An Offline Writer-Independent Signature Verification Method with Robustness Against Scalings and Rotations

Felix Eduardo Huaroto Pachas
Instituto de Informática – UFRGS
Porto Alegre, RS, Brazil
fehpachas@inf.ufrgs.br

Eduardo S. L. Gastal
Instituto de Informática – UFRGS
Porto Alegre, RS, Brazil
eslgastal@inf.ufrgs.br

***Abstract*—Handwritten signatures are still one of the most used and accepted methods for user identification and authentication. They are used in a wide range of human daily tasks, including applications from banking to legal processes. The *signature verification* problem consists of verifying whether a given handwritten signature was generated by a particular person, by comparing it (directly or indirectly) to genuine signatures from that person.**

In this paper, we introduce a new offline writer-independent signature verification method based on a combination of handcrafted Moving Least-Squares features and features transferred from a convolutional neural network. In our experiments, our method outperforms state-of-the-art techniques on Western-style signatures (CEDAR dataset), while also obtaining good results on South Asian-style handwriting (Bangla and Hindi datasets). Furthermore, we demonstrate that the proposed method is the most robust in relation to differences in scale and rotation of the signature images. We also present a discussion on dataset bias and a small user study, showing that our technique outperforms the expected human accuracy on the signature-verification task.

I. Introduction

Signature Verification consists of verifying whether a particular handwritten signature is genuine (i.e., was generated by a particular person of interest) or if it is a forgery (i.e., was generated by someone else, maybe trying to mimic the genuine signature of the first person, in order to impersonate them). This problem can be approached using either *online* or *offline* information [1]. In the online approach, signature information is collected using a helper device such as a tablet, a special pen or handwriting digitizer, which collects writing-time information about the signature, such as: x and y coordinates for each signature point (at each moment in time), total time employed by the writer to perform the signature, pen inclination, applied pressure, etc. On the other hand, in the offline approach the signature is only available as a static image, normally a scanned piece of paper containing the handwritten signature (Figure 2a). Offline signature verification is a more difficult problem to solve [1], since less information about the signature is available [2]. Nonetheless, it is still the most needed approach, since many legal and banking processes base their legality on the verification of the truthfulness of handwritten signatures.

Signature verification methods can be classified as either *writer dependent* or *writer independent* [3]. A writer-dependent technique builds a distinct and unique classification model for each person (writer), which is then used to distinguish genuine from forgery signatures for that specific person. On the other hand, a writer-independent technique uses a single classification model to deal with signatures from any person. This is usually performed by operating on top of *signature pairs* [4]; that is, given a pair of images (each image containing a single handwritten signature), the classification model classifies the pair as either genuine (if both signatures are believed to be written by the same person), or as a forgery (if the signatures seem to be from two different writers). Writer-independent models have the benefit of being able to work with new signature examples or new writers without requiring retraining or the generation of new person-specific models [3]. Furthermore, since they are composed of a single classifier trained on signature-pair examples of several writers, writer-independent models perform well even when a small number of signatures is available per writer [5].

In this work, we present a new offline writer-independent technique for signature verification. As shown by our experiments (Table II), our approach significantly outperforms state-of-the-art techniques on the CEDAR dataset (Western-style handwriting) [2], while performing on-par with the state-of-the-art on the Bangla and Hindi datasets (South Asian-style handwriting) [6]. Furthermore, our proposed method outperforms the listed state-of-the-art techniques when one considers the more challenging situation where the signatures are subjected to differences in scale and rotation. As such, our method is more robust against data variability (Section IV-C).

Our approach works by describing each signature image by a set of hybrid features: some handcrafted and some obtained from a pretrained CLIP neural network [7]. We design our handcrafted features based on Moving Least-Squares goodness-of-fit, with simplicity and robustness in mind. Furthermore, we propose an alternative pipeline for obtaining the CLIP features for each signature image, and we validate our choices with an ablation study (Table III). For classification we use a simple linear Support Vector Machine (SVM) classifier.

The **contributions** of our work include:

- A new offline writer-independent signature verification method, which outperforms the state-of-the-art in our

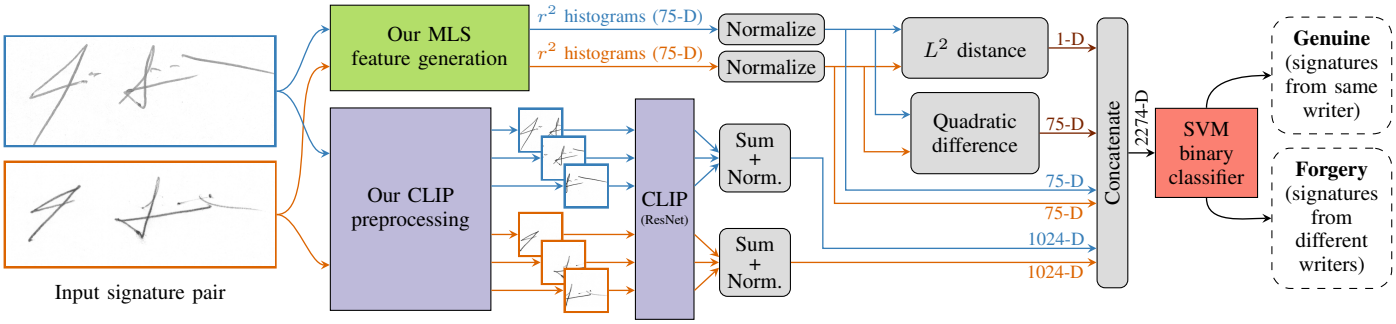


Fig. 1. Overview of our offline writer-independent technique for signature verification. Given a pair of signature images (left), our method generates features based on a Moving Least-Squares strategy (Section III-A), in addition to CNN-transferred features (Section III-B). The resulting 2274-D feature vector for the pair is fed to a binary SVM classifier, which distinguishes between genuine and forged signature pairs. Signature images provided by CEDAR [2].

experiments and is robust against changes in scale and rotation of the signature images. Our method uses a novel set of handcrated features based on the idea of Moving Least-Squares (MLS) [8].

- A discussion of unintended bias on signature-verification datasets, and a proposal of how to remove such bias (defining new Unbiased datasets), supported by experiments;
- The proposal of variations on existing datasets considering scalings and rotations of the images, and experiments on scale and rotation invariance of the signature-verification methods and of humans.

