Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

DATA SCIENCE AND COMPUTATION

Ciclo 34

Settore Concorsuale: 09/H1 – SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 – SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

DEVELOPMENT OF UNSUPERVISED LEARNING METHODS WITH APPLICATIONS TO LIFE SCIENCES DATA

Presentata da: Erika Gardini

Coordinatore Dottorato
Daniele Bonacorsi

Supervisore
Andrea Cavalli

Co-supervisore
Sergio Decherchi

Esame finale anno 2023

Abstract

Machine Learning makes computers capable of performing tasks typically requiring human intelligence. A domain where it is having a considerable impact is the life sciences, allowing to devise new biological analysis protocols, develop patients' treatments efficiently and faster, and reduce healthcare costs. This Thesis work presents new Machine Learning methods and pipelines for the life sciences focusing on the unsupervised field.

At a methodological level, two methods are presented. The first is an "Ab Initio Local Principal Path" and it is a revised and improved version of a pre-existing algorithm in the manifold learning realm. The second contribution is an improvement over the Import Vector Domain Description (one-class learning) through the Kullback-Leibler divergence. It hybridizes kernel methods to Deep Learning obtaining a scalable solution, an improved probabilistic model, and state-of-the-art performances. Both methods are tested through several experiments, with a central focus on their relevance in life sciences. Results show that they improve the performances achieved by their previous versions.

At the applicative level, two pipelines are presented. The first one is for the analysis of RNA-Seq datasets, both transcriptomic and single-cell data, and is aimed at identifying genes that may be involved in biological processes (e.g., the transition of tissues from normal to cancer). In this project, an R package is released on CRAN to make the pipeline accessible to the bioinformatic Community through high-level APIs. The second pipeline is in the drug discovery domain and is useful for identifying druggable pockets, namely regions of a protein with a high probability of accepting a small molecule (a drug). Both these pipelines achieve remarkable results.

Lastly, a detour application is developed to identify the strengths/limitations of the

"Principal Path" algorithm by analyzing Convolutional Neural Networks induced vector spaces. This application is conducted in the music and visual arts domains.

Keywords: Machine Learning, Unsupervised Learning, Life Sciences, Manifold Learning, One-class Learning, Self-Supervised Learning.

Acknowledgements

First, I express my sincere gratitude to my supervisor, Professor Andrea Cavalli, and my co-supervisor, Sergio Decherchi, Ph.D., for allowing me to conduct this research under their auspices. I am incredibly grateful for their confidence and the freedom they gave me to do this work. They supported me in all the stages of this work. They always gave me constant encouragement and advice; without coherent and illuminating instruction, this Thesis would not have reached its present form.

Also, I want to thank my mother, father, and sister for their psychological support. Without them, I would never reach this vital goal, and I would never find the courage to overcome all the difficulties encountered during this work. My thanks also go to all my family members for their love during all these years.

I want to thank all my colleagues and friends, particularly Luca, Nicola, Chiara, and Daniele, for their direct or indirect encouragement during these years. They always supported me and helped me overcome difficulties without complaining.

I want to extend my warmest thanks to my boyfriend, Mattia. He always encouraged and supported me; he kept me company during my last sleepless nights. He helped me address all the difficulties and gave me the strength to move on.

If this work has sometimes prevented me from sharing essential moments of life with all of you, know that I never stopped thinking about you.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
AE	Auto-Encoder
AI	Artificial Intelligence
AUC	Area Under the Curve
ANN	Artificial Neural Network
CI	Computational Intelligence
CNN	Convolutional Neural Network
CRAN	Comprehensive R Archive Network
CV	Computer Vision
DAE	Denoising Auto-Encoder
DGEA	Differential Gene Expression Analysis
DL	Deep Learning
EBM	Energy-Based Modelling
EM	Expectation Maximization
GAN	Generative Adversarial Network
HPC	High Performance Computing
IVDD	Import Vector Domain Description
KL	Kullback-Leibler
MCC	Metthews Correlation Coefficient
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNIST	Modified National Institute of Standard and Technology
NGS	Next-Generation Sequences
OC-SVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
PDTD	Potential Drug Target Database
PP	Principal Path
PPI	Protein-Protein Interface
ReLU	Rectifier Linear Unit
SES	Solvent Excluded Surface
SGVB	Stochastic Gradient Variational Bayes
SMO	Sequential Minimal Optimization
SPSD	Symmetric Positive Semidefinite
SSIM	Structural Similarity Index
SSL	Self-Supervised Learning
SVDD	Support Vector Domain Description
SV	Support Vector
SVM	Support Vector Machine
TSG	Tumor Suppressor Gene
t-SNE	t-Distributed Stochastic Neighbour Embedding
VAE	Variational Auto-Encoder

Chapter 1

Introduction

Artificial Intelligence (AI) is a relatively young field in science and engineering, with the first ideas dating back to Turing, and comprises all the theories and the development of computer systems able to perform tasks typically requiring human intelligence (1).

The most successful AI applications are based on Machine Learning (ML) technologies, which include all the algorithms capable of *learning from experience* (2). ML is inherently a multidisciplinary field and is highly interconnected in many areas, including Data Mining (3), Statistical Learning (4), Pattern Recognition (5), and Mathematics (6) (see Figure 1.1). Topics like natural language (text and speech) processing (7), translation between languages (8), genetic sequence analysis (9), robotics (10), customer (financial risk) evaluation (11), terrorist threat detection (12), compiler optimization (13), semantic web (14), computer security (15), computer vision (CV) (16), visual perception (17), object detection (18), decision-making (19), recommendation systems (20) are just examples of the possible application domains of ML.

In ML, experience exists in the form of input data. Therefore the *learning from experience* concept involves finding a learning algorithm that builds models from these data. A ML *model* is a compact representation of the data (usually a function) parameterized by learnable *parameters*, often termed weights. On the other hand, data are collections of *records* (or *instances* or *samples*). Each record is a description of an event/object that, in turn, is a set of *attributes* (or *features*) of the event/object

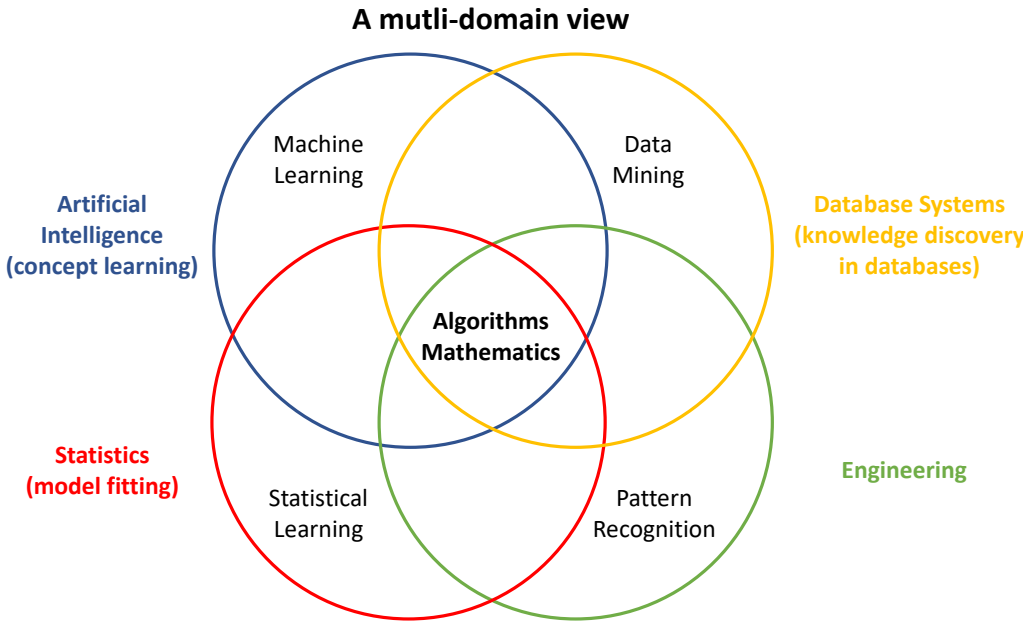

Figure 1.1: *A multi-domain view of ML and closely related fields*

(e.g., color, shape) (21). More generally, a dataset composed of n instances is a set:

$$\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}, \tag{1.1}$$

and each of its entries can be described by d attributes:

$$\boldsymbol{x}_i = (x_{i1}, ..., x_{id}) \in \mathcal{X}. \tag{1.2}$$

Hence each instance of a dataset is a vector in the d-dimensional sample space $\mathcal{X}$ (21). The subset of the dataset that is used by the learning algorithm for creating the model is called *training set*. In addition, the learning algorithm involves some hyper-parameters that control the learning process and determine the values of the model parameters. Different hyper-parameters may result in other models. For this reason, a second subset, called *validation set*, is used during the training phase to identify the best model. This second independent set is used to avoid the so-called over-fitting, preventing memorization in favor of abstraction. The ability to work on unseen samples is called *generalization ability*, and a well-generalizing model should work well on the whole sample space. Finally, the portion of the dataset used for making predictions through a learned model is called *test set* (21). When the model

produces an explicit output value, this is called *label*. More generally, the ith sample of the dataset $\mathcal{D}$ is $(\boldsymbol{x}_i, y_i)$, where $y_i \in \mathcal{Y}$ is the label of the sample $\boldsymbol{x}_i$ and $\mathcal{Y}$ is the set of all labels, also called label space or output space (21) which is provided at training time. For measuring how well the learning algorithm models the dataset, a *loss function* is needed, which is an error measure. This loss function is often enriched with further terms that do not measure the error but consider numerical stability issues or a priori expectations on the model.

In summary, a ML problem is characterized by a well-specified task. During the training phase, the training examples are fed into the model, and the prediction error made by the model is estimated through the loss function. The model then modifies its internal adjustable parameters to reduce this error often through a gradient descent procedure which brings the loss to a local minimum (see Figure 1.2). The parameters, w, are updated as per:

$$w^{t+1} = w^t - \eta \nabla_w C, \tag{1.3}$$

where w^{t+1} is the parameters vector at step $t+1$, w^t is the set of parameters at step t, η is called learning rate and rules the descent speed, and $\nabla_w C$ is the gradient of the cost function C. After the training procedure is completed, and hence the loss is minimized, the model's performance is measured on the validation set to select the best model. Finally, the test set is used to evaluate the generalization ability of the machine (22). Figure 1.3 graphically summarizes this procedure.

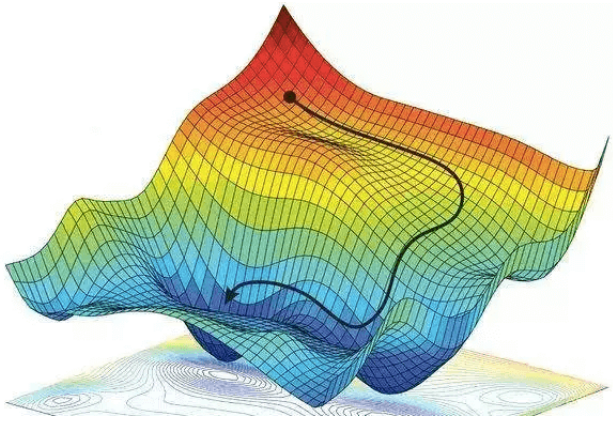

Figure 1.2: *Gradient descent: a schematic representation.*

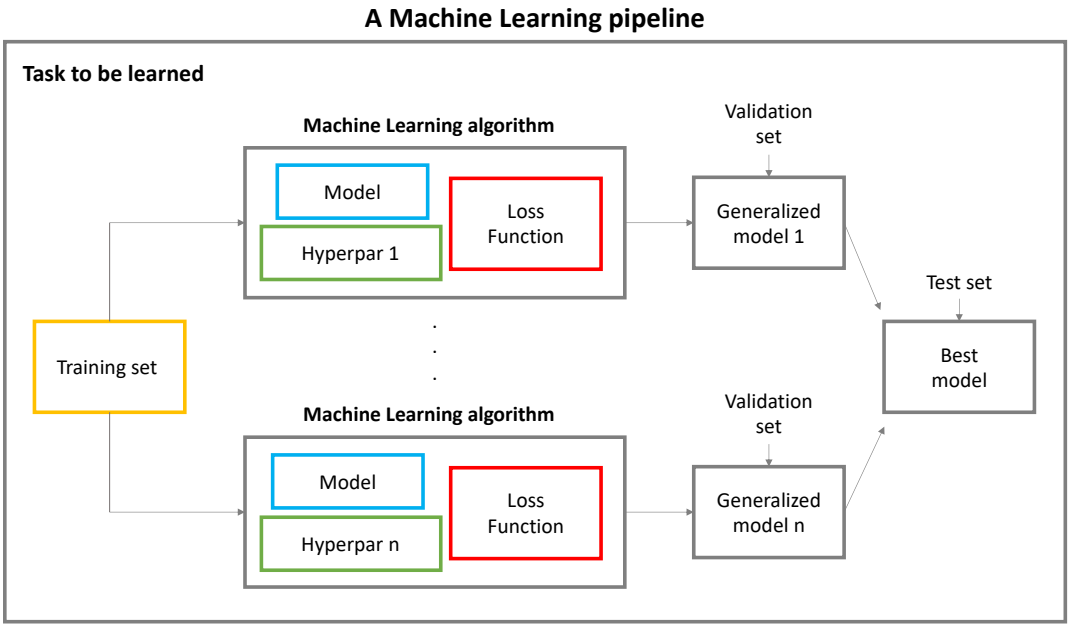

Figure 1.3: *A typical ML pipeline*

Depending on the available information on the training examples, ML methods can be subdivided into two main categories: *supervised* methods and *unsupervised* methods. The first class includes systems and algorithms that learn a predictive model using samples with known labels. On the other hand, unsupervised learning deals with unlabelled data and aims at studying the intrinsic and hidden structure of the data to get meaningful insights. Examples of unsupervised learning tasks include but are not limited to clustering ([23]), manifold learning ([24]), and anomaly detection ([25]). Purely supervised learning algorithms have been successfully applied in various application areas. However, they require expensive, possibly manual, labeling to produce large training datasets. For this reason, more and more researchers have focused on developing unsupervised learning algorithms. This class of methods bears several challenges and is definitely more complex than supervised learning. In addition, natural learning, in general, is mainly unsupervised: humans and animals discover the structure of the world by observing, sensing, and modifying it, not by being told the name of every entity ([22]).

This Thesis work is focused on unsupervised learning strategies, aiming at developing, customizing, and improving methods. The effectiveness of the proposed methods

and approaches is proven by performing several experiments in diverse Data Science domains. A central focus of this work is the use and relevance of the proposed methods in the Life Science domain (e.g., drug discovery, omics data) through chemical and biological datasets analysis. It is widely known that the development, maturation, and advancement of AI techniques have a considerable impact on the life science industry ([26]), allowing companies to develop treatments efficiently and faster, reducing the cost of health care, and making it more accessible to patients. In the drug discovery domain, ML methods can be deployed to design the proper structure for drugs ([27]). They can also be used for making predictions around bioactivity, toxicity, and physicochemical properties ([28]; [29]). This computational strategy speeds up the drug development process and helps ensure that the drugs deliver the optimal therapeutic response when administered to patients. AI and ML are also effective at identifying characteristics in images and data that can be difficult to detect by a human operator ([30]) in the clinical domain. ML strategies can improve cervical and prostate cancer screening and identify specific gene mutations from tumor pathology images or genome sequences ([31]; [32]). AI may also be employed to diagnose other conditions, including heart diseases and diabetic retinopathy ([33]). ML techniques can help people enjoy longer, healthier lives by enabling the early detection of life-threatening diseases.

In this Thesis work, Chapter 2 introduces the unsupervised learning domains which are covered in subsequent Chapters. In particular, Section 2.1 describes some of the most important methods in the dimensionality reduction and manifold learning realms. Among others, the Principal Curve concept ([34]) and the Principal Path (PP) algorithm ([35]) are also presented. Section 2.2 describes the most relevant and recent algorithms for one-class classification and discusses some of the open problems in this field. Finally, section 2.3 describes the representation learning domain together with some of the methods adopted for the learning strategy: Artificial Neural Networks (ANNs), Autoencoders (AEs) and self-supervised learning (SSL) approaches.

Chapter 3 describes original methodological contributions to the manifold learning field and their application to omics data analysis. In particular: Section 3.1 describes the PP algorithm presented in ([35]). Sections 3.2.1 and 3.2.2 present two applications

of the PP. The first one is in the life science domain. It consists of a pipeline, for the analysis of RNA-Seq datasets, both transcriptomic and single-cell, and aims at identifying genes that may be involved in biological processes (e.g., the transition of tissues from normal to cancer). The second one is a detour application which aims to highlight the strengths and identify the limitations of the PP algorithm by analyzing the complex Convolutional Neural Network (CNN) induced vector space. This application is conducted in the music and visual arts domains. Finally, Section 3.3 presents a revised version of the PP algorithm, dubbed Ab Initio Local PP, which solves some of the drawbacks and limitations of the original method.

Chapter 4 is devoted to the development of one-class learning methods and their application to drug discovery. In particular, Section 4.1 describes the Import Vector Domain Description (IVDD) method and its variants ([36; 37]). Section 4.2, instead, presents an application in the drug discovery domain. It involves the IVDD method and is devoted to the development of a pipeline for identifying druggable pockets. Finally, Section 4.3 presents an enhanced version of the IVDD method, dubbed Import Vector Domain Description Kullback-Leibler (IVDD-KL), which hybridizes kernel methods to Deep Learning (DL) approaches so as to obtain a scalable solution together with an improved probabilistic model. To close the Thesis, Chapter 5 presents some final remarks and conclusions.

Chapter 2

Unsupervised learning

Unsupervised learning is a sub-field of ML in which, as anticipated, the learning algorithm discovers patterns and regularities that are not easily detectable by human inspection. It eventually studies the intrinsic and hidden structure of the data (38). In unsupervised learning, only unlabelled data is available.

Figure 2.1 shows some of the subcategories of unsupervised learning. Association rules are one of the most important and well-researched techniques of data mining and are used to extract interesting correlations, frequent patterns, associations, or casual structures among sets of items in transaction databases or other data repositories

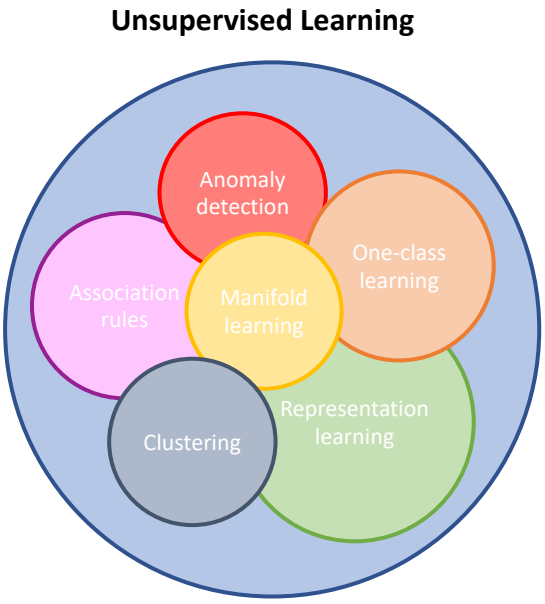

Figure 2.1: *Unsupervised learning and its subcategories*

(39). Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior (25). Manifold learning is a popular approach that allows to get insights into complex data spaces; non-linear dimensionality reduction represents a typical application (24). One-class classification algorithms aim at building classification models when the negative class is either absent, poorly sampled, or not well defined (40). Clustering techniques allow grouping data in a certain number of sets where the elements within each set should be as similar as possible to each other and simultaneously dissimilar from those of other sets (41). Finally, representation learning strategies include pre-processing pipelines and data transformations that allow the extraction and organization of discriminant information from the data (42). This Thesis work deals with manifold and one-class learning.

2.1 Dimensionality reduction and manifold learning

Dimensionality reduction comprises all the methodologies and strategies for reducing the number of attributes/features in a dataset while preserving the *structure* in the original dataset as much as possible. The meaning of *structure* heavily depends on the employed methods, which may have different aims and underlying topological/metric hypotheses. Dimensionality reduction methods can be divided into two groups (see

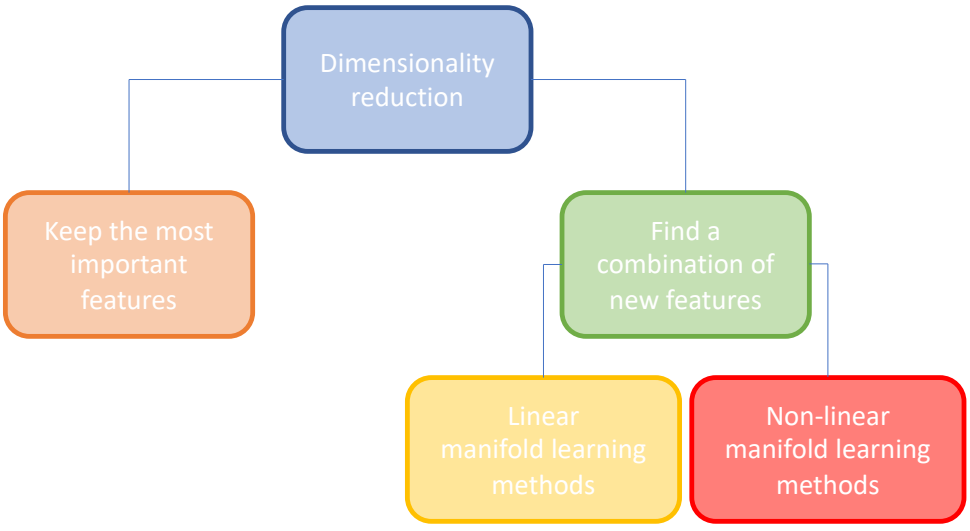

Figure 2.2: *Manifold learning as part of the dimensionality reduction domain.*

Figure 2.2). The first group comprises the methodologies which keep the most important features in a dataset and remove the redundant ones (e.g., variance threshold (43), recursive feature elimination (44), select from model (45)). In this group, any transformation is applied to the features. The other group comprises methods that define new features as, possibly non-linear, transformations of the original ones. All these methods are based on the so-called manifold hypothesis (46), which states that the data spans a sub-manifold of the ambient space and whose intrinsic dimensionality is far smaller than this ambient space (i.e., $\mathbb{R}^n$). This group can be further divided into two subcategories: linear manifold learning methods and non-linear manifold learning methods.

Manifold learning algorithms identify a low-dimensional manifold from data located on that manifold, which is embedded within a higher dimensional ambient space (48); in addition, they capture global, holistic aspects within the analyzed data set. Particularly, they can identify the possibly non-linear low-dimensional manifold underlying a given data set, thus extracting relevant topological information (see

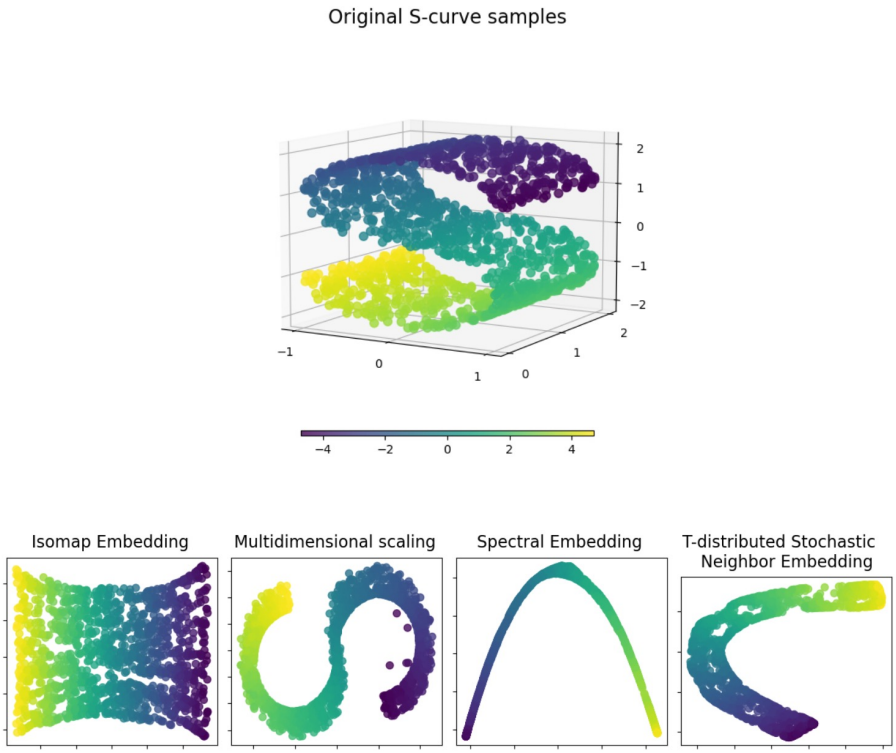

Figure 2.3: *Examples of manifold learning algorithms (47).*

Figure 2.3). Within manifold learning methods, the Principal Curve concept and algorithm (34) is particularly interesting as it is a non-linear 1d projection method. Principal Curves are smooth one-dimensional curves that pass through the *middle* of a p-dimensional dataset, providing a summary of the data. They are non-parametric, and their shape is suggested by the data. More formally, given a random variable $X = (x_1, ..., x_d) \in \mathbb{R}^d$ with known distribution, a smooth (infinitely differentiable) parametrized curve $f(t) = (f_1(t), ..., f_d(t))$ is a Principal Curve for X if f does not intersect itself, if it has finite length inside any bounded subset of $\mathbb{R}^d$, and if it is self-consistent. This last requirement means that

$$f(t) = E[X|t_f(X) = t], \tag{2.1}$$

where the projection index $t_f(x)$ of x is the largest value t for which $f(t)$ is closest to x:

$$t_f(x) = \sup_t \{t : ||x - f(t)|| = \inf_{t'} ||X - f(t')||\}. \tag{2.2}$$

The self-consistency property is like saying that each point of the curve f is the mean of the observations projecting on f around this point (49). In (34), an iterative algorithm is presented, alternating between a projection and a conditional expectation step, which yields an approximate Principal Curve. A visual representation of the Principal Curve concept is depicted in Figures 2.4 and 2.5.

Recently, a local version of the Principal Curve has been presented in (35), dubbed PP algorithm, which will be described in more detail in Chapter 3. The term *local* means

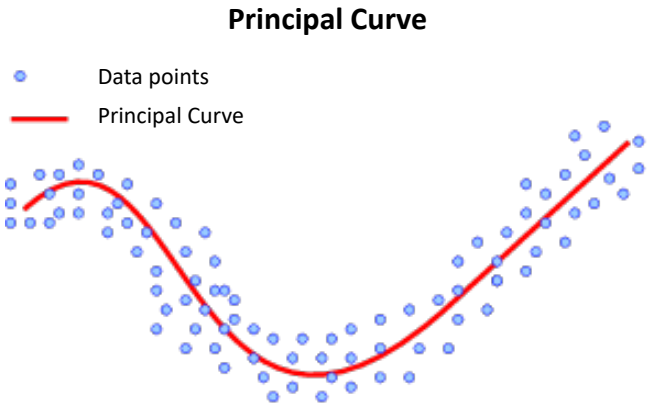

Figure 2.4: *Principal Curve: an example.*

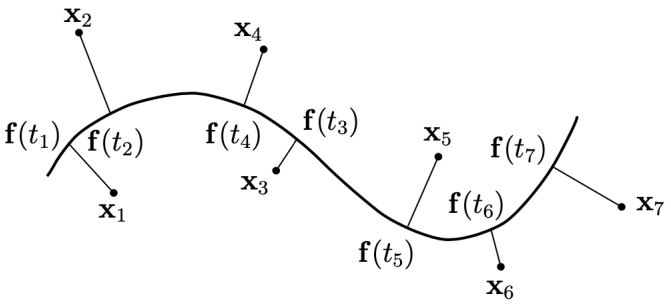

Figure 2.5: *Principal Curve: the projection index t_f.* For all i, t_i stands for $t_f(x_i)$ (49).

that it is not required that the solution is influenced by all the data points but just a subset (see Figure 2.6). The strictness of this constraint is a parameter-dependent aspect that intuitively reminds one of a regularization concept (50). The PP method was introduced to infer relevant transitions through a smooth morphing path starting and ending in two samples belonging to data space. Part of this Thesis is devoted to applying the PP method to diverse contexts also proving the algorithm's generality. These domains include life sciences data, particularly omic ones. In addition, a more robust and stabler version of the PP is introduced here. The devised methods and the results are discussed in Chapter 3.

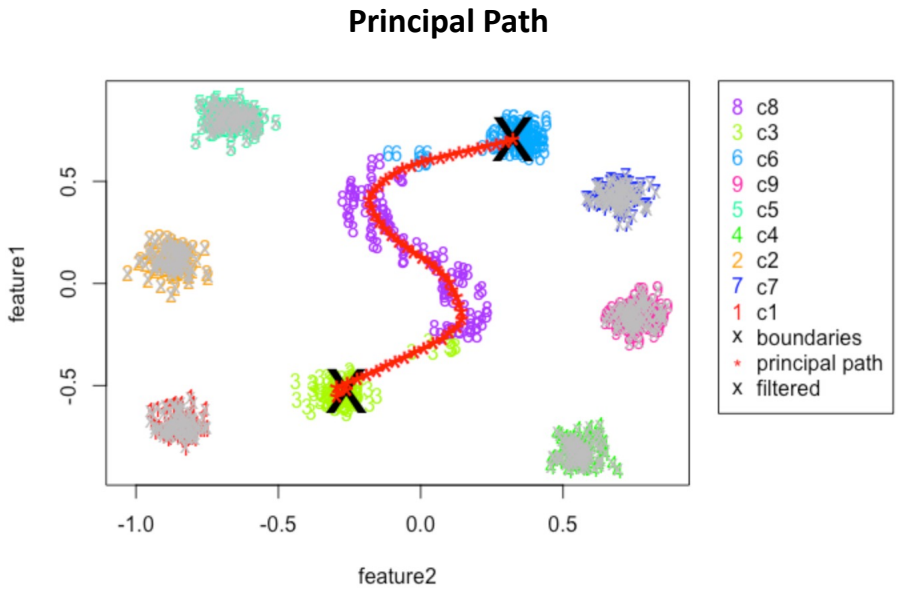

Figure 2.6: *Local Principal Curve: the PP algorithm.*

2.2 One-class learning

A widely important topic in the unsupervised learning realm is one-class learning. The objective of one-class learning is to learn a representation of a single class (one implicit label) such that one can distinguish this class from others that are unknown and not available at training time. One-class learning strategies can be adopted in the anomaly detection domain, aiming to detect unusual data samples for the expected behavior ([25]) (see Figure 2.7). When one-class learning strategies are adopted in the anomaly detection domain, the training process is performed only on *normal* samples (the ones respecting the expected behavior), and no a priori knowledge is given about the other (or unusual) samples ([51]). Typical domains where one-class classification methods are used include but are not limited to: network security ([52]), fraud detection ([53]), medical diagnosis ([54]) and in general, the monitoring of the correct functioning of facilities, plants, and devices ([55]).

Several algorithms have been proposed to address the one-class classification problem. Among others, kernel-based one-class classification methods include the One-Class Support Vector Machine ([56]) (OC-SVM) and the Support Vector Domain Description ([57]) (SVDD) methods. The OC-SVM method aims at finding a maximum margin hyper-plane in feature space that best separates the mapped data from the origin. Particularly, given a dataset $\mathcal{D}_n = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ with $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, OC-SVM solves

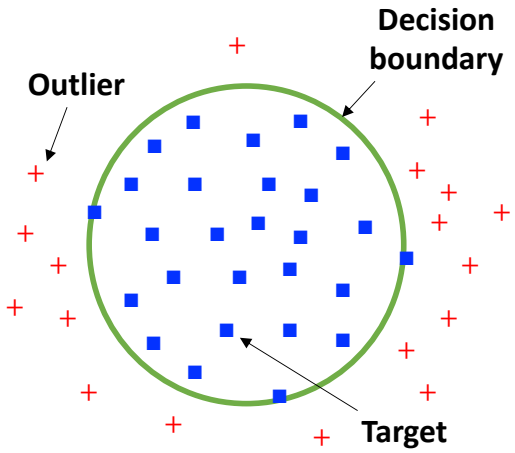

Figure 2.7: *One-class learning: a schematic representation.*

the primal problem:

$$\begin{aligned} &\min_{\boldsymbol{w},\rho,\boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{w}||^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \\ &s.t.(\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i)) \geq \rho - \xi_i, \xi_i \geq 0, \forall i, \end{aligned} \tag{2.3}$$

where ρ is the distance from the origin to hyper-plane $\boldsymbol{w}$, $\boldsymbol{\xi} = (\xi_1, ..., \xi_n)^\top$ are non-negative slack variables, $||\boldsymbol{w}||^2$ is a regularizer of the hyper-plane $\boldsymbol{w}$, the hyper-parameter $\nu \in (0, 1]$ controls the trade-off between the loss and the margin and $\phi(\boldsymbol{x}_i)$ is a, possibly non-linear, map for $\boldsymbol{x}_i$. At the optimum, $\boldsymbol{w}$ and ρ solve the problem in Equation 2.3 and the decision function $f(\boldsymbol{x}) = sgn((\boldsymbol{w}\cdot\phi(\boldsymbol{x})) - \rho)$ separates the data from the origin: it identifies points lying outside the half-space, i.e. $(\boldsymbol{w}\cdot\phi(\boldsymbol{x})) < \rho$, as anomalous. Through a dual Lagrangian derivation, the solution admits a Support Vectors (SVs) expansion:

$$f(\boldsymbol{x}) = sgn\left(\sum_i \alpha_i \mathbf{K}(\boldsymbol{x}_i, \boldsymbol{x}) - \rho\right), \tag{2.4}$$

where the coefficients are found as the solution of the dual problem:

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\sum_{ij}\alpha_i\alpha_j\mathbf{K}(\boldsymbol{x}_i, \boldsymbol{x_j}) \\ &s.t. 0 \leq \alpha_i \leq \frac{1}{\nu n}, \sum_i \alpha_i = 1. \end{aligned} \tag{2.5}$$

Patterns $\boldsymbol{x}_i$ with nonzero α_i are called SVs, while $\mathbf{K}$ is a kernel matrix:

$$\mathbf{K}(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} K(x_1, y_1) & K(x_2, y_1) & \cdots \\ K(x_1, y_2) & \ddots & \\ \vdots & & \end{bmatrix} = (\phi(\boldsymbol{x})\cdot\phi(\boldsymbol{y}))_{\mathcal{F}}, \tag{2.6}$$

where $\phi : \mathcal{X} \to \mathcal{F}$ is a feature map (58). The equivalence in Equation 2.6 is possible if and only if $\mathbf{K}(\boldsymbol{x}, \boldsymbol{y})$ is positive semidefinite (Mercer's Theorem), i.e.:

$$\sum_{i,j}^{N} a_i a_j \mathbf{K}(x_i, y_j) \geq 0, \forall a_i, a_j. \tag{2.7}$$

Differently from OC-SVM, SVDD uses a hypersphere to separate the data instead of a hyper-plane. This time, the smallest hypersphere with center $\boldsymbol{c}$ and radius $R > 0$ that encloses most of the data in feature space is learned. Formally, SVDD solves the following problem:

$$\begin{aligned} &\min_{R,\boldsymbol{c},\boldsymbol{\xi}} R^2 + \frac{1}{\nu n}\sum_i \xi_i \\ &s.t. ||\phi(\boldsymbol{x}_i) - \boldsymbol{c}||^2 \leqslant R^2 + \xi_i, \xi_i \geq 0, \forall i, \end{aligned} \tag{2.8}$$

where slack variables $\xi_i \geq 0$ allow a soft boundary, and the hyper-parameter $\nu \in (0, 1]$ controls the trade-off between penalties ξ_i and the volume of the hypersphere. Here, the data outside the hypersphere, i.e. $||\phi_k(\boldsymbol{x}) - \boldsymbol{c}||^2 > R^2$, are identified as anomalous. SVDD is a kernel method and admits a dual representation; interestingly, if the kernel function admits a unitary self-similarity, then OC-SVM and SVDD give the same solution. Another kernel approach has been proposed in (36), dubbed IVDD. Similarly to SVDD, it uses a hypersphere to separate the data. It also has the advantage of delivering the probability estimate for each sample (based on the distance to a hypersphere that encloses the data). This method is formally presented in Chapter 4.

Two problems arise, however, when using unmodified kernel-based approaches: they do not efficiently scale (in memory and computing time) to big data problems and, being shallow, they do not learn features representations, offloading this task to the design of the input features and the kernel function. To cope with the scaling issue, the Nyström method can represent a valid solution, also endowing regularization properties (59; 60). In particular, the Nyström approximation of a symmetric positive semidefinite (SPSD) matrix $\mathbf{K}$ is based on a sample of $m \ll n$ columns of $\mathbf{K}$ (61; 62). Let $\mathbf{C}$ denote the $n \times m$ matrix formed by these columns and $\mathbf{W}$ the $m \times m$ matrix consisting of the intersection of these m columns with the corresponding m rows of $\mathbf{K}$. The columns and rows of $\mathbf{K}$ can be rearranged based on this sampling so that $\mathbf{K}$

and $\mathbf{C}$ be written as follows:

$$\mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{K}_{21}^{\top} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{K}_{21} \end{bmatrix}. \tag{2.9}$$

Note that $\mathbf{W}$ is also a SPSD since $\mathbf{K}$ is SPSD. For a uniform sampling of the columns, the Nyström method generates a rank-k approximation $\widetilde{\mathbf{K}}$ of $\mathbf{K}$ for $k \leqslant m$ defined by:

$$\widetilde{\mathbf{K}} = \mathbf{C}\mathbf{W}_k^{+}\mathbf{C}^{\top} \approx \mathbf{K}, \tag{2.10}$$

where $\mathbf{W}_k$ is the best k-rank approximation of $\mathbf{W}$ for the Frobenius norm, that is $\mathbf{W}_k = \arg\min_{rank(\mathbf{V})=k} ||\mathbf{W} - \mathbf{V}||_F$ and $\mathbf{W}_k^{+}$ denotes the pseudo-inverse of $\mathbf{W}_k$ (63). $\mathbf{W}_k^{+}$ can be derived from the singular value decomposition (SVD) of $\mathbf{W}$, $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\top}$, where $\mathbf{U}$ is orthonormal and $\mathbf{\Sigma} = diag(\sigma_1, ..., \sigma_m)$ is a real diagonal matrix with $\sigma_1 \geq ... \geq \sigma_m \geq 0$. For $k \leqslant rank(\mathbf{W})$, it is given by $\mathbf{W}_k^{+} = \sum_{i=1}^{k} \sigma_i^{-1}\mathbf{U}_i\mathbf{U}_i^{\top}$, where $\mathbf{U}_i$ denotes the ith column of $\mathbf{U}$. Since the running time complexity of SVD is $\mathcal{O}(m^3)$ and $\mathcal{O}(nmk)$ is required for the multiplication with $\mathbf{C}$, the total complexity of the Nyström approximation computation is $\mathcal{O}(m^3 + nmk)$ (64). A Nyström-like approximation was applied to IVDD in (37), where a fast and memory-efficient version was presented. Further details are provided in Chapter 4.

Despite this significant computational improvement, coping with high-dimensional datasets still requires a feature and/or a kernel engineering effort. For this reason, DL can be a suitable alternative for the kernel layer. Indeed, DL approaches directly provide learned representations of data with multiple levels of abstraction (22). In particular, DL methods can be used to learn a low-dimensional feature representation which is then used in a disjoint and independent anomaly scoring step (65; 66). The idea behind several proposed approaches (65; 66) is that, after the training step, they have learned how to reconstruct normal input samples or represent normal samples in the latent feature space; therefore, they will fail in reconstructing anomalous samples or in generating them. In this case, the anomaly score is computed on the reconstruction error. Another option is to couple feature learning with anomaly scoring. Typical architectures used for this purpose are AEs (67), Denoising AEs

(DAEs) ([68; 69]), Generative Adversarial Networks (GANs) ([70]) or Variational AEs (VAEs) ([71]), which aim to learn a low-dimensional feature representation space in which normal and anomalous samples can be easily distinguished. Emergent and effective methods belonging to this category are, among others, AnoGAN ([72]), ADGAN ([73]), P-KDGAN ([74]), OCGAN ([75]), ICS ([76]), DROCC ([77]), and HRN ([78]). Particularly interesting is the DeepSVDD method presented in ([79]), which is a hybrid approach between the kernel-based and the DL methods. Similarly to SVDD, DeepSVDD proposes to build a hypersphere to enclose all the given class data. Still, it also uses a neural network to learn a good feature representation of the data with the one-class classification objective, simplifying the boundary creation. More formally, given a dataset, $\mathcal{D}_n = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ with $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and a neural network $\phi(\cdot; \mathcal{W}) : \mathcal{X} \to \mathcal{F}$ that maps the input into the output space $\mathcal{F} \subseteq \mathbb{R}^p$ with $L \in N$ hidden layers and the set of weights $\mathcal{W} = \{\mathbf{W}^1, ..., \mathbf{W}^L\}$ where $\mathbf{W}^\ell$ are the weights of layer $\ell \in \{1, ...L\}$, the soft-boundary DeepSVDD objective is defined as:

$$\min_{R,\mathcal{W}} R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max\{0, ||\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}||^2 - R^2\} + \frac{\lambda}{2} \sum_{\ell=1}^{L} ||\mathbf{W}^\ell||_F^2, \tag{2.11}$$

where $\phi(\boldsymbol{x}; \mathcal{W}) \in \mathcal{F}$ is the feature representation of $\boldsymbol{x} \in \mathcal{X}$ given by the network ϕ with parameters $\mathcal{W}$. Here, the network parameters $\mathcal{W}$ are learned simultaneously with the hypersphere created in the output space $\mathcal{F}$. The first term allows to minimize the volume of the hypersphere, the second term is a penalty term for points lying outside the hypersphere, and the last term is a weight decay term for the network parameters $\mathcal{W}$ with $\lambda > 0$ and $||\cdot||$ the Frobenius norm. The hyper-parameter $\nu \in (0, 1]$ controls the trade-off between the volume of the hypersphere and violations of the boundary. Authors ([79]) also propose an alternate cost function dubbed One-Class Deep SVDD:

$$\min_{\mathcal{W}} \frac{1}{n} \sum_{i=1}^{n} ||\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}||^2 + \frac{\lambda}{2} \sum_{\ell=1}^{L} ||\mathbf{W}^\ell||_F^2. \tag{2.12}$$

Here, the hypersphere is contracted by minimizing the mean distance of all data representations to the center. Even if the method is very effective compared to competing ones, it has some weaknesses. As the Authors discuss, the training process

without proper management can degenerate into trivial uninformative solutions. This is a structural problem; hence a change in the functional form would be desirable. A possible solution to this drawback is presented in (80), while in (81) a solution based on VAEs is proposed.

In this Thesis, the one-class learning domain is investigated, and a new method is presented and discussed in Chapter 4. In addition, an application of the original IVDD in the life science context is presented, showing the effectiveness and usefulness of the one-class learning strategies in this area.

2.3 Representation learning

Data representation and its learning is an important topic in the field of ML, being its role crucial for the effectiveness of any ML method. Natural language (text and speech) processing (7) and object detection (18) are just a few examples of the possible application areas of data representation learning and engineering strategies. Traditional feature engineering methodologies do not learn the representation, as this one is managed via feature and kernel engineering. These techniques are labor-intensive, possibly biased, and often limited in their representation ability. Representation learning is instead particularly effective for transfer learning processes, where the learning algorithm exploits commonalities between different learning tasks to share statistical strength and transfer knowledge across them (42). In the last years, the ML community has therefore focused on designing learned pre-processing neural layers and data transformations able to output good representations of the data to support subsequent, often supervised, learning methods (42). Among the various techniques of representations learning, DL methods allow to derive abstract representations through multiple non-linear transformations. The learning strategy can be either supervised or unsupervised and takes place, among others, employing ANNs (22), AEs (67), and SSL approaches (82). These strategies have been used in this Thesis, and the following Sections describe them in detail.

2.3.1 Artificial Neural Networks

ANNs are function approximators whose hierarchical architecture resembles brain neural connectivity. They have a high representation power, and often find and recognize patterns which are complex even for humans (22). ANNs generally comprise an input layer, an output layer, and some hidden layers which transform the input data (as vectors or structured data such as graphs) for the final processing, typically a linear combination (see Figure 2.8).

