Torsion in Persistent Homology and Neural Networks

Maria Walch

Abstract

1 Introduction

Over the past years, topological data analysis (TDA) has gained significance in various fields, ranging from the analysis of images relevant to cancer research [22] to theoretical particle physics [28]. In line with this development, there are several high-performance, easy-to-use, and open-source TDA libraries available (e.g.: [4, 26, 27]). Fitting with the dominant role of (deep) neural networks for data analysis, algorithms and ideas from topological data analysis are also being used for machine learning, in particular deep learning. In the context of hybrid methods, it is essential that assumptions are made both from the side of machine learning and from the side of topological data analysis. Since the data sets that are processed at the intersection of topological data analysis and machine learning exhibit usually high dimensionality and high cardinality of data points, some of these assumptions have a computational background.
With regard to the interpretability and reliability of the results achieved using these (hybrid) methods, it is essential to be aware not only of assumptions but also of the limitations and uncertainties associated with them.
In this work, we address the fact that field coefficient persistent homology is typically used by default for computations. We investigate whether, and if so, what effects this has on deep learning applications that involve the predominant invariants from field coefficient persistent homology – the persistence diagrams. Since some hybrid applications in deep learning utilize vectorized persistence diagrams, it is generally assumed that the choice of coefficient field does not significantly influence subsequent computations or affect the reconstruction of torsion features within the data. Since the classical matching distance between persistence diagrams cannot be applied to reveal information about torsion [10], it is also inherently assumed that torsion does not matter in the data and its processing steps in machine learning algorithms.
From the perspective of neural networks, we examine the following assumption.

Assumption 1.1. *The output of each layer in a deep neural network is treated as an element of a Euclidean space* $\mathbb{R}^d$.

2 Background: Persistent Homology

We define persistent homology with regard to its application to machine learning settings. In this context, we understand a data set $\mathcal{D} = \{x_i \in \mathbb{R}^d \mid 1 \leq i \leq n,\ n \in \mathbb{N}\}$ as a finite (Euclidian) metric space, also called a *point cloud*. On this data set we construct a simplicial complex - a topological space constructed by gluing together dimensional simplices (points, line segments, triangles, tetrahedrons, etc.). The underlying combinatorial structure is called an *abstract* simplicial complex.
(Abstract) simplicial complexes form the objects of a category **Simp** whose morphisms are given by simplicial maps. A simplicial map is a function between two simplicial complexes that maps vertices of one complex to vertices of another while preserving the structure of simplices. A filtration of a simplicial complex K is a sequence $\{K_i\}_{i\in I}$ of subcomplexes, indexed by a totally ordered set I, such that for any $i, j \in I$ with $i \leq j$, we have $K_i \subseteq K_j$. Considering the totally ordered set I as a category $\mathcal{I}$, where the objects are the elementss of the set and a unique morphism exists from an object i to an object j if and only if $i \leq j$, a filtered simplicial complex can be described as a functor

$$\mathcal{F} : \mathcal{I} \to \mathbf{Simp}.$$

Simplicial homology defines a functor

$$H_*(-; R) : \mathbf{Simp} \to \mathrm{Mod}_{\mathbb{Z}}^R, \tag{1}$$

where $\mathrm{Mod}_{\mathbb{Z}}^R$ represents the category of $\mathbb{Z}$-graded R-modules, with R being a commutative ring (referred to as the *coefficient ring*), and $*$ denotes the homological degree.
The p-dimensional persistent homology of simplicial complex K constructed on a finite metric space $\mathcal{D}$ is defined as the composition of the filtration functor and the simplicial homology functor

$$\begin{aligned} PH_p(K) &:= H_p \circ \mathcal{F}, \\ PH_p(K) &: \mathcal{I} \to \mathrm{Mod}_{\mathbb{Z}}^R. \end{aligned} \tag{2}$$

The functor in Eq. (2) is a *finite-type* example of a *persistence module*. Note that originally [31], the term 'persistence module' referred to functors from the indexing category of natural numbers to the target category **Vect**, which denotes the category of vector spaces over a field. Our definition stems from the more general definition of *generalized persistence modules* provided in [6].
Persistence modules form the objects of a functor category whose morphisms are given by natural transformations between functors. Choosing a field as the coefficient ring in Eq. (2), the isomorphism type of the corresponding persistence modules exhibits a particular nice description as a multiset of intervals in I. This description is the famous **barcode** or **persistence diagram** $D_p(K)$. It comprises a list of intervals of the form $[b, d) = [a_i, a_j)$ or $[a_i, +\infty)$, where $a_i, a_j \in I$. Each such interval represents a topological 'feature' that is 'born' at b and 'dies' at d.

2.1 Structure Theorem and Torsion

We consider a finite-dimensional filtered simplicial complex and thus wlog a filtration over $\mathbb{N}$. Explicitly, Eq. (2) states that the p-th persistent homology of a finite-dimensional filtered simplicial complex

$$K : \emptyset = K_0 \subset K_1 \subset ... \subset K_N \tag{3}$$

with coefficients in a commutative ring R is given by

$$PH_p(K; R) : H_p(K_0; R) \to H_p(K_1; R) \to ... \to H_p(K_N; R), \tag{4}$$

where $H_p(K_i, R)$ denotes the simplicial homology group for a subcomplex K_i, $0 \leq i \leq N$ in the filtration.

