Learning features for Offline Handwritten Signature Verification

by

Luiz Gustavo HAFEMANN

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, APRIL 23, 2019

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Robert Sabourin, Thesis Supervisor
Systems Engineering Department, École de technologie supérieure

Mr. Luiz Eduardo Soares de Oliveira, Co-supervisor
Department of Informatics, Federal University of Paraná

Mr. Patrick Cardinal, President of the Board of Examiners
Software and IT Engineering Department, École de technologie supérieure

Mr. Matthew Toews, Member of the jury
Systems Engineering Department, École de technologie supérieure

Mr. Tiago Henrique Falk, External Examiner
Centre Énergie, Matériaux, Télécommunications, Institut National de la Recherche Scientifique

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON APRIL 15, 2019

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

First of all, I would like to thank my wife Renata for being by my side in this journey. Her encouragement and support were fundamental during these four years.

I would like to thank my thesis advisor Robert Sabourin and co-advisor Luiz Oliveira for sharing their wisdom and experience in conducting high-level research. I learned much from them, and I really appreciate the discussions and the progress we made during these four years that we worked together.

Thanks for all the colleagues at LIVIA for their help on my research, for teaching me new things and for being good friends. Special thanks to Rafael Menelau, Le Thanh Nguyen-Meidine, Hiba Zakane, Sajjad Abdoli, Jérôme Rony, Saypraseuth Mounsaveng, Mariana Araújo, Jonathan de Matos and Rafooneh Jafarian Bahri. I would like to thank professors Eric Granger and Ismail Ben Ayed for the work we did together. I would also like to thank professor Marco Pedersoli for all the discussions, and for supporting the LIVIA reading group.

Finally, I would like to thank the Fonds de Recherche du Québec – Nature et technologies (FRQNT), the École de technologie supérieure (ÉTS) and the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the financial support.

L'apprentissage profond appliqué à la vérification de signatures manuscrites

Luiz Gustavo HAFEMANN

RÉSUMÉ

La signature manuscrite est, encore aujourd'hui, la modalité biométrique la plus acceptée socialement et légalement, pour authentifier les documents manuscrits et imprimés. Au cours des dernières décennies, plusieurs chercheurs ont abordé le problème de l'automatisation du processus de la vérification des signatures, en utilisant diverses techniques d'apprentissage machine et de reconnaissance de formes. La majorité des travaux de recherche ont porté sur la définition et l'évaluation de représentations discriminantes des signatures, soit en concevant de nouveaux extracteurs de caractéristiques spécialisés, soit en utilisant des extracteurs de caractéristiques développées à d'autres fins. Les chercheurs ont notamment utilisé des connaissances en graphologie, en vision par ordinateur et en traitement du signal. Malgré les progrès réalisés à ce jour, la classification des signatures authentiques et des faux avec imitation demeure un problème de recherche non résolu.

Dans cette thèse, nous proposons d'aborder ce problème sous un autre angle, en apprenant les représentations de caractéristiques directement à partir d'images de signature. L'hypothèse est que, en l'absence d'un bon modèle du processus de génération de données, il est préférable d'apprendre les caractéristiques à partir des données. Dans la première contribution, nous proposons une méthode pour apprendre les caractéristiques génériques pour plusieurs personnes, en utilisant un objectif d'optimisation substitut. Après la phase d'apprentissage des caractéristiques, un classificateur est entrainé pour chaque personne, en utilisant les caractéristiques apprises à partir d'un corpus de scripteurs indépendant. De plus, on considère une extension de notre méthode qui permet l'apprentissage de caractéristiques en utilisant des images de faux avec imitation (disponibles pour un sous-ensemble d'utilisateurs). On a observé sur quatre bases de signatures que ces caractéristiques sont bien adaptées pour la vérification des signatures provenant de nouveaux utilisateurs qui n'ont pas été utilisés pour l'apprentissage de cette représentation. Les résultats expérimentaux obtenus sur les quatre bases de données ont montré les meilleures performances considérant l'état de l'art.

Dans la deuxième contribution, nous examinons trois problèmes liés aux systèmes de vérification de signature: (i) l'apprentissage d'une représentation de cardinalité fixe pour des images de signatures de tailles variées; (ii) l'impact de la résolution des images de signatures numérisées sur la performance du système de vérification et (iii) comment les caractéristiques généralisent dans de nouvelles conditions de fonctionnement avec et sans raffinement. Nous proposons des méthodes qui permettent de traiter des signatures de tailles variées et nos simulations montrent des résultats comparables à ceux publiés dans la littérature, sans la contrainte de normaliser la taille des images présentées à l'entrée du réseaux de neurones.

Dans la troisième contribution, nous proposons de formuler le problème de la vérification des signatures manuscrites comme un méta-problème. Cette formulation apprend également directement à partir des images de signatures et permet l'optimisation directe de l'objectif (séparer les signatures authentiques des falsifications qualifiées) au lieu de s'appuyer sur des objectifs substituts pour l'apprentissage des caractéristiques. De plus, on montre que cette méthode peut naturellement faciliter l'adaptation du classificateur pour des nouveaux utilisateurs avec un classificateurs à une classe.

Pour ce qui est de la quatrième contribution, nous analysons les limites des systèmes de vérification hors ligne de signatures manuscrites dans un contexte d'apprentissage machine antagoniste (adversarial learning), où un adversaire actif essaie de perturber le système. Une nouvelle taxonomie des menaces représentées par les exemples antagonistes pour les systèmes biométrique est proposée. Plusieurs stratégies ont été considérées pour évaluer expérimentalement les attaques basées sur différents scénarios d'objectifs et de connaissance du système attaqué. Nous avons observé que les systèmes de vérification qui reposent sur des caractéristiques définies manuellement et ceux basés sur des caractéristiques apprises aux chapitres précédents sont susceptibles d'attaques antagonistes dans plusieurs scénarios. Notamment, ces systèmes de vérification sont vulnérables aux attaques basées sur une connaissance partielle des mécanismes internes du système, pour lesquels l'attaquant n'a pas accès aux classificateurs déjà entraînés. Bien que certaines défenses proposées dans la littérature augmentent la robustesse de ces systèmes, cette recherche met en évidence les scénarios pour lesquels ceux-ci sont vulnérables.

Mots-clés: Verification de signatures, biométrie, apprentissage de représentations

Learning features for Offline Handwritten Signature Verification

Luiz Gustavo HAFEMANN

ABSTRACT

Handwritten signatures are the most socially and legally accepted means for identifying a person. Over the last few decades, several researchers have approached the problem of automating their recognition, using a variety of techniques from machine learning and pattern recognition. In particular, most of the research effort has been devoted to obtaining good feature representations for signatures, by designing new feature extractors, as well as experimenting with feature extractors developed for other purposes. To this end, researchers have used insights from graphology, computer vision, signal processing, among other areas. In spite of the advancements in the field, building classifiers that can separate between genuine signatures and skilled forgeries (forgeries made targeting a particular individual) is still an open research problem.

In this thesis, we propose to address this problem from another perspective, by learning the feature representations directly from signature images. The hypothesis is that, in the absence of a good model of the data generation process, it is better to learn the features from data. As a first contribution, we propose a method to learn Writer-Independent features using a surrogate objective, followed by training Writer-Dependent classifiers using the learned features. Furthermore, we define an extension that allows leveraging the knowledge of skilled forgeries (from a subset of users) in the feature learning process. We observed that such features generalize well to new users, obtaining state-of-the-art results on four widely used datasets in the literature.

As a second contribution, we investigate three issues of signature verification systems: (i) learning a fixed-sized vector representation for signatures of varied size; (ii) analyzing the impact of the resolution of the scanned signatures in system performance and (iii) how features generalize to new operating conditions with and without fine-tuning. We propose methods to handle signatures of varied size and our experiments show results comparable to state-of-the-art while removing the requirement that all input images have the same size.

As a third contribution, we propose to formulate the problem of signature verification as a meta-learning problem. This formulation also learns directly from signatures images, and allows the direct optimization of the objective (separating genuine signatures and skilled forgeries), instead of relying on surrogate objectives for learning the features. Furthermore, we show that this method is naturally extended to formulate the adaptation (training) for new users as one-class classification.

As a fourth contribution, we analyze the limitations of these systems in an Adversarial Machine Learning setting, where an active adversary attempts to disrupt the system. We characterize new threats posed by Adversarial Examples on a taxonomy of threats to biometric systems, and conduct extensive experiments to evaluate the success of attacks under different scenarios of attacker's goals and knowledge of the system under attack. We observed that both systems that rely on handcrafted features, as well as those using learned features, are susceptible to adversarial attacks in a wide range of scenarios, including partial-knowledge scenarios where the attacker does not have full access to the trained classifiers. While some defenses proposed in the literature increase the robustness of the systems, this research highlights the scenarios where such systems are still vulnerable.

Keywords: Signature Verification, Biometrics, Feature Learning, Adversarial Machine Learning

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ALGORITHMS

LIST OF ABREVIATIONS

AER	Average Error Rate
CLBP	Completed Local Binary Patterns
CNN	Convolutional Neural Network
DPDF	Directional Probability Density Function
DMML	Deep Multitask Metric Learning
EER	Equal Error Rate
ESC	Extended Shadow Code
FAR	False Acceptance Rate
FGM	Fast Gradient Method
FRR	False Rejection Rate
GAN	Generative Adversarial Network
FLOPS	Floating point operations per second
GLCM	Gray-Level Co-Occurrence Matrix
HOG	Histogram of Oriented Gradients
HMM	Hidden Markov Model
LBP	Local Binary Patterns
LK	Limited Knowledge
LSLR	Learning Per-Layer Per-Step Learning Rates
LSTM	Long Short-Term Memory

MAML	Model Agnostic Meta Learning
MLP	Multi-Layer Perceptron
NCA	Neighborhood Component Analysis
NES	Natural Evolution Strategy
NN	Neural Network
OC-SVM	One-Class Support Vector Machine
PHOG	Pyramid Histogram of Oriented Gradients
PK	Perfect Knowledge
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
RMSE	Root Mean Squared Error
SIFT	Scale-invariant feature transform
SPP	Spatial Pyramid Pooling
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
WD	Writer-Dependent
WI	Writer-Independent

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\phi(.)$	Feature extraction function
θ	Learnable parameters in a model
$\mathscr{D}$	Development dataset
$\mathscr{E}$	Exploitation dataset
$\mathscr{L}_c$	Learning set for classification
$\mathscr{L}_v$	Learning set for verification
$\mathscr{V}_c$	Validation set for classification
$\mathscr{V}_v$	Validation set for verification
$\mathscr{T}_v$	Testing set for verification
L_c	Loss function for user classification
L_f	Loss function for forgery classification
M	Number of users
N	Number of features (dimensionality of feature vectors)
$P(f\|X)$	Probability of signature X being a forgery
$P(\mathbf{y}\|X)$	Probability of user y being the creator of signature X
X	Input sample (signature)
$\tilde{X}$	Adversarial example
δ	Adversarial noise
y	User that created the signature

$\mathscr{X}$	Feature space
$\mathscr{Y}$	Set of all users
$\mathscr{Y}_{\mathscr{D}}$	Set of all users in the development set
γ	Weight to trade-off between different loss functions
C^{-}	Weight for the negative class
C^{+}	Weight for the positive class
H_{max}	Maximum height of signatures in a dataset
W_{max}	Maximum width of signatures in a dataset
$\mathscr{T}$	Distribution of tasks (i.e. users)
$\mathscr{T}_u$	Task for user u
$\mathscr{D}_{\text{meta-train}}$	Training set for the meta-learner
$\mathscr{D}_{\text{meta-test}}$	Testing set for the meta-learner
$\mathscr{D}_u$	Samples for weight adaptation for user u
$\mathscr{D}'_u$	Samples for meta-update for user u
G_u	Genuine signatures for user u
S_u	Skilled forgeries for user u
$\theta_k^{(u)}$	Parameters adapted to user u after k descent steps
L	Loss function for weight adaptation
L'	Loss function for meta-update

INTRODUCTION

Biometric recognition technology is used in a wide variety of security applications. The aim of such systems is to recognize a person based on physiological or behavioral traits. In the first case, recognition is based on measurements of biological traits, such as the fingerprint, face, iris, etc. The second is concerned with behavioral traits such as voice and the handwritten signature (Jain *et al.* (2004)).

The handwritten signature is a particularly important type of biometric trait, mainly due to its ubiquitous use to verify a person's identity in legal, financial and administrative areas. One of the reasons for its widespread use is that the process of collecting handwritten signatures is non-invasive, and people are familiar with the use of signatures in their daily life (Plamondon & Srihari (2000)).

Signature verification systems aim to automatically discriminate if a signature sample is indeed of a particular person or not. This type of system usually consists of two phases: in an enrollment phase, users of the system provide signature samples. During the verification phase, a user claims a particular identity and provide a signature sample. The system then uses the samples provided during enrollment (or a model built using them) to verify if the user is indeed who he or she claims to be - that is, it classifies the sample as genuine or forgery.

Depending on the acquisition method, signature verification systems are divided into two categories: online (dynamic) and offline (static). In the online case, an acquisition device, such as a digitizing tablet, is used to acquire the user's signature. The data is collected as a sequence over time, containing the position of the pen, and in some cases including additional information such as the pen inclination, pressure, etc. In offline signature verification, the signature is acquired after the writing process is completed. In this case, the signature is represented as a digital image (Impedovo & Pirlo (2008)).

Problem statement

The problem of Handwritten Signature Verification can be modeled as a Pattern Recognition problem. Given a dataset $\mathscr{L}_v$ (**L**earning set for **v**erification) of signature images, the objective is to learn a model that can distinguish between genuine signatures and forgeries. In the most common formulation, namely writer-dependent classification, one classifier is built for each writer. In this formulation, we consider a dataset for each user: $\mathscr{L}_v = \{(X^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $X^{(i)}$ is a signature image, and $y^{(i)}$ is a binary variable that indicates if the signature is genuine ($y^{(i)} = 1$) or a forgery ($y^{(i)} = 0$). Commonly, genuine signatures from other users are used as negative samples (in this context, they are called Random Forgeries), since in real application scenarios, skilled forgeries cannot be expected to be available for every user. Given the dataset $\mathscr{L}_v$, a feature extraction function ϕ is used, that receives an image X as input, and outputs a feature vector $\phi(X) \in \mathbb{R}^N$, where N is the dimensionality of the vector (the number of features). The feature vectors, as well as the labels $\mathbf{y}$ are used to train a binary classifier f. In the generalization phase, a new signature X_{new} is acquired, and the same feature extraction process is applied. The feature vector $\phi(X_{\text{new}})$ is then fed to the classifier, that outputs a decision: $f(\phi(X_{\text{new}}))$, which is a prediction of whether the signature is genuine or a forgery. This process is illustrated in Figure 0.1.

Most of the research focus in the literature has been devoted to obtaining good feature representations for signatures, that is, designing good feature extractors $\phi(X)$ (Hafemann *et al.* (2017b)). 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. Recent work uses texture features, such as Local Binary Patterns (LBP) (Yılmaz & Yanıkoğlu (2016), Hu & Chen (2013)) and Gray-Level Co-occurrence Matrix (GLCM) (Hu & Chen (2013)); directional-based features such as Histogram of Oriented Gradients (HOG) (Yılmaz & Yanıkoğlu (2016)) and Directional-PDF (Rivard *et al.* (2013), Eskander

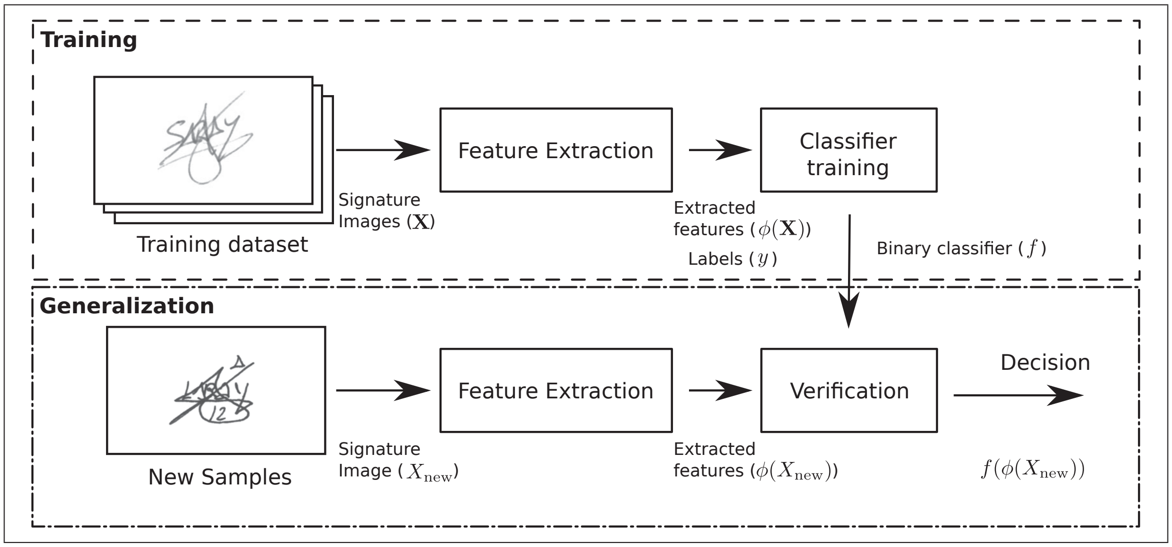

Training
Training dataset
Signature Images ($\mathbf{X}$)
Feature Extraction
Extracted features ($\phi(\mathbf{X})$)
Labels (y)
Classifier training
Binary classifier (f)
Generalization
New Samples
Signature Image (X_{new})
Feature Extraction
Extracted features ($\phi(X_{\text{new}})$)
Verification
Decision
$f(\phi(X_{\text{new}}))$

Figure 0.1 Example of writer-dependent classification.

et al. (2013)); feature extractors specifically designed for signatures, such as the estimation of strokes by fitting Bezier curves (Bertolini *et al.* (2010)); among others. No single 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, especially to distinguish genuine signatures from skilled forgeries - forgeries that are made targeting a particular individual.

Challenges

In this section, we summarize the key challenges in signature verification:

- **Obtaining good feature representations**: The performance of a signature verification system is highly dependent on the features used to train the classifiers. The choice of which feature descriptors to use is problem-dependent, and it is often hard to design good feature extractors. Ideally, the features reflect the process used to generate the data - for instance, neuromotor models of the hand movement. Although this has been explored in the context of online signature verification (e.g. the work from Ferrer *et al.* (2015)), there is not a widely accepted “best" way to extract features for the problem, especially for Offline (static) signature verification, where the dynamic information of the signature generation process is not available.

- **Partial knowledge during training**: A very challenging aspect of Handwritten signature verification is the presence of partial knowledge during training. The objective of the system is to distinguish between genuine signatures and forgeries. However, considering a realistic application scenario, during the training phase only information about the genuine class is available for the majority of users. This fact, combined with the observation that forgers try to make their forgeries very similar to genuine signatures, makes the problem quite hard.
- **Low number of samples per user**: Another challenging aspect is the requirement of these systems to work with a low number of samples per user. Even for applications having millions of users, for a new user, it is common to have only a few samples (e.g. between 1 and 5 samples).

Research Objectives and Contributions

The main research question that we address in this work is whether we can learn better feature representations for offline signature verification, from data, instead of using hand-designed feature extractors.

The core contributions of this thesis are:

- In Chapter 2, the issue of obtaining good features for signature verification is addressed with a two-step process that uses Writer-Independent feature learning, followed by Writer-Dependent classification. Features are learned from a collection of users, partially addressing the problem of having few samples per user. We also present a formulation to incorporate knowledge of forgeries (from a subset of users) in the feature learning process, that addresses some of the issues of partial knowledge during training.

Related publications:

Learning features for offline handwritten signature verification using deep convolutional neural networks (*published in Elsevier's Pattern Recognition, 2017*)

Analyzing features learned for offline signature verification using Deep CNNs (*published in the proceedings of the 23rd International Conference on Pattern Recognition (ICPR), 2016*)

Writer-independent feature learning for offline signature verification using deep convolutional neural networks (*published in proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), 2016*).

- Chapter 3 presents a method to address the issue of learning a fixed-sized representation space for signatures of variable size. This chapter also presents results on the impact of the resolution of the images used for training, and the impact of adapting (fine-tuning) the representations to new operating conditions (different acquisition protocols, such as writing instruments and scan resolution).

  Related publication:

  Fixed-sized representation learning from Offline Handwritten Signatures of different sizes (*published in Springer's International Journal on Document Analysis and Recognition (IJDAR), 2018*)

- Chapter 4 considers an approach to directly address the issue of partial knowledge during training, by formulating signature verification as a meta-learning problem. This formulation directly optimizes for the objective of separating genuine signatures and forgeries, even when forgeries are not available for all users.

Related publication:

Meta-learning for fast classifier adaptation to new users of Signature Verification systems (*to be submitted to the IEEE Transactions on Information Forensics and Security*)

- Chapter 5 explores the limitations of signature verification systems under the presence of an active adversary. The issue of adversarial examples is characterized for signature verification, under an existing taxonomy of threats to biometric systems.

  Related publication:

  Characterizing and evaluating adversarial examples for Offline Handwritten Signature Verification (*published in IEEE Transactions on Information Forensics and Security, 2019*)

To facilitate further research on this field, all the code associated to the papers above was made publicly available [1] [2].

An additional contribution was made in the field of adversarial machine learning, by developing a fast algorithm to generate adversarial examples, and its usage for training defense mechanisms (joint work with colleagues, described in appendix I). This method was used to win one of the competitions of the NIPS 2018 Adversarial Vision Challenge (Brendel *et al.* (2018b)) [3]:

Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses (*accepted to the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2019)*)

[1] https://github.com/luizgh/sigver_wiwd

[2] https://github.com/luizgh/adversarial_signatures

[3] https://medium.com/bethgelab/results-of-the-nips-adversarial-vision-challenge-2018-e1e21b690149

Organization of the thesis

This is a thesis by articles, with Chapters 2 to 5 in the main text corresponding to a journal paper. Chapter 1 presents a literature review on Offline Handwritten Signature Verification. This chapter is based on a literature review published in the 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA) (Hafemann *et al.* (2017b)), that was updated with the recent advancements on the field.

The second chapter presents a method to learn Writer-Independent features from a collection of users, that is subsequently used to train Writer-Dependent classifiers. In particular, it introduces 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). This feature representation is used to train classifiers for another set of users, for which skilled forgeries are not available. Preliminary results of this method were presented in two conference papers (Hafemann *et al.* (2016b,a)), and the complete paper was published in Elsevier's Pattern Recognition (Hafemann *et al.* (2017a)).

The third chapter explores three issues with signature verification systems that rely on learned features: (i) How to obtain a fixed-sized vector representation for signatures of varied size, (ii) How the resolution of the scanned signatures impact system performance, and (iii) the impact of fine-tuning representations to other operating conditions (e.g. different acquisition protocols, signatures from people of different locations), by using transfer learning to other datasets. This chapter was published in Springer's International Journal on Document Analysis and Recognition (IJDAR) (Hafemann *et al.* (2018)).

In the fourth chapter, we proposed to formulate Signature Verification as a meta-learning problem. This approach enables directly optimizing for the metric of interest (separating genuine signatures and forgeries), by considering two levels of learning: a task-level (learning a classifier for a particular user), and a meta-level (leverage knowledge across multiple tasks). In particular, the meta-learner can guide the user classifiers to be discriminative of forgeries, even when the classifiers themselves are not trained with them. We also show that this method naturally extends for a one-class formulation, such that for a new user, only genuine signatures from the users is necessary for training the classifier.

The fifth chapter investigates the limitations of signature verification systems in the presence of an active adversary. In particular, it investigates the impact of Adversarial examples (examples crafted to induce misclassifications) on systems that used learned features, as well as a handcrafted feature extractor (CLBP), considering different scenarios of attacker's goal and knowledge of the system. The contents of this chapter were published in the IEEE Transactions on Information Forensics and Security (Hafemann *et al.* (2019)).

Appendix I presents a fast method for generating adversarial examples, that is also used in adversarial training for learning a model that is more robust against this type of attack. The contents of this chapter will be published in the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (Rony *et al.* (2018)).

Appendix II lists the supplementary material for the contribution described in Chapter 5, with a comprehensive list of experiments conducted on four datasets.

CHAPTER 1

LITERATURE REVIEW

The area of automatic Offline Signature Verification has been researched since the decade of 1970. Over the years, the problem has been addressed from many different perspectives, as summarized by surveys in the late 80's by Plamondon & Lorette (1989), 90's by Leclerc & Plamondon (1994) and 2000's by Impedovo & Pirlo (2008). In this chapter we summarize the main concepts relevant to the field, the main methods used for addressing the problem, and the recent work on the area. This chapter is based on a literature review published as a conference paper (Hafemann *et al.* (2017b)), and was updated to include recent developments in the field of signature verification, as well as the literature on adversarial examples, which is the subject of chapter 5.

The task of automated signature verification refers to verifying the identity of a person based on his/her handwritten signature. During a system's operation, a user claims an identity and provides a signature sample. This sample is then evaluated using a reference set of genuine signatures from the claimed individual (or a model built using them), and it is classified as genuine (created by the claimed individual) or a forgery (created by someone else).

Forgeries are often classified in three types: random, simple and skilled (or simulated) forgeries. In the case of random forgeries, the forger has no information about the user or his signature and uses his own signature instead. In this case, the forgery contains a different semantic meaning than the genuine signatures from the user, presenting a very different overall shape. In the case of simple forgeries, the forger has knowledge of the user's name, but not about the user's signature. In this case, the forgery may present more similarities to the genuine signature, in particular for users that sign with their full name, or part of it. In skilled forgeries, the forger has access for both the user's name and signature, and often practices imitating the user's signature. This result in forgeries that have higher resemblance to the genuine signature, and therefore are harder to detect.

This problem can be formulated as a Pattern Recognition task: given a dataset of genuine signatures from the users enrolled in the system, a classifier is trained for a user, so that it can discriminate new samples (not seen during training) as genuine signatures or forgeries. It follows a standard Pattern Classification pipeline:

- **Data acquisition**: The first step is to obtain the data. For the problem at hand, this is accomplished by scanning the documents containing the signatures.
- **Preprocessing**: After the document has been scanned, the signature is extracted from the document and image processing techniques are applied to reduce noise or in general enhance the quality of the samples.
- **Feature Extraction**: This step consists in obtaining a set of measurements from the samples. Given a preprocessed signature image, this step produces a real-valued feature vector.
- **Training**: Once we have feature vectors from the samples, we train a machine learning model, by optimizing its parameters to minimize a cost function in the training set.

The initial steps of data acquisition and signature extraction are not considered in the majority of the studies, that already consider individual signature images as the input for the systems. As an exception, this has been explored for bank cheques, that contain a complex background (Dimauro *et al.* (1997), Djeziri *et al.* (1998)). Other preprocessing steps vary among different studies, but the majority use simple techniques such as noise removal, size normalization and centering (Huang & Yan (1997), Pourshahabi *et al.* (2009)), and a variety of morphological transformations such as binarization and thinning (Huang & Yan (1997), Ferrer *et al.* (2005)). The following sections detail the developements on the other steps of this pipeline.

1.1 Preprocessing

As with most pattern recognition problems, preprocessing plays an important role in signature verification. Signature images may present variations in terms of pen thickness, scale, rotation, etc., even among authentic signatures of a person. Bellow we summarize the main preprocessing techniques:

- **Signature extraction** - This is an initial step that consists in finding and extracting a signature from a document. This is a particular challenging problem in bank cheques, where the signature is often written on top of a complex background (Dimauro *et al.* (1997), Djeziri *et al.* (1998)). This step is, however, not considered in most signature verification studies, that already consider signatures extracted from the documents.

- **Noise Removal** - Scanned signature images often contain noise. A common strategy to address this problem is to apply a noise removal filter to the image, such a median filter (Huang & Yan (1997)). It is also common to apply morphological operations to fill small holes and remove small regions of connected components (Huang & Yan (1997), Yılmaz & Yanıkoğlu (2016)).

- **Size normalization and centering** - Depending on the properties of the features to be used, different size normalization strategies are adopted. The simplest strategy is to crop the signature images to have a tight box on the signature (Ghandali & Moghaddam (2008)). Another strategy is to user a narrower bounding box, such as cutting strokes that are far from the image centroid, that are often subject to more variance in a user's signature (Yılmaz & Yanıkoğlu (2016)). Other authors use a fixed frame size (width and height), and center the signature in this frame (Pourshahabi *et al.* (2009), Hafemann *et al.* (2017a)) .

- **Signature representation** - Besides just using the gray-level image as input to the feature extractors, other representations have been considered. For instance, using the signature's skeleton, outline, ink distribution, high pressure regions and directional frontiers (Huang & Yan (1997)).

- **Signature Alignment** - alignment is a common strategy in online signature verification, but not broadly applied for the offline scenario. Yılmaz & Yanıkoğlu (2016) propose aligning the signatures for training, by applying rotation, scaling and translation. Kalera *et al.* (2004)

propose a method to perform Rotation normalization, using first and second order moments of the signature image.

1.2 Feature Extraction

Offline signature verification has been studied from many perspectives, yielding multiple alternatives for feature extraction. Broadly speaking, the feature extraction techniques can be classified as *Static* or *Pseudo-dynamic*, where pseudo-dynamic features attempt to recover dynamic information from the signature execution process (such as speed, pressure, etc.). Another broad categorization of the feature extraction methods is between *Global* and *Local* features. Global features describe the signature images as a whole - for example, features such as height, width of the signature, or in general feature extractors that are applied to the entire signature image. In contrast, local features describe parts of the images, either by segmenting the image (e.g. according to connected components) or most commonly by the dividing the image in a grid (of Cartesian or polar coordinates), and applying feature extractors in each part of the image.

Recent studies approach the problem from a representation learning perspective (Hafemann *et al.* (2016b), Hafemann *et al.* (2017a), Rantzsch *et al.* (2016), Zhang *et al.* (2016)): instead of designing feature extractors for the task, these methods rely on learning feature representations directly from signature images.

1.2.1 Handcrafted feature extractors

A large part of the research efforts on the field has been devoted to finding good feature representations for offline signatures. In this section we summarize the main descriptors proposed for the problem.

1.2.1.1 Geometric Features

Geometric features measure the overall shape of a signature. This includes basic descriptors, such as the signature height, width, caliber (height-to-width ration) and area. More complex descriptors include the count of endpoints and closed loops (Baltzakis & Papamarkos (2001)). Besides using global descriptors, several authors also generate local geometric features by dividing the signature in a grid and calculating features from each cell. For example, using the pixel density within grids (Baltzakis & Papamarkos (2001), El-Yacoubi *et al.* (2000), Justino *et al.* (2000)).

1.2.1.2 Graphometric features

Forensic document examiners use the concepts of graphology and graphometry to examine handwriting for several purposes, including detecting authenticity and forgery. Oliveira *et al.* (2005) investigated applying such features for automated signature verification. They selected a subset of graphometric features that could be described algorithmically, and proposed a set of feature descriptors. They considered the following static features: *Calibre* - the ratio of Height / Width of the image; *Proportion*, referring to the symmetry of the signature, *Alignment to baseline* - describing the angular displacement to an horizontal baseline, and *Spacing* - describing empty spaces between strokes.

1.2.1.3 Directional features

Directional features seek to describe the image in terms of the direction of the strokes in the signature. Sabourin & Drouhard (1992) and Drouhard *et al.* (1996) extracted Directional-PDF (Probability Density Function) from the gradient of the signature outline. Rivard *et al.* (2013) used this method of feature extraction using grids of multiple scales. Zhang (2010) investigated the usage of pyramid histogram of oriented gradients (PHOG). This descriptor represents local shapes in a image by a histogram of edge orientations, also in multiple scales.

1.2.1.4 Mathematical transformations

Researchers have used a variety of mathematical transformations as feature extractors. Nemcek & Lin (1974) investigated the usage of a fast *Hadamart* transform and spectrum analysis for feature extraction. Pourshahabi *et al.* (2009) used a *Contourlet* transform as feature extraction, stating that it is an appropriate tool for capturing smooth contours. Coetzer (2005) used the discrete *Radon* transform to extract sequences of observations, for a subsequent HMM training. Deng *et al.* (1999) proposed a signature verification system based on the *Wavelet* transform. Zouari *et al.* (2014) investigated the usage of the *Fractal* transform for the problem.

1.2.1.5 Shadow-code

Sabourin & Genest (1994) proposed an Extended Shadow Code for signature verification. A grid is overlaid on top of the signature image, containing horizontal, vertical and diagonal bars, each bar containing a fixed number of bins. Each pixel of the signature image is then projected to its closest bar in each direction, activating the respective bin. The count of active bins in the projections is then used as a descriptor of the signature. This feature extractor has been used by Rivard *et al.* (2013) and Eskander *et al.* (2013) with multiple resolutions, together with directional features, to achieve promising results on writer-independent and writer-dependent classification, respectively.

1.2.1.6 Texture features

Texture features, in particular variants of Local Binary Patterns (LBP), have been used in many experiments in recent years. The LBP operator describe the local patterns in the image, and the histogram of these patterns is used as a feature descriptor. LBP variantions have been used in many studies (Yilmaz *et al.* (2011), Yılmaz & Yanıkoğlu (2016), Serdouk *et al.* (2014), Serdouk *et al.* (2015b), Hu & Chen (2013)), and have demonstrated to be among the best handcrafted feature extractors for this task. Another important texture descriptor is GLCM (Gray Level Co-occurrence Matrix). This feature uses relative frequencies of neighboring pixels, and was used in a few papers (Hu & Chen (2013), Vargas *et al.* (2011)).

1.2.1.7 Interest point matching

Interest point matching methods, such as SIFT (Scale-Invariant Feature Transform) and SURF (Speeded Up Robust Features) have been largely used for computer vision tasks. Ruiz-del Solar *et al.* (2008) used SIFT to extract local interest points from both query and reference samples to build a writer-dependent classifier. After extracting interest points from both images, they generated a set of 12 features, using information such as the number of SIFT matches between the two images, and processing time. Malik *et al.* (2014) used SURF to extract interest points in the signature images, and used these features to assess the local stability of the signatures. During classification, only the stable interest points are used for matching. The number of keypoints in the query image, and the number of matched keypoints were used to classify the signature as genuine or forgery.

1.2.1.8 Pseudo-dynamic features

Oliveira *et al.* (2005) presented a set of pseudo-dynamic features, based on graphometric studies: *Distribution of pixels*, *Progression* - that measures the tension in the strokes, providing information about the speed, continuity and uniformity, *Slant* and *Form* - measuring the concavities in the signature.

More recently, Bertolini *et al.* (2010) proposed a descriptor that considers the *curvature* of the signature. This was accomplished by fitting Benzier curves to the signature outline (more specially, to the largest segment of the signature), and using the parameters of the curves as features.

1.2.2 Deep learning

There has been an increased interest in recent years on techniques that do not rely on hand-engineered feature extractors. Instead, feature representations are learned from *raw* data (pixels, in the case of images). This is the case of Deep Learning models (Bengio (2009), LeCun *et al.* (2015)).

Early work applying representation learning for the task used private datasets and did not report much success: Ribeiro *et al.* (2011) used RBMs to learn a representation for signatures, but only reported a visual representation of the learned weights, and not the results of using such features to discriminate between genuine signatures and forgeries. Khalajzadeh *et al.* (2012) used CNNs for Persian signature verification, but only considered random forgeries in their tests.

Considering work that targeted the classification between genuine signatures and skilled forgeries, we find two main approaches in recent literature: 1) learning writer-independent features in a subset of users, to be used for training writer-dependent classifiers; 2) learning feature representations and a writer-independent system at once, using metric learning.

Concurrent to the work on this thesis, Rantzsch *et al.* (2016) proposed a Writer-Independent approach using metric learning. In this approach, the system learns a distance between signatures. During training, tuples composed of three signatures are fed to the network: (X_r, X_+, X_-), where X_r is a reference signature, X_+ is a genuine signature from the same user, and X_- is a forgery (either a random or skilled forgery). The system is trained to minimize the distance between X_r and X_+, and maximize the distance between X_r and X_-. The central idea is to a learn a feature representation that will therefore assign small distances when comparing a genuine signature to another (reference) genuine signature, and larger distances when comparing a skilled forgery with a reference. Zhang *et al.* (2016) proposed using Generative Adversarial Networks (GANs) (Goodfellow *et al.* (2014)) for learning the features from a subset of users. In this case, two networks are trained: a generator, that learns to generates signatures, and a discriminator, that learns to discriminate if an image is from a real signature or one that was automatically generated. After training, the authors used the convolutional layers of the discriminator as the features for new signatures. Zois *et al.* (2018a,b) consider sparse coding and dictionary learning for learning representations, that are subsequently used for training writer-dependent classifiers. Tsourounis *et al.* (2018) considered a Deep Sparse Coding method, using sparse representations from different layers, that are processed with a Local Spatial Pooling layer followed by using dimensionaly reduction.

1.3 Model Training

Classifiers for signature verification can be broadly classified in two groups: *writer-dependent* and *writer-independent*. In the first case, which is more common in the literature, a model is trained for each user, using the user's genuine signatures, and random forgeries (by using genuine signature from other users). During the operations phase, the model trained for the claimed identity is used to classify query signatures as genuine or forgery. The writer-independent approach, on the other hand, involves only a single classifier for all users. In this case, the system learn to compare a query signature with a reference. During the test phase, the model is used to compare a query signature with reference genuine samples from the claimed individual, to make a decision. One common way of training WI systems is to use a dissimilarity representation, where the inputs to the classifiers are differences between two feature vectors: $|x_q - x_r|$, with a binary label that indicates whether the two signatures are from the same user or not (Rivard *et al.* (2013); Eskander *et al.* (2013)).

Some authors use a combination of both approaches. For example, Eskander *et al.* (2013) and Zhang *et al.* (2016) trained hybrid writer-independent-writer-dependent solutions, where a writer-independent classifier is used for classification when only a few genuine signatures are available. When the number of collected genuine samples passes a threshold, a writer-dependent classifier is trained for the user. Yılmaz & Yanıkoğlu (2016) propose a hybrid approach, where the results of both a writer-independent and writer-dependent classifiers are combined.

Besides the most basic classifiers (e.g. simple thresholding and nearest-neighbors), several strategies have been tried for the task of signature verification. The following sections cover the main models used for the task.

1.3.1 Hidden Markov Models

Several authors have proposed using Hidden Markov Models for the task of signature verification (Justino *et al.* (2000), Oliveira *et al.* (2005), Batista *et al.* (2012)). In particular, HMMs with a left-to-right topology have been mostly studied, as they match the dynamic characteristics of American and European handwriting (with hand movements from left to right).

In the work from Justino *et al.* (2000), Oliveira *et al.* (2005) and Batista *et al.* (2012), the signatures are divided in a grid format. Each column of the grid is used as an observation of the HMM, and features are extracted from the different cells within each column, and subsequently quantized in a codebook. In the verification phase, a sequence of feature vectors is extracted from the signature and quantized using the codebook. The HMM is then used to calculate the likelihood of the observations given the model. After calculating the likelihood, a simple threshold can be used to discriminate between genuine signatures and forgeries (Justino *et al.* (2000)), or the likelihood itself can be used for more complex classification mechanisms (Batista *et al.* (2012)).

1.3.2 Support Vector Machines

Support Vector Machines have been extensively used for signature verification, for both writer-dependent and writer-independent classification (Özgündüz *et al.* (2005), Justino *et al.* (2005), Bertolini *et al.* (2010), Kumar *et al.* (2012), Yılmaz & Yanıkoğlu (2016), Hafemann *et al.* (2017a)), empirically showing to be the one of the most effective classifiers for the task. In recent years, Guerbai *et al.* (2015) used One-Class SVMs for the task. This type of model attempt to only model one class (in the case of signature verification, only the genuine signatures), which is a desirable property, since for the actual users enrolled in the system we only have the genuine signatures to train the model. However, the low number of genuine signatures present an important challenge for this strategy.

1.3.3 Neural Networks and Deep Learning

Neural Networks have been explored for both writer-dependent and writer-independent systems. Huang & Yan (1997) used Neural Networks to classify between genuine signatures and random and targeted forgeries. They trained multiple networks on features extracted at different resolutions, and another network to make a decision, based on the outputs of these networks. Shekar *et al.* (2015) presented a comparison of neural networks and support vector machines in three datasets.

More recently, Soleimani *et al.* (2016) proposed a Deep Multitask Metric Learning (DMML) system for signature verification. In this approach, the system learns to compare two signatures, by learning a distance metric between them. The signatures are processed using a feedforward neural-network, where the bottom layers are shared among all users (i.e. the same weights are used), and the last layer is specific to each individual, and specializes for the individual. In the work of Rantzsch *et al.* (2016), a metric learning classifier is learned, jointly learning a feature representation, and a writer-independent classifier.

1.3.4 Ensemble of classifiers

Some authors have adopted strategies to train multiple classifiers, and combine their predictions when classifying a new sample. Bertolini *et al.* (2010) used a static ensemble selection with graphometric features. They generate a large pool of classifiers (trained with different grid sizes), and used a genetic algorithm to select a subset of the models, building an ensemble of classifiers. Batista *et al.* (2012) used dynamic selection of classifiers for building a writer-dependent system. They used a bank of HMMs as base classifiers, and for a given sample, the posterior likelihood is calculated for all HMMs. The set of likelihoods is considered as a feature vector, and a specialized random subspace method is used to train an ensemble of classifiers . Yılmaz & Yanıkoğlu (2016) proposed a system that combines writer-dependent and writer-independent models (trained with a variety of feature descriptors). The scores from all classifiers is subsequently aggregated using a linear combination, obtaining a final decision of the ensemble.

1.3.5 Data augmentation

One of the main challenges for building an automated signature verification system is the low number of samples per user for training. To address this issue, some researchers have proposed ways to generate more samples based on existing genuine signatures.

Huang & Yan (1997) have proposed a set of “perturbations" to be applied to each genuine signature, to generate new samples: slant, rotation, scaling and perspective. In their work, they considered a set of “slight distortions", used to create new genuine samples, and “heavy distortions" to generate forgeries from the genuine samples. More recently, signature synthesis approaches inspired on a neuromotor model have been proposed (Ferrer *et al.* (2013), Ferrer *et al.* (2015), Diaz *et al.* (2017)).

1.4 Security of biometric systems

The security of machine learning systems has been object of study in the past two decades. Barreno *et al.* (2006, 2010) categorize attacks to biometric systems along three axes: (i) the influence of the attack (causative or exploratory); (ii) the specificity of the attack (targeted or indiscriminate); and (iii) the security violation of the attack: integrity violation (e.g. intrusion) or availability disruption (e.g. make the system unusable for legitimate users).

Biggio *et al.* (2014, 2015) considered a model of the adversary that includes its *goals*, *knowledge* of the target system, and *capabilities* of manipulating the input data or system components. The goals of an attacker are mainly divided in: 1) *Denial of service*: preventing real users from using the system; 2) *Intrusion*: impersonating another user; 3) *Privacy violation*: stealing private information from an user (such as the biometric templates). The *knowledge* of the adversary refers to the information of the target system that is available to the adversary, such as perfect knowledge (e.g. knowledge of the feature extractor, type of classifier and model parameters) or limited (partial) knowledge of the system. The *capabilities* of the adversary refer to what it can *change* in the target system, such as changing the training set (poisoning attack), or the inputs to the system at test time (evasion attack).

1.4.1 Adversarial Examples

Adversarial examples are a relatively new threat identified for machine learning systems. These are samples similar to the true data distribution, but that fool a classifier. In computer vision tasks, these are images $\tilde{X}$ that are visually similar to an image X (from the true data distribution), but that fool a classifier (i.e. the classifier predicts an incorrect class for $\tilde{X}$: $\text{argmax}_y P(y|\tilde{X}) \neq \text{argmax}_y P(y|X)$).

Szegedy *et al.* (2014) showed that for deep neural networks, an optimization procedure to be used to find a small change δ to an image, such that $\tilde{X} = X + \delta$ is an adversarial example. They found that in many cases, this change in imperceptible for a person, while inducing a misclassification in the system. Perhaps more surprisingly, they also discovered that an attack that is created to fool one network also fools other networks (trained on different subsets of data), meaning that attacks can be created even without full knowledge of the classifier under attack. It was later shown that such attacks can be done in the physical world (Kurakin *et al.* (2017a)), where adversarial images printed on paper and later captured with a camera also fooled a classifier. During the past few years, several attacks have been proposed in the literature (Goodfellow *et al.* (2015); Moosavi-Dezfooli *et al.* (2016); Carlini & Wagner (2017b)), as well as several defense strategies (Goodfellow *et al.* (2015), Tramèr *et al.* (2018), Papernot *et al.* (2016), Guo *et al.* (2018), Madry *et al.* (2018)). However, this problem remains an open research question, as these defenses have either shown not to be robust against strong iterative attacks, or not to scale to larger datasets (Athalye *et al.* (2018a); Athalye & Carlini (2018)). Lastly, it is worth noting that even *detecting* that an input is adversarial is a hard task, and it has been shown that such detectors can also be easily fooled (Carlini & Wagner (2017a)).

1.5 Summary

Reviewing the current literature on Signature Verification, we identify that separating genuine signatures from skilled forgeries remains a challenging task, and we identify the feature representation as being a key factor limiting the performance of such systems. Therefore, learning features from signature data is the main topic of this thesis, which is explored in chapters 2, 3 and 4, by proposing formulations of the problem that allow learning features using convolutional neural networks, directly from signature images.

Another key challenge is the presence of partial knowledge during training: we cannot expect to have skilled forgeries available for every user of the system, since these are only obtained during actual forgery attempts. We address this problem in two ways: in chapter 2, we consider a formulation for feature learning that includes knowledge of forgeries, by optimizing a multi-task loss. This allows the features to be discriminant of visual cues commonly present in forgeries (e.g. limp strokes), that generalize to new users, for which we only have genuine signatures for training. In chapter 4 we consider a meta-learning problem, with two levels of learning: a task-level (learning a classifier for a user) and a meta-level (learning across tasks). This addresses the issue of partial knowledge by employing only genuine signatures on the task-level, and forgeries (if available for some users) as part of the meta-level learning. Therefore, the meta-learner can guide the task-level classifiers to be discriminative of forgeries, even if the task-level classifiers themselves do not use forgeries for training.

Having a low number of samples per user is another key aspect of signature verification. In this thesis, we address this issue by (i) learning features using data from multiple users (chapters 2, 3 and 4), and (ii) modeling the system as a meta-learning problem, such that the meta-learner have access to signatures from many users (chapter 4).

Finally, recent research on adversarial machine learning highlights the threat of *adversarial examples*, that affects most machine learning systems, and in particular deep learning. While most research in the literature was conducted in classification tasks, we evaluate the threats that they pose for signature verification. We identify important differences that arise in verification problems: in such systems, each new user effectively introduces a new *class*, which raises questions about the transferability of attacks in scenarios where the attacker does not have full access to the system under attack. Therefore, we conduct a thorough study to characterize the problem of adversarial examples in the context of biometric system security, identifying new threats to handwritten signature verification, and evaluating attack performance under different scenarios of attacker knowledge.

CHAPTER 2

LEARNING FEATURES FOR OFFLINE HANDWRITTEN SIGNATURE VERIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

Luiz G. Hafemann[1], Robert Sabourin[1], Luiz S. Oliveira[2]

[1] Department of Automated Manufacturing Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Departement of Informatics, Federal University of Parana (UFPR),
Curitiba, PR, Brazil

Article Published in « Pattern Recognition » 2017.

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.

2.1 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 (Plamondon & Srihari (2000)).

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 (Impedovo & Pirlo (2008)). 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. 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 (Hafemann *et al.* (2017b)). Recent work uses texture features, such as Local Binary Patterns (LBP) (Yılmaz & Yanıkoğlu (2016), Hu & Chen (2013)) and Gray-Level Co-occurrence Matrix (GLCM) (Hu & Chen (2013)); directional-based features such as Histogram of Oriented Gradients (HOG) (Yılmaz & Yanıkoğlu (2016)) and Directional-PDF (Rivard *et al.* (2013), Eskander *et al.* (2013)); feature extractors specifically designed for signatures, such as the estimation of strokes by fitting Bezier curves (Bertolini *et al.* (2010)); among others. No feature extractor has emerged as particularly suitable for signature verification, and most recent work uses a combination of many such techniques.

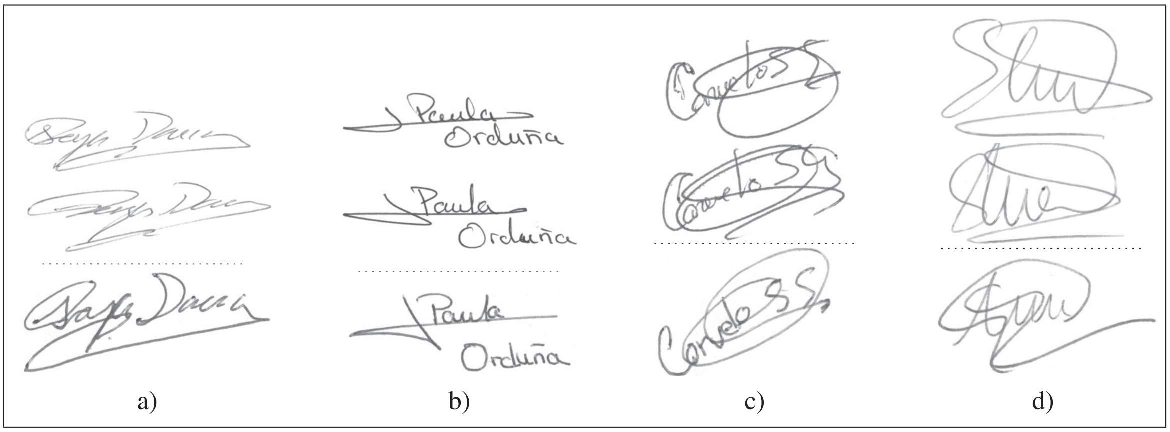


Figure 2.1 Examples of challenges in designing feature extractors for offline signatures, and the challenge of classifying skilled forgeries. Each column shows two genuine signatures from the same user in the GPDS dataset, and a skilled forgery created for the user. We notice that skilled forgeries resemble 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.

