Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks

Luiz G. Hafemann[a,*], Robert Sabourin[a], Luiz S. Oliveira[b]

[a]*LIVIA, École de Technologie Supérieure, University of Quebec, Montreal, Quebec, Canada*
[b]*Department of Informatics, Federal University of Parana (UFPR), Curitiba, PR, Brazil*

Abstract

Verifying the identity of a person using handwritten signatures is challenging in the presence of skilled forgeries, where a forger has access to a person's signature and deliberately attempt to imitate it. In offline (static) signature verification, the dynamic information of the signature writing process is lost, and it is difficult to design good feature extractors that can distinguish genuine signatures and skilled forgeries. This reflects in a relatively poor performance, with verification errors around 7% in the best systems in the literature. To address both the difficulty of obtaining good features, as well as improve system performance, we propose learning the representations from signature images, in a Writer-Independent format, using Convolutional Neural Networks. In particular, we propose a novel formulation of the problem that includes knowledge of skilled forgeries from a subset of users in the feature learning process, that aims to capture visual cues that distinguish genuine signatures and forgeries regardless of the user. Extensive experiments were conducted on four datasets: GPDS, MCYT, CEDAR and Brazilian PUC-PR datasets. On GPDS-160, we obtained a large improvement in state-of-the-art performance, achieving 1.72% Equal Error Rate, compared to 6.97% in the literature. We also verified that the features generalize beyond the GPDS dataset, surpassing the state-of-the-art performance in the other datasets, without requiring the representation to be fine-tuned to each particular dataset.

Keywords: Signature Verification, Convolutional Neural Networks, Feature Learning, Deep Learning

*Corresponding author
Email addresses: lghafemann@livia.etsmtl.ca (Luiz G. Hafemann), robert.sabourin@etsmtl.ca (Robert Sabourin), lesoliveira@inf.ufpr.br (Luiz S. Oliveira)

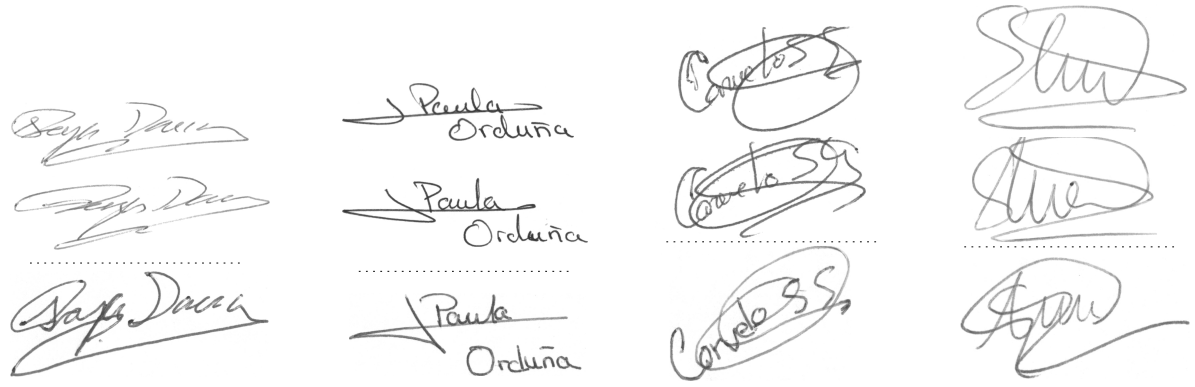
1. Introduction

Signature verification systems aim to verify the identity of individuals by recognizing their handwritten signature. They rely on recognizing a specific, well-learned gesture, in order to identify a person. This is in contrast with systems based on the possession of an object (e.g. key, smartcard) or the knowledge of something (e.g. password), and also differ from other biometric systems, such as fingerprint, since the signature remains the most socially and legally accepted means for identification [1].

In offline (static) signature verification, the signature is acquired after the writing process is completed, by scanning a document containing the signature, and representing it as a digital image [2]. Therefore, the dynamic information about the signature generation process is lost (e.g. position and velocity of the pen over time), which makes the problem very challenging.

Defining discriminative feature extractors for offline signatures is a hard task. The question "What characterizes a signature" is a difficult concept to implement as a feature descriptor, as illustrated in Figure 1. This can be observed in the literature, where most of the research efforts on this field have been devoted to finding a good representation for signatures, that is, designing feature extractors tailored for signature verification, as well as using feature extractors created for other purposes [3]. Recent work uses texture features, such as Local Binary Patterns (LBP) [4], [5] and Gray-Level Co-occurrence Matrix (GLCM) [5]; directional-based features such as Histogram of Oriented Gradients (HOG) [4] and Directional-PDF [6], [7]; feature extractors specifically designed for signatures, such as the estimation of strokes by fitting Bezier curves [8]; among others. No feature extractor has emerged as particularly suitable for signature verification, and most recent work uses a combination of many such techniques.

The difficulty of finding a good representation for signatures reflects on the classification performance of signature verification systems, in particular to distinguish genuine signatures and skilled forgeries - forgeries that are made targeting a particular individual. When we consider experiments conducted on large public datasets, such as GPDS [9], the best reported results achieve



(a) (b) (c) (d)

Figure 1: Examples of challenges in designing feature extractors for offline signatures, and the challenge of classifying skilled forgeries. Each column shows two genuine signatures from the same user in the GPDS dataset, and a skilled forgery created for the user. We notice that skilled forgeries resemble genuine signatures to a large extent. Since we do not have examples from the forgery class for training, the problem is even more challenging. We also note the challenges of creating feature extractors for these genuine signatures: **(a)** The shape of the first name is very different among the two genuine samples. A feature descriptor based on grid features would have very different vectors for the two samples. **(b)** The shape of the characters in the first name ("Paula") is very different. An analysis based on the design of individual letters would perform poorly for this user. **(c)** Large variation in flourishes may impact directional-based descriptors (such as HOG or D-PDF). **(d)** For some users, it is difficult to pinpoint the common attributes of two signatures even after carefully analyzing the samples.

Equal Error Rates around 7%, even when the number of samples for training is around 10-15, with worse results using fewer samples per user.

To address both the issue of obtaining a good feature representation for signatures, as well as improving classification performance, we propose a framework for learning the representations directly from the signature images, using convolutional neural networks. In particular, we propose a novel formulation of the problem, that incorporates knowledge of skilled forgeries from a subset of users, using a multi-task learning strategy. The hypothesis is that the model can learn visual cues present in the signature images, that are discriminative between genuine signatures and forgeries in general (i.e. not specific to a particular individual). We then evaluate if this feature representation generalizes for other users, for whom we do not have skilled forgeries available.

Our main contributions are as follows: 1) we present formulations to learn features for offline signature verification in a Writer-Independent format. We introduce a novel formulation that uses skilled forgeries from a subset of users to guide the feature learning process, using a multi-task framework to jointly optimize the model to discriminate between users (addressing random forg-

eries), and to discriminate between genuine signatures and skilled forgeries; 2) we propose a strict experimental protocol, in which all design decisions are made using a validation set composed of a separate set of users. Generalization performance is estimated in a disjoint set of users, from whom we do not use any forgeries for training; 3) we present a visual analysis of the learned representations, which shows that genuine signatures and skilled forgeries get better separated in different parts of the feature space; 4) lastly, we are making two trained models available for the research community[1], so that other researchers can use them as specialized feature extractors for the task.

Experiments were conducted on four datasets, including the largest publicly available signature verification dataset (GPDS), achieving a large performance improvement in the state-of-the-art, reducing Equal Error Rates from 6.97% to 1.72% in GPDS-160. We used the features learned on this dataset to train classifiers for users in the MCYT, CEDAR and Brazilian PUC-PR datasets, also surpassing the state-of-the-art performance, and showing that the learned feature space not only generalizes to other users in the GPDS set, but also to other datasets.

Preliminary results, using only genuine signatures for learning the features, were published as two conference papers. In [10], we introduced the formulation to learn features from genuine signatures from a development dataset, using them to train Writer-Dependent classifiers to another set of users. In [11], we analyzed the learned feature space and optimized the CNN architecture, obtaining state-of-the-art results on GPDS. The present work includes this formulation of the problem for completeness, with additional experiments on two other datasets (MCYT and CEDAR), a clearer explanation of the method and the experimental protocol, as well as the novel formulation that leverages knowledge of skilled forgeries for feature learning.

The remaining of this paper is organized as follows: Section 2 reviews the related work on signature verification and on feature learning techniques. Section 3 details the formulation and methodology to learn features for offline signature verification, and section 4 describes our experimental protocol. Section 5 presents and discusses the results of our experiments. Lastly, section 6 concludes the paper.

[1] https://www.etsmtl.ca/Unites-de-recherche/LIVIA/Recherche-et-innovation/Projets

2. Related works

The review of related works is divided below into two parts: we first present a review of previous work on Offline Signature Verification, followed by a brief review of representation learning methods.

2.1. Related works on Offline Signature Verification

The area of automatic Offline Signature Verification has been researched at least since the decade of 1970. Over the years, the problem has been addressed from many different perspectives, as summarized by [12], [13] and [2].

In this problem, given a set of genuine signatures, the objective is to learn a model that can distinguish between genuine signatures and forgeries. Forgeries are signatures not created by a claimed individual, and are often subdivided into different types. The most common classification of forgeries in the literature considers: Random Forgeries, where a person uses his or her own signature to impersonate another individual, and Skilled Forgeries, where a person tries to imitate the signature of the claimed individual. While the former is a relatively easier task, discriminating skilled forgeries is an open pattern recognition problem, and is the focus of this paper. This problem is challenging due to a few factors: First, there is a large similarity between genuine signatures and skilled forgeries, as forgers will attempt to imitate the user's signature, often practicing the signature beforehand. Second, in a practical application scenario, we cannot expect to have skilled forgeries for all users in the system, therefore the classifiers should be trained only with genuine signatures in order to be most widely applicable. Lastly, the number of genuine samples per user is often small, especially for new users of the system, for whom we may have only 3 or 5 signatures. This is especially problematic as many users have large intra-class variability, and a few signatures are not sufficient to capture the full range of variation.