2.1.1 Integral and Field Coefficients

For $R = \mathbb{Z}$, simplicial homology groups $H_p(K, \mathbb{Z})$ admit a primary decomposition according to the fundamental theorem of finitely generated Abelian groups (Theorem 11.1 in [11]):

$$H_p(K, \mathbb{Z}) \cong \mathbb{Z}^{\beta_p(\mathbb{Z})} \bigoplus_{q \text{ prime}} \left(\mathbb{Z}/q^{k_1}\mathbb{Z} \oplus \ldots \oplus \mathbb{Z}/q^{k_{t(d,q)}}\mathbb{Z} \right), \tag{5}$$

with integers $t(p, q) \geq 0$ and $k_i > 0$ for every prime number q [5]. For $t(p, q) > 0$, the integers $q^{k_1}, \ldots, q^{k_{t(p,q)}}$ are called **torsion coefficients**, with q as the unique prime divisor. The uniquely defined integer $\beta_p(K, \mathbb{Z})$ is called the pth **integral Betti number**. When a field $\mathbb{F}$ is chosen as coefficient ring, the simplicial homology groups are vector spaces, fully determined by their ranks $\beta_p(\mathbb{F})$ called the pth **field Betti numbers**.

2.1.2 Structure Theorem

In [31] Theorem 3.1, an equivalence of categories between the category of persistence modules of finite type over R and the category of finitely generated non-negatively graded modules over the polynomial ring $R[t]$ is proven. From this categorical equivalence, it becomes clear why there is no simple description for persistence modules over the integers $\mathbb{Z}$: It is known from commutative algebra that the classification of modules over $\mathbb{Z}(t)$ is extremely complicated. However, taking a field $\mathbb{F}$ as the coefficient ring, $\mathbb{F}(t)$ is a PID and the structure of the corresponding persistence module can be described by standard structure theorem (Theorem 2.1 in [31]):

$$\left(\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} \mathbb{F}[t] \right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{\gamma_j} \mathbb{F}[t]/(t^{n_j}) \right). \tag{6}$$

In Eq. (6), Σ denotes an upward shift in grading for $\alpha_i, \gamma_j \in \mathbb{Z}$. Now the isomorphism classes of $\mathbb{F}[t]$-modules are parametrized by the two types of intervals

mentioned above, namely $[b, d) = [a_i, a_j)$ or $[a_i, +\infty)$, where $a_i, a_j \in I$. Accordingly, we denote the collection of intervals for a given finite simplicial complex K and choice of coefficient field $\mathbb{F}$ as $D_p(K; \mathbb{F})$.

2.2 Dependence on the Coefficient Field

From Eq. (5) and the above discussion, it is apparent that homology groups with field coefficients are blind to the presence of torsion. Torsion phenomena in the integral (relative) homology groups, however, cause the corresponding computed field coefficient invariants $D_p(K; \mathbb{F})$ to exhibit a dependence on the choice of the coefficient field. In [20], Theorem 1.6, the following result is given to determine when this is the case:

Theorem 2.1. *A persistence diagram $D_p(K; \mathbb{F})$ is independent of the choice of the field, if the relative homology group $H_p(K_i, K_j; \mathbb{Z})$ is free for any $0 \leq i < j \leq N$ in the filtration Eq. (3), and $H_{p-1}(K_i; \mathbb{Z})$ is free for any $0 \leq i \leq N$.*

If the requirements of Theorem 2.1 are not fulfilled, then, depending on the torsional structure inherent to the data set, the intervals in the persistence diagram might differ for different coefficient fields. In particular, longer intervals could be split into shorter ones, which is critical with regard to the common assumption in TDA that short intervals correspond to insignificant features or noise [12] (see also Theorem 1.17 in [20]). For the following analysis in Section **??**, we particularly need the following Corollary to Theorem 2.1 (Corollary 1.8 in [20]).

Corollary 2.1. *Let K_X be a filtered simplicial complex whose vertices are the elements of a set $X \subset \mathbb{R}^M$. Then the persistence diagrams $D_p(K_X, \mathbb{F})$ for homology dimension $p = (M - 1)$ are independent of the choice of field $\mathbb{F}$.*

3 Occurrence of Torsion in Data

Algorithmic persistent homology was specifically developed to analyze high-dimensional, complex data that typically arises from the temporal evolution of physical variables [31]. This is also a type of data that is particularly interesting for machine learning applications. Regardless of the fact that high-dimensional data already presents a range of computational problems for both TDA algorithms and machine learning algorithms, it has also been empirically demonstrated in [20] that torsion phenomena occur more frequently in high dimensions. Usually, one associates torsion with features such as the non-orientability of a surface. Typical examples are the Klein bottle and the Möbius strip. Yet, the empirical study in [20] implies that torsion can also be expected to be found in real world data sets. Even in simple robotics applications, examples occur where torsion appears in the corresponding persistent homology groups [10]. This already raises two questions:

I. Can real-world datasets be analyzed for torsion?

II. Can neural networks be a method to solve the problems that arise in this context?

3.1 Algorithmic Test for Torsion

To the best of our knowledge, there currently exist three algorithmic methods to test a given data set for torsion, with only one of these methods directly outputting the existence of torsion as a binary result. Torsion manifests in the fact that persistence diagrams differ for various coefficient fields. Therefore, a blindfold approach to test for torsion is to compute persistence diagrams over different coefficient fields and compare the results, an approach taken by [19]. This is computationally costly.
Thus in [5], the Chinese Remainder Theorem (CRT) is used to compute persistence across multiple fields in a single pass. By doing so, it combines information from different coefficient fields, enabling a more efficient detection of torsion without separate computations for each field. However, the algorithm of [5] is computationally advantageous over the brute force approach only when there are few torsion effects and the torsion subgroups exhibit low order in their homology.
In contrast, [20] constructed an algorithm to check for torsion directly by analyzing the structure of the boundary matrix without computing persistence diagrams over multiple fields. This algorithm leverages algebraic properties to detect torsion and outputs a binary result, indicating the presence or absence of torsion in the homology. By avoiding the need for computations across different fields, the method in [20] has the lowest computational complexity among the available torsion detection algorithms. However, it still retains the cubic complexity of the persistent homology algorithm, so the limitations discussed in the subsequent Section 3.2 also apply to this approach.

