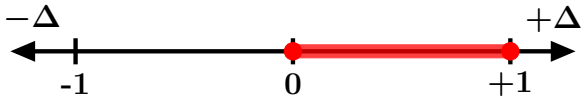


Figura 4.13: Range of the Absolute Silhouette Index function.

$$|\Delta| = \left| \frac{\sum_{i=1}^{n_C} \delta(\phi(X_i))}{n_C} \right|, |\Delta| \in [0, 1] \tag{4.25}$$

First, the PSO generates a set of particles in the search space. These particles represent the parameter vectors used to generate the synthetic samples, in the image or feature space domain. Considering only the image space domain, the duplicator generates a duplicate X_D for each signature X_o of the writer. After normalizing the signatures and duplicates, the feature vectors $\phi(X_o)$ and $\phi(X_D)$ are extracted from them. Considering only the feature space domain, the signatures of the writer are normalized and the feature vectors $\phi(X_o)$ are extracted. The Gaussian filter generates a new synthetic feature vector $\phi(X_D)$ for each original feature vector $\phi(X_o)$. The absolute silhouette index $|\Delta|$ is calculated using the feature vectors of the genuine signatures $\phi(X_o)$ and the synthetic samples $\phi(X_D)$. If the clusters of genuine and synthetic samples have a similar or equal variability, the $|\Delta|$ is equals to 0 or close to it. For each writer, the parameter vectors with the lowest absolute silhouette indices are chosen and saved. The parameter vectors are updated using the Equations 4.18 and 4.19. This process is repeated until the stop criteria is achieved. At the end of the process, there is optimized parameter vector π_ω for each one of the writers present in the optimization database. Subsequently, the average parameter vector (π_{avg}) is calculated with these parameter vectors. The π_{avg} describes the common behavioral biometric traits shared by these writers in the optimization database [92].

Despite this work does not consider the visual quality of the synthetic samples directly, it is possible to evaluate them using their feature vectors $\phi(X_D)$. Therefore, the quality of the duplicates generated by the proposed method and the duplicator with the default parameters π_{def} was assessed using the absolute silhouette index. Section 5.1 details how the synthetic samples were assessed.

4.8 TRAINING

After optimizing the parameter vectors, they can be used to generate synthetic samples to train the Writer-Dependent classifiers. These classifiers were trained taking into account four different scenarios. In the first scenario that is used as baseline, only the original genuine signatures are used to train the SVMs. In the second one, the original genuine signatures and the duplicates generated using the default parameter vector π_{def} presented in Table 4.3 are used. In the third one, the duplicates are generated with the parameters found by the optimization process. Finally, in the last scenario, the synthetic feature vectors are generated using the Gaussian filter and the parameters found by the optimization process [92].

The same signature verification system proposed by Hafemann et al. (2017a) [56] is adopted in this work. It is used to assess the impacts of the proposed optimization method and to fairly compare this work with their work. The CNN SigNet-F is used to extract the feature vectors of the normalized genuine and the random forgery signatures. A Support Vector Machine (SVM) classifier with a RBF kernel was trained for each writer with these feature vectors. The genuine signatures of other writer are used as random forgeries. When the proportion of positive and negative examples for training is different, it can create models that tend to select one class more frequently than other. One way to avoid this problem is using different C weights for positive and negative classes. For the negative class, C^- is equal to 1. Before computing the C^+, it is necessary to calculate the skew ψ. The skew was calculated using the number of positive examples P (genuine signatures) and the number of negative examples N (random forgeries) used for training (Equation 4.26). Finally, the weight for the positive class C^+ was calculated multiplying C^- by ψ (Equation 4.27) [92] [56].

$$\psi = \frac{N}{P} \tag{4.26}$$

$$C^+ = \psi C^- \tag{4.27}$$

4.9 VERIFICATION

With the Writer-Dependent classifiers trained, the signature verification system is ready to be used. After normalizing each query signature image X_Q and extracting the feature vector $\phi(X_Q)$ with the SigNet-F, this feature vector is sent to an SVM classifier f. The classifier decides $f(\phi(X_Q))$ whether the signature X_Q is classified as genuine or a forgery sample. The mean Equal Error Rate (EER), mean False Rejection Rate (FRR), mean False Acceptance Rate of random forgeries (FAR_{random}), and the mean False Acceptance Rate of skilled forgeries ($FAR_{skilled}$) were used to assess the performance of the offline signature verification system [92]. The system was assessed taking into account the four previously exposed scenarios.

5 EXPERIMENTS

Solving an optimization problem is a time-consuming task and depends on several factors. The number of writers used to optimize the parameter vectors is one of them. Instead of using all writers of subset $\mathcal{D}_{\mathcal{L}}$, 20 writers of them 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) to compose the optimization dataset. As previously exposed in Section 4.7, a PSO algorithm is used to optimize the parameter vectors. They are optimized using 100 particles during 20 iterations.

After the optimization process, there is an optimized parameter vector for each writer, which represents the writer's variability. These parameter vectors are used to calculate the average parameter vector (π_{avg}). The duplicator's average parameter vector is denoted by π_{dup}, the Gaussian filter's one is denoted by π_{gauss}, and the variation of the Knop's method one is denoted by π_{knop}. The global parameter vector proposed by Diaz et al. (2017a) [21] is denoted by π_{def}. To expose the difference among these parameter vectors, they were used to compute the average absolute silhouette index $|\Delta|_{avg}$ using the optimization dataset $20\mathcal{D}_{\mathcal{L}}$.

For each writer of $20\mathcal{D}_{\mathcal{L}}$, the first 12 genuine signatures are used to build a genuine cluster in the 2048 dimensional SigNet-F feature space. Similarly, four clusters with 12 feature vectors were created: one with duplicates generated using the optimized parameter vector π_{dup}, one with duplicates generated using the optimized parameter vector π_{gauss}, one with duplicates generated using the optimized parameter vector π_{knop}, and one with duplicates generated using the default parameter vector π_{def}. For each writer, four absolute silhouette indices were computed: one between the genuine cluster and the duplicates generated with π_{dup}, one between the genuine cluster and the duplicates generated with π_{def}, one between the genuine cluster and the duplicates generated with π_{knop}, and one between the genuine cluster and the duplicates generated with π_{gauss}. For each of these parameter vectors (π_{dup}, π_{def}, π_{knop}, and π_{gauss}), an average of absolute silhouette index $|\Delta|_{avg}$ was computed.

Table 5.1 presents the parameter vectors π_{def}, π_{dup}, π_{knop}, and π_{gauss}, and their respective $|\Delta|_{avg}$s. As can be seen, the average parameter vector π_{dup} is very different from the parameter vector π_{def} proposed by Diaz et al. (2017a) [21]. Especially, the parameters α_A^{min} and α_A^{max} have greater values than those determined in π_{def}. As previously explained in Section 4.4, the parameters α_A^{min} and α_A^{max} control the distortion on the upper and right side of the signature images. When these parameters values decrease, the distortion applied in the signature images increases. Moreover, the average absolute silhouette index $|\Delta|_{avg}$ from π_{dup} is lower than the $|\Delta|_{avg}$ from π_{def}. It means that the parameter vector π_{dup} better models the writers' variability present in $20\mathcal{D}_{\mathcal{L}}$ than π_{def} does.

For the signature augmentation in the feature space, the Gaussian filter with the parameter vector π_{gauss} presented an $|\Delta|_{avg}$ lower than the duplicator's $|\Delta|_{avg}$ with π_{dup}. It suggests that

Tabela 5.1: Average parameter vectors optimized using subset $20\mathcal{D}_{\mathcal{L}}$ and their average absolute silhouette indices ($|\Delta|_{avg}$).

Parameters	Parameter Vectors			
	π_{def}	π_{dup}	π_{gauss}	π_{knop}
α_A^{min}	5.000	69.300	-	-
α_A^{max}	30.000	88.700	-	-
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	-
$scale_{min}$	-	-	-	0.827
$scale_{max}$	-	-	-	0.932
$\|\Delta\|_{avg}$	0.0153	0.047	0.040	0.004

the Gaussian filter with π_{gauss} may better represent the writers' variability than the duplicator with π_{dup} and π_{def}. As in duplicator, the distortions applied by the Gaussian filter depends on its parameter values. As previously explained in the Section 4.5, the Gaussian filter uses σ_{min} and σ_{max} to model the writer variability in the feature space. These parameters are used to determine σ, which is effectively used by the Gaussian filter to generate duplicates. Figure 5.1 shows the effect of the σ values in the quality of the duplicates for each writer of $20\mathcal{D}_{\mathcal{L}}$. It can be seen that the concavities represent the ideal intervals of σ for each writer. These intervals can change from one writer to another. After analyzing the information from Table 5.1 and Figure 5.1, it is evident that the optimization process combines the σ intervals of different writers to determine the average parameter vector π_{gauss}. Furthermore, despite the Gaussian filter is a simple technique, it needs optimized parameters to generate more realistic duplicates.

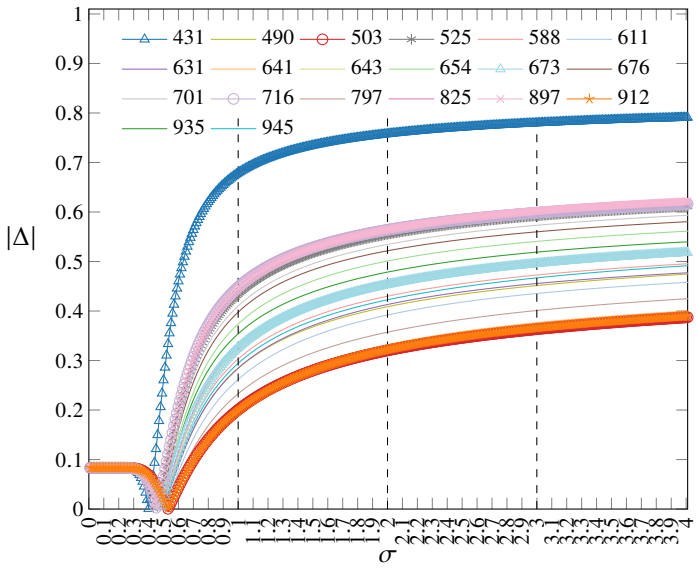


Figura 5.1: Effect of the Gaussian filter's parameter σ in the quality of the duplicates $|\Delta|$ for each writer of $20\mathcal{D}_{\mathcal{L}}$.

The proposed method was assessed considering the quality of the duplicates in the feature space (Section 5.1), and considering the performance achieved by the signature verification system when the duplicates were used for training (Sections 5.2 and 5.3). The limitations and advantages of the duplicator, and the Gaussian filter also were addressed (Section 5.5). All the experiments were carried out using at least one of the datasets exposed in Section 4.1.

5.1 QUALITY OF DUPLICATES IN THE FEATURE SPACE

As previously exposed, the main hypothesis of this work is that the writer variability observed in the image space can be reflected in the feature space. Unlike other works that assess the quality of duplicates considering the image space [21] [38], this work assesses the quality of duplicates considering the feature domain. It is expected that the original signatures and their duplicates will have similar characteristics for the same writer, and consequently, the signature and duplicate feature vectors should be similar as well. Furthermore, the duplicates should also preserve the original writer variability. However, this is only valid when a sufficiently discriminant feature descriptor is used to measure the dissimilarity between the signatures and the duplicates [56]. If the feature descriptor is weak, then the discrimination between them will be weak as well [123] [92]. This will be explored in the experiments of the Section 5.5.

As previously explained, the absolute silhouette index $|\Delta|_{avg}$ measures how close the duplicates are to the genuine signatures in the feature space. If the duplicates have a similar writer variability to the genuine signatures, then $|\Delta| \rightarrow 0$. Therefore, the three datasets GPDS-300, CEDAR, and MCYT-75 were used to assess the quality of the duplicates. For each writer of the dataset, the first 12 genuine signatures were used to build a genuine cluster in the 2048 dimensional SigNet-F feature space. Similarly, four clusters with 12 feature vectors were created: one with duplicates generated using the optimized parameter vector π_{dup}, one with duplicates generated using the optimized parameter vector π_{gauss}, one with duplicates generated using the optimized parameter vector π_{knop}, and one with duplicates generated using the default parameter vector π_{def}. For each writer, four absolute silhouette indices were computed: one between the genuine cluster and the duplicates generated with π_{dup}, one between the genuine cluster and the duplicates generated with π_{def}, one between the genuine cluster and the duplicates generated with π_{knop}, and one between the genuine cluster and the duplicates generated with π_{gauss}. For each of these parameter vectors (π_{dup}, π_{def}, π_{knop}, and π_{gauss}), the average of absolute silhouette index $|\Delta|_{avg}$ and the standard deviation were computed.

To show how big is the writer variability of each dataset in the feature space, the average cohesion co_{avg} of all writers was computed for each one of them. The cohesion measures the sparsity between all elements $\phi(X_i)$ of cluster in the feature space and its centroid μ (Figure 5.2). It is calculated using the squared differences between all $\phi(X_i)$ and μ (Equation 5.1) [125, p. 578]. The cohesions of all genuine clusters, their average co_{avg}, and their standard deviations

were computed for the three datasets: GPDS-300, MCYT-75, and CEDAR. Table 5.2 presents the $|\Delta|_{avg}$, the co_{avg}, and the standard devations of the three datasets.

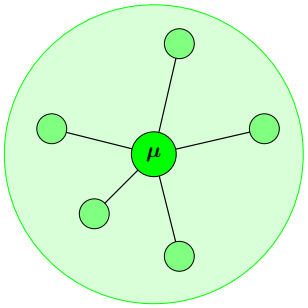


Figura 5.2: Graphical representation of the cluster's cohesion or cluster's sparsity.

$$co = \sum_{i=1}^{n} (\phi(X_i) - \mu)^2 \tag{5.1}$$

Tabela 5.2: Average Absolute Silhouette Indices, Average Sparsity of Genuine Clusters, and Standard Deviations for GPDS-300, CEDAR, and MCYT-75.

