UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

TOPOGEN: TOPOLOGY-INFORMED GENERATIVE MODELS

Autor: Jack Benarroch Jedlicki

Directors: Dr. Sergio Escalera
Dr. Carles Casacuberta
Realitzat a: Departament de Matemàtiques
i Informàtica

Barcelona, 10 de juny de 2024

Contents

Abstract

The main goal of generative models is to learn an unknown distribution of data points in a high-dimensional space. However, generative models often face challenges such as mode collapse and time-extensive training processes. In order to address these problems, we propose a new way of training generative models, which relies on the implementation of topological regularizers extracting information from persistence diagrams. These topological loss terms inform about meaningful features of the spaces formed by the true data and the generated data, such as the presence of clusters, loops or higher-dimensional holes. We provide original proofs showing that the functions used are stable with respect to the data and generically differentiable, allowing their gradient descent based optimization. Some of the results obtained in this thesis are new results extending the current knowledge of differentiability through barcode space to more general classes of functions. As a consequence, this work expands the current possibilities of differentiable topological regularization. The developed topological regularizers are first tested in synthetic datasets, demonstrating their ability to continuously deform point clouds in order to obtain ground truth topological features. Then, the regularizers are tested in variational autoencoders in the FashionMNIST dataset, and we observe that they provide an improved performance compared to their non-regularized counterparts. Furthermore, these loss terms can be applied at any layer of the generative models, opening new ways of controlling the performance of inner layers and the spatial distribution of data codes in the latent space. We thus explore this possible line of application, and the striking effects observed suggest that topological regularization may be a useful ingredient for training generative models.

2020 Mathematics Subject Classification. 55N31, 62R40, 68T01, 68T07.

Resum

L'objectiu principal dels models generatius és aprendre una distribució desconeguda de punts de dades en un espai de dimensió alta. Tanmateix, aquests models sovint es troben amb reptes com el col·lapse de modes i processos d'entrenament que requereixen extensos períodes de temps. Per resoldre aquests problemes, proposem una nova manera d'entrenar models generatius, que es basa en la implementació de regularitzadors topológics que extreuen informació de diagrames de persistència. Aquests nous termes topológics de pèrdua informen sobre característiques significatives dels espais formats per les dades reals i les generades, com la presència de clústers, llaços o forats de dimensions superiors. Donem demostracions originals del fet que els regularitzadors utilitzats són estables respecte a les dades i genèricament diferenciables, permetent la seva optimització basada en el descens de gradient. Alguns dels resultats obtinguts en aquesta tesi són nous resultats que amplien el coneixement actual de la diferenciabilitat a través de l'espai de codis de barres a classes més generals de funcions. Com a conseqüència, aquest treball amplia les possibilitats actuals de la regularització topològica diferenciable. Els regularitzadors topològics desenvolupats són provats primer en conjunts de dades sintètics, demostrant la seva capacitat per deformar contínuament núvols de punts per obtenir característiques topològiques de referència. Després, provem els regularitzadors en *autoencoders* variacionals en el conjunt de dades FashionMNIST i mostrem que proporcionen un rendiment notablement millorat en comparació amb els seus homòlegs no regularitzats. A més, aquests termes de pèrdua es poden aplicar en qualsevol capa dels models generatius, obrint noves maneres de controlar el rendiment de les capes interiors i la distribució espacial dels codis de dades en l'espai latent. Així doncs, explorem aquesta possible línia d'aplicació, i els efectes sorprenents observats suggereixen que la regularització topològica pot ser un ingredient útil per a l'entrenament de models generatius.

Acknowledgements

I would like to to express my most sincere gratitude to Sergio Escalera, Carles Casacuberta, and Rubén Ballester.

I would like to thank Sergio for giving me the unique opportunity to conduct the Mathematics Bachelor Thesis under his co-supervision, for arranging such an amazing team, and for the invaluable support, guidance, and advice. I would also like to thank Carles for the invaluable advice and detailed explanations. And a special thanks to Rubén, this work would have not been possible without your persistent support.

I would also like to thank all the previous mathematicians and scientists whose groundbreaking contributions paved the way for this thesis, such as Carlsson, Zomorodian, Edelsbrunner, Cohen, Bauer, Leygonie, Kingma, Welling, and many others. This thesis is but a humble grain of salt amidst the vast mountain of knowledge they have been building.

Introduction

This work merges two different realms of Mathematics and Artificial Intelligence: persistent homology and generative models. We begin with brief overviews of each concept, before delving into the details of our proposed methodologies.

Generative models are a class of machine learning models capable of generating new data instances that resemble a given dataset. These models learn the probability distribution of inputs, which allows them to sample from the distribution and generate new instances consistent with the learned data distribution. Generative models can be used for data compression, image generation, denoising, unsupervised feature learning, and so on. In general, these models learn to reconstruct the distribution of the data using an intermediate step where the data is encoded in a lower-dimensional space, the so-called latent space. However, a problem most generative models face is the difficulty to learn the original distribution of the data. For instance, generative adversarial models are prone to mode-collapse, i.e., the generation of only a subset of the data; diffusion models require time-expensive training processes, and variational autoencoders may learn a data distribution in the latent space incompatible with the true data distribution, making them unable to generate realistic new data [1].

On the other hand, persistent homology emerges from the field of computational algebraic topology as a tool providing topological features describing the "shape" of complex data sets. For instance, given a point cloud in a (high-dimensional) metric space, persistent homology can be used to estimate the number of clusters, the number and rough size of loops formed by the data points, and the number of higher-dimensional holes. The information captured by persistent homology can be summarized in so-called *persistence diagrams* or *barcodes*. These objects can be seen as fingerprints of point clouds providing a description of their shape, even when the concept of shape is something we cannot visually grasp. Given the apparent usefulness of persistent homology, one could thus consider its use in generative models. In fact, in 2018 Khrulkov and Oseledets merged these two concepts with the development of the Geometry Score [2], a measure of the performance of generative adversarial networks based on the comparison of persistence diagrams of the true and the generated data. However, the method was created as an evaluation tool used outside of the training process, only useful for comparison of models and hyperparameter tuning. Similarly, in 2019 Charlier et al. [3] developed PHom-GeM, a measure based on persistence diagrams for comparing the nature of the true and the generated distributions by generative models, and Schiff et al. [4] used in 2020 persistent homology for measuring and understanding how information is encoded in the latent space of variational autoencoders (VAEs), when being trained for predicting 3D molecular structures. But again, these metrics were used only for evaluating the performance after training, and not as loss terms.

In this work, we use persistent homology *inside* the training process. In particular, we introduce a family of topological regularizers of generative models employing persistence diagrams arising from the true and the generated data. These regularizers are both stable with respect to perturbations of the data and differentiable under mild conditions —two results we prove in this work. The idea is that the minimization of these new loss terms can help ensure that both the true and generated data point clouds have similar shapes. Hence, this union seeks to enhance generative models with previously unexplored information related to the shape of data.

Previous work on this idea is very recent and not extended [5, 6, 7, 8]. Chen et al. developed TopoGAN in 2020 [5], a generative adversarial network (GAN) that used information from the persistence diagrams of individual images for generating images preserving topological properties such as connectedness and loopy-ness. However, this approach is limited to very specific datasets (e.g., images of neurons or road networks) and can be extremely time-expensive when working with large datasets, since an individual persistence diagram has to be computed for each image. Similarly, Mezghanni et al. [8] developed in 2021 a differentiable loss term based on persistence diagrams obtained from cubical complexes, with the goal of introducing physical and topological information (such as connectivity) about the shape of objects. The regularizer was applied on a GAN for generating 3D objects, and the results were promising in terms of reconstruction quality. However, their method relied on computing a persistence diagram for each 3D object, which can be time-consuming in large datasets, and the scope of application of the method proposed is restricted to a very specific type of problem. On the other hand, Moor et al. [6] developed in 2020 a topological regularizer applied on the latent batches of an autoencoder and proved its differentiability under weak theoretical assumptions. However, their study was limited to a single regularizer, which remained untested on the output data. Moreover, while their experiments managed to modulate latent vector distributions for two-dimensional latent spaces, they did not enhance the quality or diversity of the generated images. It is also important to note that, by the time of conducting this work, a new work was published [7] focusing on the use of persistent homology for learning disentanglement. The authors developed a topological loss term applied on VAEs and GANs relying on the Representation Topology Divergence measure [9] for unsupervised learning of disentangled representations. Their method did enhance disentangling according to several metrics; however its effect on the reconstruction quality and diversity of generated images was not a focus of the paper.

Building upon these findings, this thesis expands the scope of topology-informed generative models by presenting a diverse set of topological regularizers and applying them on generative models. The implementation of topological regularizers for training a variational autoencoder is illustrated in Figure A.1, and can be summarized as follows. Take a generative model that produces images, songs, or any type of data that can be represented as an array of real numbers, such as a variational autoencoder or a diffusion model, and view each data element as a point in Euclidean space $\mathbb{R}^d$ for some value d. For instance, 32×32 grayscale images are viewed as points in $\mathbb{R}^{1024}$. During each training iteration, a batch of N data points is given to the model and N new points are generated as output. Then, some measure of dissimilarity between the true and the generated data (e.g., binary cross-entropy (BCE) loss) is computed and used as a loss function. When implementing our regularizers, in each training iteration we compute the persistence diagram of the batch of N true data

points, and the persistence diagram of the N generated points, both viewed as point clouds in $\mathbb{R}^d$. The two resulting persistence diagrams are then compared using some measure of dissimilarity, and the regularizer captures this measure. As a consequence, the modification of the weights of the generative model through backpropagation aims to produce data with a spatial distribution that *looks like* the distribution of the true data. Furthermore, there is also an extension of this method, illustrated in Figure A.1 and described in more detail in Section 3.5.5, which relies on applying the topological regularizers on the batch of latent vectors instead of the final outputs of the model, in order to control the distribution in the latent space.

Structure of the thesis

In Chapter 1, we recall the mathematical foundations of simplicial homology and persistent homology. We begin with an introduction to simplicial homology in Section 1.1. In particular, we make an emphasis in the physical meaning of homology groups, which justifies their use. Then, in Sections 1.2 and 1.3, we describe how to obtain the persistent homology of a point cloud embedded in a metric space. In Sections 1.4 and 1.5 we follow a sequence of transformations from persistent homology to persistence diagrams. We finish the chapter with an essential property of persistence diagrams: their stability under small perturbations of the initial point cloud. In particular, we provide in Section 1.6 a theoretical bound of the stability of persistence diagrams with respect to perturbations of point clouds.

Chapter 2 delves into the differentiability of functions through barcode space, with the goal of giving conditions for smoothness of the proposed topological regularizers. In fact, the (generic) differentiability of these functions is the key ingredient for their gradient descent based optimization with neural networks, making it an essential property for their efficient implementation in generative models. However, topological regularizers are maps of the form $\mathcal{M} \to \text{Bar} \to \mathbb{R}$, where $\mathcal{M}$ is a smooth manifold —which in practical scenarios will often be some Euclidean space $\mathbb{R}^d$—, and Bar is the space of barcodes. Due to the structure of these functions, the proof of their smoothness is not straightforward. As a consequence, we follow the differentiability framework defined in [10]. The idea, roughly speaking, relies on decomposing a function $\mathcal{M} \to \text{Bar} \to \mathbb{R}$ into maps $\mathcal{M} \to \text{Bar}$ and maps $\text{Bar} \to \mathbb{R}$, followed by proving, if possible, the differentiability of both maps, where the concept of "differentiability" is a concept adapted to the structure of Bar. Then, using a result analogous to the chain rule, one can show that the composition map is actually differentiable with the usual definition of differentiability, i.e., as a map between smooth manifolds. We thus begin in Section 2.1 with an introduction to the framework of differentiability through Bar, and a proof of the chain rule in barcode space. Then, in Section 2.2 we prove the generic smoothness of the barcode generator of degree p (via the Rips filtration) B_p, with respect to the coordinates of the point cloud. Finally, in Section 2.3, which is original work except for Subsection 2.3.5, we prove the differentiability, under certain conditions, of five different types of functions from the space of point clouds to $\mathbb{R}$, factoring through barcode space. In Section 2.4, we prove another new result, which allows us to apply topological regularizers, which according to previous work would not be differentiable, in a differentiable manner. This opens the possibility of applying new topological regularizers in a differentiable way, such as persistent entropy-based regularizers.

Finally, we experimentally test the effect of these loss terms in Chapter 3. We first properly define in Definition 3.1 five types of topological regularizers, and unify and summarize their differentiability properties in Theorem 3.2, which stem directly from the work in the previous chapter. We then begin with a set of synthetic experiments with deformable point clouds in the plane in Section 3.3. Next, we apply the regularizers in variational autoencoders in Sections 3.4 and 3.5. We begin with a description of the working principle of VAEs and of topology-informed VAEs, so-called TopoVAEs. Then, we conduct a series of qualitative and quantitative tests comparing VAEs to TopoVAEs. The promising results confirm that the regularizers have a positive effect on the learning process of generative models, paving the way to the exploration of their use in other scenarios.

Contributions

The main theoretical contributions of this work are the following.

1. A stability bound for persistence diagrams arising from a point cloud embedded in a Euclidean space (in Section 1.6).

2. Original proofs of several differentiability results from [10] (in Sections 2.1 and 2.2). In general, unless stated otherwise, all proofs written down in this work rely on the author's own arguments.

3. The construction, in Sections 2.3 and 2.4, of six types of functions factoring through barcode space that can be used as regularizers in generative models or as loss functions in other machine learning problems. These functions are 1) push functions of any degree; 2) the squared Reininghaus dissimilarity between two persistence diagrams; 3) the difference between the scaled Gaussian density estimators of two persistence diagrams; 4) the squared difference between the 0-degree persistent entropy of two diagrams; 5) the bottleneck distance to a fixed persistence diagram, and 6) selective regularizers, which include, as a particular case, p-entropy regularizers. In addition, function 3) is a new description of the density of points in a 0-degree persistence diagram. Taking particular cases of these functions, we obtain five main topological regularizers that are then used experimentally in Chapter 3: the *p-push regularizer*, the *p-Reininghaus regularizer*, the *4SGDE regularizer*, the *p-bottleneck regularizer*, and the *p-entropy regularizer*. These regularizers are properly defined in Definition 3.1.

4. The proof of the generic differentiability in the space of point clouds (i.e., the differentiability in an open and dense subset of the space of point clouds) of functions 1), 2), 3) and 4) (the latter only for homology in degree 0) in Section 2.3. Proofs of the generic differentiability of selective regularizers in Section 2.4, and of the generic differentiability of selective p-persistent entropy generators in Section 2.4. In particular, the results involving functions 2), 3), 4), selective regularizers, and p-persistent entropy generators are, to the best of our knowledge, new results.

5. The proof of the differentiability of function 5) under computationally mild assumptions, which is also a new result, in Section 2.3.

6. These differentiability results are unified in Theorem 3.2, where we state that from the five topological regularizers developed, four are generically differentiable in the space of point clouds, and one is differentiable when the point cloud lies in a generic subspace of the space of point clouds and its persistence diagram satisfies computationally mild conditions.

In the experimental part, the main contributions are the following.

1. Implementation of p-bottleneck regularizers, p-Reininghaus regularizers, 4SGDE regularizers, and p-entropy regularizers in VAEs, and qualitative and quantitative analysis of their effect on the training process. The functions corresponding to these regularizers are given in Definition 3.1. Among the tests, we include experiments where the regularizers are applied directly on the latent vectors, allowing to control the distribution of the latent space and showing remarkable modifications of their distribution. Although the page limit has not allowed us to delve deeper into a formal analysis of the latter behaviour, we believe that the experiments presented show the promising potential of topological regularization and possible future avenues of research.

2. The code for implementing these loss terms and conducting all experiments mentioned in this work, which has been made available in the GitHub repository https://github.com/JackBJ23/TopoGEN.

Chapter 1

Persistent homology

This chapter mainly relies on existing literature in the field of persistent homology [10, 11, 12, 13, 14], textbooks on algebraic topology [15] and topological data analysis [16], and lecture notes on simplicial homology [17].

1.1 Simplicial homology

Simplicial homology is an algebraic characteristic of a specific type of objects, known as abstract simplicial complexes, which are defined as follows.

Definition 1.1. An *abstract simplicial complex* is a pair (K, S), where K is a set and S is a collection of subsets of K, such that for all $v \in K$ we have $\{v\} \in S$, and for all $\sigma \in S$, if $\tau \subseteq \sigma$ then $\tau \in S$.

We name each element $\sigma \in S$ a *k-simplex*, where $k \coloneqq |\sigma| - 1$ is its *dimension*, and a simplex of the form $\{v\}$ is called a *vertex*. More in general, we may refer to any element of S as a *simplex*. In the following, we denote an abstract simplicial complex as K, and specify the set of simplices S only when required. A *subcomplex* of K is a subset $J \subseteq K$ that is also an abstract simplicial complex. In the following we will refer to abstract simplicial complexes simply as simplicial complexes. We next define an *orientation* on simplices as follows [16].

Definition 1.2. Given a simplicial complex K and a k-simplex $\sigma = \{v_0, v_1, \ldots, v_k\}$, an orientation of σ is an equivalence class of orderings of the vertices of σ, where two orderings are equivalent if they differ by an even permutation. An oriented simplex is denoted as $\sigma = [v_0, v_1, \ldots, v_k]$.

Using this definition, an *oriented simplicial complex* is an abstract simplicial complex whose simplices have been given an orientation. Note that a simplex can have only two orientations; for instance, in a 1-simplex (i.e., a segment connecting two vertices) the orientation represents the direction in which the simplex points.

Definition 1.3. Consider an abstract simplicial complex K and a field F. The *k-chain group* of K with coefficients in F, denoted $C_k(K, F)$ or simply C_k, is the F-vector space with basis the oriented k-simplices of K, with the relation $[\sigma] = -[\tau]$ if $\sigma = \tau$ and they have different orientation. In other words, C_k is composed of all the finite sums of the form

$\sum_i n_i[\sigma_i]$, where σ_i are k-simplices of K and $n_i \in F$. We refer to the elements of C_k as *k-chains*.

From now on, we assume that simplicial complexes are oriented, and thus we do not use a specific notation for representing the orientation —oriented simplices are still denoted as σ and oriented simplicial complexes are still denoted as K.

Definition 1.4. Consider an abstract simplicial complex K and a field F. Let C_k be the k-chain group of K with coefficients in F. The *k-th boundary operator* ∂_k is the map $\partial_k \colon C_k \to C_{k-1}$ defined on the generators of C_k (i.e., the oriented k-simplices of K) as

$$\partial_k([v_0, v_1, \ldots, v_k]) = \sum_i (-1)^i [v_0, v_1, \ldots, \hat{v_i}, \ldots, v_k], \tag{1.1}$$

where $\hat{v_i}$ means that v_i is deleted from the simplex. The map ∂_k is extended linearly to all C_k, making it a homomorphism of F-vector spaces. The elements of $\mathrm{Ker}(\partial_k)$ are called *k-cycles*.

In other words, the boundary operator maps a k-simplex to the alternating sum of faces that make up its boundary. In addition, ∂_k connects the chain groups into a unified structure, the *chain complex* C_*:

$$\ldots \to C_{k+1} \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \to \ldots. \tag{1.2}$$

Lemma 1.5. *For any abstract simplicial complex K and $k \in \mathbb{N}$, we have $\partial_k \circ \partial_{k+1} = 0$.*

Proof. Let $\sigma = [v_0, v_1, \ldots, v_{k+1}] \in C_{k+1}$ be an oriented simplex. Then,

$$(\partial_k \circ \partial_{k+1})(\sigma) = \partial_k \sum_i (-1)^i [v_0, \ldots, \hat{v_i}, \ldots, v_{k+1}] = \sum_i (-1)^i \partial_k [v_0, \ldots, \hat{v_i}, \ldots, v_{k+1}] \tag{1.3}$$

where the last equality comes from the linearity of ∂_k. In addition, we have:

$$\begin{aligned} \partial_k([v_0, \ldots, \hat{v_i}, \ldots, v_{k+1}]) = &\sum_{j<i} (-1)^j [v_0, \ldots \hat{v_j}, \ldots, \hat{v_i}, \ldots, v_{k+1}] \\ &+ \sum_{j>i} (-1)^{j-1} [v_0, \ldots \hat{v_i}, \ldots, \hat{v_j}, \ldots, v_{k+1}] \end{aligned} \tag{1.4}$$

where the separation into two summations comes from the fact that the sign corresponding to the deleted term in the n-th position ($n = 0, 1, 2 \ldots$) is $(-1)^n$, and when $j > i$, the position of the deleted term v_j is not j, but $j-1$ (since the term v_i appearing before v_j has been deleted). Replacing equation (1.4) into equation (1.3) we obtain:

$$\begin{aligned} (\partial_k \circ \partial_{k+1})(\sigma) = &\sum_{i,j<i} (-1)^{i+j} [v_0, \ldots \hat{v_j}, \ldots, \hat{v_i}, \ldots, v_{k+1}] \\ &+ \sum_{i,j>i} (-1)^{i+j-1} [v_0, \ldots \hat{v_i}, \ldots, \hat{v_j}, \ldots, v_{k+1}] \end{aligned} \tag{1.5}$$

and the terms from each summation cancel each other out —the terms with v_i and v_j deleted, with $j < i$ without loss of generality, appear once in each summation with opposite signs, thus canceling each other out. Therefore, $(\partial_k \circ \partial_{k+1})(\sigma) = 0$, and from the linearity of ∂_k and ∂_{k+1}, we have $(\partial_k \circ \partial_{k+1})(c) = 0$ for all $c \in C_{k+1}$. □

In other words, Lemma 1.5 states that *the boundary of a boundary is always* 0. Using this result, we can define homology groups as follows.

Definition 1.6. Consider an abstract simplicial complex K and a field F. The *k-th homology group* of K with F-coefficients, denoted by H_k, is the quotient group

$$H_k = \mathrm{Ker}(\partial_k)/\mathrm{Im}(\partial_{k+1}). \tag{1.6}$$

The classes of H_k are made of homologous cycles and we refer to them as *k-homology classes*.

Homology groups are well-defined due to the fact that the inclusion $\mathrm{Im}(\partial_{k+1}) \subseteq \mathrm{Ker}(\partial_k)$ always holds: if $\sigma \in \mathrm{Im}(\partial_{k+1})$, then $\sigma = \partial_{k+1}(\tau)$ for some $\tau \in C_{k+1}$, and from Lemma 1.5, $\partial_k(\partial_{k+1}(\tau)) = 0$, i.e., $\partial_k(\sigma) = 0$, and therefore $\sigma \in \mathrm{Ker}(\partial_k)$. In addition, since F is a field, H_k is a vector space fully described by its dimension.

We briefly explain the meaning of Definition 1.6. When we take the quotient with respect to $\mathrm{Im}(\partial_{k+1})$, we send all k-cycles that *are also the image of some k-chain via ∂_{k+1}* to the zero class. Now, recall that the image of a $(k+1)$-chain via the boundary operator is its boundary. Hence, H_k is the group formed by *the k-cycles that are not the boundary of a $(k+1)$-chain*. A particular property illustrating its meaning in the case $k = 0$ is given by the following proposition. Note that we use the concept of *connected component* and *connection* between vertices in an abstract simplicial complex. Simply put, two vertices v, w are *connected* if there is a sequence of 1-simplices of the form $[v, v_1], [v_1, v_2], \ldots, [v_{l-1}, v_l], [v_l, w]$, and a *connected component* is a subcomplex $S \subseteq K$ of connected vertices such that if $v \notin S$, then no vertex of S is connected to v.

Proposition 1.7. *Consider a finite abstract simplicial complex K, and let F be a field. Let C_0 be the 0-chain group of K and H_0 the 0-th homology group of K, with coefficients in F. Denote by $S_1, \ldots, S_n$ the connected components of K, and let $\{p_i\}_{i=1}^n$ be a collection of n vertices where $p_i \in S_i$ for $i = 1, \ldots, n$. Then, the homology classes of $p_1, \ldots, p_n$ form a basis of H_0.*

Proof. First, we note that C_0, which is in particular an F-vector space, decomposes as the direct sum of the 0-chain groups of the connected components $S_1, \ldots, S_n$ of K, i.e., $C_0 = C_0(S_1) \oplus \ldots \oplus C_0(S_n)$. This comes directly from the fact that a basis of C_0 is $\{p_0^1, \ldots, p_{k_1}^1, \ldots, p_0^n, \ldots, p_{k_n}^n\}$, where $\{p_0^i, \ldots, p_{k_i}^i\}$ is the set of vertices of S_i for $i = 1, \ldots, n$, so a basis of C_0 can be expressed as a union of the bases $\{p_0^i, \ldots, p_{k_i}^i\}$. Similarly, the 1-chain group of K is a direct sum of the 1-chain groups of $S_1, \ldots, S_n$, i.e., $C_1 = C_1(S_1) \oplus \ldots \oplus C_1(S_n)$. Furthermore, the boundary operator ∂_1 maps each $C_1(S_i)$ onto $C_0(S_i)$, so from the definition of the k-th homology groups, the 0-th homology group H_0 is the direct sum of the 0-th homology groups of each S_i: $H_0 = H_0(S_1) \oplus \ldots \oplus H_0(S_n)$ —where we denote by $H_0(S_i)$ the 0-th homology group of S_i. Hence, in order to find a basis of H_0, we need to find a basis of each $H_0(S_i)$. We will now see that for each connected simplicial complex S_i, $H_0(S_i)$ is generated by an arbitrary vertex of S_i. First, note that $\mathrm{Ker}(\partial_0)$ is an F-vector space with basis the vertices of S_i, since the image of a vertex by ∂_0 is 0. In addition, all the vertices of S_i are in the same homology class (recall the definition of the 0-th homology group; $H_0(S_1) = \mathrm{Ker}(\partial_0)/\mathrm{Im}(\partial_1)$). To see this, let p be any vertex in S_i and q any other vertex in S_i. Then, since S_i is connected, there exists a sequence of 1-simplices of the form $[p, p_1], [p_1, p_2], \ldots, [p_{l-1}, p_l], [p_l, q]$ in S_i, and in particular, $[p, p_1] + [p_1, p_2] + \ldots + [p_l, q]$ is a

1-chain, with $\partial_1([p, p_1] + [p_1, p_2] + \ldots + [p_l, q]) = q - p$, so p and q are in the same homology class; $\bar{p} = \bar{q}$ for any q in S_i. As a consequence, $H_0(S_i)$ is generated by p. Now, if we take an arbitrary point p_i in S_i for each $i = 1, \ldots, n$, then a basis of $H_0(S_i)$ is the homology class $\bar{p_i}$ of p_i for each i, and since $H_0 = H_0(S_1) \oplus \ldots \oplus H_0(S_n)$, it follows that $\{\bar{p_1}, \ldots, \bar{p_n}\}$ is a basis of H_0. □

As a consequence, the dimension of H_0 is equal to the number of connected components of K. For higher homology degrees, there is also a relation between the "shape" of the simplicial complex and the dimension of the homology spaces. We illustrate this relation in the case of a simplicial complex K where the vertices of K are points $v_0, \ldots, v_n$ in $\mathbb{R}^d$, and $\mathbb{R}^d$ is equipped with the Euclidean metric. We also impose that the vertices are affinely independent, i.e., the vectors issuing from an arbitrarily chosen vertex to the rest of the points are linearly independent —for instance, we can assume $d \geq n + 1$ and let v_i be the unit point of the i-th coordinate axis of $\mathbb{R}^d$. In this situation, we can use an alternative representation of an abstract simplicial complex: a *geometric simplicial complex*. The geometric simplicial complex K_G associated with K is a subspace of $\mathbb{R}^d$ formed by the convex hulls of all sets of points $\{v_{i_0}, v_{i_1}, \ldots, v_{i_k}\}$ where $[v_{i_0}, v_{i_1}, \ldots, v_{i_k}]$ is any k-simplex of K. Recalling that the convex hull of a set of points is the smallest convex set containing it, we can deduce that for a 0-simplex, it is simply the vertex; for a 1-simplex, it is the line segment connecting the two points; for a 2-simplex it forms a triangle with vertices the points of the 2-simplex, and so on. In other words, the set K_G is a union of vertices, line segments, triangles, tetrahedra and so on, each of them representing a simplex of K. In addition, we can define the *underlying topological space* $|K_G|$ of K_G, which is the set K_G with the topology induced by the union of the convex hulls forming it, each being equipped with the Euclidean topology. Assume now that K admits a representation as a geometric simplicial complex K_G. Then, the dimension of H_1 counts the number of linearly independent loops in K_G, and, more in general, the dimension of each k-th homology group H_k detects k-dimensional geometric features (k-dimensional "holes") of K_G [16]. This inspires the generalized terminology *k-hole* for referring to a non-zero class of the k-th homology group of an abstract simplicial complex.

1.2 Point clouds and filtrations

We define a *point cloud* as an ordered finite set $\{p_i\}_{i=1}^n$ where the elements $p_1, \ldots, p_n$ are points in a metric space. Note that, while point clouds are typically defined as unordered sets, for the objectives of our forthcoming discussion on differentiability in the next chapter, it is advantageous to consider point clouds in an ordered format. The goal of this section is to define a method to infer a description of the "shape" of point clouds. In order to do so, the idea is to build a simplicial complex with vertices the elements of the point cloud, and only allow the existence of simplices when certain conditions related to the point cloud are met. One construction of this form is the Čech complex, which exhibits several beneficial properties as highlighted in [16] and [18]. However, its computation can involve checking the intersection of balls in high-dimensional spaces, which is computationally time-prohibitive. As a consequence, we employ a simpler construction: the Rips complex. Note that there is still a close relation between the Rips and the Čech constructions [16].

Definition 1.8. Let M be a metric space with metric d, and let $P = \{p_i\}_{i=1}^n \subseteq M$ be a point cloud in M. Given $\epsilon > 0$, the *Rips complex* $K_\epsilon(P)$ is the abstract simplicial complex with vertices the points of P, and given any ordered subset $\{i_0, \dots, i_k\} \subseteq \{1, \dots, n\}$, the k-simplex $[p_{i_0}, \dots, p_{i_k}]$ exists in $K_\epsilon(P)$ if $d(p_x, p_y) \leq \epsilon$ for all $x, y \in \{i_0, \dots, i_k\}$.

Definition 1.9. A *filtration* or *filtered complex* is a nested family of simplicial complexes $(K_\epsilon)_{\epsilon \in I}$, where $I \subseteq \mathbb{R}$ and for all $\epsilon, \epsilon' \in I$, if $\epsilon \leq \epsilon'$ then $K_\epsilon \subseteq K_{\epsilon'}$.

Definition 1.10. Let M be a metric space with metric d, and let $P = \{p_i\}_{i=1}^n$ be a point cloud in M. The *Rips filtration* of P is the filtration $(K_\epsilon)_{\epsilon \in [0;+\infty)}$ where each K_ϵ is the Rips complex obtained from P with scale value ϵ.

In the remainder of this work we focus on the Rips filtration due to its convenient computational and theoretical properties. For instance, it only depends on the pairwise distances of the points in the point cloud, making its computation fast and efficient. In addition, the experimental work conducted in Chapter 3 relies on the computation of persistence diagrams using Rips filtrations, making this type of filtration of special interest.

It is important to note that there is an alternative way of describing Rips filtrations, which is preferable for the work of Chapter 2, relying on the use of filter functions. A filter function is a map $f \colon K \to \mathbb{R}$ were K is a simplicial complex (and its elements are its simplices), such that for any $\sigma, \tau \in K$, if $\sigma \subseteq \tau$, then $f(\sigma) \leq f(\tau)$. Given a simplicial complex K, we denote the space of all filter functions with support K by $\mathbb{R}^K$.

Definition 1.11. Consider a metric space M with metric d, and let $P = \{p_i\}_{i=1}^n$ be a point cloud in M. Let $K = 2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ be the total complex. The *Rips filter function* of P is defined as

$$\begin{aligned} f_P \colon K &\to \mathbb{R} \\ \sigma &\mapsto \max_{i,j \in \sigma} \{d(p_i, p_j)\}. \end{aligned} \tag{1.7}$$

Note that $2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ is the set of all possible sets of the form $\{a_1, a_2, \dots, a_n\}$ where each a_i is either 0 or 1. Equivalently, each element of $2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ can be associated with the simplex formed by the points $\{p_i\}_{a_i \neq 0}$ of P, so K represents all possible simplices that can be formed from points of P.

From Definition 1.11, for any $\sigma = [i_0, \dots, i_k] \in K$, $f_P(\sigma)$ is the scale value corresponding to the appearance of $[p_{i_0}, \dots, p_{i_k}]$ in the Rips filtration of P. In fact, the Rips filter function of P completely determines its Rips filtration: the simplicial complex K_ϵ of the Rips filtration of P is determined by $f_P^{-1}([0; \epsilon])$. Furthermore, there is a key property of Rips filtrations that has to be highlighted. Given a point cloud of n points, the number of simplices contained in the total complex $2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ is finite. Therefore, the Rips filter function of P only outputs a finite number N of strictly positive values $0 = \epsilon_0 < \epsilon_1 < \ldots < \epsilon_N$. As a consequence, we have a relation between complexes of the filtration of the form:

$$\begin{aligned} K_0 &= K_\epsilon \text{ for all } \epsilon \in [0; \epsilon_1);\ K_0 \subsetneq K_{\epsilon_1}, \\ K_{\epsilon_1} &= K_\epsilon \text{ for all } \epsilon \in [\epsilon_1; \epsilon_2);\ K_{\epsilon_1} \subsetneq K_{\epsilon_2}, \\ &\dots \\ K_{\epsilon_{N-1}} &= K_\epsilon \text{ for all } \epsilon \in [\epsilon_{N-1};\ \epsilon_N);\ K_{\epsilon_{N-1}} \subsetneq K_{\epsilon_N}, \\ K_{\epsilon_N} &= K_\epsilon \text{ for all } \epsilon \geq \epsilon_N. \end{aligned} \tag{1.8}$$

Therefore, the information encoded by the filtration can be reduced to the $N+1$ complexes $K_0, K_{\epsilon_1}, \ldots, K_{\epsilon_N}$. In other words, from the Rips filter function we obtain a finite filtration $(K^i)_{i=0}^N$ where each K^i corresponds to a scale value $0, \epsilon_1, \ldots, \epsilon_N$. Due to their underlying meaning, we may also refer to the scale values corresponding to the appearance of new simplices as *times of appearance*.

In addition, if the point cloud is embedded in some Euclidean space $\mathbb{R}^d$, we can visually interpret the filtration. To do so, we make a correspondence between vertices $i \in K$ and the points $\{p_1, \ldots, p_n\}$ of P. In particular, we associate the vertex $\{i\} \in K$, where $i \in \{1, \ldots, n\}$, with the point p_i of P. Similarly, we associate each simplex $[i_0, \ldots, i_k]$ with the simplex made of the corresponding points of P, $[p_{i_0}, \ldots, p_{i_k}]$. As a consequence, each complex K_{ϵ_i} of (1.8) corresponds to a complex $\tilde{K}_{\epsilon_i}$ whose vertices are points of P. Furthermore, we consider the geometric simplicial complexes $\tilde{K}_{G,\epsilon_i} \subseteq \mathbb{R}^d$ associated with each $\tilde{K}_{\epsilon_i}$, only allowing the existence of simplices of dimension equal to or lower than d. This yields a sequence of nested spaces $\tilde{K}_G^0 \subseteq \tilde{K}_G^1 \subseteq \ldots \subseteq \tilde{K}_G^N \subseteq \mathbb{R}^d$. This sequence begins with the point cloud and, as the filtration index increases, the convex hulls of new simplices are added to the geometric simplicial complexes, and the geometric simplicial complexes undergo changes related to the merging of connected components, the birth and death of loops, of voids, and of higher-dimensional holes. One might wonder how we can effectively trace the emergence and disappearance of these features within the filtration. The answer lies in *persistent homology*.

1.3 Persistent homology

For convenience, in the remaining of this chapter we assume the filtration is of the form $(K^i)_{i\in\mathbb{N}}$ (also denoted as $(K^i)_{i\geq 0}$). In particular, finite filtrations $K^0 \subseteq K^1 \subseteq \ldots \subseteq K^n$ can be extended to a filtration of that form by setting $K^j \coloneqq K^n$ for all $j \geq n$.

Definition 1.12. Let $(K^i)_{i\geq 0}$ be a filtration, where each K^i has boundary operators ∂_k^i, and groups C_k^i, $Z_k^i = \mathrm{Ker}(\partial_k^i)$, $B_k^i = \mathrm{Im}(\partial_{k+1}^i)$, and H_k^i for all $k \in \mathbb{N}$. Let $i, p, k \in \mathbb{N}$. The *p-persistent k-th homology group of K^i* is defined as

$$H_k^{i,p} = Z_k^i / (B_k^{i+p} \cap Z_k^i). \tag{1.9}$$

The *persistent homology of degree k*, or *k-persistent homology*, is the collection of all the groups $(H_k^{i,p})_{i,p\in\mathbb{N}}$. We refer to groups of the form (1.9) as *persistent homology groups*.

In the following, when the homology degree k is clear we may refer to the persistent homology of degree k simply as *persistent homology*.

To prove that the persistent homology groups are well-defined, we proceed as follows. From their definitions, Z_k^i is a subgroup of C_k^i and B_k^{i+p} is a subgroup of C_k^{i+p}. In addition, from the inclusions $K^0 \subseteq K^1 \subseteq \ldots$, we have the inclusions $C_k^0 \subseteq C_k^1 \subseteq \ldots$ for any $k \geq 0$, so both Z_k^i and B_k^{i+p} are subgroups of C_k^{i+p}, hence their intersection is a subgroup of Z_k^i, making the quotient of equation (1.9) well-defined. In addition, since the chain groups are vector spaces, it follows that the persistent homology groups are also vector spaces.

We now briefly illustrate the meaning behind Definition 1.12. Recall that Z_k^i are the k-cycles in K^i, and B_k^{i+p} are the boundaries of $(k+1)$-cycles in K^{i+p}. The intersection of Z_k^i and B_k^{i+p} is thus the group of k-cycles from K^i that are boundaries of $(k+1)$-cycles in

K^{i+p}. Therefore, when taking the quotient of Z_k^i with respect to the denominator, we take the non-zero elements of Z_k^i, i.e., the k-cycles that are not boundaries of $(k+1)$-cycles in K^i, and send them to the zero class *if they have become boundaries in* K^{i+p}. Hence, the non-zero classes of H_k^{i+p} are the k-holes of K^i that *persist* as k-holes until K^{i+p}. In other words, the persistent homology groups track the persistence of k-holes along the filtration.

For the following work, we need to express persistent homology in a compact and unified way. In order to do so, we unify persistent homology into a single algebraic structure, namely *persistence modules*.

1.4 Persistence modules

Definition 1.13. A *persistence module* $\mathcal{M}$ is a family of vector spaces $\{M^i\}_{i\in\mathbb{N}}$ equipped with linear maps $\varphi^i \colon M^i \to M^{i+1}$.