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 (Vargas *et al.* (2007)), the best reported results achieve 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 forgeries), 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.

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

Preliminary results, using only genuine signatures for learning the features, were published as two conference papers. In (Hafemann *et al.* (2016b)), 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 (Hafemann *et al.* (2016a)), 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 2.2 reviews the related work on signature verification and on feature learning techniques. Section 2.3 details the formulation and methodology to learn features for offline signature verification, and section 2.4 describes our experimental protocol. Section 2.5 presents and discusses the results of our experiments. Lastly, section 2.6 concludes the paper.

2.2 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.

2.2.1 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 Plamondon & Lorette (1989), Leclerc & Plamondon (1994) and Impedovo & Pirlo (2008).

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 (Bertolini *et al.* (2010), Eskander *et al.* (2013)). 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 (Nagel & Rosenfeld (1977), Justino *et al.* (2000)), descriptors inspired in graphology and graphometry (Oliveira *et al.* (2005)), directional-based descriptors such as HOG Yılmaz & Yanıkoğlu (2016) and D-PDF (Sabourin & Drouhard (1992), Rivard *et al.* (2013), Eskander *et al.* (2013)), interest-point based, such as SIFT (Yılmaz & Yanıkoğlu (2016)), to texture descriptors, such as Local Binary Patterns (LBP) (Yılmaz & Yanıkoğlu (2016)) and Gray-Level Co-occurrence Matrix (GLCM) (Hu & Chen (2013)). 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.* (2011) 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 *et al.* (2012) 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.* (2016) 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 *et al.* (2013) 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.

2.2.2 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 (Bengio (2009), Bengio (2013), LeCun *et al.* (2015)). 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 (LeCun *et al.* (2015)). In particular, Convolutional Neural Networks (CNNs) (LeCun *et al.* (1989)) have been used to achieve state-of-the-art performance in many computer vision tasks (LeCun *et al.* (2015), Krizhevsky *et al.* (2012), Szegedy *et al.* (2015)). 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 (LeCun *et al.* (2015)). 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 (Bengio *et al.* (2007))), and more recently with transfer learning (Donahue *et al.* (2014), Oquab *et al.* (2014), Nanni & Ghidoni (2017)). 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 (Donahue *et al.* (2014), Oquab *et al.* (2014)), or the last layer (that corresponds to the predictions for classes in the original task, as in (Nanni & Ghidoni (2017))), 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.

2.3 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 (Krizhevsky *et al.* (2012))). 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 $\mathscr{D}$ 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 $\mathscr{E}$, 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.

2.3.1 Learning features from genuine signatures

Let $\mathscr{D}$ be a dataset consisting of genuine signatures from a set of users $\mathscr{Y}_{\mathscr{D}}$. 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 $\mathscr{D}$. This formulation has been introduced in (Hafemann *et al.* (2016b)), and it is included here for completeness.

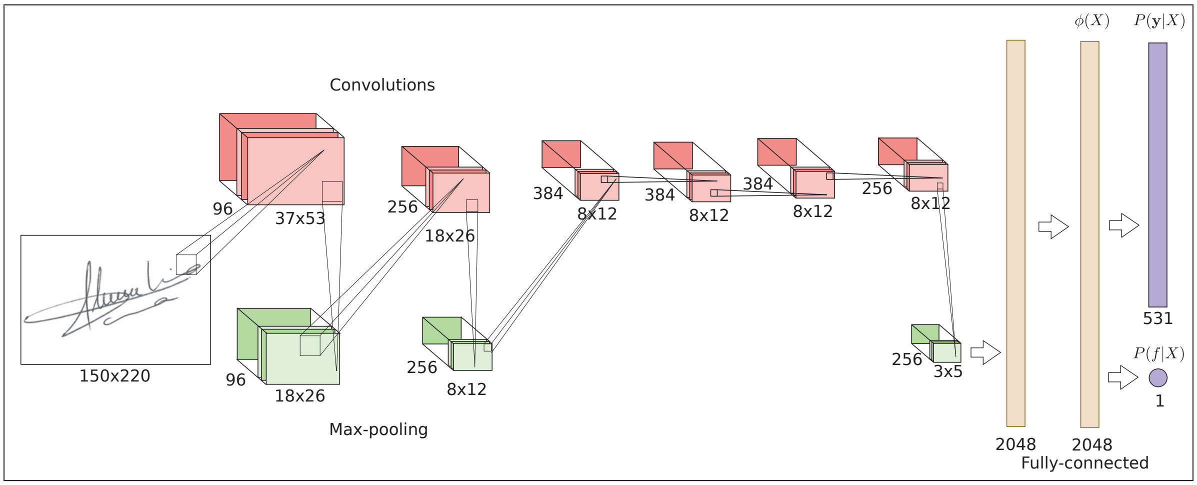


Figure 2.2 Illustration of a CNN architecture used in this work. The input image goes through a sequence of transformations with convolutional layers, max-pooling layers and fully-connected layers. During feature learning, $P(\mathbf{y}|X)$ (and also $P(f|X)$ in the formulation from sec 2.3.2.2) are 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 $\phi(X)$.

Formally, we consider a training set composed of tuples (X, y) where X is the signature image, and y is the user, that is, $y \in \mathscr{Y}_{\mathscr{D}}$. 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, ($M = |\mathscr{Y}_{\mathscr{D}}|$), and estimates $P(\mathbf{y}|X)$. Figure 2.2 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:

$$L = -\sum_j y_{ij} \log P(y_j|X_i) \tag{2.1}$$

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.

2.3.2 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 $\mathscr{D}$ contains genuine signatures and skilled forgeries for a set of users, while the exploitation set $\mathscr{E}$ 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.

2.3.2.1 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, $M = 2|\mathscr{Y}_{\mathscr{D}}|$.

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.

2.3.2.2 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 2.2, where the user classification loss (L_c) is a multi-class cross-entropy, and the forgery classification (L_f) is a binary cross-entropy:

$$\begin{aligned} L_c &= -\sum_j y_{ij} \log P(y_j|X_i) \\ L_f &= -f_i \log(P(f|X_i)) - (1-f_i)\log(1-P(f|X_i)) \end{aligned} \tag{2.2}$$

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:

$$\begin{aligned} L_1 &= (1-\lambda)L_c + \lambda L_f \\ &= -(1-\lambda)\sum_j y_{ij} \log P(y_j|X_i) + \\ &\lambda\big(-f_i \log(P(f|X_i)) - (1-f_i)\log(1-P(f|X_i))\big) \end{aligned} \tag{2.3}$$

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

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

$$\begin{aligned} L_2 &= (1-f_i)(1-\lambda)L_c + \lambda L_f \\ &= -(1-f_i)(1-\lambda)\sum_j y_{ij}\log P(y_j|X_i) + \\ \lambda\big(&-f_i\log(P(f|X_i)) - (1-f_i)\log(1-P(f|X_i))\big) \end{aligned} \tag{2.4}$$

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 2.3.1, where only genuine signatures are used for training, since the forgeries would not contribute to the loss function.

2.3.3 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 (Otsu (1975)), 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.

2.3.4 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.

Table 2.1 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	

In order to use a suitable architecture for signature verification, in (Hafemann *et al.* (2016a)) we investigated different architectures for learning feature representations using the objective from section 2.3.1 (training using only genuine signatures). In this work we use the architecture that performed best for this formulation, which is described in table 2.1. 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 (Ioffe & Szegedy (2015)), 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 $\mathscr{Y}_{\mathscr{D}}$. For the formulation in section 2.3.2.2, the neuron that estimates $P(f|X)$ uses the sigmoid function. Both output layers receive as input the result of layer FC7. Table 2.2 lists the operations mentioned above.

Optimization was conducted by minimizing the loss with Stochastic Gradient Descent with Nesterov Momentum, with a momentum factor of 0.9. In each iteration, a batch of 32 images is used, and the loss function is averaged across all images in the batch (equations 2.1 to 2.4, depending on the loss being considered). 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 (Ioffe & Szegedy (2015)), 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.

Table 2.2 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	$\mathrm{ReLU}(z_i) = \max(0, z_i)$
Sigmoid	$\sigma(z_i) = \frac{1}{1+e^{-z_i}}$
Softmax	$\mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$
Batch Normalization	$\mathrm{BN}(z_i) = \gamma_i \hat{z}_i + \beta_i,$
	$\hat{z}_i = \frac{z_i - \mathrm{E}[z_i]}{\sqrt{\mathrm{Var}[z_i]}}$

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

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.

2.3.5 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 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 (Donahue *et al.* (2014)) 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 (Osuna *et al.* (1997)):

$$
\begin{aligned}
& \min \frac{1}{2}\|\mathbf{w}\|^2 + C^+\Big(\sum_{i:y_i=+1}\xi_i\Big) + C^-\Big(\sum_{i:y_i=-1}\xi_i\Big) \\
& \text{subject to} \\
& \qquad y_i(\mathbf{w}x_i + b) \geq 1 - \xi_i \\
& \qquad \xi_i \geq 0
\end{aligned}
\tag{2.5}
$$

Where the change to the standard SVM formulation is the usage of different weights C for the two classes (we refer the reader to (Osuna *et al.* (1997)) 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:

$$\psi = \frac{N}{P} \qquad\qquad C^+ = \psi C^- \tag{2.6}$$

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.

2.4 Experimental Protocol

We conducted experiments using the datasets GPDS-960 (Vargas *et al.* (2007)), MCYT-75 (Ortega-Garcia *et al.* (2003)), CEDAR (Kalera *et al.* (2004)) and the Brazilian PUC-PR (Freitas *et al.* (2000)). Table 2.3 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 $\mathscr{D}$) and evaluating how these features generalize to other users in this dataset (the exploitation set $\mathscr{E}$). To enable comparison with previous work, we performed experiments on GPDS having the set $\mathscr{E}$ 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.

Table 2.3 Summary of the datasets used in this work

Dataset Name	**Users**	**Genuine signatures**	**Forgeries**	$S_{\textbf{canvas}}$
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

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

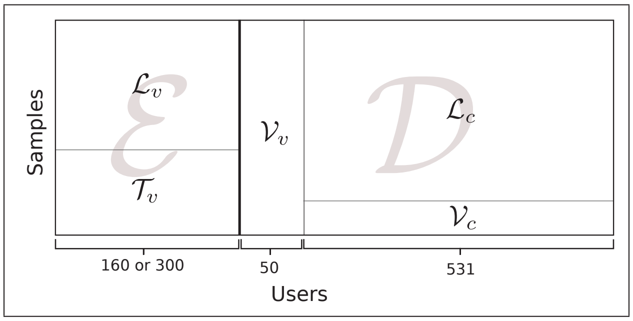


Figure 2.3 The GPDS dataset is separated into an exploitation set $\mathscr{E}$ and Development set $\mathscr{D}$. The development set is used for learning the features, and making all model decisions. The exploitation set represents the users enrolled to the system, where we train Writer-Dependent classifiers using only genuine signatures.

The GPDS dataset is divided as follows, as illustrated in Figure 2.3: The Convolutional Neural Networks are trained on a set $\mathscr{L}_c$ (denoting **L**earning set for **c**lassification) consisting of 531 users. We monitor the progress on a validation set $\mathscr{V}_c$ (**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 $\mathscr{V}_v$ (**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 $\mathscr{L}_c$ 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 $\mathscr{E}$. We trained Support Vector Machines on the set $\mathscr{L}_v$ (denoting **L**earning set for **v**erification) and tested on $\mathscr{T}_v$ (**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 2.4. For the GPDS and the Brazilian PUC-PR datasets, we used signatures from users that are not in the Exploitation set as 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.

Table 2.4 Separation into training and testing for each dataset

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

We used the same hyperparameters for training the SVM classifiers as in previous work (Hafemann *et al.* (2016a)): for the linear SVM, we used $C^{-} = 1$ (C^{+} is calculated according to equation 2.6). 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 2.3.2.2, 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 (Razavian *et al.* (2014)), 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 (Krizhevsky *et al.* (2012))), and VGG-19 (Simonyan & Zisserman (2014)). 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 (FAR_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 $\text{AER}_\text{genuine + skilled} = (\text{FRR} + \text{FAR}_\text{skilled})/2$.

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

2.5 Results and Discussion

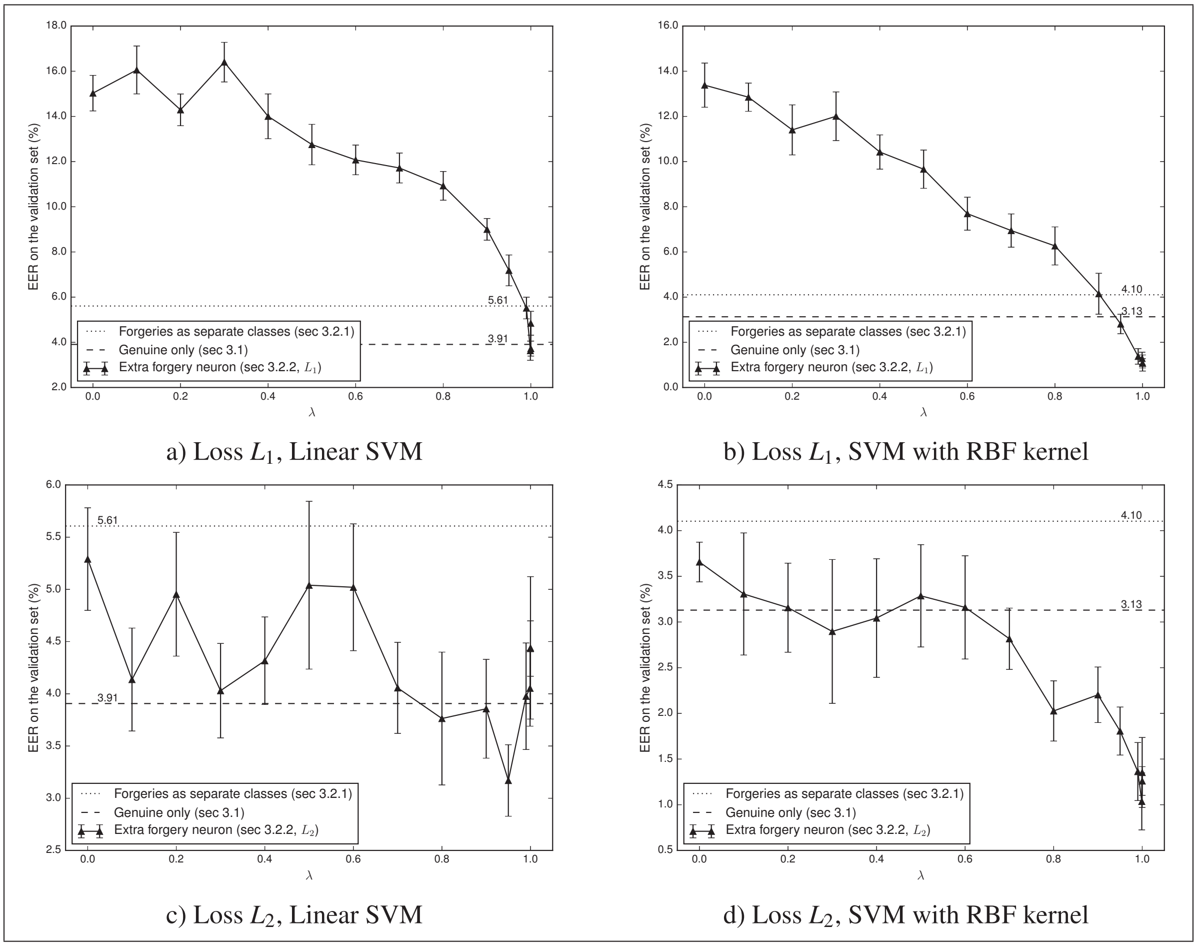
The experimental results with the proposed method are listed and discussed in this section. The first part presents the experiments on the Development set, which was used for making all the design decisions for the proposed method: evaluating different loss functions and other hyper-parameters. The second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

2.5.1 Signature Verification System Design

In these experiments, we trained the CNN architectures using the loss functions defined in section 2.3, 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 2.3.2.2, 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 2.4 shows the results on the validation set using loss L_1 (from equation 2.3), and loss L_2 (from equation 2.4). 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 2.3.2.1).

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



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 2.4 Performance on the validation set ($\mathscr{V}_v$), using features learned from genuine signatures and forgeries (sec 2.3.2.2), as we vary the hyperparameter λ. For reference, the performance of models using features learned from genuine signatures only (sec 2.3.1) and using forgeries as different classes (sec 2.3.2.1) are also included.

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 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 2.5 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 2.3.2.1) 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

Table 2.5 Performance of the WD classifiers on the validation set $\mathcal{V}_v$ (subset of 50 users in GPDS; Errors and Standard deviations in %)

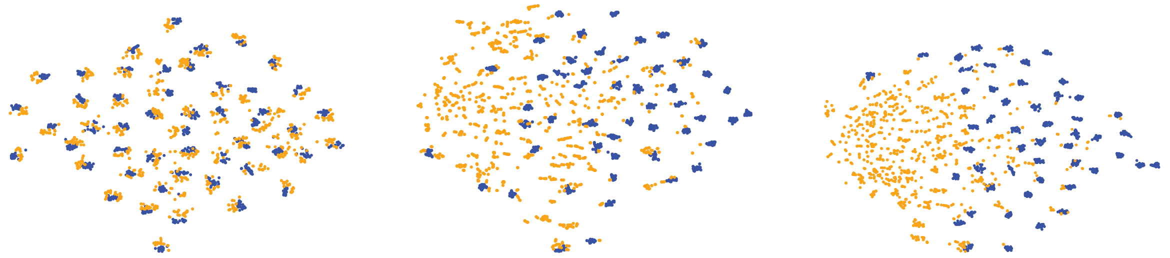
Classifier	**Formulation used to learn the features**	**EER$_{\textbf{global threshold}}$**	**EER$_{\textbf{user thresholds}}$**	**Mean AUC**
Linear SVM	Baseline (Caffenet, layer pool5)	14.09 (+- 2.80)	10.59 (+- 2.96)	0.9453 (+- 0.0198)
	Using genuine signatures only (sec 2.3.1)	6.80 (+- 0.57)	3.91 (+- 0.64)	0.9876 (+- 0.0022)
	Forgeries as separate classes (sec 2.3.2.1)	9.45 (+- 0.51)	5.61 (+- 0.63)	0.9749 (+- 0.0028)
	Forgery neuron (sec 2.3.2.2, loss L_1, $\lambda = 0.999$)	7.01 (+- 0.42)	3.63 (+- 0.43)	0.9844 (+- 0.0024)
	Forgery neuron (sec 2.3.2.2, 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 2.3.1)	5.93 (+- 0.43)	3.13 (+- 0.46)	0.9903 (+- 0.0018)
	Forgeries as separate classes (sec 2.3.2.1)	7.79 (+- 0.43)	4.10 (+- 0.41)	0.9857 (+- 0.0012)
	Forgery neuron (sec 2.3.2.2, loss L1, $\lambda = 1$)	2.41 (+- 0.32)	1.08 (+- 0.36)	0.9978 (+- 0.0008)
	Forgery neuron (sec 2.3.2.2, loss L2, $\lambda = 0.999$)	2.51 (+- 0.33)	1.04 (+- 0.31)	0.9971 (+- 0.0009)

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 2.3.1), 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.

2.5.1.1 Visualizing the learned representation space

We performed an analysis of the feature space learned by the models, by using the t-SNE algorithm (Van der Maaten & Hinton (2008)) 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 $\mathscr{L}_c$, 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. 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.



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 2.5 2D projections (using t-SNE) of the feature vectors from the 50 users in the validation set $\mathscr{V}_v$. Each point represents a signature sample: genuine signatures are displayed in blue (dark), while skilled forgeries are displayed in orange (light).

The result can be seen in Figure 2.5. 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.

2.5.2 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 $\mathscr{V}_v$, 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 used the architectures that performed best in the validation set, as seen in Table 2.5. In particular, we consider a model that was learned using genuine signatures only (sec 2.3.1), which we call simply by **SigNet** in this section. We also consider a model learned using genuine signatures and forgeries (sec 2.3.2.2), 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 $\lambda = 0.999$.

2.5.2.1 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 $\text{EER}_{\text{global threshold}}$ on the validation set $\mathscr{V}_v$) as a global threshold for all users. In this work, we do not explore techniques for setting user-specific thresholds, but simply report $\text{EER}_{\text{user thresholds}}$, which is the equal error rate obtained by using the optimal decision thresholds for each user.

Table 2.6 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,

Table 2.6 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	$\text{EER}_{\text{global threshold}}$	$\text{EER}_{\text{user thresholds}}$	Mean AUC
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)

which is consistent with previous findings that the definition of user-specific thresholds is key in obtaining 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 2.7 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 performance using SigNet-F to extract the features compared to using SigNet.

Table 2.7 Comparison with state-of-the art on the GPDS dataset (errors in %)

Reference	Dataset	#samples per user	Features	EER
Hu & Chen (2013)	GPDS-150	10	LBP, GLCM, HOG	7.66
Guerbai *et al.* (2015)	GPDS-160	12	Curvelet transform	15.07
Serdouk *et al.* (2015a)	GPDS-100	16	GLBP, LRF	12.52
Yılmaz & Yanıkoğlu (2016)	GPDS-160	5	LBP, HOG, SIFT	7.98
Yılmaz & Yanıkoğlu (2016)	GPDS-160	12	LBP, HOG, SIFT	6.97
Soleimani *et al.* (2016)	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)**

2.5.2.2 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 2.8, 2.9 and 2.10 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 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.* (2015) obtained good results on CEDAR, but poor results on GPDS; Soleimani *et al.* (2016) 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

Table 2.8 Comparison with the state-of-the-art in MCYT

Reference	**# Samples**	**Features**	**EER**
Gilperez *et al.* (2008)	5	Contours (chi squared distance)	10.18
Gilperez *et al.* (2008)	10	Contours (chi squared distance)	6.44
Wen *et al.* (2009)	5	RPF (HMM)	15.02
Vargas *et al.* (2011)	5	LBP (SVM)	11.9
Vargas *et al.* (2011)	10	LBP (SVM)	7.08
Ooi *et al.* (2016)	5	DRT + PCA (PNN)	13.86
Ooi *et al.* (2016)	10	DRT + PCA (PNN)	9.87
Soleimani *et al.* (2016)	5	HOG (DMML)	13.44
Soleimani *et al.* (2016)	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 2.9 Comparison with the state-of-the-art in CEDAR

Reference	**# Samples**	**Features**	**AER/EER**
Chen & Srihari (2006)	16	Graph Matching	7.9
Kumar *et al.* (2010)	1	morphology (SVM)	11.81
Kumar *et al.* (2012)	1	Surroundness (NN)	8.33
Bharathi & Shekar (2013)	12	Chain code (SVM)	7.84
Guerbai *et al.* (2015)	4	Curvelet transform (OC-SVM)	8.7
Guerbai *et al.* (2015)	8	Curvelet transform (OC-SVM)	7.83
Guerbai *et al.* (2015)	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)**

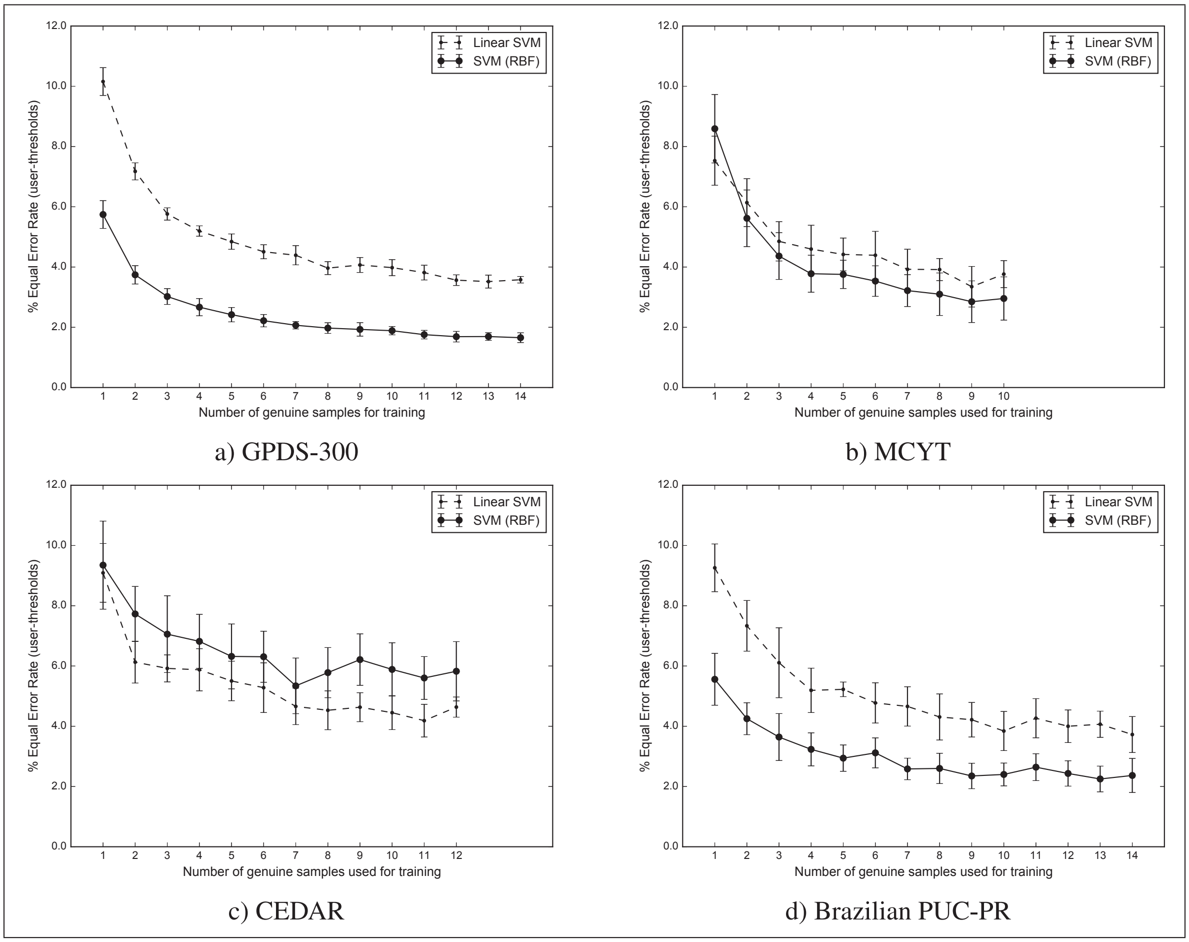
Table 2.10 Comparison with the state-of-the-art on the Brazilian PUC-PR dataset (errors in %)

Reference	**#samples per user**	**Features**	**FRR**	**FAR$_{\text{random}}$**	**FAR$_{\text{simple}}$**	**FAR$_{\text{skilled}}$**	**AER**	**AER$_{\text{genuine + skilled}}$**	**EER$_{\text{genuine + skilled}}$**
Bertolini *et al.* (2010)	15	Graphometric	10.16	3.16	2.8	6.48	5.65	8.32	-
Batista *et al.* (2012)	30	Pixel density	7.5	0.33	0.5	13.5	5.46	10.5	-
Rivard *et al.* (2013)	15	ESC + DPDF	11	0	0.19	11.15	5.59	11.08	-
Eskander *et al.* (2013)	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)

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$_{\text{skilled}}$.

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 (Vargas *et al.* (2007)).

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 (Impedovo & Pirlo (2008)), 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 % $\text{EER}_{\text{user thresholds}}$ using 10 samples per user. Using online data, the best results reported in the literature achieve 2.85 % EER (Rua & Castro (2012)) and 3.36 % EER (Fierrez *et al.* (2007)), 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 (Rua & Castro (2012)) and (Fierrez *et al.* (2007)) use score normalization, followed by a single global threshold, so the comparison of these papers to our work is not direct.



a) GPDS-300

b) MCYT

c) CEDAR

d) Brazilian PUC-PR

Figure 2.6 Average performance of the Writer-Dependent classifiers for each dataset, as we vary the number of genuine signatures (per user) available for training.

2.5.2.3 Varying the number of genuine samples available for training

Figure 2.6 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 (Eskander *et al.* (2013), Hafemann *et al.* (2016b)), 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.

2.6 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 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.

CHAPTER 3

FIXED-SIZED REPRESENTATION LEARNING FROM OFFLINE HANDWRITTEN SIGNATURES OF DIFFERENT SIZES

Luiz G. Hafemann[1], Robert Sabourin[1], Luiz S. Oliveira[2]

[1] Department of Automated Manufacturing Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Departement of Informatics, Federal University of Parana (UFPR),
Curitiba, PR, Brazil

Article Published in « International Journal on Document Analysis and Recognition (IJDAR) » 2018.

Abstract

Methods for learning feature representations for Offline Handwritten Signature Verification have been successfully proposed in recent literature, using Deep Convolutional Neural Networks to learn representations from signature pixels. Such methods reported large performance improvements compared to handcrafted feature extractors. However, they also introduced an important constraint: the inputs to the neural networks must have a fixed size, while signatures vary significantly in size between different users. In this paper we propose addressing this issue by learning a fixed-sized representation from variable-sized signatures by modifying the network architecture, using Spatial Pyramid Pooling. We also investigate the impact of the resolution of the images used for training, and the impact of adapting (fine-tuning) the representations to new operating conditions (different acquisition protocols, such as writing instruments and scan resolution). On the GPDS dataset, we achieve results comparable with the state-of-the-art, while removing the constraint of having a maximum size for the signatures to be processed. We also show that using higher resolutions (300 or 600dpi) can improve performance when skilled forgeries from a subset of users are available for feature learning, but lower resolutions (around 100dpi) can be used if only genuine signatures are used. Lastly, we show that fine-tuning can improve performance when the operating conditions change.

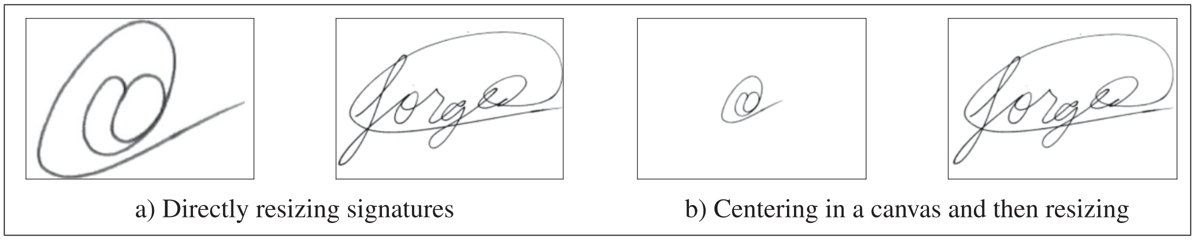
3.1 Introduction

The handwritten signature is a behavioral biometric trait that is extensively used to verify a person's identity in legal, financial and administrative areas. Automating the verification of handwritten signatures has been a subject of research since the decade of 1970 (Plamondon & Lorette (1989); Leclerc & Plamondon (1994); Impedovo & Pirlo (2008); Hafemann *et al.* (2017b)), considering two scenarios: online (dynamic) and offline (static). In the online case, signatures are captured using a special device, such as a pen tablet, that records the dynamic information of the writing process (e.g. position of the pen over time). For offline signature verification, we consider signatures written on paper, that are subsequently scanned to be used as input.

Most of the research effort in offline signature verification has been devoted to finding good feature representations for signatures, by proposing new feature descriptors for the problem (Hafemann *et al.* (2017b)). Recent work, however, showed that learning features from data (signature images) can improve system performance to a large extent (Hafemann *et al.* (2016b, 2017a); Rantzsch *et al.* (2016); Zhang *et al.* (2016)). These work rely on training Deep Convolutional Neural Networks (CNNs) to learn a hierarchy of representations directly from signature pixels.

Although these methods present good performance, they also introduce some issues. Signatures from different users vary significantly in size, while a feature descriptor should provide a fixed-sized representation for classification. This is not a problem in many feature descriptions used for signature verification, that by design are able to accommodate signatures of different sizes. Neural networks, on the other hand, in general require fixed-sized inputs, and thus these methods require pre-processing the signatures such that they all have the same size. Most commonly, signatures are either a) resized to a common size or b) first centered in a blank image of a "maximum signature size", and then resized. Figure 3.1 illustrates the problems with these approaches. In alternative (a), the width of the strokes become very different depending on the size of the original image, while in alternative (b) the width of strokes is not affected, but instead we may lose detail on small signatures, that would otherwise be preserved in the first alternative. Empirically, alternative (b) presented much better results (Hafemann *et al.* (2016b)), but it also creates the problem that now a “maximum size" is defined, and if a new signature is larger than this size, it would need to be reduced (causing similar problems to (a) regarding the width of the strokes).

Another problem in learning the representations from signature images is the selection of the resolution of the input images. The methods proposed in the literature use small images (e.g. 96 × 192 in (Rantzsch *et al.* (2016)), 170 × 242 in (Hafemann *et al.* (2016a))). For the signatures used in these papers, this is equivalently of using a resolution around 100 dpi. However, as illustrated in figure 3.2, the distinction of genuine signatures and skilled forgeries often rely on the line quality of the strokes (in particular for slowly-traced forgeries, as noted in (Hafemann *et al.* (2016a))). This suggests that using higher resolutions may improve performance on this task.


a) Directly resizing signatures

b) Centering in a canvas and then resizing

Figure 3.1 Two approaches for normalizing the signatures to a common size. The signature on the left is small (176 × 229 pixels) while the signature on the right is large (484 × 819 pixels). (a) directly resizing the signatures to the input size of the network (170 × 242); (b) centering the signatures in a canvas of a “maximum size" (600 × 850) and then resizing to 170 × 242 pixels.

In this paper, we propose learning a fixed-sized representation for signatures of variable size, by adapting the architecture of the neural network, using Spatial Pyramid Pooling (SPP) (He *et al.* (2014), He *et al.* (2015)). Our contributions are as follows: we define and evaluate different training protocols for networks with SPP applied to offline handwritten signatures. After training, signatures of any size can be fed to the network in order to obtain a fixed-sized representation. We also evaluate the impact of the image resolution on the classification

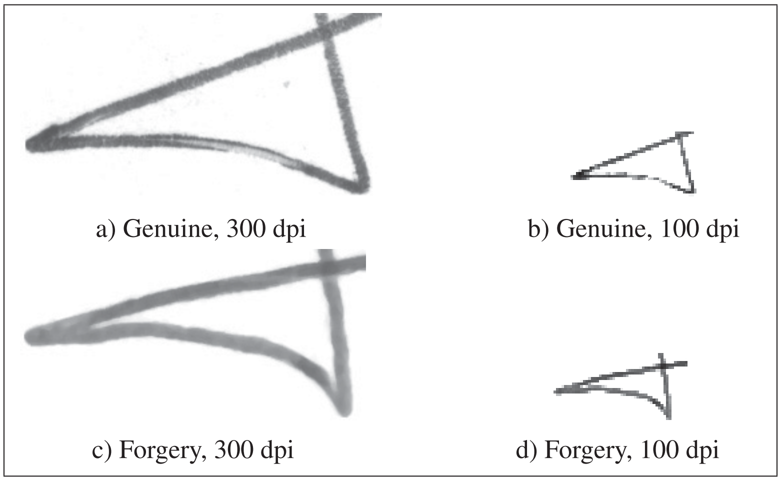


Figure 3.2 Detail of a genuine signature and a skilled forgery for user 244 in the GPDS dataset. At 300 dpi, we can notice the limp strokes of the skilled forgery, most likely due to slow hand movements while attempting to reproduce the overall shape of the genuine signature. At 100 dpi, information about line quality is mostly lost.

accuracy, and the generalization of features learned in one dataset to other operating conditions (e.g. different acquisition protocols, signatures from people of different locations), by using transfer learning to other datasets.

For feature learning, we used the problem formulation presented in (Hafemann *et al.* (2017a)), where Writer-Independent features are learned using a subset of users, and subsequently used to train Writer-Dependent classifiers for another set of users. We also use the architecture defined in this work as baseline (SigNet). We adapt this architecture to learn fixed-sized representations (proposing different training protocols) and modifying the architectures to handle images of higher resolution. We conducted experiments on four widely used signature verification datasets: GPDS, MCYT, CEDAR and the Brazilian PUC-PR dataset; and two synthetic datasets (Bengali and Devanagari scripts). Using the proposed architecture, we obtain a similar performance compared to the state-of-the-art, while removing the constraint of having a fixed maximum signature size. We also note that using higher resolutions (300dpi) greatly improves performance when skilled forgeries (from a subset of users) is available for training. On the other hand, if only genuine signatures are used for feature learning, higher resolutions did not improve performance. We verify that the learned features generalize to different operating conditions (by testing them on other datasets), and that fine-tuning the representation for the different conditions further improves performance. We observed that the features learned on GPDS generalize better to other western signature datasets (MCYT, CEDAR and Brazilian PUC-PR) than to other types of scripts (Bengali and Devanagari), and that fine-tuning also largely addresses this problem.

3.2 Related Work

The problem of Offline Signature Verification is either formulated as Writer-Dependent, with one classification task defined for each user enrolled to the system, or as a Writer-Independent problem, where we consider a single problem, of comparing a questioned signature to a reference signature. In the literature, Writer-Dependent classification is most commonly used: for each user, a set of reference (genuine) signatures are used as positive samples, and a set of genuine signatures from other users (in this context called “Random forgeries") are used as negative samples, and a binary classifier is trained. Alternatively, some authors propose using one-class classifiers for the Writer-Dependent formulation, using only genuine signatures from the user as positive samples (e.g. Guerbai *et al.* (2015)). Writer-Independent classification, on the other hand, is often used by training a binary classifier on a dissimilarity space, where the inputs are the absolute difference of two feature vectors: $\mathbf{x} = |\mathbf{f}_1 - \mathbf{f}_2|$, where $\mathbf{f}_1$ and $\mathbf{f}_2$ are feature vectors extracted from two signatures, and we consider a binary label: $y = 1$ if both signatures are from the same user, and $y = -1$ otherwise (Bertolini *et al.* (2010); Rivard *et al.* (2013); Eskander *et al.* (2013)).

After training the classifiers, we verify the performance of the system in distinguishing genuine signatures from forgeries. We adopt the following definitions of forgery, which are the most common in the Pattern Recognition community: “Random Forgeries" are forgeries made without any knowledge of the user’s genuine signature, where the forger uses his own signature instead. In the case of “Simple forgeries", the forger has access to the person’s name. In this case, the forgery may present more similarities to the genuine signature, in particular for users that sign with their full name, or part of it. Lastly, for “Skilled Forgeries", the forger has access to the user's signature, and often practices imitating it. This result in forgeries that have higher resemblance to the genuine signature, and therefore are harder to detect. While discriminating Random and Simple forgeries are relatively simpler tasks (as reflected in lower error rates in the literature), discriminating genuine signatures and skilled forgeries remains a challenging task.

A critical aspect of designing signature verification systems is how to extract discriminant features from the signatures. A large part of the research efforts on this field addresses this question, by proposing new feature descriptors for the problem. These features range from simple descriptors such as the size of the signature and inclination (Nagel & Rosenfeld (1977)), graphometric features (Justino *et al.* (2000), Oliveira *et al.* (2005)), texture-based (Vargas *et al.* (2011); Yılmaz & Yanıkoğlu (2016)), interest point-based (Pal *et al.* (2012)), among others. Recent advancements in this field include using multiple classifiers trained with different representations (Yılmaz & Yanıkoğlu (2016)), using interval symbolic representations (Alaei *et al.* (2017)) and augmenting datasets by duplicating existing signatures or creating synthetic ones (Ferrer *et al.* (2015); Diaz *et al.* (2017); Ferrer *et al.* (2017)). More recently, methods for learning features from signature images have been proposed (Hafemann *et al.* (2016b, 2017a); Rantzsch *et al.* (2016); Zhang *et al.* (2016)). Although these methods demonstrated improved performance, they introduced some issues, notably by requiring that all signature images have the same size, which is the problem addressed in this paper. We note that this problem is not present in many handcrafted feature descriptions used for signature verification: for instance Local Binary Patterns (LBP) (Ojala *et al.* (2002)) and Histogram of Oriented Gradients (HOG) (Dalal & Triggs (2005)) use histograms over the entire image, therefore resulting in feature vectors of the same size regardless of the input size; Extended Shadow Code (ESC) (Sabourin *et al.* (1993)) divides the image in the same number of windows (adapting the size the windows), therefore also working with signatures of variable sizes.

The problem of requiring inputs of a fixed size for neural networks also affects other applications, such as object recognition. This problem is often handled by simply resizing and cropping images. While these are common operations for object recognition, we argue that they are less interesting for signature verification. In object recognition, the classification task considers objects at different scales. Therefore, resizing an image to fit a particular size is a reasonable action to take, since it is aligned with the invariance to scale that we expect from the classifiers (as long as the change in scale does not distort the image, such as scaling height and width by different factors). On the other hand, for signature verification we have control of how the signatures are acquired: all signatures are usually scanned at the same resolution, usually 300 or 600 dpi. Therefore, changes in scale, introduced by resizing the image, alter the signal is ways that would not otherwise be present. In this case, a better solution would not require resizing the signature images by different factors depending on their original size.

In the context of object recognition, He at al. proposed a solution for working with inputs of variable size, by using Spatial Pyramid Pooling (He *et al.* (2015)). However, the training procedure still requires fixed-sized images. He *et al.* (2015) proposed using two image sizes, resizing each image to the these sizes (i.e. duplicating the dataset in two different scales). Learning is then conducted by alternating between the two sets in each training epoch. This is sub-optimal for signature images, since we would like to avoid resizing the images entirely. In this work we propose and test other training protocols for training networks with SPP on signature data.

3.3 Proposed Method

In this work we consider the two-stage approach described in (Hafemann *et al.* (2016b)) and (Hafemann *et al.* (2017a)), where we train Writer-Dependent classifiers on a set of users, using a feature representation learned on another set of users. We note, however, that the methods described in this paper can be used for other feature learning strategies, such as the ones used in (Rantzsch *et al.* (2016); Zhang *et al.* (2016)).

We consider two disjoint sets of users: a development set $\mathscr{D}$, where we learn feature representations, and an exploitation set $\mathscr{E}$ that consists of the users “enrolled to the system", for whom we train Writer-Dependent classifiers. The first phase consists in learning a function $\phi(X)$, using the data from $\mathscr{D}$, that takes a signature X as input, and returns a fixed-sized feature vector. In the second phase, we use this learned function to “extract features" for the signatures in $\mathscr{E}$, and train a binary classifier for each user. While we could use all users for learning the representations, this separation in two sets allows us to estimate the generalization performance of using this learned representation for new users. This is important since the set of users in a system is not fixed - new users may enroll at any time, and in this formulation we simply use the learned function $\phi(X)$ to obtain a representation for the signatures of this new user, and train a binary classifier.

In order to handle signatures of different sizes, we change both the feature learning process, as well as the process to obtain representations for new signatures using the learned network.

3.3.1 Network architecture and objective function

The definition of a Convolutional Neural Network architecture usually specifies the input size of the images for training and testing. However, as noted in (He *et al.* (2015)), this constraint is not caused by the usage of convolution and pooling layers, but rather by the usage of fully-connected layers at the end of network architectures. The reason is that the convolution and pooling operations are well defined for inputs of variable sizes, simply resulting in an output of a larger size. This presents a problem between the last pooling layer and the first fully-connected layer of the network (layer FC1 in figure 3.3): the last pooling layer is “flattened" to a vector of dimensionality K (e.g. a pooling output of size $32 \times 3 \times 2$ becomes a vector of $K = 192$ elements), and the fully-connected layer uses a weight matrix of size $K \times M$, where M is the output size of the fully-connected layer. If we use the network to process an input $\hat{X}$ of a different size, the output of the last pooling layer will have a different size. Flattening the representation results in a vector of dimensionality $\hat{K} \neq K$, and therefore the vector-matrix product in the fully-connected layer will not be defined.

The central idea of Spatial Pyramid Pooling (SPP) (He *et al.* (2015)) is to obtain a fixed-size representation for variable-sized input images. This is done by adapting the size of the pooling

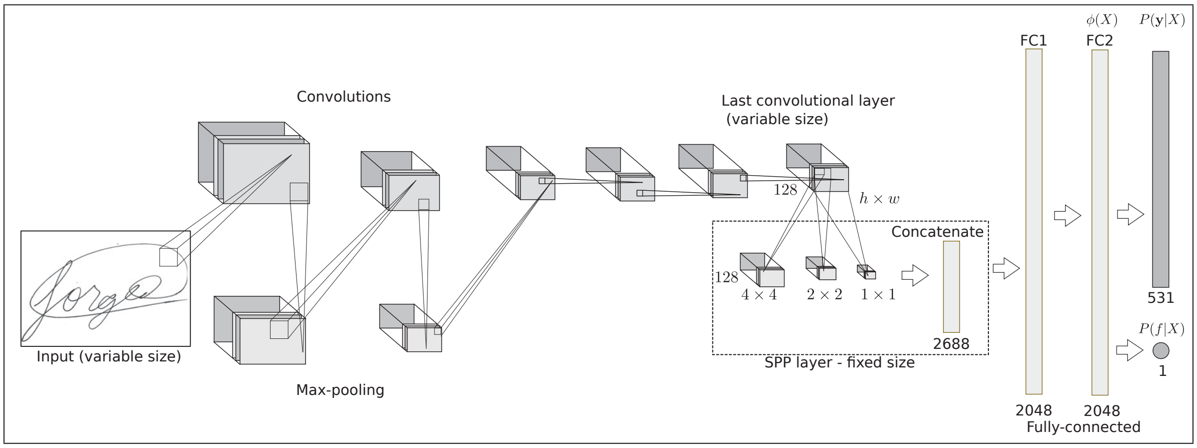


Figure 3.3 A CNN architecture with SPP used in this work. The input signature (of variable size) is transformed in a sequence of convolution and max-pooling operations. The last convolutional layer results in 128 maps of size $h \times w$ (the actual size varies according to the signature size). The Spatial Pyramid Pooling layer (SPP) is then used to obtain a fixed-sized representation, by adapting the size of pooling regions, to obtain pooled results in three sizes: 4×4, 2×2 and 1×1. These are concatenated in a single vector of size $21 \times 128 = 2688$ units, which is then used as input to the fully-connected layers. During training, the network outputs $P(\mathbf{y}|X)$ (and, if forgeries are used during training, also $P(f|X)$), and the network is trained to minimize the cross-entropy with respect to the training dataset. For obtaining representations for new signatures, we perform forward propagation until the last layer before softmax, obtaining $\phi(X)$, a vector of 2048 dimensions, regardless of the signature size.

region (and strides) for each image size, such that the output of the last pooling operation has a fixed size, and therefore can be used as input to fully-connected layers. In SPP, a set of fixed-sized outputs is chosen, and the result of them is concatenated. This is illustrated in the “SPP Layer" box in figure 3.3: we consider pooling with output sizes 1×1, 2×2 and 4×4, which has a total of $1+4+16 = 21$ outputs for each channel. Each image would therefore output a fixed representation of size $21 \times C$, where C is the number of feature maps/channels in the last convolutional layer.

Figure 3.3 illustrates a CNN architecture used in this work. The network contains a series of convolutions and max-pooling operations, with a Spatial Pyramid Pooling layer between the last convolutional layer and the first fully-connected layer. This layer outputs a fixed-sized output regardless of the size of the input signature.

We consider two application scenarios, as in (Hafemann *et al.* (2017a)): one in which we have only genuine signatures available for training, and one in which we also have access to skilled forgeries for a subset of users. In the first scenario, we consider a training objective of distinguishing between different users in the development set: the network outputs $P(\mathbf{y}|X)$: the probability of the signature belonging to one of the users in $\mathscr{D}$. Therefore, the network learns to identify the users that produced the signatures in $\mathscr{D}$. In the second scenario, we would like to leverage the information of forgeries in the feature learning process, and we use the multi-task approach defined in (Hafemann *et al.* (2017a)). In this formulation, the network also predicts whether or not the signature is a forgery: $P(f|X)$. We simultaneously train the network to optimize both objectives (distinguish between different users, and between genuine signatures and skilled forgeries), by using the loss function defined in equation 3.1:

$$\begin{aligned} L &= (1-f_i)(1-\lambda)L_c + \lambda L_f \\ &= -(1-f_i)(1-\lambda)\sum_j y_{ij} \log P(y_j|X_i) + \\ &\lambda\big(-f_i \log(P(f|X_i)) - (1-f_i)\log(1-P(f|X_i))\big) \end{aligned} \tag{3.1}$$

Where λ is a hyperparameter that trades-off between the two objectives, X_i is the signature, $\mathbf{y}_i$ is the actual user of signature ($y_{ij} = 1$ if the signature i belongs to user j), and f_i indicates whether or not the signature i is a forgery. L_c and L_f indicate the loss functions for user classification and forgery classification, respectively, which are expanded in the second and third lines. We refer the reader to (Hafemann *et al.* (2017a)) for more details on this formulation.

