UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pòs-Graduação em Ciência da Computação

Pedro Henrique Silva Souza Barros

Um novo espaço de similaridade sob medida para aprendizado métrico profundo supervisionado

Belo Horizonte
2021

Pedro Henrique Silva Souza Barros

Um novo espaço de similaridade sob medida para aprendizado métrico profundo supervisionado

Versão final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Heitor Soares Ramos Filho
Coorientadora: Fabiane da Silva Queiroz

Belo Horizonte
2021

Pedro Henrique Silva Souza Barros

A New Similarity Space Tailored for Supervised Deep Metric Learning

Final version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Heitor Soares Ramos Filho
Co-Advisor: Fabiane da Silva Queiroz

Belo Horizonte
2021

Barros, Pedro Henrique Silva Souza.

B277n A new similarity space tailored for supervised deep metric learning [manuscrito] / Pedro Henrique Silva Souza Barros– 2021.
xxv, 83 f. il.

Orientador: Heitor Soares Ramos Filho
Coorientadora: Fabiane da Silva Queiroz.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.71-83
.
1. Computação – Teses. 2. Aprendizado profundo – Teses. I. Ramos Filho, Heitor Soares. II. Queiroz, Fabiane da Silva. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. I V.Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

A NEW SIMILARITY SPACE TAILORED FOR SUPERVISED DEEP METRIC LEARNING

PEDRO HENRIQUE SILVA SOUZA BARROS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

Prof. Heitor Soares Ramos Filho - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]

Profa. Fabiane da Silva Queiroz - Coorientadora
Centro de Ciências Agrárias - UFAL

[illegible]

Profa. Gisele Lobo Pappa
Departamento de Ciência da Computação - UFMG

[illegible]

Prof. Alejandro César Frery Orgambide
School of Mathematics and Statistics - Victoria University of Wellington

Belo Horizonte, 5 de Março de 2021.

Acknowledgments

Primeiramente, agradeço a Deus por todas as graças alcançadas em minha vida.

Gostaria de agradecer a minha família, especialmente meus pais, Rafael e Vera, bem como meu irmão João, por todo apoio (seja ele financeiro, emocional ou educacional) que me foi oferecido durante todo o meu período acadêmico. Tenho certeza que sem esse apoio, não teria conseguido finalizar essa dissertação. Vocês são minha inspiração e motivação da minha vida.

Agradeço também pelo maior presente que Belo Horizonte me apresentou, Joyce Elisa Herédia. Muito obrigado por toda paciência, apoio, conversas, sushis e conselhos durante todo esse ano. Espero que nosso amor consiga durar por muito mais tempo.

Além disso, agradeço ao meu mentor, amigo e orientador Dr. Heitor Ramos, por todo apoio que venho recebendo nesses últimos anos, desde minha graduação. Através de sua sugestão, comecei a analisar a hipótese de realizar mestrado na UFMG.

Meus grandes amigos do Conversa Mole, por todas brincadeiras e resenhas do dia-a-dia. Em especial, meus parceiros do return (Demetrios, Alvino e Matheus), que mesmo longe, ainda mantemos a amizade construída durante todo período de graduação. Além disso, agradeço meu parceiro Cristopher por todas conversas sobre o mestrado, bem como eventos e viagem acadêmicas.

Minhas companheiras de Laccan House, Duda e Julie, que encararam dividir o apartamento comigo durante esses dois anos, além de me apresentar os dois gatos mais legais de BH: Miguel e Nina. Obrigado por tudo meninas.

E por fim, agradeço a todos que contribuíram de alguma forma neste trabalho, bem como a banca avaliadora pela leitura e comentários acerca desta dissertação.

"Não há briga entre nós, mas vivemos brigando
Vivemos brigando"
(Baco Exu do Blues)

Resumo

No presente trabalho, propomos um novo método de aprendizagem métrica profunda que diferentemente de muitos trabalhos nesta área, define um novo espaço latente obtido por meio de um autoencoder. O novo espaço, chamado de espaço S, é dividido em diferentes regiões que descrevem as posições onde pares de objetos são similares/dissimilares. Localizamos marcadores para identificar essas regiões. Em seguida, estimamos as semelhanças entre objetos por meio de uma distribuição t-student baseada em kernel para medir a distância dos marcadores e a nova representação de dados. Assim, estimamos simultaneamente a posição dos marcadores no espaço S e representamos os objetos no mesmo espaço em nossa abordagem. Além disso, propomos uma nova função de regularização para evitar que marcadores similares entrem em colapso. Apresentamos evidências de que nossa proposta pode representar espaços complexos, por exemplo, quando grupos de objetos semelhantes estão localizados em regiões disjuntas. Comparamos nossa proposta com 9 abordagens diferentes de aprendizagem métrica a distância (quatro delas são baseadas em aprendizagem profunda) em 28 conjuntos de dados heterogêneos do mundo real. De acordo com as quatro métricas quantitativas utilizadas, nosso método supera todas as nove estratégias da literatura. Além disso, investigamos alguns estudos de caso em diferentes domínios, para verificar a eficácia de nossa proposta.

Palavras-chave: Espaço de Similaridade, Espaço Latente, Aprendizagem de Métrica Profunda.

Abstract

We propose a novel deep metric learning method. Differently from many works in this area, we defined a novel latent space obtained through an autoencoder. The new space, namely S-space, is divided into different regions that describe the positions where pairs of objects are similar/dissimilar. We locate makers to identify these regions. We estimate the similarities between objects through a kernel-based t-student distribution to measure the markers' distance and the new data representation. We simultaneously estimate the markers' position in the S-space and represent the objects in the same space in our approach. Moreover, we propose a new regularization function to avoid similar markers to collapse altogether. We present evidence that our proposal can represent complex spaces, for instance, when groups of similar objects are located in disjoint regions. We compare our proposal to 9 different distance metric learning approaches (four of them are based on deep-learning) on 28 real-world heterogeneous datasets. According to the four quantitative metrics used, our method overcomes all the nine strategies from the literature. In addition, we investigated some case studies in different domains, to verify the effectiveness of our proposal.

Palavras-chave: Similarity Space, Latent Space, Deep Metric Learning.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

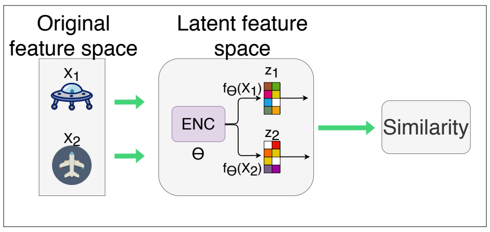
1.1 Motivation

A distance metric is a function that provides a way to measure how far apart two elements of a set are from each other. Among various works involving machine learning applications, the most commonly used metric is the Euclidean distance [21]. Methods that use Euclidean distance usually consider that all variables' covariance is zero, i.e., there is no correlation among them, but this assumption is hardly found in the real world [110]. Euclidean distance and cosine similarity are popular for many applications. For instance, the cosine similarity is vastly used for text mining [51]. Even showing its effectiveness in several applications, the cosine similarity assumes equal weight for every dimension, limiting its application [51].

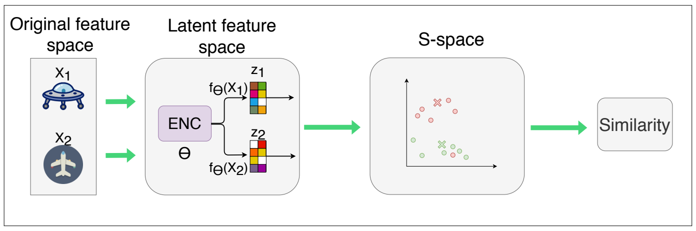
Euclidean and cosine distance are known as data-independent techniques, once they are defined without any prior knowledge about the data. Learning distances, a.k.a Metric Learning (MeL), from data is a common attempt to improve machine learning approaches [23, 31, 40, 41, 52, 103]. In modern machine learning research, MeL is a fundamental technique for several different applications such as sorting [57], classification (e.g., k-nearest neighbors), clustering [109], and ranking [95].

MeL aims to estimate distance function parameters based on a given training set. Thus, a distance d can be defined as $d_{\boldsymbol{M}}(x, y) = \sqrt{(x-y)^T \boldsymbol{M}(x-y)}$, in which $\boldsymbol{M}$ is a positive semi-definite matrix. In the case $\boldsymbol{M}$ is the covariance matrix, $d_{\boldsymbol{M}}$ is the *Mahalanobis* distance [21]. Classic methods proposed for metric learning use $d_{\boldsymbol{M}}$ to search for the best linear space that captures the semantics of the data (e.g., in a classification setting, we search for $\boldsymbol{M}$ that minimizes the miss-classification loss). However, according to Cao et al. [13], the linear transformation $\boldsymbol{M}$ (covariance matrix) has some limitations, as it cannot model high-order correlations between the original data dimensions.

Using MeL, we can define metrics that consider the covariance of attributes. Additionally, MeL approaches do not necessarily assume linear relationships, although classical MeL techniques like the Mahalanobis distance [21] assume a linear space. Moreover, MeL



(a) Canonical scheme of DMeL.



(b) Our scheme of DMeL with S-space.

Figure 1.1. Comparison between the canonical model of DMeL and the model used in this chapter.

does not assume equal weights for every attribute [51]. The assumption that MeL can be treated as a convex optimization problem can also be relaxed using the appropriate model.

To tackle the issues mentioned above, deep learning techniques are currently being used for MeL [39, 56, 67, 82, 69, 120]. Since these proposals seek to learn a non-linear feature representation, they usually overperform standard techniques found in the literature. Neural Networks (NNs) are natural candidates and are typically used to learn similarity metrics [15, 33].

1.2 Objectives

The representation of compressed data found by a Neural Network (NN) is commonly named as latent feature space and the data in this space as latent data. Our proposal hypothesizes that the latent feature space captured by NNs can be improved with an auxiliary space. For instance, common NNs-based Deep Metric Learning (DMeL) approaches extract a latent space that encodes similar and dissimilar *points*, but not the separability between them. However, this single representation is limited, as it does not capture *pairwise information.*

Unlike the literature, our approach employs NNs, fed by labeled original pairwise data, to find a *latent pairwise space with markers.* These approaches are shown in Figures 1.1a and 1.1b as we now detail. In our method, data come in pairs of vectors $(\mathbf{x}_i, \mathbf{x}_j)$ which are deemed as similar or dissimilar. The first part of our architecture is an autoencoder. After encoding the pair of input objects, our major novelty is on converting *data pairs* to a new Similarity-space (called S-space). A data point $\mathbf{x}_i$ is mapped into $\mathbf{z}_i = f_\Theta(\mathbf{x}_i)$ in the latent space with a enconfoder function f_Θ, where Θ are model parameters. The S-Space, for a pair of points i and j is composed of two novel ideas. Firstly, we represent points as a similarity vector between pairs, i.e., $\mathbf{s}_{ij} = |\mathbf{z}_i - \mathbf{z}_j|$. Secondly, *and* *more importantly*, we define markers that act as reference points to similar and dissimilar regions. Markers' position are learned in the optimization process.

Our loss function is comprised of three parts. Firstly, an autoencoder loss function takes care of data encoding and decoding. The second loss function captures the sum of distances between similarity vectors ($\mathbf{s}_{ij}$) and markers ($\boldsymbol{\mu}_m$), in this proposal, we used a Cauchy distribution (discrete version) to estimate this distance and we apply a cross-entropy loss function between the input labels and the model output. The last part of our loss function is called a *repulsive regularizer*. It is inversely proportional to the distance of the markers of the same class. This loss function ensures that markers are different (the loss increases as markers become similar), ensuring some diversity level on the marker set. It attempts that markers capture complex similarity regions such as disjoint similarity/dissimilarity regions.

We named our approach as *Supervised Distance Metric learning Encoder with Similarity Space* (SMELL). Our method is herein described as supervised learning, but it can be appropriately extended to unsupervised and semi-supervised learning. Through a wide range of experiments on 28 datasets, we show that SMELL provides gains over the state-of-the-art techniques in all of them. To explain its accuracy, we show evidence supporting the following two hypotheses.

Hypothesis 1. (**H1**) Using SMELL, the markers group data points considered similar (in our context, which have the same labels) and dissimilar (different labels) into disjoint regions in S-space.

Hypothesis 2. (**H2**) SMELL increases the input pairs' separability in the latent feature space for different types of pairs (similar/dissimilar).

1.3 Contributions

Overall, the main contributions of our dissertation are:

(i) a new data representation space called *Similarity space* (S-space) that separates regions where similar/dissimilar objects lie together and help the convergence of the model. We also investigate interpretability and data visualization in this space. S-space can capture complex regions that can model similar points in disjoint regions;

(ii) a new distance metric learning method that simultaneously learns a latent representation of the data and the markers' position in the S-space;

(iii) we found evidence that the number of markers is a virtual hyperparameter of the model and does not need to be tuned.

(iv) a new regularization function to avoid model overfitting called *repulsive regularizer*.

1.4 Outline

The remainder of this dissertation is organized as follows: Chapter 2 presents the related works to distance metric learning; Chapter 3 describes our proposal and some notations and a background review for the good understanding our proposal; Chapter 4 describes the experimental setup used to analyze the data; Chapter 5 presents the main results and discussions; Chapter 6 and Chapter 7 describes twice case study to our propose and Chapter 8 concludes this dissertation.

Chapter 2

Related Work

In the distance metric learning task, prior research usually assumes that the datasets are represented by an incomplete set of features (i.e., we can never collect all the features of an object). This subset of features may not thoroughly inform the semantics of the data space. Thus, the objective is to learn a similarity matrix that encodes how these features should be combined to compute distances best.

One of the first successful cases to solve this problem was learning the linear matrix (Mahalanobis) metric to find a new representation in the feature space [29, 44, 113]. This paradigm requires the decomposition of eigenvalues, an operation that is cubic in the dataset dimensionality (i.e., number of features). This issue severely impacts the training time. Also, approaches like this one are limited to similarity matrices, which encode linear combinations of features.

Other approaches proposed techniques based on Information Theory to tackle the distance metric learning problem [20, 58, 66]. These works start from a reference distribution to train distance functions based on divergences (e.g., Kullback-Leibler or Jeffrey) to obtain reference probability distributions of the data. Through this reference distribution, the authors estimate the similarity. These methods usually suffer from convergence issues [20] when optimizing.

In kernel-based methods, the input data is usually transformed into a higher dimensional space. The algorithm learns object similarities using the new space obtained from the kernel function [58, 64, 89, 100]. These methods also suffer from a cubic computation cost (on the number of features) or suffer from convergence issues, limiting its applicability due to training time.

In the context of Deep Learning methods, a typical family of Deep Neural Network models that learns distance metrics is the Siamese Neural Networks (SNNs). One of the first works using this approach can be seen in [11], where the authors propose a model composed of two neural networks that share their weights. This architecture was initially proposed for the signature verification problem.

Neural Networks (NNs) seek to find nonlinear similarities between comparable data examples by extracting a feature vector representing the difference between the data examples. There are several work in the context of NNs developed for different applications [14, 16, 46, 81, 108, 116]. They are easily scalable (do not suffer from the cubic cost as before), as they do not explore eigenvalues decomposition. NNs are typically optimized with functions that consider pairs of inputs, called pairwise loss function, and these proposals tend to find a new representation of the data. Therefore, a similarity function is defined in this new representation (for example, Euclidean).

More recent work present deep metric learning with contrastive loss [15, 33] and triplet loss [80]. Even showing promising results, these proposals present issues, such as slow convergence and poor local optima, optimizing the model is a challenging task. The embedding obtained by contrastive loss is highly dependent on the quality of the representation of the training.

More recent work present deep metric learning with contrastive loss [15, 33] and triplet loss [80]. Even showing promising results, these proposals present issues, such as slow convergence and poor local optima, optimizing the model is a challenging task. The embedding obtained by contrastive loss is highly dependent on the quality of the representation of the training data. The training set must contain real-valued precision for pairwise samples. This consideration is typically hard to satisfy, which is usually not available in practice [99]. For the triplet model, the loss function defines an inequality between positive and negative examples for a given anchor example. These methods suffer from what is called the hard negative problem [65, 107]. Here, some specific negative examples deteriorate the quality of the model, making the training unstable [18]. Hard negative data mining is a proposal to work around this problem. However, the computational cost of searching for these examples becomes high. In addition, it is unclear what defines "good" hard triplets [83].

Recent work, including N-pair loss [86], Lifted Structure [68], and the Multi-Similarity Loss [101] propose strategies to capture relationships within a mini-batch selection. Typically, these strategies consider a weight function that associates the pairs of elements in the loss calculation. Nevertheless, these work are based on distance measurements between pairs of similar and dissimilar objects in the space found by the neural network.

The methods mentioned in this section indicate the feasibility of learning a similarity function from the input data. Some of these methods inspire the present work [11, 54]; for instance, we use Neural Networks to extract the data representation and the t-student kernel distribution to create a similarity metric.

However, we devised a novel deep metric learning method differently from the literature using a new representation space (S-space) obtained through autoencoders. As defined herein, the S-space helps the convergence of the proposal and, thanks to the possibility of having multiple markers to represent similar objects, it models even complex spaces such as noncontinuous spaces where similar objects lie in disjoints regions. Therefore, we propose a new similarity space that helps the learning of autoencoders.

Unlike pairwise loss, our proposal does not require any specific sample selection strategy.

Chapter 3

Methodology

3.1 Background and Notation

In SMELL, we map pairwise input data into a latent space and a Similarity space. In this section, we provide some technical background about data representation with autoencoders and a mathematical notation essential to the proposed method understanding.

Throughout the dissertation, we apply the following notation. We denote vectors by boldface lowercase letters, such as $\boldsymbol{x}$, $\boldsymbol{z}$ and $\boldsymbol{\mu}$; all scalars by lowercase letters, such as m and n; sets of parameters by greek uppercase letters, such as Θ and Σ; and sets by calligraphic uppercase letters, such as $\mathcal{X}$ and $\mathcal{Z}$. The zero-mean normal distribution will be denoted by $\mathcal{N}(\mu = 0, \sigma)$. Table 3.1 summarizes this notation.

Let the set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{v}$, with $\boldsymbol{x}_i \in \mathbb{R}^m$, be v data examples defined in an m-

Notation	Description
$\mathcal{X}$	input data examples set
$\boldsymbol{x}_i$	m-dimensional single element in $\mathcal{X}$
$\mathcal{Y}$	Label set for set $\mathcal{X}$
y_i	single element in $\mathcal{Y}$
$\mathcal{Z}$	Latent Feature Space from $\mathcal{X}$
$\boldsymbol{z}_i$	n-dimensional single element in $\mathcal{Z}$
f_Θ	Encoder function
Θ	set of weights for encoder
$f_{\Theta'}$	Decoder function
Θ'	set of weights for decoder
ℓ	label function for a element in set $\mathcal{X}$
ℓ'	label function for a element in set $\mathcal{Z}$
$\mathcal{S}$	The *similarity space* from $\mathcal{Z}$
$\boldsymbol{s}_{ij}$	n-dimensional single element in $\mathcal{S}$
$\mathcal{M}$	The *markers set*, subset of S
μ_i	n-dimensional single element in $\mathcal{M}$
f^S	Function that maps a pair in $\mathcal{X}$ to an element in $\mathcal{S}$
ψ	The similarity function
Σ	Set of parameters of ψ (Θ, Θ' and $\mathcal{M}$)

Table 3.1. Notation used in this article.

dimensional feature space. For each $\boldsymbol{x}_i \in \mathcal{X}$ there is an associated label $y_i \in \mathcal{Y} = \{y_i\}_{i=1}^{v}$, where $y_i \in \{1, ..., b\}$. In this way, the pair $(\boldsymbol{x}_i, y_i)$ indicates to which of b classes an input $\boldsymbol{x}_i$ belongs to. In a supervised Machine Learning classification problem, we seek to find a function $\ell : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an unlabeled example $\boldsymbol{x}_i$ into their respective label y_i.

To develop the proposed work, we introduce here some important definitions:

Definition 3.1.1. (*The latent feature space*) Consider the set $\mathcal{X}$ as the original feature space and the representation function $f_\Theta : \mathcal{X} \rightarrow \mathcal{Z}$, in which $f_\Theta(\boldsymbol{x}_i) = \boldsymbol{z}_i \implies \ell(\boldsymbol{x}_i) = \ell'(\boldsymbol{z}_i) = y_i$ and the function $\ell' : \mathcal{Z} \rightarrow \mathcal{Y}$, which maps the latent data into their respective labels. We can define the representation space $\mathcal{Z}$ called *latent feature space* from $\mathcal{X}$ as $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^{v}$, with $\boldsymbol{z}_i \in \mathbb{R}^n$.

An autoencoder is a Neural Network trained to attempt to copy a data input to its output. It can be seen as consisting of two parts: an encoder and a decoder that produces an input-based reconstruction [32]. An encoder is a representation learning algorithm that seeks to find a representation function $f_\Theta : \mathcal{X} \rightarrow \mathcal{Z}$ for a set of weights Θ that maps the set $\mathcal{X}$ to the *latent feature space* $\mathcal{Z}$.

Similarly, the decoder function can be defined as the inverse encoder function $f_{\Theta'}^{-1} : \mathcal{Z} \rightarrow \mathcal{X}$ where Θ' is a set of weights for the decoder. Autoencoders are trained to minimize reconstruction errors (typically, Mean Squared Errors - MSE), and its training is performed through *Backpropagation* of the error, just like a regular Feedforward Neural Network [36].

A neural network model [11, 102] receives a pair of input examples $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{X} \times \mathcal{X}$ and transforms each of them to a latent data $(\boldsymbol{z}_i, \boldsymbol{z}_j) \in \mathcal{Z} \times \mathcal{Z}$ through the encoder f_Θ.

In the context of supervised learning, for a data pairwise $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{X} \times \mathcal{X}$, we say they are similar iff $\ell(\boldsymbol{x}_i) = \ell(\boldsymbol{x}_j)$. Analogously, they are dissimilar iff $\ell(\boldsymbol{x}_i) \neq \ell(\boldsymbol{x}_j)$.

Definition 3.1.2. (*The similarity space*) The representation space called *Similarity space* (or *S-space*) is a space built from the set $\mathcal{X} \times \mathcal{X}$. So, be the function $f^S : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{S}$, the *similarity space* is defined as $\mathcal{S} = \{\boldsymbol{s}_{ij}\}$, with $\boldsymbol{s}_{ij} \in S \subset \mathbb{R}^n$, where if $\ell(\boldsymbol{x}_i) = \ell(\boldsymbol{x}_j)$, then $\boldsymbol{s}_{ij}$ represents the similarity vector and if $\ell(\boldsymbol{x}_i) \neq \ell(\boldsymbol{x}_j)$, then $\boldsymbol{s}_{ij}$ represents the dissimilarity vector.

In this paper, we define the map function $f^S : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{S}$ for a pairwise $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ by the following element-wise absolute value operation:

$$\begin{aligned} \boldsymbol{s}_{ij} &= f^S(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= |f_\Theta(\boldsymbol{x}_i) - f_\Theta(\boldsymbol{x}_j)| \\ &= |\boldsymbol{z}_i - \boldsymbol{z}_j| \\ &= (|z_i^1 - z_j^1|, |z_i^2 - z_j^2|, \ldots, |z_i^n - z_j^n|) \end{aligned} \tag{3.1}$$

it is worth noting that since $\boldsymbol{s}_{ij}$ is obtained by an element-wise process, it has the same dimension as $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, where z_i^n is the n-th feature of the i-th data example in a latent space representation $\mathcal{Z}$ (see Definition 3.1.1).