Assume we have a filtration $(K^i)_{i\in\mathbb{N}}$, a homology degree k, and a field F. Each complex K^i has its respective chain group C_k^i. Consider the inclusion maps $i_j \colon C_k^j \to C_k^{j+1}$. We then have the following relation between chain groups:

$$C_k^0 \xrightarrow{i_0} C_k^1 \xrightarrow{i_1} C_k^2 \longrightarrow \dots \tag{1.10}$$

Applying k-th homology to this sequence, we obtain a sequence of k-th homology vector spaces. These spaces are connected by linear maps $\varphi^j \colon H_k^j \to H_k^{j+1}$, where each φ^j maps each homology class to the one that contains it. In other words, we have:

$$H_k^0 \xrightarrow{\varphi^0} H_k^1 \xrightarrow{\varphi^1} H_k^2 \longrightarrow \dots \tag{1.11}$$

Hence, the family $\{H_k^i, \varphi^i\}_{i\geq 0}$ is a persistence module, which captures the k-th homology groups of the filtration into a single algebraic structure. In addition, a particular type of persistence modules, given by the following definition, exhibit useful properties allowing a simple and more meaningful description of their structure.

Definition 1.14. A persistence module $\{M^i, \varphi^i\}_{i\geq 0}$ is of *finite type* if each component vector space is of finite dimension, and if the maps φ^i are isomorphisms for all $i \geq m$ for some $m \in \mathbb{N}$.

In particular, Rips filtrations of finite point clouds are finite and composed of finite complexes, and using coefficients in a field, the associated homology groups are of finite dimension. This directly implies that the resulting persistence modules, for any homology degree, are of finite type. We now develop an alternative representation of persistence modules of finite type. To achieve this, we adopt the approach employed in [13]. We begin by defining a map from persistence modules to polynomial rings as follows. First, assume we have a persistence module $\mathcal{M} = \{M^i, \varphi^i\}_{i\geq 0}$ with coefficients in a field F. We denote by $F[t]$ the commutative ring formed by the polynomials with coefficients in F, i.e., functions of the form $f(t) = \sum_{i=0}^{n} a_i t^i$, where $a_i \in F$ and t is a variable. We equip $F[t]$ with the standard grading, using the sum decomposition $F[t] \cong \bigoplus_{i=0}^{+\infty} F t^i$ as vector spaces. We then define a *graded module over the graded ring* $F[t]$ as

$$\alpha(\mathcal{M}) = \bigoplus_{i\geq 0} M^i, \tag{1.12}$$

where its F-module structure comes from the sum of the structures over each individual component, and the action of t is given by

$$t \cdot (m^0, m^1, m^2, \ldots) \mapsto (0, \varphi^0(m^0), \varphi^1(m^1), \ldots). \tag{1.13}$$

In other words, t shifts the F-modules of $\mathcal{M}$ up in the gradation. We briefly illustrate the meaning of this transformation in the case of persistence modules arising from filtrations. Assume we have a filtration $(K^i)_{i\geq 0}$ and the persistence module $\mathcal{M} = \{M^i, \varphi^i\}_{i\geq 0}$ over a field F obtained from the k-th homology of the filtration. Recall that each M^i is the k-th homology group of the complex K^i. Assume the persistence module is of finite type; then α maps $\mathcal{M}$ to a finitely generated non-negatively graded module over $F[t]$ denoted by $\alpha(\mathcal{M})$. Take now an arbitrary k-hole with birth in K^b for some $b \geq 0$, and (possible) death in K^d for some $d > b$ (if it does not die, we set $d = +\infty$). The homology class representing this k-hole appears for the first time in M^b, and is mapped to a non-zero class in the subsequent F-modules $M^{b+1}, \ldots, M^{d-1}$, followed by being mapped to the zero class in M^d. Alternatively, the first component of $\alpha(\mathcal{M})$ containing the non-zero homology class representing the selected k-hole is the b-th component. Applying t on $\alpha(\mathcal{M})$ repeatedly, this class is shifted to a non-zero class on the next component of the gradation until it reaches the d-th component, where it becomes the zero class. Therefore, $\alpha(\mathcal{M})$ captures the birth and death times of the selected k-hole. Hence, the map α associates the persistence module to a single object containing the birth and death times of the k-holes in the underlying filtration.

Now, using the fact that the graded ring $F[t]$ is a principal ideal domain (PID), we can use the following result from [13] to describe the structure of $F[t]$-modules.

Theorem 1.15. *Any finitely generated graded module M over a graded PID D decomposes uniquely into the form*

$$\left(\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} D\right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{\gamma_j} D/d_j D\right), \tag{1.14}$$

where Σ^α is an α-shift upward in the grading, and $d_j \in D$ are homogeneous elements such that $d_j \mid d_{j+1}$, and α_i, γ_j are nonnegative integers.

Although the result was given in [13] and it is a standard result in the case of non-graded PIDs, adjustments have to be made for its proof in the case of a graded PID. A detailed proof of the theorem was recently provided in [19].

Since the only graded ideals of $F[t]$ are of the form (t^n), for $n \in \mathbb{N}$, Theorem 1.15 gives a decomposition of graded modules over $F[t]$:

$$\left(\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} F[t]\right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{\gamma_j} F[t]/(t^{n_j})\right), \tag{1.15}$$

where the n_j are positive integers. Furthermore, the graded $F[t]$-module decomposes into two structures: the *free* portion (on the left), which has a form of a free module, and the *torsion* (on the right), which is a vector space whose dimension is finite. Our next step is to represent this decomposition with simpler and more meaningful objects. To do so, we define a $\mathcal{P}$-*interval* as an ordered pair (i, j) with $0 \leq i < j \in \mathbb{Z} \cup \{\infty\}$, and associate to any

set $\mathcal{S}$ of $\mathcal{P}$-intervals a graded $F[t]$-module, in the following way. Given a $\mathcal{P}$-interval (i,j), we define $Q(i,j) = \Sigma^i F[t]/(t^{j-i})$ —if $j = +\infty$, then $Q(i,+\infty) = \Sigma^i F[t]$. Given a set of intervals $\mathcal{S} = \{(i_1,j_1),\ldots,(i_n,j_n)\}$ we define

$$Q(\mathcal{S}) \coloneqq \bigoplus_{l=1}^{n} Q(i_l,j_l). \tag{1.16}$$

The correspondence $\mathcal{S} \mapsto Q(\mathcal{S})$ induces a bjection between the finite sets of $\mathcal{P}$-intervals and the finitely generated graded modules over the graded ring $F[t]$. As a consequence, any persistence module of finite type is represented by a unique finite set of $\mathcal{P}$-intervals.

We summarize the results we have obtained so far in this chapter to our scenario of interest: analyzing a point cloud embedded in a metric space. Given a point cloud P in a metric space, a homology degree $k \geq 0$ and a field F, we first generate the Rips filter function of the point cloud, f_P, yielding the family of simplicial complexes $(K^i)_{i\in\mathbb{N}}$ —where, after some $N \in \mathbb{N}$, all complexes are equal. Each of these complexes has an associated k-th homology group, which is a vector space over F. In order to connect these spaces using compatible bases, we construct the persistence module, which is of finite type due to the nature of the filtration. The map α then connects the persistence module to a finitely generated non-negatively graded module over $F[t]$, which, roughly speaking, represents a direct sum of the homology vector spaces. From Theorem 1.15, there exists a basis for this structure including compatible bases for all the vector spaces, and the bijection induced by Q ensures that this graded module is represented by a multi-set of $\mathcal{P}$-intervals, denoted $\mathcal{I}$, where each $(i,j) \in \mathcal{I}$ corresponds to a basis element for the homology vector spaces $H_k^i,\ldots,H_k^{j-1}$. Equivalently, each $\mathcal{P}$-interval (i,j) corresponds to a k-hole with birth in K^i and death in K^j. In addition, we have seen that each persistent homology group $H_k^{i,p}$ is a vector space fully described by its dimension, so-called the *p-persistent k-th Betti number* and denoted $\beta_k^{i,p}$. The following lemma shows that the representation of the persistent homology based on $\mathcal{P}$-intervals is indeed sufficient for determining the dimensions of all the k-persistent homology groups.

Lemma 1.16. *Consider the multi-set of $\mathcal{P}$-intervals denoted by $\mathcal{I}$ representing a persistence module associated with a k-persistent homology $(H_k^{l,p})_{l,p\geq 0}$. For any $(i,j) \in \mathcal{I}$, denote by $\mathcal{T}(i,j)$ the triangle defined as $\{(x,y) \in \mathbb{R}^2 \mid x \geq i,\ y \geq 0,\ x+y < j\}$. Then, the rank $\beta_k^{l,p}$ of $H_k^{l,p}$ is the number of triangles $\mathcal{T}$ containing the point (l,p).*

Proof. Consider a persistent homology group $H_k^{l,p}$. Let $(i,j) \in \mathcal{I}$, which represents a basis element $\bar{e}$ for $H_i,\ldots,H_{j-1}$. If $(l,p) \in \mathcal{T}(i,j)$, then $i \leq l \leq l+p < j$, so $\bar{e}$ persists as a basis element from H_k^l to H_k^{l+p} (both included). If $(l,p) \notin \mathcal{T}(i,j)$, then either $\bar{e}$ is not yet a basis element in H_k^l (case $l > i$), or $\bar{e}$ has become a boundary at H_k^{l+p} (case $l+p \geq j$), so $\bar{e}$ is not a basis element of $H_k^{l,p}$. Hence a triangle $\mathcal{T}(i,j)$ containing (l,p) is equivalent to the basis element associated to (i,j) being a basis element of $H_k^{l,p}$. □

Therefore, the multi-set of $\mathcal{P}$-intervals representing the persistence module completely determines the persistent homology of the filtration (in the case that the persistence module is of finite type and the ground ring is a field). In other words, computing the persistent homology is equivalent to computing the multi-set of $\mathcal{P}$-intervals describing it.

1.5 Persistence diagrams and barcode space

From the previous section, a k-persistent homology $\mathcal{H}$ has an associated persistence module $\mathcal{M}$, which can be represented with a multi-set of $\mathcal{P}$-intervals $\mathcal{I}$. We now provide an alternative representation of the persistence module (and thus of the persistent homology). First, for each $\mathcal{P}$-interval (i, j) of $\mathcal{I}$, we replace its endpoints $i < j \in \mathbb{N} \cup \{+\infty\}$ with the respective times in the filtration $\epsilon(i) < \epsilon(j) \in \bar{\mathbb{R}}$, where $\bar{\mathbb{R}} \coloneqq \mathbb{R} \cup \{+\infty\}$. The persistence module is thus mapped to a finite multi-set of intervals of the form $(b, d) \subseteq \bar{\mathbb{R}}$, which we call the *barcode* of $\mathcal{M}$.

We can also view the barcode as a finite multi-set of points in $\mathbb{R} \times \bar{\mathbb{R}}$: each interval $J \in \mathcal{J}$ is represented by the point $(\inf J, \sup J)$. Adding to this set the diagonal $\Delta = \{(b, b) \mid b \in \mathbb{R}\}$ with infinite multiplicity, noted Δ^∞, we obtain what we call the *persistence diagram* of $\mathcal{M}$ (and of $\mathcal{H}$). This representation is more suitable for the work we will pursue in the following chapters. However, a question that arises is the following: if we slightly perturb the point cloud, how much will the barcode (or persistence diagram) change? Is the persistence diagram stable under small perturbations of the point cloud? To answer, we need to define a concept of distance between persistence diagrams. We first generalize the notion of persistence diagram to an object independent of concepts of persistent homology and persistence modules, and then define a metric space that contains all observable persistence diagrams.

Definition 1.17. A persistence diagram is the union $B \cup \Delta^\infty \subseteq \mathbb{R} \times \bar{\mathbb{R}}$, where B is a finite multi-set of elements in $\mathbb{R} \times \bar{\mathbb{R}}$. The space Bar (so-called barcode space) is the space of all persistence diagrams.

From now on, we will use the terminology *persistence diagram*, *persistence barcode* or *barcode* for referring to any element of Bar. We now define a distance in Bar.

Definition 1.18. Let $D, D' \in \text{Bar}$ be two barcodes. A *matching* between D and D' is a bijection between the two multi-sets $\gamma\colon D \to D'$. The *cost* of a matching is defined as

$$c(\gamma) \coloneqq \sup_{x \in D} \|x - \gamma(x)\|_\infty, \tag{1.17}$$

where $\|\cdot\|_\infty$ is the supremum norm, given by $\|(a, b)\|_\infty = \max(|a|, |b|)$. We denote by $\Gamma(D, D')$ the set of all matchings between D and D'. The *bottleneck distance* between D and D' is then

$$d_\infty(D, D') \coloneqq \inf_{\gamma \in \Gamma(D, D')} c(\gamma). \tag{1.18}$$

The bottleneck distance is a metric in Bar. Finally, we equip the barcode space with the *bottleneck topology*, i.e., the topology induced by the bottleneck distance —the open sets of the bottleneck topology in Bar are generated by the open balls of Bar with respect to the bottleneck distance.

1.6 Stability

We now aim to establish a bound on the distance between persistence diagrams in relation to the distance between point clouds embedded in a Euclidean space. This is crucial for

practical applications: in Chapters 2 and 3, we explore the use of persistence diagrams for comparing a point cloud of true images with a point cloud of generated images. Our goal is to minimize the distance between persistence diagrams to produce data point clouds with a similar "shape" to the true data. Hence, we have to prove that persistence diagrams are *stable* with respect to the data; in other words, we aim to show that small perturbations in the data result in small perturbations in the persistence diagrams. If this stability holds, similar input and output data will imply similar input and output persistence diagrams, thereby making it possible to minimize both the distance between persistence diagrams and the distance between point clouds, making topological regularization via persistence diagrams a possible way to enhance the training process of generative models. Furthermore, in order to better understand the relation between the distance of point clouds and the distance between persistence diagrams, we aim to find a specific bound of this stability.

To answer these questions, we will first recall an important property of persistence diagrams, the stability theorem given by Cohen et al. in [14]. We use the notion of p-persistent homology operator Dgm_p, which, given an abstract simplicial complex K, maps a filter function $f\colon K \to \mathbb{R}$ to the persistence diagram of degree p induced by f. The stability theorem from [14] can be re-stated as follows.

Theorem 1.19. *Let K be a total abstract simplicial complex, and let $f, g\colon K \to \mathbb{R}$ be two filter functions. We denote by $\mathrm{Dgm}_k(f)$ and $\mathrm{Dgm}_k(g)$ the persistence diagrams of degree k induced by f and g, respectively. Then, we have:*

$$d_\infty(\mathrm{Dgm}_k(f), \mathrm{Dgm}_k(g)) \leq \|f - g\|_\infty, \tag{1.19}$$

where d_∞ is the bottleneck distance, and $\|f - g\|_\infty = \sup_{x \in K} |f(x) - g(x)|$.

The proof can be found in [14]. In order to translate this result into the stability of persistence diagrams (obtained via Rips filtrations) with respect to perturbations of the point cloud, we proceed as follows. First, fix any homology degree $k \in \mathbb{N}$, and assume we work with point clouds consisting of n points in $\mathbb{R}^d$, for some fixed $d, n \in \mathbb{N} \setminus \{0\}$. We view point clouds as individual points in $\mathbb{R}^{nd}$; if $\{p_i\}_{i=1}^n$ is a point cloud, we represent it as the point $P = (p_1, p_2, \ldots, p_n)$ in $\mathbb{R}^{nd}$. We equip $\mathbb{R}^{nd}$ with the Euclidean norm, which induces a distance between point clouds. Now, let $P \in \mathbb{R}^{nd}$ be any point cloud, and consider a second point cloud resulting from a small perturbation of the points of P. Specifically, we assume there is some $\epsilon > 0$ such that each p_i is perturbed into a new point p_i' satisfying $\|p_i - p_i'\|_2 < \epsilon$. We denote the perturbed point cloud by $P' \in \mathbb{R}^{nd}$. We will now prove that the bottleneck distance between their k-persistence diagrams is bounded by 2ϵ.

We denote the Rips filter functions of P and P' by f and f', respectively. We aim to show that $\|f - f'\|_\infty < 2\epsilon$. Let σ be any simplex of the total complex $2^{\{0,1,\ldots,N\}} \setminus \{\emptyset\}$. We have $f(\sigma) = \|p_i - p_j\|_2$ and $f'(\sigma) = \|p_k' - p_l'\|_2$ for some indices i, j, k, l. In addition, we have

$$\|p_i' - p_j'\|_2 \leq \|p_i' - p_i\|_2 + \|p_i - p_j\|_2 + \|p_j - p_j'\|_2 < \|p_i - p_j\|_2 + 2\epsilon. \tag{1.20}$$

Analogously, we have $\|p_i - p_j\|_2 < \|p_i' - p_j'\|_2 + 2\epsilon$, so

$$\big|f(\sigma) - \|p_i' - p_j'\|_2\big| < 2\epsilon. \tag{1.21}$$

Therefore, if $\{k,l\} = \{i,j\}$ (up to a permutation), then $|f(\sigma) - f'(\sigma)| < 2\epsilon$. Assume now that $\{k,l\} \neq \{i,j\}$. We have the inequality

$$\|p'_k - p'_l\|_2 \leq \|p'_k - p_k\|_2 + \|p_k - p_l\|_2 + \|p_l - p'_l\|_2 < \|p_i - p_j\|_2 + 2\epsilon = f(\sigma) + 2\epsilon. \quad (1.22)$$

Using the fact that $\|p'_i - p'_j\|_2 \leq \|p'_k - p'_l\|_2$ (from the definition of f'; see Definition 1.11), and combining it with equations (1.21) and (1.22), we obtain

$$f(\sigma) - 2\epsilon < \|p'_i - p'_j\|_2 \leq \|p'_k - p'_l\|_2 = f'(\sigma) < f(\sigma) + 2\epsilon. \quad (1.23)$$

Therefore, $|f(\sigma) - f'(\sigma)| < 2\epsilon$.

Applying Theorem 1.19 with $K = 2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ and f and f' as the functions f and g of the theorem, we obtain the bound

$$d_\infty(\mathrm{Dgm}_k(f), \mathrm{Dgm}_k(f')) \leq \|f - f'\|_\infty < 2\epsilon. \quad (1.24)$$

Therefore, if the points of the two point clouds are at a distance smaller than ϵ, then the bottleneck distance between their persistence diagrams is lower than 2ϵ. We can also say that if $\|P - P'\|_2 < \epsilon$, then $d_\infty(\mathrm{Dgm}_k(f), \mathrm{Dgm}_k(f')) < 2\epsilon$. Hence, the map from point clouds in $\mathbb{R}^{nd}$ to persistence diagrams is stable with respect to perturbations of the point cloud. In fact, the map is 1-Lipschitz continuous.

The implications of this result are twofold. First, it ensures that in a generative model learning to produce data points similar to the true dataset, if the distance between data point clouds converges to zero, then the distance between persistence diagrams also converges to zero. Hence, we can implement a regularizer with the goal of minimizing the distance between persistence diagrams, since these two quantities mutually converge. Second, it is important to note that real-life datasets come with some degree of noise, and thus a dataset can include images of say, cats, with varying degrees of noise. But we do not want small amounts of noise to have a considerable effect on the resulting persistence diagrams. Our stability bound ensures that the distance between the persistence diagrams is stable with respect to the degree of noise added (i.e., the perturbation) in the images.

In fact, we can experimentally confirm these two results through the following experiment, with the results presented in Figure B.1. We take a set of 128 random images from the FashionMNIST dataset (consisting of grayscale images of clothes; the dataset is of particular interest since it is used in the experiments in Chapter 3). Then, we apply Gaussian noise at three levels of intensity: using a variance of the Gaussian distribution of 0.1 (left, low noise level), variance of 0.3 (middle, medium noise level), and of 0.7 (right, high noise level). We then compute the bottleneck distance between the noise-free ground truth batch (top right) and each noisy batch. We then repeat the same process for 50 different batches. The results can be seen on the top left of the image: the solid lines correspond to the values of the bottleneck distance between the persistence diagrams of degree 0, while dashed lines correspond to degree 1. Blue lines correspond to low noise ("noise level 1"), green lines to medium noise ("noise level 2"), and red lines to high noise ("noise level 3"). We confirm that more noise correlates with a higher bottleneck distance, both for degree 0 and degree 1 —although the correlation is more pronounced for degree 0 [1]. Hence, we can see that

[1] We have also measured the δ-selective persistent entropy between persistence diagrams (top middle of Figure B.1), another measure of properties of the diagram, which we explain in Chapter 2; also in this case, there is a correlation between noise level and magnitude of the dissimilarity measure.

point clouds that are closer to each other have persistence diagrams that are also at closer distance. We repeat the same experiments in a colored dataset, the CIFAR10 dataset (also widely used for training and evaluating generative models); the results are shown in Figure B.2. Again, we also observe a correlation between distance between point clouds (i.e., noise level) and distance between persistence diagrams. It is important to note, however, that different point clouds may have similar persistence diagrams; we have worked here on showing that similar point clouds have similar persistence diagrams. We are thus slowly bridging the gap between persistent homology and generative modelling. However, a question that arises is: is the transformation from point cloud to persistence diagram differentiable with respect to the coordinates of the point cloud? Or, under what conditions is the bottleneck distance (or other measures of dissimilarity), between two persistence diagrams, differentiable? We address these questions in the next chapter.

1.7 Computation of persistent homology

The first algorithm for computing persistent homology was proposed in [12], and more general and efficient algorithms have been developed since, e.g., [13, 20, 21]. However, the algorithm providing persistence diagrams is not our focus here; in fact, for our purposes it can be considered to be a black box. The reason is that we are not interested in the calculation of persistence diagrams, but rather on the (possible) smoothness of the transformation from point clouds to persistence diagrams, and, as we will see in the next chapter, we can prove results about the differentiability of these maps that are independent of the specific algorithm used.

Chapter 2

Differentiability through barcode space

Sections 2.1 and 2.2 and Subsection 2.3.5 mainly rely on [10], and Sections 2.3 and 2.4, excepted Subsection 2.3.5, are original work.

From now on, we refer to an (n, d)-*point cloud* as an ordered set of n points $\{p_1, \ldots, p_n\}$ in $\mathbb{R}^d$ (for some positive integers n and d). When we work with (n, d)-point clouds for some given n and d, we represent each point cloud as a single point in $\mathbb{R}^{nd}$, and we refer to $\mathbb{R}^{nd}$ as the *space of point clouds*. More precisely, an (n, d)-point cloud $\{p_i\}_{i=1}^n$ is viewed as a point $(p_1, \ldots, p_n) \in \mathbb{R}^{nd}$, and alternatively, a point $(x_1, x_2, \ldots, x_{nd}) \in \mathbb{R}^{nd}$ corresponds to the point cloud where the i-th point is given by the components $x_{d(i-1)+1}, \ldots, x_{di}$ (for $i = 1, \ldots, n$). For convenience, in the following we will simply say "Assume $\mathbb{R}^{nd}$ is the space of point clouds" when we work with (n, d)-point clouds for some pair n, d.

2.1 Framework of differentiability through barcode space

We now introduce the framework we will use for working with the differentiation of maps from and onto the barcode space, i.e., maps of the form $B\colon \mathcal{M} \to \text{Bar}$ and maps of the form $V\colon \text{Bar} \to \mathcal{M}$, where $\mathcal{M}$ is a smooth manifold. Since Bar is not naturally equipped with a differential structure [10], we study the smoothness of maps through the barcode space using a *lift* of the maps to another space. This alternative space is the Euclidean space $\mathbb{R}^x$ for a suitable integer x, where the concepts of differentiability are well-defined. A more detailed exposition of this framework is provided in the remaining of this section.

We begin by defining the concept of *differentiability* of a map $V\colon \mathcal{M} \to \text{Bar}$. The idea is to view each barcode, composed of m bounded intervals and n infinite intervals (i.e., of the form $(a, +\infty)$, $a \in \mathbb{R}$), as points in $\mathbb{R}^{2m+n}$. More specifically, the connection between these two objects is given by a map $Q_{m,n}\colon \mathbb{R}^{2m+n} \to \text{Bar}$, which maps a point in $\mathbb{R}^{2m+n}$ to its corresponding barcode:

$$Q_{m,n}(b_1, d_1, \ldots, b_m, d_m, v_1, \ldots, v_n) = \{(b_i, d_i)\}_{i=1}^m \cup \{(v_i, +\infty)\}_{i=1}^n \cup \Delta^\infty, \tag{2.1}$$

where (b_i, d_i) are the bounded off-diagonal points of the persistence diagram, and $(v_i, +\infty)$ are its unbounded off-diagonal points. Recall that Δ^∞ is the diagonal with infinite multiplicity. Simply put, $Q_{m,n}$ *forgets* about the order of points in $(b_1, d_1, \ldots, b_m, d_m, v_1, \ldots, v_n)$:

points that differ only by permutations of coordinates (b_i, d_i) or of coordinates (v_i) yield the same barcode. An essential property of $Q_{m,n}$ is the following.

Proposition 2.1. *For any $m, n \in \mathbb{N}$, $Q_{m,n}$ is 1-Lipschitz when* Bar *has the bottleneck topology.*

Proof. Denote by d_∞ the bottleneck distance, and take any $P_1, P_2 \in \mathbb{R}^{2m+n}$. We want to see that

$$d_\infty(Q_{m,n}(P_1), Q_{m,n}(P_2)) \leq \|P_1 - P_2\|_2. \tag{2.2}$$

From the defintion of bottleneck distance, we have, for any matching γ between the barcodes $Q_{m,n}(P_1)$ and $Q_{m,n}(P_2)$:

$$d_\infty(Q_{m,n}(P_1), Q_{m,n}(P_2)) \leq c(\gamma), \tag{2.3}$$

where $c(\gamma)$ denotes the cost of γ. In particular, consider the matching that sends the i-th component of P_1 to the i-th component of P_2 (for $i = 1, \ldots, 2m+n$). This matching has a corresponding matching γ in the barcode space between $Q_{m,n}(P_1)$ and $Q_{m,n}(P_2)$ —note that this matching sends the remaining elements in Δ^∞ of P_1 to the corresponding points in P_2. Using the notation $P_1 = (a_1, \ldots, a_{2m+n})$ and $P_2 = (b_1, \ldots, b_{2m+n})$, the cost of γ is given by

$$c(\gamma) = \max_i \|x_i - \gamma(x_i)\|_\infty \leq \max_i |a_i - b_i|, \tag{2.4}$$

where the last equality is given by the fact that $\max_i \|x_i - \gamma(x_i)\|_\infty$ is either equal to $\|(a_i, a_{i+1}) - (b_i, b_{i+1})\|_\infty$ for some $i = 2k+1$; $k < m$, which is equal to $|a_i - b_i|$ or $|a_{i+1} - b_{i+1}|$, or the maximum is obtained at some $i > 2m$, and in that case the cost is $|a_i - b_i|$. Therefore, in all cases (2.4) holds. Hence, we have:

$$d_\infty(Q_{m,n}(P_1), Q_{m,n}(P_2)) \leq \max_i |a_i - b_i| \leq \sqrt{\sum_i (a_i - b_i)^2} = \|P_1 - P_2\|_2. \tag{2.5}$$

□

In order to define the differentiability of maps $\mathcal{M} \to$ Bar, there are two alternative definitions. We first give the two definitions and next formally state their equivalence.

Definition 2.2. Let $B\colon \mathcal{M} \to$ Bar, where $\mathcal{M}$ is a smooth manifold, and let $x \in \mathcal{M}$ and $r \in \mathbb{N} \cup \{\infty\}$. The map B is said to be *r-differentiable* if there exists an open neighborhood U of x, integers $m, n \in \mathbb{N}$, and a map $\tilde{B}\colon U \to \mathbb{R}^{2m+n}$ of class C^r such that $B = Q_{m,n} \circ \tilde{B}$ on U. The differential $d_{x,\tilde{B}}B$ at $x \in \mathcal{M}$ with respect to the lift $\tilde{B}$ is then defined as the differential of $\tilde{B}$ at x:

$$d_{x,\tilde{B}} \equiv d_x\tilde{B}\colon T_x\mathcal{M} \to \mathbb{R}^{2m+n}. \tag{2.6}$$

The idea behind Definition 2.2 is to produce a lift $\tilde{B}$ from $\mathcal{M}$ to $\mathbb{R}^{2m+n}$ and work with the usual differentiability between those two spaces, and then connect $\tilde{B}$ to the barcode space via $Q_{m,n}$. Then, the differential is a map from small perturbations of the point x on the tangent space to $\mathcal{M}$ at x to small variations of the ordered barcodes, seen as vectors in $\mathbb{R}^{2m+n}$.

The second definition of differentiability is more suitable computationally [10] and is based on the concept of point tracking. It is also a better concept to work with for the proof of Proposition 2.16 which we will give later.

Definition 2.3. Let $B\colon \mathcal{M} \to \mathrm{Bar}$, and let $x \in \mathcal{M}$ and $r \in \overline{\mathbb{N}}$. We define a C^r *local coordinate system* for B at x as a collection of maps $\{b_i, d_i\colon U \to \mathbb{R}\}_{i\in I}$ and $\{v_j\colon U \to \mathbb{R}\}_{j\in J}$ defined on an open neighborhood U of x such that the following two conditions hold.

(i) (*Smoothness*) The maps b_i, d_i, v_j are C^r on U.

(ii) (*Tracking*) For all $x' \in U$, we have:

$$B(x') = \{(b_i(x'), d_i(x'))\}_{i\in I} \cup \{(v_j(x'), +\infty)\}_{j\in J} \cup \Delta^\infty.$$

Local coordinate systems at some point x track smoothly the endpoints of the intervals of images of B, on an open neighborhood of x. The equivalence between r-differentiability (Definition 2.2) and having a C^r local coordinate system (Definition 2.3) is given by the following proposition.

Proposition 2.4. *Let $B\colon \mathcal{M} \to \mathrm{Bar}$, where $\mathcal{M}$ is a smooth manifold, and let $x \in \mathcal{M}$. Then B is r-differentiable at x if and only if it admits a C^r local coordinate system at x. More specifically, post-composing a local lift $\tilde{B}\colon \mathcal{M} \to \mathbb{R}^{2m+n}$ with the quotient map $Q_{m,n}$ yields a C^r local coordinate system, and on the other hand, fixing an order on the functions of a C^r coordinate system yields a C^r local lift.*

The proof can be found in Section 3.1 of [10]. We next define the differentiability of maps $\mathrm{Bar} \to \mathcal{N}$.

Definition 2.5. Let $V\colon \mathrm{Bar} \to \mathcal{N}$, where $\mathcal{N}$ is a smooth manifold, and let $D \in \mathrm{Bar}$ and $r \in \overline{\mathbb{N}}$. The map V is said to be *r-differentiable* at D if for all $m, n \in \mathbb{N}$ and $\tilde{D} \in \mathbb{R}^{2m+n}$ such that $Q_{m,n}(\tilde{D}) = D$, the map $V \circ Q_{m,n}\colon \mathbb{R}^{2m+n} \to \mathcal{N}$ is C^r on an open neighborhood of $\tilde{D}$. In this case, the differential of V at D is defined as

$$d_{D,\tilde{D}}V \coloneqq d_{\tilde{D}}(V \circ Q_{m,n})\colon \mathbb{R}^{2m+n} \to T_{V(D)}\mathcal{N}. \tag{2.7}$$

In this case, the idea is to pre-compose the map V with $Q_{m,n}$ so that the composite $V \circ Q_{m,n}$ is a map $\mathbb{R}^{2m+n} \to \mathcal{N}$, and then check that for any pre-image $\tilde{D}$ of D, the composite is differentiable. Then, the differential of V at D is seen as the differential of $V \circ Q_{m,n}$ seen as a map between manifolds, at some pre-image $\tilde{D}$ of D —hence the differential may depend on the choice of $\tilde{D}$.

To end this section, we end with one of the most essential results about the differentiability framework we have just exposed, which connects both definitions of differentiability.

Proposition 2.6. *Let $B\colon \mathcal{M} \to \mathrm{Bar}$ be r-differentiable at some $x \in \mathcal{M}$ and $V\colon \mathrm{Bar} \to \mathcal{N}$ be r-differentiable at $B(x) \in \mathrm{Bar}$. Then:*

(i) *$V \circ B$ is C^r at x as a map between manifolds.*

(ii) *If $r \geq 1$ then, for any local C^1 lift $\tilde{B}\colon U \to \mathbb{R}^{2m+n}$ of B around x, we have:*

$$d_x(V \circ B) = d_{B(x),\tilde{B}(x)}V \circ d_{x,\tilde{B}}B.$$

Proof. Proof of (i). It is a direct consequence of the definitions of B and V. On one hand, B is r-differentiable at x, so there exists an open neighborhood U of x, and a local C^r

lift $\tilde{B}\colon U \to \mathbb{R}^{2m+n}$ for some m, n such that $B = Q_{m,n} \circ \tilde{B}$ in U. On the other hand, V is r-differentiable at $B(x)$, so $V \circ Q_{m,n}\colon \mathbb{R}^{2m+n} \to \mathcal{N}$ is C^r in an open neighborhood W of $\tilde{B}(x)$ (since $Q_{m,n}(\tilde{B}(x)) = B(x)$). We can reduce U if necessary so that $\tilde{B}(U) \subseteq W$. Therefore we have the following composition:

$$\begin{aligned} \mathcal{M} &\xrightarrow{\tilde{B}} \mathbb{R}^{2m+n} \xrightarrow{V \circ Q_{m,n}} \mathcal{N} \\ x' \in U &\mapsto \tilde{B}(x') \in W \mapsto (V \circ Q_{m,n})(\tilde{B}(x')), \end{aligned}$$

and since $\tilde{B}$ is C^r on U and $V \circ Q_{m,n}$ is C^r on $\tilde{B}(U)$, by the composition of C^r maps between manifolds, we have that $(V \circ Q_{m,n}) \circ \tilde{B} = V \circ (Q_{m,n}) \circ \tilde{B}) = V \circ B$ is C^r as a map between manifolds on U and in particular at x.

Proof of (ii). Notice that $V \circ B$ is C^r as a map between manifolds, and $d_x(V \circ B)$ is independent of $\tilde{B}$ from (i) (since for any $x \in \mathcal{M}$, in a neighborhood of x the function $V \circ B$ does not depend on the lift). In addition, we have $V \circ B = (V \circ Q_{m,n}) \circ \tilde{B}$ for all $x' \in U \subseteq \mathcal{M}$, so differentiating, we have $d_x(V \circ B) = d_{\tilde{B}(x)}(V \circ Q_{m,n}) \circ (d_x\tilde{B})$. Using the notation introduced in this section, we can also write it as $d_x(V \circ B) = d_{B(x),\tilde{B}(x)}V \circ d_{x,\tilde{B}}B$. Hence the composition of the differentials is independent of the lift $\tilde{B}$. □

This result shows that, although the differential of two functions $B\colon \mathcal{M} \to \text{Bar}$ and $V\colon \text{Bar} \to \mathcal{N}$ depends on the choice of lift $\tilde{B}$, the composition $V \circ B$ does not depend on the choice of local lift. In addition, this proposition is a version of the chain rule for maps factoring through Bar, which will be essential for the proofs of smoothness of the topological regularizers viewed as maps between manifolds.

2.2 Differentiability of barcode generators

The main result of this section can be stated as *the generation of a barcode via the Rips filtration from a point cloud in $\mathbb{R}^{nd}$ is differentiable in a generic subspace of $\mathbb{R}^{nd}$*. The section is organized as follows. We first define two main notions needed for stating the main result. First, we introduce the notion of point clouds in *general position*, and we prove that the set of point clouds in general position is generic, i.e., open and dense, in $\mathbb{R}^{nd}$. Second, we define the Rips parametrization. Once equipped with these concepts, we formally state the main result of the section in Theorem 2.11. The rest of the section is dedicated to its proof, which relies on several intermediate results, mainly Proposition 2.16 and Proposition 2.17.

Definition 2.7. A point cloud $P = (p_1, \ldots, p_n) \in \mathbb{R}^{nd}$ is in *general position* if

(i) for any pair $i \neq j \in \{1, \ldots, n\}$, we have $p_i \neq p_j$, and

(ii) for any $\{i, j\} \neq \{k, l\}$ with $i, j, k, l \in \{1, \ldots, n\}$ and $i \neq j$, $k \neq l$, we have $\|p_i - p_j\|_2 \neq \|p_k - p_l\|_2$.

We denote the space of point clouds in $\mathbb{R}^{nd}$ in general position by $\tilde{\mathcal{P}}$.

In other words, in a point cloud in general position, no two points are overlapped at the same position and no two different pairs of points are at equal distance.

Proposition 2.8. *The space $\tilde{\mathcal{P}}$ is generic in $\mathbb{R}^{nd}$.*

Proof. Let U and V be the sets of points in $\mathbb{R}^{nd}$ that satisfy (i) and (ii), respectively. Therefore, $\tilde{\mathcal{P}} = U \cap V$. We will prove that both U and V are generic and, since the finite intersection of generic spaces is generic, it will follow that $\tilde{\mathcal{P}}$ is generic. The space of points not satisfying (i) from Definition 2.7, U^c, is the finite union of hyperplanes

$$U^c = \bigcup_{1 \leq i < j \leq n} \{p_i = p_j\}, \tag{2.8}$$

therefore its complementary U is generic. We now want to see that V is generic. We first define the functions f_{ijkl}, with $\{i,j\} \neq \{k,l\}$, $i,j,k,l \in \{1,\dots,n\}$, $i \neq j$ and $k \neq l$, as

$$\begin{aligned} f_{ijkl} \colon \mathbb{R}^{nd} &\to \mathbb{R} \\ P &\mapsto \|p_i - p_j\|_2^2 - \|p_k - p_l\|_2^2. \end{aligned} \tag{2.9}$$

These functions are C^1 in $\mathbb{R}^{nd}$, and $\{P \colon \nabla f_{ijkl}(P) \neq \mathbf{0}\}$ is a generic subset of $\mathbb{R}^{nd}$. Indeed, differentiating f_{ijkl} we obtain $\{P \colon \nabla f_{ijkl}(P) = \mathbf{0}\} = \{p_i = p_j\} \cap \{p_k = p_l\}$, which is a finite intersection of hyperplanes and therefore its complement is generic [1]. We now prove that the set $G_{ifkl} = \{P \colon f_{ijkl}(P) \neq 0\}$ is generic. We have $G_{ifkl} = f_{ijkl}^{-1}(\mathbb{R} \setminus \{0\})$, which is the pre-image of an open set by a continuous function, so G_{ifkl} is open in $\mathbb{R}^{nd}$. Now, assume G_{ijkl} is not dense. Then, there exist $P \in \mathbb{R}^{nd}$ and $\epsilon > 0$ such that $f_{ijkl}(P) = 0$ (i.e., $P \notin G_{ijkl}$) and $f_{ijkl}(P') = 0$ for all $P' \in B(P,\epsilon)$. However, the fact that $f_{ijkl} \equiv 0$ in $B(P,\epsilon)$ implies that f_{ijkl} is constant in $B(P,\epsilon)$, and therefore $B(P,\epsilon) \subseteq \{P' \colon \nabla f_{ijkl}(P') = \mathbf{0}\}$. This is in contradiction with the fact that $\{P' \colon \nabla f_{ijkl}(P') \neq \mathbf{0}\}$ is generic. Having reached a contradiction, we conclude that G_{ijkl} is generic. Now, $V = \bigcap_{\{i,j\} \neq \{k,l\}; i \neq j, k \neq l} G_{ijkl}$ is a finite intersection of generic subspaces of $\mathbb{R}^{nd}$, so V is generic in $\mathbb{R}^{nd}$, hence $\tilde{\mathcal{P}}$ is generic in $\mathbb{R}^{nd}$. □

In order to show that the generation of a barcode from a point cloud is an ∞-differentiable transformation with respect to the coordinates of the point cloud, we need to work through the space of filter functions. This is due to the fact that the p-persistent homology operators Dgm_p are only defined on the space of filter functions, and not directly on the space of point clouds. As a consequence, we define an intermediate function that parametrizes the filter function of the point cloud with the coordinates of the point cloud as parameters. Since computationally we will use the Rips filtration, we focus here on the Rips parametrization.