Table 3.1 lists the CNN architectures used in this paper. We consider a total of six architectures, considering three different resolutions (around 100 dpi, 300 dpi and 600 dpi), and with or without Spatial Pyramid Pooling. Each line in the table represents a layer of the CNN. For convolutional layers, we specify the size and number of feature maps (filters), the stride and the padding. For instance, conv11-32-s4-p5 refers to a convolutional layer with 32 filters of size

Table 3.1 CNN architectures used in this paper

SigNet	**SigNet-SPP**	**SigNet-300dpi**	**SigNet-SPP-300dpi**	**SigNet-600dpi**	**SigNet-SPP-600dpi**
conv11-96-s4-p0	conv11-96-s4-p0	conv11-32-s3-p5	conv11-32-s3-p5	conv11-32-s4-p5	conv11-32-s4-p5
pool3-s2-p0	pool3-s2-p0	pool3-s2-p0	pool3-s2-p0	pool3-s3-p0	pool3-s3-p0
conv5-256-p2	conv5-256-p2	conv5-64-p2	conv5-64-p2	conv5-64-p2	conv5-64-p2
pool3-s2-p0	pool3-s2-p0	pool3-s3-p0	pool3-s3-p0	pool3-s2-p0	pool3-s2-p0
conv3-384-p1	conv3-384-p1	conv3-128-p1	conv3-128-p1	conv3-128-p1	conv3-128-p1
conv3-384-p1	conv3-384-p1	conv3-128-p1	conv3-128-p1	conv3-128-p1	conv3-128-p1
		pool3-s2-p0	pool3-s2-p0	pool2-s2-p0	pool2-s2-p0
conv3-256-p1	**conv3-180-p1**	conv3-128-p1	conv3-128-p1	conv3-128-p1	conv3-128-p1
pool3-s2-p0	**spp-4-2-1**	pool3-s3-p0	**spp-4-2-1**	pool4-s4-p0	**spp-4-2-1**
FC1-2048	FC1-2048	FC1-2048	FC1-2048	FC1-2048	FC1-2048
FC2-2048	FC2-2048	FC2-2048	FC2-2048	FC2-2048	FC2-2048
FC-M + softmax ; FC-1 + sigmoid					

11×11, with stride $s = 4$ and padding $p = 5$. For pooling operations, we inform the pool size, the stride and padding. When not specified, we use stride $s = 1$. After each learnable layer (with the exception of the output layers) use a Batch Normalization layer (Ioffe & Szegedy (2015)). The SigNet architecture was defined in previous work (Hafemann *et al.* (2017a)), while the other architectures are adapted versions to handle larger images. For higher resolutions, we notice that images are very large (e.g. 780x1095 pixels for a 600 dpi signature). To handle these larger images, we used a smaller number of feature maps, and a more rapid reduction in size across the layers, by using a more aggressive pooling. For each of the three resolutions, we consider both a version with SPP (that accepts inputs of any size), and without SPP (that accepts inputs of a fixed size). The two versions have the same overall structure, but diverge on the last pooling layer. The network SigNet-SPP has another difference (lower number of convolutional maps in the last convolutional layer) to keep the number of parameters between the SPP and non-SPP version similar. In the table, the differences between the non-SPP and SPP versions are highlighted in bold. In all architectures, the last layer outputs M neurons, which estimate $P(\mathbf{y}|X)$, the probability of a signature X belonging to a particular user in $\mathscr{D}$. For the experiments using forgeries during feature learning, the network also outputs a single neuron that predicts $P(f|X)$, the probability that the signature is a forgery. We report experiments with both scenarios (with and without forgeries for feature learning). In the cases were forgeries are used, we append a suffix **-F** to the architecture name. For example, *SigNet-300dpi-F* refers to using the architecture SigNet-300dpi using both genuine signatures and skilled forgeries for training, while *SigNet-300dpi* refers to using the same architecture, but trained with only genuine signatures.

The Spatial Pyramid Pooling layer was implemented as in (He *et al.* (2015)): we use pooling regions of sizes 4×4, 2×2 and 1×1, resulting in a total of 21 outputs for each feature map. The pooling region and strides are dynamically determined for each input size. Let h and w be the output size of the last convolutional layer. For the pyramid level of size $n \times n$, the pooling region of the unit (j,i) is defined as rows between: $\left\lfloor \frac{j-1}{n} h \right\rfloor$ and $\left\lceil \frac{j}{n} h \right\rceil$ and columns between $\left\lfloor \frac{i-1}{n} w \right\rfloor$ and $\left\lceil \frac{i}{n} w \right\rceil$, where $\lfloor . \rfloor$ and $\lceil . \rceil$ denote the floor and ceiling operations. Similarly to max-pooling, we take the max of this pooling region, that is, the output of unit (j,i) is the maximum value of the pooling region defined above. The implementation of this layer has been made publicly available in the Lasagne library [1].

3.3.2 Training protocol

The neural networks are initialized with random weights following (Glorot & Bengio (2010)), and training is performed with stochastic gradient descent to minimize the loss function defined in section 3.3.1. We use mini-batches of data (which is required in order to use Batch Normalization, and also speeds up training), and we consider different protocols for generating the mini-batches, as described below.

The networks without SPP require a fixed input size for all images. In this case, we pre-process all signatures by centering them in a canvas of a “maximum size", and then resizing to the desired input size for the network.

When using Spatial Pyramid Pooling, we can process signatures of any size, but during training we need to design a protocol that provides batches of signatures having the exact same size. We consider two alternatives:

[1] https://github.com/Lasagne/Lasagne/

1. **Fixed size**: Using a single “maximum signature size" (as in the training for networks without SPP);
2. **Multiple sizes**: Defining multiple canvas sizes, and centering each signature on the smallest canvas that fit the signature.

In the first case, all the images on the training set have the same size, and we simply process the images in mini-batches in a random order.

For the second alternative, we define different image sizes based on statistics of the development set (the set of signatures used to train the CNN). Consider the following definitions:

- μ_h, σ_h, $\max_h$: height of the signatures in the development set (mean, standard deviation and maximum, respectively);
- μ_w, σ_w, $\max_w$: width of the signatures in the development set (mean, standard deviation and maximum, respectively).

We divide the dataset into 5 different parts, as follows:

1. Images larger than 3 standard deviations are considered “outliers". In particular, images taller than $\tau_h = \mu_h + 3\sigma_h$ or wider than $\tau_w = \mu_w + 3\sigma_w$ are all assigned to the largest canvas, of size ($\max_h \times \max_w$);
2. The remaining signatures are split in four groups, by using the medians of the height and width. Given the medians $\tilde{H}$ and $\tilde{W}$ for the height and width, respectively, we consider the following canvas sizes: ($\tilde{H} \times \tilde{W}$), ($\tilde{H} \times \tau_w$), ($\tau_h \times \tilde{W}$), ($\tau_h \times \tau_w$).

Each signature is centered (not resized) in the smallest canvas size that fits the signature. Therefore, this creates a total of 5 datasets, one for each canvas size.

During training, we create an iterator for each of the 5 datasets: each iterator returns batches of signatures of the same size (within the batch). We then train the model by taking batches of different image sizes, alternating the sizes after each mini-batch (contrary to (He *et al.* (2015)) that alternated after an entire epoch). This procedure is detailed in Algorithm 13. In this algorithm, the *train* function is a single step of Stochastic Gradient Descent, with Nesterov Momentum.

Algorithm 3.1 Training algorithm for “multiple sizes", for one epoch.

```
Input: S: set of image sizes; iterators: list of data iterators, for each image size
active $\leftarrow$ S ;
while active $\neq \emptyset$ do
    for $s \in$ active do
        if iterator[s].has_next_batch() then
            mini_batch $\leftarrow$ iterator[s].next_batch() ;
            train(mini_batch) ;
        end
        else
            active $\leftarrow$ active $\setminus$ s
        end
    end
end
```

We trained the networks for 60 epochs, using mini-batches of size 32, L2 penalty with weight decay set to 10^{-4} and momentum factor of 0.9. Training started with a learning rate of 10^{-3}, which was decreased twice (at epochs 20 and 40) by dividing it by 10 each time.

3.3.2.1 Data augmentation

In previous work (Hafemann *et al.* (2016a)) we performed data augmentation by performing random crops of the input images. We adopt the same protocol for the “Fixed size" training. However, in the “Multiple sizes" protocol defined above, where we use smaller canvas sizes, cropping the images could result in cropping part of the signature, not only the background. Instead, we use the opposite strategy, of enlarging the signature images, by padding the signatures with the background color. We use this strategy to avoid losing part of the signal due to cropping. For example, consider an image of size 300x300, and a padding of size 20x20: we pad the image so that it has size 320x320, positioning the original image to randomly start between 0 and 20 pixels in height and width (i.e. not necessarily in the center of this new image). For even greater variability, we consider a maximum value of padding, and in each mini-batch we randomly select the padding between 0 and this maximum value.

3.3.3 Fine-tuning representations

When considering the generalization of the learned features to new operating conditions (e.g. new acquisition protocol), it is possible to fine-tune the representations to the new conditions. In order to evaluate the impact of fine-tuning the representations, we consider a network trained in one dataset as a starting point, and subsequently train it for users of another dataset.

Similarly to previous work on transferring representations (Oquab *et al.* (2014); Chatfield *et al.* (2014)), we perform the following steps for fine-tuning representation to a new dataset:

- Duplicate the network that was trained in the first dataset;
- Remove the last layer (that correspond to $P(\mathbf{y}|X)$ for the users in the first dataset);
- Add a new softmax layer, with M_2 units, corresponding to $P(\mathbf{y}|X)$, the probability of a signature image belonging to one of the M_2 users of the second dataset;
- Train the network on the second dataset with a reduced learning rate (5×10^{-4}).

The training procedure during finetuning is similar to the training algorithm used for learning the features in the first dataset. The exception is for the SPP models trained with a “Fixed size". In this case, we consider two distinct sizes: the original maximum signature size from the first dataset, and the maximum signature size from the target dataset.

Since different datasets have different acquisition protocols (e.g. type of writing instrument, instructions for the forgers, and the resolution of scanned images), we expect that fine-tuning the representations to a set of users from the same domain should improve performance.

3.3.4 Training WD classifiers

After we learn the CNN in one set of users, we use it to obtain representations for signatures of users in the exploitation set $\mathscr{E}$, and train Writer-Dependent classifiers. The procedure to obtain the representations vary slightly depending on the training method:

- Networks without SPP: The signatures from $\mathscr{E}$ are centered in a canvas of maximum size ($H_{\max} \times W_{\max}$). During transfer learning, we consider the maximum size of the target dataset, and resize all images to the size of the original dataset;
- SPP trained with “fixed size": The signatures from $\mathscr{E}$ are centered in a canvas of size ($H_{\max} \times W_{\max}$). During transfer learning, signatures that are larger than this canvas are processed in their original size;
- SPP trained with “multiple sizes": The signatures are processed in their original size.

The differences among the three training alternatives are summarized in table 3.2.

For each user in the set $\mathscr{E}$, we build a dataset consisted of r genuine signatures from the user as positive samples, and genuine signatures from other users as negative samples. We then train a binary classifier for the user: a linear SVM or an SVM with the RBF kernel. We usually have many more negative than positive samples for training, since we only have a few genuine signatures for the user, while we can use samples from many users as negative samples. For this reason, we correct this skew by giving more weight to the positive samples, as described in (Hafemann *et al.* (2017a)). After the classifiers are trained, we measure their capability of classifying genuine signatures are different types of forgery.

3.4 Experimental Protocol

We conducted experiments on four offline handwritten signature datasets: GPDS-960 (Vargas *et al.* (2007)), MCYT-75 (Ortega-Garcia *et al.* (2003)), CEDAR (Kalera *et al.* (2004)) and

Table 3.2 Summary of differences between the training/testing protocols

	Without SPP	**SPP (training with fixed size)**	**SPP (training with multiple sizes)**
Training images	Centered in a fixed size	Centered in a fixed size	Consider 5 different sizes
CNN architecture	Use pooling with fixed pooling size	Use SPP (variable pooling size, fixed output)	Use SPP (variable pooling size, fixed output)
Generalization (extracting features)	Center images in a fixed size. Larger images are resized	Center images in a fixed size. Larger images are processed in their original size	All images are processed in their original size
Finetuning	Center images in the maximum size of the target dataset. All images are resized to the maximum size of the source dataset	Center images in two canvases: maximum size of the target dataset, and maximum size of the source dataset	Consider 5 different sizes (defined in the target dataset)

Brazilian PUC-PR (Freitas *et al.* (2000)); and two synthetic datasets, for Bengali and Devanagari scripts (Ferrer *et al.* (2017)). We used a subset of the GPDS-960 dataset for learning feature representations, using the different architectures and training methods described in this article. We then evaluate the performance of Writer-Dependent classifiers trained with these feature representations, on a disjoint subset of GPDS, as well as the other datasets.

In order to allow comparison with previous work, we used the development set $\mathscr{D}$ as the last 531 users of GPDS (users 350-881) for training the CNNs. For the training protocol using "multiple sizes", we followed the procedure detailed in section 3.3.2 to process the development dataset into 5 different canvas sizes. For instance, at 600 dpi we used canvases of size 338×684, 338×1183, 619×684, 619×1183 and 778×1212. The networks were then trained as defined in section 3.3.2.

The images were pre-processed to remove noise, by applying Otsu's algorithm to find the threshold between background and foreground. The background pixels were set to white, leaving the signature pixels in grayscale. The images were then inverted by subtracting them from the maximum pixel intensity: $I_P(x,y) = 255 - I(x,y)$. In the resulting images the background is therefore zero-valued. The OTSU algorithm was not applied to the two synthetic datasets, since they do not contain any noise.

In the literature, slightly different protocols are used for each dataset, in particular regarding how many reference signatures are used for training, and which signatures are used as negative samples. We use the following protocols: In the GPDS dataset, we trained Writer-Dependent classifiers for the first 300 users (to compare to results using the GPDS-300 dataset), using $r = 12$ reference signatures as positive samples. We used 12 signatures from each user in the development $\mathscr{D}$ as negative samples ($12 \times 531 = 6372$ signatures). This protocol is similar to the Brazilian dataset, where we have a separate development set $\mathscr{D}$. We train classifiers for the first 60 users using $r = 15$, and 15 signatures from each of the remaining 108 users as negative samples ($15 \times 108 = 1620$ signatures). In the MCYT and Cedar datasets, we used $r = 10$ and $r = 12$, respectively, and the same number of signatures from each other user in the exploitation set $\mathscr{E}$ as negative samples. In all cases, we trained a binary SVM, with an RBF kernel. We used the same hyperparameters as previous research (Hafemann *et al.* (2016a)): $C = 1$ and $\gamma = 2^{-11}$, that were selected using a subset of the GPDS validation set. In this paper we did not explore optimizing these hyperparameters for each dataset (or even each user), but rather keep the same set of parameters for comparison with previous work.

For the experiments generalizing to different conditions (datasets), we considered two scenarios: using the CNN trained on GPDS to extract features without any changes, and fine-tuning the representation on these datasets. In these experiments, we used the network trained on GPDS images of the same resolution of the datasets (300 dpi for Cedar and Brazilian, 600 dpi for other datasets).

In order to assess the generalization performance of the fine-tuned representations, we conducted cross-validation experiments as follows:

1. The dataset in randomly split in two sets of users (50%/50%). Following the same terminology as before, we can consider them to be a development set $\mathscr{D}$ and exploitation set $\mathscr{E}$;
2. We fine-tune the CNN (originally trained on GPDS) for the development set $\mathscr{D}$;
3. We use the fine-tuned CNN to extract features for the exploitation set $\mathscr{E}$ and train WD classifiers.

This protocol allows for an unbiased estimation of the performance on new users, whose signatures match the same operating characteristics of the dataset. We performed cross-validation running the steps above 10 times, each time randomly splitting the dataset in half, fine-tuning the CNN and training WD classifiers. For each fine-tuned CNN, we performed 10 runs on the WD classifier training with different signatures used for training/testing. Therefore, we fine-tuned a total of 10 CNNs for each dataset, and trained a total of 100 WD classifiers for each user in each dataset, and for each architecture. We then report the mean and standard deviation across these 100 runs.

We evaluate the generalization of the learned representations to different scripts by training WD classifiers on synthetic signatures for two indian scripts: Bengali and Devanagari (Ferrer *et al.* (2017)). For these datasets, since skilled forgeries are not available (the generation procedure is only defined for genuine signatures in (Ferrer *et al.* (2017))) we evaluate the performance of the system on random forgeries. To allow for comparison with previous work, we train the WD classifiers with $r = 5$ genuine signatures as positive samples. We also evaluated the errors on random forgeries on the other four datasets, which allows us to verify the generalization performance to other western scripts (on MCYT, CEDAR and Brazilian PUC-PR) and for other types of scripts (Bengali and Devanagari).

We evaluate the performance primarily using the Equal Error Rate (EER): which is the error when False Acceptance (misclassifying a forgery as being genuine) is equal to False Rejection (misclassifying a genuine as being a forgery). 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. For most experiments, we report the Equal Error Rates using only skilled forgeries. In the experiment where we compare the generalization to different scripts, we report the Equal Error Rates calculated with random forgeries.

For the Brazilian PUC-PR dataset, we used the same metrics as 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 $\text{AER} = (\text{FRR} + \text{FAR}_{\text{random}} + \text{FAR}_{\text{simple}} + \text{FAR}_{\text{skilled}})/4$. We also reported the average error rate considering only genuine signatures and skilled forgeries: $\text{AER}_{\text{genuine + skilled}} = (\text{FRR} + \text{FAR}_{\text{skilled}})/2$.

For the comparison between different training types, and to measure the impact of finetuning, we use t-tests to compare the classifiers (using the $\text{EER}_{\text{user thresholds}}$ metric). We considered results significantly different for $p < 0.01$.

3.5 Results

We first present our analysis on using different image resolutions, followed by the analysis of the methods trained with SPP for handling signatures of variable size, and a comparison with the state-of-the-art.

The results with varying the image resolution are summarized in figure 3.4. This figure shows the classification performance (EER) of Writer-Dependent classifiers trained on the GPDS-300 dataset, as we increase the resolution of the images. For these experiments, we consider the models trained without SPP, and consider two training scenarios: when we only use genuine signatures, and when skilled forgeries from a subset of users is used for feature learning (note that for training the WD classifiers, no skilled forgeries are used). The objective of this experiment is to verify the hypothesis that higher image resolutions are required to discriminate skilled forgeries. We notice an interesting trend in this figure: when using both genuine signatures and skilled forgeries, increasing the resolution greatly improves performance, reducing errors from 2.10% using 100 dpi to 0.4% using 300 dpi. On the other hand, increasing resolution did not improve performance when only genuine signatures are used for feature learning. We argued in the introduction (in figure 3.2) that low resolutions lose information about the line quality. These results suggest that, although fine details are present in higher resolution images, they are not taken into account when only genuine signatures are used for training the CNN. In other words, since the network does not have access to any skilled forgery, it does not learn features that discriminate line quality. Therefore, when only genuine signatures are available for training, low resolutions (100 dpi) are sufficient, but if forgeries from a subset of users are available, higher resolutions (e.g. 300 dpi) greatly improve performance.

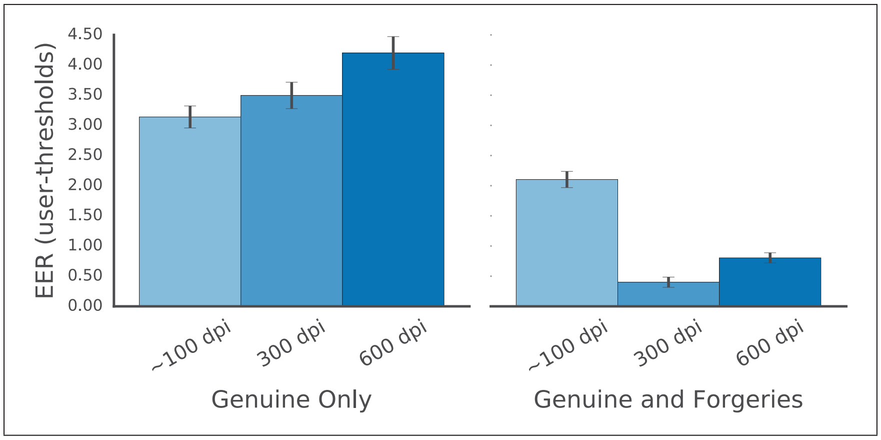


Figure 3.4 Impact of the image resolution on system performance: EER of Writer-Dependent classifiers trained on GPDS-300, with representations learned in $\mathscr{D}$ at different resolutions. Left: Using only genuine signatures for feature learning; Right: Using genuine signatures and skilled forgeries for feature learning. Error bars indicate one standard deviation of the mean error (across 10 replications)

We now consider the experiments using SPP for learning a fixed-sized representation for signatures of different sizes. Table 3.3 compares the performance of the WD classifiers on the GPDS dataset, as we change the training method. We consider both the baseline (network without SPP), and the two proposed training protocols for using SPP: with a single fixed canvas for training (denoted “Fixed" in the table), and using the 5 different canvases, defined in the development set (denoted “Multi" in the table). The results that are significantly better than the baseline (at $p < 0.01$) are denoted with a bullet point (●). We notice that the performance between the baseline and SPP Fixed is very similar, while the method using multiple canvases

Table 3.3 Performance of WD classifiers on GPDS-300, using 12 reference signatures (Errors and standard deviations in %)

Feature	**Training Algorithm**	$\textbf{EER}_{\textbf{global threshold}}$	$\textbf{EER}_{\textbf{user thresholds}}$
SigNet-300dpi		5.72 (±0.21)	3.5 (±0.22)
SigNet-SPP-300dpi	Fixed	5.63 (±0.22)	3.15 (±0.14) ●
SigNet-SPP-300dpi	Multi	7.75 (±0.28)	4.86 (±0.24)
SigNet-300dpi-F		1.78 (±0.12)	0.4 (±0.08)
SigNet-SPP-300dpi-F	Fixed	1.69 (±0.1)	0.41 (±0.05)
SigNet-SPP-300dpi-F	Multi	2.52 (±0.09)	0.8 (±0.07)
SigNet-600dpi		7.11 (±0.17)	4.2 (±0.27)
SigNet-SPP-600dpi	Fixed	7.06 (±0.13)	4.02 (±0.18)
SigNet-SPP-600dpi	Multi	6.36 (±0.16)	3.96 (±0.23)
SigNet-600dpi-F		2.46 (±0.09)	0.8 (±0.08)
SigNet-SPP-600dpi-F	Fixed	2.27 (±0.18)	0.65 (±0.11) ●
SigNet-SPP-600dpi-F	Multi	2.85 (±0.16)	0.86 (±0.1)

during training performs a little worse. The proposed method using SPP Fixed keeps about the same level of performance as the baseline, while removing the constraint of having a maximum signature size (since both SPP methods accept larger signatures for processing).

The results on transferring representations to different operating conditions are summarized in figure 3.5. We considered models trained on the GPDS dataset, and used these models to extract features and train WD classifiers on other operating conditions, that is, three other datasets: Brazilian PUC-PR, Cedar and MCYT. In all cases, we verify the impact of fine-tuning the representations for the new operating conditions, following the procedure detailed in section 3.3.3. For the first two datasets, that were scanned in 300 dpi, we used the representations learned in GPDS at 300 dpi, while for MCYT we used the representations learned at 600 dpi. In this experiment, we did not use any forgeries for training (neither from GPDS nor the

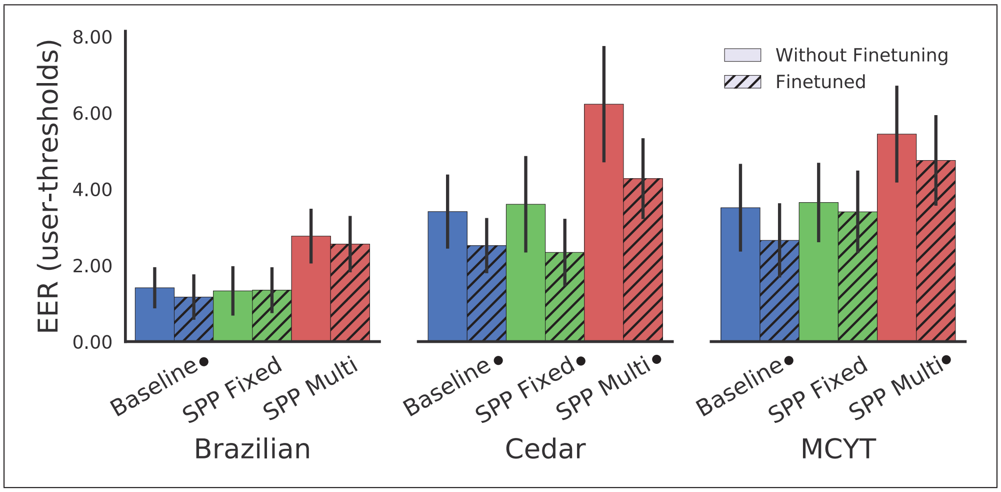


Figure 3.5 Classification performance of Writer-Dependent classifiers trained with representations learned in the GPDS dataset. The hatched bars denote results with features fine-tuned in each particular dataset. Error bars denote the standard deviation across 100 replications.

target dataset). We performed t-tests to verify if fine-tuning the representation significantly improved the classification performance (marked with a bullet point next to the dataset name). We can see that the baseline (without SPP) and the SPP model trained with fixed image sizes performed similarly, while SPP trained on multiple canvas sizes performed worse for transfer. We also consistently see that fine-tuning representations on the target datasets helps the domain adaptation, reducing the errors on average.

Table 3.4 shows the results of the experiments on transferring the representation to other types of scripts. The objective of this experiment was to verify if the features learned on the GPDS dataset generalizes to other types of script (in particular, we tested for Bengali and Devanagari). Differently from the previous analysis, for these datasets we consider the performance on discriminating genuine signatures and random forgeries (signatures from other users), since skilled forgeries are not available in these synthetic datasets. We consider experiments using the network trained on GPDS with no changes, and experiments where we finetune the representation to the particular dataset, following the protocol from section 3.3.3. Results that are significantly better (at $p < 0.01$) are shown with a bullet. We noticed an interesting

Table 3.4 Generalization performance on other datasets, with and without fine-tuning (for *random forgeries*)

Dataset	**Finetuned**	$\textbf{EER}_{\textbf{global threshold}}$	$\textbf{EER}_{\textbf{user thresholds}}$
Bengali		5.07 (±0.8)	3.41 (±0.81)
Bengali	Yes	0.77 (±0.27)	0.16 (±0.14) ●
Devanagari		4.65 (±0.92)	2.93 (±0.8)
Devanagari	Yes	0.33 (±0.2)	0.06 (±0.09) ●
MCYT		0.19 (±0.39)	0.03 (±0.13)
MCYT	Yes	0.04 (±0.12)	0.0 (±0.0)
CEDAR		1.14 (±0.75)	0.37 (±0.42)
CEDAR	Yes	0.23 (±0.26)	0.08 (±0.19) ●
Brazilian		0.47 (±0.3)	0.2 (±0.25)
Brazilian	Yes	0.5 (±0.38)	0.16 (±0.24)
Bengali (Results from Ferrer *et al.* (2017))	-	0.67	-
Devanagari (Results from Ferrer *et al.* (2017))	-	0.47	-

trend in these results, where without finetuning, the performance on other datasets that contain western-style signatures is already good (around or less than 1% for MCYT, CEDAR and Brazilian PUC-PR), but for the indian scripts the performance was much worse (3-5% EER). By finetuning the representation for the scripts, we obtain good performance (comparable with the previously reported in (Ferrer *et al.* (2017))). This suggests that the learned representation generalize better to users with western scripts than to other scripts. Multi-script learning approaches could be considered to improve performance on all scripts.

Lastly, tables 3.5, 3.6, 3.7 and 3.8 compare the results we obtained with SPP-Fixed (considering EER$_{\text{user thresholds}}$ using genuine signatures and skilled forgeries) with the state-of-the-art in GPDS, MCYT, Cedar and Brazilian PUC-PR, respectively. We observe results competitive to the state of the art in all datasets. In particular, in the GPDS dataset we notice big gains in performance (0.41% EER compared to 1.69% EER).

It is also worth noting that the MCYT dataset contains both Offline and Online signature data (for the same users). Historically, performance on online systems was greatly superior, but recent work on offline signature verification is closing the gap between the two strategies.

Table 3.5 Comparison with state-of-the art on the GPDS dataset (errors in %)

Reference	Dataset	#samples per user	Features	EER
Hu & Chen (2013)	GPDS-150	10	LBP, GLCM, HOG	7.66
Guerbai *et al.* (2015)	GPDS-160	12	Curvelet transform	15.07
Serdouk *et al.* (2015a)	GPDS-100	16	GLBP, LRF	12.52
Yılmaz & Yanıkoğlu (2016)	GPDS-160	5	LBP, HOG, SIFT	7.98
Yılmaz & Yanıkoğlu (2016)	GPDS-160	12	LBP, HOG, SIFT	6.97
Soleimani *et al.* (2016)	GPDS-300	10	LBP	20.94
Hafemann *et al.* (2017a)	GPDS-300	12	SigNet-F	1.69 (±0.18)
Present Work	GPDS-300	12	SigNet-SPP-300dpi	3.15 (±0.14)
Present Work	GPDS-300	12	SigNet-SPP-300dpi-F	0.41 (±0.05)

Table 3.6 Comparison with the state-of-the-art in MCYT (errors in %)

Reference	#samples per user	Features	EER
Gilperez *et al.* (2008)	5	Contours (chi squared distance)	10.18
Gilperez *et al.* (2008)	10	Contours (chi squared distance)	6.44
Wen *et al.* (2009)	5	RPF (HMM)	15.02
Vargas *et al.* (2011)	5	LBP (SVM)	11.9
Vargas *et al.* (2011)	10	LBP (SVM)	7.08
Ooi *et al.* (2016)	5	DRT + PCA (PNN)	13.86
Ooi *et al.* (2016)	10	DRT + PCA (PNN)	9.87
Soleimani *et al.* (2016)	5	HOG (DMML)	13.44
Soleimani *et al.* (2016)	10	HOG (DMML)	9.86
Hafemann *et al.* (2017a)	10	SigNet (SVM)	2.87 (± 0.42)
Present Work	10	SigNet-SPP-600dpi	3.64 (± 1.04)
Present Work	10	SigNet-SPP-600dpi (finetuned)	3.40 (± 1.08)

The best results on the literature achieve 2.85% EER (Rua & Castro (2012)) and 3.36% EER (Fierrez *et al.* (2007)) on the Online MCYT dataset, while for offline signature verification, performance is achieving around 3% $\text{EER}_{\text{user thresholds}}$. Although these results are not directly comparable (both Rua & Castro (2012) and Fierrez *et al.* (2007) implement per-user score

Table 3.7 Comparison with the state-of-the-art in CEDAR (errors in %)

Reference	**#samples per user**	**Features**	**AER/EER**
Chen & Srihari (2006)	16	Graph Matching	7.9
Kumar *et al.* (2010)	1	morphology (SVM)	11.81
Kumar *et al.* (2012)	1	Surroundness (NN)	8.33
Bharathi & Shekar (2013)	12	Chain code (SVM)	7.84
Guerbai *et al.* (2015)	4	Curvelet transform (OC-SVM)	8.7
Guerbai *et al.* (2015)	8	Curvelet transform (OC-SVM)	7.83
Guerbai *et al.* (2015)	12	Curvelet transform (OC-SVM)	5.6
Hafemann *et al.* (2017a)	12	SigNet-F (SVM)	4.63 (± 0.42)
Present Work	10	SigNet-SPP-300dpi	3.60 (± 1.26)
Present Work	10	SigNet-SPP-300dpi (finetuned)	2.33 (± 0.88)

Table 3.8 Comparison with the state-of-the-art on the Brazilian PUC-PR dataset (errors in %)

Reference	**#samples per user**	**Features**	**FRR**	**FAR$_{\text{random}}$**	**FAR$_{\text{simple}}$**	**FAR$_{\text{skilled}}$**	**AER**	**AER$_{\text{genuine + skilled}}$**	**EER$_{\text{genuine + skilled}}$**
Bertolini *et al.* (2010)	15	Graphometric	10.16	3.16	2.8	6.48	5.65	8.32	-
Batista *et al.* (2012)	30	Pixel density	7.5	0.33	0.5	13.5	5.46	10.5	-
Rivard *et al.* (2013)	15	ESC + DPDF	11	0	0.19	11.15	5.59	11.08	-
Eskander *et al.* (2013)	30	ESC + DPDF	7.83	0.02	0.17	13.5	5.38	10.67	-
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	15	SigNet-SPP-300dpi	0.69 (±0.51)	0.04 (±0.07)	0.14 (±0.2)	9.51 (±1.27)	2.59 (±0.35)	5.1 (±0.69)	1.33 (±0.65)
Present Work	15	SigNet-SPP-300dpi (finetuned)	0.63 (±0.57)	0.03 (±0.07)	0.14 (±0.2)	8.78 (±1.55)	2.39 (±0.39)	4.7 (±0.77)	1.35 (±0.6)

normalization with a single global threshold), it shows that the gap between the two approaches is being reduced.

3.6 Conclusion

In this work we proposed and evaluated two methods for adapting the CNN architectures to learn a fixed-size representation for signatures of different sizes. A simple method, of training a network with SPP in images of a fixed sized (and generalizing to signatures of any size) showed similar performance to previous methods, while removing the constraint of having a maximum signature size that could be processed.

Our experiments with different resolutions showed that using larger image resolutions do not always lead to improved performance. In particular, we empirically showed that using resolution higher than 100 dpi greatly improves performance if skilled forgeries (from a subset of users) is used for feature learning, but does not improve performance if only genuine signatures are used. This suggests that when learning features from skilled forgeries, the network can use detailed information about the signature strokes (e.g. if the writing is shaky, with limp strokes), while this information is ignored when only genuine signatures are used for training the CNN (when the network is only attempting to distinguish between different users).

Lastly, our experiments with transfer learning confirm previous results that features learned in one signature dataset generalize to other operating conditions. Our results also suggest that fine-tuning the representations (on a subset of the users in the new dataset) is useful to adapt the representations to the new conditions, improving performance. Especially for signatures from different styles than used for training (e.g. CNN trained on western signatures and generalizing to other types of script), finetuning showed to be particularly important. Other techniques, such as multi-script learning are also be promising for this scenario.

CHAPTER 4

META-LEARNING FOR FAST CLASSIFIER ADAPTATION TO NEW USERS OF SIGNATURE VERIFICATION SYSTEMS

Luiz G. Hafemann[1], Robert Sabourin[1], Luiz S. Oliveira[2]

[1] Department of Automated Manufacturing Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Departement of Informatics, Federal University of Parana (UFPR),
Curitiba, PR, Brazil

Article to be submitted to the « IEEE Transactions on Information Forensics & Security »

Abstract

Handwritten Signature verification presents a challenging Pattern Recognition problem, where only knowledge of the positive class is available for training. While classifiers have access to a few genuine signatures for training, during generalization they also need to discriminate forgeries. This is particularly challenging for *skilled* forgeries, where a forger practices imitating the user's signature, and often is able to create forgeries visually close to the original signatures. Most work in the literature address this issue by training for a surrogate objective: discriminating genuine signatures of a user and random forgeries (signatures from other users). In this work, we propose a solution for this problem based on meta-learning, where there are two levels of learning: a task-level (where a task is to learn a classifier for a given user) and a meta-level (learning across tasks). In particular, the meta-learner guides the adaptation (learning) of a classifier for each user, which is a lightweight operation that only requires genuine signatures. The meta-learning procedure learns what is common for the classification across different users. In a scenario where skilled forgeries from a subset of users are available, the meta-learner can guide classifiers to be discriminative of skilled forgeries even if the classifiers themselves do not use skilled forgeries for learning. Experiments conducted on the GPDS-960 dataset show improved performance compared to Writer-Independent systems, and achieve re-

sults comparable to state-of-the-art Writer-Dependent systems in the regime of few samples per user (5 reference signatures).

4.1 Introduction

Handwritten signature verification remains a challenging problem in the presence of skilled forgeries, where the forger has access to the user's signature and practices imitating it (Hafemann *et al.* (2017b)). This problem is particularly challenging since in a practical application scenario we cannot expect to have access to skilled forgeries for every user in the system for training the classifiers.

This problem is mainly addressed in three ways in the literature: (i) training a classifier for each user using a surrogate objective, where the negative samples are genuine signatures from other users (called *random forgeries* in this context) (Vargas *et al.* (2010); Yılmaz & Yanıkoğlu (2016); Hafemann *et al.* (2017a)) (ii) training a one-class classifier for each user (Guerbai *et al.* (2015)); (iii) training a global, writer-independent classifier (Kumar *et al.* (2012); Eskander *et al.* (2013); Rantzsch *et al.* (2016)). The first alternative (Writer Dependent (WD) classification) optimizes a surrogate objective, which therefore can be sub-optimal. The second alternative (one class Writer Dependent classification) is an appropriate formulation of the problem, but empirical results show that this approach performs worse than the first. A possible reason is that for signature verification tasks we normally have only a small number of samples per user, which makes it hard to estimate the support (or probability density) of the positive class. Lastly, the third alternative (Writer Independent (WI) classification) alleviates the problem of a small number of samples per user by transforming the problem in a binary classification problem: comparing a query signature with a reference (template) signature, where the same classifier is used for all users. However, empirically these approaches also show worse performance than WD classification, at least when the number of signatures available for training (per user) is larger than 1. We hypothesize that a reason for this gap in performance is that the WI classifiers compare a query signature with a reference signature one at a time, while the

WD classifiers are trained with multiple references at the same time, and therefore can better estimate the invariances in a person's signature (intra-class variation).

Considering different approaches, WD classification (alternative (i) above) shows better empirical performance (Hafemann *et al.* (2017b)). However, this approach has other shortcomings compared to WI approaches: they require training a classifier for each user, which is not desirable in some scenarios: For instance, when the number of users is very large, and each user do not use the system often - many classifiers are trained but are almost never used. Also, in the cases where features are learned from data (e.g. Hafemann *et al.* (2017a)), if we want to change the feature representation, for instance by training with new data, we would need to re-train all WD classifiers in the system, while a global (WI) classifier would not require any extra step. WI systems also naturally handles the issue of adding more signatures to the reference set.

In this work, we propose to formulate the task as a meta-learning problem, inspired by the work of a Forensics Handwritten Expert: the expert acquires knowledge examining genuine signatures and forgeries from several people along his/her training and work experience. For a new case, along with knowledge of signatures from the individual, this previous experience is also used when analyzing a signature of interest. We consider a meta-learner that learns across-tasks (classification for specific individuals), that is then adapted to a particular user in order to make a prediction on a query signature. In particular, we consider an established meta-learning algorithm: Model Agnostic Meta Learning (MAML) (Finn *et al.* (2017)), that we extend to use different loss functions during classifier adaptation and meta-learning, to address the issue of partial-knowledge during training. This approach learns directly from signature pixels, and the meta-learning procedure learns a representation that is highly *adaptable*: it is adapted to a new user by one (or a few) gradient descent steps. In this way, the adaptation of the classifier for a user is a lightweight operation that can be done on demand (e.g. for each query). This results in a system that is scalable as a WI system, but that is also adaptable for individual users. Additionally, contrary to other work that learns representations to train WD classifiers (Hafemann *et al.* (2017a)), not only the final classification layer is adapted to the new user, but the *feature representation* is also adapted.

We conducted most experiments with the GPDS-960 dataset (Vargas *et al.* (2007)). We evaluated the performance of proposed method in different scenarios such as varying the training set size and the number of gradient descent steps. On the GPDS-300 subset, the proposed method obtains better results than WI systems proposed in the literature, and approach the performance of WD systems, especially when few samples are available for training. With 5 reference signatures, the proposed method obtains 5.16% EER (using a global threshold), compared to 9.05% of a WI system and 5.25% of a WD system. For a larger number of references the WD system still performs better, but the gap in performance is greatly reduced. Considering 12 reference signatures, the method obtains 4.39% EER (with a global threshold), vs 3.74% for the WD system, while being more scalable (single meta-classifier). We also discuss some limitations of the system, most notably the requirement of using data from a large number of users for training, and worse results when transferring the meta-learner to the other datasets, by testing the system in the MCYT-75 (Ortega-Garcia *et al.* (2003)), CEDAR (Kalera *et al.* (2004)) and Brazilian PUC-PR (Freitas *et al.* (2000)) datasets.

The paper is organized as follows: section 4.2 reviews the related work on signature verification and meta-learning, and describes the common strategy of performing WD classification using learned features. Section 4.3 introduces the formulation of signature verification as a meta-learning problem, and the proposed algorithm. Section 4.4 describes the experimental protocol, and section 4.5 presents and discusses our results. Finally, section 4.6 concludes the paper.

4.2 Related Work

The objective of signature verification systems is to classify a query signature as being genuine (produced by the claimed individual), or a forgery (produced by another person). In the Pattern Recognition community, different forgeries are considered: Random forgeries - in which the forger has no knowledge of the user's signature, and use his signature instead; Simple forgeries - in which the forger knows the person's name, but not their signature; Skilled forgeries - where the forger has access to the user's signature, and practices imitating it. While the problem of distinguishing random and simple forgeries is relatively easy (i.e. low error rates in state-of-the-art classifiers), skilled forgeries still present a significant challenge for classification.

These systems can be broadly categorized as Writer-Dependent (WD, also called User-Dependent) and Writer-Independent (WI, also called User-Independent). For Writer-Dependent classifiers, we consider a dataset for each user $\{x,y\}_{i=1}^{n}$, where x are signatures, and y indicate whether they are genuine signatures from the user ($y = 1$) or random forgeries ($y = 0$) (Vargas *et al.* (2010); Yılmaz & Yanıkoğlu (2016); Hafemann *et al.* (2017a)). Some work consider one-class WD classifiers, in which only genuine signatures from the user are used for training (only $y = 1$) (Guerbai *et al.* (2015)). For WI classifiers, there are two main approaches: training a single classifier in a *dissimilarity* space, and *metric learning* approaches. In the first case, the training samples are difference of feature vectors: $|\phi(x_1) - \phi(x_2)|$, with $y = 1$ if both signatures are from the same user, and $y = 0$ otherwise (Kumar *et al.* (2012); Eskander *et al.* (2013)). The metric learning approaches use a siamese network architecture (Bromley *et al.* (1994)), which takes two signatures (x_1, x_2) as input, and outputs a metric (distance) between them.

Recent work on signature verification rely on feature learning methods (Hafemann *et al.* (2016b, 2017a); Zois *et al.* (2017, 2018a); Rantzsch *et al.* (2016)), in which learning is conducted directly from signature pixels, instead of relying on handcrafted feature extractors. When used in conjunction with WD classifiers, feature learning is conducted for another surrogate objective, e.g. dictionary learning (Zois *et al.* (2017, 2018a)), or classifying the user that produce the signatures (Hafemann *et al.* (2017a)). For WI classification, the system can be trained jointly (feature extraction and classification) (Rantzsch *et al.* (2016)). Despite being jointly trained, such WI systems still perform worse than WD classifiers trained with features learned with surrogate objectives, at least when more than one signature references are used. A possible reason for this gap is the fact that WI systems compare the query signature to each reference individually (or comparing with the centroid of the signatures), which is less powerful than training a classifier for the user, in capturing the invariances of the person's signature.

4.2.1 Meta-learning

In a broad sense, meta-learning is concerned with the problem of *learning to learn*, with origins in the 80's and 90's (Schmidhuber (1987), Bengio *et al.* (1991)). More recently, algorithms based on meta-learning have achieved state-of-the-art results in tasks such as hyperparameter optimization (Maclaurin *et al.* (2015)), neural network architecture search (Baker *et al.* (2017))), and few-shot learning (Ravi & Larochelle (2017); Finn *et al.* (2017)). Few-shot learning considers a scenario where only a few samples from each class are available for training, which is similar to actual application scenarios in handwritten signature verification.

The goal of these meta-learning approaches for few-shot learning is to train a model that can quickly (i.e. in a few iterations) adapt to a new task using only a few samples. A new task in this context refers, for instance, to classify a new object, for which only a few samples are known. Ravi & Larochelle (2017) proposed learning an *optimizer* and *initialization* for the tasks (Meta Nets). They propose using a Long short-term memory (LSTM) model to learn the update rule for adapting the network parameters to a new task. Finn *et al.* (2017) proposed a Model Agnostic Meta Learning (MAML) procedure that does not require any extra parameters. This model optimizes the *sensitivity* of the weights, that is, obtain a feature representation that is highly adaptive, such that a single (or a few) gradient descent iterations are sufficient to optimize to new tasks.

4.2.2 Revisiting WD classification using learned features

Current state-of-the-art signature verification methods consider a feature-learning phase, where a function $\phi(x)$ is learned to *extract features* from signature images x. This is commonly learned with a surrogate objective (e.g. classifying users (Hafemann *et al.* (2017a)), dictionary learning (Zois *et al.* (2017))). This feature representation is learned from a Development dataset $\mathscr{D}$, which is then used to extract features and train Writer-Dependent classifiers for the users of interest (exploitation set $\mathscr{E}$). A dataset $\mathscr{V}_v$ (Validation set for verification) is used to optimize hyperparameters for feature learning. The exploitation set if further divided in training ($\mathscr{L}_v$,

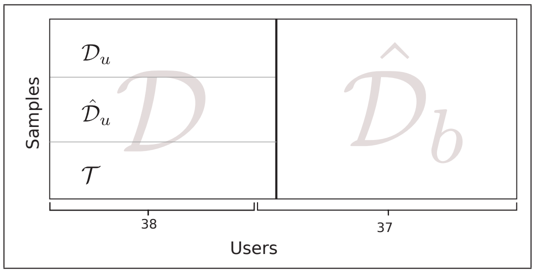


Figure 4.1 Common dataset separation for Feature Learning followed by WD classification.

used to train the WD classifiers) and test ($\mathscr{T}_v$, use to evaluate generalization performance). Furthermore, some formulations consider that skilled forgeries from a subset of users (from $\mathscr{D}$) are available for training, and measure the impact of classification for users in $\mathscr{E}$. This dataset division is illustrated in figure 4.1.

While this approach achieved state-of-the-art classification (Hafemann *et al.* (2017a)), it has some shortcomings: it requires training one classifier for each user, which may be an expensive operation (e.g. best results were reported with an SVM trained with the RBF kernel for each user). If the feature extractor is updated (e.g. trained with more data), then all classifiers need to be retrained. Also, these systems use a fixed representation for all users, and it is possible that adapting the representation for each user would yield improvements in classification performance.

4.3 Proposed Method

In this work we propose a meta-learning approach for signature verification. This formulation considers a meta-learner that guides the adaptation of a classifier for each user. We consider that each user describes a *task*: discriminating between genuine signatures (created by the user) and forgeries. Figure 4.2 illustrates the data available for one task: we consider a reference (support) dataset that is used for training a classifier that can classify new queries as genuine or forgery.

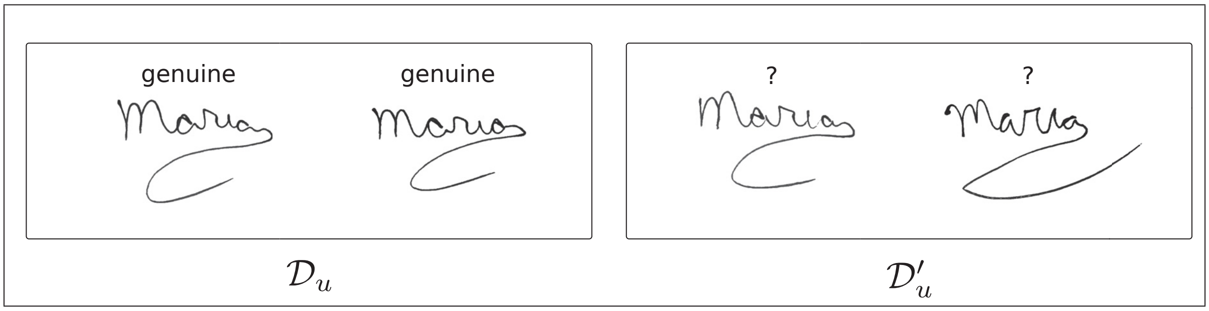


Figure 4.2 Illustration of the data available for one task (user). Left: the reference (support) set. Right: query samples.