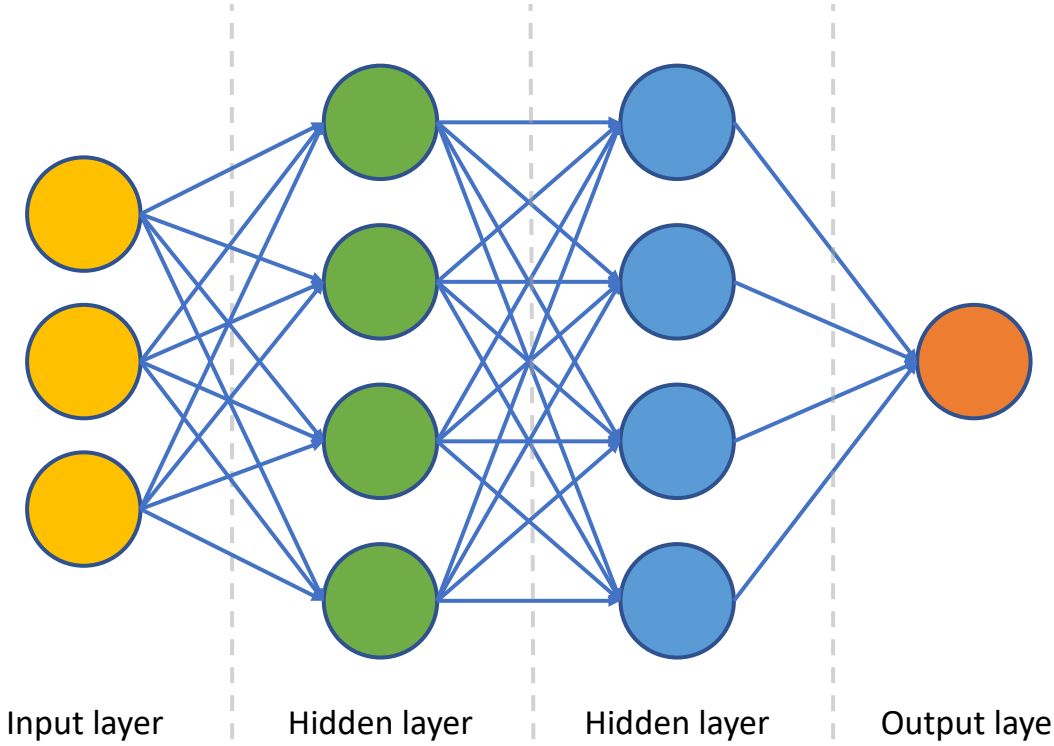

Figure 2.8: *ANN: graphical representation.*

In a standard feed-forward neural network, each layer is composed of nodes, called neurons, which perform the weighted sum of their input and feed it to a non-linear activation function (usually a sigmoid or a rectifying linear unit (ReLU)) (see Figure 2.9). Given the input $\{a_1, ..., a_n\}$, the output of the jth node of the layer ℓ is:

$$z_j^\ell = \sum_{k=1}^{m} w_{jk}^\ell f(a_k^{\ell-1}) + b_j^\ell, \tag{2.13}$$

where m is the number of neurons of the $\ell - 1$th layer, $\boldsymbol{w}_j$ is the weights vector, b_j is the bias, and the function f is the activation function. Possible activation

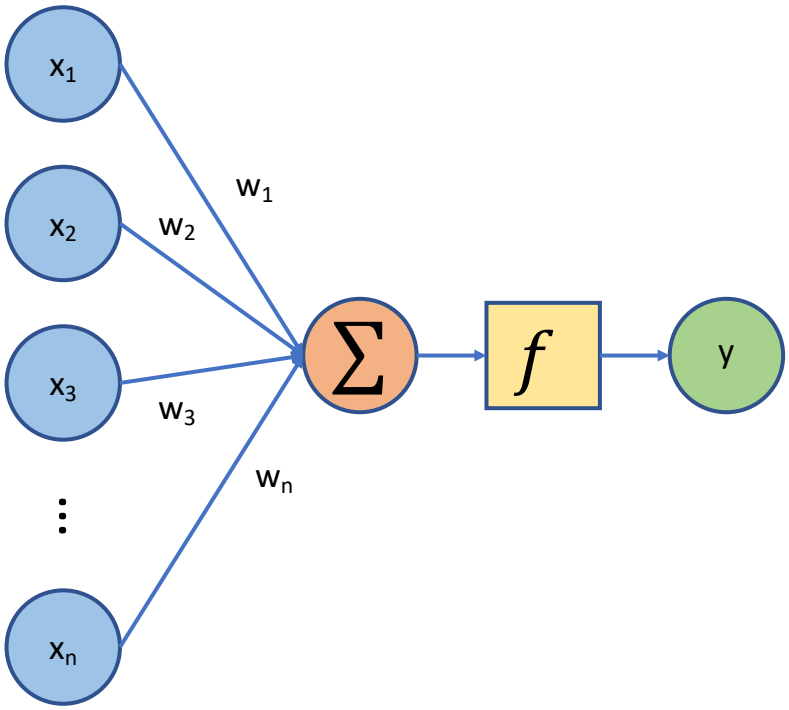

Figure 2.9: *Schematic representation of a computational unit: a neuron.*

functions are depicted in Figure 2.10. The training phase of a ANN modifies the network parameters (weights and bias) associated with all the edges to minimize a prescribed cost function. For training, the network parameters are randomly initialized; then, the input data are fed into the ANN (foreword step), and the resulting output is compared with the target to compute the error. Finally, each parameter's contribution to the error is calculated, and the parameters are adjusted accordingly using gradient descent along the network (back-propagation step (84)).

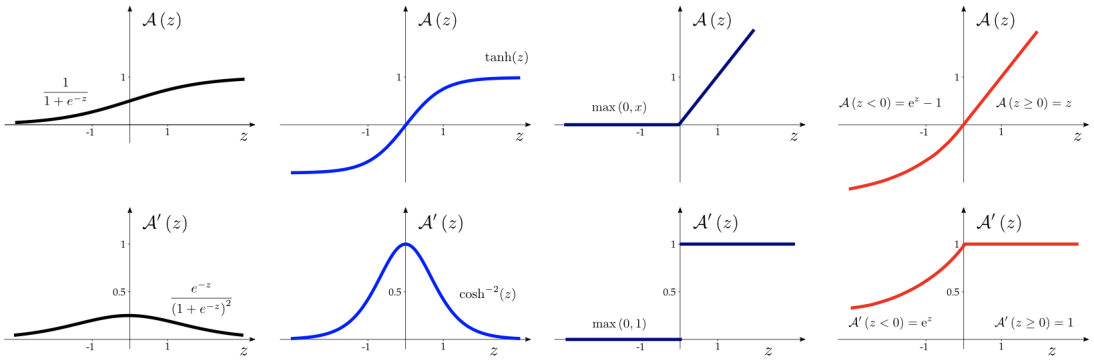

Figure 2.10: *ANN: activation functions (83).*

Given a dataset $\mathcal{D}_n = \boldsymbol{x}_1, ..., \boldsymbol{x}_n$ with $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and a neural network $\phi(\cdot; (\mathcal{W}, b)) : \mathcal{X} \to \mathcal{F}$ that maps the input into the output space $\mathcal{F} \subseteq \mathbb{R}^p$ with $L \in \mathbb{N}$ hidden layers and set of weights $\mathcal{W} = \{\mathbf{W}^1, ..., \mathbf{W}^L\}$ where $\mathbf{W}^\ell$ are the weights of layer $\ell \in \{1, ...L\}$, the total cost C is the sum a loss function $\mathcal{L}$ plus an optional regularization term:

$$C = \mathcal{L}(\phi(x; (\mathcal{W}, b)), y) + \lambda \mathcal{R}((\mathcal{W}, b))), \tag{2.14}$$

where $\phi(x; (\mathcal{W}, b))$ is the output of the network, y is the target, and $\mathcal{L}$ is the loss function (e.g., mean squared error, cross-entropy) $\mathcal{R}((\mathcal{W}, b)))$ is a regularization operator, and λ is a positive weighting factor. To minimize C by gradient descent, it is necessary to compute the partial derivative of C with respect to each weight in the network. In particular, for a single weight, the gradient is:

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l}. \tag{2.15}$$

Differentiating, one obtains:

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1}, \tag{2.16}$$

therefore:

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1}. \tag{2.17}$$

Similarly, for a single bias term the gradient is:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l}. \tag{2.18}$$

Differentiating, one obtains:

$$\frac{\partial z_j^l}{\partial b_j^l} = 1, \tag{2.19}$$

therefore:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \tag{2.20}$$

The common part in Equations 2.15 and 2.18 is called *local gradient*. It can be easily determined using the chain rule (85), and it is expressed as follows:

$$\delta_l^l = \frac{\partial C}{\partial z_j^l}. \tag{2.21}$$

Finally, the network parameters are updated as per:

$$\begin{aligned} w_k^\ell(t+1) &= w^\ell(t) - \eta \frac{\partial C}{\partial w_k^\ell} \\ b^\ell(t+1) &= b^\ell(t) - \eta \frac{\partial C}{\partial b^\ell}, \end{aligned} \tag{2.22}$$

where η is the learning rate and determines the gradient descent speed. The forward step and the backpropagation steps are iterated (in the neural jargon, these steps are called epochs) to reduce at each step the loss until the convergence is reached (which is often assessed by checking the gradient norm).

Feed-forward neural networks of the just described kind are called Multi-Layer Perceptrons (MLP) (86) and are one of the possible architectures. CNNs are another type of ANNs, particularly effective in large-scale image recognition, with layers of convolving filters applied to local features (87). Convolutional layers are followed by pooling layers, forming modules. These are followed by fully connected layers (MLP), as in the standard feed-forward neural networks (see Figure 2.11). Convolutional layers comprise neurons arranged in feature maps. Each neuron has a receptive field connected to a neighborhood of neurons in the previous layer by a set of trainable weights (kernel/filter). Input images are convolved with the kernel to create a

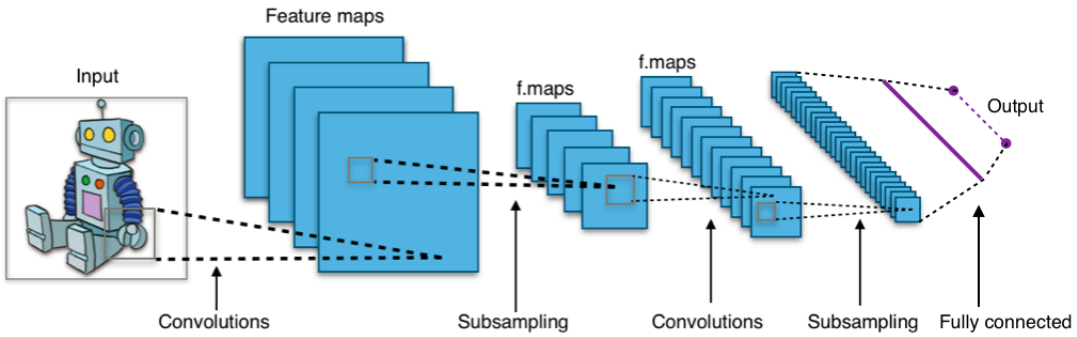

Figure 2.11: *CNN: graphical representation (88).*

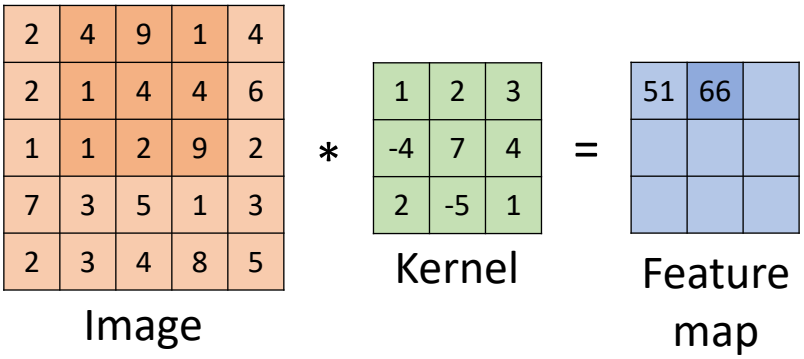

Figure 2.12: *CNN: convolution graphically explained.* The kernel window is shifted along the input image pixel by pixel. For each window position, one convolution of the kernel with the corresponding portion of the input image is computed and the result populates the feature map (see Equation 2.22).

new feature map (see Figure 2.12). The convolved results are then sent through a non-linear activation function (89). Subsequent feature map values are computed as:

$$G[m,n] = f\left((W * \boldsymbol{x})[m,n]\right) = f\left(\sum_{j=0}^{J-1}\sum_{k=0}^{K-1} W[j,k]\boldsymbol{x}[m+j,n+k]\right), \tag{2.23}$$

where $\boldsymbol{x}$ denotes the input image, W denotes the kernel (which is a $J \times K$ matrix), $*$ denotes the convolution, and $f(\cdot)$ the non-linear activation function (89). Rows and columns indices of the result matrix are m and n respectively. Pooling layers reduce the spatial resolution of the feature maps; max pooling aggregation layers are typically used. They propagate the maximum value to the next layer within the receptive field (89). Finally, fully connected layers flatten the intermediate maps to get the feature representation. The softmax operator is usually used for the classification task as the very last neuron for each class (89):

$$\sigma(\boldsymbol{z}) = \frac{e^{z_i}}{\sum_{j=1} K e^{z_j}}, \tag{2.24}$$

where $\boldsymbol{z}$ is the output vector. In this way, a vector $\boldsymbol{z}$ of K real values can be transformed into a vector of K real values $\in [0,1]$ that sum to 1 and can be interpreted as probabilities. The label predicted by the model can be easily obtained as the index of the output vector with the maximum probability.

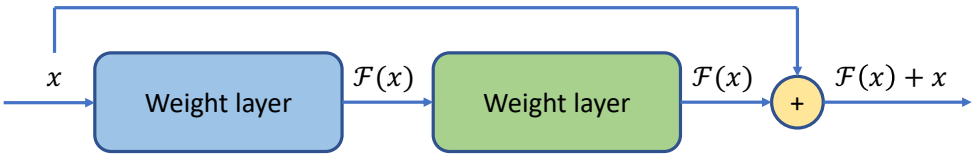

Figure 2.13: *Residual block: graphical representation.*

Residual CNNs (e.g., ResNet-50 ([90])) are a type of CNNs that introduce the deep residual learning framework blocks. In particular, if one denotes by $H(x)$ the desired underlying mapping of a few stacked layers (with x being the input to the first of these layers) and hypothesizes that multiple non-linear layers can approximate any function, one can then assume that the residual functions $H(x) - x$ can be learned correctly. The stacked layers can thus approximate a residual function $F(x) = H(x) - x$ and the original function becomes $F(x) + x$. CNNs can realize the latter formulation with shortcut connections, which perform identity mapping. Their output is then added to the output of the stacked layers (see Figure 2.13). This type of CNNs produces substantially better prediction results than previous networks, is deeper, and can deal better with the vanishing gradient issue ([90; 91]).

In this Thesis, ANNs, particularly CNNs, are used for learning a low-dimensional feature representation of the input, which is used, in a second step, for manifold learning experiments. Full details on these experiments are described in Chapter 3.

2.3.2 Autoencoders

AEs ([67]) are a specific type of ANNs, where the output is imposed to be equal to the input via an ad-hoc loss function. They are a deep solution to the dimensionality reduction problem and are capable to learn a data's compressed representation (latent-space representation). From this compressed representation, they can approximately reconstruct the input .

An AE consists of 3 components: an encoder, a bottleneck, and a decoder layer. The encoder compresses the input and produces the bottleneck; the decoder reconstructs the input only using the bottleneck features (see Figure 2.14). Both the encoder and decoder are ANNs, and the loss used during the training phase is the reconstruction

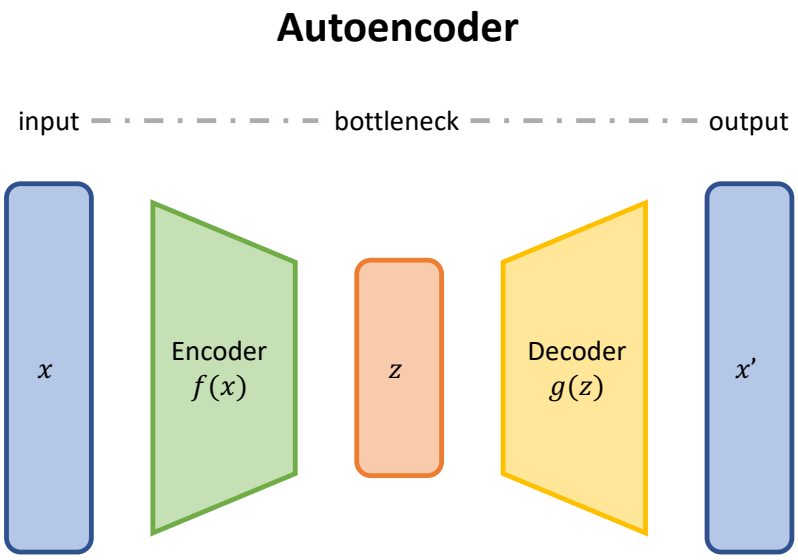

Figure 2.14: *AE: graphical representation.*

error (e.g., mean squared error):

$$\mathcal{L}_{AE} = \mathcal{L}_{rec}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i), \tag{2.25}$$

where $\boldsymbol{x}_i$ is the input sample and $\hat{\boldsymbol{x}}_i$ is the output of the architecture.

Variational AEs (VAEs) are a particular type of AEs proposed in (71). They formalize the learning process in probabilistic terms. In particular, given a dataset, $\mathcal{D}_n = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, the encoder (or recognition model) $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ with parameters ϕ, maps the data $\boldsymbol{x}$ into a probability distribution (e.g., Gaussian) over the possible values of the latent representation $\boldsymbol{z}$. It approximates the intractable true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. The decoder, $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, with parameters θ, is a generative model that, given the latent representation $\boldsymbol{z}$, produces a distribution over the possible corresponding values of $\boldsymbol{x}$ (see Figure 2.15). The marginal likelihood is composed by a sum over the marginal likelihoods of individual datapoints:

$$\log p_\theta(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) = \sum_{i=1}^{n} \log p_\theta(\boldsymbol{x}_i), \tag{2.26}$$

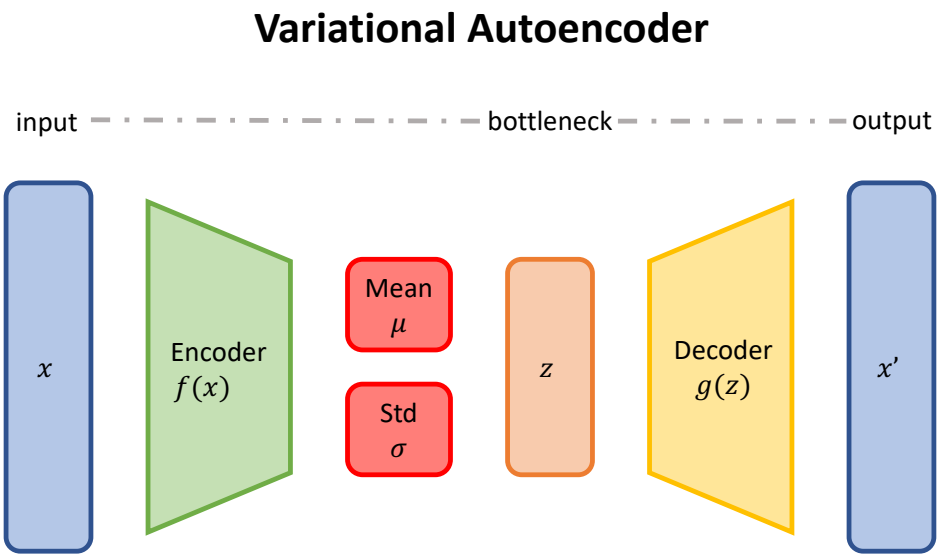

Figure 2.15: *VAE: graphical representation.*

and can be rewritten as:

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)}\left[\log p_\theta(\boldsymbol{x})\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right]\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right]\right] \\
= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]\right] &+ \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right]\right] \\
&= \mathcal{L}(\theta,\phi,\boldsymbol{x}) + D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})).
\end{aligned} \tag{2.27}$$

The term D_{KL} is the KL divergence, and measures the distance between the approximated probability distributions $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, while the term $\mathcal{L}(\theta,\phi,\boldsymbol{x})$ is the variational lower bound, also called evidence lower bound (ELBO):

$$\mathcal{L}(\theta,\phi,\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)}[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})]. \tag{2.28}$$

Since the KL divergence is non-negative, the ELBO is a lower bound on the log-likelihood of the data:

$$\begin{aligned}
\mathcal{L}(\theta,\phi,\boldsymbol{x}) &= \log p_\theta(\boldsymbol{x}) - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})) \\
= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x})\right] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})) &\leq \log p_\theta(\boldsymbol{x}).
\end{aligned} \tag{2.29}$$

Differentiating $\mathcal{L}(\theta, \phi, \boldsymbol{x})$ thrugh the usual Monte Carlo gradient estimator is impractical in this case, for this reason the reparametrizion trick is used. More generally, for an approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, the random variable $\widetilde{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$ can be reparametrized using a differentiable transformation $g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x})$ of an auxiliary noise variable $\boldsymbol{\epsilon}$:

$$\widetilde{z} = g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x}), \tag{2.30}$$

with $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. The Monte Carlo approximation of the expectation of some function $f(\boldsymbol{z})$ w.r.t. $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ can be estimated as:

$$\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[f(\boldsymbol{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x}))\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\boldsymbol{\epsilon}_l, \boldsymbol{x})), \tag{2.31}$$

where $\boldsymbol{\epsilon}_l \sim p(\boldsymbol{\epsilon})$. Applying this technique to the variational lower bound in Equation 2.29, a first Stochastic Gradient Variational Bayes (SGVB) can be obtained as:

$$\begin{aligned} \widetilde{\mathcal{L}}^A(\theta, \phi, \boldsymbol{x}) &\simeq \mathcal{L}(\theta, \phi, \boldsymbol{x}) \\ \widetilde{\mathcal{L}}^A(\theta, \phi, \boldsymbol{x}) &= \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\boldsymbol{x}, \boldsymbol{z}_l) - \log q_\phi(\boldsymbol{z}_l|\boldsymbol{x}), \end{aligned} \tag{2.32}$$

where $\boldsymbol{z}_l = g_\phi(\boldsymbol{\epsilon}_l, \boldsymbol{x})$ and $\boldsymbol{\epsilon}_l \sim p(\boldsymbol{\epsilon})$. The KL divergence can be integrated analytically and can be interpreted as a regularizer, since it encourages the approximate posterior to be close to the prior $p_\theta(\boldsymbol{z})$. Therefore, another SGVB can be obtained as:

$$\widetilde{\mathcal{L}}^B(\theta, \phi, \boldsymbol{x}) = \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(\boldsymbol{x}, \boldsymbol{z}_l)) - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})), \tag{2.33}$$

where $\boldsymbol{z}_l = g_\phi(\boldsymbol{\epsilon}_l, \boldsymbol{x})$ and $\boldsymbol{\epsilon}_l \sim p(\boldsymbol{\epsilon})$. Equation 2.33 is clearly connected with auto-encoders: the first term is a negative reconstruction error and the second one is a regularizer. Typically, the probability distributions $p_\theta(\boldsymbol{z})$ and $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ are assumed to be multivariate Gaussian:

$$\begin{aligned} p_\theta(\boldsymbol{z}) &= \mathcal{N}(\boldsymbol{z}, 0, \mathbf{I}) \\ q_\phi(\boldsymbol{z}|\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{z}, \mu, \sigma^2), \end{aligned} \tag{2.34}$$

where the mean and the standard deviation μ and σ are output of the encoder network. Further details can be found in ([92]; [93]).

In this Thesis, VAEs are used for learning a low-dimensional representation effective for one-class learning. Since the prior distribution $p_\theta(z)$ is assumed to follow a Gaussian distribution, the representation obtained with VAEs should make distinguishing normal and anomalous samples easier when a hypersphere is learned. Further details on this topic are presented in Chapter [4].

2.3.3 Self-supervised learning strategies

SSL strategies provide an appealing solution to the representation learning problem. They are particularly effective in the field of natural language processing (NLP), and examples of their effectiveness are, among others, the Collobert-Weston 2008 model ([94]), Word2Vec ([95]), GloVE ([96]), fastText ([97]), and, more recently, BERT ([98]), RoBERTa ([99]), XLM-R ([100]). Systems pretrained with self-supervised techniques yield considerably higher performance than when solely trained in a supervised manner. This progress has also led to using SSL strategies in complex real-world settings CV problems ([101]).

Self-supervised can be imagined as an energy-based modeling attempt (EBM) (see Figure [2.16]). A trainable system is given by two input, x and y, and it learns their level of incompatibility (energy). Low energy values mean that the two input are compatible, whereas high energy values indicate that the two input are incompatible.

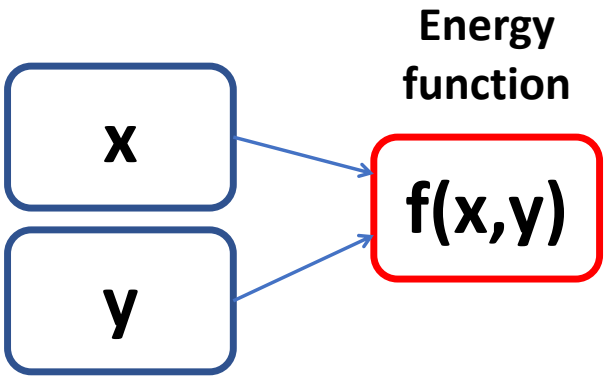

Figure 2.16: *SSL: an EBM.* Given two different versions, x and y, of an image, an energy function is computed (e.g., similarity) in order to estimate the level of compatibility of the two input.

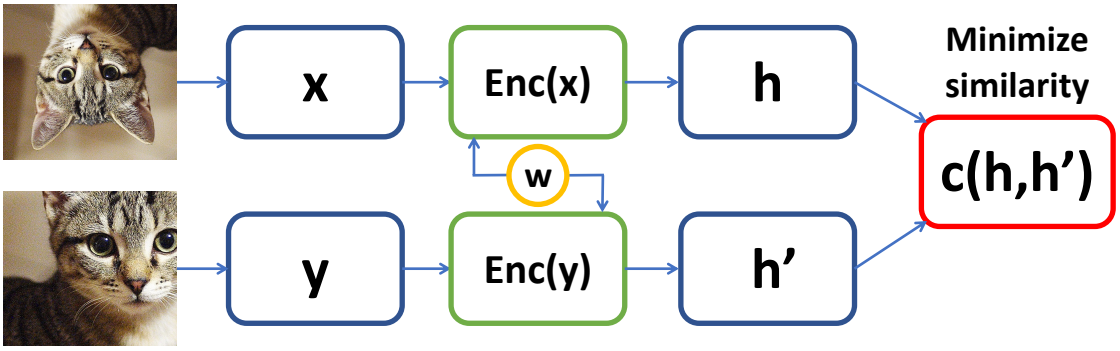

Figure 2.17: *SSL: Siamese architecture.*

During the training phase, the system considers only compatible samples and is trained to produce low energy for compatible samples and high energy for incompatible ones. This last aspect is the most challenging part of the training procedure. For image recognition problems, the input images are generally distorted to obtain two different versions, x and y, from which the model can be trained.

A well-suited DL architecture to implement this mechanism is called Siamese network or joint embedding architecture, initially introduced in the literature in the early 1990s and mid-2000s ([102]; [103]; [104]; [105]; [68]) (see Figure 2.17). It is composed of two identical networks that output two different representations (embeddings) given as input x and y (two different versions of the same image). In this case, the energy is computed as the distance between the two embedding vectors. At the same time, the weights are adjusted during the training phase so as to minimize the energy and produce similar representations for the two input.

This type of architectures have attracted the SSL community in the last years since they do not require labels during the learning phase and take advantage of all the pros of unsupervised learning approaches. The main challenge when the Siamese architecture is trained is making the network capable of outputting high energy values when incompatible samples are used as input, and impeding the network from producing an undesired trivial solution in which all output collapse to a constant. Collapse happens when the network is not capable of distinguishing different samples. Some recent results seem promising, even if the domain remains largely unexplored. Among others, the most interesting solutions include DeeperCluster ([106]), ClusterFit ([107]), MoCo-v2 ([108]), SwAV ([109]), SimSiam ([110]), Barlow Twins ([111]), and BYOL

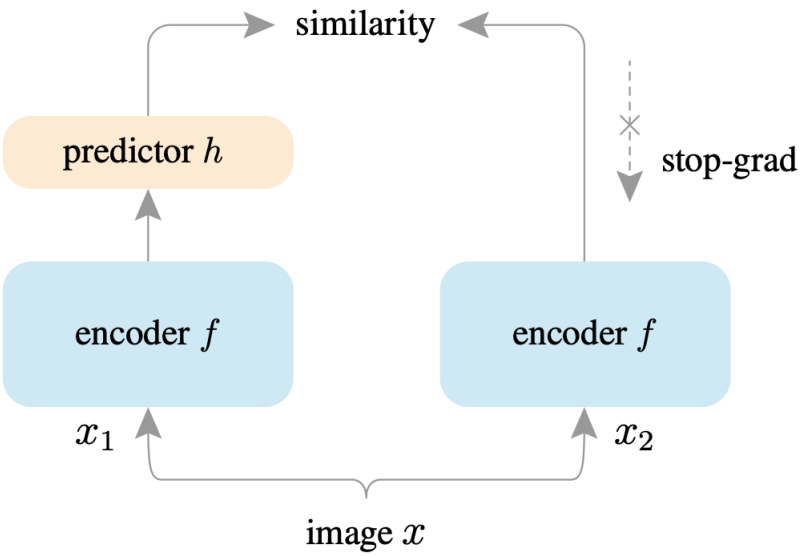

Figure 2.18: *SSL: a Siamese solution (110).*

(112). These solutions propose different strategies and tricks (e.g., computing virtual target embedding) to prevent the network from collapsing. Particularly interesting is the solution presented in (110). As in a standard Siamese approach, the network takes as input two randomly augmented views, x_1 and x_2, from an image. The network comprises an encoder f (a backbone architecture concatenated to a projection MLP) and a prediction MLP h. The two output vectors of the architecture can be defined as $p_1 = h(f(x_1))$ and $z_2 = f(x_2)$ (see Figure 2.18 for further details on the structure of the network), and the problem to minimize is defined as follows:

$$\mathcal{H}(p_1, z_2) = -\frac{p_1}{||p_1||_2} \cdot \frac{z_2}{||z_2||_2}, \tag{2.35}$$

where $||\cdot||_2$ is ℓ_2-norm, and $\mathcal{H}$ is the negative cosine similarity. The symmetrized loss can be therefore defined as:

$$\mathcal{L} = \frac{1}{2}\mathcal{H}(p_1, z_2) + \frac{1}{2}\mathcal{H}(p_2, z_1). \tag{2.36}$$

This loss is defined for each image and averaged. The minimum value is -1. To prevent the network from collapsing, the Authors of the method proposed in (110) a

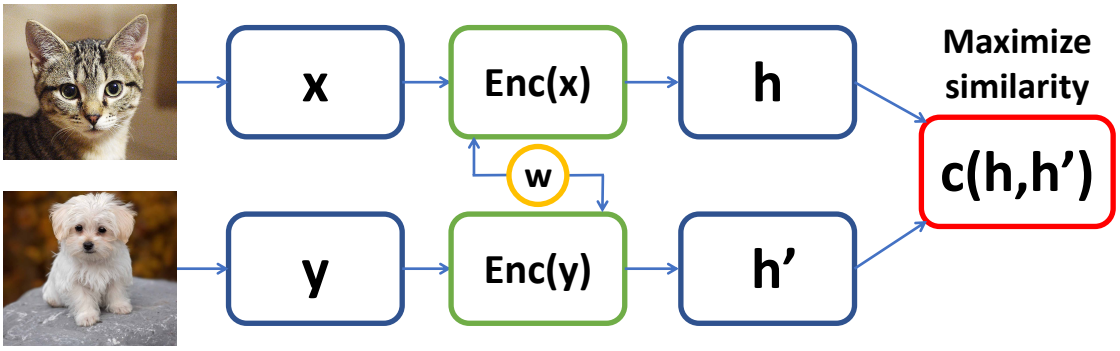

Figure 2.19: *SSL: contrastive architecture.*

stop-gradient strategy, which consists in considering z as a constant:

$$\mathcal{L} = \frac{1}{2}\mathcal{H}(p_1, stopgrad(z_2)) + \frac{1}{2}\mathcal{H}(p_2, stopgrad(z_1)). \tag{2.37}$$

In addition to Siamese architectures, contrastive (see Figure 2.19) and regularization methods have also been devised to prevent collapse. Contrastive learning methods are self-supervised strategies that allow learning a high-level feature representation considering similarities and dissimilarities among data points. They are supervised approaches for learning representations. Particularly, given an image, one can compute two augmented combinations (e.g., cropping, resizing, and recoloring) and train a model to maximize the similarity of the two corresponding vector representations (one for each augmented image). When, at training time, a dataset composed of different classes is used (e.g., images of cats and dogs), the model will learn that images belonging to the same category should have an equal representation that is different when images belong to another category. More generally, let $\phi(x) = normalize(g(x))$, i.e. $||\phi(x)|| = 1$,

$$\mathcal{L}_{clr} = -\mathbb{E}_{x,x_i \sim P_{\mathcal{X}},\mathcal{A},\mathcal{A}'}\left[\log \frac{sim(\mathcal{A}(x), \mathcal{A}'(x))}{sim(\mathcal{A}(x), \mathcal{A}'(x)) + \sum_{i=1}^{M-1} diff(\mathcal{A}(x), \mathcal{A}(x_i))}\right], \tag{2.38}$$

where $\mathcal{A}$ and $\mathcal{A}'$ are identical but independently augmented versions of the image x, $sim(\mathcal{A}(x), \mathcal{A}'(x))$ is:

$$\exp\left(\frac{1}{\tau}\phi(\mathcal{A}(x))^\top \phi(\mathcal{A}'(x))\right), \tag{2.39}$$

and $diff(\mathcal{A}(x), \mathcal{A}(x_i))$ is:

$$\exp\left(\frac{1}{\tau}\phi(\mathcal{A}(x))^\top \phi(\mathcal{A}(x_i))\right). \tag{2.40}$$

$\mathcal{L}_{clr}$ regularizes representations of the same instance with different views $(\mathcal{A}(x), \mathcal{A}'(x))$ to be similar, while those of other instances $(\mathcal{A}(x), \mathcal{A}(x_i))$ to be unlike.

Contrastive approaches have proven effective for multi-class classification ([114]; [115]). However they require labels during training. Recently, they have been adopted for solving unsupervised problems, particularly one-class learning problems, by introducing the *distribution augmentation* concept ([113]). In more detail, instead of modeling the training data distribution $P_{\mathcal{X}}$, the union of augmented training distribution is considered, that is $P_{\bigcup_a a(\mathcal{X})}$ where $a(\mathcal{X}) = \{a(x)|x \in \mathcal{X}\}$. The augmented training distribution is obtained by using augmentation strategies to the training set, which differ from those used for generating different views of the same image $\mathcal{A}$. Images augmented with these strategies are considered negative samples and are encouraged to be distant from a positive sample in the representation space. An example of this approach is depicted in Figure 2.20.

In this Thesis, the solutions presented in ([110]) and ([113]) have been adopted as representation learning strategies for the one-class learning domain. Further details are discussed in Chapter 4.

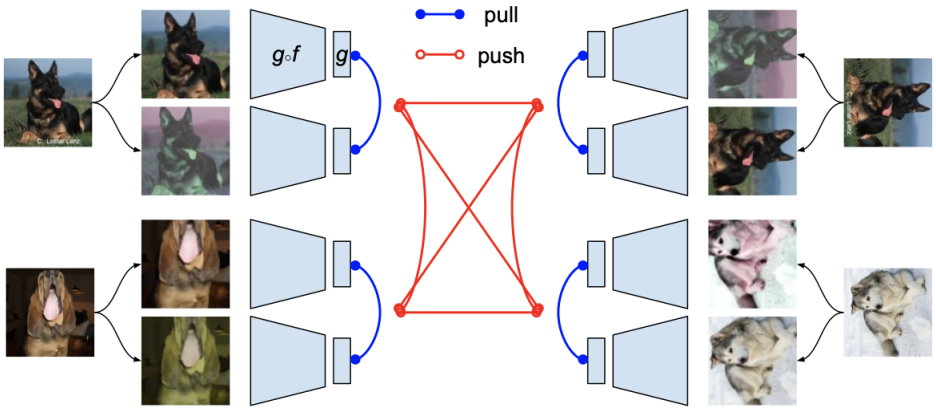

Figure 2.20: *SSL: distribution augmentation concept ([113]).*

Chapter 3

Principal Path algorithm: applications and variants

This Chapter introduces the PP algorithm ([35]), its applications and the improvements of the method developed in this Thesis. Particularly, two different experiments are described, aiming to prove its usefulness in the life science context, and its flexibility. These works have been published respectively in ([116]) and ([117]). Finally, this Chapter presents an improved version of the PP, which modifies some key steps of the algorithm and makes the method more robust, stabler, and ab-initio local by removing some drawbacks of the original version. This version of the PP algorithm has been published in ([118]).

The following notations will be used throughout the Chapter:

- Vectors are underlined, e.g., $\underline{v}$, while matrices are bold and capitalized, e.g., $\mathbf{M}$.
- N is the number of samples.
- N_c is the number of waypoints.
- A dot $\cdot$ is used as column or row index to represent the entire set of columns or rows, respectively, e.g., $\mathbf{M}_{i,\cdot} = \underline{m}_i$ is a vector representing the i-th row of the matrix M.
- $\mathbf{X}$ is the $N \times d$ matrix of samples $\underline{x}_i$ arranged in a rowwise fashion, i.e. $\mathbf{X}_{i,\cdot} = \underline{x}_i$.

- $\mathbf{W}$ is the $N_c \times d'$ matrix of waypoints $\underline{w}_i$ arranged in a rowwise fashion.
- $\underline{w}_0$ and $\underline{w}_{N_c+1}$ represent the starting and ending points of the path.
- $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d'}$ is the, possibly non-linear, transformation mapping the d-dimensional input space into a d'-dimensional transformed one. Underlining of the $\phi(\cdot)$ symbol is omitted for simplicity.
- $|w_i|$ represents the number of samples associated with the ith waypoints (i.e., cardinality of the ith cluster).
- $u_i \in [1, N_c] \cap \mathbb{N}$ is the label associated with the ith sample $\underline{x}_i$.
- $\delta(u_i, j)$ is the Kronecker delta.

3.1 Principal Path algorithm: the method

The PP algorithm (35; 119) is a one-dimensional manifold learning algorithm, which is particularly well suited to capture time-dependent trends in data (e.g., for dynamical systems, genomic data, cancer evolution, ...). It is a local version of the Principal Curve (34). The term *local* means that the PP does not necessarily consider all the data points for its solution. Unlike the Dijkstra shortest path (120) and more similarly to the minimum free-energy path in statistical mechanics, the PP is smooth. The smoothness is achieved through a regularized loss function (50). Formally, the PP is implemented as an extension of k-means clustering (121; 122), where the first and last clusters are constrained (i.e., w_0 and w_{NC+1}) and the other clusters are evolved according to the regular k-means cost function plus a regularization term, which induces a string topology. All the clusters are waypoints for the path and are topologically connected by a chain of springs. The PP cost function is:

$$\min_{W,u} \sum_{i=1}^{N} \sum_{j=1}^{N_c} \|\phi(x_i) - w_j\|^2 \delta(u_i, j) + s \sum_{i=0}^{N_c} \|w_{i+1} - w_i\|^2, \tag{3.1}$$

where the first term represents the k-means objective (with the cluster corresponding to the starting and the ending points kept fixed) and the second term is the regularization part. The regularization parameter s regulates the trade-off between the

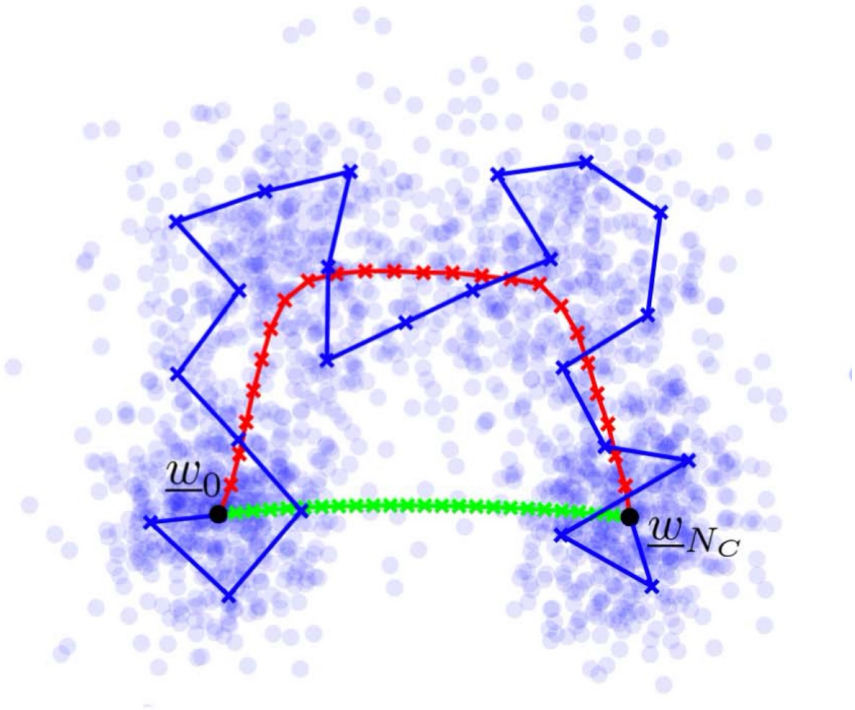

Figure 3.1: *The PP algorithm: pictorial description of s tradeoff.* If $s = 0$ (blue), the resulting path is irregular, showing an overfitting effect, if $s \sim \infty$ (green), the resulting path is a straight line connecting the boundary points. Meaningful paths are obtained with intermediate values of s.

data fitting and the smoothness of the inferred path (see Figure 3.1), while ϕ is a possibly non-linear map.

The PP algorithm is based on the following steps:

1. *Set the boundary condition*: the starting and the ending points, $\underline{w}_0$ and $\underline{w}_{N_c+1}$, are chosen among the samples.

2. *Prefilter the data* (optional, see Figure 3.2): to stabilize the method, the algorithm can consider just a subset of samples, which can be computed as follows:

   - *Medoids extraction*: a set of N_f medoids $\mathbf{M} = \underline{m}_i \subset \mathbf{X}$ is selected via the k-means++ algorithm ([123]), where N_f is a user-defined parameter.

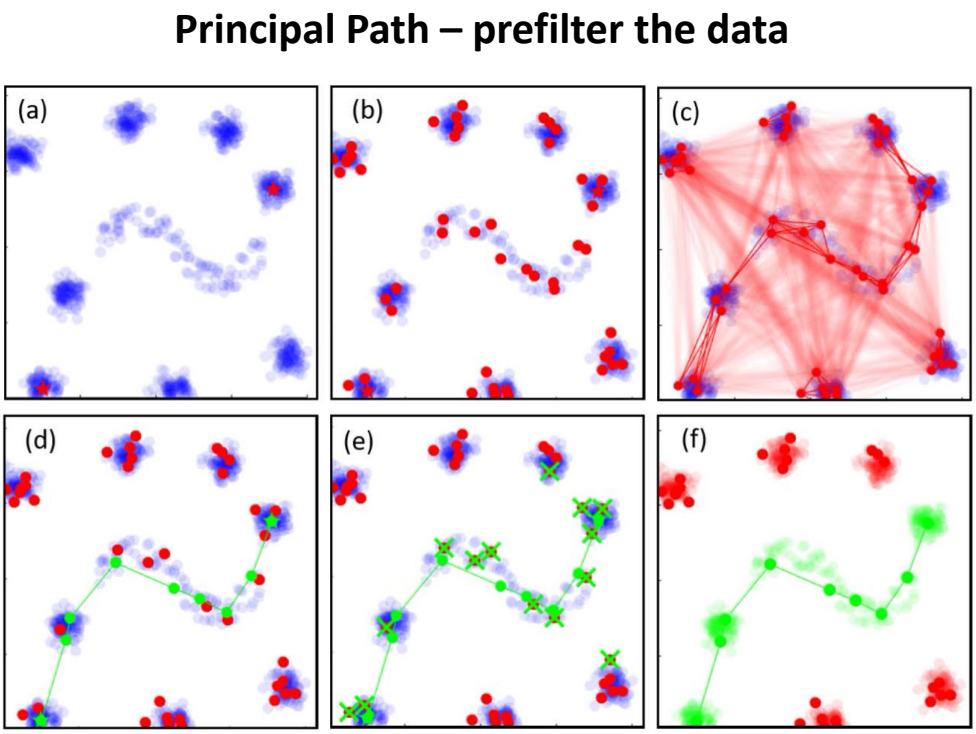

Figure 3.2: *PP algorithm: prefilter the data.* (a) Set the boundary condition. (b) Medoids extraction. (c) Build a penalized graph. (d) Find the shortest path. (e) Medoids selection. (f) Filter the samples. Image from (35).

- *Build a penalized graph*: first, the distance matrix is computed among the medoids $\mathbf{M}$. In this case, the squared Euclidean distances are considered. Then, the distance matrix is penalized according to the neighborhood of each medoid $\mathbf{M}$ with a penalty factor of p. For each medoid, the first k nearest points are considered as its neighborhood, where k is a user-defined parameter. The final distance matrix can be summarized as follows:

$$d_p^2(\underline{m}_i, \underline{m}_j) = \begin{cases} d^2(\underline{m}_i, \underline{m}_j), & \underline{m}_i \in \mathrm{nn}_k(\underline{m}_j) \\ pd^2(\underline{m}_i, \underline{m}_j), & \text{otherwise.} \end{cases}$$

  Note that the weight d_p represents the weight of the edge connecting the nodes $\underline{m}_i$ and $\underline{m}_j$ in the final penalized graph.

- *Find the shortest path*: the Dijkstra algorithm (120) is used to find the shortest path connecting the boundaries $\underline{w}_0$ and $\underline{w}_{N_c+1}$ and walking

through an ordered subset of medoids $\mathbf{S} \subset \mathbf{M}$.

- *Medoids selection*: the medoids $\mathbf{M}$ too close to $\mathbf{S}$, according to a user-defined threshold T, are removed (here identified as $\mathbf{Q}$). A final subset of medoids $\mathbf{S}_{final} = \mathbf{S} \cup (\mathbf{M} - \mathbf{Q})$ is obtained.
- *Filter the samples*: each sample of the input data matrix $\mathbf{X}$ is labeled according to the nearest medoids $\mathbf{S}_{final}$, and the subset of data N_k associated with the medoids in $\mathbf{S}$ is kept.

3. *Init the medoids*: a set of N_c medoids are selected via the k-means++ algorithm ([123]), representing the waypoints' initial configuration.