Metric	**Parameter Vector**	**Dataset**		
		GPDS-300	**MCYT-75**	**CEDAR**
$\|\Delta\|_{avg}$	π_{def}	0.140 ± 0.100	0.370 ± 0.120	0.700 ± 0.140
	π_{dup}	0.040 ± 0.050	0.150 ± 0.100	0.560 ± 0.180
	π_{gauss}	0.040 ± 0.040	0.100 ± 0.060	0.280 ± 0.130
	π_{knop}	0.005 ± 0.003	0.011 ± 0.003	0.010 ± 0.003
co_{avg}	-	18860.60 ± 1854.13	15900.48 ± 945.49	13788.87 ± 804.96

Despite the average parameter vector π_{dup} only represents the most common writer variability traits shared by a group of writers, it models the writer variability better than the π_{def}. Furthermore, as it can be seen in Figure 5.3, duplicates generated using the average parameter vector π_{def} may present some distortions that do not resemble the original signatures anymore. Consequently, these distortions are transferred to the feature space and increase the silhouette index values (Table 5.2). On the other hand, for the average parameter vectors π_{dup} and π_{gauss}, the lower silhouette indices show duplicates with a better quality in the feature space. Since both average parameter vectors π_{dup} and π_{gauss} were optimized using a subset of GPDS-960, these parameter vectors present the lowest $|\Delta|_{avg}$ for GPDS-300. Even the proposed method used a small number of writers (20 writers), it was able to model the writer variability of 300 different writers. Moreover, it also was able to transfer the writer variability modeled in GPDS $20\mathcal{D}_{\mathcal{L}}$ to CEDAR and MCYT-75 datasets.

The parameter vectors π_{def}, π_{dup}, π_{knop}, and π_{gauss} have difficulty to model the writer variability of CEDAR dataset. Since the parameter vectors π_{dup}, π_{knop}, and π_{gauss} were optimized using the GPDS dataset with highly sparse signatures in the feature space, it is expected that these parameter vectors represent the sparsity (co_{avg}) of similar signatures. However, the CEDAR

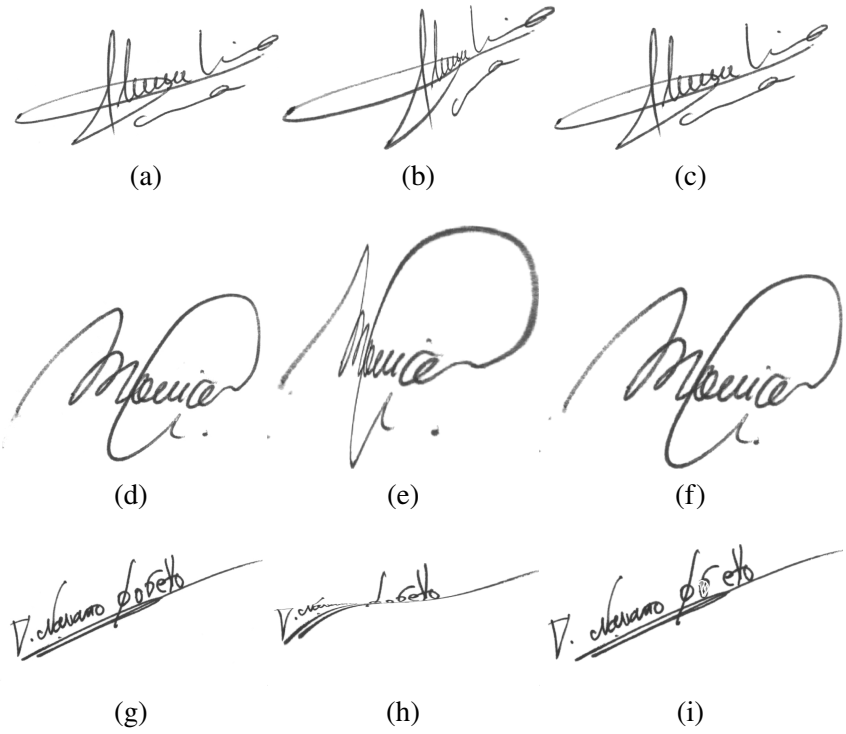


Figura 5.3: Examples of genuine signatures (a,d,g), duplicates (b,e,h) generated using Duplicator with the default parameter vector π_{def}, and duplicates (c,f,i) generated using Duplicator with the average parameter vector π_{dup}.

dataset presents signatures in the feature space with a smaller sparsity than those in the other datasets. Therefore, the parameter vectors π_{dup}, π_{knop} and π_{gauss} may not represent the writer variability of the CEDAR dataset so well as the other datasets. Despite of that, these parameter vectors represent the writer variability of the CEDAR dataset better than π_{def} does.

Another aspect that may contribute to the quality of the duplicates is the nature of the transformations applied to generate them. While the duplicator applies sinusoidal transformations in the image space and indirectly transfers them to the feature space, the Gaussian filter applies normal transformations directly in the feature space. As can be seen in Table 5.2, the variation of the Knop's method presents better duplicates for the three datasets, specially for GPDS dataset. The Gaussian filter presents better duplicates for two of three datasets, specially for CEDAR dataset. Furthermore, the $|\Delta|_{avg}$ and co_{avg} values suggest that the signature augmentation methods have more difficult to generate duplicates for low-sparsity signature clusters than for high-sparsity ones.

5.2 PERFORMANCE OF THE VERIFICATION SYSTEM USING DUPLICATOR

To compare the performance achieved by Hafemann et al. (2017a) [56] and the duplicates generated by the Duplicator, a similar experimental protocol was adopted in this work. Furthermore, to reproduce the limitation of the number of signatures in the real-world scenario, no more than three signatures per writer were used to train the classifiers of the signature verification system detailed in Chapter 4. While the genuine signatures of the writer and the

corresponding duplicates were used as a positive class to train classifiers, the genuine signatures of the other writers and the corresponding duplicates were used as a negative class (random forgeries) to train them. The genuine signatures and random forgeries were randomly selected for training and testing. For each genuine signature, the duplicator generated up to 22 duplicates to assess the effects of the number of duplicates in the EER, FRR, and $FAR_{skilled}$. For each number of duplicates, the experiment was carried out 10 times, and the average EER, average FRR, average $FAR_{skilled}$, and standard deviations were calculated for GPDS-300, MCYT-75, and CEDAR datasets. This set of experiments only considers the duplicates generated in the image space. Therefore, the duplicator using the parameter vector π_{def} and π_{dup} was used to generate the duplicates [92].

Table 5.3 presents the number of signature samples and duplicates used for training, and testing of each dataset. For GPDS, 14 genuine signatures of the other 581 writers ($\mathcal{D}_{\mathcal{L}}$ and $\mathcal{D}_{\mathcal{V}}$) and their duplicates were used as random forgeries. If for each one of the 14 genuine signatures there are 22 duplicates for 581 writers, there are a total of 187,082 random forgeries for training (14×581 genuine signatures plus $14 \times 581 \times 22$ corresponding duplicates for training). The SVM classifiers were tested with 10 genuine signatures, 10 random forgeries, and 10 skilled forgeries per writer from $\mathcal{E}_{\mathcal{T}}$. Since MCYT-75 does not have a development subset, 10 genuine signatures of other 74 writers and their duplicates were used as random forgeries. The SVMs were tested using 5 genuine signatures and 15 skilled forgeries. In CEDAR dataset, 12 genuine signatures of other 54 writers and their duplicates were used as random forgeries. The SVMs were tested using 10 genuine signatures, 10 random forgeries, and 10 skilled forgeries [92].

Tabela 5.3: Number of samples used for 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 subset**		**Testing subset**		
	G	**R**	**G**	**R**	**S**
GPDS-300	r $\in \{1, ..., 3\}$ + r $\times$ (d $\in \{0, ..., 22\}$)	$(14 \times 581) + (14 \times 581 \times$ d)	10	10	10
MCYT-75	r $\in \{1, ..., 3\}$ + r $\times$ (d $\in \{0, ..., 22\}$)	$(10 \times 74) + (10 \times 74 \times$ d)	5	-	15
CEDAR	r $\in \{1, ..., 3\}$ + r $\times$ (d $\in \{0, ..., 22\}$)	$(12 \times 54) + (12 \times 54 \times$ d)	10	-	10

Due to the limited number of signatures for test, only the GPDS-300 was used to asses the False Acceptance Rate of random forgeries (FAR_{random}). Figure 5.4 presents the FAR_{random} for GPDS-300 dataset. As can be seen, for any number of duplicates and genuine samples, the FAR_{random} is lower than 0.10 %. Moreover, the greater the number of genuine samples, the greater the FAR_{random}. It may indicate that some signatures of different writer may be considered of the same writer in the feature space, when a greater writer variability is used.

Figure 5.5 presents the FRR and $FAR_{skilled}$ for GPDS-300, MCYT-75, and CEDAR datasets, respectively. The greater the number of duplicates, the smaller the FRR. In another words, when the number of duplicates increase, there is more information about the signatures of a specific writer to distinguish them of the other signatures. On the other hand, when the numbers

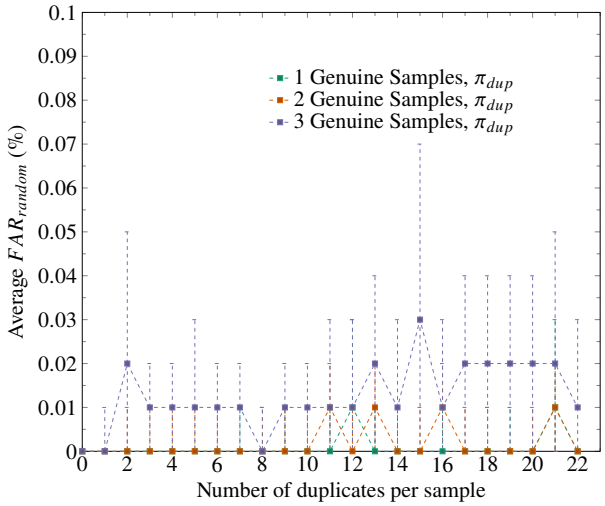


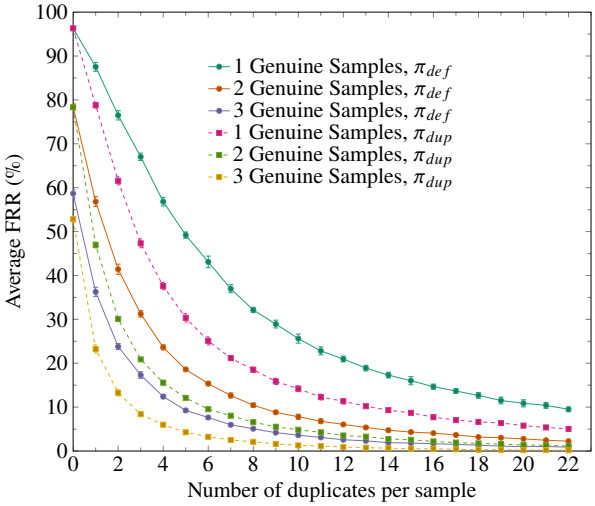
Figura 5.4: Average FAR_{random} achieved using GPDS-300 dataset with the proposed method.

of duplicates increase, there is more information that can increase the confusion between genuine signatures and skilled forgeries. Consequently, it increases the $FAR_{skilled}$.

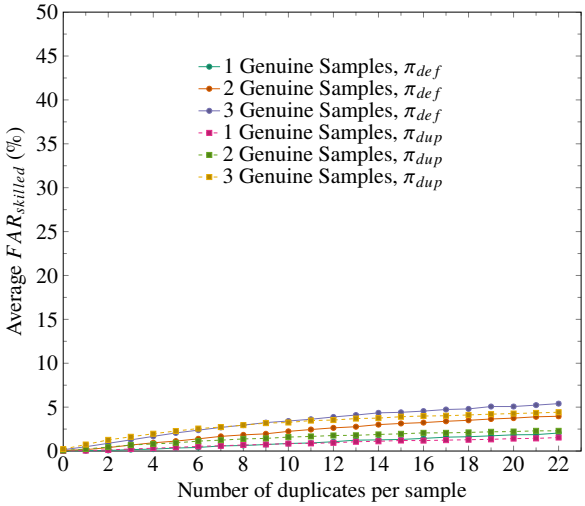
As previously explained in Section 2.1.2, there is a trade off between the False Rejection Rate (FRR) and the False Acceptance Rate (FAR) [107, p. 12]. Despite that, the trade off between the FRR and the $FAR_{skilled}$ is not directly proportional. With the number of duplicates increasing, while the FRR decreases subtly, the $FAR_{skilled}$ increases discretely. Furthermore, the number of genuine samples also affects the FRR and $FAR_{skilled}$. For only one genuine signature, there is less information about the writer variability. Therefore, the FRR is higher. On the other hand, there are less duplicates to confuse the classifiers as well. Consequently, the $FAR_{skilled}$ is lower. Moreover, the duplicates generated with the parameter vector π_{dup} achieved lower FRRs and FARs than did the ones generated using the π_{def}. It is an indicator that the duplicator generates more realistic duplicates with the optimized parameter vector π_{dup} than with the one proposed by Diaz et al. (2017a) [21]. Besides providing new information about the writer's signatures, π_{dup} also provides new information to distinguish between genuine signatures and skilled forgeries.

As can be observed, the duplicates are mainly responsible by decreasing the probability of the system consider the writer's signatures as forgeries. However, the new information introduced by the duplicates also increases the probability of the system consider skilled forgeries as genuine signatures. This behavior becomes more evident when the writer variability decreases, specially for CEDAR dataset. Due to small writer variability of the CEDAR dataset, the duplicates make it easy to distinguish between the signatures of different writers. However, they also make it difficult to distinguish between the skilled forgeries and genuine signatures. It strengths the hypothesis that is harder to generate duplicates with a small writer variability than with a great writer variability.