II. Background and Related Work

A lot of research has been done in the field of signature verification, either using online or offline approaches, as well as using writer-dependent or writer-independent models [1], [3]. Most of this research effort has been focused in extracting good feature-representations of an individual signature, in order to discriminate effectively between genuine signatures and forgeries.

A. Types of Forgeries

A signature is a forgery when it is generated by a person (the forger) that is trying to impersonate someone else. Forgeries can be ascribed to one of three categories, in order of decreasing complexity: skilled, unskilled, or random [3]. A *skilled forgery* is one in which the forger tries to mimic the original writer's signature after having seen it, and has time to practice the signature in order to reproduce it accurately. An *unskilled (or simple) forgery* is one where the forger does not have access to the actual signature of the original writer, but knows his or her name. Finally, a *random forgery* is produced knowing neither the original signature nor the name of the original writer (in this case signature pairs are usually generated by pairing genuine signatures from two different writers [3]).

B. Offline Writer-Independent Signature Verification

In this work we focus on offline and writer-independent signature verification, aimed specifically at distinguishing genuine signatures from skilled forgeries (the most difficult type of forgery to handle). As mentioned in the introduction, this scenario is most applicable to current practices in legal and banking processes, which still rely heavily on handwritten signatures. For online and/or writer-dependent methods, we refer the reader to recent surveys and techniques in the literature [1], [3], [9].

Most signature verification methods consider the whole signature image as source for feature extraction. For example, Kalera et al. [2] use a set of gradient, structural and concavity features to determine statistical distance distributions as discriminators. Other researchers proposed camera-based approaches [10], geometric features [11], [12], spatial distribution features (describing signature shape) [13], grid-based methods [14], texture-based features [6], and also moment-based representations with envelope characteristics and tree-structured Wavelet features [15]. Some works extract features from the signature images using neural networks in a writer-independent approach [16]–[19], while others propose hybrid methods considering handcrafted features and features learned from probabilistic neural networks [20].

One of the top-performing offline writer-independent methods is SigNet by Dey et al. [4]. It uses a Convolutional Siamese Network to extract feature information from the two images in a signature pair. Before feeding each image to the network, some preprocessing steps are required, including: resizing the images to a standard size, inverting the pixel values (so that background pixels have value zero), and "normalizing" the images by dividing the pixel values by the standard deviation of all the image pixels in the dataset. The network architecture consists of two twin networks joined by a cost function, which computes a distance metric between the 128-D feature vectors of the two images in a pair.

Since SigNet outputs a continuous numerical distance $D(s_1, s_2)$ that measures the dissimilarity between two signature images s_1 and s_2, one has to fix a threshold value τ to generate the binary classifier prediction. That is, if $D(s_1, s_2) \leq \tau$ the classifier predicts that the signatures are genuine (came from the same writer), otherwise, if $D(s_1, s_2) > \tau$, the classifier predicts that the pair (i.e., one of the signatures) is a forgery. For performance evaluation of SigNet, the authors compute metrics such as accuracy (which depends on the threshold τ) by selecting the maximum accuracy that is obtained in the test set over all possible thresholds τ [4]. According to their results, the proposed technique outperforms the state-of-the-art in datasets with a sufficient number of signatures for training,

but its accuracy is negatively affected when signatures vary too much in style and the number of signature samples is low. The authors also perform an experiment training only with unskilled forgeries, demonstrating that the results are worse than training only with skilled forgeries. The accuracy for SigNet is listed as 100% on the CEDAR dataset, 86.11% on the Bangla dataset and 84.64% on the Hindi dataset [4]. As shown by our experiments, the 100%-accuracy number is likely caused by an inherent bias in the CEDAR dataset (Section IV-B).

Dutta et al. [21] also propose a top-performing writer-independent method for offline signature verification, based on handcrafted local features and global statistics. They employ BRISK feature points, which are used as input to a Delaunay triangulation step. Their hypothesis is that, for genuine signatures, most of the edges from the triangulation remain stable, while for forged signatures they suffer distortions. Additionally, histogram of oriented gradient (HOG) descriptors are computed for each feature point (local descriptors), and for points connected by an edge in the triangulation, their HOG descriptors are concatenated (pairwise descriptors). These descriptors go through an encoding process, followed by the computation of weighted-histograms that define global image statistics (weights are defined based on the areas of signature features, from a 'water reservoir model'). Finally, a kernel function that measures the similarity between two images is defined, and an SVM is trained to perform classification. The SVM hyperparameters are optimized for each cross-validation fold based on the maximum accuracy obtained over a range of possible parameters [21].

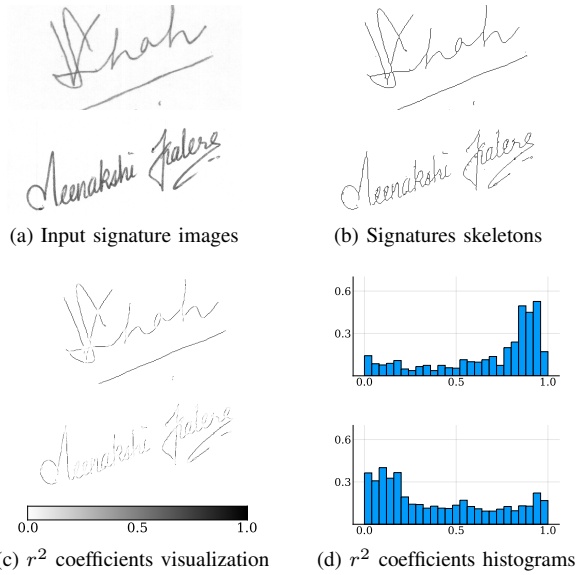
III. Proposed Method

Our method receives as input a signature pair, which must be classified as either *genuine* (if both signatures belong to the same writer), or as a *forgery* (if the signatures belong to different writers). As illustrated in Figure 1, this is treated as a binary classification problem, where the pair of signature images is converted to a high-dimensional feature vector, which is then given as input to an SVM classifier.

We propose a set of hybrid features extracted from the signature images: handcrafted features obtained by a novel Moving Least-Squares (MLS) strategy (Section III-A), in addition to features obtained from a pretrained convolutional neural network (CNN) (Section III-B). No parameters require manual tuning. We train the SVM classifier in a supervised manner (Section III-C), using default hyperparameter settings.