There are mainly two approaches for building offline signature verification systems. The most common approach is to design Writer-Dependent classifiers. In this scenario, a training set is constructed for each user of the system, consisting of genuine signatures as positive examples and genuine signatures from other users (random forgeries) as negative samples. A binary classifier is then trained on this dataset, resulting in one model for each user. This approach has shown

to work well for the task, but since it requires one model to be trained for each user, complexity increases as more users are enrolled. An alternative is Writer-Independent classification. In this case, a single model is trained for all users, by training a classifier in a dissimilarity space [8], [7]. The inputs for classification are dissimilarity vectors, that represent the difference between the features of a query signature, and the features of a template signature (a genuine signature of the user). In spite of the reduced complexity, Writer-Independent systems often perform worse, and the best results in standard benchmarks are obtained with Writer-Dependent systems.

A large variety of feature extractors have been investigated for this problem, from simple geometric descriptors [14], [15], descriptors inspired in graphology and graphometry [16], directional-based descriptors such as HOG [4] and D-PDF [17], [6], [7], descriptors based on interest-point, such as SIFT [4], to texture descriptors, such as Local Binary Patterns (LBP) [4] and Gray-Level Co-occurrence Matrix (GLCM) [5]. These features are commonly extracted locally from the signature images, by dividing the image in a grid and computing descriptors for each cell (either in Cartesian or polar coordinates).

Methods to learn features from data have not yet been widely explored for offline signature verification. Ribeiro et al. [18] used Restricted Boltzmann Machines (RBMs) to learn features from signature images. However, in this work they only showed the visual appearance of the weights, and did not test the features for classification. Khalajzadeh [19] used Convolutional Neural Networks (CNNs) for signature verification on a dataset of Persian signatures, but only considered the classification between different users (e.g. detecting random forgeries), and did not considered skilled forgeries. Soleimani et al. [20] proposed a solution using deep neural networks for Multitask Metric Learning. In their work, a distance metric between pairs of signatures is learned. Contrary to our work, the authors used handcrafted feature extractors (LBP in the experiments with the GPDS dataset), while in our work the inputs to the system are the signature themselves (pixel intensities), and the feature representation is learned. In a similar vein to our work, Eskander [7] presented a hybrid Writer-Independent Writer-Dependent solution, using a Development dataset for feature selection, followed by training WD classifiers using the selected features. However, in the present work we use a Development dataset for feature learning instead of feature selection.

2.2. *Related work on Representation Learning for computer vision tasks*

In recent years, there has been a large interest in methods that do not rely on hand-crafted features, but rather learn the representations for a problem using *raw* data, such as pixels, in the case of images. Methods based on learning multiple levels of representation have shown to be very effective to process natural data, especially in computer vision and natural language processing [21], [22], [23]. The intuition is to use such methods to learn multiple intermediate representations of the input, in layers, in order to better represent a given problem. In a classification task, the higher layers amplify aspects of the input that are important for classification, while disregarding irrelevant variations [23]. In particular, Convolutional Neural Networks (CNNs) [24] have been used to achieve state-of-the-art performance [23] in many computer vision tasks [25], [26]. These models use local connections and shared weights, taking advantage of the spatial correlations of pixels in images by learning and using the same filters in multiple positions of an input image [23]. With large datasets, these networks can be trained with a purely supervised criteria. With small datasets, other strategies have been used successfully, such as unsupervised pre-training (e.g. in a greedy layer-wise fashion [27]), and more recently with transfer learning [28], [29], [30]. CNNs have been used to transfer learning of representations, by first training a model in a large dataset, and subsequently using this model in another task (often, a task for which a smaller dataset is available), by using the network as a "feature extractor": performing forward-propagation of the samples until one of the last layers before softmax [28], [29], or the last layer (that corresponds to the predictions for classes in the original task, as in [30]), and using the activation at that layer as a feature vector. Alternatively, this pre-trained model can be used to initialize the weights of a model for the task of interest, and training proceeds normally with gradient descent.

3. Feature learning for Signature Verification

In this work we present formulations for learning features for Offline Signature Verification, and evaluate the performance of such features for training Writer-Dependent classifiers. We first note that a supervised feature learning approach directly applied for Writer-Dependent classification is not practical, since the number of samples per user is very small (commonly around 1-14

samples), while most feature learning algorithms have a large number of parameters (in the order of millions of parameters, for many computer vision problems, such as object recognition [25]). On the other hand, we expect that signatures from different users share some properties, and we would like to exploit this intuition by learning features across signatures from different writers.

We consider a two-phase approach for the problem: a Writer-Independent feature learning phase followed by Writer-Dependent classification. The central idea is to leverage data from many users to learn a feature space that captures intrinsic properties of handwritten signatures. We subsequently train classifiers for each user, using this feature space, that model the characteristics of each user. Since in real applications the list of users of the system is not fixed, we consider a disjoint set of users for learning the features and training the writer-dependent classifiers, to verify if the learned feature space is useful (i.e. generalizes) to new users. We use the term Writer-Independent for the feature learning process, since the learned representation space is therefore not specific for a set of users.

Given a development set $\mathcal{D}$ of signatures, we train Deep Convolutional Neural Networks (CNNs) using the formulations defined below. Subsequently, we use the trained network to project the input signatures onto the representation space learned by the CNN for an Exploitation set $\mathcal{E}$, and train a binary classifier for each user. The hypothesis is that genuine signatures and forgeries will be easier to separate in this feature space, if the network succeeds in capturing intrinsic properties of the signatures, that generalizes to other users.

Convolutional Neural Networks are a particularly suitable architecture for signature verification. This type of architecture scales better than fully connected models for larger input sizes, having a smaller number of trainable parameters. This is a desirable property for the problem at hand, since we cannot reduce the signature images too much without risking losing the details that enable discriminating between skilled forgeries and genuine signatures (e.g. the quality of the pen strokes). We also note that this type of architecture shares some properties with handcrafted feature extractors used in the literature, as features are extracted locally (in an overlapping grid of patches) and combined in non-linear ways (in subsequent layers). In the sections below we present our proposed formulations for the problem, first considering only genuine signatures, and then considering learning from skilled forgeries.

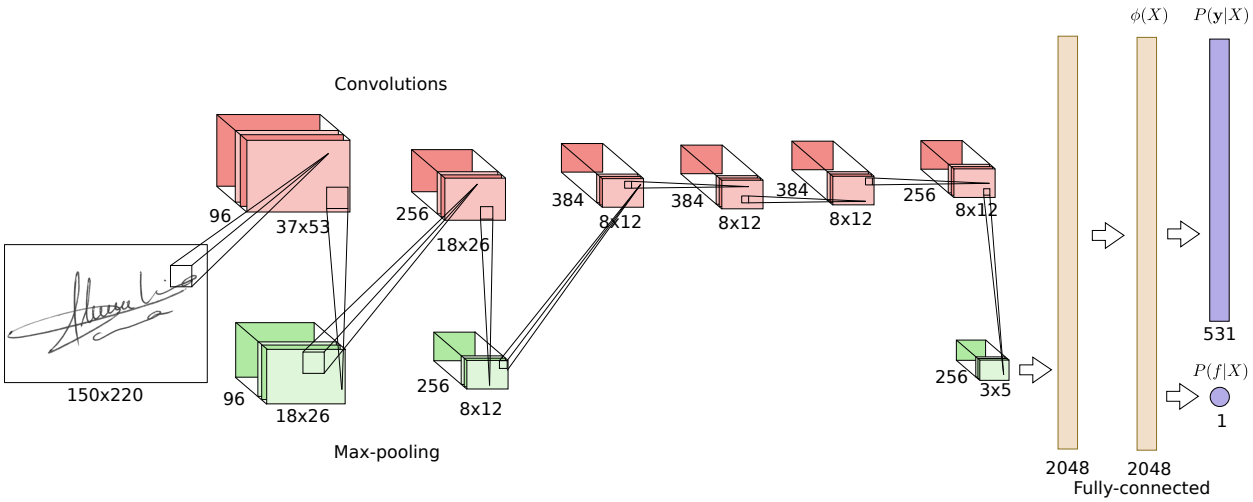


Figure 2: Illustration of the CNN architecture used in this work. The input image goes through a sequence of transformations with convolutional layers, max-pooling layers and fully-connected layers. During feature learning, $P(\mathbf{y}|X)$ (and also $P(f|X)$ in the formulation from sec 3.2.2) are estimated by performing forward propagation through the model. The weights are optimized by minimizing one of the loss functions defined in the next sections. For new users of the system, this CNN is used to project the signature images onto another feature space (analogous to "extract features"), by performing feed-forward propagation until one of the last layers before the final classification layer, obtaining the feature vector $\phi(X)$.

3.1. Learning features from genuine signatures

Let $\mathcal{D}$ be a dataset consisting of genuine signatures from a set of users $\mathcal{Y}_{\mathcal{D}}$. The objective is to learn a function $\phi(X)$ that projects signatures X onto a representation space where signatures and forgeries are better separated. To address this task, we consider learning a Convolutional Neural Network to discriminate between users in $\mathcal{D}$. This formulation has been introduced in [10], and it is included here for completeness.

Formally, we consider a training set composed of tuples (X, y) where X is the signature image, and y is the user, that is, $y \in \mathcal{Y}_{\mathcal{D}}$. We create a neural network with multiple layers, where the objective is to discriminate between the users in the Development set. The last layer of the neural network has M units with a softmax activation, where M is the number of users in the Development set, ($M = |\mathcal{Y}_{\mathcal{D}}|$), and estimates $P(\mathbf{y}|X)$. Figure 2 illustrates one of the architectures used in this work, with $M = 531$ users. We train the network to minimize the negative log likelihood of the correct user given the signature image:

$$L = -\sum_{j} y_{ij} \log P(y_j|X_i) \tag{1}$$

Where y_{ij} is the true target for example i ($y_{ij} = 1$ if the signature belongs to user j), X_i is the signature image, and $P(y_j|X_i)$ is the probability assigned to class j for the input X_i, given by the model. This cost function can then be minimized with a gradient-based method.

The key idea behind this approach is that by training the network to distinguish between users, we expect it to learn a hierarchy of representations, and that the representations on the last layers capture relevant properties of signatures. In particular, if the network succeeds in distinguishing between different users of the development set, then the representation of signatures from these users will be linearly separable in the representation space defined by $\phi(X)$, since the last layer is a linear classifier with respect to its input $\phi(X)$. We test, therefore, the hypothesis that this feature space generalizes well to signatures from other users.

3.2. *Learning features from genuine signatures and skilled forgeries*

One limitation of the formulation above is that there is nothing in the training process to drive the features to be good in distinguishing skilled forgeries. Since this is one of the main goals of a signature verification system, it would be beneficial to incorporate knowledge about skilled forgeries in the feature learning process.