4. *Optimize the cost function*: the cost function is optimized via an Expectation Maximization (EM) algorithm ([124]), which can be summarized with the following steps:

   - *E-step*: for each iteration t of the optimization process, it computes the labels (or memberships) as in the standard k-means algorithm:

$$u_{i,t+1} = \arg\min_j \|x_i - w_{j,t}\|^2. \tag{3.2}$$

   - *M-step*: based on the newly computed $u_{i,t+1}$, it updates the waypoints matrix W as follows:

$$W_{t+1} = (A_X(u_t) + sA_W)^{-1}(C + \frac{s}{2}B), \tag{3.3}$$

   where:

– $B_{i,\cdot}$ is the $N_c \times d'$ boundary conditions matrix:

$$B_{i,\cdot} = \begin{cases} w_0 & \text{if } i = 1 \\ w_{N_c+1} & \text{if } i = N_c \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

– $C_{i,\cdot}$ is the $N_c \times d'$ centroid matrix:

$$C_{i,\cdot} = \sum_{j=1}^{N} x_j \delta(u_i, j) \tag{3.5}$$

– A_W is the Toeplitz matrix representing the hessian of the regularization term in the cost function:

$$A_W = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & \dots & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & \dots & 0 & -\frac{1}{2} & 1 \end{bmatrix} \tag{3.6}$$

– $A_{X_{i,j}}$ is the $N_c \times N_c$ hessian matrix of the standard k-means cost function:

$$A_{X_{i,j}} = \begin{cases} |w_i| & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

– s is the regularization parameter.

These steps are repeated until the convergence condition ($u_{i,t} = u_{i,t-1}$) is satisfied for every i. The optimization is repeated, giving as input different values of the regularization parameter s. For this reason, different final configurations of the waypoints are obtained in this step, one for each value of s, and a model selection strategy is required. Among others, a fast and straightforward model selection strategy can be devised as it follows. Given a path W, let $l_{i,i+1} = Dst(w_i, w_{i+1})$ be the length of the segment connecting two subsequent waypoints w_i, w_{i+1}, and let σ be the variance of such quantity along the path.

For large values of s, σ is expected to be equal to 0, since the algorithm asymptotically finds the trivial path connecting w_0 to w_{N_c+1} with an evenly sampled straight line. Instead, for a small value of s, the algorithm finds an arbitrarily tortuous path, usually leading to high values of σ and boiling down to a classical k-means result. For each experiment, the best path W_{best} can be therefore selected with the elbow criterion (125) on the plot of σ vs. $\log(s)$ (see Figure 3.3). This simple heuristic can select non-trivial paths with similarly spaced waypoints which seamlessly translate into evenly sampled morphing processes. Another model selection strategy can be performed via Bayesian Evidence Maximization. Further details on this model selection strategy can be found in (35). Note that when the prefiltering procedure is used, the cost function to optimize becomes:

$$\min_{W,u} \sum_{i=1}^{N_k} \sum_{j=1}^{N_c} \|\phi(x_i) - w_j\|^2 \delta(u_i, j) + s \sum_{i=0}^{N_c} \|w_{i+1} - w_i\|^2, \tag{3.8}$$

since just the subset of N_k samples (not filtered out) is considered.

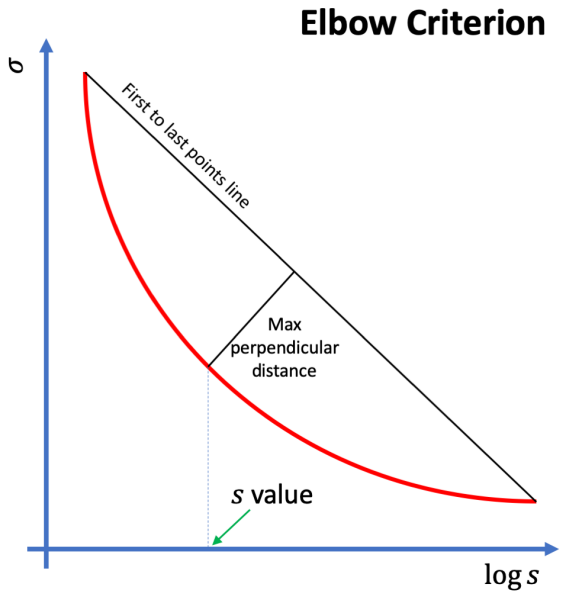

Figure 3.3: *PP algorithm: model selection with elbow criterion.* For each value of $\log(s)$ one can obtain a PP and the corresponding variance (σ) on waypoints interdistance for that path. The elbow is the furthest point from the line which connects the first and the last points.

3.2 Principal Path algorithm: applications

In this Section two applications are presented, which involve the just discussed PP method. The first one shows how the PP can be used in the life science domain and describes a package that makes the PP available to users through high-level APIs. The second application aims to challenge the PP to measure its strengths and limitations in a context where the ground truth is known.

3.2.1 Principal Path applied to Life Sciences

In the previous Section, the PP has been presented as a general method that can be adopted in any context where it makes sense to reason in terms of evolutionary connections between data points or a pseudo-time that connects them in succession. One interesting application domain for this method is omic/biological datasets (e.g., transcriptomics or metabolomics). In fact, they can be used for analyzing time-dependent phenomena, such as tumor evolution, cell cycle, cell differentiation, and organogenesis. The recent advent of next-generation sequencing (NGS) techniques has produced a massive amount of high-throughput data for quantitative biology, especially in the field of transcriptomics. This allows to obtain increased depth and sample size of transcript measurements making possible the usage of ML algorithms for analyzing these highly complex datasets and increasing the understanding of biological phenomena (126). Examples of these complex transcriptomic datasets are those generated by the TCGA (127) and GTEX (128) consortia, which collect tens of thousands of human RNA-Seq samples from tumor and physiological tissues, respectively. In the past few years, the development of single-cell sequencing technologies has further increased the sample size of RNA-Seq datasets, albeit at at the cost of transcript coverage (129). RNA-Seq analyses to understand changes in gene expression have built on the previous generation of technological platforms (microarrays), and have focused on characterizing the quantitative differences between two or more groups of samples, a process known as differential gene expression analysis (DGEA). As the sample sizes increase, so does the ability to detect and study the natural heterogeneity of living systems, whether they are bulk tissues (e.g., interpatient variance

in cancer) or single cells (e.g., different cells and cell states in a microenvironment). Moreover, large datasets allow to measure biological transition processes such as cell differentiation and tumor progression (130). The literature contains several studies on defining cell trajectories (131) (i.e., Monocle (132; 133; 134)); however, there are not so many general and flexible algorithms/packages for modeling continuous processes and extracting the associated features (i.e., genes or transcripts).
This Section introduces a novel R package, dubbed *spathial* (117). *Spathial* is an easy-to-use implementation of the PP algorithm. It allows the analysis of progressions in large-scale transcriptomic datasets, such as those arising from bulk and single-cell RNA-Seq.

Software package

The package *spathial*[1],[2] allows to run the PP algorithm using very high-level functions. In particular:

- *spathialBoundaryIds*: allows the selection of the boundaries (start and end points). *spathial* provides three options: a visual selection by the user, classes centroids, or selection of the samples using their row-name.

- *spathialPrefiltering* (optional): allows obtaining a local solution, which does not involve the entire dataset. This procedure removes some data points and forces the PP algorithm to go through a restricted number of samples. This can create smoother paths but, at the same time, can prune some available data;

- *spathialWay*: run the PP algorithm with the boundaries selected during the first step and with the input data (filtered or not filtered).

The function *spathialWay* is the core of the package *spathial*. It runs the PP algorithm several times (according to the s domain) given the boundaries, performs the model selection, and returns the best PP.
The choice in *spathial* to automatically perform model selection allows to obtain

[1]https://cran.r-project.org/web/packages/spathial/index.htm
[2]https://github.com/erikagardini/spathial

the best performing PP in few simple steps and also increases reproducibility. This, however, reduces the performance and increases the execution time, as picking one single s value would be faster yet more user-dependent.

The PP algorithm has two main computational steps: the centroid matrix C calculation, where one needs to compute the mean of the points in each cluster (each point has to be considered once, which implies that this step has complexity $\mathcal{O}(Nd')$) and the *E-step*, where one computes the distances between all points and waypoints to find the closest one in each iteration. The latter is of complexity $\mathcal{O}(Nd'N_c)$, and if the input matrix X is large, it prevents the algorithm from being scalable to large datasets and high dimensions. For this reason, the *spathial* package is optimized considering that:

$$\|x_i - w_j\|^2 = x_i^2 - 2x_i w_j + w_j^2, \tag{3.9}$$

where x_i is a row (a sample) of the input matrix X and w_j is a row (a waypoint) of the waypoints matrix W. In this way, the distance matrix D between all points and waypoints with N rows and N_c columns can be computed as:

$$D = S - 2(XW^T) + V, \tag{3.10}$$

where S is the $N \times N_c$ matrix where the vector:

$$s_i = \sum_{k=1}^{d'} x_{i,k}^2 \tag{3.11}$$

of length N is repeated N_c times along the columns, and V is the $N \times N_c$ matrix where the vector:

$$v_j = \sum_{k=1}^{d'} w_{i,k}^2 \tag{3.12}$$

of length N_c is repeated N times along the rows. This optimization allows to pre-compute the S matrix speeding up the execution time. Figure 3.4 shows the performances of the method *spathialWay* varying the dimension of the input matrix and the number of waypoints. As the total execution time is proportional to the

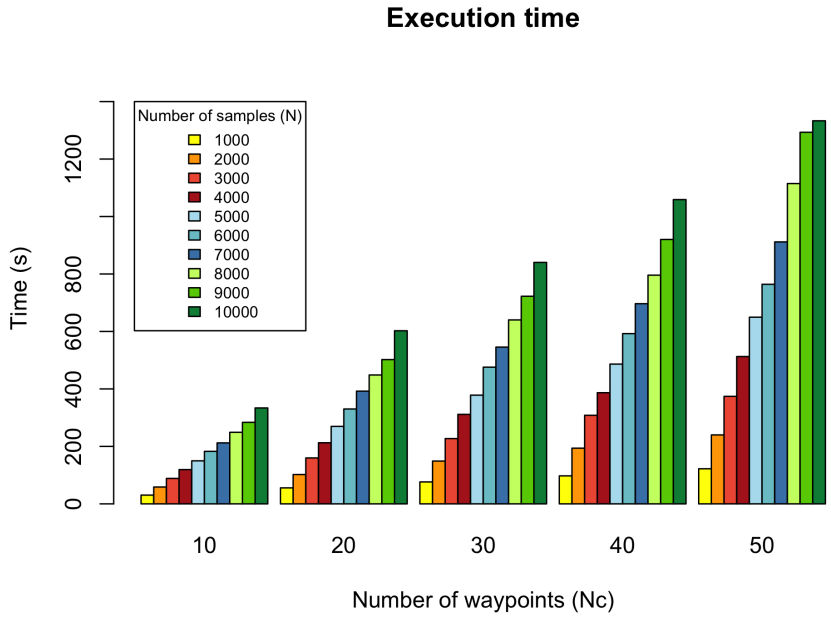

Figure 3.4: *Spathial: execution time analysis.* The input matrix X dimension varies from 1000×20000 to 10000×20000 with step 1000. The number of waypoints varies from 10 to 50 with step 10. The x axis is the number of waypoints, the y axis is the time in seconds. The highest execution time is around 25 minutes. Experiments were performed with a laptop equipped with a 2,4 GHz Intel Core i5 and 8 GB RAM.

number of s values tested (N_s), then:

$$execTime(spathialWay) = N_s \times execTime(PPalgorithm). \tag{3.13}$$

The following experiments were performed with a laptop equipped with a 2,4 GHz Intel Core i5 and 8 GB RAM.

When the function *spathiaWay* is run, users obtain the coordinate of the waypoints (new interpolating samples). *Spathial* provides some utility functions for the analysis of the output. In particular, users can compute the labels of the waypoints (assigned as the label of the nearest point) using the function *spathialLabels*. Additionally, they can plot the 2D visual representation of the datapoints together with the path waypoints running the function *spathialPlot*. This utility function takes as input the path's data points and waypoints. If those are in 2D, the function directly plots them. If not, it first performs a dimensionality reduction using t-SNE (t-Distributed Stochastic Neighbour Embedding) (135). Users can always adopt their preferred

dimensionality reduction strategy and directly give 2D coordinates to the function. Finally, *spathial* allows the user to compute some statistical information about the waypoints through the function *spathialStatistics*. It allows to obtain the Pearson's correlation of the waypoint features with the path progression. Path progression is defined as the ordered sequence of waypoint indices from 0 (the start point) to $N_c + 1$ (the end point). In this way, users can obtain the features that are correlated with the progression (features involved in the transition between the start point and the end point), and can perform a feature selection according to the Pearson's correlation scores. The function also provides the associated p (136) and q (adjusted p) values (137). They can help finding genes that show abundance differences between experiment groups, thereby, genes that may be involved in the process.

Results

Some experiments were performed to compare *spathial* with other existing tools and to demonstrate its flexibility. They were run with a laptop equipped with a 2,4 GHz Intel Core i5 and 8 GB RAM.

A first experiment was conducted on the TCGA lung adenocarcinoma RNA-Seq dataset[3], comprising 562 gene expression profiles RPM-normalized (19637 genes each) where each sample is labeled as *tumor* or *normal* according to the TCGA barcode (138). The experiment aims to navigate the space from normal samples to tumor samples. In this case, the start point was the most distant normal sample from the tumor centroid, and the end point was the most distant tumor sample from the normal centroid. These start and end points were selected to obtain the extremes, conceptually the most *normal* and *diseased* samples. As ground truth, the oncogenes and tumor suppressor genes (TSG) listed in the Cancer Gene Census (139) were considered. The prefiltering was not executed since the search was for a global solution. Finally, the PP algorithm was run with 50 intermediate points (waypoints) plus the boundaries. Figure 3.5 shows the samples (colored according to the labels) and the path. Then, a comparison between the first 1000 best q-value ranked genes for *spathial* with the relevant genes extracted by a commonly used

[3]https://gdac.broadinstitute.org

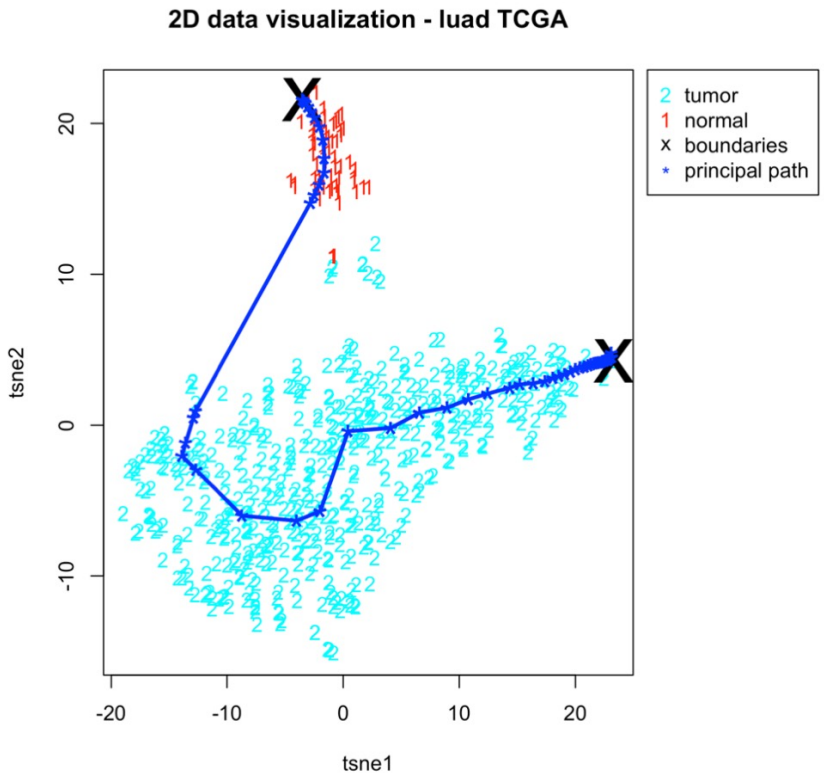

Figure 3.5: *Spathial: PP through the TCGA Lung Adenocarcinoma dataset.* 2D visualization of the PP together with the data points. The x and the y coordinates are the output of the dimensionality reduction performed with tSNE (135). The start point and the end point of the PP are the most distant points from the centroid of the tumor samples and the centroid of the normal samples, respectively. The PP is composed of 50 intermediate points (waypoints) plus the boundaries.

tool for DGEA, *edgeR* (140) (again, the first 1000 best q-values genes), is performed. Tables 3.1 and 3.2 show the details for this comparison. Some genes that *spathial* identified as being involved in the progression were not identified by *edgeR* (and vice versa). To further highlight the genes found by *spathial* and missed by *edgeR*, the most correlated genes for *spathial* were selected using the quantiles and setting two thresholds such that 70% of values fell below the first threshold and 30% fell above the second threshold, representing the most positively and negatively correlated

Table 3.1: *Spathial vs. edgeR: TCGA Lung Adenocarcinoma dataset.* Common genes are those found by both the methods in the subset of 1000 best q-value ranked genes.

	Common genes	Discordant genes
N. genes	11	989

Table 3.2: *Spathial vs. edgeR - oncogenes and TSG: TCGA Lung Adenocarcinoma dataset.* Common genes are the oncogenes and the TSG found by both the methods in the subset of 1000 best q-value ranked genes. *spathial* and *edgerR* genes are the oncogenes and TSG found respectively only by *spathial* and *edgeR* in the subset of 1000 best q-value ranked genes.

	Common genes	*spathial*	*edgeR*
N. Oncogenes	0	10	17
N. TSG	0	16	10

genes. Among the positively correlated genes for *spathial*, the oncogenes were selected and compared with the *edgeR* results. In particular, the analysis was focused on how these oncogenes are placed by the *edgeR* and *spathial* ranking, respectively, according to their statistical significance and their Pearson's correlation scores. The statistical significance was estimated as:

$$-\log_{10}(pvalue) * (FoldChange). \tag{3.14}$$

Finally, the first 30 genes for which *spathial* disagreed the most strongly with *edgeR* and for which the *spathial* rank was better than *edgeR* were selected. The same comparison can be performed by choosing the TSG from the most negatively correlated genes for *spathial*. Figure 3.6 shows the subset of 60 genes selected as described above. The x and y coordinates are the positions in the rank for *edgeR* and *spathial*, respectively. Oncogenes and TSG should be placed at the end and the beginning of the rank, respectively (the rank is with a sign and ascending), because they should be highly positively and highly negatively correlated in the transition from normal to tumor. Therefore, the red genes on the left (oncogenes) and the blue genes (TSG) on the right are those that *spathial* (but not *edgeR*) identified as involved in the transition from normal to tumor.

An equivalent analysis was also performed on the TCGA liver hepatocellular carcinoma RNA-Seq dataset[4], composed of 343 gene expression profiles RPM-normalized (18761 genes each), and the breast invasive carcinoma RNA-Seq datasets[5], consisting of 1092 gene expression profiles RPM-normalized (19733 genes each) (138). Figures

[4]https://gdac.broadinstitute.org
[5]https://gdac.broadinstitute.org

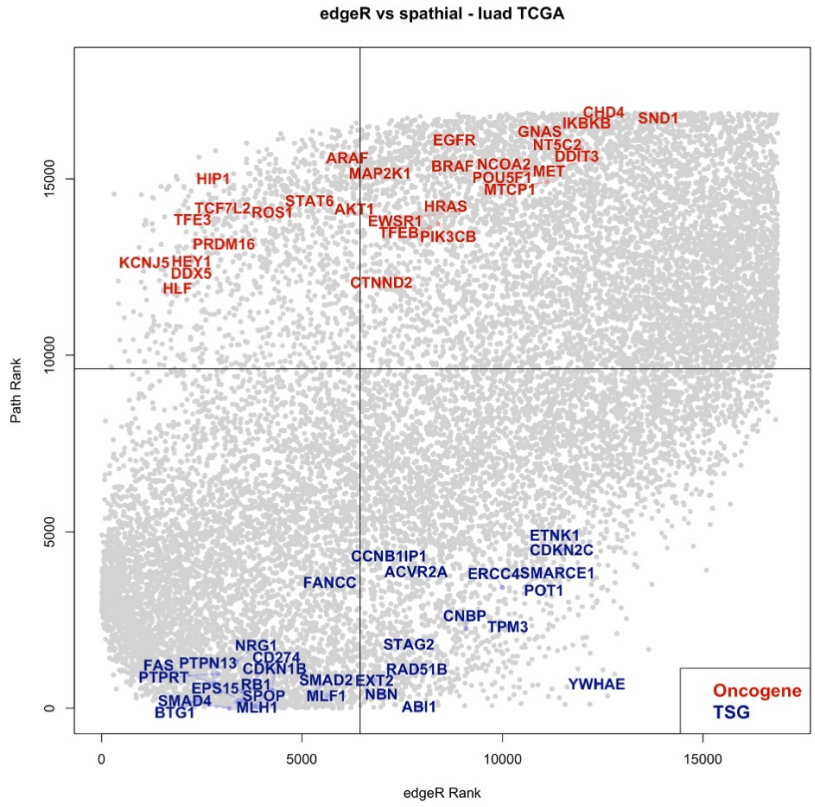

Figure 3.6: *Spathial vs. edgeR: TCGA Lung Adenocarcinoma dataset.* Comparison of *spathial* and *edgeR*. The x and y coordinates are the position in the rank for *edgeR* and *spathial*, respectively. The ranks are the statistical significance (see Equation 3.14) for *edgeR* and the Pearson's correlation score for *spathial*. Colored genes are the 60 oncogenes and TSG (from the Cancer Gene Census list (v86 - (139)) for which *spathial* disagreed the most strongly with *edgeR* and for which the *spathial* rank was better than *edgeR*.

3.7 and 3.8 and Tables 3.3, 3.4, 3.5, 3.6 show the resulting path and the comparison with *edgeR*.

A second experiment was performed to test the *spathial* performances on a single-cell RNA-Seq dataset. In this case, the dataset used for the experiments is presented in (141)[6]. This dataset comprises 96 human myxoid liposarcoma cells, and each is described with a gene expression profile (23928 genes each). Cells are labeled as *G1*, *S*, *G2/M* according to their experimentally determined cell cycle phase. The experiment aimed to navigate the space from the *G1* samples to the *G2/M* samples. The start point was the *G1* centroid, and the end point was the *G2/M* centroid. There was no prefiltering because the search was for a global solution. Finally, the

[6]https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-614

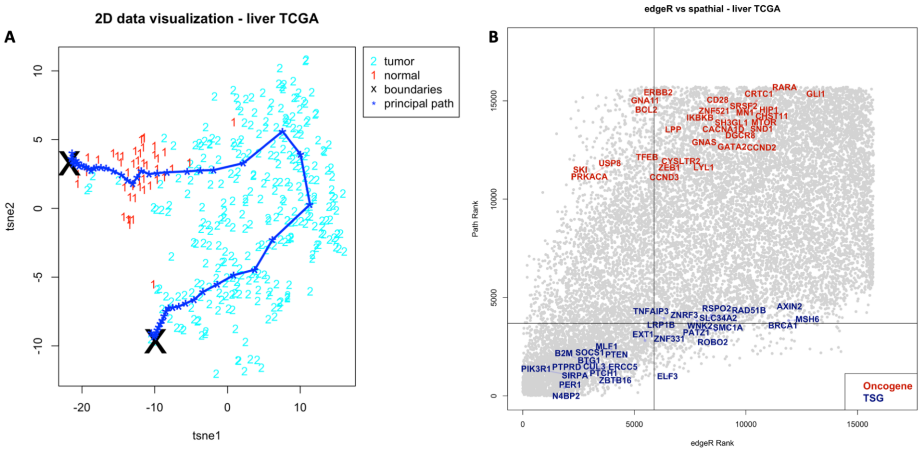

Figure 3.7: *Spathial: results on the TCGA Liver Hepatocellular Carcinoma dataset.* (A) 2D visualization of the PP together with the data points. The x and the y coordinates are the output of the dimensionality reduction performed with tSNE ([135]). The start point and the end point are the most distant normal and tumor points respectively from the centroid of the tumor samples and the centroid of the normal samples. The PP is composed of 50 intermediate points (waypoints) plus the boundaries. (B) Comparison of *spathial* and *edgeR*. The x and y coordinates are the position in the rank for *edgeR* and *spathial* respectively. The ranks are the statistical significance (see Equation 3.14) for *edgeR* and the Pearson's correlation score for *spathial*. Colored genes are the 60 oncogenes and TSG (from the Cancer Gene Census list (v86 - [139])) for which *spathial* disagreed the most strongly with *edgeR* and for which the *spathial* rank was better than *edgeR*.

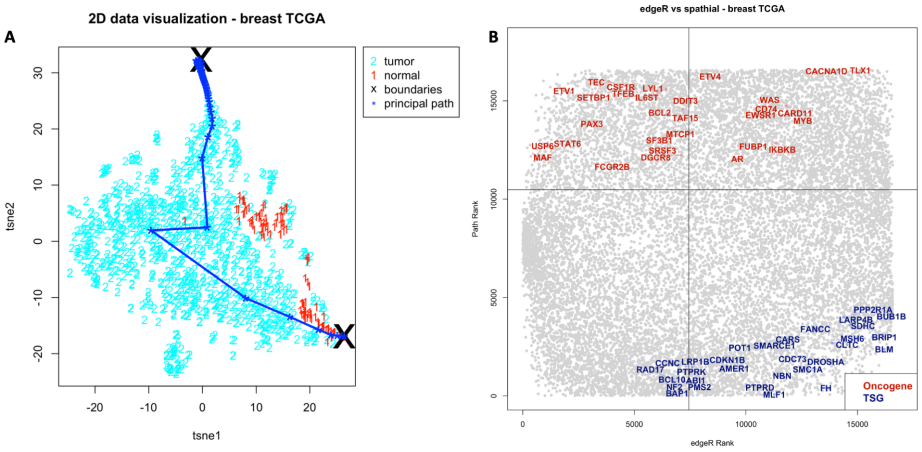

Figure 3.8: *Spathial: results on the TCGA Breast Invasive Carcinoma dataset.* (A) 2D visualization of the PP together with the data points. The x and the y coordinates are the output of the dimensionality reduction performed with tSNE ([135]). The start point and the end point are the most distant normal and tumor points respectively from the centroid of the tumor samples and the centroid of the normal samples. The PP is composed of 50 intermediate points (waypoints) plus the boundaries. (B) Comparison of *spathial* and *edgeR*. The x and y coordinates are the position in the rank for *edgeR* and *spathial* respectively. The ranks are the statistical significance (see Equation 3.14) for *edgeR* and the Pearson's correlation score for *spathial*. Colored genes are the 60 oncogenes and TSG (from the Cancer Gene Census list (v86 - [139])) for which *spathial* disagreed the most strongly with *edgeR* and for which the *spathial* rank was better than *edgeR*.

Table 3.3: *Spathial vs. edgeR: TCGA Liver Hepatocellular Carcinoma dataset.* Common genes are those found by both the methods in the subset of 1000 best q-value ranked genes.

	Common genes	Discordant genes
N. genes	92	908

Table 3.4: *Spathial vs. edgeR - oncogenes and TSG: TCGA Liver Hepatocellular Carcinoma dataset.* Common genes are the oncogenes and the TSG found by both the methods in the subset of 1000 best q-value ranked genes. *spathial* and *edgerR* genes are the oncogenes and the TSG found respectively only by *spathial* and *edgeR* in the subset of 1000 best q-value ranked genes.

	Common genes	*spathial*	*edgeR*
N. Oncogenes	1	12	6
N. TSG	0	13	20

Table 3.5: *Spathial vs. edgeR: TCGA Breast Invasive Carcinoma dataset.* Common genes are those found by both the methods in the subset of 1000 best q-value ranked genes.

	Common genes	Discordant genes
N. genes	1	999

Table 3.6: *Spathial vs. edgeR - oncogenes and TSG: TCGA Breast Invasive Carcinoma dataset.* Common genes are the oncogenes and the TSG found by both the methods in the subset of 1000 best q-value ranked genes. *spathial* and *edgerR* genes are the oncogenes and the TSG found respectively only by *spathial* and *edgeR* in the subset of 1000 best q-value ranked genes.

	Common genes	*spathial*	*edgeR*
N. Oncogenes	0	9	11
N. TSG	0	10	12

PP algorithm was run with 50 intermediate points (waypoints) and the boundaries. Figure 3.9 shows the samples (colored according to the labels) and the path. The q-value for each gene was computed, then the genes with high statistical significance (the first 1000 best q-value ranked genes) were selected. Finally, they were compared with the statistical information extracted by *monocle3*, a package for single-cell trajectory analysis ([132]; [133]; [134]). In particular, one can use *monocle3* to learn the graph and find genes that are differentially expressed across a single-cell trajectory by computing the Moran's I test. Here too, the first 1000 best q-value ranked genes were

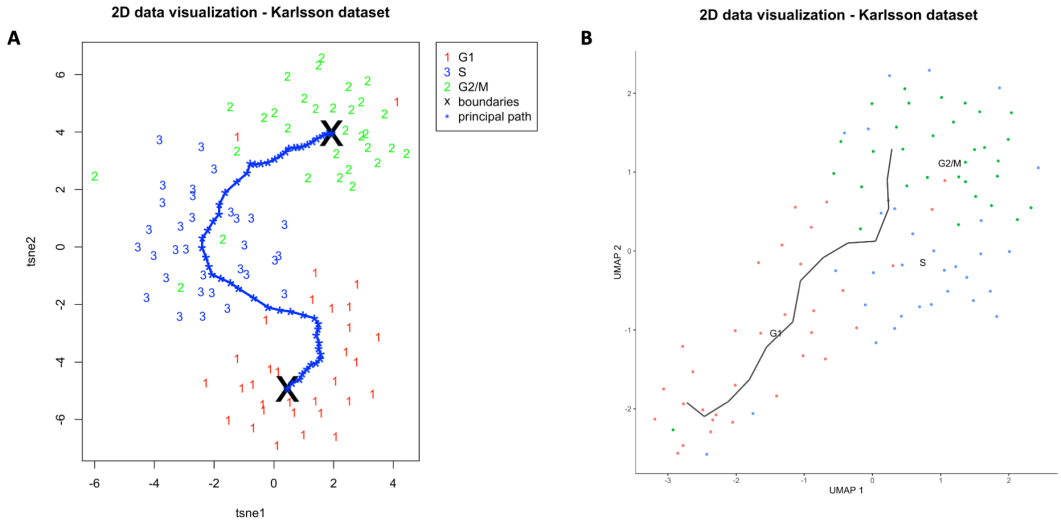

Figure 3.9: *Spathial: results on the Karlsson dataset.* (A) 2D visualization of the PP created with *spathial* together with the data points. The x and the y coordinates are the output of the dimensionality reduction performed with tSNE ([135]). The start point and the end point are respectively the centroid of the *G1* samples and the *G2/M* samples. The PP is composed of 50 intermediate points (waypoints) plus the boundaries. (B) 2D visualization (with UMAP) of the trajectories learned with *monocle3*.

selected (this q-value is computed on the Moran's I scores and adjusted according to the Benjamini-Hochberg method). A significant overlap between *monocle3* and *spathial* gene predictions was detected (see Tables 3.7 and 3.8). However, some genes identified by *spathial* as being involved in the progression were not identified by *monocle3* (and vice versa). Some of these genes belong to *Group2* and *Group3* of

Table 3.7: *Spathial vs. Monocle: Karlsson dataset.* Common genes are those found by both the methods in the subset of 1000 best q-value ranked genes.

	Common genes	Discordant genes
N. genes	206	794

Table 3.8: *Spathial vs. Monocle - Group2 and Group3: Karlsson dataset.* Common genes are those belonging to Group2 and Group3 found by both the methods in the subset of 1000 best q-value ranked genes. *spathial* and *monocle3* genes are those belonging to Group2 and and Group3 found respectively only by *spathial* and *monocle3* in the subset of 1000 best q-value ranked genes.

	Common genes	*spathial*	*monocle3*
N. Group2	26	9	7
N. Group3	42	23	17

the (141) experiment and are known to decrease and increase in expression from G1 toward mitosis respectively; the decrease/increase information is used as ground truth.

Reproducibility

The R package *spathial* is available respectively on the Comprehensive R Archive Network (CRAN) and GitHub:

- https://cran.r-project.org/web/packages/spathial/index.html
- https://github.com/erikagardini/spathial.

The datasets can be retrieved from:

- TCGA dataset:
  https://gdac.broadinstitute.org.
- Karlsson dataset:
  https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-6142/

Additional files are also available as supplementary data:

- https://academic.oup.com/bioinformatics/article/36/17/4664/5841659.

The *edgeR* (140) analysis can be reproduced using the R script *edger_spathial.R*. The TCGA datasets (**_tcga.rda*, where * is the tumor type) and the Cancer Gene Census list (v86 - (139)) used as ground truth (*cgc_genelists.rda*) are provided. The first variable inside the script is *tumor_type*. Users can change its content to repeat the same analysis with a different dataset.
Similarly to the steps for the *edgeR* experiments, the R script *monocle3_spathial.R*, the Karlsson dataset (*karlsson_rawcounts.rda*) and the list of genes in Group2 and Group3 (*karlsson_genelists.rda*) are provided to reproduce the analysis with *monocle3* (132; 133; 134). The list of genes is from the Supplementary Materials of the Karlsson experiment (141). Finally, the R script *spathial_speed_test.R* is provided to reproduce the experiments for the execution time analysis. The obtained results are saved in *spathial_time_table.rda*.

All the experiments were run on a laptop equipped with a 2,4 GHz Intel Core i5 and 8 GB RAM.

Final remarks

In this Chapter, the PP algorithm has been used to analyze transcriptomic and single-cell RNA-Seq datasets proving its effectiveness and flexibility in coping with different problems. The transcriptomic and the single-cell RNA-Seq dataset are examples of the application of the PP path, which can be more in general used for the analysis of any omics. Results obtained in these experiments show that the tool can retrieve information missed from other packages and vice versa, which could be further investigated in the future and could have important clinical implications for the early detection of tumors or staging in general.
In addition, an R implementation of the PP algorithm, dubbed *spathial*, is presented. It provides various high-level functions to the final users, making the algorithm accessible to the R-user Community and simplifying its systematic application.

3.2.2 Challenging the Principal Path in CNN-induced spaces

Creative and innovative applications have recently attracted the attention of the Computational Intelligence (CI) community, including cognitive tasks such as sentiment analysis (142; 143; 144), neural style transfer (145), artistic style recognition (146; 147; 148), and musical genre identification (149). CNNs (150; 151), presented in Chapter 2, are particularly effective for recognizing, analyzing, and classifying images and videos (22; 89; 152; 153), hence are suitable candidates for the above mentioned tasks.
In sentiment analysis, works like (154; 143; 144) and many others have tried transforming abstract concepts like emotions into images and sounds. Neural-style transfer applications use CNNs to recombine one image's content with another's style, like in (145). In terms of recognizing styles of visual art, some of the most interesting solutions have been proposed by Lecoutre et al. (146), Karayev et al. (147), and Tan et al. (148). They achieved promising results using two publicly available CNN architectures (AlexNet (152) and ResNet (90)) with the same dataset (Wikipainting).

Recently, (155) and (156) addressed the problem of how learning systems *perceive* visual art aspects, such as stylistic properties, when compared to human-derived artistic principles. Regarding the classification of musical genres, Bahuleyan recently proposed a particularly interesting solution using CNNs (149). This approach uses spectrograms (visual representations of an audio signal's time and frequency information) generated from a subset of songs in Audio Set (157).

This Section introduces a protocol for analyzing the historical evolution of events in music and art (116). Music and visual art share a fundamental trait: their historical evolutionary nature. Cultural and historical events influence changes in visual art movements and musical genres, they can thus be considered in evolutionary terms. This work aims at challenging the PP in such an environment by checking its capability to reconstruct the historical evolution of styles by analyzing the input data manifold. The capabilities and limits of the PP are hence measured in a context in which the evolution of the events (ground truth) is known.

The starting observation technically supporting this approach is that visual arts and music can be represented as images. The visual representation and hence featurization of music can be obtained by extracting the mel-spectrograms from raw audio data. Consequently, one can represent an audio file as a point in a high-dimensional space. Using the PP directly for navigating the high-dimensional space can be challenging, considering the curse of dimensionality problem (158). A possible solution involves ANNs, particularly CNNs, to learn a reasonable, low-dimensional, representation. Then, the manifolds induced by the CNNs can be navigated using the PP and focusing on the historical evolutionary changes from one song/visual artwork to another. Additionally, the resulting waypoints, which are newly generated samples obtained as output of the PP algorithm, can be classified using their neighbor samples and can be visually represented in a 2-dimensional space through the t-SNE projection method (135). This way, the insights extracted through the PP can be easily inspected and discussed.

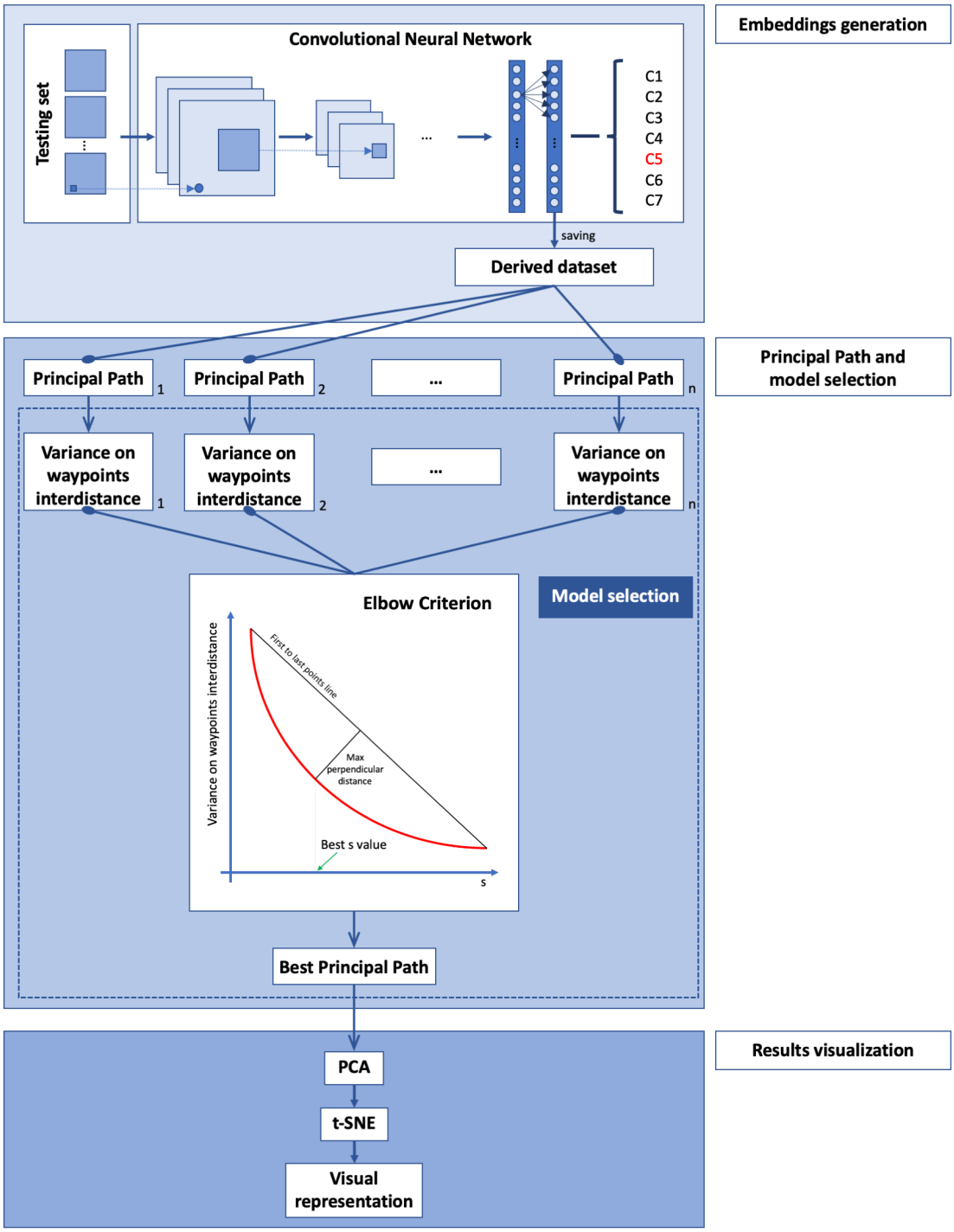

Figure 3.10: *PP in music and visual art spaces: protocol. Step 1:* featurization with CNNs. *Step 2:* PP on the induced vector space with different s values and model selection with the elbow criterion. *Step 3:* visualization of the best PP.

Analysis protocol

The approach mentioned above (visually represented in Figure 3.10) can be summarized in three main steps:

1. Embedding space generation through CNNs.
2. Running of the PP algorithm in the induced space.
3. Visualization of the results (via the t-SNE method), quantitative and qualitative comparison to the true historical style evolution.

The first step of the proposed protocol involves CNN architectures to learn a feasible representation of the input samples. This representation is extracted from the penultimate layer of the CNN. Here, two existing and widely used CNN architectures have been considered, namely ResNet-50 (90) and VGG-16 (159). The VGG-16 architecture differs from the original ConvNet architecture (152) because it has more layers (sixteen) but small (3x3) convolution filters. As a result, the network has better performance and accuracy. However, increasing the network depth is not always beneficial. Beyond a certain point, degradation issues arise, and the number of training errors grows larger. Such a problem is successfully addressed by the ResNet-50 architecture introducing deep residual learning framework blocks, as previously mentioned in Section 2.3.1. Table 3.9 shows the original configurations for VGG-16 (159) and ResNet-50 (90). Here, the original ResNet-50 architecture is used for the visual art space experiments (146). The architecture proposed in (149) is used for the music space experiments, with the original VGG-16 core connected to a new feedforward neural network. In both cases, the predicted labels of the CNNs are ignored. Instead, the output from the last pooling layer is used as feature vector for the subsequent steps.

The second step of the proposed protocol consists in navigating the CNN-induced space with the PP algorithm, aiming at extracting relevant information about the data topology. In this work, the PP algorithm is run several times with decreasing values of the regularization parameter s on an evenly spaced log scale from 10^5 to 10^{-5}, without changing start and end points. In this way, several PPs can be obtained

Table 3.9: *VGG-16 and ResNet-50 standard architectures.*

VGG-16	**ResNet50**
16 weight layers	50 weight layers
Input 224x224 RGB image	Input 224x224 RGB image
Conv3-64 Conv3-64	Conv7-64
Maxpool	Maxpool
Conv3-128 Conv3-128	Conv1-64 Conv3-64 X3 Conv1-256
Maxpool	Conv1-128 Conv3-128 X4 Conv1-512
Conv3-256 Conv3-256 Conv3-256	
Maxpool	Conv1-256 Conv3-256 X6 Conv1-1024
Conv3-512 Conv3-512 Conv3-512	
Maxpool	Conv1-512 Conv3-512 X3 Conv1-2048
Conv3-512 Conv3-512 Conv3-512	
Maxpool	Avgpool
FC-4096 FC-4096 FC-1000	FC-1000
Soft-max	Soft-max

(one for each s value), and the best path, W_{best}, is selected using the previously introduced elbow criterion as model selection strategy (see Figure 3.3).

The last step of the proposed protocol consists in qualitatively and quantitatively comparing with the ground truth. To achieve this aim, two different techniques are explored. The first technique uses t-SNE (135) to provide a 2D data visualization of the data points, the waypoints of the PP, and the points of a trivial path. A trivial path is defined here as a mere linear connection of the end points and it is used as a *null hypothesis* to prove that results are not obvious. To improve the t-SNE efficiency, the Principal Component Analysis (PCA) (160) algorithm is used to reduce the dimensionality before t-SNE itself. The second check detects the sample songs and

visual artworks nearest to the waypoints of the PP and the points of the trivial path.

Results

Several experiments have been performed in both the visual art and music spaces to validate the proposed protocol.