A. Moving Least-Squares Feature Generation

Our handcrafted features are aimed at extracting geometrical and topological characteristics of the signatures, which are good discriminators for signature verification [2]. Our secondary goal is to do this in a way that is simple and robust. To do this, our idea is to quantify the occurence of pronounced curvatures and stroke-intersections in the signatures, based on the observation that forgers would need to mimic these exact characteristics in order to obtain a convincing forged signature. Furthermore, while straight or nearly straight parts of the signature may be easier for forgers to copy, the exact proportion between curved and straight regions is more difficult to imitate.



(a) Input signature images (b) Signatures skeletons

(c) r^2 coefficients visualization (d) r^2 coefficients histograms

Fig. 2. Illustration of our Moving Least-Squares (MLS) feature generation pipeline, for two example signatures from CEDAR. (a) An input signature is converted to a (b) binary skeleton, followed by (c) MLS goodness-of-fit computation (r^2 coefficients), which detects curved and stroke-intersection signature regions. The resulting (d) r^2 histograms describe the distribution of curved and straight regions of the signature. Input images from CEDAR [2].

Overview: We quantify the curvature of each location in the signature by evaluating how well it fits a straight line. More precisely, we fit a weighted least-squares line to each small neighborhood along the signature's stroke, and evaluate its goodness-of-fit using the r^2 (r-squared) coefficient [22]. Since locations with high curvature or stroke-intersections are not well described by a straight line, they will be associated with small r^2 values (Figure 2c). Furthermore, by computing histograms of all r^2 values obtained along the signature, one is able to describe the distribution between curved and straight regions of the signature (Figure 2d).

Detailed Algorithm: For each signature image, we use the following algorithm to compute a 75-D feature vector based on the idea of r^2 histograms described above:

1) Binarize the input signature image. We use Otsu's algorithm [23], which determines an optimal global binarization threshold from the image's histogram;
2) Compute the signature's skeleton from the binarized image (Figure 2b). We use the thinning algorithm of Zhang and Suen [24], since it generates good results;
3) For each $L \times L$ pixel neighborhood Ω_p centered at each point p in the signature's skeleton, we fit a straight line through all skeleton points existing in Ω_p, using a Moving Least-Squares (MLS) strategy (described below in Section III-A1). We discard all neighborhoods which contain less than 5 skeleton points, and we fix $L = 11$ pixels in our implementation (according to our experiments, the final result is not sensitive to the exact value of L, likely due to the weighting in our least-squares fits);

4) For each neighborhood Ω_p, we compute its associated r^2 goodness-of-fit coefficient from the weighted least-squares residuals (Eq. (2), below);
5) Finally, we collect all r^2 values and compute their distributions at several scales, using histograms with 5, 10, 15, 20 and 25 bins (uniformly distributed in the $[0, 1]$ interval, since $r^2 \in [0, 1]$). Figure 2d shows the 25-bin version of the r^2 histograms. By concatenating all these histograms, we obtain a 75-D feature vector for the signature image.

Given a signature pair, as shown in Figure 1(left), we compute the 75-D r^2 histograms descriptors (feature vectors) for each image in the pair. These feature vectors are then normalized by dividing them by their L^2 norm (this makes the histograms invariant to the absolute number of pixels in the signature image). Furthermore, we compute the L^2 distance between the normalized feature vectors $\vec{u}$ and $\vec{v}$ of both images (resulting in a positive scalar $d = \|\vec{u} - \vec{v}\|_{L^2}$), in addition to their quadratic difference (resulting in another 75-D vector $\vec{w}$, whose i-th component is $\vec{w}_i = (\vec{u}_i - \vec{v}_i)^2$). Concatenating all these quantities as $[\vec{u}, \vec{v}, \vec{w}, d]$ results in our final 226-D handcrafted feature descriptor for the signature pair. This will later be concatenated with the features generated by the neural network backend (Section III-B).

1) Moving Least-Squares Fit and r^2: The idea of performing a sequence of least-squares line fits with a "sliding window" is related to the concept of Moving Least-Squares [8], a technique widely used to reconstruct continuous functions from scattered data [25]. Of particular importance in this method is the use of weights in the least-squares functionals, which are normally inversely proportional to each data point's distance from the center of the window/neighborhood. We adapt this idea for our purposes by also considering pixel intensity in the weighting.

Given a particular neighborhood Ω_p and a collection of points $(x_i, y_i) \in \Omega_p \cap \mathcal{S}$ that are also part of the signature's skeleton $\mathcal{S}$ (Figure 2b), we fit a straight line $y = ax + b$ through the points by minimizing the weighted least-squares functional:

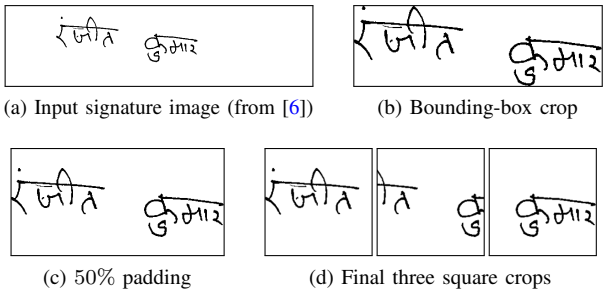
$$\mathcal{E}(a, b) = \sum_i w_i^2 \, (y_i - ax_i - b)^2. \qquad (1)$$

The weights w_i are inversely proportional to the L^2 distance between the data points (x_i, y_i) and the point $p = (x_p, y_p)$ at the center of the neighborhood Ω_p. That is, pixels that are closer to the center of the neighborhood are given more importance (*greater* weights) in the line fit. The weights are also inversely proportional to the intensities $f(x_i, y_i)$ of the pixels. Thus, darker pixels also receive greater weights, as they are more likely to be an important part of the signature's stroke: $w_i = 1/(1 + f(x_i, y_i) \, \|(x_i, y_i) - (x_p, y_p)\|_{L^2})$.

The optimal line parameters $a^\star$ and $b^\star$ that minimize $\mathcal{E}$ are the ones where $\partial\mathcal{E}(a^\star, b^\star)/\partial a = \partial\mathcal{E}(a^\star, b^\star)/\partial b = 0$. One can then compute the goodness-of-fit r^2 value as:

$$r^2 = 1 - \frac{\mathcal{E}(a^\star, b^\star)}{\sum_i w_i^2 (y_i - \bar{y})^2}, \quad \text{where} \quad \bar{y} = \frac{\sum_i w_i y_i}{\sum_i w_i}. \qquad (2)$$

For lines with slope $a^\star > 1$, we instead fit x as a function of y, with $x = my + c$ (this is done by simply swapping x and y in the equations above). This avoids numerical problems for lines that are close to vertical (where the slope $|a^\star| \to \infty$).