Definition 3.1.3. (*The Markers set*) In *S-space*, we defined the markers set $\mathcal{M} \subset \mathbb{R}^n$ (same space as $\mathcal{S}$) to improve similarity calculations. We define the set $\mathcal{M}^+$ representing the set of markers responsible for quantifying the similarity between the input pairs. Likewise, markers in set $\mathcal{M}^-$ quantify the dissimilarity. The Markers set is defined as

$$\mathcal{M} = \mathcal{M}^+ \cup \mathcal{M}^- = \{\boldsymbol{\mu}_i^+\}_{i=1}^k \cup \{\boldsymbol{\mu}_j^-\}_{j=k+1}^w. \tag{3.2}$$

Therefore, in this dissertation, we seek to calculate the similarity function $\psi_\Sigma : \mathcal{X} \times \mathcal{X} \to [0, 1]$. The parameters of ψ are defined by the set $\Sigma = \{\Theta, \Theta', \mathcal{M}\}$, respectively the weights of encoder, decoder and the Markers set in S-space. SMELL relies in simultaneously learning all elements of Σ. More details about the proposed method are described in Section 3.2.

3.2 Supervised Distance Metric learning Encoder with Similarity Space (SMELL)

Our proposal, namely SMELL, simultaneously optimizes a latent data representation (using a DMeL model) and a similarity function that indicates the similarity of two objects in the learned data S-space. This kind of technique can be useful for a wide variety of applications, such as to feed a predictor (e.g., a classifier) with a new metric learned from the data. This section details our proposal. Figure 3.1 shows a simple schematic for our proposal.

3.2.1 Metric learning algorithm

There are several ways to find a similarity metric ψ_Σ [98, 4]. In this chapter, we propose ψ_Σ being estimated from the latent representation obtained by the encoder $f_\Theta : \mathcal{X} \to \mathcal{Z}$.

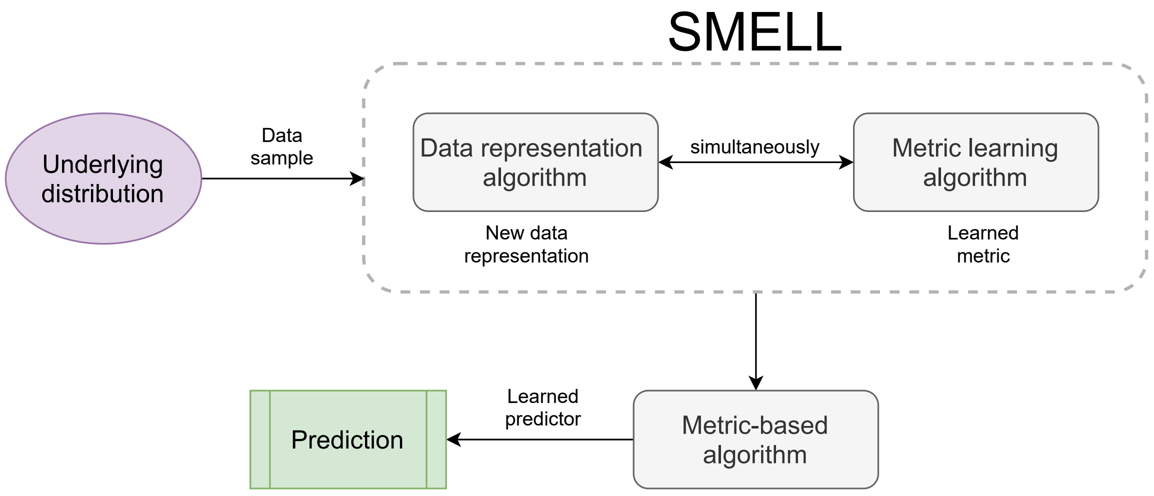


Figure 3.1. Simple black box schematic for our proposal.

3.2.2 The S-space

As can be seen in in Definition 3.1.2, we define a new representation space namely S-space $\mathcal{S}$, which quantifies the similarity between pairs of objects. In Equation 3.1, we propose a map function $f^S : \mathcal{X} \times \mathcal{X} \to \mathcal{S}$ for a data pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as an element-wise absolute value operation representing the pairwise difference between the pair of data. Note, in Equation 3.1, that $\boldsymbol{s}_{ij} \in \mathbb{R}^n$ (it has the same dimension then *latent representation space*).

Regarding the pairwise labeling, we have two options for a given pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$: similar or dissimilar. Thus, we define the *Markers set* $\mathcal{M}$ so that each marker of $\mathcal{M}^+$ or $\mathcal{M}^-$ represents one of these possibilities (see Definition 3.1.3). The closer the vector $\boldsymbol{s}_{ij}$ is to a marker $\boldsymbol{\mu}^+ \in \mathcal{M}^+$ or $\boldsymbol{\mu}^- \in \mathcal{M}^-$, the greater the probability that the elements of the pair $(\boldsymbol{x}_i,\ \boldsymbol{x}_j)$ are similar or dissimilar to each other, respectively. Then, we have, in this case, k similarity markers and $w - k$ dissimilarity markers for $\mathcal{M}$, and $\mathcal{M}^+ \cap \mathcal{M}^- = \emptyset$.

Inspired by [49, 54, 112] we use a Cauchy distribution (discrete version) as base for a kernel to measure the similarity between $\boldsymbol{s}_{ij}$ and a specific marker $\boldsymbol{\mu}_m \in \mathcal{M}$, as

$$q_{ij}^m = \frac{\left(1 + ||\boldsymbol{s}_{ij} - \boldsymbol{\mu}_m||_2^2\right)^{-1}}{\sum_{\mu_{m'} \in \mathcal{M}} \left(1 + ||\boldsymbol{s}_{ij} - \boldsymbol{\mu}_{m'}||_2^2\right)^{-1}}, \tag{3.3}$$

where $q_{ij}^m \in \mathbb{R}$ is the similarity/dissimilarity of $\boldsymbol{s}_{ij}$ in relation to the markers μ_m (it is normalized by the sum of all markers in $\mathcal{M}$). So, we calculate $q_{ij}^+ = \sum_p q_{ij}^p$ for all $\boldsymbol{\mu}_p \in \mathcal{M}^+$ and $q_{ij}^- = \sum_n q_{ij}^n$ for all $\boldsymbol{\mu}_n \in \mathcal{M}^-$. In other words, q_{ij}^+ is the probability of $\boldsymbol{x}_i$ have the same label as $\boldsymbol{x}_j$ and q_{ij}^- is the probability of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ have different labels. Since $\mathcal{M}^+$ and $\mathcal{M}^-$ are two disjoint sets, we have $q_{ij}^+ + q_{ij}^- = 1$.

The Cauchy distribution is a probability distribution that belongs to the student-t distributions' family (with 1 degree of freedom). Among the student-t distributions, the

Cauchy distribution has the biggest uncertainty in the tails, i.e., it presents a the most heavier tail. Also, this distribution does not have finite moments of order greater than or equal to one; only fractional absolute moments exist [42]. It is a few stable distributions (a linear combination of two independent random variables with this distribution has the same distribution, up to location and scale parameters). It has a probability density function that can express analytically (e.g., normal distribution and the Lévy distribution).

It is worth noting that we use a different version of the Deep Metric Learning canonical model. Thus, we use the representation of the difference vector $\boldsymbol{s}_{ij}$ defined in S-pace. In Section 5.4 we show more details about this choice.

3.2.3 Loss function and regularization

SMELL relies on simultaneously learning a latent representation of the data (with parameters Θ and Θ' for the encoder and decoder functions, respectively) and the positioning of the markers of the set $\mathcal{M}$ in S-space. Therefore, we seek to find the parameters $\Sigma = \{\Theta, \Theta', \mathcal{M}\}$ of the function $\psi_\Sigma(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is defined as an optimization problem. Let the cost function be $J(\{\mathcal{X} \times \mathcal{X}\})$, we estimate the optimal parameters set Σ^* with Cross-entropy loss H_c. We define regularization functions R_r and R_d to avoid overfitting in the training process. In training, the cross-entropy is applied between the output of SMELL and object's classes.

Similarly to [24], R_r regards to the autoencoder's reconstruction error. In our proposal, for all training pairs $(\boldsymbol{x}_i,\ \boldsymbol{x}_j)$ and for all reconstructed pairs $(\boldsymbol{x}'_i, \boldsymbol{x}'_j)$ we have $R_r = r_r N^{-1} \sum_i \sum_j \left(||\boldsymbol{x}_i - \boldsymbol{x}'_i||_2^2 + ||\boldsymbol{x}_j - \boldsymbol{x}'_j||_2^2 \right)$, where r_r is a constant to calibrate the loss reconstruction function and N is the number of pairs in train the dataset.

When we use more than one maker as reference points to the similarity/ dissimilarity regions, markers of the same set $\mathcal{M}^+$ (or $\mathcal{M}^-$) tend to group altogether, hidering the efficiency of our method. In this context, we propose a new regularization term R_d we called *Repulsive Regularizer*, to avoid this undesirable behavior. It is defined as

$$R_d^+ = \frac{1}{c^+} \left[\sum_{\mu_i \in \boldsymbol{M}^+} \sum_{\mu_j \in \boldsymbol{M}^+} \frac{1}{||\mu_i - \mu_j||_2^2 + \epsilon} \right], \tag{3.4}$$

where $\boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j$ and c^+ is a constant value defined as $c^+ = \binom{k}{2}$, in which k is the number of elements in $\mathcal{M}^+$ (see Definition 3.1.3). R_d is inversely proportional to the square distance of the markers. To avoid a possible division by 0, we added the corrective term ϵ to the

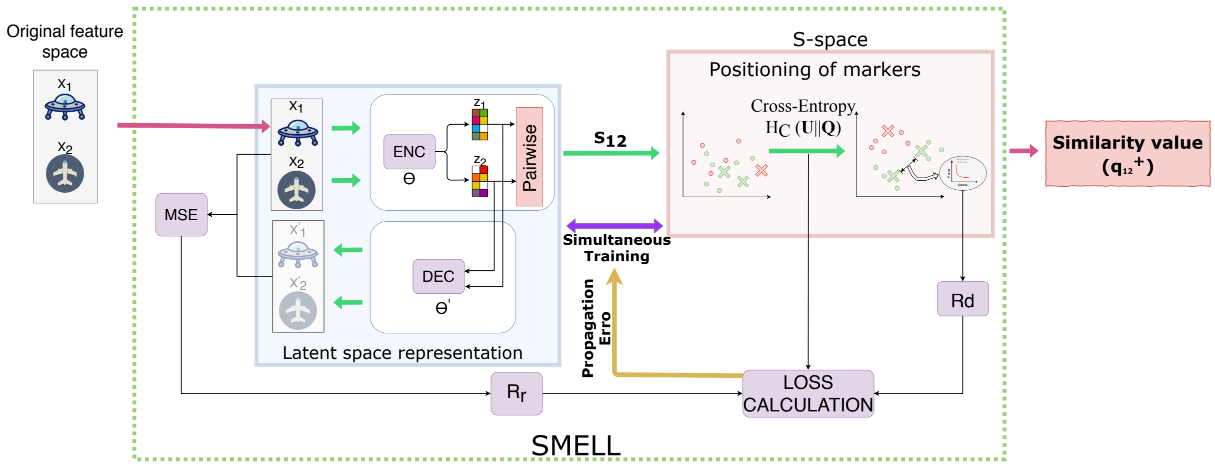


Figure 3.2. The left side represents the Encoder with reconstruction, and the right side represents the optimization process for the markers' position for $\mathcal{M} = \{\boldsymbol{\mu}_1^+, \boldsymbol{\mu}_2^+, \boldsymbol{\mu}_3^-\}$. In this example, we used two positives and one negative marker. Green and red crosses represent $\boldsymbol{\mu}_1^+$, $\boldsymbol{\mu}_2^+$, and $\boldsymbol{\mu}_3^-$, respectively, green and red dots represent the similar and dissimilar input pairs. The rightmost green arrow shows a representation of the markers' position optimization step by using Cross-Entropy divergence and some regularization functions. Observe that the number of positive and negative markers are hyperparameters.

denominator. We conducted a manual investigation with a grid search, and we adopted for our experiments $\epsilon = 10^{-3}$. In the same way, we define R_d^-, and with that, we have

$$R_d = r_d(R_d^+ + R_d^-), \tag{3.5}$$

with a constant value r_d for calibration. Note that $R_d^+ = 0$ if we have a single positive marker $k = 1$. In the same way, if we have a single negative marker, $R_d^- = 0$ if $w - k = 1$.

Let $\mathcal{Q} = \{q_{ij}\}$, the SMELL output, be the set that contains the pairs $q_{ij} = (q_{ij}^+, q_{ij}^-)$ corresponding to the probability of the elements of a pairwise input $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be similar or dissimilar, respectively. The optimal hyperparameters set can be defined as $\Sigma^* = \arg\min_{\boldsymbol{\Sigma}} J(\{\mathcal{X} \times \mathcal{X}\})$, where

$$J(\{\mathcal{X} \times \mathcal{X}\}) = H_c(\mathcal{U}||\mathcal{Q})r_{HC} + R_r + R_d, \tag{3.6}$$

where r_{HC} is a constant for calibration and $\boldsymbol{u}_{ij} \in \mathcal{U}$ is defined as $\boldsymbol{u}_{ij} = (1, 0)$ if i has same label as j and $\boldsymbol{u}_{ij} = (0, 1)$, otherwise.

SMELL learns all parameters in the set Σ^* simultaneously. The representation found in S-space aims at grouping the elements $\boldsymbol{s}_{ij}$ around their respective markers, as defined in Loss Function J (Equation 3.6). The impact of the attractive behavior is controlled by the constant r_{HC}, i.e., the higher the r_{HC}, the greater is the tendency to group the points $\boldsymbol{s}_{ij}$ closer to the respective markers. Also, note that the regularization functions operate in different spaces, i.e. R_r operate in *latent feature space*, R_d operates in *S-space* and H_c operates in *latent feature space* and *S-space* simultaneously.

Figure 3.2 depicts the more detailed schematic of our proposal using a toy example (two positive markers and one negative). Observe that the number of positive and negative markers is a hyperparameter.

3.2.4 Optimization

To find the Σ^* set, we use mini-batch stochastic gradient decent (SGD) and back-propagation. First, we note that the decoder weights Θ' are only affected by the R_r component of the loss function J. So, we can use $\partial R_r/\partial\Theta'$ to update Θ'. Then, given a mini-batch with g samples and learning rate λ, Θ' is updated by

$$\Theta' = \Theta' - \frac{\lambda}{g}\sum_{i=1}^{g}\frac{\partial R_r}{\partial\Theta'}. \tag{3.7}$$

To optimize the markers, consider that

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_t - \frac{\lambda}{g}\sum_{i=1}^{g}\frac{\partial J}{\partial\boldsymbol{\mu}_t} = \boldsymbol{\mu}_t - \frac{\lambda}{g}\sum_{i=1}^{g}\left(\frac{\partial L_{HC}}{\partial\boldsymbol{\mu}_t} + \frac{\partial R_d}{\partial\boldsymbol{\mu}_t}\right), \tag{3.8}$$

where $\frac{\partial L_{HC}}{\partial\boldsymbol{\mu}_t}$ can be calculated for a given $\boldsymbol{\mu}_t$ and $\boldsymbol{s}_{ij}$ as

$$\frac{\partial L_{HC}}{\partial\boldsymbol{\mu}_t} = 2\frac{(q_{ij}^t - \boldsymbol{u}_{ij})(\boldsymbol{s}_{ij} - \boldsymbol{\mu}_t)}{1 + ||\boldsymbol{s}_{ij} - \boldsymbol{\mu}_t||_2^2},$$

and

$$\frac{\partial R_d}{\partial\boldsymbol{\mu}_t} = -2\sum_{\boldsymbol{\mu}_s\in M}\left[\text{sign}(\boldsymbol{\mu}_s)\frac{||\boldsymbol{\mu}_t - \boldsymbol{\mu}_s||_2}{(||\boldsymbol{\mu}_t - \boldsymbol{\mu}_s||_2^2 + \epsilon)^2}\right],$$

where $\text{sign}(\boldsymbol{\mu}_s) = 1$ if $\boldsymbol{\mu}_s \neq \boldsymbol{\mu}_t$ and $\boldsymbol{\mu}_s$ has same semantic (similarity or dissimilarity) than $\boldsymbol{\mu}_t$, and $\text{sign}(\boldsymbol{\mu}_s) = 0$, otherwise.

For training SMELL, we randomly selected the mini-batch with m pairs of elements (half are similar, and the other half are dissimilar). Also, our proposal does not have any specific batch selection criteria.

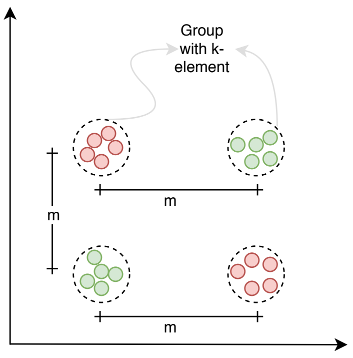


Figure 3.3. Toy example for *optimal latent space*. Same color indicates same label. S-space requires that each group has only elements of same class, but note that, this space can have different groupings with elements of the same class.

3.2.5 Theoretical proprieties

Due to the construction of the S-space, we are able to obtain some theoretical proprieties.

Definition 3.2.1. (*Optimal Latent Space*) Let $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ and a latent representation function $f_\Theta : \mathcal{X} \to \mathcal{Z}$. The transformation f_Θ generates an *optimal latent space* $\mathcal{Z}$ when the expected value $\mathbb{E}[||\boldsymbol{s}_{ij}||_2] = 0 \implies \ell(\boldsymbol{x}_i) = \ell(\boldsymbol{x}_j)$.

SMELL is able to group points of same class into clusters. It is worth noting that we defined the optimal space as a conditional instead of a biconditional statement. From this definition, we can observe that SMELL may create several different clusters of the same class, as depicted in Figure 3.3.

Proposition 3.2.1. In S-space, given k positive markers in the set $\mathcal{M}^+$ and $n-k$ negative markers in $\mathcal{M}^-$, the latent space found by SMELL, i.e., the estimation of the parameters Θ of f_Θ, generates an *optimal latent space* if $\exists\ \boldsymbol{\mu}_i \in \mathcal{M}^+$ so that $||\mu_i||_2^2 < ||\mu_j||_2^2$ for any $\boldsymbol{\mu}_j \in \mathcal{M}^-$.

Proof. Given $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, SMELL measures the similarity between the entries through the t-student kernel given by q_{ij}^+, so that for $\boldsymbol{\Sigma}^*$ it follows that $q_{ij}^+ = 1 \iff \ell(x_i) = \ell(x_j)$.

Since f_Θ generates an optimal space, we then have $\mathbb{E}[||s_{ij}||_2] = 0 \implies \ell(x_i) = \ell(x_j)$, so, it follows that for a optimal latent space, we must have

$$\begin{aligned} q_{ij}^+ &= \frac{\sum_{k\in\mathcal{M}^+}(1+||\mu_k||_2^2)^{-1}}{\sum_{k\in\mathcal{M}^+}(1+||\mu_k||_2^2)^{-1} + \sum_{s\in\mathcal{M}^-}(1+||\mu_s||_2^2)^{-1}} \\ &= \frac{1}{1+\frac{\sum_{s\in\mathcal{M}^-}(1+||\mu_s||_2^2)^{-1}}{\sum_{k\in\mathcal{M}^+}(1+||\mu_k||_2^2)^{-1}}}. \end{aligned}$$

Hence, if we want $\ell(x_i) = \ell(x_j)$, we should ideally have q_{ij}^+ tending to 1. It follows that $\sum_{s\in\mathcal{M}^-}(1+||\mu_s||_2^2)^{-1} < \sum_{k\in\mathcal{M}^+}(1+||\mu_k||_2^2)^{-1}$. Therefore, let μ^+ be the element with the smallest module in the set $\mathcal{M}^+$; we then have $\sum_{k\in\mathcal{M}^+}(1+||\mu_k||_2^2)^{-1} < k(1+||\mu^+||_2^2)^{-1}$. Analogously, we can consider μ^- as the vector with the largest module in the set $\mathcal{M}^-$, so, $\sum_{k\in\mathcal{M}^-}(1+||\mu_k||_2^2)^{-1} > (n-k)(1+||\mu^-||_2^2)^{-1}$.

We can then conclude that $k(1+||\mu^+||_2^2)^{-1} > (n-k)(1+||\mu^-||_2^2)^{-1}$, and therefore, $(n-k)(1+||\mu^+||_2^2) < k(1+||\mu^-||_2^2)$. Furthermore, adding the restriction that SMELL has a similar count of positive and negative markers (section 4.3), we have $1+||\mu^+||_2^2 < 1+||\mu^-||_2^2$.

□

From Proposition 3.2.1, if SMELL finds an *optimal latent space*, at least one positive marker has a smaller norm than the negative marker. In addition, in practice, as we can see in the Section 5, at least one positive marker is smaller than all negatives markers (the positive marker is located near the origin). We observed that the model builds a latent space of groups with elements of the same class, similar to the Figure 3.3.

Proposition 3.2.2. For S-spaces built with one marker in each group, $\mu^+ \in M^+$ and $\mu^- \in M^-$, D^- and D^+ being the Euclidean distance of an object to the negative and positive marker, respectively, the misclassification risk function of a positive marker is

$$R^+ = \frac{(D^-)^2+1}{\sqrt{(D^-)^2+2}}\left[\left(\sqrt{(D^-)^2+2}\right) log\left(\frac{1}{\sqrt{\frac{(D^+)^2}{(D^-)^2+2}+1}}\right) - \right.$$
$$- \frac{((D^-)^2+1)\left[tan^{-1}\left(\frac{(D^+)}{\sqrt{(D^-)^2+2}}\right)\right]^2}{\sqrt{(D^-)^2+2}} +$$
$$\left. + (D^+)tan^{-1}\left(\frac{(D^+)^2}{\sqrt{(D^-)^2+2}}\right)\right].$$

.

Proof. Firstly, we consider the risk of the similarity in a random negative pair to be more than the similarity in a random positive pair [91] as

$$R = \int_{-\infty}^{\infty} p^-(x)\left[\int_{-\infty}^{y} p^+(y)dy\right]dx.$$

Consider $\mathcal{M}^+ = \{\mu^+\}$ and $\mathcal{M}^- = \{\mu^-\}$, i.e., $\mathcal{M}^+$ and $\mathcal{M}^-$ have cardinality 1, and $D^+(x_i,x_j)$ and $D^-(x_i,x_j)$ are the euclidean distances from the point s_{ij} to the positive and negative markers, respectively. The risk of misclassification is

$$R^+ = \int_0^{D^+(x_i,x_j)} \left[p^-(D^+(x_i,x_j)) \left(\int_0^{y=D^+(x_i,x_j)} p^+(y)d(y) \right) d(D^+(x_i,x_j)) \right].$$

Therefore, due to the construction of S-space, we consider that the likelihood of similarity/dissimilarity between the s_{ij} representation of two samples is calculated as the relative distance to a marker. Due to this construction, we have

$$R^+ = \int_0^{D^+(x_i,x_j)} q^- \left[\int_0^{y=D^+(x_i,x_j)} q^+ d(y) \right] d(D^+(x_i,x_j)).$$

Calculating each term separately, we have that for the markers in the sets $\mathcal{M}^+$ and $\mathcal{M}^-$, the probability of a having a similar objects is

$$\begin{aligned} q^+ &= \frac{(1+D^+(x_i,x_j)^2)^{-1}}{(1+D^+(x_i,x_j)^2)^{-1} + (1+D^-(x_i,x_j)^2)^{-1}} \\ &= \frac{1+D^-(x_i,x_j)^2}{2+D^+(x_i,x_j)^2+D^-(x_i,x_j)^2}. \end{aligned}$$

Analogously, we can find that $q^- = 1 - q^+$. To simplify the notation, consider that D^+ equals $D^+(x_i,x_j)$ and D^- equals $D^-(x_i,x_j)$. We have for R^+

$$\begin{aligned} R^+ &= \int_0^{D^+} q^- \left[\int_0^{y=D^+} q^+ d(y) \right] d(D^+) \\ &= \int_0^{D^+} (1-q^+) \left[\int_0^{y=D^+} q^+ d(y) \right] d(D^+). \end{aligned}$$

Therefore, the risk of misclassification for the positive marker can be reduced to $R^+ = \int_0^{D^+} (1-q^+)\Phi(D^+)dD^+$, where Φ is the cumulative density function (CDF) for q^+.

