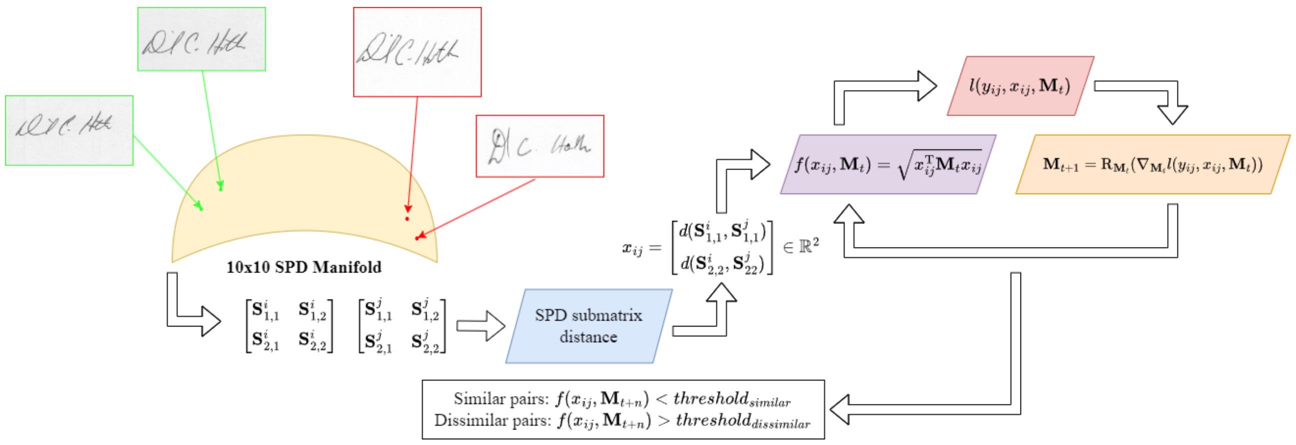
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Μέθοδοι υπολογιστική όρασης με χρήση μετα-ευρετικών αλγορίθμων βελτιστοποίησης στον χώρο πινάκων συνδιακύμανσης

Διπλωματική Εργασία



10x10 SPD Manifold
$\begin{bmatrix} S^i_{1,1} & S^i_{1,2} \\ S^i_{2,1} & S^i_{2,2} \end{bmatrix}$ $\begin{bmatrix} S^j_{1,1} & S^j_{1,2} \\ S^j_{2,1} & S^j_{2,2} \end{bmatrix}$
SPD submatrix distance
$x_{ij} = \begin{bmatrix} d(S^i_{1,1}, S^j_{1,1}) \\ d(S^i_{2,2}, S^j_{22}) \end{bmatrix} \in \mathbb{R}^2$
$f(x_{ij}, M_t) = \sqrt{x_{ij}^T M_t x_{ij}}$
$l(y_{ij}, x_{ij}, M_t)$
$M_{t+1} = R_{M_t}(\nabla_{M_t} l(y_{ij}, x_{ij}, M_t))$
Similar pairs: $f(x_{ij}, M_{t+n}) < threshold_{similar}$
Dissimilar pairs: $f(x_{ij}, M_{t+n}) > threshold_{dissimilar}$

Φοιτητής: Αλέξιος Γιαζιτζής
AM: 503262017050

Επιβλέπων Καθηγητής

Ηλίας Ζώης, Ph.D
Αναπληρωτής Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΙΟΥΛΙΟΣ 2023

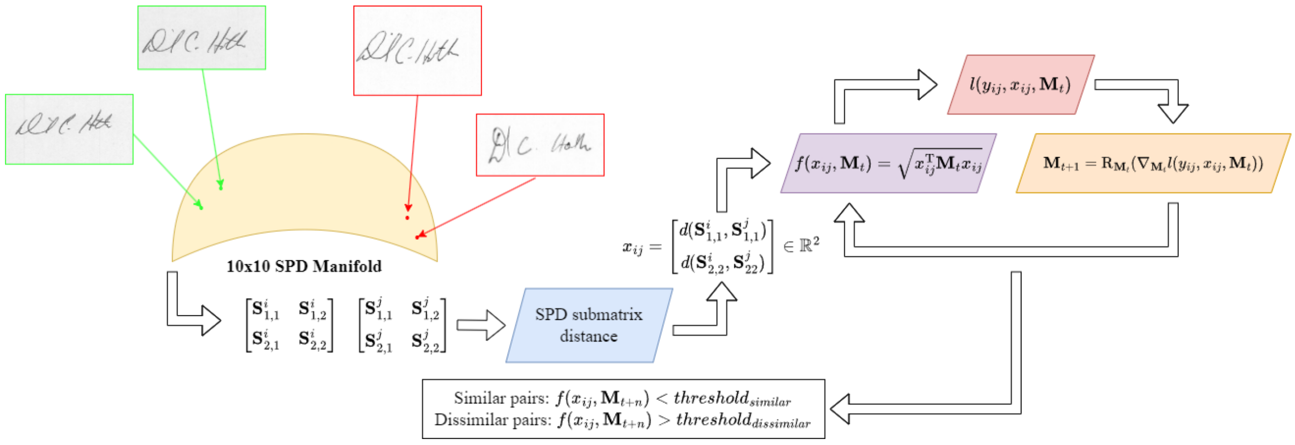
UNIVERSITY OF WEST ATTICA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Meta-heuristic optimization methods on SPD Manifolds and applications to computer vision

Diploma Thesis


10x10 SPD Manifold
$\begin{bmatrix} \mathbf{S}^i_{1,1} & \mathbf{S}^i_{1,2} \\ \mathbf{S}^i_{2,1} & \mathbf{S}^i_{2,2} \end{bmatrix}$ $\begin{bmatrix} \mathbf{S}^j_{1,1} & \mathbf{S}^j_{1,2} \\ \mathbf{S}^j_{2,1} & \mathbf{S}^j_{2,2} \end{bmatrix}$
SPD submatrix distance
$x_{ij} = \begin{bmatrix} d(\mathbf{S}^i_{1,1}, \mathbf{S}^j_{1,1}) \\ d(\mathbf{S}^i_{2,2}, \mathbf{S}^j_{22}) \end{bmatrix} \in \mathbb{R}^2$
$f(x_{ij}, \mathbf{M}_t) = \sqrt{x_{ij}^T \mathbf{M}_t x_{ij}}$
$l(y_{ij}, x_{ij}, \mathbf{M}_t)$
$\mathbf{M}_{t+1} = \mathrm{R}_{\mathbf{M}_t}(\nabla_{\mathbf{M}_t} l(y_{ij}, x_{ij}, \mathbf{M}_t))$
Similar pairs: $f(x_{ij}, \mathbf{M}_{t+n}) < threshold_{similar}$
Dissimilar pairs: $f(x_{ij}, \mathbf{M}_{t+n}) > threshold_{dissimilar}$

Student: Alexios Giazitzis
Registration Number: 503262017050

Supervisor

Elias Zois, Ph.D
Associate Professor

ATHENS-EGALEO, JULY 2023

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Ηλίας Ζώης, Αναπληρωτής Καθηγητής	Ευάγγελος Ζέρβας, Καθηγητής	Δημήτριος Καλύβας, Καθηγητής
Ilias Zois Digitally signed by Ilias Zois Date: 2023.07.24 14:12:18 +03'00' (Υπογραφή)	Evangelos Zervas Digitally signed by Evangelos Zervas Date: 2023.07.24 14:08:12 +03'00' (Υπογραφή)	Dimitrios Kalyvas Digitally signed by Dimitrios Kalyvas Date: 2023.07.24 11:02:15 +03'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Αλέξιος Γιαζιτζής του Ιωάννη, με αριθμό μητρώου 503262017050, φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.
Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι 12 μήνες και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος καθηγητή.»

Ilias Zois

Digitally signed by Ilias Zois
Date: 2023.07.24 10:32:40 +03'00'

Ο Δηλών
Αλέξιος Γιαζιτζής

Α. Γιαζιτζης

Θα ήθελα να ευχαριστήσω την οικογένεια μου, που σε όλα αυτά τα χρόνια με υποστήριξε με τον καλύτερο δυνατό τρόπο ώστε να καταφέρω να τελειώσω τις σπουδές μου αλλά και να καταφέρω όλους τους υπόλοιπους στόχους που είχα παράλληλα. Θα ήθελα επίσης να ευχαριστήσω τον κ. Ηλία Ζώη, που με καθοδηγούσε και με συμβούλευε, τόσο στην διπλωματική εργασία μου αλλά και σε ερευνητικά και μαθησιακά ενδιαφέροντα μεγαλύτερου επιπέδου.

Αλέξιος Γιαζιτζής,
Αθήνα,
Ιούλιος 2013

Περίληψη

Η επαλήθευση του κατά πόσον μια αμφισβητούμενη χειρόγραφη υπογραφή, που παρουσιάζεται σε ένα άνθρωπο ή ένα αλγόριθμο, ανήκει ή όχι σε μια διεκδικούμενη ταυτότητα είναι μια αρκετά δύσκολη εργασία, η οποία συνήθως απαιτεί προσεκτική εξέταση, δεξιότητες και γνώσεις. Ένα ακριβές και αποτελεσματικό σύστημα επαλήθευσης υπογραφών μπορεί να προσφέρει σημαντική βοήθεια, μειώνοντας τον χρόνο που απαιτείται για την ολοκλήρωση μιας χειροκίνητης εργασίας, η οποία συνήθως εκτελείται από δικαστικούς γραφολόγους – Forensic Handwriting Examiner. Προς την κατεύθυνση αυτή, η παρούσα διπλωματική εργασία προτείνει ένα πλαίσιο επαλήθευσης εικόνων, με περιεχόμενο χειρόγραφων υπογραφών, στο χώρο των συμμετρικών θετικά ορισμένων (SPD) πινάκων, χρησιμοποιώντας μια προσέγγιση μάθησης μιας μετρικής-απόστασης με τη χρήση ενός νευρωνικού δικτύου ειδικού σκοπού, του οποίου η λειτουργία διέπετε από μια μετα-ευρετική διαδικασία βελτιστοποίησης. Συγκεκριμένα, α) οι εικόνες των υπογραφών (στατική προσέγγιση) απεικονίζονται ως σημεία του Riemannian SPD χώρου - πολύπτυχου και β) εφαρμόζεται μια προσέγγιση μάθησης μιας μετρικής - απόστασης Mahalanobis, η οποία θα ελαχιστοποιεί την απόσταση δειγμάτων ίδιας προέλευσης και θα μεγιστοποιεί την απόσταση δειγμάτων διαφορετικής προέλευσης. Τα πειραματικά αποτελέσματα με τη χρήση δύο δημοφιλών συνόλων δεδομένων χειρόγραφων υπογραφών από τον επονομαζόμενο και ως δυτικό κόσμο, δείχνουν ότι το προτεινόμενο πλαίσιο είναι συγκρίσιμο τουλάχιστον με τα τρέχοντα μοντέλα αιχμής, τα οποία είθισται να υλοποιούνται υπό ένα πλαίσιο ευκλείδειας φύσης, κλειστό ως προς όλες τις γνωστές μας πράξεις.

Λέξεις – κλειδιά

Στατική Επαλήθευση Υπογραφών, Γεωμετρία Riemann, Νευρωνικά Δίκτυα, Βελτιστοποίηση, Συμμετρικοί Θετικά Οριζόμενοι Πίνακες, Επεξεργασίας Εικόνας, Υπολογιστική Όραση, Δυαδική Ταξινόμηση

Abstract

Verifying whether or not a questioned handwritten signature belongs to a claimed identity is a quite challenging task, usually requiring careful examination, skills and knowledge. An accurate and efficient signature verification system, can provide considerable assistance by reducing the time it takes to complete a manual task, usually performed by Forensic Handwriting Examiners. Towards this purpose, this thesis proposes an offline handwritten signature verification framework in the space of Symmetric Positive Definite matrices, by utilizing a metric learning approach which enables the use of a special purpose neural network, acting as a meta-heuristic optimization procedure. Specifically, the static signature images are mapped into points of the Symmetric Positive Definite Riemannian manifold and then, a metric learning approach is applied in order to learn a Mahalanobis based distance which will minimize the distance of similar samples and maximize the distance of dissimilar ones. The experimental results with the use of two popular signature datasets of western origin, shows that the proposed framework is comparable at least to State-of-the-Art models, typically realized under a framework of Euclidean nature.

Keywords

Offline Signature Verification, Riemannian Geometry, Neural Networks, Optimization, Symmetric Positive Definite Matrices, Digital Image Processing, Computer Vision, Binary Classification

Table of Contents

Table Catalogue

Image Catalogue

Alphabetical Index

2C2L: 2-Channel-2-Logit

AIM: Affine Invariant Metric

ANN: Artificial Neural Network

AUC: Area-Under-Curve

AVN: Adversarial Variation Network

BiLSTM: Bidirectional LSTM

BPPT:	Back Propagation Through Time
CEC:	Constant Error Carousel
CNN:	Convolutional Neural Network
CT:	Contourlet Transform
CVA:	Cross Validation Algorithm
DCSM:	Deep Convolutional Siamese Network
DMML:	Deep Multitask Metric Learning
DSL:	Domain Specific Language
DT:	Dichotomy transform
EER%:	Equal Error Rate Percentage
FHE:	Forensic Handwriting Examiner
FIR:	Finite Impulse Response
FPR:	False Positive Rate
gmLSTM:	generalized matrix LSTM
GSC:	Gradient, Structural and Concavity
HE:	Handwriting Examiner
HOG:	Histogram of Oriented Gradients
HSV:	Handwritten Signature Verification
IDN:	Inverse Discriminatory Network
IIR:	Infinite Impulse Response
LEM:	Log-Euclidean Metric
LSTM:	Long Short-Term Memory
MLP:	Multi-Layer Perceptron
mLSTM:	matrix LSTM
MSDN:	Mutual Signature DenseNet
NLP:	Natural Language Processing
P2S:	Point-to-Set
RBF:	Radial Basis Function
RNN:	Recurrent Neural Network
ROC:	Receiver Operating Characteristic
SGD:	Stochastic Gradient Descent
SPD:	Symmetric Positive Definite
SVM:	Support Vector Machine
TPR:	True Positive Rate
VLAD:	Vector of Aggregated Local Descriptors

WD:	Writer Dependent
WI:	Writer Independent

Introduction

We are all witnessing that as time passes by, humans rely more and more on digital technologies, which by the day are improving. It is easily noticeable that we tend to use pen and paper less, as the digital world provides the same functionality, while also offering a lot more. Due to this, the majority uses one or more text processing software in order to perform a related task. It is important to remember though that handwriting is still used in many aspects of life, with the most common being signing. Authenticating a handwritten signature by utilizing computer vision is a captivating task, whose importance and use has been seen throughout the literature [1], [2].

The signature that a person produces, typically on a sheet of paper, is the outcome of a combination of the learned common writing characteristics and the individual's hand motoric processes [3]–[6]. The handwritten signature is a trace of the hand movement a person performs, which is commonly acquired with the use of a pen, of any colour, and/or thickness, and a paper for the physical depiction. Its digital counterpart, i.e., a digital image comes from an electronic device typically a scanner, which creates the corresponding digital depiction. In terms of the signature verification community, the image of the signature, is referred to as static or offline signature. It is worth noting that, by using a digital pen or a tablet, it is possible to acquire a time indexed multivariate sequence which represents the motion rather than the visual outcome of it, termed hereafter as dynamic or online signature. Using any of the signature representations, it is possible to devise an automated signature verification digital system (simplified, SV), which basically attempts to support, in a robust and digital way, a number of manually performed authentication tasks, typically performed by a Forensic Handwriting Examiner (FHE). Initially, SV systems are categorized by the signature representation they utilize, termed dynamic-online or static-offline [5], [7]–[17], [17]–[22].

Aside from the online or offline categorization, SV systems are also classified as Writer Dependent (WD) or Writer Independent (WI) [23]–[29]. The first approach, WD, refers to training a dedicated classifier for each writer with their reference samples; this is the approach that is more frequently encountered in literature. On the other hand, in WI, a universal classifier attempts to adapt and discriminate between two types of distributions: a) the ω^+ genuine-to-genuine pairs of a learning set of signature samples, referred hereafter as the positive, similar, or intra-class, distribution, and b) the ω^- genuine-to-forgery pairs, termed hereafter as the negative, dissimilar, or inter-class, distribution [30]–[32]. Usually, this is done by transforming the feature space $F \in \mathbb{R}^d$, the space that contains the features of the signatures, either extracted from the static context or acquired from the dynamic nature of the hand motion, which contains any two signature pairs $(S_i,\ S_j)$, into the (dis)similarity distance space $D = \left|S_i - S_j\right| \in \mathbb{R}^d_{\geq 0}$ [33].

The design of both WD and WI oriented SV systems, is realized as a two-class classification problem. The positive class refers to the ω^+ genuine-to-genuine pairs, while the negative class refers to the ω^- genuine-to-forgery pairs, the latter being composed of forgeries of different origins. Therefore, it is important to note that there are different types of ω^- genuine-to-forgery pairs, due to different kinds of forgeries. According to some literature [24], [25], [34], [35], the most commonly encountered types of forgeries are: i) *Random*, ii) *Casual/Zero Effort*, iii) *Simulated* and iv) *Skilled*; all of which are explained in the following chapter. In case of a WI-SV system, in order to ensure an unbiased learning and evaluation procedure, the testing dataset must not belong to the signature data that the system was trained with. And since these systems are trained to discriminate between the $\omega^+$and ω^- classes, it is possible to form the negative distribution with two distinct ratios of genuine-to-random or genuine-to-skilled forgeries.