(a) Input signature image (from [6]) (b) Bounding-box crop

(c) 50% padding (d) Final three square crops

Fig. 3. Our proposed CLIP preprocessing pipeline (Section III-B).

B. CLIP Feature Generation

Convolutional Neural Networks (CNNs) are able to automatically learn features for distinguishing between images in classification tasks [26]. The discriminative power of CNN features also generalizes well between different problem domains, which is the idea behind transfer learning [27]. This technique allows one to use features learned in a domain containing large amounts of training data, in another domain where training data may be scarcer or harder to obtain.

In our experiments, we found that the recent CLIP network (Contrastive Language-Image Pre-training) [7] contains features which generalize well to our signature-verification problem. This CNN was trained on 400 million images with associated textual captions, with the goal of predicting which caption belongs to which image. We use the publicly available pre-trained CLIP network (with the ResNet-50 backend) to generate 1024-D feature vectors for each image in a signature pair (without retraining). However, giving the "raw" signature image as input to CLIP is not optimal due to variations in the signatures' aspect ratios, and also because signatures are scanned with varying degrees of padding (empty paper) around the signature.

We propose the following preprocessing pipeline for CLIP, illustrated in Figure 3, which significantly improves classification accuracy (Section IV-D). For each signature image:

1) Crop the image to the bounding-box of the signature. We do this by removing pixel rows and columns (around the border of the image) whose average pixel values are below the threshold 254 (considering 8-bit encoded grayscale, with pixel values in $[0, 255]$);
2) Pad the image with empty rows/columns (i.e., with white pixels), to guarantee that the smallest dimension is no smaller than 50% the size of the largest dimension. This is done in order to guarantee that the crops generated in the next step (3) cover the whole signature, while also overlapping by $\geq 50\%$;
3) Generate three square crops from the signature image. If the image has width $>$ height, we extract crops aligned to the left, center, and right of the image. Otherwise, the crops are extracted from the top, center, and bottom of the image.

At the end of this preprocessing step, features are generated for each of the crops, using the pretrained CLIP network. Each

square crop is resized to 224×224 pixels and fed to CLIP, resulting in three 1024-D feature vectors (one for each crop). We empirically evaluated the best way to join these three vectors in order to obtain a single feature vector, and concluded that performing their elementwise sum, followed by L^2 normalization, generates the best results.[1] This gives a single 1024-D feature vector that describes the whole signature.

C. Classification with SVM

For each signature pair, we concatenate our MLS features (Section III-A) and CLIP features (Section III-B), resulting in a 2274-D feature vector for the pair (Figure 1, right). We use this as input information to a binary classifier, which has the task of determining if the signature pair contains either two genuine signatures, or one genuine and one forged signature. We use a linear Support Vector Machine (SVM) for the binary classification task, which performs well in this context. Our implementation is based on the `LinearSVC` class of scikit-learn [28], with default hyperparameters.[2]

We train this classifier in a supervised way, using a training set composed of example signature pairs (from many different writers) with known genuine/forgery status. Since our method is writer-independent, we train a single model for all writers.

IV. Experimental Results

We performed exhaustive experiments with our proposed method over three commonly used skilled-forgeries datasets: the CEDAR dataset [2], provided by the Center of Excellence for Document Analysis and Recognition from Buffalo University;[3] and BHSig260 [6], which contains two different datasets, for Hindi and Bangla writers. We analyze unbiased versions of these datasets (Section IV-B), as well as versions subjected to rotations and scalings of the signatures (Section IV-C).

We compare our technique against the best-performing state-of-the-art offline writer-independent methods: Dutta et al. [21], which is based on handcrafted features, and Dey et al. [4], which is based on a neural network called SigNet. We use the source code provided by the authors. Table II summarizes our results, which we discuss in detail in the following sections.

A. Experimental Methodology

Each dataset is composed by a collection of writers, each writer having a certain number of genuine signature images (G) and skilled-forgery signature images (F). By pairing signatures from the same writer, one can generate examples of genuine-genuine signature pairs (G-G), and genuine-forgery signature pairs (G-F). Table I summarizes these numbers for the CEDAR, Bengla and Hindi datasets. To get a balanced dataset, one must use the same quantity of G-G and G-F pairs. This is done by randomly sampling 276 pairs from all possible G-F pairs (since the number of G-G pairs is 276, smaller than that of G-F pairs). In summary, we end up with 552 examples per writer composed by 276 G-G examples and 276 G-F examples.

Dataset	Writers	Genuine (G)	Forgery (F)	G-G	G-F
CEDAR	55	24	24	276	576
Bengla	100	24	30	276	720
Hindi	160	24	30	276	720

TABLE I
STATISTICS OF THE DATASETS USED IN OUR EXPERIMENTS.

We retrain from scratch and perform 10-fold cross validation for all methods, separating the training and test data *by writer* for each fold (i.e., if a writer is included in the training set, none of his or her signatures are included in the test set). The implementation of Dutta et al. [21] requires large amounts of memory for computation, and their source code provides a parameter to control which percentage of the total number of examples is used for training and testing in each fold. We use the default value of 30% for CEDAR, which is hard-coded in their source code, and we select 15% for Bangla and 10% for Hindi, which leads to 828, 828 and 884 signature pairs, respectively, for the testing set in each fold. The source code of SigNet [4] does not implement cross validation, so we implemented a cross-validation mechanism with the characteristics mentioned above. Since SigNet is a CNN, it separates a portion of the training data as validation data, in order to save the best obtained model state across all training epochs.

In our evaluation tables, we report mean and standard deviation of all metrics over the cross-validation folds. The best results in each group are marked in bold, and the results which are statistically equivalent to the best result (according to Welch's t-test with $p < 0.05$) are highlighted in green. We include the Equal Error Rate (EER) metric, which is widely used to compare signature verification techniques [1]. EER is defined as the value at the intersection of the False Acceptance Rate (FAR) and False Rejection Rate (FRR) curves (which are generated by varying the discrimination threshold). Lower EER means better classification performance. According to Mohammed et al. [1], EER for offline signature verification systems ranges from 10% to 30% (versus 2% to 5% for online systems).

As mentioned in Section II, the SigNet method selects the discrimination threshold that maximizes its accuracy *over the test set*, possibly using a different threshold in each different fold. In contrast, our technique and Dutta et al.'s method use SVM classifiers with a fixed threshold defined by the particular implementation over the *training* set (normally the threshold used is 0). Therefore, the EER metric is the most suitable when comparing experimental results against the SigNet method, since EER is not affected by the test-set threshold selection of SigNet (we mark the names of the metrics that are affected by this selection with an asterisk in our tables).