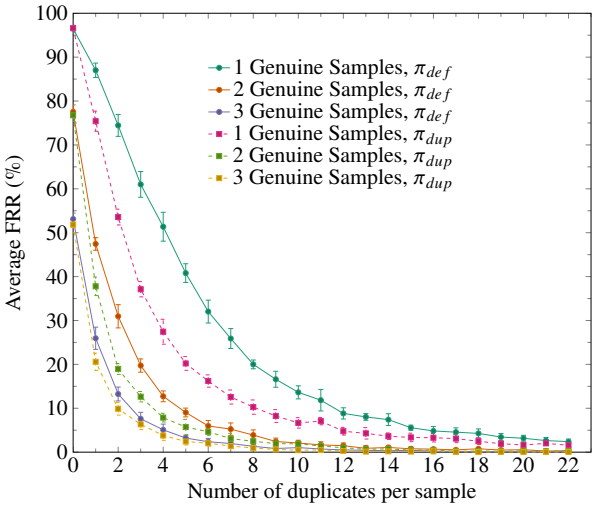
The Equal Error Rate (EER) summarizes the performance of False Rejection Rate and False Acceptance Rate [49]. Figure 5.6 shows the average EER achieved in GPDS dataset for each



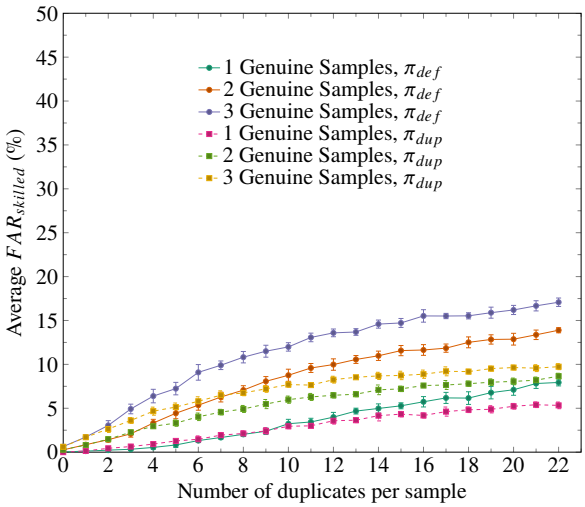
(a) GPDS-300: FRR



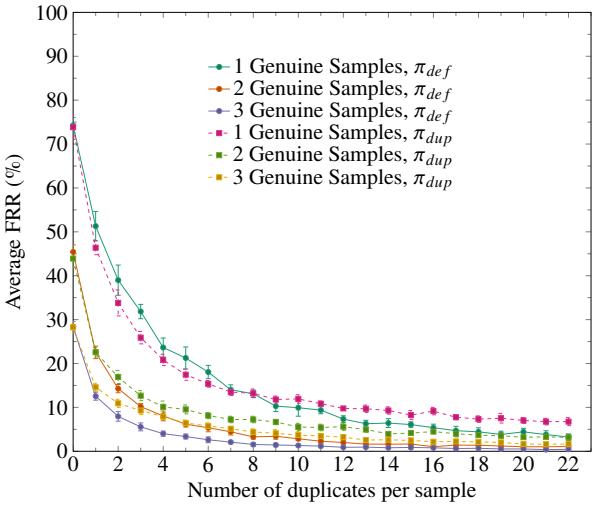
(b) GPDS-300: $FAR_{skilled}$



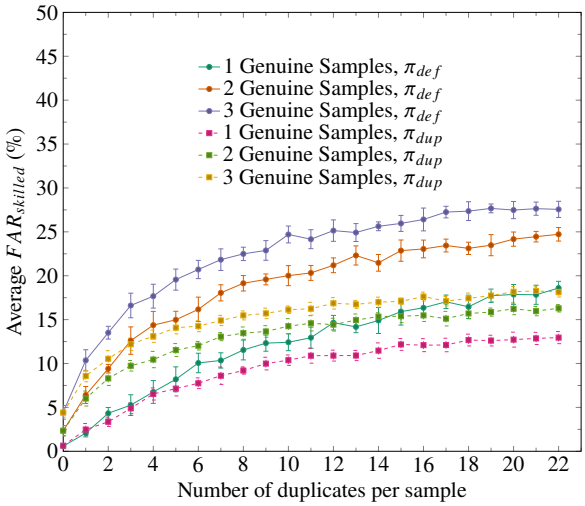
(c) MCYT-75: FRR



(d) MCYT-75: $FAR_{skilled}$



(e) CEDAR: FRR



(f) CEDAR: $FAR_{skilled}$

Figura 5.5: Average False Acceptance Rates and Average False Rejection Rates achieved with the proposed method.

number of duplicates. As can be seen, the number of duplicates is inversely proportional to the the average EER. Even using synthetic samples (duplicates) to train the classifiers, they provide additional information about the original signatures of the writers. Moreover, for the same number of genuine samples, the SVSs that used duplicates generated using π_{dup} outperforms the ones that used the duplicates generated using π_{def}. As previously exposed, the optimized parameter vector π_{dup} generates more realistic samples than those generated by π_{def}. Consequently, the samples generated by π_{dup} contributes to increase the performance of the signature verification system.

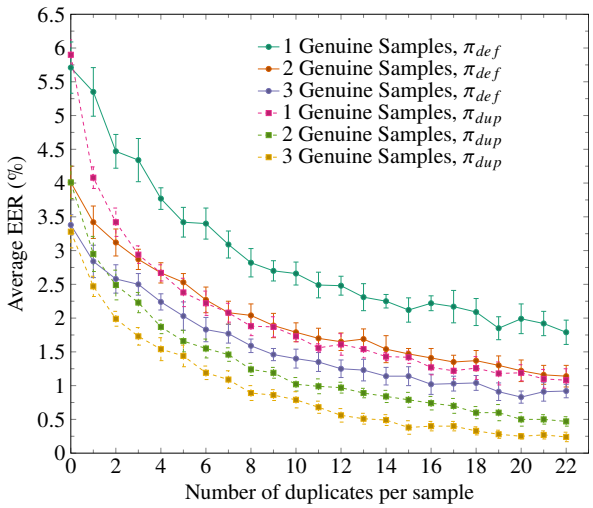


Figura 5.6: Average EER achieved using GPDS-300 dataset with the proposed method.

Figure 5.7 shows the performance achieved in MCYT-75 dataset for each number of duplicates. As observed in GPDS, the number of duplicates is inversely proportional to the the average EER in MCYT-75. Despite the average parameter vector π_{dup} was optimized using a subet of GPDS-960, it was able to model the writer variability present in MCYT-75. Since GPDS and MCYT-75 datasets present high-sparsity signature clusters in the feature space, it is expected that the proposed method generates high-quality duplicates for MCYT-75 as well. Consequently, when these duplicates are used for training, they increase the performance of the signature verification system. This behavior illustrates the generalization capability of the proposed method.

Figure 5.8 shows the performance achieved in CEDAR dataset for each number of duplicates. As in previous datasets, the greater the number of duplicates, the smaller the average EER. However, this trend is more subtle for CEDAR dataset. As previously exposed in Section 5.1, the duplicates for CEDAR dataset are less realistic than for other datasets. Consequently, these duplicates provide less information to distinguish between genuine signatures and skilled forgeries. Therefore, this negatively affect the performance of the verification system. It strengths the hypothesis that is hard to model the writer variability of the CEDAR dataset due the difference between its intrapersonal variability and the optimization dataset. Nevertheless, in terms of performance, the optimized parameter vector π_{dup} was able to better represent the writer variability present in CEDAR dataset than π_{def}.

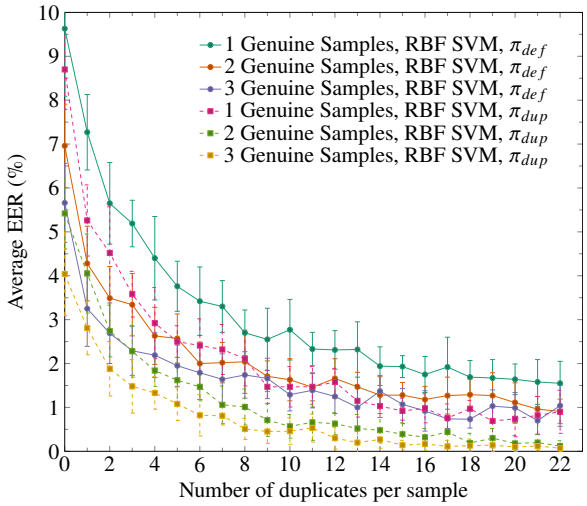


Figura 5.7: Average EER achieved using MCYT-75 dataset with the proposed method.

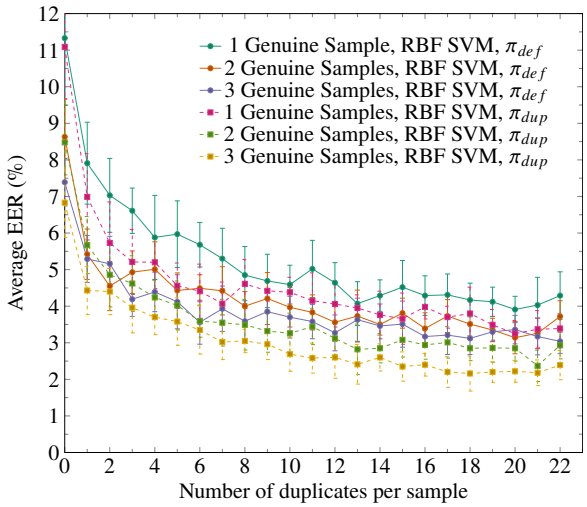


Figura 5.8: Average EER achieved using CEDAR dataset with the proposed method.

5.3 PERFORMANCE OF THE VERIFICATION SYSTEM USING THE GAUSSIAN FILTER

As explained before, the main hypothesis of this work is that the intrapersonal variability observed in the image space induces an intrapersonal variability in the feature space as well. Therefore, it enables the quality assessment of the duplicates in the feature space. To show the proof of this concept, the Gaussian filter [85] also is used to generate duplicates in the feature space. It includes new samples in the feature space by perturbing the genuine feature vectors [92].

A experimental protocol similar to the adopted in Section 5.2 was adopted here. Instead of using the duplicator to generate duplicates, the Gaussian filter was used to generate them with the average parameter vector π_{gauss}. Subsequently, they were used to train the SVM classifiers. As for the Duplicator, only the GPDS-300 was used to assess the False Acceptance Rate of random forgeries (FAR_{random}) with the Gaussian filter. Figure 5.9 presents the FAR_{random} for GPDS-300 dataset. As can be observed, the system trained with duplicates generated by the Gaussian filter achieved even lower FARs than the one trained with the duplicator's duplicates.

In both cases, the signature verification system almost does not have problems with random forgeries.

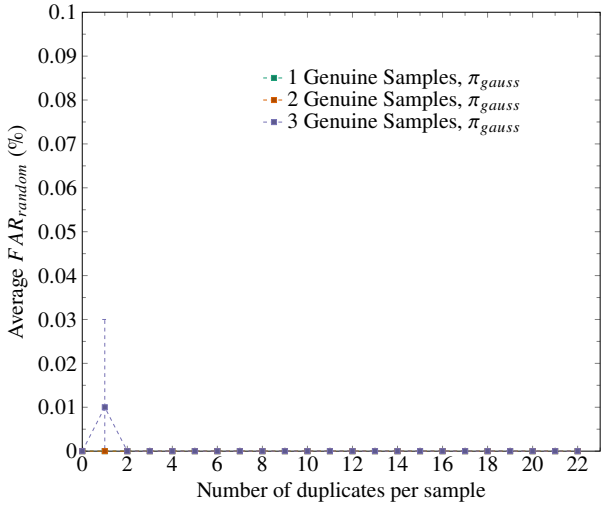
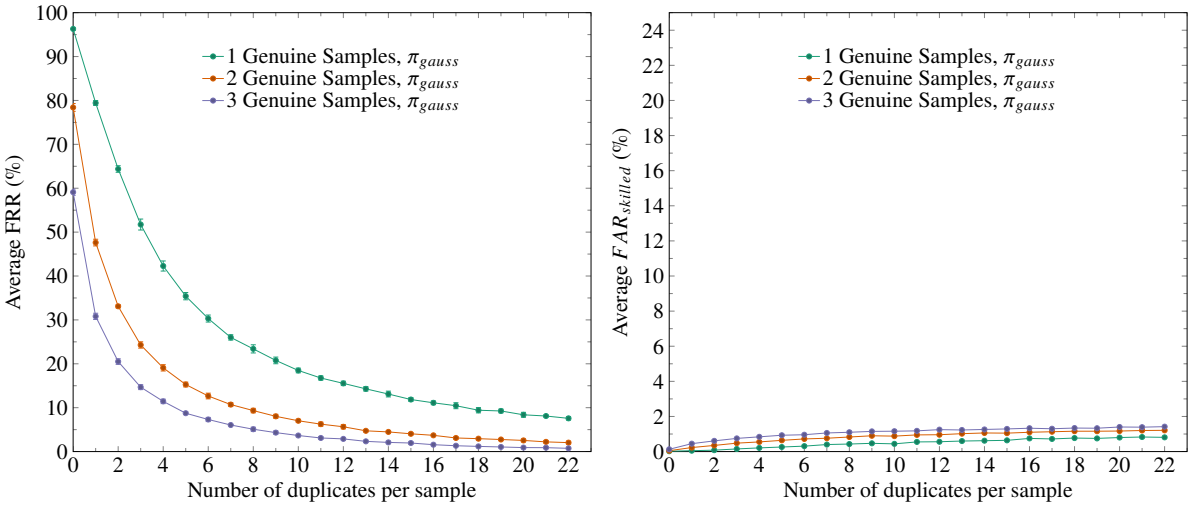


Figura 5.9: Average FAR_{random} achieved using GPDS-300 dataset with the Gaussian filter.

Figure 5.10 shows the average False Rejection Rates (FRR) and average False Acceptance Rates for skilled forgeries ($FAR_{skilled}$). The FRR and $FAR_{skilled}$ present the same trend of the duplicates generated by the duplicator with the optimized parameter vector (Section 5.2). For the three datasets, the duplicates generated by the Gaussian filter achieved lower $FAR_{skilled}$s than did the duplicates generated by duplicator. It may indicate that it is easier for the Gaussian filter to generate duplicates with a small intrapersonal variability than duplicator.

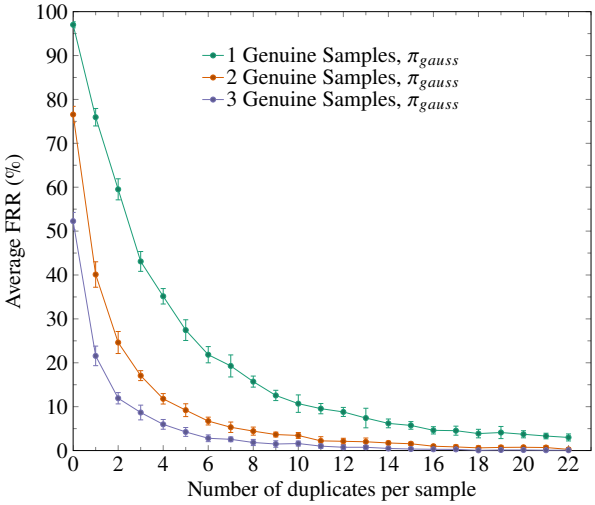
Figures 5.11 presents the average EERs achieved in GPDS-300, MCYT-75, and CEDAR datasets, respectively. Except for the CEDAR dataset, the average EERs achieved by the classifiers are similar to those reported in Section 5.2. For the CEDAR dataset, the duplicates generated by the Gaussian filter achieved a better performance than did the duplicates generated by the duplicator. Even using the Gaussian filter, the performance achieved in the CEDAR dataset is still inferior to other datasets. It strengthen the hypothesis that is harder to generate duplicates for writers with small intrapersonal variabilities than with great ones.