In a real application scenario, we cannot expect to have skilled forgeries available for each user enrolled in the system. We consider, however, a scenario where we obtain skilled forgeries for a subset of the users. Assuming such forgeries are available, we would like to formulate the feature learning process to take advantage of this data. Using the same notation as above, we consider that the development set $\mathcal{D}$ contains genuine signatures and skilled forgeries for a set of users, while the exploitation set $\mathcal{E}$ contains only genuine signatures available for training, and represent the users enrolled to the system.

In this section we introduce novel formulations for the problem, that incorporate forgeries in the feature learning process. The first approach considers the forgeries of each user as a separate class, while the second formulation considers a multi-task learning framework.

3.2.1. Treat forgeries as separate classes

A simple formulation to incorporate knowledge of skilled forgeries into training is to consider the forgeries of each user as a different class. In this formulation, we have two classes for each user (genuine signatures and forgeries), that is, $M = 2|\mathcal{Y}_\mathcal{D}|$.

We note that this alternative is somewhat extreme, as it considers genuine signatures and forgeries as completely separate entities, while we would expect genuine signatures and skilled forgeries to have a high level of resemblance.

3.2.2. Add a separate output for detecting forgeries

Another formulation is to consider a multi-task framework, by considering two terms in the cost function for feature learning. The first term drives the model to distinguish between different users (as in the formulations above), while the second term drives the model to distinguish between genuine signatures and skilled forgeries. Formally, we consider another output of the model: $P(f|X)$, a single sigmoid unit, that seeks to predict whether or not the signature is a forgery. The intuition is that in order to classify between genuine signatures and forgeries (regardless of the user), the network will need to learn visual cues that are particular to each class (e.g. bad line quality in the pen strokes, often present in forgeries).

We consider a training dataset containing tuples of the form (X, y, f), where X is the signature image, y is the author of the signature (or the target user, if the signature is a forgery), and f is a binary variable that reflects if the sample is a forgery or not ($f = 1$ indicates a forgery). Note that contrary to the previous formulation, genuine signatures and forgeries targeted to the same user have the same y. For training the model, we consider a loss function that combines both the classification loss (correctly classifying the user), and a loss on the binary neuron that predicts whether or not the signature is a forgery. The individual losses are shown in Equation 2, where the user classification loss (L_c) is a multi-class cross-entropy, and the forgery classification (L_f) is a binary cross-entropy:

$$
\begin{aligned}
L_c &= -\sum_j y_{ij} \log P(y_j|X_i) \\
L_f &= -f_i \log(P(f|X_i)) - (1 - f_i) \log(1 - P(f|X_i))
\end{aligned}
\tag{2}
$$

For training the model, we combine the two loss functions and minimize both at the same time. We considered two approaches for combining the losses. The first approach considers a weighted sum of both individual losses:

$$
\begin{aligned}
L_1 &= (1 - \lambda) L_c + \lambda L_f \\
&= -(1 - \lambda) \sum_j y_{ij} \log P(y_j|X_i) + \\
&\lambda\big(- f_i \log(P(f|X_i)) - (1 - f_i) \log(1 - P(f|X_i))\big)
\end{aligned}
\tag{3}
$$

Where λ is a hyperparameter that trades-off between the two objectives (separating the users in the set $\mathcal{D}$, and detecting forgeries)

In a second approach we consider the user classification loss only for genuine signatures:

$$
\begin{aligned}
L_2 &= (1 - f_i)(1 - \lambda) L_c + \lambda L_f \\
&= -(1 - f_i)(1 - \lambda) \sum_j y_{ij} \log P(y_j|X_i) + \\
&\lambda\big(- f_i \log(P(f|X_i)) - (1 - f_i) \log(1 - P(f|X_i))\big)
\end{aligned}
\tag{4}
$$

In this case, the model is not penalized for misclassifying for which user a forgery was made.

In both cases, the expectation is that the first term will guide the model to learn features that can distinguish between different users (i.e. detect random forgeries), while the second term will focus on particular characteristics that distinguish between genuine signatures and forgeries (such as limp strokes). It is worth noting that, in the second formulation, using $\lambda = 0$ is equivalent to the formulation in section 3.1, where only genuine signatures are used for training, since the forgeries

would not contribute to the loss function.

3.3. Preprocessing

The signatures from the datasets used in our experiments are already extracted from the documents where they were written, so signature extraction is not investigated in this paper. Some few preprocessing steps are required, though. The neural networks expect inputs of a fixed size, where signatures vary significantly in shape (in GPDS, they range from small signatures of size 153x258 to large signatures of size 819x1137 pixels).

We first center the signatures in a large canvas of size $S_{\text{canvas}} = H \times W$, by using the signatures' center of mass. We remove the background using OTSU's algorithm [31], setting background pixels to white (intensity 255), and leaving the foreground pixels in grayscale. The image is then inverted by subtracting each pixel from the maximum brightness $I(x, y) = 255 - I(x, y)$, such that the background is zero-valued. Lastly, we resize the image to the input size of the network.

3.4. Training the Convolutional Neural Networks

For each strategy described above, we learn a feature representation $\phi(.)$ on the Development set of signatures by training a Deep Convolutional Neural Network on this set. This section describes the details of the CNN training.

In order to use a suitable architecture for signature verification, in [11] we investigated different architectures for learning feature representations using the objective from section 3.1 (training using only genuine signatures). In this work we use the architecture that performed best for this formulation, which is described in table 1. The CNN consists of multiple layers, considering the following operations: convolutions, max-pooling and dot products (fully-connected layers), where convolutional layers and fully-connected layers have learnable parameters, that are optimized during training. With the exception of the last layer in the network, after each learnable layer we apply Batch Normalization [32], followed by the ReLU non-linearity. The last layer uses the softmax non-linearity, which is interpreted as $P(\mathbf{y}|X)$ - the probability assigned by the network to each possible user in $\mathcal{Y}_{\mathcal{D}}$. For the formulation in section 3.2.2, the neuron that estimates $P(f|X)$ uses the sigmoid function. Both output layers receive as input the result of layer FC7. Table 2 lists the operations mentioned above.

Table 1: Summary of the CNN layers

Layer	Size	Other Parameters
Input	1x150x220	
Convolution (C1)	96x11x11	stride = 4, pad=0
Pooling	96x3x3	stride = 2
Convolution (C2)	256x5x5	stride = 1, pad=2
Pooling	256x3x3	stride = 2
Convolution (C3)	384x3x3	stride = 1, pad=1
Convolution (C4)	384x3x3	stride = 1, pad=1
Convolution (C5)	256x3x3	stride = 1, pad=1
Pooling	256x3x3	stride = 2
Fully Connected (FC6)	2048	
Fully Connected (FC7)	2048	
Fully Connected + Softmax ($P(\mathbf{y}\|X)$)	M	
Fully Connected + Sigmoid ($P(f\|X)$)	1	

Optimization was conducted by minimizing the loss with Stochastic Gradient Descent with Nesterov Momentum, using mini-batches of size 32, and momentum factor of 0.9. As regularization, we applied L2 penalty with weight decay 10^{-4}. The models were trained for 60 epochs, with an initial learning rate of 10^{-3}, that was divided by 10 every 20 epochs. We used simple translations as data augmentation, by using random crops of size 150x220 from the 170x242 signature image. As in [32], the batch normalization terms (mean and variance) are calculated from the mini-batches during training. For generalization, the mean ($\mathrm{E}[z_i]$) and variance ($\mathrm{Var}[z_i]$) for each neuron were calculated from the entire training set.

It is worth noting that, in our experiments, we found Batch Normalization to be crucial to train deeper networks. Without using this technique, we could not train architectures with more than 4 convolutional layers and 2 fully-connected layers. In these cases, the performance in both a training and validation set remained the same as chance, not indicating overfitting, but rather problems in the optimization process.

3.5. Training Writer-Dependent Classifiers

After training the CNN, we use the network to extract feature representations for signatures from the Exploitation set, and train Writer-Dependent classifiers. To do so, we crop the center

Table 2: List of feedforward operations

Operation	Formula
Convolution	$\mathbf{z}^l = \mathbf{h}^{l-1} * W^l$
MaxPooling	$h^l_{xy} = \max_{i=0,..,s,j=0,..,s} \mathbf{h}^{l-1}_{(x+i)(y+j)}$
Fully-connected layer	$\mathbf{z}^l = W^l \mathbf{h}^{l-1}$
ReLU	$\text{ReLU}(z_i) = \max(0, z_i)$
Sigmoid	$\sigma(z_i) = \frac{1}{1+e^{-z_i}}$
Softmax	$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$
Batch Normalization	$\text{BN}(z_i) = \gamma_i \hat{z}_i + \beta_i,$ $\hat{z}_i = \frac{z_i - \text{E}[z_i]}{\sqrt{\text{Var}[z_i]}}$

$\mathbf{z}^l$: pre-activation output of layer l
$\mathbf{h}^l$: activation of layer l
$*$: discrete convolution operator
W, γ, β: learnable parameters

150x220 pixels from the 170x242 signature image, perform feedforward propagation until the last layer before softmax (obtaining $\phi(X)$), and use the activations at that layer as the feature vector for the image. This can be seen as a form of transfer learning (similar to [28]) between the two sets of users. For each user, we build a training set consisting of genuine signatures from the user as positive samples, and genuine signatures from other users as negative samples. We trained Support Vector Machines (SVM), both in a linear formulation and with the Radial Basis Function (RBF) kernel.

We used different weights for the positive and negative class to account for the imbalance of having many more negative samples than positive. The SVM objective becomes [33]:

$$\begin{aligned} &\min \frac{1}{2} \|\mathbf{w}\|^2 + C^+ \Big(\sum_{i:y_i=+1} \xi_i \Big) + C^- \Big(\sum_{i:y_i=-1} \xi_i \Big) \\ &\text{subject to} \\ &\quad y_i(\mathbf{w} x_i + b) \geq 1 - \xi_i \\ &\quad \xi_i \geq 0 \end{aligned} \tag{5}$$

Where the change to the standard SVM formulation is the usage of different weights C for the two

Table 3: Summary of the datasets used in this work

Dataset Name	Users	Genuine signatures	Forgeries	S_{canvas}
Brazilian (PUC-PR)	60 + 108	40	10 simple, 10 skilled[2]	700×1000
CEDAR	55	24	24	730×1042
MCYT-75	75	15	15	600×850
GPDS Signature 960 Grayscale	881	24	30	952×1360

classes (we refer the reader to [33] for the dual formulation). We set the weight of the positive class (genuine signatures) to match the skew (denoted below as ψ). Let P be the number of positive (genuine) samples for training, and N the number of negative (random forgery) samples:

$$\psi = \frac{N}{P} \qquad\qquad C^+ = \psi C^- \tag{6}$$

For testing, we used a disjoint set of genuine signatures from the user (that is, not used for training) and the skilled forgeries made targeting the user's signature.