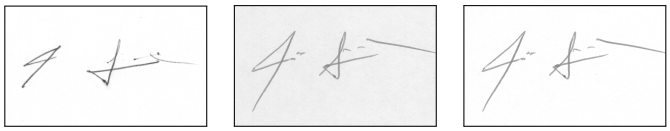
B. Evaluation on "Unbiased" Datasets

When performing experiments over the CEDAR dataset, we noticed that genuine signature images have an extremely

[1] We experimented with elementwise sum, mean, and max operations, as well as concatenation, all optionally followed by renormalization. The concatenation option significantly increases memory consumption and execution time since the number of CNN features will be 3 times that of the other options.

[2] We optimize for the primal problem, which is the recommended procedure when the number of training examples is greater than the number of features.

[3] http://www.cedar.buffalo.edu/NIJ/data/signatures.rar



(a) Forged signature (b) Genuine signature (c) Genuine signature after our unbiasing step

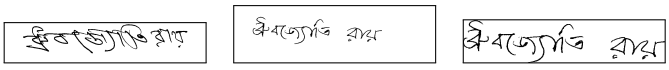
Fig. 4. Example signatures from CEDAR dataset [2], illustrating the significantly different background color between the (a) genuine and (b) forged signature images. This is a source of bias for Machine Learning algorithms.

different background when compared against forged signature images (Figures 4a and 4b). More precisely, genuine signature images have light-gray background, in constrast with forged signature images with completely white backgrounds. This pattern is repeated for all writers in CEDAR, and is a significant source of bias for Machine Learning algorithms, which could "cheat" by differentiating between the two classes just based on the background color. In particular, we hypothesize that the 100%-accuracy listed in the SigNet paper for CEDAR is a consequence of this bias.

To prove this hypothesis, we applied a histogram transformation to all genuine signatures in CEDAR in order to remove the background color discrepancy (i.e., make all images have a white background). The pixel operation used was: $[\text{new pixel color}] = 1.23*([\text{original pixel color}]-0.4)+0.35$. This is illustrated in Figure 4c. From this "unbiasing" procedure we define a new version of the CEDAR dataset which we call C-Unbiased, and we rename the original CEDAR as C-Biased.

As shown in Table II, the accuracy for SigNet drops from 100% in C-Biased to 79.51% in C-Unbiased, supporting our hypothesis that SigNet is heavily influenced just by the background color (i.e., the classifier was not differentiating between the signatures, just their backgrounds). The method of Dutta et al., on the other hand, is not corrupted by this bias, instead *increasing* its accuracy from 90.93% in C-Biased to 92.87% in C-Unbiased (likely the white-background images work better for their BRISK feature extraction). Finally, our proposed method also includes CNN features and thus is sensitive to this type of bias, but proves significantly more robust (versus SigNet): its accuracy drops just 2.15 percentage points, from 99.57% in C-Biased to 97.42% in C-Unbiased. Overall, our method significantly outperforms both SigNet and the technique of Dutta et al. in C-Unbiased, achieving an EER of just 1.95% (and thus FAR = FRR = 1.95%).

Following this bias analysis on the CEDAR dataset, we analyzed both the Bangla and Hindi datasets also looking for possible sources of bias. We found that genuine image signatures from Bangla and Hindi writers are not centered. In fact, all the genuine signature images have the signature positioned at the left side, as shown in Figure 5b. To test whether this could introduce bias, we applied a trimming operation over all images in both datasets, cropping empty regions around the signature images (Figure 5c). We thus defined B-Unbiased and B-Biased datasets based on Bangla, and H-Unbiased and H-Biased datasets based on Hindi dataset.



(a) Forged signature (b) Genuine signature (c) Genuine signature after our unbiasing step

Fig. 5. Example signatures from Bengla dataset [6], illustrating the different signature position between (a) genuine and (b) forged signature images.

As shown in Table II, the centering of the signatures did not prove to be a significant source of bias.

C. Evaluation of Rotation and Scale Invariance

In real-world scenarios, signatures can be found in different orientations and sizes. Therefore, it is important for signature-verification methods to perform well in situations where the signatures being compared are subjected to differences in rotation and scale. To evaluate this property, we propose the following variations of our previously-proposed *X*-Unbiased datasets (for $X \in \{\text{C, B, H}\}$, respectively CEDAR, Bangla, and Hindi), generated with imagemagick's `convert` command:

- *X*-**UR** = *X*-Unbiased + **Rotations**, where each image in the original *X*-Unbiased dataset has been subjected to a different random rotation between 0 and 360 degrees;
- *X*-**US** = *X*-Unbiased + **Scalings**, where each image in the original *X*-Unbiased dataset has been subjected to a different random downscaling between 50% and 100% of its original size;
- *X*-**URS** = *X*-Unbiased + **Rotations** + **Scalings**, which combines both random rotations and downscalings (with different random parameters versus UR and US datasets).

Table II summarizes our results, grouped by dataset. As one can see, our proposed method is significantly more robust against Rotations (UR datasets), with an average loss in accuracy of just 2.2 pp (percentage points) when replacing Unbiased by UR data. In comparison, the next-best method, SigNet, suffers a 8.4 pp loss in accuracy, while Dutta et al. loses 14.8 pp. Furthermore, our technique is adequately robust against Scalings (US datasets), with an average loss in accuracy of 2.2 pp (versus 0.9 pp for SigNet and 9.9 pp for Dutta et al.). Finally, our method is the most robust against combined Rotations and Scalings (URS datasets), with an average loss in accuracy of just 4.5 pp, versus 8.9 pp for SigNet and 22.7 for Dutta et al.

D. Ablation Study

Table III presents an analysis of our proposed set of features and their individual performance, averaged over the CEDAR, Bangla and Hindi datasets. Note how the proposed CLIP feature extraction, which uses our preprocessing pipeline described in Section III-B, performs better in the signature-verification context than the original CLIP pipeline (which simply crops the input image to a square, discarding information from the signatures, which are often rectangular in shape). Furthermore, combining our CLIP features with our MLS features (Proposed Method), results in further increase in performance (higher accuracy and lower EER), while also reducing the standard deviation between different folds of the cross validation experiment.