Calculating the accumulated histogram

$$\Phi(x) = \int_0^x \frac{1+(D^-)^2}{2+(x)^2+(D^-)^2} dx,$$

we get

$$\Phi(x) = \frac{((D^-)^2+1)tan^{-1}\left(\frac{x}{\sqrt{(D^-)^2+2}}\right)}{\sqrt{(D^-)^2+2}}.$$

Therefore, by solving the integral $\int_0^{D^+}(1-q^+)\Phi(D^+)dD^+$, we have the exact analytical value of the misclassification risk function for the positive marker. With that, it follows that

$$R^+ = \frac{(D^-)^2+1}{\sqrt{(D^-)^2+2}}\Bigg[\left(\sqrt{(D^-)^2+2}\right) log\left(\frac{1}{\sqrt{\frac{(D^+)^2}{(D^-)^2+2}+1}}\right) - \\ - \frac{((D^-)^2+1)\left[tan^{-1}\left(\frac{(D^+)}{\sqrt{(D^-)^2+2}}\right)\right]^2}{\sqrt{(D^-)^2+2}} + \\ + (D^+)tan^{-1}\left(\frac{(D^+)^2}{\sqrt{(D^-)^2+2}}\right)\Bigg].$$

□

Due to the S-space formulation, we obtain the probability of a pair being similar analytically, given the distance of that pair to the positive marker (typically this probability is estimated, as we can see in [91]).

Chapter 4

Experimental setup

We conducted an extensive set of experiments in several scenarios with different setups to understand SMELL behavior and effectiveness better. Section 4.1 describes the datasets we have employed. Section 4.2 details the classification protocol designed to evaluate our method and the baselines. Section 4.3 discusses the initialization and the architecture of the proposed approach.

4.1 Dataset

4.1.1 General purpose datasets

KEEL [90] is an open source[1] Java software tool that can be used for a large number of different knowledge data discovery tasks. We used 28 datasets provided by KEEL to evaluate our proposal. All datasets are numeric and have no elements missing. Furthermore, all datasets have been min-max normalized to the interval $[0, 1]$, a precondition to the experiments' execution.

There is a wide variety of data in KEEL. The 28 datasets used in our experiments are divided into Medical data (Bupa, Cleveland, Appendicitis, Newthyroid, Pima, Wdbc, Wisconsin, and Thyroid); natural Language Processing data (Vowel, Letter, and Phoneme); experimental psychological data (Balance); feature-based image data (Magic, and Satimage); hierarchical decision-making data (Monk-2, Ring, and Twonorm); nature data (Iris and Banana); disaster prediction data (Titanic); Weather data (Ionosphere); chemical data (Glass, Wine, Winequality-red); and Object/shape recognition (Sonar, Movement_libras, and Vehicle).

These 28 datasets have substantial diversity in terms of the number of examples,

[1]http://keel.es/datasets.php

Dataset	#Examples	#Features	#Classes
Appendicitis	106	7	2
Balance	625	4	3
Banana (10%)	530	2	2
Bupa	345	6	2
Cleveland	297	13	5
Glass	214	9	7
Ionosphere	351	33	2
Iris	150	4	3
Letter (10%)	2003	16	26
Magic (10%)	1902	10	2
Monk-2	432	6	2
Movement-libras	360	90	15
Newthyroid	215	5	3
Phoneme (10%)	541	5	2
Pima	768	8	2
Ring (10%)	740	20	2
Satimage (10%)	643	36	7
Segment (10%)	231	19	7
Sonar	208	60	2
Thyroid (10%)	720	21	3
Titanic (10%)	221	3	2
Twonorm (10%)	683	20	2
Vehicle	846	18	4
Vowel	990	13	11
Wdbc	569	30	2
Wine	176	13	3
Winequal-red (10%)	160	11	11
Wisconsin	683	9	2

Table 4.1. Datasets used in our experimets.

the number of features, and the number of classes. Specifically, the number of examples ranges from 106 to 2003, and the number of features ranges from 2 to 90. The datasets contain both binary and multiple class datasets with a maximum of 26 classes for one dataset.

Although our method scales up to large datasets, some methods do not; hence, due to a large number of datasets, we downsampled some of them (the ones with more than 1000 samples) to 10% of the original size. The characteristics of datasets are described in Table 4.1.

4.1.2 The MNIST dataset

The MNIST dataset[2] is one of the most common datasets used for image classification and accessible from many different sources. The data set consists of grayscale images with 28x28 dimensions. Following [33], the training set is built from all hand-written digits 4 and 9 from the MNIST dataset.

Due to a large number of MNIST features, the spatial correlation found in the images, and a large number of samples, we consider the dataset suitable for this evaluation. All images were normalized to the interval $[0, 1]$, resulting in 6958 and 6824 images corresponding to hand-written digits 4 and 9, respectively.

4.2 Network evaluation

To evaluate all metric learning techniques assessed in this chapter, including our approach, we apply a *K-Nearest Neighbor* (KNN) classifier, with three neighbors, in agreement with [22]. The KNN classification performance can often be significantly improved through (supervised) metric learning. In this dissertation, the KNN classification can be exchanged for any other algorithm that uses a metric.

Since we used several datasets to validate our proposal, we divided our assessment into two approaches. The first is an individual evaluation for each dataset, and the second is a general evaluation for all datasets.

For each dataset, we calculate the accuracy. For all datasets (except the MNIST), we calculate the average accuracy (Accuracy_AVG), the average rank position value (Ranking_AVG), and the difference of the accuracy average for the best proposal (Diff_AVG). We also calculated the number of times our algorithm was in the first position (# of 1st). For the MNIST dataset, we evaluated the proposals' accuracy for different *latent space representation* dimensions.

We used 10-fold cross-validation. This validation can largely retain heterogeneous distributions in the training set and improve statistical confidence in the results. For the sake of reproducibility, our proposal is publicly available on a Gitlab repository[3].

[2] http://yann.lecun.com/exdb/mnist/
[3] https://gitlab.com/sufex00/smell

4.3 Parameters initialization and network architecture

We initialize all weights of the autoencoder layers from a zero-mean normal distribution $\mathcal{N}(\mu = 0, \sigma = 0.01)$. Biases were also initialized as outcomes of a normal distribution $\mathcal{N}(\mu = 0.5, \sigma = 0.01)$, following [46]. Markers position are initialized with Lloyd's algorithm [53]. Furthermore, we pre-trained an autoencoder (without markers) and further transfer the learn to the complete model (with markers) to improve the convergence speed.

The encoder of all deep metric learning approaches used as baseline is identical to the one we used in SMELL. According to [112], we set network dimensions to m-512-512-2048-n for all datasets, where m is the number of features of the input data, and n is the *latent space representation* dimension. All layers are fully connected, and we used as activation function the Rectified Linear Unit (ReLU) [60].

In addition, we used mini-batch Stochastic Gradient Descent (SGD) where learning rate is 0.01 and momentum is 0.9. All parameters previously mentioned (except for the calibration of the markers) were used in all deep metric learning baseline and our proposal.

Since the optimization model depends on some hyperparameters (r_{HC}, r_d, r_r, w, k), we performed an investigation to determine which value of these variables would maximize the model accuracy. Therefore, we randomly chose the Vehicle dataset to train the model and select the hyperparameters.

In [59], the author proposed a method called Bayesian Optimization, which consists of optimizing functions such as a "black box" to determine the shape of the estimate function with a Gaussian Process regression. This function to quantify (through a heuristic) the possible values of the hyperparameters to optimize the model's performance.

Because some hyperparameters are defined in a discrete interval, such as the number of markers, it was necessary to perform the discretization of the Bayesian Optimization values. We used five random starting points, and then 20 rounds of the algorithm, where we found $r_{HC} = 1$, $r_d = 10^{-1}$, $r_r = 10^{-3}$, similarity markers $k = 3$, dissimilarity markers $w - k = 2$, these values were used in the rest of this dissertation. We realized that our proposal typically performs well when k has a value similar to $w - k$.

We configured all baselines with the hyperparameters recommended in their original articles. These parameters are listed below. Observe that three approaches (Euclidean, NCA and NPair) do not have hyperparameters to set.

- Metric Learning algorithm

  - ANMM [96]: The size of the homogeneous and heterogeneous neighborhoods for each data point is set to 10;
  - KDMLMJ [66]: Let k_1, k_2 denote the number of neighbors for constructing the positive and negative difference spaces, we used $k_1 = k_2 = 5$.

- Deep Metric Learning algorithm
  - ContrastiveLoss [33]: The margin term equals 1;
  - Triplet [80]: The margin term equals 0.2;
  - MultiSimilarityLoss [101]: The hyper-parameters for the model are : $\alpha = 2$, $\lambda = 1$, $\beta = 50$;
  - FastAPLoss [12]: The number of soft histogram bins for calculating average precision is 10.

Chapter 5

Results and Discussion

In this section, we present the results of the SMELL's assessment. We also discuss the interpretability of the similarity space (S-space) and conduct a performance evaluation comparing SMELL with three distance metric learning approaches from pyDML[1] [31, 66, 96], five deep metric learning approaches [12, 33, 80, 86, 102], and Euclidean distance.

5.1 Ablation Study

For a better understanding of our proposal, we conducted an ablation study. Therefore, we evaluated SMELL for different regularization calibration values. We evaluated SMELL with and without the reconstruction error ($r_r = 0$), with and without the repulsive error ($r_d = 0$), and without both ($r_d = r_r = 0$). The other default values adopted in our experiments, can be found in Section 4.

Besides, we also evaluated the behavior of our proposal when using S-space only for training. We then use for prediction a version of SMELL without the S-space (using Euclidean distance), we named this approach SMELL (Euclidean). Therefore, after training the model using S-space, we observe only the latent space to perform the similarity metric's extraction, i.e., we consider that the similarity between two objects is the Euclidean distance between them in the latent space. It is also worth noting that we use the same default values adopted in our experiments (without any restriction on r_d and r_r). We provide a complete report of our results in Table 5.1.

We see that among the usual SMELL methods when we take $r_d = 0$, the proposal tends to have performance degradation. This behavior is easily seen in the dataset ring, in which SMELL ($r_d = 0$) and SMELL ($r_d = r_r = 0$) has an accuracy of 0.6536 and 0.6610, respectively. Comparing this value with the best result, we have a difference of more than 20%. The proposal with $r_d = 0$ has the worst performance in all four metrics analyzed (excluding Euclidean).

[1]https://pydml.readthedocs.io/en/latest/index.html

Dataset	SMELL	SMELL	SMELL	SMELL	SMELL
	($r_r = 0$)	($r_d = 0$)	($r_r = r_d = 0$)	(Euclidian)	(S-space)
Appendicitis	78.90 ± 11.12	79.09 ± 11.02	79.09 ± 09.59	77.18 ± 9.13	**80.19 ± 7.74**
Balance	97.00 ± 01.03	97.00 ± 01.12	97.00 ± 1.02	98.40 ± 1.22	**98.88 ± 1.02**
Banana (10%)	89.44 ± 4.89	90.76 ± 08.83	90.01 ± 3.76	87.73 ± 1.14	**90.95 ± 4.42**
Bupa	64.58 ± 09.92	**67.05 ± 09.62**	66.37 ± 8.83	55.15 ± 10.12	63.92 ± 6.85
Cleveland	**52.22 ± 07.42**	51.20 ± 05.65	52.21 ± 6.32	51.93 ± 7.22	51.24 ± 8.49
Glass	67.52 ± 14.71	67.81 ± 12.16	66.91 ± 11.31	**70.75 ± 11.46**	66.94 ± 13.24
Ionosphere	**89.75 ± 5.14**	88.89 ± 02.37	**89.75 ± 4.98**	84.88 ± 6.79	89.47 ± 4.59
Iris	95.33 ± 04.00	94.67 ± 04.47	94.67 ± 4.27	95.33 ± 4.27	**96.00 ± 3.26**
Letter (10%)	**80.30 ± 3.72**	77.58 ± 08.40	77.77 ± 3.82	62.96 ± 8.62	77.77 ± 4.72
Magic (10%)	84.44 ± 02.98	84.44 ± 02.74	**84.49 ± 2.37**	78.92 ± 7.11	83.49 ± 2.58
Monk-2	**100 ± 0.00**	**100.00 ± 0.00**	**100 ± 0.00**	**100 ± 0.00**	**100 ± 0.00**
Movement-libras	86.21 ± 04.44	84.72 ± 04.44	85.56 ± 5.39	83.33 ± 6.92	**87.78 ± 3.61**
Newthyroid	**97.71 ± 02.25**	**97.71 ± 03.57**	96.77 ± 2.29	90.71 ± 10.97	96.77 ± 2.11
Phoneme (10%)	82.77 ± 03.89	**83.87 ± 04.97**	82.21 ± 3.04	81.29 ± 4.89	81.48 ± 4.31
Pima	70.44 ± 05.17	70.98 ± 05.96	**71.75 ± 2.65**	69.68 ± 3.87	70.44 ± 3.52
Ring (10%)	**89.32 ± 21.56**	65.36 ± 19.21	66.10 ± 3.11	71.80 ± 16.55	89.22 ± 4.56
Satimage (10%)	83.49 ± 02.94	85.20 ± 03.12	84.58 ± 4.45	**85.37 ± 3.11**	84.11 ± 3.57
Segment (10%)	**90.48 ± 04.48**	89.05 ± 07.22	89.05 ± 4.42	90.27 ± 4.76	89.76 ± 3.96
Sonar	84.05 ± 10.53	**85.52 ± 10.92**	84.55 ± 10.44	81.76 ± 12.88	83.59 ± 11.26
Thyroid (10%)	95.59 ± 02.43	95.56 ± 02.15	**95.69 ± 2.51**	94.71 ± 2.24	94.71 ± 2.45
Titanic (10%)	62.41 ± 16.07	62.56 ± 05.23	63.76 ± 16.22	**73.13 ± 6.13**	66.97 ± 15.85
Twonorm (10%)	96.00 ± 01.41	96.75 ± 06.07	97.00 ± 1.48	93.78 ± 13.33	**97.43 ± 1.76**
Vehicle	84.40 ± 02.00	83.34 ± 12.02	**84.87 ± 3.33**	78.32 ± 18.33	84.40 ± 2.63
Vowel	98.12 ± 00.79	98.12 ± 15.63	**98.99 ± 1.09**	98.48 ± 1.04	**98.99 ± 0.90**
Wdbc	**96.65 ± 02.99**	96.47 ± 02.40	**96.65 ± 2.99**	89.11 ± 14.63	**96.65 ± 2.89**
Wine	97.77 ± 03.59	97.71 ± 05.12	**98.30 ± 5.11**	97.19 ± 5.14	97.77 ± 5.11
Wisconsin	**96.21 ± 02.14**	95.91 ± 01.84	95.62 ± 1.87	95.91 ± 2.48	**96.21 ± 2.26**
Accuracy_AVG	0.8254	0.8169	0.8178	0.7994	**0.8268**
Ranking_AVG	**2.2857**	2.8571	**2.2857**	3.6429	**2.2857**
Diff_AVG	0.0116	0.0201	0.0193	0.0376	**0.0102**
# of 1st	9	5	9	4	**10**

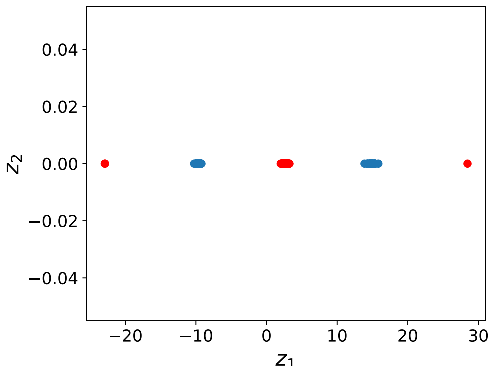
Table 5.1. Performance comparison of ablation study when using KNN classification for 27 different datasets.

When $r_r = 0$, we observe a slight impact on the result (when compared to $r_d = 0$), but for the datasets Appendicitis, Vowel, Banana (10%) and Twonorm (10%) (see Table 5.1), changing r_r to zero, made SMELL stop being the first position (when analyzing accuracy), to the second last position.

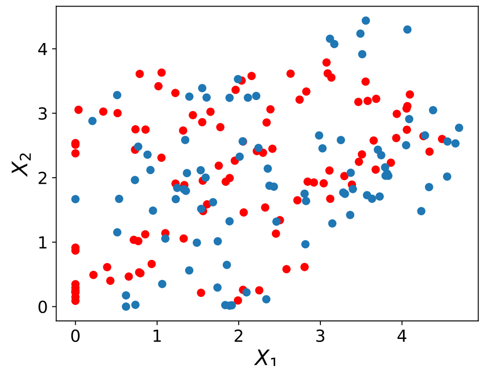
When we consider the case of SMELL with the Euclidean metric (instead of S-space), our proposal has the worst performance among the cases analyzed for the four metrics adopted. In particular, we see that the average difference for the first place (DIFF_AVG) has increased 200%. In addition, it is worth noting that for the datasets Twonorm (10%), Banana (10%), Wdbc, Movement_libras and Appendicites (see Table 5.1), SMELL goes from first place to last place. This evidencing the limitation of the Euclidean metric (even using the function f_Θ found by our proposal).

We hypothesized that SMELL tends to find a representation in S-space that captures similarity semantics. SMELL (S-space) has the best performance among all the versions used, evidence of the last statement. In addition, Repulsive regularizer tends to increase the separability of latent space. This fact shows the importance of the repulsive regularizer.

Moreover, we evaluated the behavior of SMELL in comparison with an autoencoder (without markers). Figure 5.1 shows the behavior of the encoder output under SMELL

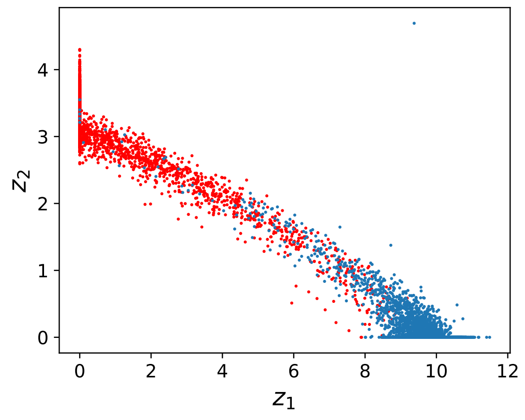


(a) Latent space ($n = 2$) from SMELL.

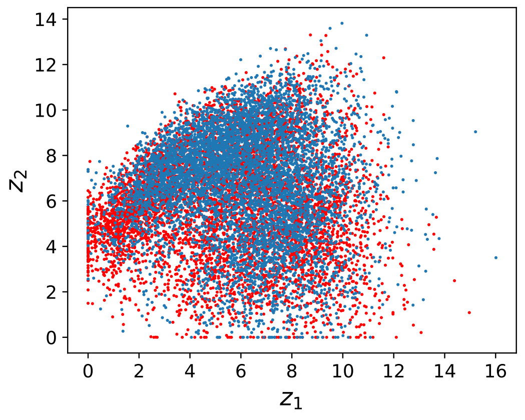


(b) Encoder ($n = 2$) without markers.

Figure 5.1. Sonar Dataset: Latent Space and Encoder space Analysis.



(a) Latent space from SMELL.



(b) Encoder output without markers.

Figure 5.2. MNIST Dataset: Latent Space and Encoder space Analysis.

and the autoencoder. In Figure 5.1a, whose encoder was used with SMELL, the classes have well-defined groups, differently to Figure 5.1b, where there is a greater dispersion of the classes, with no clustering pattern being observed. The same behavior is found in the MNIST dataset, as shown in Figure 5.2.

5.2 Performance Evaluation

To compare our results to other techniques present in the literature, we used the datasets and the metrics appointed in Section 4, with $n = 64$ (latent dimension). The summary of results can be found in Table 5.2. The second column indicates whether the approach is based on Metric Learning (MeL) or Deep Metric Learning (DMeL) techniques. We compare SMELL to metric learning approaches [96, 31, 66], deep metric learning