3.2 Persistent Homology of Complex Data Sets

We define a *complex* data set as a collection $\mathcal{D} = \{x_i \in \mathbb{R}^d \mid 1 \leq i \leq n,\ n \in \mathbb{N}\}$ for which $n \gg 100$ and $d \gg 3$.
Question (I) is already challenging for this setting because, although in theory persistent homology can be computed in higher dimensions, computational limitations make it infeasible for datasets with more than a few thousand points in $\mathbb{R}^3$ [17]. Problems are caused by three factors: the cardinality of the dataset, the dimension of the Euclidian space in which it is embedded, and the resulting computational demand of the persistent homology algorithm itself.
Beginning with the *first aspect*, the size of certain simplicial complexes, including the two most common ones—the Rips complex and the Čech complex—grows exponentially with the cardinality of the dataset [21]. Furthermore, the maximum dimension of an abstract simplicial complex constructed on a data set of cardinality n is $n - 1$ [8]. This dimension can be much higher than the dimen-

sion of the ambient Euclidean space $\mathbb{R}^d$ of the points x_i, $i \in \{0, ..., n\}$ on which the simplicial complex is constructed. It is a topic of ongoing research how to prevent a simplicial complex from having too many simplices, especially in high dimensions [21].
With regard to the *second aspect*, the computation of the distance matrix required for persistent homology becomes more costly as the dimensionality of the data points increases. Additionally, there is the problem of the "curse of dimensionality," which leads to the requirement of exponentially more data points in higher dimensions to maintain the sampling density. This curse also affects the computation of topological invariants with persistent homology: When intrinsically low dimensional data is embedded in a high-dimensional ambient space persistent homology becomes very sensitive to noise and fails to detect the correct topology of the data [7, 1, 16].
With regard to the *third aspect*, the computational complexity of persistent homology for a fixed homology dimension p scales as $\mathcal{O}(n^3)$ in the worst case, where n denotes the number of simplices [21]. For a data set embedded in $\mathbb{R}^d$, it is *a priori* necessary to compute Betti numbers up to dimension $d-1$ to fully capture the topological features of the space. verweis auf kahle, [9, 13]
Summarizing the above three aspects with reference to Section 3.1, we therefore answer Question (I) in the negative: It is currently not possible to comprehensively (and algorithmically) analyze real-world data sets for torsion. This leads us to addressing Question II, for which we first need to establish the necessary foundations regarding neural networks.

4 Neural Networks

Neural networks represent a class of machine learning models composed of interconnected computational units, inspired by the architecture of biological neurons. These units, or neurons, are typically organized into layers. A network with more than three layers is called *deep*.
Let $\mathcal{D} = \{x_i \in \mathbb{R}^d \mid 1 \leq i \leq n,\ n \in \mathbb{N}\}$ be a data set serving as the input to the neural network. For a given layer f_j containing $M \in \mathbb{N}$ neurons, the output for an input $x_{i,l} \in \mathbb{R}^v$ *to the layer* is assumed to be a vector $f_j(x_{i,l}) \in \mathbb{R}^M$, defined as:

$$f_j(x_{i,l}) = [o_{ij1}, o_{ij2}, ..., o_{ijM}], \tag{7}$$

where each $o_{ijk} \in \mathbb{R}$ for $1 \leq k \leq M$ denotes the output of the k-th neuron in layer j for the input $x_{i,l}$. We denote the term in Eq. (7) as the *layer state* associated to the layer f_j.
We consider a neural network f as a composition of $N \in \mathbb{N}$ sequentially connected layers f_j, $1 \leq j \leq N$. This is also known as a *feedforward network*. Now viewing $x_{i,nn} \in \mathcal{D}$ as an input sample *to the entire network* , the output of the neural network is obtained as:

$$f(x_{i,nn}) = f_N \circ f_{N-1} \circ \cdots \circ f_2 \circ f_1(x_{i,nn}). \tag{8}$$

Here, the composition $f_N \circ f_{N-1} \circ \cdots \circ f_1$ represents the forward propagation of $x_{i,nn}$ through the network's layers, yielding the final output.

4.1 Neuronal Computations

The output o_{ijk} of neuron k in layer j for an input $x_i \in \mathbb{R}^v$ is computed as:

$$o_{ijk} = \sigma\left(W_{jk} \cdot x_i + b_{jk}\right), \tag{9}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ denotes a typically nonlinear *activation function*. The term $W_{jk} \in \mathbb{R}^v$ represents the *weight vector* associated with neuron k in layer j, and $x_i \in \mathbb{R}^v$ is the input to this layer. Finally, $b_{jk} \in \mathbb{R}$ denotes the *bias* term associated with neuron k in layer j.

4.2 Neural Network Training

The weights and biases are *parameters* of the neuronal network that are *optimized* during an iterative process called *training*. The metric difference (also called *loss*) between an output $f(x_{i,nn})$ associated with a specific input $x_{i,nn}$ to a neuronal network is measured with respect to a pre-selected metric distance (the *error metric* or *loss term*). This is selected in a way that a minimum loss would indicate the network to have learned a desirable characteristic of the input data.
The loss information is subsequently *propagated back* [24] in order to update the parameters (mainly the weights) of a neural network until a minimum loss is attained. The aim of this procedure is to adjust the parameters of the network so that the loss on the data it was trained with is as low as possible, while also ensuring that the loss on data known to the user of the neural network to be similar to the training data is also low. If this condition is met, one says that the network *generalizes well*.