4. Experimental Protocol

We conducted experiments using the datasets GPDS-960 [9], MCYT-75 [34], CEDAR [35] and the Brazilian PUC-PR [36]. Table 3 summarizes these datasets, including the size used to normalize the images in each dataset (height x width). GPDS-960 is the largest publicly available dataset for offline signature verification with 881 users, having 24 genuine samples and 30 skilled forgeries per user. We used a subset of users from this dataset for learning the features (the development set $\mathcal{D}$) and evaluating how these features generalize to other users in this dataset (the exploitation set $\mathcal{E}$). To enable comparison with previous work, we performed experiments on GPDS having the set $\mathcal{E}$ as the first 160 or the first 300 users of the dataset (to allow comparison with the datasets GPDS-160, and GPDS-300, respectively). In order to evaluate if the features generalize to other datasets, we use the same models learned on GPDS to train Writer-Dependent classifiers for the MCYT, CEDAR and Brazilian PUC-PR datasets.

[2]This dataset contains simple and skilled forgeries for the first 60 users

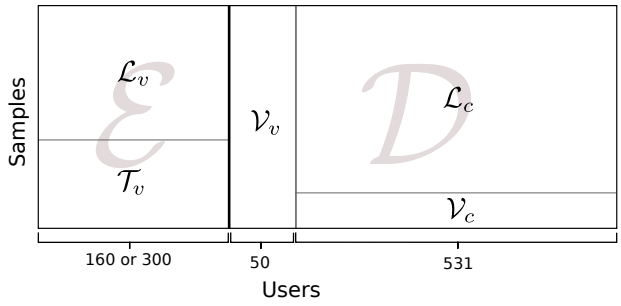


Figure 3: The GPDS dataset is separated into an exploitation set $\mathcal{E}$ and Development set $\mathcal{D}$. The development set is used for learning the features, and making all model decisions. The exploitation set represents the users enrolled to the system, where we train Writer-Dependent classifiers using only genuine signatures.

The GPDS dataset is divided as follows, as illustrated in Figure 3: The Convolutional Neural Networks are trained on a set $\mathcal{L}_c$ (denoting **L**earning set for **c**lassification) consisting of 531 users. We monitor the progress on a validation set $\mathcal{V}_c$ (**V**alidation set for **c**lassification). Both sets contains the same users, but a disjoint set of signature samples from these users. We split 90% of the signatures for training, and 10% for this validation set.

After the CNNs are trained, we train Writer-Dependent classifiers on a validation set $\mathcal{V}_v$ (**V**alidation set for **v**erification) consisting of 50 users. The purpose of this set is to allow the estimation of the performance of Writer-Dependent classifiers trained with the representation space learned by the CNN. We use this validation set to make all model choices (CNN architecture and values hyperparameters). On this validation phase, we follow the same protocol for Writer-Dependent classifier training, using a fixed number of 12 genuine signatures for the user as positive samples, and random forgeries from $\mathcal{L}_c$ as negative samples.

Finally, we use the models and hyperparameters that performed best in the validation set, to train and test classifiers for the exploitation set $\mathcal{E}$. We trained Support Vector Machines on the set $\mathcal{L}_v$ (denoting **L**earning set for **v**erification) and tested on $\mathcal{T}_v$ (**T**esting set for **v**erification). For each user, we build a dataset consisting of r genuine signatures from the user as positive samples, and genuine signatures from other users as negative samples. Taking into consideration the differences in datasets and experimental protocols that used them in the literature, we used a different number of signatures for training and testing, which is summarized in table 4. For the GPDS and the Brazilian PUC-PR datasets, we used signatures from users that are not in the Exploitation set as

Table 4: Separation into training and testing for each dataset

Dataset Name	Training set		Testing set
	Genuine	Random Forgeries	
Brazilian (PUC-PR)	$r \in \{1, \ldots, 30\}$	$30 \times 108 = 3240$	10 genuine, 10 random, 10 simple, 10 skilled
CEDAR	$r \in \{1, \ldots, 12\}$	$12 \times 54 = 648$	10 genuine, 10 skilled
MCYT-75	$r \in \{1, \ldots, 10\}$	$10 \times 74 = 588$	5 genuine, 15 skilled
GPDS-160	$r \in \{1, \ldots, 14\}$	$14 \times 721 = 10094$	10 genuine, 10 random, 10 skilled
GPDS-300	$r \in \{1, \ldots, 14\}$	$14 \times 581 = 8134$	10 genuine, 10 random, 10 skilled

random forgeries (i.e. signatures from users 301-881 for GPDS-300 and users 61-168 for the Brazilian PUC-PR). For MCYT and CEDAR, we consider genuine samples from other users from the exploitation set as negative samples for training the WD classifier. In each experiment, we performed the WD training 10 times, using different splits for the data. We report the mean and variance of the performance across these executions.

We used the same hyperparameters for training the SVM classifiers as in previous work [11]: for the linear SVM, we used $C^- = 1$ (C^+ is calculated according to equation 6). For the SVM with RBF kernel, we used $C^- = 1$ and $\gamma = 2^{-11}$. We found these hyperparameters to work well for the problem, on a range of architectures and users, but we note that they could be further optimized (to each model, or even to each user), which is not explored in this study.

For learning features using forgery data, specifically the formulation on section 3.2.2, we tested values of λ from 0 to 1 is steps of 0.1. The boundaries are special cases: with $\lambda = 0$, the forgery neuron is not used at all, and the model only classifies among different users; with $\lambda = 1$ the model does no try to separate among different users, but only classifies whether or not the input is a forgery. In our experiments, we found better results on the right end of this range, and therefore we refined the search for the appropriate λ with the following cases: $\lambda \in \{0.95, 0.99, 0.999\}$.

Besides comparing the performance with the state-of-the-art in this dataset, we also considered a baseline consisted of a CNN pre-trained on the Imagenet dataset. As argued in [37], these pre-trained models offer a strong baseline for Computer Vision tasks. We used two pre-trained models[3], namely Caffenet (Caffe reference network, based on AlexNet [25]), and VGG-19 [38].

[3] https://github.com/BVLC/caffe/wiki/Model-Zoo

We used these networks to extract the feature representations $\phi(X)$ for signatures, and followed the same protocol for training Writing-Dependent classifiers using these representations. We considered the following layers to obtain the representations: pool5, fc6 and fc7.

We evaluate the performance on the testing set using the following metrics: False Rejection Rate (FRR): the fraction of genuine signatures rejected as forgeries; False Acceptance Rate ($\text{FAR}_{\text{random}}$ and $\text{FAR}_{\text{skilled}}$): the fraction of forgeries accepted as genuine (considering random forgeries and skilled forgeries). We also report the Equal Error Rate (EER): which is the error when FAR = FRR. We considered two forms of calculating the EER: $\text{EER}_{\text{user thresholds}}$: using user-specific decision thresholds; and $\text{EER}_{\text{global threshold}}$: using a global decision threshold. In both cases, to calculate the Equal Error Rate we only considered skilled forgeries (not random forgeries) - that is, we use only FRR and $\text{FAR}_{\text{skilled}}$ to estimate the optimum threshold and report the Equal Error Rate. We also report the mean Area Under the Curve (AUC), considering ROC curves created for each user individually. For calculating FAR and FRR in the GPDS exploitation set, we used a decision threshold selected from the validation set $\mathcal{V}_v$ (the threshold that achieved EER using a global decision threshold).

For the Brazilian PUC-PR dataset, we followed the convention of previous research in this dataset, and also report the individual errors (False Rejection Rate and False Acceptance Rate for different types of forgery) and the Average error rate, calculate as $\text{AER} = (\text{FRR} + \text{FAR}_{\text{random}} + \text{FAR}_{\text{simple}} + \text{FAR}_{\text{skilled}})/4$. Since in this work we are mostly interested in the problem of distinguishing genuine signatures and skilled forgeries, we also report $\text{AER}_{\text{genuine + skilled}} = (\text{FRR} + \text{FAR}_{\text{skilled}})/2$.

5. Results and Discussion

The experimental results with the proposed method are listed and discussed in this section. The first part presents the experiments on the Development set, which was used for making all the design decisions for the proposed method: evaluating different loss functions and other hyperparameters. The second part presents the results on the Exploitation set, and the comparison with the state-of-the-art for all four datasets.