Dataset	ANMM[96]	KDMLMJ[66]	Contrastive[33]	MSLoss[102]	Triplet[80]	NCA[31]	NPair[86]	FastAP[12]	Euclidian	SMELL
Appendicitis	84.27 ± 10.64	**85.09 ± 9.94**	84.18 ± 9.95	84.09 ± 11.02	81.27 ± 10.79	84.09 ± 9.39	84.90 ± 08.61	84.09 ± 10.07	84.27 ± 10.64	81.09 ± 7.74
Balance	80.81 ± 4.22	79.83 ± 04.32	78.93 ± 23.54	98.24 ± 01.12	96.17 ± 2.03	95.84 ± 2.28	89.74 ± 08.00	98.24 ± 01.31	80.16 ± 5.06	**98.88 ± 1.02**
Banana (10%)	56.18 ± 12.43	55.62 ± 13.25	89.06 ± 12.97	84.38 ± 08.83	90.19 ± 3.85	86.79 ± 2.15	89.44 ± 05.25	72.59 ± 13.55	56.18 ± 12.43	**90.95 ± 4.42**
Bupa	60.95 ± 9.04	65.00 ± 7.888	49.63 ± 5.01	67.05 ± 09.62	**68.63 ± 5.28**	57.79 ± 6.62	57.16 ± 08.84	58.77 ± 11.71	64.95 ± 9.04	63.92 ± 6.85
Cleveland	55.14 ± 6.92	48.83 ± 06.64	55.54 ± 2.76	54.55 ± 05.65	56.92 ± 6.92	50.23 ± 6.76	**56.99 ± 06.61**	55.80 ± 8.29	55.13 ± 6.92	51.24 ± 8.49
Glass	68.13 ± 10.26	**68.95 ± 10.78**	51.11 ± 17.89	66.71 ± 12.16	68.30 ± 11.82	67.02 ± 10.40	60.66 ± 08.20	64.65 ± 9.33	68.13 ± 10.25	66.94 ± 13.24
Ionosphere	85.18 ± 4.92	84.04 ± 03.43	86.16 ± 19.30	**94.02 ± 02.37**	92.89 ± 2.89	88.88 ± 4.15	86.33 ± 07.32	92.60 ± 3.39	85.18 ± 4.92	89.47 ± 4.59
Iris	94.00 ± 4.67	96.00 ± 04.00	**96.67 ± 4.47**	**96.67 ± 04.47**	96.00 ± 2.72	95.33 ± 4.27	94.67 ± 04.00	**96.67 ± 3.33**	94.00 ± 4.67	96.00 ± 3.26
Letter (10%)	78.93 ± 10.95	**85.7 ± 11.83**	25.95 ± 21.15	77.64 ± 08.40	46.54 ± 23.79	61.55 ± 21.08	46.10 ± 08.06	46.55 ± 23.79	79.13 ± 8.95	77.77 ± 4.72
Magic (10%)	62.28 ± 9.68	77.34 ± 09.54	73.5 ± 5.57	**84.65 ± 02.74**	84.07 ± 2.64	67.99 ± 9.85	82.07 ± 02.58	84.57 ± 2.60	62.82 ± 9.68	83.49 ± 2.58
Monk-2	95.89 ± 3.92	**100 ± 0.00**	71.44 ± 12.13	96.52 ± 03.29	98.37 ± 1.48	**100 ± 0.00**	95.21 ± 04.92	98.86 ± 1.52	98.18 ± 2.72	**100 ± 0.00**
Movement-libras	80.83 ± 3.39	87.22 ± 05.28	53.33 ± 21.64	82.78 ± 04.44	75.00 ± 10.54	81.67 ± 4.51	33.61 ± 11.81	67.78 ± 20.38	80.12 ± 3.39	**87.78 ± 3.61**
Newthyroid	95.36 ± 2.95	94.87 ± 03.58	97.25 ± 3.16	96.77 ± 03.57	95.84 ± 3.84	**97.73 ± 3.05**	96.32 ± 04.95	97.25 ± 3.65	95.37 ± 2.95	96.77 ± 2.11
Phoneme (10%)	81.91 ± 8.87	81.91 ± 08.70	70.16 ± 9.79	82.39 ± 04.97	79.96 ± 5.88	**83.4 ± 9.81**	75.73 ± 04.41	76.64 ± 8.12	62.71 ± 8.87	81.48 ± 4.31
Pima	72.93 ± 4.30	69.28 ± 04.04	64.46 ± 4.65	74.01 ± 05.96	**74.11 ± 3.96**	70.44 ± 3.78	72.80 ± 05.22	72.55 ± 3.94	72.93 ± 4.30	70.44 ± 3.52
Ring (10%)	51.70 ± 6.39	59.61 ± 07.46	73.15 ± 11.33	79.45 ± 19.21	**93.91 ± 4.84**	51.19 ± 9.57	82.71 ± 04.11	70.10 ± 20.91	51.77 ± 6.39	89.22 ± 4.56
Satimage (10%)	84.18 ± 9.65	74.58 ± 09.15	59.50 ± 17.88	**86.35 ± 03.12**	82.89 ± 4.25	64.93 ± 21.39	69.82 ± 11.49	83.99 ± 4.72	52.93 ± 19.66	84.11 ± 3.57
Segment (10%)	73.12 ± 12.48	83.69 ± 04.05	67.11 ± 17.72	84.76 ± 07.22	92.21 ± 4.68	64.88 ± 15.37	57.02 ± 16.15	**92.74 ± 3.85**	53.13 ± 22.48	89.76 ± 3.96
Sonar	83.07 ± 10.58	82.09 ± 12.12	59.26 ± 10.03	86.00 ± 10.92	**88.43 ± 9.13**	86.50 ± 9.55	79.33 ± 11.93	85.05 ± 10.91	82.70 ± 10.57	83.59 ± 11.26
Thyroid (10%)	90.99 ± 2.07	92.65 ± 02.21	91.68 ± 1.79	94.48 ± 02.15	93.69 ± 2.19	90.14 ± 2.18	93.74 ± 02.50	64.17 ± 31.81	90.99 ± 2.07	**94.71 ± 2.45**
Titanic (10%)	61.74 ± 16.72	71.82 ± 12.67	73.13 ± 5.05	72.67 ± 05.23	73.18 ± 5.16	64.16 ± 9.05	72.67 ± 05.34	73.61 ± 5.78	**74.14 ± 10.96**	66.97 ± 15.85
Twonorm (10%)	93.83 ± 4.54	95.95 ± 04.39	91.89 ± 10.69	95.95 ± 06.07	96.62 ± 1.38	95.95 ± 8.16	**98.11 ± 01.38**	93.78 ± 10.22	97.5 ± 4.59	97.43 ± 1.76
Vehicle	70.21 ± 3.66	65.95 ± 03.46	40.23 ± 10.55	74.13 ± 12.02	81.91 ± 4.02	74.71 ± 3.13	45.17 ± 08.40	**85.58 ± 3.45**	65.95 ± 3.66	84.40 ± 2.63
Vowel	97.77 ± 0.98	98.28 ± 01.37	88.69 ± 5.58	57.37 ± 15.63	95.25 ± 3.10	97.68 ± 1.20	72.12 ± 06.70	78.79 ± 10.75	98.28 ± 0.98	**98.99 ± 0.90**
Wdbc	96.48 ± 2.49	92.79 ± 03.32	95.61 ± 4.72	**97.36 ± 02.40**	95.18 ± 1.97	92.44 ± 3.16	93.86 ± 07.49	96.48 ±2.72	92.79 ± 2.49	96.65 ± 2.89
Wine	95.52 ± 4.17	97.71 ± 02.80	95.51 ± 4.72	97.22 ± 05.12	82.56 ± 2.22	97.44 ± 2.54	89.25 ± 13.83	96.63 ± 5.11	69.18 ± 4.16	**97.77 ± 5.11**
Wisconsin	96.52 ± 2.79	**96.66 ± 02.71**	74.78 ± 2.51	96.05 ± 01.84	96.05 ± 2.36	95.82 ± 1.73	95.92 ± 02.58	96.22 ± 2.33	96.39 ± 2.94	96.21 ± 2.26
Accuracy_AVG	0.7768	0.7827	0.6923	0.8076	0.8115	0.7732	0.7380	0.7801	0.7486	**0.8268**
Ranking_AVG	5.71429	4.96429	7.17857	3.7857	3.9643	5.67857	6.2143	4.7857	5.89286	**4.3214**
Diff_AVG	0.0726	0.0667	0.1571	0.0418	0.0379	0.0762	0.1114	0.0693	0.1007	**0.0242**
# of 1st	0	5	1	5	4	3	2	3	1	**7**

Table 5.2. Performance comparison of some distance metrics approaches and SMELL when using KNN classification for 27 different datasets.

approaches [12, 33, 80, 86, 102], and the usual Euclidean distance. The k-fold cross-validation results are shown by averaging the standard deviation and accuracy values reported by the process.

SMELL achieved the best accuracy results in 7 (# of 1st) datasets, thus surpassing all other analyzed algorithms (improving 40% more datasets when compared with second best). KDMLMJ and MSLoss, the second-best, achieved the best result in 5 datasets. SMELL achieved an accuracy of 0.8268 (Accuracy_AVG). The second-best, Triplet, achieves 0.8115, and the third-best, MSLoss, achieves 0.8081. In a simple dataset (Monk-2), SMELL achieves 100%. It is worth mentioning that SMELL, even in some situations its performance is not the best, reaches accuracy close to the best algorithm. For instance, the average distance between SMELL and the best algorithm is 2.26% (Diff_AVG), improving its average distance by 67.70% and 84.96% compared to Triplet (second-best) and MSLoss (third-best), respectively. Finally, when we average the ranking, SMELL achieved an average of 3.6429 (Ranking_AVG), the smallest value among all algorithms. The second-best was MSLoss, reaching 3.7857.

In Figure 5.3, we compare SMELL's accuracy with all othe approaches used in this paper. We noticed that SMELL, in all cases, manages to overcome the techniques presented when we compare the number of individual hits, i.e., the number of datasets that SMELL exceeds the accuracy of the analyzed baseline. Besides, we noticed a small scattering of the blue dots around the black line compared to the red triangles' behavior. It indicates that even when SMELL performs worst than another approach, its results are close to the best.

Analyzing the metric learning approaches only (see Table 5.2), we see that KDMLMJ and NCA algorithms achieve better results (among the algorithms adopted as baseline) when considering the metric that counts the number of times that the algorithm's accuracy surpassed all the others. This behavior is because the algorithms have been evaluated with KNN, and these algorithms were specifically designed to improve this classifier.

Considering the MNIST data, we can see that our proposals achieves considerably better results, particularly for lower dimensions ($d = \{1, 2, 4\}$). This characteristic is highlighted by the area under the curve, as seen in Figure 5.4. We noticed that some techniques are highly dependent on the feature extractor. For example, the Contrastive loss [33] was proposed to capture coherent semantics in a latent space. However, the proposal aims to capture the semantics of the data, but, without the aid of convolutions layers, we observe a performance degradation when compared to other techniques.

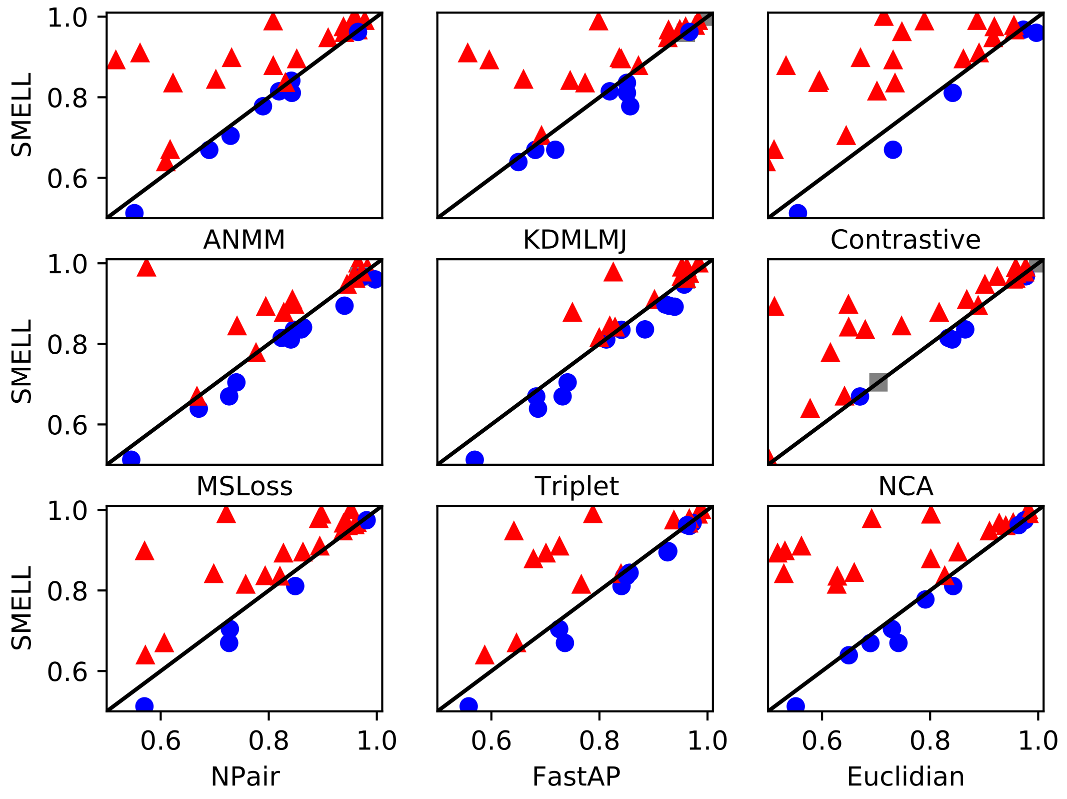


Figure 5.3. Comparison between SMELL and other proposals. Red triangles, blue dots and gray squares indicate when the SMELL is superior, inferior and has the same mean accuracy result, respectively, when compared to other methods.

5.3 Behavior Analysis

Our proposal is based on optimizing the parameter set $\boldsymbol{\Sigma} = \{\Theta, \Theta', \boldsymbol{M}\}$ using markers (with a t-student kernel).

SMELL learns a representation of input pairs that groups the points with similar and dissimilar labels around their respective markers. We can observe this behavior in Figure 5.5. In this Figure, the input pairs of similar and dissimilar labels are represented by pink circles and gray triangles, respectively. In addition μ^+ and μ^- markers are represented by green and red crosses respectively. We plot some s_{ij} vectors for input pairs (x_i, x_y) of the test set for the Balance dataset. Initially, after training the autoencoder, a two-dimensional plot was created by the aid of PCA before (first figure) and after (the other figures) the optimization process.

We can observe in fist plot in Figure 5.5 (before adjusting the markers' positions) that the points do not present a well-defined cluster structure. This behavior changes when we analyze the last plot in Figure 5.5 (after adjusting the markers).

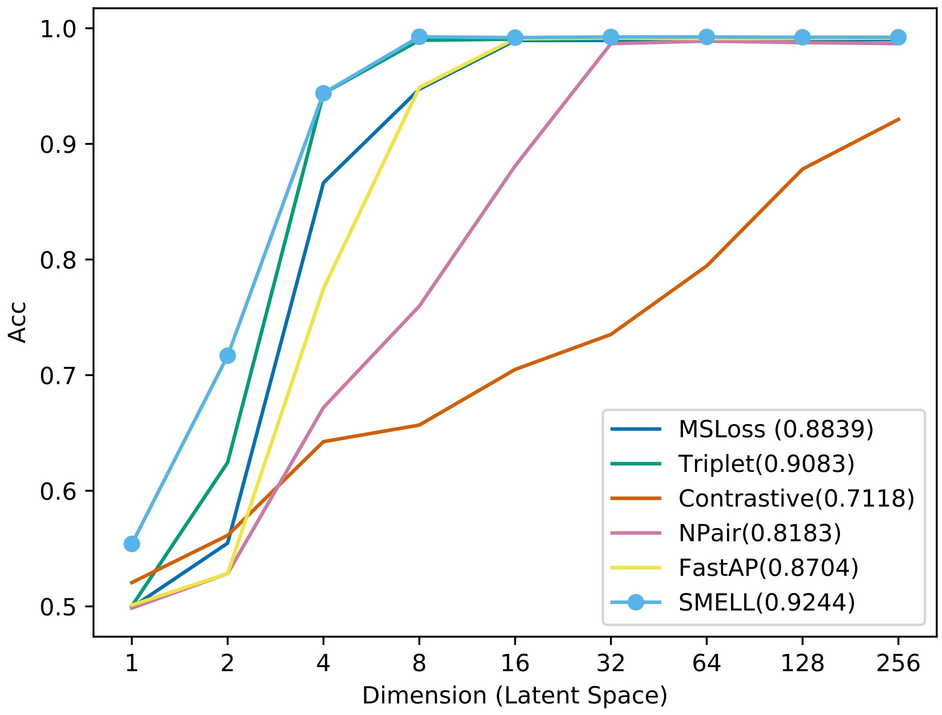


Figure 5.4. MNIST evaluation for different dimension of latent space. The AUC is reported in parentheses.

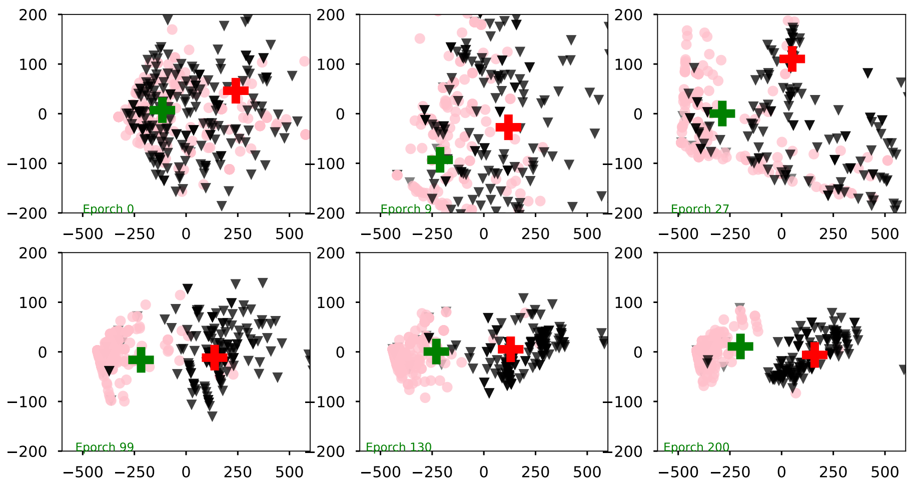
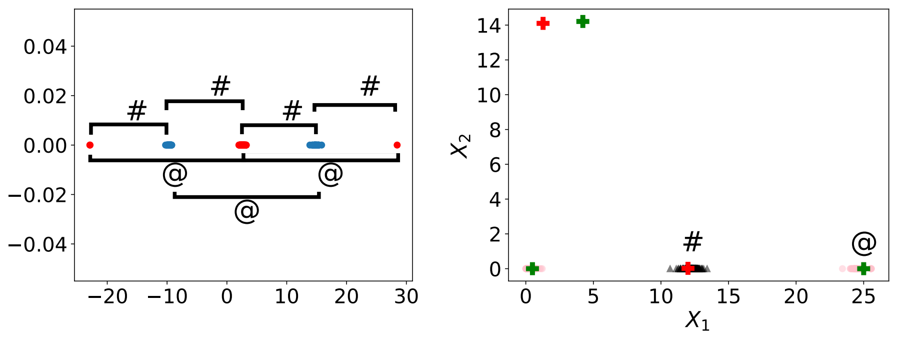


Figure 5.5. Simultaneous training of μ^+ (green) and μ^- (red) markers's position and data representation for some training epochs.

In the last plot in Figure 5.5, we can see that there are well-defined groups around the markers. Moreover, by comparing the scale of the Figures 5.5, we see that in the last case, points are more spaced, i.e., our proposal tends to group points around their respective markers. This behavior corroborates our initial hypothesis described in (**H1**).



(a) Latent space ($n = 2$) **(b)** S-space ($n = 2$)

Figure 5.6. Sonar Datasets: Latent Space and S-space Analysis for SMELL.

We observed that our proposal acts as an attractive potential. In this sense, the marker "pulls" the favorable points (similarity mark "pulls" similar points). Therefore, their movement resembles a Group Mobility Model [38], i.e., the marker is being positioned, and the points go "following" the leader as a "caravan" of nomads. At the same time, the markers tend to repulse themselves.

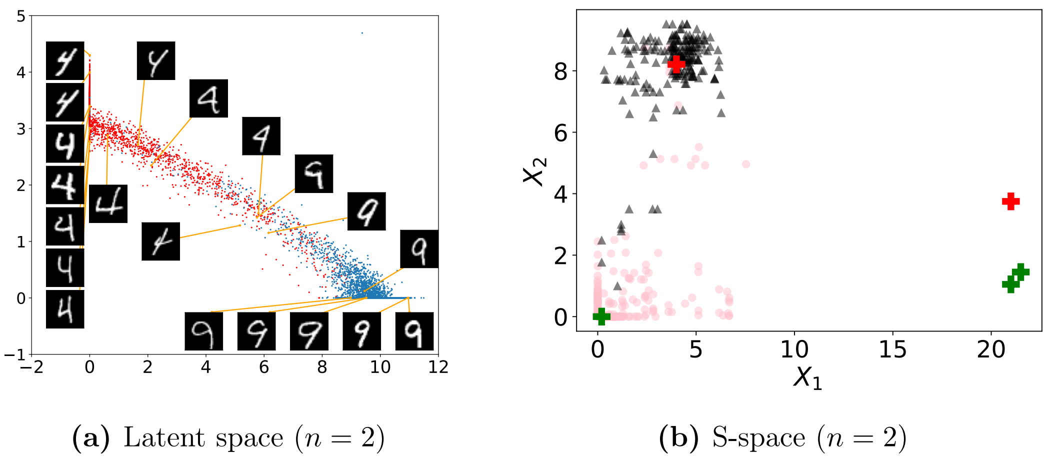
This can be seen as such an intense attracting field, which locks the movement dynamics of the points closest to the markers.

5.4 Latent space and S-space analysis

For a better understanding of the latent space found by SMELL, we analyzed the behavior of our proposal using the sonar and MNIST datasets as shown in Figure 5.7 and 5.6. For the sake of visualization, in this analysis, we use the setup discussed in Section 4 with $n = 2$ (latent dimension). Figures 5.6a and 5.7a show the *latent feature space* (output of encoder). Observe that in these figures, points represent individual objects. Red and blue points represent different classes. There are two classes in sonar dataset, and we show only two classes of MNIST (handcraft digits 4 and 9). These *latent feature spaces* result from the joint optimization process of the autoencoder and the S-space.

In Figure 5.6a, we observe that red points are grouped in different regions far apart at a distance approximately constant, denoted by @. Similarly, blue points are apart at a distance approximately constant, @. Different clusters are apart at a distance approximately constant, denoted as #.

Figures 5.6b and 5.7b show a random sample of 400 data pairs from the sonar dataset mapped to S-space (200 similar and 200 dissimilar pairs). In S-space, points



(a) Latent space ($n = 2$) **(b)** S-space ($n = 2$)

Figure 5.7. MNIST Datasets: Latent Space and S-space Analysis for SMELL.

represent a pair of objects. Pink circles and black triangles represent similar and dissimilar labels, respectively. Also, similarity and dissimilarity markers are represented by green and red crosses, respectively.

Figure 5.6b show some clustered regions. The region grouped by the similarity marker (closer to the origin) is responsible for grouping elements of similar classes with a distance closer to 0. This result corroborates with the Proposition 3.2.1. The same behavior is found in Figure 5.7b, where we observe a green cross close to the origin.

However, in Figure 5.6b, we observe some similar objects mapped to points that have distance close to @, instead of zero. The green cross located at @ is responsible for creating the similarity region that represents this situation. Other regions of similarity and/or dissimilarity can occur, depending on the data complexity, and are represented by other green/red crosses. Dissimilarity regions are depicted as #. Therefore, in the space found by SMELL, we see the behavior of multiple groups, separated by distances determined by the similarity/dissimilarity markers (labels # and @).

Observe in Figure 5.7a the soft transition from digit 4 to 9, which shows that the S-space preserves the connection between these two similar digits. This effect is captured even though we do not use any data-specific feature extractor, such as convolution layers. In SMELL, the encoder can be switched by any feature extractor tailored explicitly for the input data.

In Figure 5.7a, we observe that the handcrafted digits four are grouped (on the left). We observe that even in this group, the similarities between the digits remain. The first two digits in the top-left region correspond to numbers with thicker writing and slightly rotated, and as we go down in the latent space, the shape of the digit starts to become thinner. This behavior indicates a gradient that represents the thickness of the object. This same behavior occurs similarly to digit 9. There is a transition from groups of digits 4 to 9, i.e., there is a semantic in this transition. As we move along the diagonal that connects the two groups, gradually, the numbers 4 resemble the number 9, so that, in the middle of the diagonal, it is tough to differentiate between these two numbers. It is also worth noting that, the further away from the denser regions of the points cloud, the less readable are the numbers, for instance, the two digits four depicted below the transition diagonal. We see that our proposal uses markers to help in the convergence and finds a latent space that preserves the semantics of the original data. This behavior corroborates our initial hypothesis described in (**H2**).

It is also worth noting that our proposal has no sensitive learning in the presence of multiple markers, i.e., even in this experiment that we have defined three similarity markers and two dissimilarity markers, our proposals does not use all. This behavior is emphasized in Figures 5.6b and 5.7b , where our proposal removes excessive markers from the groupings by locating these markers far away from the data. This behavior is an indication that the number of markers is a virtual parameter of the model.

We hypothesize that markers group data points considered similar (in our context, which have the same labels) and dissimilar (different labels) in disjoint regions. Figures 5.7b and 5.6b show this behavior, where we can see similar and dissimilar groups in distinct (and disjoint) regions in S-space. In addition, we can notice in Figure 5.6a that our proposal allows different clusters for the same class (*optimal latent space*), as mentioned in Definition 3.2.1.

Chapter 6

Case of Study: Diagnostic Aid Software for Leishmaniosis Detection

6.1 Introduction

Leishmaniasis, caused by species of the intracellular protozoan of the genus *Leishmania*, is a neglected and infectious vector-borne disease. It occurs in poorest countries and most vulnerable populations with difficult access to health services. Different species of *Leishmania* can cause a variety of clinical manifestations, including the Visceral Leishmaniasis (VL), that is more severe and often fatal if not diagnosed and treated [105]. In the Americas, VL is endemic in 12 countries. South America countries, such as Brazil, Argentina, Colombia, Paraguay, and Venezuela are among those with the highest case records, knowing that Brazil reports 96% of the total of cases. Some Central America countries, such as Honduras and Guatemala, previously presented sporadic VL cases, but they have reported an increasing number of cases in last years [105]. In southern Europe, it is a primary opportunistic infection in patients with acquired Immunodeficiency Syndrome [70].

Serological or parasitological techniques and the Polymerase Chain Reaction (PCR) are typically diagnostic tools used for the VL diagnosis. PCR-based assays are the main molecular diagnostic tools, especially in immunosuppressed patients [6]. However, it remains complex and expensive, and in most VL-endemic countries, they are restricted to a few teaching hospitals and research centers [26]. Serological diagnosis may lack specificity due to asymptomatic infections [104], but parasitological techniques are highly specific. In this process, smears or biopsies can be obtained from liver, lymph nodes, bone marrow, and spleen, but only smears from bone marrow and spleen have been used routinely [37].