4.3 Network Architectures

The activation function is an example of a *hyperparameter* that remains usually fixed over the training phase. The patterns of interconnections between the neurons in a network and the hyperparameters define the *architecture* of a neural network. In this sense, novel network architectures for a wide range of applications and data are developed on a daily basis. In the following Section, and with regard to Question (II), we present autoencoder architectures and topologically inspired modifications to their training process.
In [2] the definition of an architecture is subdivided into the *combinatorial*, the *weight* and the *activation* architecture. Herein, the combinatorial architecture captures only the interconnectivity pattern of the neurons themselves, i.e., the number of layers, the number of neurons per layer, and how they are connected to each other.

4.4 Autoencoders

Autoencoders [23] constitute a network architecture mainly used for dimensionality reduction [14]. Via the use of nonlinear activation functions, autoencoders represent a nonlinear dimensionality reduction method.
Autoencoders can be seen as a special case of feedforward networks. They distinguish by their combinatorial architecture which is designed in a way to enable the network to copy the input [14]. Due to (architectural) constraints, a perfect reconstruction of the input is inhibited. The aim is to learn a (lower dimensional) representation of the input that captures the defining features of the data.
A basic (or *vanilla*) autoencoder architecture comprises at least three layers: an input layer, a *latent* layer, and an output layer. The part of the neural network mapping the input to the latent state is called *encoder*, whereas the function mapping the latent state to the output state is called the *decoder*. In a deep autoencoder, additional *hidden* layers are introduced into the encoder and/or the decoder.
The defining characteristic of a basic autoencoder is that the latent layer contains significantly fewer neurons than the input and output layers. Figure 1 illustrates this structure. When the features being reconstructed correspond to the Euclidean dimensions of the input space, having fewer latent neurons effectively reduces the dimensionality of the input. From the topological perspective, two immediate questions arise with regard to the respective layer states of an autoencoder:

III. How do the persistence modules constructed on the input, the latent and the output state differ?

IV. Can an autoencoder, under suitable architectural constraints, be trained to induce quasi-isomorphisms between chain complexes derived from simplicial complexes on the layer states?

V. Is this a desirable property for learning meaningful representations?

Question (III) is solvable as far as persistent homology is computable, by calculating a (pseudo) metric distance between the associated persistence diagrams (in the 1-parameter case). With regard to Question (III), there are various algorithmic approaches, three of which we present in the subsequent Section 4.5 and further examine in connection with Question (II) concerning torsion.

4.5 Autoencoders with TDA Losses

As mentioned in Section 4.2, loss terms are designed such that a minimum loss ideally guarantees that certain characteristica of the data are learned. Regarding Question (III), it appears desirable for the loss term of an autoencoder to be designed to maximize the similarity between the corresponding persistence modules.

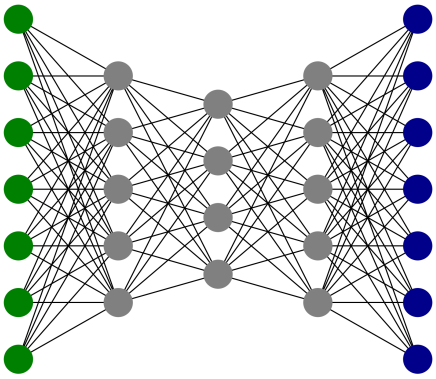


Figure 1: An example of a vanilla autoencoder with three hidden layers, whose neurons are marked in grey. The neurons in the input layer are green, the neurons in the output layer are dark blue.

With regard to Question (V) we restate Corollary 3.A.7. (b) from [15]. fehlt: universal coefficient theorem

Corollary 4.1. *A map $f : X \to Y$ induces isomorphisms on homology with $\mathbb{Z}$ coefficients if and only if it induces isomorphisms on homology with $\mathbb{Q}$ and $\mathbb{Z}_p$ coefficients for all prime numbers p.*

Therefore, from the perspective of reconstructing torsion effects, it is desirable that both the encoder and the decoder each induce quasi-isomorphisms — and this should hold for every coefficient field $\mathbb{Q}$ and $\mathbb{Z}_p$ for all primes p. ausführungen dass es aus machine learning perspective aber nicht wünschenswert ist, ggf in arbeit

4.5.1 Topological Autoencoder

In a filtered simplicial complex, via the functoriality in Eq. (4), the intervals in the barcode are each associated with a specific "creator simplex" and a "destroyer simplex" from the filtration. If the simplicial complex constructed on the dataset is built based on the metric distance between the dataset elements, there is accordingly a "creator edge" and a "destroyer edge" for each interval, where for simplices with a dimension > 1, this edge is selected from the simplex

according to a weighting scheme.
In [18], a loss term (in the following: "TopoAE loss term") is introduced with the aim to align these creator and destroyer edges between the input and the latent space. Imposing Assumption 1.1, whereby a simplicial complex is constructed on both the input data and the latent space according to the respective inherited Euclidean metric of each embedding space, the TopoAE loss term aligns the corresponding creator and destroyer simplices bidirectionally. Namely,

$$\mathcal{L}_{\text{TopoAE}} := \mathcal{L}_{\mathcal{X}\to\mathcal{Z}} + \mathcal{L}_{\mathcal{Z}\to\mathcal{X}},$$

where $\mathcal{X}$ denotes the input and $\mathcal{Z}$ denotes the latent space, with:

$$\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} := \frac{1}{2}\left\|A^X[\pi^X] - A^Z[\pi^X]\right\|^2, \tag{10}$$

$$\mathcal{L}_{\mathcal{Z}\to\mathcal{X}} := \frac{1}{2}\left\|A^Z[\pi^Z] - A^X[\pi^Z]\right\|^2. \tag{11}$$