In offline SV systems, the signature samples are in the form of images, which implies the use of Computer Vision related algorithms. An efficient and descriptive representation of images was introduced in [36], in which the images was were mapped to a non-Euclidean representation, by means of region covariance descriptors, which introduced the principles of Differential Geometry in computer vision, with the most common form being the space of Symmetric Positive Definite (SPD) matrices. This representation has been abundantly used in applications, such as fine-grained and generic image classification [37]–[40], few-shot learning [41], medical imaging [42], etc. Since similarity-based methods are geometry agnostic, it is reasonable to wonder how can the SPD manifold be manipulated in order to build an efficient offline SV algorithm.

Geometry aside, similarity between two distinct samples can be easily computed by their corresponding distance in the feature space they reside. In the case of the SPD manifold, there is a plethora of formulas that measure the distance of two samples, exploiting the underlying geometry in a manner of ways, like the geodesic distance induced by the affine invariant Riemannian metric (AIM) [42], log-Euclidean metric (LEM) [43], the Jensen-Bregman LogDet Divergence [44], the Jeffrey Divergence [45] and the geodesic distance induced by the Bures-Wasserstein metric [46], Generalized Bures-Wasserstein metric [47] and log-Cholesky metric [48]. Besides the conventional distance measuring, metric learning has also been developed in the SPD manifold, under three major frameworks [49]: a) learning a distance metric in the Euclidean tangent space which is a vector space, b) learn a distance metric in the kernel space, and c) by projecting an initial high-dimensional SPD space into another, typically of lower dimension, SPD space, which preserves the manifold structure, and learn a corresponding metric.

To the author's knowledge, no prior work has been proposed in the literature in which the original SPD representation of an individual's signature images is mapped to a two-dimensional distance space, followed by a metric learning approach for WI-SV. In literature, relative research direction mainly focuses on deep metric learning with methods like Signature Embedding [50], Histograms of Oriented Gradients (HOGs) with Deep Multitask Metric Learning [31], graph edit distance [51], Mutual Signature DenseNet with Deep Convolutional Siamese Network [52] and use of HOGS and Local Binary Patters in a Mahalanobis distance based learning algorithm [53].

The above examples, along with many more, theorize the existence of a Euclidean structure both for the data and the algorithmic development approach in the SV literature. Therefore, machine learning algorithms in SV are developed under the assumption that the underlying nature of the signature descriptors is Euclidean, as it can be seen by the amount of literature work on SV systems. Given the fact that the use of non-Euclidean descriptors has improved the performance of numerous computer vision related tasks, it is likely that the same performance improvement will arise if a similar approach is taken for SV methods. The proposed thesis framework's innovative characteristics are:

1. We transform any signature image to a 10×10 SPD space, followed by a mapping into an associated two-dimensional distance space. We begin by selecting a pair of signatures (a, b) and there corresponding SPD matrices $\mathcal{S}_{10,a}^{++}$ and $\mathcal{S}_{10,b}^{++}$. Next, any initial SPD matrix $\mathcal{S}_{10,(a,b)}^{++}$ is partitioned into two $\mathcal{S}_{5,(a,b)}^{++}$ submatrices (the block diagonal elements of the $\mathcal{S}_{10}^{++}$), followed by two distance measures between corresponding $\mathcal{S}_{5,(a,b)}^{++}$ submatrices. This attempts to depict pairwise characteristics in a distance-based space originating from initial SPD matrix representations.
2. We, then, perform a low-dimensional Mahalanobis metric learning algorithm, by computing the 2x2 SPD matrix that essentially dichotomized the distance space, by using a meta-heuristic optimization algorithm, in the form of a neural network, proposed in [54]. Specifically, the

optimization network is trained and used in parallel, with a complete iteration of the algorithm being first training the network for a few epochs and the using it to compute the SPD parameter for the Mahalanobis distance formula for another few epochs. The algorithm terminates when either the SPD parameter converges or the predefined total iterations are reached.

This thesis is organized in the following manner: Chapter 1 presents in a glance the handwritten signatures concept, along with a number of characteristics that affect its formation and different features that can be extracted from it. Chapter 2 presents the foundations of the components utilized in the proposed framework, such as the Recurrent Neural Network topology and the postulates of the Riemannian geometry of the SPD manifold. Chapter 3 provides the full metric learning algorithm proposition, the experiments performed and their results. Finally, Chapter 4 provides the conclusion.

1 Handwriting

Handwriting is considered to be a skill which humans learn in their early stages of life. It is a skill that is affected by a number of conditions, both of the human side as well as on the environment they are in, eventually becoming an automated task for a human to perform. As an automated task, it is hard to correct for potential errors once the hand movement is performed [55]. As a trait, it is unique to each human [3], although it is possible for some so look alike, as seen in the research done in [55], where they found two writers with similar handwriting. Handwriting, is affected by a plethora of factors, resulting in what is termed as the intra-variability of the produced content, without the loss of the unique writing characteristics of the person. This uniqueness means that it is difficult to reproduce a person’s writing, as one has to suppress their own hand motor skill that they have developed and then perform the task at due with the writing characteristics and variability of the original writer [55]. However, it is possible, through practice and given enough information of the writing capabilities of the person, for someone, to learn to produce simulations of high quality of their writing [55].

As mentioned, handwriting is affected by several factors, both internal and external [3]. The gender and age of the writer, their nationality, their dominant hand, the handwriting system of the country they were taught in, if they are writing from memory, i.e., if they are copying or they are dictated the content, their psychological state, stress levels and many more are some of the internal factors, while the writing device, environmental noise, the stance they have to take due to the table they are writing on or the chair they may be sitting in and others are external factors [3], [8]. Depending on which factors are involved in the process and at what level, the writer’s produced handwriting may change severely, possibly exceeding the normal variability that appears.

Generally speaking, handwriting is not comprised solely on the outcome of the hand movement the person performs, but it is the result of a compound activity of the eye-hand coordination and the perceptual abilities of the writer [8]. Due to this, handwriting is considered a biometric trait [8], which like all other, cannot be easily modified, if at all possible. As a biometric trait, it can be and is used in many applications, including, but not limited to, medical, forensic and, primarily for person authentication and verification purposes [8]. In each application, the handwritten trace is possibly represented in a different manner, like an image of the handwriting, or a time-indexed series of different measurements of it, such as position of the vertical and horizontal axis, hand velocity, etc [3], [8], [55]–[57]. In the following, we will be focusing on the person authentication and verification aspect of handwriting, which is also part of forensic science.

Any person may claim any given identity, by producing a look-alike handwritten signature. Since handwriting is unique to each person, so is the signature they produce, even more so if one considers that there can be many ways for one person to define its signature trace, such as having a legible one by signing their name, stylizing the ink trace with flourishes, etc [55], [58]. Using this unique aspect of signatures, it is possible to establish if any given questioned document or signature belongs to the claimed identity. This is a job usually performed by a Handwriting Examiner (HE) or Forensic Handwriting Examiner (FHE). The (F)HE must objectively reach a conclusion by comparing the questioned document or signature to known references of the claimed identity’s, either manually or through an automated verification process. In either occasion, objective features and measurement must be used in order for the result to not be biased.

Given a handwritten document, a (F)HE can derive objective characteristics such as line lengths, turning points, intersections, legibility and per word/character features, such as line component ratios [3], [55], either manually or automatically should they have access to the dynamic information of the hand movement. It is also possible to use digital image processing techniques if one digitizes the ink trace, as some aspects of these kinds of images and algorithms that use them have been correlated with some of the characteristics a person has during writing [3]. Since (F)HEs generally can only use a static representation of the signature [8], using image processing or computer vision techniques can be favourable as they are objective measurements, explainable and able to be correlated with writing aspects, in order to decide if the questioned document is genuine or a forgery. According to the literature, forgeries can be categorized as: a) *Random*: using the genuine signature of the person claiming to be another, b) *Casual*/*Simple*: the person forging the signature produces a simulating sample by only knowing the name of the claimed identity, c) *Simulation*: the (inexperienced) forger produces a sample after practicing given one or more reference samples, and d) *Skilled*: the (experienced) forger produces a sample after practicing, by using some kind of technique such as calligraphy [8], [34], [58].

When it comes to signature verification specifically, many automated systems have been developed, with the goal of assisting the (F)HEs [8], [9]. These systems work similarly to a (F)HE; they take as input at least one reference and the questioned document and they output either the probability of being genuine/forgery or the distance of the questioned document with the reference(s). Such a system could be of great help to the examiner, as it could provide a hint as to which questioned documents are definitely genuine/forged and which may require a more thorough examination conducted by the (F)HE. It is important though, for the automated verification system to be explainable and well defined in terms of the objective examination guidelines, as the results can directly affect people’s lives, since, e.g., they may be administered to courts of law [57], [59], [60].

2 Background

The proposed framework relies on neural network based around a variant of the LSTM RNN topology, used in-place of a handcrafted optimization algorithm, employed in the space of SPD matrices. This type of optimization approach, i.e., using a non-handcrafter optimization algorithm that learns how to optimize, is called Meta-learning. In order to build intuition behind these terms, how they are incorporated in the proposition and why they were chosen, this chapter focuses on an analytical overview of these concepts. Firstly, the basic RNN and the LSTM variant are covered, following up with Meta-learning and the foundations of the SPD Manifold. Lastly, the meta-learning neural network is presented.

Moving forward, unless otherwise specified, matrices are denoted by capital letters, e.g., X, with symmetric matrices being in italic, e.g., X, while SPD matrices being in bold, e.g., **X**. A vector is denoted by a lowercase italic letter, e.g., x. The d-dimensional SPD manifold is specified by $\mathcal{S}_d^{++}$, while the tangent space T of a point $\mathbf{M}$ on the SPD manifold is denoted by $\mathrm{T}_{\mathbf{M}}\mathcal{S}_d^{++}$. Lastly, $\odot$ denotes the Hadamard product, $\oplus$ denotes the element-wise addition, $\mathbf{I}_d$ and $\mathbf{0}_d$ represent the identity and zero filled matrices while any number with an arrow above it, e.g., $\vec{0}$, corresponds to a vector filled with that number.

2.1 Recurrent Neural Networks

Sequential data or time-indexed series exhibit relations between samples, since usually at any given time step, the value of a sample might be affected by any values of its previous time steps. This can also lead to data with variable dimensionality, which cannot be easily modelled with a standard MLP network topology, as the MLP does not reuse any information from a previous prediction besides the trained weights thus not being able to capture the entire relation between any time steps. In order to address this, RNN, a variant of the baseline ANN was proposed, that a) allows the use of data comprised of variable time steps and b) can also capture any potential associations between current and previous samples [61], [62]. Figure 1 shows a block diagram of the elementary building blocks of the basic RNN network topology. In their uninitialized state, eq. (1) and eq. (2) provide the typical mathematical representation and operation of the network, and are referred to as a RNN cell [63]. The vectors $x_{(t)} \in \mathbb{R}^f$ and $h_{(t-1)} \in \mathbb{R}^d$, where f and d denote the size of each vector, correspond to the input data to the network and the hidden state at time step $t-1$. The matrices $\mathrm{W}_{hh} \in \mathbb{R}^{d \times d}$, $\mathrm{W}_{xh} \in \mathbb{R}^{f \times d}$, $\mathrm{W}_{hy} \in \mathbb{R}^{d \times o}$, with f and d being the same as before and o denoting the output vector size, are trainable weights of dimensions that match the output vector, denoted by the second letter in their subscript, when multiplied with the corresponding variable, denoted by the first letter in their subscript. The vectors $b_h \in \mathbb{R}^d$ and $b_y \in \mathbb{R}^o$ are trainable biases, while $h_{(t)} \in \mathbb{R}^d$ and $\hat{y}_{(t)} \in \mathbb{R}^o$ are the hidden state and output vectors of the network at time step t. We define the first time step the network will get input data and output a prediction as $t = 0$, which is why the hidden state, h_t, at time step $t = -1$, is either initialized with a zero valued vector or it is the output of some other network earlier in the architecture.

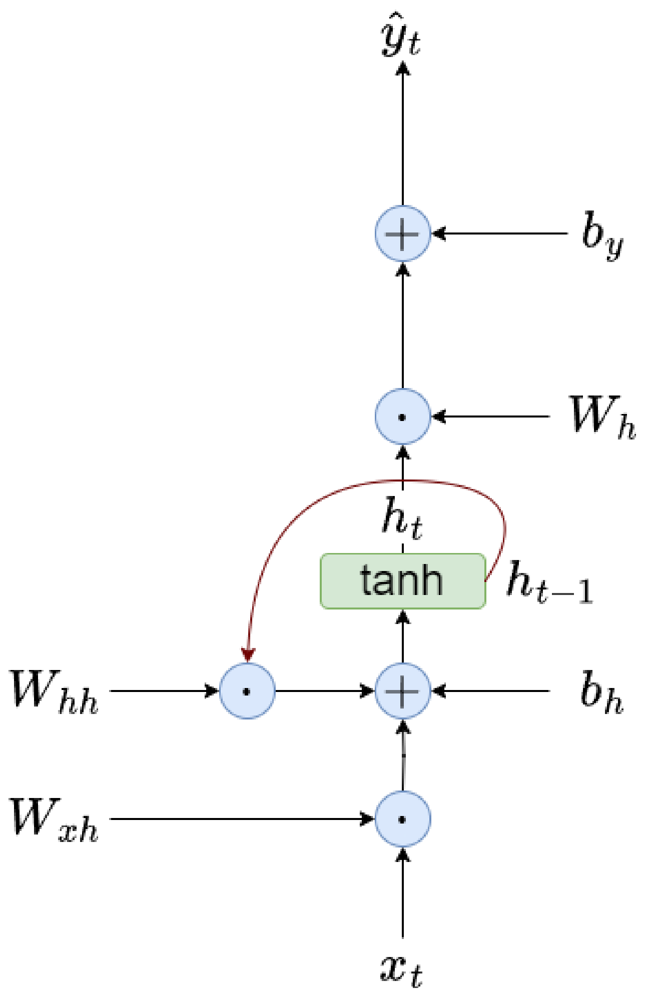


Figure 1: RNN Block Diagram. The blue blocks represent mathematical operations between variables and the green blocks depict operations using a single variable. The red path denotes the flow of the hidden state.

$$h_{(t)} = \tanh\left(\mathrm{W}_{hh}^{T} h_{(t-1)} + \mathrm{W}_{xh}^{T} x_{(t)} + b_h\right) \tag{1}$$

$$\hat{y}_{(t)} = \mathrm{W}_{hy}^{T} h_{(t)} + b_y \tag{2}$$

The dependencies of the data that the RNN can exploit, are easily identified in eq. (1). The hidden state, h_t, represents the relations between each data point that the network has calculated, which is used for the prediction made for the next time step. The relation capturing characteristic of the hidden state is the result of the reuse of it from the previous time step, which in turn is also the result of the reuse of it from two time steps back, and so on. The hidden state can be thought of as the network’s memory of data it has seen, compared to the standard ANN topology which does not have such a mechanism, as explained earlier. Due to these, the RNN is widely used for a number of tasks, especially in the area of NLP, for the tasks of text generation, sentiment classification, machine translation, etc.

Eq. (1) can be regarded a) as recurrent one, when computed from the origin of the time steps, (i.e. $t = 0$), as each next calculations reuses a previous one, or b) as recursive one, if needed to be calculated at a later step, so at time step $t = k$, $h_{(k)}$ contains all information from all previous time steps, with $t = 0$ being the end [64]. This kind of information flow clearly suggests that the RNN topology is an actual IIR system. Following the example #1 of reference [63], the IIR aspect of eq.

(1) and eq. (2) can be seen by setting the initial hidden state $h_{(-1)} = \vec{0}$, the input data $x_{(t)} = \delta[t]$, where $\delta[t]$ is the Kronecker Delta function, and the bias terms $b_h = \vec{0}$. So, the evaluation of eq. (1) for some time steps, provide some sort of the infinite impulse characteristic of the network according to:

$$\begin{aligned} h_{(-1)} &= \vec{0} \\ h_{(0)} &= \tanh\left(W_{xh}^T\, \vec{1}\right) \\ h_{(1)} &= \tanh(W_{hh}^T \tanh\left(W_{xh}^T \vec{1}\right)) \\ h_{(2)} &= \tanh(W_{hh}^T \tanh\left(W_{hh}^T \tanh\left(W_{xh}^T\, \vec{1}\right)\right)) \\ &\dots \end{aligned}$$

Similarly, eq. (2), which is the output of the network, is just a linear combination of the corresponding hidden state values of the above example, which similarly displays the IIR characteristic, since the hidden state is defined for all time steps. Even with a single initial impulse, the RNN topology can produce an output for any positive time step. With this kind of representation, and regardless of the initial input signal, the RNN can be "unfolded" for any number of time steps, something that visually, is presented Figure 2.