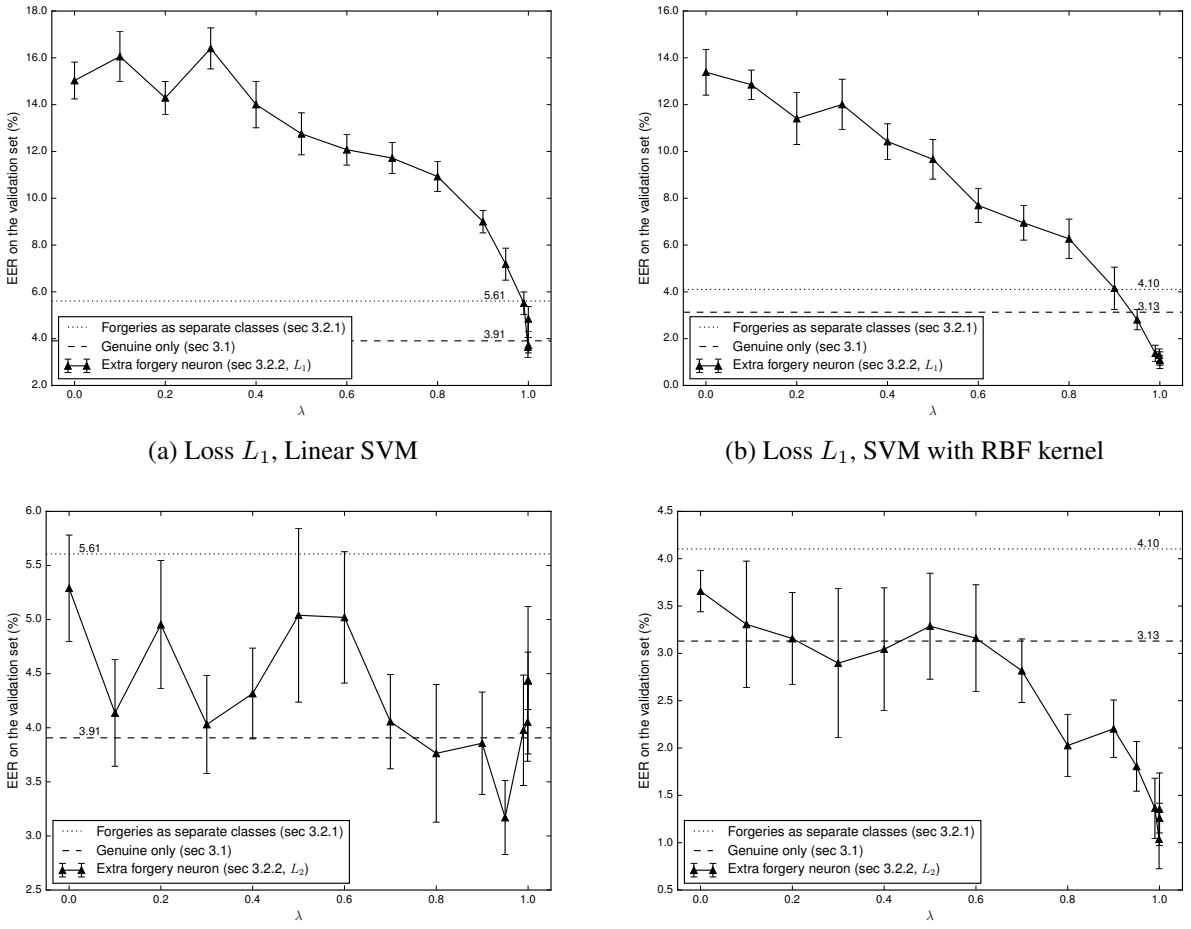
5.1. Signature Verification System Design

In these experiments, we trained the CNN architectures using the loss functions defined in section 3, used them to extract features for the users in the validation set $\mathcal{V}_v$, and trained Writer-Dependent classifiers for these users using 12 reference signatures. We then analyzed the impact in classification performance of the different formulations of the problem.

For the formulation on section 3.2.2, where we have a separate neuron to estimate if a signature is a forgery or not, we trained models with variable values of λ. Figure 4 shows the results on the validation set using loss L_1 (from equation 3), and loss L_2 (from equation 4). The models with loss L_2 only consider the user-classification loss for genuine signatures, while the models using L_1 consider user-classification loss for all signatures (genuine and forgeries). As a performance reference, we also show the results using a model trained with genuine signatures only, as well as the model trained with forgeries as separate classes (sec 3.2.1).

Both using a linear SVM or using an SVM with RBF kernel, the results using the loss L_1 were very poor for low values of λ. This is likely caused by the fact that, in this formulation, both genuine signatures and forgeries of the same user are assigned to the same class y, and the loss function guides the model to be less discriminative between the genuine signatures and forgeries of the same user. This behavior is not present when we use the loss L_2, since the model is not penalized for misclassifying for which user the forgery was created. We also noticed that the best results were closer to the right end of the range, suggesting that the distinction of forgeries (regardless of the user) in the development set may be more relevant than the distinguishing genuine signatures from different users. In the extreme case, with $\lambda = 1$, the model is only learning to discriminate between genuine signatures and forgeries (the output is a single binary unit), and the performance is still reasonable, although worse than the performance when both loss functions are combined. It is worth noting that the scale of L_c is larger than L_f by definition: L_c is a cross-entropy loss among 531 users. A random classifier would have loss $L_c \approx \log(531) \approx 6.27$. On the other hand, L_f is a cross-entropy loss among 2 alternatives, and a random classifier would have loss around $L_f \approx \log(2) \approx 0.69$, which also partially explains larger λ values.

We noticed an unexpected behavior using loss L_2 with $\lambda = 0$. This loss function is equivalent to the loss when using only genuine signatures, but actually performed worse during our



(a) Loss L_1, Linear SVM

(b) Loss L_1, SVM with RBF kernel

(c) Loss L_2, Linear SVM

(d) Loss L_2, SVM with RBF kernel

Figure 4: Performance on the validation set ($\mathcal{V}_v$), using features learned from genuine signatures and forgeries (sec 3.2.2), as we vary the hyperparameter λ. For reference, the performance of models using features learned from genuine signatures only (sec 3.1) and using forgeries as different classes (sec 3.2.1) are also included.

Table 5: Performance of the WD classifiers on the validation set $\mathcal{V}_v$ (subset of 50 users in GPDS; Errors and Standard deviations in %)

Classifier	Formulation used to learn the features	$EER_{\text{global threshold}}$	$EER_{\text{user thresholds}}$	Mean AUC
Linear SVM	Baseline (Caffenet, layer pool5)	14.09 (+- 2.80)	10.59 (+- 2.96)	0.9453 (+- 0.0198)
	Using genuine signatures only (sec 3.1)	6.80 (+- 0.57)	3.91 (+- 0.64)	0.9876 (+- 0.0022)
	Forgeries as separate classes (sec 3.2.1)	9.45 (+- 0.51)	5.61 (+- 0.63)	0.9749 (+- 0.0028)
	Forgery neuron (sec 3.2.2, loss L_1, $\lambda = 0.999$)	7.01 (+- 0.42)	3.63 (+- 0.43)	0.9844 (+- 0.0024)
	Forgery neuron (sec 3.2.2, loss L_2, $\lambda = 0.95$)	6.09 (+- 0.29)	3.17 (+- 0.34)	0.9899 (+- 0.0017)
SVM (RBF)	Baseline (Caffenet, layer fc6)	16.20 (+- 0.94)	13.51 (+- 0.99)	0.9261 (+- 0.0054)
	Using genuine signatures only (sec 3.1)	5.93 (+- 0.43)	3.13 (+- 0.46)	0.9903 (+- 0.0018)
	Forgeries as separate classes (sec 3.2.1)	7.79 (+- 0.43)	4.10 (+- 0.41)	0.9857 (+- 0.0012)
	Forgery neuron (sec 3.2.2, loss L1, $\lambda = 1$)	2.41 (+- 0.32)	1.08 (+- 0.36)	0.9978 (+- 0.0008)
	Forgery neuron (sec 3.2.2, loss L2, $\lambda = 0.999$)	2.51 (+- 0.33)	1.04 (+- 0.31)	0.9971 (+- 0.0009)

experiments. Analyzing this abnormal behavior, we identified that, although the forgeries do not contribute to the loss function directly, they do have some indirect effect on loss function due to the usage of batch normalization. During training, the skilled forgeries are used, together with genuine signatures, when computing the batch statistics (mean and variance), therefore affecting the output of the network. However, it is unclear why this effect results in worse performance, instead of simply adding more variance to the results.

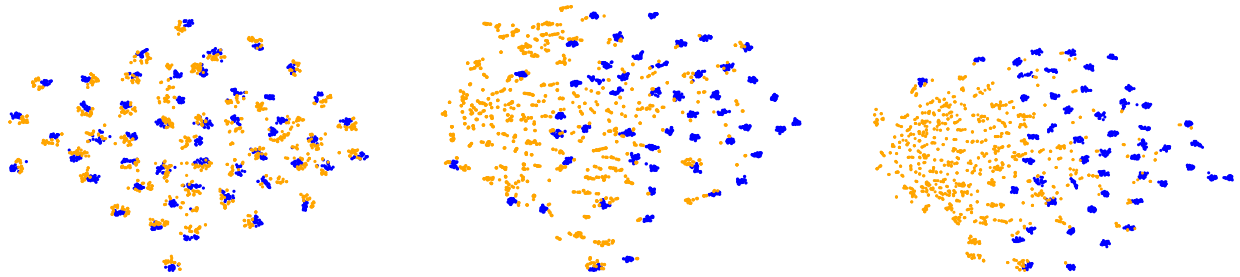
We also verified if the forgery neuron generalized well to other users. Since this neuron is not related to a particular user in the development set, we can use it to estimate $P(f|X)$ for signature images from other users. In this case, we estimate if a signature is a forgery only by looking at the questioned specimen, and not comparing it to other genuine signatures from the same user. We used the neuron trained with loss L_2 and $\lambda = 0.999$ to classify all signatures from the validation set $\mathcal{V}_v$, achieving an error rate of 14.37%. In comparison, for classifying signatures from the same set of users where the CNN was trained (i.e. testing on $\mathcal{V}_c$), the model achieved 2.21% of error. This suggests that using this neuron is mostly helpful to guide the system to obtain better representations (and subsequently train WD classifiers), than to use it directly as a classifier for new samples, since it mainly generalizes to other signatures from the same users used to train the CNN.

Table 5 consolidates the performance obtained in the validation set $\mathcal{V}_v$ using the proposed methods. The baseline, using a CNN pre-trained on the ImageNet dataset, performed reasonably well compared to previous work on the GPDS dataset, but still much worse than the methods that learned on signature data. An interesting result is that the naive formulation to use forgeries (treat forgeries as separate classes - section 3.2.1) performed worse than the formulation that used only genuine signatures for training the CNN. Using the model trained with genuine signatures, we obtained EER of 3.91% using a linear SVM, and 3.13% using the RBF kernel. Using the model trained with forgeries as separate classes, we obtained EER of 5.61% using Linear SVM and 4.10% using the RBF kernel. A possible explanation for this effect is that this formulation effectively doubles the number of classes, making the classification problem much harder. This fact, combined with the observation that genuine signatures and forgeries for the same user usually share several characteristics, may justify this drop in performance. On the other hand, the formulation using the forgery neuron performed much better in the validation set, showing that this is a promising formulation of the problem. We reiterate that forgeries are used only in the feature learning process, and that no forgeries from the validation set $\mathcal{V}_v$ were used for training.

Although it is not the focus of this paper, we note that these models could also be used for user identification from signatures. Using the features learned from genuine signatures only (sec 3.1), the performance on the validation set $\mathcal{V}_c$ (classification between the 531 users) is 99.23%, showing that using CNNs for this task is very effective.