In Eq. (10), $A^X[\pi^X]$ is a vector containing the creator and destroyer edges of the filtration on the input space and $A^Z[\pi^X]$ is their image under the encoder. Accordingly, in Eq. (11), $A^Z[\pi^Z]$ denotes the vector of creator and destroyer edges of the filtration constructed on the latent data and $A^X[\pi^Z]$ denotes their preimages under the encoder. ausführungen, dass topo ae keinen quasi isomorphismus induziert, siehe Arbeit

4.5.2 RTD Autoencoder

Representation Topology Divergence ("RTD"), as defined in [3] and adapted in [29] to define a loss term, is intended to measure the extent to which the chain map induced by a bijective mapping between two point clouds deviates from a quasi-isomorphism [3]. Since the homotopy category of chain complexes associated with a simplicial complex is triangulated, a chain map is a quasi-isomorphism if its cone is acyclic [ref?].
For each of the two point clouds $\mathcal{P}$ and $\tilde{\mathcal{P}}$, we construct a Vietoris-Rips complex represented as a weighted graph, where the weights correspond to the Euclidean distances between points. Specifically, let $G^{w\leq\alpha}$ and $G^{\tilde{w}\leq\alpha}$ denote the weighted graphs for $\mathcal{P}$ and $\tilde{\mathcal{P}}$, respectively, and let $R_\alpha(G^w)$ and $R_\alpha(G^{\tilde{w}})$ represent their associated Vietoris-Rips complexes. Let f be a bijective map between the points in $\mathcal{P}$ and $\tilde{\mathcal{P}}$.
Define $G^{\min(w,\tilde{w})\leq\alpha}$ as the weighted graph formed by the union of the edges in $G^{w\leq\alpha}$ and $G^{\tilde{w}\leq\alpha}$. Specifically, $G^{\min(w,\tilde{w})\leq\alpha}$ includes an edge between two points $p, p' \in \mathcal{P}$ if either the distance between p and p' or the distance between their images under f, $f(p)$ and $f(p')$, is $\leq \alpha$.
Let

$$i : R_\alpha(G^w) \hookrightarrow R_\alpha(G^{\min(w,\tilde{w})}) \tag{12}$$

denote the inclusion map from the Vietoris-Rips complex associated with G^w to the Vietoris-Rips complex associated with $G^{\min(w,\tilde{w})}$. In [3], an auxiliary complex $R_\alpha(\tilde{G}^{(w,\tilde{w})})$ is constructed, which is homotopy equivalent to the cone

of i. The analogous construction is carried out for the inclusion $\tilde{i} : R_\alpha(G^{\tilde{w}}) \hookrightarrow R_\alpha(G^{\min(w,\tilde{w})})$, yielding an auxiliary complex $R_\alpha(\tilde{G}^{(\tilde{w},w)})$ homotopy equivalent to the cone of $\tilde{i}$.
The representation topology divergence $\mathrm{RTD}_p(P, \tilde{P})$ (RTD) in homology dimension $p \geq 1$ associated with $R_\alpha(\tilde{G}^{(w,\tilde{w})})$ is defined as the sum of the interval's length in the p-dimensional barcode of $R_\alpha(\tilde{G}^{(w,\tilde{w})})$ and likewise for $R_\alpha(\tilde{G}^{(\tilde{w},w)})$. For the inclusions i and $\tilde{i}$ to induce quasi-isomorphisms, the homology of $R_\alpha(\tilde{G}^{(w,\tilde{w})})$ and likewise for $R_\alpha(\tilde{G}^{(\tilde{w},w)})$ should be trivial and thus the RTD should be 0.
Based on this, [29] define a loss term ("RTD loss") as follows:

$$\sum_{i \geq 1} \left(\mathrm{RTD}_p(P, \tilde{P}) + \mathrm{RTD}_p(\tilde{P}, P) \right), \tag{13}$$

for a homology dimension $p \geq 1$. when this loss term is used for the optimisation of the trianing of an autoencoder architecture, the two point clouds under consideration P and $\tilde{P}$ would be identified with the input and output data set respectively.
In practice, [29] used

$$\mathrm{RTD}_1(P, \tilde{P}) + \mathrm{RTD}_1(\tilde{P}, P). \tag{14}$$

4.5.3 DIPOLE

The DIPOLE functional, as introduced in [30], is a loss functional specifically designed to enhance the capacity of a given surjective embedding function

$$f : X \to Y, \tag{15}$$

where (X, d_X) and (Y, d_Y) are finite metric spaces, to preserve both local and global properties of the metric space to be embedded. DIPOLE is based on distributive persistent homology and, unlike the global application of persistent homology, is therefore also computable on complex data sets.

Definition 4.1 (Distributed Invariant, after [25])**.** *Consider an invariant λ for a finite metric space (X, d_X). For an integer $k \in \mathbb{Z}$, let the associated distributed invariant Λ_k map X to $\{(S, \lambda(S)) \mid S \subset X, |S| = k\}$ when $k > 0$ and to $\emptyset$ for $k \leq 0$. Essentially, $\Lambda_k(X)$ captures the values of λ on subsets of X with precisely k elements.*

Let the invariant under consideration be λ^m, the persistent homology
(Eq. (2)) computed on the m-skeleton of a finite metric space (X, d_X) based on the Vietoris-Rips complex. For short, we call λ^m the m-th 'Rips-persistence' of (X, d_X). The main result in [30] states conditions under which the embedding function f from Eq. (15) is a quasi-isometry and gives an estimate for its metric distortion.