Definition 2.9. Let $\mathbb{R}^{nd}$ be the space of point clouds, and let $K = 2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ be the total complex. The *Rips parametrization* is defined as

$$\begin{aligned} F \colon \mathbb{R}^{nd} &\to \mathbb{R}^K \\ P &\mapsto F(P) = \{f \colon K \to \mathbb{R},\ f(\sigma) = \max_{i,j \in \sigma} \|p_i - p_j\|_2\}. \end{aligned} \tag{2.10}$$

Hence, F *parametrizes* the Rips filter function f of P with the parameters being the coordinates of the points of P.

Definition 2.10. Let $\mathbb{R}^{nd}$ be the space of point clouds. Given $p \in \mathbb{N}$, the *barcode generator of degree* p (via the Rips filtration) is the map

$$\begin{aligned} B_p \colon \mathbb{R}^{nd} &\to \mathrm{Bar} \\ P &\mapsto B_p(P) = \mathrm{Dgm}_p(F(P)), \end{aligned} \tag{2.11}$$

[1] A detailed proof of the expression of the set $\{P \colon \nabla f_{ijkl}(P) = \mathbf{0}\}$ is given in Appendix B.1, Lemma B.2.

where F is the Rips parametrization and Dgm_p is the persistence diagram generator of degree p.

Theorem 2.11. *Let $\mathbb{R}^{nd}$ be the space of point clouds. Given any $p \in \mathbb{N}$, the barcode generator B_p of degree p is ∞-differentiable in $\tilde{\mathcal{P}} \subseteq \mathbb{R}^{nd}$.*

The sketch of the proof is the following: we first state several notions related to filter functions and we prove two important properties about filter functions. Using these properties, we then prove a result on the local differentiability of B_p in Proposition 2.16. However, the theorem assumes the differentiability of F, and thus the proof of the differentiability of F remains. Therefore, we prove the differentiability of F in $\tilde{\mathcal{P}}$ in Proposition 2.17. We conclude by connecting Proposition 2.16 and Proposition 2.17 in the proof of Theorem 2.11.

Definition 2.12. Given a filter function $f\colon K \to \mathbb{R}$, the values it takes in each simplex induce a pre-ordering on the simplices of K. Two filter functions $f, g\colon K \to \mathbb{R}$ are *order equivalent* if they induce the same pre-order on K. This is an equivalence relation and is noted $\sim$. The equivalence class of each filter function is noted $[f] = \{g \in \mathbb{R}^K \mid f \sim g\}$. The space of equivalence classes is noted $\Omega(\mathbb{R}^K)$.

Next, given a simplicial complex K, we define a set of simplices from K that are "enough to fully describe the persistence diagram". This set is the so-called *barcode template*, and is more formally described as follows.

Definition 2.13. Assume $\mathbb{R}^{nd}$ is the space of point clouds and let $K = 2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ be the total complex. Given a filter function $f \in \mathbb{R}^K$ and a homology degree $0 \leq p \leq d$, a *barcode template* is a pair (P_p, U_p) where P_p is a multi-set of pairs of simplices of K, and U_p is a multi-set of simplices in K, such that

$$\mathrm{Dgm}_p(f) = \{(f(\sigma), f(\sigma'))\}_{(\sigma,\sigma') \in P_p} \cup \{(f(\sigma), +\infty\}_{\sigma \in U_p} \cup \Delta^{\infty}. \tag{2.12}$$

In other words, a barcode template is a set of simplices that are associated with the birth and death of all the p-holes in the p-persistence diagram. In the remaining of this chapter, we restrict our work to Rips filter functions. In the next two propositions, we prove two key results about these filter functions: 1) for any Rips filter function there always exists a barcode template, and 2) order equivalent filter functions have the same set of barcode templates.

Proposition 2.14. *Let $K = 2^{\{1,\dots,n\}} \setminus \{\emptyset\}$ be the total complex and $f \in \mathbb{R}^K$ its Rips filter function parametrized by some $P \in \mathbb{R}^{nd}$. Then, for any homology degree $0 \leq p \leq d$, there exists a barcode template (P_p, U_p) of f.*

Proof. We know from the previous chapter that the persistence module of degree p, $\mathbf{H}_p(f)$, can be expressed as an interval decomposition $\bigoplus_{J \in \mathcal{J}} \mathbb{I}_J$, where each J is either a bounded interval with endpoints $b < d$, or an unbounded interval $(b, +\infty)$. The intervals are in bijective correspondence with the points in the persistence diagram. Note that for any bounded interval J in $\mathcal{J}$ with endpoints b and d, we have $f^{-1}(b) \times f^{-1}(d) \neq \emptyset$. The reason is that J represents the life time of a p-hole, and the birth (or formation) of the p-hole at time b corresponds to the birth of a simplex σ_J. By the definition of f, since σ_J is born at time b, we have $f(\sigma_J) = b$. Analogously, the death of the p-hole at time d is marked by the

birth of another simplex τ_J that "kills" the hole, and since τ_J is born at time d, we have $f(\tau_J) = d$. Therefore, $(\sigma_J, \tau_J) \in f^{-1}(b) \times f^{-1}(d)$. Hence, proceeding in this fashion, we obtain the set

$$P_p = \{(\sigma_J, \tau_J) \mid J \in \mathcal{J},\ J = (b, d),\ b, d \in \mathbb{R}\}. \tag{2.13}$$

An analogous reasoning can be done for intervals of the form $J = (b, +\infty)$ —in that case, we only need the simplex σ_J that marks the birth of the p-hole, satisfying $f(\sigma_J) = b$. In this case we obtain the set

$$U_p = \{(\sigma_J) \mid J \in \mathcal{J},\ J = (b, +\infty),\ b \in \mathbb{R}\}. \tag{2.14}$$

And, by construction, (P_p, U_p) is a barcode template. □

Proposition 2.15. *If $f, f' \in \mathbb{R}^K$ are two Rips filter functions parametrized by points P, P' in $\mathbb{R}^{nd}$, and $f \sim f'$, then a barcode template of f is also a barcode template of f' and vice-versa.*

Proof. To enhance clarity we use the notation $(K_i)_i$ and $(K'_j)_j$ for the filtrations induced by f and f', respectively, where the subscripts are the times (induced by f and f' respectively) of appearance of each new complex. Since $f \sim f'$, the order of appearance of each simplex is the same under the two filtrations, and therefore we can arrange the simplices of $(K_i)_i$ noted σ_s and the simplices of $(K'_j)_j$ noted σ'_t into the sets $\{\sigma_{s_1}, \dots, \sigma_{s_N}\}$ and $\{\sigma'_{t_1}, \dots, \sigma'_{t_N}\}$ respectively, where s_i and t_i are the birth times of each simplex according to the respective filtration (i.e., $f(\sigma_s) = s$ and $f'(\sigma'_t) = t$). In addition, we can choose the arrangement of these two sets so that $\sigma_{s_k} = \sigma'_{t_k}$ for all $k = 1, \dots, N$, and so that s_i and t_i are increasing sequences. Let $g\colon \{s_i\}_i \to \{t_i\}_i$ be such that $g(s_k) = t_k$ for $k = 1, \dots, N$. Therefore g is an increasing function and maps $f(\sigma)$ to $f'(\sigma)$ for any simplex σ of K. As a consequence, $\sigma_{s_k} = \sigma'_{g(s_k)}$ for $k = 1, \dots, N$. Therefore, the filtrations induced by f and f' are shifted; the only difference is the time associated to each complex; we have $K_{s_k} = K'_{g(s_k)}$ for $k = 1, \dots, N$.

From the construction of the persistent homology and persistence module (which only depends on the filtration, see Chapter 1), for any homology degree p, the two filtrations $(K_i)_i$ and $(K'_j)_j$ have the same persistence module and thus the same decomposition into $\mathcal{P}$-intervals. Therefore, there is a one-to-one correspondence between the barcodes of the two filtrations.

We can finally show that a barcode template of f is also of f'. Take any off-diagonal interval (b_i, d_i) from the barcode of f. It has a corresponding pair (σ_J, τ_J) in the barcode template, and a corresponding interval $(g(b_i), g(d_i))$ in the barcode of f' that represents the life-time of the same p-hole. Since $f(\sigma_J) = b_i$, $f(\tau_J) = d_i$, from the definition of g we have $f'(\sigma_J) = g(b_i)$, $f'(\tau_J) = g(d_i)$, and therefore (σ_J, τ_J) can be used as the element representing the interval $(g(b_i), g(d_i))$ in a barcode template of f'. Using the same argument for all the intervals of the barcode of f (and a similar argument for non-bounded intervals) we conclude that (P_p, U_p) is a barcode template of f'. □

Now we can state and prove the local differentiability of B_p.

Proposition 2.16. *Let $P \in \mathbb{R}^{nd}$, and assume that the Rips parametrization $F\colon \mathbb{R}^{nd} \to \mathbb{R}^K$ is of class $\mathcal{C}^r$ (for some $r \geq 0$) on some open neighborhood U of P, such that for all $P' \in U$, $F(P) \sim F(P')$. Then $B_p\colon \mathbb{R}^{nd} \to \mathrm{Bar}$ is r-differentiable at P.*

Proof. The space U is an open subset of $\mathbb{R}^{nd}$. From Proposition 2.14, we can take a barcode template (P_p, U_p) for $F(P)$. From Proposition 2.15, (P_p, U_p) is consistent as a barcode template for all $F(P')$, $P' \in U$. Therefore for all $P' \in U$ we have

$$B_p(P') = \{(F(P')(\sigma), F(P')(\tau))\}_{(\sigma,\tau)\in P_p} \cup \Delta^\infty \tag{2.15}$$

which is a local coordinate system for B_p at P, and is C^r since F is C^r over U. Therefore, by Proposition 2.4, B_p is r-differentiable at P. □

Proposition 2.17. *The Rips parametrization $F\colon \mathbb{R}^{nd} \to \mathbb{R}^K$ is C^∞ over $\tilde{\mathcal{P}}$.*

Proof. First, notice that F is continuous in $\mathbb{R}^{nd}$ due to the continuity of the norm and max functions. If $P \in \tilde{\mathcal{P}}$, then, from the definition of $\tilde{\mathcal{P}}$, the distances $\|p_i - p_j\|_2$, $1 \le i < j \le n$ are strictly ordered at P. We define, for each $\sigma \subseteq K$,

$$\{i(\sigma), j(\sigma)\} \coloneqq \operatorname*{argmax}_{k,l\in\sigma} \|p_k - p_l\|_2, \tag{2.16}$$

and we define a new parametrization $\tilde{F}$ as $\tilde{F}(P')(\sigma) = \|p'_{i(\sigma)} - p'_{j(\sigma)}\|_2$ for all $\sigma \in K$. Clearly $\tilde{F}(P) = F(P)$. Since the functions $P' \mapsto \|p'_i - p'_j\|_2$ are continuous, it follows that the pre-order induced by P, $\|p_{i_1} - p_{j_1}\|_2 < \ldots < \|p_{i_N} - p_{j_N}\|_2$, remains the same in an open neighborhood $U \subseteq \tilde{\mathcal{P}}$ of P. If we now take any $P' \in U$ and $\sigma \in K$, then $F(P')(\sigma)$ is the largest element of the ordering $\|p'_{i_1} - p'_{j_1}\|_2 < \ldots < \|p'_{i_N} - p'_{j_N}\|_2$ that involves a pair $\{i, j\} \in \sigma$. From the definition of $\{i(\sigma), j(\sigma)\}$ and the fact that the strict ordering of pairwise distances is preserved, $F(P')(\sigma)$ must then be equal to $\|p'_{i(\sigma)} - p'_{j(\sigma)}\|_2 = \tilde{F}(P')(\sigma)$. Therefore, $F = \tilde{F}$ in U. In summary, we have proved that for all $P' \in U$, $\sigma \in K$, we have

$$F(P')(\sigma) = \|p'_{i(\sigma)} - p'_{j(\sigma)}\|_2, \tag{2.17}$$

with the indices fixed by P. Since $U \subseteq \tilde{\mathcal{P}}$, it follows that all the distances in (2.17) are strictly positive, so F is completely determined by functions $P' \mapsto \|p'_{i(\sigma)} - p'_{j(\sigma)}\|_2$, which are C^∞ in U, so F is C^∞ in U, and in particular in P. □

The proof of Theorem 2.11 now follows from Proposition 2.17 and Proposition 2.16.

Proof of Theorem 2.11. Take any $P \in \tilde{\mathcal{P}}$. Then from Proposition 2.17 there exists an open neighborhood U of P such that $U \subseteq \tilde{\mathcal{P}}$ and F is C^∞ in U. In addition, from the proof of Proposition 2.17, we can choose U such that (i) the strict order of pairwise distances in P remains constant in U, and (ii) for any $P' \in U$ and $\sigma \in K$, we have the equality $F(P')(\sigma) = \|p'_{i(\sigma)} - p'_{j(\sigma)}\|_2$. Then, (i) and (ii) imply that for any $P' \in U$, $F(P')$ and $F(P)$ induce the same pre-order on the simplices of K, i.e., $F(P') \sim F(P)$ for all $P' \in U$. Hence, from Proposition 2.16, B_p is ∞-differentiable at P. □

2.3 Differentiability of topological regularizers

We now study the differentiability of several functions from Bar to $\mathbb{R}$. Furthermore, we expand the work, using the composition with B_p, to maps from the space of point clouds $\mathbb{R}^{nd}$ to $\mathbb{R}$, factoring through Bar. We refer to these functions as *topological regularizers*, since they can be applied in generative modelling problems where the goal is to generate point clouds

with specific topological features, and thus the use of loss functions extracting information from their persistence diagrams can be used to enhance the learning process. For instance, if the goal is to generate batches of data points (seen as point clouds) with similar spatial distributions to ground truth point clouds (e.g., input batches), the use of additional loss terms providing a measure of dissimilarity between their persistence diagrams can be used as a tool for regularizing the training process. In fact, the minimization of such loss terms through gradient descent can lead to a model that produces data with a distribution with a similar "shape", i.e., with a similar persistence diagram, to the true data —and, with similar, we mean *with a low value of the topological regularizer used.*

Proposition 2.18. *Let $F\colon \mathbb{R}^2 \to \mathbb{R}$ be a function such that $F(b,b) = 0$ for all $b \in \mathbb{R}$. If F is C^r in $\mathbb{R}^2$, then the map on barcodes $f(D) \coloneqq \sum_{(b,d)\in D} F(b,d)$ is r-differentiable in* Bar.

Proof. Recall the Definition 2.5 of r-differentiability of a map on barcodes. Let $D \in \text{Bar}$, and take $m, n \in \mathbb{N}$ and $\tilde{D} \in \mathbb{R}^{2m+n}$ such that $Q_{m,n}(\tilde{D}) = D$. We want to prove that the function $f \circ Q_{m,n}\colon \mathbb{R}^{2m+n} \to \mathcal{N}$ is C^r on an open neighborhood of $\tilde{D}$. But F is C^r in each $(b_i, d_i) \in D$, so $\tilde{F}_i\colon \tilde{D} = (\dots, b_i, d_i, \dots) \mapsto F(b_i, d_i)$ is C^r in $\tilde{D}$. Therefore, the sum $\sum_i \tilde{F}_i = f \circ Q_{m,n}$ is C^r in $\tilde{D}$. Finally, notice that for any choice of pre-image $\tilde{D}$ we have $(f \circ Q_{m,n})(\tilde{D}) = f(D)$. This is due to the fact that $F(b,b) = 0$ for all $b \in \mathbb{R}$, so adding diagonal points in the pre-image does not change the value of $(f \circ Q_{m,n})(\tilde{D})$. □

2.3.1 Push functions

Definition 2.19. Let $\mathbb{R}^{nd}$ be the space of point clouds, let p be a non-negative integer and denote by B_p the barcode generator of degree p. The *push function of degree p* is defined as

$$\begin{aligned} P_p^\circ\colon \mathbb{R}^{nd} &\to \mathbb{R} \\ P &\mapsto -\sum_{(b,d)\in B_p(P)} (d-b). \end{aligned} \tag{2.18}$$

The notation P_p° is used for avoiding confusion with the notation of point cloud P. The push function computes the persistence diagram of degree p, and outputs the sum of all distances of off-diagonal points to the diagonal, multiplied by a factor of -2, since the distance of an off-diagonal point to Δ^∞ is $(d-b)/2$ —note that, although we allow the existence of points below the diagonal, persistence diagrams arising from Rips filtrations only have points above the diagonal, hence their distance to the diagonal is $(d-b)/2$. The choice of multiplying by -2 comes from the computational use of this function. In fact, assume we use $P_p^\circ(P)$ as a loss function and perform backpropagation for updating the coordinates of the point cloud. Then, gradient descent provides a small perturbation of P that decreases $P_p^\circ(D)$. By the definition of P_p°, this means that the computed perturbation of P results on slightly moving the off-diagonal bounded points of $B_p(P)$ away from the diagonal. In other words, backpropagation through P_p° *pushes* bounded off-diagonal points of $B_p(P)$ away from the diagonal. This has several applications. For instance, the application of P_0° can lead to pushing apart the clusters of the point cloud and increasing their life-times, and the implementation of P_1° can increase the size of loops.

Proposition 2.20. *Let $\mathbb{R}^{nd}$ be the space of point clouds, and let p be a non-negative integer. The push function P_p° of degree p is C^∞ in $\tilde{\mathcal{P}}$, which is generic in $\mathbb{R}^{nd}$.*

Proof. Let T be the map on barcodes given by $T(D) = \sum_{(b,d)\in D}(b-d)$. Then $P_p^\circ = T \circ B_p$. Denote by F the function $F(b,d) = b - d$; in particular F is C^∞ for any $(b,d) \in \mathbb{R}^2$, and $F(b,b) = 0$ for all $b \in \mathbb{R}$. From Proposition 2.18, $\sum_{(b,d)\in D} F(b,d) = T$ is ∞-differentiable in Bar. Assume now P is any point cloud in $\tilde{\mathcal{P}}$. Then from Theorem 2.11, B_p is ∞-differentiable in P, and T is ∞-differentiable in $B_p(P) \in$ Bar, so from Proposition 2.6 the composition $T \circ B_p = P_p^\circ$ is C^∞ in P. □

2.3.2 Reininghaus dissimilarity

We now focus on a measure of dissimilarity between persistence diagrams that is simple to compute and shows promising potential in machine learning problems. This measure is based on the *persistence scale space kernel* or *Reininghaus kernel* k_σ. We will not go through the full derivation here, but we refer the reader to [22] for details. The main conclusions that can be highlighted from the function k_σ are: 1) it is stable with respect to the 1-Wasserstein distance, and 2) it has shown promising capabilities for vision tasks such as shape classification and texture image classification. A simple expression of the persistence scale space kernel k_σ is the following:

$$k_\sigma(F,G) = \frac{1}{8\pi\sigma}\sum_{p\in F,\, q\in G} e^{-\frac{\|p-q\|_2^2}{8\sigma}} - e^{-\frac{\|p-\bar{q}\|_2^2}{8\sigma}}, \tag{2.19}$$

where $\bar{q} = (q_2, q_1)$ for $q = (q_1, q_2)$, and $\|\cdot\|_2$ is the Euclidean norm. The sum goes over all pairs $p \in F$, $q \in G$ of off-diagonal bounded points of both diagrams. This kernel induces a pseudo-distance d_σ in Bar, which is equal to the distance between feature maps in the feature space [22]. The pseudo-distance, which we refer to as *Reininghaus dissimilarity*, is given by the following expression:

$$d_\sigma(F,G) = \sqrt{k_\sigma(F,F) + k_\sigma(G,G) - 2k_\sigma(F,G)}. \tag{2.20}$$

Now, assume given a machine learning problem where we want to generate a point cloud with topological properties similar to those given by a reference persistence diagram D_0. If the coordinates of the point cloud are learnable (i.e., they can be updated through gradient descent), and D is the persistence diagram of the learnable point cloud, a question that arises is: can we introduce the Reininghaus dissimilarity in the loss function and minimize it through gradient descent? In other words, is $d_\sigma(D, D_0)$ differentiable with respect to the coordinates of the learnable point cloud? The answer is given by the following two results. We define the space $\mathrm{Bar}_0 \subseteq \mathrm{Bar}$ as the space of barcodes with no unbounded intervals.

Proposition 2.21. *Let D_0 be a barcode in Bar_0, and let $d_\sigma(\cdot, D_0)\colon \mathrm{Bar}_0 \to \mathbb{R}$ be the map assigning to a barcode its Reininghaus dissimilarity to D_0. Then, the squared Reininghaus dissimilarity $d_\sigma^2(\cdot, D_0)$ is ∞-differentiable in Bar_0.*

Proof. First, note that $k_\sigma(D, D_0)$ can be expressed as

$$k_\sigma(D, D_0) = \sum_{p\in D} F_\sigma(p), \tag{2.21}$$

where F_σ is defined as:

$$F_\sigma(p) = \frac{1}{8\pi\sigma} \sum_{q \in D_0} e^{-\frac{\|p-q\|_2^2}{8\sigma}} - e^{-\frac{\|p-\bar{q}\|_2^2}{8\sigma}}. \tag{2.22}$$

Here p, q are points in $\mathbb{R}^2$, so, if $p = (p_1, p_2)$, $q = (q_1, q_2)$, then $\|p-q\|_2^2 = (p_1-q_1)^2+(p_2-q_2)^2$ and $\|p - \bar{q}\|_2^2 = (p_1 - q_2)^2 + (p_2 - q_1)^2$. In particular, F_σ is C^∞ in $\mathbb{R}^2$, and $F(p) = 0$ for all $p \in \Delta^\infty$. Therefore, we can apply Proposition 2.18, and as a consequence $k_\sigma(\cdot, D_0)$ is ∞-differentiable at any $D \in \mathrm{Bar}$ with no unbounded intervals. With a similar argument we can prove that $k_\sigma(D, D)$ is ∞-differentiable in D for any $D \in \mathrm{Bar}_0$. In addition, $k_\sigma(D_0, D_0)$ is a constant since it is independent of D.

In summary, the maps $k_\sigma(D, D_0)$, $k_\sigma(D, D)$ and $k_\sigma(D_0, D_0)$ are ∞-differentiable with respect to D in Bar_0. Using the fact that the sum of ∞-differentiable maps is ∞-differentiable —see Lemma B.1—, we have that $d_\sigma^2(D, D_0) = k_\sigma(D, D) + k_\sigma(D_0, D_0) - 2k_\sigma(D, D_0)$ is ∞-differentiable at D, for any $D \in \mathrm{Bar}_0$. □

Corollary 2.22. *Denote by $\mathbb{R}^{nd}$ the space of point clouds, let p be a non-negative integer, and let $\tilde{d}_{\sigma,D_0} \colon \mathbb{R}^{nd} \to \mathbb{R}$ be the map defined as $\tilde{d}_{\sigma,D_0}(P) = d_\sigma^2(B_p(P), D_0)$ for any $P \in \mathbb{R}^{nd}$. Then, $\tilde{d}_{\sigma,D_0}$ is C^∞ in $\tilde{\mathcal{P}}$, which is generic in $\mathbb{R}^{nd}$.*

Proof. From Theorem 2.11, we have that $B_p \colon \mathbb{R}^{nd} \to \mathrm{Bar}$ is ∞-differentiable in $\tilde{\mathcal{P}}$, which is generic in $\mathbb{R}^{nd}$. In addition, Proposition 2.21 states that $d_\sigma^2(\cdot, D_0) \colon \mathrm{Bar} \to \mathbb{R}$ is ∞-differentiable in Bar_0. We now have to see that $B_p(P) \in \mathrm{Bar}_0$ for any $P \in \mathbb{R}^{nd}$. If $p > 0$, it is true since all p-holes die at some finite time (when they are filled with higher-dimensional simplices). If $p = 0$, we always have a single unbounded point $(0, +\infty)$ in the persistence diagram; however, it does not provide useful information, so it can be neglected. Finally, using Proposition 2.6, we have that $d_\sigma^2(\cdot, D_0) \circ B_p = \tilde{d}_{\sigma,D_0}$ is C^∞ in $\tilde{\mathcal{P}}$. □

2.3.3 Scaled Gaussian density estimators

In the case of 0-dimensional persistence diagrams, all points are of the form $(0, d)$, so the goal is to make a set of points in the y-axis imitate a distribution of reference. Hence, in this case, there are approaches that may be more efficient and faster (computationally) than the persistence scale space kernel. For instance, we can approximate the density of points of the form $(0, d)$ with a continuous function. We propose here a new approximator of the density of points for persistence diagrams obtained from the barcode generator of degree 0. The idea is to place a Gaussian function over each point of the diagram, whose height is scaled by a factor $H(d - b)$, where $H(x)$ is a smooth and increasing function on x and $H(0) = 0$. With these restrictions, points near the origin produce Gaussian curves with low height, while points with higher persistence produce larger Gaussians. This allows us to give more importance to points away from the origin, which can be associated with clusters in the original point cloud. The total density estimator is then the sum of all the Gaussians produced by off-diagonal points of the diagram. We define this type of density function more generally and formally as follows.

Definition 2.23. A *scaled Gaussian density estimator* or *SGDE* is a function of the form

$$\begin{aligned} E \colon \mathrm{Bar} \times [0; +\infty) &\to \mathbb{R} \\ (D, x) &\mapsto E(D, x) = \sum_{(b,d) \in D} H(d - b) e^{((d-x)/\sigma)^2} \end{aligned} \tag{2.23}$$

where $\sigma > 0$, $H\colon \mathbb{R}^2 \to \mathbb{R}$ is C^∞ in $\mathbb{R}^2$, and $H(0) = 0$.

Although we are only interested in diagrams of degree 0, the definition applies to any diagram in order to facilitate the proof of its smoothness. Furthermore, when using the scaled Gaussian density estimator as a loss function, we will only evaluate it at some fixed values of x. Hence, we want to see that for a given x, $E(D, x)$ is ∞-differentiable as a map $\mathrm{Bar} \to \mathbb{R}$ with D the only variable.

Proposition 2.24. *For any fixed $x \in [0; +\infty)$, and any SGDE E, the map on barcodes $E(\cdot; x)$ (where we have changed the notation to highlight that x is now a parameter) is ∞-differentiable as a map in* Bar.

Proof. Recall that $E(D; x)$ has the form given by Definition 2.23, for some function H and $\sigma > 0$. We define the function

$$G_x((b, d)) = H(d - b)e^{((d-x)/\sigma)^2}. \tag{2.24}$$

Here G_x satisfies the conditions of the function F from Proposition 2.18: it is C^∞ in $\mathbb{R}^2$ and $G_x((b, b)) = 0$ for all $b \in \mathbb{R}$, since $H(0) = 0$. In addition, we have

$$E(D; x) = \sum_{(b,d)\in D} G_x((b, d)). \tag{2.25}$$

Therefore, applying Proposition 2.18, $E(D; x)$ is ∞-differentiable in Bar. □

Corollary 2.25. *Let $\mathbb{R}^{nd}$ be the space of point clouds. Consider any SGDE E, and let $x \in [0; +\infty)$. Then, the composition map*

$$E(\cdot; x) \circ B_0 = E(B_0(P); x)\colon \mathbb{R}^{nd} \to \mathbb{R} \tag{2.26}$$

is C^∞ in $\tilde{\mathcal{P}} \subseteq \mathbb{R}^{nd}$, which is generic in $\mathbb{R}^{nd}$.

Proof. Let $x \in [0, +\infty)$. From Theorem 2.11, the map $B_0\colon \mathbb{R}^{nd} \to \mathrm{Bar}$ is ∞-differentiable in $\tilde{\mathcal{P}}$, and from Proposition 2.24, the map $E(\cdot; x)\colon \mathrm{Bar} \to \mathbb{R}$ is ∞-differentiable in Bar. Then, from Proposition 2.6, the composition $E(\cdot; x) \circ B_0\colon \mathbb{R}^{nd} \to \mathbb{R}$ is C^∞ in $\tilde{\mathcal{P}}$. □

For machine learning applications, the function we are interested in is not just the density at a point x, $E(\cdot; x) \circ B_0$, but rather the *difference* between two SGDEs. In order to compute this difference, we can take m values $x_1, \ldots, x_m \in [0; +\infty)$ and compute the mean of the squared difference between the density functions of D and D_0 at each x_i. The generic differentiability, in the space of point clouds, of the resulting function is given by the following proposition.

Proposition 2.26. *Let E be a SGDE, $D_0 \in \mathrm{Bar}$, and $\{x_1, \ldots, x_m\} \subseteq [0; +\infty)$. Assume that the space of point clouds is $\mathbb{R}^{nd}$. Then, the function $L_E\colon \mathbb{R}^{nd} \to \mathbb{R}$ given by*

$$L_E(P) = \frac{1}{m}\sum_{i=1}^{m}[E(B_0(P); x_i) - E(D_0; x_i)]^2 \tag{2.27}$$

is C^∞ in $\tilde{\mathcal{P}}$, which is generic in $\mathbb{R}^{nd}$.

Proof. Take any $i \in \{1, \dots, m\}$, and define the function $L_{E,i}(P)$ as

$$L_{E,i}(P) = \frac{1}{m}[E(B_0(P); x_i) - E(D_0; x_i)]^2. \tag{2.28}$$

This function corresponds to the composite $S \circ A_i \circ E(B_0(\cdot); x_i)$, where $S(z) = z^2$ and $A_i(z) = z - E(D_0; x_i)$. Hence, both A_i and S are C^∞ in $\mathbb{R}$. Since $E(B_0(\cdot); x_i)$ is C^∞ in $\tilde{\mathcal{P}}$ from the previous proposition, we have that $S \circ A_i \circ E(B_0(\cdot); x_i)$ is C^∞ in $\tilde{\mathcal{P}}$. This is true for all $i = 1, \dots, m$, so each $L_{E,i}(P)$ is C^∞ in $\tilde{\mathcal{P}}$. Finally, since $L_E(P) = \sum_{i=1}^{m} L_{E,i}(P)$ for all $P \in \mathbb{R}^{nd}$, we have that L_E is a sum of C^∞ functions in $\tilde{\mathcal{P}}$, so L_E is C^∞ in $\tilde{\mathcal{P}}$. □

Finally, we define the specific SGDE we use in our experiments.

Definition 2.27. A *4SGDE*, denoted by E_4, is a SGDE parametrized by $s > 0$ and $\sigma > 0$, and given by

$$E_4(D, x; \sigma, s) = s \sum_{(b,d) \in D} (d - b)^4 e^{((d-x)/\sigma)^2}. \tag{2.29}$$

An example of a 4SGDE applied to a persistence diagram of degree 0 is given in Figure A.2. From Corollary 2.25, the composition map $E_4(\cdot; x; \sigma, s) \circ B_0$ is ∞-differentiable in $\tilde{\mathcal{P}}$ for any $x \geq 0$. In addition, from Proposition 2.26, any function of the form 2.27 that compares the 4SGDE of the persistence diagram obtained from a learnable point cloud to the 4SGDE of a reference diagram D_0 is C^∞ in $\tilde{\mathcal{P}}$.

2.3.4 Persistent entropy

The concept of *persistent entropy* is an adaptation of the idea of entropy from the framework of information theory to the context of persistence diagrams. The *persistent entropy* of a persistence diagram D is defined as follows [23]:

$$\epsilon(D) = -\sum_{(b,d) \in D} \frac{|d - b|}{L} \log\left(\frac{|d - b|}{L}\right) \tag{2.30}$$

where $L = \sum_{(b,d) \in D} |d - b|$, and the summation only includes bounded off-diagonal points.

One can see that the smoothness of ϵ in Bar does not seem straightforward to prove and, in fact, may not be true —due to the non-differentiability of $\log(\frac{|d-b|}{L})$ in the diagonal. However, using a different approach, we can prove that the function computing the persistent entropy of persistence diagrams of degree 0 is generically C^∞ in the space of point clouds. To prove it, we first define the following type of maps from the space of point clouds to $\mathbb{R}$.

Definition 2.28. Let $\mathbb{R}^{nd}$ be the space of point clouds and $p \geq 0$, and let $B_p \colon \mathbb{R}^{nd} \to \text{Bar}$ be the barcode generator of degree p, as defined in Definition 2.10. The *p-persistent entropy generator* G_p is the function

$$\begin{aligned} G_p \colon \mathbb{R}^{nd} &\to \mathbb{R} \\ P &\mapsto \epsilon(B_p(P)). \end{aligned} \tag{2.31}$$

Proposition 2.29. *Let $\mathbb{R}^{nd}$ be the space of point clouds. The 0-persistent entropy generator G_0 is C^∞ in $\tilde{\mathcal{P}}$, which is generic in $\mathbb{R}^{nd}$.*

Proof. Let $P \in \tilde{\mathcal{P}}$. From the definition of $\tilde{\mathcal{P}}$, there is an open neighborhood of U of P, $U \subseteq \tilde{\mathcal{P}}$, where every point cloud $P' \in U$ has no overlapped points. As a consequence, for any $P' \in U$, $B_0(P')$ (without providing it with Δ^∞) has no points in the diagonal, and exactly n off-diagonal points ($n-1$ bounded points and one unbounded point). Therefore, for any $P' \in U$, the minimal pre-images of $B_0(P)$ lie in $\mathbb{R}^{2(n-1)} \times \mathbb{R}^1$.

From Theorem 2.11, the map B_0 is ∞-differentiable in $\tilde{\mathcal{P}}$. As a consequence, any lift $\tilde{B}_0' \colon \tilde{\mathcal{P}} \to \mathbb{R}^{2(n-1)} \times \mathbb{R}^1$ is C^∞ in $\tilde{\mathcal{P}}$. We fix one of these lifts and note it $\tilde{B}_0$. This map fixes an indexation of the points of the diagrams obtained from point clouds in U. As a consequence, the function

$$U \xrightarrow{\tilde{B}_0} \mathbb{R}^{2(n-1)+1} \xrightarrow{Q_{2(n-1),1}} \text{Bar} \xrightarrow{\epsilon} \mathbb{R} \tag{2.32}$$

is equal to G_0 in U. For any $P' \in U$, we use the notation

$$\tilde{B}_0(P') = (b_1(P'), d_1(P'), \ldots, b_{n-1}(P'), d_{n-1}(P'), v_1(P')). \tag{2.33}$$

Then, for any $P' \in U$, and for any $i = 1, \ldots, n-1$, the sign of $d_i(P') - b_i(P')$ does not change (since off-diagonal points remain outside the diagonal and thus stay either above or below the diagonal, and no points arise from the diagonal), hence $|d_i(P') - b_i(P')|$ is differentiable. Therefore, the function

$$\begin{aligned} &\epsilon \circ Q_{2(n-1)+1} \colon \mathbb{R}^{2(n-1)} \times \mathbb{R}^1 \to \mathbb{R} \\ &(b_1, d_1, \ldots, b_{n-1}, d_{n-1}, v_1) \mapsto -\sum_{i=1}^{n-1} \frac{|d_i - b_i|}{L} \log\left(\frac{|d_i - b_i|}{L}\right) \end{aligned} \tag{2.34}$$

is C^∞ in $\tilde{B}_0(P)$. Finally, since $\epsilon \circ Q_{2(n-1)+1}$ is C^∞ in $\tilde{B}_0(P)$, and $\tilde{B}_0$ is C^∞ in P, then the composite $\epsilon \circ Q_{2(n-1)+1} \circ \tilde{B}_0$ is C^∞ in P. Since $\epsilon \circ Q_{2(n-1)+1} \circ \tilde{B}_0$ is C^∞ is equal to G_0 in U (see (2.32)), we conclude that G_0 is also C^∞ in P. □

We have not proven the smoothness of maps employing lifts to non-minimal pre-images of $B_0(P)$, due to the following reason. Any such lift would use points (b_i, b_i), which correspond to points in the diagonal of the persistence diagrams. However, in a neighborhood of P these points will remain fixed in the diagonal —since, as we have seen, the number of off-diagonal points of 0-persistence diagrams obtained from point clouds in U remains constant. Therefore, these points have no effect on the persistent entropy of diagrams obtained from point clouds in U, and should not have an effect on the gradients of G_0. Hence, the only lift of interest is a lift that maps P to a minimal pre-image of $B_0(P)$.

2.3.5 Bottleneck distance to a fixed diagram

We now focus on the bottleneck distance between a fixed barcode D_0 and the barcode resulting from the Rips filtration of a point cloud $P \in \mathbb{R}^{kd}$. In this subsection, and the one that follows, we denote the number of points as k, and the dimension of each point as d, while n represents the number of unbounded intervals in D_0. Furthermore, the space of barcodes with n unbounded intervals is referred to as Bar_n. From now on, we assume that n is fixed. In practical applications, since the barcodes come from finite complexes, we will either work on Bar_1 when computing barcodes of degree 0, or on Bar_0 when the homology degree is $p \geq 1$.

In this subsection we prove that the map $d_{D_0}\colon \mathrm{Bar} \to \mathbb{R}$ defined as $d_{D_0}(D) = d_\infty(D, D_0)$ is generically ∞-differentiable in Bar_n. However, we will see that this result is not sufficient for showing the differentiability of the full map $d_{D_0} \circ B_p$ from $\mathbb{R}^{kd}$ to $\mathbb{R}$. The next subsection will provide conditions for knowing whether $d_{D_0} \circ B_p$ is C^∞. The main result of this section is the following.