In terms of the visual art space, the manifold induced by the network defined in (146) was analyzed without performing any retraining procedure. The network implementation is the one in (161) because it was explicitly trained to recognize visual art styles and because the present analysis aimed to understand their relations. The Wikipainting dataset[7] is used, which is an extensive and widely used dataset collection from the WikiArt website (162). It comprises 80,000 images, each tagged with one of the following 25 styles: *Abstract Art*, *Abstract Expressionism*, *Art Informel*, *Art Nouveau (Modern)*, *Baroque*, *Color Field Painting*, *Cubism*, *Early Renaissance*, *Expressionism*, *High Renaissance*, *Impressionism*, *Magic Realism*, *Mannerism (Late Renaissance)*, *Minimalism*, *Naive Art (Primitivism)*, *Neoclassicism*, *Northern Renaissance*, *Pop Art*, *Post-Impressionism*, *Realism*, *Rococo*, *Romanticism*, *Surrealism*, *Symbolism*, and *Ukiyo-e*. The featurization led to 2048 variables from the penultimate layer of the network. The space was navigated through the PP, selecting three or four classes at a time with a reasonable historical distance to simplify the analysis and using the most recent and the oldest visual artworks as start/end points. A path comprising 50 waypoints plus the start and end points was generated. To understand the transitions, the nearest picture for each waypoint was retrieved. To check the significance of the results and for comparison, a trivial path was generated by connecting the start and end points with a straight line and splitting it into 50 equally distributed points. Figure 3.11a shows the class (the true style label) of the retrieved nearest pictures for the PP and the trivial path, considering different styles. To visualize the points together with the paths, the dimension was reduced via PCA (sklearn implementation with n_components = 50 and random_state = 5) (160) followed by t-SNE (sklearn implementation with n_components=2, random_state=20, perplexity=50 and learning_rate=300) (135). PCA reduced the number of features

[7] www.lamsade.dauphine.fr/ bnegrevergne/webpage/software/rasta/wikipaintings$_{f}ull.tgz$

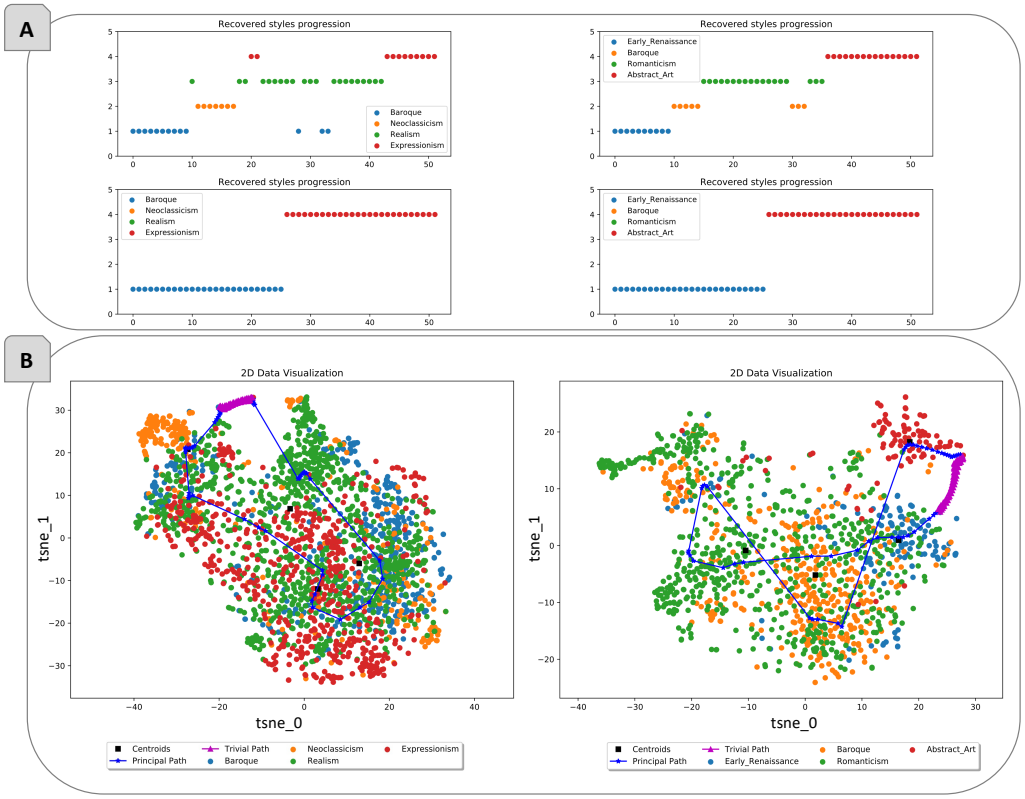

Figure 3.11: *PP in music and visual art spaces: labeling and 2D visualization.* (a) Labels of the nearest artwork for each waypoint of the PP (top) and the trivial path (bottom). On the left, results for the following classes: Baroque, Neoclassicism, Realism, Expressionism. On the right, results for the following classes: Early Renaissance, Baroque, Romanticism, Abstract Art. (b) 2D representation of the PP and the trivial path through four different styles: Baroque, Neoclassicism, Realism, Expressionism (on the left); Early Renaissance, Baroque, Romanticism, Abstract Art (on the right). The x and the y coordinates are the output of the dimensionality reduction performed with tSNE ([135]). The start point and the end point are the most recent and the oldest visual artworks, respectively. The PP is composed of 50 intermediate points (waypoints) plus the boundaries.

from 2048 to 50, improving the t-SNE algorithm's efficiency (Figure 3.11b). Within this simplified and class-reduced setting, the PP correctly recovered the historical evolution of the artistic style and the content of the visual artworks. This indicates that the employed CNN-induced spaces at least partially reflect the historical evolution of the styles. In contrast, the trivial path moved blindly and jumped from the start class to the end class without any interesting intermediate. These aspects are emphasized when the start and end points are the historically oldest and newest visual artworks, respectively. In addition to this analysis, other paths were generated by perturbing the start/end points (e.g., using the second/third most

recent and the second/third oldest visual artworks as start/end points). Results are available in the GitHub repository[8] (folders /results/images/mode=2_1-22-18-25 and /results/images/mode=2_22-14-17-3) and show that the PP is robust to perturbations. The PP's ability to capture the historical evolution of visual art is highlighted when one selects and plots the nearest visual artworks to the points of the two paths, as shown in Figure 3.12 (repeated consecutive artworks along the path were not shown in this, and all the subsequent figures for clarity of representation). In Figure 3.12a, the PP finds visual artworks that depict people and then landscapes. The style varies from black and white to color, gradually changing from dark, cool colors to light warm ones. Except for some noisy visual artworks, the evolution of the classes reflects the historical evolution of the art (i.e., Baroque ∼17th-18th century, Neoclassicism ∼18th-19th century, Realism ∼19th century, Expressionism ∼20th century). In contrast, the trivial path passes through two visual artworks with different styles and content and belonging to the first and last class, respectively. The same gradual morphing can be observed in Figure 3.12b. The evolution of the pictures reflects the historical evolution of the art (i.e., Early Renaissance ∼14th-16th century, Baroque ∼17th-18th century, Romanticism ∼18th-19th century, Abstract Art ∼20th century), with the colors and content gradually changing from one artwork to the next. There is an interesting jump from romanticism to abstract art, with a strong similarity in shape: the man and the sculpture are both in the center of the image, and the top of the sculpture resembles the man's hat in the portrait. Once again, the trivial path passes through two visual artworks with different styles and content belonging to the first and last classes, respectively. The historical evolution is not always found by the PP solution, as shown in Figure 3.13. In this case, among a subset of five classes (Early Renaissance, Mannerism, Baroque, Neoclassicism and Impressionism), the PP retrieved only visual artworks belonging to the Early Renaissance and the Impressionism, with a little deviation to the Baroque. Even if the historical evolution is not respected, the PP can perform a gradual morphing from the start to the end points. This is particularly clear when compared to the performance of the trivial

[8]https://github.com/erikagardini/Using-PP-to-walk-through-music-and-visual-art-style-spaces-induced-by-CNN

Figure 3.12: *PP in music and visual art spaces: visual artworks results.* Nearest visual artworks to the waypoints of the PP and the trivial path, removing consecutive repeated artworks and considering four classes: (a) Baroque (blue), Neoclassicism (orange), Realism (green), Expressionism (red); (b) Early Renaissance (blue), Baroque (orange), Romanticism (green), Abstract Art (red).

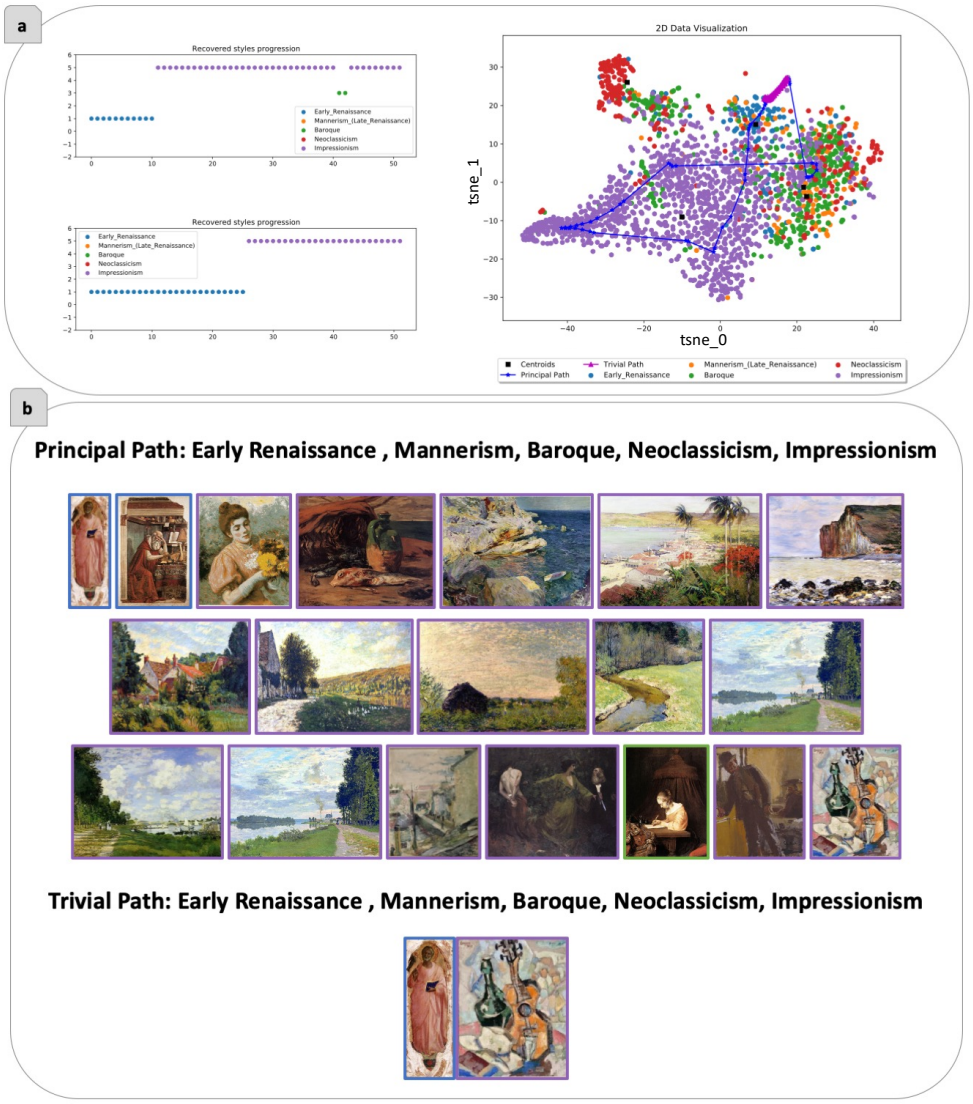

Figure 3.13: *PP in music and visual art spaces: failure.* (a) On the left, labels of the nearest artworks for each waypoint of the PP (top) and the trivial path (bottom). On the right, 2D representation of the PP and the trivial path through four different styles: Early Renaissance, Mannerism, Baroque, Impressionism. The x and the y coordinates are the output of the dimensionality reduction performed with tSNE (135). The start point and the end point are the most recent and the oldest visual artworks, respectively. The PP comprises 50 intermediate points (waypoints) plus the boundaries. (b) Nearest visual artworks to the waypoints of the PP, removing consecutive repeated artworks and considering four classes: Early Renaissance (blue), Mannerism (orange), Baroque (green), Impressionism (purple).

path, which once again passes through two visual artworks with different styles and content belonging to the first and last class, respectively. Further results for the visual art space are freely available in the GitHub repository[9], where the results

[9]https://github.com/erikagardini/Using-PP-to-walk-through-music-and-visual-art-style-spaces-induced-by-CNN

for different subsets of classes and different starting/ending points selections are collected.

In the music space, the above experiments were repeated, analyzing the manifold induced by the network defined in (149) and available at (163). This time, the network is trained to recognize music genres. In contrast to the visual art context, the literature contains music experiments that used datasets with very different traits. The best-known datasets with audio tracks include RWC (465 entries) (164; 165), GZTAN genre (1000 entries) (166), Magnatagatune (25863 entries) (167), and AudioSet (40540) (157). Here, the experiments are performed considering the Magnatagatune dataset[10] (167), instead of the AudioSet (157) mentioned in (149). This is because the Magnatagatune dataset is one of the largest with available audio tracks; additionally, it is enriched with tag annotation files that assign a list of genres and instruments to each song. Specifically, a subset of songs is selected considering the following genres: Baroque, *Classical*, *Jazz*, *Medieval*, *Opera*, and *Rock*. A univocal class is established for each song by choosing the most specific genre (e.g., a song labeled as Baroque and Classical becomes Baroque). In this case, the featurization led to 512 features from the penultimate layer of the network. The dataset was randomly split into training (80%) and test (20%) sets. The validation set was 10% of the training set. The training phase was run for no more than 20 epochs using the same batch size (100) and using the Adam optimizer (default learning rate = 0.001)(92). Figure 3.14 shows the learning curves of the model. The best model was the one obtained at the sixth epoch. This model had the highest accuracy and the lowest loss on the validation set. Finally, the best model was tested on the test set for accuracy, F1-score, Area Under the Curve (AUC), and Matthews Correlation Coefficient (MCC), obtaining 0.86, 0.55, 0.9, and 0.77, respectively. Again, the induced spaces from the CNN were navigated using the PP and the trivial path, considering three or four classes at a time with a reasonable historical distance, with the start/end points selected visually and generating 50 intermediate waypoints. Figure 3.15a compares the class variation for the nearest song to each waypoint (as a class, the true style label of each nearest song is considered). Figure 3.15b shows the 2D visual representation

[10]http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset

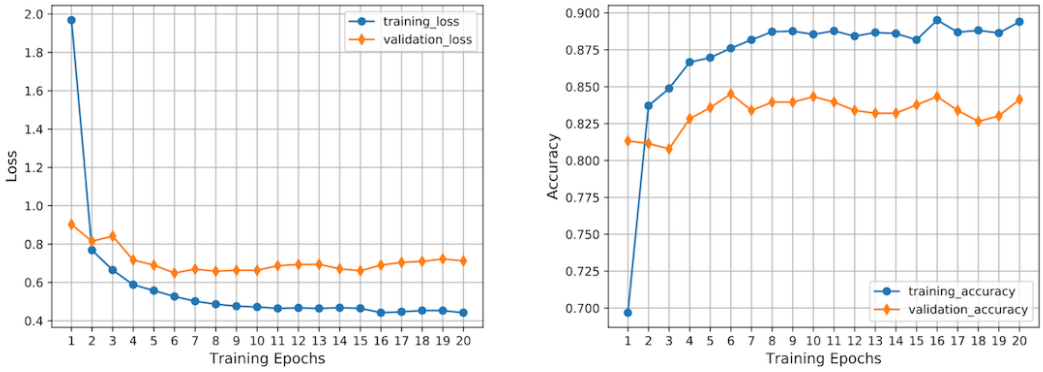

Figure 3.14: *PP in music and visual art spaces: music CNN learning curves for model selection.*

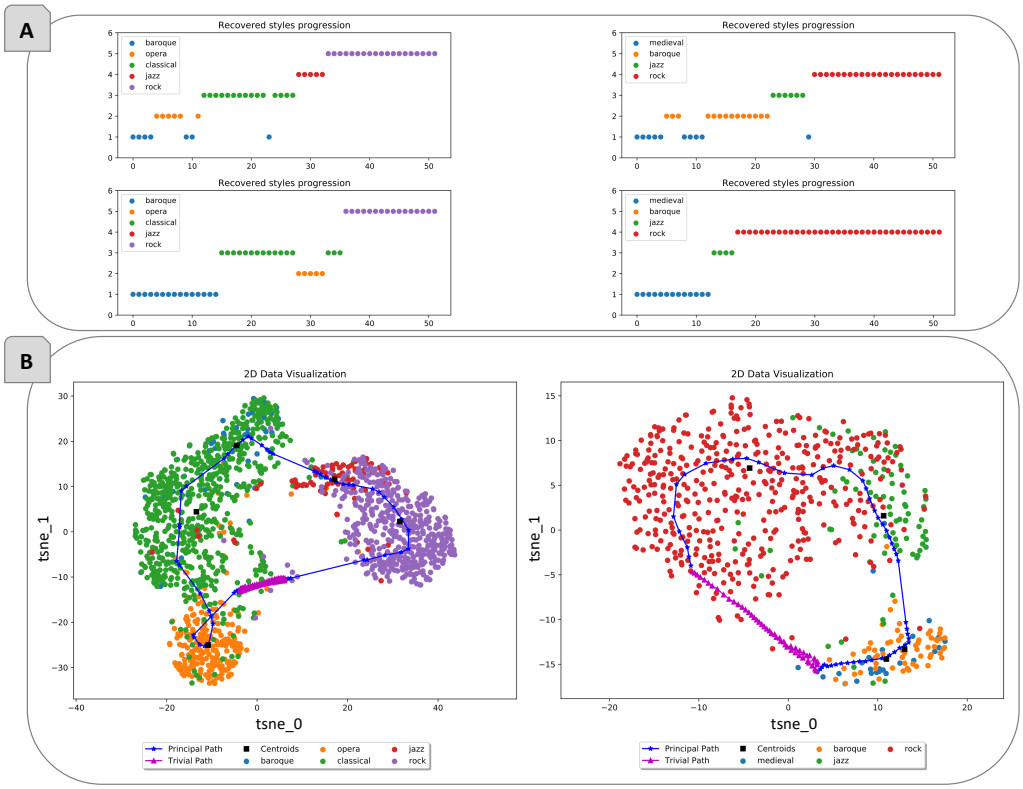

Figure 3.15: *PP in music and visual art spaces: music results.* (a) Labels of the nearest songs for each waypoint of the PP (top) and the trivial path (bottom). On the left, results for the following classes: Baroque, Classical, Opera, Jazz, Rock. On the right, results for the following classes: Medieval, Baroque, Jazz, Rock. (b) 2D representation of the PP and the trivial path through three different styles: Baroque, Classical, Opera, Jazz, Rock (on the left); Medieval, Baroque, Jazz, Rock (on the right). The x and the y coordinates are the output of the dimensionality reduction performed with tSNE ([135]). The start point and the end points are selected visually. The PP comprises 50 intermediate points (waypoints) plus the boundaries.

of the two paths. This was obtained by reducing the number of features from 512 to 50 with the PCA algorithm (sklearn implementation with n_components=50 and random_state=5) (160), then reducing the number of features from 50 to 2 with the t-SNE algorithm (sklearn implementation with n_components=2, random_state=20, perplexity=50 and learning_rate=300) (135). Here, the resulting spectrograms are omitted because they are difficult to interpret. The corresponding songs are listed in the GitHub repository[11], and Table 3.10 shows a summary. Figure 3.15a highlights the PP's ability to navigate the space, considering the evolution of the musical genres, despite the complexity of the problem. The trivial path seems to find intermediate genres between the start and end classes. However, by analyzing Table 3.10, one can observe that the PP performs a smooth transition finding different songs along its way. In contrast, the trivial path finds few songs, and the transition from the start point to the end point is not gradual. Other examples using different classes and start/end points are freely available in the GitHub repository of the project[12].

[11] https://github.com/erikagardini/Using-PP-to-walk-through-music-and-visual-art-style-spaces-induced-by-CNN

[12] https://github.com/erikagardini/Using-PP-to-walk-through-music-and-visual-art-style-spaces-induced-by-CNN

Table 3.10: *PP in music and visual art spaces: music results.* Number of songs found by the PP and the trivial path along their way (without duplicated samples and grouped by classes). Baroque, Opera, Classical, Jazz, Rock (on the top), Medieval, Baroque, Jazz, Rock (on the bottom).

	Principal Path	**Trivial Path**
Baroque	3	1
Opera	3	1
Classical	8	3
Jazz	2	-
Rock	11	1
Total	27	6
	Principal Path	**Trivial Path**
Medieval	4	1
Baroque	6	0
Jazz	6	1
Rock	16	5
Total	28	7

Reproducibility

The implementation was written in the Python/Keras ([168]) framework with Tensorflow ([169]) as backend. The code to reproduce all the experiments is available on GitHub: the code to perform the featurization is at links:

- https://github.com/erikagardini/ImageFeaturization;
- https://github.com/erikagardini/MusicFeaturization.

The code to reproduce the results at:

- https://github.com/erikagardini/Using-PP-to-walk-through-music-and-visual-art-style-spaces-induced-by-CNN.

All the experiments were run on a High Performance Computing (HPC) cluster node equipped with a nVidia P100 GPU and 2 Intel Xeon E5-2650 v4 CPUs.

Final remarks

In this work, the PP algorithm was combined with CNN-induced vector spaces to analyze the historical evolution of the events in visual art and music. Based on the obtained results, the PP found reasonable connections between visual artworks and songs from very different genres, partially respecting their historical evolution in a simplified setting. Albeit far from perfect results, also due to the sub-optimal CNN input manifold, the findings might suggest that many evolutionary processes could be studied by approaching them as a minimum free-energy path-finding problem ([170]). The waypoints along the PP make the presented model implicitly generative, even though a probability density function is not available. In the future, with further development, the explicit creation of new objects could be achieved via generative models (e.g., VAEs ([71])).

3.3 An ab initio local Principal Path algorithm

Despite the promising results described in previous Sections, the algorithmic steps of the PP proposed in ([35]) present some drawbacks. The main one is that, because it is

based on the k-means clustering algorithm, the original PP algorithm is intrinsically global and could lie on a global manifold (involving the entire data set). This affects and can distort the final solution, especially when the manifold is not continuous and samples are gathered in clusters instead. To address this, in (35) Authors proposed to apply a prefiltering procedure before the PP algorithm is run. The prefiltering method selects a subset of datapoints and removes others. The PP thus considers a restricted number of samples, producing a local solution. Although the prefiltering procedure partially solved the problem, it introduced some additional drawbacks.

First and foremost, the prefiltering procedure is optional. This means that it is a user-dependent choice. The use of the prefiltering is strictly related to the distribution of the input data $\mathbf{X}$ and also to the final solution itself. The prefiltering, indeed, should be avoided when one can assume that the solution spans the whole data set, while it becomes crucial when the solution lies in a submanifold; this last situation holds true in naturally clustered data sets. While this discrimination is visual in low-dimensional data sets, it becomes more difficult for high-dimensional ones.

Additionally, at a technical level, the prefiltering procedure depends on three different user-defined parameters: the number of the initial medoids N_f, the value of the threshold T, and the size of the neighborhood k, respectively. Of these, the parameters N_f and T control the strength of the prefiltering, which will be more relaxed for high values, and stricter otherwise. On the other side, the value of k defines the resulting Dijkstra shortest path algorithm, which will comprise close intermediate nodes for low values and loosely spaced nodes otherwise. The number of filtered samples, and consequently the number of samples kept N_k, is not known a priori, but it is strictly correlated to the initial configuration of the input parameters. This is particularly true also for the parameter T. This means that a good configuration of this parameter is crucial for the goodness of the prefiltering step. However, tuning this parameter is not trivial and the wrong configuration may lead the prefiltering to choose an extremely small number of samples to keep, which hinders the correct finding of the PP when $N_k < N_c$.

Another weakness caused by the prefiltering, but more in general, present in the algorithm pipeline, is that the k-means++ (123) algorithm is used for the medoid

initializations (steps 2.1 and 3 in Section 3.1). This adds some degree of randomness to the solution. If the algorithm is repeated multiple times with the same input samples $\mathbf{X}$, it may result in different filtered samples and initial configurations of the waypoints N_c. This affects the final solution when the initialization leads to sub-optimally distributed waypoints. In fact, the cost function of the PP is not convex and heavily depends on the path initialization. For this reason, an annealing procedure was used in the algorithm's original version to solve the problem partially. This procedure involves training several models with decreasing values of the regularization parameters s. The first time the optimization step is performed (with the highest value of s), it takes as input the initial configuration of the waypoints from step 3. In contrast, for all other values of s, the optimization step is performed by using as initialization the resulting path at the previous iteration.

For these reasons, this Section presents a revised version of the PP, based on more robust prefiltering and initialization steps. This way, the method is almost parameter-free and results in an always intrinsically local final solution, which is the very aim of the algorithm differently from the Principal Curve. In the revised algorithm, the prefiltering step is no longer a pre-processing step but an integral, mandatory part of the method. This enhancement was published in (118).

Method

Assuming that choosing the boundaries remains unaltered as in the original algorithm (see Section 3.1), the core steps can be summarized as follows:

1. *Path initialization:* given the starting and the ending points $\underline{w}_0$ and $\underline{w}_{N_c+1}$, the Dijkstra algorithm (120) is run over a penalized graph obtained by computing the penalized distance matrix among all the samples $\underline{x}_i$ in $\mathbf{X}$ defined by:

$$d_p^2(\underline{x}_i, \underline{x}_j) = \begin{cases} d^2(\underline{x}_i, \underline{x}_j), & \underline{x}_i \in \mathrm{nn}_k(\underline{x}_j) \\ pd^2(\underline{x}_i, \underline{x}_j), & \text{otherwise.} \end{cases}$$

   In this way, the shortest path connecting $\underline{w}_0$ to $\underline{w}_{N_c+1}$ is found. Note that the squared Euclidean distance is used because it amplifies long distances and

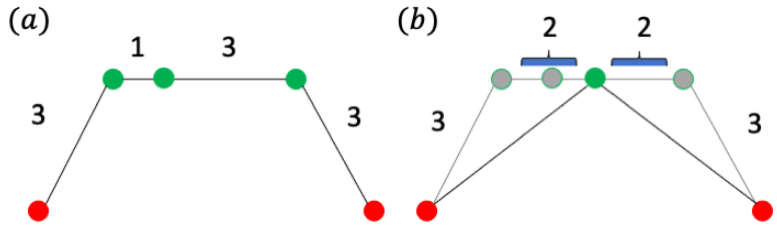

Figure 3.16: *Ab-initio local PP: waypoints positioning.* (a) Hypothetical shortest path after the Dijkstra algorithm with some intermediate nodes not equally distributed. The total length of the path is supposed to be 10. (b) New distribution of the waypoints, considering $N_c = 1$.

reduces small ones. Also, note that this shortest path does not operate on the original distance matrix but in a penalized one that allows it to avoid shortcuts that may stay outside the manifold. This is crucial to obtaining valuable results.

2. *Waypoints positioning:* the Dijkstra shortest path algorithm gives as output a path, which moves from the starting point to the ending point and walks through some intermediate nodes (existing samples in $\mathbf{X}$). The number of intermediate nodes and their distribution is not known a priori. Therefore an adjustment step has been proposed, as summarized in Figure 3.16. This step allows one to obtain the desired number of intermediate waypoints N_c, positioning them equally distributed along the initial path. In particular, given the desired number of intermediate waypoints N_c, one can compute the distance among them as $\Delta = \frac{l}{N_c+1}$, where l is the distance along the path. Each waypoint will therefore be positioned at a distance Δ from the previous point along the path. Note that intermediate nodes are existing samples, while the final waypoints are newly generated samples.

3. *Optimization of the cost function*: as for the original algorithm, the cost function is optimized via the EM algorithm; here, the original feature space is considered. In contrast to the original version, the waypoint configuration $\mathbf{W}_{init}$ from the previous step is used as waypoint initialization and as input

matrix $\mathbf{X}$ ($\mathbf{W}_{init} = \mathbf{X}$). This means that the cost function can be rewritten as:

$$\min_{W,u} \sum_{i=1}^{N_c} \sum_{j=1}^{N_c} \|\phi(x_i) - w_j\|^2 \delta(u_i, j) + s \sum_{i=0}^{N_c} \|w_{i+1} - w_i\|^2. \tag{3.15}$$

The proposed method allows to obtain a more robust initialization step, which is fully deterministic and generative. The initialization is thus a reparameterization of the Dijkstra shortest path. Provided that the metric underlying the graph correctly captures the density (and hence probability) variations, the Dijkstra algorithm can extract the key elements in the transition from the starting to the ending points. However, since it provides an in-sample solution, it may result in an irregular path, with intermediate points gathered only in dense space areas. For this reason, an adjustment of the Dijkstra shortest path is needed, allowing one to obtain equally distributed points that morph more gradually from the starting to the ending points. In this way, the optimization step only involves a refinement to smooth out the initial path and gain an even smoother transition.

Notably, by setting $N = N_c$, the solution for $s = 0$ is completely different from the previous one. Indeed, the previous algorithm solution would have been the k-means waypoint distribution, which is largely distant from a path. Here, instead, the $s = 0$ solution is represented by the re-parameterized shortest path, which is already very close to the final solution. Thus, while the previous algorithm has a regularization path that spans from the straight line to the k-means solution, the solutions space here is much more reduced because its spans from the straight line to the shortest path. This dramatically increases the solution's stability. Additionally, the annealing procedure is not strictly required here because the basin of attraction is much more well-defined. Figure 3.17 shows a schematic representation of the two algorithms, the original and the revised one.

Results

This Section presents some experiments to compare the proposed method with the original version of the PP algorithm.

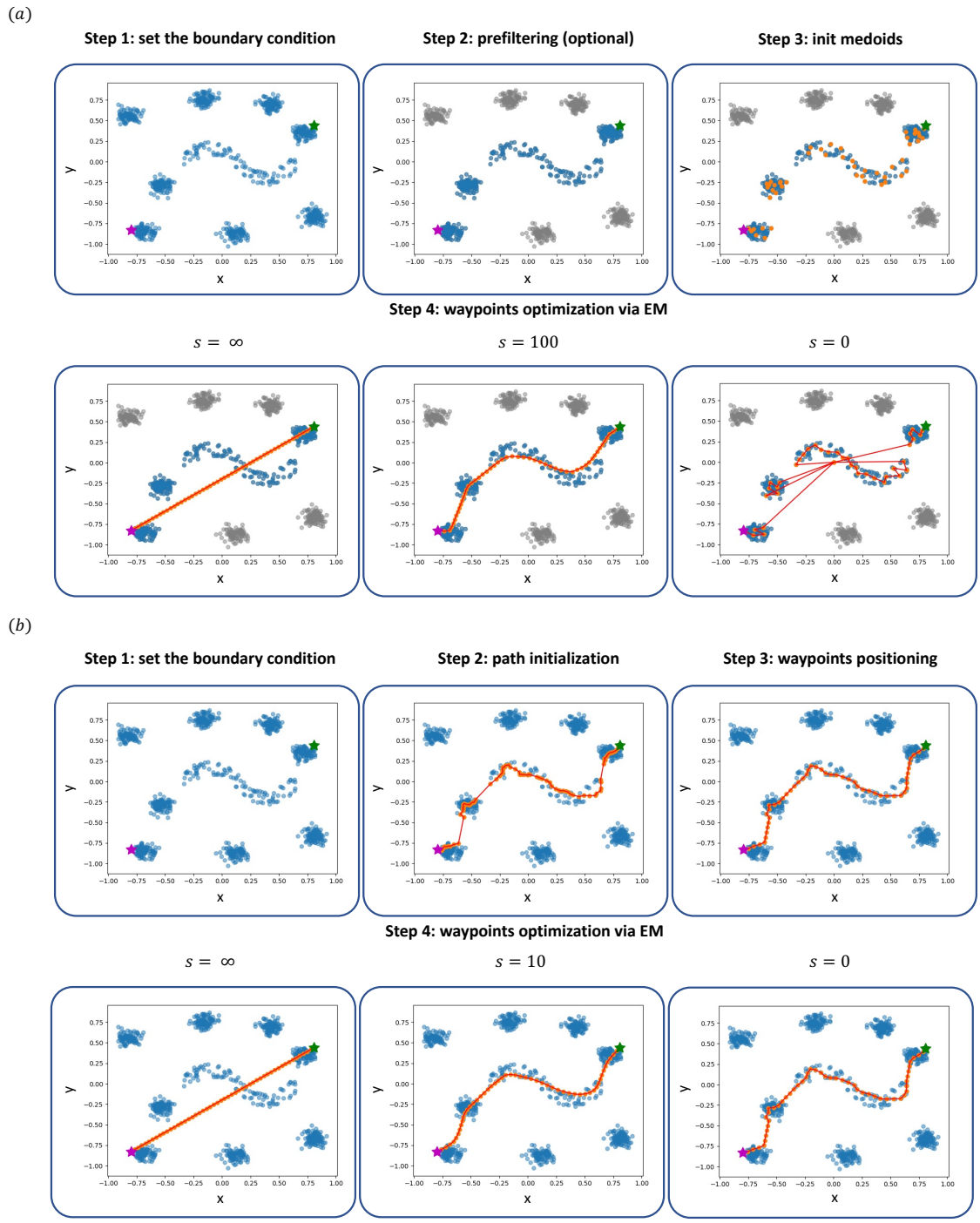

Figure 3.17: *Ab-initio local PP: comparison with the original PP.* (a) Original PP workflow. Step 1: boundary selection. Step 2: prefiltering. Step 3: waypoints initialization. Step 4: waypoints optimization. (b) The proposed workflow. Step 1: boundary selection. Step 2: Dijkstra shortest path. Step 3: waypoints positioning. Step 4: waypoint optimization.

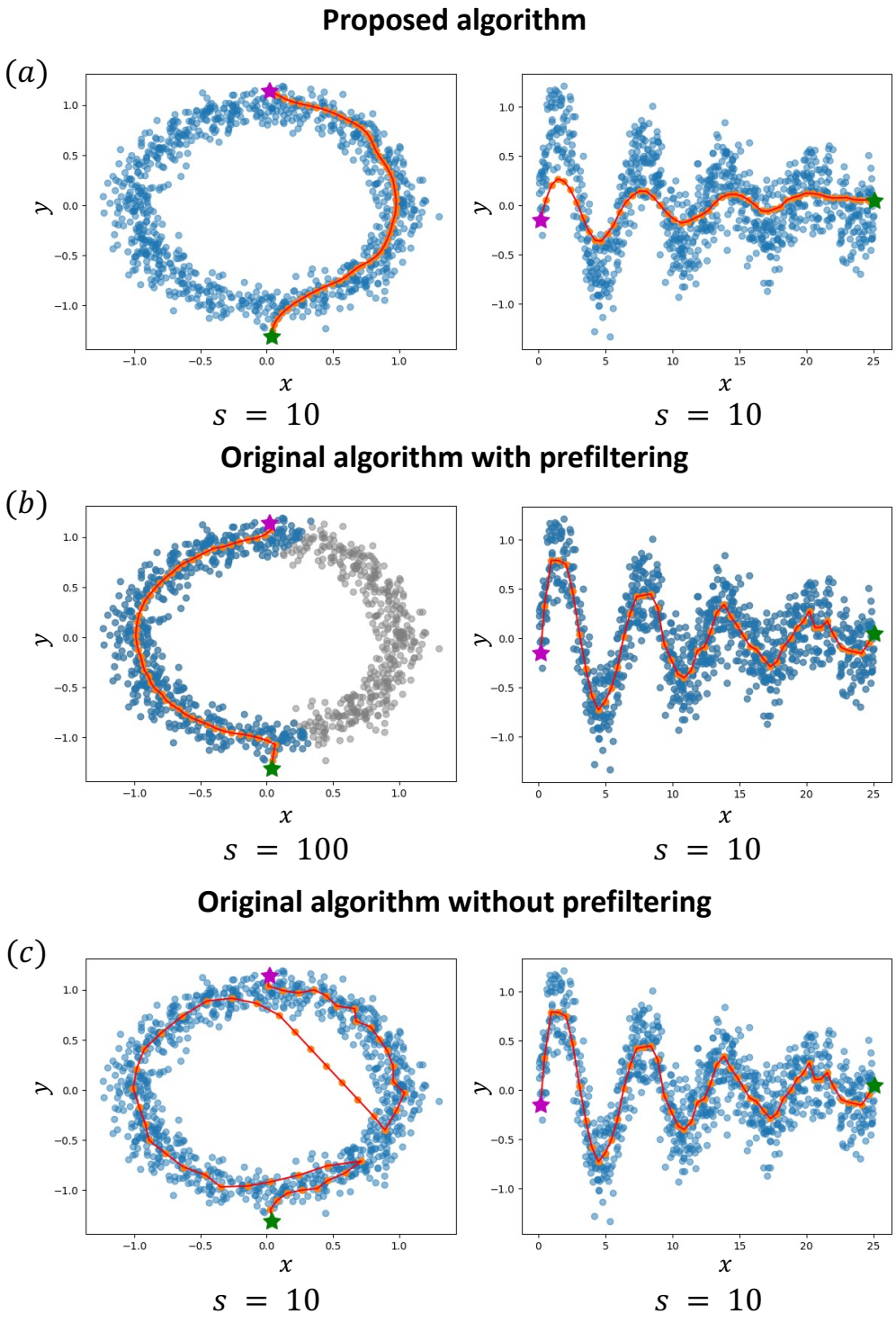

Figure 3.18: *Ab-initio local PP: morphing paths on 2D toy data sets.* Comparison of the results obtained with the revised method (top) and the original method with and without prefiltering (center and bottom, respectively) on two different 2D data sets. The s value selected via the max-evidence model selection is written at the bottom of each figure. The starting point is the green star, the ending point is the magenta star. The samples kept after the prefiltering are in blue. The samples removed are in grey. The final path (red line) and the intermediate waypoints (orange points) are depicted in each figure.

First, experiments with 2D toy datasets[13] have been performed to validate the algorithm steps in a fully controlled environment. Here, the starting and ending waypoints were selected by visual inspection. All the experiments were executed with the following parameters: $N_c = 50$, $N_f = 200$, $T = 0.1$, $k = 5$. Figure 3.18 shows one of the results obtained with the proposed method, compared to the result obtained with the original method with and without prefiltering. In this case, the best s parameter in the domain $\mathbb{D}_s = \{1000000, 100000, 10000, 1000, 100, 10, 0\}$ is selected, using as model selection the Bayesian Evidence Maximization approach proposed in (35). Other experiments are available in the GitHub repository of the project[14]. All the experiments clearly show that searching for a global solution can lead to wrong paths, which try to walk through all the data even if samples are clustered. To make the solution local, the prefiltering step proposed in (35) removes some samples. However, the solution quality heavily depends on the goodness of the filtering (as depicted in Figure 3.18b on the left). Additionally, the prefiltering output is not predictable a priori. There are some cases in which it does not produce any effect (as depicted in Figure 3.18b on the right). This means the path goes through the global manifold even if the prefiltering procedure has been performed. In contrast, the revised method is stabler and more controlled (no decision on global/local solution needed) because it is always intrinsically local, and the solution is known when $s = \infty$ or $s = 0$.

Additionally, some experiments on high-dimensional data sets are presented, which include the standard handwritten digits data set of the Modified National Institute of Standard and Technology database (MNIST)[15] (171), and the Olivetti faces data set[16] (172), highlighting the strengths of the proposed method. For this experiment, the following parameters were used: $N_c = 20$, $k = 10$. Figure 3.19 shows the results obtained with the revised method compared to those obtained with the original PP algorithm for different values of the regularization parameter s. In this case, the prefiltering procedure is always used. However, results obtained without the

[13] https://github.com/erikagardini/pp2/tree/master/datasets
[14] https://github.com/erikagardini/pp2
[15] http://yann.lecun.com/exdb/mnist/
[16] https://scikit-learn.org/0.19/datasets/olivetti$_f$*aces.html*

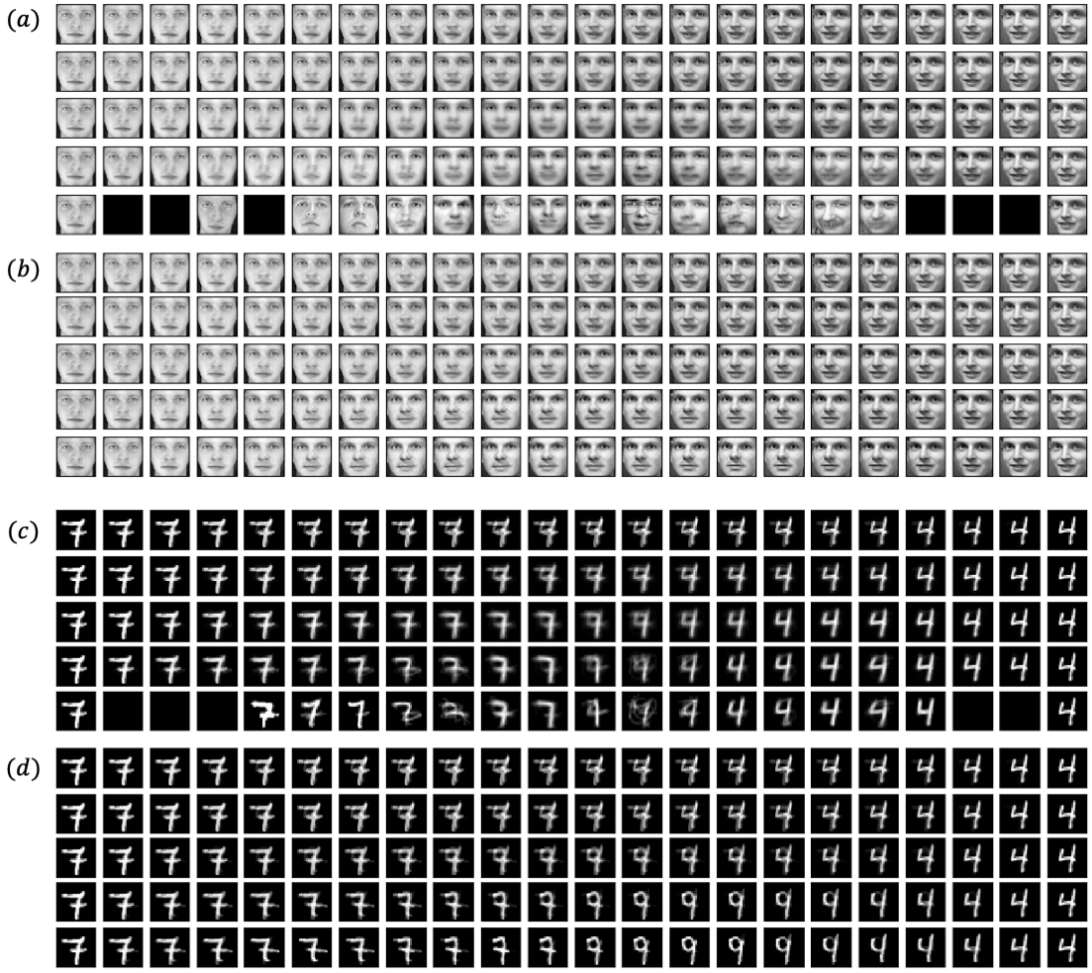

Figure 3.19: *Ab-initio local PP: morphing paths on Olivetti and MNIST data sets.* Comparison between the revised method and the original PP algorithm in high-dimensional data space. Each set of images reports the results for $s = 10000, 1000, 100, 10, 0$. (a) Original version of the PP algorithm on the Olivetti data set. (b) Revised version of the PP algorithm on the Olivetti data set. (c) Original version of the PP algorithm on MNIST data set. (d) Revised version of the PP algorithm on MNIST data set. When $s = 0$, the standard k-means algorithm is obtained in (a) and (c). The black figures represents the k-means centroids without any assigned samples.

prefiltering can be found in the GitHub repository of the project[17] together with other experiments with different starting and ending points. In all the experiments, the path runs outside the manifolds for $s = \infty$, when the it is equal to the straight line. This is clear when one moves from one digit to another. In fact, no intermediate digits are found; the intermediate waypoints are just an overlapping of the starting and the ending picture. A similar effect can be observed in the face experiment. In contrast, when s assumes low values, the stability of the new proposed path can be clearly seen. In fact, while the original path visits many different submanifolds,

[17] https://github.com/erikagardini/pp2

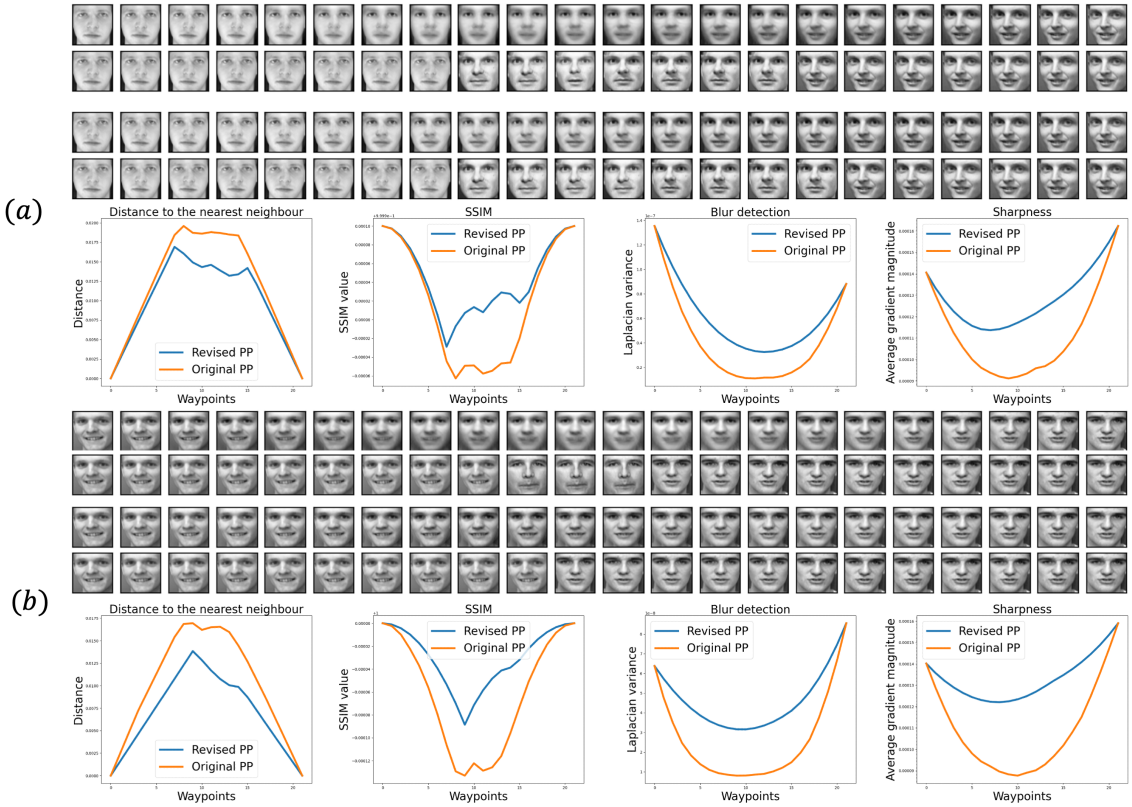

Figure 3.20: *Ab-initio local PP: quantitative and qualitative results analysis.* Comparison of the results from the original method with respect to the revised one. Two pairs of faces (two experiments) are considered, the first in a) and the second in b). For each experiment the generated images and the nearest neighbors are reported for both the the original algorithm and the revised method. The results are obtained with $s = 100$. Secondly, each generated face along the paths (the revised and the original ones) is quantitatively compared to the nearest one in data space in terms of distance and similarity (first two plots from left). Finally, the Laplacian variance and the average gradient magnitude are computed for each generated image in order to estimate the blur and the sharpness respectively (third and fourth plot from left).

becoming global, the revised method allows one to obtain the reparameterized Dijkstra path as a reference solution. Note that the original path solution is linked to the goodness of the prefiltering threshold, which can be challenging to guess for high-dimensional data. The smoothing effect is clearly visible for the solutions with intermediate values of the s parameter. However, when the original PP is used, a clear over-smoothing effect is visible, which produces blurred intermediate pictures. However, the solution seems more realistic when the revised method is used. This is because the intermediate waypoints are placed much closer to the existing samples with respect to the original method, thus more locally and through dense areas of the data space. This is numerically confirmed in Figures 3.20a and 3.20b (bottom left). As the proposed method allows one to obtain waypoints closer

to the original input samples, its generative capability is increased. This can be qualitatively evinced from Figures 3.20a and 3.20b (top), which show the nearest sample to each generated waypoint for the revised and the original path, respectively. In particular, in both examples, the newly generated pictures/people are not in the input data set. However, the proposed method is more powerful in generating more realistic facial expressions. This is particularly evident in Figure 3.20b (top), in which the proposed path moves from a smiling person to a serious-looking person by gradually closing the mouth, removing the tooth, and changing the expression of the eyes. The direction of the face is also gradually modified, rotating from left to right. A similar effect is obtained with the original PP, but the pixel area around the mouth is blurred in all the intermediate pictures, making the results less realistic visually.