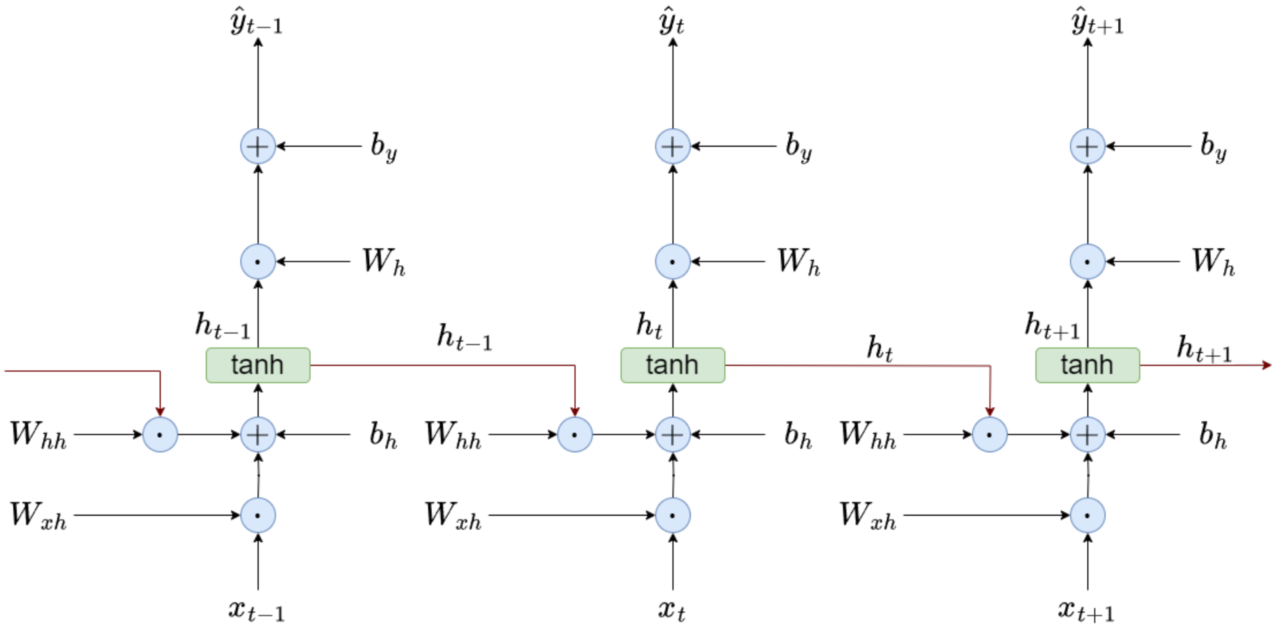


Figure 2: The unfolding property of an RNN. In this depiction, the recurrence characteristic of the topology is shown by the use of another hidden state and input vector by the same network.

In practice, the RNN is usually trained as an FIR system approximation, by splitting sequences in places where they show no dependence with each other. This allows for finite computations, sufficient convergence of the given loss function and similar results[1] [63].

The back propagation algorithm works in the RNN with a slight modification. Specifically, since at each time step the weights and biases are shared, since they are not time dependent functions and only change during back propagation, the gradient computation starts at the output of each time step, moves backwards in time and is accumulated at time step $t = 0$, performing an update to the weight for the next epoch. This procedure is termed as the "Back Propagation Through Time (BPPT)" and visually is depicted in Figure 3. During training, the RNN topology can exhibit two kinds of

[1] For the interested reader, the proof for this can be seen in Proposition 1 of reference [63].

problems: the exploding and the vanishing gradients. The first one (exploding) refers to the gradient computation getting larger, in an exponential rate, during the back propagation stage while the latter to gradient getting smaller in an exponential rate. This occurs due to the recurrence relation of eq. (1), as the gradients flow backwards in time, they may be getting multiplied by quite large or small numbers, leading to the aforementioned problems. Exploding gradients can be easily solved by gradient clipping, which tests if the gradient is large in some manner, either through the Frobenius or max norm or eigenvalues; if so, then we scale it back by that metric.

On the other hand, vanishing gradients though are harder to solve. This prompted the proposal of a number of variants over the original RNN topology. Specifically, the original architecture of the RNN topology was modified in order to introduce nonlinear, data dependent structures, termed gates, which ensure that the gradient flow does not vanish, through training. As a result, the Long Short-Term Memory (LSTM) network cell was created [65].

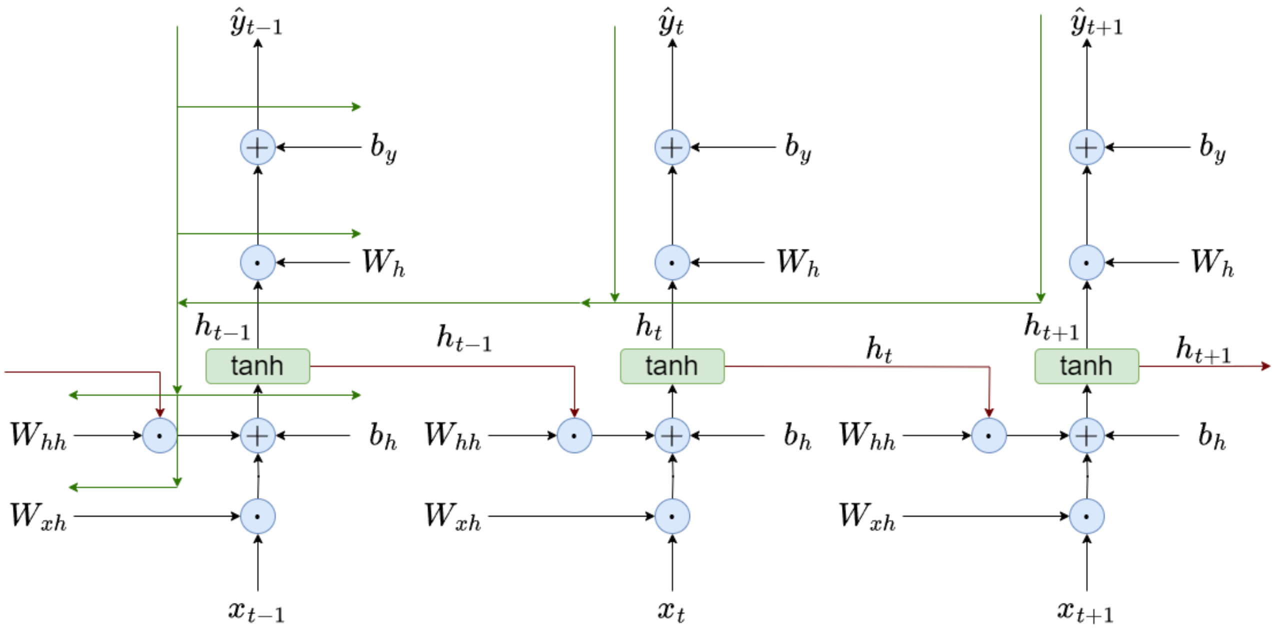


Figure 3: Gradient flow on a RNN. The gradient starts from the last step of the predicted sequence and moves backwards towards the first, updating the weights and biases in a cumulative manner.

2.1.1 Long Short-Term Memory Network

LSTM cells are a modified version of the RNN, which allow for better information flow in both forward and backward propagation. It was made possible, by introducing nonlinear structures that are trained to ensure that only relevant information, (i.e. time dependencies), affect the predictions made by the model in the forward pass, while making sure that the same relevance affects the network parameters during the backward pass. Figure 4a illustrates the LSTM architecture. It can be seen that besides the original hidden state vector, there exists a new one, the cell memory, $c \in \mathbb{R}^d$, which is computed in a much simpler way. Due to the easiness of its computation, the gradient flow is easier to compute and not as prone to vanishing gradients in that route, contrary to the gradient flow from the hidden state, as seen in Figure 4b. In principle, the LSTM cell is represented by the following formulas.

$$i_{(t)} = \sigma\left(\mathrm{W}_{xi}^{T} x_{(t)} + \mathrm{W}_{hi}^{\mathrm{T}} h_{(t-1)} + b_i\right) \tag{3}$$

$$f_{(t)} = \sigma\left(\mathrm{W}_{xf}^{\mathrm{T}}\, x_{(t)} + \mathrm{W}_{hf}^{\mathrm{T}}\, h_{(t-1)} + b_f\right) \tag{4}$$

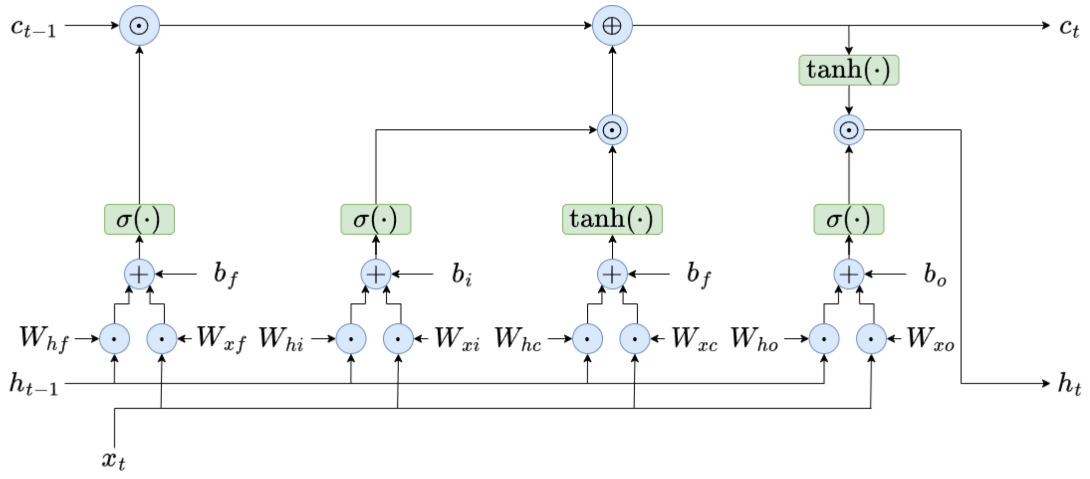
$$o_{(t)} = \sigma\left(\mathrm{W}_{xo}^{\mathrm{T}}\, x_{(t)} + \mathrm{W}_{ho}^{\mathrm{T}} h_{(t-1)} + b_o\right) \tag{5}$$

$$\tilde{c}_{(t)} = \tanh\left(\mathrm{W}_{xc}^{\mathrm{T}} x_{(t)} + \mathrm{W}_{hc}^{\mathrm{T}} h_{(t-1)} + b_c\right) \tag{6}$$

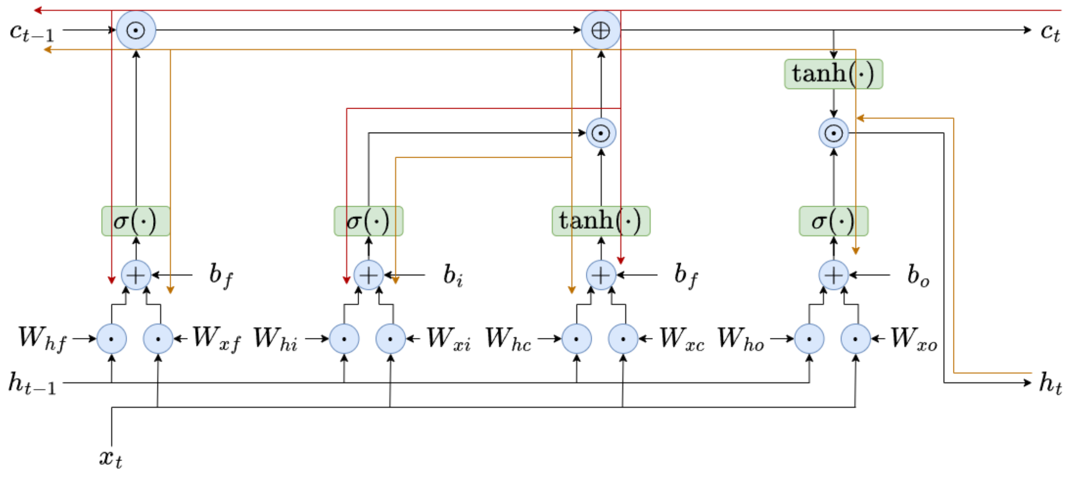
$$c_{(t)} = f_{(t)} \odot c_{(t-1)} \oplus i_{(t)} \odot \tilde{c}_{(t)} \tag{7}$$

$$h_{(t)} = o_{(t)} \odot \tanh\left(c_{(t)}\right) \tag{8}$$

The LSTM retains the IIR nature of the basic RNN cell and can also be unfolded. Eqs. (3)-(5) define the input, forget and output gates respectively. Each one of them controls different parts of the information flow. The input gate $i_{(t)} \in \mathbb{R}^d$ controls how much and which kind of information from the candidate memory cell $\tilde{c}_{(t)} \in \mathbb{R}^d$ are allowed to exist in the new memory cell, whilst the forget gate $f_{(t)} \in \mathbb{R}^d$ is responsible for how much of the previous memory cell $c_{(t-1)}$, information is going to be added to the new one. The output gate $o_{(t)} \in \mathbb{R}^d$ controls also the information for the predictions. Due to these, the LSTM can capture long term dependencies better than the basic RNN, while also offering multiple gradient paths, which in turn resolves the vanishing gradient problem. Given the fact that the gates control the information flow, they are numerically bounded between 0 and 1; this is the outcome of applying the sigmoid function σ. Due to this, the data used in training need to be standardized, i.e., have a mean value of 0 and a standard deviation of 1, otherwise the LSTM performs poorly, due to the nonlinear activation functions mapping data either to 0 or 1 most of the times.



(a)



(b)

Figure 4: (a) LSTM Block Diagram. (b) Two-way gradient flow of LSTM, red path is easier to compute than yellow path.

Although in theory, it is still possible, given some set of badly formed parameter values, for the vanishing gradient issue to appear, in practice it has been seen that it does not, as LSTM are mostly numerically stable [66]. The reason for this can be seen by setting $f_{(t)} = \vec{1}$ and $i_{(t)} = o_{(t)} = \vec{0}$. It provides an uninterrupted, computationally efficient, gradient flow in its backward pass, as can be seen inFigure 5, which the creators named Constant Error Carousel (CEC) mode [65]. So, even in the case that the gradient flow from the hidden state output diminishes due to the nonlinear structures, the memory cell path does not due to linearities and the CEC.

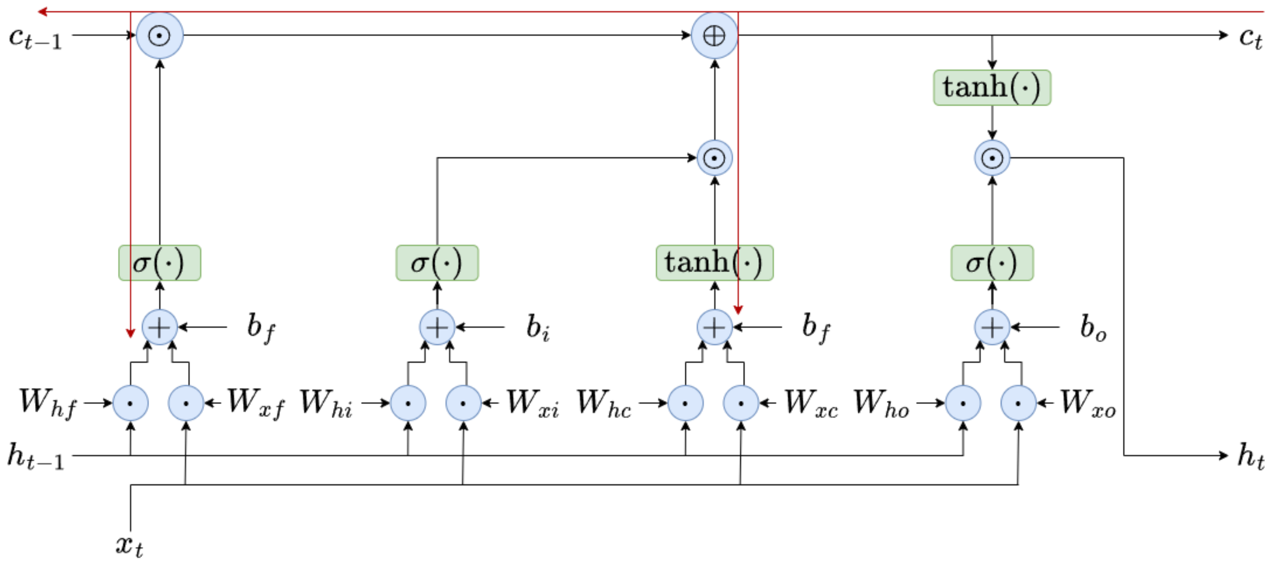


Figure 5: LSTM CEC mode gradient flow, denoted by the red path.

2.2 Meta-learning

Usually, neural networks are trained using a handcrafted styled optimizer, which utilizes a typical form of gradient descent. Such algorithms and its variants are the SGD with and without momentum [67], [68], Rprop [69] , RMSProp [70], Adagrad [71], Adam and Adamax [72]. Although capable, these do not always find a good or total minima for the loss function of the task at hand, even when adding additional constraints such as L_2 or L_1 penalties. This can be also accounted to not being able to consider the task's characteristics, such as the underlying data's distribution. Due to this, different learning rules have been formulated [73]–[76], that either use genetic algorithms, synaptic learning rules, or biologically matching functionalities. These formulations, along with training neural network models to perform weight and bias updating, are based on biological reasoning and knowledge reuse, thus creating task specific optimization algorithms, generally known as meta learning algorithms. RNNs have been used to create meta-learning algorithms, basically neural networks who learn to optimize parameters in a specific task, while also being able to transfer learn to other similar tasks with unseen data.

Meta-learning has been practiced using many different ways of training in an extensive range of tasks. Hochreiter et al. in [77] trained an LSTM network, by gradient descent, to come up with an optimization algorithm for approximating quadratic functions using 35 examples. Li and Malik in [78] used reinforcement learning with a guided policy to learn better optimization algorithms in a number of tasks. Andrychowicz et al. in [79], Ravi and Larochelle in [80], Wichrowska in [81] and Rusu et al. in [82] trained recurrent models using the parameter gradients as input for a number of tasks. Bello et al. in [83] used reinforcement learning in order to train an RNN to generate strings in a Domain Specific Language (DSL) that correspond to parameter update equations given a list of base functions, like gradients, etc. Lee et al. in [84] formulated a meta-learning algorithm which can optimize convex learners by few-shot learning. Lastly, Gao et al. in [54], [85], created the first meta-learning optimizer for Riemannian manifolds, firstly on SPD and next generalized to other manifolds, by using a new LSTM variants, that utilize specific matrix operations that respect the manifold structure.

In all the aforementioned tests, meta-learning algorithms have been shown to outperform handcrafted optimizers, therefore generalize in a more efficient way the learners in the given tasks. But these algorithms come also at the cost of having to train yet another type of network, which can be time consuming and also possibly prone to the same issues as when training an ANN by basic optimizers, such as over- or underfitting, non-convexity, suboptimal optima, etc.