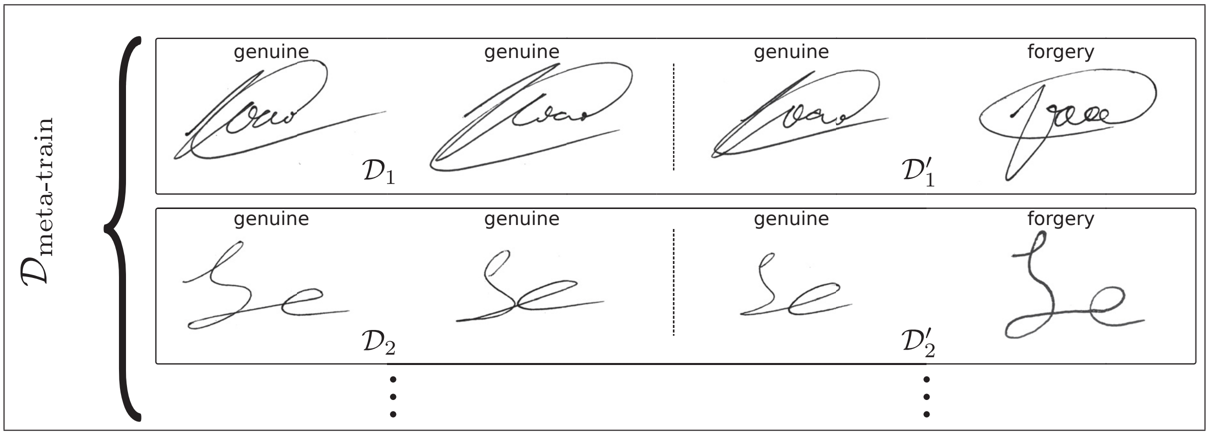


Figure 4.3 Example of the meta-learning setup. Each user represents an *episode*, where $\mathscr{D}_u$ is used for classifier adaptation and $\mathscr{D}'_u$ is used for meta-update.

In a meta-learning setting, we consider that training a classifier for a particular user is guided by a meta-learner, that leverages data from multiple tasks for learning. For this we consider a dataset $\mathscr{D}_{\text{meta-train}}$, and then evaluate the generalization performance on unseen users $\mathscr{D}_{\text{meta-test}}$.

We note that this approach has a direct correspondence to previous work that used feature learning followed by WD classification (section 4.2.2), and here we make the association between the terminology in the meta-learning research and previous work on Signature Verification. In both cases we use a separate set of users for feature learning ($\mathscr{D}_{\text{meta-train}}$ is analogous to the development set in sec. 4.2.2), which is then used for to train and test classifiers on a new set of users ($\mathscr{D}_{\text{meta-test}}$ is analogous to the exploitation set). The key differences of meta-learning is that: (i) The loss optimized for feature learning is directly related to the final objective (separate genuine signatures and forgeries); (ii) training a classifier for a new user is a lightweight process (a few gradient descent iterations); (iii) not only the classifier, but the features are also adapted for each user.

In the next section we formalize the problem of signature verification as a meta-learning task.

4.3.1 Problem formulation

Table 4.1 Table of symbols

Symbol	**Description**
$\mathscr{T}$	Distribution of tasks (i.e. users)
$\mathscr{T}_u$	Task for user u
$\mathscr{D}_{\text{meta-train}}$	Training set for the meta-learner
$\mathscr{D}_{\text{meta-test}}$	Testing set for the meta-learner
$\mathscr{D}_u$	Samples for weight adaptation for user u
$\mathscr{D}'_u$	Samples for meta-update for user u
G_u	Genuine signatures for user u
S_u	Skilled forgeries for user u
θ	Network parameters
$\theta_k^{(u)}$	Parameters adapted to user u after k descent steps
L	Loss function for weight adaptation
L'	Loss function for meta-update

We consider that each user describes a task $T_u \in \mathscr{T}$, where the task consists in classifying a signature image as genuine (created by the user) or forgery (not created by the user). A collection of users therefore describes a distribution of tasks $\mathscr{T}$, and the aim of the meta-learner is to explore the structure present in this distribution. We consider a dataset $\mathscr{D}_{\text{meta-train}}$ containing tasks from $\mathscr{T}$, that is used for meta-learning. For each user we consider a set $\mathscr{D}_u$, that is used to adapt the classifier, and a set $\mathscr{D}'_u$ that is used for updating the meta-learner. Lastly, to verify the generalization to unseen users, we consider a set $\mathscr{D}_{\text{meta-test}}$, that contains data from a disjoint set of users ($\mathscr{D}_{\text{meta-train}} \cap \mathscr{D}_{\text{meta-test}} = \emptyset$). Figure 4.3 illustrates the meta-learning setup, and the symbols used in this paper are listed in Table 4.1 for clarity.

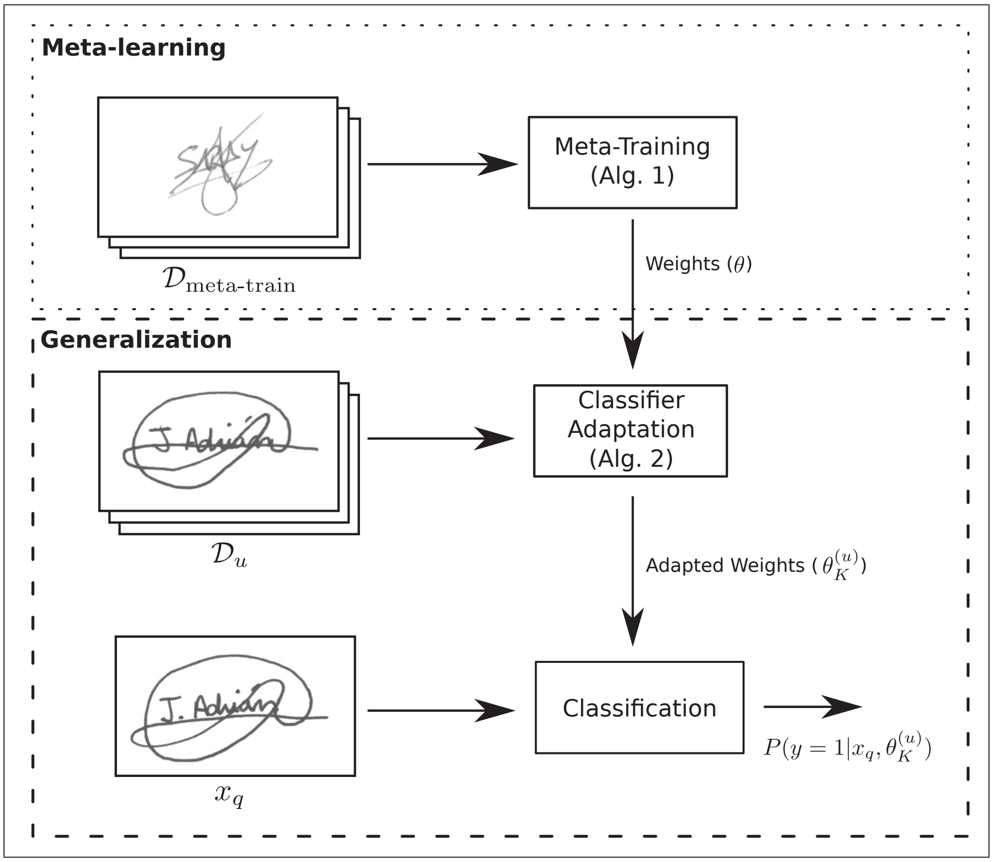

Meta-learning
Meta-Training (Alg. 1)
$\mathcal{D}_{\text{meta-train}}$
Weights (θ)
Generalization
Classifier Adaptation (Alg. 2)
$\mathcal{D}_u$
Adapted Weights ($\theta_K^{(u)}$)
Classification
$P(y=1|x_q, \theta_K^{(u)})$
x_q

Figure 4.4 Overview of the meta-learning system for signature verification.

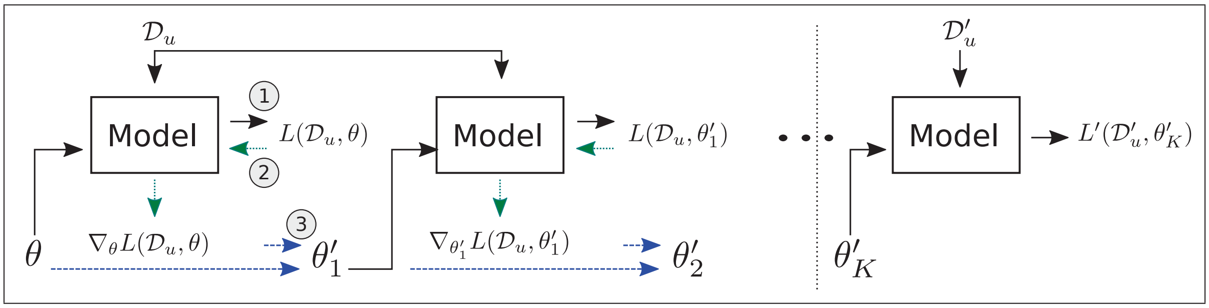

$\mathcal{D}_u$
Model
1
$L(\mathcal{D}_u, \theta)$
2
Model
$L(\mathcal{D}_u, \theta'_1)$
$\mathcal{D}'_u$
Model
$L'(\mathcal{D}'_u, \theta'_K)$
3
θ
$\nabla_\theta L(\mathcal{D}_u, \theta)$
θ'_1
$\nabla_{\theta'_1} L(\mathcal{D}_u, \theta'_1)$
θ'_2
θ'_K

Figure 4.5 Illustration of one iteration of meta-training for one task T_u. Starting with parameters θ, the weights are specialized for the task in K gradient descent steps. Each step involves computing the loss (1), back-propagating the loss w.r.t to θ'_{k-1} (2) and updating the weights (3). For the meta-update, the loss L' is backpropagated through the entire chain (from L' back to the initial θ), computing $\nabla_\theta L'(\mathscr{D}'_u, \theta^u_K)$.

4.3.2 Model Agnostic Meta-Learning for signature verification

In this work we propose an extended version of Model-Agnostic Meta-Learning (MAML) (Finn *et al.* (2017)), by considering different criteria for classifier adaptation and meta-learning. An overview of the system can be seem in figure 4.4. We consider a development set for meta-training, that consists in learning the weights θ of a Convolutional Neural Network, that are highly *adaptable* to new tasks. During generalization, for a user u, a reference set $\mathscr{D}_u$ is used to adapt the classifier to this user (using K gradient descent steps) obtaining weights $\theta_K^{(u)}$. This adapted classifier is then used to classify a query image x_q, obtaining $P(y=1|x_q,\theta_K^{(u)})$.

Algorithm 4.1 Meta-Training algorithm

```
1  Input: M: Meta-batch size
2  Input: K: Number of gradient descent steps
3  Input: α, β Learning rates
4  Output: θ: Meta-learned weights
5  Randomly initialize θ
6  while not done do
7      Sample a batch of tasks $\{T_u\}_{u=1}^{M} \sim \mathscr{T}$
8      $\theta_{\text{grad}} \leftarrow \vec{0}$
9      for $u \leftarrow 1$ to $M$ do
10         Sample $\mathscr{D}_u$ // Genuine only
11         $\theta'_0 \leftarrow \theta$
12         for $k \leftarrow 1$ to $K$ // Adapt weights to $u$
13          do
14             $\theta'_k \leftarrow \theta'_{k-1} - \alpha\nabla_{\theta'_{k-1}} L(\mathscr{D}_u, \theta'_{k-1})$
15         end
16         Sample $\mathscr{D}'_u$ // Genuine and forgeries
17         Compute predictions: $P(\mathbf{y}_{\text{test}}|\mathbf{X}_{\text{test}}, \theta'_K)$
18         $\theta_{\text{grad}} \leftarrow \theta_{\text{grad}} + \frac{1}{M}\nabla_\theta L'(\mathscr{D}'_u, \theta'_K)$
19     end
20     $\theta \leftarrow \theta - \beta\theta_{\text{grad}}$ // Meta-update
21 end
```

Algorithm 4.1 describes the full meta-training algorithm. Meta-training is conducted in *episodes* (Figure 4.3). In each episode, the classifier is adapted to a particular user using $\mathscr{D}_u$ (lines 7 to 10), and the adapted classifier is used to classify the set $\mathscr{D}'_u$. The loss is then back-propagated through all intermediate steps of the classifier adaptation (lines 11 and 12), and is used to update the meta-learner weights θ (line 14). Therefore, instead of having a feature representation that is directly applicable for any user, they are learned to work well for new users *after K* gradient descent steps on the user's signatures. For stability during training, we train on “mini-batches" of episodes, by accumulating the gradients for M episodes before updating θ.

Figure 4.5 illustrates the classifier adaptation procedure. In this work, we adapt the MAML algorithm to use different loss functions for the classifier adaptation and the final loss (used for the meta-update). In particular, we consider a loss function L that only uses genuine signatures for the classifier adaptation, and a loss function L' that use both genuine signatures and forgeries. Let $\mathscr{D}_u = G_u \cup G_{i \neq u}$ be the training set consisted of genuine signatures from the user (G_u) and random forgeries ($G_{i \neq u}$). We consider the following loss for classifier adaptation:

$$\begin{aligned} L(\mathscr{D}_u, \theta) = & -\frac{1}{|G_u|} \sum_{x:G_u} \log(P(y|x,\theta)) \\ & -\frac{1}{|G_{i \neq u}|} \sum_{x:G_{i \neq u}} \log(P(y|x,\theta)) \end{aligned} \tag{4.1}$$

where $|G_u|$ and $|G_{i \neq u}|$ are the number of users in the sets, which is used to correct for the imbalance between the two classes.

Let $\mathscr{D}'{}_u = G'_u \cup G'_{i \neq u} \cup S'_u$ be the a disjoint set of signatures for user u: genuine signatures (G'_u), random forgeries ($G'_{i \neq u}$), and (if available), skilled forgeries S'_u. We define the loss function for meta-update as follows:

$$\begin{aligned} L'(\mathscr{D}'_u, \theta) = & -\frac{1}{|G'_u|} \sum_{x:G'_u} \log(P(y|x,\theta_K^{(u)})) \\ & -\frac{1}{|G'_{i \neq u}|} \sum_{x:G'_{i \neq u}} \log(P(y|x,\theta_K^{(u)})) \\ & -\frac{1}{|S'_u|} \sum_{x:S'_u} \log(P(y|x,\theta_K^{(u)})) \end{aligned} \tag{4.2}$$

Algorithm 4.2 Classifier adaptation

```
**Input:** $K$: Number of gradient descent steps
**Input:** $\alpha$ Learning rate
**Input:** $\theta$ Meta-learned weights
**Input:** $\mathscr{D}_u$ Reference set for user $u$
**Output:** $\theta'_K$: Weights adapted to the user after $K$ steps
$\theta'_0 \leftarrow \theta$
for $k \leftarrow 1$ to $K$ do
    $\theta'_k \leftarrow \theta'_{k-1} - \alpha \nabla_{\theta'_{k-1}} L(\mathscr{D}_u, \theta'_{k-1})$
end
```

On generalization, for a new user we first adapt the weights to this user using a set of reference signatures $\mathscr{D}_u$, and then classify a new query signature using the adapted weights. Algorithm 4.2 describes the classifier adaptation to a new user. We note that only the loss function L is used, and therefore only genuine signatures are used when adapting a classifier for a new user.

4.3.3 Meta-learning for one-class classification

The approach defined above can also be extended for one-class classification, where the classifier adaptation is done with only genuine signatures from the user of interest. This is easily implemented by considering $\mathscr{D}_u = G_u$. It is worth noting that similarity-based methods and one-class methods that involve feature learning often suffer from the problem of collapsing representations into a point (Perera & Patel (2018)). This is often addressed by adding a penalty in the loss function that requires dissimilar items to be far apart in the feature space. In our formulation, while the user's classifier is only trained with data from one class, we observe that training does not collapse to a single point since the meta-training procedure directly optimizes the performance on separating forgeries in $\mathscr{D}'_u$.

4.4 Experimental Protocol

We conducted most experiments on the GPDS-960 dataset (Vargas *et al.* (2007)), that consists of 881 users, with 24 genuine signatures per user and 30 skilled forgeries. We follow the same dataset separation as previous work (figure 4.1), with users 350-881 as $\mathscr{D}_{\text{meta-train}}$, 300-350 as $\mathscr{D}_{\text{meta-val}}$ and users 0-300 as $\mathscr{D}_{\text{meta-test}}$. We used the same pre-processing method from previous work (Hafemann *et al.* (2016b, 2017a)), by removing the background noise using Otsu, centering the images in a canvas of size 952×1360 and resizing them to 170×242. We also conducted experiments with the datasets MCYT-75 (Ortega-Garcia *et al.* (2003)), CEDAR (Kalera *et al.* (2004)) and Brazilian PUC-PR (Freitas *et al.* (2000)), to investigate the transferability of the meta-learner.

We analyze the impact of the hyperparameters in the classifier's performance, measured in $\mathscr{D}_{\text{meta-val}}$. We consider the experiments by varying these parameters:

- Number of gradient descent steps in the classifier adaptation: $K \in \{1,5\}$
- One-class classification vs adaptation using genuine signatures and random forgeries
- Fraction of users with skilled forgeries available for training
- Performance as we vary the number of reference genuine signatures

We compare the results on $\mathscr{D}_{\text{meta-val}}$ with a baseline using feature learning followed WD classification (Hafemann *et al.* (2017a)). As in (Hafemann *et al.* (2017a)), we evaluate each model with repeated random subsampling: we randomly partition the validation set into training ($\mathscr{D}_u$) and testing ($\mathscr{D}'_u$), repeating the experiment 10 times with different partitions. We report the mean and standard deviation of the metrics.

In all experiments, we train the meta-classifier for a total of 100 epochs, considering a meta-batch size $M = 4$. We consider an initial meta-learning rate $\beta = 0.001$, that is reduced (with cosine annealing) to 10^{-5} by the last epoch. We used early stopping, by keeping the meta-learner weights that performed best in the validation set. Following (Antoniou *et al.* (2019)), we used Multi-Step Loss Optimization (MSL) for improving training stability. For the first 20 iterations, instead of computing the loss function L' only after K steps (step 12 of algorithm 4.1), we compute the loss function for all intermediate θ'_k, and consider a weighted average of the losses. In the first epoch the loss using each θ'_k contributes equally to the loss function, and the weights are annealed to give more weight to the last step until iteration 20, after which only the loss function at the final step K contributes to the loss. We found this procedure effective in stabilizing training (measured by the variation in validation accuracy across epochs). We also attempted to use learnable task learning rates (LSLR) described in Antoniou *et al.* (2019) without success. Empirically, we also noticed that when using only genuine signatures the task learning rate needs to be larger than the case where skilled forgeries are available for training. In our experiments, if the fraction of users with skilled forgeries is less than 10% we used a task learning rate $\alpha = 0.01$, and a learning rate of $\alpha = 0.001$ for the other experiments.

In order to evaluate the transferability of the features to other operating conditions, we conducted experiments on other datasets, (that were collected in different regions, and followed different collection processes). We conducted two experiments: (i) use the meta-learner trained on GPDS directly for new users of these datasets; (ii) train a meta-learner with data from the four datasets. It is worth noting that, with the exception of GPDS, the datasets are relatively small, with 55, 75 and 60 users for CEDAR, MCYT and Brazilian PUC-PR. We observed that the formulations from this work require a large amount of users for training, and for this reason, we conducted 10-fold cross validation. We divide each dataset in 10 folds (by users), and for each run we consider 1 fold as meta-test, and the remaining folders for meta-training and validation. As in the previous experiments, we further use repeated subsampling for evaluating the adaptation for the new users. In total, for experiment (ii), we train 10 CNN models and perform 10 adaptations for each user. We report the mean error rates over all runs, and the standard deviation across the 10 different adaptations (each based on different train/test splits of the repeated subsampling).

The CNN architecture used in the experiments is listed in table 4.2. We found that using a smaller network, compared to previous work using feature learning followed by WD classification, was successful in the meta-learning setting. This network has a total of 1.4M weights and uses 0.1 GFLOPS for forward propagation, while SigNet (Hafemann *et al.* (2017a)) has

Table 4.2 Base architecture used in this work

Layer	**Size**
Input	1x150x220
Convolution (C1)	32x5x5
Max Pooling	32x5x5
Convolution (C2)	32x5x5
Pooling	32x5x5
Fully Connected (FC3)	1024
Fully Connected (FC4)	256
Fully Connected + Sigmoid	1

15.8M weights and uses 0.6 GFLOPS. That is, the CNN used in this work is 10x smaller and 6x times faster.

We evaluate the performance 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: $\text{EER}_{\text{global }\tau}$: using a global decision threshold and $\text{EER}_{\text{user }\tau}$: using user-specific decision thresholds. In both cases, to calculate the Equal Error Rate we only considered skilled forgeries. For FRR and FAR, we report the values with a threshold of 0.5 (i.e. if $p(y=1|x,\theta'_K) \geq 0.5$ we consider the model predicting x as a genuine signature).

Table 4.3 Performance on $\mathscr{D}_{\text{meta-val}}$ with one-class and two-class formulations

Type	**#Gen**	**#RF**	**FRR**	$\textbf{FAR}_{\textbf{random}}$	$\textbf{FAR}_{\textbf{skilled}}$	$\textbf{EER}_{\textbf{global }\tau}$	$\textbf{EER}_{\textbf{user }\tau}$
SigNet* + WD	5	7434	10.48 (±2.24)	0.03 (±0.01)	24.67 (±0.99)	17.03 (±1.06)	13.17 (±0.94)
SigNet-F* + WD	5	7434	18.08 (±1.49)	0.16 (±0.04)	1.55 (±0.22)	4.6 (±0.59)	3.08 (±0.38)
Meta-learning One-class	5	-	2.54 (±0.61)	2.74 (±0.93)	4.24 (±0.93)	3.48 (±0.57)	1.69 (±0.43)
Meta-learning Two-class	5	5	2.82 (±0.59)	1.98 (±0.55)	4.18 (±0.48)	3.8 (±0.48)	2.04 (±0.41)
	5	10	5.1 (±0.99)	1.94 (±0.27)	2.66 (±0.76)	3.56 (±0.57)	1.85 (±0.57)
	5	20	2.84 (±0.97)	1.98 (±0.48)	3.1 (±0.83)	2.86 (±0.59)	1.78 (±0.27)
	5	30	2.62 (±0.82)	2.48 (±0.39)	3.46 (±0.62)	3.17 (±0.37)	1.4 (±0.48)

4.5 Results

4.5.1 System design

In this section we report the results on $\mathscr{D}_{\text{meta-val}}$ (GPDS users 300-350), considering the experiments defined in section 4.4. The objective is to evaluate different aspects of the system, such as the number of gradient steps (that trades-off comstarutation complexity and accuracy), as well as investigate the performance of the model in different data scenarios.

In a first experiment we consider the results of the one-class formulation and the two-class formulation as we vary the number of Random Forgeries used for classifier adaptation (#RF). For this experiment, use used 5 genuine signatures for classifier adaptation, and $K = 5$ gradient descent steps; for meta-training we considered that skilled forgeries were available on $\mathscr{D}_{\text{meta-train}}$ (users 350-881). Note that for validation, no skilled forgeries were used for training. Table 4.3 reports the results of these experiments. We observe similar verification performance on the two formulations. Note that the formulation using random forgeries is more computationally expensive, as the classifier adaptation involves a larger batch of images (e.g. computing the loss for one-class uses 5 images, while for two-class with #RF=30 uses 35 images). We also compare with a method using feature learning followed by WD classification (Hafemann *et al.* (2017a)). The entries denoted *SigNet** used the same approach proposed in (Hafemann *et al.* (2017a)), but using the CNN architecture defined for this work (table 4.2). We note that the meta-learning formulation performed much better, while being a simpler model (single model for all users). A comparison with the SigNet CNN architecture from (Hafemann *et al.* (2017a)) is conducted in section 4.5.2, where we compare to the state-of-the-art.

Figure 4.6 shows the results on verification performance as we vary the number of gradient descent steps K. For each value of K, we meta-trained a network and evaluate its performance on $\mathscr{D}_{\text{meta-val}}$. We observed improved performance with larger number of steps, but with diminishing returns. As we increase the number of steps, however, we increase the computational cost. If we consider that forward propagation and backward propagation have similar cost, the

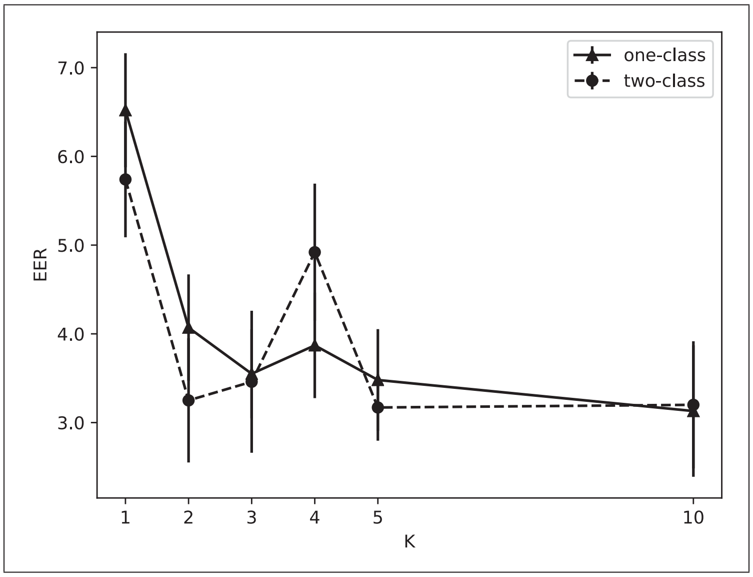


Figure 4.6 Performance on $\mathscr{D}_{\text{meta-val}}$ as we vary the number of update steps K.

classifier adaptation for a new user takes $2K$ the time for a single forward pass. A higher K also requires more memory (in the order of $2K$) during meta-training, since the whole update sequence needs to be stored in memory in order to compute the gradient for meta-update (as can be seen in figure 4.5).

In figures 4.7 and 4.8 we analyze the impact in performance as we vary the size of the $\mathscr{D}_{\text{meta-train}}$ set. As noted in section 4.3.2, if skilled forgeries from a subset of users are available, we can incorporate them into the meta-update loss function L'. In this experiment we considered that $\mathscr{D}_{\text{meta-train}}$ contains all 531 users, and vary the number of users for which skilled forgeries are available. For each case, we build a dataset consisting of genuine signatures for all users and skilled forgeries for the selected users, and trained a model. Figure 4.7 shows the performance as we vary the number of users for which skilled forgeries as available. We re-iterate that we evaluate the performance on a disjoint set of users ($\mathscr{D}_{\text{meta-val}}$) for which only genuine signatures are used. We observed that the meta-learning formulation of the problem is well suited to incorporating information from skilled forgeries (when it is available), and this generalizes well to unseen users, for which we only have genuine signatures. However, we observed that the performance is not very good when there are only genuine signatures for meta-training: the

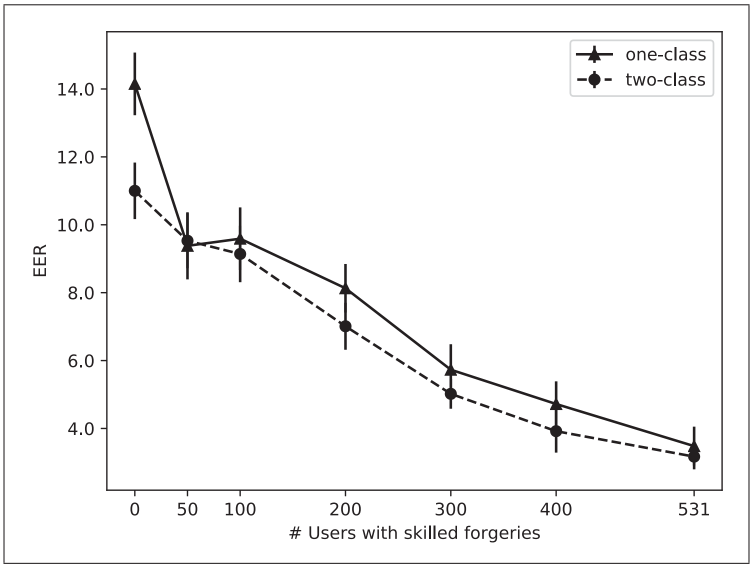
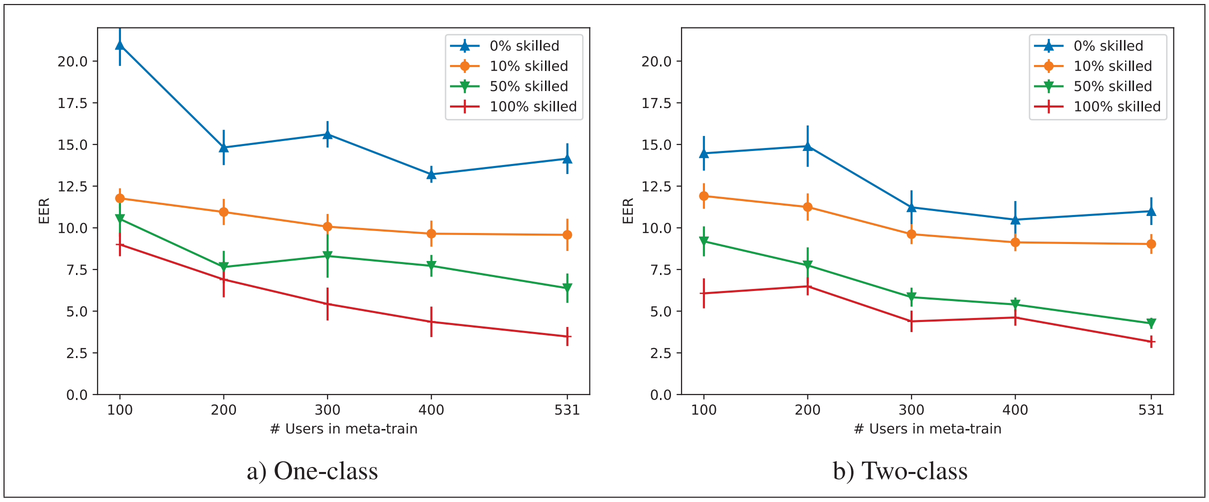


Figure 4.7 Performance on $\mathscr{D}_{\text{meta-val}}$ as we vary the number of users in $\mathscr{D}_{\text{meta-train}}$ for which skilled forgeries are available.



a) One-class

b) Two-class

Figure 4.8 Performance on $\mathscr{D}_{\text{meta-val}}$ as we vary the number of users available for meta-training. (a): one-class formulation; (b) two-class formulation.

one-class formulation achieves 14.15% EER when only genuine signatures are available, and 3.48% EER when skilled forgeries are available for all 531 users in meta-training.

In figure 4.8, we evaluate the performance of the system as we vary the number of users in $\mathscr{D}_{\text{meta-train}}$. We also consider 4 levels of availability of skilled forgeries in the meta-training set: 0% (genuine only), 10%, 50% and 100%, where the percentages refer to the number of users for which skilled forgeries are available (e.g. 10% with 100 users means that forgeries for 10 users are considered, where the remaining 90 users have only genuine signatures). For a given number of users and skilled forgery percentage, we construct a dataset with randomly selected users (taken from the 531 users in the development set), with genuine signatures from all the selected users, and skilled forgeries for a fraction of the users. We then use this dataset for meta-training a model, and evaluate its performance on $\mathscr{D}_{\text{meta-val}}$. We observed improved performance both as more users are available for meta-training, as well as when more knowledge of skilled forgeries is available. Most surprisingly, we observed that for the two-class formulation, a classifier trained with 100 users with 100% forgeries (i.e. forgeries for every user in meta-train) performed better than a model trained with 531 users with forgeries for only 100 users (comparing figures 4.8b and 4.7): 6.07% EER vs 9.14% EER. We re-iterate that this measures the performance on discriminate genuine signatures and skilled forgeries, and the model that has access to more users (with the same amount of users with skilled forgeries) has better performance on discriminating random forgeries, since its optimization consisted mostly of this problem.

4.5.2 Comparison with the state-of-the-art

We now compare our results with the state-of-the-art in the GPDS-300 dataset. For these comparisons, we considered a model trained with the one-class formulation, and a model trained with the two-class formulation, with $r = 30$ forgeries. In both cases, we used the whole dataset $\mathscr{D}_{\text{meta-train}}$ for training, and used 5 genuine signatures for training and $k = 5$ updates. While training was conducted with 5 reference signatures, we evaluate the performance of the system with different number or references.

Table 4.4 compares our results with the state-of-the-art. We observe an improved performance compared to other WI systems, achieving 5.16% EER (global τ) with 5 reference signatures, compared to 9.05% from (Souza *et al.* (2018)). Comparing to WD systems, we observed similar performance in some scenarios (5 reference signatures), and worse results otherwise.

Table 4.4 Comparison with state-of-the art on the GPDS dataset (errors in %)

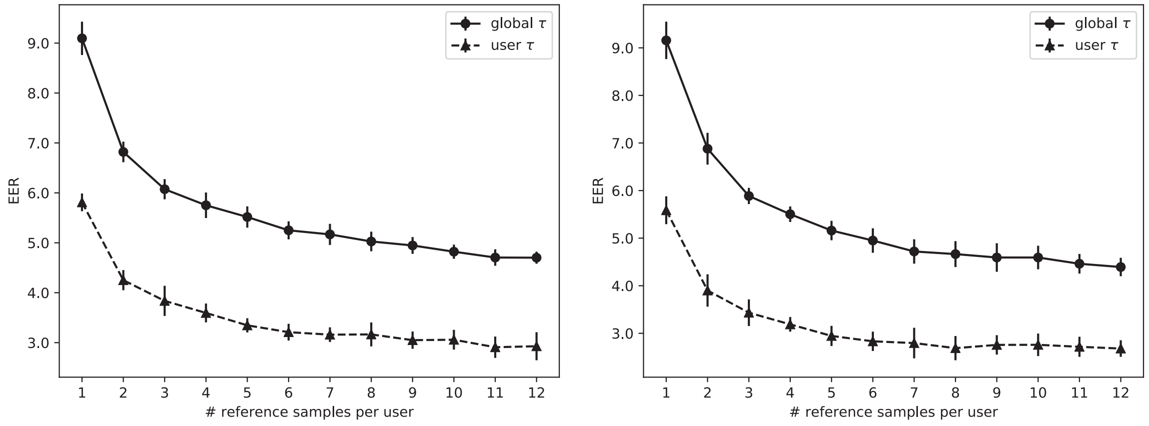
Reference	Type	Dataset	#samples per user	Features	EER
Hu & Chen (2013)	WI	GPDS-150	10	LBP, GLCM, HOG	7.66
Guerbai *et al.* (2015)	WD	GPDS-160	12	Curvelet transform	15.07
Serdouk *et al.* (2015a)	WD	GPDS-100	16	GLBP, LRF	12.52
Yılmaz & Yanıkoğlu (2016)	WD	GPDS-160	5	LBP, HOG, SIFT	7.98
Yılmaz & Yanıkoğlu (2016)	WD	GPDS-160	12	LBP, HOG, SIFT	6.97
Soleimani *et al.* (2016)	WI	GPDS-300	10	LBP	20.94
Hafemann *et al.* (2017a)	WD	GPDS-300	5	SigNet-F (global τ)	5.25 (±0.15)
Hafemann *et al.* (2017a)	WD	GPDS-300	5	SigNet-F (user τ)	2.42 (±0.24)
Hafemann *et al.* (2017a)	WD	GPDS-300	12	SigNet-F (global τ)	3.74 (±0.15)
Hafemann *et al.* (2017a)	WD	GPDS-300	12	SigNet-F (user τ)	1.69 (±0.18)
Souza *et al.* (2018)	WI	GPDS-300	5	SigNet (global τ)	9.05 (±0.34)
Souza *et al.* (2018)	WI	GPDS-300	5	SigNet (user τ)	4.40 (±0.34)
Souza *et al.* (2018)	WI	GPDS-300	12	SigNet (global τ)	7.96 (±0.26)
Souza *et al.* (2018)	WI	GPDS-300	12	SigNet (user τ)	3.34 (±0.22)
Present work	WI/WD	GPDS-300	5	MAML one-class (global τ)	5.52 (±0.20)
Present work	WI/WD	GPDS-300	5	MAML one-class (user τ)	3.35 (±0.13)
Present work	WI/WD	GPDS-300	5	MAML two-class (global τ)	5.16 (±0.19)
Present work	WI/WD	GPDS-300	5	MAML two-class (user τ)	2.94 (±0.20)
Present work	WI/WD	GPDS-300	12	MAML one-class (global τ)	4.70 (±0.11)
Present work	WI/WD	GPDS-300	12	MAML one-class (user τ)	2.93 (±0.27)
Present work	WI/WD	GPDS-300	12	MAML two-class (global τ)	4.39 (±0.18)
Present work	WI/WD	GPDS-300	12	MAML two-class (user τ)	2.68 (±0.17)

With 12 reference signatures, the proposed system obtained 4.39% EER (global τ), compared to 3.74 for the WD system (Hafemann *et al.* (2017a)). However, the proposed system is more scalable, as a single model is stored for all users.

Figure 4.9 shows the performance on GPDS-300 as we vary the number of reference samples available for each user. As commonly observed in WD systems (e.g. Hafemann *et al.* (2017a)), the performance greatly improves as more reference samples are available for training: For the one-class formulation, performance with a single reference is 9.09% EER (global τ) and 5.81% EER (user τ). With 12 references, we obtain 4.70% EER (global τ) and 2.93% EER (user τ).

4.5.3 Transfer to other datasets

We now consider results on three other datasets: MCYT, CEDAR and the Brazilian PUC-PR. Table 4.5 shows the performance in two scenarios: (i) meta-learner trained only in GPDS, with



a) One-class

b) Two-class

Figure 4.9 Performance on GPDS-300 as we vary the number reference signatures available for each user. (a): one-class formulation; (b) two-class formulation.

Table 4.5 Transfer performance to the other datasets

Target Dataset	**Training Dataset**	**EER (global)**	**EER (user)**
MCYT	GPDS	15.48 (± 1.00)	12.54 (± 1.86)
	All	15.37 (± 0.97)	12.77 (± 0.46)
CEDAR	GPDS	15.98 (± 1.09)	12.07 (± 1.01)
	All	10.69 (± 1.76)	8.02 (± 1.22)
Brazilian	GPDS	8.05 (± 0.95)	4.83 (± 1.07)
	All	8.55 (±0.55)	6.7 (± 0.87)

its generalization to new operating conditions and (ii) meta-learned trained on all four datasets (using 10-fold cross validation, as described in section 4.4). While the method generalized well to unseen GPDS users, we see that the generalization performance to other datasets is much worse. Furthermore, we notice that even when training with a subset of users from all datasets, the performance does not improve for all datasets. A possible explanation is that the GPDS dataset is still much larger (10 times larger than the others) and dominates training. Overall, this suggests that the proposed method requires a large amount of data from the target application, and is sensitive to changes in operating conditions. Finally, tables 4.6, 4.7 and 4.8 compares de results with the state-of-the-art on MCYT, CEDAR and Brazilian PUC-PR, respectively.

Table 4.6 Comparison with the state-of-the-art in MCYT

Reference	**Type**	**# Samples**	**Features**	**EER**
Wen *et al.* (2009)	WD	5	RPF	15.02
Vargas *et al.* (2011)	WD	5	LBP	11.9
Vargas *et al.* (2011)	WD	10	LBP	7.08
Ooi *et al.* (2016)	WD	5	DRT + PCA	13.86
Ooi *et al.* (2016)	WD	10	DRT + PCA	9.87
Soleimani *et al.* (2016)	WD	5	HOG	13.44
Soleimani *et al.* (2016)	WD	10	HOG	9.86
Hafemann *et al.* (2017a)	WD	5	SigNet (user τ)	3.58 (± 0.54)
Hafemann *et al.* (2017a)	WD	10	SigNet (user τ)	2.87 (± 0.42)
Present Work	WI/WD	5	MAML one-class (global τ)	15.37(± 0.97)
Present Work	WI/WD	5	MAML one-class (user τ)	12.77(± 0.46)
Present Work	WI/WD	10	MAML one-class (global τ)	14.50(± 0.77)
Present Work	WI/WD	10	MAML one-class (user τ)	12.44(± 0.97)

Table 4.7 Comparison with the state-of-the-art in CEDAR

Reference	**Type**	**# Samples**	**Features**	**AER/EER**
Chen & Srihari (2006)	WD	16	Graph Matching	7.9
Kumar *et al.* (2010)	WI	1	morphology	11.81
Kumar *et al.* (2012)	WI	1	Surroundness	8.33
Bharathi & Shekar (2013)	WD	12	Chain code	7.84
Guerbai *et al.* (2015)	WD	4	Curvelet transform	8.7
Guerbai *et al.* (2015)	WD	8	Curvelet transform	7.83
Guerbai *et al.* (2015)	WD	12	Curvelet transform	5.6
Hafemann *et al.* (2017a)	WD	4	SigNet (SVM)	5.87 (± 0.73)
Hafemann *et al.* (2017a)	WD	8	SigNet (SVM)	5.03 (± 0.75)
Present Work	WI/WD	4	MAML one-class (global τ)	11.06(± 1.12)
Present Work	WI/WD	4	MAML one-class (user τ)	8.27(± 1.45)
Present Work	WI/WD	8	MAML one-class (global τ)	10.21(± 1.21)
Present Work	WI/WD	8	MAML one-class (user τ)	7.07(± 1.08)

4.6 Conclusion

In this paper we proposed to formulate Signature Verification as a meta-learning problem, where each user defines a task. This formulation enables directly optimizing for the objective (separating genuine signatures and forgeries) even when forgeries are not available for all users. The resulting system is scalable and yet adaptable for individual users: a single meta-classifier is learned and stored, and for the verification of a given signature, the classifier is

Table 4.8 Comparison with the state-of-the-art on the Brazilian PUC-PR dataset (errors in %)

Reference	**Type**	**#samples**	**Features**	**AER$_{\textbf{genuine + skilled}}$/EER**
Bertolini *et al.* (2010)	WI	15	Graphometric	8.32
Batista *et al.* (2012)	WD	30	Pixel density	10.5
Rivard *et al.* (2013)	WI	15	ESC + DPDF	11.08
Eskander *et al.* (2013)	WD	30	ESC + DPDF	10.67
Hafemann *et al.* (2017a)	WD	5	SigNet (user τ)	2.92 (± 0.44)
Hafemann *et al.* (2017a)	WD	15	SigNet (user τ)	2.07 (± 0.63)
Souza *et al.* (2018)	WI	5	SigNet (global τ)	5.95 (± 0.68)
Souza *et al.* (2018)	WI	5	SigNet (user τ)	2.58 (± 0.72)
Souza *et al.* (2018)	WI	15	SigNet (global τ)	5.13 (± 0.23)
Souza *et al.* (2018)	WI	15	SigNet (user τ)	1.70 (± 0.40)
Present Work	WI/WD	5	MAML one-class (global τ)	8.55 (± 0.55)
Present Work	WI/WD	5	MAML one-class (user τ)	6.70(± 0.87)
Present Work	WI/WD	15	MAML one-class (global τ)	6.93(± 0.73)
Present Work	WI/WD	15	MAML one-class (user τ)	5.74(± 0.84)

adapted to the claimed user and subsequently used for verification. The proposed method is also able to naturally incorporate new reference signatures for a user, and enable adapting the representation as more training data is available. The drawbacks of this solution are twofold: increased computational cost and worse transferability to new conditions. The method is $2K$ slower, when using K updates for the classification adaptation, although it allows the option to trade storage and computational cost - the adapted weights for a given user can be stored for faster classification. Our experiments transferring the meta-learner to other datasets show reduced performance, highlighting the need for better adaptation to new conditions, which will be explored in future work. Future work also includes considering a dynamic scenario, where the meta-classifier is updated as new training data is available.

CHAPTER 5

CHARACTERIZING AND EVALUATING ADVERSARIAL EXAMPLES FOR OFFLINE HANDWRITTEN SIGNATURE VERIFICATION

Luiz G. Hafemann[1], Robert Sabourin[1], Luiz S. Oliveira[2]

[1] Department of Automated Manufacturing Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Departement of Informatics, Federal University of Parana (UFPR),
Curitiba, PR, Brazil

Article Published in « IEEE Transactions on Information Forensics & Security » 2019.

Abstract

The phenomenon of Adversarial Examples is attracting increasing interest from the Machine Learning community, due to its significant impact to the security of Machine Learning systems. Adversarial examples are similar (from a perceptual notion of similarity) to samples from the data distribution, that “fool" a machine learning classifier. For computer vision applications, these are images with carefully crafted but almost imperceptible changes, that are misclassified. In this work, we characterize this phenomenon under an existing taxonomy of threats to biometric systems, in particular identifying new attacks for Offline Handwritten Signature Verification systems. We conducted an extensive set of experiments on four widely used datasets: MCYT-75, CEDAR, GPDS-160 and the Brazilian PUC-PR, considering both a CNN-based system and a system using a handcrafted feature extractor (CLBP). We found that attacks that aim to get a genuine signature rejected are easy to generate, even in a limited knowledge scenario, where the attacker does not have access to the trained classifier nor the signatures used for training. Attacks that get a forgery to be accepted are harder to produce, and often require a higher level of noise - in most cases, no longer “imperceptible" as previous findings in object recognition. We also evaluated the impact of two countermeasures on the success rate of the attacks and the amount of noise required for generating successful attacks.

5.1 Introduction

Biometric systems are extensively used to establish a person's identity in legal and administrative tasks (Jain *et al.* (2004)). They are commonly modeled as Pattern Recognition systems, in which biometric data from an individual is acquired (e.g. during an enrollment process), and stored as a “template" for future comparisons, or used to train a classifier that can discriminate if new samples belong to this user.

The reliability of these systems have security implications, and in the last decade these systems have been analyzed from an Adversarial Machine Learning perspective. From this viewpoint, we consider an active adversary, with its own goals (e.g. getting access to a system), knowledge (e.g. knowing the classifier parameters, or the learning algorithm) and capabilities (e.g. ability to manipulate the training data, or the inputs during test time). In particular, Ratha *et al.* (2001) and later Biggio *et al.* (2015) characterize the different components of a biometric system that can be attacked.

However, an emerging issue of “Adversarial Examples" pose new security concerns for such systems. This issue refers to adversarial input perturbations specially crafted to induce misclassifications. Szegedy *et al.* (2014) showed that very small perturbations on images (almost imperceptible) could be crafted to mislead a state-of-the-art CNN-based classifier. Moreover, attacks crafted for one model often transfer to other models, meaning that an attacker could train its own surrogate classifier to generate attacks, as long as it has access to data from the same data distribution. This issue has been analyzed in many recent papers (Goodfellow *et al.* (2015); Papernot *et al.* (2016); Tramèr *et al.* (2018); Carlini & Wagner (2017b,a)), but the theoretical reasons are not fully understood, and most defenses are weak (i.e. they fail if the attacker knows about the defense).

We evaluate this new threat for biometric systems, by characterizing the potential new attacks under a taxonomy of threats to such systems (Ratha *et al.* (2001), Biggio *et al.* (2015)). We consider particular attack scenarios to Offline Handwritten Signature Verification, identifying the attacker's goals, required knowledge and capabilities.

It is worth noting that attacking verification systems can present difficulties not present in classification problems. In particular, as new users join the system, they introduce a new *class*, not only unseen examples of existing classes. We present a refined version of the adversary's knowledge model that explicitly makes the distinction of whether access to data from a particular individual of interest is available to the attacker.

We conducted experiments on Writer-Dependent classifiers trained with a CNN-based representation (SigNet) and a handcrafted feature extractor (CLBP), considering four widely used Datasets: MCYT, CEDAR, GPDS-160 and the Brazilian PUC-PR. We defined a comprehensive set of experiments to evaluate such systems under different scenarios of the adversary's knowledge level and objectives, using four attack methods (gradient-based and gradient-free). Our main contributions are as follows:

- We characterize different attack scenarios for Offline Handwritten Signature Verification systems, focused on new threats introduced by Adversarial Examples.
- We identify that there is an asymmetry in the attacks, empirically showing that attacking genuine signatures (so that they are rejected) can be done with high success rate and a relatively low amount of noise, while attacking forgeries (so that they are accepted) is a much harder task.
- Our experiments with different scenarios of attacker knowledge show that attacks can be done even with Limited Knowledge, where the attacker has no access to the signatures used to train the classifiers, showing that this transferability affects both CNN-based systems and systems based on handcrafted features. We also identify that attack transferability is greatly reduced if the CNN is trained on a different subset of *users*, contrasting with previous findings that attacks transfer well if the CNN is trained on a different subset of samples from the same classes (Szegedy *et al.* (2014)).
- Lastly, we evaluate the impact of countermeasures and find that the Madry defense (Madry *et al.* (2018)) is effective in increasing the amount of noise necessary to make a sample

adversarial, even when it is applied only to the feature learning phase, and not on training the WD classifiers. Code for reproducing the experiments will be made publicly available at https://github.com/luizgh/adversarial_signatures.

The paper is organized as follows: in section 5.2 we introduce the main concepts of security in biometric systems; in section 5.3 we present the issue of adversarial examples and in section 5.4 we present particular attack scenarios for offline signature verification, and a refinement of the adversary's knowledge model. Section 5.5 describes the experimental protocol, and the results are discussed in section 5.6. Finally, our conclusions are listed in section 5.7.

5.2 Security in biometric systems

The security of machine learning systems have been widely studied in the past decade. Barreno *et al.* (2006, 2010) categorize attacks to such systems along three axes: (i) the influence of the attack, that can be causative (when training data is compromised) or exploratory (probing the learner to acquire information); (ii) the specificity of the attack: *targeted*, in which a particular point or a set of points is targeted or *indiscriminate*; and (iii) the security violation of the attack, that can seek an integrity violation (e.g. intrusion) or availability disruption (e.g. make the system unusable for legitimate users).