Smears are simple to prepare, and their direct examination is usually the best diagnostic method in poorer areas where PCR is not available [26]. In this way, bone marrow examination is an authentic method for diagnosis of VL. This procedure includes

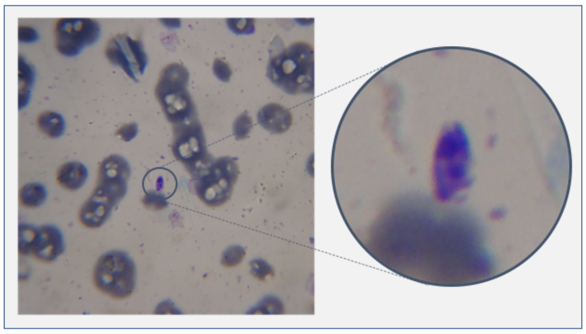


Figure 6.1. *Leishmania* amastigote (magnification: 400×). Source: [27]

the directly microscopic observation of the parasite. However, the miniature size of the protozoan makes this a tedious task that can be very time-consuming [26]. Also, when faced with the presence of the protozoan, the physician may not be sure about whether it is a parasite, once it may resemble other structures present in the image content. Figure 6.1 presents an example of *Leishmania* amastigotes in a bone marrow microscopic color image.

Several Deep Learning-based methods for image analysis have been used in the interpretation of medical and biomedical images. In particular, Convolutional Neural Networks (CNNs) have rapidly become a methodological way for analyzing these images. The most common methods include learning algorithms for image classification [74], object detection, object segmentation image registration, and other tasks.

In the last years, several approaches have been proposed for histopathology image analysis [87], including that ones that aim the automated detection of parasites [115].

Relli et al. [75] propose an automatic method to counting trypanosomatid amastigotes in human cells infected with Chagas disease. This method is divided in a sequence of steps that include a initial morphological operation that removes complex background from original image; an unsupervised classification that clustered the remain objects and a thresholding operation to preserve the interest objects: the infected cell.

In this chapter, we propose a new version of the SMELL called L-SMELL for the classification of leishmaniasis samples. In this way we use S-space to obtain a new representation of the microscope samples, however with some modifications in the regularization function, as will be shown below.

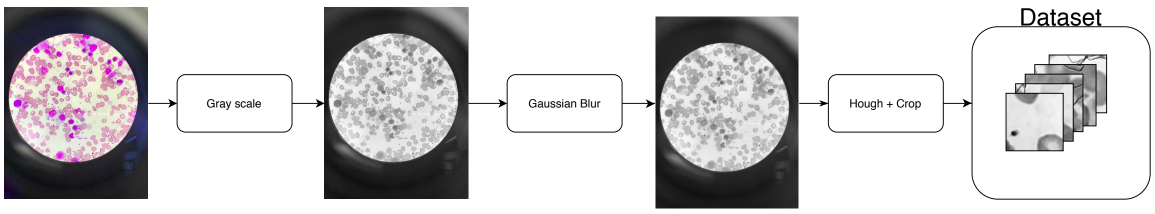


Figure 6.2. Step-by-step of the pre-processing used in this chapter.

6.2 Methodology

6.2.1 Data Acquisition

6.2.1.1 Our data

In this chapter, we built a new dataset containing samples of leishmaniasis labeled by specialists. This dataset was collected by doctors at the University Hospital linked to the Federal University of Alagoas. All capture preprocessing was carried out by the doctors, thus resulting in 76 samples containing leishmaniasis.

6.2.1.2 Dataset 2

In [27], the authors propose a new dataset for patients with visceral leishmaniasis. For data acquisition, a digital camera (Sony DSC-H9) coupled on an optical microscope (Olampus-CH40RF200) were used and 45 data were captured for Automatic Boundary Extraction of Leishman Bodies.

6.2.2 Data Preprocessing: Region of Interest Identification

To analyze the images obtained in the Section 6.2.1, we performed some transformation steps, as we can see in the Figure 6.2.

Initially, We captured the images in color and transformed them into a gray-scale. We used a Gaussian convolution filter to blur an image and to reduce the noise present in the object of interest. The result of this operation is a smoothing of the image, reminding the visualization through a translucent screen. This smoothing is widely used in the image preprocessing stage to highlight the image structure at different scales. In practice, the Gaussian filter is a convolution operation in the image with a Gaussian function.

Given the image smoothed with a Gaussian convolution filter, we apply a circular transformation. Circle Hough Transform (CHT) is a traditional technique used in Digital Image Processing for detecting circular objects in a digital image. We can see in the schematic in the Figure 6.2, we removed the edge from the microscope to eliminate regions that we are not interested in analyzing. After removing the circular border of the images, we crop them in a patch of size 200 x 200 pixels.

6.2.3 Experiments

Typically medical image datasets are unbalanced. So, we designed experiments with some versions of the datasets described in the previous section, as we can see:

– Experiment I: We carried out a subsample to mitigate the problem of imbalance. Thus, for each image of the minority class, choose 1 random image of the marjoritary class. Then, at the end of the sampling, we have a new reduced version of the original dataset having the same number of samples from both classes.

– Experiment II: Following the previous experiment, we built a new dataset using a subsample for a 4:1 ratio, i.e., for each image of the minority class, we have 4 images of the marjoritary class.

– Experiment III: We use all the images contained in the dataset.

6.3 Our Proposal

6.3.1 L-SMELL

Our proposal can be divided into 3 steps:

- Data representation algorithm: In this step we will build a representation for the problem data.
- Metric learning algorithm: For a given new representation obtained from the previous step, we will build a similarity function for a given input data pair with S-space as we defined in section 3.2.2.
- Metric-based algorithm: We will apply a metric based classification algorithm to solve the problem.

6.3.2 Loss Function

Our proposal estimates the latent feature space and the S-space simultaneously like Section 3.2.2. We can define through our proposal the similarity function $d_{\Omega}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that receives a pair of images and returns the similarity between them. To estimate the parameter set $\Omega = \{\Theta, \mathcal{M}\}$, we built a cost function J as based on Cross-entropy loss H_c.

In addition, we defined the contrastive loss R_c as a regularization function to effect overfitting and assist in the convergence of the model, as we discussed in Section 6.4.3.

Proposed by [33], contrastive loss is a function that minimizes the distance between similar points and imposes a restrictive distance between the points of dissimilar classes. We can define the contrastive loss for a pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as

$$R_c(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} ||f_{\Theta}(\boldsymbol{x}_i) - f_{\Theta}(\boldsymbol{x}_j)||_2^2, \text{ if } (\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ is similar} \\ [p - ||f_{\Theta}(\boldsymbol{x}_i) - f_{\Theta}(\boldsymbol{x}_j)||_2]_+^2, \text{ if } (\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ if dissimilar,} \end{cases} \tag{6.1}$$

where operator $[.] = max(0, .)$ is the hinge function. In this chapter, following [33], we use the margin $p = 1$.

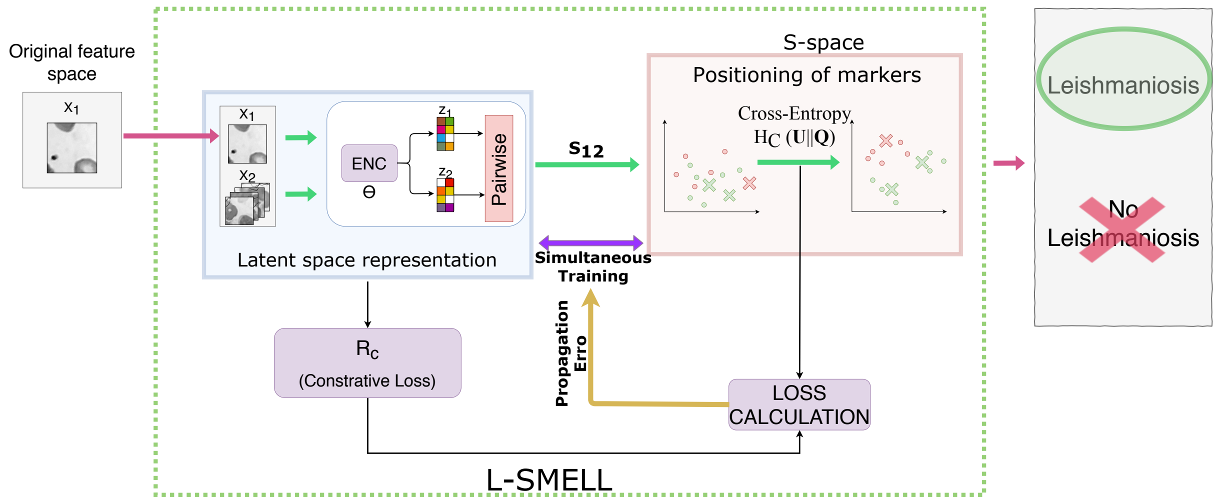


Figure 6.3. The left side represents the Encoder, and the right side represents the optimization process for the markers' position for $\mathcal{M} = \{\boldsymbol{\mu}_1,\ \boldsymbol{\mu}_2,\ \boldsymbol{\mu}_3\}$. In this example, we used two positives and one negative marker. The rightmost green arrow shows a representation of the markers' position optimization step by using Cross-Entropy divergence and Contrastive regularization functions.

Let $\mathcal{Q} = \{q_{ij}\}$, the L-SMELL output, be the set that contains the pairs $q_{ij} = (q^+_{ij},\ q^-_{ij})$ corresponding to the probability of the elements of a pairwise input $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be similar or dissimilar, respectively. The optimal hyperparameters set can be defined as $\Omega^* = \arg\min_{\boldsymbol{\Omega}} J(\{\mathcal{X} \times \mathcal{X}\})$, where

$$J(\{\mathcal{X} \times \mathcal{X}\}) = H_c(\mathcal{U}||\mathcal{Q})r_{HC} + r_c R_c, \tag{6.2}$$

where r_{HC} is a constant for calibration for cross-entropy, r_c is a calibration term for contrastive loss, and $\boldsymbol{u}_{ij} \in \mathcal{U}$ is defined as $\boldsymbol{u}_{ij} = (1, 0)$ if $\boldsymbol{x}_i$ has same label as $\boldsymbol{x}_j$ and $\boldsymbol{u}_{ij} = (0, 1)$, otherwise. The scheme for our proposer can be see in Figure 6.3.

6.3.3 Revisiting Contrastive loss

In this chapter, we present L-SMELL, a novel deep metric learning formulation to improve the deep metric learning approach. We define a *k-optimal representation* as a space that the K-nearest neighbor classification (KNN) with $(k-1)$ neighbors obtains 100% for accuracy.

Contrastive loss [33] was proposed to compare a pair of objects (x_i, x_j) and output if the objects have equal/different classes. It maps the objects to a representation space using an encoder function f_Θ. It defines a loss function $J(x_i, x_j, y_{ij}) = \frac{y_{ij}}{2}||f_\Theta(x_i) -$

$f_\Theta(x_j)||_2^2 + \frac{(1-y_{ij})}{2}[max(0, m - ||f_\Theta(x_i) - f_\Theta(x_j)||_2)]^2$, where m is a margin parameter; y_{ij} equals 1 when the pair has the same class and 0 otherwise.

Thus, Contrastive loss can be interpreted as an energy function that minimizes the distance between points of similar classes and imposing the distance between objects of different classes to be larger than m.

Suppose that an encoder finds a *k-optimal representation* for binary classification problems, and this space is formed by clusterings so that two elements of the same group belong to the same class. So, given a pair (k_i, k_j) of the same group, we have $||k_i - k_j||_2 = 0$, and the groups of elements of different classes are distant of each other by m. We can see a schematic for this space in Figure 3.3.

Therefore, given two elements (k_i, k_j) of different classes, we have that $\mathbb{E}[||k_i - k_j||_2] = (k * m + k * m)/(2k) = m$, so that the contrastive loss equation is 0. However, analyzing the case of a pair of elements (k_i, k_j) of the same class, we have $\mathbb{E}[||k_i - k_j||_2] = ((k-1) * 0 + k * m\sqrt{2})/(2k) = m\sqrt{2}/2$. Thus, the contrastive function is grater than 0 and tends to deform the space found by the encoder.

Unlike contrastive, our proposal, considered the space mentioned above, tends to conserve the k-optimal representation space. Thus, by adding the markers in positions $\mathcal{M}^+ = \{(0,0), (m,m)\}$ and $\mathcal{M}^- = \{(m,0), (0,m)\}$, the loss of our proposal equals zero, and therefore conserves the space found.

6.3.4 Parameter Setting

We initialize all weights of the autoencoder layers from a normal distribution with mean $(\mu) = 0$ and standard deviation $(\sigma) = 0.01$. Biases were also initialized from a normal distribution, but for this case it was used mean $(\mu) = 0.5$ and standard deviation $(\sigma) = 0.01$, where this values can be found in Koch et al. [46].

A different learning rate was used for each round of training of the proposal, where the rate dropped systematically in specific periods of time during the training. This method has been implemented so that the learning rate decreases by 70% to a fixed number of training rounds. Therefore, we defined the learning rate l_t in a given training round t as being $l_t = l_0 * r^{\lfloor t/d \rfloor}$, where l_0 corresponds to the initial learning rate, r is the fact of decay and d the number of rounds that the learning rate remains constant. For this chapter we used $l_0 = 0.1$, $r = 0.7$ and $d = 100$.

We used VGG19 [84] for all deep metric learning approaches used as this works. All layers are fully connected, and we used as activation function the Rectified Linear Unit (ReLU). With grid search, we found $r_{HC} = 1$, $r_c = 10^{-1}$, similarity markers $k = 3$,

Table 6.1. Table showing the results for the analyzed dataset. The best values are represented in **bold** and * represents the values without a statistically significant difference.

Experiment I				
Proposal	ACC	PREC	REC	F1-Score
SURF + SVM	50.77% (1.54%)	25.38% (0.77%)	50.00% (0.00%)	33.67% (0.67%)
SIFT+SVM	53.64% (0.25%)	26.82% (0.13%)	50.00% (0.00%)	34.91% (0.11%)
DeCaf [25]	54.64% (1.74%)	36.32% (0.86%)	50.00% (1.56%)	39.48% (0.75%)
NPairLoss [86]	73.57% (13.65%)	74.78% (14.25%)	73.69% (13.83%)	73.25% (13.76%)
Triplet [80]	*84.26% (4.96%)	*85.33% (5.24%)	*84.19% (4.89%)	*84.15% (4.97%)
CosFaceLoss [97]	*80.38% (6.69%)	*83.14% (7.27%)	*80.36% (6.46%)	*79.99% (6.69%)
MultiSimilarLoss [101]	*81.01% (8.62%)	*82.78% (8.33%)	*81.19% (8.54%)	80.71% (8.89%)
L-SMELL	***85.47% (10.11%)**	***86.46% (9.69%)**	***85.45% (10.08%)**	***85.26% (10.28%)**
Experiment II				
Proposal	ACC	PREC	REC	F1-Score
SURF + SVM	75.24% (4.64%)	56.03% (13.57%)	57.19% (7.19%)	55.48% 9.90%
SIFT+SVM	76.66% (0.64%)	38.33% (0.32%)	50.00% (0.00%)	43.39% (0.21%)
DeCaf [25]	73.29% (0.95%)	36.64% (0.48%)	50.00% (0.00%)	42.29% (0.32%)
NPairLoss [86]	78.36% (5.00%)	72.57% (5.76%)	75.45% (5.84%)	72.81% (5.24%)
Triplet [80]	*86.76% (5.86%)	81.71% (7.91%)	*80.00% (6.55%)	80.54% (5.08%)
CosFaceLoss [97]	79.88% (6.01%)	74.27% (7.49%)	74.43% (5.85%)	73.78% (6.41%)
MultiSimilarLoss [101]	81.52% (5.66%)	75.84% (7.93%)	74.26% (8.43%)	74.28% (8.17%)
L-SMELL	***89.41% (6.77%)**	***88.77% (9.36%)**	***85.07% (8.14%)**	***86.15% (8.20%)**
Experiment III				
Proposal	ACC	PREC	REC	F1-Score
SURF + SVM	***89.93% (0.23%)**	44.97% (0.11%)	50.00% (0.00%)	47.35% (0.06%)
SIFT+SVM	89.26% (0.32%)	44.63% (0.16%)	50.00% (0.00%)	47.16% (0.09%)
DeCaf [25]	88.41% (0.06%)	44.20% (0.03%)	50.00% (0.00%)	48.92% (0.02%)
NPairLoss [86]	85.36% (3.55%)	64.13% (7.02%)	*67.34% (8.71%)	65.07% (7.31%)
Triplet [80]	82.22% (2.19%)	67.12% (9.97%)	*66.39% (8.45%)	66.93% (7.82%)
CosFaceLoss [97]	85.86% (3.57%)	65.99% (7.26%)	*69.08% (6.77%)	67.12% (7.03%)
MultiSimilarLoss [101]	*89.09% (1.61%)	71.30% (5.01%)	*66.98% (6.72%)	67.95% (5.10%)
L-SMELL	*89.76% (1.26%)	***77.54% (8.90%)**	***70.32% (5.11%)**	***72.18% (5.54%)**

dissimilarity markers $w - k = 2$, these values were used in the rest of this chapter. We realized that our proposal typically performs well when k has a value similar to $w - k$.

6.4 Results and Discussion

6.4.1 Numerical results

To compare our results with other proposals found in the literature, we used the experiments described in Section 6.2.3.

For dataset presented in 6.2.1.1, we carried out three experiments, and we show results in Table 6.1. In the first experiment, we verified the behavior of proposals with a balanced dataset. L-SMELL achieves the best value when we consider all the metrics analyzed in this experiment. The Triplet obtains the second-best result, with a difference of 1.13 %, 1.23%, 1.26 %, and 1.11% for the four metrics analyzed. The dataset was balanced, so the values of metrics are similar.

Evaluating the second experiment in Table 6.1, we found that L-SMELL and Triplet achieve the first and second-best results for all metrics analyzed in this chapter. However, we observed that in this experiment, the difference between them becomes more evident. L-SMELL obtains 89.41 % for ACC, while Triplet obtains 86.76 %, resulting in a difference of 2.65 % (a result greater than 1.13 % previously found). We analyze metrics (PREC, REC, and F1-Score) and calculate a difference of 7.06 %, 5.07 %, and 5.61 % between L-SMELL and the second-best result.

For the last experiment in Table 6.1, we analyzed the proposals for a highly unbalanced dataset, as we see **experiment III** in Section 6.2.3. For the ACC metric, we observe SURF + SVM proposal has the best value, and L-SMELL has the second-best result. Probably this behavior occurs because the dataset is unbalanced. Therefore, other metrics are needed to evaluate the performance of the techniques. Looking at the other metrics, we see that the SURF + SVM proposal has a low performance. Therefore for PREC, we found that the L-SMELL achieves the best result among the analyzed techniques with a value equal to 77.54%. MultiSimilarLoss achieved the second-best result with a value of 71.30 % for the PREC metric, resulting in a difference of 6.24 %.

For the REC metric, L-SMELL and CosFaceLoss achieve the first and second-best results with 70.32% and 69.08%. Finally, L-SMELL had the best value for the F1-Score metric with 72.18%, being the second-best result obtained by MultiSimilarLoss with 67.95%, totaling a difference of 4.23%.

For the dataset proposed by this chapter, we summarize the results for the three experiments in the Table 6.2. For the first experiment, we observed that SURF + SVM achieves the best ACC result with 83.19 %, and L-SMELL reaches the second-best value with 80.80 %. For the PREC metric, we observed that the first and second-best results corresponding to the proposals SURF + SVM and L-SMELL obtained 87.18 % and 81.12 %. In the analysis of REC, SURF + SVM has the best results with 80.16 %. The second-best position is the L-SMELL that reaches to REC metric of 79.41 %, with a difference of 0.75 % for the first result. In the last evaluation, SURF + SVM achieves the best performance for the F1-Score metric, while L-SMELL obtains the second-best result.

In the second experiment, SURF + SVM reaches the best value for the ACC metric with 80.91 %, while the proposal SIFT + SVM achieves the second-best result with 76.66 %. However, for the other three metrics analyzed, we noticed that the SIFT + SVM has a low performance. Probably, it happens in this experiment because the dataset is unbalanced. For all three remaining metrics, we identified that SURF + SVM had the best result. We see that the CosFaceLoss technique achieved the second-best result for the PREC and F1-Score metrics, while the MultiSimilarLoss proposal achieves the second-best results for REC.

For the last experiment for this dataset, we investigated the unbalanced version

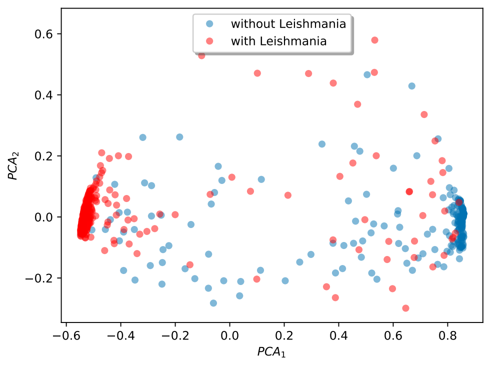
Table 6.2. Table showing the results for the analyzed dataset. We represent the best values in **bold** and * represents the values without a statistically significant difference.

Experiment I				
Proposal	ACC	PREC	REC	F1-Score
SURF + SVM	***83.19% (3.97)%**	***87.18% (3.93%)**	***80.16% (4.45%)**	***81.25% (4.70%)**
SIFT+SVM	58.57% (0.14%)	29.29% (0.07%)	53.21% (0.35%)	36.94% (0.06%)
DeCaf [25]	58.82% (0.33%)	29.41% (0.17%)	51.23% (0.12%)	37.03% (0.13%)
NPairLoss [86]	72.27% (4.88%)	71.82% (5.20%)	70.31% (5.25%)	70.56% (5.21%)
Triplet [80]	74.12% (6.51%)	70.63% (4.33%)	73.99% (3.54%)	72.11% (4.71%)
CosFaceLoss [97]	73.43% (6.20%)	72.71% (6.61%)	71.94% (6.99%)	72.04% (6.90%)
MultiSimilarLoss [101]	75.39% (2.86%)	75.82% (3.21%)	74.19% (3.04%)	74.65% (3.02%)
L-SMELL	*80.80% (5.01%)	*81.12% (5.49%)	*79.41% (5.45%)	*79.71% (5.44%)
Experiment II				
Proposal	ACC	PREC	REC	F1-Score
SURF + SVM	***80.91% (2.16%)**	***77.34% (2.17%)**	***84.04% (2.75%)**	***78.48% (2.39%)**
SIFT+SVM	76.66% (0.32%)	38.33% (0.32%)	50.00% (0.00%)	43.39% (0.21%)
DeCaf [25]	74.13% (0.15%)	37.06% (0.08%)	50.00% (0.00%)	42.57% (0.05%)
NPairLoss [86]	67.58% (2.73%)	62.04% (3.56%)	64.55% (4.77%)	62.25% (3.67%)
Triplet [80]	71.62% (3.15%)	61.05% (1.23%)	65.32% (6.52%)	64.21% (4.16%)
CosFaceLoss [97]	74.51% (9.36%)	70.73% (10.16%)	71.84% (9.55%)	71.04% (10.24%)
MultiSimilarLoss [101]	71.99% (1.89%)	69.57% (2.77%)	73.75% (2.39%)	69.72% (2.36%)
L-SMELL	76.31% (4.79%)	69.21% (6.25%)	67.83% (6.36%)	68.16% (6.09%)
Experiment III				
Proposal	ACC	PREC	REC	F1-Score
SURF + SVM	***91.88% (0.04%)**	45.94% (0.01%)	50.00% (0.00%)	47.88% (0.01%)
SIFT+SVM	***91.88% (0.02%)**	45.94% (0.02%)	50.00% (0.00%)	47.88% (0.01%)
DeCaf [25]	***91.88% (0.02%)**	45.94% (0.02%)	50.00% (0.00%)	47.88% (0.01%)
NPairLoss [86]	86.81% (3.85%)	*63.75% (7.58%)	*70.61% (9.97%)	*65.95% (8.72%)
Triplet [80]	85.61% (0.44%)	*67.81% (2.31%)	64.70% (3.42%)	*66.05% (1.82%)
CosFaceLoss [97]	90.31% (0.60%)	*68.06% (1.89%)	62.46% (3.69%)	*64.62% (2.65%)
MultiSimilarLoss [101]	89.81% (2.75%)	*62.71% (6.87%)	***72.46% (8.31%)**	*67.38% (7.18%)
L-SMELL	90.46% (2.20%)	***68.63% (11.61%)**	*68.39% (12.22%)	***68.49% (12.52%)**

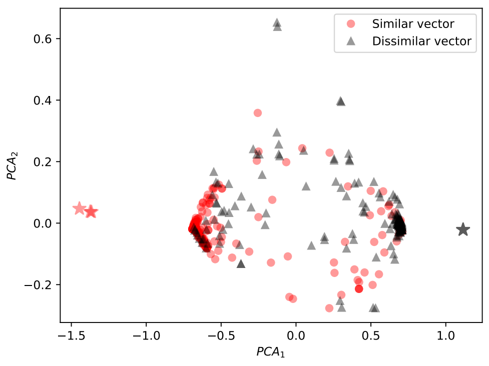
of our dataset for the approaches used in this chapter. We verified the ACC metric, we decide SURF + SVM, SIFT + SVM, and DeCaf was the best results. However, observing other metrics we concluded the performance of these proposals was affected by the unbalanced dataset. We noticed L-SMELL gets the best results when we analyze PREC, while CosFaceLoss begets the second-best position. MultiSimilarLoss and NpairLoss were first and second-best results for the REC metric. For F1-Score, L-SMELL obtains the best result with 68.49 %, while MultiSimilarLoss gets the second-best results.