2.3 Symmetric Positive Definite Manifold

Riemannian submanifolds[2] are smooth (i.e. infinitely differentiable, C^{∞}) submanifolds embedded in the Euclidean space, $\mathcal{E}$, endowed with a Riemannian metric that is obtained based on restrictions of the metric of $\mathcal{E}$ [86], [87]. The most frequently used manifolds fall under this category, including the manifold of Symmetric Positive Definite Matrices. Some manifolds are curved, which makes simple tasks like addition, subtraction and gradient descent nontrivial. The SPD manifold of 2x2 matrices can be visualized as two cones which span $\mathbb{R}^3$, as shown in Figure 6.

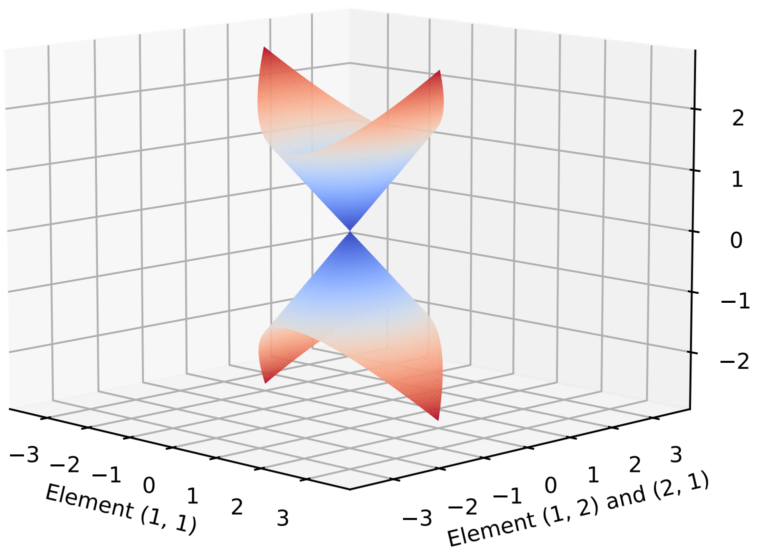


Figure 6: Visualization of the 2x2 SPD manifold.

The SPD Manifold is defined to be the set of all symmetric, positive definite matrices. The set formula describing it can be seen in eq. (9).

[2] According to Absil et al. in [86] and Boumal in [87], these are as well Riemannian Manifolds and the most commonly encountered type. But in general, Riemannian Manifolds do not follow the same principle, i.e., being submanifolds of the Euclidean space.

$$\mathcal{S}_d^{++} = \left\{\mathbf{M} \in \mathbb{R}^{\mathrm{d}\times\mathrm{d}} \mid \mathbf{M} = \mathbf{M}^{\mathrm{T}}, x^{\mathrm{T}}\mathbf{M}x \geq 0, \forall\, x \in \mathbb{R}^d \setminus \{\vec{0}\}\right\} \tag{9}$$

The tangent space of an SPD matrix $\mathbf{M}$, $\mathrm{T}_{\mathbf{M}}\mathcal{S}_d^{++}$ is the space of symmetric matrices. The tangent bundle, $\mathrm{T}\mathcal{S}_{\mathrm{d}}^{++}$, formed by this manifold and the corresponding tangent planes is a vector space of symmetric matrices and is seen in eq. (10).

$$\mathrm{T}\mathcal{S}_d^{++} = \{\, \forall\, (\mathbf{X}, Y) \mid \mathbf{X} \in \mathcal{S}_d^{++}, Y \in \mathrm{T}_{\mathbf{X}}\mathcal{S}_d^{++} \,\} \tag{10}$$

The SPD Manifold when equipped with the Riemannian metric of Eq. (11), the Rao-Fisher metric [88], is globally diffeomorphic [89], meaning there is a one-to-one, continuously differentiable mapping in both directions, in the form of eq. (12) and eq. (13).

$$\langle\, X, Y \,\rangle_{\mathbf{M}} = \mathrm{tr}(\mathbf{M}^{-1}X\mathbf{M}^{-1}Y)^2 \tag{11}$$

$$\mathbf{Y} \equiv \exp_{\mathbf{X}}(Y) = \mathbf{X}^{\frac{1}{2}} \exp\left(\mathbf{X}^{-\frac{1}{2}}Y\mathbf{X}^{-\frac{1}{2}}\right)\mathbf{X}^{\frac{1}{2}} \tag{12}$$

$$Y \equiv \log_{\mathbf{X}}(\mathbf{Y}) = \mathbf{X}^{\frac{1}{2}} \log\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}}\right)\mathbf{X}^{\frac{1}{2}} \tag{13}$$

The exp and log operations are defined to be the matrix exponential and logarithm, accordingly, which can be computed by using the eigen-decomposition of symmetric matrices, $X = \mathrm{UDU}^{\mathrm{T}}$, where U, D are the eigenvectors and eigenvalues respectively, as seen in eq. (14) and eq. (15).

$$\exp(X) = \mathrm{U}\exp(\mathrm{D})\,\mathrm{U}^{\mathrm{T}} \tag{14}$$

$$\log(X) = \mathrm{U}\log(\mathrm{D})\,\mathrm{U}^{\mathrm{T}} \tag{15}$$

Intuitively, the exponential map, maps a point on the SPD manifold to the tangent space $\mathrm{T}_{\mathbf{M}}\mathcal{S}_d^{++}$of $\mathbf{M}$, i.e., $f : \mathcal{S}_d^{++} \mapsto \mathrm{T}_{\mathbf{M}}\mathcal{S}_d^{++}$, while the logarithmic map operates the reverse action, viz. $f : \mathrm{T}_{\mathbf{M}}\mathcal{S}_d^{++} \mapsto \mathcal{S}_d^{++}$. By substituting Eq. (13) in (12), it is possible to derive the single parametric equation connecting two points, $\mathbf{X}, \mathbf{Y} \in \mathcal{S}_{\mathrm{d}}^{++}$, with a geodesic curve $\mathbf{\Gamma}_{\{\mathbf{X},\mathbf{Y}\}}(t)$, which has both ends $\mathbf{X}$, for $t = 0$ and $\mathbf{Y}$ for $t = 1$. The corresponding formula is provided in eq. (16).

$$\mathbf{\Gamma}_{\{\mathbf{X},\mathbf{Y}\}}(t) = \mathbf{X}^{\frac{1}{2}}\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}}\right)^{t}\mathbf{X}^{-\frac{1}{2}} \tag{16}$$

Using Eq. (11), the distance between two points on the manifold can be measured by substitution of Eq. (13), using one SPD matrix as the tangent origin.

$$d(\mathbf{X}, \mathbf{Y}) = \langle\, \log_{\mathbf{X}}(\mathbf{Y}), \log_{\mathbf{X}}(\mathbf{Y})\rangle_{\mathbf{X}} = \mathrm{tr}\left[\log^2\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}}\right)\right] = \left\|\log\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}}\right)\right\|_{\mathrm{F}} \tag{17}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. Last, but not least, given the fact that SPD matrices do have a symmetric nature, it is easy to derive that they only have $\frac{d(d+1)}{2}$ independent values. Also, since the logarithmic map, maps an SPD point to a tangent space, which is a vector space, due to it being a linear subspace of $\mathbb{R}^{d,d}$ [87], a map can be derived such that $f : \mathrm{T}_{\mathbf{M}}\mathcal{S}_d^{++} \mapsto \mathbb{R}^{\frac{d(d+1)}{2}}$. This map, termed the ***vec*** operator [42], gives the orthonormal coordinates of the tangent vector Y in the tangent space of $\mathbf{X}$. This map is described by eq. (18) and eq. (19) while eq. (20) provides the relation between the Riemannian metric of Eq. (11) with Euclidean metric in $\mathbb{R}^{\frac{d(d+1)}{2}}$.

$$\boldsymbol{vec}_{\mathbf{I}_d}(Y) = [\, Y_{1,1}, \sqrt{2}Y_{1,2}, \sqrt{2}Y_{1,3}, \dots, Y_{2,2}, \sqrt{2}Y_{2,3}, \dots, Y_{d,d}] \tag{18}$$

$$\boldsymbol{vec}_{\mathbf{X}}(Y) = \boldsymbol{vec}_{\mathbf{I}_d}\left(\mathbf{X}^{-\frac{1}{2}}Y\mathbf{X}^{-\frac{1}{2}}\right) \tag{19}$$

$$\langle Y, Y\rangle_{\mathbf{X}} = \|\boldsymbol{vec}_{\mathbf{X}}(Y)\|_2^2 \tag{20}$$

To the author's opinion, the SPD Manifold is one the few manifolds that can be endowed with different Riemannian metrics, with some of them defining the exponential and logarithmic map differently, which leads to the geometry of the space being exploited in a different manner. Those metrics are the Log-Euclidean [90], Log-Cholesky [48], Bures-Wasserstein [46] and Generalized Bures-Wasserstein metrics [47].

2.4 Optimization in SPD Manifolds

The Euclidean space is considered to be a trivial manifold with zero curvature, making it among the handiest to work with. Its associated metric, exponential and logarithmic maps for two vectors belonging to the Euclidean space, $x, y \in \mathbb{R}^n$, are described by Eqs. (21)-(23), in which $\mathbf{I}_\mathrm{n} \in \mathbb{R}^{n \times n}$ is the diagonal identical matrix.

$$\langle x, y \rangle = y^\mathrm{T} x = y^\mathrm{T} \mathbf{I}_\mathrm{n} x \tag{21}$$

$$\exp_x(y) = x + y \tag{22}$$

$$\log_x(y) = x - y \tag{23}$$

As seen, the two maps, are the well-known addition and subtraction operators of vectors. If compared to the SPD maps, one may easy find out that both in general and in the case of the SPD manifold, the exponential map is the equivalent of the addition operation, while the logarithmic map is the equivalent of the subtraction operation, due to the fact that the tangent space being is a vector space. With this, it is easy to formulate a concrete formula for gradient descent in the SPD manifold. And since the derivative of a function $f : \mathbf{M} \mapsto \mathbb{R}^1$, w.r.t. $\mathbf{M}$, yields a symmetric matrix at the tangent space[3] of $\mathbf{M}$, i.e., $\nabla_\mathbf{M} f(\mathbf{M}) \in \mathrm{T}_\mathbf{M}\mathcal{S}_d^{++}$, the gradient descent formula with a step size of λ is described by Eq. (24). This operation is frequently referred to as the retraction operation on the manifold at a point $\mathbf{M}$, $\mathrm{R}_\mathbf{M}(Y)$, which in this case corresponds to the exponential map.

$$\mathbf{M}_{(t+1)} = \mathrm{R}_\mathbf{M}\left(\nabla_{\mathbf{M}_{(t)}} f\left(\mathbf{M}_{(t)}\right)\right) = \exp_\mathbf{M}\left(-\lambda\, \nabla_{\mathbf{M}_{(t)}} f\left(\mathbf{M}_{(t)}\right)\right) \tag{24}$$

In general, the retraction operation and the exponential map are not necessarily the same operation mathematically. The retraction operation is defined as a map from the tangent space to the manifold with a local rigidity condition that preserves gradients at the point $\mathbf{X}$ [86]. In the case of the SPD Manifold, the retraction operation and exponential map are the same due to the Riemannian metric the space is endowed with, eq. (11). With another Riemannian metric in this or another manifold, the retraction operation could be different from the exponential map, e.g., the Stiefel manifold, has multiple retraction operations with only one of them using the exponential map in it [91].

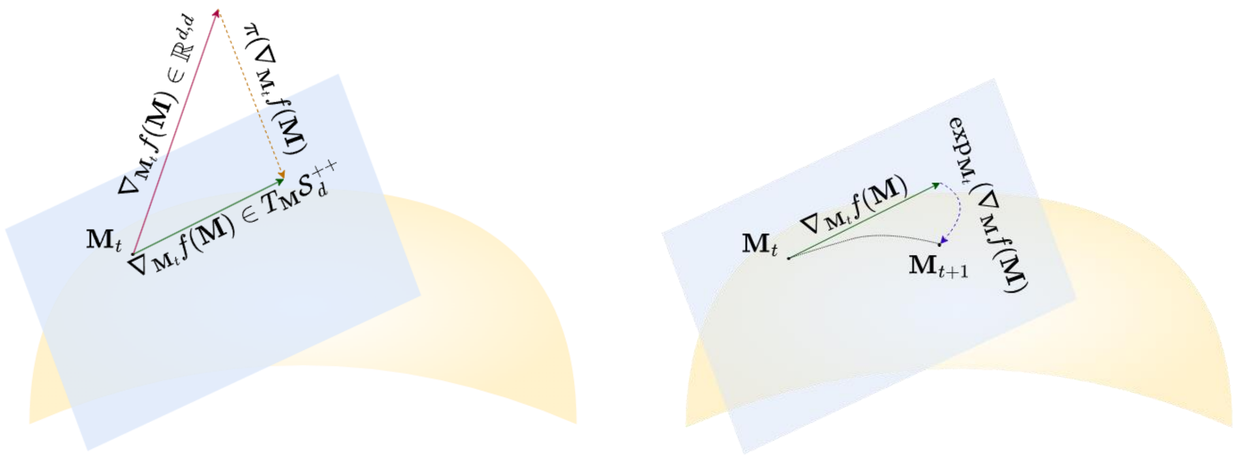
2.4.1 Gradient Computation

The gradient of a loss w.r.t. an SPD matrix can be computed in two ways. The first being the standard differentiation of the equations w.r.t. the SPD matrices while taking into account the structure of the manifold. The other way, which is commonly utilized [91]–[94], is differentiating w.r.t. the matrices without taking into account any information regarding their structure. This leads to Euclidean gradients which are not part of the tangent space of each matrix. They are then projected onto the tangent space using the respective orthogonal projection formula of each Riemannian submanifold. For the SPD Manifolds, the orthogonal projection map of a Euclidean matrix to the

[3] The reason as to why the gradient of an SPD matrix is a point in its tangent space, is thoroughly explained in [86] and [87].

tangent space of an SPD matrix, $\pi_{\mathbf{M}}(\cdot) : \mathbb{R}^{d\times d} \mapsto \mathrm{T}_{\mathbf{M}}\mathcal{S}_d^{++}$, is defined in Eq. (25). A visualization of the gradient descent algorithm on an SPD manifold, is provided in Figure 7.

$$\pi_{\mathbf{M}}(\mathrm{X}) = \mathbf{M}\left(\frac{\mathrm{X}+\mathrm{X}^{\mathrm{T}}}{2}\right)\mathbf{M} \tag{25}$$



$\nabla_{\mathbf{M}_t} f(\mathbf{M}) \in \mathbb{R}^{d,d}$
$\pi(\nabla_{\mathbf{M}_t} f(\mathbf{M}))$
$\mathbf{M}_t$
$\nabla_{\mathbf{M}_t} f(\mathbf{M}) \in T_{\mathbf{M}}\mathcal{S}_d^{++}$
$\mathbf{M}_t$
$\nabla_{\mathbf{M}_t} f(\mathbf{M})$
$\exp_{\mathbf{M}_t}(\nabla_{\mathbf{M}_t} f(\mathbf{M}))$
$\mathbf{M}_{t+1}$

(a) (b)

Figure 7: (a) Step 1: Orthogonal projection of the Euclidean gradient to $\mathrm{T}_{\mathbf{M}_t}\mathcal{S}_d^{++}$. (b) Step 2: Move on the manifold using the retraction operation.

As an example, the clustering task on the SPD manifold can be defined with eq. (26). Basically, for each class we need to find a point on the manifold for which the intra-class distances, calculated by eq. (17), will be minimized. Assuming the solution is not the mean value of points for each class, this problem is possible to be solved using gradient descent. Starting with the second approach, i.e., computing Euclidean gradients which will then be mapped to the tangent space, the derivative of eq. (26) w.r.t. point **X**, is computed through eq. (27)[4]. Once mapped to the tangent space, as seen in eq. (28), using the retraction operation, gradient descent is performed. For the initial method, i.e., computing the derivatives directly on the manifold, one starts by deriving eq. (28) then perform the retraction in eq. (29).

$$\min_{\mathbf{X}\in\mathcal{S}_d^{++}} f(\boldsymbol{X},\boldsymbol{Y}_i) = \min_{\boldsymbol{X}\in\mathcal{S}_d^{++}} d(\boldsymbol{X},\boldsymbol{Y}_i) \tag{26}$$

$$\frac{\partial f}{\partial \mathbf{X}_{(t)}} = \frac{-1}{d\left(\mathbf{X}_{(t)},\mathbf{Y}_i\right)}\mathbf{X}_{(t)}^{-\frac{1}{2}}\log\left(\mathbf{X}_{(t)}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}_{(t)}^{-\frac{1}{2}}\right)\mathbf{X}_{(t)}^{-\frac{1}{2}} \in \mathbb{R}^{d\times d} \tag{27}$$

$$\pi_{\mathbf{X}_{(t)}}\left(\frac{\partial f}{\partial \mathbf{X}_{(t)}}\right) = \boldsymbol{X}_{(t)}\left(\frac{\dfrac{\partial f}{\partial \mathbf{X}_{(t)}} + \dfrac{\partial f}{\partial \mathbf{X}_{(t)}}^{\mathrm{T}}}{2}\right)\boldsymbol{X}_{(t)} = \frac{-1}{d\left(\mathbf{X}_{(t)},\mathbf{Y}_i\right)}\mathbf{X}_{(t)}^{\frac{1}{2}}\log\left(\mathbf{X}_{(t)}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}_{(t)}^{-\frac{1}{2}}\right)\mathbf{X}_{(t)}^{\frac{1}{2}} \in \mathrm{T}_{\mathbf{X}_{(t)}}\mathcal{S}_d^{++} \tag{28}$$

$$\mathbf{X}_{(t+1)} = \mathrm{R}_{\mathbf{X}_{(t)}}\left(\pi_{\mathbf{X}_{(t)}}\left(\frac{\partial f}{\partial \mathbf{X}_{(t)}}\right)\right) = \exp_{\mathbf{X}_{(t)}}\left(-\lambda\pi_{\mathbf{X}_{(t)}}\left(\frac{\partial f}{\partial \mathbf{X}_{(t)}}\right)\right) \tag{29}$$

[4] The computation of the derivatives in detail is out of the scope of the thesis. For the interested reader, it is suggested to use references [86], [87], [95]–[99] to derive the presented equations.