To quantitatively compare the generated images with both methods, a commonly used metric for evaluating generative models is used, the SSIM (Structural Similarity Index) ([173]; [174]), which compares two images in terms of luminance, contrast, and structure. In this case, each generated image is compared to the nearest sample in the data space. Additionally, the quantity of blur in the image can be measured by considering the Laplacian variance and the sharpness. Figures 3.20a and 3.20b (bottom) show that in all the experiments the images generated with the proposed method have a more similar structure to the existing samples in the data space. Additionally, they have a higher Laplacian variance meaning that they have clear edges, representing a normal in-focus image. Finally, the images generated with the proposed method have a higher sharpness (estimated by the average gradient magnitude), meaning that the details are clearer and are more realistic than those generated with the original method. All these measures confirmed the results obtained with the previous qualitative analysis. Similar results have also been obtained for the other experiments and can be found in the GitHub repository of the project [18].

[18] https://github.com/erikagardini/pp2

Reproducibility

The experiments' implementation was written in Python, modifying the original PP code available at:

- https://github.com/mjf-89/PrincipalPath.

All the experiments can be reproduced using the code available on GitHub:

- https://github.com/erikagardini/pp2.

They were run on a laptop equipped with a 2,6 GHz Intel Core i7 and 16 GB RAM.

Final remarks

In this Section, a revised version of the PP has been presented, more robust and capable of providing more controlled and stabler solutions. In all the experiments, visually pleasant results are obtained, gaining interesting insights into how to morph two concepts in space. This aspect is particularly evident for the Olivetti and the MNIST data sets where, as for the experiments proposed in (35), the new algorithm morphed one figure into another along the transition path with meaningful and realistic intermediates.

The revised method, like the original one, is implicitly generative due to the presence of waypoints, even though a probability density function is unavailable. However, the generative potential of the PP is improved, with the suggested solution lying more on the space between a geodetic and a straight line. In this way, the waypoints are not attracted by nonlocal submanifolds that may corrupt the solution. For a particular value of s, one can find solutions that visually resemble those obtainable from GANs (70) and VAEs (71; 175). Indeed, one could argue that higher s values resemble the classical over-smoothing effect of VAEs, whereas small s values correspond to the GANs, which stay closer to the data manifold and thus deliver more realistic samples.

In terms of computational efficiency, the new method reduces the optimization cost because it is based on a very restricted set of samples, always equal to N_c. However, the initialization step requires computing a penalized distance matrix

among all the samples in the data sets. This is not scalable when the input matrix is high-dimensional and comprises several samples. In the future, techniques like the Nyström approximation can be used to improve the efficiency of this step.

Chapter 4

Import Vector Domain Description: applications and variants

This Chapter introduces and discusses the IVDD ([36; 37]). An application in the life science domain is presented, particularly in the drug discovery domain ([176]). In addition, a deep version of the IVDD algorithm is proposed to combine the kernel methods' advantages and DL strengths.

The following notation will be used throughout the Chapter:

- $\mathbf{X}$ is the $n \times d$ matrix of samples, where n is the number of samples and d is the dimension of the input space.
- $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d'}$ is the, possibly non-linear, transformation mapping the d-dimensional input space into a d'-dimensional transformed one.
- $\mathbf{K}$ is the kernel matrix.
- r is the radius of the embedding hypersphere, and $\Gamma = r^2$.
- $\mathbf{a}$ is the center of the embedding hypersphere.
- f_i is the decision function, i.e. the function that tells if a sample is inside or outside the hypersphere. If $f_i > 0$, the ith pattern is outside the hypersphere, otherwise is inside.

- p_i is the probability of the i-th sample being inside the hypersphere.
- $C \geq 0$ is an inverse regularization coefficient, and $\hat{C} = C/n$.
- $\hat{\mathbf{x}}_i = g(h(\mathbf{x}_i; \mathcal{W}_h); \mathcal{W}_g)$ is the reconstructed sample in a VAE (or AE) and h and g are respectively the coder and decoder functions parameterized by their respective weights.

4.1 Import Vector Domain Description: the method

IVDD is a one-class classification kernel method (36) inspired by the OC-SVM (56) and the SVDD methods (177). IVDD not only performs one-class classification using a hypersphere but also provides a probability estimation. It solves the following minimization problem:

$$\min_{\Gamma, a} \Gamma^2 - \hat{C} \sum_{i=1}^{n} \log(p_i), \tag{4.1}$$

where Γ is the square of the radius of the hypersphere, the constant $\hat{C}$ represents the trade-off between the radius size and the error minimization and p_i is the probability defined by a logistic model:

$$p_i = \frac{1}{1 + \exp(\beta f_i)}, \tag{4.2}$$

where f_i is the decision function defined as:

$$f_i = ||\phi(\mathbf{x}_i) - \mathbf{a}||^2 - \Gamma, \tag{4.3}$$

and β is a fixed coefficient. If the center $\mathbf{a}$ in the cost (Equation 4.1) is expressed as a linear combination of the input patterns:

$$\mathbf{a} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i), \tag{4.4}$$

than it is easy to show that the method can be set in the framework of kernel methods as only dot products involving $\phi(\mathbf{x}_i)$ are needed (36). In a reproducing kernel Hilbert space the dot product $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ can be implicitly computed by a single function evaluation $k(\mathbf{x}_i, \mathbf{x}_j)$ where k is a positive definite kernel function;

hence the knowledge of $\phi(\cdot)$ is not required. Thanks to this property one can expand the distance between the samples and the center as:

$$d(\mathbf{x}_i, \mathbf{a}) = \left\| \phi(\mathbf{x}_i) - \sum_{k=1}^{n} \alpha_k \phi(\mathbf{x}_k) \right\|^2 = \boldsymbol{\alpha}^t \mathbf{K} \boldsymbol{\alpha} - 2\mathbf{K}_{i,(\cdot)}\boldsymbol{\alpha} + k_{ii}, \tag{4.5}$$

where $\mathbf{K}$ is the kernel matrix of size $n \times n$, $\mathbf{K}_{i,(\cdot)}$ is the i-th row of the kernel matrix $\mathbf{K}$ of size $n \times n$, and k_{ii} is the self-similarity of the i-th sample. Figure 4.1 shows a schematic representation of the IVDD method. In (37), an efficient optimization algorithm is described that can be ascribed to the class of Sequential Minimal Optimization (SMO) methods (178). It combines with SMO features typical of EM algorithms (179) as the sub-minimization problem is solved via self-consistent iterations. The method has shown to be very effective; however, it suffers from the usual limits of kernel methods namely the scalability problem, as the expansion is carried over all the samples n. To improve the scaling, the Nyström approximation can be used (37), as anticipated in Chapter 2. In particular, a restricted number of samples can be selected a-priori from the original samples as landmarks. Then, one can assume that the decision function lives in the space spanned by these landmarks

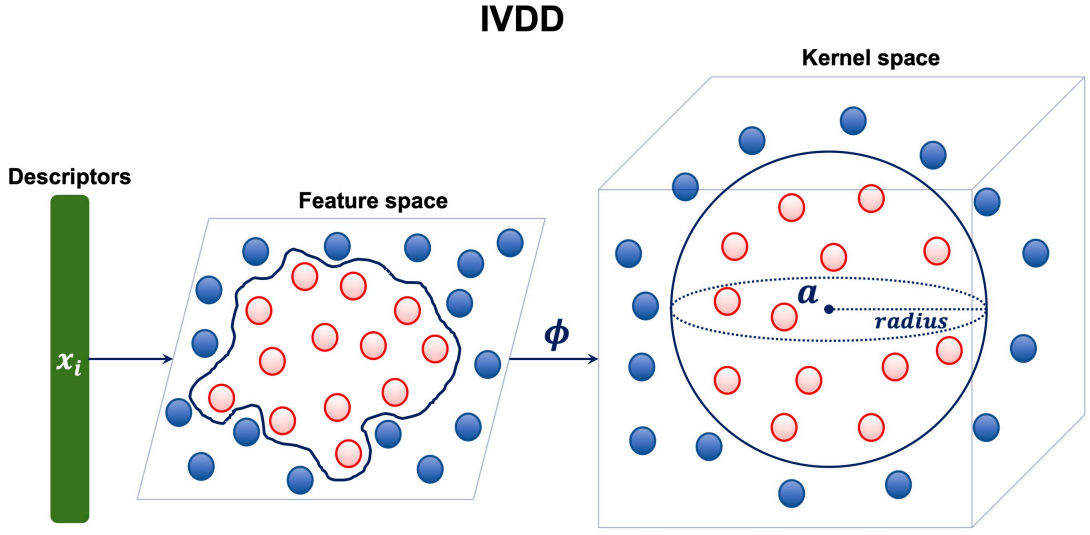

Figure 4.1: *IVDD method.* Each sample (*druggable* pocket) is a single point in a n dimensional space. The hypersphere is created in a kernel space. The mapping between the feature space and the kernel space is given by the function ϕ.

instead of the whole set of samples in the training set. As a consequence, the center $\mathbf{a}$ can be expanded in the landmarks subset as:

$$\mathbf{a} = \sum_{k=1}^{n_l} \alpha_k \phi(\mathbf{x}_k), \tag{4.6}$$

where n_l is the number of landmarks. The cost to be minimized in this case is therefore the following:

$$\Gamma^2 - \hat{C} \sum_{i=1}^{n} \log \left(1 + \exp \left(\beta \left(\left|\left| \phi(\mathbf{x}_i) - \sum_{k=1}^{n_l} \alpha_k \phi(\mathbf{x}_k) \right|\right|^2 - \Gamma \right) \right) \right). \tag{4.7}$$

Again kernel properties lead to:

$$d(\mathbf{x}_i, \mathbf{a}) = \left|\left| \phi(\mathbf{x}_i) - \sum_{k=1}^{n_l} \alpha_k \phi(\mathbf{x}_k) \right|\right|^2 = \boldsymbol{\alpha}^t \hat{\mathbf{K}}_r \boldsymbol{\alpha} - 2\hat{\mathbf{K}}_{i,(\cdot)} \boldsymbol{\alpha} + k_{ii}, \tag{4.8}$$

where $\hat{\mathbf{K}}_r$ is the kernel matrix of size $n_l \times n_l$, $\hat{\mathbf{K}}_{i,(\cdot)}$ is the i-th row of the kernel matrix $\hat{\mathbf{K}}$ of size $n \times n_l$, and k_{ii} is the self-similarity of the i-th sample. This version is much faster, bears a limited memory footprint and shows interesting regularization properties (37) confirming the findings in (60) for the unsupervised scenario. Yet it assumes an already engineered input representation; this point will be addressed in the methodological section of this Chapter.

4.2 Import Vector Domain Description for Drug Discovery

Drug discovery is a time-consuming and complex task (180). It requires a multistep pipeline from biological understanding to the finetuning of the lead candidate (for small molecules), often via computational means (181; 182). In the past 20 years, computation has significantly contributed to many drug discovery steps via physics-based simulation, ML modeling, and a combination of the two (183; 184). In particular, computational modeling can help find a putatively druggable target and hence a pocket that may accept a small molecule. A protein of interest is considered druggable when a drug has been found to inhibit it. However, some Authors consider

ligandability to be a more appropriate term for the propensity of the target/protein to accept drug-like molecules, irrespective of the more complex pharmacokinetic and pharmacodynamic mechanisms implied by the term druggability (185).

Hereafter, the term *druggable pocket* indicates a protein region with a high probability of accepting a small molecule. The reliable in silico identification of potentially druggable pockets is important for drug discovery. Finding new druggable *hot spots* can be particularly relevant when searching for an allosteric binder and to boost selectivity. Selectivity, in turn, is particularly important when designing chemical entities like PROTACs (186; 187), even more important than optimizing the affinity of the warhead itself. While researchers often know about the orthosteric pocket of a specific protein, it requires geometric and chemical insight to detect alternate druggable pockets, making it a much more complex task. Therefore, effective tools are required to support the computational medicinal chemist in detecting and ranking new pockets to design highly selective drugs.

The literature contains many reports on the computational estimation of druggability (188). The available tools for this task include standalone software (e.g., P2Rank (189)) and webservers (e.g., PockDrug (190)). Prediction often involves defining geometric and chemical features to support ML techniques (191) (e.g., DrugPred (192)). Alternatively, more recent DL methodologies often use 3D grids (voxels) of physico-chemical fields. Indeed, there are several methods for predicting the probability of a pocket's druggability. DoGSiteScorer (193) is an algorithm that detects pockets and estimates druggability by considering global and local pocket proprieties. It uses the SVM method to build a predictive model. PRANK (194) uses Decision Trees and Random Forests to re-rank/re-score the pockets predicted by other software, such as ConCavity (195) and Fpocket (196). PRANK could help improve the performance of existing prediction methods; PRANK aims to predict the ligandability of a specific point near the surface of the pocket. TRAPP is a powerful method for analyzing molecular dynamics trajectories. It was recently endowed with druggability assessment capabilities, extending its analysis to entire ensembles of structures (197). Druggability can also be assessed with pharmacophores (198) by using either straightforward geometric considerations (e.g., Cavity (199)) or fully

fledged DL approaches. There are many such DL approaches, which often leverage CNNs coupled to 3D grids. In (200), the Authors use both the pocket and the ligand with the DenseNet architecture. In contrast, (201) uses CNNs specialized for nucleotide and heme binding sites, again starting from 3D grids. InDeep (202) is another contribution based on convolutional architectures. Here, the focus is on characterizing protein-protein interfaces (PPI) to allow the design of PPI disruptors. The capabilities of CNNs were boosted by pocket segmentation in (203). This work and others (e.g., (204)) demonstrate that both prediction and other activities, like segmentation, are beneficial, so that one can devise a more complex framework than a pure predictor. Along these lines, PUResNet (205) uses an interesting deep residual (skip connections) decoder/encoder architecture derived from the U-net concept. This work presented both an architecture and a clean-up procedure for the training set. This class of deep methods is very accurate but lacks native interpretability.

From the protein dataset perspective, some datasets used in published papers are suitable benchmarks. They are often used to train and test ML protocols, thus creating a shared base. For instance, in (206), the Authors created an online dataset containing 72 unique protein binding sites. The Authors in (207) published two datasets: a large but redundant dataset (DD, with 1070 structures) and a non-redundant subset (70 binding sites).

In the following, the work published in (176) is presented. It introduces a protocol that employs the IVDD method to learn a hypersphere capable of enclosing druggable pockets. Here, the druggability estimation problem is addressed as a one-class unsupervised learning task and not a classification one. In this way, only a definition of a druggable pocket is required at training time, avoiding the induction of any bias due to the definition of the not-druggable class.

Method

The training phase of the protocol allows one to estimate (learn) the model and comprises three main steps (see Figure 4.2):

1. Computation of the descriptors for the proteins of the training set. In particular, for each protein:

(a) the protein part is filtered from the input PDB and the radii of the Amber99SB-ildn force field are assigned to it;

(b) the PDB is thus converted to a .xyzr file, then passed to NanoShaper to detect all the pockets;

(c) a main druggable pocket is identified (one for each training protein);

(d) the geometric-chemical descriptors of the pocket are computed.

2. All the information from the previous step is aggregated to form the training dataset, which is therefore composed of the descriptors of each main druggable pocket of the training targets.

3. Finally, the training dataset is used to train the IVDD ML method. In this phase, a hypersphere is learned and allows to assign a probability value to each pocket and consequently distinguishing druggable (probability ≥ 0.5) and nondruggable pockets (probability < 0.5).

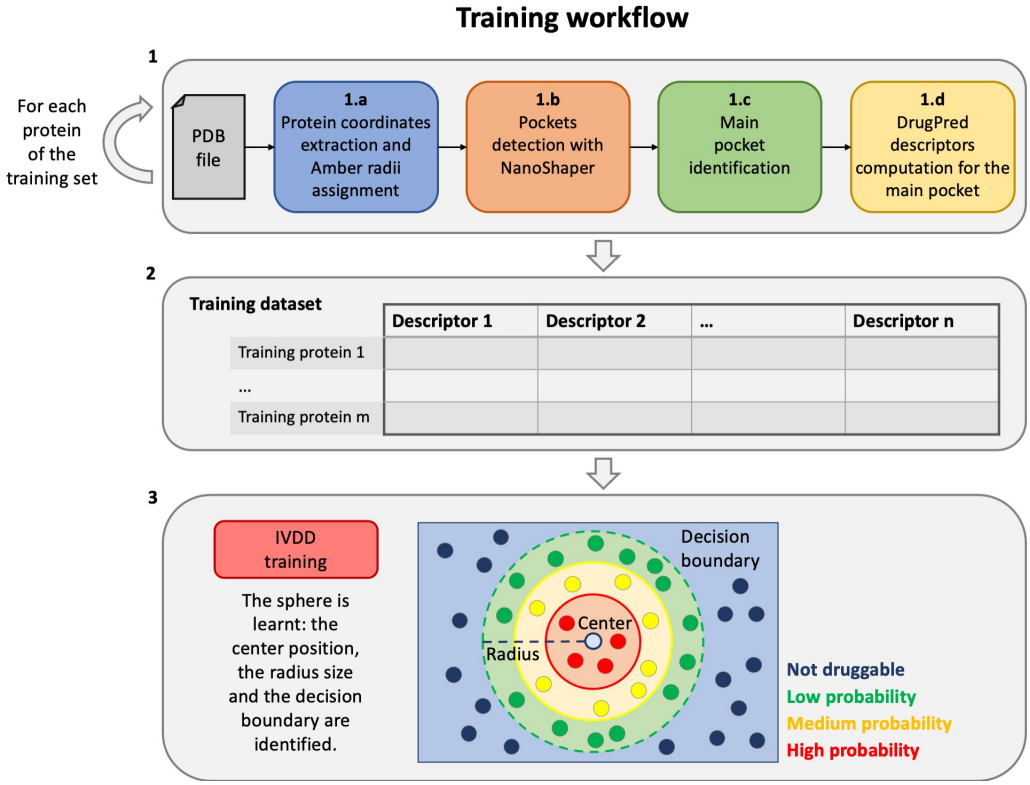

Figure 4.2: *Druggability prediction: training workflow.* From the creation of the training dataset to the training phase of the IVDD method.

On the other hand, the testing/operative protocol, that is when the model is used for predictions only, comprises three main steps (see Figure 4.3):

1. Computation of the descriptors for the current target protein. In particular:
   (a) the protein part is filtered from the input PDB and the radii of the Amber99SB-ildn force field are assigned to it;
   (b) the PDB is thus converted to a .xyzr file, then passed to NanoShaper to detect all the pockets;
   (d) the geometric-chemical descriptors of the pockets are computed.
2. All the information from the previous step is aggregated, obtaining a single file composed of the descriptors of each pocket of the current target.
3. Finally, the previously estimated hypersphere is used to predict the probability for each of the newly detected pockets. The pockets with the highest probability are most likely to be druggable.

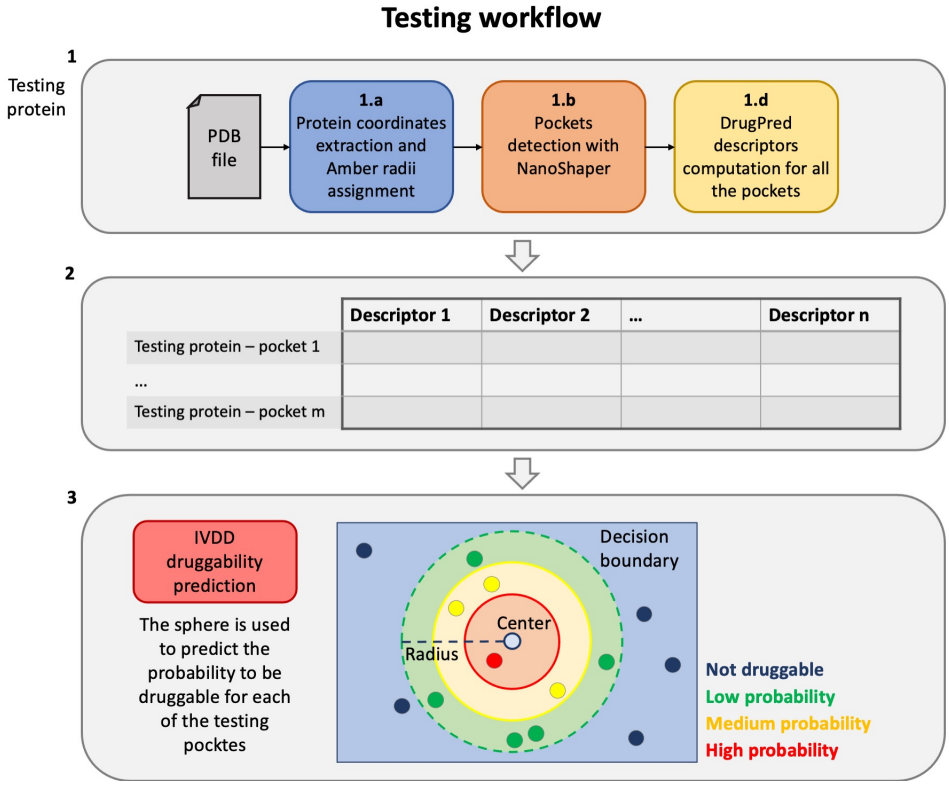

Figure 4.3: *Druggability prediction: testing workflow.* From the protein PDB file to the druggability prediction with the trained IVDD.

Detecting all the available pockets is instrumental in estimating the druggability of each pocket in the protein of interest. For this step, the NanoShaper tool is used ([208]) to deliver the set of pockets on a protein efficiently. NanoShaper was chosen as it accurately estimates the molecular surface ([209]); the detected pockets are triangulated with the same technique used for molecular surface triangulation, hence providing smooth triangulated meshes.

The detected pockets are saved as mesh files in MSMS or .off format, and can be easily parsed to support the subsequent descriptor building step. NanoShaper also provides the volume, surface area, and a list of constituting atoms for all the internal cavities and pockets identified for the given molecular system. These are identified and computed via an intuitive approach, which involves the volumetric difference of the regions of space between the system's solvent excluded surfaces (SESs), with two probe radii, dubbed large probe (with radius R) and small probe (with radius r) ([208]). The probe sizes encode the expectation onto the shape of the pockets. High R values allow the identification of shallow pockets, whereas high r values will smooth inner surface gaps. Default values are 3.0 Å and 1.4 Å for the large and small probes, respectively. The large radius is based on empirical evidence and the small radius mimics the water molecule. In this work, the default value of the small radius is used, but the large radius is fine-tuned to a value of 3.5 Å. With respect to the default value of 3.0 Å, the new one allows a better detection of slightly more shallow pockets (larger surface size of pocket entrance).

To create the training dataset, an automated method is needed to detect the orthosteric/main pocket (where the ligand is located) and discriminate it among the others delivered by NanoShaper. Because the orthosteric pocket is well-identified in the analyzed PDB, the surrounding atoms of the ligand are used. In detail, the Jaccard index on the atom indices is computed to detect the orthosteric pocket easily; the Jaccard index of atoms is an accurate proxy of Discretized Volume Overlap, often found in druggability predictors. The orthosteric pocket is defined as the pocket detected by NanoShaper with the maximal Jaccard index with respect to the reference indices. This is easily obtainable by localizing the atom indices around the

target's natural substrate (or drug). The Jaccard index is defined by:

$$J(O, P_i) = \frac{|O \cap P_i|}{|O \cup P_i|} \tag{4.9}$$

where O is the indices set for the orthosteric site and P_i is the set of detected atom indices in the i-th pocket. The Jaccard index is hence a natural measure of the quality of the detected pocket with respect to the ligand's envelope.
The Jaccard index can be decomposed into two components, which account for the degree of overimposition of the pocket and the reference ligand volume in two different ways. The first component is the normalized intersection component J_{int}:

$$J_{int}(O, P_i) = \frac{|O \cap P_i|}{|O|} \tag{4.10}$$

and the second component is the normalized union component J_{or}:

$$J_{or}(O, P_i) = \frac{|O|}{|O \cup P_i|} \tag{4.11}$$

They both belong to the interval $[0, 1]$. They account, respectively, for the ability to detect all the reference atoms (J_{int}) and for the tightness of the detection (J_{or}). Both properties are desirable and consistently lead to the Jaccard index upon multiplication.
To characterize each pocket identified by NanoShaper, the descriptors defined by (192) can be used, together with the entrance area provided by NanoShaper (Table 4.1). Binding site properties describing size, shape, polarity, and amino acid composition were calculated using NanoShaper output files as input to the descriptors builder. In particular, to compute the volume (vol), total surface area (area_b), and entrance area (area_e) (which describe the area of the pocket mouth), the estimations provided by NanoShaper are directly used. The other descriptors are computed starting from the NanoShaper output files, which describe the atoms and meshes of each pocket. The hydrogen-bond donor and acceptor properties of each pocket were calculated by considering the surface area surrounding all the polar atoms (dsa_t, asa_t). Based on these descriptors, the hydrophobic surface area (hsa_t) is defined as the difference between the total surface area and the sum of the hydrogen-bond donor and acceptor

Table 4.1: *Druggability prediction: descriptors of the datasets.* The incidence is calculated for every amino acid X.

Descriptor	**Abbr.**
Binding site volume	vol
Total surface area	area_b
Entrance area	area_e
Binding site compactness	cness
Relative hydrogen bond donor surface area	dsa_r
Hydrogen bond donor surface area	dsa_t
Relative hydrogen bond acceptor surface area	asa_r
Hydrogen bond acceptor surface area	asa_t
Relative hydrophobic surface area	hsa_r
Hydrophobic surface area	hsa_t
Relative occurrence of polar amino acids	paa
Relative occurrence of apolar amino acids	haa
Relative occurrence of multifunctional amino acids	maa
Relative occurrence of charged amino acids	caa
Relative polar surface area (dsa_r+asa_r)	psa_r
Incidence of X amino acid in binding site relative to surface	in_X

surface areas. Moreover, the relative amplitude of the hydrogen-bond donor and acceptor surface areas (dsa_r, asa_r) and the hydrophobic surface area (hsa_r) were computed by dividing each descriptor by the total surface area of the binding site. Finally, the relative polar surface area (psa_r) is defined as the sum between the relative hydrogen-bond donor and acceptor surface areas. To characterize the shape of the different cavities, the compactness descriptor can be used and it is defined by ([192]) as:

$$cness = \frac{4\pi \left(\sqrt[3]{\frac{vol}{\frac{4}{3}\pi}} \right)^2}{area\_b} \tag{4.12}$$

According to this equation, the closer the compactness is to 1, the more spherical is the pocket. The remaining descriptors, relating to the amino acid composition, were

calculated by considering the occurrence of different classes of amino acids grouped by their overall physico-chemical properties. In particular, all the amino acids were grouped into the following classes:

- Apolar: Ala, Gly, Val, Ile, Leu, Met, Phe, Pro
- Polar: Thr, Lys, Arg, Glu, Asp, Gln, Asn, Ser
- Charged : Lys, Arg, His, Asp, Glu
- Multifunctional : Trp, Tyr, His, Cys

To define the relative occurrence of hydrophobic amino acids (haa), polar amino acids (paa), charged amino acids (caa), and multifunctional amino acids (maa), the fraction of each group of amino acids with respect to the total number of amino acids comprising each cavity is computed. Finally, the incidence of each amino acid of type (in_X), defined as the sum of all the surface areas surrounding the X amino acids, is also considered as a descriptor.

When the descriptors are extracted, they can be used as input for IVDD algorithm. As anticipated, only the samples (pockets) that are known be to druggable are considered during the training phase, and the method tries to create an enclosing surface that contains all the training samples. The enclosing surface is endowed with a probabilistic model, which assigns the probability of belonging (or not) to the enclosing hypersphere.

When the training procedure is concluded, the hypersphere configuration (center position and radius size) that best minimizes the IVDD cost function (see Equation 4.1) is found, meaning that as many samples as possible are kept inside the hypersphere controlling at the same time the radius size. The range of acceptance of the fraction of training examples inside the hypersphere is called $[\pi_{low}, \pi_{high}]$. It can be shown that the optimal hypersphere (the solution to the minimization problem) is unique, as the problem is convex.

Once the final hypersphere configuration is found, it determines the predictions during the operative phase. The non-druggable nature of a pocket is just an interpretation of the probability values; strictly speaking, one-class learning just describes

the adherence of a sample (a pocket) to a concept (druggability). If a crisp classification is needed, the probability threshold of 0.5 can be used. Samples outside the hypersphere (decision boundary) are predicted as non-druggable (with a corresponding probability lower than 0.5). In contrast, samples inside the hypersphere are predicted as druggable (with a corresponding probability higher than 0.5). Clearly, the inner and most central pockets are estimated to have the highest probabilities of being druggable. Indeed, this probability is high at the core of the hypersphere and decreases towards the edges.

Datasets

Two different datasets were considered for all the experiments, and two different versions are generated: with and without hydrogens atoms. The first dataset is the NRDLD dataset, presented in (192). It is the largest publicly accessible non-redundant dataset for model building and the validation of structure-based druggability methods. The dataset comprises 115 structures (protein binding sites), including 71 druggable and 44 less druggable (which becomes 42 after the analysis in (192)). Following the aforementioned strategy, 35 different descriptors are calculated for each binding site (see Table 4.1). In addition to the NRDLD set, another dataset was created comprising the binding sites of 100 different proteins. Those targets are taken from the PDTD (Potential Drug Target Database) (210), a free online collection of 1100 3D structures of proteins. The targets in the 100-protein dataset include enzymes, receptors, antibodies, signaling proteins, and lipid binding proteins, obtaining 5692 and 4807 binding sites without and with hydrogens, respectively. Of these, 100 are orthosteric (one for each target). For each structure, the pocket that hosts the drug or substrate is selected and defined as orthosteric (or main) (because the drug is often co-crystallized in the orthosteric site). As for the NRDLD, 35 different descriptors are obtained (see Table 4.1). See Appendix A for more information on the targets of the NRDLD and the PDTD datasets.

Results

This Section describes the experiments performed with the IVDD on the above mentioned datasets. The descriptors were computed on a HPC cluster equipped with a nVidia Tesla V100 32Gb GPU and 2 Intel(R) Xeon(R) Gold 6240 @ 2.60GHz Cascade Lake CPUs. IVDD was run on a laptop equipped with a 2,6 GHz Intel Core i7 and 16 GB RAM. The implementation was written in Matlab and Python.

First, the IVDD objective (see Equations 4.1, 4.2, 4.3, and 4.4) was trained considering the descriptors of $n = 70$ druggable structures in the NRDLD set. The *1nvj* (PDB code) structure was excluded since it is a small oligonucleotide, while this work considers only proteins. The following parameters were adopted: the kernel used is the RBF with $\sigma = \max_{ij}(d_{ij})/\log(n)$ (where d_{ij} is the distance between i and j samples), the value of C is initialized at the value 0.5, the value of β is set as 25, while the range of accepted inner samples is set to $[\pi_{low}, \pi_{high}] = [0.8, 0.9]$. The values of $[\pi_{low}, \pi_{high}]$ may vary according to the reliability of the training dataset. In this case, a conservative approach was preferred, with 80-90% of samples included inside the hypersphere and the remaining peripheral 20-10% as outliers, in order to avoid overfitting. The learning phase is stopped when the range of inner samples is hit. Each time the training is repeated, the C is increased/reduced by 0.01 (increased if the percentage of samples inside the hypersphere is lower than the desired range, reduced otherwise). In this experiment, the training procedure ended with 90% of samples inside the hypersphere and a final C value of 0.1 for the solution without hydrogens, with 90% of samples inside the hypersphere and a final C value of 0.12 for the solution with hydrogens. Figure 4.4 shows a 2D representation of the training set obtained by reducing the dimensionality via PCA ([160]). For some samples, the corresponding 3D structure is also provided. In both cases, most of the training samples coherently obtained a high probability of druggability (dark red points in Figure 4.4). This outcome is obtained because the solution is forced to include at least 80% of the training samples inside the hypersphere.

Considering the solution without hydrogens (Figure 4.4a), the sample *1udt* has the highest probability and is the sample nearest to the center of the hypersphere. In this structure, the pocket identified by NanoShaper is very compact and well-defined.

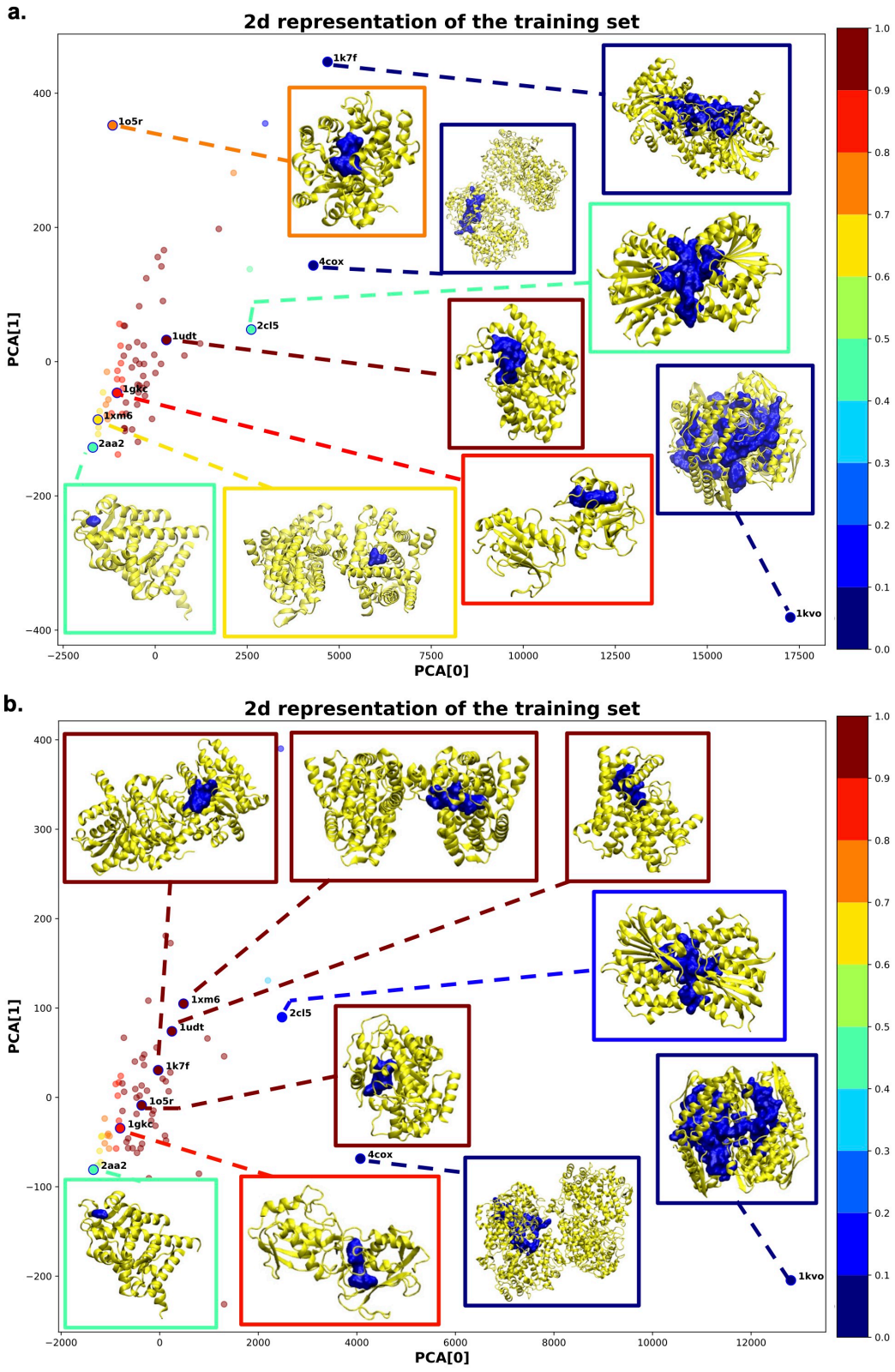

Figure 4.4: *Druggability prediction: 2D representation of the training samples via PCA dimensionality reduction.* Each point corresponds to a training sample (protein binding site). The color of each point corresponds to the probability assigned by IVDD (graded according to the color map on the right). For some training samples, the corresponding 3D structure is shown. Panel a) is without hydrogens whereas in b) hydrogens were added.

IVDD performs the best in cases where the pocket closely surrounds the ligand bound in it. The samples outside the hypersphere (corresponding to 10% of the samples) obtained low probability scores. These scores are explainable by looking at the pocket shape. Structures like *1kvo*, *4cox*, and *1k7f* do not look like well-defined pockets, but rather like a fusion of more than one pocket. This leads to descriptors that are quite distant from those that the algorithm is learning as the druggable reference. As a consequence, those structures are scored as outliers. This highlights that the ex post segmentation can be a powerful pre-processing tool before the ML step. Nevertheless, IVDD can cope with this situation by excluding or marginalizing percolating pockets. It is possible to identify another case where NanoShaper did not correctly identify the orthosteric pocket (e.g., *2aa2*). Here, the pocket is very shallow and the bound ligand is not deeply buried. The identified pocket is much smaller than it should be, leading to a low probability. This effect is expected because NanoShaper can only detect shallow pockets via a proper tuning of the big probe, whereas the selected value is expected to work mainly for deep buried prototypical, pockets.

The solution with hydrogens (Figure 4.4b) identifies the sample *1xm6* as having the highest probability. In contrast to the solution without hydrogens, its structure is now more compact around the ligand with a greater J_{int}. Since the presence of the hydrogens better defined the orthosteric pocket, NanoShaper improved its accuracy, leading to a high IVDD probability. This happens similarly for *1k7f*, where the channel that led to a big pocket was closed by the presence of the hydrogens. In this specific case, NanoShaper identified the orthosteric pocket with a Jaccard index three times better than the solution without hydrogens. Although the solution with hydrogens solves some NanoShaper errors (wide percolation), pockets such as *1kvo*, *4cox*, and *2aa2* remain more or less unchanged, with very big or shallow structures. The option to use hydrogen atoms (or not) is partially data-dependent and is further studied in the NRDLD and the new dataset.

After this training phase, a testing phase follows. The first experiment is performed considering the 42 less druggable structures described in (192). Figures 4.5 and 4.6 show the probability assigned to each structure by the IVDD method for the

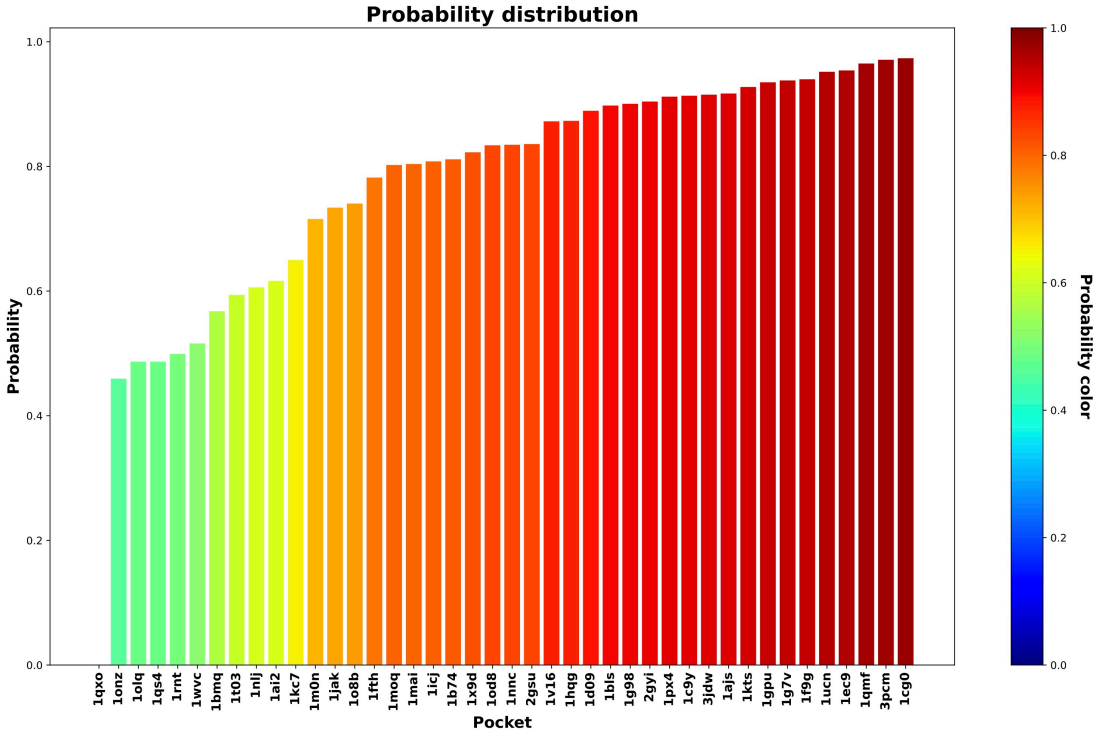

Figure 4.5: *Druggability prediction: experiment on the less druggable subset of NRDLD without adding hydrogens.* For each protein binding site (x axis), the druggability probability is predicted (y axis and color of the bar).

solutions without and with hydrogens, respectively.

Generally speaking, the two results are relatively similar. The resulting trend shows that IVDD predicts a probability greater than 0.8 for around half of the less druggable set. This points to a possible bias in the *less druggable* set. Indeed, a purely unsupervised approach like this one, in which no a priori dichotomy is created, shows that several of the pockets are not judged to be less druggable. On the contrary, more than half are scored with high probability values. The less druggable nature can be ascribed partially to the shallow nature of this set; however, thanks to the large probe set to 3.5 Angstroms, NanoShaper can still detect them.

This result hence partially contrasts with the *less druggable* labeling of this dataset. One should consider the principles behind this previous classification. Authors (192) postulate that a protein (not just the pocket) can be ascribed to the less druggable realm if none of the two conditions are met: 1) at least one ligand is orally available as judged by the Lipinski's rule-of-five 2) the ligands must have a clogP ≥ -2. Addi-

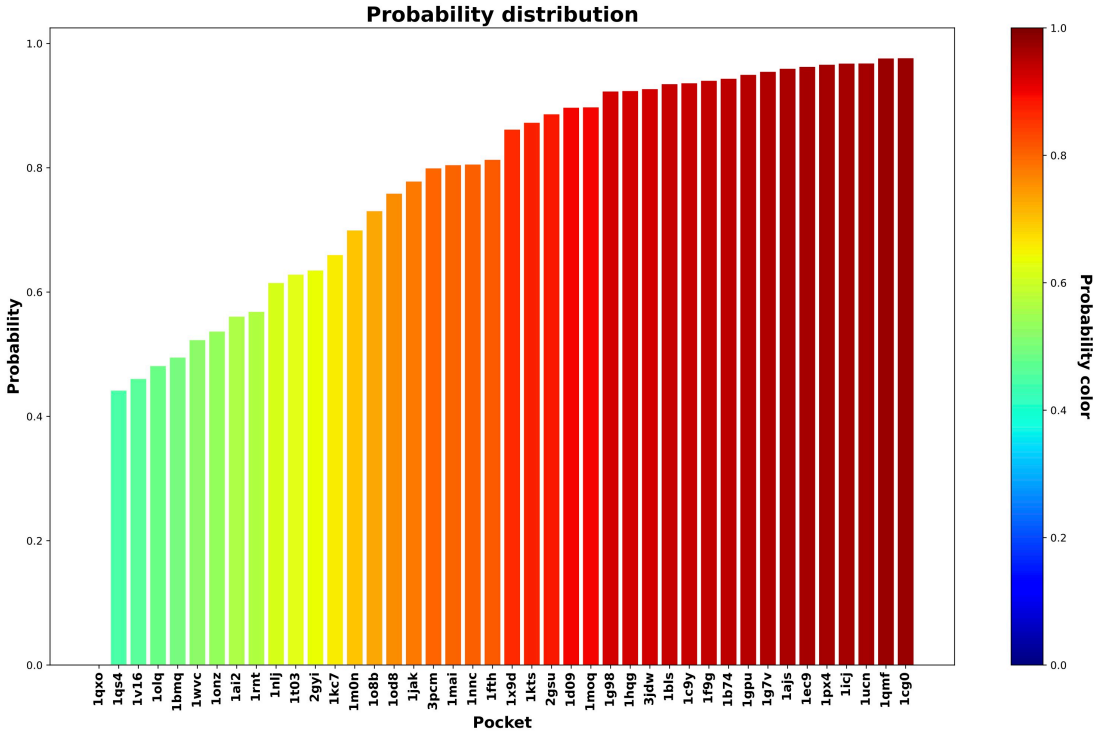

Figure 4.6: *Druggability prediction: experiment on the less druggable subset of NRDLD with the addition of hydrogens.* For each protein binding site (x axis), the druggability probability is predicted (y axis and color of the bar).

tionally, the ligand efficiency of at least one of the ligands fulfilling criteria 1) and 2) must be ≥ 0.3 kcal mol^{-1} per heavy atom. To correctly fulfill the requirements, one should be able to test all the chemical space before making any conclusion. Indeed, ideally, and more correctly, one could define the *true druggability* of a pocket as the value of the activity of the best possible ligand for that pocket in the chemical space. As the sampling of the chemical space is limited, and further biases are due to the drug discovery Community's interest and efforts for a specific protein, this classification is questionable and not necessarily reliable. The problem of a druggability classification of a pocket, or a protein, that is ligand-dependent hence is that it would require the true sampling of the chemical space. In the proposed method, instead, the labels are not defined a-priori and the attention was focused on the only reliable information that is, druggable pockets. The final result of this is that some pockets previously labeled as less druggable instead obtain high druggability probability values.