6.4.2 Latent Space and Similarity space

We investigate the behavior of L-SMELL using the dataset proposed in this chapter. For better visualization, we use PCA as a dimensionality reducer for 2-D as can see in Figure 6.4. The result of the encoder output (*latent feature space*) can be see in Figure 6.4a shows . Each dot represents a single object, red dots represent samples with Leishmania parasite, and the blue dots represent samples without the parasite. The joint optimization process of autoencoder and S-space generated this *latent feature space*.



(a) Latent space ($n = 128$)



(b) S-space ($n = 128$)

Figure 6.4. Latent Space and S-space Analysis for Our Dataset in **Experiment I**.

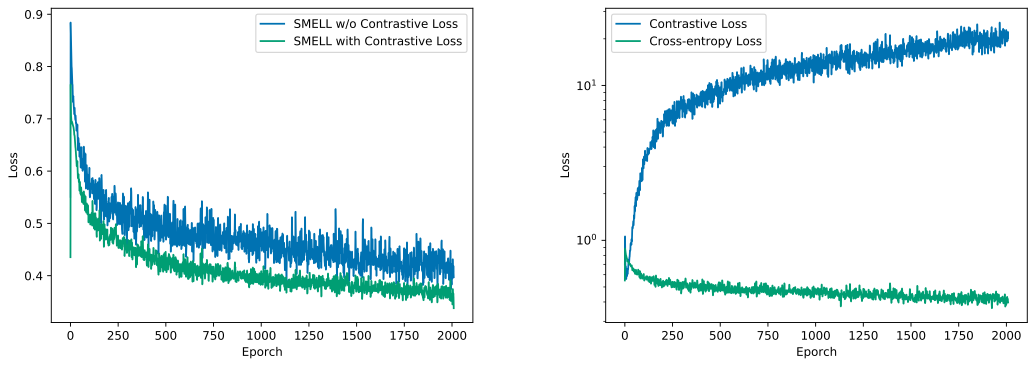
We realized that training process used by L-SMELL is able to separate the latent space into two disjunct regions, thus making the inferential process for identifying the parasite in samples.

In addition to *latent feature space*, our proposal estimates a new representation space called S-space. We can see S-space in Figure 6.4b. Note that each point represents a pair of elements, as we can see in Section 3.2.1. The pairs of similar images correspond to red circles, while the black triangles represent dissimilar image pairs. In this space, we estimate markers as we can see in Figure 6.4b. The red stars represent the positive markers $\mathcal{M}^+$, while the points represented by the black stars are the markers of dissimilarities $\mathcal{M}^-$. Both spaces are estimated simultaneously by our proposal.

In S-space, shown in Figure 6.4b, we can see that L-SMELL also estimates two disjunct regions. However, we can notice that the markers act as attractive components, e.g., the similarity markers in $\mathcal{M}^+$ attract similar points, while the dissimilarity markers in $\mathcal{M}^-$ attract the dissimilar points. We use three positive markers and two negative markers, but our proposal collapse some markers not used. This behavior indicates that the number of markers is a virtual hyper-parameter of the model.

6.4.3 Contrastive loss effect

As described in section 6.3.2, we use the cross-entropy loss for training L-SMELL. Also, we use the contrastive loss as a regularization function to avoid overfitting. We can see in Figure 6.5a a comparison of the cross-entropy function for our proposal when we train L-SMELL with the regularization function and L-SMELL without the regularization function. For the case in which we do not use contrastive loss, the loss function has a



(a) Comparison of L-SMELL versions with and without Contrastive Loss

(b) Cross entropy and Contrastive loss for L-SMELL training

Figure 6.5. Loss analysis for our propose.

greater dispersion of the loss values and a higher average loss value.

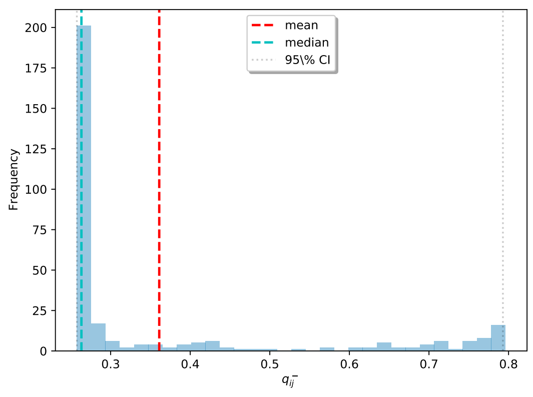
In Figure 6.5b, we observe separately the behavior of the two components that define our loss, i.e., R_c and H_c. We noticed that, in the first moment, the value of H_c has an upward behavior, while for R_c we see a drop in value. However, in the training of L-SMELL, H_c has value reduced, while R_c has a growth trend.

Our proposal optimizes two spaces simultaneously (*latent feature space* and S-space). So we believe that the model does not use any prior information, it has an extensive search space solution, and therefore, it has some difficulty to find an effective solution for both representations. With the addition of Contrastive loss as a regularization function, we observed a tendency for L-SMELL to estimate a *latent feature space* in than similar points are close together. However, as we can see in Figure 3.3, our proposal can build more complex latent space, so as L-SMELL finds effective solutions, the contrastive loss becomes less relevant (and thus its value increases), the measure that these solutions are estimating.

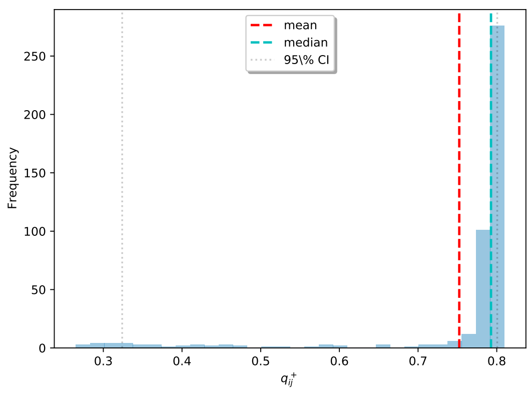
6.4.4 Metric learning visualization

To investigate the behavior of the proposal, we performed a qualitative assessment on L-SMELL. Thus, we select one image for each class of dataset proposed in this chapter. We compared each selected image with all other images used in experiment 1 with our dataset. This evaluation can be seen in Figure 6.6 and 6.7.

In Figure 6.6, we randomly select an image that contains the parasite that causes leishmaniasis and perform two comparisons. In the first comparison, we verified the

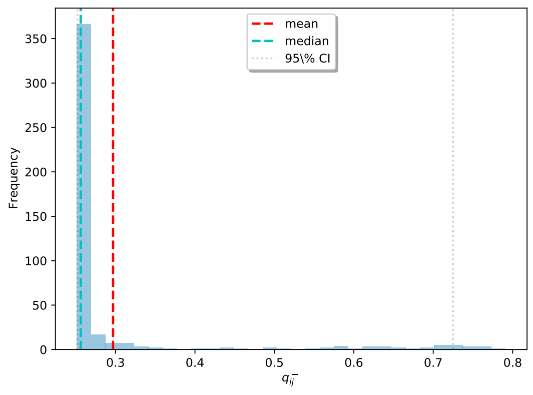


(a) Histogram for q_i^-

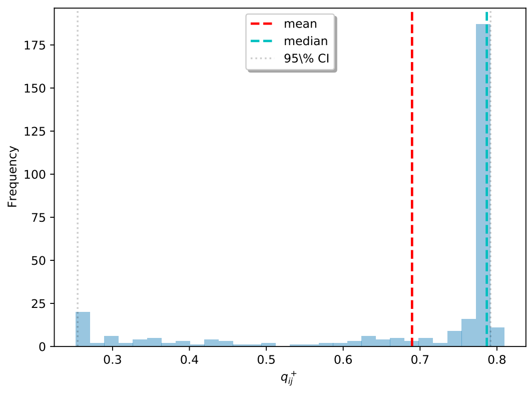


(b) Histogram for q_i^+

Figure 6.6. Quantitative evaluation of our proposal given a reference image with the parasite that causes Leishmaniasis.



(a) Histogram for q_i^-



(b) Histogram for q_i^+

Figure 6.7. Quantitative evaluation of our proposal given a reference image without the parasite that causes Leishmaniasis.

similarity of the selected sample with all images used in experiment 1 that also have the parasite, as we can see in Figure 6.6b. Finally, we compared all dissimilar images to the selected image, i.e., that do not have the parasite caused by leishmaniasis, resulting in the histogram shown in Figure 6.6a.

In Figure 6.6a, we see the distribution of q^- for the image with parasite. We verified that this histogram has a median equal to 0.26 and an average with a value of 0.3621 denote the behavior of heavy tail to the right. Analogously, Figure 6.6b shows the histogram of q^+ for the pairs of similar images. We observe heavy tail behavior with mean and median equal to 0.7508 and 0.79242. The heavy tail indicates that our proposal captures the context of dissimilarity/similarity. This behavior shows that most of the points have a high dissimilarity/similarity value compare to our reference image.

In Figure 6.7, we randomly select an image that does not contain the Leishmania parasite and compare it with all the remaining images in our dataset during experiment 1. Thus, we present in Figure 6.7a, the distribution of q_j^-. We found a heavy tail behavior, with a median value around 0.2, i.e., our metric obtained a large count of dissimilarity values. Also, there are few points with high similarity.

In Figure 6.6b, we observe the histogram for all images similar. We verified a heavy tail behavior, a median value close to 0.8. This result is similar to the previous experiments, indicating the good performance of our proposal.

With this quantitative experiment, we verified that our proposal can capture the context of similarities to which the data are inserted, thus showing that L-SMELL is a promising candidate to detect the Leishmaniasis parasite.

6.5 Conclusion

In this chapter, we propose L-SMELL. We consider that our proposals can simultaneously learn *latent feature space* and the similarity space.

We hypothesized that due to the complexity of the estimation of *latent feature space* and *S-space*, It is difficult for L-SMELL to estimate an effective solution. So, with the addition of contrastive loss as a regularization function, we were able to tend to search the feature spaces. However, our proposal can build more complex latent space, so as L-SMELL finds effective solutions, the contrastive loss becomes less relevant (and thus its value increases), the measure that these solutions are estimating.

We conclude, with these experiments, if a dataset becomes unbalanced, then proposals performance is degraded. However, the proposals based on Deep Metric Learning had a less evident degradation if we compared them with other techniques used in this chapter. Unlike classic literature proposals, which use information from a single element to perform the inference, the deep metric learning approaches consider mutual information obtained by pairs of elements, thus achieving a greater diversity of samples to be used in training, and therefore, better performance in the classification.

Chapter 7

Case of Study: Malware Classification

7.1 Introduction

Cyberattacks are today one of the main concerns in the realm of computer networks. Many attacks ranging from naive viruses to ransomware and then to sophisticated malware like Stuxnet[48] jeopardize individual users' privacy/safety and endanger entire nations' sovereignty. In fact, cyberattacks are so relevant nowadays that president Biden said during his inauguration that his administration would make cybersecurity a top priority [1].

The detection of malicious software that "deliberately fulfills its intention to harm a system"–or *malware* for short [71]– is one of the most relevant security problems. For instance, the incidence of a single malicious software can cause up to millions of dollars in damage [5].

Per Raff et al. [73], some of the biggest challenges in malware detection are:

– Software bytes can indicate a wide range of information. The meaning of any particular byte is context-sensitive and can be encoded by human-readable text (for example, function names from the import table), binary code, among others.

– Software codes, and consequently bytes of binary code, exhibit several types of spatial correlation. These correlations, in turn, have discontinuities in function calls and jump commands.

– By treating each byte as a unit in a sequence, we are dealing with a classification problem of an order of two million steps. In this sense, it is impracticable to use a naive sequential neural network for this task.

Antivirus is perhaps the most popular approach for malware detection. Traditionally, those products employ signature-based strategies for malware detection. Broadly,

[1]https://www.voanews.com/usa/us-biden-voice-new-alarm-about-cyberattack

whenever a suspect software is reported, it goes through an analysis process. If the software is regarded as malware, then a unique signature is generated and then added to the antivirus product database [7, 106].

Unfortunately, however, malwares also have their ways of tricking antivirus and security countermeasures alike. For example, malwares can employ encryption to disguise their real signature and then not decrease the effectiveness of antivirus [88]. One of the reasons antivirus misclassifies malware is that they mainly focus on each malware file individually instead of considering cross-files information.

As a solution to this problem, we came up with Malware-SMELL, a novel similarity function meant to detect malware. Malware-SMELL gets two new data representation spaces: *latent feature space* and *similarity space*.

Malware-SMELL receives a pair of objects associated with a similarity label, i.e., if a pair of objects have the same class, they are called similar. Similarly, we define a pair of objects to be dissimilar, if they have different classes. Therefore, for each input pair, Malware-SMELL estimates a new normalized feature vector using an encoder function. This new representation is called a *latent feature space*.

Also, Malware-SMELL uses an auxiliary representation, called the similarity space (or S-Space), which helps in the construction of *latent feature space*. In S-space, we define similarity and dissimilarity markers to group the vectors into their respective representations (similar or dissimilar). Note that, for an input pair, we have two representations in the *latent feature space*, while that same pair is represented by a single vector in S-Space.

In training, Malware-SMELL simultaneously estimates *latent feature space* and S-Space. S-Space consists of disjunct regions, forming clusters referring to similar and dissimilar data pairs. We estimate the similarity through a T-student distribution between the new representations of the data in S-Space and the markers found by our approach. As we perform data normalization in *latent feature space*, the Euclidean distance is equivalent to Pearson's Correlation [50].

Overall, the main contributions of this chapter are:

– We propose a new approach to malware detection based on image representation of files extracted from binary code. Through the spaces estimated by Malware-SMELL (latent feature space and S-space), our proposal estimates a new similarity function.

– Due to the S-space estimate, our proposal can classify new unknown families of malware that had not been used in training (Zero-shot learning).

– Due to our pair selection strategy to model training, Malware-SMELL can make inferences on unbalanced data sets.

– Due to the restriction of normality of vectors in *latent feature space* (norm equal to 1), Euclidean distance in this space corresponds to Pearson's Correlation. Therefore, grouping a set of similar pairs corresponds to identifying a new representation of the data that have sections with high Pearson's correlation between them.

We have performed a series of experiments over the analyzed dataset and shown that Malware-SMELL outperforms baseline methods by a ratio of 0.56 %. In addition, if we consider the classification problem in the context of Zero-shot learning, Malware-SMELL outperforms baseline methods by a ratio of 29.09 %.

7.2 Related Work

To obtain efficiency and robustness malware analysis techniques study the behavior and structure of executables, extracting resources that can describe their malicious intent. Traditionally, the methods are classified based on the static and dynamic analysis of the program files and by the feature extraction approach used.

The dynamic analysis consists of an automated classification system based on heuristics that examine the sample execution behavior in its network activities, reading, and writing operations in a restricted area environment. Although a dynamic analysis component is likely to be important for a long-term solution, such an approach is still slow, requires high computational power, and requires a controlled environment to ensure good results.

On the other hand, static analysis detects an executable file without executing code execution in real-time. Using signature-based methods, it identifies patterns of strings extracted through observations of the binary code or the internal file structure, so it requires unzipping the program before doing the analysis [79].

Among the set of static malware analysis, image-based approaches are gaining popularity due to their ease of use and the existence of infrastructure for synthetic images [28, 62]. In such methods, executable files are converted into grayscale images, making it possible to use traditional image processing methods. Once the construction of the malware image from the executable file is able to maintain information from the program's code, text and resources sections, promising results have been acquired [61].

Many methods have calculated the similarity between different representations of the malware images to perform the classification. Lim et al. [34] generated arrays of RGB images of Opcode strings extracted from the malware and calculated similarities between the arrays using simHash. His experimental results showed that his method ranks effectively with 98.96% accuracy. Han et al. [35] converted executable files into

Table 7.1. Main articles that used a visual representation of malware to infer the binary condition. In addition, we verify which articles use an approach to imbalanced dataset (I. D.) and Zero-shot learning approaches (Z. S. L.)

Paper	Approach	Taxonomy	I. D.	Z. S. L.	year
[45]	Garbor + Wavelet	Features extraction	✗	✗	2013
[111]	SURF + LHS	Features extraction	✗	✗	2014
[47]	CMYK + RF	Features extraction	✗	✗	2016
[2]	CNN + SVM	NN	✗	✗	2017
[43]	Hamming Distance + CNN	Features extraction + NN	✗	✗	2018
[19]	Subsampling + CNN	NN	✓	✗	2018
[1]	GMM	Features extract	✗	✗	2019
[8]	SOINN	NN	✗	✗	2019
[118]	SPP + CNN	NN	✗	✗	2019
[17]	Gabor + RF	Features extraction	✗	✗	2019
[85]	CNN	NN	✗	✗	2019
[76]	CNN features + SVM	NN	✗	✗	2019
[94]	CNN + LSTM	NN	✗	✗	2019
[72]	CNN + SVM	NN	✗	✗	2019
[114]	CNN + Attention	NN	✗	✗	2019
[30]	ResNeXt	NN	✗	✗	2020
[117]	RCNN	NN	✗	✗	2020
[119]	R-CNN	NN	✗	✗	2020
[77]	gcForest	Deep Learning	✗	✗	2020
[93]	Enseble CNN	NN	✗	✗	2020
[121]	CapsNet	NN	✓	✗	2021
[10]	UMAP	Manifold Learning	✗	✓	2021
[92]	CNN + Finetuning	NN	✓	✗	2021
Our approach	Malware-SMELL	Deep metric learning	✓	✓	2021

grayscale images and introduced a similarity technique based on entropy graphs. His experimental results showed that his method achieves a 97.9% similarity rate.

The analysis of binary texture resources using local and global descriptors also proved to be an alternative for the classification of malware. Ban et al. [111] extracted local features from the images using the SURF (Speeded up robust feature) descriptor and performed the correspondence using LSH (Local sensitive hashing). Such a method achieved 85% accuracy in the ranking task.

Applying a global features analysis approach, Lakshman et al. [63] and Aziz et al. [55] proposed the use of the GIST descriptor and the K-nearest neighbors (KNN) algorithm for classification. His experimental results indicated that his technique achieves 96% accuracy. Kumar et al. [47] analyze images based on malware with different color representation systems. The authors use different color systems (grayscale, RGB, CMYK and HSL) and extract features using GIST to perform classification with Random Forest.

Following a different path, Barath et al. [61] introduced a new technique of static visual analysis for extracting global resources from images in the Microsoft Malware Classification Challenge (BIG 2015) dataset, using PCA. Three algorithms, namely, support vector machine (SVM), and K-nearest neighbors (KNN) are applied to classify malware based on PCA resources. Such a method achieved 96.6 % accuracy with KNN.

Deep learning has been applied in various fields of knowledge such as speech recognition, image classification and dimension reduction as a powerful feature extract tool. Neural networks (NN) seek to learn a non-linear feature representation, they usually overperform standard features description (e.g. GIST, SURF, PCA) found in the literature.

Roseline et al. [77] use a Deep Forest (Forest) to classify malware. This method generates a deep forest ensemble, with a cascade structure which enables gcForest to representation learning. Its representational learning ability can be further enhanced by multi-grained scanning when the inputs are with high dimensionality, potentially enabling gcForest to be contextual or structural aware.

We see in Singh et al. [85], the authors use a convolutional neural network to classify malware files. The authors apply a color map and based on the new visual representation, they obtained 96.08 % accuracy for their dataset. In Go et al. [30], the authors use a neural network based on ResNeXt to classify images based on malware.

Vasan et al. [93] propose an ensemble formed with convolutional neural networks for malware detection. Thus, for each neural network architecture, the authors apply the PCA to the features found by the convolutional layers, and with this new representation, they use a classifier obtaining 98.11% accuracy.

Yakura et al. [114] propose the use of an attention mechanism to calculate the attention map, which is expected to specify regions having higher importance for classification. This distinction of regions enables the extraction of characteristic byte sequences peculiar to the malware family without prior knowledge.

We also see that several studies use convolutional layers (CNN's) associated with different classification approaches. Vinayakumar et al. [94] proposes a neural network architecture that combines convolutional (CNN) and recurrent (LSTM) layers to identify malware based on visual representation.

The authors in [117] propose a new neural network architecture called recurrent convolutional neural network (RCNN). In this architecture, the authors construct some stages of CNN (feature extraction and max polling) recursively. In the recurring stage, the authors resize the images in order to identify different representations of the same object. With RCNN, the authors identify malware through the visual representation (grayscale) obtained through the binary file, obtaining 91.64 % accuracy based on Exploit kits.

The authors in [119] use a Region-Convolutional Neural Networks (R-CNN) to classify malware. R-CNN was initially proposed for the identification of regions and, therefore, for object detection. The authors used a trained architecture and performed transfer learning for the target dataset (which contains the visual representations of the malware). The authors obtained 92.8 % accuracy for a set of JPEG images infected with malware.

In [2, 72, 76], the authors use the features extracted from the convolution layers and are used by the SVM algorithm to classify malware. Note that [2, 72] SVM are designed specifically for the problem and we can train CNN and SVM simultaneously whereas Rezende et al. [76] use the features found in CNN training and, this new representation was used with an SVM classifier.

We noticed that the datasets referring to malware have imbalanced data (I. D.). Thus, techniques that consider this type of problem are interesting for the task of classifying malware.

In [121], the authors propose to use a Capsule Neural Network (CapsNet) for malware classification. CapsNet is a Neural Networks that can be used to better model hierarchical relationships. The idea is to add structures called "capsules" to a convolutional neural network (CNN), and to reuse output from several of those capsules to form more stable (with respect to various perturbations) representations for higher capsules [78] and imbalance datasets inference.

In another work, Cui et al. [19] propose to use a convolutional neural network to identify malware based on its visual representation, but differently from some approaches, a sub-sample of the dataset was performed for each training round of the algorithm for mitigate imbalance class problem.

Approaches that can classify unknown malware (that were not in the training set), are very promising in this type of task. Zero-shot learning (Z. S. L.) is a problem setup in machine learning, where at test time, a learner observes samples from classes that were not observed during training.

The authors in [10] propose the use of a manifold learning and dimension reduction technique called UMAP for the problem domain and evaluated its contribution to unknown malware detection problem (zero short learning).