Computationally, it is much simpler to use the second approach, as many commonly used deep learning frameworks, such as Python's PyTorch, Tensorflow, Apache MXNet and others, MATLAB's Deep Learning Toolbox and Manopt toolbox [93], Julia's Manifolds.jl and many others, implement a concept called automatic differentiation, which basically uses the chain rule to easily calculate the derivatives of scalar outputs w.r.t. vectors or matrices. The computed gradients are in the Euclidean space, which means it is simply a matter of projecting the gradients to the tangent space of the point in the manifold and then performing the update step.

2.5 Meta-learning on SPD Manifold

Meta-learning on any kind of Riemannian (sub)manifold was not performed until the proposed method of [54] which was formulated on the SPD manifold, and later generalized to many Riemannian (sub)manifolds, with [85]. As the work of this thesis utilized the proposed methods, it will be presented in this chapter, as it is the only meta-learning algorithm so far formulated in spaces of differential geometry.

The RNN topology, and specifically the LSTM variant, has been used in many meta-learning methods as stated earlier, proving that they are a good choice when it comes to developing meta-learning algorithms that are based on neural networks. They are formulated to work with Euclidean vectors, which hinders the usage of matrix-shaped data, but this is easily solved by modifying the shapes of the weights and biases to match the matrix shape that is needed. Even so, the main concern of using the LSTM topology, even if changed to use with matrices, is that the operations performed do not guarantee that the structure of the data used is preserved. For instance, in the SPD manifold, the meta-learning network would be needed to output symmetric matrices, as these would be used as gradients to the SPD matrix parameters that are optimized, since as stated earlier, the gradient generated from an objective function when differentiated w.r.t a SPD matrix, are symmetric matrices in the tangent space defined from that SPD matrix. This means, that the basic matrix multiplications used in the LSTM formulas, must be replaced with another kind of operation that not only performs a similar task but also preserves the structure of the given data. The symmetry of a matrix can be preserved by a bidirectional matrix multiplication, termed bilinear projection, of the data with a weight matrix and its transpose, $O = \mathrm{W}^{\mathrm{T}} X \mathrm{W}$, as proposed in [54], with $\mathrm{W} \in \mathbb{R}^{\mathrm{o} \times \mathrm{d}}$, being the weight matrix, $X \in Sym^d$ and $O \in Sym^o$ being symmetric matrices, with the former being the input and the latter the output, and o being the output dimension and d the input dimension without them necessarily being of different values, i.e., it is possible that $o = d$. Thus, the typical LSTM formulas are changed as seen in eqs. (30) - (35), which is a modification of the LSTM termed matrix LSTM (mLSTM). By replacing the bilinear projection with another one, suitable for the structure of the manifold the data reside in, the authors proposed the generalized matrix LSTM (gmLSTM) in [85].

$$I_{(t)} = \sigma\left(\mathrm{W}_{xi}^{\mathrm{T}} \mathrm{X}_{(t)} \mathrm{W}_{xi} + \mathrm{W}_{hi}^{\mathrm{T}} H_{(t-1)} \mathrm{W}_{hi}\right) \in Sym^o \tag{30}$$

$$F_{(t)} = \sigma\left(\mathrm{W}_{xf}^{\mathrm{T}} \mathrm{X}_{(t)} \mathrm{W}_{xf} + \mathrm{W}_{hf}^{\mathrm{T}} H_{(t-1)} \mathrm{W}_{hf}\right) \in Sym^o \tag{31}$$

$$O_{(t)} = \sigma\left(\mathrm{W}_{xo}^{\mathrm{T}} \mathrm{X}_{(t)} \mathrm{W}_{xo} + \mathrm{W}_{ho}^{\mathrm{T}} H_{(t-1)} \mathrm{W}_{ho}\right) \in Sym^o \tag{32}$$

$$\tilde{C}_{(t)} = \tanh\left(\mathrm{W}_{xc}^{\mathrm{T}} \mathrm{X}_{(t)} \mathrm{W}_{xc} + \mathrm{W}_{hc}^{\mathrm{T}} H_{(t-1)} \mathrm{W}_{hc}\right) \in Sym^o \tag{33}$$

$$C_{(t)} = F_{(t)} \odot C_{(t-1)} + I_{(t)} \odot \tilde{C}_{(t)} \in Sym^o \tag{34}$$

$$H_{(t)} = O_{(t)} \odot \tanh\left(C_{(t)}\right) \in Sym^o \tag{35}$$

Utilizing the mLSTM as well as the retraction and projection equations of (24) and (25), the authors constructed the meta-learning network topology of Figure 8, where $\Gamma_{M^{(t)}}$ is eq. (24). The network learns to calculate the proper value update and learning rate for any given gradient, by having one mLSTM output matrices used to compute a new gradient-like matrix and one mLSTM whose output is used to calculate the learning rate. The training algorithm developed for it only requires the definition of an objective function for the task at hand, as it is defined to be task agnostic. This objective function is transformed to a meta-objective function which should accommodate multiple SPD matrices, the amount of which is denoted by m, if it does not by default, as this reduced oscillations of the updates to the network during training. As one can notice, especially when taking into account the experiments performed, the network learns to optimize and optimizes a single SPD parameter, which means it is not formulated to work with networks that may contain SPD matrices for weights. This is a drawback when compared with traditional handcrafted Riemannian optimizers, but it is alleviated by the presented fact that it converges a lot faster and a lot better. Mathematically, the network topology is expressed by eq. (36) through (43). The mLSTM memory matrices are shared by calculating their element-wise product, as this allows the optimization information from both the weight update and learning rate be shared.

$$S_{l,(t-1)} = \left[H_{l,(t-1)}, C_{l,(t-1)}\right] \tag{36}$$

$$S_{s,(t-1)} = \left[H_{s,(t-1)}, C_{s,(t-1)}\right] \tag{37}$$

$$S_{(t-1)} = S_{l,(t-1)} \odot S_{s,(t-1)} \tag{38}$$

$$S_{l,(t)} = \mathrm{mLSTM}_l\left(\nabla_{\mathbf{M},(t)}, S_{(t-1)}\right) \tag{39}$$

$$S_{s,(t)} = \mathrm{mLSTM}_s\left(\nabla_{\mathbf{M},(t)}, S_{(t-1)}\right) \tag{40}$$

$$U_{(t)} = \pi_{\mathbf{M}_{(t)}}\left(\mathrm{W}_s^{\mathrm{T}}\left(H_{s,(t)} + \nabla_{\mathbf{M},(t)}\right)\mathrm{W}_s\right) \tag{41}$$

$$\lambda_{(t)} = w_l^{\mathrm{T}} H_{l,(t)} w_l \tag{42}$$

$$\mathbf{M}_{(t+1)} = \Gamma_{\mathbf{M}_{(t)}}\left(-\lambda_{(t)} U_{(t)}\right) \tag{43}$$

where $S = [H, C]$ the output of the mLSTM network. In total, the meta-learning network's parameters can be combined in a set $\phi = \{\mathrm{W}_{s,xi}, \mathrm{W}_{s,hi}, \mathrm{W}_{s,xf}, \mathrm{W}_{s,hf}, \mathrm{W}_{s,xo}, \mathrm{W}_{s,ho}, \mathrm{W}_{s,xc}, \mathrm{W}_s, \mathrm{W}_{l,xi}, \mathrm{W}_{l,hi}, \mathrm{W}_{l,xf}, \mathrm{W}_{l,hf}, \mathrm{W}_{l,xo}, \mathrm{W}_{l,ho}, \mathrm{W}_{l,xc}, w_l\}$[5].

The training algorithm for the network is split into two phases. Initially, an experience pool is filled by using a simple Riemannian Gradient Descent optimizer on the task's meta-objective function starting with the initial SPD matrix being the identity matrix, for a predefined number of iterations. Once this observation stage, as it was named, is complete, the learning stage commences. For another predefined number of iterations, the experience pool is randomly sampled and the meta-learning network performs a specific number of optimization steps on the meta-objective function, accumulating a global loss value. Then the gradients of the accumulated loss w.r.t the network's mLSTMs weights are calculated which are used with the ADAM optimization algorithm to update

[5] Assuming one mLSTM is used for in each cell. If multiple are used in sequence, then the set includes their parameters as well.

the network's trainable parameters. If a specific number of training epochs have passed, the SPD matrix is set to the identity and the state matrices are set to matrices filled with zeros, then pushed into the experience pool. If not, the optimized SPD parameter along with the network's state matrices, $\boldsymbol{S}^{(t)}$, are pushed into the experience pool. Once the predetermined learning stage epochs are reached, the training ends and the trained meta-learning optimization network is returned.

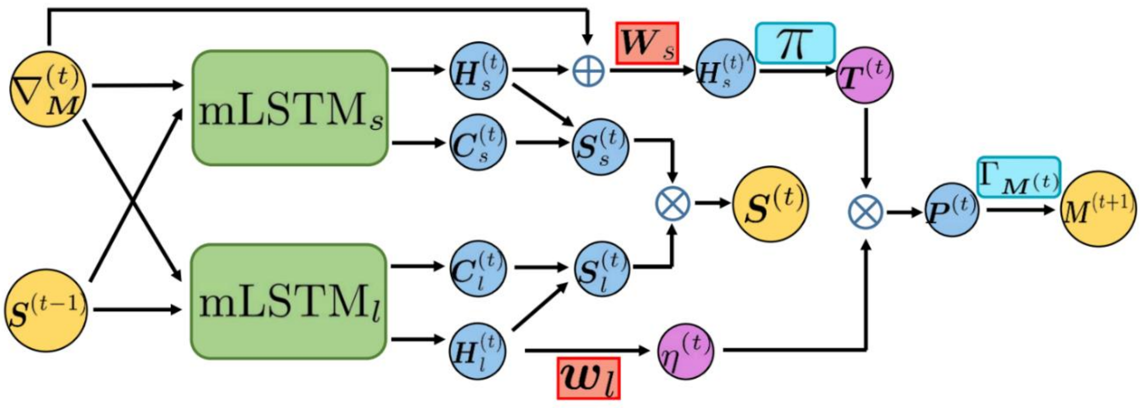


Figure 8: SPD meta-learning network topology. Source: [55], Learning to Optimize on SPD Manifolds, Gao et al.

3 Proposition, Experiments & Results

This chapter covers the data preprocessing steps performed to map the static signature images to SPD points and then to pairwise distance vectors, the verification framework that was developed, the experiments performed with it and the respective results. As a reminder, the target objective is to calculate a 2x2 SPD matrix which when used as the covariance matrix parameter on a Mahalanobis distance of the pairwise distances, it will minimize the result for the similar pairs and maximize it for the dissimilar.

3.1 Datasets & Preprocessing

For the learning and testing purposes of the proposed method, two static signature datasets of Western origin were used: a) the CEDAR Signature Dataset [100], consisting of 55 writers with 24 genuine samples each, and 24 (skilled) forgery samples which were obtained by 20 skillful forgers, and b) the offline part of the MCYT signature corpus [101], MCYT-75 (henceforth MCYT), consisting of 75 writers with 15 genuine and 15 (skilled) forgery samples each, with the forgeries being acquired from subsequent users of the signature acquisition process.

Following the novel work which was initially proposed in [18] on the preprocessing step, after the images are binarized using Otsu's thresholding [102], the binary signature images are morphologically thinned as many times as the corresponding optimal level was found, that being one for the CEDAR database and two for the MCYT dataset. Afterwards, the original image is masked over with its binary, skeletonized counterpart, giving back the grayscale equivalent of the morphologically thinned mask. Lastly, the masked grayscale image are cropped such that the surrounding background is removed.

The feature extraction from the processed grayscale image is performed by applying a typical 10-layer stack filter of eq. (44), resulting in an equivalent 10-layer stack image, and keeping only the values corresponding to signature trace, i.e., the background noise is not taken into account. Once extracted, using eq. (45), each signature image is mapped to a point $\mathbf{C}$ on the SPD manifold.

$$\mathrm{F(I)} = \{\mathrm{I}, |\mathrm{I_x}|, |\mathrm{I_y}|, \sqrt{\mathrm{I_x^2} + \mathrm{I_y^2}}, \tan^{-1}\left(\frac{\mathrm{I_y}}{\mathrm{I_x}}\right), |\mathrm{I_{xx}}|, |\mathrm{I_{yy}}|, |\mathrm{I_{xy}}|, \mathrm{x}, \mathrm{y}\} \in \mathbb{R}^{N \times Mx10} \tag{44}$$

$$\mathbf{C} = \frac{1}{\mathrm{S}-1}\sum_{i=1}^{S}\left(\mathrm{F(I)}_i - mean\big(\mathrm{F(I)}\big)\right)\left(\mathrm{F(I)}_i - mean\big(\mathrm{F(I)}\big)\right)^{\mathrm{T}} \tag{45}$$

In eq. (44), I corresponds to the preprocessed grayscale image, $\mathrm{I_x}, \mathrm{I_y}$ and $\mathrm{I_{xx}}, \mathrm{I_{yy}}$ are the first and seconds directional image derivatives, $\mathrm{I_{xy}}$ is the image derivative w.r.t both directions, $\sqrt{\mathrm{I_x^2} + \mathrm{I_y^2}}$ and $\tan^{-1}\left(\frac{\mathrm{I_y}}{\mathrm{I_x}}\right)$ are the magnitude and direction of the gradient, with the second being normalized in to radians ranging from $[-\pi, \pi)$, and x, y are the signature trace pixel coordinates normalized by the number of rows and columns the resized grayscale signature image has. In eq. (45), S refers to the number of samples contained in the feature matrix $\mathrm{F(I)}$, $\mathrm{F(I)}_i$ corresponds to the i-th feature vector containing all 10 filter values of the corresponding signature trace pixel and $mean\big(\mathrm{F(I)}\big)$ corresponds to the vector containing the mean value of each extracted feature.

From each SPD matrix $\mathbf{C}$, two 5x5 block-diagonal submatrices are identified and extracted, which are also SPD. Then, for each genuine-to-genuine and genuine-to-forgery pair, their submatrix pairs, are employed in order to calculate the geodesic distance, with the help of eq. (17). This equation basically utilizes the difference between two SPD matrices to measure the distance between them, so similarly to [56], the feature domain is transformed to the distance feature domain, with the difference being that the original feature domain is the SPD manifold instead of the Euclidean space, and the distance space is comprised of 2D distance vectors rather than a single distance measure. A visualization of this method is presented in Figure 10, while a visualization of the resulting space for a writer of the CEDAR dataset is presented in Figure 9. Graphs for the rest of the writers of that dataset and all of the writers from the MCYT dataset are presented in the images folder of the online repository of the Python source code used for the experiments[6].

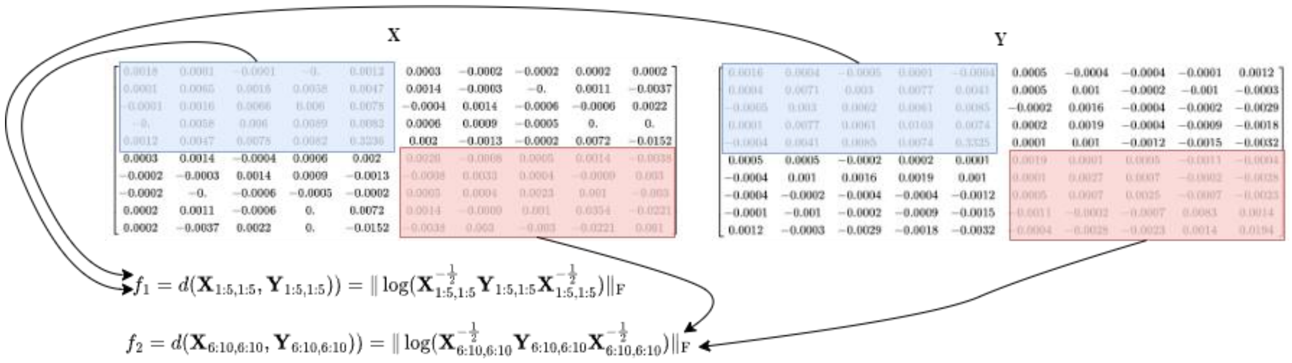

X
Y
$f_1 = d(\mathbf{X}_{1:5,1:5}, \mathbf{Y}_{1:5,1:5}) = \| \log(\mathbf{X}_{1:5,1:5}^{-\frac{1}{2}} \mathbf{Y}_{1:5,1:5} \mathbf{X}_{1:5,1:5}^{-\frac{1}{2}}) \|_F$
$f_2 = d(\mathbf{X}_{6:10,6:10}, \mathbf{Y}_{6:10,6:10}) = \| \log(\mathbf{X}_{6:10,6:10}^{-\frac{1}{2}} \mathbf{Y}_{6:10,6:10} \mathbf{X}_{6:10,6:10}^{-\frac{1}{2}}) \|_F$

Figure 10: A pair of SPD matrices is mapped to a 2D Euclidean distance vector by SPD geodesic distance on SPD submatrices. The subscripts on the matrices denote the range of indices taken in both directions.