As observed in Figure 5.1, the sigma interval controls the intensity of the transformation applied in the feature vectors. If the sigma interval is great, it will provide synthetic feature vectors that are far from the original feature vectors. Therefore, the synthetic feature vectors will not resemble the original feature vectors anymore. As consequence, the average ERR will be higher for the synthetic samples using a great interval than using a small one [92].

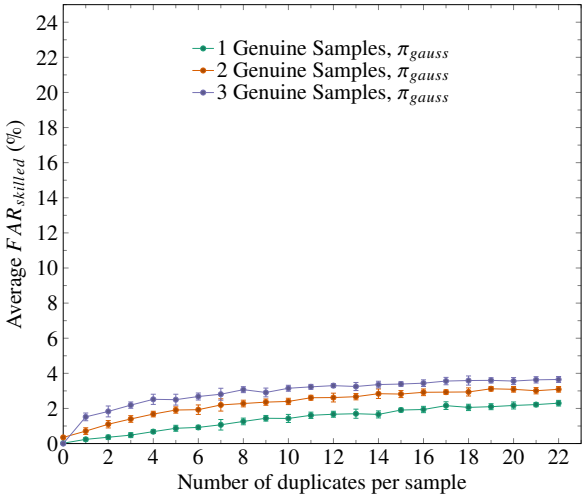


(a) GPDS-300: FRR

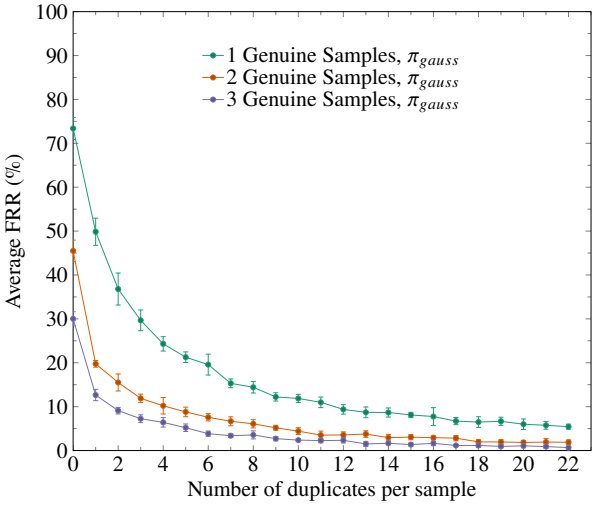
(b) GPDS-300: $FAR_{skilled}$



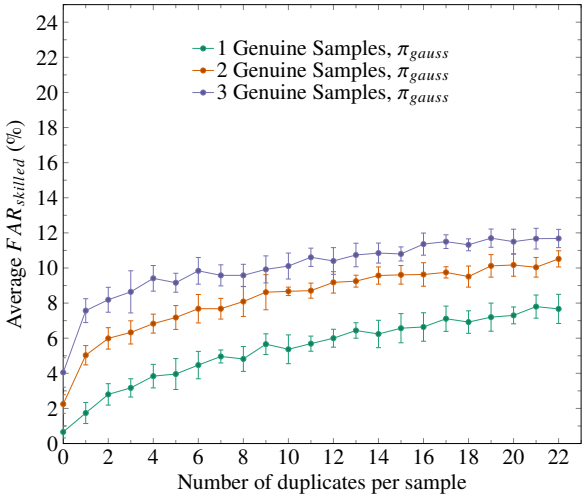
(c) MCYT-75: FRR



(d) MCYT-75: $FAR_{skilled}$

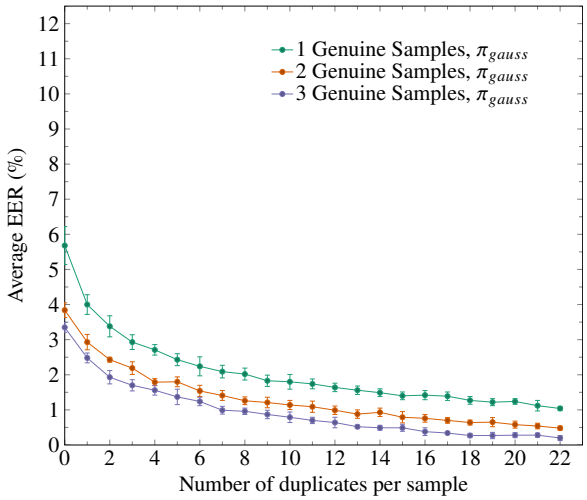


(e) CEDAR: FRR

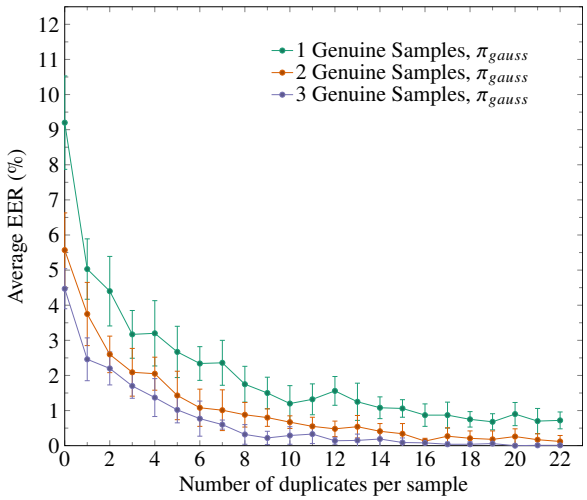


(f) CEDAR: $FAR_{skilled}$

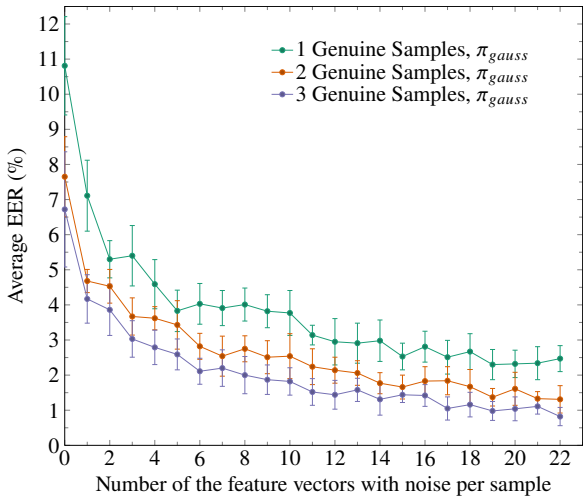
Figura 5.10: Average False Acceptance Rates and Average False Rejection Rates achieved with the Gaussian filter.



(a) GPDS-300



(b) MCYT-75



(c) CEDAR

Figura 5.11: Average EER achieved using duplicates generated by the Gaussian filter.

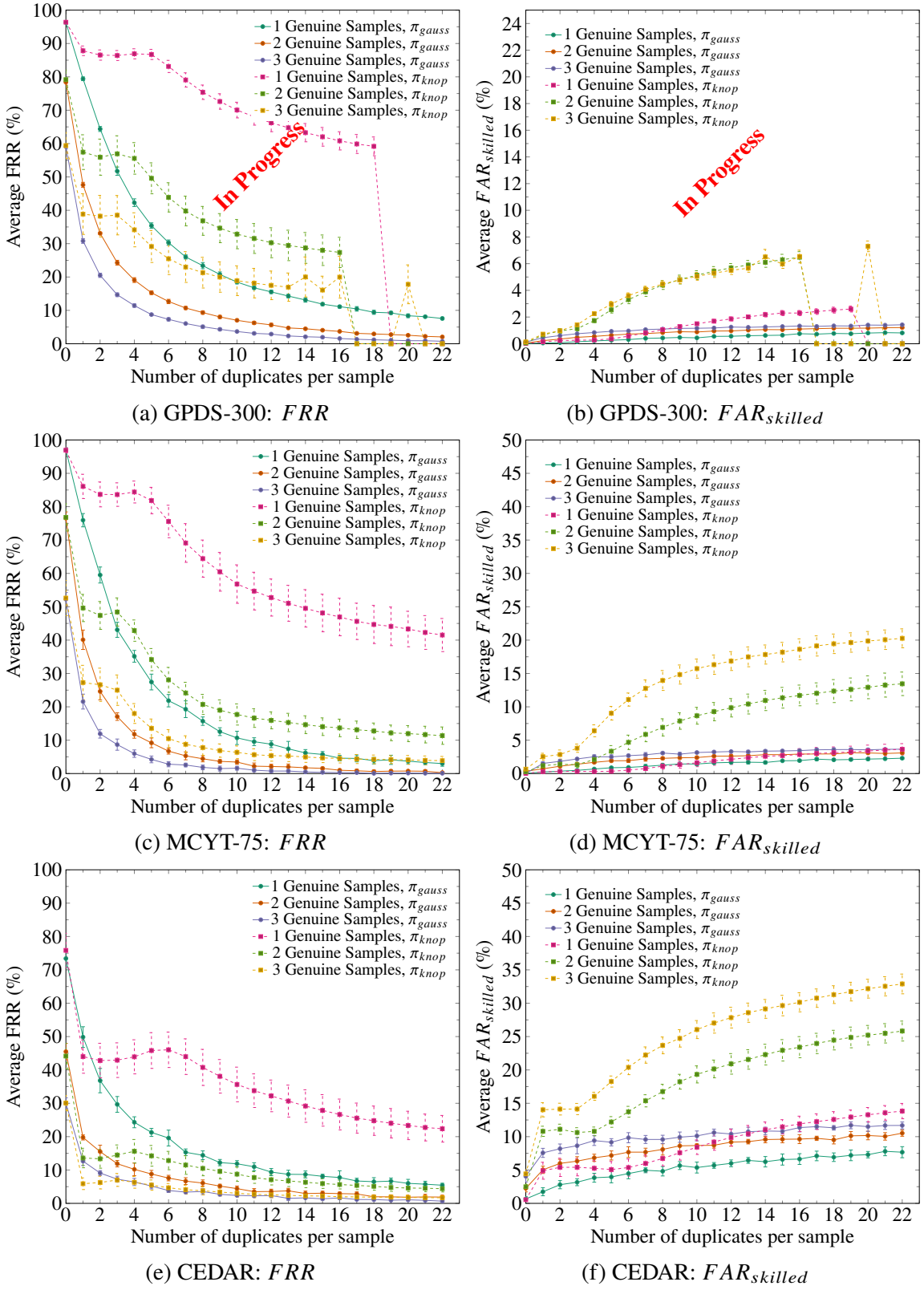
5.4 PERFORMANCE OF THE VERIFICATION SYSTEM USING THE VARIATION OF THE KNOP'S METHOD

A experimental protocol similar to the adopted in Section 5.3 was adopted here. Instead of using the Gaussian filter to generate duplicates, a variation of the Knop's method [77] was used to generate them with the average parameter vector π_{knop}. Subsequently, they were used to train the SVM classifiers. Since the variation of the Knop's method generates duplicates surrounding the genuine signatures in the feature space, it is expected that this method can cover a greater region of the feature space than the Gaussian filter. Consequently, the variation of the Knop's method may represent the writer's signatures better than the Gaussian filter does.

False Rejection Rate (FRR) and the False Acceptance Rate for skilled forgeries ($FAR_{skilled}$) were analyzed. Figure 5.12 presents the FRR and $FAR_{skilled}$ achieved on the three datasets for the Gaussian filter and the variation of the Knop's method. As can be observed, the information provided by the duplicates of the variation of the Knop's method is not so good as the Gaussian filter's duplicates. The duplicates generated by both methods change the balance between the FRR and $FAR_{skilled}$. When new duplicates are used for training, the FRR decreases and the $FAR_{skilled}$ increases. It means that the duplicates provides more information about the genuine signatures. On the other hand, this information also increases the confusion between the genuine signatures and skilled forgeries.

Figure 5.13 presents the average Equal Error Rate for each number of duplicates achieved on each of the three datasets: GPDS-300, MCYT-75, and CEDAR. As can be observed, the duplicates generated by the Gaussian filter achieved lower average EERs than did the ones generated by the variation of the Knop's method. For GPDS-300 dataset, when the number of the duplicates used for training increases, the EER increases well. It is more evident when we use 1 or 2 genuine samples per writer for training. On the other hand, the duplicates generated by the variation of the Knop's method help to decrease the average EER in MCYT-75 and CEDAR datasets.

Figure 5.14 shows the 3 examples of issues that the signature augmentation methods can present in a two dimensional feature space. In a 2D space, the circles and ellipses represent the regions that the duplicates can cover in the feature space. As can be observed in Figure Figure 5.14a, the Gaussian filter does not cover so well the feature space. The Gaussian filter generates duplicates that are aligned in the feature space. Since the Gaussian filter uses a uniform distribution to select the σ value, the duplicates have the same probability of being close to the genuine signature or far of it. Therefore, even generating few duplicates, the Gaussian filter can generate some duplicates that are close to the genuine samples and another ones that far from them. With the duplicates covering a restrict region of the feature space, it is possible to define a clearer boundary between genuine signatures and skilled forgeries. Despite of that, some skilled forgeries can be located close to the duplicates.



(a) GPDS-300: FRR

(b) GPDS-300: $FAR_{skilled}$

(c) MCYT-75: FRR

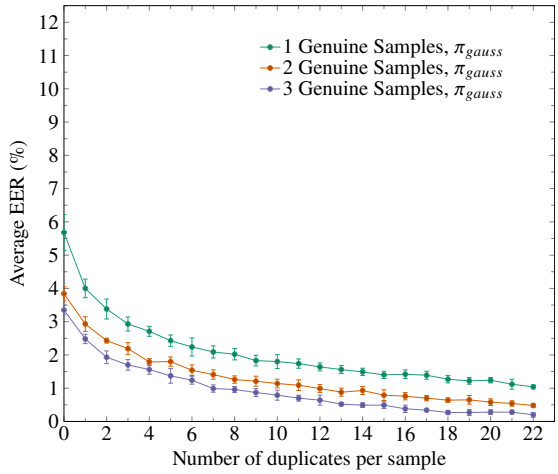
(d) MCYT-75: $FAR_{skilled}$

(e) CEDAR: FRR

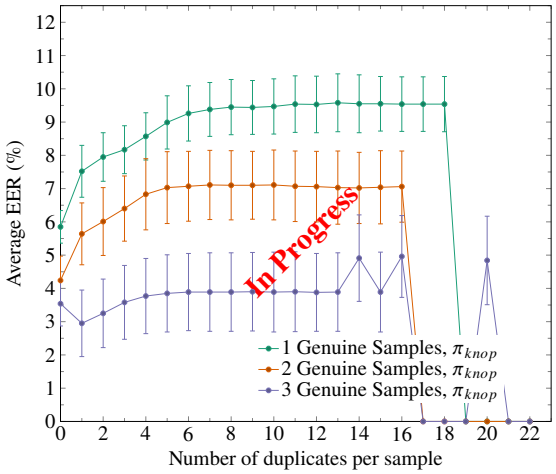
(f) CEDAR: $FAR_{skilled}$

Figura 5.12: Average FRR and Average $FAR_{skilled}$ achieved with the Gaussian filter and the variation of Knop's method.