Theorem 2.30. *Let D_0 be a barcode in Bar_n. Then, the map d_{D_0} is ∞-differentiable in a generic subset of Bar_n.*

The sketch of the proof is as follows. First, from the definition of ∞-differentiability, we have to prove that for *any* pre-image $\tilde{D}$ of D, which can be a-priori in any space $\mathbb{R}^{2m+n}$ for $m, n \geq 0$, the map $d_{D_0} \circ Q_{m,n}$ is C^∞ at $\tilde{D}$. However, proving simultaneously that all the possible compositions $d_{D_0} \circ Q_{m,n}$ are C^∞ at pre-images is not an easy task. Instead, we use a trick: we show that there is a space $\widehat{\mathrm{Bar}} \subseteq \mathrm{Bar}_n$ where, for any $D \in \widehat{\mathrm{Bar}}$, the smoothness of all possible lifts at pre-images of D can be inferred from the smoothness of the lift at a single pre-image, called *minimal pre-image*. This idea is stated more formally in Lemma 2.34. In addition, for proving it we need another result which, roughly speaking, says: *there is an open ball around D such that for all the barcodes D' in that open ball, $d_\infty(D', D_0)$ does not depend on the points that arise from the diagonal.* We prove it in Lemma 2.33. This result allows us to connect the differentiability of $d_{D_0} \circ Q_{m,n}$ at a minimal pre-image $\tilde{D}$ to the differentiability of $d_{D_0} \circ Q_{m',n}$ at any other pre-image $\tilde{D}'$, using the fact that in open neighborhoods of $\tilde{D}$ and $\tilde{D}'$, both maps use the same points for the bottleneck distance. Now, recall that we are interested in the *generic* smoothness of d_{D_0} in Bar_n, and not just at some point. Therefore, we prove:

(1) In Bar_n, for any given space $\mathbb{R}^{2m+n}$, the maps $d_{D_0} \circ Q_{m,n}$ are generically smooth (in Lemma 2.31).

(2) The space $\widehat{\mathrm{Bar}}$ is generic in Bar_n (in Lemma 2.32).

(3) A (small) open ball around any $D \in \mathrm{Bar}_n$ is contained in the image of the union of open balls of same radius centered at the pre-images of D (in Lemma 2.35).

We finally connect the five aforementioned lemmas in the final proof of Theorem 2.30, where we show that the *space of barcodes in Bar_n that admit an open neighborhood where d_{D_0} is smooth*, noted $\mathcal{W}$, is generic in Bar_n. To show it, we take any barcode D in Bar_n, and perform two infinitesimal perturbations of the barcode: one for having D in $\widehat{\mathrm{Bar}}$, and a second one for having a minimal pre-image $\tilde{D} \in \mathbb{R}^{2m+n}$ in the space of smoothness of $d_{D_0} \circ Q_{m,n}$. The second perturbation can be kept infinitesimal while keeping D in $\widehat{\mathrm{Bar}}$ since $\widehat{\mathrm{Bar}}$ is open and $Q_{m,n}$ is 1-Lipschitz. Then, we use Lemma 2.31 for proving the smoothness of $d_{D_0} \circ Q_{m,n}$ at a minimal pre-image of D, and then Lemma 2.34 for proving the smoothness of $d_{D_0} \circ Q_{m',n}$ at any other pre-image. In addition, the statement of Lemma 2.34 provides an even stronger result: all the maps $d_{D_0} \circ Q_{m',n}$ at pre-images of D are C^∞ on open balls centered at the pre-images, of same radius $\epsilon > 0$. We use this result and Lemma 2.35 to show the smoothness of d_{D_0} in an open neighborhood of D. This shows that $D \in \mathcal{W}$. The fact that we only needed two infinitesimal perturbations of D for making it belong to $\mathcal{W}$ makes $\mathcal{W}$ generic. We proceed now to the full proof.

Lemma 2.31. *For any $m \in \mathbb{N}$, the map $d_{D_0} \circ Q_{m,n}\colon \mathbb{R}^{2m+n} \to \mathbb{R}$ is generically C^∞.*

Proof. The proof consists in proving the generic differentiability of a "nicer" function $\tilde{d}_{D_0,m}\colon \mathbb{R}^{2m+n} \to \mathbb{R}$, and then proving that the function is equal to $d_{D_0} \circ Q_{m,n}$ in $\mathbb{R}^{2m+n}$. Let $m \in \mathbb{N}$, and we define an ordered matching $\tilde{\gamma}\colon \mathbb{R}^{2m+n} \to \mathbb{R}^{2m+n}$ as follows: the first m pairs of coordinate functions are of the form $\tilde{D} \mapsto (b_i, d_i) - (b_{0,i}, d_{0,i})$, where $(b_{0,i}, d_{0,i})$ is either an off-diagonal point of D_0 or the orthogonal projection of (b_i, d_i) into the diagonal, i.e., $(\frac{b_i+d_i}{2}, \frac{b_i+d_i}{2})$. Analogously, the last n coordinate functions are of the form $\tilde{D} \mapsto v_j - v_{0,j}$, where $(v_{0,j}, +\infty)$ is in D_0 —notice that in this case there is directly a matching between unbounded intervals, since an optimal matching will always match together pairs of unbounded intervals. We also impose that the intervals $(b_{0,i}, d_{0,i})$ are all different elements of D_0. Therefore $\tilde{\gamma}$ induces a matching between the persistence barcode given by $\tilde{D}$ and D_0. We denote by $D_0(\tilde{\gamma})$ the set of off-diagonal points $(b_0, d_0) \in D_0$ that are not used by $\tilde{\gamma}$ —by construction, these are all bounded points. We now define the function $\tilde{c}(\tilde{\gamma})$, corresponding to the cost of the matching, as follows:

$$\begin{aligned} \tilde{c}(\tilde{\gamma})\colon \mathbb{R}^{2m+n} &\to \mathbb{R} \\ \tilde{D} &\mapsto \max\left(\|\tilde{\gamma}(\tilde{D})\|_\infty,\ \left\{ \frac{|d_0 - b_0|}{2} \right\}_{(b_0,d_0)\in D_0(\tilde{\gamma})} \right). \end{aligned} \tag{2.35}$$

Given m and D_0, we denote by $\tilde{\Gamma}_m$ the set of all possible functions $\tilde{\gamma}$ as previously defined. Then $\tilde{\Gamma}_m$ is clearly finite and non-empty. Finally, we can define the map $\tilde{d}_{D_0,m}$ as

$$\begin{aligned} \tilde{d}_{D_0,m}\colon \mathbb{R}^{2m+n} &\to \mathbb{R} \\ \tilde{D} &\mapsto \min_{\tilde{\gamma}\in\tilde{\Gamma}_m} \tilde{c}(\tilde{\gamma})(\tilde{D}), \end{aligned} \tag{2.36}$$

and $\tilde{d}_{D_0,m}$ is C^∞ in a generic subset of $\mathbb{R}^{2m+n}$. This comes from the fact that each function $\tilde{c}(\tilde{\gamma})$ is the maximum of generically C^∞ functions in $\mathbb{R}^{2m+n}$, and the maximum of generically C^∞ functions is generically C^∞; therefore each $\tilde{c}(\tilde{\gamma})$ is generically C^∞ in $\mathbb{R}^{2m+n}$. Since the minimum of generically C^∞ functions is also generically C^∞, it follows that $\tilde{d}_{D_0,m}$ is generically C^∞ in $\mathbb{R}^{2m+n}$.

Now, we proceed to show that $\tilde{d}_{D_0,m} = d_{D_0} \circ Q_{m,n}$ in $\mathbb{R}^{2m+n}$. Fix a point $\tilde{D} \in \mathbb{R}^{2m+n}$ and let $D \coloneqq Q_{m,n}(\tilde{D})$. If $\tilde{\gamma}\colon \mathbb{R}^{2m+n} \to \mathbb{R}$ is an ordered matching, then it induces a matching γ between the barcodes D and D_0. In this matching, the m bounded intervals of D are matched to either off-diagonal points of D_0 or to the orthogonal projections onto the diagonal; this matching is determined by the first m pairs of components of $\tilde{\gamma}$. The last n components of $\tilde{\gamma}(\tilde{D})$ induce a matching between the unbounded intervals of D and D_0. In addition, by construction, $\tilde{c}(\tilde{\gamma})(\tilde{D})$ is equal to the cost $c(\gamma)$ of the matching γ —recall the definition of cost in the previous chapter and equation (2.35). Finally, we have to see that the minimum attained by $\tilde{d}_{D_0,m}$ (recall its definition in equation (2.36)) is the absolute minimum of all possible matchings in barcode space. First, notice that $\tilde{d}_{D_0,m}(\tilde{D}) \geq d_{D_0} \circ Q_{m,n}(\tilde{D})$, since $\tilde{d}_{D_0,m}$ attains the minimum of the cost for *some* of the matchings between D and D_0. On the other hand, if we take any optimal matching between D and D_0, the projections of points of D or D_0 onto the diagonal (if there are any) must be orthogonal projections —since this is the projection that minimizes the distance to the diagonal— and unbounded intervals of D must be matched to unbounded intervals of D_0 —otherwise the distance would be infinite, and $D, D_0 \in \mathrm{Bar}_n$ so the distance is finite. Therefore, we can find an ordered matching $\tilde{\gamma}$ satisfying the definition at the beginning of

the proof, such that the points $(b_{0,i}, d_{0,i})$ and $v_{0,j}$ are taken in accordance to the matching defined by γ. Therefore, $\tilde{c}(\tilde{\gamma})(\tilde{D}) = c(\gamma)$. As a consequence, $\tilde{d}_{D_0,m}(\tilde{D}) = d_{D_0} \circ Q_{m,n}(\tilde{D})$. Since this is true for any $\tilde{D}$ in $\mathbb{R}^{2m+n}$, we have $\tilde{d}_{D_0,m} = d_{D_0} \circ Q_{m,n}$ in $\mathbb{R}^{2m+n}$. Therefore, $d_{D_0} \circ Q_{m,n}$ is C^∞ in $\mathbb{R}^{2m+n}$. □

It seems that this lemma could directly prove Theorem 2.30, but it does not: Lemma 2.31 says that for each m, there is a generic subspace of $\mathbb{R}^{2m+n}$ where $d_{D_0} \circ Q_{m,n}$ is C^∞, but these generic spaces may not be equal for different values of m. In fact, recall Definition 2.5 on the differentiability of maps $\text{Bar} \to \mathbb{R}$. The definition imposes that for d_{D_0} being ∞-differentiable at some $D \in \text{Bar}_n$, we need $d_{D_0} \circ Q_{m,n}$ to be simultaneously C^∞ in open neighborhoods of all the pre-images of D via $Q_{m,n}$, for all possible m. Therefore, we would need to evaluate in some way all the generic spaces of $\mathbb{R}^{2m+n}$ for any m that Lemma 2.31 refers to, and check that their intersection is in fact generic. But this is not straightorward: the intersection of an infinite number of spaces generic in some space S is not necessarily generic in S. Therefore, we employ a different approach. We first define the space $\widehat{\text{Bar}}$ as the set of barcodes $D \in \text{Bar}_n$ such that no point of D_0 is at distance $d_\infty(D, D_0)$ to its diagonal projection. This space has two key properties, which are given by the next two lemmas.

Lemma 2.32. *The space* $\widehat{\text{Bar}}$ *is generic in* Bar_n.

Proof. First, notice that in this context a space is generic in Bar_n when it is open with respect to the bottleneck topology, and dense with respect to the bottleneck distance. Since D_0 has a finite number m of off-diagonal intervals, we have a finite number of values d_i, $i = 1, \dots, m$ that are forbidden for $d_\infty(D, D_0)$. For each d_i, denote by $\widehat{\text{Bar}}_i$ the space of barcodes $D \in \text{Bar}_n$ such that $d_\infty(D, D_0) \neq d_i$. Then, $\widehat{\text{Bar}} = \cap_i \widehat{\text{Bar}}_i$. In addition, we denote by $\overline{\mathcal{B}}(D_0, d_i)$ the closed ball (with the bottleneck distance) centered at D_0 and of radius d_i, and by $\partial\overline{\mathcal{B}}(D_0, d_i)$ its boundary. In particular, $\partial\overline{\mathcal{B}}(D_0, d_i)$ is closed, hence its complement is open. Furthermore, its complement is also dense. In fact, given $D \in \partial\overline{\mathcal{B}}(D_0, d_i)$, we can find a barcode D' arbitrarily close to D such that $d_\infty(D', D_0) \neq d_i$. As a consequence, the complement of $\partial\overline{\mathcal{B}}(D_0, d_i)$ is generic. Since $\widehat{\text{Bar}}_i = \text{Bar}_n \setminus \partial\overline{\mathcal{B}}(D_0, d_i)$, we infer that $\widehat{\text{Bar}}_i$ is generic for each i. Since the finite intersection of generic spaces is generic, $\widehat{\text{Bar}}$ is generic. □

We introduce now the multi-set $\Delta_\epsilon := \{(b, d) \in \mathbb{R}^2 \mid \frac{|b-d|}{2} < \epsilon\}$, i.e., the space of points that are ϵ-close to the diagonal $\{(b, b) \in \mathbb{R}^2 \mid b \in \mathbb{R}\}$, with the bottleneck distance. The second property of $\widehat{\text{Bar}}$ is the following.

Lemma 2.33. *If* $D \in \widehat{\text{Bar}}$, *then there exists* $\epsilon > 0$ *such that, for all* D' ϵ*-close to* D, $d_\infty(D', D_0) > \epsilon$ *and there exists an optimal matching between* D' *and* D_0 *sending* $D' \cap \Delta_\epsilon$ *onto* Δ^∞.

Proof. The proof can be found in [10]. Due to the page limit, we have not added the proof here. However, we have included a detailed proof of the lemma, including a sketch of the demonstration, in Appendix B.1, Lemma B.3. □

This lemma is saying that, for the chosen ϵ and for any $D' \in \mathcal{B}(D, \epsilon)$, $d_\infty(D', D_0)$ does not depend on the points of D' that are in Δ_ϵ. We now define the concept of *minimal barcode*:

an ordered barcode $\tilde{D} = [(b_i, d_i)_{i=1}^m, (v_j)_{j=1}^n]$ is minimal if $b_i \neq d_i$ for all $i = 1, \dots m$. The meaning of this concept is that if $D := Q_{m,n}(\tilde{D})$, then any other ordered barcode representing D is in a space $\mathbb{R}^{2m'+n}$ with $m' \geq m$, and $\tilde{D}$ lies in the smallest possible space for representing D. Using this concept, we will now prove that given a barcode D and under specific conditions, the differentiability of $d_{D_0} \circ Q_{m,n}$ at a minimal pre-image induces the differentiability of the maps $d_{D_0} \circ Q_{m',n}$ at all other possible pre-images of D. Notice in addition that a direct consequence of the definition of minimal barcode is that, for each $m \in \mathbb{N}$, the space of minimal barcodes is open.

Lemma 2.34. *Let $\tilde{D}_m \in \mathbb{R}^{2m+n}$ be a minimal barcode such that $D := Q_{m,n}(\tilde{D_m}) \in \widehat{\text{Bar}}$, and assume that $d_{D_0} \circ Q_{m,n}$ is C^∞ in an open neighborhood U of $\tilde{D_m}$. Then there exists $\epsilon > 0$ such that, for any other pre-image $\tilde{D}_{m'}$ of D, the map $d_{D_0} \circ Q_{m',n}$ is C^∞ in $B(\tilde{D}_{m'}, \epsilon)$.*

Proof. The proof can be found in [10] and it is not given here due to the page limit. □

A direct consequence of Lemma 2.34 is that we can deduce *at once* the simultaneous differentiablility of all the maps $d_{D_0} \circ Q_{m',n}$ over open balls *of the same size*. However, for the proof of Theorem 2.30, we need to refer to differentiability in a neighborhood of the barcode D, hence we need a last result connecting an open neighborhood of D in Bar_n to these open balls.

Lemma 2.35. *For all $D \in \text{Bar}_n$, there exists $\epsilon > 0$ such that for any $m' \in \mathbb{N}$, we have*

$$Q_{m',n}^{-1}(B(D, \epsilon)) \subseteq \bigcup_{\tilde{D}_{m'} \in \mathbb{R}^{2m'+n};\, Q_{m',n}(\tilde{D}_{m'}) = D} \mathcal{B}(\tilde{D}_{m'}, \epsilon). \tag{2.37}$$

Proof. The proof can be found in [10]. □

Now that we have all necessary results, we can proceed to prove Theorem 2.30.

Proof of Theorem 2.30. Let $\mathcal{W}$ be the set of barcodes in Bar_n admitting an open neighborhood where d_{D_0} is ∞-differentiable. The goal of this proof is to show that 1) $\mathcal{W}$ is open in Bar_n, and 2) $\mathcal{W}$ is dense. 1) is true by definition of $\mathcal{W}$. One way of showing 2) is taking any $D \in \text{Bar}_n$ and showing that we can perform a finite number of infinitesimal —i.e., arbitrarily small— perturbations of D such that the new (after perturbations) D admits an open neighborhood over which d_{D_0} is ∞-differentiable, i.e., $D \in \mathcal{W}$. For showing this, we proceed as follows.

Let $D \in \text{Bar}_n$. Since $\widehat{\text{Bar}}$ is generic in Bar_n, we can perform an infinitesimal perturbation to D such that D is in $\widehat{\text{Bar}}$ —more specifically, we can replace D by another $D' \in \widehat{\text{Bar}}$ with $d_\infty(D, D') = \epsilon_1$ with $\epsilon_1 > 0$ arbitrarily small, and for convenience we denote D' as D.

Let now $\tilde{D}_m \in \mathbb{R}^{2m+n}$ be a minimal pre-image of D. We have seen that the set of minimal barcodes is open, and by Lemma 2.31, $d_{D_0} \circ Q_{m,n}$ is C^∞ on a generic subset of $\mathbb{R}^{2m+n}$. So up to an infinitesimal perturbation of $\tilde{D}_m$ (which induces an infinitesimal perturbation of D in Bar_n since $Q_{m,n}$ is 1-Lipschitz) we can assume that $\tilde{D}_m$ is in that generic space, so $d_{D_0} \circ Q_{m,n}$ is C^∞ on $\mathcal{B}(\tilde{D}_m, \epsilon)$ for some $\epsilon > 0$, while keeping $\tilde{D}_m$ minimal. In addition, since $\widehat{\text{Bar}}$ is open and $Q_{m,n}$ is 1-Lipschitz, we can make the perturbation sufficiently small such that the new D stays in $\widehat{\text{Bar}}$. Therefore, we now have $D \in \widehat{\text{Bar}}$, $\tilde{D}_m$ minimal, and $d_{D_0} \circ Q_{m,n}$ is C^∞ on $\mathcal{B}(\tilde{D}_m, \epsilon)$. Using now Lemma 2.34 and reducing ϵ further if necessary according to the lemma, we have that all the maps $d_{D_0} \circ Q_{m',n}$ are C^∞ over the balls $\mathcal{B}(\tilde{D}_{m'}, \epsilon)$, for all

the pre-images $D_{m'}$ of D via the maps $Q_{m',n}$ (for all $m' \in \mathbb{N}$). Now, reducing ϵ if necessary, by Lemma 2.35, we have that d_{D_0} is ∞-differentiable over $\mathcal{B}(D, \epsilon)$.

Although this conclusion comes directly from the definitions of ∞-differentiability and Lemma 2.35, we can prove it in more detail for clarity. Given any $D^* \in \mathcal{B}(D, \epsilon)$, we want to see that d_{D_0} is ∞-differentiable at D^*. Since all the pre-images of D^* lie in $\cup_{m' \in \mathbb{N}} Q^{-1}_{m',n}(\mathcal{B}(D, \epsilon))$, they also lie in the union of ϵ-balls given in the right-hand side of equation (2.37). But from the choice of ϵ, all the maps $d_{D_0} \circ Q_{m',n}$ are C^∞ in the set of the right-hand side of equation (2.37) (with the values of m' correctly chosen for each ball). As a consequence, all the pre-images of D^* admit open neighborhoods where $d_{D_0} \circ Q_{m',n}$ —for the right m'— is C^∞. So by the definition of r-differentiability, d_{D_0} is ∞-differentiable at D^*. Since the proof holds for any $D^* \in \mathcal{B}(D, \epsilon)$, d_{D_0} is ∞-differentiable at $\mathcal{B}(D, \epsilon)$. Therefore $D \in \mathcal{W}$. Recall that this new D only differs from the original D by two infinitesimal perturbations, hence $\mathcal{W}$ is dense in Bar_n. □

2.3.6 Conditions for smoothness of the bottleneck distance

Theorem 2.30 ensures that there is a generic subset of Bar_n where d_{D_0} is ∞-differentiable. However, in practical applications we do not only compute d_{D_0} from the space of barcodes, but rather the composition $d_{D_0} \circ B_p$ that maps a point cloud in $\mathbb{R}^{kd}$ to its Rips filtration, then to its persistence diagram D, and then to $d_{D_0}(D)$. Therefore, for computational applications we are interested on the smoothness of the composite $d_{D_0} \circ B_p$, rather than only d_{D_0}. This section gives a simple numerical condition that, if satisfied, ensures that $d_{D_0} \circ B_p$ is C^∞. First, we summarize the main conclusions from the previous sections:

1) from Theorem 2.11, the barcode generator $B_p \colon \mathbb{R}^{kd} \to \mathrm{Bar}_n$ is ∞-differentiable in the space of point clouds in general position $\tilde{\mathcal{P}}$, which is generic in $\mathbb{R}^{kd}$, and

2) from Theorem 2.30, the bottleneck distance to a fixed diagram D_0, $d_{D_0} \colon \mathrm{Bar}_n \to \mathbb{R}$, is ∞-differentiable in $\mathcal{W}$, where $\mathcal{W}$ is defined as

$$\mathcal{W} = \widehat{\mathrm{Bar}} \cap \{D \in \mathrm{Bar}_n \mid \exists \text{ minimal } \tilde{D} \in \mathbb{R}^{2m+n} \text{ such that } \tilde{d}_{D_0,m} \text{ is } C^\infty \text{ at } \tilde{D}\}. \quad (2.38)$$

Remark: Note that when the barcode distance is attained simultaneously by different pairs of points $p \in D, p_0 \in D_0$, all with same length, then it may not be differentiable. For instance, if $d_\infty(D, D_0)$ is given by $\|p - p_0\|_\infty$ and also by $\|p' - p'_0\|_\infty$ using two different optimal matchings, then locally around D, we do not have a well defined gradient. The reason is that if we keep p constant and move p' away from p'_0, then d_∞ is given by $\|p - p_0\|_\infty$. However, if we move p away from p_0 and make p' closer to p'_0, then d_∞ is given by $\|p' - p'_0\|_\infty$. Therefore, if we approach D by different paths/directions, we get different functions d_∞ that are only equal at D, but different in a neighborhood. So it seems that d_∞ is not differentiable in D. The solution we propose is showing that if all optimal matchings use a unique pair $p \in D, p_0 \in D_0$, then d_∞ is locally a smooth function around D.

If we could computationally check whether a barcode D is in $\mathcal{W}$, then we would be done. However, $\mathcal{W}$ is a-priori a very complex space and it is not straightforward to know if a barcode belongs to it. Hence, we will take a simpler space $\mathcal{X} \subseteq \mathrm{Bar}_n$, defined such that it is easy to check, for any barcode D, if D is in $\mathcal{X}$. In addition, we will see that the conditions

of a barcode D to be in $\mathcal{X}$ are computationally mild and likely to be almost always satisfied. The main result of this section is Lemma 2.37, where we show that $\mathcal{X} \subseteq \mathcal{W}$. This is then used in Corollary 2.38 for proving that $d_{D_0} \circ B_p$ is C^∞ when $P \in \tilde{\mathcal{P}}$ and $D \in \mathcal{X}$.

Definition 2.36. Given $D \in \mathrm{Bar}_n$, let Γ be the set of optimal matchings between D and D_0. Then, the space $\mathcal{X} \subseteq \mathrm{Bar}_n$ is defined as follows

$$\begin{aligned}\mathcal{X} \coloneqq \widehat{\mathrm{Bar}} \cap \{D \in \mathrm{Bar}_n \mid \exists\, p \in D, p_0 \in D_0 \colon \forall \gamma \in \Gamma,\ c(\gamma) = \|p - p_0\|_\infty \\ \wedge \|p - p_0\|_\infty > \|p' - \gamma(p')\|_\infty \forall p' \in D \setminus \{p\} \wedge (p - p_0) \nparallel (1,1)\}.\end{aligned} \tag{2.39}$$

Although the expression of $\mathcal{X}$ in Definition 2.36 is seemingly complex, this space is quite simple: $\mathcal{X}$ is the set of barcodes in $\widehat{\mathrm{Bar}}$ such that all optimal matchings have a cost uniquely given by a single pair $p \in D, p_0 \in D_0$, and such that these two points are not aligned parallel to the diagonal. Note that this second condition is given by the condition $(p - p_0) \nparallel (1,1)$, and is needed for the differentiability of the supremum norm, as we will at the end of the proof of the next lemma. Therefore, it is easy to know if a barcode belongs to $\mathcal{X}$. In addition, the definition of $\mathcal{X}$ implies that for all $D \in \mathcal{X}$, and for all optimal matchings γ between D and D_0, there is an ordering of the lengths of all the matches made by γ of the following form:

$$\|p - \gamma(p)\|_\infty > \|p_1 - \gamma(p_1)\|_\infty \geq \|p_2 - \gamma(p_2)\|_\infty \geq \dots \tag{2.40}$$

where $p, p_1, p_2, \dots$ are different points of D.

Lemma 2.37. *The space $\mathcal{X}$ is contained in $\mathcal{W}$.*

Proof. Let $D \in \mathcal{X}$, and let $\tilde{D} \in \mathbb{R}^{2m+n}$ be a minimal pre-image of D. Recall the definition of $\tilde{d}_{D_0,m}$ in Lemma 2.31 using equations (2.35) and (2.36). We want to show that $\tilde{d}_{D_0,m}$ is C^∞ in $\tilde{D}$ using these definitions. First, notice that we only have to look at the optimal matchings between $\tilde{D}$ and D_0 for computing $\tilde{d}_{D_0,m}(D)$. From the definition of $\mathcal{X}$, the minimum computed in (2.36) is attained by a unique pair $p \in D, p_0 \in D_0$ for all optimal matchings. Therefore, $\tilde{d}_{D_0,m}(D) = \|p - p_0\|_\infty > 0$. The strict positivity comes because $D \in \widehat{\mathrm{Bar}}$, so $D \neq D_0$. In addition, since $D \in \widehat{\mathrm{Bar}}$, $p \notin \Delta^\infty$. Now, using $\tilde{D} = (x_1, x_2, \dots, x_{2m}, y_1, \dots, y_n)$, we infer that p is given by some components x_{2i+1}, x_{2i+2} of $\tilde{D}$ (with $0 \leq i \leq m-1$).

Recall that $\tilde{\Gamma}_m$ is the finite and non-empty set of ordered matchings as defined in Lemma 2.31. Take any matching with minimal cost $\tilde{\gamma} \in \tilde{\Gamma}_m$. Then its cost is given by $\|(x_{2i+1}, x_{2i+2}) - p_0\|_\infty$. In addition, from the definition of $\mathcal{X}$, this cost is strictly larger than all the other lengths of matches made by $\tilde{\gamma}$ (which are the lengths inside the max in (2.35)). More formally, the lengths of all the matches induced by $\tilde{\gamma}$ are a finite set of N elements of the form

$$\{C_j(x_{2j+1}, x_{2j+2});\ j = 1, \dots, m-1\} \cup \left\{ \frac{|d_0 - b_0|}{2} \right\}_{(b_0,d_0) \in D_0(\tilde{\gamma})}, \tag{2.41}$$

where each C_j is equal to either $\|(x_{2j+1}, x_{2j+2}) - (a_j, b_j)\|_\infty$ for some $(a_j, b_j) \in D_0 \setminus \Delta^\infty$, or $|x_{2j+1} - x_{2j+2}|/2$ when $\tilde{\gamma}$ maps it to the diagonal. Recall that $D_0(\tilde{\gamma})$ are the off-diagonal points of D_0 that are not mapped to off-diagonal points of $\tilde{D}$, and thus they are mapped to the diagonal. For convenience, we generalize the elements of (2.41) to maps $C_j \colon \mathbb{R}^{2m} \to \mathbb{R}$, $j = 1, \dots, N$ that take the coordinates of ordered barcodes as inputs. In addition, these maps are continuous on $\mathbb{R}^{2m}$. From the definition of $\mathcal{X}$, in $\tilde{D}$ we have a strict ordering

$$C_{i_1}(\tilde{D}) > C_{i_2}(\tilde{D}) \geq C_{i_3}(\tilde{D}) \geq 0 \geq C_{i_N}(\tilde{D}), \tag{2.42}$$

where the indexes $i_1, i_2, \ldots, i_N$ are a re-ordering of $\{1, \ldots, N\}$. From the definition of $\mathcal{X}$, i_1 is the same function for all optimal ordered matchings, but the order of the remaining $N-1$ functions depends on each matching. From the continuity of these maps, and recalling that $\mathbb{D}$ is a point in $\mathbb{R}^{2m+n}$, there is some $\epsilon > 0$ such that for any $\tilde{D}' \in B_\epsilon(\tilde{D})$, $C_{i_1}(\tilde{D}')$ is strictly larger than the other $N-1$ functions $C_j(\tilde{D}')$, $j \neq i_1$. As a consequence, for all $\tilde{D}' \in B_\epsilon(\tilde{D})$, $\tilde{c}(\tilde{\gamma})(\tilde{D}') = C_{i_1}(\tilde{D}')$.

Using the same argument with all the other optimal ordered matchings, and using the fact that there is a finite number, we obtain an $\epsilon > 0$ such that for all optimal ordered matchings $\tilde{\gamma}$, its cost is the function $C_{i_1}(\tilde{D}')$ for all $\tilde{D}' \in B_\epsilon(\tilde{D})$. In addition, from the definition of $\mathcal{X}$, the function C_{i_1} is the same for all optimal matchings. We thus denote it by C for convenience, and it can take the two following forms, depending on where the point p_0 in D_0 that is used for $d_\infty(D, D_0)$ lies (recall the definition of $\mathcal{X}$).

(i) If p_0 is the projection of $p \in D$ onto the diagonal, then $C_{\tilde{\gamma}} = |x_{2i+1} - x_{2i+2}|/2$ for some $0 \leq i \leq m-1$ independent of the optimal ordered matching.

(ii) If $p_0 = (p_{0,1}, p_{0,2})$ is an off-diagonal point, then $C = \|(x_{2i+1}, x_{2i+2}) - (p_{0,1}, p_{0,2})\|_\infty$ for all optimal ordered matchings.

In summary, we have proved that for any optimal ordered matching $\tilde{\gamma} \in \tilde{\Gamma}_m$, and for any $\tilde{D}' \in B_\epsilon(\tilde{D})$, we have $\tilde{c}(\gamma)(\tilde{D}') = C(x_{2i+1}, x_{2i+2})$, a continuous function that is the same for all optimal ordered matchings. In addition, we have shown that C is either of type (i) or (ii) in all $B_\epsilon(\tilde{D})$. What remains now to prove is that an optimal ordered matching in $\tilde{D}$ remains an optimal ordered matching in an open neighborhood of $\tilde{D}$.

The proof is simple but requires some formalism with the notation. We have a finite number of non-optimal ordered matchings in $\tilde{\Gamma}_m$. Take one of these non-optimal matchings $\tilde{\gamma}'$. It has a cost that is given by the length of some of the pairings it creates. Let k be the number of different pairings that yield the same equal maximal length that gives the cost $\tilde{c}(\tilde{\gamma}')$. In particular k is finite. Each of these k lengths is a function of a pair of coordinates x_{2j_k}, x_{2j_k+1}, and we denote it as $Q_k(x_{2j_k}, x_{2j_k+1})$. In particular, these k functions Q_k are equal on $\tilde{D}$, and strictly larger than the length of the other pairings induced by $\tilde{\gamma}'$. Therefore, in a sufficiently small neighborhood of $\tilde{D}$, the cost of $\tilde{\gamma}_k$ is given by the maximum of these k functions. In addition, because $\tilde{\gamma}_k$ is non-optimal, at $\tilde{D}$ we have $C(\tilde{D}) < Q_k(\tilde{D})$ for all k. Hence, we have an open neighborhood U of $\tilde{D}$ where $C(\tilde{D}') < Q_k(\tilde{D}') \leq \tilde{c}(\tilde{\gamma}')(\tilde{D}')$, for all $\tilde{D}' \in U$. We repeat the same argument for all non-optimal ordered matchings in $\tilde{\Gamma}_m$, and we obtain an $\epsilon' > 0$ with $\epsilon' < \epsilon$ such that, for all $\tilde{D}' \in B_{\epsilon'}(\tilde{D})$ and for all non-optimal $\tilde{\gamma}'$, we have $C(\tilde{D}') < \tilde{c}(\tilde{\gamma}')(\tilde{D}')$. As a consequence, for all $\tilde{D}' \in B_{\epsilon'}(\tilde{D})$, the minimum of the cost of ordered matchings is $C(\tilde{D}')$. Hence, $\tilde{d}_{D_0,m}(\tilde{D}') = C(\tilde{D}')$ for all $B_{\epsilon'}(\tilde{D})$. Finally, recall that $\tilde{D} \neq D_0$ so $C(\tilde{D}) > 0$. Therefore, if C is of type (i), then in an open neighborhood of $\tilde{D}$, $\tilde{d}_{D_0,m} = \alpha \cdot (x_{2i+1} - x_{2i+2})/2$ where $\alpha = 1$ or -1, hence it is C^∞ on $\tilde{D}$. If C is of type (ii), then $\tilde{d}_{D_0,m} = \max(|x_{2i+1} - p_{0,1}|, |x_{2i+2} - p_{0,2}|)$. But from the definition of $\mathcal{X}$, we have that $(p - p_0) \nparallel (1,1)$, or, equivalently, $|x_{2i+1} - p_{0,1}| > |x_{2i+2} - p_{0,2}|$ (or vice-versa) in $\tilde{D}$. Thus, there is an open neighborhood of $\tilde{D}$ where $\tilde{d}_{D_0,m}$ only depends on one of the two terms, so it is C^∞ in $\tilde{D}$. In conclusion, we have proved that if $D \in \mathcal{X}$, then $\tilde{d}_{D_0,m}$ is C^∞ at all minimal pre-images of D, so $D \in \mathcal{W}$. □

Corollary 2.38. *Let $P \in \tilde{\mathcal{P}} \subseteq \mathbb{R}^{kd}$ and $D = \mathrm{Dgm}_p(F(P))$, where F is the Rips filtration. If $D \in \mathcal{X}$, then $d_{D_0} \circ B_p$ is C^∞ in P.*

Proof. From Lemma 2.37, $D \in \mathcal{W}$, so d_{D_0} is ∞-differentiable at $D = B_p(P)$. Since $P \in \tilde{\mathcal{P}}$, from Theorem 2.11 we infer that B_p is ∞-differentiable at P. From Proposition 2.6, the composite of two ∞-differentiable functions is C^∞ as a map between manifolds, so $d_{D_0} \circ B_p$ is C^∞ at P. □

2.4 Selective regularizers

The *persistence* of a point (b, d) in a persistence diagram is defined as the value $|d - b|$. In many cases it can be useful to work with maps on barcodes that only take into account points with persistence larger than some threshold $\delta > 0$. For instance, when the function is not differentiable in the diagonal, or when the function is not zero in the diagonal, but also in cases where one wants to avoid computational costs and noise. In this section, we study these functions and prove a new result, given in Proposition 2.41, related to the differentiability of this type of functions. We only work in this section with Bar_0, i.e., the diagrams with no unbounded points —if the diagrams are of degree 0, we can neglect the point $(0, +\infty)$.

Definition 2.39. Given $\delta > 0$, a *δ-selective* function is a function $f_\delta \colon \mathrm{Bar}_0 \to \mathbb{R}$ of the form

$$f_\delta(D) = \sum_{(b,d)\in D\colon |d-b|>\delta} F(b, d) \tag{2.43}$$

where $F : \mathbb{R}^2 \to \mathbb{R}$ is C^∞ in $\{(x, y) \colon |x - y| > \epsilon\}$ for some $\epsilon < \delta$.

Proposition 2.40. *Any δ-selective function is generically ∞-differentiable in* Bar_0.

Proof. Denote by Δ^*_δ the set of points in $\mathbb{R}^2$ with persistence δ, or equivalently, at a distance $\delta/2$ to the diagonal (i.e., $\Delta^*_\delta = \{(x, x+\delta) \mid x \in \mathbb{R}\} \cup \{(x, x-\delta) \mid x \in \mathbb{R}\}$), and let $C_\delta \subseteq \mathrm{Bar}_0$ be the set of barcodes with no points in Δ^*_δ, which is generic in Bar_0 (it is open and dense). Now, let f_δ be any δ-selective function, which uses some function $F \colon \mathbb{R}^2 \to \mathbb{R}$ in the summation. We define the function

$$\mathrm{act}(b, d) \coloneqq \begin{cases} 1 \colon |d - b| > \delta \\ 0 \colon |d - b| \leq \delta \end{cases} \tag{2.44}$$

and we define the function $G(b, d) \coloneqq F(b, d) \cdot \mathrm{act}(b, d)$, which is C^∞ in $\mathbb{R}^2 \setminus \Delta^*_\delta$ and equal to 0 in the diagonal. The function f_δ can be re-written as:

$$f_\delta(D) = \sum_{(b,d)\in D} G(b, d). \tag{2.45}$$

Let now D be any barcode in C_δ and $\tilde{D}$ be any pre-image of D, with $Q_{m,0}(\tilde{D}) = D$ for some m. Then the linear representation $\tilde{f}_\delta$ of f_δ in an open neighborhood of $\tilde{D}$, given by

$$(b_1, d_1, \ldots, b_m, d_m) \in \mathbb{R}^{2m} \mapsto \tilde{f}_\delta(b_1, d_1, \ldots, b_m, d_m) = \sum_i G(b_i, d_i) \tag{2.46}$$

is C^∞ in $\tilde{D}$, since all the points (b_i, d_i) are in $\mathbb{R}^2 \setminus \Delta^*_\delta$. As a consequence f_δ is ∞-differentiable in C_δ, which is generic in Bar_0. □

Proposition 2.41. *Let f_δ be a δ-selective function, $\mathbb{R}^{nd}$ the space of point clouds, and let B_p be the barcode generator of degree p (via the Rips filtration) for any $p \geq 0$. Then, the composition $f_\delta \circ B_p$ is generically C^∞ in $\mathbb{R}^{nd}$. More specifically, $f_\delta \circ B_p$ is C^∞ in the subspace of $\mathbb{R}^{nd}$*

$$\mathcal{Y} \coloneqq \tilde{\mathcal{P}} \cap \{P \in \mathbb{R}^{nd} \colon \; |\|p_i - p_j\|_2 - \|p_k - p_l\|_2| \neq \delta \text{ for all } 1 \leq i,j,k,l \leq n\}, \tag{2.47}$$

where we use the notation $P = (p_1, \ldots, p_n)$. In addition, $\mathcal{Y}$ is generic in $\mathbb{R}^{nd}$.