Biggio *et al.* (2014, 2015) further expands this analysis for biometric systems, incorporating a model of the adversary that includes its *goals*, *knowledge* of the target system, and *capabilities* of manipulating the input data or system components. The goals of an attacker are mainly divided in: 1) *Denial of service*: preventing real users from using the system; 2) *Intrusion*: impersonating another user; 3) *Privacy violation*: stealing private information from an user (such as the biometric templates). The *knowledge* of the adversary refers to the information of the target system that is available to the adversary, such as perfect knowledge (e.g. knowledge of the feature extractor, type of classifier and model parameters) or limited (partial) knowledge of the system. The *capabilities* of the adversary refer to what it can *change* in the target system, such as changing the training set (poisoning attack), or the inputs to the system at test time (evasion attack).

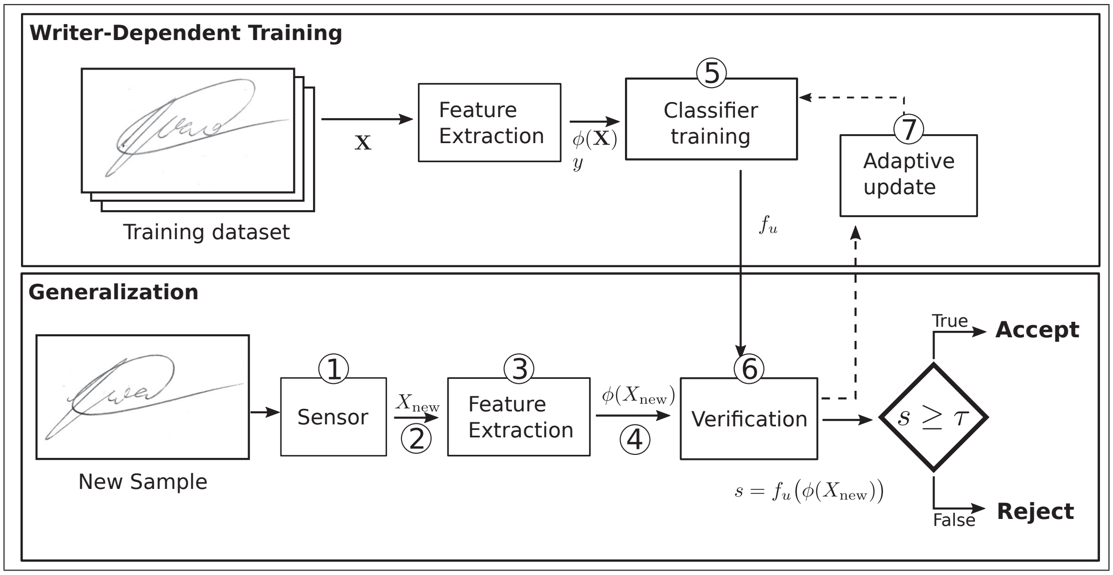


Figure 5.1 A typical writer-dependent signature verification system, with annotated points of attack. On the training phase, a classifier f_u is trained for each user. During operations, for a new sample X_{new} we obtain a feature vector $\phi(X_{\text{new}})$, and use the classifier f_u to accept or reject the signature. For adaptive systems, an update rule select signatures for classifier adaptation.

Modeling the knowledge of the adversary was formalized by Biggio & Roli (2018). Let $\mathscr{X}$ and $\mathscr{Y}$ be the feature and label spaces, respectively, and $\mathscr{D}$ be a dataset $\mathscr{D} = \{x_i, y_i\}_{i=1}^n$ of n training samples. Let f be a training algorithm (classifier), and w be a collection of its parameters and hyper-parameters. The knowledge of the attacker can be formalized as a set θ, containing the components of the system that are known to the attacker. Perfect-Knowledge (PK) attacks consider full knowledge of the system, that is, $\theta_{PK} = (\mathscr{D}, \mathscr{X}, f, w)$. We can also consider Limited Knowledge (LK) attacks, in which some of the information is not available to the adversary. As an example, if the adversary does not have access to the learned weights of the model, but has access to the training data, a surrogate classifier f can be trained (learning parameters $\hat{w}$) and used to generate the attack. Similarly, if the training data is not available, the adversary may be able to collect another training set from the same data distribution and use it to train the surrogate classifier. In this last scenario, the knowledge of the attacker would be represented as $\theta_{LK} = (\hat{\mathscr{D}}, \mathscr{X}, f, \hat{w})$. The hat symbol (ˆ) indicate limited knowledge of a component (such as getting a surrogate dataset from the same data distribution).

Biometric systems are composed of several components, such as the sensors capturing the biometric, and software to extract features, store templates and perform classification. Ratha *et al.* (2001) identified eight points of attack on biometric security systems, that were later grouped by Jain *et al.* (2008) and extended by Biggio *et al.* (2015) to include multi-modal systems and adaptive systems. The set of this attack points is considered the *attack surface* of the system. Figure 5.1 shows a typical User-Dependent classification system, with the main attack points. Below we discuss the main threats to the different points of attack.

The first point of attack (#1) in a biometric system is the user interface that collects the sample (e.g. a scanner capturing a document with a signature, or a mobile application taking a picture of a bank cheque). For many biometrics, attacks on this first point mainly consist of spoofing attacks, that normally use a fabricated fake biometric trait. Possible defenses for such attacks rely on liveness detection. On the signature verification task, simulated and traced forgeries can be considered attacks targeting this stage. A second set of attack points refer to attacks in the communication between different components of the system (#2, #4) (for example, intercepting and replacing the sensor input or the extracted features, that is input to the subsequent module). Defenses for such attacks involve encrypting the communication between the different modules. The software modules (#1, #3, #5, #6, #7) may present vulnerabilities in the code (such as buffer overflow) that can be exploited by a malicious user. The classifier training (#5) can be targeted for poisoning attacks (e.g. adding samples from another user in the training data for subsequent intrusion). For adaptive systems, the template update rule (#7) can be targeted to update the template database (e.g. for intrusion).

5.3 Adversarial Examples

Adversarial examples are samples similar to the true data distribution, but that fool a classifier. In computer vision, these are images $\tilde{X}$ that are visually similar to a “real" image X, but

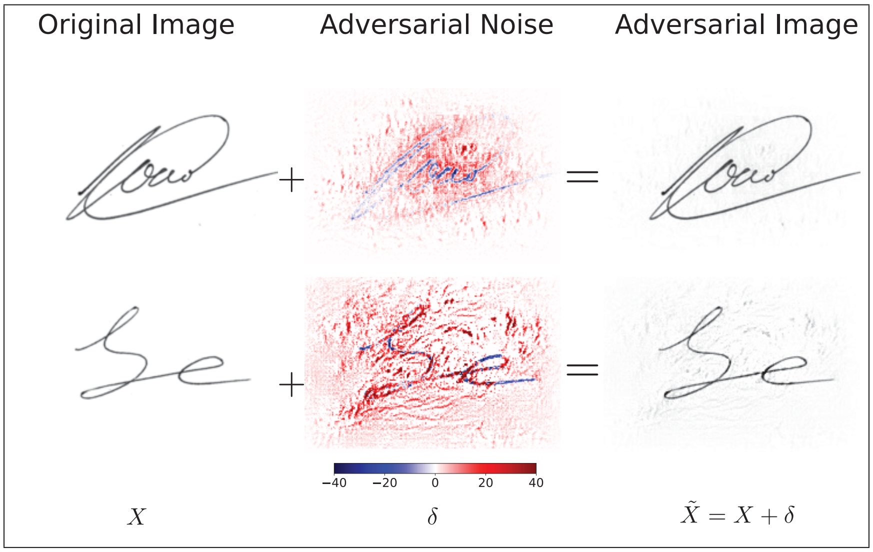


Figure 5.2 Illustration of adversarial examples. An adversarial noise δ is added to original images X, such that the resulting image $\tilde{X}$ is misclassified. **Top**: Type-I attack: a genuine signature from user u_1 (left) is attacked to be classified as a forgery (right). **Bottom**: Type-II attack. The original image (left) is from user u_2 (i.e. a random forgery for u_1), and is attacked to be classified as a genuine (right).

that fool a classifier (i.e. the classifier predicts an incorrect class for $\tilde{X}$: $\text{argmax}_y P(y|\tilde{X}) \neq \text{argmax}_y P(y|X)$).

Szegedy *et al.* (2014) showed that for deep neural networks, we can run an optimization procedure to produce a small change δ to an image, such that $\tilde{X} = X + \delta$ is an adversarial example, as illustrated in Figure 5.2. Perhaps more surprisingly, they also discovered that an attack that is created to fool one network also fools other networks (trained on different subsets of data), meaning that attacks can be created even without full knowledge of the classifier under attack. It was later shown that such attacks can be done in the physical world (Kurakin *et al.* (2017a)), where adversarial images printed on paper and later captured with a camera also fooled a classifier. Lastly, although some defense strategies have been proposed (Goodfellow *et al.* (2015); Papernot *et al.* (2016); Tramèr *et al.* (2018); Madry *et al.* (2018)), most solutions are not robust to strong iterative attacks. Even *detecting* that an input is adversarial is a hard task (Carlini & Wagner (2017a)).

Most of the recent research on this area concentrates on differentiable classifiers (usually Deep Learning models), creating attacks that use gradient information of the loss function with respect to the inputs. However, most feature extractors used in signature verification (such as LBP, HOG) are non-differentiable, and therefore attacks to systems using these features could not rely on gradient-based methods. Some methods proposed in the literature do not rely on gradient information, and could potentially be used for this task. Papernot *et al.* (2017) proposed *Substitute model training*, in which the attacker train a substitute (differentiable) model, and use it to generate the attack. Brendel *et al.* (2018a) proposed a *Decision-based attack*, that relies only on the decision (prediction) of the model under attack. Its strategy is the opposite of most attacks: given an image X and an image $\tilde{X}^0$ that is from another class, the algorithm iteratively refines $\tilde{X}^k$ to be closer to $\tilde{X}$ (e.g. in L_2 norm). The image $\tilde{X}^0$ can be a random image (e.g. sampled at random until it is from the desired class), or an actual image from a target class. Chen *et al.* (2017) proposed a *Zeroth order optimization* method, where the gradient is estimated numerically. Doing so naively is impractical (due to the dimensionality of the input), so the authors employ techniques to reduce the computational complexity of this estimation (block coordinate descent, attack-space dimension reduction, hierarchical attacks and importance sampling). With all these techniques, the attack has shown to scale to imagenet ($299 \times 299 \times 3$ pixels), producing an attack in 20 minutes. This method requires the function to be smooth and Lipshitz continuous. Ilyas *et al.* (2018) proposed using *Natural Evolution Strategy (NES) gradient estimate* - instead of using numerical methods to estimate the gradient (as above), the authors propose using Natural Evolution Strategies for the gradient estimate. These estimates are given by computing the loss function along random directions. The authors claim that this method require 1-2 orders of magnitude less computations of the loss function. Lastly, Ramanathan *et al.* (2017) explored using *Simulated annealing* for creating adversarial examples for a system based on HOG features with a linear SVM classifier. In each iteration, a small perturbation is applied to the image, and the distance of the new image to the SVM hyperplane is used as a condition to accept the new point. With this approach the authors were able to craft adversarial images with imperceptible noise that fooled the HOG+SVM classifier.

5.3.1 Attacks considered in this paper

In this paper, we consider two gradient-based attacks (that can be used when the classifiers are differentiable with respect to the input), and two gradient-free attacks, that can be used even if the features and/or classifiers are non-differentiable. In this paper we are mostly interested in feature extractors widely used for signature verification, and chose the LBP descriptor, which is used in several studies (Vargas *et al.* (2011); Hu & Chen (2013); Yılmaz & Yanıkoğlu (2016)). Since LBP is highly discontinuous (due to the thresholding using the center pixel's value), methods that estimate the gradient are less interesting: the gradient should be very discontinuous (0 almost everywhere), since for each pixel, the transition from one pattern to the other is a step function. For this reason we selected two methods that do not rely on estimating the gradients: the decision-based attack (Brendel *et al.* (2018a)) and the optimization using Simulated annealing. For the gradient-based attacks, we considered the Fast Gradient Method (FGM) (Goodfellow *et al.* (2015)) and the Carlini & Wagner attack (Carlini & Wagner (2017b)).

The decision-based attack (Brendel *et al.* (2018a)) is an iterative method: given an image X from class y_i, the objective is to find an image $\tilde{X}^k$ that is classified as a different class, and minimizes the distance $D(X, \tilde{X}^k)$ for some distance measure. It starts with a sample $\tilde{X}^0$ from a class $y \neq y_i$. In each step, first the sample is projected in a random direction that is orthogonal to $(\tilde{X}^{k-1} - X)$ (i.e. orthogonal to a straight line to the sample X), and then takes a step in the direction of X. If the point is still from a class different than y_i, it is accepted as the next point $\tilde{X}^k$, otherwise a new point is searched in another random direction. This method therefore only requires the decision of the model (which class a sample $\tilde{X}^k$ belongs to).

The annealing method uses the well known simulated annealing method as a gradient-free optimization method. Starting from the image X, we add a small perturbation obtaining $\tilde{X}^k$. If the resulting image is closer to the decision boundary of the SVM (i.e the score decreases/increases depending on the type of attack), it is accepted as the next point. Otherwise, with a probability inversely proportional to the current step, it is still accepted as the next point. In the work from Ramanathan *et al.* (2017), the authors consider as the objective function simply to reduce the distance to the SVM hyperplane, and stop optimization when the boundary is crossed. In our experiments, we found it necessary to include a penalty on the L_2 norm of the noise added to the image. This is further detailed in section 5.5.

The FGM attack is a one-step gradient-based attack. In this paper we consider the version of this attack focused on the L_2 norm:

$$\tilde{X} = X + \varepsilon \frac{\nabla J(x,y)}{\|\nabla J(x,y)\|_2} \tag{5.1}$$

Where X is the original image, $\nabla J(x,y)$ is the gradient of the loss function with respect to the input, and ε is a hyperparameter that controls the size of the update. The adversarial image is then clipped to the allowed range of the input (e.g. pixels between 0 and 255).

The Carlini & Wagner L_2 attack uses an iterative gradient attack, using a gradient descent method (the Adam optimizer). The objective to be minimized contains two terms: a term minimizing the noise δ and a term encouraging the model to misclassify the image:

$$\min_{w} \|\delta\|_2 + cf(X+\delta) \tag{5.2}$$

Where c trades-off between the two objectives, and is chosen with a binary search (the smallest c that still obtains a misclassified image). Instead of enforcing hard constraints on the adversarial image (to keep pixel values between 0 and 255), the authors propose a change of variable. First, they consider images normalized between 0 and 1. Then, to enforce that $X+\delta \in [0,1]$ they consider the following change of variable:

$$\delta_i = \frac{1}{2}(\tanh(w_i)+1) - x_i \tag{5.3}$$

Since $-1 \leq \tanh(w_i) \leq 1$, it follows that $0 \leq X_i + \delta_1 \leq 1$, satisfying the box constraints on the resulting image, but putting no constraints on the variable under optimization (w). As for the term that encourages the model to misclassify the image, they choose a term that seeks to increase the distance between the logits (pre-softmax activation) of the target class t and the class with maximum prediction (other than the target class):

$$f(X) = \max(\max_{i \neq t}(Z(X)_i) - Z(X)_t, -\kappa) \tag{5.4}$$

Where $Z(X)$ is the logit (pre-softmax activation) and κ is a constant that can be used to select how confident the model must be in the wrong class prediction. This loss function has no constraints, and can be solved by any gradient-based method.

5.3.2 Countermeasures

Under a paradigm of *Security by design*, systems should be designed to be secure from the ground up. In the case of Machine Learning, systems should be designed explicitly considering an adversary (Biggio & Roli (2018)). Dalvi *et al.* (2004) presented one of the first formulations of this problem, by considering a *game* between the classifier and the adversary. They propose a solution of this game for naive bayes classifiers, considering a classifier that performs as well as possible against an optimal adversary. This has some resemblance to recent approaches proposed for adversarial examples called *Adversarial Training* (Goodfellow *et al.* (2015); Tramèr *et al.* (2018)), in which the training procedure is augmented with adversarial samples, with the objective of increasing robustness of the systems.

In this work we are concerned with the new vulnerabilities introduced by adversarial changes in the input images that induce misclassifications in Signature Verification systems. In this setting, some defenses become harder to implement - for instance, Biggio *et al.* (2013) propose learning the support ($P(X)$) and incorporating this knowledge on the classifier training.

Learning this support when X is high dimensional (which is the case in signature images, eg. 150×200 pixels in this work) is a hard task, specially when just a few samples per user are available. The problem of working with large models and input dimensions is explored in recent work in adversarial examples for deep neural networks. For instance by *Adversarial training* (Goodfellow *et al.* (2015), Tramèr *et al.* (2018)); *defensive distilation* (retraining a network with knowledge extract from a previous training) (Papernot *et al.* (2016)); and techniques to add non-differentiable steps in the inference process (e.g. transforming the input with non-differentiable operations (Guo *et al.* (2018))). Most defenses, however, have been shown to fail when the attacker has knowledge of them. Tramèr *et al.* (2018) showed that Adversarial training is not robust to iterative attacks on a white-box (PK) scenario; Carlini & Wagner (2017b) showed that distillation is also not effective in this scenario. More recently, Athalye *et al.* (2018a); Athalye & Carlini (2018) showed that almost all defenses presented in recent ICLR and CVPR conferences can be bypassed. The only exception was the work of Madry *et al.* (2018), that propose a framework that provides guarantees against attacks with a maximum L_∞ norm. However, as noted in (Athalye & Carlini (2018)), this defense is hard to scale (the authors only reported results on the CIFAR-10 dataset, which consists of small images of 32×32 pixels), and that resistance to L_∞ attacks does not guarantee resistance to other scenarios (e.g. when the attacker is limited by a maximum L_2 norm of the noise). This problem therefore remains as an open research question.

In this paper we focus our attention in defenses for the CNN-based models, in particular by evaluating two defenses: Ensemble Adversarial Training (Tramèr *et al.* (2018)) and the Madry defense (Madry *et al.* (2018)). The first has demonstrated some robustness in Limited Knowledge scenarios, while the second is a proposed defense against perfect-knowledge attacks.

For the ensemble adversarial training, we first train M models on the task at hand. Then we train another model with the following loss function:

$$\tilde{J}(X, y, \theta) = \alpha J(X, y, \theta) + (1 - \alpha) J(\tilde{X}, y, \theta) \tag{5.5}$$

Where $J(X, y, \theta)$ is the cross-entropy loss function of a sample X with true label y, and $\tilde{X}$ is an adversarial sample generated using FGM (equation 5.1) either using the model being trained, or one of the M previously trained models.

The Madry defense involves a saddle point optimization problem, in which we optimize for the worst case:

$$\begin{aligned} &\min_{\theta} p(\theta) \\ \text{where} \quad &p(\theta) = \mathbb{E}_{(x,y)\sim\mathscr{D}}\big[\max_{\delta\in\mathscr{S}} J(X+\delta, y, \theta)\big] \end{aligned} \tag{5.6}$$

Where $\mathscr{S}$ defines a feasible region of the attack (i.e. the attacker capability). For instance, to add robustness against attacks that minimize the L_2 norm of the attacks, we train the classifier with an adversary constrained to $\mathscr{S} = \{\delta : \|\delta\|_2 < \epsilon\}$, for a given maximum perturbation ϵ.

Lastly, we also consider a countermeasure using background removal. Handwritten data has an important difference compared to other vision tasks, such as object recognition, where we have a clear and simple separation of background and foreground. This is an important distinction because adversarial samples usually involve adding a crafted “noise" all around the image. To this end, we investigate the impact (on the attack success rate) of removing the background after the adversarial samples are generated.

5.4 Attack scenarios for Offline Handwritten Signature Verification

We now consider the possible attacks to biometric systems based on adversarial examples $\tilde{X}$. In particular, we identify possible attack points, and provide specific scenarios for Offline Handwritten Signature Verification.

The attacks using adversarial examples involve changing the inputs to the classifier, and therefore we identify two potential areas of vulnerability: at the sensor level, or the template storage/update level. The most prominent aspect of adversarial examples is that they fool a machine learning system without fooling humans (i.e. $\tilde{X}$ being visually similar to X). This is an important difference to spoofing attacks (that also target the sensor level), since these fake biometric traits, such as a “gummy finger”, are clearly identified as different from a real finger by a human. We identify the following new attacks on signature verification systems, along with possible goals of an attacker:

1. **Attacks on the data capture** (targets point #1). In this case the adversarial image is crafted before the image is collected for the system. That is, an adversary can craft adversarial images $\tilde{X}$, and present them to the system, for instance using a banking application that allows a customer to use a picture of a cheque to cash it; or by printing adversarial noise on a physical signature. We identify two types of attack:

    - **Type-I attack** (false rejection): Present a genuine signature that fools the system as being a forgery. This can be used for denial of service (preventing genuine users to accessing a system). We can also make a parallel to disguised signatures, where the user signs a document with the intent of later denying it (for example, the receiver of a check accepts it, but fails to cash it as the system classifies it is a forgery).
    - **Type-II attack** (false acceptance): Present a random forgery (i.e. a genuine signature from user y_i) that fools the system as being genuine for user y_j ($j \neq i$). At the same time, to a person, this sample can show no signs of being forgery (if it is not compared to a reference), since it is a genuine signature. The attacker can also use a skilled forgery as “starting point", creating noise to increase the likelihood of the forgery being accepted.

2. **Attacks on the templates** (targets point #5): If original images are stored as part of the system (e.g. for classifier re-training, or manual verification in case of system failure/rejection of a sample), the templates can be changed to still look like genuine signatures for human operators, but in a way that accept signatures from a different person as genuine.

3. **Attack on template update** (targets point #7): For adaptive systems, the attacker can craft changes on the user’s signature, so that adversarial templates are added to the gallery, to enable an intrusion later using a signature from another person. Similarly to the point above, the templates would appear as genuine to a person.

The attacks above require different capabilities from the part of the attacker. The first attack only affects the system at test time (evasion attack), and in many practical scenarios would require the creation of a *physical* attack, that is, the creation of an adversary signature in a piece of paper, for instance by printing adversarial noise on top of a handwritten signature. The second attack is a poisoning attack, that does not require a physical sample, as it impacts the stored templates of an user. However, it requires the capability of the attacker to update the template database, and can be categorized as an *insider attack* as per the terminology used by Biggio *et al.* (2015). Note that this attack differ from simply adding another user's biometric to the templates, since a manual inspection of the templates would not reveal that the templates have been tampered with. The third attack can also be seen as a poisoning attack, affecting adaptive systems, that automatically add new samples to the set of user templates.

As for the knowledge required from the adversary, we can consider different scenarios, ranging from full knowledge of the system, to scenarios where only limited information is available to the attacker.

5.4.1 Refining the adversary's knowledge model

For biometric *verification* tasks, we identify an important refinement of the adversary's knowledge model. We argued in section 5.2 that an adversary that does not have access to the training set can collect its own data $\hat{\mathscr{D}}$ from the same data distribution, and train a surrogate classifier. For verification systems, each new user to the system effectively introduces a new class, and therefore it is important to make a distinction of accessing data for a particular individual of interest, and a “background class", that are negative examples for a given user (e.g. signatures from other users). We refer therefore to two data components: $\mathscr{D}_b$ - biometric data from the background class (i.e. not for the individual under attack), and $\mathscr{D}_u$ - biometric data from the targeted individual. This allows the definition of limited knowledge scenarios where the biometric sample of the user can be collected, or for scenarios where the adversary can only collect samples from a other users.

In our experiments, we consider three attack scenarios:

- **Perfect Knowledge** scenario: the attacker has knowledge of all components of the system: $\theta_{PK} = (\mathscr{D}_b, \mathscr{D}_u, \mathscr{X}, f, w)$. This scenario serves as a tool to analyze the worst-case scenario (from the system's defense perspective).
- **Limited Knowledge #1**: we consider a scenario where the attacker does not have access to the dataset used for training the classifiers, but has access to all other components. We consider that the attacker is able to collect signatures from some users ($\hat{\mathscr{D}}_b$, that are from different users from those used to train the system), and some signatures from the user of interest, that were not used for training the system: $\hat{\mathscr{D}}_u$. In this case, $\theta_{LK1} = (\hat{\mathscr{D}}_b, \hat{\mathscr{D}}_u, \mathscr{X}, f, \hat{w})$.
- **Limited Knowledge #2**: similarly to the above, but we consider a scenario where the attacker does not have full access to the feature extraction function (that induces the space $\mathscr{X}$). In particular, we consider a scenario where the attacker does not have access to the CNN model that was used to extract the features, but trains its own CNN (with identical training procedure and architecture) on a different set of users. In this case, $\theta_{LK2} = (\hat{\mathscr{D}}_b, \hat{\mathscr{D}}_u, \hat{\mathscr{X}}, f, \hat{w})$.

5.5 Experimental Protocol

We conducted experiments using the datasets MCYT-75 (Ortega-Garcia *et al.* (2003)) (with 75 users), CEDAR (Kalera *et al.* (2004)) (55 users), GPDS-160 (Vargas *et al.* (2007)) (160 users) and the Brazilian PUC-PR (Freitas *et al.* (2000)) (60 users).

In order to simulate the different attack scenarios we split the dataset into two parts of disjoint users, as illustrated in Figure 5.3. The set $\mathscr{D}$ refers to the users “enrolled in the system", that will be under attack. This dataset is divided in training (user signatures $\mathscr{D}_u$) and testing $\mathscr{T}$. For the limited knowledge scenarios, we consider a set $\hat{\mathscr{D}}_b$ that contains signatures from other users (not those being attacked), simulating the scenario of an attacker that acquired his own signature dataset in order to generate the attacks. We also consider that the attacker has access

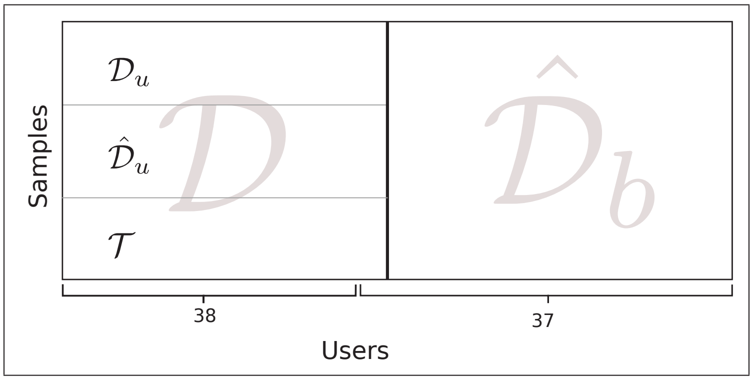


Figure 5.3 Dataset separation for the MCYT dataset. The set $\mathscr{D}_u$ is used for training the classifiers under attack, and the sets $\hat{\mathscr{D}}_b$ and $\hat{\mathscr{D}}_u$ are used by the attacker to train surrogate classifiers.

to some signatures from the user, $\hat{\mathscr{D}}_u$, that were not used for training the system (i.e. disjoint from $\mathscr{D}_u$ and $\mathscr{T}$).

The images were pre-processed in a similar way to (Hafemann *et al.* (2017a)): Signatures were first centered in a blank canvas using their center of mass. We then resize the images to 150×220 pixels and invert the image such that the background pixels are zero-valued. Lastly, we run the OTSU algorithm to identify the optimal threshold that separates background and foreground. We set the pixels with intensity smaller than the threshold to intensity 0, leaving the remaining pixels in grayscale.

We consider Writer-Dependent classifiers, training an SVM (linear or with the RBF kernel) for each user. As feature extraction $\phi(X)$, we consider: (i) a CNN-based learned representation: SigNet (Hafemann *et al.* (2017a)), and (ii) the CLBP operator (Completed Local Binary Patterns) (Guo *et al.* (2010)). We train the SVMs with 5 genuine signatures from the user as positive samples, and 5 signatures from each other user as negative.

For the scenario LK2, we consider two CNN models with the same architecture and training procedures, but trained on a disjoint set of users. The CNN used by the model under attack was trained on GPDS users 350-614, and the CNN used by the surrogate models (by the attacker) were trained with users 615-881. Training procedure followed the same as SigNet (Hafemann *et al.* (2017a)). For the Ensemble Adversarial Learning evaluation, we first trained two models with different architectures (slight variations from SigNet, as described in the Supplemental Material) and then trained a model with the SigNet architecture and the loss function defined in equation 5.5, with $\varepsilon = 5$. For the Madry defense, we also used the same architecture, and trained with $\mathscr{S} = \{\delta : \|\delta\|_2 < \varepsilon\}$ with $\varepsilon = 2$. We tried using larger values for ε and obtained worse classification performance during the CNN training, so these values represent a tradeoff between robustness and accuracy. In both cases, we trained the network with users 350-614, to enable evaluating the scenario LK2. In this scenario, we consider an attacker that trained a regular CNN (no adversarial training), with users 615-881.

After training the classifiers for each user, the SVMs implement the following decision functions:

$$s_{\text{Linear}} = \mathbf{w}^{\mathsf{T}} \phi(X) + b \tag{5.7}$$

$$s_{\text{RBF}} = \sum_{i \in \mathscr{S}} \alpha_i \exp(-\gamma \|\phi(X) - X_i\|) + b \tag{5.8}$$

Where s_{Linear} and s_{RBF} are the scores for the linear SVM and the SVM with the RBF kernel, respectively; w are the weights learned by the linear SVM, $\mathscr{S}$ is the set of support vectors, α_i and X_i are the coefficients and support vectors, γ is a hyperparameter for the RBF kernel and b is the bias. We can easily see that both functions are differentiable with respect to $\phi(X)$ (Biggio *et al.* (2013)). For the classifier using a CNN-based model to extract the features, we can calculate the gradients of the scores w.r.t the inputs X, and apply gradient-based methods to generate the attacks. For non-differentiable feature extractors, we consider only the two gradient-free methods described in section 5.3.1. When reporting the scores in Figures 5.2, 5.4 and 5.5, we consider a normalized loss as follows: $\tilde{s}(X) = s(X) - \tau$, where τ is the global threshold. This makes it easy to identify if a signature would be classified as genuine or as a forgery ($\tilde{s}(X) \geq 0$ indicates the prediction of X being a genuine signature).

For the classifiers using LBP, we consider the the operator CLBP_S/M/C (Guo *et al.* (2010)) (3D histogram of CLBP S, M and C), with the following parameters: $R = 1$ (radius of 1 pixel), $P = 8$ (eight neighbors) and rotation invariant uniform patterns (“riu2"). The feature vector has a total of 200 dimensions.

To simplify the generation of the attacks we considered a global threshold for the classifications, that obtained the Equal Error Rate on the set $\mathscr{D}$ (without any attacks).

After the classifiers are trained, we generate attacks using the four methods described in section 5.3.1. We used the FGM method with $\varepsilon = 1000$, and the Carlini & Wagner attack with $\kappa = 1$. For the Decision-based attack, we considered the implementation from the authors[1], running the attack for a maximum of 1000 iterations. For the Simulated Annealing method, we considered an open implementation of simulated annealing[2]. In each iteration, we change the state by adding gaussian noise ε ($\varepsilon \sim \mathscr{N}(0, \sigma I)$, with $\sigma = 2$), and clipping the image between 0 and 255. We consider the energy to be a mixture of the SVM score and the L_2 norm of the adversarial noise δ: $E = s(X) + \lambda \left\|\delta\right\|_2$, with $\lambda = 0.001$ being a trade-off between changing the SVM score, and not deviating too far from the original image. We used an initial temperature $T_{\text{max}} = 1$ and final temperature $T_{\text{min}} = 0.001$. These values were chosen such that around 95% of the steps that would increase the energy are still accepted in the start of the procedure, and less than 5% were accepted in the end. We ran this procedure with at most 1000 steps, with early stopping (we stop optimization if the image is adversarial).

The experiments consisted in Type-I attacks (attempting to have a genuine signature rejected by the system) and Type-II attacks (attempting to have a forgery accepted by the system). For each user, we selected one genuine signature, one random forgery and one skilled forgery, such that all four classifiers correctly classified these samples. We then used the different attack methods to generate adversarial samples, and measured the attack success rate (number of misclassified images after the attack), and the average RMSE (root mean square error) of the

[1] https://github.com/bethgelab/foolbox

[2] https://github.com/perrygeo/simanneal

adversarial noise on successful attacks. It is worth noting that we consider pixel values in the range $[0,255]$, so the RMSE of the adversarial noise is also constrained in the same range. To summarize the experiments, we considered:

- Datasets: MCYT-75, CEDAR, GPDS-160, Brazilian PUC-PR
- Feature extractor: CLBP, SigNet
- SVM type: Linear, RBF
- Attack method: Decision-based, Simulated Annealing, FGM, Carlini
- Attacker's goal: Type-I (attacking Genuine signatures) and Type-II (attacking Random and Skilled forgeries),
- Attacker's knowledge: Perfect Knowledge, Limited Knowledge LK1 and LK2
- Defense: No defense, Ens. Adv. training, Madry

It is worth mentioning that in this work we did not consider the discretization of the generated adversarial images. We worked with images in float format, instead of discretized into integers between 0 and 255. This is discussed in section 5.6.6.

5.6 Results and discussion

Before presenting the results of the attacks, we first validate the performance of the WD classifiers on the four datasets. Table 5.1 shows the EER obtained by using different features/classifiers, when trained with 5 reference signatures per user, with the protocol defined in section 5.5. We observe a large variance in the results across different datasets, which suggests different degrees of difficulty on separating genuine signatures and forgeries in them. We also observe a large difference of performance between systems trained with the SigNet and CLBP features. In order to have a fair analysis of the adversarial examples against each classifier/feature extractor, we select the same set of images for the attacks on all classifiers, ensuring that the original

Table 5.1 Results of WD classifiers using different feature sets (EER considering skilled forgeries)

Dataset	Features	EER global-τ		EER user-τ	
		Linear	RBF	Linear	RBF
MCYT-75	SigNet	7.12	7.03	7.39	5.68
	CLBP	26.49	27.03	27.21	26.85
CEDAR	SigNet	12.03	11.82	6.01	4.52
	CLBP	28.01	21.36	23.95	16.39
GPDS	Signet	7.70	6.80	4.62	4.14
	CLBP	26.74	24.58	21.79	22.37
Brazilian PUC-PR	SigNet	6.78	5.22	3.61	2.67
	CLBP	26.83	19.61	24.61	16.83

images (before the attack) were correctly classified by them. Although the classifier performance varies across different datasets, the results for the adversarial attacks showed consistent trends across them. In this paper we report the consolidated results over the four datasets, and for completeness we include the results on individual datasets in the Supplementary Material.

5.6.1 Perfect Knowledge

We consider first a scenario of Perfect Knowledge, in which the adversary has full knowledge of all components of the system: $\theta_{PK} = (\mathscr{D}_b, \mathscr{D}_u, \mathscr{X}, f, w)$. The attacker can run his own copy of the system, and use one of the proposed attacks to generate adversarial images.

Table 5.2 Success rate of Type-I attacks (% of attacks that transformed a genuine signature in a forgery)

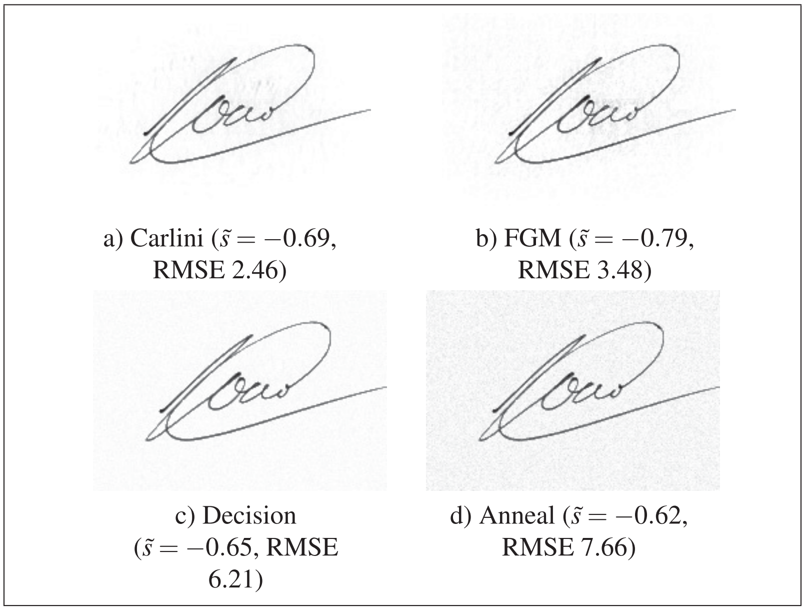
		Attack Type			
Feature	Classifier	FGM	Carlini	Anneal	Decision
CLBP	Linear	-	-	63.16	80.70
CLBP	RBF	-	-	100.00	100.00
SigNet	Linear	99.42	100.00	98.83	100.00
SigNet	RBF	98.25	100.00	98.83	100.00

For Type-I attacks, given a genuine sample X_g, the objective is to obtain an adversarial $\tilde{X} = X_g + \delta$ that is classified as a forgery. Table 5.2 shows the success rate of attacks in this scenario

Table 5.3 Distortion (RMSE of the adversarial noise) for successful Type-I attacks

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	0.40	1.57
CLBP	RBF	-	-	0.36	10^{-9}
SigNet	Linear	4.04	1.35	5.69	3.27
SigNet	RBF	4.06	1.40	5.17	3.02

(i.e. the percentage of attacks that found an adversary image), by attack type and classifier type. We see a high success rate for most attacks. Table 5.3 shows the average RMSE (root mean squared error) of the adversarial noise δ. We notice that the required amount of noise varies significantly with different classifiers and attack types. In general, gradient-based attacks find adversarial images with much less noise on the differentiable models. For the models with handcrafted features (where we do not have gradients), we noticed that even smaller changes on the image were enough to induce a misclassification. Figures 5.4 and 5.5 present examples of this type of attack.



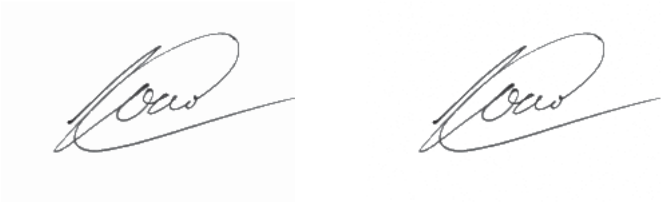
a) Carlini ($\tilde{s} = -0.69$, RMSE 2.46)

b) FGM ($\tilde{s} = -0.79$, RMSE 3.48)

c) Decision ($\tilde{s} = -0.65$, RMSE 6.21)

d) Anneal ($\tilde{s} = -0.62$, RMSE 7.66)

Figure 5.4 Example of Type-I attacks on the SVM model with RBF kernel and SigNet features. The original image is correctly classified as genuine by this model ($\tilde{s} = 0.13$).



a) Decision ($\tilde{s} = -0.39$, RMSE 0.99)

b) Anneal ($\tilde{s} = -0.27$, RMSE 0.21)

Figure 5.5 Example of Type-I attacks on the SVM model with Linear kernel and CLBP features. The original image is correctly classified as genuine by this model ($\tilde{s} = 1.60$).

Table 5.4 Success rate of Type-II attacks (% of attacks that transformed a forgery in a genuine signature)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	37.36	45.98
		skilled	-	-	38.73	46.24
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	1.15	96.55	0.00	0.00
		skilled	28.90	99.42	2.31	3.47
SigNet	RBF	random	0.57	94.83	0.00	0.00
		skilled	19.65	100.00	1.73	1.73

We now consider Type-II attacks, in which we want to modify a forgery X_f, by creating an adversary $\tilde{X} = X_f + \delta$ that is classified as a genuine signature. Table 5.4 shows the success rate of the different methods, and table 5.5 shows the level of noise required in the successful attacks.

Table 5.5 Distortion (RMSE of the adversarial noise) for successful Type-II attacks

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	0.39	1.17
		skilled	-	-	0.42	1.08
SigNet	Linear	random	4.11	6.07	-	-
		skilled	4.20	3.19	3.61	1.34
SigNet	RBF	random	4.70	6.55	-	-
		skilled	4.08	3.62	3.17	1.18

The results show that this attack is much harder to obtain compared to the Type-I attacks. For the models trained with CLBP features, we observed that the linear classifier could be attacked half of the time, while we could not generate any attack using the two gradient-free methods for the non-linear model. For the CNN-based models, a strong gradient-based method (Carlini) worked for almost all samples, while the gradient-free methods did not work in most cases - we only observed some success when using skilled forgeries as the starting point. Comparing tables 5.3 and 5.5, we observe that for the CLBP-based classifiers, a similar amount of noise was required to create successful attacks. For the CNN-based methods, when starting from a random forgery a large amount of noise was required to create successful attacks, while when starting from a skilled forgery a lower amount of noise was required. We reiterate that the skilled forgeries selected for attack were correctly classified by the model (i.e. classified as forgeries), while in successful attacks the adversarial image is classified as a genuine.

It is worth noting that in the experiments with the strong gradient-based attack, we observed a much larger amount of noise required for misclassification compared to previous results reports on object recognition. For instance, in the classification task on ImageNet, successful attacks (using the same Carlini & Wagner method) are reported with much lower noise (RMSE of 0.004 for 100% success of targeted attacks on ImageNet (Carlini & Wagner (2017b))). While for object recognition the adversarial images are often perceptually identical to the original, for signatures we noted some distinguishable noise, specially on the Type-II attacks, as can be seen in figure 5.2 (where the Type-II attack has RMSE of 10.34).

5.6.2 Limited Knowledge #1

We now consider a limited knowledge scenario, where the attacker does not have access to the signatures used for training the system, but does obtain a surrogate dataset: $\theta_{LK1} = (\hat{\mathscr{D}}_b, \hat{\mathscr{D}}_u, \mathscr{X}, f, \hat{w})$. In this case, the signatures from the *background set* (used as negative samples during training) were from a different set of users than those used to train the system. We also consider that the attacker collected some signatures from the user of interest $\hat{\mathscr{D}}_u$, but that are also different from those used to train the system. This scenario also assumes that the attacker knows the feature extractor (i.e. full knowledge of the feature extractor, including all parameters), and the learning function (the WD classifier type, but not the learned parameters). In this scenario, the attacker uses the surrogate data to train their own version of the WD classifiers, and uses this classifier to generate the attacks. We then evaluate the success rate of these attacks on the actual system.

Table 5.6 Success rate of Type-I attacks (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	42.69	43.86
CLBP	RBF	-	-	82.46	82.46
SigNet	Linear	97.08	80.12	50.88	40.35
SigNet	RBF	97.66	91.81	54.39	47.95

Table 5.6 shows the success rate of the Type-I attacks. We observe a lower success rate compared to the perfect knowledge scenario, but still we find a high success rate against most models. This suggests that indeed there is a transferability of attacks across models (as observed before in CNNs (Szegedy *et al.* (2014))), and that this transferability also impacts systems trained with handcrafted features.

Table 5.7 Success rate of Type-II attacks (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	24.71	28.74
		skilled	-	-	21.97	26.01
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	46.55	0.00	0.00
		skilled	22.54	71.68	1.73	0.00
SigNet	RBF	random	0.00	74.71	0.00	0.00
		skilled	19.08	83.24	0.58	0.00

Table 5.7 shows the success rate for Type-II attacks in a limited knowledge scenario. Again we see a drop in performance compared to the perfect knowledge scenario, but still the attacks that worked in the PK scenario also worked (to some extent) in the limited knowledge scenario.

5.6.3 Limited Knowledge #2

We now consider a limited knowledge scenario similar to the above, but where the attacker also does not have access to the CNN used to extract the features. In this case, we consider that the attacker trains a surrogate CNN using a disjoint set of users, which induces a new feature space $\hat{\mathscr{X}}$. We consider therefore $\theta_{LK2} = (\hat{\mathscr{D}}_b, \hat{\mathscr{D}}_u, \hat{\mathscr{X}}, f, \hat{w})$.

Table 5.8 Success rate of Type-I attacks (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge #2)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	60.34	6.90	48.85	19.54
SigNet	RBF	64.37	9.20	51.15	18.97

Table 5.9 Success rate of Type-II attacks (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge #2)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	random	0.00	0.00	0.00	0.00
		skilled	2.30	2.30	0.57	0.57
SigNet	RBF	random	0.00	0.00	0.00	0.00
		skilled	1.72	1.72	1.15	0.00

Tables 5.8 5.9 show the success rate of the Type-I and Type-II attacks, respectively. We observe much lower success rates, especially for Type-II attacks, where no attacks were successful when starting from a random forgery, and starting with a skilled forgery the success was as low as 1-2%. For the Type-I attacks, we notice lower success rates compared to the PK and LK1 scenarios. Overall, these results suggest that transferability of the attacks is much worse when the models are trained with a different subset of users, that is, when the attacker does not have access to signatures from the same users that were used to train the CNN model. This contrasts

with findings in object classification, where attacks trained on a subset of data transfer well to a model trained with another subset of data (different samples from the same classes) (Szegedy *et al.* (2014)). Also, it is worth noting that the strong Carlini attack (that achieves close to 100% success in the Perfect Knowledge scenario) drops in performance in the LK scenarios, confirming previous findings that such iterative attacks transfer less than single-step attacks such as FGM (Kurakin *et al.* (2017b)).

5.6.4 Evaluating countermeasures

Table 5.10 Success rate of Type-I attacks considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
Baseline	Linear	100.00	95.40	60.34	100.00	78.16	6.90
	RBF	100.00	97.70	64.37	100.00	85.63	9.20
Ens. Adv	Linear	91.38	85.63	45.40	100.00	79.89	4.60
	RBF	90.23	83.91	46.55	100.00	90.23	5.75
Madry	Linear	91.38	83.33	22.99	100.00	74.71	1.72
	RBF	89.08	86.21	21.84	100.00	89.08	0.57

Table 5.11 Distortion (RMSE of the adversarial noise) for Type-I attacks, considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
Baseline	Linear	4.17	4.19	4.30	1.31	1.33	1.37
	RBF	4.20	4.21	4.30	1.40	1.38	1.55
Ens. Adv.	SigNet & Linear	4.37	4.30	4.20	1.35	1.43	1.85
	RBF	4.36	4.32	4.20	1.44	1.43	1.63
Madry	SigNet & Linear	4.76	4.72	4.26	3.19	3.28	1.59
	RBF	4.77	4.74	4.27	3.48	3.52	2.19

We now consider the impact of two counter-measures for the CNN-based systems: Ensemble Adversarial Learning (Tramèr *et al.* (2018)) and the Madry defense (Madry *et al.* (2018)). Tables 5.10 and 5.11 show the success rate and distortion (RMSE) for Type-I attacks. We consider the three Knowledge scenarios discussed in section 5.4.1 (Perfect Knowledge and two Limited Knowledge scenarios), and the two gradient-based attacks (FGM and Carlini). We notice that both defenses provide some robustness against the FGM attack in all knowledge scenarios. Considering the Carlini attack, we see that in a Perfect-Knowledge scenario the attack was always successful, but Table 5.11 shows that the Madry defense greatly increase the amount of noise required to generate adversarial examples, going from a RMSE of 1.4 to around 3.3.

Table 5.12 Success rate of Type-II attacks considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			**Carlini**		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
Baseline	Linear	random	2.87	1.15	0.00	98.85	42.53	0.00
		skilled	40.80	29.31	2.30	100.00	66.67	2.30
	RBF	random	1.72	1.15	0.00	95.98	68.39	0.00
		skilled	34.48	27.59	1.72	100.00	83.91	1.72
Ens. Adv.	Linear	random	1.72	0.57	0.00	93.10	41.38	0.00
		skilled	29.31	14.94	1.15	100.00	64.37	3.45
	RBF	random	2.30	0.00	0.00	93.10	69.54	0.00
		skilled	22.99	17.24	1.15	100.00	83.91	2.30
Madry	Linear	random	1.72	0.57	0.00	98.28	45.98	0.00
		skilled	48.85	38.51	8.05	100.00	73.56	3.45
	RBF	random	2.30	0.57	0.00	97.70	75.86	0.00
		skilled	45.98	37.36	6.32	100.00	87.93	2.87

Tables 5.12 and 5.13 shows the results on Type-II attacks. In these experiments, we again observe that the Carlini attack finds attacks most of the time, and that the Madry defense showed to be effective in increasing the amount of noise required to obtain an adversarial example (e.g. the average RMSE is increased from 5.98 to 10.81 when starting with a random forgery, comparing the baseline and the Madry defense). It is worth noting that the RMSE values only consider the successful attacks, and therefore the results on the Limited Knowledge scenarios (where the success rate is very low) are likely skewed by a few forgeries that were already close to the decision boundary.