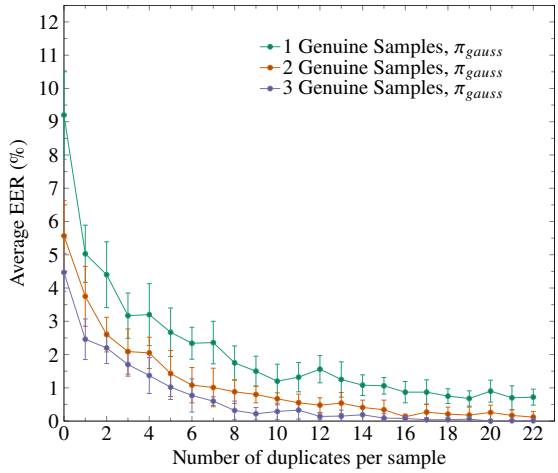
The variation of the Knop's method assumes that the data has a Gaussian distribution in the feature space [77]. Furthermore, it tries to fill a set of hyperspheres with duplicates in the



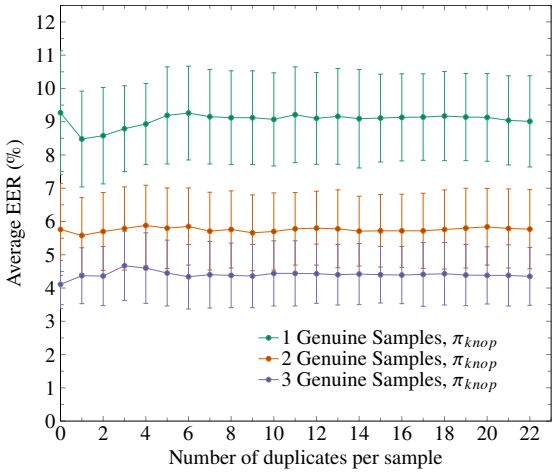
(a) GPDS-300: Gaussian filter



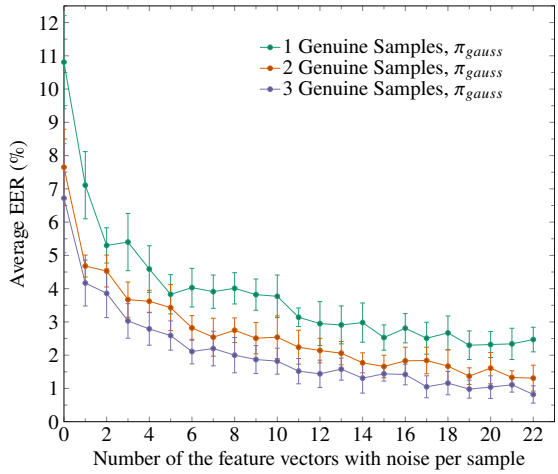
(b) GPDS-300: variation of the Knop's method



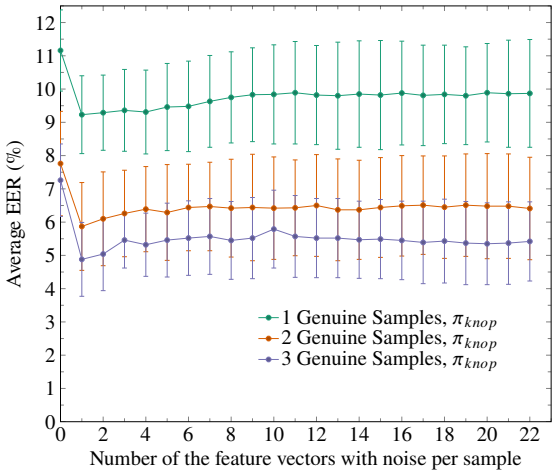
(c) MCYT-75: Gaussian filter



(d) MCYT-75: variation of the Knop's method



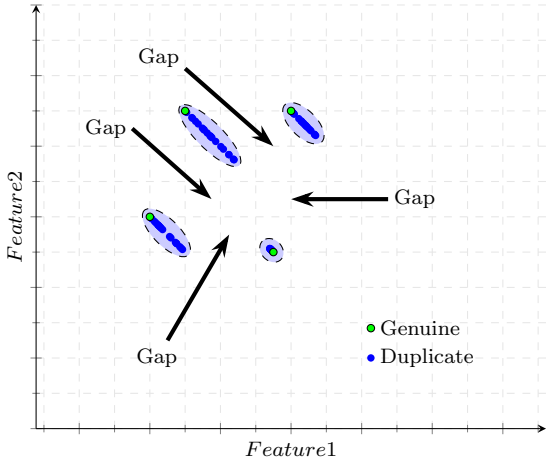
(e) CEDAR: Gaussian filter



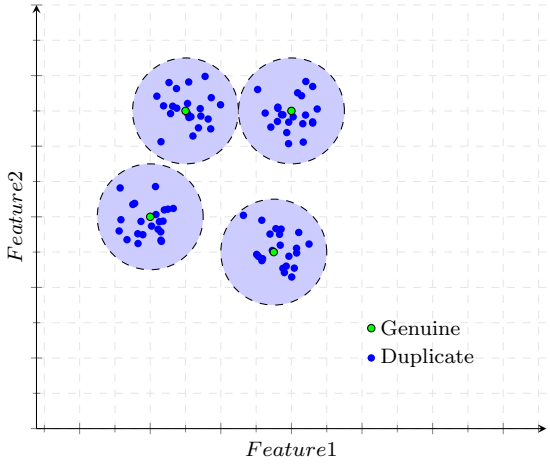
(f) CEDAR: variation of the Knop's method

Figura 5.13: Average EER achieved using duplicates generated by the Feature Space Augmentation methods.

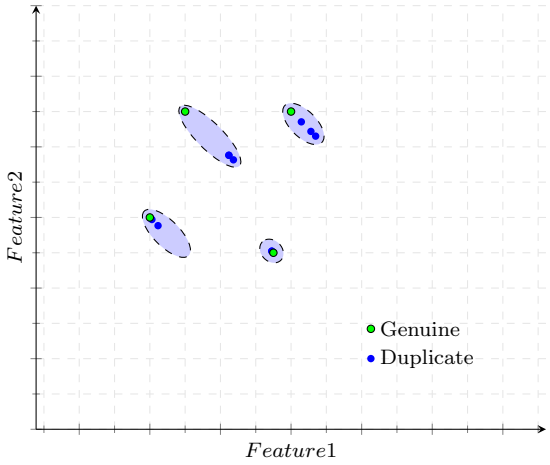
feature space. Since the distribution of the genuine signatures can assume several forms, these hyperspheres may be not enough to cover all the regions of the genuine signatures (Figure 5.14b).



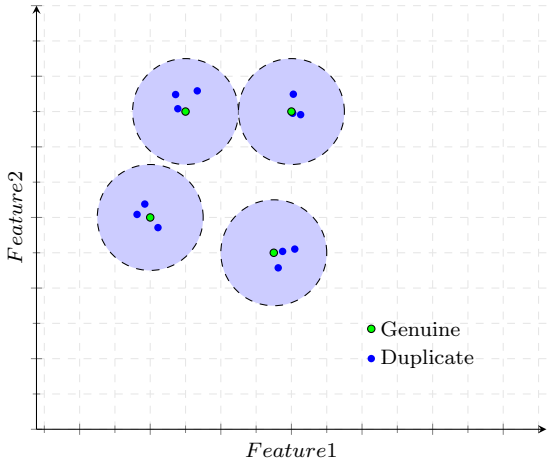
(a) Gaussin filter: gaps in the coverage region



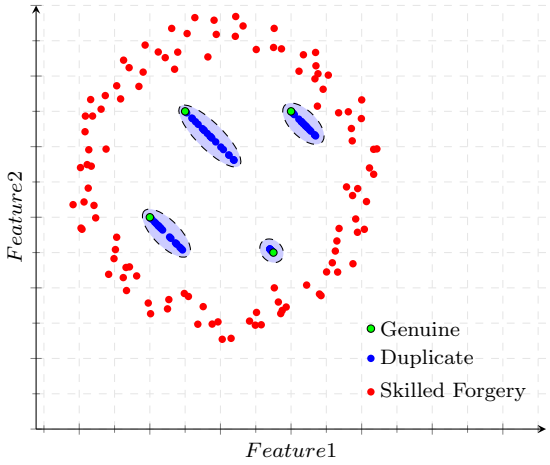
(b) Variation of Knop's method: gaps in the coverage region



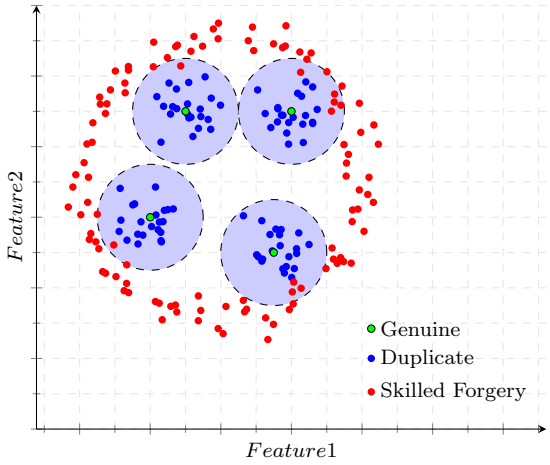
(c) Gaussian filter: some duplicates can be too close to the genuine signatures when there are few duplicates



(d) Variation of Knop's method: duplicates are too close to the genuine signatures when there are few duplicates



(e) Gaussian filter: some skilled forgeries can be considered as genuine signatures



(f) Variation of Knop's method: skilled forgeries can be considered as genuine signatures

Figura 5.14: Examples of issues that the signature augmentation methods can present in a 2D feature space.

Moreover, the genuine samples are used as the mean point of a Gaussian distribution. Therefore, there is a greater probability of the duplicates being generated close to the mean point than far from it. Consequently, it can create duplicates that are almost like the genuine signatures in the feature space (Figure 5.14d). Therefore, these duplicates may not provide enough information to improve the description of the genuine signatures. This behavior can be observed when few duplicates are generated (Figure 5.12 and Figure 5.14b). This particular behavior may explain the small $|\Delta|_{avg}$ using the three datasets (Table 5.2).

Furthermore, the hyperspheres generated by the variation of the Knop's method can cover regions that are out of the distribution of the genuine signatures. Particularly, when these hyperspheres cover the boundaries between the genuine signatures and the skilled forgeries simultaneously, the skilled forgeries can be considered as genuine signatures (Figure 5.14f).

Tables 5.4, 5.5, and 5.6 summarize the average EERs achieved in each experiment and in some state-of-the-art researches [21] [56] [147] in GPDS, MCYT, and CEDAR datasets, respectively. While Hafemann et al. [56] used up to 12 genuine signatures in their signature verification system, Zois et al. [147] used 10 and 12 genuine signatures. It is important to highlight that the results presented in [21] and [147] use other experimental protocols and signature features. Therefore, it is not possible to compare directly the results. As can be observed, the proposed method achieved state-of-the-art results using up to 3 genuine signatures. Despite the writer variability being modeled using GPDS dataset, the proposed method was able to generate more realistic duplicates for MCYT-75 and CEDAR datasets as well. Moreover, it was able to improve the quality of duplicates in the image (duplicator) and the feature space (Gaussian filter). These duplicates provide additional information about the writers, enabling the verification system distinguish one from another.

Besides the variation of the Knop's method achieved the lowest performance among the offline signature augmentation techniques, it achieved better results than not using duplicates (Baseline) in MCYT-75 and CEDAR datasets. As can be observed, the duplicates generated by this method also provide additional information about the genuine signatures. On the other hand, they also increase the confusion between the genuine samples and the skilled forgeries.

Tabela 5.4: Summary of the experimental results using GPDS dataset, 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**	**#W**	**#S**	**#D**	**Performance (%)**		
						EER	**FRR**	$FAR_{skilled}$
Diaz et al., 2017 [21]	LDerivP	SVM	300	2	20	21.63	-	-
				5	20	17.19	-	-
				8	20	14.58	-	-
Hafemann et al., 2017 [56]	SigNet-F	SVM	300	5	0	2.42	-	-
				12	0	1.69	-	-
Zois et al. 2019 [147]	KSVD/ OMP (F_3)	SVM	300	12	0	0.70	-	-
Baseline (Without Duplicates)	SigNet-F	SVM	300	1	0	5.71	96.31	0.02
				2	0	4.01	78.30	0.06
				3	0	3.38	52.83	0.22
Duplicator π_{def}	SigNet-F	SVM	300	1	22	1.79	9.50	2.05
				2	22	1.14	2.25	3.96
				3	22	0.92	0.88	5.40
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{dup}	SigNet-F	SVM	300	1	22	1.08	5.02	**1.54**
				2	22	0.47	1.15	2.30
				3	22	**0.24**	**0.17**	4.42
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{gauss}	SigNet-F	SVM	300	1	22	1.04	7.57	**0.81**
				2	22	0.48	2.04	1.21
				3	22	**0.20**	**0.74**	1.42
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{knop}	SigNet-F	SVM	300	1	22			
				2	22			
				3	22			

Tabela 5.5: Summary of the experimental results using MCYT-75 dataset, 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**	**#W**	**#S**	**#D**	**Performance (%)**		
						EER	**FRR**	$\boldsymbol{FAR_{skilled}}$
Diaz et al., 2017 [21]	LDerivP	SVM	75	2	20	16.06	-	-
				5	20	11.90	-	-
				8	20	9.12	-	-
Hafemann et al., 2017 [56]	SigNet-F	SVM	75	5	0	3.70	-	-
				10	0	3.00	-	-
Zois et al. 2019 [147]	KSVD/ OMP (F_3)	SVM	75	10	0	1.37	-	-
Baseline (Without Duplicates)	SigNet-F	SVM	75	1	0	9.63	96.61	0.00
				2	0	6.96	76.75	0.28
				3	0	5.66	51.84	0.61
Duplicator π_{def}	SigNet-F	SVM	75	1	22	1.55	2.40	7.94
				2	22	0.91	0.35	13.89
				3	22	1.04	0.13	17.08
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{dup}	SigNet-F	SVM	75	1	22	0.90	1.57	**5.33**
				2	22	0.12	0.29	8.66
				3	22	**0.07**	**0.03**	9.74
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{gauss}	SigNet-F	SVM	75	1	22	0.72	2.99	**2.30**
				2	22	0.12	0.27	3.09
				3	22	**0.01**	**0.08**	3.65
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{knop}	SigNet-F	SVM	75	1	22	9.01	41.52	**3.64**
				2	22	5.77	11.36	13.45
				3	22	**4.35**	**3.87**	20.27