It is interesting to analyze the probability shift from lower to upper values, systematically. Figure 4.7 shows the orthosteric pockets found by NanoShaper for the less druggable proteins, the structures set was sub-sampled with a ratio of 1 every 5 complexes. The pockets tend to become deeper and more compact, moving from the lesser probability to the higher. The shift is particularly evident comparing the *1onz* and the *1cg0*, where the first case is a very shallow pocket, in which a ligand can be found, but it is neither a prototypical nor ideal pocket; as such, its probability value is 0.46. In contrast, the *1cg0* shows a much better defined and large enough pocket that would host a potential ligand well; as such, IVDD classifies it as druggable with a probability value of 0.97. Except for *1qxo* (a pocket detected by NanoShaper that is too large), one observes that the lower the score, the smaller and more shallow the pocket will be. This is also evident looking at the portion of solvent exposed surface of the ligands, where the low probability pockets tend to have more solvent floating ligands.

There are some particularly interesting cases in this less druggable set, also considering the ligands found in the crystal structures. In *1kts*, *1gpu*,*1ucn*,*1cg0* the ligands are small molecules or small molecules-like ligands. Missing these pockets would be quite negative in a drug discovery campaign. All these pockets score

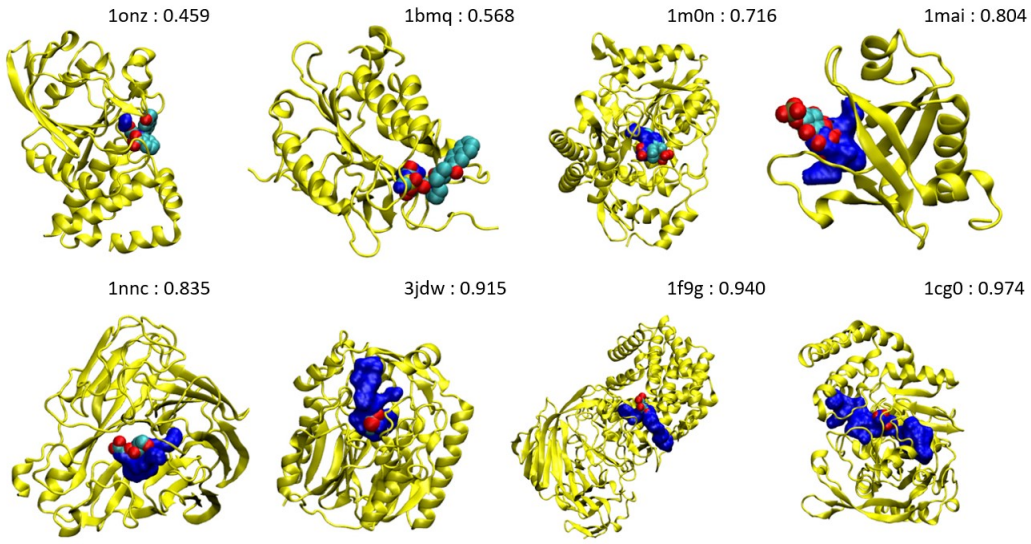

Figure 4.7: *Druggability prediction: main pockets (computed without hydrogens) of 1onz, 1bmq, 1m0n, 1mai, 1nnc, 3jdw, 1f9g and 1cg0.* The pocket surface is in blue and the complexed ligand in the pdb file is in VdW style. The number is the estimated druggability probability value.

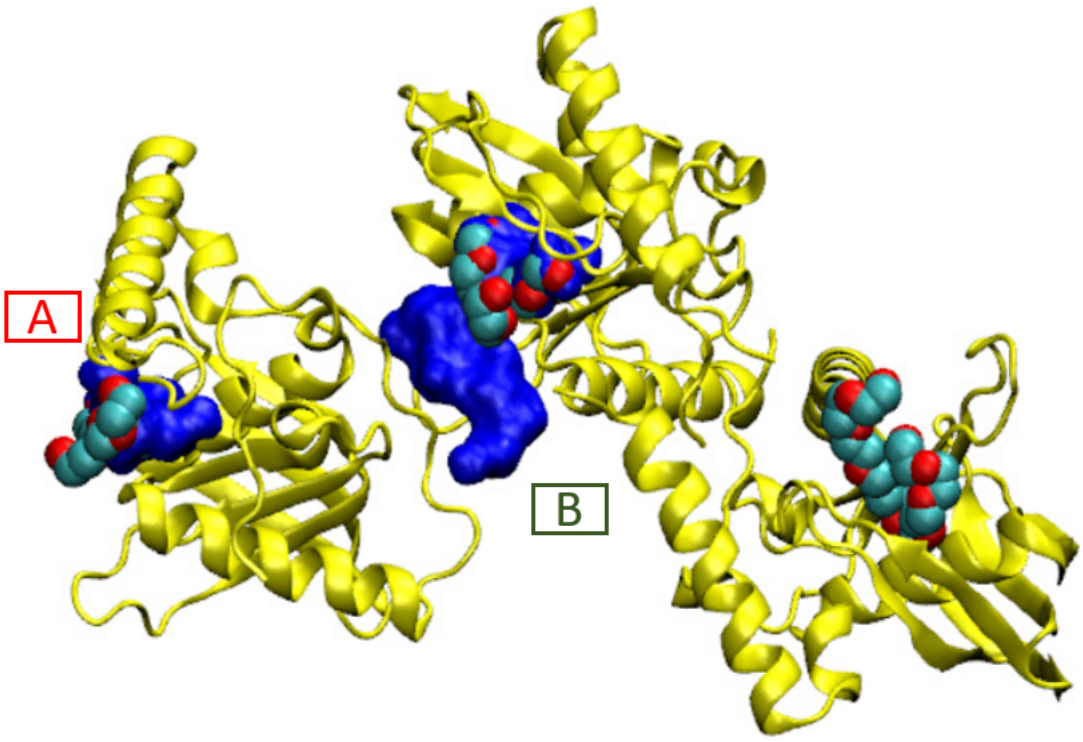

Figure 4.8: *Druggability prediction: main pocket shift for 1icj together with the co-crystallized ligand.* A) Main pocket detected when adding hydrogens. B) Main pocket without adding hydrogens. The pocket is semantically the same orthosteric pocket but changes from one monomer to another. The three structures of ligands bound in the pdb structure are also reported

quite high with the proposed method. One should also consider other than the pure small molecule paradigm; in the case one is concerned with the design of a molecular glue or a PROTAC even a warhead that is relatively not too active could be sufficient to degrade the protein. Hence the proposed method, which is agnostic to ligand-induced labeling, avoids missing or undervaluing these pockets. At a technical level, it is interesting to compare the pocket probabilities with and without hydrogen addition and to consider NanoShaper's behavior. As anticipated, adding or not adding hydrogens does not change the detection of the main pocket by NanoShaper (highest Jaccard index). However, the shape and the relative probability ranking both change. A first observation is that, in some peculiar cases, the percolating behavior of NanoShaper pockets cannot be solved by adding hydrogen atoms. Indeed, *1qxo* is still ranked last, and coherently, this pocket is percolating widely inside the protein crevices. This global invariance is confirmed by analyzing *1icj* (see Figure 4.8). In this case, the detection of the main pocket is geometrically but not

semantically changed by changing from a structure with hydrogens to a structure without hydrogens. That is, the main detected pocket is the same but in another monomer of the homotrimeric unit. Despite this finding, its druggability probability changes when adding hydrogens. This demonstrates that the same pocket in two different conformations (monomers) is well-detected and always ranked as druggable. Indeed without hydrogens, the orthosteric pocket is identified in monomer A. Upon addition of the hydrogens, the orthosteric pocket is instead identified in monomer B. In this last case, the Jaccard index is higher with improved pocket quality (the pocket is more compact and located at the interface). However, the probability value changes as the corresponding geometry (and presence or absence of hydrogens) changes, leading to a higher value for pocket B. Therefore, from one side, what is judged druggable remains druggable. However, inside the druggable set, conformational changes of the same pocket have a nontrivial role in shifting the probability value. This confirms that it is crucial to consider dynamical aspects, particularly the probability of a given site conformation (and hence its free energy), in order to obtain a complete picture of the overall druggability of a site, which may be dealt with as a physical observable. Overall, this analysis shows that the dataset definition can create nontrivial biases, including biases due to labeling and the presence or absence of hydrogens, which can induce local changes. One-class learning can mitigate the first bias because it only uses the druggable class during training.

The second experiment is performed using the 100-protein dataset, which is a curated subset of the PDTD dataset. This time, the accuracy of classification is evaluated, but also other possible sources of biases are investigated.

It is well-known that the volume value has a crucial role in determining the druggability of a site. However, just looking at the volume value may create further biases, some intrinsic, some operational, and some technical. An overly large volume could be erroneously ascribed to the main site just because a small fraction contains the true binding site. This can happen dependng on the pocket detection engine (e.g., for the percolation effect). Fortunately, this can be evaluated well via overlapping volume metrics or the Jaccard index. Here, the results obtained with the proposed method are compared to the ones obtained by considering a simple descending ranking of the

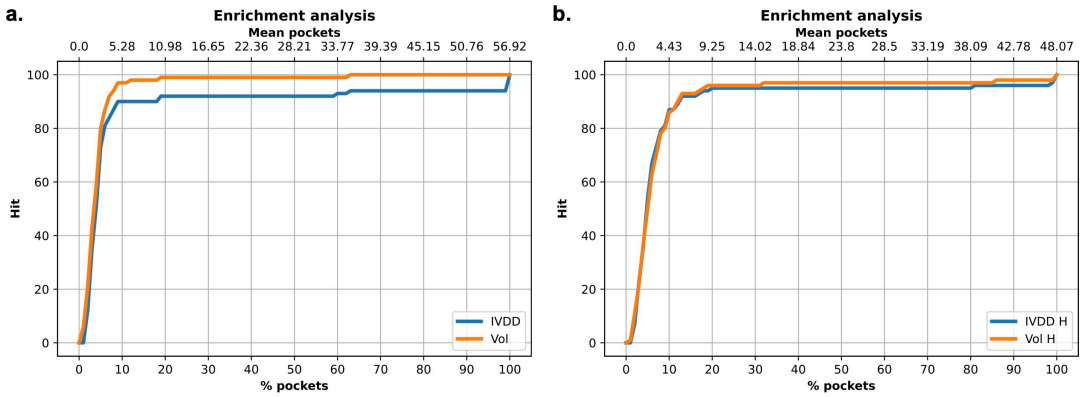

Figure 4.9: *Druggability prediction: enrichment analysis on the 100-protein experiment.* a) Solution without hydrogens. For the 10% of pockets (for each protein) with the highest probability (on average 5.28 pockets), the orthosteric site is found in 90% of cases with IVDD and in 97% of cases with the descending ranking of the pocket volumes. b) Solution with hydrogens. For the 10% of the pockets (for each protein) with the highest probability (on average 4.43 pockets), the orthosteric site is found in 87% of cases with IVDD and in 86% of cases with the descending ranking of the pocket volumes.

pocket volumes. Figure 4.9 and Table 4.2 show the results for the situations with and without hydrogens.

Using a simple ranking of the volume, a better performance at top5 can be obtained, with an accuracy of 97%. This decreases to 89% when hydrogens atoms are added. In contrast, IVDD identifies 90% of the orthosteric pockets in the top5 highest probability pockets, which increases to 92% when hydrogen atoms are added. This shows that IVDD is more stable, although lower in accuracy in absolute terms.

It is important to consider the quality of the pockets identified in both cases. The

Table 4.2: *Druggability prediction: results obtained on the PDTD dataset (with and without hydrogens) with the IVDD method and by a simple descending ranking of the pocket volumes.* All results are referred to the orthosteric/main sites.

Description	IVDD	Volume	IVDD+H	Volume+H
Top 1	50	60	50	50
Top 2	67	76	69	65
Top 3	81	87	81	79
Top 5	89	97	92	89
Top 10%	90	97	87	86

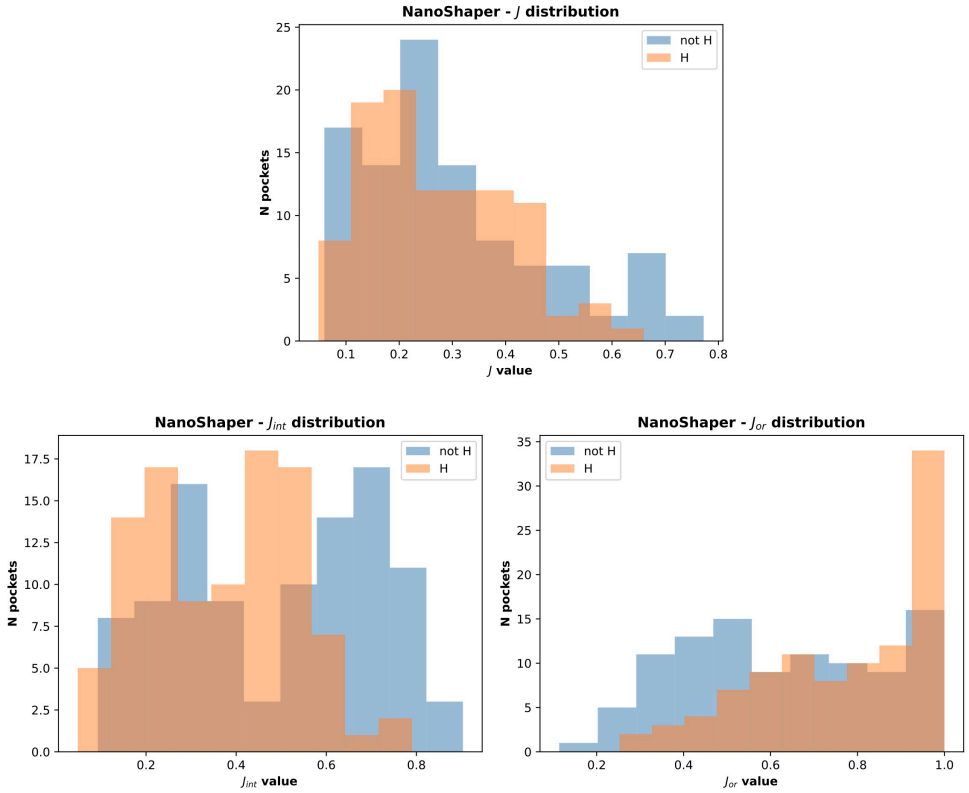

Figure 4.10: *Druggability prediction: NanoShaper scores distribution with and without hydrogens.*

presence of hydrogens sometimes allows the fragmentation of some of the overly large pockets. This increases the accuracy in terms of the main pocket druggability estimation but also affects the overall shape, which often becomes too tight. This is a NanoShaper-dependent effect, documented in Figure (4.10). This effect is combined with the volume and the IVDD classifier. Figure 4.11 shows the cumulative scores, namely J, J_{int}, J_{or}, for the volume and the IVDD ranking for the top1 pockets, ordered respectively by volume and probability. The trend shows a systematically higher value for all three scores for IVDD without hydrogens and almost indistinguishable scores with hydrogens.

Interestingly, without hydrogens, IVDD has a lower accuracy than the simple volume. This is unsurprising since an overly percolating volume allows an easier main pocket detection. However, when quality is considered, even if some pockets are lost with IVDD, the remaining pockets have significantly higher scores. Again, with the proposed method a bias can be mitigated by not overfitting the volume-induced

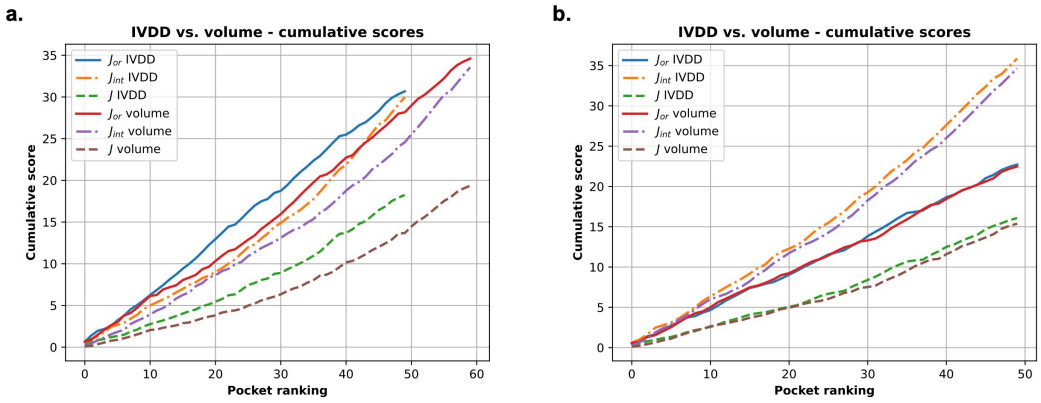

Figure 4.11: *Druggability prediction: cumulative scores (J, J_{or} and J_{int}) for IVDD and volume ranking.* Here, the orthosteric sites identified by IVDD and the volume ranking in top1 are considered and ranked according to the probability score and the volume respectively. Both the rankings are in descending order. Inset a) is results without hydrogens and inset b) is with hydrogens.

ranking. In the paradoxical case where one has a volume percolating throughout the protein, one would get a completely useless top1 with 100 % accuracy by using a pure volume ranking.

Within the IVDD results, it is also relevant to compare what happens with and without hydrogens. Examining the structures that did not land in the top5 positions with and without hydrogens, one can conclude that most (e.g., *1vkg*, *1qpb*, *1ht8*) are large pockets with low or intermediate Jaccard index or with very low J_{or} value. In some cases, there are shallow pockets (e.g., *1gp6* and *1i7g*) characterized by very high values of J_{or}. Some of those structures improve in the presence of hydrogens, reducing the number of targets that fall outside the top 5 from 11 to 8. Some shared structures (e.g., *1ht8*, *1h9u*, and *1v8b*) do not change the shape of the orthosteric pocket, leading to no significant changes in the probability.

Avoiding some of the possible biases (chiefly the labels) and considering the model without hydrogens, 81% detection accuracy in top3 and 89% in top5 can be obtained with the proposed solution. Additionally, a non-negligible fraction of the missed detections in top5 can be ascribed to NanoShaper's behavior. In comparison, the Authors of (211) obtained 88% accuracy in correctly assigning to the druggable or nondruggable class in the NRDLD with the software DoGSiteScorer, where the SVM method is used as ML backend. In contrast, DrugPred (192) obtains 91% accuracy for NRDLD. A widely used method is fpocket from Guilloux et al. (196),

which correctly identified 83% of ligandable pockets in top3 of all analyzed proteins. Overall, the accuracy obtained with the proposed method is similar to that of several existing methods, but with some ab inito safeguards such as avoiding biases due to labels and volume.

To further interpret the IVDD results, it is interesting to investigate how much each single feature affects the IVDD prediction. IVDD does not embed a feature selection method, so an ex post labeling strategy has to be used. First, one can estimate the probability obtained, on average, for each orthosteric site in the dataset, obtaining 0.852 and 0.877, respectively, without and with hydrogens. These values represent two thresholds and allow a labeling for each binding site, which is 0 when its probability is lower than the threshold value, and otherwise 1. This ex post labeling allows to fit a classifier and to estimate the feature importance. Here, tree-based classifiers are adopted since they can provide features' importance without requiring any feature normalization step (regression based methods require normalization before usage). In particular, the Random Forest ([212]) (with 100 estimators and the Gini index as criteria for the split), the Decision Tree (default parameters in scikit-learn ([213])) and the AdaBoost (default parameters in scikit-learn ([213])) classifiers are adopted and the results are depicted in Figure 4.12. In all cases, the volume (Vol) is a major impacting feature, followed by the area of the pocket surface (Area_b), the hydrophobic surface area (hsa_t), the hydrogen bond acceptor surface area (asa_t), the hydrogen bond donor surface area (dsa_t), the binding site compactness (cness), and the entrance (mouth) surface area (Area_e). This means that IVDD is influenced by the volume, but it also considers other chemical aspects in predicting probability. Of less relevance is the fact that hydrophobic residues (LEU, PHE, MET, GLY) and some charged residues (HIS, GLU) rank slightly higher. The presence of hydrophobic residues and volume as key factors is largely consistent with chemical intuition.

The correlation between the IVDD prediction and the volume can be seen in Figure 4.13, in which each binding site is depicted as a point in a 2D space, where the coordinates are the probability predicted by IVDD and the volume itself. In the presence (see Figure 4.13) and absence (data not shown) of hydrogens, the samples with the highest probability have a volume between 500 and 2000 Angstroms. The

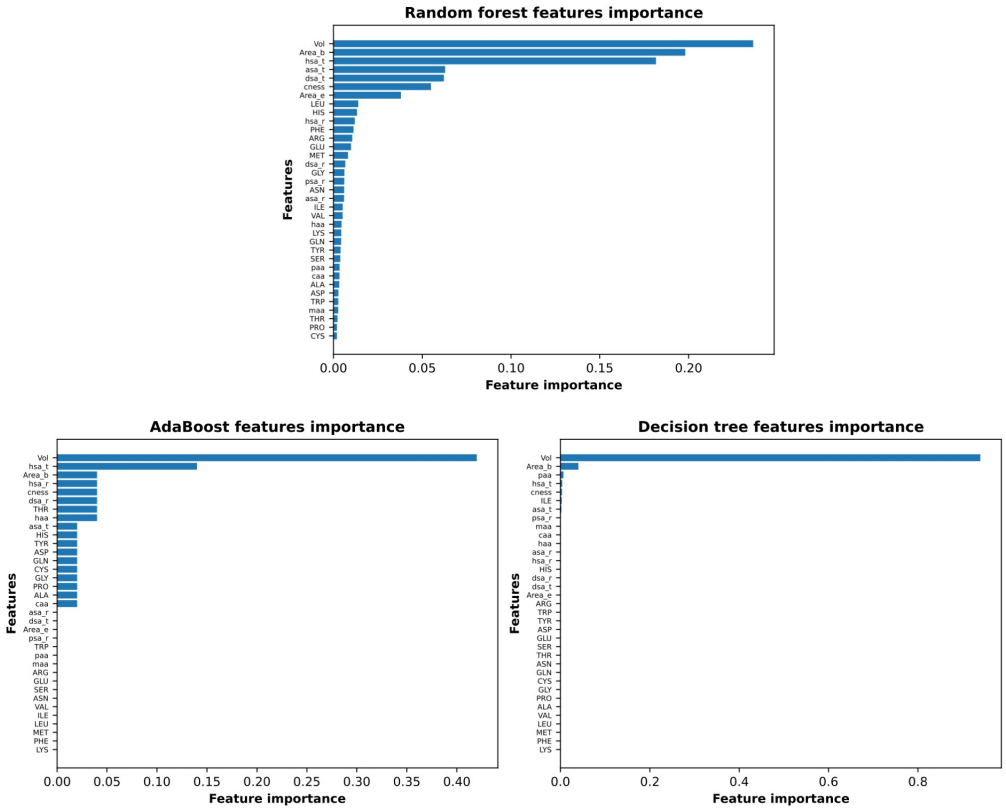

Figure 4.12: *Druggability prediction: features importance by assigning ex post labels to the IVDD predictions.* Results shown are without hydrogens; similar results are obtained with hydrogens.

orthosteric sites and the training samples are condensed on the right side of the figure, meaning that they obtained high probability scores in most cases. Not orthosteric binding sites are condensed in the bottom left of the figure since they are mostly small pockets and obtain low probability scores. However, both figures contain some not orthosteric pockets with a volume between 1000 and 2000 and lower probability scores. In such cases, the IVDD decision has been influenced by factors other than volume.

Final remarks

In this Section, an unsupervised one-class approach is presented to build a druggability estimation model. A pipeline is defined to obtain all the pockets of a protein (NanoShaper), their corresponding descriptors, and the druggability prediction. The method achieved 89% accuracy in top5, in line with other methods. Although the method was less accurate than a trivial volume-based ranking by NanoShaper, it

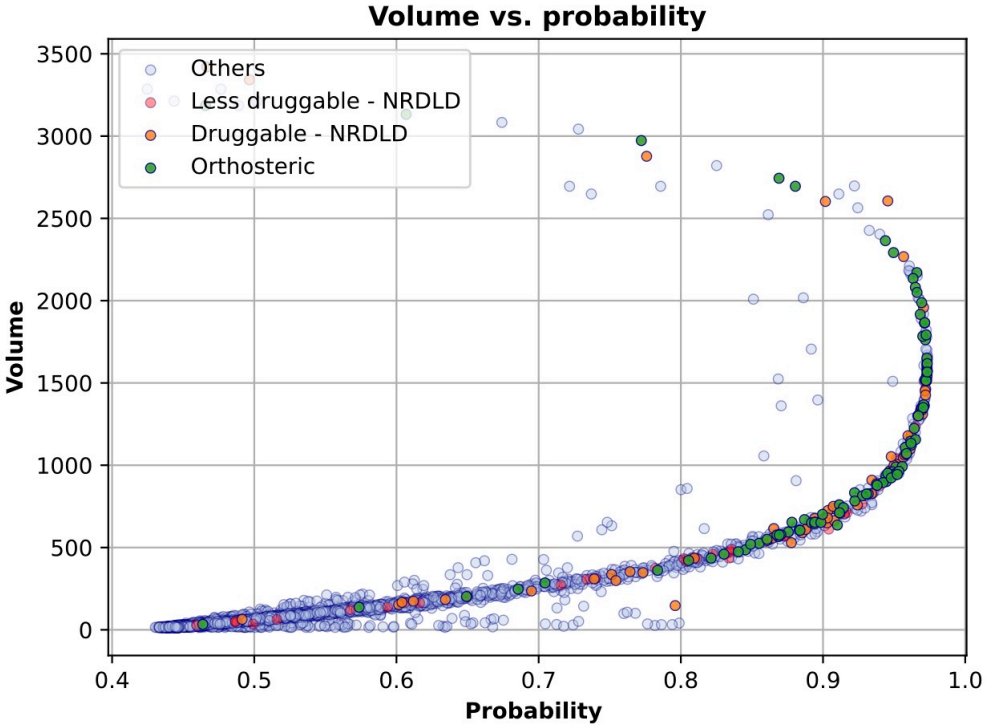

Figure 4.13: *Druggability prediction: IVDD probability scores vs volumes.* Each sample represents a pocket (colored according to the corresponding dataset). The x-axis represents the probability that a pocket is druggable, while the y-axis represents the volume of each pocket. The plot is referred to the solution with hydrogens. Similar results are obtained without hydrogens.

favors well-shaped pockets with higher J, J_{or} and J_{int} scores. This has practical relevance since a relatively tight and well-shaped pocket reduces the ambiguity and difficulty of the subsequent virtual screening and docking campaigns.

Crucially, the proposed method does not aim to distinguish between druggable and less druggable pockets (binary classification). Instead, a probability estimation for each pocket is given, which is easily interpretable and comparable across different proteins. In contrast to a score, the probability estimation does not need a posteriori calibration. Instead, the logistic model of the hypersphere naturally delivers this information. Again, a probability allows the computational medicinal chemist to easily identify the most eligible pocket for subsequent drug discovery steps, without wondering if the score value is high or low in absolute terms. This is because any probability very close to 1 is inevitably a strong indicator. Most importantly, this approach does not need to define a less druggable or nondruggable class. This

potentially ambiguous concept is bypassed by the one-class approach. The results show that druggability prediction is best considered as a concept learning problem, rather than a distinction learning problem. This approach allows debiasing from the start of the learning process, which is clear in the results from the less druggable dataset. The presence or absence of hydrogens can change the overall modeling attempt in ways that are not always obvious. This is because the effects of NanoShaper are overimposed onto the IVDD learning model.

The proposal to mitigate and reduce various biases, even at the cost of lower accuracy, is indebted to the fair ML field ([214]). While fairness concepts are usually applied to social aspects (e.g., demographic parity), here this way of thinking is introduced focusing on only certain label information. Together with explicit structural biases, technical aspects also have an important role. Here, several different values for the small and large NanoShaper probes are tested to identify the pockets. The small probe was easy because there was no reason not to choose the water-molecule-like size of 1.4 Å. For the large probe, there is no immediate physically driven quantity, with the convex hull being the extreme solution. A value of 3.5 Å performed better than 3 Å in detecting relatively shallow pockets together with the more prototypical buried ones. Larger values generally led to poorer results in terms of shape, with a systematic decrease in Jaccard index values.

In terms of future developments, several improvements can be envisioned for this methodology. A volume segmentation ad hoc algorithm could improve the accuracy, particularly when selecting the value of the large probe. Such a tool could provide more freedom of choice for this parameter. The work of ([203]), among others, has shown that many software for pocket identification tend to identify large pockets without segmentation techniques. Segmentation could be used to find subpockets that are better suited to virtual screening and docking. Another development would be a webserver to access the tool easily. Finally, the method can be combined with the Pocketron method ([215]) to not only track the pocket volume and residues over time, but also to provide a dynamic druggability score that explicitly considers the probability of the conformation ultimately delivering a Boltzmann weighted estimator.

4.3 Deep Import Vector Domain Description

Section 2.2 described some of the most significant and recently proposed methods developed in the one-class learning domain. In particular, it highlighted the crucial role that ANNs have had in the improvement of the performances in this domain. This improvement is because ANNs allow learning a hidden representation where it is generally easier to close off the normal samples. The most interesting state-of-the-art methods are OCGAN (175), TQM (216), ICS (176), Deep SVDD (179), Deep Vae SVDD (181), HRN (178), P-KDGAN (174) and the recently proposed contrastive distribution augmentation method (113) (discussed in Section 2.3.3 and hereafter mentioned as ContrDA). Typically one-class learning methods are compared in terms of performances through well-known datasets (e.g., MINIST[1] (171), f-MNIST[2] (217), and CIFAR-10[3] (218)) (see Figure 4.14). Even if these datasets have been devised to address classification problems, they can be used for one-class experiments by considering one class at a time for training and all the test set for testing. This way, different experiments can be run for each dataset, one for each class (MNIST, f-MNIST, and CIFAR-10 contain ten classes each). To evaluate the model's generalization ability, each experiment is usually repeated more than once, with different seeds, and results are averaged accordingly. The results are, in general, quantitatively evaluated through the AUC by using the ground truth labels in testing.

Figure 4.14: *Typical dataset for one-class learning experiments.*

[1]http://yann.lecun.com/exdb/mnist/
[2]https://github.com/zalandoresearch/fashion-mnist
[3]https://www.cs.toronto.edu/ kriz/cifar.html

While for some simple datasets, i.e. MNIST and f-MNIST, all of the methods mentioned above are capable of achieving good and similar performances, this only sometimes happens for some complex and variegated datasets, i.e. CIFAR-10. This is clearly visible in Table 4.3 (for compactness, class names are substituted by digits: 0 - Airplane, 1 - Autobomile, 2 - Bird, 3 - Cat, 4 - Deer, 5 - Dog, 6 - Frog, 7 - Horse, 8 - Ship, 9 - Truck) where state-of-the-art methods are compared with the performances obtained by the OC-SVM method (219) on the input space. In particular, Table 4.3 shows that the usage of complex and deep strategies does not necessarily correspond to a performance improvement (e.g., for Deep-SVDD, Deep-VAE-SVDD and OCGAN). The method that has made one-class learning strategies extremely powerful also in complex tasks is ContrDA (113). As mentioned in Section 2.3.3, this method adopted an unsupervised contrastive approach for learning a good representation of the normal samples. Then, a one-class learning method is used for making predictions, i.e. OC-SVM as the network's *last layer*. The ContrDA method proves that hidden representations have a crucial role in the effectiveness of one-class learning methods.

Authors of ContrDA adapted the representation learning strategy to one-class learning, yet they did not propose any new *last layer* but took advantage of existing ones (e.g., OC-SVM). In this Section, a solution, which combines the ContrDA approach with the IVDD method, is proposed and dubbed Deep-IVDD. This strategy combines the pros of having good representations with the pros of obtaining a probability estimation for each sample, which is preferable in some application areas, like the life sciences (176).

Table 4.3: *AUCs scores per-class and averaged on CIFAR-10.*

CIFAR-10	0	1	2	3	4	5	6	7	8	9	Mean
OC-SVM (56)	61.6	63.8	50.0	55.9	66.0	62.4	74.7	62.6	74.9	75.9	64.78
OCGAN (75)	75.7	53.1	64.0	62.0	72.3	62.0	72.3	57.5	82.0	55.4	65.66
TQM (216)	40.7	53.1	41.7	58.2	39.2	62.6	55.1	63.1	48.6	58.7	52.10
ICS (76)	76.8	71.3	63.0	60.1	74.9	66.0	71.6	64.1	78.9	66.0	69.27
Deep-SVDD (79)	61.7	65.9	50.8	59.1	60.9	65.7	67.7	67.3	75.9	73.1	64.81
Deep-VAE-SVDD (81)	64.4	65.3	57.5	60.3	61.6	64.3	66.3	64.0	76.5	74.8	65.5
HRN (78)	77.3	69.9	60.6	64.4	71.5	67.4	77.4	64.9	82.5	77.3	71.32
P-KDGAN (74)	82.5	74.4	70.3	60.5	76.5	65.2	79.7	72.3	82.7	73.5	73.76
ContrDA (113)	89.7	97.3	86.9	82.5	84.4	88.9	90.1	91.3	76.6	93.5	84.6

In a further development, a revised version of the IVDD method is introduced, dubbed IVDD-KL, that approximates the IVDD method (36) through the Nyström method (37) rendering it scalable and easily interpretable as a last layer of a neural architecture. The method aims to combine the advantages of DL with the ones of the kernel-based approaches (radial basis functions locality control), obtaining a scalable and non-degenerate one-class method. As the original IVDD method, the proposed version provides the probability estimation for each sample to be normal and, additionally to (37), it has faster training time and an improved probability model.

Method

Deep-IVDD combines the ContrDA approach presented in (113) with the IVDD method (36). In particular, a two-step framework is proposed as in (113). In the first step, SSL strategies are adopted for learning the hidden representation. In the second step, the IVDD method is trained on the learned representations. A schematic view of the Deep-IVDD method is depicted in Figure 4.15.

IVDD-KL is a modified version of the IVDD method (with the Nyström optimization presented in (37) and described in Section 4.1). In particular, the proposed objective function is:

$$\min_{\Gamma,a} D_{KL}(q||\hat{p}) - \hat{C}\sum_{i=1}^{n}\log(\hat{p}_i). \tag{4.13}$$

Here, $\hat{p}_i$ is a modification of the original probability definition defined in Equation 4.2; it is still a logistic model but the decision function is now scaled with respect to the distance of a sample from the center:

$$\hat{f}_i = \frac{||\phi(h(\mathbf{x}_i;\mathcal{W})) - \sum_{k=1}^{n_l}\alpha_k\phi(\mathbf{x}_k)||^2 - \Gamma}{||\phi(h(\mathbf{x}_i;\mathcal{W})) - \sum_{k=1}^{n_l}\alpha_k\phi(\mathbf{x}_k))||} = \frac{\boldsymbol{\alpha}^t\hat{\mathbf{K}}_r\boldsymbol{\alpha} - 2\hat{\mathbf{K}}_{i,(\cdot)}\boldsymbol{\alpha} + k_{ii} - \Gamma}{\sqrt{\boldsymbol{\alpha}^t\hat{\mathbf{K}}_r\boldsymbol{\alpha} - 2\hat{\mathbf{K}}_{i,(\cdot)}\boldsymbol{\alpha} + k_{ii}}}. \tag{4.14}$$

This modification grants that at the center of the hypersphere, the probability always reaches the maximum value of 1 as $\hat{f}_i = -\infty$ (assuming a not null Γ value). In the original IVDD formulation, the value of the probability at the center depended on

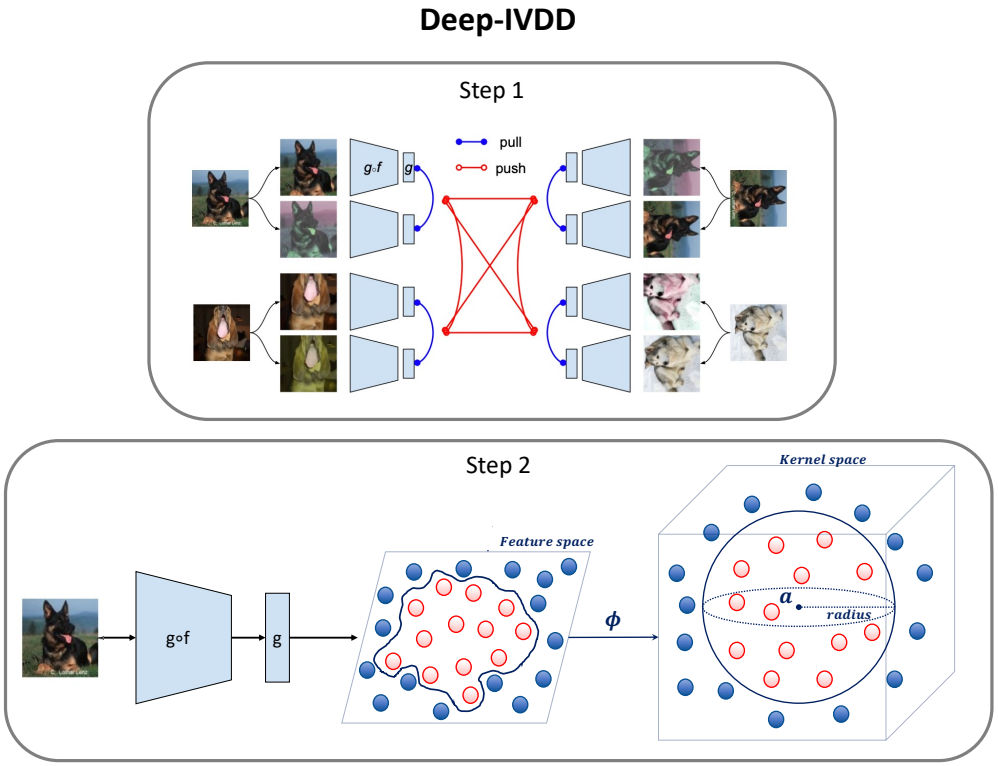

Figure 4.15: *Deep-IVDD: a schematic representation of the method.*

the specific Γ value. Here this dependence is no more present, and the probability ranges from 1 at the center to 0 for an infinitely distant sample from the center. In addition, instead of controlling the radius directly, as in the SVDD approach, the adherence of the probability distribution $\hat{p}$ to a reference probability distribution q is controlled via the KL divergence. This still allows controlling the radius, albeit indirectly. In this way, IVDD-KL endows a probability model as IVDD. Additionally the regularization takes place through a probability control mechanism.

Instead of using the optimization algorithm described in (36), this cost function is optimized via the Batch Stochastic Gradient Descent (SGD) and its variants (e.g., Adam (92)). Training by SGD, with a proper initialization and a sufficiently big C ($C = 1$ proves always to be sufficient), the size of the hypersphere is monotonically increasing with respect to the epochs. A range $[\pi_{low}, \pi_{high}]$ can be fixed a priori, representing the percentage of accepted inner samples, and the training epochs can be repeated until this range is hit for the first time. If the range is skipped as the hypersphere grows too fast, one can step back to the previous epoch parameters and

Algorithm 1 *Deep-IVDD-KL: optimization algorithm*

Require: $\mathbf{X}, \sigma, \eta$
Ensure: $\boldsymbol{\alpha}, \mathbf{W}, \Gamma$

$\eta = \eta_0, \boldsymbol{\alpha} = 1/n_l, \pi = 0.0$
while $\pi \notin [\pi_{low}, \pi_{high}]$ **do** ▷ until π is not in range
  $\pi, \hat{\boldsymbol{\alpha}}, \hat{\Gamma} = takestep(\mathbf{X}, \mathbf{W}, \eta, \Gamma, \boldsymbol{\alpha})$ ▷ do an epoch
  if $\pi > \pi_{high}$ **then** ▷ if π is too high
    $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}$ ▷ step back to the previous $\boldsymbol{\alpha}$
    $\hat{\Gamma} = \Gamma$ ▷ step back to the previous Γ
    $\eta = \eta/10.0$ ▷ reduce the learning rate
  end if
  $\boldsymbol{\alpha} = \hat{\boldsymbol{\alpha}}$
  $\Gamma = \hat{\Gamma}$
end while

decrease the learning rate. This procedure is similar to what was done for IVDD, where the prescribed range was reached by bisection on C (37). The IVDD-KL training procedure hence overcomes the IVDD sensitivity problem to the C value, as it is never changed and takes advantage of the epochs to reach the desired range; this strategy proved to be very fast and always hits the desired range. Algorithm 1 shows the pseudo-code of the proposed approach, where η indicates the learning rate, *takestep* is a gradient update step, and π is the fraction of inner samples. When this variant is combined with an a-priori learned representation, the method is dubbed Deep-IVDD-KL.

Results

Here, some experiments are presented, aiming at comparing the performances of the aforementioned state-of-the-art methods with the Deep-IVDD and Deep-IVDD-KL presented above. The code is available in the GitHub repository of the project[4]. For Deep-IVDD, and Deep-IVDD-KL the hidden representation was firstly learned and then their respective cost functions were optimized on the mapped samples. For the first activity, the code available in (220) was used without modifying any parameter. The SSL network was trained on a HPC cluster equipped with a nVidia Tesla V100 32Gb GPU and 2 Intel(R) Xeon(R) Gold 6240 @ 2.60GHz Cascade Lake CPUs.

[4]https://github.com/erikagardini/DeepIVDD

In particular, the IVDD objective function (see Equations 4.1, 4.2, 4.3, and 4.4) was optimized with $\beta = 25$ and $\hat{C}$ automatically updated and tuned for including inside the hypersphere $[80\% - 90\%]$ of the training samples. The parameters α_k were initialized at $1/n$ where n is the number of training samples. All the samples were normalized using ℓ_2 normalization, the kernel used was the RBF, and γ was set to $\gamma = \frac{10.0}{var(\mathbf{X})*d}$ where $\mathbf{X}$ is the matrix containing all the input samples, $var(\mathbf{X})$ is the variance of all the values in the input matrix $\mathbf{X}$, and d is the feature space dimension. This approach is equivalent to the one proposed in (113). For IVDD-KL, instead, the number of landmarks (see Equation 4.14) was set to $n_l =\sim 10\%$ of the training samples, following a *less is more* approach (60) that allows to reduce memory/time requirements while preserving good generalization performances. The landmarks were initialized through the k-means algorithm with $k = n_l$. This is more computationally expensive than a random selection, but it should help the method to preside over the space more correctly (see Figure 4.16).

The range $[\pi_{low}, \pi_{high}]$ was set to $[80\%, 90\%]$; this means that the training was stopped when the hypersphere was large enough to include between 80% and 90% of the training samples inside. The prior q in Equation 4.13 was a Laplace distribution $q = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$. This choice is motivated by the prior expectation that at the end of the training, one could expect that the majority of normal samples have high probability values, hence the norm-one induced sharp peak. This distribution may be preferable in principle to a Gaussian distribution, for instance, as one can

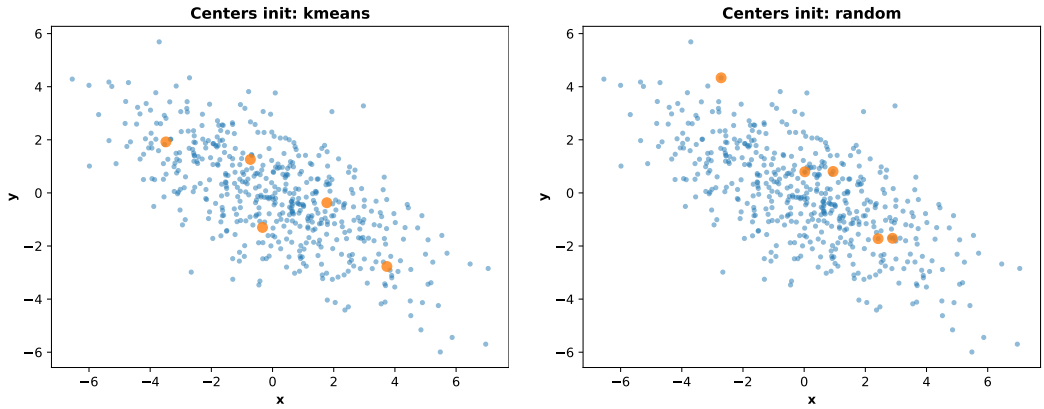

Figure 4.16: *Deep-IVDD-KL: landmark initialization.*

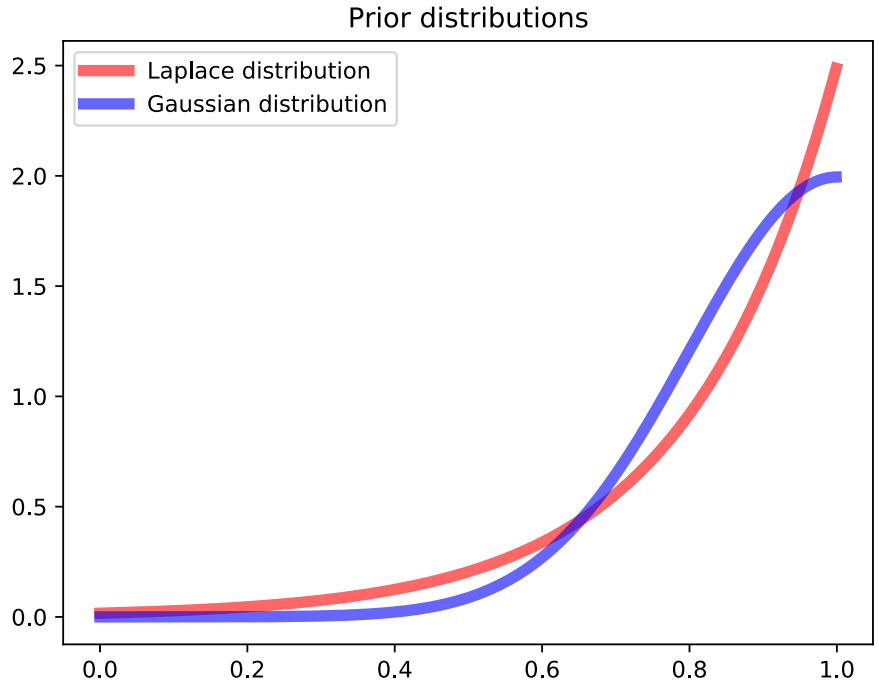

Figure 4.17: *Deep-IVDD-KL: Laplace distribution vs. Gaussian distribution.*

expect that the confidence reduction with respect to the distance from the center is a monotone function, and there is no reason to have a convexity change in this probability shift process (see Figure 4.17). The parameters of the distribution were set to: $\mu = 1$ and $b = 0.2$. Finally, the value of $\hat{C}$ in Equation 4.13 was set at the value 1 and never changed, while the batch size was set to 32, and the value of beta was kept to $\beta = 25$ (see Equation 4.2), as proposed in (36; 37). Crucial is the value of γ of the RBF kernel. While in (113) the value was set to $\gamma = \frac{10.0}{var(\mathbf{X})*d}$, which is effective also for Deep-IVDD, for Deep-IVDD-KL the choice of this parameter is not trivial as it also depends on the number of landmarks used during the training and may cause a degradation of the performances when it is not properly selected. For all the experiments, the adopted γ was $\gamma = \frac{1.0}{var(\mathbf{X}_{nl})*d}$, where $\mathbf{X}_{nl}$ are the landmarks matrix, $var(\mathbf{X}_{nl})$ is the variance of all the values in the landmarks matrix $\mathbf{X}_{nl}$ and d is the feature space dimension. This formula was effective in most of the experiments; however, further experiments are underway for improving and automatizing the choice of γ.