Dataset	Method	EER (%) (lower is better)	Accuracy* (%) (higher is better)	F1-Score* (%)	Precision* (%)	Recall* (%)	ROC AUC
C-Unbiased	**Proposed Method**	$\mathbf{1.95 \pm 1.71}$	$\mathbf{97.42 \pm 1.95}$	$\mathbf{97.43 \pm 1.97}$	$\mathbf{97.08 \pm 3.03}$	$\mathbf{97.92 \pm 3.40}$	$\mathbf{99.75 \pm 0.44}$
C-Unbiased	SigNet	21.73 ± 9.17	79.51 ± 8.40	80.25 ± 7.74	78.90 ± 10.10	82.75 ± 9.83	85.43 ± 9.39
C-Unbiased	Dutta et al.	6.67 ± 3.58	92.87 ± 3.39	92.80 ± 3.48	93.53 ± 4.65	92.49 ± 6.13	97.84 ± 1.89
C-Unbiased	*Human*		79.20 ± 5.22				
C-UR	**Proposed Method**	$\mathbf{2.82 \pm 1.56}$	$\mathbf{96.71 \pm 1.95}$	$\mathbf{96.68 \pm 2.02}$	$\mathbf{97.02 \pm 2.43}$	$\mathbf{96.48 \pm 3.74}$	$\mathbf{99.52 \pm 0.49}$
C-UR	SigNet	25.76 ± 6.18	75.83 ± 6.62	78.00 ± 6.08	71.65 ± 5.78	85.84 ± 7.98	81.31 ± 6.78
C-UR	Dutta et al.	20.07 ± 3.82	79.18 ± 4.45	77.69 ± 7.03	82.64 ± 4.87	74.83 ± 12.46	87.85 ± 4.17
C-US	**Proposed Method**	$\mathbf{5.13 \pm 2.90}$	$\mathbf{94.17 \pm 2.84}$	$\mathbf{94.09 \pm 2.97}$	$\mathbf{94.75 \pm 3.39}$	$\mathbf{93.70 \pm 5.30}$	$\mathbf{98.71 \pm 1.38}$
C-US	SigNet	25.76 ± 9.23	75.05 ± 9.15	76.46 ± 7.46	73.97 ± 10.05	79.76 ± 6.45	80.19 ± 10.54
C-US	Dutta et al.	23.68 ± 4.02	75.76 ± 4.77	73.94 ± 7.25	78.63 ± 2.66	70.87 ± 12.44	84.74 ± 4.50
C-URS	**Proposed Method**	$\mathbf{6.26 \pm 2.16}$	$\mathbf{93.39 \pm 2.09}$	$\mathbf{93.38 \pm 2.16}$	$\mathbf{93.24 \pm 2.34}$	$\mathbf{93.62 \pm 3.65}$	$\mathbf{98.34 \pm 1.24}$
C-URS	SigNet	27.10 ± 4.95	74.67 ± 5.30	76.92 ± 5.46	70.64 ± 4.91	84.96 ± 8.69	79.82 ± 5.09
C-URS	Dutta et al.	34.23 ± 2.31	64.95 ± 2.49	59.39 ± 5.27	70.47 ± 3.31	52.03 ± 8.35	71.42 ± 3.36
C-URS	*Human*		78.20 ± 4.66				
B-Unbiased	**Proposed Method**	$\mathbf{15.87 \pm 4.02}$	83.76 ± 3.68	83.57 ± 4.38	83.87 ± 2.36	83.62 ± 7.70	$\mathbf{92.30 \pm 3.32}$
B-Unbiased	SigNet	16.46 ± 4.20	$\mathbf{84.57 \pm 3.82}$	$\mathbf{85.14 \pm 3.67}$	82.31 ± 4.70	$\mathbf{88.45 \pm 5.14}$	91.20 ± 3.71
B-Unbiased	Dutta et al.	16.16 ± 3.36	83.55 ± 3.22	83.29 ± 3.44	$\mathbf{84.83 \pm 5.03}$	82.39 ± 6.60	91.90 ± 2.78
B-UR	**Proposed Method**	$\mathbf{17.66 \pm 2.62}$	$\mathbf{81.98 \pm 2.62}$	$\mathbf{81.80 \pm 2.65}$	$\mathbf{82.93 \pm 4.47}$	81.05 ± 4.81	$\mathbf{90.80 \pm 2.35}$
B-UR	SigNet	21.05 ± 4.37	79.62 ± 4.23	80.43 ± 3.88	77.51 ± 4.42	$\mathbf{83.67 \pm 3.98}$	86.94 ± 4.20
B-UR	Dutta et al.	32.74 ± 3.49	66.82 ± 3.60	65.12 ± 5.26	68.46 ± 3.76	62.68 ± 8.64	72.96 ± 4.10
B-US	**Proposed Method**	16.96 ± 4.01	82.90 ± 3.92	82.69 ± 4.45	83.40 ± 3.95	82.34 ± 7.21	90.91 ± 3.46
B-US	SigNet	$\mathbf{15.35 \pm 3.66}$	$\mathbf{85.45 \pm 3.78}$	$\mathbf{85.82 \pm 3.59}$	$\mathbf{84.03 \pm 5.06}$	$\mathbf{87.96 \pm 4.68}$	$\mathbf{91.76 \pm 3.13}$
B-US	Dutta et al.	22.15 ± 4.24	77.79 ± 4.06	77.10 ± 4.84	79.42 ± 4.75	75.39 ± 7.79	86.12 ± 4.24