Tabela 5.6: Summary of the experimental results using CEDAR dataset, 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**	**#W**	**#S**	**#D**	**Performance (%)**		
						EER	**FRR**	$FAR_{skilled}$
Hafemann et al. [56]	SigNet-F	SVM	55	4	0	5.92	-	-
				8	0	4.77	-	-
				12	0	4.63	-	-
Zois et al. 2019 [147]	KSVD/ OMP (F_3)	SVM	55	10	0	0.79	-	-
Baseline (Without Duplicates)	SigNet-F	SVM	55	1	0	11.33	73.84	0.64
				2	0	8.63	43.91	2.35
				3	0	7.39	28.24	4.40
Duplicator π_{def}	SigNet-F	SVM	55	1	22	4.29	3.25	18.60
				2	22	3.72	1.09	24.72
				3	22	3.04	0.45	27.56
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{dup}	SigNet-F	SVM	55	1	22	3.39	6.76	**12.95**
				2	22	2.93	3.16	16.29
				3	22	**2.39**	**1.65**	18.16
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{gauss}	SigNet-F	SVM	55	1	22	2.47	5.44	**7.67**
				2	22	1.31	1.89	10.52
				3	22	**0.82**	**0.67**	11.68
Proposed Method 20 $\mathcal{D}_{\mathcal{L}}$ π_{knop}	SigNet-F	SVM	300	1	22	9.87	22.36	**13.83**
				2	22	6.41	4.40	25.83
				3	22	**5.42**	**1.69**	32.89

5.5 DUPLICATOR VS GAUSSIAN FILTER WITH LESS DISCRIMINANT FEATURES

In the previous experiments, the duplicator and the Gaussian filter were used with high discriminant features to increase the performance of a signature verification system. However, they were not tested with less discriminant features. Therefore, both offline signature augmentation approaches are tested with fewer discriminant features. Some representations were defined to show how the offline signature augmentation approaches are influenced by distinct levels of feature discrimination. They were defined randomly selecting 64, 128, 256, 512, 768, 1024, 1280, 1536, and 1792 elements of the original SigNet-F feature vectors. The complete feature vectors with 2048 elements were also used. These representations were used to analyze the quality of duplicates in the feature space considering the duplicator and the Gaussian filter (Section 5.5.1). Subsequently, these representations were used to assess the performance of the previously defined signature verification system (Section 5.5.2).

5.5.1 Quality of Duplicates

A similar experimental protocol to the proposed in Section 5.1 was used here. The average absolute Silhouette index $|\Delta|$ also was used to asses the quality of duplicates in the feature space. For convenience, the average absolute Silhouette index will be called only Silhouette index. The first 12 genuine signatures of each writer of GPDS $\mathcal{D}_{\mathcal{V}}$ were used to represent the genuine cluster in the feature domain.

Three clusters of 12 feature vectors were considered: one with genuine signatures, one with duplicates generated using the duplicator using the optimized parameter vector π_{dup}, and one with duplicates generated using the Gaussian filter using the optimized parameter vector π_{gauss}. The quality of duplicates was assessed calculating the $|\Delta|$ and the standard deviation between: the genuine cluster and the cluster of duplicates generated by duplicator, and the genuine cluster and the cluster of duplicates generated by the Gaussian filter. This process was repeated 10 times, and the average of the silhouette indices and the average of standard deviations were calculated. The whole process was repeated for each representation.

If the average Silhouette Index of the representation is similar to or lower than the complete feature vector, the parameter vector optimized with the complete feature vector can be reused. Otherwise, the parameter vector should be optimized for the specific representation. Table 5.7 presents the average silhouette indices $|\Delta|_{avg}$ and the respective standard deviations for each representation with the signature augmentation approaches. For duplicator, the greater the $|\Delta|_{avg}$, the greater the dimensionality representation. It may indicate that the duplicator can generate better duplicates for less discriminant features.

The Gaussian filter presents a subtle difference between the $|\Delta|_{avg}$s. Moreover, it presents higher $|\Delta|_{avg}$s and standard deviations than duplicator does. Due to that it is difficult to compare the quality of the duplicates generated by both signature augmentation approaches.

Tabela 5.7: Average Absolute Silhouette Index and Standard Deviation for Different Representations

Feature	$\|\Delta\|_{avg}$	
Vector Size	**Duplicator**	**Gaussian Filter**
64	0.030±0.031	0.103±0.070
128	0.027±0.029	0.097±0.187
256	0.031±0.024	0.123±0.052
512	0.032±0.037	0.105±0.075
768	0.036±0.031	0.109±0.134
1024	0.037±0.031	0.106±0.098
1280	0.040±0.037	0.118±0.039
1536	0.040±0.042	0.125±0.189
1792	0.041±0.040	0.131±0.145
2048	0.055±0.055	0.118±0.121

Therefore, the performance of a signature verification system was assessed when it used each kind of duplicate for training.

5.5.2 Performance of the Offline Signature Verification System

The signature verification system described in Chapter 4 was trained and tested to assess the influence of representation on two signature augmentation approaches: the duplicator and the Gaussian filter. For GPDS datasets, the subsets $\mathcal{D}_{\mathcal{V}}$ and $\mathcal{D}_{\mathcal{L}}$ were used to perform the experiments. The genuine signatures and random forgeries were randomly selected. In $\mathcal{D}_{\mathcal{V}}$, three genuine signatures per writer were used for training. For each genuine signature used for training, the signature augmentation technique was used to generate up to 22 duplicates for training. In $\mathcal{D}_{\mathcal{L}}$, the 14 genuine signatures of each writer was used as random forgeries. For each random forgery used for training, the signature augmentation technique was also used to generate up to 22 duplicates. The classifiers were tested using 10 genuine signatures, and 10 skilled forgeries from subset $\mathcal{D}_{\mathcal{V}}$. The training and testing were repeated 10 times, and the average EER and standard deviation were calculated. The whole process was repeated for each one of the predefined representations.

A similar procedure has been adopted for MCYT-75 and CEDAR dtasets. Since MCYT-75 does not have a development dataset, the 10 genuine signatures of the other 74 writers and the respective duplicates were used as random forgeries. Five genuine signatures and 15 skilled forgeries were used to test the classifiers. For CEDAR dataset, the 12 genuine signatures of the other 54 writers and the respective duplicates were used as random forgeries. Ten genuine signatures and 10 skilled forgeries were used for testing. For each dataset, Table 5.8 presents the number of signature samples and duplicates used for training, and testing.

Figure 5.15 shows the performance achieved in each dataset for different discrimination feature levels. While the left column (Figure 5.15ace) presents the duplicator's performance, the right column (Figure 5.15bdf) presents the Gaussian filter's performance. For both signature augmentation approaches, the small feature vectors are less discriminant than the large ones.

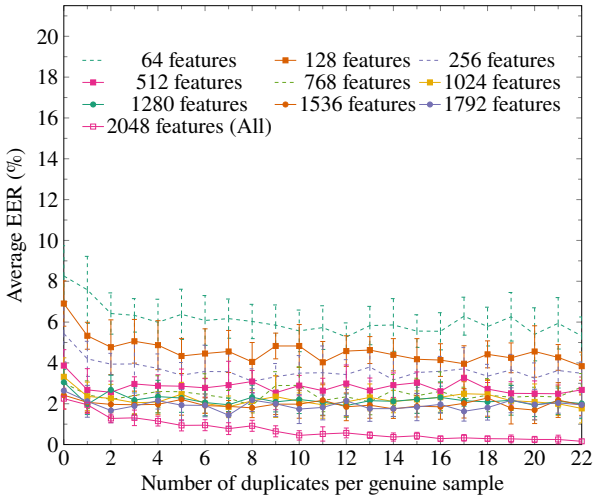
Tabela 5.8: Number of samples used for training and testing with different representations. The number of genuine signatures G, and skilled forgeries S are specified. The number of duplicates used for training also is specified.

Dataset	**Training subset**		**Testing subset**	
	G	**R**	**G**	**S**
GPDS 50$\mathcal{D}_\mathcal{V}$	3 + 3 × (d ∈ {0, ..., 22})	(14 × 581) + (14 × 581 × d)	10	10
MCYT-75	3 + 3 × (d ∈ {0, ..., 22})	(10 × 74) + (10 × 74 × d)	5	15
CEDAR	3 + 3 × (d ∈ {0, ..., 22})	(12 × 54) + (12 × 54 × d)	10	10

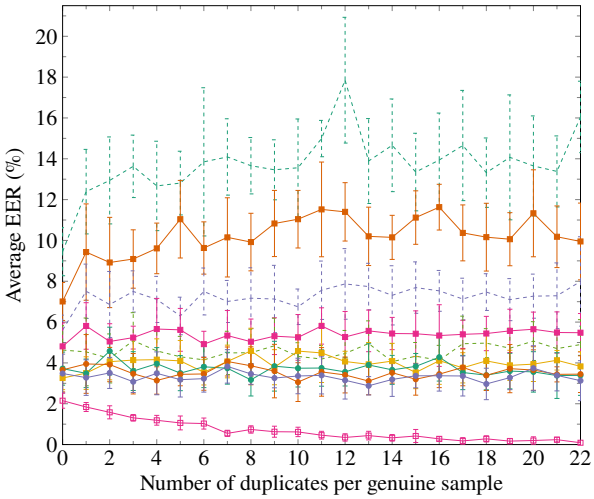
For the same signature augmentation approach, the representations followed a similar trend in the three different datasets. For the duplicator, the duplicates provide additional information about the signatures that helps to increase performance. Figures 5.15a, c, and e suggest that the contribution of the duplicates is more significant for lower discriminant levels than for greater ones. Since the duplicator uses a genuine signature as base to apply some distortions and generate a duplicate, it preserves some visual aspects of the genuine signature image. As consequence, some features are preserved in the feature space as well. Thus, they can help to characterize the duplicates as the original signatures of their writers.

When the Gaussian filter uses small feature vectors, the duplicates provide information that reduce the performance of the classifiers. For the feature vectors with 512 elements, the performance of the system starts to stabilize. It shows that the duplicates start to provide useful information when the discrimination level of the feature vectors increases, which improves the performance of the system. Even the Gaussian filter has a low computational cost and it is simpler to implement, Figures 5.15b, d, and f suggest that the duplicates generated using it do not help to increase the performance of the system with low and medium feature discrimination levels. The Gaussian filter relies on a single kind of transformation to distort the whole feature vector. It can modify the distribution of the features in such way, that they do not keep the information which characterizes the writer anymore [120]. Therefore, other techniques like duplicator may be more suitable to generate duplicates that can also be used with these kind of representations.

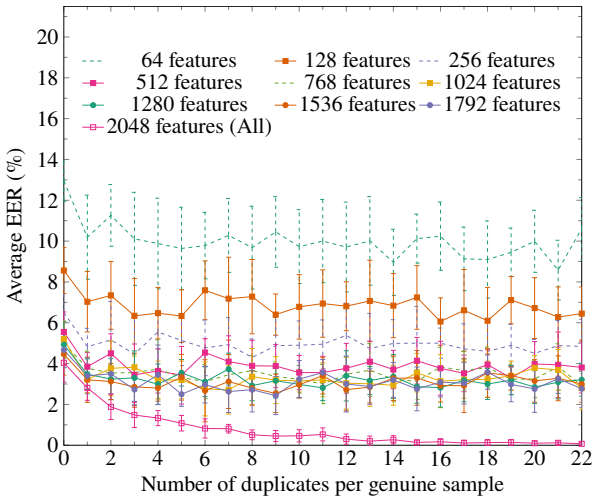
The Gaussian filter has a low computational cost and is simpler to implement than duplicator [85]. Since the Gaussian filter generates the duplicates directly in the feature space, it does not provide signature images that can be used by different feature extractors. While the duplicator is indicated to be used with low, some medium, and high discriminant features, the Gaussian filter is only indicated to be used with high discriminant features [92].



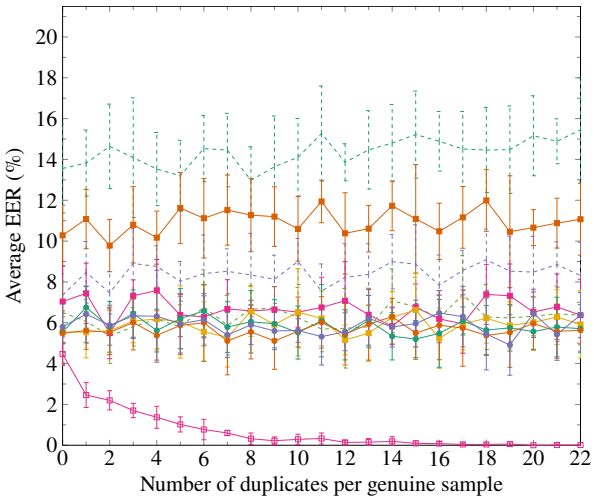
(a) GPDS 50$\mathcal{D}_\mathcal{V}$: Duplicator



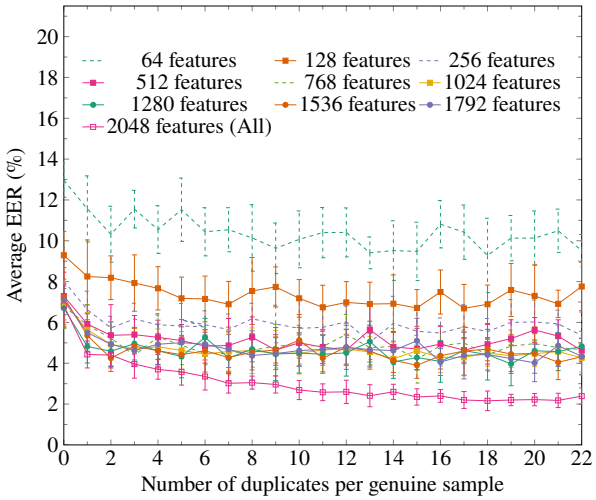
(b) GPDS 50$\mathcal{D}_\mathcal{V}$: Gaussian Filter



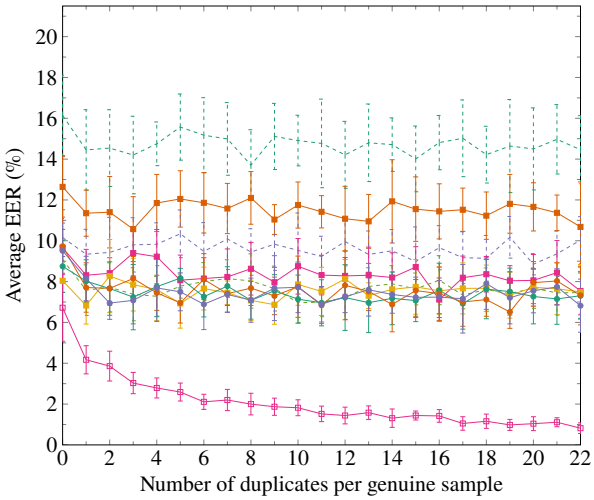
(c) MCYT-75: Duplicator



(d) MCYT-75: Gaussian Filter



(e) CEDAR: Duplicator



(f) CEDAR: Gaussian Filter

Figura 5.15: Average Equal Error Rate achieved with different feature representations, 3 genuine samples per writer, and duplicates generated by the offline signature augmentation approaches.