Table 4.4: *Comparison of different one-class classification methods.* Average AUCs scores in % for each experiment and learning the hidden representations as described in (113).

CIFAR-10	0	1	2	3	4	5	6	7	8	9	Mean
ContrDA+OC-SVM	89.7	97.3	86.9	82.5	84.4	88.9	90.1	95.6	86.0	90.6	89.2
Deep-IVDD	89.8	97.3	86.8	82.6	84.3	88.9	90.1	95.6	85.8	90.6	89.2
Deep-IVDD-KL	80.8	95.1	73.5	80.5	79.2	87.9	91.1	91.3	76.6	93.5	84.6

Considering the results in Table 4.4, the Deep-IVDD method can rival the performances obtained with the ContrDa+OC-SVM method, with the additional advantage of being probabilistic. On the other hand, the performances obtained with Deep-IVDD-KL are worse than the others. However, there may be more suitable metrics for measuring the performances of a one-class learning method than the AUC score. In fact, AUC only considers the scores distribution of the test set (see Figure 4.18). In particular, in the one-class learning domain, a high AUC score means that the normal samples (true positives) obtain higher scores than the ones assigned to the anomalies (true negatives). When the AUC is computed on scores in a known domain, i.e. probabilities, and the thresholds are set a priori to comprise the minimum and the maximum values, the metric can also be interpreted in absolute terms. On the contrary, when the AUC is computed on scores that are not in a known bounded

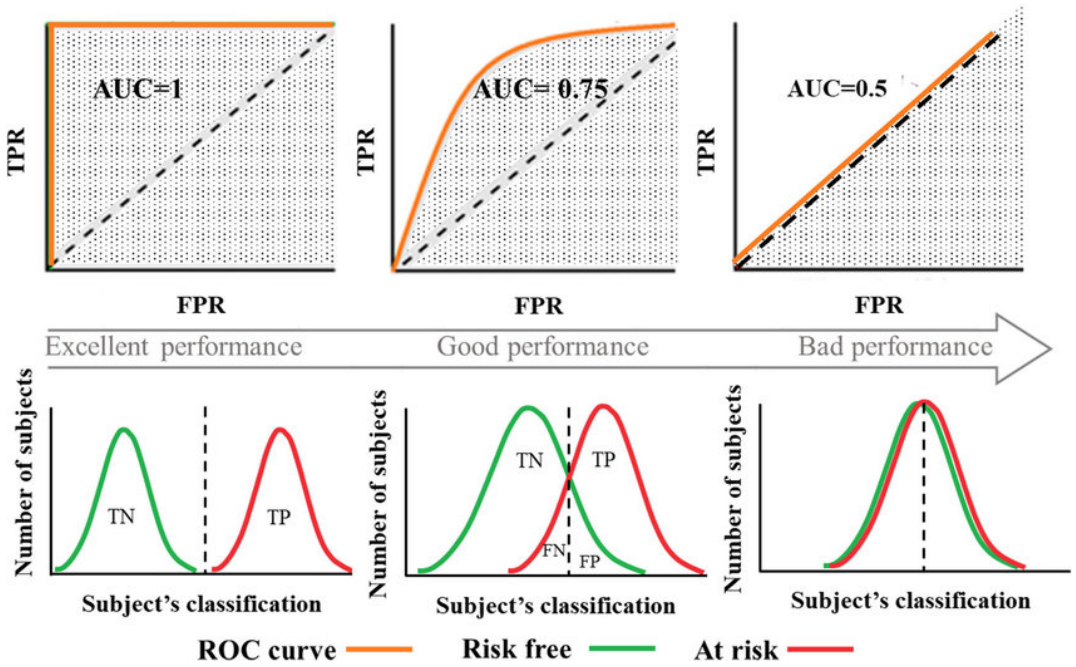

Figure 4.18: *AUC metric visually explained (221).*

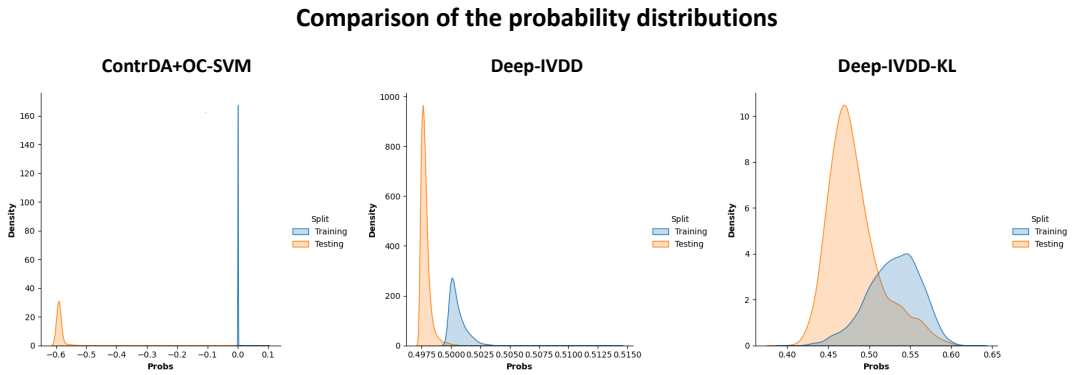

Figure 4.19: *Distributions of the scores (CIFAR dataset, class 1).*

domain, then it is just a relative metric, and it is not capable of measuring the goodness of the testing scores with respect to the training scores and a prescribed threshold. This aspect might lead to a wrong interpretation of the AUC results.

To further investigate this aspect, the distributions of the scores have been analyzed (see Figure 4.19). The picture shows that the OC-SVM method's testing scores are much lower than the training scores. In this case, all the testing samples are identified as anomalies (the distance from the separating hyper-plane is negative). Despite this, the AUC score is high. This is because the normal samples in the test set (true positives) are getting higher scores than the real anomalies (true negatives). In fact, in the OC-SVM situation, the domain of the scores is not known a priori; therefore, the AUC is just measuring the goodness of the testing scores, that is, the capability to discriminate among different testing samples, but it cannot be used as a metric for evaluating the classification performances in absence of a threshold coming from the training set.

At this stage, a word of wisdom is needed; indeed, one should distinguish between one-class learning and one-class classification. In one-class learning, given a testset, one is interested in getting the right ranking of the samples with respect to a central concept. AUC measures well this aspect as only the relative ranking counts. In the one-class classification case, one needs an explicit threshold that should be devised at training time. Hence, methods that do not deliver a threshold should be endowed with a threshold-delivering method and the typical classification metrics should be used to assess the goodness of the results.

Broadly used scores are the F1-score, the balanced accuracy (BA), the precision, and the recall metrics have been considered. They can be computed as:

$$precision = \frac{tp}{tp + fp} \tag{4.15}$$

$$recall = \frac{tp}{tp + fn} \tag{4.16}$$

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.17}$$

$$BA = \frac{tpr + tnr}{2} \tag{4.18}$$

$$tpr = \frac{tp}{tp + fn} \tag{4.19}$$

$$tnr = \frac{tn}{tn + fp}, \tag{4.20}$$

where tp means true positives (normal samples predicted as normal), fp means false positives (anomalous samples predicted as normal), fn means false negatives (normal samples predicted as anomalous), and tn means true negatives (anomalous samples predicted as anomalous).

While for Deep-IVDD and Deep-IVDD-KL the classification threshold was naturally set to 0.5 ($\hat{p}_i < 0.5$ are anomalous), for the ContrDA+OC-SVM (as for OC-SVM) the threshold was set to 0.0 (samples with a negative distance from the hyper-plane are anomalous). The results of this additional analysis are summarized in Table 4.5. The missing values mean that the denominator is zero, and therefore the value cannot be computed. In particular, here, it is numerically and quantitatively confirmed what was evinced by the visual analysis of the probability distributions. In all these experiments, the performances obtained with Deep-IVDD-KL are higher. The precision and recall values show that the Deep-IVDD-KL method is less conservative and includes some outliers inside the hypersphere (low precision values); at the same time it can correctly identify most of the correct samples and place them inside the hypersphere (high recall values). On the other hand, the ContrDA+OC-SVM solution is completely overfitted and it cannot distinguish the difference between normal and anomalous testing samples. These aspects are further confirmed by the F1 and the BA scores, which show, once again, that the Deep-IVDD-KL method can

Table 4.5: *Classification performances.* Metrics obtained by the OC-SVM, IVDD and IVDD-KL methods classifying the testing samples of the CIFAR-10 dataset.

F1-Score	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**
OC-SVM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
IVDD	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
IVDD-KL	0.2	0.6	0.2	0.2	0.2	0.3	0.4	0.4	0.2	0.4	0.3
Precision	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**
OC-SVM	1.0	1.0	-	-	1.0	-	-	-	-	1.0	1.0
IVDD	0.97	1.0	0.9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99
IVDD-KL	0.12	0.40	0.11	0.13	0.13	0.19	0.24	0.28	0.12	0.22	0.20
Recall	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**
OC-SVM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
IVDD	0.04	0.04	0.01	0.02	0.02	0.02	0.02	0.04	0.01	0.02	0.02
IVDD-KL	0.97	0.92	0.96	0.94	0.94	0.94	0.92	0.89	0.97	0.97	0.94
BA	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**
OC-SVM	50.2	50.8	50.0	50.0	50.0	50.0	50.0	50.0	0.5	0.5	50.1
IVDD	52.2	51.9	50.4	51.1	50.8	50.9	51.0	52.1	50.3	51.2	51.2
IVDD-KL	60.2	88.5	53.7	63.5	63.2	74.7	80.2	81.9	58.4	79.1	70.3

obtain a better performance in terms of classification. These observations confirm the Deep-IVDD-KL method's potential but also highlight that there is still room for improvement in most cases and that the AUC metric could be misleading in the one-class classification realm (which, we repeat, it is a different concept than pure one-class learning).

Another aspect to consider is that for Deep-IVDD-KL, as well as for Deep-IVDD,

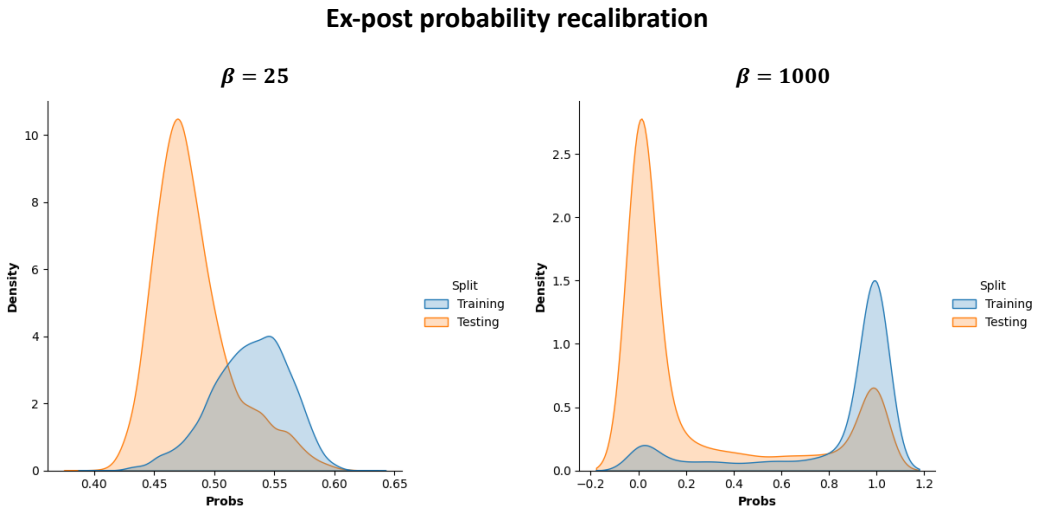

Figure 4.20: *Deep-IVDD-KL: ex-post recalibration (CIFAR dataset, class 1).*

Table 4.6: *Deep-IVDD-KL: classification performances with different thresholds.* The results are the average of metrics obtained for each class of the CIFAR-10 dataset.

Threshold	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
F1-Score	0.31	0.33	0.36	0.39	0.44
Precision	0.20	0.21	0.23	0.26	0.32
Recall	0.94	0.92	0.90	0.86	0.78
BA	70.3	72.1	74.0	75.6	76.9

it is possible to ex-post recalibrate the distributions given the decision function by simply varying the value of β. This does not change the results but improves the distributions and scores' interpretability (see Figure 4.20). When $\beta = 1000$ a better and more spread probability distribution can be obtained for all the CIFAR-10 classes.

Finally, in a one-class classification problem, sometimes a more conservative approach might be preferable, and the main attention can be given to the samples obtaining a higher probability score, as in (176). For this reason, Table 4.6 reports the metrics obtained by recalibrating the probabilities (with $\beta = 1000$) and varying the classification thresholds (from 0.5 to 0.9 with a step of 0.1). The results show that some anomalous samples are included inside the sphere (obtaining a probability score > 0.5), but their probability score is lower than those assigned to the normal samples. In fact, increasing the value of the threshold corresponds to an improvement in terms of BA.

For completeness, Appendix B shows the top-64 samples, which are the samples with the highest probability score, for each class. These images confirm what was quantitatively shown by Table 4.5: for some categories (e.g., car, frog, horse), the method can correctly distinguish normal and anomalous samples, while for other classes, there is still room for improvement.

Ablation study

Next, additional experiments to illustrate the impact of the different components of the pipeline. In particular, different strategies for learning hidden representations have

Table 4.7: *Details of the VAE encoder architecture.*

Architecture
$32 \times (5 \times 5 \times 3)$-filters + Batch Normalization + Leaky ReLU
$64 \times (5 \times 5 \times 3)$-filters + Batch Normalization + Leaky ReLU
$128 \times (5 \times 5 \times 3)$-filters + Batch Normalization + Leaky ReLU
Dense layer of 128 units

been adopted (e.g., VAEs) to explore how much the hidden representation impacts the final performance. Additionally, further experiments have been performed to understand better the importance of the ℓ_2 normalization. To this aim, the OC-SVM and the IVDD methods have also been run using representations from different models and on data that are not ℓ_2 normalized.

In particular, a VAE has been used for learning an alternative hidden representation. Its structure is equivalent to the one presented for the Deep-VAE-SVDD method (81), which is an adaptation of the network used in the Deep-SVDD method (79). The encoder structure is summarized in Table 4.7, while the decoder mirrors the architecture of the encoder. The parameters have been set following the Deep-SVDD and Deep-VAE-SVDD experiments, and in particular, $\alpha = 0.1$ for the Leaky ReLU activation function, learning rate $lr = 0.001$ with Adam optimization (92), number of epochs $n_{epochs} = 150$, and batch size $bs = 200$.

In addition, another approach has been considered as a representation learning method. The architecture was inspired by the work presented in (110), dubbed SimSiam, and described in Section 4.8. In particular, the architecture is a Siamese network composed of two different blocks: an encoder, which comprises a backbone and a projection MLP, and a prediction MLP. As backbone a CIFAR variant of the ResNet-18 architecture has been adopted (90), while the details of the projection MLP and the prediction MLP are summarized in Table 4.8. The architecture is optimized with the following parameters: number of epochs $n_{epochs} = 100$, learning rate $lr = 0.1$ and a batch size $bs = 256$. Further details on the implementation can be found in (222). Here, instead of using a multi-class learning approach, just one class at a time has been considered, as in a standard one-class learning experiment.

Table 4.8: *Details of the SimSiam architecture.*

Backbone	**Projection MLP**	**Prediction MLP**
CIFAR version of ResNet-18	Dense layer of 128 units Batch Normalization ReLU Dense layer of 128 units Batch Normalization	Dense layer of 32 units Batch Normalization ReLU Dense layer of 128 units

Table 4.9 shows the results of all these experiments. In particular, it clearly shows that learning a good representation is key to solving a one-class learning problem. In fact, the performances achieved with the ContrDA approach are, in most cases, higher than those obtained with the VAE and the SimSiam architectures. However, more than high-quality hidden representations are required. In fact, the a-posteriori ℓ_2 normalization of the samples has a high contribution to the final performance of the ContrDA method. This is not the first time ℓ_2 normalization has had a significant impact; in fact, it has been used by the recent HRN method (78) (as a normalization technique for the input) and allowed a significant improvement of the final performances.

Table 4.9: *Experiments with different hidden representations and with/without a-posteriori ℓ_2 normalization.* Average AUCs scores in % of the OC-SVM and the IVDD methods.

OC-SVM	**l2**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**
ContrDA	✓	89.7	97.3	86.9	82.5	84.4	88.9	90.1	95.6	86.0	90.6	89.2
ContrDA	✗	64.2	81.4	53.6	50.7	67.9	61.8	76.8	75.4	61.9	74.5	66.8
VAE	✓	67.6	62.5	62.1	51.8	63.9	53.1	66.2	61.8	61.1	57.7	60.8
VAE	✗	68.4	41.9	66.2	49.5	73.5	49.4	70.6	52.2	67.6	43.0	58.2
SimSiam	✓	59.2	60.1	53.1	60.5	56.1	62.2	64.3	63.2	72.5	57.7	60.9
SimSiam	✗	62.6	56.3	48.8	53.5	59.1	62.9	58.8	62.0	67.9	58.9	59.1
IVDD	**l2**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Mean**
ContrDA	✓	89.8	97.3	86.8	82.6	84.3	88.9	90.1	95.6	85.8	90.6	89.2
ContrDA	✗	65.7	80.5	54.2	51.0	68.1	62.0	76.8	75.4	61.9	75.0	67.1
VAE	✓	67.6	62.4	62.1	51.8	63.9	53.1	66.2	61.8	61.1	57.7	60.8
VAE	✗	68.3	41.8	66.2	49.5	73.5	49.4	70.5	52.1	67.6	43.0	58.2
SimSiam	✓	58.4	60.6	52.8	60.4	56.1	62.3	63.8	63.0	72.2	58.3	60.8
SimSiam	✗	62.7	56.7	49.4	53.5	59.3	63.0	59.1	61.9	67.8	59.2	59.3

Reproducibility

The implementation was written in Python, and all the experiments can be reproduced using the code available in the GitHub repository:

- https://github.com/erikagardini/DeepIVDD.

The representations were learned on a HPC cluster equipped with a nVidia Tesla V100 32Gb GPU and 2 Intel(R) Xeon(R) Gold 6240 @ 2.60GHz Cascade Lake CPUs. All the other experiments were run on a laptop equipped with a 2,6 GHz Intel Core i7 and 16 GB RAM.

Final remarks

This Section presented the Deep-IVDD method. It combines deep representations with the probabilistic one-class learning method IVDD. In addition, the Deep-IVDD-KL method is proposed, which is a scalable version of the Deep-IVDD method based on the SGD optimization and the Nyström approximation and endows an enhanced probability model.

The main advantage of both the Deep-IVDD and the Deep-IVDD-KL methods is that they combine deep approaches for learning a good hidden representation with a probabilistic one-class classification method. In this way, not only the classification is easier to perform, with a classification threshold naturally set as 0.5, but also a more interpretable result (with a known domain) is obtained. This is preferable in some applications, especially for experiments in life science (176).

Challenging for this type of methods, and more in general for unsupervised learning techniques, is learning the intrinsic structure of the data and identifying the relevant features. In particular, the lack of samples belonging to the anomalous class, makes the training and the identification of the decision boundary hard, as only the data belonging to one class delivers information. Here, different strategies for learning hidden representations have been adopted to distinguish the normal samples more easily from the anomalous samples. A comparison of these strategies is presented, together with the effect of the a-posteriori ℓ_2 normalization. The analysis results show that the ContrDA approach presented in (113), combined with the a-posteriori

ℓ_2 normalization, allows to obtain extremely powerful representations, which help the subsequent one-class learning method. However, good AUC performance does not necessarily correspond to good classification metrics. In fact, the AUC computed on general scores is only a relative measure, and allows to compare the goodness of a testing sample with respect to the others. These aspects have been widely discussed, and some additional metrics for evaluating the classification performances have been proposed. The eventual outcome of this analysis is that the Deep-IVDD-KL method can obtain better classification performances than those of the Deep-IVDD and the OC-SVM methods.

In the future, it would be interesting to investigate in more detail the usage of other deep architectures for learning hidden representations, knowing that SSL techniques are a good starting point. In addition, the IVDD-KL solution can be improved by reducing the number of parameters or simplifying their selection. One possible option could be to adopt an automatized method for selecting γ (e.g., the trace-criterion method ([223])) or to improve the selection of the landmarks by using other clustering techniques. In fact, the k-means algorithm adopted in this work may lead to select landmarks in sparsely populated areas of the space, which may correspond to a degradation of the performances of the proposed method.

Chapter 5

Conclusions

This Thesis has been focused on unsupervised learning methods and their applications to the life science domain. At methodological stage, two new unsupervised ML methods have been presented, aiming to improve existing ones in the one-class and manifold learning realms. The first one, dubbed Ab Initio Local PP (118), is a revised version of the PP algorithm (35) and solves some of the drawbacks and limitations of the original method. The second one, dubbed IVDD-KL, is an enhanced version of the IVDD method (36), which hybridizes kernel methods to DL approaches to obtain a scalable solution together with an improved probabilistic model. According to the results presented in Sections 3.3 and 4.3, both methods improved the performances achieved by their previous versions. For the newly introduced IVDD version, state-of-the-art performances have been reached within a coherent probabilistic model, and some systematic deficiencies in the literature in the evaluation of one-class learning methods have been found, particularly in the role of the AUC metric.

At applicative stage, particular attention has been devoted to the life science/health domain, where it is widely known there is still ample space for the deployment of AI. The first application consisted of a pipeline, based on the PP method, for the analysis of RNA-Seq datasets, both transcriptomic and single-cell, and aimed at identifying genes that may be involved in biological processes (e.g., the transition of tissues from normal to cancer (117)). In this project, an R package has been released and published on CRAN to make the pipeline accessible to the bioinformatic community through high-level APIs.

The second application was about the drug discovery domain. It has been devoted to the development of a pipeline for identifying druggable pockets, namely regions of a protein with a high probability of accepting a small molecule (176) (a drug). Both these pipelines achieved remarkable results (see Sections 3.2.1 and 4.2). Lastly, a detour application has been developed in Section 3.2.2 (116). It involved the PP algorithm and aimed to highlight its strengths and identify its limitations by analyzing the complex CNN-induced vector space. This application has been conducted in music and visual arts domains. The aim was to automatically recover the historical evolution of styles just by looking at the samples. Hence, here, the PP has been evaluated at a cognitive point of view and used to indirectly verify the goodness of the manifolds learned by the employed CNN.

Regarding the PP algorithm, its application was based on euclidean distances so far. In the future, it would be interesting to investigate its application in more complex spaces, like the ones induced by optimal transport (224). In this case, the Wasserstein distance may be adopted (225). Alternatively, spaces induced by deep architectures may be considered. An analysis of the PP applied to CNN-induced spaces has been proposed, perhaps more suitable architectures (e.g., VAEs or self-supervised architectures) may be considered to improve hidden spaces. One more interesting aspect that may deserve deepening is the implicit generative nature of the PP method. In fact, even though a probability density function is not available, the waypoints obtained throughout the algorithm are newly generated samples missing in the initial dataset. In the future, it could be interesting to analyze more deeply those generated samples, especially when they are computed in an embedding space (e.g., VAE). In terms of computational efficiency, the Ab Initio Local PP presented in this Thesis reduces the optimization cost because it is based on a very restricted set of samples. However, the initialization step requires one to compute a penalized distance matrix among all the samples in the dataset. This is not scalable when the input matrix is high-dimensional and comprises several samples. In the future, techniques like the Nyström approximation can be used to improve the efficiency of this step. Finally, the most intriguing future work connected to the PP could be to investigate its behavior from the point of view of human cognition. The question is

whether minimum free energy paths, which are the inspiring concept behind PPs, are how humans connect ideas. In doing this, one implicitly assumes that start and end points are ideal objects, literally ideas. One could hence wonder if the way humans think can be formalized as an attempt to move from an idea to another via maximal probability moves (namely minimum free energy regions or regions which bear many ideas, as the probability is linked to the number of points/ideas in the space). The conjecture is that maximal probability paths (resulting from the PP algorithm) can describe many phenomena, including ideas morphing (which is ultimately a creative cognitive process). It would be interesting to understand how general this principle is and what it can retrieve when applied to DL-induced hidden spaces.

Regarding the IVDD-KL method, in the future it could be interesting to investigate other solutions that are less parameters dependent. Alternatively, automatic or improved strategies could be adopted for their selection. Examples of these strategies are the trace-criterion method for the selection of *gamma* (223) and other clustering techniques for the landmarks selection (e.g., Kernel Density Estimation (226)). In addition, the choice of the prior distribution could be further investigated. In this work, the idea of using a probability distribution as a regularization term was inspired by the VAEs approach, and the Laplace distribution was just a possible, albeit justified, choice. In the future, other distributions and regularization terms may be further investigated. From a technological standpoint, being the IVDD-KL method scalable, it could be adopted on low-cost devices (e.g., mobile phones and embedded devices). In this case, knowledge distillation techniques may be adopted to make this application even more feasible in terms of computational complexity and storage requirements (227).

Regarding applications, particularly the life sciences realm, in order to extend the usage of the proposed methods to clinical scenarios, one ought to consider other important aspects, which overall can be ascribed to the so-called trustworthy AI (228; 229). These aspects include the safety, privacy, security, fairness, and robustness of the ML methodologies and also concern data access, usage, and co-modification. For instance, for the privacy concern federated learning (230), Differential Privacy Stochastic Gradient Descent (DPSGD) (231), and Secure Multiparty Computation

(232) are possible solutions. Other crucial aspects concern the explainability, causality, and accountability of the ML techniques used. In particular, it is extremely important to understand how a prediction of a DL method can be explained (233; 234; 235). This aspect is not trivial, as ML methods are typically complex and black-box (236). In the future, additional experiments can be performed to interpret the proposed methods' internal mechanisms and explain the reasons behind their decision/predictions.
Lastly, the methods developed in this Thesis can find many other applications in domains that are possibly very distant from what has been discussed here. For example, the Ab Initio Local PP could work as a playlist creator in the music context. In fact, it could provide a list of songs that gradually morphs from one initial song to another, working as a recommendation system (237). Such kinds of projects could be further investigated in the future along the lines of other already published applications (119; 238).

Bibliography

[1] Turing AM. Computing machinery and intelligence. In: Parsing the Turing Test. Springer; 2009. p. 23-65.

[2] Mitchell TM, Mitchell TM. Machine learning. vol. 1. McGraw-hill New York; 1997.

[3] Witten IH, Frank E, Hall MA, Pal CJ, DATA M. Practical machine learning tools and techniques. In: Data Mining. vol. 2; 2005. .

[4] James G, Witten D, Hastie T, Tibshirani R. An introduction to statistical learning. vol. 112. Springer; 2013.

[5] Bishop CM, Nasrabadi NM. Pattern recognition and machine learning. vol. 4. Springer; 2006.

[6] Deisenroth MP, Faisal AA, Ong CS. Mathematics for machine learning. Cambridge University Press; 2020.

[7] Chowdhary K. Natural language processing. Fundamentals of artificial intelligence. 2020:603-49.

[8] Lin L, Liu J, Zhang X, Liang X. Automatic translation of spoken English based on improved machine learning algorithm. Journal of Intelligent & Fuzzy Systems. 2021;40(2):2385-95.

[9] Libbrecht MW, Noble WS. Machine learning applications in genetics and genomics. Nature Reviews Genetics. 2015;16(6):321-32.

[10] Kim D, Kim SH, Kim T, Kang BB, Lee M, Park W, et al. Review of machine learning methods in soft robotics. Plos one. 2021;16(2):e0246102.

[11] Galindo J, Tamayo P. Credit risk assessment using statistical and machine learning: basic methodology and risk modeling applications. Computational economics. 2000;15(1):107-43.

[12] Uddin MI, Zada N, Aziz F, Saeed Y, Zeb A, Ali Shah SA, et al. Prediction of future terrorist activities using deep neural networks. Complexity. 2020;2020.

[13] Dubach C, Jones TM, Bonilla EV, Fursin G, O'Boyle MF. Portable compiler optimisation across embedded programs and microarchitectures using machine learning. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture; 2009. p. 78-88.

[14] Tresp V, Bundschus M, Rettinger A, Huang Y. Towards machine learning on the semantic web. In: Uncertainty reasoning for the Semantic Web I. Springer; 2006. p. 282-314.

[15] Barreno M, Nelson B, Joseph AD, Tygar JD. The security of machine learning. Machine Learning. 2010;81(2):121-48.

[16] Sebe N, Cohen I, Garg A, Huang TS. Machine learning in computer vision. vol. 29. Springer Science & Business Media; 2005.

[17] Giusti A, Guzzi J, Cireşan DC, He FL, Rodríguez JP, Fontana F, et al. A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robotics and Automation Letters. 2015;1(2):661-7.

[18] Papageorgiou C, Poggio T. A trainable system for object detection. International journal of computer vision. 2000;38(1):15-33.

[19] Coglianese C, Lehr D. Regulating by robot: Administrative decision making in the machine-learning era. Geo LJ. 2016;105:1147.

[20] Pazzani MJ, Billsus D. Content-based recommendation systems. In: The adaptive web. Springer; 2007. p. 325-41.

[21] Zhou ZH. Machine learning. Springer Nature; 2021.

[22] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015;521(7553):436-44.

[23] Rokach L, Maimon O. Clustering methods. In: Data mining and knowledge discovery handbook. Springer; 2005. p. 321-52.

[24] Cayton L. Algorithms for manifold learning. Univ of California at San Diego Tech Rep. 2005;12(1-17):1.

[25] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM computing surveys (CSUR). 2009;41(3):1-58.

[26] Ramsundar B, Eastman P, Walters P, Pande V. Deep learning for the life sciences: applying deep learning to genomics, microscopy, drug discovery, and more. O'Reilly Media; 2019.

[27] Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596(7873):583-9.

[28] Vamathevan J, Clark D, Czodrowski P, Dunham I, Ferran E, Lee G, et al. Applications of machine learning in drug discovery and development. Nature reviews Drug discovery. 2019;18(6):463-77.

[29] Lavecchia A. Machine-learning approaches in drug discovery: methods and applications. Drug discovery today. 2015;20(3):318-31.

[30] Ranjbarzadeh R, Bagherian Kasgari A, Jafarzadeh Ghoushchi S, Anari S, Naseri M, Bendechache M. Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images. Scientific Reports. 2021;11(1):1-17.

[31] Cruz JA, Wishart DS. Applications of machine learning in cancer prediction and prognosis. Cancer informatics. 2006;2:117693510600200030.

[32] Iqbal MJ, Javed Z, Sadia H, Qureshi IA, Irshad A, Ahmed R, et al. Clinical applications of artificial intelligence and machine learning in cancer diagnosis: looking into the future. Cancer cell international. 2021;21(1):1-11.

[33] Learning M. Heart disease diagnosis and prediction using machine learning and data mining techniques: a review. Advances in Computational Sciences and Technology. 2017;10(7):2137-59.

[34] Hastie T, Stuetzle W. Principal curves. Journal of the American Statistical Association. 1989;84(406):502-16.

[35] Ferrarotti MJ, Rocchia W, Decherchi S. Finding principal paths in data space. IEEE transactions on neural networks and learning systems. 2018;30(8):2449-62.

[36] Decherchi S, Rocchia W. Import vector domain description: a kernel logistic one-class learning algorithm. IEEE transactions on neural networks and learning systems. 2016;28(7):1722-9.

[37] Decherchi S, Cavalli A. Fast and Memory-Efficient Import Vector Domain Description. Neural Processing Letters. 2020;52(1):511-24.

[38] Barlow HB. Unsupervised learning. Neural computation. 1989;1(3):295-311.

[39] Kotsiantis S, Kanellopoulos D. Association rules mining: A recent overview. GESTS International Transactions on Computer Science and Engineering. 2006;32(1):71-82.

[40] Khan SS, Madden MG. One-class classification: taxonomy of study and review of techniques. The Knowledge Engineering Review. 2014;29(3):345-74.

[41] Ruspini EH. A new approach to clustering. Information and control. 1969;15(1):22-32.

[42] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence. 2013;35(8):1798-828.

[43] Munson MA, Caruana R. On feature selection, bias-variance, and bagging. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer; 2009. p. 144-59.

[44] Yan K, Zhang D. Feature selection and analysis on correlated gas sensor data with recursive feature elimination. Sensors and Actuators B: Chemical. 2015;212:353-63.

[45] Baraniuk RG. Compressive sensing [lecture notes]. IEEE signal processing magazine. 2007;24(4):118-21.

[46] Fefferman C, Mitter S, Narayanan H. Testing the manifold hypothesis. Journal of the American Mathematical Society. 2016;29(4):983-1049.

[47] Manifold Learning;. Accessed: 2022-10-09. Available from: https://scikit-learn.org/stable/modules/manifold.html.

[48] Izenman AJ. Introduction to manifold learning. Wiley Interdisciplinary Reviews: Computational Statistics. 2012;4(5):439-46.

[49] Biau G, Fischer A. Parameter selection for principal curves. IEEE Transactions on Information Theory. 2011;58(3):1924-39.

[50] Evgeniou T, Pontil M, Poggio T. Regularization networks and support vector machines. Advances in computational mathematics. 2000;13(1):1-50.

[51] Moya MM, Koch MW, Hostetler LD. One-class classifier networks for target recognition applications. NASA STI/Recon Technical Report N. 1993;93:24043.

[52] Tuor A, Kaplan S, Hutchinson B, Nichols N, Robinson S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In: Workshops at the Thirty-First AAAI Conference on Artificial Intelligence; 2017. .

[53] Adewumi AO, Akinyelu AA. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. International Journal of System Assurance Engineering and Management. 2017;8(2):937-53.

[54] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. Springer; 2015. p. 234-41.

[55] Stojanovic L, Dinic M, Stojanovic N, Stojadinovic A. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In: 2016 IEEE international conference on big data (big data). IEEE; 2016. p. 1647-52.

[56] Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC. Estimating the support of a high-dimensional distribution. Neural computation. 2001;13(7):1443-71.

[57] Tax DM, Duin RP. Support vector data description. Machine learning. 2004;54(1):45-66.

[58] Vapnik VN. An overview of statistical learning theory. IEEE transactions on neural networks. 1999;10(5):988-99.

[59] Erfani SM, Rajasegarar S, Karunasekera S, Leckie C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recognition. 2016;58:121-34.

[60] Rudi A, Camoriano R, Rosasco L. Less is More: Nyström Computational Regularization. In: NIPS; 2015. p. 1657-65.

[61] Drineas P, Mahoney MW, Cristianini N. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. journal of machine learning research. 2005;6(12).

[62] Williams C, Seeger M. Using the Nyström method to speed up kernel machines. Advances in neural information processing systems. 2000;13.

[63] Golub GH, Van Loan CF. Matrix computations. JHU press; 2013.

[64] Kumar S, Mohri M, Talwalkar A. Ensemble Nystrom Method. In: Bengio Y, Schuurmans D, Lafferty J, Williams C, Culotta A, editors. Advances in Neural Information Processing Systems. vol. 22. Curran Associates,

Inc.; 2009. Available from: https://proceedings.neurips.cc/paper/2009/file/a49e9411d64ff53eccfdd09ad10a15b3-Paper.pdf.

[65] Pang G, Shen C, Cao L, Hengel AVD. Deep learning for anomaly detection: A review. ACM Computing Surveys (CSUR). 2021;54(2):1-38.

[66] Chalapathy R, Chawla S. Deep learning for anomaly detection: A survey. arXiv preprint arXiv:190103407. 2019.

[67] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. science. 2006;313(5786):504-7.

[68] Hadsell R, Chopra S, LeCun Y. Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2. IEEE; 2006. p. 1735-42.

[69] Vincent P, Larochelle H, Bengio Y, Manzagol PA. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning; 2008. p. 1096-103.

[70] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ, editors. Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc.; 2014. p. 2672-80.

[71] Kingma DP, Welling M. Auto-encoding variational bayes. arXiv preprint arXiv:13126114. 2013.

[72] Schlegl T, Seeböck P, Waldstein SM, Schmidt-Erfurth U, Langs G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: International conference on information processing in medical imaging. Springer; 2017. p. 146-57.

[73] Deecke L, Vandermeulen R, Ruff L, Mandt S, Kloft M. Image anomaly detection with generative adversarial networks. In: Joint european conference on machine learning and knowledge discovery in databases. Springer; 2018. p. 3-17.

[74] Zhang Z, Chen S, Sun L. P-kdgan: Progressive knowledge distillation with gans for one-class novelty detection. arXiv preprint arXiv:200706963. 2020.

[75] Perera P, Nallapati R, Xiang B. Ocgan: One-class novelty detection using gans with constrained latent representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 2898-906.

[76] Schlachter P, Liao Y, Yang B. Deep one-class classification using intra-class splitting. In: 2019 IEEE Data Science Workshop (DSW). IEEE; 2019. p. 100-4.

[77] Shrestha A, Mahmood A. Review of deep learning algorithms and architectures. IEEE Access. 2019;7:53040-65.

[78] Hu W, Wang M, Qin Q, Ma J, Liu B. HRN: A holistic approach to one class learning. Advances in Neural Information Processing Systems. 2020;33:19111-24.

[79] Ruff L, Vandermeulen R, Goernitz N, Deecke L, Siddiqui SA, Binder A, et al. Deep one-class classification. In: International conference on machine learning. PMLR; 2018. p. 4393-402.

[80] Chong P, Ruff L, Kloft M, Binder A. Simple and Effective Prevention of Mode Collapse in Deep One-Class Classification. In: 2020 International Joint Conference on Neural Networks (IJCNN); 2020. p. 1-9.

[81] Zhou Y, Liang X, Zhang W, Zhang L, Song X. VAE-based Deep SVDD for anomaly detection. Neurocomputing. 2021;453:131-40.

[82] Zhai X, Oliver A, Kolesnikov A, Beyer L. S4l: Self-supervised semi-supervised learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019. p. 1476-85.

[83] Masi F, Stefanou I, Vannucci P, Maffi-Berthier V. Thermodynamics-based Artificial Neural Networks for constitutive modeling. Journal of the Mechanics and Physics of Solids. 2021;147:104277.

[84] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. nature. 1986;323(6088):533-6.

[85] Toga MIR. Calculus with analytic geometry. 1988.

[86] Murtagh F. Multilayer perceptrons for classification and regression. Neurocomputing. 1991;2(5-6):183-97.

[87] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278-324.

[88] Convolutional Neural Network;. Accessed: 2022-10-09. Available from: https://en.wikipedia.org/wiki/Convolutional_neural_network.

[89] Rawat W, Wang Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Computation. 2017;29(9):2352-449.

[90] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 770-8.

[91] Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 1998;6(02):107-16.

[92] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv. 2014;abs/1412.6980. Available from: http://arxiv.org/abs/1412.6980.

[93] Kingma DP, Welling M, et al. An introduction to variational autoencoders. Foundations and Trends® in Machine Learning. 2019;12(4):307-92.

[94] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning; 2008. p. 160-7.

[95] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013.

[96] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014. p. 1532-43.

[97] Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. arXiv preprint arXiv:160701759. 2016.

[98] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805. 2018.

[99] Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, et al. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:190711692. 2019.

[100] Conneau A, Khandelwal K, Goyal N, Chaudhary V, Wenzek G, Guzmán F, et al. Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:191102116. 2019.

[101] Goyal P, Caron M, Lefaudeux B, Xu M, Wang P, Pai V, et al. Self-supervised pretraining of visual features in the wild. arXiv preprint arXiv:210301988. 2021.

[102] Becker S, Hinton GE. Self-organizing neural network that discovers surfaces in random-dot stereograms. Nature. 1992;355(6356):161-3.

[103] Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature verification using a" siamese" time delay neural network. Advances in neural information processing systems. 1993;6.

[104] Goldberger J, Hinton GE, Roweis S, Salakhutdinov RR. Neighbourhood components analysis. Advances in neural information processing systems. 2004;17.

[105] Chopra S, Hadsell R, LeCun Y. Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1. IEEE; 2005. p. 539-46.

[106] Caron M, Bojanowski P, Mairal J, Joulin A. Unsupervised pre-training of image features on non-curated data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019. p. 2959-68.

[107] Yan X, Misra I, Gupta A, Ghadiyaram D, Mahajan D. Clusterfit: Improving generalization of visual representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020. p. 6509-18.

[108] Chen X, Fan H, Girshick R, He K. Improved baselines with momentum contrastive learning. arXiv preprint arXiv:200304297. 2020.

[109] Caron M, Misra I, Mairal J, Goyal P, Bojanowski P, Joulin A. Unsupervised learning of visual features by contrasting cluster assignments. Advances in Neural Information Processing Systems. 2020;33:9912-24.

[110] Chen X, He K. Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 15750-8.

[111] Zbontar J, Jing L, Misra I, LeCun Y, Deny S. Barlow twins: Self-supervised learning via redundancy reduction. In: International Conference on Machine Learning. PMLR; 2021. p. 12310-20.

[112] Grill JB, Strub F, Altché F, Tallec C, Richemond P, Buchatskaya E, et al. Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems. 2020;33:21271-84.

[113] Sohn K, Li CL, Yoon J, Jin M, Pfister T. Learning and evaluating representations for deep one-class classification. arXiv preprint arXiv:201102578. 2020.

[114] Saunshi N, Plevrakis O, Arora S, Khodak M, Khandeparkar H. A theoretical analysis of contrastive unsupervised representation learning. In: International Conference on Machine Learning. PMLR; 2019. p. 5628-37.

[115] Tosh C, Krishnamurthy A, Hsu D. Contrastive learning, multi-view redundancy, and linear models. In: Algorithmic Learning Theory. PMLR; 2021. p. 1179-206.

[116] Gardini E, Ferrarotti MJ, Cavalli A, Decherchi S. Using principal paths to walk through music and visual art style spaces induced by convolutional neural networks. Cognitive Computation. 2021;13(2):570-82.

[117] Gardini E, Giorgi FM, Decherchi S, Cavalli A. Spathial: an R package for the evolutionary analysis of biological data. Bioinformatics. 2020;36(17):4664-7.

[118] Gardini E, Cavalli A, Decherchi S. An ab initio local principal path algorithm. In: 2021 International Joint Conference on Neural Networks (IJCNN). IEEE; 2021. p. 1-8.

[119] Ragusa E, Gastaldo P, Zunino R, Ferrarotti MJ, Rocchia W, Decherchi S. Cognitive Insights into Sentic Spaces Using Principal Paths. Cognitive Computation. 2019;11(5):656-75.

[120] Dijkstra EW, et al. A note on two problems in connexion with graphs. Numerische mathematik. 1959;1(1):269-71.

[121] Lloyd S. Least squares quantization in PCM. IEEE transactions on information theory. 1982;28(2):129-37.

[122] MacQueen J. Classification and analysis of multivariate observations. In: 5th Berkeley Symp. Math. Statist. Probability; 1967. p. 281-97.

[123] Arthur D, Vassilvitskii S. k-means++: The advantages of careful seeding. Stanford; 2006.

[124] Bottou L, Bengio Y. Convergence properties of the k-means algorithms. Advances in neural information processing systems. 1994;7.

[125] Thorndike RL. Who belongs in the family? Psychometrika. 1953;18(4):267-76.

[126] Camacho DM, Collins KM, Powers RK, Costello JC, Collins JJ. Next-generation machine learning for biological networks. Cell. 2018;173(7):1581-92.

[127] Tomczak K, Czerwińska P, Wiznerowicz M. The Cancer Genome Atlas (TCGA): an immeasurable source of knowledge. Contemporary oncology. 2015;19(1A):A68.

[128] Lonsdale J, Thomas J, Salvatore M, Phillips R, Lo E, Shad S, et al. The genotype-tissue expression (GTEx) project. Nature genetics. 2013;45(6):580.

[129] Svensson V, Vento-Tormo R, Teichmann SA. Exponential scaling of single-cell RNA-seq in the past decade. Nature protocols. 2018;13(4):599.

[130] Pastushenko I, Blanpain C. EMT transition states during tumor progression and metastasis. Trends in cell biology. 2018.

[131] Saelens W, Cannoodt R, Todorov H, Saeys Y. A comparison of single-cell trajectory inference methods. Nature biotechnology. 2019;37(5):547-54.