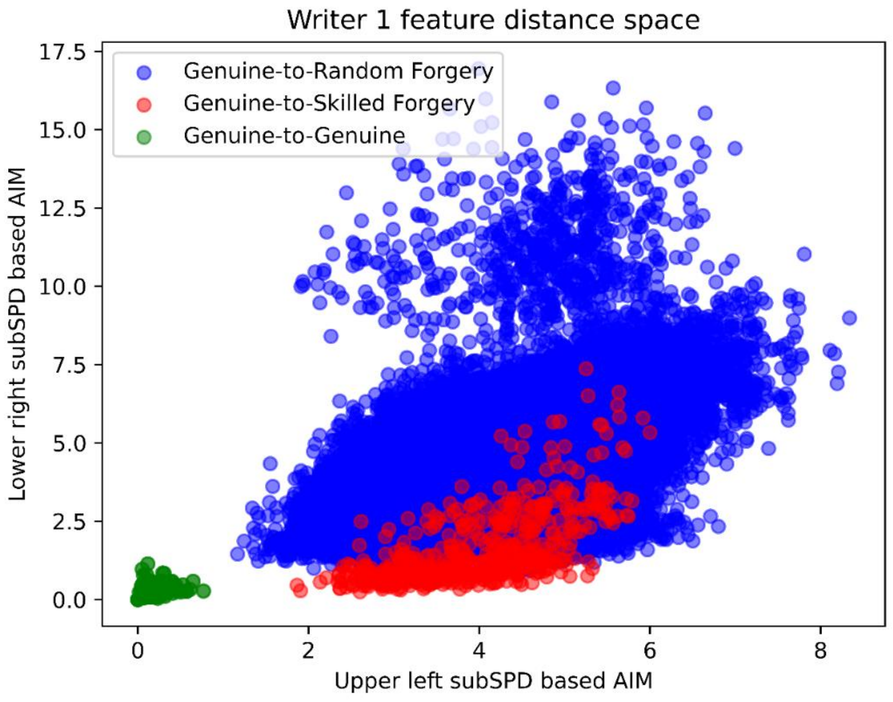


Figure 9: The two – dimensional distance feature space of writer 1 from the CEDAR dataset.

[6] https://bitbucket.org/agiaz/thesis

3.2 Verification Framework

The proposed handwritten signature verification algorithm utilizes a single as well as simple 2×2 SPD matrix which is calculated using the meta-learning method developed in [54], by modifying its training algorithm in order to introduce a parallelization in training both the optimization neural network and the 2x2 SPD parameter. Therefore, we train the optimizer for a few steps at a time and then use it to calculate the updated SPD parameter which is used for validation, with this interchange running for a predetermined number of times or until convergence of the utilized loss metric is used.

In detail, evaluation of the static signature pairwise distances are then employed in the determination of the Mahalanobis distance equation of eq. (46), in which $\mathbf{M}$ denotes the corresponding SPD parameter. Then the contrastive loss is calculated by eq. (47), by including a binary label, $\mathrm{y_{ij,kl}}$, with its value being 0 for dissimilar pairs and 1 for similar pairs, and two hyper-parameters, ζ_d and ζ_s, that are distance thresholds for dissimilar and similar pairs. The subscripts i and k refer to the writer of the dataset, while j and l refer to the sample of each writer. Due to them, the loss function drives the gradients to move the weights towards values that make the result of eq. (46) for dissimilar pairs larger, while for similar pairs smaller.

$$d(x_{ij,kl}, \mathbf{M}) = x_{ij,kl}^{\mathrm{T}} \mathbf{M} x_{ij,kl} \tag{46}$$

$$\begin{aligned} l(D, S, \mathbf{M}) = \frac{1}{\mathrm{D}} \sum_{i,j,k,l \in D} \left(1 - \mathrm{y_{ij,kl}}\right) \max\left(\zeta_d - d(x_{ij,kl}, \mathbf{M})\right)^2 \\ + \frac{1}{\mathrm{S}} \sum_{i,j,k,l \in S} \mathrm{y_{ij,kl}} \max\left(d(x_{ij,kl}, \mathbf{M}) - \zeta_s, 0\right)^2 \end{aligned} \tag{47}$$

The sets D and S are the dissimilar and similar pairs in the minibatch, with each containing the indices of the writers and the indices of their samples that are part of the set. The training process uses a mini-batch training approach, which is why in the loss function, the dissimilar pairs loss is normalized by |D|, the cardinality of the set of dissimilar pairs in the mini-batch, while the similar pairs loss by |S|, the cardinality of the set of similar pairs in the mini-batch. Eq. (47) is the objective function used in our task, which is transformed to the meta-objective function used by the meta-learning network's training algorithm, by the mean value of all loss calculations per SPD matrix, as multiple SPD parameters are utilized to reduce training oscillations. This is depicted in eq. (48).

$$\mathcal{L}(D, S, \mathbf{M}) = \frac{1}{m} \sum_{n=1}^{m} l(D, S, \mathbf{M}_n) \tag{48}$$

3.3 Experiments & Results

The outline of the training algorithm utilized for our framework can be seen in Algorithm 1. It is a modified version of the meta-learner's training algorithm, as stated earlier, by splitting the learning state in multiple segments, with each segment followed by optimizing our final SPD parameter and then performing a validation step, which allows us to stop the training process if convergence is met. Once the training loop is done, either by reaching the predefined epochs or convergence of the validation metric, the most performant SPD parameter found during training is kept and tested.

The training, validation and internal testing is performed by initially splitting the dataset in half, keeping half of the writers for training and validation and half for testing. The first half is split again, with 70% of the sample pairs of each writer being used for training, and the other 30% used for validation. For the training part, we choose to keep the number of genuine-to-forgery pairs equal to that of the genuine-to-genuine for each writer, in order to have a balanced distribution of samples from $\omega^+$and ω^-. The rest of the genuine-to-forgery pairs are used with the other 30% for validation. The validation step utilized the AUC metric of the Receiver Operating Characteristic (ROC) curve in order to declare the optimal operating parameters of the proposed model. The testing step utilizes the True Positive Rate (TPR) and False Positive Rate (FPR) measures from the ROC Curve in order to calculate an EER%. Once the algorithm terminates, the initial two splits are interchanged and the algorithm is run again, i.e., the testing part of the dataset becomes the training-validation one and vice-versa. We perform this form of cross-validation 5 times, thus performing a *5-by-2* cross validation algorithm (CVA).

Using the *5-by-2* CVA, we perform an exhaustive hyperparameter grid search over discrete value ranges for some of the hyperparameters with the CEDAR dataset, with the results of it shown in Table 4[7]. Next, we pick the best performing hyperparameter set, designated by a bold line on Table 4, in order to perform two new *5-by-2* CVA for both the CEDAR and MCYT datasets. For testing, a reference population of 10 genuine samples from each writer are randomly sampled, while the rest of the genuine samples along with the skilled forgeries are used as the set of questioned samples. For each pair between each genuine sample and each questioned sample, the distances are calculated as shown in Figure 9, and are used with the SPD parameter found during the training stage. The resulting distances are stored for post-processing for a total of 10 times. Afterwards, for each questioned sample's result, we keep the minimum, mean and maximum distances found, perform a ROC analysis for each distance type and use the TPR and FPR to calculate the EER%. The results of these are depicted in Table 1. Finally, we calculate a new SPD parameter, with each dataset without splitting for testing, which is then tested on the other. This allows us to see the generalization capabilities of the model to unseen data distributions. Table 2 contains the inter-dataset testing results. The rest of the hyperparameters that were predetermined and not found by the CVA, were set as:

- Batch size = 512 samples
- mLSTM layers = 2
- $\tau =$ 105 if O.I = 300, 205 if O.I = 600
- $T = 5$
- $m = 12$
- $\zeta_s = 0.01$
- $\zeta_d = 1$

[7] Table 1 is in Appendix A due to its size. A legend is also included regarding the table headers.

Table 1
Intra-dataset testing results.

Dataset	D.T.	Fold 1 EER%	Fold 2 EER%	Fold 3 EER%	Fold 4 EER%	Fold 5 EER%	Mean EER%
CEDAR	min	0.91	0.98	0.75	0.75	0.6	**0.8**
CEDAR	mean	2.07	2.29	2.31	2.02	1.71	2.08
CEDAR	max	5.64	6.25	6.73	5.62	5.31	5.91
MCYT	min	4.67	3.91	4.59	4.11	4.71	**4.4**
MCYT	mean	5.71	4.57	5.25	4.93	6.03	5.3
MCYT	max	11.41	8.47	9.31	9.16	11.69	10.01

Table 2
Inter-dataset testing EER%

	CEDAR	MCYT
CEDAR	-	4.51
MCYT	1.22	-

Table 3
Summary of WI-SV on testing EER%

1st Author Ref.	Method	#Sig. Refs.	CEDAR	MCYT-75
Maergner [51]	Graph edit distance	10	5.91	3.91
Maergner [103]	Graph edit distance & Inkball Models	10	-	5.78
Soleimani [31]	HOG with DMML	5	-	13.40
	Same as above	10	-	9.86
Kumar [104]	Surroundedness feat. with MLP/RBF-SVM	1	8.50	-
Liu [52]	Metric learning with MSDN with DCSN	1	4.83	-
	Same as above	1	8.26	-
	Same as above	10	1.75	-
	Same as above	12	1.67	-
Zhu [105]	P2S metric learning	5	5.22	4.86
Hamadene [32]	CT with DCCM	5	2.10	-
Kalera [100]	GSC feats. To Bayes Classifier	16	21.90	-
Zois [25]	Partially ordered sets	5	2.90	3.50
Longjam [26]	Hybrid CNN-BiLSTM Network	N/A	0.00	-
Parcham [106]	CNNCapsNet - CBCapsNet	N/A	0.00	-
Hanif [53]	VLAD	N/A	0.00	-
Li [107]	AVN	N/A	3.77	-
Lin [108]	2C2L	N/A	0.00	-
Wei [109]	IDN	N/A	3.62	-
Souza [24]	DT with Signet	12/10	3.32	2.89
Proposed$_{\text{intra}}$	Image to SPD to $\mathbb{R}^2$, Mahalanobis distance	10	0.61	4.11

Algorithm 1 Cross Validation process of SV Framework

```
Input:  Randomly initialized meta-learning optimizer parameters $\phi$. Randomly initialized SPD
        parameter $\mathbf{M}_{(t)}$. Empty experience pool $\psi = \emptyset$. Initial optimizer state $S_{(0)} = \mathbf{0}_d$. Initial
        meta-objective SPD parameter $\mathbf{L}_{(0)} = \mathbf{I}_d$.
Output: The SPD parameter $\mathbf{M}_{(t)}$.
while   $i \neq observation\_iterations$
        Compute $l$ with $\mathbf{L}_{(t)}$ using eq. (47) and $\nabla_{\mathbf{L}_{(t)}} l$
        Compute $\mathbf{L}_{(t+1)}$by eq. (29)
        Insert $\{(\mathbf{L}_{(t),j}, S_{(0),j})\}_{j=1}^{m}$ to $\psi$
end
while   $itr \neq total\_learning\_iterations$ or $convergence$
        while   $oitr \neq optimizer\_learning\_epochs$
                Randomly sample $\{(\mathbf{L}_{(t),j}, S_{(t),j})\}_{j=1}^{m}$from $\psi$
                while   $step \neq T$
                        Compute $l$ with $\mathbf{L}_{(t)}$ using eq. (47) and $\nabla_{\mathbf{L}_{(t)}} l$
                        Update $\mathbf{L}_{(t+1)}$by eq. (36) through (43)
                end
                Compute the loss $\mathcal{L}$ of the optimizer by eq. (48) and $\nabla_{\phi}\mathcal{L}$
                Update $\phi$ using the ADAM algorithm
                if $t + T > \tau$
                    Set $\mathbf{L}_{(t),j} = \mathbf{I}_d|_{j=1}^{m}$ and $S_{(t),j} = \mathbf{0}_d|_{j=1}^{m}$
                end
                Insert $\{(\mathbf{L}_{(t),j}, S_{(t),j})\}_{j=1}^{m}$to $\psi$
        end
        while   $litr \neq learning\_epochs$
                Compute $l$ with $\mathbf{M}_{(t)}$ by eq. (47) and $\nabla_{\mathbf{M}_{(t)}} l$
                Update $\mathbf{M}_{(t+1)}$by eq. (36) through (43)
        end
        Calculate validation metric $v$
        If $v$ has not changed for a $convergence\_iterations$
            Set $convergence = True$
        end
end
Return $\mathbf{M}_{(t)}$
```

4 Chapter 4: Conclusion

In Table 4, we can see that there are more hyperparameter sets that reach equivalent results, meaning that possibly with a different weight initialization, they could perform similarly or better with the chosen one. It is also important to note, that the minimum distance type performs better in all of the test cases. As it can also be seen from Table 1, the intra-dataset results on CEDAR showcase low error rates, while on MCYT the error is comparable higher than on CEDAR, but it is comparable with that of the State-of-the-Art models referred in the literature. Table 2 inspection allows us to assert that our model is able to generalize, as the results come close with the intra-dataset ones. Cumulatively, the results showcase that the proposed framework, which utilizes low-dimensionality parameters, performs similarly to the State-of-the-Art WI-SV methods, something which is extremely important as it allows for similar results with a smaller computational effort. The mapping from the SPD space to the simple Euclidean $\mathbb{R}^2$ space through a subSPD distance measurement seems to capture important information, similarly to the map proposed in [56]. For some writers of the datasets, this mapping showcases an easily dichotomized space, while for others the dichotomy of their distance space is not that much more difficult. Finally, the combination of the mapping, with the Mahalanobis distance learning and the SPD meta-learner, offered a performant, low-dimensionality, WI-SV algorithm.

The development of a performant WI-SV system with low computational overhead is not impossible as presented in this thesis. Utilizing novelty methods, such as meta-learning on Riemannian manifolds, and the presented subSPD distance feature mapping, a system such as the one mentioned was developed. The proposed framework is able to generalize properly to unseen data with possibly different distributions, is computationally cheap, as it is only a single 2x2 SPD matrix. The results on two popular datasets of Western origin, CEDAR and MCYT-75, portray the capabilities of the proposed method. Comparing with other WI-SV methods, in Table 3, it is clear that the proposed framework is equally performant with more complex, computationally intensive algorithms that utilize a lot more parameters. Finally, on-going research includes the usage of image segments in the mapping to SPD and then to the distance domain, and finding different ways to extract subSPD matrix information between pairs.

Bibliography – References – Online Sources

[1] "Bleary-eyed U.S. election officials turn to signature-verifying software in mail-in surge," *Reuters*, Sep. 24, 2020. Accessed: Jun. 26, 2023. [Online]. Available: https://www.reuters.com/article/us-usa-election-ballot-signatures-idUSKCN26F1UG

[2] "Factbox: U.S. counties using automated signature verification software," *Reuters*, Sep. 24, 2020. Accessed: Jun. 26, 2023. [Online]. Available: https://www.reuters.com/article/us-usa-election-ballot-signatures-softwa-idUSKCN26F1U4

[3] S. N. Srihari, S.-H. Cha, H. Arora, and S. Lee, "Individuality of handwriting," *J. Forensic Sci.*, vol. 47, no. 4, pp. 1–17, 2002.

[4] H.-L. Teulings, *Handwriting Movement Control*. in Handbook of Perception and Action. London: Academic Press, 1996.

[5] N. Sae-Bae, N. Memon, and P. Sooraksa, "Distinctiveness, complexity, and repeatability of online signature templates," *Pattern Recognit.*, vol. 84, pp. 332–344, Dec. 2018, doi: 10.1016/j.patcog.2018.07.024.

[6] M. Faundez-Zanuy, J. Fierrez, M. A. Ferrer, M. Diaz, R. Tolosana, and R. Plamondon, "Handwriting Biometrics: Applications and Future Trends in e-Security and e-Health," *Cogn. Comput.*, vol. 12, no. 5, pp. 940–953, Sep. 2020, doi: 10.1007/s12559-020-09755-z.

[7] D. Impedovo and G. Pirlo, "Automatic Signature Verification: The State of the Art," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 5, pp. 609–635, 2008, doi: 10.1109/TSMCC.2008.923866.

[8] Moises Diaz, Miguel A. Ferrer, Donato Impedovo, Muhammad Imran Malik, Giuseppe Pirlo, and R. Plamondon., "A Perspective Analysis of Handwritten Signature Technology," *ACM Comput Surv 1 1 Artic. 1 January 2018*, 2018, doi: 10.1145/3274658.

[9] M. M. Hameed, R. Ahmad, M. L. M. Kiah, and G. Murtaza, "Machine learning-based offline signature verification systems: A systematic review," *Signal Process. Image Commun.*, vol. 93, p. 116139, Apr. 2021, doi: 10.1016/j.image.2021.116139.

[10] N. Sae-Bae and N. Memon, "Online Signature Verification on Mobile Devices," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 6, pp. 933–947, 2014, doi: 10.1109/TIFS.2014.2316472.

[11] V. Venugopal and S. Sundaram, "An improved online writer identification framework using codebook descriptors," *Pattern Recognit.*, vol. 78, pp. 318–330, Jun. 2018, doi: 10.1016/j.patcog.2018.01.023.

[12] M. Okawa, "Online signature verification using single-template matching with time-series averaging and gradient boosting," *Pattern Recognit.*, vol. 102, p. 107227, Jun. 2020, doi: 10.1016/j.patcog.2020.107227.

[13] M. Zalasiński, K. Łapa, and K. Cpałka, "Prediction of values of the dynamic signature features," *Expert Syst. Appl.*, vol. 104, pp. 86–96, Aug. 2018, doi: 10.1016/j.eswa.2018.03.028.