Table 5.13 Distortion (RMSE of the adversarial noise) for Type-II attacks, considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			Carlini		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
Baseline	Linear	random	3.97	3.75	-	5.98	5.71	-
		skilled	4.21	4.14	4.24	2.99	2.71	2.43
	RBF	random	3.84	3.83	-	6.27	6.03	-
		skilled	4.11	4.06	4.64	3.32	3.20	1.77
Ens. Adv.	Linear	random	4.51	4.82	-	8.61	8.83	-
		skilled	4.53	4.58	4.09	4.71	4.34	1.43
	RBF	random	4.40	-	-	9.45	9.31	-
		skilled	4.59	4.58	4.07	5.43	4.82	2.14
Madry	Linear	random	4.74	5.38	-	10.81	10.97	-
		skilled	4.90	4.93	4.15	6.18	5.87	1.94
	RBF	random	4.62	5.28	-	11.49	11.46	-
		skilled	4.91	4.88	4.16	7.00	6.71	2.40

5.6.5 Impact of background removal

Table 5.14 Success of Type-I attacks in a PK scenario, with no pre-processing and with OTSU pre-processing

		Attack Type and Preprocessing							
		FGM		Carlini		Anneal		Decision	
Feature	**Classifier**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	-	-	-	-	63.16	9.36	80.70	3.51
	RBF	-	-	-	-	100.00	0.58	100.00	0.00
SigNet baseline	Linear	100.00	88.51	100.00	18.39	96.55	0.57	100.00	1.72
	RBF	100.00	86.21	100.00	22.41	98.28	0.00	98.85	0.57
SigNet Ens. Adv.	Linear	91.38	67.24	100.00	2.87	97.70	0.00	100.00	0.00
	RBF	90.23	65.52	100.00	1.72	98.28	0.00	100.00	0.00
SigNet Madry	Linear	91.38	87.93	100.00	77.01	87.93	0.00	99.43	6.90
	RBF	89.08	87.36	100.00	75.86	88.51	0.00	100.00	5.75

We now investigate the impact of simple noise-reduction techniques on the success of the attacks. Starting from the adversarial examples found in the experiments from the previous section, we applied the OTSU algorithm to remove noise with intensity lower than a threshold (as described in section 5.5). We then evaluate if the resulting image remains adversarial.

Table 5.15 Success of Type-II attacks in a PK scenario, with no pre-processing and with OTSU pre-processing

			Attack Type and Preprocessing							
			FGM		**Carlini**		**Anneal**		**Decision**	
Feature	**Classifier**	**Forgery Type**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	random	-	-	-	-	37.36	0.57	45.98	0.00
		skilled	-	-	-	-	38.73	1.73	46.24	1.16
	RBF	random	-	-	-	-	0.00	0.00	0.00	0.00
		skilled	-	-	-	-	0.00	0.00	0.00	0.00
SigNet Baseline	Linear	random	2.87	0.57	98.85	0.00	0.00	0.00	0.00	0.00
		skilled	40.80	31.03	100.00	12.07	1.15	0.00	1.15	0.00
	RBF	random	1.72	0.57	95.98	0.00	0.00	0.00	0.00	0.00
		skilled	34.48	24.14	100.00	14.37	1.72	0.00	1.72	0.00
SigNet Ens Adv.	Linear	random	1.72	1.15	93.10	7.47	0.00	0.00	1.72	0.00
		skilled	29.31	22.99	100.00	21.84	0.57	0.00	2.87	0.57
	RBF	random	2.30	1.15	93.10	12.64	0.00	0.00	0.00	0.00
		skilled	22.99	14.94	100.00	27.59	0.57	0.00	0.57	0.00
SigNet Madry	Linear	random	1.72	1.15	98.28	45.40	0.00	0.00	1.72	0.00
		skilled	48.85	43.10	100.00	77.01	0.57	0.00	2.87	0.57
	RBF	random	2.30	1.72	97.70	62.64	0.00	0.00	0.00	0.00
		skilled	45.98	40.23	100.00	84.48	0.57	0.00	0.57	0.00

Tables 5.14 and 5.15 evaluate the impact of processing the adversarial images with OTSU on the success rate of the attacks, for Type-I and Type-II attacks, respectively. We noticed that this pre-processing was effective against the gradient-free attacks, and provided some reduction in the success rate using gradient-based attacks. A possible explanation for this difference is that on the gradient-free methods used in these experiments, only small changes to a random set of pixels in done in each iteration, while the gradient-based methods can select larger changes to a smaller set of pixels (the regions where we have a large gradient of the loss w.r.t to the pixels).

5.6.6 Limitations and practical considerations

In this work we evaluated different attack scenarios (knowledge and capabilities for the attacker), but we would like to highlight some practical aspects to take into consideration for actual attacks:

- **Discretization**: In this work, we use images in floating point representation, which is appropriate for the optimization methods. Images are commonly stored in 8-bits per channel (i.e. pixels intensities that are integer values $X_{ij} \in \{0, ..., 255\}$). Simply rounding the pixel intensities to the nearest integer degrades the quality of adversarial examples. An alternative is to conduct a greedy search (changing each pixel at a time and checking if the image is still adversarial). This solution is computationally intensive, but can solve the problem (Carlini & Wagner (2017b) reported success with this search - i.e. by using this method, the discretized version of an adversarial image is still adversarial, for all images). For figures 5.4 and 5.5 we used the discretized images (and reported the score and RMSE using the discretized version of the images), so this step mainly adds more computational complexity for the attacker.

- **Physical Attacks**: We considered only attacks using digital images (i.e. after the sensor acquisition) which are limited for scenarios where digital images are used: services where the client provides a digital image (e.g. an app where the user scans a picture of a bank cheque). It has been shown that physical attacks are possible (Kurakin *et al.* (2017a), Athalye *et al.* (2018b)), where adversarial images were printed, subsequently captured using a camera, and still fooled classifiers. However, this often requires more noise to be added, to account to transformations such as slight rotations or translations of the image. Also, it is worth noting that, if noise is printed on top of a handwritten signature, the noise δ needs to be constrained to be positive. In some early experiments in this scenario, we found it to also require more noise (50% higher RMSE) than if δ does not have this constraint.

- **Knowledge of noise-removal**: In section 5.6.5, we considered a pre-processing step to remove noise, that is effective (to some extent) in many scenarios. We note, however, that this cannot be considered a robust defense, and that if the adversary is aware of it, it can use this information as part of generating the adversarial images (e.g. knowing that a threshold τ is used, consider adding only pixels with intensity larger than τ). This still increases the difficulty for gradient-based methods, since the problem becomes discontinuous (the pixel intensities can be 0 or greater than τ).

5.7 Conclusion

In this paper we investigated the impact of adversarial examples on biometric systems, in particular by identifying threats to Offline Handwritten Signature Verification under the point of view of Adversarial Machine Learning. Our experiments indicate that the issue of adversarial examples present new threats to such systems in several scenarios, including both systems using handcrafted feature extractors and systems that learn directly from image pixels. In particular, we identify that Type-I attacks (changing a genuine signature so that it is rejected by the system) were successful is all systems investigated, even in a limited knowledge scenario, where the attacker does not have access to the signatures used for training the writer-dependent classifiers. The results in this scenario confirm previous findings that attacks transfer across different CNN classifiers (Szegedy *et al.* (2014)), and show that this transferability is also present on attacks on systems using a handcrafted feature extractor (CLBP). We found, however, that transferability is greatly reduced when the CNN is trained with a different set of *users* (rather than a dijsoint set of *samples* from the same classes, as investigated in (Szegedy *et al.* (2014))). We identified that Type-II attacks (changing a forgery to be accepted as genuine) are much harder to craft, obtaining lower success rates overall, and requiring larger amounts of noise for the strong gradient-based method. This contrasts with results in object recognition literature, where successful attacks (even in a targeted setting) are reported with much lower noise (less than 3 orders of magnitude), that are commonly visually imperceptible (Carlini & Wagner (2017b)).

Lastly, we investigated some countermeasures for this problem, and confirmed previous findings that the Madry defense (Madry *et al.* (2018)) increase the amount of noise necessary to generate adversarial images. In this paper, we show that this defense is effective even when only applied on the feature learning phase, with no changes to the subsequent WD classifier training. We do note, however, that in spite of the increased amount of noise required, a strong attack (Carlini) is able to find adversarial examples most of the time. Our experiments with noise reduction show that this can reduce the success rate of attacks when the attacker is not aware of the defense, although we reiterate that this cannot be considered a robust defense (the adversary can incorporate this knowledge on the attack generation process). A definitive solution for this issue is yet an open research problem. Exploring the nature of the signal (a pen trajectory in 2D space) as part of the defense can be a promising direction for defenses. Another interesting area for future work is analyzing the impact of physical attacks (e.g. by printing adversarial noise on top of a signature).

CONCLUSION AND RECOMMENDATIONS

In this thesis, we proposed different methods to learn feature representations for Offline Handwritten Signatures directly from data (signature images). We analyzed the problem taking into consideration the constraints of real applications, such as the requirement of having a low number of samples per user and the lack of skilled forgeries for training writer-dependent classifiers. We also analyzed the limitations of these systems in an adversarial machine learning scenario.

First, we presented formulations for writer-independent feature learning followed by training writer-dependent classifiers. These approaches showed to be very effective for signature verification, improving performance 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 data from a subset of users, so that the learned features perform better in distinguishing forgeries for unseen users. With this formulation, we obtained state-of-the-art results on four datasets: GPDS, MCYT, CEDAR, and Brazilian PUC-PR, demonstrating that the features learned in this Writer-Independent format generalize well to new users.

In a second contribution, we evaluated two methods for adapting the CNN architectures to learn a fixed-size representation for signatures of different sizes. A simple method, of training a network with SPP in images of a fixed sized (and generalizing to signatures of any size) showed similar performance to previous methods, while removing the constraint of having a maximum signature size that could be processed. Our experiments varying the image resolution showed that larger resolutions are not always optimal: when only genuine signatures are available for feature learning, a relatively low resolution (100 dpi) sufficient, but if forgeries are available, higher resolutions are required in order to capture the low-level details on the pen strokes. We also showed that transfer learning can improve performance on new operating conditions, by fine-tuning representations on the other datasets.

In a third contribution, we considered a formulation of signature verification as a meta-learning problem. This allows to directly optimize the objective (discriminating genuine signatures and forgeries). The resulting system is scalable, requiring a single meta-classifier to be stored, that is adapted to each user on demand. This method showed state-of-the-art results compared to Writer-Independent approaches on GPDS, and closed the gap to Writer-Dependent systems. On the other hand, we show that this method requires more data (i.e. from more users) to be effective, and it does not transfer as well to new operating conditions.

Lastly, we analyzed the limitations of such methods in an Adversarial Machine Learning setting. In particular, we characterized new security threats to Offline Handwritten Signature Verification, and experimentally validated that adversarial examples present new threats to such systems in several scenarios, including both systems using handcrafted feature extractors and systems that learn directly from image pixels. In particular, we identify that Type-I attacks (changing a genuine signature so that it is rejected by the system) were successful is all systems investigated, even in a limited knowledge scenario, where the attacker does not have access to the signatures used for training the writer-dependent classifiers. The results in this scenario confirm previous findings that attacks transfer across different CNN classifiers, and show that this transferability is also present on attacks on systems using a handcrafted feature extractor (CLBP). We found, however, that transferability is greatly reduced when the CNN is trained with a different set of *users*. We identified that Type-II attacks (changing a forgery to be accepted as genuine) are much harder to craft, obtaining lower success rates overall, and requiring larger amounts of noise for the strong gradient-based method. This contrasts with results in object recognition literature, where successful attacks (even in a targeted setting) are reported with much lower noise (less than 3 orders of magnitude), that are commonly visually imperceptible. Lastly, we investigated some countermeasures for this problem, and confirmed previous findings that the Madry defense increases the amount of noise necessary to generate adversarial images. However, a definitive solution for this issue is yet an open research problem.

Future Work

The analyses of this thesis suggest the following directions for future work:

- **Explicitly address the issue of having a low number of samples per user**: low number of samples per class is a fundamental issue in signature verification. In this thesis, we partially addressed this problem with (i) learning feature across several users (chapter 2); (ii) a meta-learning formulation that leverages information across several users (chapter 4). However, we still notice a steep curve when we consider system performance as we vary the number of samples per user. Possible directions for addressing this issue involve using more advanced data augmentation techniques, such as those inspired by neuromotor models of the handwriting process (Diaz *et al.* (2017)).
- **Online learning**: In a practical application, new users are constantly joining the system. The approaches considered in this thesis consider only using data from a fixed Development set for learning the features / training a meta-classifier. However, in such dynamic scenario, it is possible to consider online learning where, as new data is available, it can be used to improve the system. This is particularly interesting for the meta-learning case developed in chapter 4, since a single model is used for all users, and therefore this update can benefit the performance of the system for all new queries, including for existing users of the system.
- **Improving defenses against adversarial examples**: we showed that current defenses (e.g. the Madry defense) can increase system robustness by requiring more noise to be added to the signatures in order to craft adversarial examples. However, these defenses are generic, and it is possible that better defenses exist, that consider the nature of the signal (a pen trajectory in 2D space).

- **Analyzing the impact of physical attacks**: In chapter 5, we analyzed the impact of adversarial examples for signature verification, but did not consider the aspect of physical attacks. While existing research shows that adversarial examples exist in the physical world (Kurakin *et al.* (2017a); Athalye *et al.* (2018b)), there are unique properties of the problem that deserve further investigation. For instance, a realistic scenario would involve printing adversarial noise on top of a signature, which in practice would restrict the noise to be positive. This, combined with the observation that physical attacks often requires more noise (to account for random elements in the data collection) suggest that these attacks may require a very high level of noise, that would no longer be imperceptible. This problem has a large impact, but a proper analysis of this hypothesis is needed to reach any conclusions.

APPENDIX I

DECOUPLING DIRECTION AND NORM FOR EFFICIENT GRADIENT-BASED L_2 ADVERSARIAL ATTACKS AND DEFENSES

Jérôme Rony* [1], Luiz G. Hafemann* [1], Luiz S. Oliveira[2], Ismail Ben Ayed[1], Robert Sabourin[1], Eric Granger[1]

* Equal contribution
[1] Department of Automated Manufacturing Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Departement of Informatics, Federal University of Parana (UFPR),
Curitiba, PR, Brazil

Article accepted to the « IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) » 2019.

Abstract

Research on adversarial examples in computer vision tasks has shown that small, often imperceptible changes to an image can induce misclassification, which has security implications for a wide range of image processing systems. Considering L_2 norm distortions, the Carlini and Wagner attack is presently the most effective white-box attack in the literature. However, this method is slow since it performs a line-search for one of the optimization terms, and often requires thousands of iterations. In this paper, an efficient approach is proposed to generate gradient-based attacks that induce misclassifications with low L_2 norm, by decoupling the direction and the norm of the adversarial perturbation that is added to the image. Experiments conducted on the MNIST, CIFAR-10 and ImageNet datasets indicate that our attack achieves comparable results to the state-of-the-art (in terms of L_2 norm) with considerably fewer iterations (as few as 100 iterations), which opens the possibility of using these attacks for adversarial training. Models trained with our attack achieve state-of-the-art robustness against white-box gradient-based L_2 attacks on the MNIST and CIFAR-10 datasets, outperforming the Madry defense when the attacks are limited to a maximum norm.

2. Introduction

Deep neural networks have achieved state-of-the-art performances on a wide variety of computer vision applications, such as image classification, object detection, tracking, and activity recognition (Gu *et al.* (2018)). In spite of their success in addressing these challenging tasks, they are vulnerable to active *adversaries*. Most notably, they are susceptible to *adversarial examples*[1], in which adding small perturbations to an image, often imperceptible to a human observer, causes a misclassification (Biggio & Roli (2018); Szegedy *et al.* (2014)).

Recent research on adversarial examples developed *attacks* that allow for evaluating the robustness of models, as well as *defenses* against these attacks. Attacks have been proposed to achieve different objectives, such as minimizing the amount of noise that induces misclassification (Carlini & Wagner (2017b); Szegedy *et al.* (2014)), or being fast enough to be incorporated into the training procedure (Goodfellow *et al.* (2015); Tramèr *et al.* (2018)). In particular, considering the case of obtaining adversarial examples with lowest perturbation (measured by its L_2 norm), the state-of-the-art attack has been proposed by Carlini and Wagner (C&W) (Carlini & Wagner (2017b)). While this attack generates adversarial examples with low L_2 noise, it also requires a high number of iterations, which makes it impractical for training a robust model to defend against such attacks. In contrast, one-step attacks are fast to generate, but using them for training does not increase model robustness on white-box scenarios, with full knowledge of the model under attack (Tramèr *et al.* (2018)). Developing an attack that finds adversarial examples with low noise in few iterations would enable adversarial training with such examples, which could potentially increase model robustness against white-box attacks.

Developing attacks that minimize the norm of the adversarial perturbations requires optimizing two objectives: 1) obtaining a low L_2 norm, while 2) inducing a misclassification. With the current state-of-the-art method, C&W (Carlini & Wagner (2017b)), this is addressed by using a two-term loss function, with the weight balancing the two competing objectives found via an

[1] This also affects other machine learning classifiers, but we restrict our analysis to CNNs, that are most commonly used in computer vision tasks.

expensive line search, requiring a large number of iterations. This makes the evaluation of a system's robustness very slow and it is unpractical for adversarial training.

In this paper, we propose an efficient gradient-based attack called *Decoupled Direction and Norm*[2] (DDN) that induces misclassification with a low L_2 norm. This attack optimizes the cross-entropy loss, and instead of penalizing the norm in each iteration, projects the perturbation onto a L_2-sphere centered at the original image. The change in norm is then based on whether the sample is adversarial or not. Using this approach to decouple the direction and norm of the adversarial noise leads to an attack that needs significantly fewer iterations, achieving a level of performance comparable to state-of-the-art, while being amenable to be used for adversarial training.

A comprehensive set of experiments was conducted using the MNIST, CIFAR-10 and ImageNet datasets. Our attack obtains comparable results to the state-of-the-art while requiring much fewer iterations (~100 times less than C&W). For untargeted attacks on the ImageNet dataset, our attack achieves better performance than the C&W attack, taking less than 10 minutes to attack 1 000 images, versus over 35 hours to run the C&W attack.

Results for adversarial training on the MNIST and CIFAR-10 datasets indicate that DDN can achieve state-of-the-art robustness compared to the Madry defense (Madry *et al.* (2018)). These models require that attacks use a higher average L_2 norm to induce misclassifications. They also obtain a higher accuracy when the L_2 norm of the attacks is bounded. On MNIST, if the attack norm is restricted to 1.5, the model trained with the Madry defense achieves 67.3% accuracy, while our model achieves 87.2% accuracy. On CIFAR-10, for attacks restricted to a norm of 0.5, the Madry model achieves 56.1% accuracy, compared to 67.6% in our model.

[2] Code available at https://github.com/jeromerony/fast_adversarial.

3. Related Work

In this section, we formalize the problem of adversarial examples, the threat model, and review the main attack and defense methods proposed in the literature.

3.1 Problem Formulation

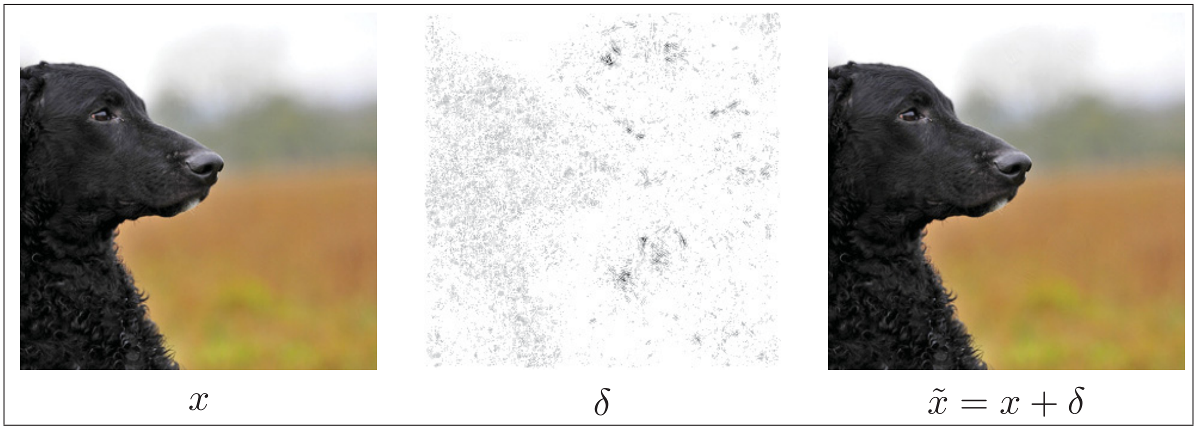


Figure-A I-1 Example of an adversarial image on the ImageNet dataset. The sample x is recognized as a Curly-coated retriever. Adding a perturbation δ we obtain an adversarial image that is classified as a microwave (with $\|\delta\|_2 = 0.7$).

Let x be an sample from the input space $\mathscr{X}$, with label y_{true} from a set of possible labels $\mathscr{Y}$. Let $D(x_1, x_2)$ be a distance measure that compares two input samples (ideally capturing their perceptual similarity). $P(\text{y}|x,\theta)$ is a model (classifier) parameterized by θ. An example $\tilde{x} \in \mathscr{X}$ is called *adversarial* (for non-targeted attacks) against the classifier if $\text{argmax}_j P(y_j|\tilde{x},\theta) \neq y_{\text{true}}$ and $D(x,\tilde{x}) \leq \varepsilon$, for a given maximum perturbation ε. A *targeted attack* with a given desired class y_{target} further requires that $\text{argmax}_j P(y_j|\tilde{x},\theta) = y_{\text{target}}$. We denote as $J(x,y,\theta)$, the cross-entropy between the prediction of the model for an input x and a label y. Figure-A I-1 illustrates a targeted attack on the ImageNet dataset, against an Inception v3 model (Szegedy *et al.* (2016)).

In this paper, attacks are considered to be generated by a gradient-based optimization procedure, restricting our analysis to differentiable classifiers. These attacks can be formulated either to obtain a minimum distortion $D(x,\tilde{x})$, or to obtain the worst possible loss in a region

$D(x,\tilde{x}) \leq \varepsilon$. As an example, consider that the distance function is a norm (e.g., L_0, L_2 or L_∞), and the inputs are images (where each pixel's value is constrained between 0 and M). In a white-box scenario, the optimization procedure to obtain an non-targeted attack with minimum distortion δ can be formulated as:

$$\begin{aligned} \min_{\delta} \|\delta\| \;\; \text{subject to} \;\; &\operatorname*{argmax}_{j} P(y_j|x+\delta,\theta) \neq y_{\text{true}} \\ \text{and} \;\; &0 \leq x+\delta \leq M \end{aligned} \tag{A I-1}$$

With a similar formulation for *targeted attacks*, by changing the constraint to be equal to the target class.

If the objective is to obtain the worst possible loss for a given maximum noise of norm ε, the problem can be formulated as:

$$\begin{aligned} \min_{\delta} P(y_{\text{true}}|x+\delta,\theta) \;\; \text{subject to} \;\; &\|\delta\| \leq \varepsilon \\ \text{and} \;\; &0 \leq x+\delta \leq M \end{aligned} \tag{A I-2}$$

With a similar formulation for *targeted attacks*, by maximizing $P(y_{\text{target}}|x+\delta,\theta)$.

We focus on gradient-based attacks that optimize the L_2 norm of the distortion. While this distance does not perfectly capture perceptual similarity, it is widely used in computer vision to measure similarity between images (e.g. comparing image compression algorithms, where Peak Signal-to-Noise Ratio is used, which is directly related to the L_2 measure). A differentiable distance measure that captures perceptual similarity is still an open research problem.

3.2 Threat Model

In this paper, a *white-box* scenario is considered, also known as a Perfect Knowledge scenario (Biggio & Roli (2018)). In this scenario, we consider that an attacker has perfect knowledge of the system, including the neural network architecture and the learned weights θ. This threat model serves to evaluate system security under the *worst case* scenario. Other scenarios can be conceived to evaluate attacks under different assumptions on the attacker's knowledge, for instance, no access to the trained model, no access to the same training set, among others. These scenarios are referred as *black-box* or Limited-Knowledge (Biggio & Roli (2018)).

3.3 Attacks

Several attacks were proposed in the literature, either focusing on obtaining adversarial examples with a small δ (Equation A I-1) (Carlini & Wagner (2017b); Moosavi-Dezfooli *et al.* (2016); Szegedy *et al.* (2014)), or on obtaining adversarial examples in one (or few) steps for adversarial training (Goodfellow *et al.* (2015); Kurakin *et al.* (2017b)).

L-BFGS. Szegedy *et al.* (2014) proposed an attack for minimally distorted examples (Equation A I-1), by considering the following approximation:

$$\begin{aligned} &\min_{\delta} C \, \|\delta\|_2 + \log P(y_{\text{true}}|x+\delta,\theta) \\ &\text{subject to } \; 0 \le x+\delta \le M \end{aligned} \qquad \text{(A I-3)}$$

where the constraint $x+\delta \in [0,M]^n$ was addressed by using a box-constrained optimizer (L-BFGS: Limited memory Broyden–Fletcher–Goldfarb–Shanno), and a line-search to find an appropriate value of C.

FGSM. Goodfellow *et al.* (2015) proposed the Fast Gradient Sign Method, a one-step method that could generate adversarial examples. The original formulation was developed considering the L_∞ norm, but it has also been used to generate attacks that focus on the L_2 norm as follows:

$$\tilde{x} = x + \varepsilon \frac{\nabla_x J(x,y,\theta)}{\|\nabla_x J(x,y,\theta)\|} \qquad \text{(A I-4)}$$

where the constraint $\tilde{x} \in [0,M]^n$ was addressed by simply clipping the resulting adversarial example.

DeepFool. This method considers a linear approximation of the model, and iteratively refines an adversary example by choosing the point that would cross the decision boundary under this approximation. This method was developed for untargeted attacks, and for any L_p norm (Moosavi-Dezfooli *et al.* (2016)).

C&W. Similarly to the L-BFGS method, the C&W L_2 attack (Carlini & Wagner (2017b)) minimizes two criteria at the same time – the perturbation that makes the sample adversarial (e.g., misclassified by the model), and the L_2 norm of the perturbation. Instead of using a box-constrained optimization method, they propose changing variables using the tanh function, and instead of optimizing the cross-entropy of the adversarial example, they use a difference between logits. For a targeted attack aiming to obtain class t, with Z denoting the model output before the softmax activation (logits), it optimizes:

$$\begin{aligned} &\min_{\delta}\left[\|\tilde{x}-x\|_2^2+Cf(\tilde{x})\right] \\ \text{where} \quad &f(\tilde{x})=\max(\max_{i\neq t}\{Z(\tilde{x})_i\}-Z(\tilde{x})_t,-\kappa) \\ \text{and} \quad &\tilde{x}=\frac{1}{2}(\tanh(\operatorname{arctanh}(x)+\delta)+1) \end{aligned} \tag{A I-5}$$

where $Z(\tilde{x})_i$ denotes the logit corresponding to the i-th class. By increasing the confidence parameter κ, the adversarial sample will be misclassified with higher confidence. To use this attack in the untargeted setting, the definition of f is modified to $f(\tilde{x})=\max(Z(\tilde{x})_y-\max_{i\neq y}\{Z(\tilde{x})_i\},-\kappa)$ where y is the original label.

3.4 Defenses

Developing defenses against adversarial examples is an active area of research. To some extent, there is an *arms race* on developing defenses and attacks that break them. Goodfellow *et al.*proposed a method called *adversarial training* (Goodfellow *et al.* (2015)), in which the training data is augmented with FGSM samples. This was later shown not to be robust against iterative white-box attacks, nor black-box single-step attacks (Tramèr *et al.* (2018)). Papernot *et al.* (2016) proposed a *distillation* procedure to train robust networks, which was shown to be easily broken by iterative white-box attacks (Carlini & Wagner (2017b)). Other defenses involve *obfuscated gradients* (Athalye *et al.* (2018a)), where models either incorporate non-differentiable steps (such that the gradient cannot be computed) (Buckman *et al.* (2018); Guo *et al.* (2018)), or randomized elements (to induce incorrect estimations of the gradient) (Dhillon *et al.* (2018); Xie *et al.* (2018)). These defenses were later shown to be ineffective when attacked with Backward Pass Differentiable Approximation (BPDA) (Athalye *et al.* (2018a)), where the actual model is used for forward propagation, and the gradient in the backward-pass is approximated. The Madry defense (Madry *et al.* (2018)), which considers a worst-case optimization, is the only defense that has been shown to be somewhat robust (on the MNIST and CIFAR-10 datasets). Below we provide more detail on the general approach of adversarial training, and the Madry defense.

Adversarial Training. This defense considers augmenting the training objective with adversarial examples (Goodfellow *et al.* (2015)), with the intention of improving robustness. Given a model with loss function $J(x,y,\theta)$, training is augmented as follows:

$$\tilde{J}(x,y,\theta) = \alpha J(x,y,\theta) + (1-\alpha)J(\tilde{x},y,\theta) \tag{A I-6}$$

where $\tilde{x}$ is an adversarial sample. In (Goodfellow *et al.* (2015)), the FGSM is used to generate the adversarial example in a single step. Tramèr *et al.* (2018) extended this method, showing that generating one-step attacks using the model under training introduced an issue. The model can converge to a degenerate solution where its gradients produce "easy" adversarial samples, causing the adversarial loss to have a limited influence on the training objective. They proposed a method in which an ensemble of models is also used to generate the adversarial examples $\tilde{x}$. This method displays some robustness against black-box attacks using surrogate models, but does not increase robustness in white-box scenarios.

Madry Defense. Madry *et al.* (2018) proposed a saddle point optimization problem, in which we optimize for the worst case:

$$\begin{gathered}\min_{\theta} p(\theta) \\ \text{where} \quad p(\theta) = \mathbb{E}_{(x,y)\sim\mathscr{D}}\big[\max_{\delta\in\mathscr{S}} J(x+\delta,y,\theta)\big]\end{gathered} \tag{A I-7}$$

where $\mathscr{D}$ is the training set, and $\mathscr{S}$ indicates the feasible region for the attacker (e.g. $\mathscr{S} = \{\delta : \|\delta\| < \varepsilon\}$). They show that Equation A I-7 can be optimized by stochastic gradient descent – during each training iteration, it first finds the adversarial example that maximizes the loss around the current training sample x (i.e., maximizing the loss over δ, which is equivalent to minimizing the probability of the correct class as in Equation A I-2), and then, it minimizes the loss over θ. Experiments by Athalye *et al.* (2018a) show that it was the only defense not broken under white-box attacks.

4. Decoupled Direction and Norm Attack

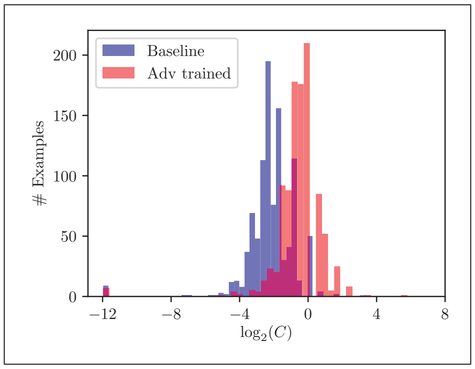
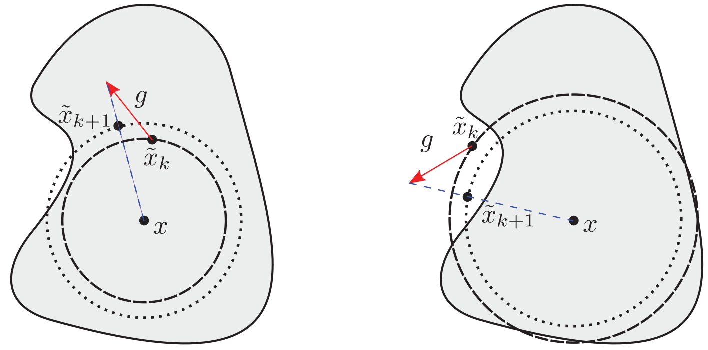


Figure-A I-2 Histogram of the best C found by the C&W algorithm with 9 search steps on the MNIST dataset.

From the problem definition, we see that finding the worst adversary in a fixed region is an easier task. In Equation A I-2, both constraints can be expressed in terms of δ, and the resulting equation can be optimized using projected gradient descent. Finding the closest adversarial example is harder: Equation A I-1 has a constraint on the prediction of the model, which cannot be addressed by a simple projection. A common approach, which is used by Szegedy *et al.* (2014) and Carlini & Wagner (2017b) is to approximate the constrained problem in Equation A I-1 by an unconstrained one, replacing the constraint with a *penalty*. This amounts to jointly optimizing both terms, the norm of δ and a classification term (see Eq. A I-3 and A I-5), with a sufficiently high parameter C. In the general context of constrained optimization, such a penalty-based approach is a well known general principle (Jensen & Bard (2003)). While tackling an unconstrained problem is convenient, penalty methods have well-known difficulties in practice. The main difficulty is that one has to choose parameter C in an *ad hoc* way. For instance, if C is too small in Equation A I-5, the example will not be adversarial; if it is too large, this term will dominate, and result in an adversarial example with more noise. This can be particularly problematic when optimizing with a low number of steps (e.g. to enable its use in adversarial training). Figure-A I-2 plots a histogram of the values of C that were obtained by running the C&W attack on the MNIST dataset. We can see that the optimum C varies significantly among different examples, ranging from 2^{-11} to 2^{5}. We also see that the distribution of the best constant C changes whether we attack a model with or without adversarial training (adversarially trained models often require higher C). Furthermore, penalty methods typically result in slow convergence (Jensen & Bard (2003)).

Algorithm-A I-1 Decoupled Direction and Norm Attack

```
**Input:** $x$: *original image to be attacked*
**Input:** $y$: *true label (untargeted) or target label (targeted)*
**Input:** $K$: *number of iterations*
**Input:** $\alpha$: *step size*
**Input:** $\gamma$: *factor to modify the norm in each iteration*
**Output:** $\tilde{x}$: *adversarial image*
Initialize $\delta_0 \leftarrow \mathbf{0}$, $\tilde{x}_0 \leftarrow x$, $\varepsilon_0 \leftarrow 1$
If targeted attack: $m \leftarrow -1$ else $m \leftarrow +1$
**for** $k \leftarrow 1$ *to* $K$ **do**
    $g \leftarrow m\nabla_{\tilde{x}_{k-1}} J(\tilde{x}_{k-1}, y, \theta)$
    $g \leftarrow \alpha \frac{g}{\|g\|_2}$ // Step of size $\alpha$ in the direction of $g$
    $\delta_k \leftarrow \delta_{k-1} + g$
    **if** $\tilde{x}_{k-1}$ *is adversarial* **then**
        $\varepsilon_k \leftarrow (1-\gamma)\varepsilon_{k-1}$ // Decrease norm
    **end**
    **else**
        $\varepsilon_k \leftarrow (1+\gamma)\varepsilon_{k-1}$ // Increase norm
    **end**
    $\tilde{x}_k \leftarrow x + \varepsilon_k \frac{\delta_k}{\|\delta_k\|_2}$ // Project $\delta_k$ onto an $\varepsilon_k$-sphere around $x$
    $\tilde{x}_k \leftarrow \mathrm{clip}(\tilde{x}_k, 0, 1)$ // Ensure $\tilde{x}_k \in \mathscr{X}$
**end**
Return $\tilde{x}_k$ that has lowest norm $\|\tilde{x}_k - x\|_2$ and is adversarial
```



(a) $\tilde{x}_k$ not adversarial

(b) $\tilde{x}_k$ adversarial

Figure-A I-3 Illustration of an untargeted attack. The shaded area denotes the region of the input space classified as y_{true}. In (a), $\tilde{x}_k$ is still not adversarial, and we increase the norm ε_{k+1} for the next iteration, otherwise it is reduced in (b). In both cases, we take a step g starting from the current point $\tilde{x}$, and project back to an ε_{k+1}-sphere centered at x.

Given the difficulty of finding the appropriate constant C for this optimization, we propose an algorithm that does not impose a penalty on the L_2 norm during the optimization. Instead, the norm is constrained by projecting the adversarial perturbation δ on an ε-sphere around the original image x. Then, the L_2 norm is modified through a binary decision. If the sample x_k is not adversarial at step k, the norm is increased for step $k+1$, otherwise it is decreased.

We also note that optimizing the cross-entropy may present two other difficulties. First, the function is not bounded, which can make it dominate in the optimization of Equation A I-3. Second, when attacking trained models, often the predicted probability of the correct class for the original image is very close to 1, which causes the cross entropy to start very low and increase by several orders of magnitude during the search for an adversarial example. This affects the norm of the gradient, making it hard to find an appropriate learning rate. C&W address these issues by optimizing the difference between logits instead of the cross-entropy. In this work, the issue of it being unbounded does not affect the attack procedure, since the decision to update the norm is done on the model's prediction (not on the cross-entropy). In order to handle the issue of large changes in gradient norm, we normalize the gradient to have unit norm before taking a step in its direction.

The full procedure is described in algorithm I-1 and illustrated in Figure-A I-3. We start from the original image x, and iteratively refine the noise δ_k. In iteration k, if the current sample $\tilde{x}_k = x + \delta_k$ is still not adversarial, we consider a larger norm $\varepsilon_{k+1} = (1+\gamma)\varepsilon_k$. Otherwise, if the sample is adversarial, we consider a smaller $\varepsilon_{k+1} = (1-\gamma)\varepsilon_k$. In both cases, we take a step g (step 5 of algorithm I-1) from the point $\tilde{x}_k$ (red arrow in Figure-A I-3), and project it back onto an ε_{k+1}-sphere centered at x (the direction given by the dashed blue line in Figure-A I-3), obtaining $\tilde{x}_{k+1}$. Lastly, $\tilde{x}_{k+1}$ is projected onto the feasible region of the input space $\mathscr{X}$. In the case of images normalized to $[0,1]$, we simply clip the value of each pixel to be inside this range (step 13 of algorithm I-1). Besides this step, we can also consider quantizing the image in each iteration, to ensure the attack is a valid image.

It's worth noting that, when reaching a point where the decision boundary is tangent to the ε_k-sphere, g will have the same direction as δ_{k+1}. This means that δ_{k+1} will be projected on the direction of δ_k. Therefore, the norm will oscillate between the two sides of the decision boundary in this direction. Multiplying ε by $1+\gamma$ and $1-\gamma$ will result in a global decrease (on two steps) of the norm by $1-\gamma^2$, leading to a finer search of the best norm.

5. Attack Evaluation

Experiments were conducted on the MNIST, CIFAR-10 and ImageNet datasets, comparing the proposed attack to the state-of-the-art L_2 attacks proposed in the literature: DeepFool (Moosavi-Dezfooli *et al.* (2016)) and C&W L_2 attack (Carlini & Wagner (2017b)). We use the same model architectures with identical hyperparameters for training as in (Carlini & Wagner (2017b)) for MNIST and CIFAR-10 (see the supplementary material for details). Our base classifiers obtain 99.44% and 85.51% accuracy on the test sets of MNIST and CIFAR-10, respectively. For the ImageNet experiments, we use a pre-trained Inception V3 (Szegedy *et al.* (2016)), that achieves 22.51% top-1 error on the validation set. Inception V3 takes images of size 299×299 as input, which are cropped from images of size 342×342.

Table-A I-1 Performance of our DDN attack compared to C&W and DeepFool attacks on MNIST, CIFAR-10 and ImageNet in the untargeted scenario.

	Attack	Budget	Success	Mean L_2	Median L_2	#Grads	Run-time (s)
MNIST	C&W	4×25	100.0	1.7382	1.7400	100	1.7
		1×100	99.4	1.5917	1.6405	100	1.7
		9×10 000	100.0	**1.3961**	1.4121	54 007	856.8
	DeepFool	100	75.4	1.9685	2.2909	98	-
	DDN	100	100.0	1.4563	1.4506	100	1.5
		300	100.0	1.4357	1.4386	300	4.5
		1 000	100.0	1.4240	1.4342	1 000	14.9
CIFAR-10	C&W	4×25	100.0	0.1924	0.1541	60	3.0
		1×100	99.8	0.1728	0.1620	91	4.6
		9×10 000	100.0	0.1543	0.1453	36 009	1 793.2
	DeepFool	100	99.7	0.1796	0.1497	25	-
	DDN	100	100.0	0.1503	0.1333	100	4.7
		300	100.0	0.1487	0.1322	300	14.2
		1 000	100.0	**0.1480**	0.1317	1 000	47.6
ImageNet	C&W	4×25	100.0	1.5812	1.3382	63	379.3
		1×100	100.0	0.9858	0.9587	48	287.1
		9×10 000	100.0	0.4692	0.3980	21 309	127 755.6
	DeepFool	100	98.5	0.3800	0.2655	41	-
	DDN	100	99.6	0.3831	0.3227	100	593.6
		300	100.0	0.3749	0.3210	300	1 779.4
		1 000	100.0	**0.3617**	0.3188	1 000	5 933.6

Table-A I-2 Comparison of the DDN attack to the C&W L_2 attack on MNIST.

	Average case		Least Likely	
Attack	Success	Mean L_2	Success	Mean L_2
C&W 4×25	96.11	2.8254	69.9	5.0090
C&W 1×100	86.89	2.0940	31.7	2.6062
C&W 9×10 000	100.00	1.9481	100.0	2.5370
DDN 100	100.00	1.9763	100.0	2.6008
DDN 300	100.00	1.9577	100.0	2.5503
DDN 1 000	100.00	1.9511	100.0	2.5348

For experiments with DeepFool (Moosavi-Dezfooli *et al.* (2016)), we used the implementation from Foolbox (Rauber *et al.* (2017)), with a budget of 100 iterations. For the experiments with C&W, we ported the attack (originally implemented on TensorFlow) on PyTorch to evaluate the models in the frameworks in which they were trained. We use the same hyperparameters

Table-A I-3 Comparison of the DDN attack to the C&W L_2 attack on CIFAR-10.

	Average case		Least Likely	
Attack	**Success**	**Mean L_2**	**Success**	**Mean L_2**
C&W 4×25	99.78	0.3247	98.7	0.5060
C&W 1×100	99.32	0.3104	95.8	0.4159
C&W 9×10 000	100.00	0.2798	100.0	0.3905
DDN 100	100.00	0.2925	100.0	0.4170
DDN 300	100.00	0.2887	100.0	0.4090
DDN 1 000	100.00	0.2867	100.0	0.4050

Table-A I-4 Comparison of the DDN attack to the C&W L_2 attack on ImageNet. For C&W 9×10 000, we report the results from Carlini & Wagner (2017b).

	Average case		Least Likely	
Attack	**Success**	**Mean L_2**	**Success**	**Mean L_2**
C&W 4×25	99.13	4.2826	80.6	8.7336
C&W 1×100	96.74	1.7718	66.2	2.2997
C&W 9×10 000	100.00	0.96	100.0	2.22
DDN 100	99.98	1.0260	99.5	1.7074
DDN 300	100.00	0.9021	100.0	1.3634
DDN 1 000	100.00	0.8444	100.0	1.2240

from (Carlini & Wagner (2017b)): 9 search steps on C with an initial constant of 0.01, with 10 000 iterations for each search step (with early stopping) - we refer to this scenario as C&W 9×10 000 in the tables. As we are interested in obtaining attacks that require few iterations, we also report experiments in a scenario where the number of iterations is limited to 100. We consider a scenario of running 100 steps with a fixed C (1×100), and a scenario of running 4 search steps on C, of 25 iterations each (4×25). Since the hyperparameters proposed in (Carlini & Wagner (2017b)) were tuned for a larger number of iterations and search steps, we performed a grid search for each dataset, using learning rates in the range [0.01, 0.05, 0.1, 0.5, 1], and C in the range [0.001, 0.01, 0.1, 1, 10, 100, 1 000]. We report the results for C&W with the hyperparameters that achieve best Median L_2. Selected parameters are listed in the supplementary material.

For the experiments using DDN, we ran attacks with budgets of 100, 300 and 1 000 iterations, in all cases, using $\varepsilon_0 = 1$ and $\gamma = 0.05$. The initial step size $\alpha = 1$, was reduced with cosine annealing to 0.01 in the last iteration. The choice of γ is based on the encoding of images. For any correctly classified image, the smallest possible perturbation consists in changing one pixel by $1/255$ (for images encoded in 8 bit values), corresponding to a norm of $1/255$. Since we perform quantization, the values are rounded, meaning that the algorithm must be able to achieve a norm lower than $1.5/255 = 3/510$. When using K steps, this imposes:

$$\varepsilon_0(1-\gamma)^K < \frac{3}{510} \Rightarrow \gamma > 1 - \left(\frac{3}{510\,\varepsilon_0}\right)^{\frac{1}{K}} \tag{A I-8}$$

Using $\varepsilon_0 = 1$ and $K = 100$ yields $\gamma \simeq 0.05$. Therefore, if there exists an adversarial example with smallest perturbation, the algorithm may find it in a fixed number of steps.

For the results with DDN, we consider quantized images (to 256 levels). The quantization step is included in each iteration (see step 13 of algorithm I-1). All results reported in the paper consider images in the $[0,1]$ range.

Two sets of experiments were conducted: untargeted attacks and targeted attacks. As in (Carlini & Wagner (2017b)), we generated attacks on the first 1 000 images of the test set for MNIST and CIFAR-10, while for ImageNet we randomly chose 1 000 images from the validation set that are correctly classified. For the untargeted attacks, we report the success rate of the attack (percentage of samples for which an attack was found), the mean L_2 norm of the adversarial noise (for successful attacks), and the median L_2 norm over all attacks while considering unsuccessful attacks as worst-case adversarial (distance to a uniform gray image, as introduced by (Brendel *et al.* (2018b))). We also report the average number (for batch execution) of gradient computations and the total run-times (in seconds) on a NVIDIA GTX 1080 Ti with 11GB of memory. We did not report run-times for the DeepFool attack, since the implementation from foolbox generates adversarial examples one-by-one and is executed on CPU, leading to unrepresentative run-times. Attacks on MNIST and CIFAR-10 have been executed in a single batch of 1 000 samples, whereas attacks on ImageNet have been executed in 20 batches of 50 samples.

For the targeted attacks, following the protocol from (Carlini & Wagner (2017b)), we generate attacks against all possible classes on MNIST and CIFAR-10 (9 attacks per image), and against 100 randomly chosen classes for ImageNet (10% of the number of classes). Therefore, in each targeted attack experiment, we run 9 000 attacks on MNIST and CIFAR-10, and 100 000 attacks on ImageNet. Results are reported for two scenarios: 1) average over all attacks; 2) average performance when choosing the least likely class (i.e. choosing the worst attack performance over all target classes, for each image). The reported L_2 norms are, as in the untargeted scenario, the means over successful attacks.

Table-A I-1 reports the results of DDN compared to the C&W L_2 and DeepFool attacks on the MNIST, CIFAR-10 and ImageNet datasets. For the MNIST and CIFAR-10 datasets, results with DDN are comparable to the state-of-the-art. DDN obtains slightly worse L_2 norms on the MNIST dataset (when compared to the C&W 9×10 000), however, our attack is able to get within 5% of the norm found by C&W in only 100 iterations compared to the 54 007 iterations required for the C&W L_2 attack. When the C&W attack is restricted to use a maximum of 100 iterations, it always performed worse than DDN with 100 iterations. On the ImageNet dataset, our attack obtains better Mean L_2 norms than both other attacks. The DDN attack needs 300 iterations to reach 100% success rate. DeepFool obtains close results but fails to reach 100% success rate. It is also worth noting that DeepFool seems to performs worse against adversarially trained models (discussed in section 7). Supplementary material reports curves of the perturbation size against accuracy of the models for the three attacks.

Tables I-2, I-3 and I-4 present the results on targeted attacks on the MNIST, CIFAR-10 and ImageNet datasets, respectively. For the MNIST and CIFAR-10 datasets, DDN yields similar performance compared to the C&W attack with 9×10 000 iterations, and always perform better than the C&W attack when it is restricted to 100 iterations (we re-iterate that the hyperparameters for the C&W attack were tuned for each dataset, while the hyperparameters for DDN are fixed for all experiments). On the ImageNet dataset, DDN run with 100 iterations obtains superior performance than C&W. For all datasets, with the scenario restricted to 100 iterations, the C&W algorithm has a noticeable drop in success rate for finding adversarial examples to the least likely class.

6. Adversarial Training with DDN

Since the DDN attack can produce adversarial examples in relatively few iterations, it can be used for adversarial training. For this, we consider the following loss function:

$$\tilde{J}(x,y,\theta) = J(\tilde{x},y,\theta) \tag{A I-9}$$

where $\tilde{x}$ is an adversarial example produced by the DDN algorithm, that is projected to an ε-ball around x, such that the classifier is trained with adversarial examples with a maximum norm of ε. It is worth making a parallel of this approach with the Madry defense (Madry *et al.* (2018)) where, in each iteration, the loss of the worst-case adversarial (see Equation A I-2) in an ε-ball around the original sample x is used for optimization. In our proposed adversarial training procedure, we optimize the loss of the closest adversarial example (see Equation A I-1). The intuition of this defense is to push the decision boundary away from x in each iteration. We do note that this method does not have the theoretical guarantees of the Madry defense. However, since in practice the Madry defense uses approximations (when searching for the global maximum of the loss around x), we argue that both methods deserve empirical comparison.

7. Defense Evaluation

We trained models using the same architectures as (Carlini & Wagner (2017b)) for MNIST, and a Wide ResNet (WRN) 28-10 (Zagoruyko & Komodakis (2016)) for CIFAR-10 (similar to (Madry *et al.* (2018)) where they use a WRN 34-10). As described in section 6, we augment the training images with adversarial perturbations. For each training step, we run the DDN attack with a budget of 100 iterations, and limit the norm of the perturbation to a maximum $\varepsilon = 2.4$ on the MNIST experiments, and $\varepsilon = 1$ for the CIFAR-10 experiments. For MNIST, we train the model for 30 epochs with a learning rate of 0.01 and then for 20 epochs with a learning rate of 0.001. To reduce the training time with CIFAR-10, we first train the model on original images for 200 epochs using the hyperparameters from (Zagoruyko & Komodakis (2016)). Then, we continue training for 30 more epochs using Equation A I-9, keeping the same final learning rate of 0.0008. Our robust MNIST model has a test accuracy of 99.01% on the clean samples, while the Madry model has an accuracy of 98.53%. On CIFAR-10, our model reaches a test accuracy of 89.0% while the model by Madry *et al.*obtains 87.3%.