The methods mentioned in this section indicate the feasibility of learning to classify malware from the input data using a visual representation. Table 7.1 summarizes the malware classification works using a visual representation.

However, we found that literature proposals typically use descriptors to capture information from a single object. Therefore, we developed a new method that captures pairwise information from data pairs. Thus, we propose a new measure of similarity designed for the context of malware identification. To the best of our knowledgment, this chapter is the first deep metric learning approach in the context of identifying malware through a visual representation.

7.3 Malware-SMELL

As defined in Section 3.2, we use, in Malware-SMELL, a *latent feature space* and S-space, but we propose a new version of SMELL for image classification. Note that, unlike SMELL, in this chapter we normalize the vector in *latent feature space* ($||f_\Theta(x_i)||_2 = 1$). We describe the loss function for this version of SMELL.

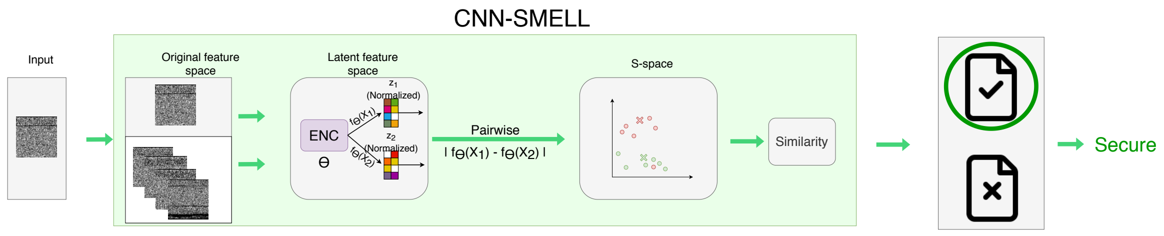


Figure 7.1. The left side represents the encoder function for a image pair in original feature space. At the end of this propose, we decide if image has leishmaniasis.

7.3.1 Loss Function

Our proposal conducts the simultaneous training of the set $\mathcal{M}$, and *latent feature space* with parameters Θ and Θ' for the encoder and decoder functions, respectively.

We defined the cost function

$$J(\{\mathcal{X} \times \mathcal{X}\}) = H_c(\mathcal{U}||\mathcal{Q}) + R_c + R_r, \tag{7.1}$$

where H_c is the cross-entropy loss, R_r is reconstruction function and R_c is a regularization function to avoid overfitting in the training process.

We can define the contrastive loss as a function that minimizes the distance between similar points and imposes a restrictive distance between the points of dissimilar classes [33]. So, we can define the contrastive loss for a pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as

$$R_c(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} r_c||f_\Theta(\boldsymbol{x}_i) - f_\Theta(\boldsymbol{x}_j)||_2^2, \text{ if } \ell(\boldsymbol{x}_i) = \ell(\boldsymbol{x}_j) \\ r_c[p - ||f_\Theta(\boldsymbol{x}_i) - f_\Theta(\boldsymbol{x}_j)||_2]_+^2, \text{ on the contrary,} \end{cases} \tag{7.2}$$

where r_c is a calibration term, and operator $[.] = max(0, .)$ is the hinge function. In this chapter, following [33] we use the margin $p = 1$.

Note than R_c function works only in the *latent feature space*, minimizing the Euclidean distance between similar points. The H_c function acts in the *latent feature space* and the *S-space* simultaneously, grouping the elements $\boldsymbol{s_{ij}}$ around the respective markers.

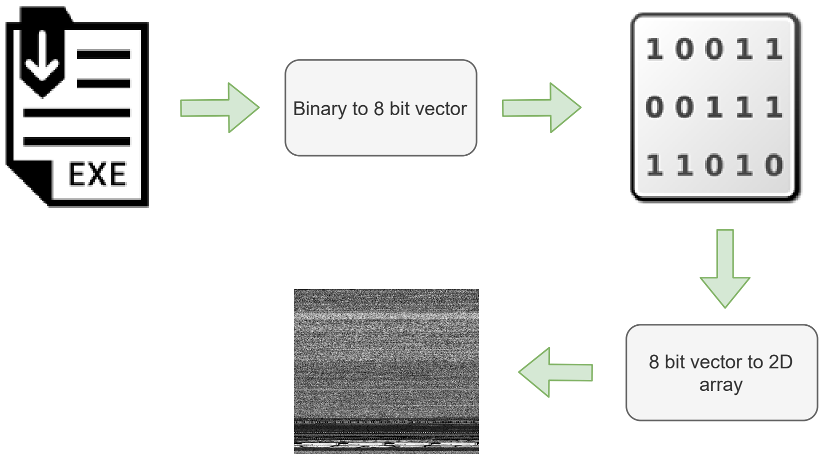


Figure 7.2. Schemer for malware dataset creation

7.4 Methodology

7.4.1 Dataset

To analyze the performance of Malware-SMELL we used the Malimg Dataset [62], developed by the University of California's vision research laboratory [2]. This data set contains 9339 samples from 25 malware families, obtained through experiments of mixtures of network and the Windows operating system malware.

For the generation of the dataset, each malware binary data, which consists of an 8-bit unsigned integer vector, is organized in a two-dimensional matrix resized to 64 x 64 dimension and viewed as a grayscale image, as can see in Figure 7.2. Thus, sample variants belonging to the same malware family will have similar visual and textural characteristics. Details about the dataset can be found in Table 7.2.

7.4.2 Network evaluation

To evaluate our approach, we apply a *K-Nearest Neighbor* classifier, with three neighbors, in agreement with [9]. The KNN classification performance can often be significantly improved through (supervised) metric learning. In addition, we use KNN with the Euclidean distance to compare with our proposal, e.g., sanity check.

[2]https://vision.ece.ucsb.edu/research/signal-processingmalware-analysis

Table 7.2. Description of the malware classes and families present in the Mailing dataset

Class	Family	# of samples
Backdoor	Agent.FYI	116
Backdoor	Rbot!gen	158
Dialer	Adialer.C	125
Dialer	Dialplatform.B	177
Dialer	Instantaccess	431
PWS	Lolyda.AA 1	213
PWS	Lolyda.AA 2	184
PWS	Lolyda.AA 3	123
PWS	Lolyda.AT	159
Rogue	Fakerean	381
TDownloader	Dontovo.A	162
TDownloader	Obfuscator.AD	142
TDownloader	Swizzot.gen!I	132
TDownloader	Swizzot.gen!E	128
TDownloader	Wintrim.BX	97
Trojan	Alueron.gen!J	198
Trojan	C2Lop.gen!g	200
Trojan	C2Lop.P	146
Trojan	Malex.gen!J	136
Trojan	Skintrim.N	80
Worm	Allaple.L	1591
Worm	Allaple.A	2949
Worm	VB.AT	408
Worm	Yuner.A	800
Worm:AutoIT	Autorun.K	106

We used 10-fold cross-validation. This validation can largely retain heterogeneous distributions in the training set and improve statistical confidence in the results.

For evaluation, we calculate average accuracy (ACC), average precision (PREC), average recall (REC), and average F1-Score (F1-Score).

These evaluation metrics have been extensively used in research community to provide detailed assessments of methods.

So, we can define **True Positive** (TP): means correct detection of a benign; **True Negative** (TN): means correct identification of malware; **False Positive** (FP): means false identification of malware as a benign application; **False Negative** (FN): means false identification of a benign file as malware, and

– Accuracy is defined as the ratio of correctly predicted outcomes to the sum of all predictions, as follows:

$$ACC = \frac{TN + TP}{TN + FN + TP + FP}.$$

– Precision determines if the positive predictions of the model are correct and is calculated by dividing the sum of true positives by all positive predictions as:

$$PREC = \frac{TP}{FP + TP}.$$

– Recall is the positives identified by the model among all possible positives and is obtained by dividing true positives by the sum of actual positives as:

$$REC = \frac{TP}{TP + FN}.$$

– F1 score is the weighted average of recall and precision. Balanced F-score (F1 score) or F-measure is the harmonic mean of recall and precision,

$$F1 - Score = 2 * \frac{PREC * REC}{PREC + REC}.$$

To evaluate Zero-shot learning, we used Recall@K, as seen in [101]. Each test image (query) first retrieves K nearest neighbors from the test set and receives score 1 if an image of the same class is retrieved among the K nearest neighbors and 0 otherwise. Recall@K averages this score over all the images.

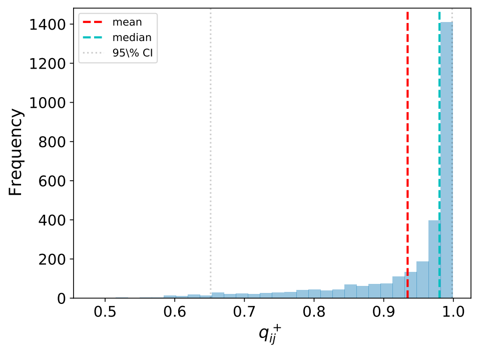
7.4.3 Parameters initialization and network architecture

We initialize all weights of the autoencoder layers following [46]. We initialized Markers position with Lloyd's algorithm [53]. For the encoder, we used the convolutional layers of the VGG 19 architecture as a feature extractor, proposed by [84].

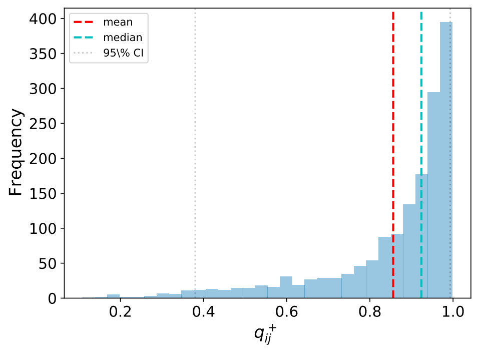
Thus, after the convolutional layer, we designed a feedforward network with three layers of dimensions f-4096-d, where f is the output of the convolutional feature extractor, and d is the *latent feature space* representation. For this chapter, we used d = 256. All layers are fully connected, and we used as activation function the Rectified Linear Unit(ReLU).

Besides, we used mini-batch Stochastic Gradient Descent (SGD) where the learning rate is 0.01, and momentum is 0.9.

Since the optimization model depends on some hyperparameters (r_c, w, k), we performed an investigation to determine which value of these variables would maximize the model accuracy.



(a) Similarity frequency distribution for *Allaple.A* class.



(b) Similarity frequency distribution for *Allaple.L* class.

Figure 7.3. Two analysis of the similarity of Malware-SMELL a random choice.

We used a grid search approach for hiperparameters optimization, where we found $r_c = 10^{-1}$, similarity markers $k = 3$, dissimilarity markers $w - k = 2$, these values were used in the rest of this chapter.

7.5 Results and Discussion

7.5.1 Metric Learning

To qualitatively evaluate the performance of our proposal, we investigated the behavior of the similarity function found by Malware-SMELL. Thus, we randomly select an element x and calculate the similarity with Malware-SMELL for all similar pairs (x, x_j), i.e., x has the same label as x_j. Among the classes with the largest number of elements, we selected a unique object for each evaluation, as we can see in Figure 7.3.

Initially, for the class *Allaple.A*, we can see in Figure 7.3a similarity distribution for q^+. We can see that the distribution is asymmetrical, with a heavy tail on the left. The median (50 % of the analyzed values) is 0.982, indicating that our metric had a satisfactory result since all the analyzed pairs are similar. This behavior indicates that most points have high similarity. Also, we see that the average has a 0.936, even in a heavy tail distribution, which tends to decrease the value of the average considerably.

Looking at Figure 7.3b, we see similar behavior for an element of the class *Allaple.L*. The median has a 0.93, indicating the behavior of a heavy tail.

We observed the similarity space formed by Malware-SMELL. We selected a ran-

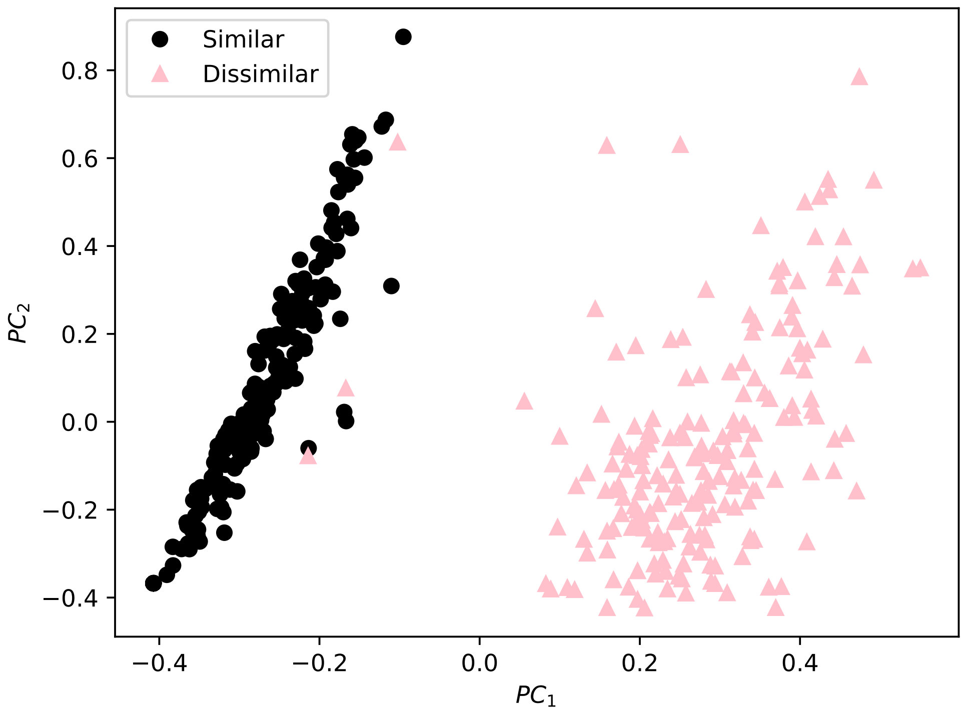


Figure 7.4. The similarity space for our proposal. We randomly selected 400 pairs of data (200 similar and 200 dissimilar).

dom sample of 400 pairs (200 similar and 200 dissimilar). As defined in section 3.2, a pair of objects represents a single point in S-space. Pink circles and black triangles represent dissimilar and similar labels, respectively.

We see that S-space forms two well-defined disjoint regions. This behavior indicates that the model obtained a good estimate of similarity.

7.5.2 Comparative results

We compared our proposal with 12 standard techniques found in literature and a sanity check with the usual KNN for Euclidean distance. We use the metrics described in Section 7.4.2. The results were reported in Table 7.3.

For the sanity check, we noticed a significant difference between our proposal and standard KNN equal 47.70% for the F1-Score metric.

Vinayakumar et al. [94] achieves the best average accuracy result when compared with other techniques, resulting in 98.59%. The second and third best results were obtained by Malware-SMELL and Roseline et al. [77] with 97.76% and 97.21% respectively,

Table 7.3. Table showing the results for the analyzed dataset. The best values are represented in **bold** and * represents the values without a statistically significant difference.

Proposal	ACC	PREC	REC	F1-Score
SURF + SVM	31.58% (0.02%)	1.26% (0.01%)	3.99% (0.01%)	1.92% (0.01%)
KNN (Euclidian)	39.86% (0.07%)	64.62% (0.09%)	39.85% (0.07%)	41.99% (0.09%)
GLCM + SVM [19]	62.96% (0.81%)	56.04% (1.51%)	62.96% (0.81%)	53.64% (1.09%)
BAT [19]	69.15 % (43.39 %)	66.45 % (43.43 %)	67.80 % (41.78 %)	66.42 % (42.35 %)
Garbor + RF [17]	80.89% (0.79%)	78.74% (0.72%)	80.89% (0.79%)	78.49% (0.80%)
HOG + SVM	89.09% (0.84%)	88.28% (0.93%)	89.09% (0.84%)	88.23% (0.91%)
GIST + SVM [76]	92.20% (1.35%)	92.50% (2.57%)	91.40% (3.57%)	92.41% (3.20%)
CNN + SVM [3]	94.12% (5.55 %)	88.86 % (8.69 %)	88.82 % (8.62 %)	88.44 % (9.16 %)
VGG 19 [85]	95.15% (0.58%)	89.84% (1.25%)	90.23% (1.15%)	89.81% (1.23%)
Singh et al. [85]	95.57 % (4.39 %)	95.89 % (3.58 %)	95.56 % (4.39 %)	95.21 % (4.91 %)
UMPA [10]	96.62 % (0.33 %)	95.26 % (0.62 %)	94.29 % (0.81 %)	94.54 % (0.74 %)
Roseline et al. [77]	97.21 % (0.38 %)	97.23 % (0.38 %)	97.21 % (0.37 %)	97.14 % (0.38 %)
Vinayakumar et al. [94]	*__98.59 % (0.26 %)__	96.57 % (0.76 %)	96.33 % (0.74 %)	96.33 % (0.74 %)
Malware-SMELL	97.76% (0.49%)	*__97.84% (0.52%)__	*__97.76% (0.49%)__	*__97.69% (0.51%)__

representing 0.83% and 1.36% difference for Vinayakumar et al. [94]. However, when we analyze the other metrics, this difference becomes more evident. Probably this behavior occurs because the dataset is unbalanced, with the sum of the two major classes equal to 4540 (representing 48.61% of the dataset). Therefore, other metrics are needed to evaluate the performance of the techniques.

Analyzing the precision, we see that Malware-SMELL achieves 97.84%. The second-best result obtained 97.23% with Roseline et al. [77] technique, thus totaling 0.61% difference. This difference becomes greater when compared to Vinayakumar et al. [94] (third-best result), equal 1.27%.

Observing the recall, we see that the second and third-best performance in the analyzed dataset is obtained by the proposals Roseline et al. [77] and Vinayakumar et al. [94], with 97.21% and 96.33% respectively. When compared to Malware-SMELL, which obtained the best performance with 97.76%, we noticed a difference of 0.36% and 1.43%.

Finally, we evaluated the performance of the proposals using the F1-score metric (harmonic average of the precision and recall). Malware-SMELL again obtained the best result of 97.69%, when compared with the second and third best approach 97.14% and 96.33% obtained by Roseline et al. [77] and Vinayakumar et al. [94], respectively.

We observed that Malware-SMELL managed to be the best technique among all analyzed. In addition, we performed the t-student test for static comparison between the results. For PREC, REC and F1-Score, the result obtained by Malware-SMELL has a static difference to the other approaches. Figure 7.5 show the latent feature space for Mailing dataset found by Malware-SMELL.

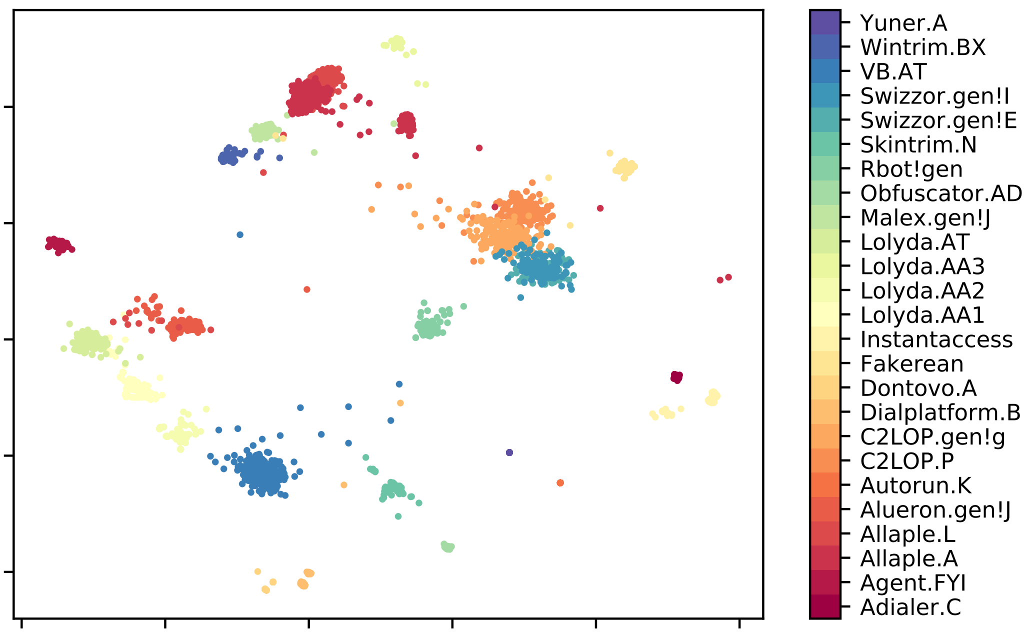


Figure 7.5. *Latent feature space* for malware dataset (n=2)

7.5.3 Zero short learning results

Our proposal estimates two representation spaces: *latent feature space* and S-space. In this way, we were able to estimate similarity between unknown malware, i.e., class of malware that did not appear in training set (Zero-shot learning). Therefore, we investigate the behavior of malware detection proposals that consider Zero-shot learning, as seen in Table 7.1.

In Table 7.4, we evaluate the behavior of Malware-SMELL and UMPA for Malimg Dataset. Initially, we randomly selected 12 classes for training (Adialer.C, Agent.FYI, Allaple.A, Allaple.L, Alueron.gen!J, Autorun.K, C2LOP.P, C2LOP.gen!g, Dialplatform.B, Dontovo.A, Fakerean, Instantaccess) totaling 6579 images. All other 14 classes remaining (Lolyda.AA1, Lolyda.AA2, Lolyda.AA3, Lolyda.AT, Malex.gen!J, Obfuscator.AD, Rbot!gen, Skintrim.N, Swizzor.gen!E, Swizzor.gen!I, VB.AT, Wintrim.BX, Yuner.A) were used for testing totaling 2760 images.

As described in Section 7.4.2, we use Recall@K to evaluate the behavior of Zero-Shot-Learing proposals for different K values. Thus, we adopt $K = \{1, 4, 8, 10, 40, 80\}$. Notice how *Skintrim.N* has only 80 samples, we cannot adopt $K > 80$.

In Table 7.4, we see that for all K values analyzed, Malware-SMELL has the best

Table 7.4. Description of the malware classes and families present in the Mailing dataset

	Recall@K (%)					
Propose	K = 1	K = 4	K = 8	K = 10	K = 40	K = 80
UMPA [10]	82.72 %	76.98 %	76.25 %	72.56 %	63.29 %	53.33 %
Malware-SMELL	**95.47 %**	**94.79 %**	**94.08 %**	**93.32 %**	**89.26 %**	**86.69 %**

performance. In addition, as the K value increases, we realize that the degradation of the result in Malware-SMELL is less than UMPA. For $K = 1$ we have that our proposals reach 95.47 % of Recall@K, while UMPA has 82.72 %, resulting in a difference of 12.75 %. However, as the value increases, i.e., $K = \{4, 8, 10, 40, 80\}$ we have a difference between Malware-SMELL and UMPA of {17.81%, 17.83%, 20.76%, 25.97%, 33.36% } respectively.

7.6 Conclusions and Futures Work

In this chapter, we propose a new metric of similarity to identify malware using visual representation. Our proposal, called Malware-SMELL, finds two representations (*latent feature space* and *similarity space*) that quantify similarity between objects. Due to the fact that Malware-SMELL is the first deep metric learning proposal for identifying malware using image representation, our approach is a promising technique, when compared to techniques found in the literature.

In future works, we will build the space of similarity with the Generative Adversarial Network. We believe that with a generative model, we will be able to improve the metric extracted by our proposal. In addition, we will extend our analysis to more datasets.

Chapter 8

Conclusion

This chapter presents the final thoughts about this dissertation. In Section 8.1, we reinforce the achieved contributions while we provide our view about future research directions that can follow from this dissertation. Section 8.2 shows the list of publications we achieved during this dissertation.

8.1 Conclusions and outlook

In this dissertation, we proposed a Supervised Distance Metric learning Encoder with Similarity Space (SMELL), based on the fact that the distance metrics can be simultaneously learned along with a latent representation of the data and the similarity markers.