Theorem 4.2 (Theorem 1 in [30])**.** *Let (X, d_X) be a finite metric space and λ^m be the Rips-persistence of its m-skeleton as described above. Let $k > m > 0, k \in$*

$\mathbb{Z}$. *Let $f : X \to Y$ be a surjection as in Eq.* (15)*, ensuring that for all subsets $S \subseteq X$ with cardinality $|S| \in \{k, k-1, ..., k-m-1\}$, the bottleneck distance (where the maximum over all homology dimensions is taken) satisfies:*

$$d_B(\lambda^m(S), \lambda^m(f(S))) \leq \epsilon. \tag{16}$$

Then f is a quasi-isometry and $\forall x_1, x_2 \in X$, the metric distortion of f can be estimated as:

$$|d_X(x_1, x_2) - d_Y(f(x_1), f(x_2))| \leq 112k^2\epsilon. \tag{17}$$

In [25], it is demonstrated that for Theorem 4.2 to be applicable, it is not necessary to consider the collection Γ of all subsets of X with cardinalities ranging from k to $k-m-1$. It is sufficient to have a collection of subsets that satisfies the covering and closure properties in Theorem 20 of [25] (see Section 5.5 and Appendix D in [25]).
In order to guarantee that an embedding function as given in Eq. (**??**) preserves the essential metric and topological properties of (X, d_X), the method presented in [30] constructs a gradient-differentiable loss term L_{dip}, which is defined as the sum of a local 'metric' term LMR_p and a global 'topological' term DP_k^m.

$$L_{\text{dip}} := \alpha \cdot \text{LMR}_p + \text{DP}_k^m, \tag{18}$$

where $\alpha \in \mathbb{R}$ is a hyperparameter controlling the strength of the contribution of the local term LMR_p.

Definition 4.3 (Local Term)**.** *Let (X, d_X) be a finite metric space and let $P \subseteq X \times X$ be a subset of pairs of points in X selected to be close to one another, for example via picking a radius parameter $\delta > 0$, such that:*

$$P = \{(x_i, x_j) \ \in X \times X \mid d_X(x_i, x_j) \leq \delta\}. \tag{19}$$

The local term in Eq. (18) *is defined as follows:*

$$\text{LMR}_p(f) = \sum_{(x_1,x_2)\in P} \left(d_X(x_1, x_2) - \|f(x_1) - f(x_2)\|\right)^2, \tag{20}$$

where f denotes an embedding function as given in Eq. (**??**)*.*

The global part of the loss term uses the above-given definition of a distributed invariant (Definition 4.1).

Definition 4.4 (Global Term)**.** *Let λ denote the Rips persistence of the finite metric space (X, d_X). Let λ^m be the invariant that associates to the elements of X the Rips persistence of its m-skeleton. Let $k \in \mathbb{Z}$ with $k > m > 0$. Then the global topological term of the loss term in Eq.* (18) *is defined as follows:*

$$\text{DP}_k^m(f) = \frac{1}{\binom{|X|}{k}} \sum_{\substack{S\subseteq X \\ |S|=k}} W_p^p(\lambda_m(S), \lambda_m(f(S))), \tag{21}$$

where $|X|$ respectively $|S|$ denotes the cardinality of elements in X and S respectively. W_p^p denotes the p-th power of the p-th Wasserstein distance between $\lambda^m(S)$ and $\lambda^m(f(S))$, which is computed for all possible homology dimensions and summed up.

4.5.4 DIPOLE Autoencoder

The encoder f_{enc} of a trained non-variational autoencoder represents a surjective function that can be considered as an embedding function under Assumption 1.1. Thus, one possible way to integrate the DIPOLE loss functional into an autoencoder setting is to apply it as a post-processing step to the trained encoder. We have implemented this approach and used it in the experimental section.
Another approach is to use DIPOLE directly as a loss term for training an autoencoder. possible with almost sure convergence? Clarke subdifferential? Fehelerabschätzung cohen steiner + arbeit

5 Neural Networks and Torsion

Definition 5.1 (Basis Architecture)**.** *In the following, the 'basis architecture' refers to a trained autoencoder with n layers for the encoder and n layers for the decoder part, whose activation architecture is purely RELU. It has a nonzero mean squared reconstruction error denoted by δ.*

Proposition 5.1 (Bijectivity of the Basis Architecture)**.** *If the basis architecture $\mathcal{A} := f_{\text{dec}} \circ f_{\text{enc}}$ is assumed to be bijective, then the encoder f_{enc} as well as the decoder f_{dec} is bijective.*

Proof. For $\mathcal{A}$ to be bijective, the encoder needs to be at least injective and the decoder needs to be surjective on $f_{\text{enc}}(\mathcal{D})$ for an input data set $\mathcal{D}$. By definition, the encoder is deterministic and thus also surjective. Hence for $\mathcal{A}$ to be bijective, the encoder constitutes a bijection. Thus also the deecoder is bijective. □

Remark 5.1 (Generalization Error)**.** *A generalization error of δ implies that there exists at least one point in the input space $x \in \mathcal{D} \subset \mathbb{R}^d$, such that the reconstruction error with respect to the inherited Euclidian metric satisfies*

$$\|x - \mathcal{A}(x)\| = \sqrt{n\delta}, \tag{22}$$

where $\mathcal{A}$ denotes the map induced by the autoencoder and $n = |\mathcal{D}|$ is the cardinality of the input space.