6 CONCLUSION

In this work, a method to automatically model the intrapersonal variability of writers to generate synthetic samples of offline handwritten signatures was proposed. The proposed method was used with two kinds of offline signature augmentation approaches: one in the image space (duplicator) [21] and another two in the feature space (Gaussian filter and the variation of the Knop's method). The advantages and limitations of these approaches were also explored. Moreover, the synthetic samples generated by these approaches can be used to train an automatic handwritten signature verification system. Besides improving the performance of the system, the method was able to model the most common writer variability traits to produce better synthetic samples than the method proposed by Diaz et al. (2017a) [21]. Furthermore, a new approach to validate the quality of synthetic samples with their features was proposed. The results support the hypothesis that the writer variability observed in the image space induces a writer variability in the feature space as well [92].

The proposed method achieved a performance comparable to the state of the art in GPDS-300, MCYT-75, and CEDAR datasets using up to 3 genuine signatures per writer. It achieved EERs close to zero in the MCYT-75 dataset. Since the parameter vector was optimized using a GPDS subset which has a writer variability close to the observed in MCYT-75, this behavior is expected. The proposed method also achieved low EERs in the CEDAR dataset. However, the optimization of six parameters of the duplicator may not be sufficient to reproduce the small writer variability observed in the CEDAR dataset. Therefore, the other duplicator's parameters can be optimized to generate more compact clusters in the feature space. The different distributions observed in the three datasets suggest that other transformations may be investigated to improve the performance in each one of them, specially in CEDAR dataset. Perhaps, some nonlinear transformations like the ones used in feature extractors [2] [95] should be investigated. The proposed method showed promising results for three distinct offline signature datasets based on the Latin alphabet. Therefore, the proposed method can be tested using signature datasets based on another alphabet systems [92].

According to the previous experiments, the offline signature augmentation approaches depends on the parameter vector used to model the writer variability. Moreover, the proposed method was able to optimize their parameter vectors. It can be tested with other offline signature augmentation approaches or even other parameter optimization problems. Despite the low complexity and its simple implementation, the Gaussian filter is only indicated to be used with high discriminant features. On the other hand, the duplicator is indicated to be used with low, some medium, and high discriminant features. Unlike the Gaussian filter, the duplicator provides a signature image that can be used by different feature descriptors. These synthetic signature

images can be used to create more robust offline handwritten signature datasets [39]. Furthermore, they can also be used to train more robust deep learning models [56] [58] [135] [92].

The main constraint of this work is the number of random forgeries needed to train the signature verification system. Therefore, an opposite approach can be used to generate random forgeries. Instead modeling the most common intrapersonal variability traits among a set of writers, the less common intrapersonal variability traits can be modeled to generate the random forgeries. It is expected that the model can describe the surrounding of the genuine signatures in the feature space. By definition, the genuine signatures and random forgeries are not visually alike. Therefore, other transformations can also be explored to generate the random forgeries.

In this work, the Gaussian filter is used to increase the number of samples in the feature space using as input a feature vector. However, it uses the same σ for all elements of the same feature vector. As consequence, it limits the diversity of the synthetic samples. To increase this diversity, the Gaussian filter could use a different σ for each element of the feature vector. Perhaps, it can also be applied to generate random forgeries directly in the feature space.

As previous exposed, the variation of the Knop's method uses a Gaussian distribution to generate the duplicates in the feature space. Consequently, it has a higher probability in creating duplicates too similar to the genuine signatures than creating different ones. Therefore, these duplicates can provide less information about the genuine signatures. This behavior is particularly evident when a small number of duplicates is generated for each genuine signature. A possible solution is use a uniform distribution to generate the duplicates surrounding the genuine signatures in the feature space.

When the genuine signatures are close to the skilled forgeries and the duplicates are surrounding the genuine samples, the variation of the Knop's method can increase the confusion between the genuine signatures and the skilled forgeries. Instead of generating the duplicates surrounding each genuine signature, they can be generated surrounding the centroid of the cluster of genuine signatures.

REFERÊNCIAS

[1] M. N. Abdi and M. Khemakhem. A model-based approach to offline text-independent arabic writer identification and verification. *Pattern Recognition*, 48:1890–1903, 2015.

[2] P.F. Alcantarilla, A. Bartoli, and A.J. Davison. Kaze features. In *European Conference on Computer Vision*, pages 214–227, 2012.

[3] M. J. Allen. *Foundations of Forensic Document Analysis: Theory and Practice*. Wiley Blackwell, 2016.

[4] L. Batista, E. Granger, and R. Sabourin. Dynamic selection of generative-discriminative ensembles for off-line signature verification. *Pattern Recognition*, 45(4):1326–1340, 2012.

[5] J. R. Bell. Algorithm 334 normal random deviates [g5]. *Communications of ACM*, 11(7):498, July 1968.

[6] C. Bergamini, L. S. Oliveira, A. L. Koerich, and R. Sabourin. Combining different biometric traits with one-class classification. *Signal Processing*, 89:2117–2127, 2009.

[7] N. Berry. *The Story of Writing*. Children's Book Trust, 2004.

[8] D. Bertolini, L. S. Oliveira, E. Justino, and R. Sabourin. Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers. *Pattern Recognition*, 43:387–396, 2010.

[9] D. Bertolini, L. S. Oliveira, E. Justino, and R. Sabourin. Texture-based descriptors for writer identification and verification. *Expert Systems with Applications*, 40:2069–2080, 2013.

[10] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] W. Bouamra, Chawki Djeddi, B. Nini, M. Diaz, and I. Siddiqi. Towards the design of an offline signature verifier based on a small number of genuine samples for training. *Expert Systems With Applications*, 107:182–195, 2018.

[12] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. 29(2):610–611, June 1958.

[13] F. Chollet. *Deep Learning with Python*. Manning, 2018.

[14] E. Clayton. *The Golden Thread: The Story of Writing*. Counterpoint Berkeley, 2013.

[15] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[16] S. A. Daramola and T. S. Ibiyemi. Offline signature recognition. *International Journal of Computer Applications*, 10(2):17–22, November 2010.

[17] A. Das, M. A. Ferrer, U. Pal, S. Pal, M. Diaz, and M. Blumenstein. Multi-script versus single-script scenarios in automatic off-line signature verification. *IET Biometrics*, 5(4):305–313, July 2016.

[18] T. DeVries and G. W. Taylor. Dataset augmentation in feature space. In *International Conference on Learning Representations*, pages 1–12, 2017.

[19] M. Diaz, S. Chanda, M. A. Ferrer, C. Kr. Banerjee, A. Majumdar, C. Carmona-Duarte, P. Acharya, and U. Pal. Multiple generation of bengali static signatures. In *15^{th} International Conference on Frontiers in Handwriting Recognition*, October 2016.

[20] M. Diaz and M. A. Ferrer. Assessing the common authorship of a set of questioned signature images. In *2017 International Carnahan Conference on Security Technology*, October 2017.

[21] M. Diaz, M. A. Ferrer, G. S. Eskander, and R. Sabourin. Generation of duplicated off-line signature images for verification systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5):961–964, April 2017.

[22] M. Diaz, M. A. Ferrer, D. Impedovo, M. I. Malik, G. Pirlo, and R. Plamondon. A perspective analysis of handwritten signature technology. *ACM Computing Surveys*, 51(6):117, 2019.

[23] M. Diaz, M. A. Ferrer, and R. Sabourin. Approaching the intra-class variability in multi-script static signature evaluation. In *23^{rd} International Conference on Pattern Recognition*, pages 1147–1152, December 2016.

[24] 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 *14^{th} International Conference on Frontiers in Handwriting Recognition*, pages 482–487, 2014.

[25] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo. A multi-expert signature verification system for bankcheck processing. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(5):827–844, 1997.

[26] D. Diringer. *The Book Before Printing: Ancient, Medieval and Oriental*. Dover Publications, 1982.

[27] A. R. Disney. *A History of Portugal and the Portuguese Empire: From Beginnings to 1807 Volume 1*. Cambridge University Press, 2009.

[28] M. Djioua and R. Plamondon. A new algorithm and system for characterization of handwriting strokes with delta-lognormal parameters. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 31(11):2060–2072, November 2009.

[29] E. Domany, J.L. Van Hemmen, and K. Schulten. *Models of Neural Networks III: Association, Generalization and Representation*. Springer, 1996.

[30] J.P. Drouhard, R. Sabourin, and M. Godbout. A neural network approach to off-line signature verification using directional pdf. *Pattern Recognition*, 29(3):415–424, 1996.

[31] A. G. Dyer, B. Found, and D. Rogers. Visual attention and expertise for forensic signature analysis. *Journal of Forensic Sciences*, 51(6):1397–1404, November 2006.

[32] A. G. Dyer, B. Found, and D. Rogers. An insight into forensic document examiner expertise for discriminating between forged and disguised signatures. *Journal of Forensic Sciences*, 53(5):1154–1159, 2008.

[33] G. S. Eskander, R. Sabourin, and E. Granger. Hybrid writer-independent-writer-dependent offline signature verification system. *IET Biometrics*, 2(4):169–181, 2013.

[34] 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. *Applied Soft Computing*, 91:106242, 2020.

[35] 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. *IEEE Proceedings in Vision, Image and Signal Processing*, 149(2):85–90, April 2002.

[36] M. Faundez-Zanuy, J. Fierrez, M. A. Ferrer, M. Diaz, R. Tolosana, and R. Plamondon. Handwriting biometrics: Applications and future trends in e-security and e-health. *Cognitive Computation*, 12:940–953, August 2020.

[37] A. Ferreira and G. Giraldi. Convolutional neural network approaches to granite tiles classification. *Expert Systems with Applications*, 84:1–11, 2017.

[38] M. A. Ferrer, S. Chanda, M. Diaz, C. K. Banerjee, A. Majumdar, C. Carmona-Duarte, P. Acharya, and U. Pal. Static and dynamic synthesis of bengali and devanagari signatures. *IEEE Transactions on Cybernetics*, 48(10):2896–2907, October 2018.

[39] M. A. Ferrer, M. Diaz, C. Carmona-Duarte, and A. Morales. A behavioral handwriting model for static and dynamic synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1041–1053, June 2016.

[40] M. A. Ferrer, M. Diaz-Cabrera, and A. Morales. Robustness of offline signature verification based on gray level features. *IEEE Transactions on Information Forensics and Security*, 7(3):966–977, June 2012.

[41] M. A. Ferrer, M. Diaz-Cabrera, and A. Morales. Synthetic off-line signature image generation. In *2013 International Conference on Biometrics*, pages 1–7, June 2013.

[42] M. A. Ferrer, M. Diaz-Cabrera, and A. Morales. Static signature synthesis: A neuromotor inspired approach for biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):667–680, March 2015.

[43] 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 *2013 47th International Carnahan Conference on Security Technology*, October 2013.

[44] J. Fierrez-Aguilar, N. Alonso-Hermira, G. Moreno-Marquez, and J. Ortega-Garcia. An off-line signature verification system based on fusion of local and global information. In *International Workshop on Biometric Authentication*, pages 295–306, 2004.

[45] S. R. Fischer. *A History of Writing*. Reaktion Books, 2001.

[46] K. B. Florey. *Script and Scribble: The Rise and Fall of Handwriting*. Melville House, 2009.

[47] C. Freitas, M. Morita, L. Oliveira, E. Justino, A. Yacoubi, E. Lethelier, F. Bortolozzi, and R. Sabourin. Bases de dados de cheques bancários brasileiros. In *XXVI Conferência Latinoamericana de Informatica*, 2000.

[48] E. Frias-Martinez, A. Sanchez, and J. Velez. Support vector machine versus multi-layer perceptrons for efficient off-line signature recognition. *Engineering Applications of Artificial Intelligence*, 19:693–704, March 2006.

[49] R. Gad, A. El-Sayed, N. El-Fishawy, and M. Zorkany. Multi-biometric systems: A state of the art survey and reasearch directions. *International Journal of Advanced Computer Science and Applications*, 6(6):128–138, 2015.

[50] J. Galballly, M. Martinez-Diaz, and J. Fierrez. Aging in biometrics: An experimental analysis on online signature. *PLOS One*, 8(7):1–17, July 2013.

[51] J. Georges. *Writing: The Story of Alphabets and Scripts*. New York: H.N. Abrams, 1992.

[52] R. C. Gonzalez and R. E. Woods. *Digital Image Processing 4th global edition*. Pearson, 2018.

[53] Y. Guerbai, Y. Chibani, and B. Hadjadji. The effective use of the one-class svm classifier for handwritten signature verification based on writer-independent parameters. *Pattern Recognition*, 48(1):103–113, 2015.

[54] L. G. Hafemann, R. Sabourin, and L. S. Oliveira. Analyzing features learned for offline signature verification using deep cnns. In *23rd International Conference on Pattern Recognition*, pages 2989–2994, December 2016.

[55] L. G. Hafemann, R. Sabourin, and L. S. Oliveira. Writer-independent feature learning for offline signature verification using deep convolutional neural networks. In *2016 International Joint Conference on Neural Networks*, pages 2576–2583, 2016.

[56] L. G. Hafemann, R. Sabourin, and L. S. Oliveira. Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition*, 70:163–176, 2017.

[57] L. G. Hafemann, R. Sabourin, and L. S. Oliveira. Offline handwritten signature verification - literature review. In *Seventh International Conference on Image Processing Theory, Tools and Applications*, pages 1–8, 2017.

[58] L. G. Hafemann, R. Sabourin, and L. S. Oliveira. Fixed-sized representation learning from offline handwritten signatures of different sizes. *International Journal of Document Analysis and Recognition*, pages 1–14, 2018.