It is worth noticing that all four parts play a major role. Firstly, the autoencoder maps data to the latent space. Secondly, the sum of distances assures that markers capture similarities. Thirdly, the cross-entropy loss function assures that markers captures the similar and dissimilar classes. Finally, the repulsive regularizer ensures some level of diversity on the marker set, guaranteeing that markers capture complex similarity regions such as disjoint similarity/dissimilarity regions.

We hypothesized that SMELL groups data points consider similar and increases classes separability. We showed evidences that support our hypothesis by a comprehensive behavior analysis.

Due to the construction of the auxiliary space (S-space), we were able to map the *latent feature space* in the S-space, and, we extracted some theoretical guarantees about the SMELL observing only the position of the markers. In addition, for a particular case of our proposal, we calculate the theoretical misclassification risk function.

We also conducted an extensive validation of our proposal comparing it to many methods over different type of input data. We obtained promising results and, in general context, we got best results. We also carried out a study on the SMELL hyper-parameters.

In the ablation study, we checked several versions of SMELL in order to understand the contribution of each component.

Specifically for the MNIST, image dataset, we were able to carry out a qualitative assessment about SMELL for this dataset. We note that, besides separating the groups, our proposal preserves the semantics of the data.

We intend to investigate the possible applications for this type of approach, as well as to use the Proposition 3.2.2 to build a novel loss function specifically tailored for SMELL.

For chapter 6, we propose, a new method of deep metric learning for Leishmaniasis Parasite Detection in Bone Marrow Smears. In this chapter, we use Contrastive Loss as a regularization function, to improve SMELL convergence.

We hypothesized that due to the complexity of the estimation of *latent feature space* and *S-space*, It is difficult for L-SMELL to estimate an effective solution. So, with the addition of contrastive loss as a regularization function, we were able to tend to search the feature spaces. However, our proposal can build more complex latent space, so as L-SMELL finds effective solutions, the contrastive loss becomes less relevant (and thus its value increases).

For chapter 7, we propose, a new method of deep metric learning to classify malware using visual representation. We compare our proposal to many different approaches. According to the four quantitative metrics used, our method overcomes all baseline strategies from the literature.

We have performed a series of experiments over the analyzed dataset and shown that Malware-SMELL outperforms baseline methods by a ratio of 0.56 %. In addition, if we consider the classification problem in the context of Zero-shot learning, Malware-SMELL outperforms baseline methods by a ratio of 29.09 %. The best of our knowledge, this is the first proposal of deep metric learning in the context of classification of malware with image representation.

8.2 Publication

In the following sections we list the publications achieved during this dissertation. The list is divided into four categories: (i) periodical papers, (ii) conference papers, (iii) under submission, and (iv) short course. Works that start with a (X) mark are related to a direct contribution of this dissertation.

8.2.1 Periodical papers

Barros, P. H., Cardoso-Pereira, I., Allende-Cid, H., Rosso, O. A., Ramos, H. S. (2020). Leveraging Phase Transition of Topics for Event Detection in Social Media. IEEE Access, 8, 70505-70518.

8.2.2 Conference papers

Barros, P. H., Cardoso-Pereira, I., Foschini, L., Corradi, A., Ramos, H. S. (2019, June). Load balancing in D2D networks Using Reinforcement Learning. In 2019 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-6). IEEE.

Cardoso, I., Barros, P. H., Borges, J., Loureiro, A. A. F., Ramos, H. S. (2019, May). Classificação de Séries Temporais Através de Grafos de Transição de Padrões Ordinais. In Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (pp. 622-635). SBC.

Tenorio, F. F. A., Chagas, E., Barros, P. H., Ramos, H. S. (2019, September). Detecçao de eventos no twitter através de grafos de visibilidade natural. In Anais do III Workshop de Computação Urbana (pp. 181-193). SBC.

8.2.3 UnderSubmission

(X) **Neuralcomputing** : Barros, P. H., Queiroz, F., Figueredo, F., Santos, J. A. D., Ramos, H. S. (2020). A New Similarity Space Tailored for Supervised Deep Metric Learning. arXiv preprint arXiv:2011.08325.

8.2.4 Short course

https://www2.dcc.ufmg.br/cursos/deeplearning/

8.2.5 Open Work

In addition, we have 4 papers in the writing process.

– The paper analyzes mobility patterns in the context of the SARS-Cov-2 pandemic. In this work we use the epidemiological model to characterize mobility. This work will be sent to the PLOS ONE Journal.

– (X) Titled paper: *Supervised Distance Metric learning Encoder with Similarity Space for malware classification through image representation.* This paper is an extension of Chapter 7. This work will be sent to the Computer Networks Journal.

– (X) Titled paper: *Deep Learning for Leishmaniasis Parasite Detection in Bone Marrow Smears.* This paper is an extension of Chapter 6. This work will be sent to the IEEE Transactions on Medical Imaging.

– (X) Titled paper: *Zero-shot learning tailored Similarity Space for Space-time correlation grouping sensors in smart buildings.* This work groups sensors located in same environment (for example, room), based on Zero-Short Learning paradigm with a new version of SMELL adapted for time series. This work will be sent to the Expert Systems with Applications Journal.

Bibliography

[1] Abdullayeva, F. (2019). Malware detection in cloud computing using an image visualization technique. In *2019 IEEE 13th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–5.

[2] Agarap, A. F. and Pepito, F. J. H. (2018a). Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification. *CoRR*, abs/1801.00318.

[3] Agarap, A. F. and Pepito, F. J. H. (2018b). Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification. *CoRR*, abs/1801.00318.

[4] Ahmed, E., Jones, M., and Marks, T. K. (2015). An improved deep learning architecture for person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3908–3916.

[5] Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M. J., Levi, M., Moore, T., and Savage, S. (2013). Measuring the cost of cybercrime. In *The economics of information security and privacy*, pages 265--300. Springer.

[6] Antinori, S., Calattini, S., Longhi, E., Bestetti, G., Piolini, R., Magni, C., Orlando, G., Gramiccia, M., Acquaviva, V., Foschi, A., Corvasce, S., Colomba, C., Titone, L., Parravicini, C., Cascio, A., and Corbellino, M. (2007). Clinical Use of Polymerase Chain Reaction Performed on Peripheral Blood and Bone Marrow Samples for the Diagnosis and Monitoring of Visceral Leishmaniasis in HIV-Infected and HIV-Uninfected Patients: A Single-Center, 8-Year Experience in Italy and Review of the Literature. *Clinical Infectious Diseases*, 44(12):1602–1610. ISSN 1058-4838.

[7] Bailey, M., Oberheide, J., Andersen, J., Mao, Z. M., Jahanian, F., and Nazario, J. (2007). Automated classification and analysis of internet malware. In Kruegel, C., Lippmann, R., and Clark, A., editors, *Recent Advances in Intrusion Detection*, pages 178--197, Berlin, Heidelberg. Springer Berlin Heidelberg.

[8] Baptista, I., Shiaeles, S., and Kolokotronis, N. (2019). A novel malware detection system based on machine learning and binary visualization. *CoRR*, abs/1904.00859.

[9] Barros, P. H., Queiroz, F., Figueredo, F., dos Santos, J. A., and Ramos, H. S. (2020). A new similarity space tailored for supervised deep metric learning. *arXiv preprint arXiv:2011.08325.*

[10] Bozkir, A. S., Tahillioglu, E., Aydos, M., and Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers and Security*, 103:102166. ISSN 0167-4048.

[11] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 737--744.

[12] Cakir, F., He, K., Xia, X., Kulis, B., and Sclaroff, S. (2019). Deep metric learning to rank. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1861–1870.

[13] Cao, X., Ge, Y., Li, R., Zhao, J., and Jiao, L. (2019). Hyperspectral imagery classification with deep metric learning. *Neurocomputing*, 356:217 – 227. ISSN 0925-2312.

[14] Cheng, G., Yang, C., Yao, X., Guo, L., and Han, J. (2018). When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative cnns. *IEEE Transactions on Geoscience and Remote Sensing*, 56(5):2811–2821.

[15] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539--546.

[16] Chopra, S., Hadsell, R., LeCun, Y., et al. (2005). Learning a similarity metric discriminatively, with application to face verification. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 539--546.

[17] Corum, A., Jenkins, D., and Zheng, J. (2019). Robust pdf malware detection with image visualization and processing techniques. In *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*, pages 108–114.

[18] Cui, Y., Zhou, F., Lin, Y., and Belongie, S. (2016). Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[19] Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G., and Chen, J. (2018). Detection of malicious code variants based on deep learning. *IEEE Transactions on Industrial Informatics*, 14(7):3187–3196.

[20] Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *International Conference on Machine Learning (ICML)*, pages 209--216.

[21] De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. (2000). The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1--18.

[22] Deng, Z., Zhu, X., Cheng, D., Zong, M., and Zhang, S. (2016). Efficient knn classification algorithm for big data. *Neurocomputing*, 195:143 – 148.

[23] Deudon, M. (2018). Learning semantic similarity in a continuous space. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 986--997.

[24] Dizaji, K. G., Herandi, A., Deng, C., Cai, W., and Huang, H. (2017). Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5747–5756.

[25] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, number 1 in Proceedings of Machine Learning Research, pages 647--655, Bejing, China. PMLR.

[26] Elmahallawy, E. K., Sampedro, A. M., Rodriguez-Granger, J., Hoyos-Mallecot, Y., Agil, A., Navarro, J. M., and Fernández, J. (2014). Diagnosis of leishmaniasis. *Journal of Infection in Developing Countries*, 8(8):961 – 972. ISSN 2036-6590.

[27] Farahi, M., Rabbani, H., and Mehri, A. (2014). Automatic boundary extraction of leishman bodies in bone marrow samples from patients with visceral leishmaniasis. *Journal of Isfahan Medical School*, 32(286).

[28] Fu, J., Xue, J., Wang, Y., Liu, Z., and Shan, C. (2018). Malware visualization for fine-grained classification. *IEEE Access*, 6:14510--14523.

[29] Globerson, A. and Roweis, S. (2005). Metric learning by collapsing classes. In *International Conference on Neural Information Processing Systems*, NIPS'05, pages 451--458.

[30] Go, J. H., Jan, T., Mohanty, M., Patel, O. P., Puthal, D., and Prasad, M. (2020). Visualization approach for malware classification with resnext. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7.

[31] Goldberger, J., Hinton, G. E., Roweis, S. T., and Salakhutdinov, R. R. (2005). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 513--520.

[32] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[33] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735--1742. IEEE.

[34] Han, K., Kang, B., and Im, E. G. (2014). Malware analysis using visualized image matrices. *The Scientific World Journal*, 2014.

[35] Han, K. S., Lim, J. H., Kang, B., and Im, E. G. (2015). Malware analysis using visualized images and entropy graphs. *International Journal of Information Security*, 14(1):1--14.

[36] Hecht-Nielsen (1989). Theory of the backpropagation neural network. In *Conference on Neural Networks*, pages 593–605.

[37] Ho, E. A., Soong, T.-H., and Li, Y. (1948). Comparative merits of sternum, spleen and liver punctures in the study of human visceral leishmaniasis. *Transactions of The Royal Society of Tropical Medicine and Hygiene*, 41(5):629–636. ISSN 0035-9203.

[38] Hong, X., Gerla, M., Pei, G., and Chiang, C.-C. (1999). A group mobility model for ad hoc wireless networks. In *International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile systems*, pages 53--60.

[39] Hou, R., Ma, B., Chang, H., Gu, X., Shan, S., and Chen, X. (2019). Interaction-and-aggregation network for person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[40] Huai, M., Miao, C., Li, Y., Suo, Q., Su, L., and Zhang, A. (2018). Metric learning from probabilistic labels. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1541--1550.

[41] Inaba, S., Fakhry, C. T., Kulkarni, R. V., and Zarringhalam, K. (2019). A free energy based approach for distance metric learning. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 5--13.

[42] Johnson, N. L., Kotz, S., and Balakrishnan, N. (1995). *Continuous univariate distributions, volume 2*, volume 289. John wiley & sons.

[43] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D. B., Wang, Y., and Iqbal, F. (2018). Malware classification with deep convolutional neural networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5.

[44] Kan, S., Zhang, L., He, Z., Cen, Y., Chen, S., and Zhou, J. (2020). Metric learning-based kernel transformer with triplets and label constraints for feature fusion. *Pattern Recognition*, 99:107086.

[45] Kancherla, K. and Mukkamala, S. (2013). Image visualization based malware detection. In *2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pages 40–44.

[46] Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

[47] Kumar, A., Sagar, K. P., Kuppusamy, K. S., and Aghila, G. (2016). Machine learning based malware classification for android applications using multimodal image representations. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–6.

[48] Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security Privacy*, 9(3):49–51.

[49] Li, F., Qiao, H., and Zhang, B. (2018). Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83:161--173.

[50] Li, S., Hong, D., and Wang, H. (2020). Relation inference among sensor time series in smart buildings with metric learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4683--4690. AAAI Press.

[51] Lin, Y., Jiang, J., and Lee, S. (2014). A similarity measure for text classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1575–1590.

[52] Liu, Y., Zhao, K., and Cong, G. (2018). Efficient similar region search with deep metric learning. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1850--1859.

[53] Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

[54] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(Nov):2579--2605.

[55] Makandar, A. and Patrot, A. (2015). Malware analysis and classification using artificial neural network. In *2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, pages 1–6.

[56] Mao, C., Zhong, Z., Yang, J., Vondrick, C., and Ray, B. (2019). Metric learning for adversarial robustness. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 480--491. Curran Associates, Inc.

[57] McFee, B. and Lanckriet, G. R. (2010). Metric learning to rank. In *International Conference on Machine Learning (ICML)*, pages 775--782.

[58] Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K. R. (1999). Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48.

[59] Mockus, J. (1975). On the bayes methods for seeking the extremal point. *IFAC Proceedings Volumes*, 8:428 – 431.

[60] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *International Conference on International Conference on Machine Learning (ICML)*, pages 807--814.

[61] Narayanan, B. N., Djaneye-Boundjou, O., and Kebede, T. M. (2016). Performance analysis of machine learning and pattern recognition algorithms for malware classification. In *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, pages 338--342. IEEE.

[62] Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. (2011a). Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*, pages 1--7.

[63] Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. (2011b). Malware images: Visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, New York, NY, USA. Association for Computing Machinery.

[64] Nguyen, B. and De Baets, B. (2019). Kernel-based distance metric learning for supervised k -means clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 30(10):3084–3095.

[65] Nguyen, B. and De Baets, B. (2020). Improved deep embedding learning based on stochastic symmetric triplet loss and local sampling. *Neurocomputing*, 402:209 – 219. ISSN 0925-2312.

[66] Nguyen, B., Morell, C., and Baets, B. D. (2017). Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition*, 64:215 – 225. ISSN 0031-3203.

[67] Niethammer, M., Kwitt, R., and Vialard, F.-X. (2019). Metric learning for image registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[68] Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[69] Paixao, T. M., Berriel, R. F., Boeres, M. C. S., Koerich, A. L., Badue, C., Souza, A. F. D., and Oliveira-Santos, T. (2020). Fast(er) reconstruction of shredded text documents via self-supervised deep asymmetric metric learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[70] Peters, B. S., Fish, D., Golden, R., Evans, D. A., Bryceson, A. D. M., and Pinching, A. J. (1990). Visceral Leishmaniasis in HIV Infection and AIDS: Clinical Features and Response to Therapy. *QJM: An International Journal of Medicine*, 77(2):1101–1111. ISSN 1460-2725.

[71] Petrik, R., Arik, B., and Smith, J. M. (2018). Towards architecture and os-independent malware detection via memory forensics. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 2267–2269, New York, NY, USA. Association for Computing Machinery.

[72] Priyamvada Davuluru, V. S., Narayanan Narayanan, B., and Balster, E. J. (2019). Convolutional neural networks as classification tools and feature extractors for distinguishing malware programs. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 273–278.

[73] Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., and Nicholas, C. K. (2018). Malware detection by eating a whole exe. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

[74] Rasti, R., Rabbani, H., Mehridehnavi, A., and Hajizadeh, F. (2018). Macular oct classification using a multi-scale convolutional neural network ensemble. *IEEE Transactions on Medical Imaging*, 37(4):1024–1034. ISSN 0278-0062.

[75] Relli, C., Facon, J., Ayala, H. L., and Britto Jr, A. D. S. (2017). Automatic counting of trypanosomatid amastigotes in infected human cells. *Computers in Biology and Medicine*, 89:222--235.

[76] Rezende, E., Ruppert, G., Carvalho, T., Theophilo, A., Ramos, F., and Geus, P. d. (2018). Malicious software classification using vgg16 deep neural network's bottleneck features. In Latifi, S., editor, *Information Technology - New Generations*, pages 51--59, Cham. Springer International Publishing.

[77] Roseline, S. A., Geetha, S., Kadry, S., and Nam, Y. (2020). Intelligent vision-based malware detection and classification using deep random forest paradigm. *IEEE Access*, 8:206303–206324.

[78] Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829.*

[79] Saeed, I. A., Selamat, A., and Abuagoub, A. M. (2013). A survey on malware and malware detection systems. *International Journal of Computer Applications*, 67(16).

[80] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[81] Shen, C., Jin, Z., Zhao, Y., Fu, Z., Jiang, R., Chen, Y., and Hua, X.-S. (2017). Deep siamese network with multi-level similarity perception for person re-identification. In *Proceedings of the 25th ACM International Conference on Multimedia*, pages 1942--1950.

[82] Shen, J., Wang, H., Zhang, A., Qiu, Q., Zhen, X., and Cao, X. (2020). Model-agnostic metric for zero-shot learning. In *The IEEE Winter Conference on Applications of Computer Vision (WACV).*

[83] Shi, H., Yang, Y., Zhu, X., Liao, S., Lei, Z., Zheng, W., and Li, S. Z. (2016). Embedding deep metric for person re-identification: A study against large variations. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 732--748, Cham. Springer International Publishing.

[84] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of ICLR.*

[85] Singh, A., Handa, A., Kumar, N., and Shukla, S. K. (2019). Malware classification using image representation. In Dolev, S., Hendler, D., Lodha, S., and Yung, M., editors, *Cyber Security Cryptography and Machine Learning*, pages 75--92, Cham. Springer International Publishing.

[86] Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 1857--1865. Curran Associates, Inc.

[87] Swiderska-Chadaj, Z., Pinckaers, H., [van Rijthoven], M., Balkenhol, M., Melnikova, M., Geessink, O., Manson, Q., Sherman, M., Polonia, A., Parry, J., Abubakar, M., Litjens, G., [van der Laak], J., and Ciompi, F. (2019). Learning to detect lymphocytes

in immunohistochemistry with deep learning. *Medical Image Analysis*, 58:101547. ISSN 1361-8415.

[88] Szor, P. (2005). *The Art of Computer Virus Research and Defense: ART COMP VIRUS RES DEFENSE _p1*. Pearson Education.

[89] Torresani, L. and Lee, K.-c. (2007). Large margin component analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1385--1392.

[90] Triguero, I., González, S., Moyano, J. M., García, S., Alcalá-Fdez, J., Luengo, J., Fernández, A., del Jesús, M. J., Sánchez, L., and Herrera, F. (2017). Keel 3.0: An open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10:1238–1249.

[91] Ustinova, E. and Lempitsky, V. (2016). Learning deep embeddings with histogram loss. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4170--4178. Curran Associates, Inc.

[92] Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., and Zheng, Q. (2020a). Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171:107138. ISSN 1389-1286.

[93] Vasan, D., Alazab, M., Wassan, S., Safaei, B., and Zheng, Q. (2020b). Image-based malware classification using ensemble of cnn architectures (imcec). *Computers and Security*, 92:101748. ISSN 0167-4048.

[94] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., and Venkatraman, S. (2019). Robust intelligent malware detection using deep learning. *IEEE Access*, 7:46717–46738.

[95] Vogel, R., Bellet, A., and Clémençon, S. (2018). A probabilistic theory of supervised similarity learning for pointwise ROC curve optimization. In *International Conference on Machine Learning (ICML)*, pages 5065--5074.

[96] Wang, F. and Zhang, C. (2007). Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[97] Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., and Liu, W. (2018). Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5265–5274.

[98] Wang, J., Gao, X., Wang, Q., and Li, Y. (2012). Prodis-contshc: learning protein dissimilarity measures and hierarchical context coherently for protein-protein comparison in protein database retrieval. *BMC Bioinformatics*, 13(7):S2.

[99] Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. (2017). Deep metric learning with angular loss. In *The IEEE International Conference on Computer Vision (ICCV).*

[100] Wang, L., Yang, B., Chen, Y., Zhang, X., and Orchard, J. (2017). Improving neural-network classifiers using nearest neighbor partitioning. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2255–2267.

[101] Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. (2019a). Multi-similarity loss with general pair weighting for deep metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[102] Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. (2019b). Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5022--5030.

[103] Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 10:207--244.

[104] Werneck, G., Rodrigues, L., Santos, M., Araújo, I., Moura, L., Lima, S., Gomes, R., Maguire, J., and Costa, C. (2002). The burden of leishmania chagasi infection during an urban outbreak of visceral leishmaniasis in brazil. *Acta Tropica*, 83(1):13 – 18. ISSN 0001-706X.

[105] World Health Organization. Pan American Health Organization (2019). Epidemiological report of the americas. leishmaniases.

[106] Wressnegger, C., Freeman, K., Yamaguchi, F., and Rieck, K. (2017). Automatically inferring malware signatures for anti-virus assisted attacks. ASIA CCS '17, page 587–598, New York, NY, USA. Association for Computing Machinery.

[107] Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. (2017). Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840--2848.

[108] Wu, H., Zhou, Q., Nie, R., and Cao, J. (2020). Effective metric learning with co-occurrence embedding for collaborative recommendations. *Neural Networks.*

[109] Wu, L., Hoi, S. C. H., Jin, R., Zhu, J., and Yu, N. (2012). Learning bregman distance functions for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):478–491.

[110] Xiang, S., Nie, F., and Zhang, C. (2008). Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600 – 3612. ISSN 0031-3203.

[111] Xiaofang, B., Li, C., Weihua, H., and Qu, W. (2014). Malware variant detection using similarity search over content fingerprint. In *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pages 5334--5339. IEEE.

[112] Xie, J., Girshick, R., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, pages 478--487.

[113] Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2002). Distance metric learning, with application to clustering with side-information. In *International Conference on Neural Information Processing Systems*, NIPS'02, pages 521--528.

[114] Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y., and Sakuma, J. (2019). Neural malware analysis with attention mechanism. *Computers and Security*, 87:101592. ISSN 0167-4048.

[115] Yang, F., Poostchi, M., Yu, H., Zhou, Z., Silamut, K., Yu, J., Maude, R. J., Jaeger, S., and Antani, S. (2020). Deep learning for smartphone-based malaria parasite detection in thick blood smears. *IEEE Journal of Biomedical and Health Informatics*, 24(5):1427–1438.

[116] Yang, Y., Chen, H., and Shao, J. (2019). Triplet enhanced autoencoder: Model-free discriminative network embedding. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5363--5369.

[117] Yoo, S., Kim, S., and Kang, B. B. (2020). The image game: Exploit kit detection based on recursive convolutional neural networks. *IEEE Access*, 8:18808–18821.

[118] Zhang, P., Sun, B., Ma, R., and Li, A. (2019). A novel visualization malware detection method based on spp-net. In *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, pages 510–514.

[119] Zhao, Y., Cui, W., Geng, S., Bo, B., Feng, Y., and Zhang, W. (2020). A malware detection method of code texture visualization based on an improved faster rcnn combining transfer learning. *IEEE Access*, 8:166630–166641.

[120] Zheng, F., Deng, C., Sun, X., Jiang, X., Guo, X., Yu, Z., Huang, F., and Ji, R. (2019). Pyramidal person re-identification via multi-loss dynamic training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[121] Çayır, A., Ünal, U., and Dağ, H. (2020). Random capsnet forest model for imbalanced malware type classification task.