By Assumption 1.1, we define a filtered simplicial complex on each layer space, constructed using the Vietoris–Rips construction with a filtration over $\mathbb{R}$. A filtered complex is an instance of a persistence complex (Definition 3.1 in [31]). In the following, the input, latent and output space are of particular relevance.

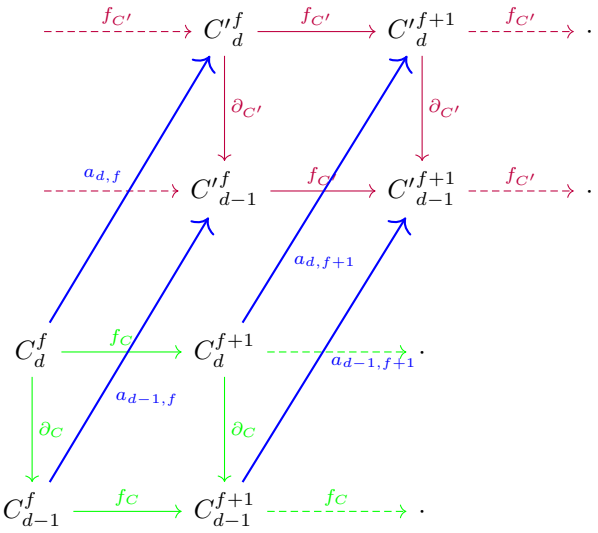


Figure 2: The chain complexes on the input space at dimension d and filtration level f are denoted by C^f_d, and those on the layer space by C'^f_d. The chain maps are represented as f_C and $f_{C'}$, while the boundary operators are ∂_C and $\partial_{C'}$. The persistence complex on the input space is indicated in green, and its counterpart on the layer space in purple. The mappings $a_{d,f}$ induced by a trained autoencoder between C^f_d and C'^f_d are shown in blue.

Proposition 5.2 (Basic Architecture as Chain Isomorphism)**.** *The autoencoder basis architecture does not induce a chain isomorphism on the persistence complexes associated to the input and output space.*

Proof. If the basis architecture is not bijective, this result follows immediately, as a chain isomorphism between the persistence complexes requires also an isomorphism on the zero-chains.
Assuming bijectivity, in the persistence complex on the input space, there exists a filtration index ϕ where x fullfills the Rips condition with $N \in \mathbb{N}$ elements of $\mathcal{D}$, thus generating at least one k-chain at ϕ with $k \leq N$. Wlog, we assume that in the persistence complex on the output space, the shifted $\mathcal{A}(x)$ fullfills the Rips condition with $N' \in \mathbb{N}$ elements of $\mathcal{O} := \mathcal{A}(\mathcal{D})$ at a filtration level $\phi' \neq \phi$, generating at least one k'-chain with $k \neq k'$. The base architecture removes at least one k-chain at the filtration level ϕ and adds at least one k'-chain at the filtration level ϕ'. Consequently, it does not induce an isomorphism of k-chains at filtration level ϕ, nor of k'-chains at filtration level ϕ'. Therefore, no isomorphism exists between the corresponding persistence complexes. □

Example 5.1 (Chain Modification via Delta Error Shift)**.** *As illustrated in Figure 3, a large generalization error δ can also cause a k-chain to be removed at a filtration level $f = \phi$ and re-added at the same filtration level $f = \phi$.*

However, we excluded this case wlog in the proof of Proposition 5.2.

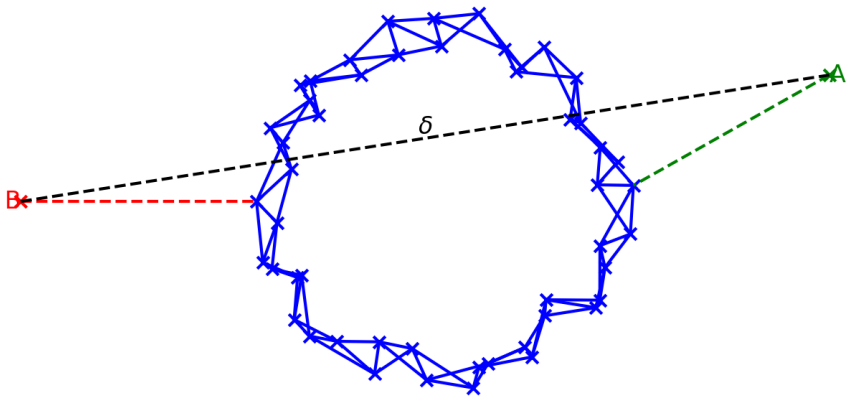


Figure 3: The point A is shifted via the generalization error to position B, having the same inter point distance as A and thus forming a 1-chain at the same filtration level. Thus, albeit the generalization error δ is large, the corresponding 1-chain is removed and added at the same filtration level $f = \phi$.