[132] Trapnell C, Cacchiarelli D, Grimsby J, Pokharel P, Li S, Morse M, et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nature biotechnology. 2014;32(4):381-6.

[133] Qiu X, Mao Q, Tang Y, Wang L, Chawla R, Pliner HA, et al. Reversed graph embedding resolves complex single-cell trajectories. Nature methods. 2017;14(10):979-82.

[134] Cao J, Spielmann M, Qiu X, Huang X, Ibrahim DM, Hill AJ, et al. The single-cell transcriptional landscape of mammalian organogenesis. Nature. 2019;566(7745):496-502.

[135] van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research. 2008;9(86):2579-605.

[136] Wasserstein RL, Lazar NA. The ASA statement on p-values: context, process, and purpose. Taylor & Francis; 2016.

[137] Storey JD. The positive false discovery rate: a Bayesian interpretation and the q-value. The annals of statistics. 2003;31(6):2013-35.

[138] Weinstein JN, Collisson EA, Mills GB, Shaw KR, Ozenberger BA, Ellrott K, et al. The cancer genome atlas pan-cancer analysis project. Nature genetics. 2013;45(10):1113-20.

[139] Sondka Z, Bamford S, Cole CG, Ward SA, Dunham I, Forbes SA. The COSMIC Cancer Gene Census: describing genetic dysfunction across all human cancers. Nature Reviews Cancer. 2018;18(11):696-705.

[140] Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. bioinformatics. 2010;26(1):139-40.

[141] Karlsson J, Kroneis T, Jonasson E, Larsson E, Ståhlberg A. Transcriptomic characterization of the human cell cycle in individual unsynchronized cells. Journal of molecular biology. 2017;429(24):3909-24.

[142] Sorensen V, Lansing JS, Thummanapalli N, Cambria E. Mood of the planet: Challenging visions of big data in the arts. Cognitive Computation. 2022;14(1):310-21.

[143] Sensity; 2004- 2009. Accessed: 2020-12-02. Available from: https://stanza.co.uk/sensity/.

[144] Illuminations; 2013. Accessed: 2020-09-18. Available from: http://vibeke.info/illuminations.

[145] Jing Y, Yang Y, Feng Z, Ye J, Yu Y, Song M. Neural Style Transfer: A Review. IEEE Transactions on Visualization and Computer Graphics. 2020;26(11):3365-85.

[146] Lecoutre A, Negrevergne B, Yger F. Recognizing Art Style Automatically in Painting with Deep Learning. In: Zhang ML, Noh YK, editors. Proceedings of the Ninth Asian Conference on Machine Learning. vol. 77 of Proceedings of Machine Learning Research. PMLR; 2017. p. 327-42.

[147] Karayev S, Trentacoste M, Han H, Agarwala A, Darrell T, Hertzmann A, et al. Recognizing Image Style. arXiv. 2014;abs/1311.3715. Available from: http://arxiv.org/abs/1311.3715.

[148] Tan WR, Chan CS, Aguirre HE, Tanaka K. Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification. In: 2016 IEEE International Conference on Image Processing (ICIP); 2016. p. 3703-7.

[149] Bahuleyan H. Music Genre Classification using Machine Learning Techniques. arXiv. 2018;abs/1804.01149. Available from: http://arxiv.org/abs/1804.01149.

[150] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation. 1989;1(4):541-51.

[151] LeCun Y, Boser BE, Denker JS, Henderson D, Howard RE, Hubbard WE, et al. Handwritten Digit Recognition with a Back-Propagation Network. In: Touretzky DS, editor. Advances in Neural Information Processing Systems. vol. 2. Morgan-Kaufmann; 1990. p. 396-404.

[152] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc.; 2012. p. 1097-105.

[153] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision. 2015;115(3):211-52.

[154] Mood of the Planet;. Accessed: 2020-12-02. Available from: https://sentic.net/mood-of-the-planet.pdf.

[155] Cetinic E, Lipic T, Grgic S. Learning the Principles of Art History with convolutional neural networks. Pattern Recognition Letters. 2020;129:56-62.

[156] Elgammal A, Liu B, Kim D, Elhoseiny M, Mazzone M. The Shape of Art History in the Eyes of the Machine. In: Proceedings of the 32nd AAAI conference on Artificial Intelligence; 2018. p. 2183-91.

[157] Gemmeke JF, Ellis DPW, Freedman D, Jansen A, Lawrence W, Moore RC, et al. Audio Set: An ontology and human-labeled dataset for audio events. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2017. p. 776-80.

[158] Bellman RE. Dynamic Programming. Princeton University Press; 2021. Available from: https://doi.org/10.1515/9781400835386 [cited 2022-06-26].

[159] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv. 2014;abs/1409.1556. Available from: http://arxiv.org/abs/1409.1556.

[160] Jolliffe I. Principal Component Analysis. In: Lovric M, editor. International Encyclopedia of Statistical Science. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 1094-6.

[161] bnegreve/rasta;. Accessed: 2020-12-04. Available from: https://github.com/bnegreve/rasta.

[162] WikiArt.org - Visual Art Encyclopedia;. Accessed: 2020-12-04. Available from: https://www.wikiart.org/.

[163] Recognizing the genre of music files using machine learning and deep learning models;. Accessed: 2020-12-04. Available from: https://github.com/HareeshBahuleyan/music-genre-classification.

[164] Goto M, Hashiguchi H, Nishimura T, Oka R. RWC Music Database: Popular, Classical and Jazz Music Databases. In: Proceedings of the 3rd International Conference on Music Information Retrieval (Ismir). vol. 2; 2002. p. 287-8.

[165] Goto M. Development of the RWC music database. In: Proceedings of the 18th International Congress on Acoustics (ICA). vol. 1; 2004. p. 553-6.

[166] Tzanetakis G, Cook PR. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing. 2002;10(5):293-302.

[167] The MagnaTagATune Dataset |City University MIRG;. Accessed: 2020-12-04. Available from: http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset.

[168] Chollet F, et al.. Keras; 2015. Available from: https://keras.io.

[169] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al.. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; 2015. Available from: https://www.tensorflow.org/.

[170] Cambria E, Liu Q, Decherchi S, Xing F, Kwok K. SenticNet 7: a commonsense-based neurosymbolic AI framework for explainable sentiment analysis. Proceedings of LREC 2022. 2022.

[171] LeCun Y, Cortes C, Burges CJC. MNIST handwritten digit database;. Accessed: 2021-02-05. Available from: http://yann.lecun.com/exdb/mnist/.

[172] Olivetti Faces data set;. Accessed: 2021-02-05. Available from: https://scikit-learn.org/0.19/datasets/olivetti_faces.html.

[173] Borji A. Pros and Cons of GAN Evaluation Measures. CoRR. 2018;abs/1802.03446. Available from: http://arxiv.org/abs/1802.03446.

[174] Ridgeway K, Snell J, Roads B, Zemel RS, Mozer MC. Learning to generate images with perceptual similarity metrics. CoRR. 2015;abs/1511.06409. Available from: http://arxiv.org/abs/1511.06409.

[175] Rezende DJ, Mohamed S, Wierstra D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In: Xing EP, Jebara T, editors. Proceedings of the 31st International Conference on Machine Learning. vol. 32 of Proceedings of Machine Learning Research. Bejing, China: PMLR; 2014. p. 1278-86.

[176] Aguti R, Gardini E, Bertazzo M, Decherchi S, Cavalli A. Probabilistic pocket druggability prediction via one-class learning. Frontiers in Pharmacology:1242.

[177] Tax DMJ, Duin RPW. Support vector domain description. Pattern Recognition Letters. 1999;20(11):1191-9.

[178] Zeng ZQ, Yu HB, Xu HR, Xie YQ, Gao J. Fast training support vector machines using parallel sequential minimal optimization. In: 2008 3rd international conference on intelligent system and knowledge engineering. vol. 1. IEEE; 2008. p. 997-1001.

[179] Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Methodological). 1977;39(1):1-22.

[180] Nicolaou KC. Advancing the Drug Discovery and Development Process. Angewandte Chemie International Edition. 2014;53(35):9128-40. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201404761.

[181] Jamali AA, Ferdousi R, Razzaghi S, Li J, Safdari R, Ebrahimie E. DrugMiner: comparative analysis of machine learning algorithms for prediction of potential druggable proteins. Drug Discovery Today. 2016;21(5):718-24. Available from: https://www.sciencedirect.com/science/article/pii/S1359644616000271.

[182] Csermely P, Korcsmáros T, Kiss HJM, London G, Nussinov R. Structure and dynamics of molecular networks: A novel paradigm of drug discovery: A comprehensive review. Pharmacology Therapeutics. 2013;138(3):333-408. Available from: https://www.sciencedirect.com/science/article/pii/S0163725813000284.

[183] Decherchi S, Cavalli A. Thermodynamics and Kinetics of Drug-Target Binding by Molecular Simulation. Chemical Reviews. 2020;120(23):12788-833. PMID: 33006893.

[184] Decherchi S, Grisoni F, Tiwary P, Cavalli A. Editorial: Molecular Dynamics and Machine Learning in Drug Discovery. Frontiers in Molecular Biosciences.

2021;8:231. Available from: https://www.frontiersin.org/article/10.3389/fmolb.2021.673773.

[185] Edfeldt FN, Folmer RH, Breeze AL. Fragment screening to predict druggability (ligandability) and lead discovery success. Drug discovery today. 2011;16(7-8):284-7.

[186] Qi SM, Dong J, Xu ZY, Cheng XD, Zhang WD, Qin JJ. PROTAC: An Effective Targeted Protein Degradation Strategy for Cancer Therapy. Frontiers in Pharmacology. 2021;12:1124.

[187] Shimokawa K, Shibata N, Sameshima T, Miyamoto N, Ujikawa O, Nara H, et al. Targeting the Allosteric Site of Oncoprotein BCR-ABL as an Alternative Strategy for Effective Target Protein Degradation. ACS Medicinal Chemistry Letters. 2017;8(10):1042-7. Available from: https://doi.org/10.1021/acsmedchemlett.7b00247.

[188] Agoni C, Olotu FA, Ramharack P, Soliman ME. Druggability and drug-likeness concepts in drug design: are biomodelling and predictive tools having their say? Journal of molecular modeling. 2020;26(6):1-11.

[189] Krivák R, Hoksza D. P2Rank: machine learning based tool for rapid and accurate prediction of ligand binding sites from protein structure. Journal of Cheminformatics. 2018 Aug;10(1):39. Available from: https://doi.org/10.1186/s13321-018-0285-8.

[190] Hussein HA, Borrel A, Geneix C, Petitjean M, Regad L, Camproux AC. PockDrug-Server: a new web server for predicting pocket druggability on holo and apo proteins. Nucleic acids research. 2015 Jul;43(W1):W436-42. 25956651[pmid]. Available from: https://pubmed.ncbi.nlm.nih.gov/25956651.

[191] Xie L, Li J, Xie L, Bourne PE. Drug discovery using chemical systems biology: identification of the protein-ligand binding network to explain the side effects of CETP inhibitors. PLoS computational biology. 2009;5(5):e1000387.

[192] Krasowski A, Muthas D, Sarkar A, Schmitt S, Brenk R. DrugPred: a structure-based approach to predict protein druggability developed using an extensive nonredundant data set. Journal of chemical information and modeling. 2011;51(11):2829-42.

[193] Volkamer A, Kuhn D, Rippmann F, Rarey M. DoGSiteScorer: a web server for automatic binding site prediction, analysis and druggability assessment. Bioinformatics. 2012 05;28(15):2074-5. Available from: https://doi.org/10.1093/bioinformatics/bts310.

[194] Krivák R, Hoksza D. Improving protein-ligand binding site prediction accuracy by classification of inner pocket points using local features. Journal of cheminformatics. 2015;7(1):1-13.

[195] Capra JA, Laskowski RA, Thornton JM, Singh M, Funkhouser TA. Predicting protein ligand binding sites by combining evolutionary sequence conservation and 3D structure. PLoS computational biology. 2009;5(12):e1000585.

[196] Le Guilloux V, Schmidtke P, Tuffery P. Fpocket: an open source platform for ligand pocket detection. BMC bioinformatics. 2009;10(1):1-11.

[197] Yuan JH, Han SB, Richter S, Wade RC, Kokh DB. Druggability Assessment in TRAPP Using Machine Learning Approaches. Journal of Chemical Information and Modeling. 2020 Mar;60(3):1685-99. Available from: https://doi.org/10.1021/acs.jcim.9b01185.

[198] Desaphy J, Azdimousa K, Kellenberger E, Rognan D. Comparison and Druggability Prediction of Protein–Ligand Binding Sites from Pharmacophore-Annotated Cavity Shapes. Journal of Chemical Information and Modeling. 2012;52(8):2287-99. PMID: 22834646. Available from: https://doi.org/10.1021/ci300184x.

[199] Yuan Y, Pei J, Lai L. Binding Site Detection and Druggability Prediction of Protein Targets for Structure-Based Drug Design. Current Pharmaceutical Design. 2013;19(12):2326-33.

[200] Zhang H, Saravanan KM, Lin J, Liao L, Ng JTY, Zhou J, et al. DeepBindPoc: a deep learning method to rank ligand binding pockets using molecular vector representation. PeerJ. 2020 Apr;8:e8864. Available from: https://doi.org/10.7717/peerj.8864.

[201] Pu L, Govindaraj RG, Lemoine JM, Wu HC, Brylinski M. DeepDrug3D: Classification of ligand-binding pockets in proteins with a convolutional neural network. PLOS Computational Biology. 2019 02;15(2):1-23. Available from: https://doi.org/10.1371/journal.pcbi.1006718.

[202] Mallet V, Checa Ruano L, Moine Franel A, Nilges M, Druart K, Bouvier G, et al. InDeep: 3D fully convolutional neural networks to assist in silico drug design on protein–protein interactions. Bioinformatics. 2021 12. Btab849. Available from: https://doi.org/10.1093/bioinformatics/btab849.

[203] Aggarwal R, Gupta A, Chelur V, Jawahar CV, Priyakumar UD. DeepPocket: Ligand Binding Site Detection and Segmentation using 3D Convolutional Neural Networks. Journal of Chemical Information and Modeling. 2022;accepted. PMID: 34374539. Available from: https://doi.org/10.1021/acs.jcim.1c00799.

[204] Stepniewska-Dziubinska MM, Zielenkiewicz P, Siedlecki P. Improving detection of protein-ligand binding sites with 3D segmentation. Scientific Reports. 2020 Mar;10(1):5035. Available from: https://doi.org/10.1038/s41598-020-61860-z.

[205] Kandel J, Tayara H, Chong KT. PUResNet: prediction of protein-ligand binding sites using deep residual neural network. Journal of Cheminformatics. 2021 Sep;13(1):65. Available from: https://doi.org/10.1186/s13321-021-00547-7.

[206] Hajduk PJ, Huth JR, Fesik SW. Druggability indices for protein targets derived from NMR-based screening data. Journal of medicinal chemistry. 2005;48(7):2518-25.

[207] Schmidtke P, Barril X. Understanding and Predicting Druggability. A High-Throughput Method for Detection of Drug Binding Sites. Journal of Medicinal Chemistry. 2010;53(15):5858-67. PMID: 20684613.

[208] Decherchi S, Spitaleri A, Stone J, Rocchia W. NanoShaper–VMD interface: computing and visualizing surfaces, pockets and channels in molecular systems. Bioinformatics. 2018 08;35(7):1241-3. Available from: https://doi.org/10.1093/bioinformatics/bty761.

[209] Wilson L, Krasny R. Comparison of the MSMS and NanoShaper molecular surface triangulation codes in the TABI Poisson–Boltzmann solver. Journal of Computational Chemistry. 2021;42(22):1552-60.

[210] Gao Z, Li H, Zhang H, Liu X, Kang L, Luo X, et al. PDTD: a web-accessible protein database for drug target identification. BMC bioinformatics. 2008;9(1):1-7.

[211] Volkamer A, Kuhn D, Grombacher T, Rippmann F, Rarey M. Combining Global and Local Measures for Structure-Based Druggability Predictions. Journal of Chemical Information and Modeling. 2012;52(2):360-72. PMID: 22148551.

[212] Breiman L. Random forests. Machine learning. 2001;45(1):5-32.

[213] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-30.

[214] Jiang H, Nachum O. Identifying and Correcting Label Bias in Machine Learning; 2019.

[215] La Sala G, Decherchi S, De Vivo M, Rocchia W. Allosteric Communication Networks in Proteins Revealed through Pocket Crosstalk Analysis. ACS Central Science. 2017;3(9):949-60. PMID: 28979936. Available from: https://doi.org/10.1021/acscentsci.7b00211.

[216] Wang S, Zeng Y, Liu X, Zhu E, Yin J, Xu C, et al. Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. Advances in neural information processing systems. 2019;32.

[217] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms; 2017.

[218] Krizhevsky A, Hinton G, et al. Learning multiple layers of features from tiny images. 2009.

[219] Schölkopf B, Smola A, Müller KR. Kernel principal component analysis. In: International conference on artificial neural networks. Springer; 1997. p. 583-8.

[220] Deep representation one class;. Accessed: 2020-12-04. Available from: https://github.com/google-research/deep_representation_one_class.

[221] Aldraimli M, Soria D, Parkinson J, Thomas E, Bell J, Dwek M, et al. Machine learning prediction of susceptibility to visceral fat associated diseases. Health and Technology. 2020;10(4):925-44.

[222] Self-supervised contrastive learning with SimSiam;. Available from: https://keras.io/examples/vision/simsiam/.

[223] Chaudhuri A, Sadek C, Kakde D, Wang H, Hu W, Jiang H, et al. The trace kernel bandwidth criterion for support vector data description. Pattern Recognition. 2021;111:107662.

[224] Peyré G, Cuturi M, et al. Computational optimal transport: With applications to data science. Foundations and Trends® in Machine Learning. 2019;11(5-6):355-607.

[225] Piccoli B, Rossi F. Generalized Wasserstein distance and its application to transport equations with source. Archive for Rational Mechanics and Analysis. 2014;211(1):335-58.

[226] Botev ZI, Grotowski JF, Kroese DP. Kernel density estimation via diffusion. The annals of Statistics. 2010;38(5):2916-57.

[227] Gou J, Yu B, Maybank SJ, Tao D. Knowledge distillation: A survey. International Journal of Computer Vision. 2021;129(6):1789-819.

[228] Shneiderman B. Human-centered artificial intelligence: Reliable, safe & trustworthy. International Journal of Human–Computer Interaction. 2020;36(6):495-504.

[229] Bærøe K, Miyata-Sturm A, Henden E. How to achieve trustworthy artificial intelligence for health. Bulletin of the World Health Organization. 2020;98(4):257.

[230] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST). 2019;10(2):1-19.

[231] Song S, Chaudhuri K, Sarwate AD. Stochastic gradient descent with differentially private updates. In: 2013 IEEE Global Conference on Signal and Information Processing. IEEE; 2013. p. 245-8.

[232] Jagadeesh KA, Wu DJ, Birgmeier JA, Boneh D, Bejerano G. Deriving genomic diagnoses without revealing patient genomes. Science. 2017;357(6352):692-5.

[233] Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L. Explaining explanations: An overview of interpretability of machine learning. In: 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA). IEEE; 2018. p. 80-9.

[234] Doshi-Velez F, Kim B. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:170208608. 2017.

[235] Lakkaraju H, Kamar E, Caruana R, Leskovec J. Faithful and customizable explanations of black box models. In: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society; 2019. p. 131-8.

[236] Lipton ZC. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue. 2018;16(3):31-57.

[237] Boil the Frog;. Accessed: 2020-12-04. Available from: http://boilthefrog.playlistmachinery.com.

[238] Cambria E, Li Y, Xing F, Poria S, Kwok K. SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. Association for Computing Machinery; 2020. p. 105-14.

A Proteins description

Table A.1 describes all the proteins included in the druggable (training) and the less druggable datasets. Druggable proteins are marked with d (training set), less druggable proteins are marked with n.

PDB code	**Name**	**Category**
1pwm	Aldose reductase	d
1lox	15-lipoxygenase	d
3etr	Xanthine oxidase	d
3f1q	Dihydroorotate dehydrogenase	d
3ia4	Dihydrofolate reductase	d
2cl5	Catechol-O-methyltransferase	d
1uou	Human thymidine phosphorylase	d
1t46	c-Kit kinase	d
1unl	cyclin-dependent kinase5	d
1q41	Glycogen synthase kinase 3	d
2i1m	FMS kinase	d
1pmn	c-Jun kinases	d
1fk9	HIV reverse transcriptase (nonnucleoside reverse transcriptase inhibitor binding site)	d
1e66	Acetylcholinesterase	d

1xoz	Phosphodiesterase 5A	d
1owe	Urokinase plasminogen activator	d
1r55	A disintegrin and metalloprotease	d
3f0r	Histone Deacetylase 8	d
1oq5	Carbonic anhydrase II	d
1kzn	DNA gyrase	d
2aa2	Mineralocorticoid receptor	d
3b68	Androgen receptor	d
1sqn	Progesterone receptor	d
1v16	Branched-chain alpha-keto acid dehydrogenase	n
3jdw	Arginine:glycine amidinotransferase	n
1ajs	Aspartate aminotransferase	n
1wvc	CDP-D-glucose synthase	n
1kc7	Pyruvate phosphate dikinase	n
1mai	Phospholipase C	n
1px4	Beta-galactosidase	n
1od8	Xylanase	n
1bmq	Interleukin-1 beta-converting enzyme 1	n

1bls	Beta-lactamase	n
1m0n	Dialkylglycine Decarboxylase	n
1ec9	D-glucarate dehydratase	n
1b74	Glutamate racemase	n
1g98	Phosphoglucose isomerase	n
1e9x	Cytochrome P450 14alpha -sterol demethylase	d
1hw8	3-hydroxy-3-methylglutaryl-CoA	d
1sqi	4-hydroxyphenylpyruvate dioxygenase	d
1r9o	Cytochrome P450 2C9	d
4cox	Cyclooxygenase 2	d
1c14	Enoyl reductase	d
2bxr	Monoamine oxidase A	d
2gh5	Glutathione reductase	d
1hvy	Thymidylate synthase	d
1rsz	Purine nucleoside phosphorylase	d
1n2v	tRNA-guanine transglycosylase	d
1v4s	Hexokinase	d
1u4d	ACK1 kinase	d

1m17	Epidermal growth factor receptor kinase	d
2dq7	Fyn kinase	d
1qpe	Lck kinase	d
1qhi	Thymidine kinase	d
2fb8	B-Raf kinase	d
1ke6	cyclin-dependent kinase2	d
2br1	Chk1 kinase	d
1ywr	p38 Mitogen-activated protein kinases	d
2ivu	RET kinase	d
2hiw	Abl tyrosin kinase	d
2i0e	Protein kinase C	d
1ywn	Vascular endothelial growth factor receptor-2	d
1ig3	Thiamin pyrophosphokinase	d
1yvf	Hepatitis C virus polymerase NS5B	d
1k8q	Gastric lipase	d
1kvo	Phospholipase A 2	d
1xm6	Phosphodiesterase 4B	d
1udt	Phosphodiesterase 5	d

1u30	Amylase	d
1r58	Methionine aminopeptidase-2	d
1rwq	Dipeptidyl peptidase-IV	d
1lpz	Factor Xa	d
2g24	Renin	d
1hvr	HIV protease	d
1gkc	Matrix metalloproteinase-9	d
1yqy	Lethal factor	d
1o5r	Adenosine deaminase	d
1js3	DOPA decarboxylase	d
1k7f	Tryptophan synthase	d
1j4i	FKBP13	d
1vbm	Tyrosyl-tRNA synthetase	d
1rv1	Ubiquitin-protein ligase E3 Mdm2	d
1gwr	Estrogen receptor	d
1m2z	Glucocorticoid receptor	d
3d4s	Beta-2-adrenergic receptor	d
1ai2	Isocitrate dehydrogenase	n
3pcm	3,4-dioxygenas	n
1d09	Aspartate transcarbamoylase	n

1c9y	Ornithine carbamoyltransferase	n
1gpu	Transketolase	n
1qmf	Penicillin binding protein-2X	n
1moq	Glucosamine 6-phosphate synthas	n
1ucn	Nucleoside diphosphate kinase	n
1t03	HIV reverse transcriptase (nucleoside binding site)	n
1qs4	HIV integrase	n
1fth	Acyl carrier protein synthase	n
1rnt	Ribonuclease T2	n
1onz	Protein-tyrosine phosphatase 1B	n
1x9d	Mannosidase	n
1nnc	Neuraminidase	n
1olq	Endo-beta-1,4-glucanase	n
1jak	Beta-N-Acetylhexosaminidases	n
1kts	Thrombin	n
1nlj	Cathepsin K	n
1icj	Peptide deformylase	n
1hqg	Arginase	n
2gsu	Phosphodiesterase-nucleotide Pyrophosphatase	n

1g7v	3-deoxy-D-manno-2-octulosonate-8-phosphate synthase	n
1f9g	Hyaluronate lyase	n
1qxo	Chorismate synthase	n
2gyi	D-xylose isomerase	n
1o8b	Ribose-5-phosphate isomerase	n
1cg0	Adenylosuccinate synthetase	n

Table A.1: *Druggability prediction: proteins description of the NRDLD dataset.*

Table A.2 describes all the proteins included in the PDTD (100-proteins) dataset.

PDB code	**Name**	**Category**
1a28	Progesterone receptor	d
1acj	Acetylcholine esterase	d
1aco	Aconite with transaconitate bound	d
1adc	NAD analogues bound to alcohol dehydrogenase	d
1coy	Cholesterol oxidases	d
1cqe	Prostaglandin H2 synthase-1	d
1d3g	Dihydroorotate dehydrogenase	d

1d6u	E. Coli amine oxidase	d
1db1	Nuclear receptor for vitamin D	d
1dht	Estrogenic 17-beta hydroxysteroid dehydrogenase	d
1diy	Cyclooxygenase active site of PGHS-1	d
1dkf	Heterodimeric complex of RAR and RXR	d
1e1f	Beta-glucosidase	d
1e3g	Androgen receptor	d
1e3k	Progesteron receptor	d
1e55	Mutant Monocut beta-glucosidase	d
1eet	HIV-1 reverse transcriptase	d
1efh	Hydroxysteroid sulfotransferase	d
1f2a	Cruzain hydrolase	d
1fm6	Heterodimer of the RXR-α and PPAR-γ	d
1gii	Cyclin dependent kinase	d
1gos	Monoamine oxidase B	d
1gp6	Anthocyanidin synthase	d
1gpk	Acetylcholinesterase	d

1gqs	Acetylcholinesterase complexed with NAP	d
1gs4	Androgen receptor ARccr	d
1h5u	Glycogen phosphorylase B	d
1h9u	Retinoid X receptor beta	d
1hb2	Isopenicillin N synthase	d
1hdy	Alcohol dehydrogenase variant	d
1hfc	Fibroblast collagenase	d
1hj1	Estrogen receptor beta	d
1hld	Liver alcohol dehydrogenase	d
1ho4	Pyridoxine 5-phosphate	d
1ht8	Oxidoreductase COX-1	d
1hy3	Estrogen sulfotransferase V269E	d
1hzx	Bovine Rhodopsin	d
1i7g	Human PPAR-α	d
1ie9	Nuclear receptor for vitamin D	d
1iiu	Plasma retinol-binding protein	d
1j90	Deoxyribonuclease kinase	d
1jbp	Catalytic subunit of c-AMP dependent protein kinase	d
1jkh	HIV-1 reverse transcriptase	d

1js3	Dopa decarboxylase	d
1k3u	Tryptophan synthase	d
1k4w	Nuclear receptor ROR-	d
1k74	Heterodimer of PPAR- and RXR-	d
1k7l	Human PPAR-	d
1lde	Liver alcohol dehydrogenase	d
1ldy	Liver alcohol dehydrogenase complexed to NADH and cyclohexyl formamide	d
1mup	Pheromone binding to two urinary proteins	d
1n7i	Phenylethanolamine N-metyltransferase	d
1nwk	Monomeric actin in the ATP state	d
1og5	Human cytochrome P450 CYP2C9	d
1oi9	Human thr160-phospho CDK2/cyclin A	d
1p1n	GluR2 ligand binding core (S1S2J) mutant	d
1p2d	Glycogen phosphorylase B	d

1p4g	Glycogen phosphorylase B in complex with C-(1-azido-alpha-D-glucopyranosyl) formamide	d
1p93	Glucocorticoid receptor	d
1pcg	Helix-stabilized cyclic peptides	d
1pha	Cytochrome P450-CAM	d
1pig	Pancreatic alpha-amylase	d
1ppl	Aspartyl proteinases	d
1qab	Retinol binding protein	d
1kvo	Phospholipase A 2	d
1qkm	Estrogen receptor β	d
1qkt	Mutant estrogen nuclear receptor	d
1qpb	Pyruvate decarboxylase	d
1r18	Isoaspartyl methyltransferase	d
1r1k	Heterodimer EcR/USP bound to ponasterone A	d
1rbp	Serum retinol binding protein	d
1rlb	Retinol binding protein complexed with transthyretin	d
1rt6	HIV-1 reverse transcriptase	d
1tvr	HIV-1 RT/9-CL TIBO	d
1uhl	LXRα-RXRβ LBD heterodimer	d

1ulb	Purine nucleoside phosphorylase	d
1uom	Estrogen receptor complexed with Tetrahydroisochiolin	d
1upv	Liver X receptor β	d
1v8b	Hydrolase	d
1vkg	HDAC8	d
1vlb	Aldehyde oxidoreductase	d
1vot	Acetylcholine esterase	d
1w6k	Human OSC	d
1x07	Undecaprenyl pyrophosphate synthase	d
1xnx	Androstane receptor	d
1y0s	PPAR-γ	d
1zhy	Oxysterol binding protein Osh4	d
2a3i	Mineralocorticoid receptor	d
2a3l	Adenosine 5'-Monophosphate deaminase	d
2ack	Acetylcholinesterase	d
2ae2	Tropinone reductase-II	d
2bx8	Human serum albumin	d
2dln	D-alanine ligase	d

2mas	Purine nucleoside hydrolase	d
3bto	Liver alcohol dehydrogenase	d
3ert	Estrogen receptor-α	d
3hvt	Human immunodeficiency virus type 1 reverse transcriptase heterodimer	d
4thi	Thiaminase I	d
6cox	Cyclooxygenase-2	d
8cat	Liver catalase	d

Table A.2: *Druggability prediction: proteins description of the PDTD dataset.*

B Deep-IVDD-KL additional experiments

The 64 testing samples with the highest and the lowest scores are reported below for each class of the CIFAR-10 dataset.

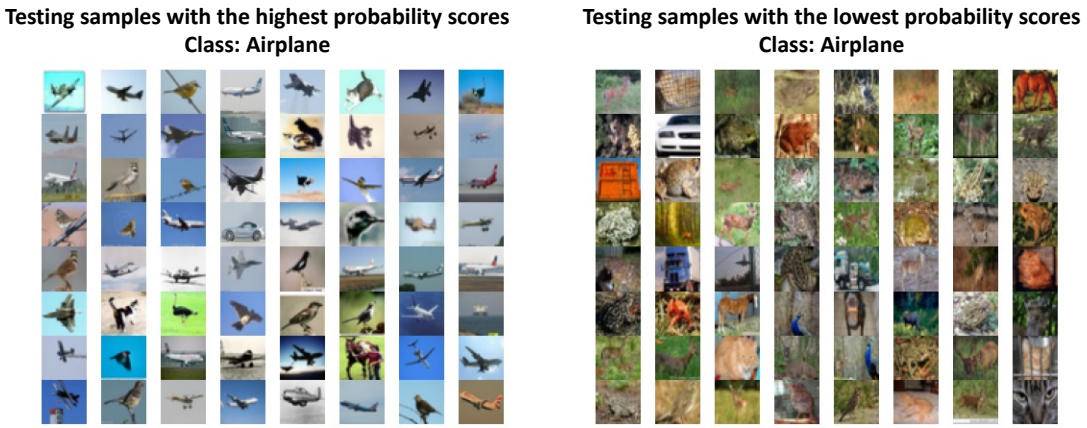

Figure B.1: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Airplane).*

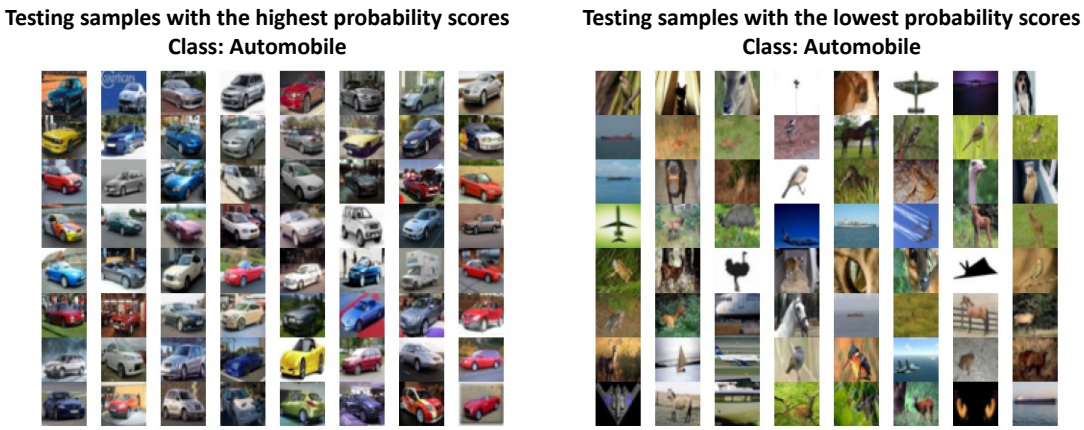

Figure B.2: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Automobile).*

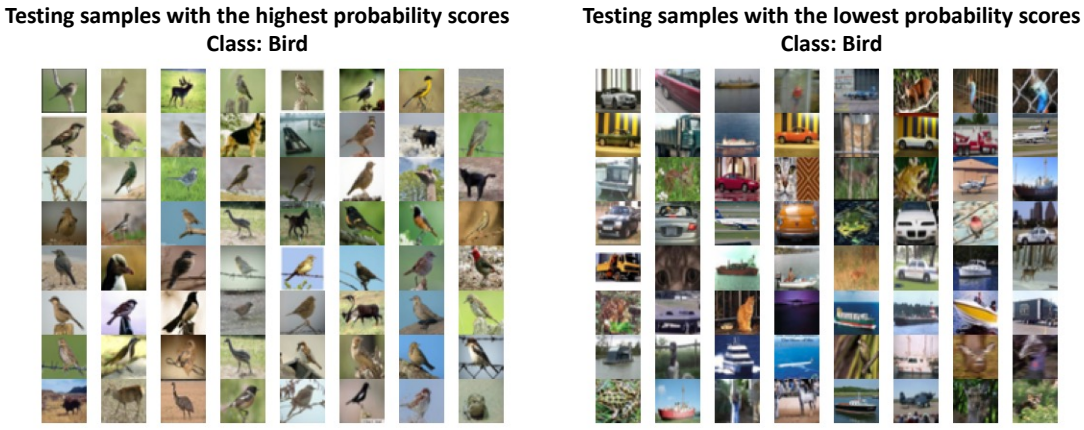

Figure B.3: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Bird).*

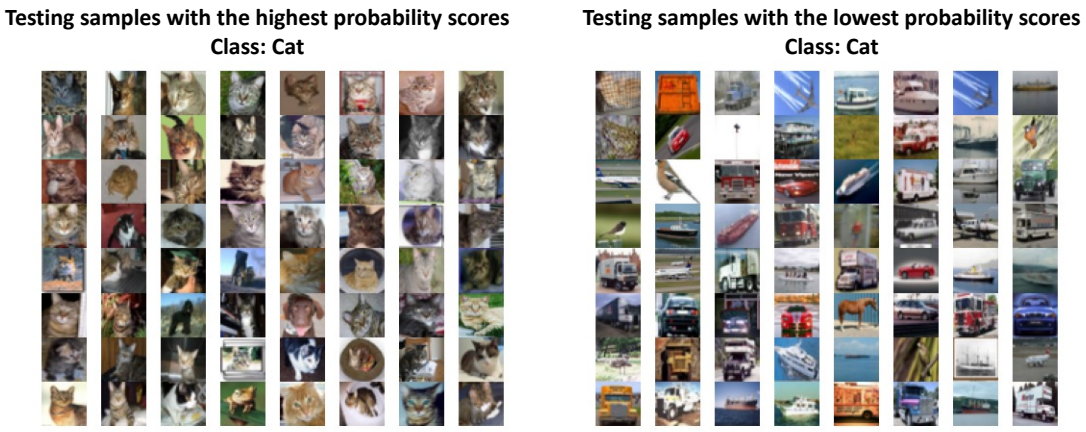

Figure B.4: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Cat).*

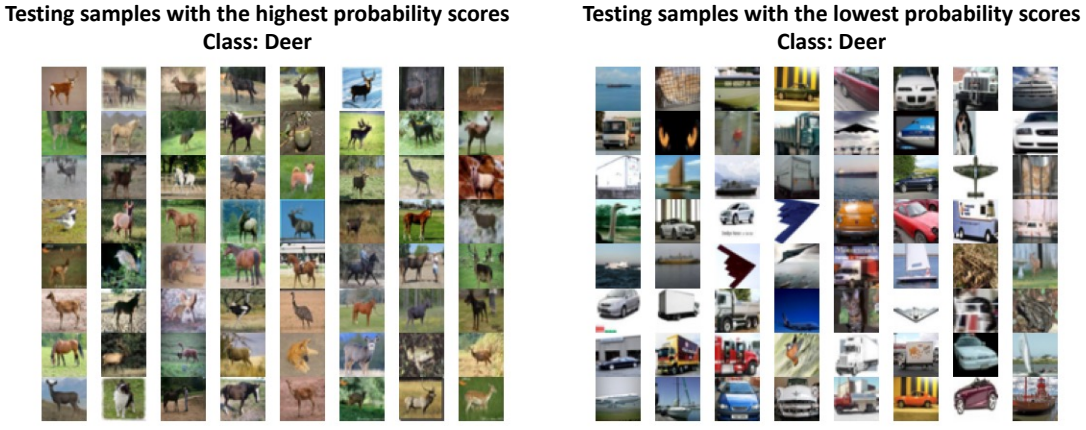

Figure B.5: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Deer).*

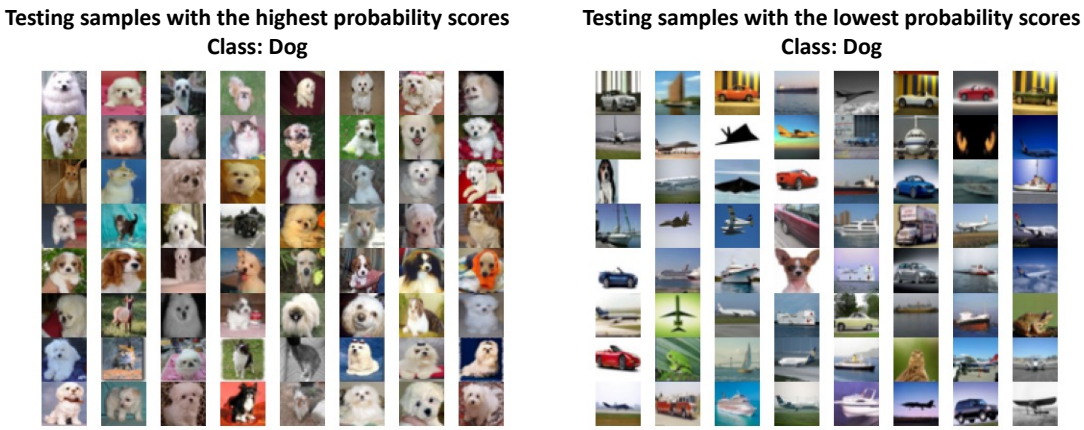

Figure B.6: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Dog).*

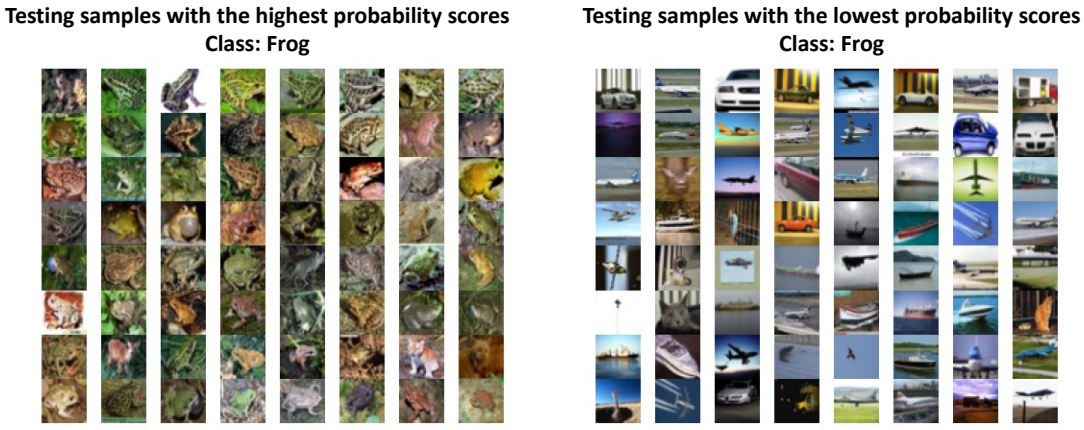

Figure B.7: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Frog).*

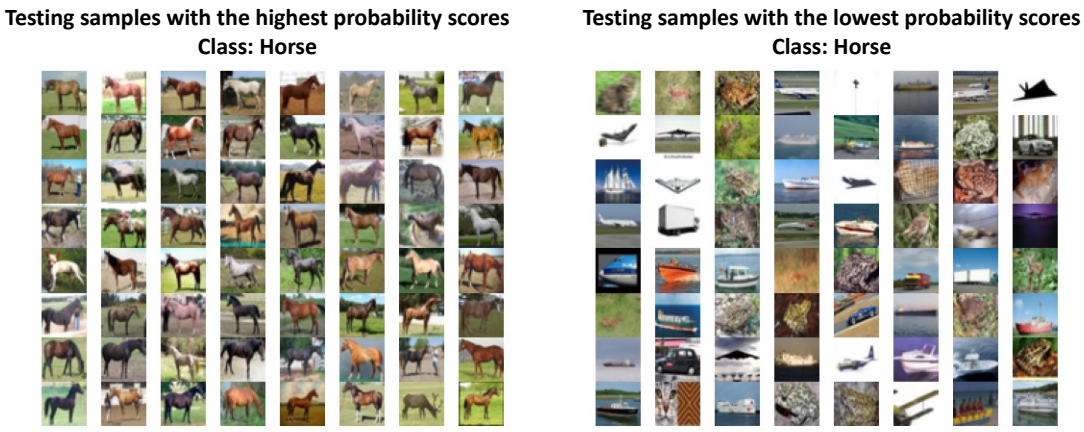

Figure B.8: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Horse).*

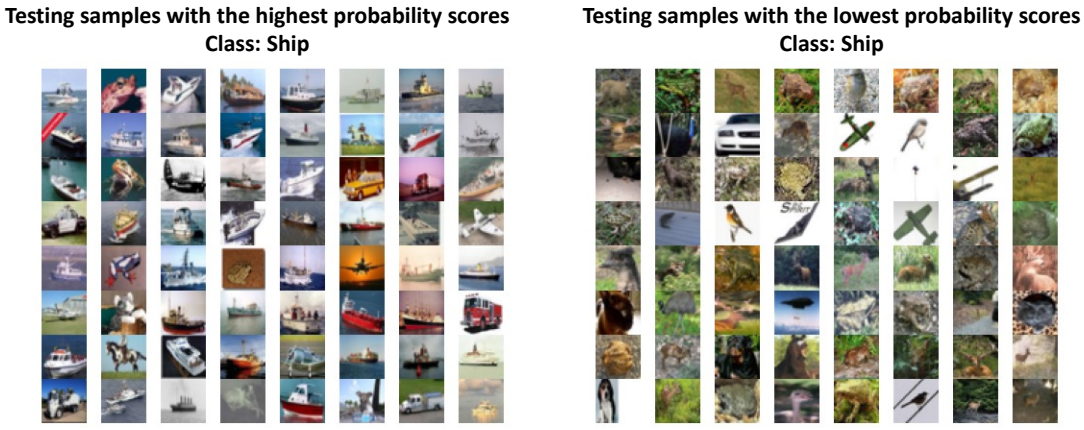

Figure B.9: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Ship).*

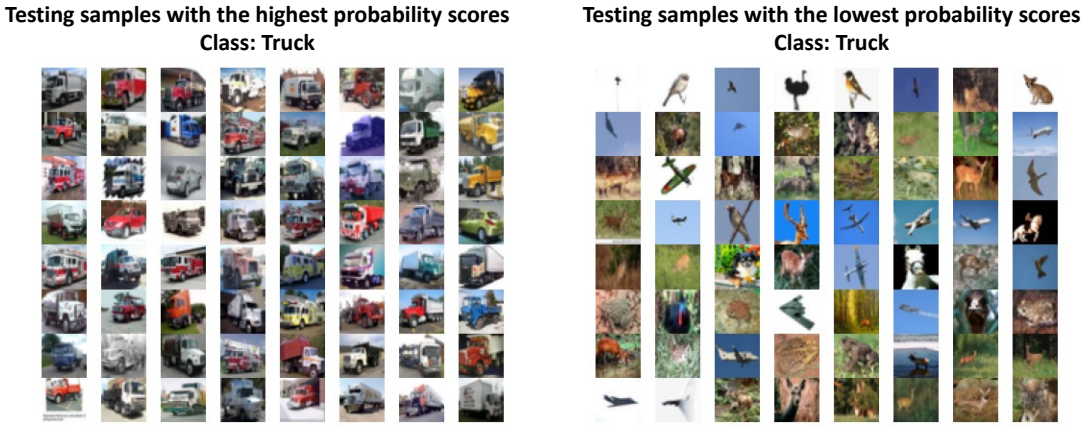

Figure B.10: *Deep-IVDD-KL: test samples with the highest/lowest scores (class: Truck).*