Proof. Let $P = (p_1, \ldots, p_n)$ be a point cloud in $\mathcal{Y}$. If its barcode $B_p(P)$ had a point in Δ^*_δ, then it would have a point of the form $(x, x \pm \delta)$. However, the points of the barcode are of the form $(\|p_i - p_j\|_2, \|p_k - p_l\|_2)$ for some i,j,k,l, which would imply that $|\|p_i - p_j\|_2 - \|p_k - p_l\|_2| = \delta$, a contradiction. As a consequence, the barcode of P does not have points in Δ^*_δ, so from Proposition 2.40, f_δ is ∞-differentiable in $B_p(P)$. Using the fact that B_p is ∞-differentiable in P (since $\mathcal{Y} \subseteq \tilde{\mathcal{P}}$), and applying Proposition 2.6, it follows that $f_\delta \circ B_p$ is C^∞ in P. To prove that $\mathcal{Y}$ is generic in $\mathbb{R}^{nd}$, we re-express the set as

$$\mathcal{Y} = \tilde{\mathcal{P}} \cap \bigcap_{i,j,k,l} Z_{ijkl} \tag{2.48}$$

where $Z_{ijkl} \coloneqq \{P \in \mathbb{R}^{nd} \colon \; |\|p_i - p_j\|_2 - \|p_k - p_l\|_2| \neq \delta\}$, and the values i,j,k,l are taken over all possible 4-vectors $(i,j,k,l) \in \{1, \ldots, n\}^4$. We will now prove that each Z_{ijkl} is generic in $\mathbb{R}^{nd}$. Fix a 4-vector $(i,j,k,l) \in \{1, \ldots, n\}^4$ and consider Z_{ijkl}. We define the function $f_{ijkl}(P) \coloneqq |\|p_i - p_j\|_2 - \|p_k - p_l\|_2| - \delta$, which is continuous in $\mathbb{R}^{nd}$. Since $\mathbb{R} \setminus \{0\}$ is open, the pre-image $f^{-1}_{ijkl}(\mathbb{R} \setminus \{0\})$ is open in $\mathbb{R}^{nd}$, so $Z_{ijkl} = f^{-1}_{ijkl}(\mathbb{R} \setminus \{0\})$ is open in $\mathbb{R}^{nd}$. To show that Z_{ijkl} is dense, take any $P \notin Z_{ijkl}$, and consider any value $\epsilon > 0$. We have $f_{ijkl}(P) = 0$, which implies that in (i,j,k,l) there is one index that is not repeated. In fact, if we assume that all indices in (i,j,k,l) are repeated, we either have $i = j = k = l$, or $i = j$ and $k = l \neq i$, or $i = k$ and $j = l$ (and $i \neq j$) (up to permutation of the indices, these are all possible cases). In all these cases, we have $f_{ijkl}(P) = -\delta \neq 0$, which is a contradiction. As a consequence, there is one index, say i, that is different from the other three indexes j,k,l. In particular, we can transform p_i into some $p'_i \neq p_i$ with $\|p_i - p'_i\|_2 < \epsilon$ and such that $\|p'_i - p_j\|_2 \neq \|p_i - p_j\|_2$. From the choice of index, this change does not affect the second term $\|p_j - p_k\|_2$, so, denoting by P' the new point $(p_1, \ldots, p'_i, \ldots, p_n)$, we have $f_{ijkl}(P') \neq 0$. In other words, we have a point $P' \in Z_{ijkl}$ with $\|P - P'\|_2 < \epsilon$, so Z_{ijkl} is dense in $\mathbb{R}^{nd}$, and in particular generic. Since a finite intersection of generic spaces is generic, we have that $\mathcal{Y}$ is generic in $\mathbb{R}^{nd}$. □

In other words, Proposition 2.41 states that any regularizer that "only looks at points in the diagram with persistence larger than some δ" is C^∞ in an open and dense subset of the space of point clouds. This has many applications, such as allowing us to use gradient descent optimization on functions through barcode space even when they take zero values in the diagonal or when they are not differentiable in the diagonal. One important implication of this result is related to regularizers based on persistent entropy.

Definition 2.42. Let $\mathbb{R}^{nd}$ be the space of point clouds. Consider the persistent entropy ϵ as defined in Definition 2.30, and let B_p be the barcode generator of degree p via the Rips

parametrization. Given any (arbitrarily small) $\delta > 0$, the *δ-selective p-persistent entropy generator* $G_{p,\delta}$ is defined as

$$\begin{aligned} G_{p,\delta} &\colon \mathbb{R}^{nd} \to \mathbb{R} \\ P &\mapsto \epsilon(B_p(P) \setminus \bar{\Delta}_{\delta/2}) = -\sum_{(b,d)\in B_p(P)\colon |d-b|>\delta} \frac{|d-b|}{L} \log\left(\frac{|d-b|}{L}\right), \end{aligned} \tag{2.49}$$

where $L = \sum_{(b,d)\in B_p(P)\colon |d-b|>\delta} |d-b|$, and $\bar{\Delta}_{\delta/2} = \{(b,d) \in \mathbb{R}^2 \mid |d-b| \leq \delta\}$ is the set of points with persistence smaller or equal than δ. Equivalently, it is the set of points that are at a distance smaller or equal than $\delta/2$ to the diagonal. Therefore, $G_{p,\delta}$ computes the persistent entropy of $B_p(P)$ without considering the points with persistence lower than δ.

Proposition 2.43. *Let $\mathbb{R}^{nd}$ be the space of point clouds. For any $p \in \mathbb{N}$ and $\delta > 0$, the δ-selective p-persistent entropy generator $G_{p,\delta}\colon \mathbb{R}^{nd} \to \mathbb{R}$ is generically C^∞ in $\mathbb{R}^{nd}$. In particular, $G_{p,\delta}$ is C^∞ in $\mathcal{Y}$, which is generic in $\mathbb{R}^{nd}$.*

Proof. The map $G_{p,\delta}$ can be written as $f_\delta \circ B_p$, where

$$f_\delta(D) = -\sum_{(b,d)\in D\colon |d-b|>\delta} \frac{|d-b|}{L(D)} \log\left(\frac{|d-b|}{L(D)}\right), \tag{2.50}$$

where $L(D) = \sum_{(b,d)\in D\colon |d-b|>\delta} |d-b|$. Assume D is a barcode in C_δ, and let $\tilde{D}$ be any pre-image of D with $Q_{m,0}(\tilde{D}) = D$ for some m. Then, the linear representation of f_δ in an open neighborhood of $\tilde{D}$ is given by

$$(b_1, \ldots, d_m) \in \mathbb{R}^{2m} \mapsto \tilde{f}_\delta((b_1, \ldots, d_m)) = H \circ G(b_1, \ldots, d_m), \tag{2.51}$$

where the functions G and H are defined as follows. The map G is given by

$$G\colon (b_1, \ldots, d_m) \mapsto (b_1, \ldots, d_m, \sum_{i=1}^{m} |d_i - b_i| \cdot \mathrm{act}(b_i, d_i)), \tag{2.52}$$

and we denote the last element returned by G as $L(\tilde{D})$. Since D is in C_δ, for all points of $\tilde{D}$ we have $|d_i - b_i| \neq \delta$, so on an open neighborhood U of $\tilde{D}$, all inequalities $|d_i - b_i| > \delta$ (or $|d_i - b_i| < \delta$) do not change. As a consequence, the act function is locally 0 or 1 for each (b_i, d_i) in U, and the absolute value $|d_i - b_i|$ becomes either $d_i - b_i$ or $b_i - d_i$ in U, so G becomes a linear function in U, so G is C^∞ in an open neighborhood of $\tilde{D}$. Then, we compose G with the map

$$H(b_1, \ldots, d_m, L) = -\sum_{i=1}^{m} \frac{|d_i - b_i|}{L} \log\left(\frac{|d_i - b_i|}{L}\right) \cdot \mathrm{act}(b_i, d_i), \tag{2.53}$$

which is C^∞ in an open neighborhood of $(b_1, \ldots, d_m, L)$ whenever $L > 0$ and $|d_i - b_i| \neq \delta$ for all i. If $L = 0$, then there are no points in U that satisfy $|d_i - b_i| > \delta$, so $\tilde{f}_\delta = 0$ in U, hence the map is C^∞ in $\tilde{D}$. If $L \neq 0$ (and as a consequence $L > 0$), then by the composition of C^∞ maps, we have that f_δ is C^∞ in $\tilde{D}$. This holds for an arbitrary pre-image of D, so f_δ is ∞-differentiable in C_δ. Now, assume P is a point cloud in $\mathcal{Y} \subseteq \mathbb{R}^{nd}$. Then, B_p is ∞-differentiable in P and $B_p(P) \in C_\delta$ (by definition of $\mathcal{Y}$), so f_δ is ∞-differentiable in $B_p(P)$. From the chain rule through barcode space (Proposition 2.6), the composition $f_\delta \circ B_p = G_{p,\delta}$ is C^∞ in P. Hence, $G_{p,\delta}$ is C^∞ in $\mathcal{Y}$, which by Proposition 2.41 is generic in $\mathbb{R}^{nd}$. □

Chapter 3

Topology-informed generative models

The construction of the variational autoencoder and the explanation of its working principle in Subsection 3.4.2 rely on the paper that first proposed variational autoencoders [24] and the lecture notes [25]. The rest of the chapter is original work.

3.1 Topological regularizers

In Chapter 2, we have obtained several results related to the differentiability of different classes of functions from the space of point clouds to $\mathbb{R}$, factoring through barcode space. We will now define five types of topological regularizers from the space of point clouds $\mathbb{R}^{nd}$ to $\mathbb{R}$ —four of them comparing the persistence diagram of a point cloud to a reference persistence diagram—, which we will use in the following experiments. In order to define these functions, it is important to recall the functions introduced in the previous chapters; notably, if p is a homology degree, the map B_p is the barcode generator of degree p via the Rips parametrization (Definition 2.10); d_∞ is the bottleneck distance (Definition 1.18); d_σ is the Reininghaus dissimilarity for some $\sigma > 0$ (see equation (2.20)); E_4 is the 4SGDE parametrized by some $\sigma_d > 0$ and $s > 0$ (Definition 2.27); and $G_{p,\delta}$ is the δ-selective p-persistent entropy generator, for some $\delta > 0$ (Definition 2.42).

Definition 3.1. Let $\mathbb{R}^{nd}$ be the space of point clouds, $D_0 \in \mathrm{Bar}$ a fixed reference barcode, and $p \geq 0$ an homology degree. We have the four following types of topological regularizers:

$$\begin{aligned} L_{\mathrm{Rh},p}(\sigma)\colon P \in \mathbb{R}^{nd} &\mapsto d_\sigma^2(B_p(P), D_0) \in \mathbb{R}, \\ L_{\mathrm{4SGDE},0}(\mathbf{x}, \sigma_d, s)\colon P \in \mathbb{R}^{nd} &\mapsto \sum_{i=1}^{m} [E_4(B_0(P); x_i, \sigma_d, s) - E_4(D_0; x_i, \sigma_d, s)]^2 \in \mathbb{R}, \\ L_{\mathrm{bottleneck},p}\colon P \in \mathbb{R}^{nd} &\mapsto d_\infty(B_p(P), D_0) \in \mathbb{R}, \\ L_{\mathrm{entropy},p}(\delta)\colon P \in \mathbb{R}^{nd} &\mapsto (G_{p,\delta}(P) - G_{p,\delta}(D_0))^2 \in \mathbb{R}. \end{aligned} \tag{3.1}$$

We refer to these functions, respectively, as the *p-Reininghaus regularizer*, the *4SGDE regularizer* (only used with degree 0), the *p-bottleneck regularizer*, and the *p-entropy regularizer*. We have also added in parentheses, when needed, their hyperparameters; σ, σ_d, s and δ are strictly positive real values, and $\mathbf{x} \coloneqq \{x_i\}_{i=1}^m$ is a finite set of evaluation points in $[0, +\infty)$. We also define the *p-push regularizer* as

$$L_{\mathrm{push},p}\colon P \in \mathbb{R}^{nd} \mapsto -\sum_{(b,d)\in B_p(P)} (d - b) \in \mathbb{R}, \tag{3.2}$$

which, although it does not compare the persistence diagram to another diagram, it can also be used as a regularizer.

The differentiability of these regularizers follows directly from the previous results, and can be summarized as follows.

Theorem 3.2. *Let $\mathbb{R}^{nd}$ be the space of point clouds. Then, the p-push regularizers, p-Reininghaus regularizers,* 4SGDE *regularizers, and p-entropy regularizers are generically differentiable in $\mathbb{R}^{nd}$. In addition, the p-bottleneck regularizers are differentiable whenever the point cloud is in the space of point clouds in general position $\tilde{\mathcal{P}}$, which is generic in $\mathbb{R}^{nd}$, and its persistence diagram is in the subspace $\mathcal{X}$ of* Bar *as defined in Definition 2.36.*

Proof. The generic differentiability of p-push regularizers has been proven in Proposition 2.20; for p-Reininghaus regularizers, it has been proven in Corollary 2.22; and for 4SGDE regularizers, it has been proven in Proposition 2.26 —the result is more general, and includes the particular case of 4SGDEs. In the case of p-entropy regularizers parametrized by some $\delta > 0$, recall that, from Proposition 2.43, the δ-selective p-persistent entropy generator $G_{p,\delta}$ from $\mathbb{R}^{nd}$ to $\mathbb{R}$ is generically differentiable in $\mathbb{R}^{nd}$. As a consequence, if we consider the map $\varphi\colon \mathbb{R} \to \mathbb{R}$ given by $\varphi(x) = (x - G_{p,\delta}(D_0))^2$ (where $G_{p,\delta}(D_0)$ is a constant), then φ is differentiable in $\mathbb{R}$, and thus the composition $\varphi \circ G_p$ is generically differentiable in $\mathbb{R}^{nd}$, and $\varphi \circ G_p$ is equal to the p-entropy regularizer, hence the p-entropy regularizer is generically differentiable in $\mathbb{R}^{nd}$. The result for the p-bottleneck regularizer follows from Corollary 2.38. □

3.2 Environment for the experiments

The code has been implemented using Python version 3.10.12, with PyTorch and Persim as the main libraries. In our machine learning problems, automatic differentiation is employed to compute gradients. This approach allows us to directly express the loss in terms of the point cloud coordinates, eliminating the need for explicit gradient calculations.

For computing the persistence diagrams, we use the function `ripser_parallel` from the library Persim, which, given a point cloud, returns the persistence diagram of any homology degree, using coefficients in the field $\mathbb{Z}/2\mathbb{Z}$ and the Euclidean metric for the distance between points. The method combines optimization techniques and ideas from [20, 26, 27], achieving state-of-the-art performance. More details about the algorithm can be found in [21]. Furthermore, although the computation of the persistence diagrams is done without backpropagation, the final loss depends on the coordinates of the point cloud allowing the backpropagation. This is due to the fact that the coordinates of each off-diagonal point of the persistence diagram D^0 and D^1 correspond to pairwise distances of points of the point cloud. In particular, in D^0 each point $(0, d_i)$ is given by $d_i = \|\hat{X}_{i_1} - \hat{X}_{i_2}\|_2$ for two points of the generated batch $(\hat{X}_i)_{i=1}^N$. Similarly, in D^1 each point (b_i, d_i) is given by $b_i = \|\hat{X}_{i_1} - \hat{X}_{i_2}\|_2$ and $d_i = \|\hat{X}_{i_3} - \hat{X}_{i_4}\|_2$ for four points (which may be repeated) of the generated batch. We have access to these relations between points of the diagrams and points of the point clouds through the "generators" part of the object computed by the `ripser_parallel` function, allowing us to trace back the points of a persistence diagram to the original points of the point clouds, making backpropagation through gradient descent affect the coordinates of the point cloud. Furthermore, if the point cloud is the output of

a neural network (which is the case in generative models) the gradients trace back again, using the chain rule, to the weights of the neural network. More details about the algorithms developed can be found in the code; see Appendix A.6.

3.3 Synthetic experiments

We provide three examples of the use of topological losses for controlling the shape of arbitrary point clouds. We begin in each case with a point cloud P of 64 points at random initial positions in $\mathbb{R}^2$, and set a reference persistence diagram D_0 associated to some ground truth topological features. The goal is to make P learn, via backpropagation, to rearrange itself to produce the topological features described by D_0. We will not perform experiments for all functions of Definition 3.1 but rather give three proof-of-concept examples using regularizers $L_{\text{push},0}$ and $L_{\text{bottleneck},p}$. More precisely, we define the loss function

$$L_1 = \begin{cases} L_{\text{bottleneck},0} + L_{\text{bottleneck},1} & \text{if dependent on points of } P, \\ L_{\text{push},0} & \text{otherwise.} \end{cases} \tag{3.3}$$

Note that separation between these two cases is necessary since, whenever the function $L_{\text{bottleneck},0} + L_{\text{bottleneck},1}$ only depends on points of D_0, differentiation with respect to coordinates of points of P results in vanishing gradients, and as a consequence stagnation of the state of P. To avoid this issue, whenever L_1 does not depend on coordinates of P, we have to replace the total loss by a term that slightly perturbs the point cloud. The choice of $L_{\text{push},0}$ comes from the fact that this term perturbs points of P in such a way that points in its persistence diagram are *pushed* away from the diagonal. Experimentally, this phenomenon does take place and results in the appearance of new points in the diagrams of degree 0 and 1. These effects cause in general the dependence of $L_{\text{bottleneck},0}$ and $L_{\text{bottleneck},1}$ on coordinates of P in the next steps. Note that this is only an empirical observation and we do not provide a theoretical justification. The three experiments performed are the following. All figures and results are given in Appendix A.3.

Experiment A: Collapse of clusters. We begin with a point cloud P consisting of 5 clusters. The reference persistence diagram describes a point cloud with 3 clusters. Hence, the goal is to make P *understand* from the topological loss that it has to decrease the number of clusters from 5 to 3. Results are given in Figure A.5, and an animation of the evolution of P can be found in synthetic1_video.mov. We can see that the point cloud does learn to deform itself achieving the desired number of clusters.

Experiment B: Creation of clusters. We begin with the point cloud P consisting of 3 clusters, and the reference persistence diagram represents a point cloud with 5 clusters. Hence, the goal is to make the point cloud expand itself to appropriately increase the number of clusters. This example is an illustration of one of the possible applications of topological regularizers in generative models: avoiding an under-representation of the real space where the data lies, generating only a subset of the real data. With this example we show that the point cloud learns, through the topological loss, to expand the points and generate new clusters at distances given by the reference barcode. Results are given in Figure A.6, and an animation of the point cloud evolution can be found in synthetic2_video.mov.

Experiment C: From lines to circles. We begin with a point cloud P arranged for forming two non-intersecting line segments. The reference persistence diagram is the diagram obtained from a circle. The goal is thus to make P deform itself for forming a circle. Results are given in Figure A.7, and an animation can be found in synthetic3_video.mov, showing that the loss does efficiently teach the point cloud to generate the circle.

These experiments confirm the capability of topological loss functions to continuously deform a point cloud into a new object with the desired topological features. Having observed these positive results, we now apply these functions to a real-life problem: enhancing the training process of generative models.

3.4 Topology-informed generative models

3.4.1 Generative models

Assume we have a dataset D (e.g., images, songs), where the data points lie in a high-dimensional space $\mathbb{X}$. For instance, RGB 64×64 images are points in $\mathbb{R}^{3 \cdot 64 \cdot 64}$. Assume there is a probability distribution $P_t(X)$ associated to the data points in $\mathbb{X}$, assigning to regions of the space their likelihood to belong the dataset. However, in general we do not know the true probability distribution P_t, and rather we have only access to a collection of points $(X_i)_i$ sampled from P_t. The goal of a generative model is to learn a probability distribution P as close as possible to P_t so that we can sample new data points using P, producing data that had never been fed to them.

There is a wide variety of generative models, such as variational autoencoders (VAEs), generative adversarial networks (GANs), or diffusion models. We focus in this section on VAEs, since their training process allows to compare topological features of the input batch and the output batch (a comparison that loses meaning in GANs, for instance), and the training process is not as time-expensive as in diffusion models. The goal of this section is to analyze changes in the behaviour of VAEs when we integrate topological loss terms in the training process. To do so, we first describe the working principle of a VAE. Then, we propose the new training algorithm when using a topological regularizer, and in the next section we evaluate experimentally the performance of these new models.

3.4.2 Variational autoencoders

A VAE is an unsupervised machine learning model that maps high-dimensional data to a lower-dimensional latent space using an encoder, and reconstructs the data from this latent space using a decoder. We denote the latent space by $\mathcal{Z}$, and refer to its elements as *latent vectors*, noted z. Both the encoder and the decoder of the VAE are neural networks, parametrized by their respective sets of parameters ϕ and θ. The encoder maps an input X to a pair of vectors $\mu(X;\phi), \Sigma(X;\phi)$, both with dimension equal to that of the latent space, and the decoder maps a latent vector z to a data point $f(z;\theta) \in \mathbb{X}$. If θ is fixed and z is sampled according to a probability density function $P(z)$ on $\mathcal{Z}$, then $f(z;\theta)$ is a random vector in $\mathbb{X}$. The objective of the VAE is to optimize θ such that the $f(z;\theta)$'s are points from the data distribution. In other words, θ has to be optimized in order to maximize, for

every X in the dataset, the following quantity:

$$P(X) = \int_{\mathcal{Z}} P(X \mid z;\theta)P(z)\,dz, \tag{3.4}$$

where $P(X)$ is the probability that the data point X is produced by the generative process $z \mapsto f(z;\theta)$ for all possible z in $\mathcal{Z}$ following the distribution $P(z)$. In VAEs, $P(z)$ is defined as a standard distribution $\mathcal{N}(z \mid 0, Id)$ [25]. The term $P(X \mid z;\theta)$ represents the probability that $f(z;\theta)$ *looks like* X. This function is often defined as a Gaussian distribution $\mathcal{N}(X \mid f(z;\theta), \sigma \cdot Id)$, where σ is a hyperparameter, although other distributions can be used as long as they are continuous on θ and can be numerically computed [25]. Hence, if the decoder maximizes $P(X)$, it will create data points that *look like* those from the dataset. Due to spatial limitations we will not explain here how to approximate $P(X)$ in a differential (i.e., allowing gradient descent) and efficient manner. A detailed derivation can be found in Appendix B.2, and we provide here directly the main results. To summarize, in the case of an individual data point X, and with a latent space dimension J and an arbitrary positive number L, one can obtain the following approximation of a lower bound of $\log P(X)$:

$$\begin{aligned}&\tilde{\mathcal{L}}(\phi,\theta;X,L) = \frac{1}{2}\sum_{j=1}^{J}(1+\log((\Sigma_j)^2) - (\mu_j)^2 - (\Sigma_j)^2) + \frac{1}{L}\sum_{l=1}^{L}\log(P(X \mid z^l))\\ &\text{where } z^l = \mu(X;\phi) + \Sigma(X;\phi)\odot\epsilon^l;\ \epsilon^l \sim \mathcal{N}(0, Id).\end{aligned} \tag{3.5}$$

Here $\odot$ is the element-wise product, and Σ_j and μ_j are the components of the vectors Σ and μ, respectively. Each z^l corresponds to a random latent vector obtained from the noise vector ϵ^l —this technique, known as the reparametrization trick, is explained in more detail in Appendix B.2. Furthermore, the function (3.5) is continuous on ϕ and θ, and $-\tilde{\mathcal{L}}$ can be computed and minimized via stochastic gradient descent [25]. The term $\log P(X \mid z^l)$ corresponds to a Bernoulli or Gaussian distribution whose parameters are given by $f(z;\theta)$, and the choice depends on the type of data. In our experiments, we have modeled the term $\log P(X \mid z^l)$ as a Bernoulli distribution; see the code (Appendix A.6) for details.

When training a VAE, the input consists of batches of N data points $X_1, \ldots, X_N$ randomly sampled from the full dataset D, and the total loss is calculated for these N inputs. Furthermore, when N is larger than 100 (which holds in our experiments), we can take a single sample ϵ_i for each X_i [24]. As a consequence, the total loss in each training iteration of the standard variational autoencoder is given by

$$\text{Loss}\left(\phi,\theta;\{X_i\}_{i=1}^N\right) = -\sum_{i=1}^{N}\tilde{\mathcal{L}}(\phi,\theta;X_i,1), \tag{3.6}$$

where the -1 factor comes from the fact that the gradient descent process aims to minimize the loss, and our goal is to maximize each $\tilde{\mathcal{L}}$. A summary of the process computed during each training iteration is illustrated in the following diagram:

$$X_i \overset{\text{Encoder}(X_i;\phi)}{\mapsto} \mu(X_i), \Sigma(X_i) \overset{\&\epsilon_i}{\mapsto} z_i \overset{\text{Decoder}(z_i;\theta)}{\mapsto} f(z_i;\theta) \mapsto \tilde{\mathcal{L}}(\phi,\theta;X_i,1) \mapsto \text{Loss}. \tag{3.7}$$

The final loss employs $\mu(X_i)$, $\Sigma(X_i)$ for calculating the first term of (3.5), and $f(z_i;\theta)$ for the second term of (3.5). A summary of the loss calculation for a standard VAE is given

in Algorithm 1. Note that the first term in equation (3.5) is the opposite of the Kullback-Leibler divergence KLD, and the second term in (3.5) is, in our code, the opposite of the Binary Cross-Entropy loss BCE. As a consequence, we use in Algorithm 1 the notation $-\tilde{\mathcal{L}} = \text{KLD} + \text{BCE}$. Furthermore, for convenience we refer to the total loss of the standard VAE given in (3.6) as L_0. It is important to note that the BCE loss at an individual pixel of an image, where x_i is the ground truth value of the pixel intensity, and $\hat{x}_i$ is the predicted value in the reconstructed pixel (both values between 0 and 1), is given by

$$-x_i \log(\hat{x}_i) - (1 - x_i) \cdot \log(1 - \hat{x}_i). \tag{3.8}$$

This function has a unique minimum when $\hat{x}_i = x_i$, and its value increases with the absolute difference $|x_i - \hat{x}_i|$. The BCE loss of the entire image is then the sum of the BCE loss in each pixel, and represents a measure of how accurate the reconstruction of the image is; minimizing its value corresponds to making the model produce an image more similar to the original image.

3.4.3 TopoVAEs

The topologically-regularized VAEs or topology-informed VAEs, abbreviated TopoVAEs, are the combination of the standard VAE with a new term in the loss function that acts as a regularizer. The goal of the regularizer is to make the model produce batches of images with a distribution similar to the batch of real images. These regularizers represent ways of "teaching" the model information from the persistence diagrams, such as how many clusters the clouds of generated data have to form, how far apart these clusters have to be, or how many loops have to be formed and how large they have to be. This can avoid early mode collapse and enhance a faster imitation of the true data distribution. In order to implement these regularizers, in each iteration a batch of N true datapoints is taken, and we view the true batch as a point cloud, and the generated batch (i.e., the N datapoints generated by the VAE) as a second point cloud. We then compute their persistence diagrams of degree 0 and/or 1 and compute some measure of dissimilarity between these two diagrams, which can be one (or a combination) of the regularizers in equation (3.1). This "distance" is taken as a new loss term and added to the total loss. This working principle is illustrated in Figure A.1 and more formally described in Algorithm 2, where we have used the notation D^p for the persistence diagram of degree p obtained from the generated batch, and D_0^p for the diagram of degree p obtained from the true batch. The training process then follows the main idea of Section 3.3. However, instead of points in the plane we now work with points in higher-dimensional spaces, and the coordinates of these points are the outputs of neural networks. Using the training procedure defined in Algorithm 2 and the topological regularizers of equation (3.1), we construct the following four topology-informed total losses:

$$\begin{aligned}
L_{T1}(P;\omega_0,\omega_1) &= L_0 + \omega_0 \cdot L_{\text{bottleneck},0}(P; D_0^0) + \omega_1 \cdot L_{\text{bottleneck},1}(P; D_0^1),\\
L_{T2}(P;\omega_0,\omega_1,\delta) &= L_0 + \omega_0 \cdot L_{\text{entropy},0}(P; D_0^0, \delta) + \omega_1 \cdot L_{\text{entropy},1}(P; D_0^1, \delta),\\
L_{T3}(P;\omega_0,\omega_1,\delta,\sigma) &= L_0 + \omega_0 \cdot L_{\text{entropy},0}(P; D_0^0, \delta) + \omega_1 \cdot L_{\text{Rh},1}(P; D_0^1, \sigma),\\
L_{T4}(P;\omega_0,\omega_1,\mathbf{x},\sigma_d,s,\sigma) &= L_0 + \omega_0 \cdot L_{\text{4SGDE},0}(P; D_0^0, \mathbf{x}, \sigma_d, s) + \omega_1 \cdot L_{\text{Rh},1}(P; D_0^1, \sigma).
\end{aligned} \tag{3.9}$$

The topological losses are only added to the total loss if they involve points of the learnable point cloud P —otherwise, the gradients are zero and do not modify the point cloud.

3.5 Experiments and results

We now fix a VAE model structure, which is described in the next subsection. Using this structure, we define VAE0 as the VAE employing the standard loss L_0, and we refer to the VAE model employing one of the topology-informed losses of equation (3.9) as a TopoVAE. In particular, the model using loss L_{Ti} is referred to as TopoVAEi.

We test the resulting four topology-informed VAEs in the FashionMNIST dataset, which consists of 60000 grayscale images of 28×28 pixels, with labels from 10 classes (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot). We first find optimal hyperparameters for each topological regularizer by a grid search approach, in order to systematically explore a range of values for key hyperparameters (a detailed explanation of the final choices is given in Appendix A.4.1). Using these fine-tuned regularizers, we compare each TopoVAE to VAE0 in a set of qualitative and quantitative experiments in Subsections 3.5.2 and 3.5.3, respectively. The promising results lead us to further explore the diverse properties of these regularizers, as detailed in Subsections 3.5.4 and 3.5.5, where we address interesting new lines of applications. In addition, in all experiments we use a batch size of 128; this choice comes from the fact that it ensures a large enough point cloud for providing rich structural information through the topological regularizer, while being sufficiently small for ensuring a computation process that is not too time-consuming —larger batch sizes result in a more time-expensive computation of the persistence diagrams and the losses, while not bringing an improved training performance. The optimal hyperparameters, training times and summary of performance measurements are summarized in Table A.1.

3.5.1 VAE structure

The VAE used is a convolutional VAE whose detailed structure is given in the code; see `topovae_tests.py`. Unless stated otherwise, in all experiments the latent space dimension is 10. We briefly describe the transformations using latent dimension 10. The input has shape $(128, 1, 28, 28)$ (i.e., a batch of 128 grayscale 28×28 images), and the encoder realizes the following shape transformations:

$$\begin{aligned}(128,1,28,28) &\mapsto (128,32,14,14) \mapsto (128,64,7,7) \mapsto (128,3136) \mapsto (128,256)\\ &\mapsto (128,10),(128,10).\end{aligned} \tag{3.10}$$

The first two maps are done via convolutional layers, the third map is a re-shaping operation of the data, the fourth map occurs via a fully-connected layer, and the last map occurs by applying in parallel two fully-connected layers. The output corresponds to two batches of μ and Σ vectors. Applying the reparametrization trick, these values are then transformed into 128 latent vectors with total shape $(128, 10)$. The latent vectors are then fed into the decoder, and the resulting shape transformations are

$$\begin{aligned}(128,10) &\mapsto (128,256) \mapsto (128,3136) \mapsto (128,64,7,7) \mapsto (128,32,14,14)\\ &\mapsto (128,10),(128,1,28,28).\end{aligned} \tag{3.11}$$

The maps correspond, respectively, to a fully-connected layer, a second fully-connected layer, a re-shaping operation, and two transposed convolutional layers. A description of the fully-connected layer, the convolutional layer, and the transposed convolutional layer, which are the building blocks of the convolutional VAE, are given in Appendix B.3.

3.5.2 Qualitative comparison

We first qualitatively compare the images generated by TopoVAE models and VAE0 during early training. We also show qualitative differences in the decay of the BCE losses. More precisely, for each TopoVAE, we train VAE0 and TopoVAE simultaneously using initially cloned models, i.e., the two models have initially the same weights. In addition, the batches fed into both models are exactly the same, thus, the only difference between the two models is the absence or presence of the topological regularizer. Both models are trained for exactly one epoch [1]. From their training processes, we generate two types of figures:

(i) A figure showing

- a set of 32 new real images (which the models have not been trained with) (on the left of the figure);
- the corresponding 32 images generated by VAE0 (on the middle of the figure);
- the 32 images produced by TopoVAE (on the right of the figure).

The training step used for plotting these images is chosen among the training steps 50, 75, 100, 125 and 150. One epoch consists of 469 training steps.

(ii) A comparison of the evolution of the BCE losses of VAE0 and TopoVAE during the initial steps of training. The amount of iterations pictured in each figure has been selected in each case in order to showcase relevant differences between the two models. In each case, after the iterations shown, the performance of the two models is roughly similar and the BCE losses are approximately equal.

All figures corresponding to these experiments can be found in Appendix A.4.2. In each figure, and unless specified otherwise, the hyperparameters of the model used are those presented in Table A.1. The figures corresponding to each TopoVAE are Figures A.8, A.9, A.10, A.11, and A.12 for TopoVAE1; Figures A.13, A.14, A.15, A.16, A.17, and A.18 for TopoVAE2; Figures A.19 and A.20 for TopoVAE3, and Figures A.21 and A.22 for TopoVAE4. In all cases, the figures show that TopoVAEs produce images with, in general, a higher quality, pointing towards an outperformance of the topologically regularized models compared to VAE0. This outperformance, from a qualitative point of view, is more pronounced in TopoVAE1, followed by TopoVAE2, and then TopoVAE3 and TopoVAE4.

3.5.3 Quantitative comparison

We have seen that TopoVAEs seem to produce images with better quality; however, we want to quantify this possible outperformance. To do so, we study how the regularizer affects the evolution of the BCE loss —which is a measure of how accurately the model reconstructs input images. Note that, although there are other metrics for evaluating the

[1] The choice of one epoch comes from the fact that the learning process in the FashionMNIST dataset is relatively fast: using batches of 128 images, after 300 training steps the standard VAE has, in general, already learned to generate high quality images and the BCE loss stagnates. Thus, we know that the stagnation of the losses will occur before the end of the first epoch (which corresponds to 469 training steps). Hence, instead of doing tests with early stopping, for simplicity we train all models for one epoch.

performance of a model, such as the Inception Score or the Fréchet Inception Distance (FID), what they measure is not completely clear [28], and the FID relies on an Inception v3 model trained on the ImageNet dataset, and not on FashionMNIST, which can lead to results that may not be meaningful. Hence, we have chosen to study the magnitude of the BCE loss, which describes the accuracy and the quality of the images produced by the VAEs. To study the effect of the topological regularizer on the BCE loss, for each TopoVAE we proceed as follows.

1) We run fifty times, using different random seeds, the chosen TopoVAEi and VAE0 being initially cloned models. At training steps 25, 50 and 75 of the first epoch, we save the BCE losses of both models, denoted by $\mathrm{BCE}_{\mathrm{VAE0}}(s)$ for the standard VAE and $\mathrm{BCE}_{\mathrm{TopoVAE}i}(s)$ for TopoVAEi. We focus only on the cases $s = 25, 50, 75$ since in the previous qualitative analysis we have observed that the difference between the BCE losses of the regularized and the non-regularized models is more pronounced in the first 20 to 100 training steps. On the other hand, after 100 iterations the BCE losses are almost equal. Note that, since we are in the first epoch, the batches of images are being fed into the models for the first time, so the BCE loss also describes the capability of the models to generalize and adapt to new data.

2) For each test j, and for each each number of training steps $s = 25$, 50 and 75, we compute the relative variation of the loss, $r(s)$, defined as

$$r(s) = 100 \cdot \frac{\mathrm{BCE}_{\mathrm{VAE0}}(s) - \mathrm{BCE}_{\mathrm{TopoVAE}i}(s)}{\mathrm{BCE}_{\mathrm{VAE0}}(s)}. \tag{3.12}$$

In other words, $r(s)$ gives a percentage measure of the relative change of the BCE loss when the regularizer is added, at training step s. It is positive if the loss of the regularized model converges faster. We denote by $r_j(s)$ the value of r measured in test j at step s. Then, for each $s = 25, 50, 75$, we compute the mean of the r_j values over the fifty tests:

$$\bar{r}(s) = \frac{1}{50}\sum_{j=1}^{50} r_j(s), \tag{3.13}$$

hence, these values are a mean of the relative change in the decay of the BCE loss when adding the topological regularizer, expressed as a percentage.

This process is performed for each TopoVAEi, where $i = 1, 2, 3, 4$, and the resulting $\bar{r}$ values for each TopoVAE are given in Table A.1. It is important to note that the $\bar{r}$ values are consistently positive in all models TopoVAE1, 2, 3 and 4, showing that the BCE loss decays consistently faster in regularized models compared to their non-regularized counterparts. Furthermore, it is noteworthy to highlight that there seems to be a correlation between the qualitative performance ranking of the TopoVAE models (with TopoVAE1 exhibiting the highest performance, followed by TopoVAE2, then TopoVAE3, and ultimately TopoVAE4) and the magnitude of the $\bar{r}$ values presented in Table A.1. Specifically, a higher $\bar{r}$ value would seem to indicate the model's enhanced ability to generate realistic and high quality images. Note that this observation is speculative and based solely on the obtained results. We also see that the $\bar{r}$ value is maximal at step 50 in all cases; this may be due to the fact that initially the TopoVAE and VAE0 models are cloned, so their behaviour is similar, and

their difference increases with the number of training steps, but, at the same time, they end up converging to the same behaviour after around 80-100 iterations.

3.5.4 Consistency across different VAE structures

We now explore how topological regularization performs across different VAE structures. In particular, we want to see if a topological regularizer with hyperparameters optimized in a certain VAE structure consistently ensures an improved performance in other VAE structures. We focus on total losses L_{T1} and L_{T2} from equation (3.9), since these have yielded the best results in the previous section.

We replace the first VAE we used, which we now refer to as VAE.A, by a simpler VAE, which we refer to as VAE.B, which has 48781 parameters while VAE.A has 1683669 parameters. Hence VAE.B represents a simpler model, less prone to overfitting. The structure of VAE.B is described in more detail in Appendix A.5. We denote by VAE0(A) and VAE0(B) the models VAE.A and VAE.B using the standard VAE loss L_0, respectively, and by TopoVAE1(A) and TopoVAE1(B) the models VAE.A and VAE.B employing the total loss L_{T1}. Similarly, TopoVAE2(A) and TopoVAE2(B) are models VAE.A and VAE.B using the loss L_{T2}. We use for both losses the same hyperpameters used in the previous section.