[14] J. Galbally, M. Diaz-Cabrera, M. A. Ferrer, M. Gomez-Barrero, A. Morales, and J. Fierrez, "On-line signature recognition through the combination of real dynamic data and synthetically generated static data," *Pattern Recognit.*, vol. 48, no. 9, pp. 2921–2934, 9AD, doi: 10.1016/j.patcog.2015.03.019.

[15] X. Xia, Z. Chen, F. Luan, and X. Song, "Signature alignment based on GMM for on-line signature verification," *Pattern Recognit.*, vol. 65, pp. 188–196, May 2017, doi: 10.1016/j.patcog.2016.12.019.

[16] S. Lai, L. Jin, Y. Zhu, Z. Li, and L. Lin, "SynSig2Vec: Forgery-Free Learning of Dynamic Signature Representations by Sigma Lognormal-Based Synthesis and 1D CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6472–6485, 2022, doi: 10.1109/TPAMI.2021.3087619.

[17] R. Ghosh, "A Recurrent Neural Network based deep learning model for offline signature verification and recognition system," *Expert Syst. Appl.*, vol. 168, p. 114249, Apr. 2021, doi: 10.1016/j.eswa.2020.114249.

[18] E. N. Zois, D. Tsourounis, I. Theodorakopoulos, A. L. Kesidis, and G. Economou, "A Comprehensive Study of Sparse Representation Techniques for Offline Signature Verification," *IEEE Trans. Biom. Behav. Identity Sci.*, vol. 1, no. 1, pp. 68–81, 2019, doi: 10.1109/TBIOM.2019.2897802.

[19] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks," *Pattern Recognit.*, vol. 70, pp. 163–176, 2017, doi: 10.1016/j.patcog.2017.05.012.

[20] T. M. Maruyama, L. S. Oliveira, A. S. Britto, and R. Sabourin, "Intrapersonal Parameter Optimization for Offline Handwritten Signature Augmentation," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1335–1350, 2021, doi: 10.1109/TIFS.2020.3033442.

[21] J. Iacovacci and L. Lacasa, "Sequential motif profile of natural visibility graphs," *Phys. Rev. E*, vol. 94, no. 5, p. 052309, 11AD, doi: 10.1103/PhysRevE.94.052309.

[22] E. N. Zois, S. Said, D. Tsourounis, and A. Alexandridis, "Subscripto multiplex: A Riemannian symmetric positive definite strategy for offline signature verification," *Pattern Recognit. Lett.*, vol. 167, pp. 67–74, Mar. 2023, doi: 10.1016/j.patrec.2023.02.002.

[23] W. Bouamra, C. Djeddi, B. Nini, M. Diaz, and I. Siddiqi, "Towards the design of an offline signature verifier based on a small number of genuine samples for training," *Expert Syst. Appl.*, vol. 107, pp. 182–195, Jan. 10AD, doi: 10.1016/j.eswa.2018.04.035.

[24] V. L. F. Souza, A. L. I. Oliveira, R. M. O. Cruz, and R. Sabourin, "A white-box analysis on the writer-independent dichotomy transformation applied to offline handwritten signature verification," *Expert Syst. Appl.*, vol. 154, p. 113397, 2020, doi: 10.1016/j.eswa.2020.113397.

[25] E. N. Zois, A. Alexandridis, and G. Economou, "Writer independent offline signature verification based on asymmetric pixel relations and unrelated training-testing datasets," *Expert Syst. Appl.*, vol. 125, pp. 14–32, Jul. 2019, doi: 10.1016/j.eswa.2019.01.058.

[26] T. Longjam, D. R. Kisku, and P. Gupta, "Writer independent handwritten signature verification on multi-scripted signatures using hybrid CNN-BiLSTM: A novel approach," *Expert Syst. Appl.*, vol. 214, p. 119111, Mar. 2023, doi: 10.1016/j.eswa.2022.119111.

[27] S. N. Srihari, A. Xu, and M. K. Kalera, "Learning strategies and classification methods for off-line signature verification," presented at the Ninth International Workshop on frontiers in Handwriting Recognition, 2004., 2004, pp. 161–166. doi: 10.1109/IWFHR.2004.61.

[28] D. Rivard, E. Granger, and R. Sabourin, "Multi-feature extraction and selection in writer-independent off-line signature verification," *Int. J. Doc. Anal. Recognit.*, vol. 16, no. 1, pp. 83–103, 2013, doi: 10.1007/s10032-011-0180-6.

[29] G. S. Eskander, R. Sabourin, and E. Granger, "Hybrid writer-independent writer-dependent offline signature verification system," *IET Biom.*, vol. 2, no. 4, pp. 169–181, 2013, doi: 10.1049/iet-bmt.2013.0024.

[30] D. Bertolini, L. S. Oliveira, E. Justino, and R. Sabourin, "Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers," *Pattern Recognit.*, vol. 43, no. 1, pp. 387–396, 2010, doi: 10.1016/j.patcog.2009.05.009.

[31] A. Soleimani, B. N. Araabi, and K. Fouladi, "Deep Multitask Metric Learning for Offline Signature Verification," *Pattern Recognit. Lett.*, vol. 80, pp. 84–90, 2016, doi: 10.1016/j.patrec.2016.05.023.

[32] A. Hamadene and Y. Chibani, "One-Class Writer-Independent Offline Signature Verification Using Feature Dissimilarity Thresholding," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1226–1238, 2016, doi: 10.1109/TIFS.2016.2521611.

[33] E. Pękalska and R. P. W. Duin, "Dissimilarity representations allow for building good classifiers," *Pattern Recognit. Lett.*, vol. 23, no. 8, pp. 943–956, 2002, doi: 10.1016/S0167-8655(02)00024-7.

[34] M. I. Malik and M. Liwicki, "From Terminology to Evaluation: Performance Assessment of Automatic Signature Verification Systems," presented at the 2012 International Conference

onFrontiers in Handwriting Recognition, Sep. 2012, pp. 613–618. doi: 10.1109/ICFHR.2012.205.

[35] S. Masoudnia, O. Mersa, B. N. Araabi, A.-H. Vahabie, M. A. Sadeghi, and M. N. Ahmadabadi, “Multi-Representational Learning for Offline Signature Verification using Multi-Loss Snapshot Ensemble of CNNs,” *Expert Syst. Appl.*, Mar. 2019, doi: 10.1016/j.eswa.2019.03.040.

[36] O. Tuzel, F. Porikli, and P. Meer, “Region Covariance: A Fast Descriptor for Detection and Classification,” presented at the European Conference on Computer Vision, ECCV 2006, H. Bischof and A. Pinz, Eds., in Computer Vision – ECCV 2006. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 589–600.

[37] P. Koniusz, F. Yan, P. Gosselin, and K. Mikolajczyk, “Higher-Order Occurrence Pooling for Bags-of-Words: Visual Concept Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 313–326, 2017, doi: 10.1109/TPAMI.2016.2545667.

[38] Z. Gao, Y. Wu, M. Harandi, and Y. Jia, “A Robust Distance Measure for Similarity-Based Classification on the SPD Manifold,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3230–3244, 2020, doi: 10.1109/TNNLS.2019.2939177.

[39] Z. Gao, J. Xie, Q. Wang, and P. Li, “Global Second-Order Pooling Convolutional Networks,” presented at the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2019, pp. 3019–3028. doi: 10.1109/CVPR.2019.00314.

[40] Q. Wang, P. Li, Q. Hu, P. Zhu, and W. Zuo, “Deep Global Generalized Gaussian Networks,” presented at the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2019, pp. 5075–5083. doi: 10.1109/CVPR.2019.00522.

[41] P. Koniusz and H. Zhang, “Power Normalizations in Fine-Grained Image, Few-Shot Image and Graph Classification,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 2, pp. 591–609, 2022, doi: 10.1109/TPAMI.2021.3107164.

[42] X. Pennec, P. Fillard, and N. Ayache, “A Riemannian Framework for Tensor Computing,” *Int J Comput Vis.*, vol. 66, no. 1, pp. 41–66, 2006, doi: 10.1007/s11263-005-3222-z.

[43] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, “Log-Euclidean metrics for fast and simple calculus on diffusion tensors,” *Magn. Reson. Med.*, vol. 56, no. 2, pp. 411–421, 2006, doi: 10.1002/mrm.20965.

[44] A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos, “Efficient similarity search for covariance matrices via the Jensen-Bregman LogDet Divergence,” in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2399–2406. doi: 10.1109/ICCV.2011.6126523.

[45] M. Hein and O. Bousquet, “Hilbertian Metrics and Positive Definite Kernels on Probability Measures,” presented at the International Conference on Artificial Intelligence and Statistics, 2005. Accessed: Jun. 27, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Hilbertian-Metrics-and-Positive-Definite-Kernels-on-Hein-Bousquet/782269ba228ab3c245ea62f8da2fee141a40f486

[46] L. Malagò, L. Montrucchio, and G. Pistone, “Wasserstein Riemannian geometry of Gaussian densities,” *Inf. Geom.*, vol. 1, no. 2, pp. 137–179, Dec. 2018, doi: 10.1007/s41884-018-0014-4.

[47] A. Han, B. Mishra, P. Jawanpuria, and J. Gao, “Learning with symmetric positive definite matrices via generalized Bures-Wasserstein geometry.” arXiv, Jun. 08, 2023. doi: 10.48550/arXiv.2110.10464.

[48] Z. Lin, “Riemannian Geometry of Symmetric Positive Definite Matrices via Cholesky Decomposition,” *SIAM J. Matrix Anal. Appl.*, vol. 40, no. 4, pp. 1353–1370, Jan. 2019, doi: 10.1137/18M1221084.

[49] L. Zhou, L. Wang, J. Zhang, Y. Shi, and Y. Gao, “Revisiting Metric Learning for SPD Matrix Based Visual Representation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 7111–7119. doi: 10.1109/CVPR.2017.752.

[50] H. Rantzsch, H. Yang, and C. Meinel, “Signature Embedding: Writer Independent Offline Signature Verification with Deep Metric Learning,” presented at the International Symposium on Visual Computing, Cham: Springer International Publishing, 2016, pp. 616–625.

[51] P. Maergner *et al.*, “Combining graph edit distance and triplet networks for offline signature verification,” *Pattern Recognit. Lett.*, vol. 125, pp. 527–533, Jul. 2019, doi: 10.1016/j.patrec.2019.06.024.

[52] L. Liu, L. Huang, F. Yin, and Y. Chen, “Offline signature verification using a region based deep metric learning network,” *Pattern Recognit.*, vol. 118, p. 108009, Oct. 2021, doi: 10.1016/j.patcog.2021.108009.

[53] M. S. Hanif and M. Bilal, “A Metric Learning Approach for Offline Writer Independent Signature Verification,” *Pattern Recognit. Image Anal.*, vol. 30, no. 4, pp. 795–804, Oct. 2020, doi: 10.1134/S1054661820040173.

[54] Z. Gao, Y. Wu, Y. Jia, and M. Harandi, “Learning to Optimize on SPD Manifolds,” presented at the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2020, pp. 7697–7706. doi: 10.1109/CVPR42600.2020.00772.

[55] L. C. Alewijnse, C.E. van den Heuvel, and R. D. Stoel, “Analysis of signature complexity,” *J. Forensic Doc. Exam.*, vol. 21, pp. 37–49, 2011.

[56] S.-H. Cha and S. Srihari, “Writer Identification: Statistical Analysis and Dichotomizer,” in *Advances in Pattern Recognition*, F. Ferri, J. Iñesta, A. Amin, and P. Pudil, Eds., in Lecture Notes in Computer Science, vol. 1876. Springer Berlin Heidelberg, 2000, pp. 123–132. doi: 10.1007/3-540-44522-6_13.

[57] M. Deviterne-Lapeyre and S. Ibrahim, “Interpol questioned documents review 2019–2022,” *Forensic Sci. Int. Synergy*, vol. 6, p. 100300, 2023, doi: 10.1016/j.fsisyn.2022.100300.

[58] A. Soleimani, K. Fouladi, and B. N. Araabi, “UTSig: A Persian offline signature dataset,” *IET Biom.*, vol. 6, no. 1, pp. 1–8, 2016, doi: 10.1049/iet-bmt.2015.0058.

[59] B. Li and N. Li, “Handwriting expertise reliability: A review,” *J. Forensic Sci. Med.*, vol. 5, no. 4, p. 181, 2019, doi: 10.4103/jfsm.jfsm_44_19.

[60] J. Sita, B. Found, and D. K. Rogers, “Forensic handwriting examiners’ expertise for signature comparison,” *J. Forensic Sci.*, vol. 47, no. 5, pp. 1117–1124, Sep. 2002.

[61] S.-I. Amari, “Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements,” *IEEE Trans. Comput.*, vol. C–21, no. 11, pp. 1197–1206, Nov. 1972, doi: 10.1109/T-C.1972.223477.

[62] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[63] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Phys. Nonlinear Phenom.*, vol. 404, p. 132306, Mar. 2020, doi: 10.1016/j.physd.2019.132306.

[64] J. L. Elman, “Finding Structure in Time,” *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990, doi: 10.1207/s15516709cog1402_1.

[65] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[66] Q. V. Le, “A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks,” 2015. Accessed: Jun. 08, 2023. [Online]. Available: https://www.semanticscholar.org/paper/A-Tutorial-on-Deep-Learning-Part-2%3A-Autoencoders%2C-Le/94ebfd63a2f8133220cd5e4ad011002270b1489b#cited-papers

[67] Y. Nesterov, “A method for solving the convex programming problem with convergence rate O(1/k^2),” *Proc. USSR Acad. Sci.*, 1983, Accessed: Jun. 17, 2023. [Online]. Available: https://www.semanticscholar.org/paper/A-method-for-solving-the-convex-programming-problem-Nesterov/8d3a318b62d2e970122da35b2a2e70a5d12cc16f

[68] P. Tseng, “An Incremental Gradient(-Projection) Method with Momentum Term and Adaptive Stepsize Rule,” *SIAM J. Optim.*, vol. 8, no. 2, pp. 506–531, May 1998, doi: 10.1137/S1052623495294797.

[69] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: the RPROP algorithm,” in *IEEE International Conference on Neural Networks*, Mar. 1993, pp. 586–591 vol.1. doi: 10.1109/ICNN.1993.298623.

[70] G. Hinton, N. Srivastava, and K. Swersky, “Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude.” University of Toronto, Nov. 02, 2014. Accessed: Jun. 17, 2023. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[71] J. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2121–2159, Jul. 2011.

[72] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” 2014, doi: 10.48550/ARXIV.1412.6980.

[73] D. J. Chalmers, “The Evolution of Learning: An Experiment in Genetic Connectionism,” in *Connectionist Models*, Elsevier, 1991, pp. 81–90. doi: 10.1016/B978-1-4832-1448-1.50014-7.

[74] Y. Bengio, S. Bengio, and J. Cloutier, “Learning a synaptic learning rule,” in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, Seattle, WA, USA: IEEE, 1991, p. 969. doi: 10.1109/IJCNN.1991.155621.

[75] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, “On the Optimization of a Synaptic Learning Rule,” 2007. Accessed: Jun. 17, 2023. [Online]. Available: https://www.semanticscholar.org/paper/On-the-Optimization-of-a-Synaptic-Learning-Rule-Bengio-Bengio/8784f905f4f9fb6fa4a3cc9b0faa5b5479c687ec

[76] S. Bengio, Y. Bengio, and J. Cloutier, “On the search for new learning rules for ANNs,” *Neural Process. Lett.*, vol. 2, no. 4, pp. 26–30, Jul. 1995, doi: 10.1007/BF02279935.

[77] S. Hochreiter, A. S. Younger, and P. R. Conwell, “Learning to Learn Using Gradient Descent,” in *Artificial Neural Networks — ICANN 2001*, G. Dorffner, H. Bischof, and K. Hornik, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 87–94. doi: 10.1007/3-540-44668-0_13.

[78] K. Li and J. Malik, “Learning to Optimize.” arXiv, Jun. 06, 2016. doi: 10.48550/arXiv.1606.01885.

[79] M. Andrychowicz *et al.*, “Learning to learn by gradient descent by gradient descent.” arXiv, Nov. 30, 2016. doi: 10.48550/arXiv.1606.04474.

[80] S. Ravi and H. Larochelle, “Optimization as a Model for Few-Shot Learning,” presented at the International Conference on Learning Representations, Nov. 2016. Accessed: Jun. 17, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Optimization-as-a-Model-for-Few-Shot-Learning-Ravi-Larochelle/29c887794eed2ca9462638ff853e6fe1ab91d5d8

[81] O. Wichrowska *et al.*, “Learned Optimizers that Scale and Generalize.” arXiv, Sep. 07, 2017. doi: 10.48550/arXiv.1703.04813.

[82] A. A. Rusu *et al.*, “Meta-Learning with Latent Embedding Optimization.” arXiv, Mar. 26, 2019. doi: 10.48550/arXiv.1807.05960.

[83] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, “Neural Optimizer Search with Reinforcement Learning.” arXiv, Sep. 22, 2017. doi: 10.48550/arXiv.1709.07417.

[84] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-Learning with Differentiable Convex Optimization.” arXiv, Apr. 23, 2019. doi: 10.48550/arXiv.1904.03758.

[85] Z. Gao, Y. Wu, X. Fan, M. Harandi, and Y. Jia, “Learning to Optimize on Riemannian Manifolds,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5935–5952, May 2023, doi: 10.1109/TPAMI.2022.3215702.

[86] P.-A. Absil, R. Mahoney, and R. Sepulchre, “Optimization Algorithms on Matrix Manifolds,” 1999. https://press.princeton.edu/absil (accessed Jun. 17, 2023).

[87] N. Boumal, *An introduction to optimization on smooth manifolds*. Cambridge ; New York, NY: Cambridge University Press, 2023.

[88] C. Atkinson and A. F. S. Mitchell, "Rao's Distance Measure," *Sankhyā Indian J. Stat. Ser. 1961-2002*, vol. 43, no. 3, pp. 345–365, 1981.