5.1.1. Visualizing the learned representation space

We performed an analysis of the feature space learned by the models, by using the t-SNE algorithm [39] to project the samples from the validation set $\mathcal{V}_v$ from $\mathbb{R}^N$ to $\mathbb{R}^2$. This analysis is useful to examine the local structure present in this high-dimensionality space. For this analysis, we used the baseline model (Caffenet, using features from layer pool5), a model learned with genuine signatures only, and a model learned with genuine signatures and forgeries (using loss L_2 and $\lambda = 0.95$). These models were trained on the set $\mathcal{L}_c$, which is a disjoint set of users from the validation set. In all cases, we used the models to "extract features" from all 1200 signatures images from the validation set, by performing forward propagation until the layer specified above.



(a) Baseline (Features learned on Imagenet)

(b) Using only genuine signatures to learn the features

(c) Using genuine signatures and forgeries to learn the features

Figure 5: 2D projections (using t-SNE) of the feature vectors from the 50 users in the validation set $\mathcal{V}_v$. Each point represents a signature sample: genuine signatures are displayed in blue (dark), while skilled forgeries are displayed in orange (light).

For the baseline model, this representation is in $\mathbb{R}^{9216}$, while for the other models it is in $\mathbb{R}^{2048}$. For each model, we used the t-SNE algorithm to project the samples to 2 dimensions.

The result can be seen in Figure 5. The baseline system (model trained on natural images) projects the samples onto a space where samples from different users are clustered in separate regions of the space, which is is quite interesting considering that this network was never presented signature images. On the other hand, skilled forgeries are also clustered together with genuine signatures in this representation. On the models trained with signature data, we can see that signatures from different users also occupy different regions of the feature space. Using the model trained with genuine signatures and forgeries, we see that the forgeries from the users in the validation set are much more grouped together in a part of the feature space, although several forgeries are still close to the genuine signatures of the users. This suggests that the network has learned characteristics that are intrinsic to many forgeries, that generalizes to other users.

5.2. Generalization performance and comparison with the state-of-the-art

We now present the results on the exploitation set, comparing the results with the state-of-the-art. In these experiments, we do not use any skilled forgeries from the users, since it is not reasonable to expect skilled forgeries to be available for all users enrolled in the system.

We reiterate that all design decisions (e.g. choice of architecture and other hyperparameters) were done using the validation set $\mathcal{V}_v$, which consists of a separate set of users, to present an unbiased estimate of the performance of the classifier in the testing set. In these experiments, we

Table 6: Detailed performance of the WD classifiers on the GPDS-160 and GPDS-300 datasets (Errors and Standard Deviations in %)

Dataset	Samples per user	Classifier	FRR	FAR_random	FAR_skilled	$EER_{\text{global threshold}}$	$EER_{\text{user thresholds}}$	meanAUC
GPDS-160	5	SVM (Linear)	9.09 (+- 0.65)	0.01 (+- 0.03)	5.75 (+- 0.12)	7.30 (+- 0.35)	3.52 (+- 0.28)	0.9880 (+- 0.0013)
		SVM (RBF)	5.16 (+- 0.41)	0.06 (+- 0.04)	5.17 (+- 0.17)	5.15 (+- 0.22)	2.41 (+- 0.12)	0.9924 (+- 0.0011)
	12	SVM (Linear)	6.39 (+- 0.67)	0.01 (+- 0.02)	3.96 (+- 0.18)	5.15 (+- 0.28)	2.60 (+- 0.39)	0.9922 (+- 0.0010)
		SVM (RBF)	3.59 (+- 0.23)	0.02 (+- 0.03)	3.66 (+- 0.15)	3.61 (+- 0.07)	1.72 (+- 0.15)	0.9952 (+- 0.0006)
GPDS-300	5	SVM (Linear)	9.28 (+- 0.36)	0.01 (+- 0.02)	8.18 (+- 0.23)	8.68 (+- 0.22)	4.84 (+- 0.26)	0.9792 (+- 0.0016)
		SVM (RBF)	6.03 (+- 0.45)	0.04 (+- 0.04)	4.68 (+- 0.18)	5.25 (+- 0.15)	2.42 (+- 0.24)	0.9923 (+- 0.0007)
	12	SVM (Linear)	6.80 (+- 0.31)	0.00 (+- 0.01)	6.16 (+- 0.17)	6.44 (+- 0.17)	3.56 (+- 0.18)	0.9857 (+- 0.0010)
		SVM (RBF)	3.94 (+- 0.29)	0.02 (+- 0.02)	3.53 (+- 0.11)	3.74 (+- 0.15)	1.69 (+- 0.18)	0.9951 (+- 0.0004)

used the architectures that performed best in the validation set, as seen in Table 5. In particular, we consider a model that was learned using genuine signatures only (sec 3.1), which we call simply by **SigNet** in this section. We also consider a model learned using genuine signatures and forgeries (sec 3.2.2), using loss $L2$, which we call **SigNet-F**. For the experiments with a linear SVM, we used the model learned with $\lambda = 0.95$, while for the experiments with the SVM with the RBF kernel, we used the model learned with $\lambda = 0.999$.

5.2.1. Experiments on GPDS-160 and GPDS-300

For these experiments, we used the models SigNet and SigNet-F to extract features of the exploitation set (GPDS-160 and GPDS-300), and trained Writer-Dependent classifiers. To report the False Rejection Rate and False Acceptance Rates, we used the validation set to find the optimum global threshold (the threshold that obtained $EER_{\text{global threshold}}$ on the validation set $\mathcal{V}_v$) as a global threshold for all users. In this work, we do not explore techniques for setting user-specific thresholds, but simply report $EER_{\text{user thresholds}}$, which is the equal error rate obtained by using the optimal decision thresholds for each user.

Table 6 lists the detailed results on the GPDS-160 and GPDS-300 datasets, for experiments using SigNet-F. We notice that the using only 5 samples per user already achieves a good average performance on these datasets, showing that the proposed strategy works well with low number of samples per user. We also note that the performance using user-specific thresholds is much better than using a single global threshold (1.72% vs 3.61%) in the GPDS-160 dataset, which is consistent with previous findings that the definition of user-specific thresholds is key in obtaining

Table 7: Comparison with state-of-the art on the GPDS dataset (errors in %)

Reference	Dataset	#samples per user	Features	EER
Hu and Chen [5]	GPDS-150	10	LBP, GLCM, HOG	7.66
Guerbai et al [40]	GPDS-160	12	Curvelet transform	15.07
Serdouk et al [41]	GPDS-100	16	GLBP, LRF	12.52
Yilmaz [4]	GPDS-160	5	LBP, HOG, SIFT	7.98
Yilmaz [4]	GPDS-160	12	LBP, HOG, SIFT	6.97
Soleimani et al [20]	GPDS-300	10	LBP	20.94
Present Work	GPDS-160	5	SigNet	3.23 (+-0.36)
Present Work	GPDS-160	12	SigNet	2.63 (+-0.36)
Present Work	GPDS-300	5	SigNet	3.92 (+-0.18)
Present Work	GPDS-300	12	SigNet	3.15 (+-0.18)
Present Work	GPDS-160	5	SigNet-F	2.41 (+-0.12)
Present Work	**GPDS-160**	**12**	**SigNet-F**	**1.72 (+-0.15)**
Present Work	GPDS-300	5	SigNet-F	2.42 (+-0.24)
Present Work	**GPDS-300**	**12**	**SigNet-F**	**1.69 (+-0.18)**

a good performance.

We notice that the performance using a linear classifier (Linear SVM) is already good, which is interesting from a practical perspective for a large-scale deployment. Since the CNN model is the same for all users, adding new users to the system requires only training the WD classifier. For a linear classifier, this requires only one weight per dimension (plus a bias term), adding to 2049 doubles to be stored (16KB per user). For the SVM with RBF kernel, the storage requirements for each user depends on the number of support vectors. In the GPDS-300 dataset, in average the classifiers used 75 support vectors. Since the set of random forgeries is the same for all users, most of these support vectors will be shared among different users. On the other hand, we noticed that the majority of genuine signatures were selected as support vectors (as expected) - in average 10.3 genuine signatures, when using 12 references for training.

Table 7 compares our results with the state-of-the-art on the GPDS dataset. We observed a large improvement in verification performance, obtaining 1.72% EER on GPDS-160, compared to a state-of-the-art of 6.97%, both using 12 samples per user for training. We also note that this result is obtained with a single classifier, while the best results in the state-of-the-art use ensembles of many classifiers. As in the experiments in the validation set, we notice an improvement in

Table 8: Comparison with the state-of-the-art in MCYT

Reference	# Samples	Features	EER
Gilperez et al.[42]	5	Contours (chi squared distance)	10.18
Gilperez et al.[42]	10	Contours (chi squared distance)	6.44
Wen et al.[43]	5	RPF (HMM)	15.02
Vargas et al.[44]	5	LBP (SVM)	11.9
Vargas et al.[44]	10	LBP (SVM)	7.08
Ooi et al[45]	5	DRT + PCA (PNN)	13.86
Ooi et al[45]	10	DRT + PCA (PNN)	9.87
Soleimani et al.[20]	5	HOG (DMML)	13.44
Soleimani et al.[20]	10	HOG (DMML)	9.86
Proposed	5	SigNet (SVM)	3.58 (+- 0.54)
Proposed	**10**	**SigNet (SVM)**	**2.87 (+- 0.42)**
Proposed	5	SigNet-F (SVM)	3.70 (+- 0.79)
Proposed	10	SigNet-F (SVM)	3.00 (+- 0.56)

Table 9: Comparison with the state-of-the-art in CEDAR

Reference	# Samples	Features	AER/EER
Chen and Srihari[46]	16	Graph Matching	7.9
Kumar et al.[47]	1	morphology (SVM)	11.81
Kumar et al.[48]	1	Surroundness (NN)	8.33
Bharathi and Shekar[49]	12	Chain code (SVM)	7.84
Guerbai et al.[40]	4	Curvelet transform (OC-SVM)	8.7
Guerbai et al.[40]	8	Curvelet transform (OC-SVM)	7.83
Guerbai et al.[40]	12	Curvelet transform (OC-SVM)	5.6
Proposed	4	SigNet (SVM)	5.87 (+- 0.73)
Proposed	8	SigNet (SVM)	5.03 (+- 0.75)
Proposed	12	SigNet (SVM)	4.76 (+- 0.36)
Proposed	4	SigNet-F (SVM)	5.92 (+- 0.48)
Proposed	8	SigNet-F (SVM)	4.77 (+- 0.76)
Proposed	**12**	**SigNet-F (SVM)**	**4.63 (+- 0.42)**

performance using SigNet-F to extract the features compared to using SigNet.