We compare the performance of TopoVAE1(B) and TopoVAE2(B) with model VAE0(B), in order to see if there is an improved training performance when applying the regularization on structure VAE.B. We also compare their evolutions to those obtained in the original structure VAE.A (for which the hyperparameters were originally fine-tuned).

The results for the loss L_{T1} are shown in Figures A.24 and A.25. In particular, Figure A.24 shows the performance of TopoVAE1(A) and VAE0(A) while Figure A.25 shows the performance of TopoVAE1(B) and VAE0(B). Note that, while $L_{\text{bottleneck},0}$ and $L_{\text{bottleneck},1}$ decrease during the training process of VAE.A, the situation changes for VAE.B. In that case (Figure A.25) the topological losses decrease more slowly, or even do not decrease in the case of $L_{\text{bottleneck},1}$. In addition, the BCE losses of TopoVAE1(B) and VAE0(B) are almost identical, see the left graph of Figure A.25.

The analogous results for L_{T2} are given in Figures A.26 and A.27. We observe a similar behaviour with the losses: while the BCE loss decreases faster in TopoVAE2(A) compared to VAE0(A), in the case of TopoVAE2(B) and VAE0(B) the BCE losses are almost identical. In addition, while the topological loss of degree 0 clearly decays in TopoVAE2(A), the decay is much less marked in TopoVAE2(B).

In summary, these two experiments seem to suggest that topological regularizers may have to be fine-tuned for each specific model structure. However, we have to note that after trial and error tests we did not obtain an improved performance of TopoVAE1(B) compared to VAE0(B), pointing to the fact that it may be possible that topological regularizers are more useful, and have a stronger effect, under the presence of more parameters in the generative model.

It is important to note that, from these tests, we have seen that the topological losses are minimized during the training process in VAE.A (the model for which the regularizers have been fine-tuned). Not only that, but the decay is clearly marked during the first epoch (i.e., the first 469 iterations) —see Figures A.24 (right) and A.26 (right). In particular, during this stage, each training iteration consists of completely new images, and therefore the model learns to minimize the topological losses even on batches of unseen images. In

other words, the model learns to produce topological features in the generated data that matches with topological features in the ground truth batches, an effect that is justified by the decay of the topological losses in the aforementioned figures. Note also that the decay is more marked for losses corresponding to diagrams of degree 0 while not so marked for homology degree 1. This may indicate that the homology degree 0 can be more meaningful in the context of learning to produce images rather than the homology degree 1.

3.5.5 Topogical regularizers in latent space

As we previously commented, VAEs sometimes learn a distribution of latent vectors in the latent space incompatible with the true data distribution, making them unable to generate realistic new data [1]. To address this problem, we now explore the use of a topological regularizer controlling the *latent* distribution. In fact, an advantage of the developed regularizers is that they can be applied at any inner layer of the model. In particular, since each topological regularizer in Definition 3.1 does not depend on the dimension of the point cloud P, we can use the batch of N latent vectors $\{z_i\}_{i=1}^N$ produced in each training iteration as point cloud P. In that case, the topological regularizers, comparing the persistence diagrams of $\{z_i\}_{i=1}^N$ and the of the batch of true images $\{X_i\}_{i=1}^N$, result in a new way of controlling the spacial distribution of the latent vectors.

Note, however, that in the FashionMNIST dataset the data points X_i lie in $\mathbb{R}^{784}$, and each component has a value between 0 and 1. On the other hand, each latent vector z_i lies in in $\mathbb{R}^{n_Z}$, where n_Z is the latent space dimension and is often smaller than 784, and has unbounded components. Hence, in order to allow a more meaningful comparison of persistence diagrams we can normalize the ground truth diagrams (i.e., those obtained from the true images) such that both diagrams have points with similar magnitudes. In fact, assume we did not normalize properly the diagrams. Then, if initially the diagrams from latent vectors have all points with death value lower than 0.1, and the ground truth diagrams have points with deaths of magnitude up to 20, then their bottleneck distance (e.g., for diagrams of degree 0) will only depend in most cases on the ground truth diagram, leading to vanishing gradients. Hence, we choose to apply an appropriate renormalization of the ground truth diagrams. In order to do so, we first train VAE0 with latent space dimension $n_Z = 10$ and plot persistence diagrams of latent batches $\{z_i\}_{i=1}^{128}$ produced during the training process; see for instance Figure A.28. From these diagrams, we can infer the magnitudes of coordinates of points in the diagrams of batches of latent vectors. In particular, the maximal death of points is in general between 3 and 4. We then compare these diagrams to the diagrams of batches of 128 true images of the FashionMNIST dataset; see for instance Figure A.4. In that case the maximal death magnitude is around 10. We thus choose to multiply the coordinates of points in the diagrams of true data by a factor of 0.35. This choice makes the initial diagrams produced by the VAE and the diagrams of true batches have similar magnitudes, which could potentially allow a more meaningful comparison of these objects through the topological regularizers.

After rescaling the persistence diagrams of the true data, we train VAE.A with the standard loss (i.e., model VAE0) and VAE.A with the loss L_{T1}, with P corresponding to the batch of latent vectors instead of the final outputs. We refer to the latter model as TopoVAE-Z1. It is important to note that, since we have only modified the ground truth persistence diagrams, the topological regularizers remain differentiable. We perform two

different tests.

Test 1: We use $n_Z = 10$ for both VAE0 and TopoVAE-Z1, i.e., the same latent dimension as in all previous experiments. The hyperparameters of TopoVAE-Z1 are finetuned for this specific example; we have found an optimal performance at $(\omega_0, \omega_1) = (15.0, 15.0)$. Figure A.29 shows images produced by both models during early training, and we can see that TopoVAE-Z1 presents a higher image quality than VAE0. In Figure A.30 (left) we can see that the BCE loss decays faster in TopoVAE-Z1 than in VAE0, supporting the improved image quality. Furthermore, Figure A.30 (middle and right) shows that the model does learn to minimize the topological losses during the first five training epochs (a minimization more marked for the loss of degree 0).

Test 2: We use $n_Z = 2$ for both VAE0 and TopoVAE-Z1 in order to give a proof-of-concept example illustrating the differences in the spatial distribution of latent vectors. We keep $(\omega_0, \omega_1) = (15.0, 15.0)$ for TopoVAE-Z1, and train both models during one epoch. In Figure A.31 we show 500 latent vectors in latent space produced by VAE0 (left) and TopoVAE-Z1 (right), at two different training stages. These vectors are obtained from randomly sampling 500 images of the true dataset, feeding them as inputs to both models, and plotting the resulting latent vectors. The colors of points correlate with image labels. In Figure A.32 we show the distribution of 1000 latent vectors in both models, at the end of the first training epoch. We can see an improved spatial distribution of latent vectors in TopoVAE-Z1 in all three cases. In fact, it seems that the regularized model has learnt to classify in an unsupervised manner the latent vectors, arranging them spatially in clusters corresponding to the types of images they arise from, an effect much more marked than in VAE0. In addition, Figure A.33 shows a comparison of images produced by TopoVAE-Z1 and by VAE0 in early training. We can observe a higher quality in the images generated by TopoVAE-Z1. Finally, Figure A.34 confirms that the BCE losses of TopoVAE-Z1 are consistently smaller than those of VAE0. Furthermore, the bottleneck losses (more clearly marked for the loss of degree 0) clearly decrease during the training process, showing that these terms are minimized by the model, even though each iteration introduces completely new batches of images.

Given the improved results in TopoVAE-Z1, we may also wonder if there is also an improved redistribution of latent vectors in latent space when the regularizer is applied on the final outputs, i.e., in the TopoVAE models. We thus test this effect in an individual example: in Figure A.23 we show 500 latent vectors produced by VAE0 and TopoVAE1 with (ω_0, ω_1) =(15.,15.) after one training epoch, both models having dimension of the latent space $n_Z = 2$. The latent vectors also arise from randomly sampling 500 true images. We can observe an improved distribution of latent vectors in the regularized model. Given these promising results, we expect further investigation on this aspect of topological regularization, for instance for improving unsupervised classification tasks.

Conclusions and future work

Future work

The methods we have developed open a wide variety of applications and extensions to other machine learning problems; some possible extensions are the following.

Regularization of inner layers of generative models Contrarily to regularizers that directly compare distances between true and generated data, the functions we have proposed can be applied at *any layer* of the generative model. This property opens new ways of controlling the distribution of the latent vectors and of regularizing inner layers of the model, which can be useful for enhancing both the diversity and fidelity of images obtained through direct latent vector sampling. For instance, we have shown that applying topological regularizers on the latent vectors causes striking redistributions of the latent vectors related to the types of images they produce via the generator. Hence, we believe that further work in this area should be conducted, for instance comparing topology-informed VAEs with VAEs specifically trained for achieving a good latent distribution.

Application in diffusion models Since VAEs have a relatively fast training process in the FashionMNIST dataset, the outperformances we have obtained are not significant, and we can still train in an effective way a VAE without a topological regularizer. In fact, while we have observed a higher diversity of the generated images in early training in TopoVAEs, this does not reduce considerably the learning time since standard VAEs already require short training times for achieving image diversity —less than one epoch and less than two minutes. However, diffusion models can have highly time-expensive training procedures, requiring many more iterations and longer training times for achieving image diversity and high image quality, and thus topological regularization could potentially avoid minutes or hours of training. The interest in this line of application was confirmed by the publication, by the time of writing this, of [29], where the authors used persistent homology to regularize diffusion models —however, their method relied on constructing persistence diagrams of individual images, which can be time-consuming, and thus our approach, viewing image batches as point clouds, should also be studied in diffusion models.

Extension to other machine learning tasks The regularizers developed in this work can be applied to many other generative modelling problems. For instance, they could be used in cases where images have to be generated with specific topological and geometric features; such as in the scenario exposed in [5]. To do so, the simplicial complexes could be replaced by cubical complexes —which are better-suited for individual image analysis—,

and the Rips filtration could be replaced by a lower-star filtration. Topological regularizers could also be used for mesh generation tasks, i.e., for training generative models that produce point clouds representing meshes of 3D objects. The regularizers could be directly applied on the generated point clouds with the goal of forcing the models to generate point clouds with specific topological features (and thus helping the model close loops or voids, generate the appropriate number of clusters, and so on). Lastly, as we have seen in Chapter 3, topological regularizers may be useful for conducting unsupervised classification tasks. A reason may be the fact that they can help identifying clusters in high dimensional spaces, even under the presence of noise.

Further analysis of the advantages provided by topological regularization Although the results obtained in this work are promising, more thorough comparisons between standard VAEs and TopoVAEs should be done in the future, for instance, through tests in diverse datasets, with different VAE structures, and using multiple performance metrics (e.g., FID or Geometry Score). It would be also interesting to explore the relation between magnitude of the outputs from the topological regularizers and properties of the generated data, such as image quality and diversity. This could bring a deeper understanding of the meaning of each function, allowing to use combinations of topological regularizers in a more informed way. In fact, we have conducted such an analysis for bottleneck and entropy regularizers, which has been done as a supplementary exploration, in Appendix B.4. The results confirm that the value of the bottleneck and entropy regularizers is related with image quality and image diversity; higher noise (in both grayscale and colored datasets) leads to a larger value of the bottleneck and entropy regularizers, and higher differences in image diversity seem to also produce larger outputs.

On the theoretical side, we also expect further research in the differentiability of loss functions employing multidimensional persistence diagrams. In fact, multidimensional persistent homology has recently been gaining attention [30], since it can lead to more useful descriptors of the data compared to single-dimensional persistent homology. Thus, if applied in generative models, regularizers based on multi-parameter persistence diagrams could provide more insightful descriptions of the data, including information related to the spatial density of points, the scale, and other parameters.

Conclusion

In this work, we have developed new topological regularizers and have provided new theoretical results ensuring their generic differentiability, making our regularizers amenable to gradient descent optimization. We have applied these regularizers in VAEs and showed that they can enhance their learning processes, achieving a higher diversity of images in early training, an improved image quality and a faster decay of the BCE loss. We have also discussed possible new lines of application, such as controlling the latent distribution, improving classification tasks or mesh generation tasks, or applying them in diffusion models. In summary, this work has reviewed and expanded topological regularization with persistence diagrams, and given the promising results, we expect further advances in this field.

Bibliography

[1] Yaniv Yacoby, Weiwei Pan, and Finale Doshi-Velez. "Characterizing and avoiding problematic global optima of variational autoencoders". In: *Symposium on Advances in Approximate Bayesian Inference*. PMLR. 2020, pp. 1–17.

[2] Valentin Khrulkov and Ivan Oseledets. "Geometry score: A method for comparing generative adversarial networks". In: *International conference on machine learning*. PMLR. 2018, pp. 2621–2629.

[3] Jeremy Charlier, Radu State, and Jean Hilger. "PHom-GeM: Persistent homology for generative models". In: *2019 6th Swiss Conference on Data Science (SDS)*. IEEE. 2019, pp. 87–92.

[4] Yair Schiff et al. "Characterizing the latent space of molecular deep generative models with persistent homology metrics". In: *arXiv:2010.08548* (2020).

[5] Fan Wang et al. "Topogan: A topology-aware generative adversarial network". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer. 2020, pp. 118–136.

[6] Michael Moor et al. "Topological autoencoders". In: *International conference on machine learning*. PMLR. 2020, pp. 7045–7054.

[7] Nikita Balabin et al. "Disentanglement Learning via Topology". In: *arXiv:2308.12696* (2023).

[8] Mariem Mezghanni et al. "Physically-aware generative network for 3d shape modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9330–9341.

[9] Serguei Barannikov et al. "Representation topology divergence: A method for comparing neural network representations". In: *arXiv:2201.00058* (2021).

[10] Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. "A framework for differential calculus on persistence barcodes". In: *Foundations of Computational Mathematics* (2021), pp. 1–63.

[11] Afra Joze Zomorodian. *Computing and comprehending topology: Persistence and hierarchical Morse complexes*. University of Illinois at Urbana-Champaign, 2001.

[12] Edelsbrunner, Letscher, and Zomorodian. "Topological persistence and simplification". In: *Discrete & Computational Geometry* 28 (2002), pp. 511–533.

[13] Afra Zomorodian and Gunnar Carlsson. "Computing persistent homology". In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. 2004, pp. 347–356.

[14] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. "Stability of persistence diagrams". In: *Proceedings of the Twenty-first Annual Symposium on Computational Geometry*. 2005, pp. 263–271.

[15] James R Munkres. *Elements of algebraic topology. Advanced Book Program. Redwood City, California etc.* 1984.

[16] Raúl Rabadán and Andrew J Blumberg. *Topological data analysis for genomics and evolution: Topology in biology*. Cambridge University Press, 2019.

[17] Carles Casacuberta. *Topological Data Analysis*. 2020.

[18] Gunnar Carlsson. "Topology and data". In: *Bulletin of the American Mathematical Society* 46.2 (2009), pp. 255–308.

[19] Clara Loeh. "A comment on the structure of graded modules over graded principal ideal domains in the context of persistent homology". In: *arXiv:2301.11756* (2023).

[20] Ulrich Bauer. "Ripser: efficient computation of Vietoris–Rips persistence barcodes". In: *Journal of Applied and Computational Topology* 5.3 (2021), pp. 391–423.

[21] Julián Burella Pérez et al. "giotto-ph: A Python library for high-performance computation of persistent homology of Vietoris-Rips filtrations". In: *arXiv:2107.05412* (2021).

[22] Jan Reininghaus et al. "A stable multi-scale kernel for topological machine learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4741–4748.

[23] Rubén Ballester et al. "Predicting the generalization gap in neural networks using topological data analysis". In: *Neurocomputing* 596 (2024), p. 127787.

[24] Diederik P Kingma and Max Welling. "Auto-encoding variational Bayes". In: *arXiv: 1312.6114* (2013).

[25] Carl Doersch. "Tutorial on variational autoencoders". In: *arXiv:1606.05908* (2016).

[26] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. "Ripser.py: A lean persistent homology library for Python". In: *Journal of Open Source Software* 3.29 (2018), p. 925.

[27] Dmitriy Morozov and Arnur Nigmetov. "Towards lockfree persistent homology". In: *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. 2020, pp. 555–557.

[28] Lucas Theis, Aäron van den Oord, and Matthias Bethge. "A note on the evaluation of generative models". In: *arXiv:1511.01844* (2015).

[29] Chen Song et al. "Topo-Diffusion: Topological diffusion model for image and point cloud generation". In: *Under review as a conference paper at ICLR 2024* (2023).

[30] David Loiseaux, Mathieu Carrière, and Andrew J Blumberg. "Efficient approximation of multiparameter persistence modules". In: *arXiv:2206.02026* (2022).

[31] Jonas Teuwen and Nikita Moriakov. "Convolutional neural networks". In: *Handbook of medical image computing and computer assisted intervention*. Elsevier, 2020, pp. 481–501.

Appendix A

Experiments and results

A.1 Working principle of topology-informed VAEs

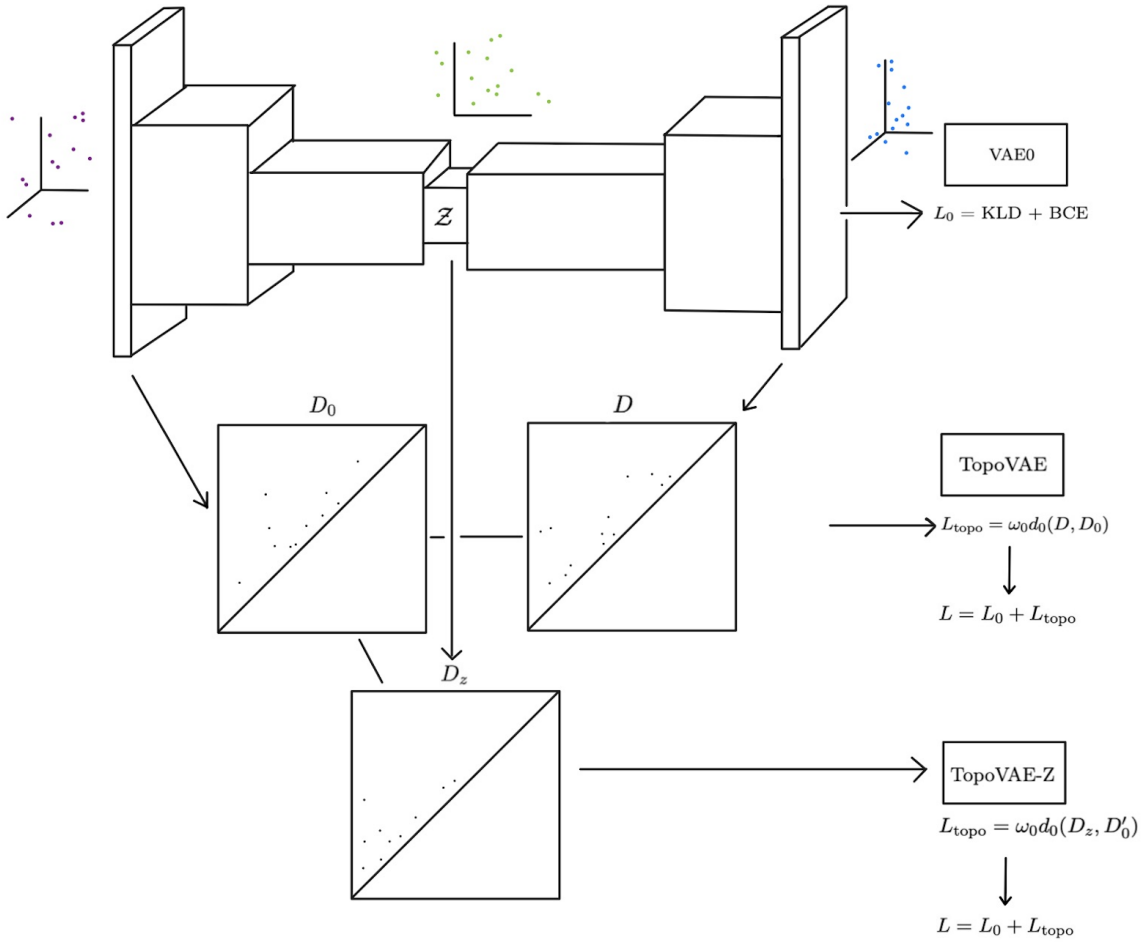

Figure A.1: Working principle of VAE0, TopoVAE and TopoVAE-Z. Persistence diagrams D_0, D, D_z and D_0' arise, respectively, from the input batch; the output batch; the batch of latent vectors, and a rescaling operation on D_0, and d_0 is a measure of dissimilarity between persistence diagrams. The method can be used for persistence diagrams of degree 0, 1, or both degrees simultaneously.

A.2 Scaled Gaussian density functions

Using the FashionMNIST dataset, the parameters of the density function have been adapted to the properties of the dataset. We choose $\sigma = 0.1$, a scale factor of 0.0005, and the sampled points to be 30 equally spaced points distributed in $[0, 15.0]$. The choice of 30 points, although leading to some inaccuracy (see Figures A.2 and A.3) is done for achieving a faster calculation of the loss.

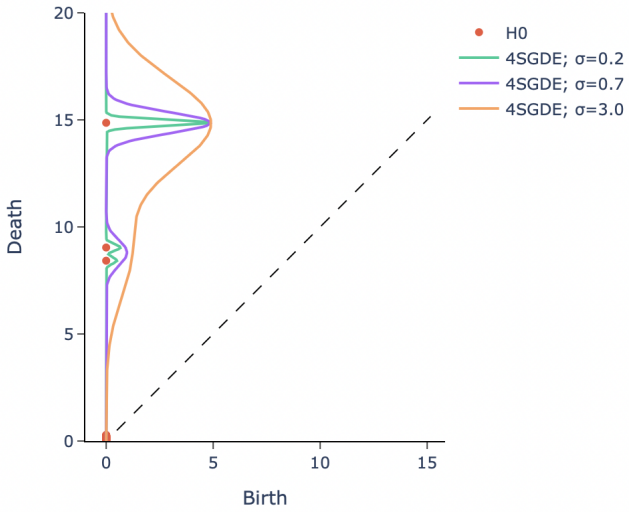

Figure A.2: 4-scaled Gaussian density function applied to a 0-dimensional persistence diagram, for three different values of σ. The function is given in equation (2.29), with $s = 0.0001$. The points near the origin (considered noise) are roughly neglected by the density function.

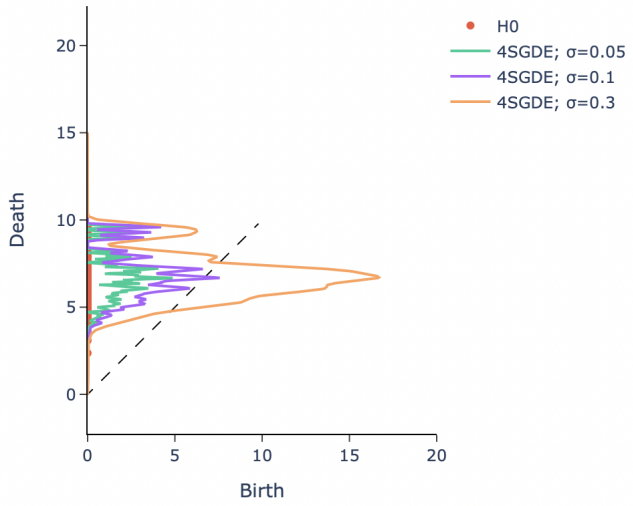

Figure A.3: 4-scaled Gaussian density function applied to a 0-dimensional persistence diagram of a batch of 128 images from the FashionMNIST dataset, for three different values of σ, and 500 sampled points in $[0, 15.0]$. The function is given in equation (2.29), with $s = 0.0005$.

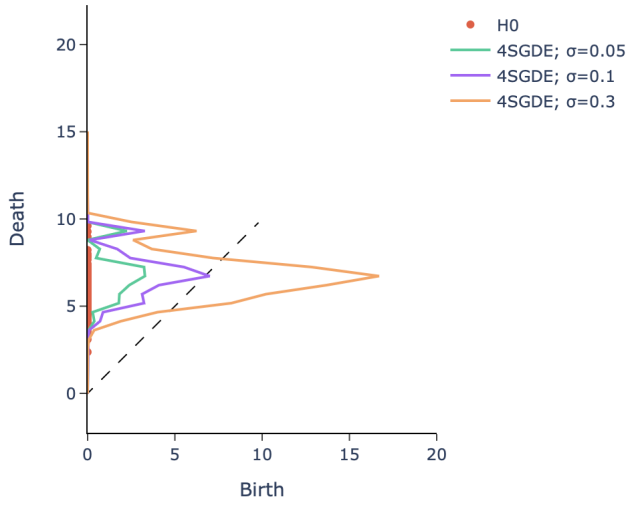

Figure A.4: 4-scaled Gaussian density function applied to a 0-dimensional persistence diagram of a batch of 128 images from the FashionMNIST dataset, for three different values of σ, and 30 sampled points in $[0, 15.0]$. The function is given in equation (2.29), with $s = 0.0005$.

A.3 Synthetic experiments

In each figure, the six images represent, from left to right and from top to bottom: (a) the initial point cloud; (b) its persistence diagram; (c) the reference persistence diagram D_0; (d) the point cloud after finalizing the learning process; (e) its persistence diagram, and (f) the evolution of L_1 during the training process. The code for running the three synthetic experiments and generating the figures and animations can be found in `synthetic_tests.py`.

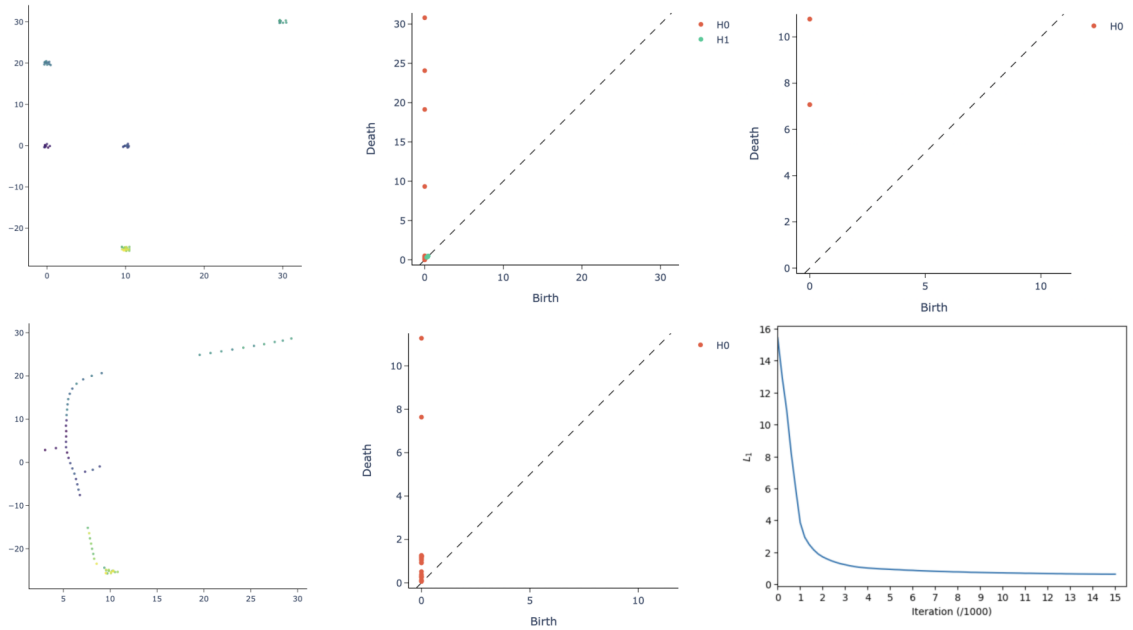

Figure A.5: Synthetic experiment 1: Cluster collapse. The full evolution of the point cloud is given in synthetic1_video.mov.

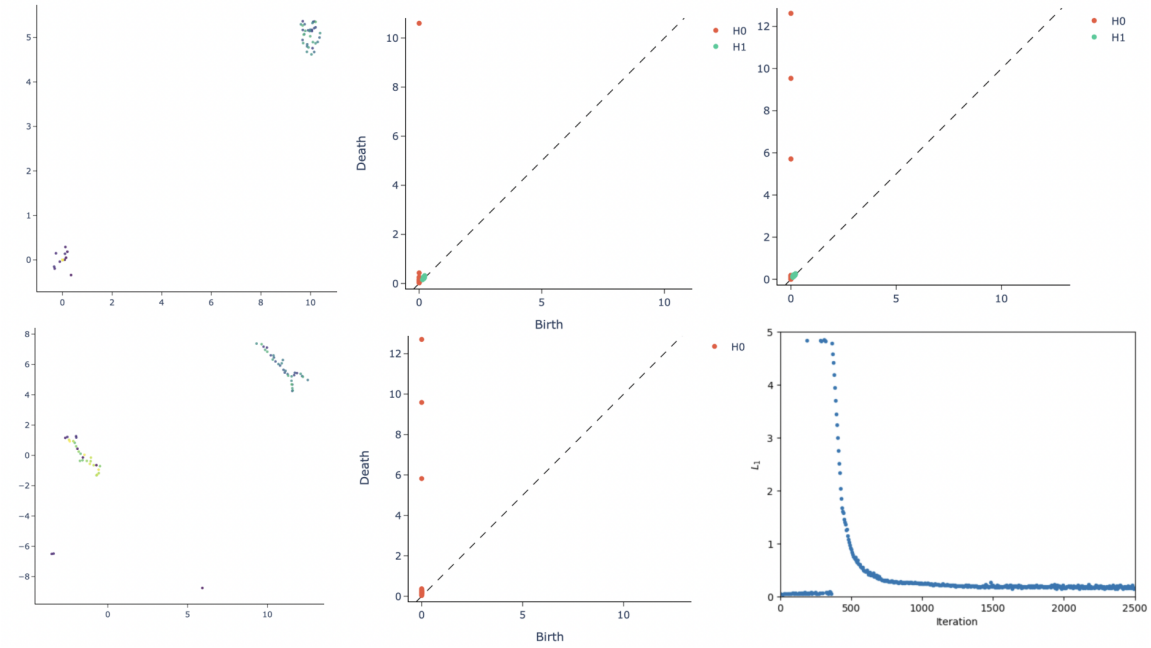

Figure A.6: Synthetic experiment 2: Cluster creation. In this case, L_1 begins with values corresponding to the $L_{\text{push},0}$ term. As a consequence, to enhance clarity we have chosen to scatter the points in the representation of the loss evolution. The full evolution of the point cloud is given in synthetic2_video.mov.

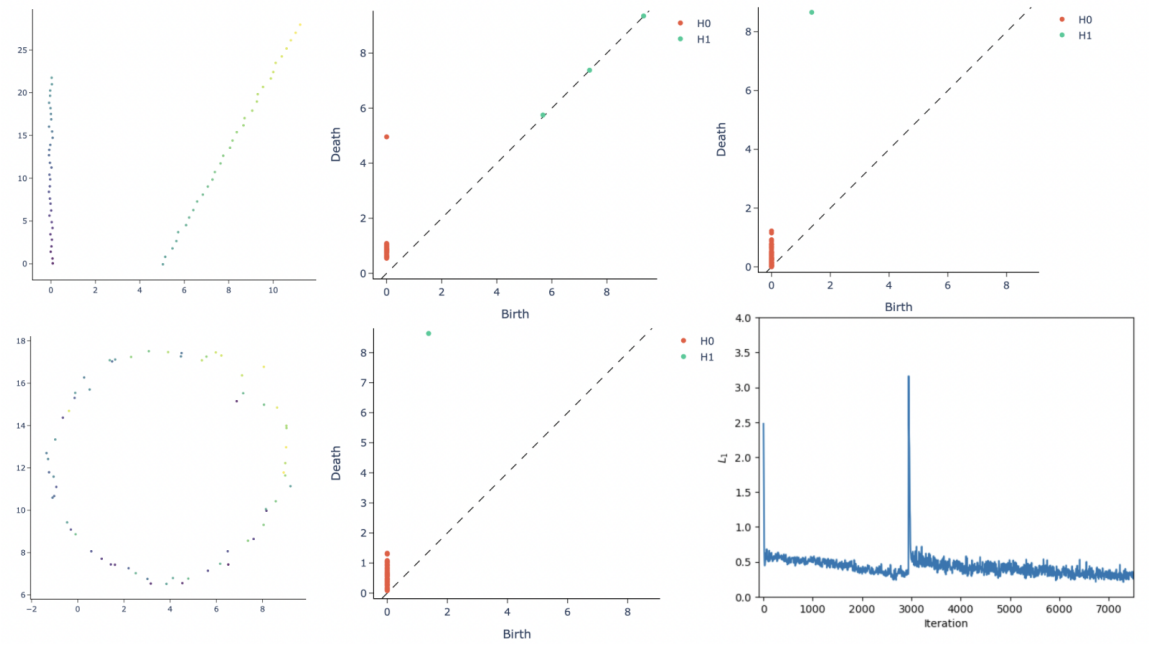

Figure A.7: Synthetic experiment 3: From lines to circles. The full evolution of the point cloud is given in synthetic3_video.mov.

A.4 TopoVAE experiments

A.4.1 Hyperparameters

The optimal hyperparameters for each TopoVAE are given in Table A.1, and have mostly been found in a grid search procedure. However, in the case of TopoVAE4, the set $\{\frac{15i}{29}\}_{i=0}^{29}$ has been chosen due to the nature of the FashionMNIST images; in fact, this set of points has been selected for efficiently capturing properties of the persistence diagrams of batches of FashionMNIST images while keeping a low computation time.

To see how this choice affects the representation of the persistence diagram, see Figures A.3 and A.4. These two images show two persistence diagrams computed from 128-image batches of the FashionMNIST dataset, and the corresponding 4SGDEs when sampling points from $[0, 15]$ at different number of sampled points. Although the choice of 500 sampling points in Figure A.3 may provide a more detailed representation of the density of points in the diagram, we used only 30 sampled points in the experiments in order to ensure a faster computation of the 4SGDE. Furthermore, the value $\sigma_d = 0.1$ seems to properly capture density variations in the diagrams, justifying its choice.

A.4.2 TopoVAEs and VAE0: qualitative comparison

The figures of this subsection, excepted the last figure, are obtained from the program `topovae_tests.py`. In order to train each different TopoVAE, a single line of code has to be changed in order to use the desired topological regularizer. The hyperparameters of each regularizer are, unless mentioned otherwise, those indicated in Table A.1. In the figures representing the evolution of losses, we label the loss corresponding to VAE0 and TopoVAE as "Conv-VAE" and "Conv-TopoVAE", respectively, where "Conv" stands for Convolutional.

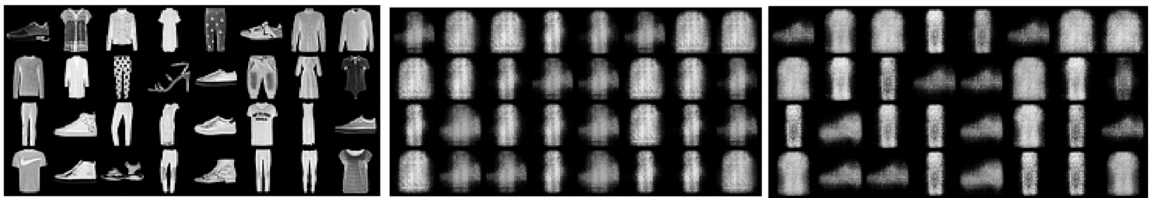

Figure A.8: Comparison of TopoVAE1 and VAE0 in early training (iteration 50/469). TopoVAE1 is used with $(\omega_0, \omega_1) = (5.0, 5.0)$.

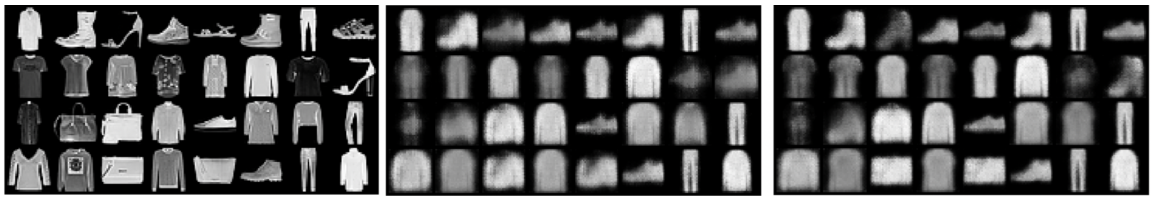

Figure A.9: Comparison of performance between TopoVAE1 and VAE0, in a more advanced stage of training (iteration 100/469).

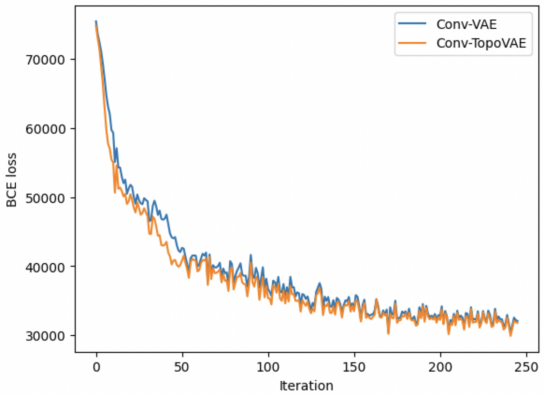

Figure A.10: Evolution of the BCE lossses of TopoVAE1 ("Conv-TopoVAE") and VAE0 ("Conv-VAE").

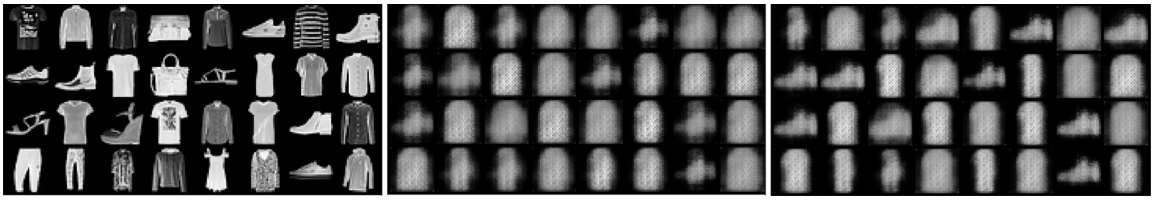

Figure A.11: Comparison of TopoVAE1 and VAE0, using initially cloned models. TopoVAE1 is used with $(\omega_0, \omega_1) = (15.0, 15.0)$. Iteration 50/469.

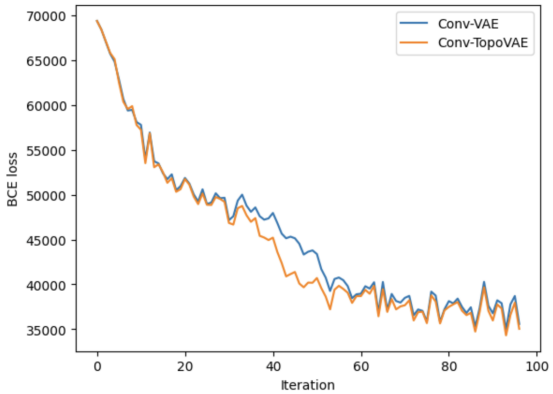

Figure A.12: Evolution of the BCE lossses of TopoVAE1 ("Conv-TopoVAE") and VAE0 ("Conv-VAE") used to the previous figure.