Table-A I-5 Evaluation of the robustness of our adversarial training on MNIST against the Madry defense.

Defense	**Attack**	**Attack Success**	**Mean L_2**	**Median L_2**	**Model Accuracy at $\varepsilon \leq 1.5$**
Baseline	C&W 9×10 000	100.0	1.3961	1.4121	42.1
	DeepFool 100	75.4	1.9685	2.2909	81.8
	DDN 1 000	100.0	1.4240	1.4342	45.2
	All	100.0	1.3778	1.3946	40.8
Madry *et al.*	C&W 9×10 000	100.0	2.0813	2.1071	73.0
	DeepFool 100	91.6	4.9585	5.2946	93.1
	DDN 1 000	99.6	1.8436	1.8994	69.9
	All	100.0	1.6917	1.8307	67.3
Ours	C&W 9×10 000	100.0	2.5181	2.6146	88.0
	DeepFool 100	94.3	3.9449	4.1754	92.7
	DDN 1 000	100.0	2.4874	2.5781	87.6
	All	100.0	**2.4497**	2.5538	**87.2**

We evaluate the adversarial robustness of the models using three untargeted attacks: Carlini 9×10 000, DeepFool 100 and DDN 1 000. For each sample, we consider the smallest adversarial perturbation produced by the three attacks and report it in the "**All**" row. Tables I-5 and I-6 report the results of this evaluation with a comparison to the defense of Madry *et al.* (2018)[3] and the baseline (without adversarial training) for CIFAR-10. For MNIST, the baseline corre-

[3] Models taken from https://github.com/MadryLab

Table-A I-6 Evaluation of the robustness of our adversarial training on CIFAR-10 against the Madry defense.

Defense	**Attack**	**Attack Success**	**Mean L_2**	**Median L_2**	**Model Accuracy at $\varepsilon \leq 0.5$**
Baseline WRN 28-10	C&W 9×10 000	100.0	0.1343	0.1273	0.2
	DeepFool 100	99.3	0.5085	0.4241	38.3
	DDN 1 000	100.0	0.1430	0.1370	0.1
	All	100.0	0.1282	0.1222	0.1
Madry *et al.* WRN 34-10	C&W 9×10 000	100.0	0.6912	0.6050	57.1
	DeepFool 100	95.6	1.4856	0.9576	64.7
	DDN 1 000	100.0	0.6732	0.5876	56.9
	All	100.0	0.6601	0.5804	56.1
Ours WRN 28-10	C&W 9×10 000	100.0	0.8860	0.8254	67.9
	DeepFool 100	99.7	1.5298	1.1163	69.9
	DDN 1 000	100.0	0.8688	0.8177	68.0
	All	100.0	**0.8597**	0.8151	**67.6**

sponds to the model used in section 5. We observe that for attacks with unbounded norm, the attacks can successfully generate adversarial examples almost 100% of the time. However, an increased L_2 norm is required to generate attacks against the model trained with DDN.

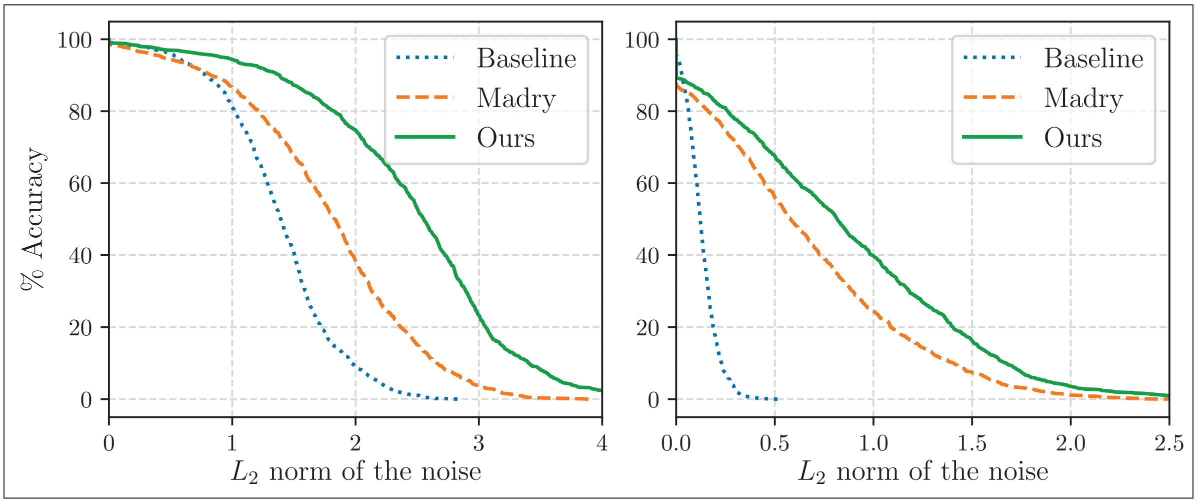


Figure-A I-4 Models robustness on MNIST (left) and CIFAR-10 (right): impact on accuracy as we increase the maximum perturbation ε.

Figure-A I-4 shows the robustness of the MNIST and CIFAR-10 models respectively for different attacks with increasing maximum L_2 norm. These figures can be interpreted as the expected accuracy of the systems in a scenario where the adversary is constrained to make changes with norm $L_2 \leq \varepsilon$. For instance on MNIST, if the attacker is limited to a maximum norm of $\varepsilon = 1.5$, the baseline performance decreases to 40.8%; Madry to 67.3% and our defense to 87.2%. At $\varepsilon = 2.0$, baseline performance decreases to 9.2%, Madry to 38.6% and our defense to 74.8%. On CIFAR-10, if the attacker is limited to a maximum norm of $\varepsilon = 0.5$, the baseline performance decreases to 0.1%; Madry to 56.1% and our defense to 67.6%. At $\varepsilon = 1.0$, baseline performance decreases to 0%, Madry to 24.4% and our defense to 39.9%. For both datasets, the model trained with DDN outperforms the model trained with the Madry defense for all values of ε.

Figure-A I-5 shows adversarial examples produced by the DDN 1 000 attack for different models on MNIST and CIFAR-10. On MNIST, adversarial examples for the baseline are not meaningful (the still visually belong to the original class), whereas some adversarial examples obtained for the adversarially trained model (DDN) actually change classes (bottom right: 0 changes to 6). For all models, there are still some adversarial examples that are very close to the original images (first column). On CIFAR-10, while the adversarially trained models require higher norms for the attacks, most adversarial examples still perceptually resemble the original images. In few cases (bottom-right example for CIFAR-10), it could cause a confusion: it can appear as changing to class 1 - a (cropped) automobile facing right.

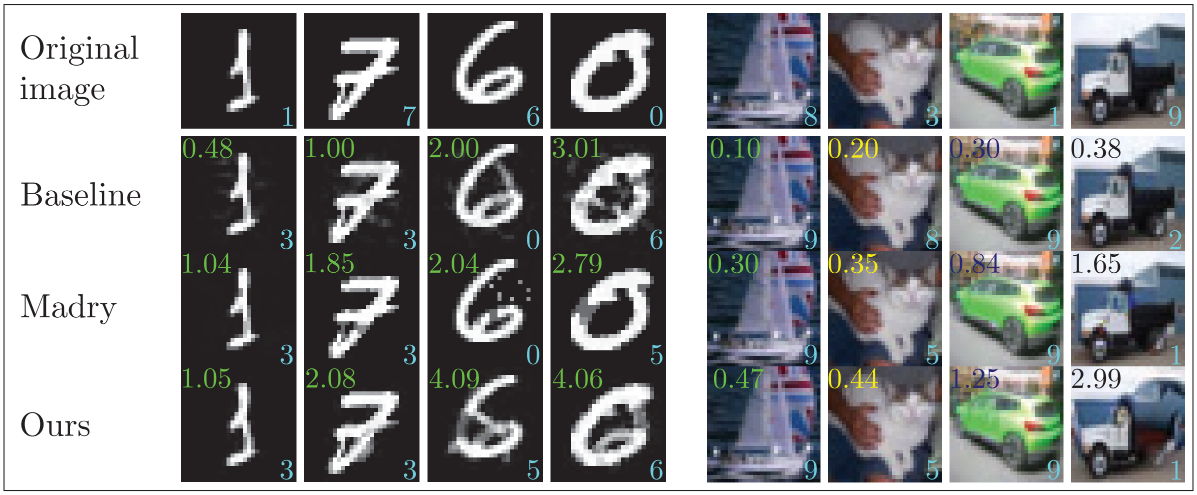


Figure-A I-5 Adversarial examples with varied levels of noise δ against three models: baseline, Madry defense and our defense. Text on top-left of each image indicate $\|\delta\|_2$; text on bottom-right indicates the predicted class[4].

8. Conclusion

We presented the *Decoupled Direction and Norm* attack, which obtains comparable results with the state-of-the-art for L_2 norm adversarial perturbations, but in much fewer iterations. Our attack allows for faster evaluation of the robustness of differentiable models, and enables a novel adversarial training, where, at each iteration, we train with examples close to the decision boundary. Our experiments with MNIST and CIFAR-10 show state-of-the-art robustness against L_2-based attacks in a white-box scenario. Future work includes the evaluation of the transferability of attacks in black-box scenarios.

The methods presented in this paper were used in NIPS 2018 Adversarial Vision Challenge Brendel *et al.* (2018b), ranking first in untargeted attacks, and third in targeted attacks and robust models (both attacks and defense in a black-box scenario). These results highlight the effectiveness of the defense mechanism, and suggest that attacks using adversarially-trained surrogate models can be effective in black-box scenarios, which is a promising future direction.

[4] For CIFAR-10: 1: automobile, 2: bird, 3: cat, 5: dog, 8: ship, 9: truck.

Supplementary material

9. Model architectures

Table-A I-7 lists the architectures of the CNNs used in the Attack Evaluation - we used the same architecture as in (Carlini & Wagner (2017b)) for a fair comparison against the C&W and DeepFool attacks. Table-A I-8 lists the architecture used in the robust model (defense) trained on CIFAR-10. We used a Wide ResNet with 28 layers and widening factor of 10 (WRN-28-10). The residual blocks used are the “basic block" (He *et al.* (2016); Zagoruyko & Komodakis (2016)), with stride 1 for the first group and stride 2 for the second an third groups. This architecture is slightly different from the one used by Madry *et al.* (2018), where they use a modified version of Wide ResNet with 5 residual blocks instead of 4 in each group, and without convolutions in the residual connections (when the shape of the output changes, e.g. with stride=2).

Table-A I-7 CNN architectures used for the Attack Evaluation

Layer Type	**MNIST Model**	**CIFAR-10 Model**
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Max Pooling	2×2	2×2
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Max Pooling	2×2	2×2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Fully Connected + Softmax	10	10

10. Hyperparameters selected for the C&W attack

We considered a scenario of running the C&W attack with 100 steps and a fixed C (1×100), and a scenario of running 4 search steps on C, of 25 iterations each (4×25). Since the hyperparameters proposed in (Carlini & Wagner (2017b)) were tuned for a larger number of iterations

Table-A I-8 CIFAR-10 architecture used for the Defense evaluation

Layer Type	**Size**
Convolution	$3 \times 3 \times 16$
Residual Block	$\begin{bmatrix} 3 \times 3, 160 \\ 3 \times 3, 160 \end{bmatrix} \times 4$
Residual Block	$\begin{bmatrix} 3 \times 3, 320 \\ 3 \times 3, 320 \end{bmatrix} \times 4$
Residual Block	$\begin{bmatrix} 3 \times 3, 640 \\ 3 \times 3, 640 \end{bmatrix} \times 4$
Batch Normalization + ReLU	-
Average Pooling	8×8
Fully Connected + Softmax	10

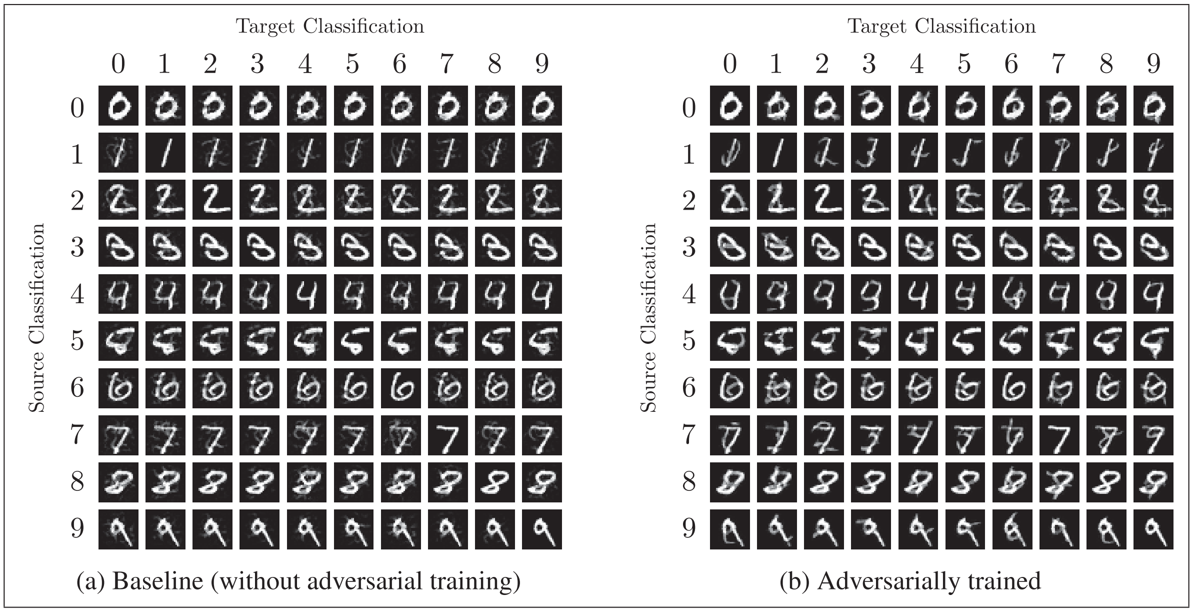
and search steps, we performed a grid search for each dataset, using learning rates in the range [0.01, 0.05, 0.1, 0.5, 1], and C in the range [0.001, 0.01, 0.1, 1, 10, 100, 1 000]. We selected the hyperparameters that resulted in targeted attacks with lowest Median L_2 for each dataset. Table-A I-9 lists the hyperparameters found through this search procedure.

Table-A I-9 Hyperparameters used for the C&W attack when restricted to 100 iterations.

Dataset	**# Iterations**	**Parameters**
MNIST	1×100	$\alpha = 0.1, C = 1$
MNIST	4×25	$\alpha = 0.5, C = 1$
CIFAR-10	1×100	$\alpha = 0.01, C = 0.1$
CIFAR-10	4×25	$\alpha = 0.01, C = 0.1$
ImageNet	1×100	$\alpha = 0.01, C = 1$
ImageNet	4×25	$\alpha = 0.01, C = 10$

11. Examples of adversarial images

Figure-A I-6 plots a grid of attacks (obtained with the C&W attack) against the first 10 examples in the MNIST dataset. The rows indicate the source classification (label), and the columns indicate the target class used to generate the attack (images on the diagonal are the original samples). We can see that in the adversarially trained model, the attacks need to introduce



(a) Baseline (without adversarial training)

(b) Adversarially trained

Figure-A I-6 Adversarial examples obtained using the C&W L_2 attack on two models: (a) Baseline, (b) model adversarially trained with our attack.

much larger changes to the samples in order to make them adversarial, and some of the adversarial samples visually resemble another class.

Figure-A I-7 shows randomly-selected adversarial examples for the CIFAR-10 dataset, comparing the baseline model (WRN 28-10), the Madry defense and our proposed defense. For each image and model, we ran three attacks (DDN 1 000, C&W 9×10 000, DeepFool 100), and present the adversarial example with minimum L_2 perturbation among them. Figure-A I-8 shows cherry-picked adversarial examples on CIFAR-10, that visually resemble another class, when attacking the proposed defense. We see that on the average case (randomly-selected), adversarial examples against the defenses still require low amounts of noise (perceptually) to induce misclassification. On the other hand, we notice that on adversarially trained models, some examples do require a much larger change on the image, making it effectively resemble another class.

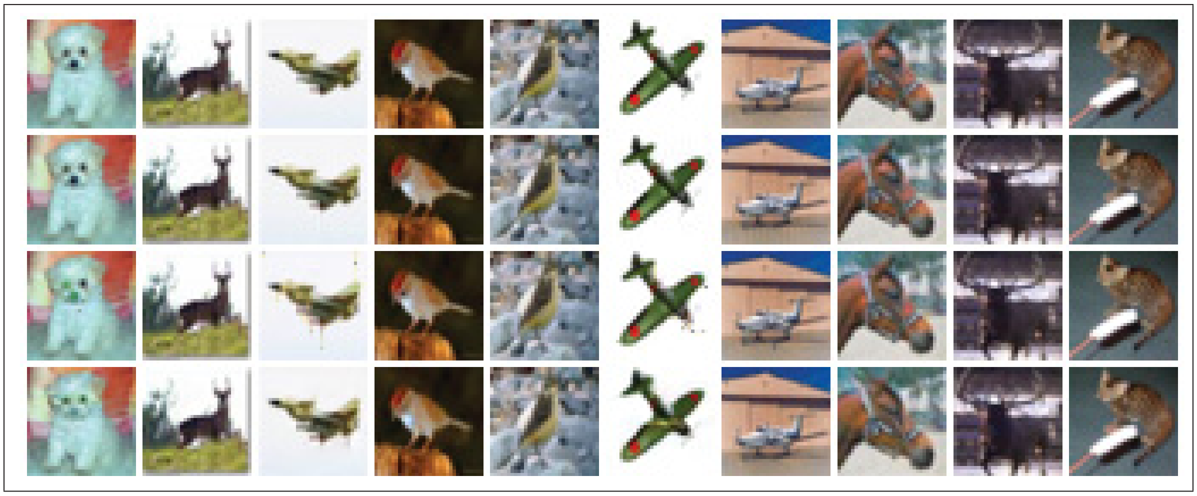


Figure-A I-7 Randomly chosen adversarial examples on CIFAR-10 for three models. **Top row**: original images; **second row**: attacks against the baseline; **third row**: attacks against the Madry defense.

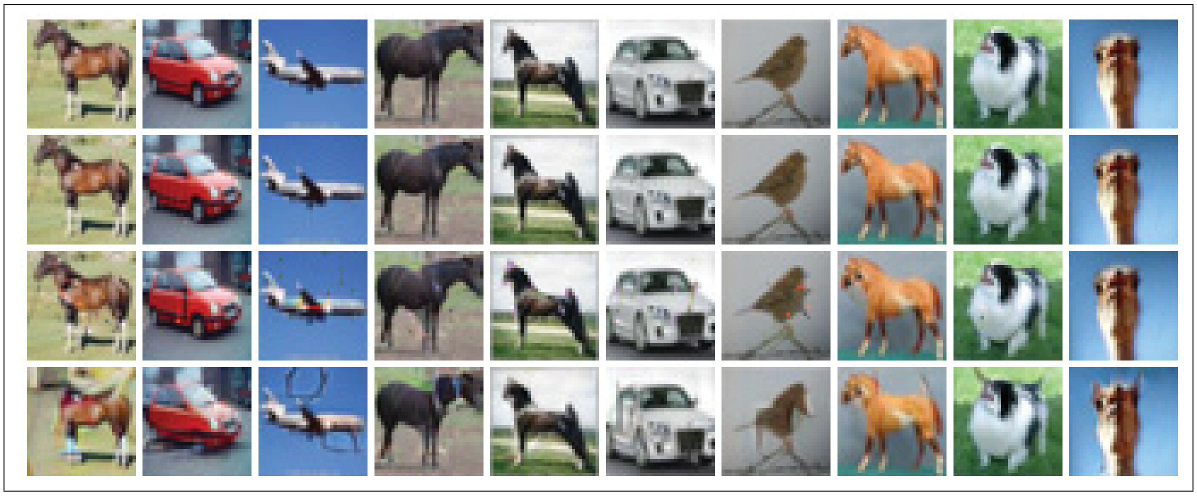
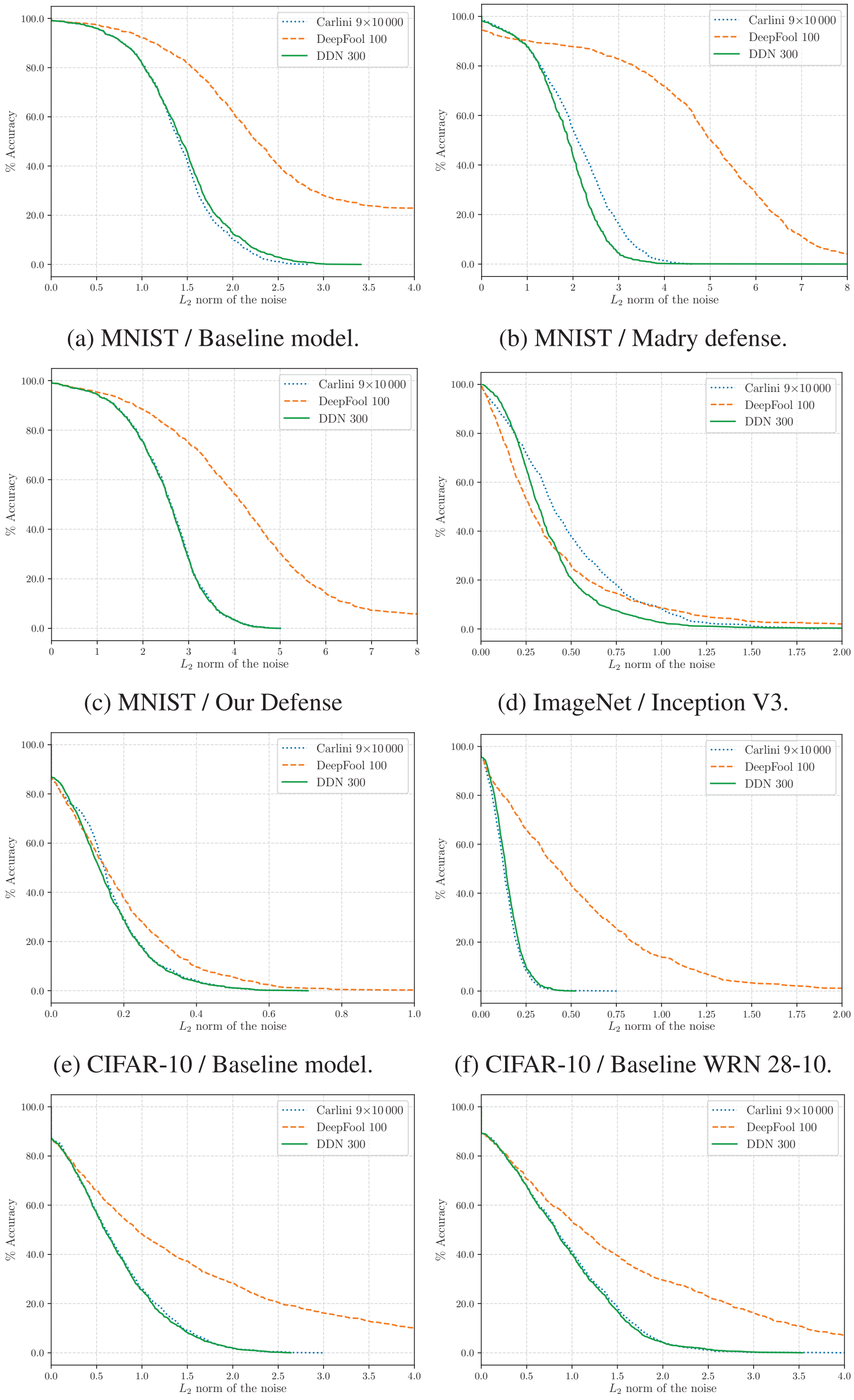


Figure-A I-8 Cherry-picked adversarial examples on CIFAR-10 for three models. **Top row**: original images; **second row**: attacks against the baseline; **third row**: attacks against the Madry defense; **bottom row**: attacks against the proposed defense. Predicted labels for the last row are, from left to right: dog, ship, deer, dog, dog, truck, horse, dog, cat, cat.

12. Attack performance curves

Figure-A I-9 reports curves of the perturbation size against accuracy of the models for three attacks: Carlini 9×10 000, DeepFool 100 and DDN 300.



(a) MNIST / Baseline model.

(b) MNIST / Madry defense.

(c) MNIST / Our Defense

(d) ImageNet / Inception V3.

(e) CIFAR-10 / Baseline model.

(f) CIFAR-10 / Baseline WRN 28-10.

(g) CIFAR-10 / Madry defense.

(h) CIFAR-10 / Our Defense.

Figure-A I-9 Attacks performances on different datasets and models.

APPENDIX II

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED CHARACTERIZING AND EVALUATING ADVERSARIAL EXAMPLES FOR OFFLINE HANDWRITTEN SIGNATURE VERIFICATION

1. CNN architectures

In this paper, we used the SigNet architecture (Hafemann *et al.* (2017a)) for the CNN-based experiments. This architecture is listed in Table II-1. Additionally, as base models for the Ensemble Adversarial Training (Tramèr *et al.* (2018)), we trained two models based on similar architectures: SigNet-thin, that has a smaller amount of channels in the convolutional layers (described in Table II-2) and SigNet-smaller that has less layers (described in Table II-3). In all cases, *M* refer to the number of users (531 in the PK and LK1 experiments, and 264 in the LK2 experiments).

Table-A II-1 SigNet architecture

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	M	

Table-A II-2 SigNet-thin architecture

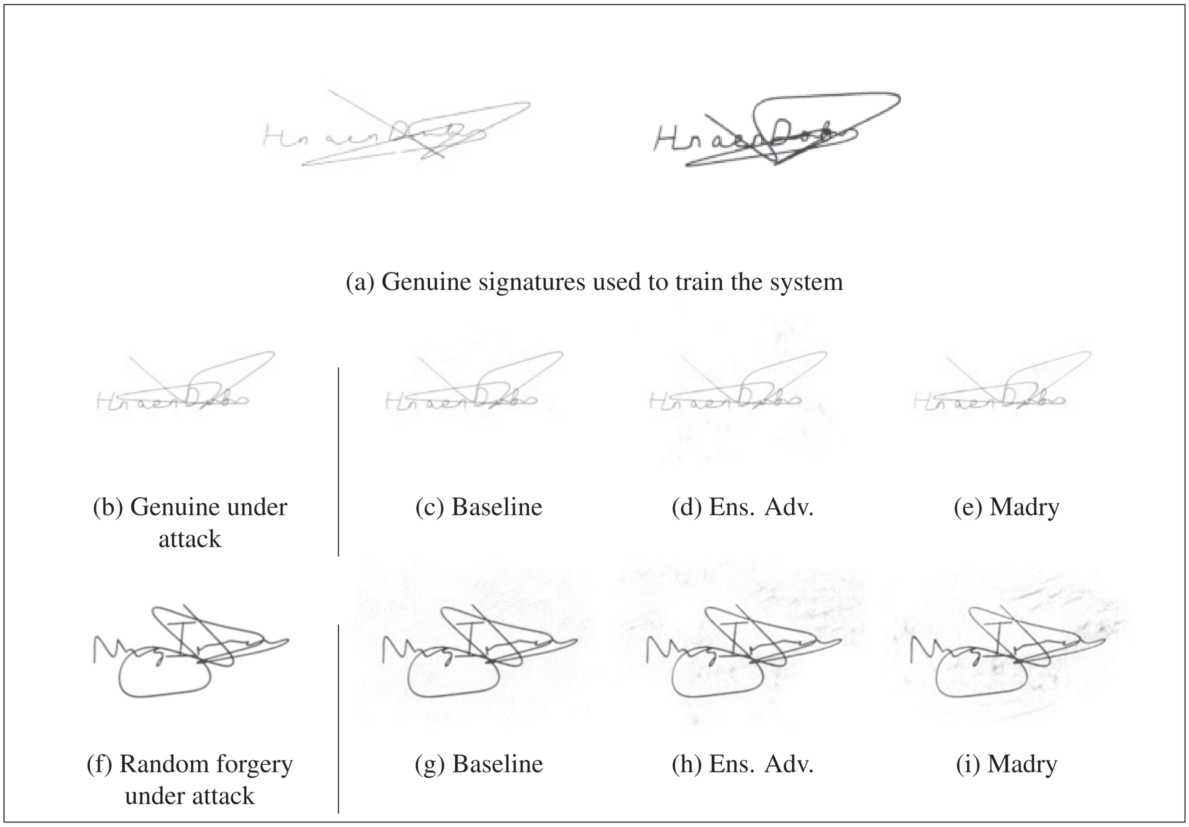
Layer	Size	Other Parameters
Input	1x150x220	
Convolution (C1)	96x11x11	stride = 4, pad=0
Pooling	96x3x3	stride = 2
Convolution (C2)	128x5x5	stride = 1, pad=2
Pooling	128x3x3	stride = 2
Convolution (C3)	128x3x3	stride = 1, pad=1
Convolution (C4)	128x3x3	stride = 1, pad=1
Convolution (C5)	128x3x3	stride = 1, pad=1
Pooling	128x3x3	stride = 2
Fully Connected (FC6)	1024	
Fully Connected (FC7)	1024	
Fully Connected + Softmax	M	

Table-A II-3 SigNet-smaller architecture

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)	256x3x3	stride = 1, pad=1
Pooling	256x3x3	stride = 2
Fully Connected (FC5)	2048	
Fully Connected + Softmax	M	

2. Visualizing adversarial examples

Figures II-1 to II-4 show adversarial examples for different users and defenses. For these visualizations, we considered the first 160 users of the GPDS Synthetic dataset (Ferrer *et al.* (2015)). We followed the experimental protocol defined in Chapter 5, using the three CNN models to extract the features, and training WD classifiers using 5 signatures as positive samples, and 5 signatures from the other users as negative samples. We then generated adversarial examples in a Perfect Knowledge scenario.



(a) Genuine signatures used to train the system

(b) Genuine under attack

(c) Baseline

(d) Ens. Adv.

(e) Madry

(f) Random forgery under attack

(g) Baseline

(h) Ens. Adv.

(i) Madry

Figure-A II-1 Adversarial examples against user 2, considering three systems: baseline (no defense), Ens. Adv training and Madry. **Top**: Two signatures used to train the classifier. **Middle**: attacking a genuine signature to be classified as forgery. **Bottom**: attacking a random forgery to be classified as genuine.

For each figure, we consider attacks against three models, that used: (i) the baseline model, (ii) the CNN trained with Ensemble Adversarial training, (iii) the CNN trained with the Madry defense. Figure II-1 consider attacks against user 2 of the dataset: the first row shows two reference signatures from this user, that were used to train the classifier. The middle row shows Type-I attacks, making a signature from this user be classified as forgery, for each of the three target systems; the bottom row shows Type-II attacks, making a random forgery be accepted as genuine for this user. We observe that, in general, Type-II attacks require much larger amounts of noise, which is reflected in a higher RMSE for these attacks. We also observe that attacking a system with the Madry defense often requires a noticeable amount of noise than the baseline. Lastly, we observe that for some users (e.g. figure II-1), the attacks have imperceptible amounts of noise, even when defenses are considered.

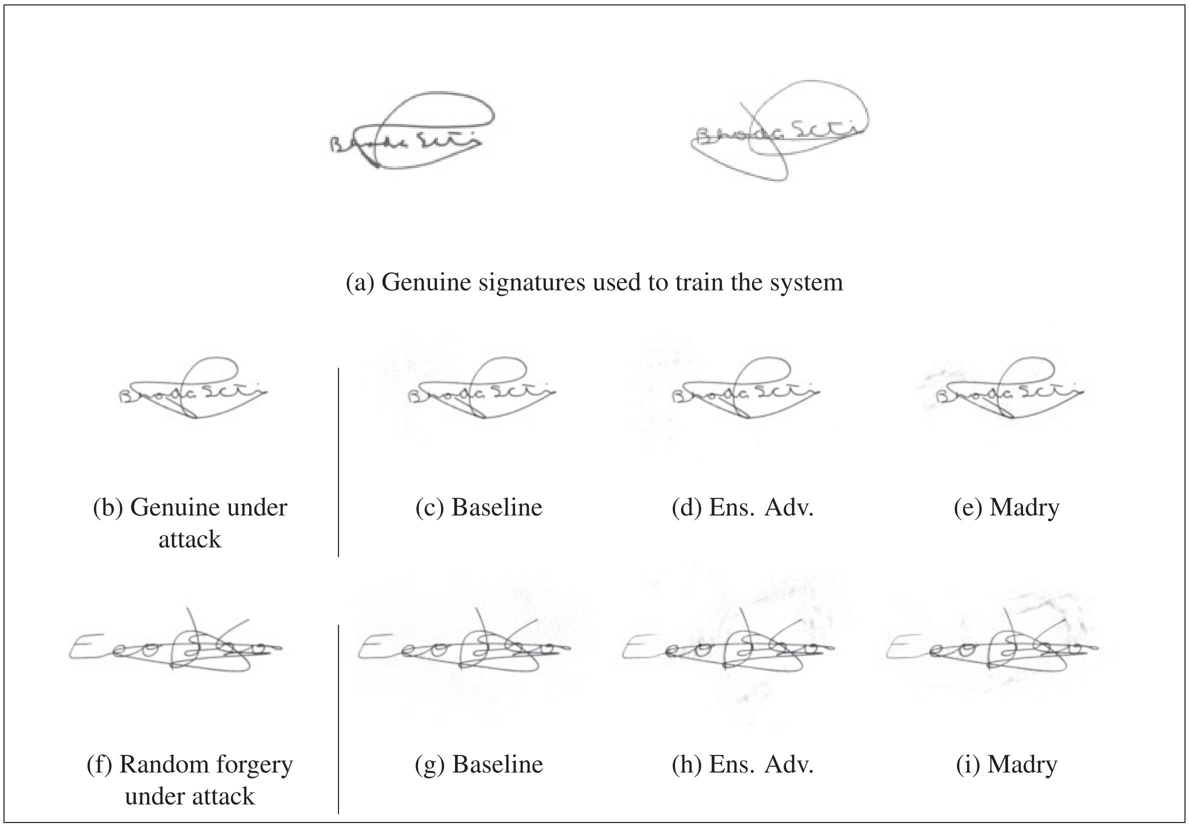


Figure-A II-2 Adversarial examples against user 4, considering three systems: baseline (no defense), Ens. Adv training and Madry. **Top**: Two signatures used to train the classifier. **Middle**: attacking a genuine signature to be classified as forgery. **Bottom**: attacking a random forgery to be classified as genuine.

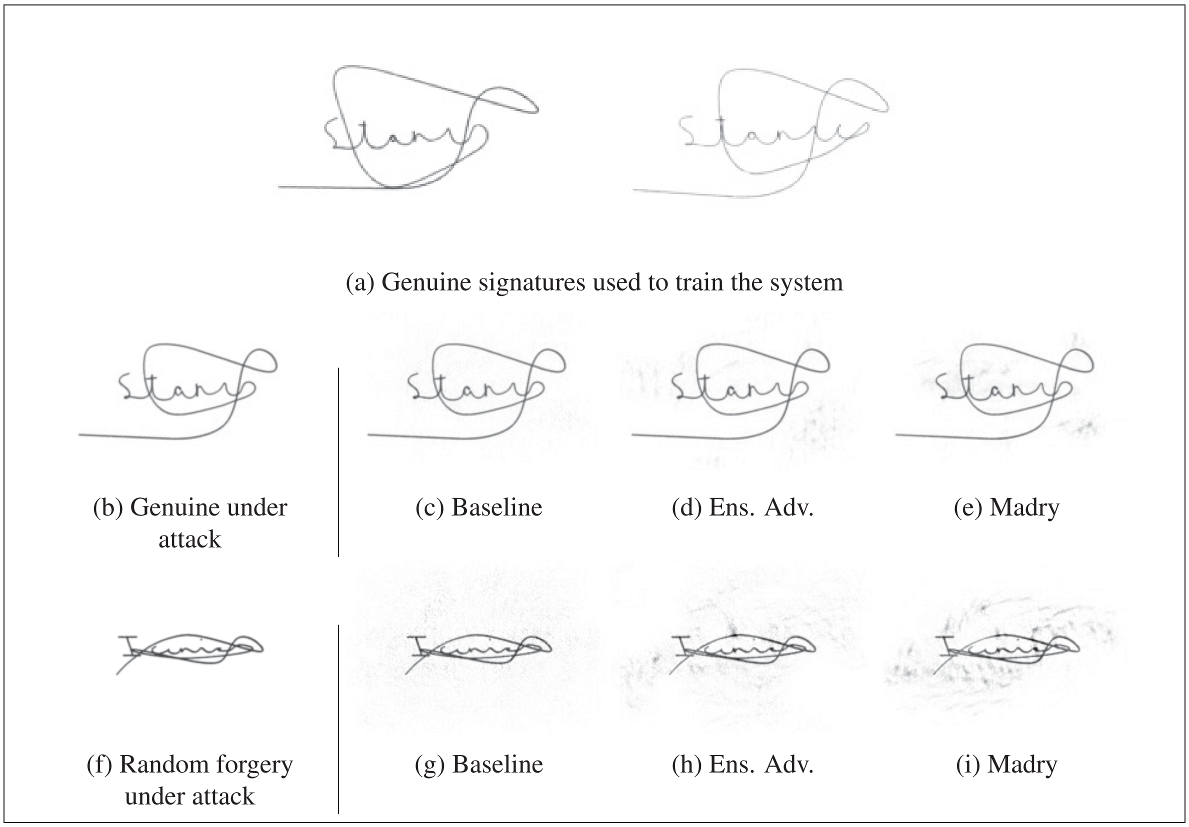


Figure-A II-3 Adversarial examples against user 16, considering three systems: baseline (no defense), Ens. Adv training and Madry. **Top**: Two signatures used to train the classifier. **Middle**: attacking a genuine signature to be classified as forgery. **Bottom**: attacking a random forgery to be classified as genuine.

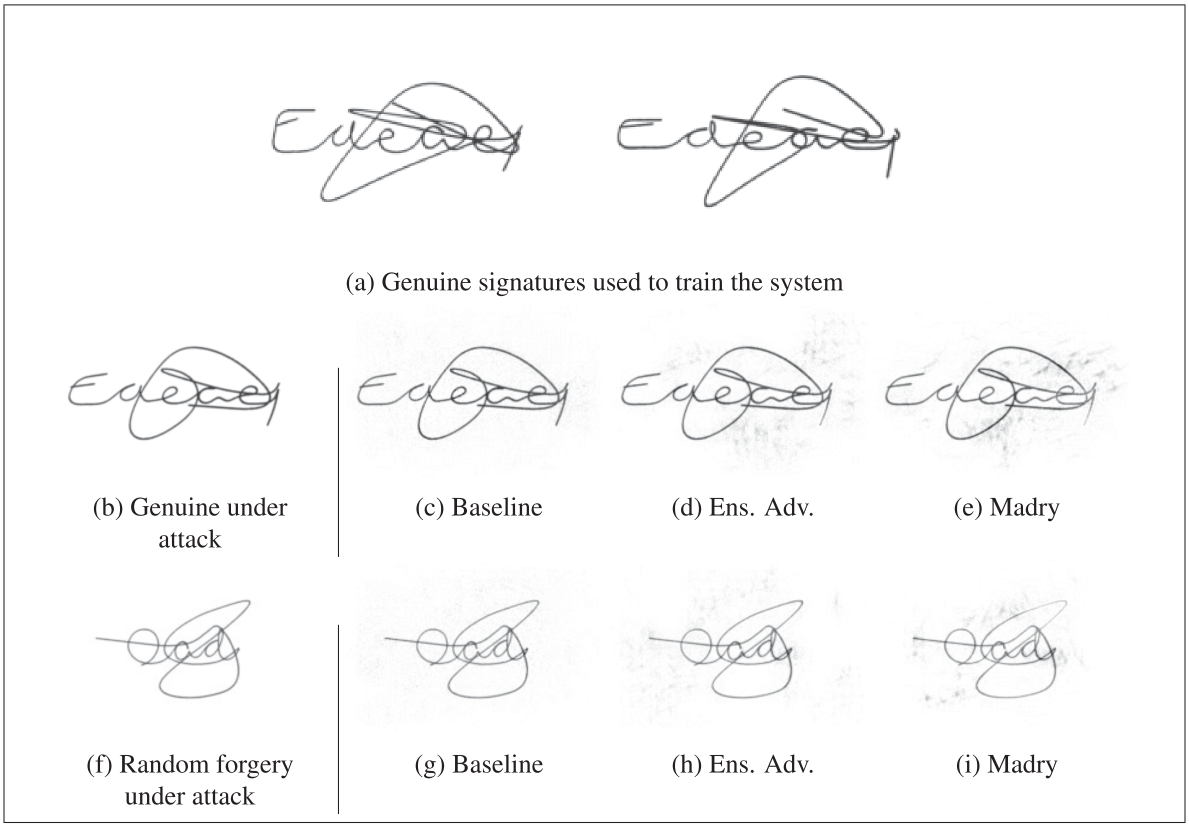


Figure-A II-4 Adversarial examples against user 25, considering three systems: baseline (no defense), Ens. Adv training and Madry. **Top**: Two signatures used to train the classifier. **Middle**: attacking a genuine signature to be classified as forgery. **Bottom**: attacking a random forgery to be classified as genuine.

3. Results on each dataset

Results on the MCYT dataset for different knowledge scenarios are shown in Tables II-4 to II-11. Results with countermeasures are shown in tables II-12 to II-15. Results with noise removal are shown in tables II-16 and II-17.

Results on the CEDAR dataset for different knowledge scenarios are shown in Tables II-18 to II-25. Results with countermeasures are shown in tables II-26 to II-29. Results with noise removal are shown in tables II-30 and II-31.

Results on the Brazilian PUC-PR dataset for different knowledge scenarios are shown in Tables II-32 to II-39. Results with countermeasures are shown in tables II-40 to II-43. Results with noise removal are shown in tables II-44 and II-45.

Results on the GPDS dataset for different knowledge scenarios are shown in Tables II-46 to II-53. Results with countermeasures are shown in tables II-54 to II-57. Results with noise removal are shown in tables II-58 and II-59.