5.2.2. Generalizing to other datasets

We now consider the generalization performance of the features learned in GPDS to other datasets. We use the same networks, namely SigNet and SigNet-F, for extracting features and training Writer-Dependent classifiers on MCYT, CEDAR and the Brazilian PUC-PR datasets.

Tables 8, 9 and 10 present the comparison with the state-of-the-art performance on MCYT, CEDAR and Brazilian PUC-PR, respectively. In all datasets we notice improvement in performance compared to the state-of-the-art, suggesting that the features learned on GPDS generalize

Table 10: Comparison with the state-of-the-art on the Brazilian PUC-PR dataset (errors in %)

Reference	#samples per user	Features	FRR	FAR_{random}	FAR_{simple}	$FAR_{skilled}$	AER	$AER_{genuine + skilled}$	$EER_{genuine + skilled}$
Bertolini et al. [8]	15	Graphometric	10.16	3.16	2.8	6.48	5.65	8.32	-
Batista et al. [50]	30	Pixel density	7.5	0.33	0.5	13.5	5.46	10.5	-
Rivard et al. [6]	15	ESC + DPDF	11	0	0.19	11.15	5.59	11.08	-
Eskander et al. [7]	30	ESC + DPDF	7.83	0.02	0.17	13.5	5.38	10.67	-
Present Work	5	SigNet	4.63 (+- 0.55)	0.00 (+- 0.00)	0.35 (+- 0.20)	7.17 (+- 0.51)	3.04 (+- 0.17)	5.90 (+- 0.32)	2.92 (+- 0.44)
Present Work	15	SigNet	1.22 (+- 0.63)	0.02 (+- 0.05)	0.43 (+- 0.09)	10.70 (+- 0.39)	3.09 (+- 0.20)	5.96 (+- 0.40)	2.07 (+- 0.63)
Present Work	**30**	**SigNet**	0.23 (+- 0.18)	0.02 (+- 0.05)	0.67 (+- 0.08)	12.62 (+- 0.22)	3.38 (+- 0.06)	6.42 (+- 0.13)	**2.01 (+- 0.43)**
Present Work	5	SigNet-F	17.17 (+- 0.68)	0.00 (+- 0.00)	0.03 (+- 0.07)	2.72 (+- 0.37)	4.98 (+- 0.16)	9.94 (+- 0.31)	5.11 (+- 0.89)
Present Work	15	SigNet-F	9.25 (+- 0.88)	0.00 (+- 0.00)	0.25 (+- 0.09)	6.55 (+- 0.37)	4.01 (+- 0.24)	7.90 (+- 0.46)	4.03 (+- 0.59)
Present Work	30	SigNet-F	5.47 (+- 0.46)	0.00 (+- 0.00)	0.38 (+- 0.11)	8.80 (+- 0.44)	3.66 (+- 0.12)	7.13 (+- 0.25)	3.44 (+- 0.37)

well to signatures from other datasets (with different protocols for signature acquisition, created with different users in different countries). We also note that other methods proposed in the literature often present better performance only in one dataset, for instance, Guerbai et al. [40] obtained good results on CEDAR, but poor results on GPDS; Soleimani et al. [20] obtained good results on MCYT, but not on GPDS. The proposed method, however, obtained state-of-the-art performance in all datasets. For MCYT we obtained EER of 2.87% compared to 6.44% in the literature. On CEDAR, we obtained EER of 4.63%, compared to 5.6%. For the Brazilian PUC-PR dataset, we notice an improvement in performance both in terms of average error rate (considering all types of forgery), and the average error rate comparing only genuine signatures and skilled forgeries. It is worth noting that in these experiments we used a global threshold = 0 to report FRR and FAR, since we did not have a validation set to learn the appropriate global threshold, hence the large differences between FRR and $FAR_{skilled}$.

We also noticed that the formulation that learned features using skilled forgeries from the GPDS dataset did not perform better in all cases. For MCYT and CEDAR the performance between SigNet and SigNet-F was not significantly different, whereas for the Brazilian PUC-PR dataset it obtained worse performance than SigNet. This suggests that the representation may have specialized to traits present in the forgeries made for the GPDS dataset, which depend on the acquisition protocol, such as if only one type of writing instrument was used, and the directions given to participants to create the forgeries. We note, however, that 1920 people participated in creating forgeries for the GPDS dataset [9].

Finally, considering that the MCYT dataset contains both an Offline dataset (with static signature images, as used in this paper), and an Online version (with dynamic information of the strokes), it is possible to compare the two approaches to the problem. In the literature, online signature verification systems empirically demonstrate better performance than offline systems [2], which is often attributed to the lack of dynamic information of the signature writing process in the offline signatures. The gains in performance using the method proposed in this paper reduce the gap between the two approaches. Using offline signatures, we obtained 2.87 % $EER_{\text{user thresholds}}$ using 10 samples per user. Using online data, the best results reported in the literature achieve 2.85 % EER [51] and 3.36 % EER [52], also using 10 samples per user. We note, however, that in our work we do not address the issue of selecting user-specific thresholds (or performing user-specific score normalization), which is left as future work. In constrast, both [51] and [52] use score normalization, followed by a single global threshold, so the comparison of these papers to our work is not direct.

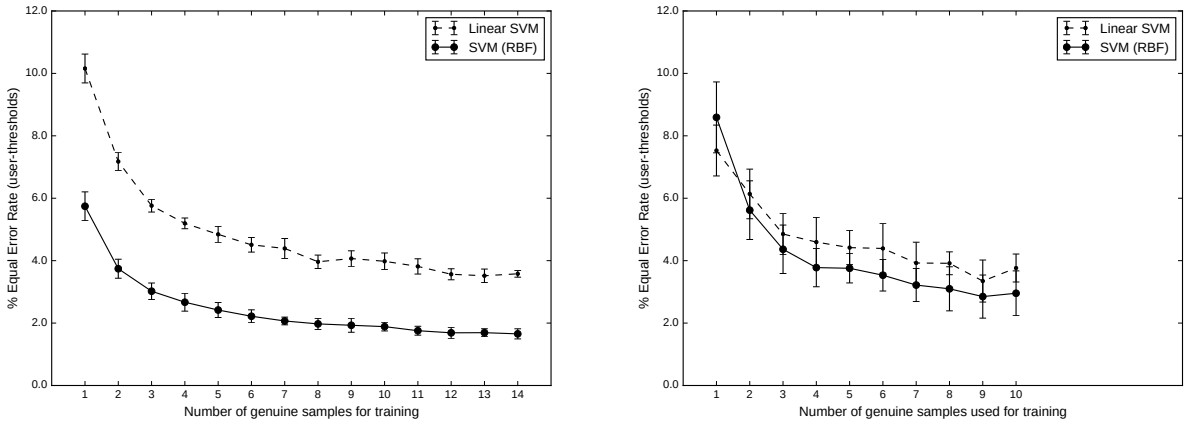
5.2.3. Varying the number of genuine samples available for training

Figure 6 shows the improvement in performance on the four datasets as we obtain more samples per user for training. Each point represents the performance of the WD classifiers trained with a given number of genuine samples (mean and standard deviation across 10 replications). As in previous work ([7], [10]), we notice diminishing returns as we collect more samples for each user. It is worth noting that in the GPDS dataset, even with a single sample per user we obtain 5.74% EER, which surpasses the state-of-the-art system that used 12 samples per user, showing that good feature representations are indeed critical to obtain good performance.

6. Conclusion

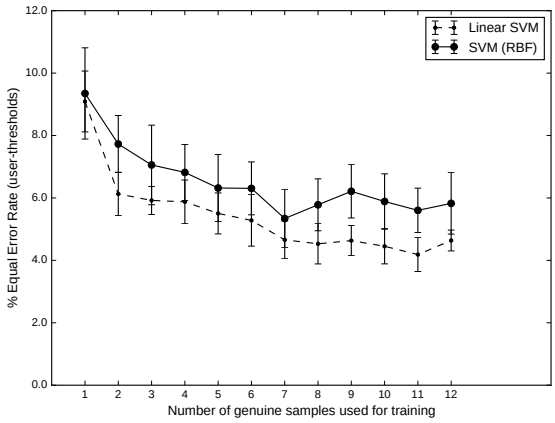
In this work, we presented different formulations for learning representations for offline signature verification. We showed that features learned in a writer-independent way can be very effective for signature verification, improving performance on the task, compared to the methods that rely on hand-engineered features.

In particular, we showed a formulation of the problem to take advantage of having forgery

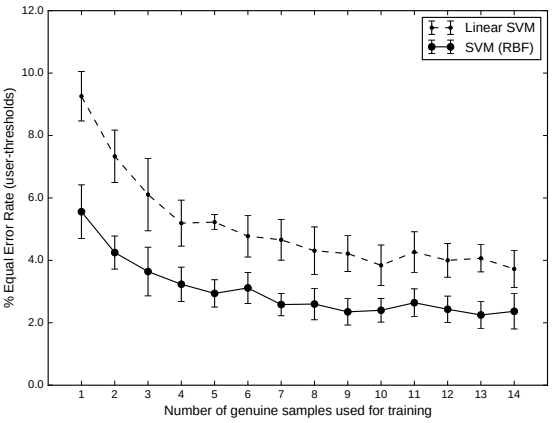


(a) GPDS-300

(b) MCYT



(c) CEDAR



(d) Brazilian PUC-PR

Figure 6: Average performance of the Writer-Dependent classifiers for each dataset, as we vary the number of genuine signatures (per user) available for training.

data from a subset of users, so that the learned features perform better in distinguishing forgeries for unseen users. With this formulation, we obtain an EER or 1.72% in the GPDS-160 dataset, compared to 6.97% reported in the literature. The visual analysis of the feature space shows that the features generalize well to unseen users, by separating genuine signatures and forgeries in different regions of the representation space. We also noted very good performance of this strategy even when few samples per user are available. For instance, with 5 samples per user, we obtained 2.41 % EER on this dataset.

The experiments with the MCYT, CEDAR and Brazilian PUC-PR datasets demonstrate that the features learned in this Writer-Independent format not only generalize to different users of the GPDS dataset, but also to users from other datasets, surpassing the state-of-the-art performance on all three. We noticed, however, that the model learned with forgeries in the GPDS dataset did not perform better in all cases, suggesting that the characteristics of forgeries in the datasets may be different - this will be further studied in future work. Another promising research direction is the combination of online and offline signature verification methods. This can improve robustness of the system since it becomes harder to create a forgery that is misclassified by both classifiers, that is, a forgery having similar strokes in terms of speed of execution, and at the same time that is visually similar to a genuine signature from the user.