B-URS	**Proposed Method**	$\mathbf{19.24 \pm 2.78}$	$\mathbf{80.65 \pm 2.96}$	$\mathbf{80.44 \pm 3.01}$	$\mathbf{81.60 \pm 4.61}$	79.71 ± 5.28	$\mathbf{89.36 \pm 2.90}$
B-URS	SigNet	21.78 ± 3.65	79.24 ± 3.46	80.39 ± 3.31	76.29 ± 3.85	$\mathbf{85.16 \pm 4.90}$	86.17 ± 3.68
B-URS	Dutta et al.	35.99 ± 3.14	63.76 ± 3.12	62.39 ± 5.27	64.60 ± 2.85	60.94 ± 8.99	69.03 ± 3.91
H-Unbiased	**Proposed Method**	16.39 ± 3.47	83.59 ± 3.37	83.53 ± 3.46	83.85 ± 3.94	83.36 ± 4.69	91.61 ± 3.31
H-Unbiased	SigNet	$\mathbf{15.22 \pm 4.02}$	$\mathbf{85.14 \pm 3.90}$	$\mathbf{85.33 \pm 3.88}$	$\mathbf{84.35 \pm 4.41}$	$\mathbf{86.48 \pm 4.63}$	$\mathbf{92.74 \pm 3.03}$
H-Unbiased	Dutta et al.	19.36 ± 2.67	80.69 ± 2.57	80.96 ± 2.61	79.83 ± 2.69	82.19 ± 3.37	88.94 ± 2.54
H-UR	**Proposed Method**	$\mathbf{20.64 \pm 4.23}$	$\mathbf{79.39 \pm 4.21}$	$\mathbf{79.21 \pm 4.26}$	$\mathbf{80.02 \pm 4.86}$	$\mathbf{78.55 \pm 4.89}$	$\mathbf{87.30 \pm 4.25}$
H-UR	SigNet	31.85 ± 3.30	68.55 ± 3.38	70.02 ± 3.53	66.84 ± 2.86	73.59 ± 4.83	74.42 ± 4.07
H-UR	Dutta et al.	33.36 ± 2.65	66.56 ± 2.23	65.34 ± 3.09	67.71 ± 1.90	63.26 ± 4.79	73.20 ± 2.86
H-US	**Proposed Method**	18.87 ± 3.73	81.03 ± 3.84	80.96 ± 4.04	81.19 ± 4.03	80.87 ± 5.26	88.88 ± 3.92
H-US	SigNet	$\mathbf{14.41 \pm 3.57}$	$\mathbf{86.06 \pm 3.26}$	$\mathbf{86.33 \pm 2.90}$	$\mathbf{85.26 \pm 4.83}$	$\mathbf{87.59 \pm 2.38}$	$\mathbf{92.89 \pm 2.92}$
H-US	Dutta et al.	26.15 ± 2.34	73.71 ± 2.40	73.26 ± 2.49	74.71 ± 3.57	72.13 ± 4.43	81.99 ± 2.75
H-URS	**Proposed Method**	$\mathbf{22.54 \pm 4.22}$	$\mathbf{77.28 \pm 4.23}$	$\mathbf{77.06 \pm 4.25}$	$\mathbf{77.87 \pm 4.64}$	$\mathbf{76.36 \pm 4.76}$	$\mathbf{84.99 \pm 4.55}$
H-URS	SigNet	31.77 ± 4.33	68.70 ± 4.29	70.13 ± 4.66	67.05 ± 3.91	73.91 ± 7.44	74.34 ± 5.10
H-URS	Dutta et al.	39.66 ± 2.23	60.23 ± 1.86	59.19 ± 2.70	60.77 ± 1.97	57.85 ± 4.40	64.14 ± 2.53
C-Biased	**Proposed Method**	0.20 ± 0.26	99.67 ± 0.34	99.67 ± 0.34	99.55 ± 0.66	99.79 ± 0.31	$\mathbf{100.00 \pm 0.00}$
C-Biased	SigNet	$\mathbf{0.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
C-Biased	Dutta et al.	8.32 ± 4.36	90.93 ± 4.52	90.66 ± 5.04	92.41 ± 6.12	90.07 ± 9.89	96.44 ± 3.43
B-Biased	**Proposed Method**	14.98 ± 3.31	84.80 ± 3.17	84.67 ± 3.49	85.17 ± 3.32	84.45 ± 6.00	$\mathbf{93.10 \pm 2.57}$
B-Biased	SigNet	$\mathbf{14.30 \pm 3.99}$	$\mathbf{86.46 \pm 3.85}$	$\mathbf{86.56 \pm 3.62}$	$\mathbf{86.37 \pm 5.36}$	$\mathbf{87.04 \pm 4.59}$	92.48 ± 3.56
B-Biased	Dutta et al.	17.25 ± 3.58	82.74 ± 3.56	82.39 ± 4.11	83.90 ± 4.20	81.45 ± 7.40	91.07 ± 3.10
H-Biased	**Proposed Method**	16.03 ± 3.58	83.95 ± 3.61	83.85 ± 3.78	$\mathbf{84.29 \pm 3.91}$	83.57 ± 5.31	91.59 ± 3.65
H-Biased	SigNet	$\mathbf{14.93 \pm 2.87}$	$\mathbf{85.73 \pm 2.66}$	$\mathbf{86.26 \pm 2.54}$	83.32 ± 3.40	$\mathbf{89.52 \pm 3.09}$	$\mathbf{93.25 \pm 1.85}$
H-Biased	Dutta et al.	18.72 ± 2.62	81.19 ± 2.04	81.40 ± 2.07	80.55 ± 2.63	82.38 ± 3.28	89.52 ± 1.98