Table-A II-4 Success rate of Type-I attacks on the MCYT dataset(% of attacks that transformed a genuine signature in a forgery)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	55.56	75.00
CLBP	RBF	-	-	100.00	100.00
SigNet	Linear	97.22	100.00	97.22	100.00
SigNet	RBF	91.67	100.00	97.22	100.00

Table-A II-5 Distortion (RMSE of the adversarial noise) for successful Type-I attacks on the MCYT dataset

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	0.36	2.34
CLBP	RBF	-	-	0.36	10^{-9}
SigNet	Linear	4.19	2.08	7.37	4.43
SigNet	RBF	4.22	2.17	6.80	4.16

Table-A II-6 Success rate of Type-II attacks on the MCYT dataset (% of attacks that transformed a forgery in a genuine signature)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	45.95	48.65
		skilled	-	-	45.95	48.65
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	100.00	0.00	0.00
		skilled	40.54	100.00	2.70	2.70
SigNet	RBF	random	0.00	100.00	0.00	0.00
		skilled	29.73	100.00	2.70	2.70

Table-A II-7 Distortion (RMSE of the adversarial noise) for successful Type-II attacks on the MCYT dataset

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	0.45	0.47
		skilled	-	-	0.53	0.67
SigNet	Linear	random	-	7.78	-	-
		skilled	4.29	4.13	1.72	0.52
SigNet	RBF	random	-	8.53	-	-
		skilled	4.24	4.54	0.94	0.27

Table-A II-8 Success rate of Type-I attacks on the MCYT dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	38.89	41.67
CLBP	RBF	-	-	77.78	77.78
SigNet	Linear	88.89	80.56	52.78	44.44
SigNet	RBF	88.89	86.11	55.56	50.00

Table-A II-9 Success rate of Type-II attacks on the MCYT dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	29.73	35.14
		skilled	-	-	27.03	29.73
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	54.05	0.00	0.00
		skilled	21.62	78.38	2.70	0.00
SigNet	RBF	random	0.00	78.38	0.00	0.00
		skilled	24.32	86.49	2.70	0.00

Table-A II-10 Success rate of Type-I attacks on the MCYT dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge #2)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	37.84	8.11	40.54	18.92
SigNet	RBF	43.24	10.81	40.54	18.92

Table-A II-11 Success rate of Type-II attacks on the MCYT dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge #2)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	random	0.00	0.00	0.00	0.00
		skilled	0.00	2.70	0.00	0.00
SigNet	RBF	random	0.00	0.00	0.00	0.00
		skilled	5.41	2.70	0.00	0.00

Table-A II-12 Success rate of Type-I attacks on the MCYT dataset considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
Baseline	Linear	100.00	86.49	37.84	100.00	75.68	8.11
	RBF	100.00	89.19	43.24	100.00	81.08	10.81
Ens. Adv.	Linear	70.27	64.86	16.22	100.00	89.19	8.11
	RBF	72.97	59.46	16.22	100.00	86.49	5.41
Madry	Linear	67.57	45.95	8.11	100.00	78.38	2.70
	RBF	56.76	48.65	8.11	100.00	86.49	0.00

Table-A II-13 Distortion (RMSE of the adversarial noise) for Type-I attacks on the MCYT dataset, considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	4.25	4.29	4.46	2.03	2.05	2.22
	RBF	4.29	4.32	4.49	2.16	2.15	2.52
ensadv	Linear	4.58	4.52	4.38	2.19	2.33	2.05
	RBF	4.55	4.50	4.34	2.35	2.48	2.23
madry	Linear	4.88	4.90	4.05	4.94	4.96	1.95
	RBF	4.87	4.90	4.07	5.26	5.54	-

Table-A II-14 Success rate of Type-II attacks on the MCYT dataset considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			**Carlini**		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	random	0.00	0.00	0.00	100.00	51.35	0.00
		skilled	35.14	32.43	0.00	100.00	67.57	2.70
	RBF	random	0.00	0.00	0.00	100.00	70.27	0.00
		skilled	35.14	29.73	5.41	100.00	83.78	2.70
ensadv	Linear	random	0.00	0.00	0.00	97.30	45.95	0.00
		skilled	29.73	13.51	0.00	100.00	78.38	0.00
	RBF	random	0.00	0.00	0.00	97.30	70.27	0.00
		skilled	24.32	21.62	0.00	100.00	83.78	0.00
madry	Linear	random	0.00	0.00	0.00	100.00	54.05	0.00
		skilled	35.14	27.03	2.70	100.00	78.38	2.70
	RBF	random	0.00	0.00	0.00	100.00	78.38	0.00
		skilled	35.14	27.03	5.41	100.00	94.59	5.41

Table-A II-15 Distortion (RMSE of the adversarial noise) for Type-II attacks on the MCYT dataset, considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			Carlini		
Defense	Classifier	Forgery Type	PK	LK1	LK2	PK	LK1	LK2
baseline	Linear	random	-	-	-	7.02	7.00	-
		skilled	4.41	4.42	-	3.69	3.97	3.56
	RBF	random	-	-	-	7.89	8.28	-
		skilled	4.39	4.45	4.75	4.22	4.39	2.26
ensadv	Linear	random	-	-	-	12.41	11.93	-
		skilled	4.49	4.41	-	6.77	7.11	-
	RBF	random	-	-	-	13.05	13.66	-
		skilled	4.51	4.49	-	7.68	7.23	-
madry	Linear	random	-	-	-	15.01	15.83	-
		skilled	4.91	4.99	4.31	8.91	9.16	3.73
	RBF	random	-	-	-	15.79	16.14	-
		skilled	4.94	5.00	4.10	9.85	9.90	3.68

Table-A II-16 Success of Type-I attacks on the MCYT dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

		Attack Type and Preprocessing							
		FGM		Carlini		Anneal		Decision	
Feature	Classifier	None	OTSU	None	OTSU	None	OTSU	None	OTSU
CLBP	Linear	-	-	-	-	55.56	8.33	75.00	2.78
	RBF	-	-	-	-	100.00	0.00	100.00	0.00
SigNet Baseline	Linear	100.00	67.57	100.00	21.62	94.59	0.00	100.00	0.00
	RBF	100.00	62.16	100.00	24.32	97.30	0.00	100.00	0.00
SigNet Ens. Adv.	Linear	70.27	43.24	100.00	0.00	91.89	0.00	100.00	0.00
	RBF	72.97	35.14	100.00	0.00	94.59	0.00	100.00	0.00
SigNet Madry	Linear	67.57	59.46	100.00	56.76	64.86	0.00	97.30	2.70
	RBF	56.76	51.35	100.00	59.46	67.57	0.00	100.00	2.70

Table-A II-17 Success of Type-II attacks on the MCYT dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

			Attack Type and Preprocessing							
			FGM		Carlini		Anneal		Decision	
Feature	Classifier	Forgery Type	None	OTSU	None	OTSU	None	OTSU	None	OTSU
CLBP	Linear	random	-	-	-	-	45.95	0.00	48.65	0.00
		skilled	-	-	-	-	45.95	0.00	48.65	2.70
	RBF	random	-	-	-	-	0.00	0.00	0.00	0.00
		skilled	-	-	-	-	0.00	0.00	0.00	0.00
SigNet Baseline	Linear	random	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
		skilled	35.14	24.32	100.00	16.22	0.00	0.00	0.00	0.00
	RBF	random	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
		skilled	35.14	24.32	100.00	16.22	0.00	0.00	0.00	0.00
SigNet Ens. Adv.	Linear	random	0.00	0.00	97.30	10.81	0.00	0.00	5.41	0.00
		skilled	29.73	13.51	100.00	21.62	0.00	0.00	5.41	0.00
	RBF	random	0.00	0.00	97.30	10.81	0.00	0.00	0.00	0.00
		skilled	24.32	10.81	100.00	27.03	0.00	0.00	0.00	0.00
SigNet Madry	Linear	random	0.00	0.00	100.00	56.76	0.00	0.00	5.41	0.00
		skilled	35.14	35.14	100.00	75.68	0.00	0.00	5.41	2.70
	RBF	random	0.00	0.00	100.00	72.97	0.00	0.00	0.00	0.00
		skilled	35.14	29.73	100.00	81.08	0.00	0.00	0.00	0.00

Table-A II-18 Success rate of Type-I attacks on the CEDAR dataset(% of attacks that transformed a genuine signature in a forgery)

		Attack Type			
Feature	Classifier	FGM	Carlini	Anneal	Decision
CLBP	Linear	-	-	70.37	85.19
CLBP	RBF	-	-	100.00	100.00
SigNet	Linear	100.00	100.00	100.00	100.00
SigNet	RBF	100.00	100.00	100.00	100.00

Table-A II-19 Distortion (RMSE of the adversarial noise) for successful Type-I attacks on the CEDAR dataset

		Attack Type			
Feature	Classifier	FGM	Carlini	Anneal	Decision
CLBP	Linear	-	-	0.36	0.69
CLBP	RBF	-	-	0.36	10^{-9}
SigNet	Linear	3.83	0.85	4.69	2.43
SigNet	RBF	3.81	0.88	3.99	2.09

Table-A II-20 Success rate of Type-II attacks on the CEDAR dataset (% of attacks that transformed a forgery in a genuine signature)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	25.93	44.44
		skilled	-	-	34.62	46.15
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	100.00	0.00	0.00
		skilled	30.77	100.00	3.85	7.69
SigNet	RBF	random	0.00	100.00	0.00	0.00
		skilled	26.92	100.00	0.00	0.00

Table-A II-21 Distortion (RMSE of the adversarial noise) for successful Type-II attacks on the CEDAR dataset

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	0.35	0.72
		skilled	-	-	0.38	1.12
SigNet	Linear	random	-	3.38	-	-
		skilled	3.96	2.16	1.30	0.88
SigNet	RBF	random	-	4.05	-	-
		skilled	3.73	2.58	-	-

Table-A II-22 Success rate of Type-I attacks on the CEDAR dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	33.33	33.33
CLBP	RBF	-	-	85.19	85.19
SigNet	Linear	100.00	77.78	62.96	40.74
SigNet	RBF	100.00	96.30	62.96	48.15

Table-A II-23 Success rate of Type-II attacks on the CEDAR dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge)

			Attack Type			
Features	Classifier	Forgery Type	FGM	Carlini	Anneal	Decision
CLBP	Linear	random	-	-	18.52	14.81
		skilled	-	-	15.38	15.38
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	37.04	0.00	0.00
		skilled	30.77	65.38	3.85	0.00
SigNet	RBF	random	0.00	70.37	0.00	0.00
		skilled	23.08	76.92	0.00	0.00

Table-A II-24 Success rate of Type-I attacks on the CEDAR dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge #2)

		Attack Type			
Feature	Classifier	FGM	Carlini	Anneal	Decision
SigNet	Linear	70.37	3.70	55.56	25.93
SigNet	RBF	74.07	11.11	59.26	22.22

Table-A II-25 Success rate of Type-II attacks on the CEDAR dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge #2)

			Attack Type			
Features	Classifier	Forgery Type	FGM	Carlini	Anneal	Decision
SigNet	Linear	random	0.00	0.00	0.00	0.00
		skilled	0.00	0.00	3.70	3.70
SigNet	RBF	random	0.00	0.00	0.00	0.00
		skilled	0.00	3.70	7.41	0.00

Table-A II-26 Success rate of Type-I attacks on the CEDAR dataset considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			Carlini		
Defense	Classifier	PK	LK1	LK2	PK	LK1	LK2
baseline	Linear	100.00	96.30	70.37	100.00	74.07	3.70
	RBF	100.00	100.00	74.07	100.00	81.48	11.11
ensadv	Linear	100.00	100.00	70.37	100.00	77.78	3.70
	RBF	100.00	100.00	70.37	100.00	96.30	3.70
madry	Linear	100.00	96.30	33.33	100.00	77.78	0.00
	RBF	100.00	100.00	25.93	100.00	92.59	0.00

Table-A II-27 Distortion (RMSE of the adversarial noise) for Type-I attacks on the CEDAR dataset, considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	3.93	3.99	4.07	0.86	0.91	0.85
	RBF	3.96	3.97	4.07	0.94	0.92	0.97
ensadv	Linear	4.22	4.09	4.03	0.82	0.86	1.10
	RBF	4.20	4.15	4.00	0.90	0.84	1.02
madry	Linear	4.49	4.43	4.14	2.01	2.10	-
	RBF	4.54	4.45	4.31	2.34	2.30	-

Table-A II-28 Success rate of Type-II attacks on the CEDAR dataset considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			**Carlini**		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	random	11.11	7.41	0.00	100.00	37.04	0.00
		skilled	48.15	25.93	0.00	100.00	48.15	0.00
	RBF	random	11.11	7.41	0.00	100.00	70.37	0.00
		skilled	40.74	37.04	0.00	100.00	74.07	3.70
ensadv	Linear	random	7.41	0.00	0.00	100.00	40.74	0.00
		skilled	29.63	11.11	0.00	100.00	62.96	7.41
	RBF	random	11.11	0.00	0.00	100.00	66.67	0.00
		skilled	22.22	14.81	0.00	100.00	77.78	3.70
madry	Linear	random	7.41	0.00	0.00	100.00	40.74	0.00
		skilled	59.26	44.44	22.22	100.00	62.96	7.41
	RBF	random	3.70	0.00	0.00	100.00	81.48	0.00
		skilled	55.56	37.04	11.11	100.00	81.48	7.41

Table-A II-29 Distortion (RMSE of the adversarial noise) for Type-II attacks on the CEDAR dataset, considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			Carlini		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	random	3.93	3.75	-	3.50	4.49	-
		skilled	4.03	3.85	-	2.04	1.60	-
	RBF	random	3.84	3.83	-	4.11	4.24	-
		skilled	3.97	3.93	-	2.40	2.33	0.77
ensadv	Linear	random	4.43	-	-	4.82	5.00	-
		skilled	4.40	4.65	-	3.13	2.47	1.11
	RBF	random	4.28	-	-	5.87	4.83	-
		skilled	4.46	4.46	-	3.76	3.13	1.82
madry	Linear	random	4.55	-	-	5.66	5.73	-
		skilled	4.88	5.00	4.26	3.71	3.20	1.32
	RBF	random	4.09	-	-	6.45	6.34	-
		skilled	4.87	4.94	4.26	4.71	4.42	1.47

Table-A II-30 Success of Type-I attacks on the CEDAR dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

		Attack Type and Preprocessing							
		FGM		Carlini		Anneal		Decision	
Feature	**Classifier**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	-	-	-	-	70.37	7.41	85.19	0.00
	RBF	-	-	-	-	100.00	0.00	100.00	0.00
SigNet Baseline	Linear	100.00	100.00	100.00	11.11	100.00	0.00	100.00	0.00
	RBF	100.00	100.00	100.00	14.81	96.30	0.00	100.00	0.00
SigNet Ens. Adv.	Linear	100.00	81.48	100.00	0.00	100.00	0.00	100.00	0.00
	RBF	100.00	81.48	100.00	0.00	100.00	0.00	100.00	0.00
SigNet Madry	Linear	100.00	100.00	100.00	85.19	100.00	0.00	100.00	3.70
	RBF	100.00	100.00	100.00	81.48	100.00	0.00	100.00	0.00

Table-A II-31 Success of Type-II attacks on the CEDAR dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

			Attack Type and Preprocessing							
			FGM		**Carlini**		**Anneal**		**Decision**	
Feature	**Classifier**	**Forgery Type**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	random	-	-	-	-	25.93	0.00	44.44	0.00
		skilled	-	-	-	-	34.62	0.00	46.15	3.85
	RBF	random	-	-	-	-	0.00	0.00	0.00	0.00
		skilled	-	-	-	-	0.00	0.00	0.00	0.00
SigNet Baseline	Linear	random	11.11	3.70	100.00	0.00	0.00	0.00	0.00	0.00
		skilled	48.15	40.74	100.00	14.81	3.70	0.00	3.70	0.00
	RBF	random	11.11	3.70	100.00	0.00	0.00	0.00	0.00	0.00
		skilled	40.74	29.63	100.00	22.22	7.41	0.00	7.41	0.00
SigNet Ens. Adv.	Linear	random	7.41	3.70	100.00	3.70	0.00	0.00	0.00	0.00
		skilled	29.63	22.22	100.00	22.22	0.00	0.00	0.00	0.00
	RBF	random	11.11	3.70	100.00	14.81	0.00	0.00	0.00	0.00
		skilled	22.22	11.11	100.00	33.33	0.00	0.00	0.00	0.00
SigNet Madry	Linear	random	7.41	3.70	100.00	25.93	0.00	0.00	0.00	0.00
		skilled	59.26	55.56	100.00	81.48	3.70	0.00	3.70	0.00
	RBF	random	3.70	3.70	100.00	59.26	0.00	0.00	0.00	0.00
		skilled	55.56	55.56	100.00	85.19	3.70	0.00	3.70	0.00

Table-A II-32 Success rate of Type-I attacks on the Brazilian PUC-PR dataset(% of attacks that transformed a genuine signature in a forgery)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	66.67	76.67
CLBP	RBF	-	-	100.00	100.00
SigNet	Linear	100.00	100.00	100.00	100.00
SigNet	RBF	100.00	100.00	100.00	100.00

Table-A II-33 Distortion (RMSE of the adversarial noise) for successful Type-I attacks on the Brazilian PUC-PR dataset

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	0.36	0.37
CLBP	RBF	-	-	0.36	10^{-9}
SigNet	Linear	3.92	1.03	5.10	2.75
SigNet	RBF	3.89	1.05	4.46	2.51

Table-A II-34 Success rate of Type-II attacks on the Brazilian PUC-PR dataset (% of attacks that transformed a forgery in a genuine signature)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	33.33	36.67
		skilled	-	-	33.33	36.67
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	6.67	100.00	0.00	0.00
		skilled	33.33	100.00	0.00	3.33
SigNet	RBF	random	3.33	100.00	0.00	0.00
		skilled	10.00	100.00	3.33	3.33

Table-A II-35 Distortion (RMSE of the adversarial noise) for successful Type-II attacks on the Brazilian PUC-PR dataset

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	0.36	0.60
		skilled	-	-	0.36	0.19
SigNet	Linear	random	4.11	4.63	-	-
		skilled	4.26	2.01	-	0.27
SigNet	RBF	random	4.70	5.70	-	-
		skilled	4.19	2.45	3.92	1.30

Table-A II-36 Success rate of Type-I attacks on the Brazilian PUC-PR dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	50.00	53.33
CLBP	RBF	-	-	90.00	90.00
SigNet	Linear	100.00	80.00	50.00	43.33
SigNet	RBF	100.00	96.67	56.67	53.33

Table-A II-37 Success rate of Type-II attacks on the Brazilian PUC-PR dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	26.67	33.33
		skilled	-	-	23.33	26.67
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	50.00	0.00	0.00
		skilled	23.33	83.33	0.00	0.00
SigNet	RBF	random	0.00	83.33	0.00	0.00
		skilled	10.00	93.33	0.00	0.00

Table-A II-38 Success rate of Type-I attacks on the Brazilian PUC-PR dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge #2)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	80.00	10.00	43.33	13.33
SigNet	RBF	83.33	10.00	56.67	20.00

Table-A II-39 Success rate of Type-II attacks on the Brazilian PUC-PR dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge #2)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	random	0.00	0.00	0.00	0.00
		skilled	10.00	10.00	0.00	0.00
SigNet	RBF	random	0.00	0.00	0.00	0.00
		skilled	0.00	3.33	0.00	0.00

Table-A II-40 Success rate of Type-I attacks on the Brazilian PUC-PR dataset considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	100.00	100.00	80.00	100.00	86.67	10.00
	RBF	100.00	100.00	83.33	100.00	96.67	10.00
ensadv	Linear	100.00	96.67	76.67	100.00	80.00	3.33
	RBF	100.00	100.00	83.33	100.00	96.67	3.33
madry	Linear	100.00	93.33	33.33	100.00	83.33	3.33
	RBF	100.00	100.00	36.67	100.00	100.00	0.00

Table-A II-41 Distortion (RMSE of the adversarial noise) for Type-I attacks on the Brazilian PUC-PR dataset, considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	4.09	4.06	4.24	0.96	0.97	1.18
	RBF	4.09	4.00	4.11	1.03	1.04	1.27
ensadv	Linear	4.28	4.26	4.15	0.94	0.92	1.51
	RBF	4.31	4.35	4.12	0.98	0.97	1.59
madry	Linear	4.57	4.54	4.39	2.16	2.35	0.83
	RBF	4.57	4.64	4.18	2.41	2.55	-

Table-A II-42 Success rate of Type-II attacks on the Brazilian PUC-PR dataset considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			**Carlini**		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	random	3.33	0.00	0.00	100.00	46.67	0.00
		skilled	50.00	30.00	10.00	100.00	76.67	10.00
	RBF	random	0.00	0.00	0.00	100.00	80.00	0.00
		skilled	30.00	26.67	0.00	100.00	93.33	3.33
ensadv	Linear	random	3.33	3.33	0.00	100.00	53.33	0.00
		skilled	46.67	20.00	3.33	100.00	60.00	6.67
	RBF	random	3.33	0.00	0.00	100.00	96.67	0.00
		skilled	23.33	16.67	3.33	100.00	100.00	3.33
madry	Linear	random	3.33	3.33	0.00	96.67	50.00	0.00
		skilled	76.67	60.00	10.00	100.00	76.67	3.33
	RBF	random	10.00	3.33	0.00	96.67	93.33	0.00
		skilled	66.67	56.67	10.00	100.00	93.33	0.00

Table-A II-43 Distortion (RMSE of the adversarial noise) for Type-II attacks on the Brazilian PUC-PR dataset, considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			**Carlini**		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	random	4.63	-	-	4.47	5.13	-
		skilled	4.25	4.06	4.21	1.86	1.79	2.05
	RBF	random	-	-	-	5.31	4.95	-
		skilled	4.01	3.85	-	2.25	2.18	2.29
ensadv	Linear	random	4.66	4.82	-	6.83	7.46	-
		skilled	4.53	4.60	3.86	2.86	2.84	1.51
	RBF	random	4.76	-	-	8.27	8.16	-
		skilled	4.62	4.60	3.84	3.89	3.69	2.45
madry	Linear	random	5.12	5.38	-	8.23	8.69	-
		skilled	4.89	4.86	3.95	3.88	3.69	1.30
	RBF	random	4.79	5.28	-	9.41	9.67	-
		skilled	4.86	4.74	4.27	4.96	4.73	-

Table-A II-44 Success of Type-I attacks on the Brazilian PUC-PR dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

		Attack Type and Preprocessing							
		FGM		**Carlini**		**Anneal**		**Decision**	
Feature	**Classifier**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	-	-	-	-	66.67	13.33	76.67	3.33
	RBF	-	-	-	-	100.00	0.00	100.00	0.00
SigNet Baseline	Linear	100.00	100.00	100.00	10.00	90.00	0.00	100.00	6.67
	RBF	100.00	96.67	100.00	23.33	100.00	0.00	96.67	0.00
SigNet Ens. Adv.	Linear	100.00	93.33	100.00	6.67	96.67	0.00	100.00	0.00
	RBF	100.00	93.33	100.00	3.33	96.67	0.00	100.00	0.00
SigNet Madry	Linear	100.00	100.00	100.00	90.00	93.33	0.00	100.00	6.67
	RBF	100.00	100.00	100.00	86.67	96.67	0.00	100.00	6.67

Table-A II-45 Success of Type-II attacks on the Brazilian PUC-PR dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

			Attack Type and Preprocessing							
			FGM		**Carlini**		**Anneal**		**Decision**	
Feature	**Classifier**	**Forgery Type**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	random	-	-	-	-	33.33	0.00	36.67	0.00
		skilled	-	-	-	-	33.33	0.00	36.67	0.00
	RBF	random	-	-	-	-	0.00	0.00	0.00	0.00
		skilled	-	-	-	-	0.00	0.00	0.00	0.00
SigNet Baseline	Linear	random	3.33	0.00	100.00	0.00	0.00	0.00	0.00	0.00
		skilled	50.00	36.67	100.00	13.33	3.33	0.00	3.33	0.00
	RBF	random	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
		skilled	30.00	26.67	100.00	16.67	3.33	0.00	3.33	0.00
SigNet Ens. Adv.	Linear	random	3.33	3.33	100.00	3.33	0.00	0.00	0.00	0.00
		skilled	46.67	43.33	100.00	20.00	3.33	0.00	6.67	0.00
	RBF	random	3.33	3.33	100.00	16.67	0.00	0.00	0.00	0.00
		skilled	23.33	23.33	100.00	36.67	3.33	0.00	3.33	0.00
SigNet Madry	Linear	random	3.33	3.33	96.67	36.67	0.00	0.00	0.00	0.00
		skilled	76.67	60.00	100.00	73.33	0.00	0.00	3.33	0.00
	RBF	random	10.00	6.67	96.67	60.00	0.00	0.00	0.00	0.00
		skilled	66.67	56.67	100.00	90.00	0.00	0.00	0.00	0.00

Table-A II-46 Success rate of Type-I attacks on the GPDS-160 dataset(% of attacks that transformed a genuine signature in a forgery)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	62.82	83.33
CLBP	RBF	-	-	100.00	100.00
SigNet	Linear	100.00	100.00	98.72	100.00
SigNet	RBF	100.00	100.00	98.72	100.00

Table-A II-47 Distortion (RMSE of the adversarial noise) for successful Type-I attacks on the GPDS-160 dataset

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	0.46	1.99
CLBP	RBF	-	-	0.36	10^{-9}
SigNet	Linear	4.09	1.30	5.52	3.22
SigNet	RBF	4.13	1.36	5.12	3.00

Table-A II-48 Success rate of Type-II attacks on the GPDS-160 dataset (% of attacks that transformed a forgery in a genuine signature)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	38.75	48.75
		skilled	-	-	38.75	48.75
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	92.50	0.00	0.00
		skilled	21.25	98.75	2.50	2.50
SigNet	RBF	random	0.00	88.75	0.00	0.00
		skilled	16.25	100.00	1.25	1.25

Table-A II-49 Distortion (RMSE of the adversarial noise) for successful Type-II attacks on the GPDS-160 dataset

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	0.37	1.80
		skilled	-	-	0.39	1.51
SigNet	Linear	random	-	6.78	-	-
		skilled	4.18	3.54	5.71	2.74
SigNet	RBF	random	-	6.83	-	-
		skilled	4.11	3.98	4.66	1.96

Table-A II-50 Success rate of Type-I attacks on the GPDS-160 dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	-	-	44.87	44.87
CLBP	RBF	-	-	80.77	80.77
SigNet	Linear	98.72	80.77	46.15	37.18
SigNet	RBF	100.00	91.03	50.00	44.87

Table-A II-51 Success rate of Type-II attacks on the GPDS-160 dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
CLBP	Linear	random	-	-	23.75	28.75
		skilled	-	-	21.25	27.50
CLBP	RBF	random	-	-	0.00	0.00
		skilled	-	-	0.00	0.00
SigNet	Linear	random	0.00	45.00	0.00	0.00
		skilled	20.00	66.25	1.25	0.00
SigNet	RBF	random	0.00	71.25	0.00	0.00
		skilled	18.75	80.00	0.00	0.00

Table-A II-52 Success rate of Type-I attacks on the GPDS dataset (% of attacks that transformed a genuine signature in a forgery) (Limited Knowledge #2)

		Attack Type			
Feature	**Classifier**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	60.00	6.25	52.50	20.00
SigNet	RBF	63.75	7.50	51.25	17.50

Table-A II-53 Success rate of Type-II attacks on the GPDS dataset (% of attacks that transformed a forgery in a genuine signature) (Limited Knowledge #2)

			Attack Type			
Features	**Classifier**	**Forgery Type**	**FGM**	**Carlini**	**Anneal**	**Decision**
SigNet	Linear	random	0.00	0.00	0.00	0.00
		skilled	1.25	0.00	0.00	0.00
SigNet	RBF	random	0.00	0.00	0.00	0.00
		skilled	1.25	0.00	0.00	0.00

Table-A II-54 Success rate of Type-I attacks on the GPDS dataset considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			Carlini		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	100.00	97.50	60.00	100.00	77.50	6.25
	RBF	100.00	100.00	63.75	100.00	85.00	7.50
ensadv	Linear	95.00	86.25	38.75	100.00	76.25	3.75
	RBF	91.25	83.75	38.75	100.00	87.50	7.50
madry	Linear	96.25	92.50	22.50	100.00	68.75	1.25
	RBF	96.25	93.75	21.25	100.00	85.00	1.25

Table-A II-55 Distortion (RMSE of the adversarial noise) for Type-I attacks on the GPDS dataset, considering different defenses and attacker knowledge scenarios

		Attack Type and Knowledge scenario					
		FGM			**Carlini**		
Defense	**Classifier**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	4.25	4.27	4.37	1.27	1.29	1.07
	RBF	4.28	4.32	4.42	1.34	1.33	1.32
ensadv	Linear	4.39	4.33	4.31	1.29	1.33	2.02
	RBF	4.38	4.33	4.36	1.37	1.36	1.53
madry	Linear	4.88	4.85	4.28	3.17	3.26	1.99
	RBF	4.90	4.85	4.34	3.46	3.45	2.19

Table-A II-56 Success rate of Type-II attacks on the GPDS dataset considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			**Carlini**		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	random	1.25	0.00	0.00	97.50	38.75	0.00
		skilled	37.50	28.75	1.25	100.00	68.75	0.00
	RBF	random	0.00	0.00	0.00	91.25	62.50	0.00
		skilled	33.75	23.75	1.25	100.00	83.75	0.00
ensadv	Linear	random	0.00	0.00	0.00	86.25	35.00	0.00
		skilled	22.50	15.00	1.25	100.00	60.00	2.50
	RBF	random	0.00	0.00	0.00	86.25	60.00	0.00
		skilled	22.50	16.25	1.25	100.00	80.00	2.50
madry	Linear	random	0.00	0.00	0.00	97.50	42.50	0.00
		skilled	41.25	33.75	5.00	100.00	73.75	2.50
	RBF	random	0.00	0.00	0.00	96.25	66.25	0.00
		skilled	40.00	35.00	3.75	100.00	85.00	1.25

Table-A II-57 Distortion (RMSE of the adversarial noise) for Type-II attacks on the GPDS dataset, considering different defenses and attacker knowledge scenarios

			Attack Type and Knowledge scenario					
			FGM			Carlini		
Defense	**Classifier**	**Forgery Type**	**PK**	**LK1**	**LK2**	**PK**	**LK1**	**LK2**
baseline	Linear	random	3.40	-	-	6.92	5.58	-
		skilled	4.17	4.12	4.32	3.40	2.78	-
	RBF	random	-	-	-	6.64	6.06	-
		skilled	4.07	3.99	4.41	3.60	3.32	-
ensadv	Linear	random	-	-	-	8.88	9.24	-
		skilled	4.59	4.61	4.32	4.98	3.89	1.67
	RBF	random	-	-	-	9.48	9.33	-
		skilled	4.66	4.66	4.31	5.54	4.74	2.14
madry	Linear	random	-	-	-	11.55	10.81	-
		skilled	4.90	4.92	4.10	6.62	5.87	1.99
	RBF	random	-	-	-	11.98	11.97	-
		skilled	4.95	4.91	3.98	7.22	6.63	1.67

Table-A II-58 Success of Type-I attacks on the GPDS dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

		Attack Type and Preprocessing							
		FGM		Carlini		Anneal		Decision	
Feature	**Classifier**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	-	-	-	-	62.82	8.97	83.33	5.13
	RBF	-	-	-	-	100.00	1.28	100.00	0.00
SigNet Baseline	Linear	100.00	90.00	100.00	22.50	98.75	1.25	100.00	1.25
	RBF	100.00	88.75	100.00	23.75	98.75	0.00	98.75	1.25
SigNet Ens. Adv.	Linear	95.00	63.75	100.00	3.75	100.00	0.00	100.00	0.00
	RBF	91.25	63.75	100.00	2.50	100.00	0.00	100.00	0.00
SigNet Madry	Linear	96.25	92.50	100.00	78.75	92.50	0.00	100.00	10.00
	RBF	96.25	95.00	100.00	77.50	91.25	0.00	100.00	8.75

Table-A II-59 Success of Type-II attacks on the GPDS dataset in a PK scenario, with no pre-processing and with OTSU pre-processing

			Attack Type and Preprocessing							
			FGM		**Carlini**		**Anneal**		**Decision**	
Feature	**Classifier**	**Forgery Type**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**	**None**	**OTSU**
CLBP	Linear	random	-	-	-	-	38.75	1.25	48.75	0.00
		skilled	-	-	-	-	38.75	3.75	48.75	0.00
	RBF	random	-	-	-	-	0.00	0.00	0.00	0.00
		skilled	-	-	-	-	0.00	0.00	0.00	0.00
SigNet Baseline	Linear	random	1.25	0.00	97.50	0.00	0.00	0.00	0.00	0.00
		skilled	37.50	28.75	100.00	8.75	0.00	0.00	0.00	0.00
	RBF	random	0.00	0.00	91.25	0.00	0.00	0.00	0.00	0.00
		skilled	33.75	21.25	100.00	10.00	0.00	0.00	0.00	0.00
SigNet Ens. Adv.	Linear	random	0.00	0.00	86.25	8.75	0.00	0.00	1.25	0.00
		skilled	22.50	20.00	100.00	22.50	0.00	0.00	1.25	1.25
	RBF	random	0.00	0.00	86.25	11.25	0.00	0.00	0.00	0.00
		skilled	22.50	15.00	100.00	22.50	0.00	0.00	0.00	0.00
SigNet Madry	Linear	random	0.00	0.00	97.50	50.00	0.00	0.00	1.25	0.00
		skilled	41.25	36.25	100.00	77.50	0.00	0.00	1.25	0.00
	RBF	random	0.00	0.00	96.25	60.00	0.00	0.00	0.00	0.00
		skilled	40.00	33.75	100.00	83.75	0.00	0.00	0.00	0.00

BIBLIOGRAPHY

Alaei, A., Pal, s., Pal, U. & Blumenstein, M. (2017). An Efficient Signature Verification Method based on an Interval Symbolic Representation and a Fuzzy Similarity Measure. *IEEE Transactions on Information Forensics and Security*, PP(99), 1–1. doi: 10.1109/TIFS.2017.2707332.

Antoniou, A., Edwards, H. & Storkey, A. (2019). How to train your MAML. *International Conference on Learning Representations*.

Athalye, A. & Carlini, N. (2018). On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses. *The Bright and Dark Sides of Computer Vision: Challenges and Opportunities for Privacy and Security (CVPR workshop)*.

Athalye, A., Carlini, N. & Wagner, D. (2018a, 10–15 Jul). Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *Proceedings of the 35th International Conference on Machine Learning*, 80, 274–283.

Athalye, A., Engstrom, L., Ilyas, A. & Kwok, K. (2018b, 10–15 Jul). Synthesizing Robust Adversarial Examples. *Proceedings of the 35th International Conference on Machine Learning*, 80, 284–293.

Baker, B., Gupta, O., Naik, N. & Raskar, R. (2017). Designing Neural Network Architectures using Reinforcement Learning. *International Conference on Learning Representations*.

Baltzakis, H. & Papamarkos, N. (2001). A new signature verification technique based on a two-stage neural network classifier. *Engineering applications of Artificial intelligence*, 14(1), 95–103.

Barreno, M., Nelson, B., Sears, R., Joseph, A. D. & Tygar, J. D. (2006). Can machine learning be secure? *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25.

Barreno, M., Nelson, B., Joseph, A. D. & Tygar, J. D. (2010). The security of machine learning. *Machine Learning*, 81(2), 121–148.

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

Bengio, Y., Bengio, S. & Cloutier, J. (1991). Learning a synaptic learning rule. *IJCNN-91-Seattle International Joint Conference on Neural Networks*, ii, 969 vol.2–. doi: 10.1109/IJCNN.1991.155621.

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

Bengio, Y. (2013). Deep Learning of Representations: Looking Forward. In *Statistical Language and Speech Processing* (pp. 1–37). Springer Berlin Heidelberg.

Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2007). Greedy Layer-Wise Training of Deep Networks. In Schölkopf, B., Platt, J. C. & Hoffman, T. (Eds.), *Advances in Neural Information Processing Systems 19* (pp. 153–160). MIT Press.

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

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

Biggio, B., fumera, g., Russu, P., Didaci, L. & Roli, F. (2015). Adversarial Biometric Recognition : A review on biometric system security from the adversarial machine-learning perspective. *IEEE Signal Processing Magazine*, 32(5), 31–41. doi: 10.1109/MSP.2015.2426728.

Biggio, B. & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 317–331. doi: 10.1016/j.patcog.2018.07.023.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G. & Roli, F. (2013). Evasion Attacks against Machine Learning at Test Time. *Machine Learning and Knowledge Discovery in Databases*, pp. 387–402.

Biggio, B., Fumera, G. & Roli, F. (2014). Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4), 984–996.

Brendel, W., Rauber, J. & Bethge, M. (2018a). Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *International Conference on Learning Representations*.

Brendel, W., Rauber, J., Kurakin, A., Papernot, N., Veliqi, B., Salathé, M., Mohanty, S. P. & Bethge, M. (2018b). Adversarial Vision Challenge. *arXiv:1808.01976*.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. & Shah, R. (1994). Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, pp. 737–744.

Buckman, J., Roy, A., Raffel, C. & Goodfellow, I. (2018). Thermometer Encoding: One Hot Way To Resist Adversarial Examples. *International Conference on Learning Representations*.

Carlini, N. & Wagner, D. (2017a). Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*.

Carlini, N. & Wagner, D. (2017b). Towards evaluating the robustness of neural networks. *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57.

Chatfield, K., Simonyan, K., Vedaldi, A. & Zisserman, A. (2014). Return of the Devil in the Details: Delving Deep into Convolutional Nets. *British Machine Vision Conference, BMVC 2014, Nottingham, UK, September 1-5, 2014*.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J. & Hsieh, C.-J. (2017). ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks Without Training Substitute Models. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26. doi: 10.1145/3128572.3140448.

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

Coetzer, J. (2005). *Off-line signature verification*. (Ph.D. thesis, Stellenbosch: University of Stellenbosch).

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1, 886–893.

Dalvi, N., Domingos, P., Mausam, Sanghai, S. & Verma, D. (2004). Adversarial Classification. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (KDD '04), 99–108. doi: 10.1145/1014052.1014066.

Deng, P. S., Liao, H.-Y. M., Ho, C. W. & Tyan, H.-R. (1999). Wavelet-Based Off-Line Handwritten Signature Verification. *Computer Vision and Image Understanding*, 76(3), 173–190. doi: 10.1006/cviu.1999.0799.

Dhillon, G. S., Azizzadenesheli, K., Lipton, Z. C., Bernstein, J., Kossaifi, J., Khanna, A. & Anandkumar, A. (2018). Stochastic Activation Pruning for Robust Adversarial Defense. *International Conference on Learning Representations*.

Diaz, M., Ferrer, M. A., Eskander, G. S. & Sabourin, R. (2017). Generation of Duplicated Off-Line Signature Images for Verification Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5), 951–964. doi: 10.1109/TPAMI.2016.2560810.

Dimauro, G., Impedovo, S., Pirlo, G. & Salzo, A. (1997). A multi-expert signature verification system for bankcheck processing. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(05), 827–844.

Djeziri, S., Nouboud, F. & Plamondon, R. (1998). Extraction of signatures from check background based on a filiformity criterion. *IEEE Transactions on Image Processing*, 7(10), 1425–1438. doi: 10.1109/83.718483.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. & Darrell, T. (2014). DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *International conference on machine learning*, pp. 647–655.

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

El-Yacoubi, A., Justino, E. J. R., Sabourin, R. & Bortolozzi, F. (2000). Off-line signature verification using HMMs and cross-validation. *Proceedings of the 2000 IEEE Signal Processing Society Workshop*, 2, 859–868 vol.2. doi: 10.1109/NNSP.2000.890166.

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

Ferrer, M. A., Alonso, J. B. & Travieso, C. M. (2005). Offline geometric parameters for automatic signature verification using fixed-point arithmetic. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6), 993–997. doi: 10.1109/TPAMI.2005.125.

Ferrer, M. A., Diaz-Cabrera, M. & Morales, A. (2015). Static Signature Synthesis: A Neuromotor Inspired Approach for Biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3), 667–680. doi: 10.1109/TPAMI.2014.2343981.

Ferrer, M. A., Diaz-Cabrera, M. & Morales, A. (2013). Synthetic off-line signature image generation. *Biometrics (ICB), 2013 International Conference on*, pp. 1–7.

Ferrer, M. A., Chanda, S., Diaz, M., Banerjee, C. K., Majumdar, A., Carmona-Duarte, C., Acharya, P. & Pal, U. (2017). Static and Dynamic Synthesis of Bengali and Devanagari Signatures. *IEEE transactions on cybernetics*.

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

Finn, C., Abbeel, P. & Levine, S. (2017, 06–11 Aug). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *Proceedings of the 34th International Conference on Machine Learning*, 70, 1126–1135.

Freitas, C., Morita, M., Oliveira, L., Justino, E., Yacoubi, A., Lethelier, E., Bortolozzi, F. & Sabourin, R. (2000). Bases de dados de cheques bancarios brasileiros. *XXVI Conferencia Latinoamericana de Informatica*.

Ghandali, S. & Moghaddam, M. (2008). A Method for Off-line Persian Signature Identification and Verification Using DWT and Image Fusion. *IEEE International Symposium on Signal Processing and Information Technology, ISSPIT*, pp. 315–319. doi: 10.1109/ISSPIT.2008.4775712.

Gilperez, A., Alonso-Fernandez, F., Pecharroman, S., Fierrez, J. & Ortega-Garcia, J. (2008). Off-line signature verification using contour features. *11th International Conference on Frontiers in Handwriting Recognition*.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Artificial Intelligence and Statistics, International conference on*, pp. 249–256.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27* (pp. 2672–2680).

Goodfellow, I. J., Shlens, J. & Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. *International Conference on Learning Representations*.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354–377. doi: 10.1016/j.patcog.2017.10.013.

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

Guo, C., Rana, M., Cissé, M. & van der Maaten, L. (2018). Countering Adversarial Images using Input Transformations. *International Conference on Learning Representations*.

Guo, Z., Zhang, L. & Zhang, D. (2010). A completed modeling of local binary pattern operator for texture classification. *Image Processing, IEEE Transactions on*, 19(6), 1657–1663.

Hafemann, L. G., Sabourin, R. & Oliveira, L. S. (2016a). Analyzing features learned for offline signature verification using Deep CNNs. *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pp. 2989–2994.

Hafemann, L. G., Sabourin, R. & Oliveira, L. S. (2016b). Writer-independent feature learning for Offline Signature Verification using Deep Convolutional Neural Networks. *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 2576–2583. doi: 10.1109/IJCNN.2016.7727521.

Hafemann, L. G., Sabourin, R. & Oliveira, L. S. (2017a). Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition*, 70, 163–176. doi: 10.1016/j.patcog.2017.05.012.

Hafemann, L. G., Sabourin, R. & Oliveira, L. S. (2017b). Offline handwritten signature verification—literature review. *Image Processing Theory, Tools and Applications (IPTA), 2017 Seventh International Conference on*, pp. 1–8.

Hafemann, L. G., Oliveira, L. S. & Sabourin, R. (2018). Fixed-sized representation learning from offline handwritten signatures of different sizes. *International Journal on Document Analysis and Recognition (IJDAR)*, 1–14. doi: 10.1007/s10032-018-0301-6.

Hafemann, L. G., Sabourin, R. & Oliveira, L. S. (2019). Characterizing and evaluating adversarial examples for Offline Handwritten Signature Verification. *IEEE Transactions on Information Forensics and Security*. doi: 10.1109/TIFS.2019.2894031.

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 1904–1916. doi: 10.1109/TPAMI.2015.2389824.

He, K., Zhang, X., Ren, S. & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. *European Conference on Computer Vision*, pp. 346–361.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

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

Huang, K. & Yan, H. (1997). Off-line signature verification based on geometric feature extraction and neural network classification. *Pattern Recognition*, 30(1), 9–17. doi: 10.1016/S0031-3203(96)00063-5.

Ilyas, A., Engstrom, L., Athalye, A. & Lin, J. (2018). Black-box Adversarial Attacks with Limited Queries and Information. *International Conference on Learning Representations*.

Impedovo, D. & Pirlo, G. (2008). Automatic Signature Verification: The State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5), 609–635. doi: 10.1109/TSMCC.2008.923866.

Ioffe, S. & Szegedy, C. (2015, 07–09 Jul). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning*, 37, 448–456.

Jain, A. K., Ross, A. & Prabhakar, S. (2004). An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1), 4–20.

Jain, A. K., Nandakumar, K. & Nagar, A. (2008). Biometric Template Security. *EURASIP J. Adv. Signal Process*, 2008, 113:1–113:17. doi: 10.1155/2008/579416.

Jensen, P. A. & Bard, J. F. a. (2003). *Operations Research Models and Methods*. Wiley.

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

Justino, E. J. R., Bortolozzi, F. & Sabourin, R. (2005). A comparison of SVM and HMM classifiers in the off-line signature verification. *Pattern Recognition Letters*, 26(9), 1377–1385. doi: 10.1016/j.patrec.2004.11.015.

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

Khalajzadeh, H., Mansouri, M. & Teshnehlab, M. (2012). Persian Signature Verification using Convolutional Neural Networks. *International Journal of Engineering Research and Technology*, 1.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25* (pp. 1097–1105).

Kumar, R., Kundu, L., Chanda, B. & Sharma, J. D. (2010). A Writer-independent Off-line Signature Verification System Based on Signature Morphology. *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia*, (IITM '10), 261–265. doi: 10.1145/1963564.1963610.

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

Kurakin, A., Goodfellow, I. & Bengio, S. (2017a). Adversarial examples in the physical world. *International Conference on Learning Representations (workshop track)*.

Kurakin, A., Goodfellow, I. & Bengio, S. (2017b). Adversarial Machine Learning at Scale. *International Conference on Learning Representations*.

Leclerc, F. & Plamondon, R. (1994). Automatic signature verification: the state of the art—1989–1993. *International Journal of Pattern Recognition and Artificial Intelligence*, 08(03), 643–660. doi: 10.1142/S0218001494000346.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4), 541–551. doi: 10.1162/neco.1989.1.4.541.

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

Maclaurin, D., Duvenaud, D. & Adams, R. (2015, 07–09 Jul). Gradient-based Hyperparameter Optimization through Reversible Learning. *Proceedings of the 32nd International Conference on Machine Learning*, 37, 2113–2122.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. *International Conference on Learning Representations*.

Malik, M. I., Liwicki, M., Dengel, A., Uchida, S. & Frinken, V. (2014). Automatic Signature Stability Analysis and Verification Using Local Features. *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pp. 621–626.

Moosavi-Dezfooli, S.-M., Fawzi, A. & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582.

Nagel, R. N. & Rosenfeld, A. (1977). Computer Detection of Freehand Forgeries. *IEEE Transactions on Computers*, C-26(9), 895–905. doi: 10.1109/TC.1977.1674937.

Nanni, L. & Ghidoni, S. (2017). 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, 1–7. doi: 10.1016/j.patrec.2016.11.011.

Nemcek, W. F. & Lin, W. C. (1974). Experimental Investigation of Automatic Signature Verification. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-4(1), 121–126. doi: 10.1109/TSMC.1974.5408537.

Ojala, T., Pietikainen, M. & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7), 971–987.

Oliveira, L. S., Justino, E., Freitas, C. & Sabourin, R. (2005). The graphology applied to signature verification. *12th Conference of the International Graphonomics Society*, pp. 286–290.

Ooi, S. Y., Teoh, A. B. J., Pang, Y. H. & Hiew, B. Y. (2016). Image-based handwritten signature verification using hybrid methods of discrete Radon transform, principal component analysis and probabilistic neural network. *Applied Soft Computing*, 40, 274–282. doi: 10.1016/j.asoc.2015.11.039.

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

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

Osuna, E., Freund, R. & Girosi, F. (1997). Support Vector Machines: Training and Applications.

Otsu, N. (1975). A threshold selection method from gray-level histograms. *Automatica*, 11(285-296), 23–27.

Pal, S., Chanda, S., Pal, U., Franke, K. & Blumenstein, M. (2012). Off-line signature verification using G-SURF. *Intelligent Systems Design and Applications, 12th International Conference on*, pp. 586–591.

Papernot, N., McDaniel, P., Wu, X., Jha, S. & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. *Security and Privacy, IEEE Symposium on*, pp. 582–597.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B. & Swami, A. (2017). Practical Black-Box Attacks Against Machine Learning. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, (ASIA CCS), 506–519. doi: 10.1145/3052973.3053009.

Perera, P. & Patel, V. M. (2018). Learning Deep Features for One-Class Classification. *arXiv:1801.05365 [cs].*

Plamondon, R. & Srihari, S. N. (2000). Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 63–84. doi: 10.1109/34.824821.

Plamondon, R. & Lorette, G. (1989). Automatic signature verification and writer identification—the state of the art. *Pattern recognition*, 22(2), 107–131.

Pourshahabi, M. R., Sigari, M. H. & Pourreza, H. R. (2009). Offline handwritten signature identification and verification using contourlet transform. *Soft Computing and Pattern Recognition, International Conference of*, pp. 670–673.

Ramanathan, A., Pullum, L., Husein, Z., Raj, S., Torosdagli, N., Pattanaik, S. & Jha, S. K. (2017). Adversarial attacks on computer vision algorithms using natural perturbations. *Contemporary Computing (IC3), 2017 Tenth International Conference on*, pp. 1–6.

Rantzsch, H., Yang, H. & Meinel, C. (2016). Signature Embedding: Writer Independent Offline Signature Verification with Deep Metric Learning. *Advances in Visual Computing*, (Lecture Notes in Computer Science), 616–625. doi: 10.1007/978-3-319-50832-0_60.

Ratha, N. K., Connell, J. H. & Bolle, R. M. (2001). An analysis of minutiae matching strength. *International Conference on Audio-and Video-Based Biometric Person Authentication*, pp. 223–228.

Rauber, J., Brendel, W. & Bethge, M. (2017). Foolbox: A Python toolbox to benchmark the robustness of machine learning models. *arXiv:1707.04131*.

Ravi, S. & Larochelle, H. (2017). Optimization as a model for few-shot learning. *International Conference on Learning Representations*.

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

Ribeiro, B., Gonçalves, I., Santos, S. & Kovacec, A. (2011). Deep Learning Networks for Off-Line Handwritten Signature Recognition. In Martin, C. S. & Kim, S.-W. (Eds.), *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (pp. 523–532). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-25085-9_62.

Rivard, D., Granger, E. & Sabourin, R. (2013). Multi-feature extraction and selection in writer-independent off-line signature verification. *International Journal on Document Analysis and Recognition (IJDAR)*, 16(1), 83–103. doi: 10.1007/s10032-011-0180-6.

Rony, J., Hafemann, L. G., Oliveira, L. S., Ayed, I. B., Sabourin, R. & Granger, E. (2018). Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses. *arXiv:1811.09600*.

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

Ruiz-del Solar, J., Devia, C., Loncomilla, P. & Concha, F. (2008). Offline Signature Verification Using Local Interest Points and Descriptors. In Ruiz-Shulcloper, J. & Kropatsch, W. G. (Eds.), *Progress in Pattern Recognition, Image Analysis and Applications* (pp. 22–29). Springer Berlin Heidelberg.

Sabourin, R. & Drouhard, J.-P. (1992). Off-line signature verification using directional PDF and neural networks. *, 11th IAPR International Conference on Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings*, pp. 321–325. doi: 10.1109/ICPR.1992.201782.

Sabourin, R. & Genest, G. (1994). An extended-shadow-code based approach for off-line signature verification. I. Evaluation of the bar mask definition. *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, 2, 450–453 vol.2. doi: 10.1109/ICPR.1994.576979.

Sabourin, R., Cheriet, M. & Genest, G. (1993). An extended-shadow-code based approach for off-line signature verification. , *Proceedings of the Second International Conference on Document Analysis and Recognition, 1993*, pp. 1–5. doi: 10.1109/ICDAR.1993.395795.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning*. (PhD Thesis, Technische Universität München, München).

Serdouk, Y., Nemmour, H. & Chibani, Y. (2014). Combination of OC-LBP and Longest Run Features for Off-Line Signature Verification. *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems (SITIS)*, pp. 84–88. doi: 10.1109/SITIS.2014.36.

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

Serdouk, Y., Nemmour, H. & Chibani, Y. (2015b). Orthogonal Combination and Rotation Invariant of Local Binary Patterns for Off-line Handwritten Signature Verification. *International Conference on Telecommunications and ICT*.

Shekar, B. H., Bharathi, R. K., Kittler, J., Vizilter, Y. & Mestestskiy, L. (2015). Grid structured morphological pattern spectrum for off-line signature verification. *International Conference on Biometrics*, pp. 430–435. doi: 10.1109/ICB.2015.7139106.

Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*.

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

Souza, V. L. F., Oliveira, A. L. I. & Sabourin, R. (2018). A Writer-Independent Approach for Offline Signature Verification using Deep Convolutional Neural Networks Features. *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 212–217. doi: 10.1109/BRACIS.2018.00044.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Tramèr, F., Kurakin, A., Papernot, N., Boneh, D. & McDaniel, P. (2018). Ensemble Adversarial Training: Attacks and Defenses. *International Conference on Learning Representations*.

Tsourounis, D., Theodorakopoulos, I., Zois, E. N., Economou, G. & Fotopoulos, S. (2018). Handwritten Signature Verification via Deep Sparse Coding Architecture. *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pp. 1–5. doi: 10.1109/IVMSPW.2018.8448687.

Van der Maaten, L. & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605), 85.

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

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

Vargas, J., Travieso, C., Alonso, J. & Ferrer, M. (2010). Off-line Signature Verification Based on Gray Level Information Using Wavelet Transform and Texture Features. *2010 International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 587–592. doi: 10.1109/ICFHR.2010.96.

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

Xie, C., Wang, J., Zhang, Z., Ren, Z. & Yuille, A. (2018). Mitigating Adversarial Effects Through Randomization. *International Conference on Learning Representations*.

Yilmaz, M. B., Yanikoglu, B., Tirkaz, C. & Kholmatov, A. (2011). Offline signature verification using classifier combination of HOG and LBP features. *Biometrics (IJCB), 2011 International Joint Conference on*, pp. 1–7.

Yılmaz, M. B. & Yanıkoğlu, B. (2016). Score level fusion of classifiers in off-line signature verification. *Information Fusion*, 32, Part B, 109–119. doi: 10.1016/j.inffus.2016.02.003.

Zagoruyko, S. & Komodakis, N. (2016). Wide Residual Networks. *Proceedings of the British Machine Vision Conference*, pp. 87.1-87.12. doi: 10.5244/C.30.87.

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

Zhang, Z., Liu, X. & Cui, Y. (2016). Multi-phase Offline Signature Verification System Using Deep Convolutional Generative Adversarial Networks. *2016 9th International Symposium on Computational Intelligence and Design (ISCID)*, 02, 103–107. doi: 10.1109/ISCID.2016.2033.

Zois, E. N., Theodorakopoulos, I., Tsourounis, D. & Economou, G. (2017). Parsimonious Coding and Verification of Offline Handwritten Signatures. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 636–645. doi: 10.1109/CVPRW.2017.92.

Zois, E. N., Papagiannopoulou, M., Tsourounis, D. & Economou, G. (2018a). Hierarchical Dictionary Learning and Sparse Coding for Static Signature Verification. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Zois, E. N., Tsourounis, D., Theodorakopoulos, I., Kesidis, A. & Economou, G. (2018b). A comprehensive study of sparse representation techniques for offline signature verification. *arXiv:1807.05039 [cs]*. arXiv: 1807.05039.

Zouari, R., Mokni, R. & Kherallah, M. (2014). Identification and verification system of offline handwritten signature using fractal approach. *Image Processing, Applications and Systems Conference (IPAS), 2014 First International*, pp. 1–4. doi: 10.1109/IPAS.2014.7043305.

Özgündüz, E., Şentürk, T. & Karslıgil, M. E. (2005). Off-line signature verification and recognition by support vector machine. *European signal processing conference, EUSIPCO*.