[59] L. G. Hafemann, R. Sabourin, and L. S. Oliveira. Characterizing and evaluating adversarial examples for offline handwritten signature verification. *IEEE Transactions on Information Forensics and Security*, 14(8):2153–2166, August 2019.

[60] A. Hamadene and Y. Chibani. One-class writer-independent offline signature verification using feature dissimilarity thresholding. *IEEE Transactions on Information Forensics and Security*, 11(6):1226–1238, June 2016.

[61] R. K. Hanusiak, L. S. Oliveira, E. Justino, and R. Sabourin. Writer verification using texture-based features. *International Journal on Document Analysis and Recognition*, 15:213–226, 2012.

[62] R. M. Harralick and L. G. Shapiro. *Computer and Robot Vision Volume I*. Addison-Wesley Publishing Company, 1992.

[63] H. H. Harralson. *Developments in Handwriting and Signature Identification in the Digital Age*. Elsevier, 2013.

[64] D. Harris. *The Art of Calligraphy*. Dorling Kindersley, 1995.

[65] K. Huang and H. Yan. Off-line signature verification based on geomtric feature extraction and neural network classification. *Pattern Recognition*, 30(1):9–17, 1997.

[66] K. Y. Huang. A hybrid particle swarm optimization approach for clustering and classification of datasets. *Knowledge-Based Systems*, 24(3):420–426, 2011.

[67] R. A. Huber and A. M. Headrick. *Handwriting Identification: Facts and Fundamentals*. CRC Press, 1999.

[68] D. Impedovo and G. Pirlo. Automatic signature verification: the state of the art. *IEEE Transactions on Systems, Man, an Cybernetics*, 38(5):609–635, September 2008.

[69] D. Impedovo, G. Pirlo, and G.Vessio. Dynamic handwriting analysis for supporting earlier parkinson's disease diagnosis. *Information*, 9(10), October 2018.

[70] D. Impedovo, G. Pirlo, L. Sarcinella, E. Stasolla, and C. A. Trullo. Analyis of stability in static signatures using cosine similarity. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 231–235, 2012.

[71] S. Ioffe and C. Szegedy. Batch normalization: Accelarating deep network training by reducing internal convariance shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[72] A. Jain, K. Nandakumar, and A. Ross. 50 years of biometric research: Accomplishments, challenges, and opportunities. *Pattern Recognition Letters*, 79:80–105, 2016.

[73] A. K. Jain, P. Flynn, and A. A. Ross. *Handbook of Biometrics*. Springer, 2008.

[74] J. L. Wayman A. L. Jain, D. Maltoni, and D. Maio. *Biometric Systems: Technology, Design and Performance Evaluation*. Springer, 2005.

[75] M. K. Kalera, S. Srihari, and A. Xu. Offline signature verification and identification using distance statistics. *International Journal of Pattern Recogtion*, 18(7):1339–1360, 2004.

[76] J. Kennedy and R. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks*, pages 1942–1948, 1995.

[77] R. Knop. Remark on algorithm 334 [g5] normal random deviates. *Communications of ACM*, 12(5):281, May 1969.

[78] K. M. Koppenhaver. *Forensic Document Examination: Principles and Practice*. Humana Press, 2007.

[79] S. Kotz and S. Nadarajah. *Extreme Value Distributions: Theory and Applications*. Imperial College Press, 2000.

[80] B. Kovari and H. Charaf. Analysis of intra-person variability of features for off-line signature verification. *WSEAS Transactions on Computers*, 9(11):1359–1368, November 2010.

[81] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[82] R. Kumar, L. Kundu, B. Chanda, and J.D. Sharma. A writer-independent off-line signature verification system based on signature morphology. In *1st International Conference on Intelligent Interactive Technologies and Multimedia*, page 261–265, 2010.

[83] R. Kumar and Puhan. Off-line signature verification: Upper and lower envelope shape analysis using chord moments. *IET Biometrics*, 3(4):347–354, July 2014.

[84] R. Kumar, J. D. Sharma, and B.Chanda. Writer-independent off-line signature verification using surroundedness feature. *Pattern Recogtion Letters*, 33:301–308, 2012.

[85] V. Kumar, H. Glaude, C. de Lichy, and W. Campbell. A closer look at feature space data augmentation for few-shot intent classification. In *Workshop on Deep Learning Approaches for Low resource Natural Language Processing*, pages 1–10, 2019.

[86] Y. LeCun, B. Yoshua, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[87] J. A. Lewis. *Forensic Document Examination: Fundamentals and Current Trends*. Academic Press, 2014.

[88] S. Z. Li and A. K. Jain and. *Encyclopedia of Biometrics 2nd Edition*. SpringerReference, 2015.

[89] L. Liu, L. Huang F. Yin, and Y. Chen and. Offline signature verification using a region based deep metric learning network. *Pattern Recognition*, 118:108009, May 2021.

[90] M. I. Malik, M. Liwicki, A. Dengel, S. Uchida, and V. Frinken. Automatic signature stability analysis and verification using local features. In *14th International Conference on Frontiers in Handwriting Recognition*, pages 621–626, 2014.

[91] H. Martin. *The History and Power of Writing*. The University of Chicago Press, 1994.

[92] T. M. Maruyama, L. S. Oliveira, A. S. Britto Jr., and R. Sabourin. Intrapersonal parameter optimization for offline handwritten signature augmentation. *Transactions on Information Forensics and Security*, 16:1335–1350, 2021.

[93] S. Mitra and M. Gofman. *Biometrics in a Data Driven World: Trends, Technologies, and Challenges*. CRC Press, 2017.

[94] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.

[95] M. Okawa. Synergy of foreground-background images for feature extraction: Offline signature verification using fisher vector with fused KAZE features. *Pattern Recognition*, 79:480–489, February 2018.

[96] L. S. Oliveira, E. Justino, C. Freitas, and R. Sabourin. Graphology applied to signature verification. In *12^{th} Conference of the International Graphonomics Society*, pages 286–290, 2005.

[97] L. S. Oliveira, E. Justino, R. Sabourin, and F. Bortolozzi. Combining classifiers in the roc-space for off-line signature verification. *Journal of Universal Computer Science*, 14(2):237–251, January 2008.

[98] S. Y. Ooi, A. B. J. Teoh, Y. H. Pang, and B.Y. Hiew. Image-based handwritten signature verification using hybrid methods of discrete radon transform, principal component analysis and probabilistic neural network. *Applied Soft Computing*, 40:274–282, March 2016.

[99] S. Pal. *Multi-script Off-line Signature Verification*. PhD thesis, Griffith University, October 2014.

[100] R. E. A. C. Paley and N. Wiener. *Fourier Transforms in the Complex Domain*. American Mathematical Society, 1934.

[101] A. L. Pepe, D. K. Rogers, and J. C. Sita. A consideration of signature complexity using simulators' gaze behaviour. *Journal of Forensic Document Examination*, 22:5–13, 2012.

[102] G. Pirlo and D. Impedovo. Cosine similarity for analysis and verification of static signatures. *IET Biometrics*, 2(4):151–158, April 2013.

[103] G. Pirlo, D. Impedovo, and M. Fairhurst. *Advances in Digital Handwritten Signature Processing: A Human Artefact for E-Society*. World Scientific, 2014.

[104] R. Plamondon and G. Lorette. Automatic signature verification and writer identification - the state of the art. *Pattern Recognition*, 125:107–131, January 1989.

[105] H. Rantzsch, H. Yang, and C. Meinel. Signature embedding: Writer independent offline signature verification with deep metric learning. In *International Symposium on Visual Computing*, pages 616–625, 2016.

[106] D. Rivard, E. Granger, and R. Sabourin. Multi-feature extraction and selection in writer-independent off-line signature verification. *International Journal on Document Analysis and Recognition*, 16(1):83–103, 2013.

[107] A. A. Ross, K. Nandakumar, and A. K. Jain. *Handbook of Multibiometrics*. Springer, 2006.

[108] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[109] 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*, 374:30–41, 2020.

[110] J. Ruiz-del-Solar, C. Devia, P. Loncomilla, and F. Concha. Offline signature verification using local interest points and descriptors. In *Iberoamerican Congress on Pattern Recognition*, pages 22–29, 2008.

[111] R. Sabourin, M. Cheriet, and G. Genest. An extended-shadow-code based approach for off-line signature verification. In *International Conference on Document Analysis and Recognition*, pages 450–453, 1994.

[112] S. Salehi, A. Selamat, M.R. Mashinchi, and H. Fujita. The synergistic combination of particle swarm optimization and fuzzy sets to design granular classifier. *Knowledge-Based Systems*, 22:200–218, 2015.

[113] S. Sayeed, A. Samraj, R. Besar, and J. Hossen. Online hand signature verification: A review. *Journal of Applied Sciences*, 10(15):1632–1643, 2010.

[114] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *International Society for Music Information Retrieval Conference*, pages 121–126, 2015.

[115] Y. Serdouk, H. Nemmour, and Y. Chibani. Combination of oc-lbp and longest run features for off-line signature verification. In *International Conference on Signal-Image Technology and Internet-Based Systems*, pages 84–88, November 2014.

[116] Y. Serdouk, H. Nemmour, and Y. Chibani. Orthogonal combination and rotation invariant of local binary patterns for off-line handwritten signature verification. In *International Conference on Telecommunications and ICT*, 2014.

[117] Y. Serdouk, H. Nemmour, and Y. Chibani. New off-line handwritten signature verification method based on artificial immune recognition system. *Expert Systems With Applications*, 51:186–194, 2016.

[118] Y. Serdouk, H. Nemmour, and Y. Chibani. Handwritten signature verification using quad tree histogram of templates and a support vector-based artificial immune classification. *Image and Vision Computing*, 66:26–35, 2017.

[119] C. A. Shoniregun and S. Crosier. *Securing Biometrics Applications*. Springer, 2008.

[120] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[121] A. Soleimani, B. N. Araabi, and K. Fouladi. Deep multitask metric learning for offline signature verification. *Pattern Recognition Letters*, 80:84–90, 2016.

[122] M. Song and Z. Sun. An immune clonal selection algorithm for synthetic signature generation. *Mathematical Problems in Engineering*, 10(15), 2014.

[123] 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 Systems With Applications*, 154:113397, March 2020.

[124] M. Taha, M. M. Selim, and A. Yousry. Secured digital handwritten signature prototype for visually impaired people. *International Journal of Intelligent Engineering & Systems*, 13(5):307–3016, 2020.

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

[126] D.K. Thara, B.G. PremaSudha, and F. Xiong. Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques. *Pattern Recognition Letters*, 128:544–550, 2019.

[127] J. F. Vargas, M. A. Ferrer, C. M. Tavieso, and J. B. Alonso. Off-line handwritten signature verification using deep convolutional neural networks. In 9^{th} *International Conference on Document Analysis and Recognition*, pages 764–768, 2007.

[128] J. F. Vargas, M. A. Ferrer, C. M. Travieso, and J. B. Alonso. Off-line signature verification based on grey level information using texture features. *Pattern Recognition*, 44(2):375–385, February 2011.

[129] J. von Neumann. Various techniques used in connection with random digits. In A. S. Householder, G. E. Forsythe, and H. H. Germond, editors, *Monte Carlo Method*, volume 12 of *National Bureau of Standards Applied Mathematics Series*, chapter 13, pages 36–38. US Government Printing Office, Washington, DC, 1951.

[130] P. Wang, F. Su, Z. Zhao, Y. Guo, Y. Zhao, and B. Zhuang. Deep class-skewed learning for face recognition. *Neurocomputing*, 363:35–45, 2019.

[131] J. Wen, B. Fang, Y.Y. Tang, and T. Zhang. Model-based signature verification with rotation invariant features. *Pattern Recognition*, 42:1458–1466, 2009.

[132] J. D. Wooodward Jr., N. M. Orlans, and P. T. Higgins. *Biometrics*. McGraw-Hill, 2003.

[133] Y. LeCun Y, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345, 1999.

[134] Q. Yang, X. Li, H. Cai, Y.M. Hsu, J. Lee, C.H. Yang, Z.L. Li, and M.Y. Lin. Fault prognosis of industrial robots in dynamic working regimes: Find degradation in variations. *Measurement*, 173:108545, 2021.

[135] M. M. Yapıcı, A. Tekerek, and N. Topaloglu. Deep learning-based data augmentation method and signature verification system for offline handwritten signature. *Pattern Analysis and Applications*, pages 1–15, 2020.

[136] M. B. Yilmaz and K. Ozturk. Hybrid user-independent and user-dependent offline signature verification with a two-channel cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 526–534, 2018.

[137] M. B. Yilmaz and B. Yanikoğlu. Score level fusion of classifiers in off-line signature verification. *Information Fusion*, 32:109–119, February 2016.

[138] M. B. Yilmaz, B. Yanikoğlu, C. Tirkaz, and A. Kholmatov. Offline signature verification using classifier combination of hog and lbp features. In *2011 International Joint Conference on Biometrics (IJCB)*, pages 1–7, 2011.

[139] B. Zhang. Off-line signature verification and identification by pyramid histogram of oriented gradients. *International Journal of Intelligent Computing and Cybernetics*, 3(4):611–630, 2010.

[140] D. D. Zhang. *Automated Biometrics: Technologies and Systems*. Kluwer Academic Publishers, 2000.

[141] Y. Zhang, X. Liu, F. Bao, J. Chi, C. Zhang, and P. Liu. Particle swarm optimization with adaptive learning strategy. *Knowledge-Based Systems*, 196:105789, 2020.

[142] Z. Zhang, X. Liu, and Y. Cui. Multi-phase offline signature verification system using deep convolutional generative adversarial networks. In *9th International Symposium on Computational Intelligence and Design*, pages 103–107, 2016.

[143] F. Zhao. Optimized algorithm for particle swarm optimization. *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, 10:96–100, 2016.

[144] Y. Zheng, B. K. Iwana, Malik M. I, S. Ahmed, W. Ohyama, and S. Uchida. Learning the micro deformations by max-pooling for offline signature verification. *Pattern Recognition*, 118:108008, May 2021.

[145] E. N. Zois, L. Alewijnse, and G. Economou. Offline signature verification and quality characterization using poset-oriented grid features. *Pattern Recognition*, 54:162–177, January 2016.

[146] E. N. Zois, A. Alexandridis, and G. Economou. Writer independent offline signature verification based on asymmetric pixel relations and unrelated training-testing datasets. *Expert Systems with Applications*, 125:14–32, 2019.

[147] 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 Transactions on Biometrics, Behavior, and Identity Science*, 1(1):68–81, January 2019.

[148] R. Zouari, R. Mokni, and M. Kherallah. Identification and verification system of offline handwritten signature using fractal approach. In *International Image Processing, Applications and Systems Conference*, pages 1–4, November 2014.