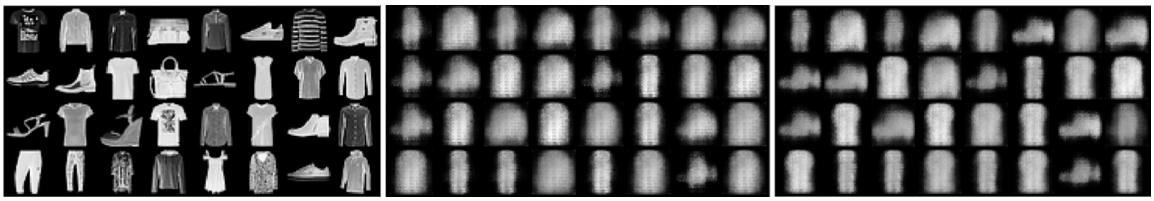

Figure A.13: Comparison of TopoVAE2 and VAE0 using initially cloned models. Iteration 50/469.

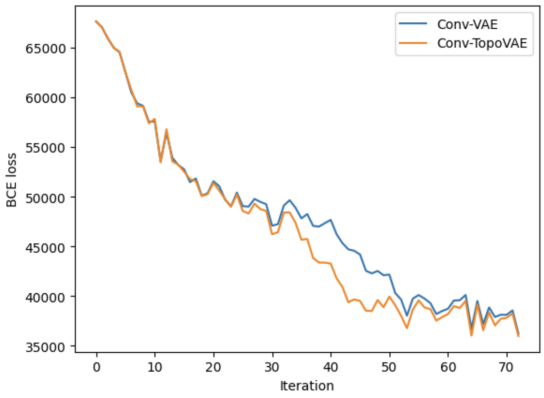

Figure A.14: Evolution of the BCE lossses of TopoVAE2 ("Conv-TopoVAE") and VAE0 ("Conv-VAE"), used in the previous figure.

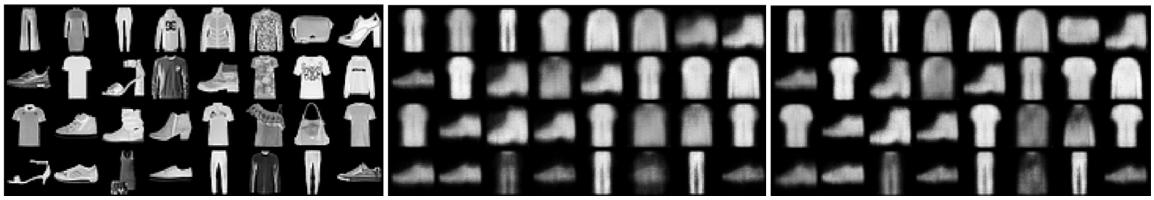

Figure A.15: Comparison of TopoVAE2 and VAE0, using initially cloned models (a different pair from the previous figure using a different seed). Iteration 125/469.

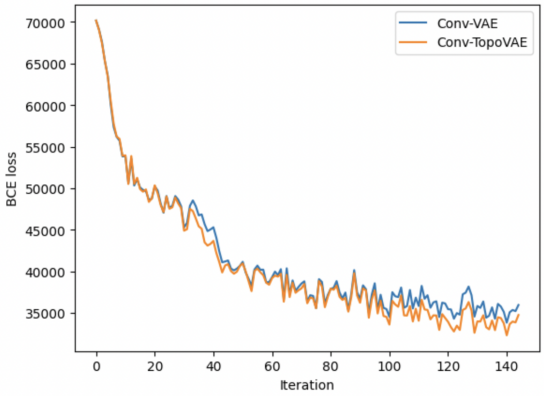

Figure A.16: Evolution of the BCE lossses of TopoVAE2 ("Conv-TopoVAE") and VAE0 ("Conv-VAE"), used in the previous figure.

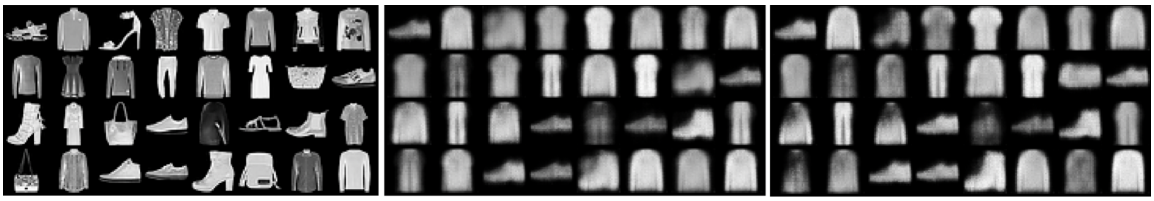

Figure A.17: Comparison of TopoVAE2 and VAE0, using initially cloned models (a different pair from the previous figure using a different seed). Iteration 125/469.

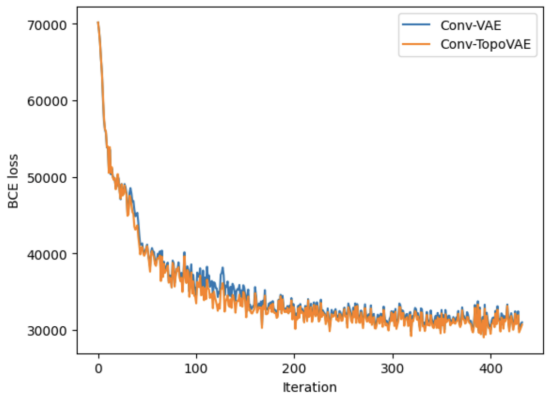

Figure A.18: Evolution of the BCE lossses of TopoVAE2 ("Conv-TopoVAE") and VAE0 ("Conv-VAE"), used in the previous figure.

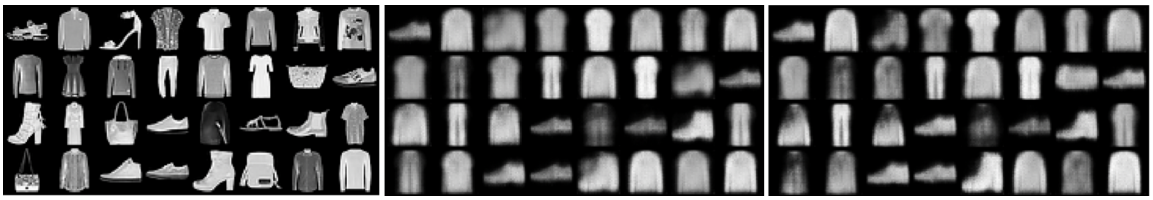

Figure A.19: Comparison of TopoVAE3 and VAE0, using initially cloned models. Iteration 125/469.

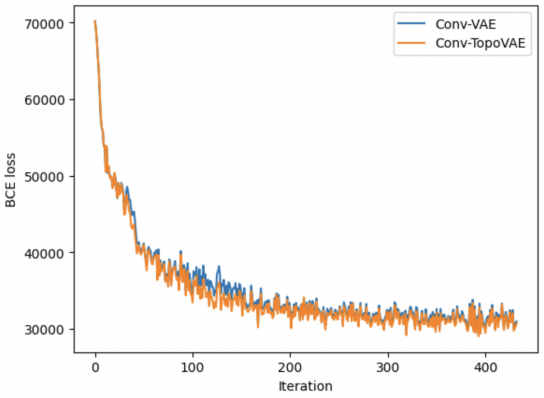

Figure A.20: Evolution of the BCE lossses of TopoVAE3 ("Conv-TopoVAE") and VAE0 ("Conv-VAE"), used in the previous figure.

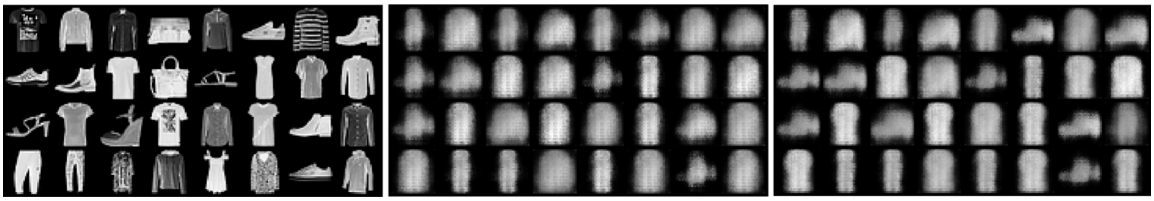

Figure A.21: Comparison of TopoVAE4 and VAE0, using initially cloned models. Iteration 50/469.

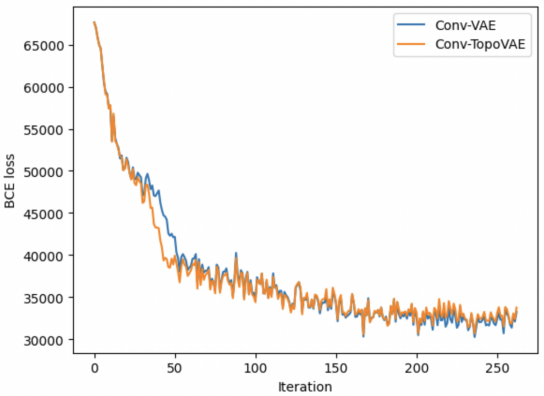

Figure A.22: Evolution of the BCE lossses of TopoVAE4 ("Conv-TopoVAE") and VAE0 ("Conv-VAE"), used in the previous figure.

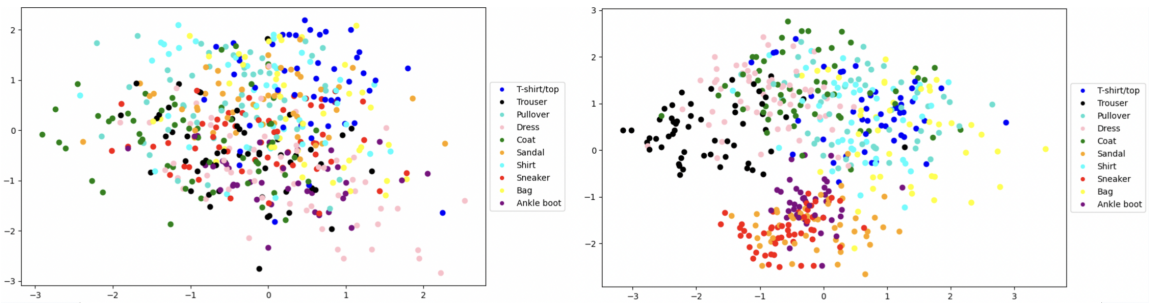

Figure A.23: Visualization of the spatial distributions of latent vectors in latent space. Left: VAE0; right: TopoVAE1(15.,15.), both with $n_Z = 2$. Image is taken after finishing epoch 1 of training.

A.4.3 Comparison across different VAE structures

We compare here the difference of performance between two VAEs. We only work with the case of TopoVAE1 and TopoVAE2 due to their more marked effects on the training process, compared to TopoVAE3 and 4. The code used to obtain the figures can be found in `topovae_simplevae.py`.

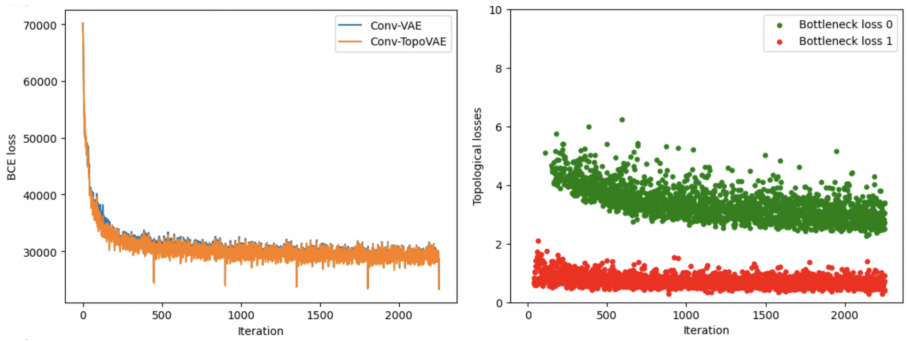

Figure A.24: Evolution of the losses during the first 5 epochs of training VAE0(A) and TopoVAE1(A). Left: BCE losses; right: topological losses. Weights: $(\omega_0, \omega_1) = (15.0, 15.0)$.

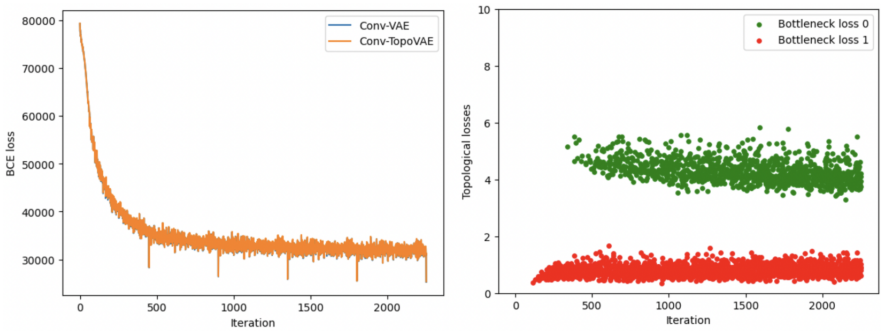

Figure A.25: Evolution of the losses during the first 5 epochs of training VAE0(B) and TopoVAE1(B). Left: BCE losses; right: topological losses. Weights: $(\omega_0, \omega_1) = (15.0, 15.0)$.

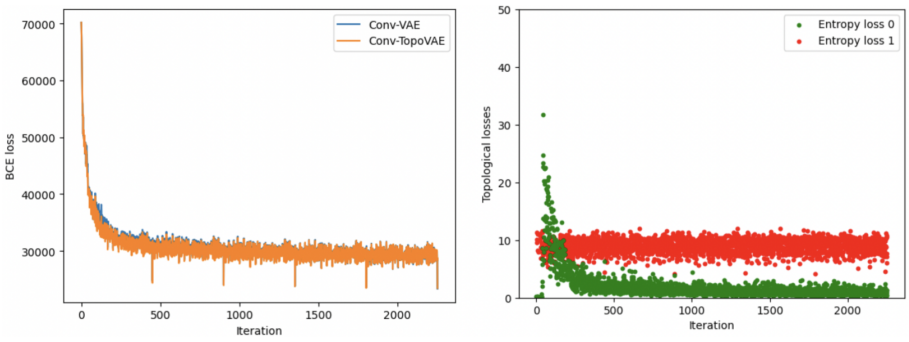

Figure A.26: Evolution of the losses during the first 5 epochs of training VAE0(A) and TopoVAE2(A). Left: BCE losses; right: topological losses. Weights: $(\omega_0, \omega_1, \delta) = (3.5, 3.5, 0.1)$. The entropy loss of degree 0 has been rescaled by a factor of 1000.

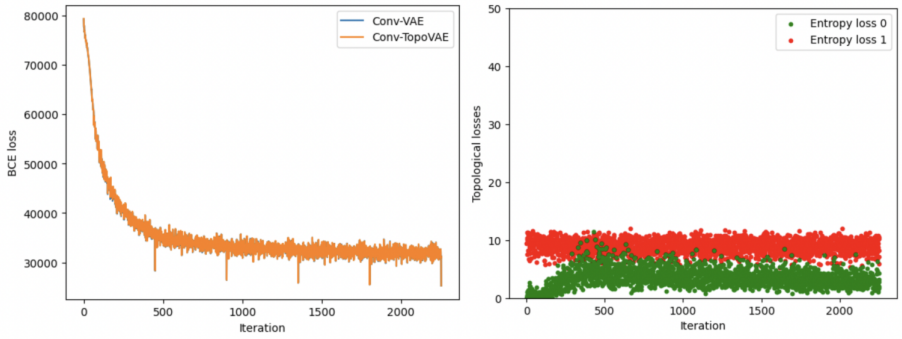

Figure A.27: Evolution of the losses during the first 5 epochs of training VAE0(B) and TopoVAE2(B). Left: BCE losses; right: topological losses. Weights: $(\omega_0, \omega_1, \delta) = (3.5, 3.5, 0.1)$. The entropy loss of degree 0 has been rescaled by a factor of 1000.

A.4.4 Topological regularization in latent space

The code used in these experiments, which involve using the batch of latent vectors in the topological regularizer, can be found in `topovae_z_tests.py`.

Test 1: $n_Z = 10$

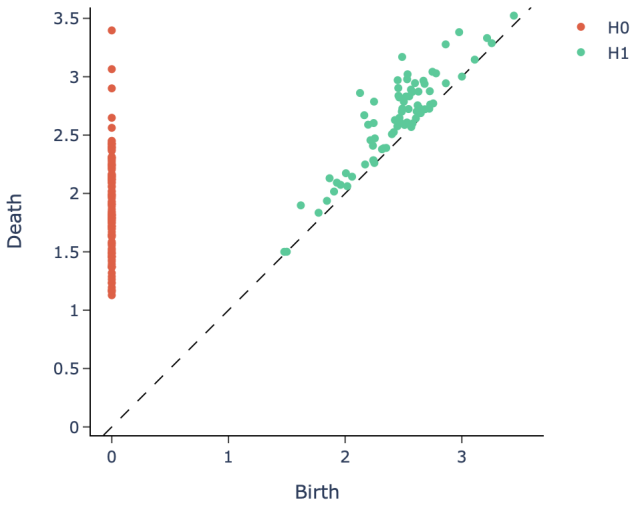

Figure A.28: Example of the persistence diagram (degrees 0 and 1) of a batch of 128 latent vectors produced by VAE0 during training, at iteration 300/469 (corresponding to an advanced stage of training, producing good quality images). The coordinates of points in the diagrams stay in the same areas during the entire training process (for instance, the largest value of death of points stays around 3.0-4.0).

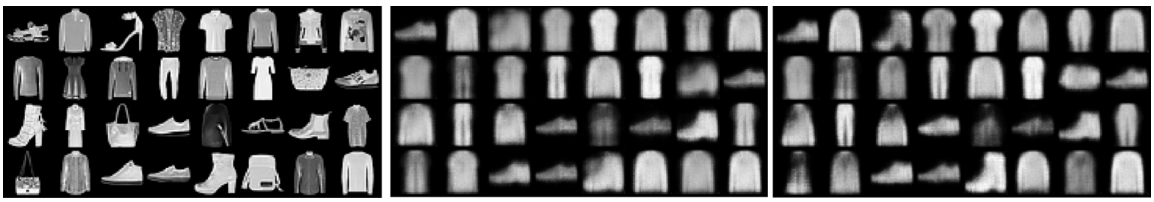

Figure A.29: Comparison of TopoVAE-Z1 and VAE0 in early training (iteration 125/469). TopoVAE-Z1 is used with $(\omega_0, \omega_1) = (15.0, 15.0)$, and the loss applied in the batch of latent vectors.

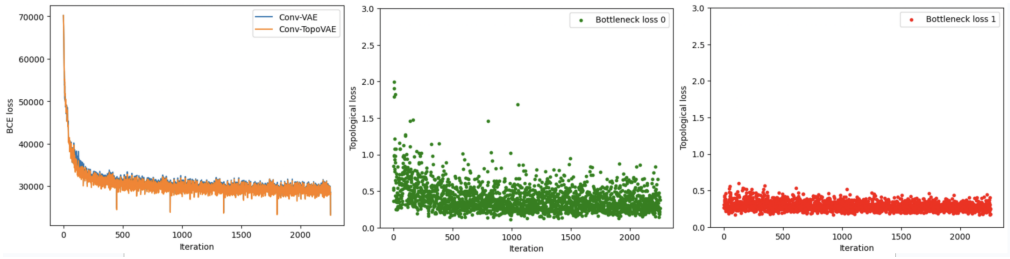

Figure A.30: Evolution of the lossses of TopoVAE-Z1 ("Conv-TopoVAE") and VAE0 ("Conv-VAE") during the first 5 training epochs. Left: BCE losses; middle: bottleneck loss of degree 0, right: bottleneck loss of degree 1.

Test 2: $n_Z = 2$; visualization of the latent distribution

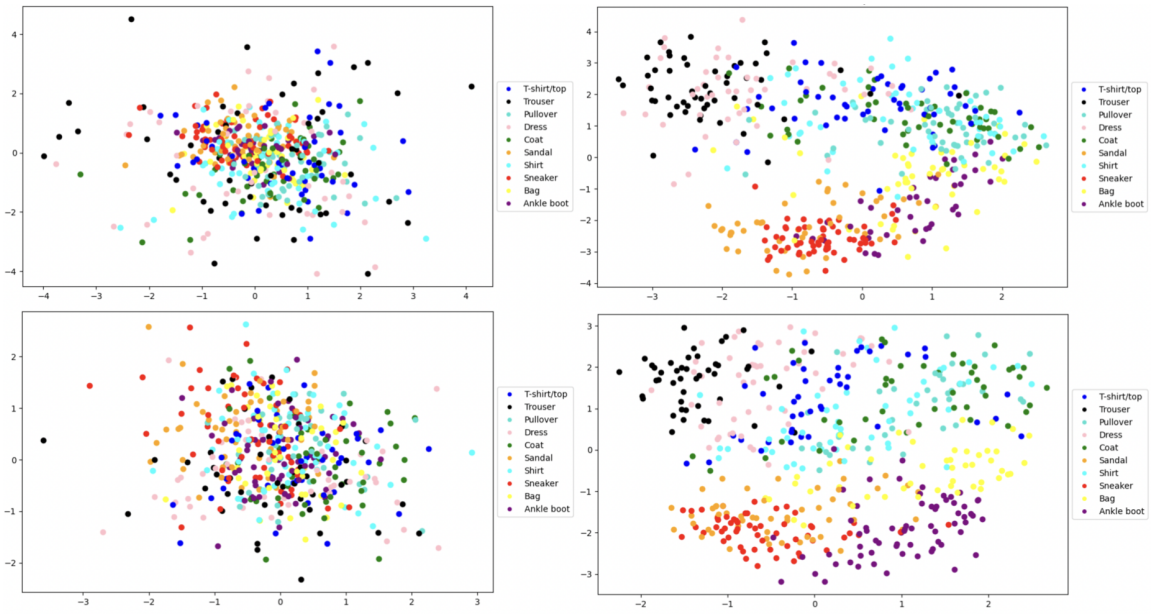

Figure A.31: Distribution of latent vectors of TopoVAE-Z1 and VAE0 during the training process. Left: VAE0; right: TopoVAE-Z1; top: iteration 50; bottom: iteration 100. TopoVAE-Z1 is used with $(\omega_0, \omega_1) = (15.0, 15.0)$, and the loss applied in the batch of latent vectors. The latent vectors come from 500 randomly sampled images from the FashionMNIST dataset. The scale factor of diagrams of the FashionMNIST dataset is kept the same as in Test 1.

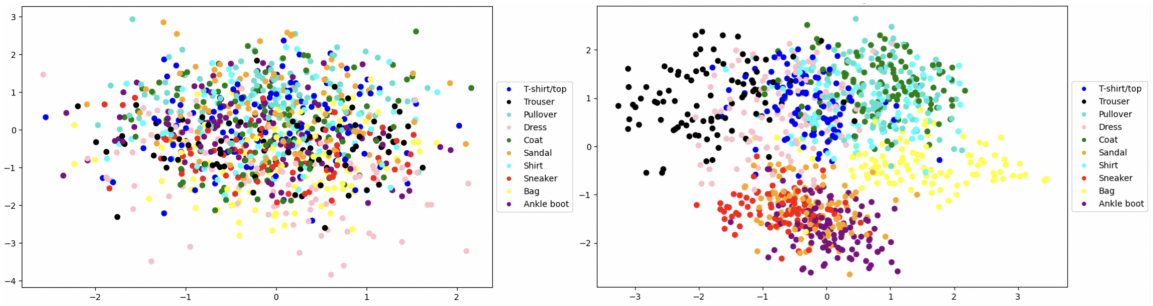

Figure A.32: Distribution of latent vectors of VAE0 (left) and TopoVAE-Z1 (right) after completing epoch 1 of training. Same conditions as previous figure; using 1000 latent vectors.

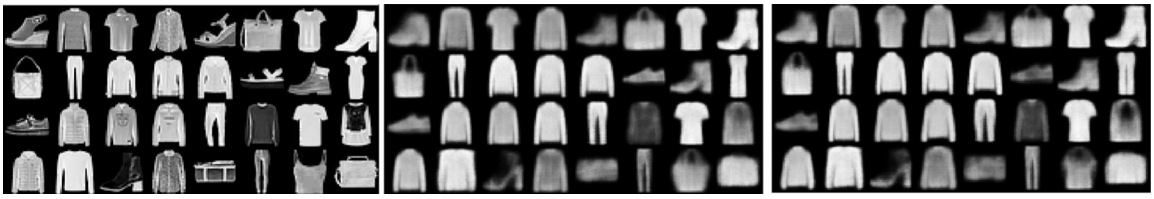

Figure A.33: Images generated by VAE0 and TopoVAE-Z1 at iteration 125. Left: input of real images; middle: VAE0; right: TopoVAE-Z1.

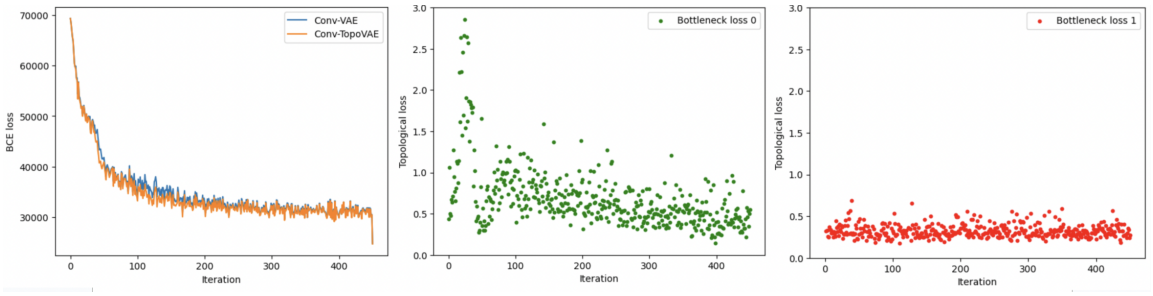

Figure A.34: Corresponding losses of TopoVAE-Z1 and VAE0 during one training epoch.

A.4.5 Summary of results

The following table summarizes the main results related to the TopoVAEs; namely, hyperparameters yielding optimal performance, running time of 10 training steps, and $\bar{r}$ values at training steps 25, 50, and 75.

Model	**Optimal hyperameters**	**Time (10 steps)**	$\bar{r}(25)$	$\bar{r}(50)$	$\bar{r}(75)$
VAE0		1.5 sec.			
TopoVAE1	$(\omega_0, \omega_1) = (5.0, 5.0)$	7.1 sec.	0.3 %	1.2 %	0.2 %
TopoVAE2	$(\omega_0, \omega_1, \delta) = (3.5, 3.5, 0.1)$	2.1 sec.	0.2 %	1.1 %	0.3 %
TopoVAE3	$(\omega_0, \omega_1, \delta, \sigma) = (5.0, 10.0, 0.1, 0.1)$	2.1 sec.	0.2 %	0.7 %	0.2 %
TopoVAE4	$(\omega_0, \omega_1, \mathbf{x}, \sigma_d, s, \sigma) = (10., 5., \{\frac{15i}{29}\}_{i=0}^{29}, 0.1, 0.0005, 0.1)$	17.0 sec.	0.2 %	0.6 %	0.2%

Table A.1: Relevant characteristics of each TopoVAE, including hyperparameters yielding optimal performance, running time of 10 training steps, and $\bar{r}$ values at training steps 25, 50, and 75.

A.5 Structure of VAE.B

The structure of the second VAE, VAE.B, is similar to that of VAE.A. Using latent dimension 10, the encoder realizes a series of transformations of the input, which has an initial shape $(128, 1, 28, 28)$ (i.e., a batch of 128 grayscale 28×28 images). These transformations result in a series of shape transformations of the data that can be expressed as follows:

$$(128, 1, 28, 28) \mapsto (128, 8, 14, 14) \mapsto (128, 1568) \mapsto (128, 10), (128, 10).$$

The first map is done via a convolutional layer, the second map is a re-shaping operation of the data, the fourth map occurs via a fully-connected layer, and the last map occurs by applying in parallel two fully-connected layers. The output corresponds to two batches of μ and Σ vectors. Applying the reparametrization trick, these values are transformed into 128 latent vectors with total shape $(128, 10)$. The latent vectors are then fed into the decoder, and the resulting shape transformations are:

$$(128, 10) \mapsto (128, 1568) \mapsto (128, 8, 14, 14) \mapsto (128, 1, 28, 28).$$

The maps correspond, respectively, to a fully-connected layer, a re-shaping operation, and a transposed convolutional layer. For more details, see `topovae_simplevae.py`.

A.6 Algorithms and code

A.6.1 Main algorithms

Algorithm 1: Total loss calculation in a standard VAE

Input: Training batch $(X_i)_{i=1}^N$
Output: Loss
for $i \leftarrow 1$ **to** N **do**
  Obtain $\mu(X_i)$ and $\Sigma(X_i)$ from the encoder Enc using X_i;
  Generate random vector ϵ_i;
  Compute $z_i = \mu(X_i) + \Sigma(X_i) \odot \epsilon_i$;
  Reconstruct data point $\hat{X}_i$ using the decoder, $\hat{X}_i = f(z_i)$;
  $\text{Loss}_i = \text{KLD}(\mu(X_i), \Sigma(X_i)) + \text{BCE}(\hat{X}_i, X_i)$
return $\sum_{i=1}^N \text{Loss}_i$;

Algorithm 2: Total loss calculation in a TopoVAE

Input: Training batch $(X_i)_{i=1}^N$
Output: Total loss
for $i \leftarrow 1$ **to** N **do**
  Obtain $\mu(X_i)$ and $\Sigma(X_i)$ from the encoder Enc using X_i;
  Generate random vector ϵ_i;
  Compute $z_i = \mu(X_i) + \Sigma(X_i) \odot \epsilon_i$;
  Reconstruct data point $\hat{X}_i$ using the decoder $f(z_i)$;
  $\text{Loss}_i = \text{KLD}(\mu(X_i), \Sigma(X_i)) + \text{BCE}(\hat{X}_i, X_i)$
$L = \sum_{i=1}^N \text{Loss}_i$;
Without backpropagation: Compute, via the Rips filtration, the persistence diagrams D^0 and D^1 (of degrees 0 and 1, respectively) of the generated batch $(\hat{X}_i)_{i=1}^N$, and D_0^0 and D_0^1 for the true batch $(X_i)_{i=1}^N$.;
Compute the loss terms $L_{\text{topo},0}(D^0, D_0^0)$ and $L_{\text{topo},1}(D^1, D_0^1)$ using backpropagation ;
if $L_{topo,0}(D^0, D_0^0)$ *depends on off-diagonal points of* D^0 **then**
  $L = L + \omega_0 \cdot L_{\text{topo},0}(D^0, D_0^0)$
if $L_{topo,1}(D^1, D_0^1)$ *depends on off-diagonal points of* D^1 **then**
  $L = L + \omega_1 \cdot L_{\text{topo},1}(D^1, D_0^1)$
return L;

A.6.2 Code

The entire code used in this work can be found in the repository TopoGEN. In particular, the following files have been used for generating the figures in the Appendix and experiments described in Chapter 3:

- `synthetic_tests.py`: for the synthetic tests of Appendix A.3.

- `densityfctn.py`: for plotting the 4SGDE function, used for plotting Figure A.2 in Appendix A.2.
- `topovae_fashionmnist_tune.py`: provides the full code for training all the TopoVAE models in the FashionMNIST dataset, but also includes a code at the end that allows to finetune the hyperparameters of the density estimator according to the shape of the persistence diagrams obtained from batches of the FashionMNIST dataset. Figures A.3 and A.4 in Appendix A.2 come from this finetuning extension.
- `topovae_tests.py`: provides the full code for training all the TopoVAE models in the FashionMNIST dataset, providing figures of produced images of the TopoVAE model trained and the VAE0 model, and the curves of evolution of the BCE and KLD losses. By changing a single one of code, any TopoVAE model can be trained.
- `topovae_plot_latent.py`: same tests as previous but done with $n_Z = 2$ and includes plots of the distribution of latent vectors. Used to produce Figure A.23.
- `tests_losses.py`: realizes the tests for obtaining the $\bar{r}$ values as described in Subsection 3.5.3.
- `topovae_simplevae.py`: used in Appendix A.4.3. Uses two structures of VAE model (normal VAE, and simpler VAE (fewer parameters)), and compares the performance of TopoVAE models and VAE0 on both VAE structures.
- `topovae_z_tests.py`: applies the topological regularizer of TopoVAE1 on batches of generated latent vectors instead of generated images. Also provides plots of curves of the evolution of losses. Used in Appendix A.4.4.
- `tests_fashionmnist.ipynb`, `tests_cifar.ipynb`, and `diversity_fashionmnist.ipynb`: used in Appendix B.4 for studying the relation between image quality and diversity and the magnitude of topological regularizers.

Note that the programs `topovae_tests.py`, `topovae_plot_latent.py` and `topovae_simplevae.py` can be modified to use any topological regularizer. In particular, each of the following lines calls the full loss function of each TopoVAE:

1. TopoVAE1: loss = loss_fctn2(recon_batch, data, mean, log_var, dgm, dgm2, 5., 5.)
2. TopoVAE2: loss = loss_fctn3(recon_batch, data, mean, log_var, dgm, dgm2, 3.5, 3.5, 0.1)
3. TopoVAE3: loss = loss_fctn4(recon_batch, data, mean, log_var, dgm, dgm2, 5., 10., 0.1)
4. TopoVAE4: loss = loss_fctn5(recon_batch, data, mean, log_var, dgm, dgm2, 10., 5.).

Appendix B

Supplementary material

B.1 Additional proofs

Lemma B.1. *Let $D \in \mathrm{Bar}$, $r \in \bar{\mathbb{N}}$, and let $V_i \colon \mathrm{Bar} \to \mathbb{R}$ be maps on barcodes, with $i = 1, \dots, N$. Assume the maps $V_1, \dots, V_N$ are r-differentiable in D. Then, the map $\left(\sum_{i=1}^{N} V_i\right) : \mathrm{Bar} \to \mathbb{R}$ is r-differentiable in D.*

Proof. We prove the result for the case $N = 2$, and by applying iterative reasoning it follows that the lemma is true for any $N \geq 2$. Recall the Definition 2.5 of r-differentiability of maps on barcodes, and take a pair m, n and a barcode $\tilde{D} \in \mathbb{R}^{2m+n}$ such that $Q_{m,n}(\tilde{D}) = D$. We want to see that there exists an open neighborhood U of $\tilde{D}$ such that the map $(V_1 + V_2) \circ Q_{m,n} \colon \mathbb{R}^{2m+n} \to \mathbb{R}$ is C^r in U. Since V_1 is r-differentiable in D, there is an open neighborhood U_1 of $\tilde{D}$ such that $V_1 \circ Q_{m,n}$ is C^r in U_1. Analogously, since V_2 is r-differentiable in D, there is another open neighborhood U_2 of $\tilde{D}$ such that $V_2 \circ Q_{m,n}$ is C^r in U_2. Therefore, the map $(V_1 + V_2) \circ Q_{m,n} \colon \mathbb{R}^{2m+n}$ is C^r in $U_1 \cap U_2$, which is an open neighborhood of $\tilde{D}$. Since this argument holds for any m, n and $\tilde{D} \in \mathbb{R}^{2m+n}$ such that $Q_{m,n}(\tilde{D}) = D$, we conclude that the map on barcodes $V_1 + V_2$ is r-differentiable in D. □

Lemma B.2. *Let $i, j, k, l \in \{1, \dots, n\}$ be four numbers such that $i \neq j$, $k \neq l$, and $\{i, j\} \neq \{k, l\}$. We define the function*

$$\begin{aligned} f_{ijkl} \colon \mathbb{R}^{nd} &\to \mathbb{R} \\ P &\mapsto \|p_i - p_j\|_2^2 - \|p_k - p_l\|_2^2. \end{aligned}$$

Then, $\{P \in \mathbb{R}^{nd} \colon \nabla f_{ijkl}(P) = \mathbf{0}\} = \{p_i = p_j\} \cap \{p_i = p_j\}$.

Proof. From the hypothesis, we can assume $i < l$; $i \neq j, k, l$ and $l \neq i, j, k$ without loss of generality. Take an arbitrary point $P = (p_1, \dots, p_n) = (p_1^1, \dots, p_1^d, \dots, p_n^1, \dots, p_n^d)$; then, the gradient of f_{ijkl} at P is given by:

$$\begin{aligned} \nabla f_{ijkl}(P) = &\left[2\sum_{\alpha=1}^{n} (p_i^\alpha - p_j^\alpha)\mathbf{e}_i^\alpha - 2\sum_{\alpha=1}^{n} (p_k^\alpha - p_l^\alpha)\mathbf{e}_l^\alpha \right] \\ &+ \left[-2\sum_{\alpha=1}^{n} (p_i^\alpha - p_j^\alpha)\mathbf{e}_j^\alpha + 2\sum_{\alpha=1}^{n} (p_k^\alpha - p_l^\alpha)\mathbf{e}_k^\alpha \right], \end{aligned}$$

where $\mathbf{e}_q^\alpha$ is the unitary vector corresponding to the α-th coordinate of the q-th point, i.e., the $((q-1)\cdot d+\alpha)$-th coordinate in $\mathbb{R}^{nd}$. In order to have $\nabla f_{ijkl}(P)=\mathbf{0}$, we need to have the first term equal to zero, which implies $(p_i^\alpha - p_j^\alpha)=0$ and $(p_k^\alpha - p_l^\alpha)=0$ for $\alpha=1,\dots,d$. In other words, we need $p_i=p_j$ and $p_k=p_l$. Conversely, if $p_i=p_j$ and $p_k=p_l$, then $\nabla f_{ijkl}(P)=\mathbf{0}$. □

Lemma B.3. *If $D\in\widehat{\mathrm{Bar}}$, then there exists $\epsilon>0$ such that, for all D' ϵ-close to D, $d_\infty(D',D_0)>\epsilon$ and there exists an optimal matching between D' and D_0 sending $D'\cap\Delta_\epsilon$ onto Δ^∞.*

Proof. The proof is based on the proof given in [10]. We first give a sketch of the proof: the idea is to first find a specific ϵ with specific properties that are used in the remainder of the proof. Next, we assume that the conclusion of the lemma is false in order to reach a contradiction. In particular, we take any D' ϵ-close to D and assume that any optimal matching γ from D' to D_0 maps some interval in $D'\cap\Delta_\epsilon$ to some off-diagonal interval of D. In addition, we impose γ to have a maximal number of off-diagonal intervals from D or D_0 matched to the diagonal. The key step of this proof is showing that if we take these two matched off-diagonal intervals and match them instead to their diagonal projections, we always obtain another optimal matching —this is mainly proved with equations (B.3) and (B.4). But this is a contradiction with the definition of γ, since the new matching has two more off-diagonal intervals sent to the diagonal than γ. We now proceed to the full proof.