Acknowledgments

This work was supported by the CNPq grant #206318/2014-6 and by grant RGPIN-2015-04490 to Robert Sabourin from the NSERC of Canada.

References

References

[1] R. Plamondon, S. N. Srihari, Online and off-line handwriting recognition: a comprehensive survey 22 (1) 63–84. doi:10.1109/34.824821.

[2] D. Impedovo, G. Pirlo, Automatic signature verification: The state of the art 38 (5) 609–635. doi:10.1109/TSMCC.2008.923866.

[3] L. G. Hafemann, R. Sabourin, L. S. Oliveira, Offline Handwritten Signature Verification-Literature Review, arXiv preprint arXiv:1507.07909.

[4] M. B. Yilmaz, B. Yanikoglu, Score level fusion of classifiers in off-line signature verification, Information Fusion 32, Part B (2016) 109–119. doi:10.1016/j.inffus.2016.02.003.

[5] J. Hu, Y. Chen, Offline Signature Verification Using Real Adaboost Classifier Combination of Pseudo-dynamic Features, in: Document Analysis and Recognition, 12th International Conference on, 2013, pp. 1345–1349. doi:10.1109/ICDAR.2013.272.

[6] D. Rivard, E. Granger, R. Sabourin, Multi-feature extraction and selection in writer-independent off-line signature verification 16 (1) 83–103. doi:10.1007/s10032-011-0180-6.

[7] G. Eskander, R. Sabourin, E. Granger, Hybrid writer-independent-writer-dependent offline signature verification system, IET Biometrics 2 (4) (2013) 169–181. doi:10.1049/iet-bmt.2013.0024.

[8] D. Bertolini, L. S. Oliveira, E. Justino, R. Sabourin, Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers, Pattern Recognition 43 (1) (2010) 387–396. doi:10.1016/j.patcog.2009.05.009.

[9] J. Vargas, M. Ferrer, C. Travieso, J. Alonso, Off-line Handwritten Signature GPDS-960 Corpus, in: Document Analysis and Recognition, 9th International Conference on, Vol. 2, 2007, pp. 764–768. doi:10.1109/ICDAR.2007.4377018.

[10] L. G. Hafemann, R. Sabourin, L. S. Oliveira, Writer-independent feature learning for offline signature verification using deep convolutional neural networks, in: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 2576–2583. doi:10.1109/IJCNN.2016.7727521.

[11] L. G. Hafemann, L. S. Oliveira, R. Sabourin, Analyzing features learned for offline signature verification using Deep CNNs, in: 23rd International Conference on Pattern Recognition, 2016.

[12] R. Plamondon, G. Lorette, Automatic signature verification and writer identification âĂŤ the state of the art 22 (2) 107–131. doi:10.1016/0031-3203(89)90059-9.

[13] F. Leclerc, R. Plamondon, Automatic signature verification: The state of the art - 1989-1993 08 (3) 643–660. doi:10.1142/S0218001494000346.

[14] R. N. Nagel, A. Rosenfeld, Computer detection of freehand forgeries C-26 (9) 895–905. doi:10.1109/TC.1977.1674937.

[15] E. J. Justino, A. El Yacoubi, F. Bortolozzi, R. Sabourin, An off-line signature verification system using HMM and graphometric features, in: Fourth IAPR International Workshop on Document Analysis Systems (DAS), Rio de, Citeseer, 2000, pp. 211–222.

[16] L. S. Oliveira, E. Justino, C. Freitas, R. Sabourin, The graphology applied to signature verification, in: 12th Conference of the International Graphonomics Society, 2005, pp. 286–290.

[17] R. Sabourin, J.-P. Drouhard, Off-line signature verification using directional PDF and neural networks, in: ,

11th IAPR International Conference on Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings, 1992, pp. 321–325. doi:10.1109/ICPR.1992.201782.

[18] B. Ribeiro, I. Goncalves, S. Santos, A. Kovacec, Deep learning networks for off-line handwritten signature recognition, in: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, no. 7042 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 523–532, DOI: 10.1007/978-3-642-25085-9_62.

[19] H. Khalajzadeh, M. Mansouri, M. Teshnehlab, Persian Signature Verification using Convolutional Neural Networks, in: International Journal of Engineering Research and Technology, Vol. 1, ESRSA Publications, 2012.

[20] A. Soleimani, B. N. Araabi, K. Fouladi, Deep Multitask Metric Learning for Offline Signature Verification, Pattern Recognition Letters 80 (2016) 84–90. doi:10.1016/j.patrec.2016.05.023.

[21] Y. Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning 2 (1) (2009) 1–127. doi:10.1561/2200000006.

[22] Y. Bengio, Deep Learning of Representations: Looking Forward, in: Statistical Language and Speech Processing, no. 7978 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 1–37.

[23] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. doi:10.1038/nature14539.

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition 1 (4) 541–551. doi:10.1162/neco.1989.1.4.541.

[25] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: Advances in Neural Information Processing Systems 25, 2012, pp. 1097–1105.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[27] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: P. B. Schölkopf, J. C. Platt, T. Hoffman (Eds.), Advances in Neural Information Processing Systems 19, MIT Press, 2007, pp. 153–160.

[28] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, arXiv:1310.1531 [cs]ArXiv: 1310.1531.

[29] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks, in: Computer Vision and Pattern Recognition, IEEE Conference on, 2014, pp. 1717–1724. doi:10.1109/CVPR.2014.222.

[30] L. Nanni, S. Ghidoni, How could a subcellular image, or a painting by Van Gogh, be similar to a great white shark or to a pizza?, Pattern Recognition Letters 85 (2017) 1–7. doi:10.1016/j.patrec.2016.11.011.

[31] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man, and

Cybernetics 9 (1) 62–66. doi:10.1109/TSMC.1979.4310076.

[32] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in: Proceedings of The 32nd International Conference on Machine Learning, 2015, pp. 448–456.

[33] E. Osuna, R. Freund, F. Girosi, Support Vector Machines: Training and Applications.

[34] J. Ortega-Garcia, J. Fierrez-Aguilar, D. Simon, J. Gonzalez, M. Faundez-Zanuy, V. Espinosa, A. Satue, I. Hernaez, J.-J. Igarza, C. Vivaracho, others, MCYT baseline corpus: a bimodal biometric database, IEE Proceedings-Vision, Image and Signal Processing 150 (6) (2003) 395–401.

[35] M. K. Kalera, S. Srihari, A. Xu, Offline signature verification and identification using distance statistics, International Journal of Pattern Recognition and Artificial Intelligence 18 (07) (2004) 1339–1360. doi:10.1142/S0218001404003630.

[36] C. Freitas, M. Morita, L. Oliveira, E. Justino, A. Yacoubi, E. Lethelier, F. Bortolozzi, R. Sabourin, Bases de dados de cheques bancarios brasileiros, in: XXVI Conferencia Latinoamericana de Informatica, 2000.

[37] A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN Features off-the-shelf: an Astounding Baseline for Recognition, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, IEEE, 2014, pp. 512–519.

[38] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556 [cs]ArXiv: 1409.1556.

[39] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (2579-2605) (2008) 85.

[40] Y. Guerbai, Y. Chibani, B. Hadjadji, The effective use of the one-class SVM classifier for handwritten signature verification based on writer-independent parameters, Pattern Recognition 48 (1) (2015) 103–113. doi:10.1016/j.patcog.2014.07.016.

[41] Y. Serdouk, H. Nemmour, Y. Chibani, New gradient features for off-line handwritten signature verification, in: 2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA), 2015, pp. 1–4. doi:10.1109/INISTA.2015.7276751.

[42] A. Gilperez, F. Alonso-Fernandez, S. Pecharroman, J. Fierrez, J. Ortega-Garcia, Off-line signature verification using contour features, in: 11th International Conference on Frontiers in Handwriting Recognition, Montreal, Quebec-Canada, August 19-21, 2008, CENPARMI, Concordia University, 2008.

[43] J. Wen, B. Fang, Y. Y. Tang, T. Zhang, Model-based signature verification with rotation invariant features, Pattern Recognition 42 (7) (2009) 1458–1466. doi:10.1016/j.patcog.2008.10.006.

[44] J. F. Vargas, M. A. Ferrer, C. M. Travieso, J. B. Alonso, Off-line signature verification based on grey level information using texture features, Pattern Recognition 44 (2) (2011) 375–385. doi:10.1016/j.patcog.2010.07.028.

[45] S. Y. Ooi, A. B. J. Teoh, Y. H. Pang, B. Y. Hiew, Image-based handwritten signature verification using hybrid

methods of discrete radon transform, principal component analysis and probabilistic neural network 40 274–282. doi:10.1016/j.asoc.2015.11.039.

[46] S. Chen, S. Srihari, A New Off-line Signature Verification Method based on Graph, in: 18th International Conference on Pattern Recognition (ICPR'06), Vol. 2, 2006, pp. 869–872. doi:10.1109/ICPR.2006.125.

[47] R. Kumar, L. Kundu, B. Chanda, J. D. Sharma, A Writer-independent Off-line Signature Verification System Based on Signature Morphology, in: Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia, IITM'10, ACM, New York, NY, USA, 2010, pp. 261–265. doi:10.1145/1963564.1963610.

[48] R. Kumar, J. D. Sharma, B. Chanda, Writer-independent off-line signature verification using surroundedness feature, Pattern Recognition Letters 33 (3) (2012) 301–308. doi:10.1016/j.patrec.2011.10.009.

[49] R. Bharathi, B. Shekar, Off-line signature verification based on chain code histogram and Support Vector Machine, in: 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2013, pp. 2063–2068. doi:10.1109/ICACCI.2013.6637499.

[50] L. Batista, E. Granger, R. Sabourin, Dynamic selection of generative-discriminative ensembles for off-line signature verification, Pattern Recognition 45 (4) (2012) 1326–1340. doi:10.1016/j.patcog.2011.10.011.

[51] E. A. Rua, J. L. A. Castro, Online Signature Verification Based on Generative Models, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (4) (2012) 1231–1242. doi:10.1109/TSMCB.2012.2188508.

[52] J. Fierrez, J. Ortega-Garcia, D. Ramos, J. Gonzalez-Rodriguez, HMM-based on-line signature verification: Feature extraction and signature modeling, Pattern Recognition Letters 28 (16) (2007) 2325–2334. doi:10.1016/j.patrec.2007.07.012.