TABLE II

CLASSIFICATION METRICS FOR THE METHODS EVALUATED IN OUR EXPERIMENTS. BEST RESULTS IN EACH GROUP ARE MARKED IN BOLD, AND RESULTS WHICH ARE STATISTICALLY EQUIVALENT TO THE BEST RESULT ARE HIGHLIGHTED IN GREEN (WELCH'S T-TEST). METRIC NAMES MARKED WITH AN ASTERISK (*) ARE AFFECTED BY SIGNET'S TEST-SET THRESHOLD SELECTION PROCEDURE (SEE SECTION IV-A).

E. User Study with Human Subjects

We designed a small user study to measure the expected human accuracy in the signature verification task. A pair of signature images is presented to the user, who is asked to determine if the signatures are genuine (written by the same writer) or if one of them is a forgery (signatures are written by different writers). We recruited a total of 5 users for the study, with no previous experience in signature verification. Each user saw a total of 200 signature pairs, being 100 from the C-Unbiased and 100 from the C-URS dataset (in this order). Within each dataset, we separated a balanced set of 50 genuine and 50 forged signature pairs, which were presented to the users in a random order. All users saw the same set of images, and no time limit was imposed (average test time was 25 minutes).

As seen in Table II (rows in light blue), the average human accuracy was $\sim 79\%$, and the users did not suffer any significant performance loss due to rotations and scalings of the images (C-URS dataset). This reinforces the importance for the classification methods to exhibit rotation and scale invariance properties. The individual accuracies for the users were $\{82\%, 86\%, 80\%, 74\%, 74\%\}$ for C-Unbiased and

Dataset	Method	EER (%) (lower is better)	Accuracy* (%) (higher is better)
Unbiased	**Proposed Method**	$\mathbf{11.40 \pm 3.07}$	$\mathbf{88.26 \pm 3.00}$
Unbiased	Our MLS Features	17.19 ± 4.87	82.71 ± 4.79
Unbiased	Our CLIP Features	14.60 ± 4.06	85.28 ± 4.04
Unbiased	Original CLIP	19.03 ± 3.35	80.59 ± 3.69
UR	**Proposed Method**	$\mathbf{13.71 \pm 2.80}$	$\mathbf{86.03 \pm 2.93}$
UR	Our MLS Features	22.88 ± 4.39	76.98 ± 4.62
UR	Our CLIP Features	14.99 ± 2.86	84.74 ± 2.84
UR	Original CLIP	19.68 ± 3.26	80.11 ± 3.26
US	**Proposed Method**	$\mathbf{13.65 \pm 3.55}$	$\mathbf{86.03 \pm 3.53}$
US	Our MLS Features	31.31 ± 3.09	68.72 ± 3.17
US	Our CLIP Features	15.18 ± 4.03	84.60 ± 3.95
US	Original CLIP	21.36 ± 4.28	78.27 ± 4.34
URS	**Proposed Method**	$\mathbf{16.01 \pm 3.05}$	$\mathbf{83.77 \pm 3.09}$
URS	Our MLS Features	35.00 ± 3.82	64.97 ± 4.01
URS	Our CLIP Features	16.19 ± 3.02	83.64 ± 3.05
URS	Original CLIP	21.48 ± 3.31	78.11 ± 3.37

TABLE III
ABLATION STUDY (AVERAGED OVER CEDAR, BANGLA AND HINDI DATASETS). PROPOSED METHOD = OUR MLS + OUR CLIP FEATURES.

$\{81\%, 81\%, 80\%, 79\%, 70\%\}$ for C-URS. We conclude that signature verification is not an easy task for (untrained) humans, and state-of-the-art algorithms are able to significantly surpass human performance.

V. CONCLUSIONS

We introduced a new offline writer-independent signature verification method, based on a combination of Moving Least-Squares handcrafted features and features transferred from a CNN. In our experiments, our method outperforms state-of-the-art techniques on the CEDAR dataset (Western-style writing), while simultaneously obtaining good results on the Bangla and Hindi datasets. Additionally, our method exhibits some degree of rotation and scale invariance, being the best-performing technique across all datasets when the signature images are subjected to random changes in rotation and scale. We also presented a discussion of unintended bias in the datasets, supported by experiments, in addition to a user study measuring the expected human performance on signature verification.

Acknowledgements. This work was partially supported by CNPq-Brazil (436932/2018-0), Petrobras (2017/00752-3), and financed in part by "Coordenação de Aperfeiçoamento de Pessoal de Nível Superior" - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] R. A. Mohammed, R. M. Nabi, M. Sardasht, R. Mahmood, and R. M. Nabi, "State-of-the-art in handwritten signature verification system," in *Intl. Conference on Computational Science and Computational Intelligence*. IEEE, 2015, pp. 519–525.

[2] M. K. Kalera, S. Srihari, and A. Xu, "Offline signature verification and identification using distance statistics," *Intl. Journal of Pattern Recog. and Artificial Intelligence*, vol. 18, no. 07, pp. 1339–1360, 2004.

[3] A. Kumar and K. Bhatia, "A survey on offline handwritten signature verification system using writer dependent and independent approaches," in *International Conference on Advances in Computing, Communication, & Automation*. IEEE, 2016, pp. 1–6.

[4] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Lladós, and U. Pal, "SigNet: Convolutional siamese network for writer independent offline signature verification," *arXiv:1707.02131*, 2017.

[5] L. S. Oliveira, E. Justino, and R. Sabourin, "Off-line signature verification using writer-independent approach," in *Intl. Joint Conference on Neural Networks*. IEEE, 2007, pp. 2539–2544.

[6] S. Pal, A. Alaei, U. Pal, and M. Blumenstein, "Performance of an off-line signature verification method based on texture features on a large indic-script signature dataset," in *12th IAPR workshop on document analysis systems*. IEEE, 2016, pp. 72–77.

[7] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," *preprint arXiv:2103.00020*, 2021.

[8] P. Lancaster and K. Salkauskas, "Surfaces generated by moving least squares methods," *Mathematics of computation*, vol. 37, no. 155, pp. 141–158, 1981.

[9] 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*, vol. 14, no. 8, pp. 2153–2166, 2019.

[10] M. E. Munich and P. Perona, "Visual identification by signature tracking," *IEEE TPAMI*, vol. 25, no. 2, pp. 200–217, 2003.

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

[12] M. A. Ferrer, J. B. Alonso, and C. M. Travieso, "Offline geometric parameters for automatic signature verification using fixed-point arithmetic," *IEEE TPAMI*, vol. 27, no. 6, pp. 993–997, 2005.

[13] R. Kumar, J. Sharma, and B. Chanda, "Writer-independent off-line signature verification using surroundedness feature," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 301–308, 2012.

[14] 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*, vol. 43, no. 1, pp. 387–396, 2010.

[15] V. Ramesh and M. N. Murty, "Off-line signature verification using genetically optimized weighted features," *Pattern Recognition*, vol. 32, no. 2, pp. 217–233, 1999.

[16] G. Alvarez, B. Sheffer, and M. Bryant, "Offline signature verification with convolutional neural networks," *Tech. Report, Stanford Univ.*, 2016.

[17] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Writer-independent feature learning for offline signature verification using deep convolutional neural networks," in *Intl. Joint Conference on Neural Networks*. IEEE, 2016, pp. 2576–2583.

[18] ——, "Learning features for offline handwritten signature verification using deep convolutional neural networks," *Pattern Recognition*, vol. 70, pp. 163–176, 2017.

[19] V. L. Souza, A. L. Oliveira, and R. Sabourin, "A writer-independent approach for offline signature verification using deep convolutional neural networks features," in *Brazilian Conference on Intelligent Systems*. IEEE, 2018, pp. 212–217.

[20] 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*, vol. 40, pp. 274–282, 2016.

[21] A. Dutta, U. Pal, and J. Lladós, "Compact correlated features for writer independent signature verification," in *Intl. Conference on Pattern Recognition*. IEEE, 2016, pp. 3422–3427.

[22] Y. Dodge, *The Concise Encyclopedia of Statistics*. Springer, 2008.

[23] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 1979.

[24] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[25] D. Levin, "The approximation power of moving least-squares," *Mathematics of computation*, vol. 67, no. 224, pp. 1517–1531, 1998.

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[27] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.