[89] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian Detection via Classification on Riemannian Manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1713–1727, 2008, doi: 10.1109/TPAMI.2008.75.

[90] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Geometric Means in a Novel Vector Space Structure on Symmetric Positive-Definite Matrices," *SIAM J. Matrix Anal. Appl.*, vol. 29, no. 1, pp. 328–347, Jan. 2007, doi: 10.1137/050637996.

[91] S. D. Axen, M. Baran, R. Bergmann, and K. Rzecki, "Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds." arXiv, Jun. 12, 2023. doi: 10.48550/arXiv.2106.08777.

[92] Bergmann, Ronny, "Manopt.jl." Zenodo, Jun. 15, 2023. doi: 10.5281/ZENODO.4290905.

[93] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab toolbox for optimization on manifolds," 2013, doi: 10.48550/ARXIV.1308.5200.

[94] J. Townsend, N. Koep, and S. Weichwald, "Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation." arXiv, Sep. 08, 2016. doi: 10.48550/arXiv.1603.03236.

[95] T. Parr and J. Howard, "The Matrix Calculus You Need For Deep Learning." arXiv, Jul. 02, 2018. doi: 10.48550/arXiv.1802.01528.

[96] K. B. Petersen and M. S. Pedersen, "The Matrix Cookbook," *Univ. Waterloo*.

[97] I. Chavel, *Riemannian Geometry: A Modern Introduction*, 2nd ed. Cambridge University Press, 2006. doi: 10.1017/CBO9780511616822.

[98] R. Bhatia, *Positive definite matrices*. in Princeton series in applied mathematics. Princeton Oxford: Princeton University Press, 2015.

[99] M. P. do Carmo, *Riemannian Geometry*. Accessed: Jun. 24, 2023. [Online]. Available: https://link.springer.com/book/9780817634902

[100] M. K. Kalera, S. Srihari, and A. Xu, "Offline signature verification and identification using distance statistics," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 18, no. 07, pp. 1339–1360, Nov. 2004, doi: 10.1142/S0218001404003630.

[101] J. Ortega-Garcia *et al.*, "MCYT baseline corpus: a bimodal biometric database," *IEE Proc. - Vis. Image Signal Process.*, vol. 150, no. 6, pp. 395–401, Dec. 2003, doi: 10.1049/ip-vis:20031078.

[102] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, 1979, doi: 10.1109/TSMC.1979.4310076.

[103] P. Maergner, N. Howe, K. Riesen, R. Ingold, and A. Fischer, "Offline Signature Verification Via Structural Methods: Graph Edit Distance and Inkball Models," presented at the 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Aug. 2018, pp. 163–168. doi: 10.1109/ICFHR-2018.2018.00037.

[104] R. Kumar, J. D. Sharma, and B. Chanda, "Writer-independent off-line signature verification using surroundedness feature," *Pattern Recognit. Lett.*, vol. 33, no. 3, pp. 301–308, 2012, doi: 10.1016/j.patrec.2011.10.009.

[105] Y. Zhu, S. Lai, Z. Li, and L. Jin, "Point-to-Set Similarity Based Deep Metric Learning for Offline Signature Verification," presented at the 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), Sep. 2020, pp. 282–287. doi: 10.1109/ICFHR2020.2020.00059.

[106] E. Parcham, M. Ilbeygi, and M. Amini, "CBCapsNet: A novel writer-independent offline signature verification model using a CNN-based architecture and capsule neural networks," *Expert Syst. Appl.*, vol. 185, p. 115649, Dec. 2021, doi: 10.1016/j.eswa.2021.115649.

[107] H. Li, P. Wei, and P. Hu, “AVN: An Adversarial Variation Network Model for Handwritten Signature Verification,” *IEEE Trans. Multimed.*, pp. 1–1, 2021, doi: 10.1109/TMM.2021.3056217.

[108] C. Li, F. Lin, Z. Wang, G. Yu, L. Yuan, and H. Wang, “DeepHSV: User-Independent Offline Signature Verification Using Two-Channel CNN,” in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, Sep. 2019, pp. 166–171. doi: 10.1109/ICDAR.2019.00035.

[109] P. Wei, H. Li, and P. Hu, “Inverse Discriminative Networks for Handwritten Signature Verification,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5764–5772.

Appendix A: Table 1

- D.T. : Distance Type
- O.I. : Observation Iterations
- O.L.R.: Optimizer Learning Rate
- H.L.R.: Handcrafted optimizer Learning Rate
- L.E.: Learner Epochs
- R.F.P. : Random Forgery Percentage

Table 4
5-by-2 cross validation on CEDAR dataset

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
min	300	200	0.001	0.1	50	0	0.81
mean	300	200	0.001	0.1	50	0	2.31
max	300	200	0.001	0.1	50	0	6.35
min	300	200	0.001	0.1	50	0.5	1.03
mean	300	200	0.001	0.1	50	0.5	2.78
max	300	200	0.001	0.1	50	0.5	7.1
min	300	200	0.001	0.1	50	1	0.86
mean	300	200	0.001	0.1	50	1	2.46
max	300	200	0.001	0.1	50	1	6.37
min	300	200	0.0001	0.1	50	0	1.04
mean	300	200	0.0001	0.1	50	0	2.54
max	300	200	0.0001	0.1	50	0	6.39
min	300	200	0.0001	0.1	50	0.5	1.08
mean	300	200	0.0001	0.1	50	0.5	2.88
max	300	200	0.0001	0.1	50	0.5	7.01
min	300	200	0.0001	0.1	50	1	1.07
mean	300	200	0.0001	0.1	50	1	2.41
max	300	200	0.0001	0.1	50	1	6.56
min	300	200	1.00E-05	0.1	50	0	0.76
mean	300	200	1.00E-05	0.1	50	0	2.09
max	300	200	1.00E-05	0.1	50	0	5.99
min	300	200	1.00E-05	0.1	50	0.5	1.09

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
mean	300	200	1.00E-05	0.1	50	0.5	2.58
max	300	200	1.00E-05	0.1	50	0.5	6.52
min	300	200	1.00E-05	0.1	50	1	1.05
mean	300	200	1.00E-05	0.1	50	1	2.35
max	300	200	1.00E-05	0.1	50	1	6.2
min	300	200	1.00E-06	0.1	50	0	0.94
mean	300	200	1.00E-06	0.1	50	0	2.51
max	300	200	1.00E-06	0.1	50	0	6.29
min	300	200	1.00E-06	0.1	50	0.5	1.03
mean	300	200	1.00E-06	0.1	50	0.5	2.63
max	300	200	1.00E-06	0.1	50	0.5	6.86
min	300	200	1.00E-06	0.1	50	1	0.77
mean	300	200	1.00E-06	0.1	50	1	2.03
max	300	200	1.00E-06	0.1	50	1	5.72
min	600	200	0.001	0.01	50	0	0.85
mean	600	200	0.001	0.01	50	0	2.3
max	600	200	0.001	0.01	50	0	5.65
min	**600**	**200**	**0.001**	**0.01**	**50**	**0.5**	**0.61**
mean	600	200	0.001	0.01	50	0.5	1.69
max	600	200	0.001	0.01	50	0.5	5.23
min	600	200	0.001	0.01	50	1	1.09
mean	600	200	0.001	0.01	50	1	2.89
max	600	200	0.001	0.01	50	1	7.14
min	600	200	0.0001	0.01	50	0	1.04
mean	600	200	0.0001	0.01	50	0	2.75
max	600	200	0.0001	0.01	50	0	7.06
min	600	200	0.0001	0.01	50	0.5	0.83
mean	600	200	0.0001	0.01	50	0.5	2.31
max	600	200	0.0001	0.01	50	0.5	6.26
min	600	200	0.0001	0.01	50	1	0.89
mean	600	200	0.0001	0.01	50	1	2.38
max	600	200	0.0001	0.01	50	1	6.45

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
min	600	200	1.00E-05	0.01	50	0	0.81
mean	600	200	1.00E-05	0.01	50	0	2.28
max	600	200	1.00E-05	0.01	50	0	6.19
min	600	200	1.00E-05	0.01	50	0.5	0.87
mean	600	200	1.00E-05	0.01	50	0.5	2.3
max	600	200	1.00E-05	0.01	50	0.5	5.96
min	600	200	1.00E-05	0.01	50	1	1.17
mean	600	200	1.00E-05	0.01	50	1	2.76
max	600	200	1.00E-05	0.01	50	1	6.97
min	600	200	1.00E-06	0.01	50	0	0.73
mean	600	200	1.00E-06	0.01	50	0	1.91
max	600	200	1.00E-06	0.01	50	0	5.59
min	600	200	1.00E-06	0.01	50	0.5	0.82
mean	600	200	1.00E-06	0.01	50	0.5	2.23
max	600	200	1.00E-06	0.01	50	0.5	5.99
min	600	200	1.00E-06	0.01	50	1	0.72
mean	600	200	1.00E-06	0.01	50	1	1.94
max	600	200	1.00E-06	0.01	50	1	5.66
min	300	200	0.001	0.1	100	0	1.64
mean	300	200	0.001	0.1	100	0	3.65
max	300	200	0.001	0.1	100	0	8.32
min	300	200	0.001	0.1	100	0.5	1.06
mean	300	200	0.001	0.1	100	0.5	2.63
max	300	200	0.001	0.1	100	0.5	6.32
min	300	200	0.001	0.1	100	1	0.67
mean	300	200	0.001	0.1	100	1	1.85
max	300	200	0.001	0.1	100	1	5.39
min	300	200	0.0001	0.1	100	0	0.78
mean	300	200	0.0001	0.1	100	0	2.19
max	300	200	0.0001	0.1	100	0	5.98
min	300	200	0.0001	0.1	100	0.5	0.88
mean	300	200	0.0001	0.1	100	0.5	2.36

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
max	300	200	0.0001	0.1	100	0.5	6.15
min	300	200	0.0001	0.1	100	1	0.93
mean	300	200	0.0001	0.1	100	1	2.5
max	300	200	0.0001	0.1	100	1	6.64
min	300	200	1.00E-05	0.1	100	0	0.83
mean	300	200	1.00E-05	0.1	100	0	2.42
max	300	200	1.00E-05	0.1	100	0	6.46
min	300	200	1.00E-05	0.1	100	0.5	0.81
mean	300	200	1.00E-05	0.1	100	0.5	2.06
max	300	200	1.00E-05	0.1	100	0.5	5.71
min	300	200	1.00E-05	0.1	100	1	1.01
mean	300	200	1.00E-05	0.1	100	1	2.59
max	300	200	1.00E-05	0.1	100	1	6.71
min	300	200	1.00E-06	0.1	100	0	0.87
mean	300	200	1.00E-06	0.1	100	0	2.27
max	300	200	1.00E-06	0.1	100	0	6.19
min	300	200	1.00E-06	0.1	100	0.5	1.36
mean	300	200	1.00E-06	0.1	100	0.5	3.33
max	300	200	1.00E-06	0.1	100	0.5	7.65
min	300	200	1.00E-06	0.1	100	1	0.91
mean	300	200	1.00E-06	0.1	100	1	2.48
max	300	200	1.00E-06	0.1	100	1	6.39
min	600	200	0.001	0.01	100	0	1.19
mean	600	200	0.001	0.01	100	0	3.06
max	600	200	0.001	0.01	100	0	7.66
min	600	200	0.001	0.01	100	0.5	0.86
mean	600	200	0.001	0.01	100	0.5	2.3
max	600	200	0.001	0.01	100	0.5	6.49
min	600	200	0.001	0.01	100	1	0.94
mean	600	200	0.001	0.01	100	1	2.52
max	600	200	0.001	0.01	100	1	6.56
min	600	200	0.0001	0.01	100	0	0.86

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
mean	600	200	0.0001	0.01	100	0	2.47
max	600	200	0.0001	0.01	100	0	6.29
min	600	200	0.0001	0.01	100	0.5	0.73
mean	600	200	0.0001	0.01	100	0.5	2.12
max	600	200	0.0001	0.01	100	0.5	5.89
min	600	200	0.0001	0.01	100	1	1.02
mean	600	200	0.0001	0.01	100	1	2.75
max	600	200	0.0001	0.01	100	1	7.04
min	600	200	1.00E-05	0.01	100	0	0.67
mean	600	200	1.00E-05	0.01	100	0	1.81
max	600	200	1.00E-05	0.01	100	0	5.27
min	600	200	1.00E-05	0.01	100	0.5	0.95
mean	600	200	1.00E-05	0.01	100	0.5	2.28
max	600	200	1.00E-05	0.01	100	0.5	6
min	600	200	1.00E-05	0.01	100	1	1.22
mean	600	200	1.00E-05	0.01	100	1	2.79
max	600	200	1.00E-05	0.01	100	1	7.18
min	600	200	1.00E-06	0.01	100	0	0.81
mean	600	200	1.00E-06	0.01	100	0	2.19
max	600	200	1.00E-06	0.01	100	0	5.87
min	600	200	1.00E-06	0.01	100	0.5	1.46
mean	600	200	1.00E-06	0.01	100	0.5	3.07
max	600	200	1.00E-06	0.01	100	0.5	7.12
min	600	200	1.00E-06	0.01	100	1	0.83
mean	600	200	1.00E-06	0.01	100	1	2.09
max	600	200	1.00E-06	0.01	100	1	5.63
min	300	200	0.001	0.1	150	0	2.53
mean	300	200	0.001	0.1	150	0	3.95
max	300	200	0.001	0.1	150	0	8.11
min	300	200	0.001	0.1	150	0.5	0.81
mean	300	200	0.001	0.1	150	0.5	2.31
max	300	200	0.001	0.1	150	0.5	6.65

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
min	300	200	0.001	0.1	150	1	1.12
mean	300	200	0.001	0.1	150	1	3
max	300	200	0.001	0.1	150	1	7.06
min	300	200	0.0001	0.1	150	0	1
mean	300	200	0.0001	0.1	150	0	2.64
max	300	200	0.0001	0.1	150	0	6.52
min	300	200	0.0001	0.1	150	0.5	1.18
mean	300	200	0.0001	0.1	150	0.5	2.97
max	300	200	0.0001	0.1	150	0.5	7.17
min	300	200	0.0001	0.1	150	1	0.97
mean	300	200	0.0001	0.1	150	1	2.52
max	300	200	0.0001	0.1	150	1	6.41
min	300	200	1.00E-05	0.1	150	0	0.79
mean	300	200	1.00E-05	0.1	150	0	2.16
max	300	200	1.00E-05	0.1	150	0	6.07
min	300	200	1.00E-05	0.1	150	0.5	1.34
mean	300	200	1.00E-05	0.1	150	0.5	3.23
max	300	200	1.00E-05	0.1	150	0.5	8.04
min	300	200	1.00E-05	0.1	150	1	0.98
mean	300	200	1.00E-05	0.1	150	1	2.66
max	300	200	1.00E-05	0.1	150	1	6.86
min	300	200	1.00E-06	0.1	150	0	1.27
mean	300	200	1.00E-06	0.1	150	0	3.26
max	300	200	1.00E-06	0.1	150	0	8.19
min	300	200	1.00E-06	0.1	150	0.5	0.93
mean	300	200	1.00E-06	0.1	150	0.5	2.48
max	300	200	1.00E-06	0.1	150	0.5	6.76
min	300	200	1.00E-06	0.1	150	1	0.97
mean	300	200	1.00E-06	0.1	150	1	2.45
max	300	200	1.00E-06	0.1	150	1	6.16
min	600	200	0.001	0.01	150	0	1.1
mean	600	200	0.001	0.01	150	0	2.52

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
max	600	200	0.001	0.01	150	0	6.71
min	600	200	0.001	0.01	150	0.5	1.36
mean	600	200	0.001	0.01	150	0.5	2.99
max	600	200	0.001	0.01	150	0.5	6.85
min	600	200	0.001	0.01	150	1	1.02
mean	600	200	0.001	0.01	150	1	2.44
max	600	200	0.001	0.01	150	1	6.3
min	600	200	0.0001	0.01	150	0	1.1
mean	600	200	0.0001	0.01	150	0	2.39
max	600	200	0.0001	0.01	150	0	5.97
min	600	200	0.0001	0.01	150	0.5	0.95
mean	600	200	0.0001	0.01	150	0.5	2.43
max	600	200	0.0001	0.01	150	0.5	6.03
min	600	200	0.0001	0.01	150	1	1.5
mean	600	200	0.0001	0.01	150	1	3.19
max	600	200	0.0001	0.01	150	1	7.32
min	600	200	1.00E-05	0.01	150	0	2.62
mean	600	200	1.00E-05	0.01	150	0	3.35
max	600	200	1.00E-05	0.01	150	0	7.04
min	600	200	1.00E-05	0.01	150	0.5	1.68
mean	600	200	1.00E-05	0.01	150	0.5	3.54
max	600	200	1.00E-05	0.01	150	0.5	8.02
min	600	200	1.00E-05	0.01	150	1	1.51
mean	600	200	1.00E-05	0.01	150	1	3.16
max	600	200	1.00E-05	0.01	150	1	7.23
min	600	200	1.00E-06	0.01	150	0	1.11
mean	600	200	1.00E-06	0.01	150	0	2.6
max	600	200	1.00E-06	0.01	150	0	6.56
min	600	200	1.00E-06	0.01	150	0.5	1.19
mean	600	200	1.00E-06	0.01	150	0.5	3.34
max	600	200	1.00E-06	0.01	150	0.5	7.93
min	600	200	1.00E-06	0.01	150	1	0.92

D.T.	O.I.	O.L.E.	O.L.R	H.L.R	L.E.	R.F.P.	EER%
mean	600	200	1.00E-06	0.01	150	1	2.35
max	600	200	1.00E-06	0.01	150	1	6.19