Let $D\in\widehat{\mathrm{Bar}}_n$, then we define α as

$$\alpha \equiv \min_{(b_0,d_0)\in D_0}\left|\frac{|d_0-b_0|}{2}-d_\infty(D,D_0)\right| \tag{B.1}$$

i.e., it is the minimal difference between the distances of off-diagonal intervals of D_0 to the diagonal, and $d_\infty(D,D_0)$. Since $D\in\widehat{\mathrm{Bar}}$ and by definition of $\widehat{\mathrm{Bar}}$, $\alpha>0$. In addition, $d_\infty(D,D_0)>0$, because for any point of D_0 in the diagonal the distance to its diagonal projection is 0, therefore by definition of $\widehat{\mathrm{Bar}}$, $d_\infty(D,D_0)$ cannot be zero. As a consequence, there exists $\epsilon>0$ such that $\epsilon<\frac{1}{2}\min(d_\infty(D,D_0),\alpha)$ —since both values are strictly positive.

We now prove that the conclusion of the lemma "$d_\infty(D',D_0)>\epsilon$ and there exists an optimal matching between D' and D_0 sending $D'\cap\Delta_\epsilon$ onto Δ^∞" holds for any $D'\in\mathcal{B}(D,\epsilon)$.

Let $D'\in\mathcal{B}(D,\epsilon)$. Since $\epsilon<\frac{d_\infty(D,D_0)}{2}$, then $d_\infty(D',D_0)>\epsilon$. This is a consequence of the triangular inequality: we have $2\epsilon<d_\infty(D,D_0)\le d_\infty(D,D')+d_\infty(D',D_0)<\epsilon+d_\infty(D',D_0)$, so $\epsilon<d_\infty(D',D_0)$.

Now, we assume for the sake of contradiction that there is no optimal matching between D' and D_0 sending $D'\cap\Delta_\epsilon$ to Δ^∞. For simplicity, we restrict the matchings to the set of optimal matchings that can send off-diagonal points of D' and D_0 to Δ^∞ *only* by orthogonal projections. We note this set $\Gamma^*(D',D_0)$, and it is finite and non-empty. Notice that $\Gamma^*(D',D_0)$ contains all optimal matchings: if an optimal matching sends an off-diagonal interval p to the diagonal not by orthogonal projection, then a matching that only differs by sending p to the diagonal by orthogonal projection would have a strictly lower cost, which is a contradiction.

We also define the Δ-degree of a matching $\gamma\in\Gamma^*(D',D_0)$ as the number of off-diagonal points of D' and D_0 that are sent to Δ^∞. We take γ with maximal Δ-degree. By the

assumption (that there is no optimal matching between D' and D_0 sending $D' \cap \Delta_\epsilon$ to Δ^∞) there is an off-diagonal interval $(b', d') \in D' \cap \Delta_\epsilon$ sent by γ to an off-diagonal interval $(b_0, d_0) \in D_0$. We will next to prove that in any possible situation, we always have the inequality $\frac{|b'-d'|}{2} < d_\infty(D', D_0)$ and $\frac{|b_0-d_0|}{2} < d_\infty(D', D_0)$. Proving these two inequalities is essential, since they show that if we replace the matching (b', d') to (b_0, d_0) by their diagonal orthogonal projections, we obtain a new optimal matching but with Δ-degree larger than γ, which is a contradiction.

From the definition of α, it follows that for the given (b_0, d_0), we have the inequality $\left|\frac{|d_0-b_0|}{2} - d_\infty(D, D_0)\right| \geq \alpha$. Therefore, we have two cases:

(i) $\frac{|d_0-b_0|}{2} \geq d_\infty(D, D_0) + \alpha$, or

(ii) $\frac{|d_0-b_0|}{2} \leq d_\infty(D, D_0) - \alpha$.

In case (i), we have:

$$\begin{aligned}
\|(b_0, d_0) - (b',d')\|_\infty &= \|(b_0, d_0) - (\frac{b'+d'}{2}, \frac{b'+d'}{2}) + (\frac{b'+d'}{2}, \frac{b'+d'}{2}) - (b', d')\|_\infty \\
&\overset{(a)}{\geq} \|(b_0, d_0) - (\frac{b'+d'}{2}, \frac{b'+d'}{2})\|_\infty - \|(\frac{b'+d'}{2}, \frac{b'+d'}{2}) - (b', d')\|_\infty \\
&\overset{(b)}{\geq} \frac{|d_0 - b_0|}{2} - \frac{|d' - b'|}{2} \overset{(c)}{\geq} d_\infty(D, D_0) + \alpha - \epsilon \\
&\overset{(d)}{\geq} d_\infty(D', D_0) - \epsilon + \alpha - \epsilon \overset{(e)}{>} d_\infty(D', D_0).
\end{aligned} \tag{B.2}$$

The first inequality (a) comes directly from the triangle inequality, and the second inequality (b) comes from the fact that the distance of (b_0, d_0) to any diagonal projection is greater or equal than the distance to its orthogonal projection into the diagonal, $\frac{|d_0-b_0|}{2}$, in other words $\|(b_0, d_0) - (\frac{b'+d'}{2})\|_\infty \geq \frac{|d_0-b_0|}{2}$. The second term after (b) comes from rewriting the second term of the line above. We next justify inequality (c): the term $d_\infty(D, D_0) + \alpha$ comes because we are in case (i), so $\frac{|d_0-b_0|}{2} \geq d_\infty(D, D_0) + \alpha$, and the term $-\epsilon$ appears because $(b', d') \in \Delta_\epsilon$, so $\frac{|d'-b'|}{2} < \epsilon$, i.e., $-\frac{|d'-b'|}{2} > -\epsilon$. (d) is justified as follows: we have $d_\infty(D, D') < \epsilon$, so $d_\infty(D_0, D') \leq d_\infty(D_0, D)+\epsilon$ from the triangle inequality, and rearranging we have $d_\infty(D, D_0) \geq d_\infty(D', D_0) - \epsilon$. Finally, (e) is true since by definition $\epsilon < \alpha/2$, so $\alpha - 2\epsilon > 0$.

Now, notice that (B.2) leads to a contradiction, since γ is an optimal matching so its cost should be lower or equal to $d_\infty(D', D_0)$. Therefore, scenario (i) cannot happen.

On the other hand, in case (ii) we have:

$$\frac{|d_0 - b_0|}{2} \leq d_\infty(D, D_0) - \alpha \leq d_\infty(D', D_0) + \epsilon - \alpha < d_\infty(D', D_0). \tag{B.3}$$

The first inequality is the definition of case (ii), the second inequality comes from the triangle inequality, and the third inequality from the fact that $\epsilon < \alpha$. In addition, we also have:

$$\frac{|d' - b'|}{2} \leq \epsilon < d_\infty(D, D_0) \leq \frac{d_\infty(D', D_0) + \epsilon}{2} < d_\infty(D', D_0). \tag{B.4}$$

The first inequality comes from the fact that $(b', d') \in \Delta_\epsilon$, the second one from the definition of ϵ, the third one from the triangle inequality and the last one is proved as follows: from

previous inequalities of (B.4) we have $\epsilon \leq \frac{d_\infty(D',D_0)}{2} + \frac{\epsilon}{2}$, so $\epsilon \leq d_\infty(D', D_0)$. So from (B.3) and (B.4) we have that both $\frac{|d_0-b_0|}{2}$ and $\frac{|d'-b'|}{2}$ are strictly smaller than $d_\infty(D', D_0)$, and this always happens since we have seen that (i) cannot happen. Therefore, modifying γ by sending (b_0, d_0) and (b', d') to their diagonal orthogonal projections, we obtain a new matching $\tilde{\gamma}$ that is also in $\Gamma^*(D', D_0)$ —which is true since (B.3) and (B.4) show that $\tilde{\gamma}$ is still an optimal matching— with Δ-degree strictly larger than the Δ-degree of γ. This is a contradiction since we had taken γ with maximal Δ-degree. Having reached a contradiction, we have proven that for any D' in $\mathcal{B}(D, \epsilon)$, there is an optimal matching between D' and D_0 that sends $D' \cap \Delta_\epsilon$ onto the diagonal. □

B.2 Variational autoencoders: full derivation of the loss

A VAE is an unsupervised machine learning model that maps high-dimensional data to a lower-dimensional latent space using an encoder, and reconstructs the data from this latent space using a decoder. We denote to the latent space as $\mathcal{Z}$, and refer to its elements as *latent vectors*, noted z.

Both the encoder and the decoder of the VAE are neural networks, parametrized by their respective sets of parameters ϕ and θ. The decoder maps a latent vector to a data point in $f(z;\theta) \in \mathbb{X}$. If θ is fixed and z is sampled according to a probability density function $P(z)$ on $\mathcal{Z}$, then $f(z;\theta)$ is a random vector in $\mathbb{X}$. The objective of the VAE is to optimize θ such that the $f(z;\theta)$'s are points from the data distribution. In other words, θ has to be optimized in order to maximize, for every X in the dataset, the following quantity:

$$P(X) = \int_{\mathcal{Z}} P(X \mid z;\theta)P(z)\, dz, \tag{B.5}$$

where $P(X)$ is the probability that the data point X is produced by the generative process $z \mapsto f(z;\theta)$ for all possible z in $\mathcal{Z}$ following the distribution $P(z)$ [25]. In VAEs, $P(z)$ is defined as a standard distribution $\mathcal{N}(z \mid 0, Id)$ [25]. The term $P(X \mid z;\theta)$ represents the probability that $f(z;\theta)$ *looks like* X. This function is often defined as a Gaussian distribution $\mathcal{N}(X \mid f(z;\theta), \sigma \cdot Id)$, where σ is a hyperparameter. The intuition is that with this definition, $P(X \mid z;\theta)$ is 1 when $X = f(z;\theta)$, and is *close* to 1 when X is a point close to $f(z;\theta)$. The larger the distance between X and $f(z;\theta)$, the closer $P(X \mid z;\theta)$ is to 0. Note that other distributions can be used as long as they are continuous on θ and can they be numerically computed. The continuity θ is key for optimizing the model: in fact, initially the generated data points do not match the points from the dataset, and gradients are computed in order to increase the outputted probabilities. If we had, for instance, a Dirac delta distribution being only 1 at $f(z;\theta) = X$ and 0 elsewhere, the gradients of $P(X \mid z;\theta)$ in these erroneous points would be 0 and we would not be able to improve θ via gradient descent. On the other hand, a Gaussian or another continuous (on θ) distribution allows non-zero gradients that improve the parameters of the decoder.

In summary, the idea behind equation (B.5) is that if the decoder maximizes $P(X)$, it will then create data points that *look like* those from the dataset.

The question is now: how can a VAE compute (B.5) while allowing backpropagation? A straightforward answer would be to sample a large number n of z's and compute a discrete sum that would approximate (B.5). However, this approach is excessively time-consuming from a computational perspective [25]. Instead, the key idea of the VAE is to

only look at values z that may yield a large value of $P(X \mid z;\theta)$, since the other z's almost do not contribute to the integral (B.5), so they can be neglected. To that aim, we use another probability distribution, $Q(z \mid X)$ which, given a data point X, tries to provide the distribution of latent vectors z that are likely to output X through the decoder. We then approximate $P(X)$ with $E_{z\sim Q}P(X \mid z)$, which is in fact equivalent to (B.5) but with z following the distribution $Q(z \mid X)$ instead of $P(z)$. The advantage of this alternative quantity is that when calculating $E_{z\sim Q}P(X \mid z)$, we only take into account the space of z's likely under $Q(z \mid X)$, which is smaller than the region of z's likely under $P(z)$. This makes the computation of $E_{z\sim Q}P(X \mid z)$ easier and faster compared to $P(X)$. The form of $Q(z \mid X)$ could be arbitrary, however in VAEs $Q(z \mid X)$ is often restricted to be a Gaussian $\mathcal{N}(z \mid \mu(X), \Sigma(X))$ where Σ is a diagonal matrix [25]. The encoder of the VAE plays then the role of $Q(z \mid X)$: given an input X, the encoder outputs $\mu(X;\phi)$ and $\Sigma(X;\phi)$, establishing the distribution $Q(z \mid X)$, $\mathcal{N}(\mu(X;\phi), \Sigma(X;\phi))$. Therefore, the role of the encoder is to provide the distribution that tells *what are the latent vectors in $\mathcal{Z}$ more likely to reproduce X via the decoder.* Note that this leads to $Q(z \mid X)$ being parametrized by ϕ, however we do not explicitly specify the dependence to avoid unnecessarily complex notation.

There is, however, a fundamental a problem with the approach we have just described: using $Q(z \mid X)$ instead of $P(z)$, we are computing $E_{z\sim Q}P(X \mid z)$, and not $P(X)$. We thus need to obtain $P(X)$ from $E_{z\sim Q}P(X \mid z)$. The answer lies in the following equation [24]:

$$\log P(X) - D_{KL}[Q(z \mid X) \,||\, P(z \mid X)] = E_{z\sim Q}(\log P(X \mid z)) - D_{KL}[Q(z \mid X) \,||\, P(z)], \tag{B.6}$$

where D_{KL} is the Kullback-Leibler (KL) divergence, and provides a measure of the dissimilarity between two probability distributions. It is defined as follows: given two probability distributions $F(z)$ and $G(z)$ over $\mathcal{Z}$, $D_{KL}[F(z) \,||\, G(z)] := E_{z\sim F}(\log F(z) - \log G(z))$. In the left-hand side of equation (B.6), the KL-divergence represents *how well Q is selecting the region we take into account for approximating $P(X)$*, or *how well Q is choosing z's that output data points similar to X*. It is important to note that equation (B.6) holds for any distribution Q, independently of how well it matches the distribution $P(z \mid X)$. This makes it applicable at any stage of the training process.

In addition, the KL divergence is non-negative [24] and only 0 if both distributions are equal. As a consequence, the right-hand side of (B.6) is a lower bound of $\log P(X)$. We thus have the inequality:

$$\log P(X) \geq \mathcal{L}(\phi, \theta; X) := -D_{KL}[Q(z \mid X) \,||\, P(z)] + E_{z\sim Q}(\log P(X \mid z)). \tag{B.7}$$

We next explain how to compute this lower bound and optimize it via stochastic gradient descent.

First, recall that Q has been restricted to be a Gaussian distribution. This particular choice has an essential advantage: since both distributions $Q(z \mid X)$ and $P(z)$ are Gaussian, their KL-divergence is given in closed form. In fact, viewing μ and Σ as vectors of dimension J, the KL-divergence of $Q(z \mid X)$ and $P(z)$ can be expressed as [24]:

$$-D_{KL}[Q(z \mid X) \,||\, P(z)] = \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\Sigma_j)^2) - (\mu_j)^2 - (\Sigma_j)^2). \tag{B.8}$$

For the second term of (B.7), we use an approximation via Monte Carlo estimates. This requires taking samples z^l according to the distribution $Q(z \mid X)$. However, this is a-priori a non-differentiable operation with respect to the parameters of $Q(z \mid X)$, ϕ. A solution to this problem is the *reparametrization trick* [24, 25]: we first sample another noise variable $\epsilon \sim \mathcal{N}(0, Id)$, and then obtain z using a differentiable transformation:

$$z = \mu(X; \phi) + \Sigma(X; \phi) \odot \epsilon, \tag{B.9}$$

where $\odot$ is the element-wise product. Using this approach, z becomes continuous on ϕ and backpropagation can be effectively performed [24]. After taking L samples ϵ^l $(l = 1, \ldots, L)$ of the auxiliary noise variable, we calculate the Monte Carlo estimate of $E_{z\sim Q}(\log P(X \mid z))$:

$$E_{z\sim Q}(\log P(X \mid z)) \simeq \frac{1}{L}\sum_{l=1}^{L} \log(P(X \mid z^l);\ z^l = \mu(X; \phi) + \Sigma(X; \phi) \odot \epsilon^l. \tag{B.10}$$

Combining equations (B.8) and (B.10), we obtain an approximation of the lower bound from equation (B.7):

$$\begin{aligned} &\mathcal{L}(\phi, \theta; X) \simeq \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\Sigma_j)^2) - (\mu_j)^2 - (\Sigma_j)^2) + \frac{1}{L}\sum_{l=1}^{L} \log(P(X \mid z^l)) \\ &\text{where } z^l = \mu(X; \phi) + \Sigma(X; \phi) \odot \epsilon^l;\ \epsilon^l \sim \mathcal{N}(0, Id). \end{aligned} \tag{B.11}$$

We denote the function in the right-hand side of equation (B.11) as $\tilde{\mathcal{L}}(\phi, \theta; X, L)$. This function is continuous on ϕ and θ, and can be computed and optimized via stochastic gradient descent [25]. The term $\log P(X \mid z^l)$ corresponds to a Bernoulli or Gaussian distribution whose parameters are given by $f(z; \theta)$, and the choice depends on the type of data. In our experiments, we have modeled $\log P(X \mid z^l)$ as a Bernoulli distribution, see Appendix A.6 for details.

Up to now we have discussed about the function we want to optimize given a single data point X used as input, however in practical applications we do not use a single data point at each training iteration. Rather, we take a batch of N data points $X_1, \ldots, X_N$ randomly sampled from the full dataset D, and calculate the total loss. In addition, the batches we use consist of more than 100 data points. Such a quantity of points allows, according to [24], to take only one sample ϵ_i for each X_i. As a consequence, the total loss for each training iteration of the (standard) variational autoencoder is

$$\text{Loss}\left(\phi, \theta; \{X_i\}_{i=1}^{N}\right) = -\sum_{i=1}^{N} \tilde{\mathcal{L}}(\phi, \theta; X_i, 1), \tag{B.12}$$

where the -1 factor comes from the fact that the gradient descent process aims to minimize the loss, and our goal is to maximize each $\tilde{\mathcal{L}}$. A summary of the process computed during each training iteration is illustrated in the following diagram:

$$X_i \overset{\text{Encoder}(X_i;\phi)}{\mapsto} \mu(X_i), \Sigma(X_i) \overset{\&\epsilon_i}{\mapsto} z_i \overset{\text{Decoder}(z_i;\theta)}{\mapsto} f(z_i; \theta) \mapsto \tilde{\mathcal{L}}(\phi, \theta; X_i, 1) \mapsto \text{Loss}. \tag{B.13}$$

The final loss $\tilde{\mathcal{L}}$ employs $\mu(X_i)$, $\Sigma(X_i)$ for calculating the first term of (B.11), and $f(z_i; \theta)$ for the second term of (B.11). A summary of the loss calculation for a standard VAE is given in Algorithm 1. For convenience, we refer to the total loss of the standard VAE as L_0.

B.3 Neural networks

Due to the page limit, we have not defined in detail the working principle of the transformations computed in a variational autoencoder. We thus describe here in detail the computation of the three main components of a VAE: the fully-connected layer, the convolutional layer, and the transposed convolutional layer. We also conclude with a remark about the full transformation computed by the VAE, from input images to topological loss, showing its differentiability under mild computational assumptions.

The simplest neural network, the *fully-connected layer*, is a map of the input $\mathbf{x} = (x_1, \ldots, x_n)$ into an output $\mathbf{y}$, of the form [31]:

$$\mathbf{x} \mapsto \mathbf{W} \cdot \mathbf{x} + \mathbf{b} = \mathbf{y}, \tag{B.14}$$

where $\mathbf{W}$ is a $m \times n$ matrix of weights, $\mathbf{b}$ is a bias vector, and m is the dimension of the output. In other words, we have:

$$y_i = \sum_{j=1}^{n} W_{ij} y_j + b_i, \text{ for } i = 1, \ldots, m. \tag{B.15}$$

Using weight matrices and biases, one can then create more complex transformations. In particular, we use the *convolutional layer* (corresponding to the conv2d function in Pytorch). This function maps an input $\mathbf{X}$ of shape (C_{in}, H, W) to an output $\mathbf{Y}$ of shape $(C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$, where C is the number of channels, H the height of the input planar image in pixels, and W is its width in pixels. The transformation is given by:

$$\mathbf{Y}(C_{\text{out}_j}) = \mathbf{b}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \mathbf{W}(C_{\text{out}_j}, k) \star \mathbf{X}(k), \tag{B.16}$$

where each $\mathbf{b}(C_{\text{out}_j})$ is a bias matrix, each $\mathbf{W}(C_{\text{out}_j}, k)$ is a weight matrix, k represents the output channel, and $\star$ is the cross-correlation operator. This operation is given by:

$$(\mathbf{W}(C_{\text{out}_j}, k) \star \mathbf{X}(k))_{ij} = \sum_m \sum_n W_{i+m,j+n} \cdot X_{m,n}, \tag{B.17}$$

where $W_{a,b}$ are the elements of $\mathbf{W}(C_{\text{out}_j}, k)$ and $X_{m,n}$ are the elements of $\mathbf{X}(k)$. The exact implementation details of the convolutional layer can be found in Conv2d.

The *transposed convolutional layer* (the ConvTranspose2d function in the code) transforms the input of shape $(C_{\text{in}}, H_{\text{in}}, W_{\text{in}})$ into an output of shape $(C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$. More details about this transformation can be found in ConvTranspose2d.

In general, a VAE relies on the sequential application of the three aforementioned transformations. In addition, between some of these transformations, we also employ activation functions such as the ReLU activation function (given by $\text{ReLU}(x) = \max\{0, x\}$), the sigmoid function, and so on. In our case, we use the ReLU function. It is important to note that the fully-connected, convolutional and transposed layers are all differential operations with respect to the parameters of the weight matrices and the bias matrices, and non-differentiability issues can only arise due to the ReLU function. However, its non-differentiability, corresponding to a zero-valued input, can be assumed to be a computationally unlikely situation.

We can connect this neural network structure with the topological loss: recall that TopoVAEs use the output of the VAE, producing a persistence diagram from it and computing the loss via a topological regularizer from Definition 3.1. We know that for most of the regularizers, the maps from point cloud to loss are generically differentiable in the space of point clouds. Combining this with the fact that the map from input to output (i.e., point cloud used by the loss) is computationally *almost always* also differentiable, we deduce, through the chain rule, that the full loss of a TopoVAE is a differentiable map under computationally mild conditions. This ensures that gradient descent can be performed, allowing the training process of TopoVAEs.

B.4 Information about image quality and diversity

As explained in the main text, the idea behind topological regularizers is to provide the generative model with information about the structure, shape or distribution of the batch of images in the high-dimensional space they lie in. This may enhance a faster way of optimizing the weights of the model such that the generated images have a similar distribution to the real data, and a higher quality and diversity. In particular, in a variational autoencoder, these additional loss terms penalize the difference between the persistence diagrams of the true and the generated data, leading to an update of the weights towards an updated model that produces data with a distribution with similar "shape", i.e., with a similar persistence diagram, to the true data —and with "similar", we mean with a low value of the topological regularizer used, which which can be bottleneck distance, squared difference between persistent entropies, and so on.

In the previous experiments, we have seen that topology-informed VAEs perform seemingly better than standard VAEs in terms of image quality and decay of the BCE loss. In addition, in some cases we observe a higher diversity of shapes in early training. In this section, we delve into the meaning behind the values outputted by topological regularizers. In particular, we explore the relation between the quality and the diversity of the images and the magnitude of the values returned by the topological regularizers. To do so, we perform two analyses, showing: 1) the magnitude of topological regularizers seems to correlate with image quality, and 2) the magnitude of topological regularizers seems to correlate with image diversity. Note that we present the following results as insightful behaviours of the regularizers, and not as formal proofs of any sort. We perform the tests with the bottleneck regularizers of degrees 0 and 1, and the δ-selective persistent entropy regularizers of degrees 0 and 1.

B.4.1 Analysis 1: image quality

We select 50 random batches, of 128 images each, from the FashionMNIST dataset. For each of these batches, we generate 3 new batches where we add different levels of Gaussian noise to the images. To do so, given a real batch B_i, we generate three new batches $B_i'(v_j)$, where v_j is the variance of the Gaussian noise added to the images of the original batch B_i. We choose $v_1 = 0.1$, $v_2 = 0.3$ and $v_3 = 0.7$ since it leads to three clearly different levels of noise. Then, we compute the dissimilarity between the persistence diagrams of B_i and $B_i'(v_j)$, via a regularizer $\tilde{d}$. In particular, given a regularizer $\tilde{d}$, we compute $\tilde{d}(B_i, B_i'(v_1))$, $\tilde{d}(B_i, B_i'(v_2))$, and $\tilde{d}(B_i, B_i'(v_3))$. In other words, we track how the dissimilarity between

persistent diagrams evolves as the noise in the batch of images increases. A good regularizer will thus present an increase of the magnitude of $\tilde{d}$ as the noise increases, and this is in fact what we observe in the experiments, which we show in Figure B.1. The dissimilarity metrics we use are the bottleneck distance of degree 0 (i.e., the bottleneck regularizer of degree 0), the bottleneck distance of degree 1 (i.e., the bottleneck regularizer of degree 1), and the values of the δ-selective persistent entropy regularizers of degree 0 and 1, with $\delta = 0.1$. (We do not use the other regularizers here for 1) avoiding a large amount of repetitive experiments, and 2) our main goal in this section is to see if dissimilarities in persistence diagrams can be related to image quality and noise levels, and not to check if each specific regularizer is a good measure of the level of noise.) The results can be seen on the top left and middle of Figure B.1: the solid lines correspond to the values of the regularizer for diagrams of degree 0, while dashed lines correspond to degree 1. Blue lines correspond to low noise ("noise level 1"), green lines to medium noise ("noise level 2"), and red lines to high noise ("noise level 3").

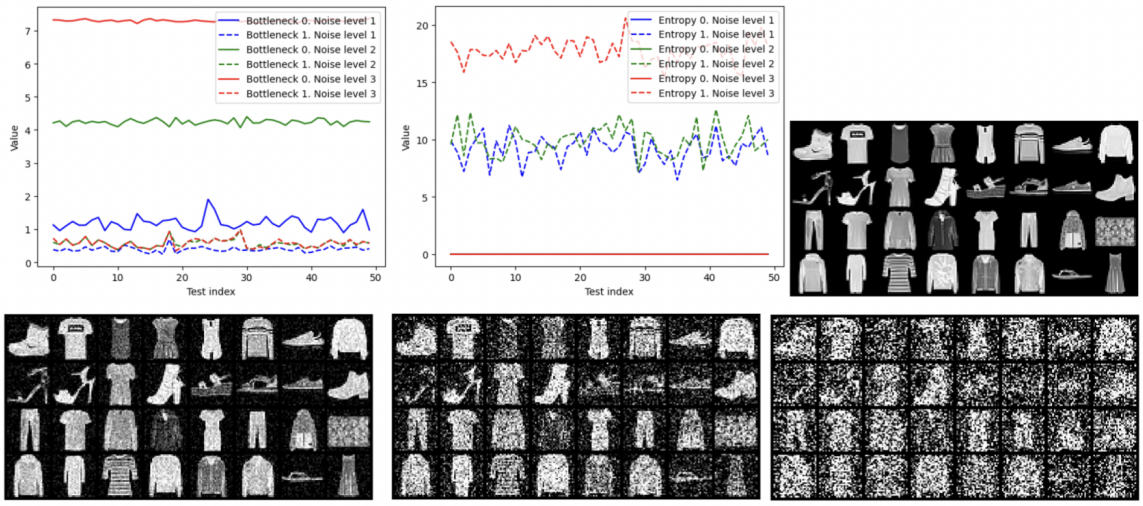

Figure B.1: Analysis 1: Correlation between regularizer magnitude and image quality. Top left: value of the bottleneck distance of persistence diagrams of degrees 0 and 1 for 50 random batches of the FashionMNIST dataset, corresponding to the bottleneck regularizers. Top middle: values of the δ-selective entropy regularizers of degres 0 and 1; $\delta = 0.1$. Top right: example of 32 images from an original batch of the dataset. Bottom: addition of Gaussian noise to the original batch, with increasing variance of the Gaussian distribution (0.1 (left), 0.3 (middle), and 0.7 (right)).

We confirm in Figure B.1 that the value of these regularizers increases with the level of noise in the batches. This correlation is clear for the bottleneck regularizer of degree 0 and the entropy regularizer of degree 1. The correlation still seems to be present in the bottleneck regularizer of degree 1 (its mean value is 0.40 for low noise, 0.54 for medium noise, and 0.59 for high noise), and the entropy regularizer of degree 0 (mean value of 0.00009 for low noise, 0.00050 for medium noise, and 0.00060 for high noise). However, for the latter two, the correlation seems to be less pronouced.

In summary, there is a clear correlation between noise level and the value of the bottleneck regularizer of degree 0 and the value of the δ-selective persistent entropy regularizer of degree 1. More analysis should be done on the bottleneck regularizer of degree 1 and persistent entropy regularizer of degree 0, but, at least, they do not seem uncorrelated to the noise level.

To explore if this behaviour is also present in colored datasets —which is a key aspect of topological regularizers for expanding their use to more real-life applications—, we repeat the same experiment in the CIFAR10 dataset, see Figure B.2. We observe again a correlation between regularizer magnitude and the amount of noise added, for both the bottleneck regularizer of degree 0 and the persistent entropy regularizer of degree 1. Furthermore, we also see, analogously to the FashionMNIST dataset, that while the entropy regularizer of degree 0 takes values orders of magnitude smaller, it also showcases an increase of its value with noise level (0.0001 for low noise, 0.0003 for medium noise, and 0.0003 for high noise). However, the values of the bottleneck regularizer of degree 1 do not seem to correlate with noise level.

Hence, in all experiments we have seen that the δ-selective persistent regularizer of degree 1 and the bottleneck regularizer of degree 0 seem to capture the concept of noise level and image quality. We can thus say that minimizing the value of these regularizers could be correlated with removing noise and enhancing the generation of more realistic images. More research should be done with the other regularizers, which we leave as future work. However, we have validated the main question we wanted to answer: dissimilarities in persistence diagrams can be related to image quality and noise level.

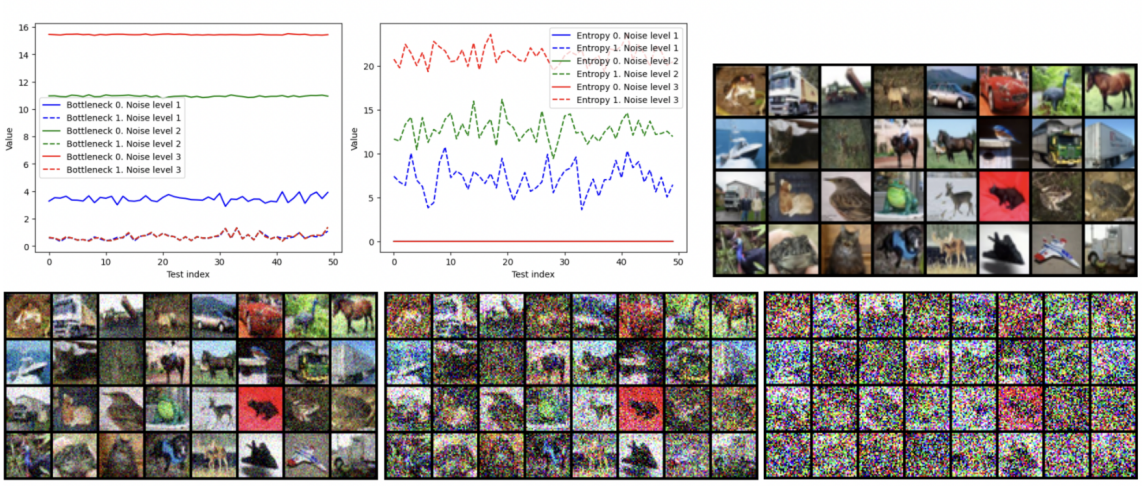

Figure B.2: Analysis 1: Correlation between regularizer magnitude and image quality. The experiments are analogous to Figure B.1, but they are performed in the CIFAR10 dataset instead of the FashionMNIST dataset.

B.4.2 Analysis 2: image diversity

To see if the difference between persistence diagrams is correlated with image diversity, we perform the following test in the FashionMNIST dataset. We select a class, such as class 0 of the dataset (corresponding to images of T-shirts/tops), and put all the images of this class in batches of 128 images, leaving us with 47 batches of T-shirts/tops. For each of these, denoted B_i, we generate three new batches $B_i'(n_j)$, where n_j is the number of new images from random classes inserted. We select $n_1 = 25$, $n_2 = 65$, and $n_3 = 105$. In other words, we generate three new batches with increasing amounts of image diversity. Then, for each true batch B_i, we select another random batch B_k T-shirts/tops, and compute the dissimilarities, via some regularizer $\tilde{d}$, of the persistence diagram of B_i and the persistence diagrams of B_k and $B_k'(n_j)$. In other words, we compute $\tilde{d}(B_i, B_k)$, $\tilde{d}(B_i, B_k'(n_1))$, $\tilde{d}(B_i, B_k'(n_2))$, and $\tilde{d}(B_i, B_k'(n_3))$. Hence, we track how the value $\tilde{d}$ evolves as the diversity in the batch increases.

We perform this test with two classes: first, with class 0 (T-shirts/tops); the result is given in Figure B.3, and second, with class 7 (sneakers); the result is given in Figure B.4.

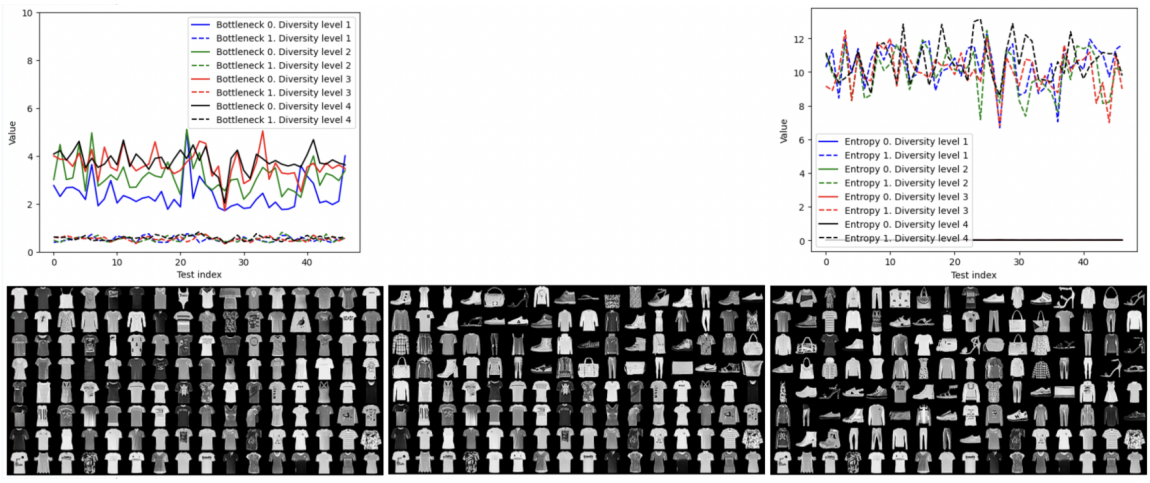

Figure B.3: Analysis 2: Correlation between regularizer magnitude and image diversity. Using class 0 (T-shirts/tops) of the FashionMNIST dataset. Top: values of the regularizers for the 47 different batches. Bottom: example of three levels of diversity (left: no noise, middle: 65 random images inserted, right: 105 random images inserted).

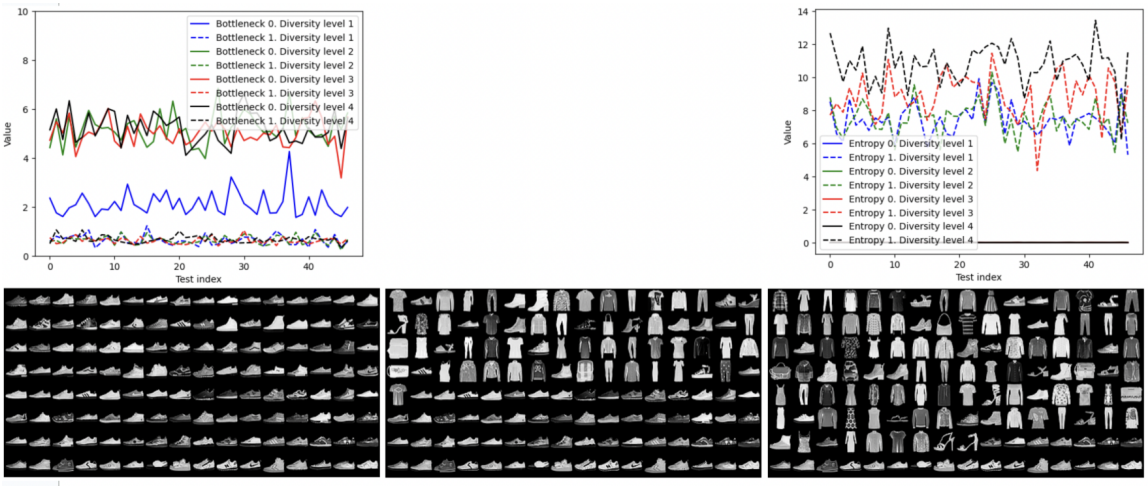

Figure B.4: Analysis 2: Correlation between regularizer magnitude and image diversity. Using class 7 (sneakers) of the FashionMNIST dataset.

In the first case, there seems to be a correlation between bottleneck distance of degree 0 and diversity. In fact, its mean value over the 47 tests for increasing diversity levels 1, 2, 3, and 4 is 2.39, 3.15, 3.61, and 3.81, repectively, increasing with diversity level. This is probably due to the fact that more diversity is related to more difference between pixel values and thus pairwise distances between points (images) of the point cloud (batch) that are perturbed, yielding persistence diagrams different from the original one, with a difference that increases with the diversity of shapes in the images. On the other hand, there does not seem to be a correlation for the other three regularizers. However, in the case of sneakers, i.e., in Figure B.4, the δ-selective persistent entropy generator of degree 1 seems to be related to image quality; its mean value for each diversity level is 7.41, 7.53, 8.87, and 10.79, increasing with diversity level. This, again, is possibly due to specific pixel intensity variations caused by the insertion of random images from other classes in the batch. Furthermore, we see again some degree of correlation between the value of the bottleneck regularizer of degree 0 and image diversity (mean values 2.13, 5.19, 5.03, and 5.21 for each diversity level).

The code used for conducting the quality tests is in `tests_fashionmnist.ipynb` for the FashionMNIST dataset, and in `tests_cifar.ipynb` for the CIFAR10 dataset. For the diversity tests, the code is in `diversity_fashionmnist.ipynb`. In summary, the results obtained in this section indicate that topological regularizers encode some information about the data, related to the quality, the noise, and the shape and diversity of the images, that can lead to enhanced training processes in generative models.