Proposition 5.3 (Removal of a Chain in Dimension 1)**.** *Consider chains up to homology dimension 1. Let a trained autoencoder architecture $\mathcal{A}$ represent a chain map between the persistence complexes of the input and output spaces. If the generalization error δ causes a 1-chain c' to be removed from the output at a filtration level $f = \phi$, this removal occurs under the condition that this chain forms a cycle such that $\partial_C(c) = 0$ with c being the 1-chain in C_1^ϕ corresponding to c' before its removal.*

Proof. We assume that the generalization error δ causes at least one 1-chain to be removed at a filtration level $f = \phi$.
For the filtration level ϕ, we consider the corresponding segment of the persistence complexes (C, ∂_C) of the input space and $(C', \partial_{C'})$ of the output space, along with the mappings a_d^ϕ induced by the autoencoder between the corresponding chain groups or modules C_d^ϕ for $d = 0, 1$. For the trained autoencoder architecture to induce a chain map, the following diagram must commute for $f = \phi$:

$$\begin{array}{ccc} C_1^\phi & \xrightarrow{a_1^\phi} & C'^\phi_1 \\ \downarrow{\scriptstyle \partial_C} & & \downarrow{\scriptstyle \partial_{C'}} \\ C_0^\phi & \xrightarrow{a_0^\phi} & C'^\phi_0 \end{array} \tag{23}$$

Without the shift of at least one point induced by the generalization error, all mappings a_d^f for $d = 0, 1$ constitute bijections. The removal of a 1-chain c' from

${C'}_1^{\phi}$ now implies that its preimage c in C_1^{ϕ} must be mapped via a_1^{ϕ} to the 0-chain in order to satisfy the condition of the chain map, which requires that k-chains are mapped to k-chains. As C_0^f constitutes the vertex set of the input set for all metric filtration levels f, a_0^f constitutes a bijection for all f. By linearity of the boundary operator, $\partial_{C'}(a_1^{\phi}(c)) = 0$. Hence, for the diagram in Eq. 23 to be commutive, $\partial_C(c) = 0$ which implies c is a cycle. □

Corollary 5.1. *For data whose associated persistence complex exhibits homology up to dimension 1, a trained bijective autoencoder basis architecture cannot induce a quasi-isomorphism.*

Proposition 5.4 (Addition of a Chain in Dimension 1)**.** *Consider chains up to homology dimension 1. Let a trained autoencoder architecture $\mathcal{A}$ represent a chain map between the persistence complexes of the input and output spaces. If the generalization error δ causes a 1-chain c'' to be added to the output at a filtration level $f = \phi'$, this addition occurs under the condition that this chain either constitutes a cycle or shares a boundary with another 1-chain c' in ${C'}_1^{\phi}$.*

Proof. We assume that the generalization error δ causes at least one 1-chain to be added at a filtration level $f = \phi'$.
Again, we investigate the commutativity of the diagram in Eq. (23). We consider a 1-chain c'' added to ${C'}_1^{\phi'}$. If the added chain c' is a cycle, the commutativity of the diagram in Eq. (23) under the bijectivity of a_0^{ϕ} immediately follows. For the second case, there must exist another 1-chain $c' \in {C'}_1^{\phi'}$ such that $\partial_{C'}(c') = \partial_{C'}(c'')$. In other words, $c' - c''$ is a cycle. □

Proposition 5.5 (Removal of a Chain in Dimension d)**.** *In dimension d, it is possible that the trained autoencoder basis architecture fullfills the condition of a chain map and the removed chain does neither induce the removal of a cycle nor a boundary.*

Proof. We assume that the generalization error causes a at least one d-simplex and thus one d-chain to be removed at a filtration level $f = \phi$. In contrast to the 1-dimensional case, the shift of a point results not only in the removal of a d-chain, but also in the removal of chains with dimension $k < d$. In particular, the removal of one vertex of a d-simplex also causes the removal of the faces of this simplex that contain this vertex and thus also the removal of the k-chains generated by the simplices constituting these faces. In particular, the number of the removed k-dimensional faces is $\binom{d}{k}$.
Case (i): Removal of a boundary.

We consider the diagram in Eq. (24).

$$\begin{array}{ccc} C^f_{d+1} & \xrightarrow{a_{d+1,f}} & C'^f_{d+1} \\ \downarrow{\scriptstyle \partial_C^{d+1}} & & \downarrow{\scriptstyle \partial_{C'}^{d+1}} \\ C^f_d & \xrightarrow{a_{d,f}} & C'^f_d \\ \downarrow{\scriptstyle \partial_C^{d}} & \circlearrowleft & \downarrow{\scriptstyle \partial_{C'}^{d}} \\ C^f_{d-1} & \xrightarrow{a_{d-1,f}} & C'^f_{d-1} \end{array} \tag{24}$$

Let c' be a d-chain removed from C'^f_d via the generalization error. Thus, a d-chain $c \in C^f_d$ is mapped to the zero-chain under $a_{d,f}$, ensuring that the mapping $a_{d,f}$ preserves the property of mapping k-chains to k-chains. If $c \in \mathrm{im}(\partial_C^{d+1})$ respectively $c' \in \mathrm{im}(\partial_{C'}^{d+1})$ then the marked diagram in Eq. (24) is commutative by the Fundamental Lemma of Homology.

Case (ii): Removal of a cycle. The argumentation for the removal of a cycle is as in dimension 1.

Case (iii):

We consider the diagram in Eq. (25).

$$\begin{array}{ccc} C^f_d & \xrightarrow{a_{d,f}} & C'^f_d - c' \\ \downarrow{\scriptstyle \partial_C^{d}} & 1 & \downarrow{\scriptstyle \partial_{C'}^{d}} \\ C^f_{d-1} & \xrightarrow{a_{d-1,f}} & C'^f_{d-1} \\ \downarrow{\scriptstyle \partial_C^{d-1}} & 2 & \downarrow{\scriptstyle \partial_{C'}^{d-1}} \\ C^f_{d-2} & \xrightarrow{a_{d-2,f}} & C'^f_{d-2} + \Delta \\ \downarrow{\scriptstyle \partial_C^{d-2}} & 3 & \downarrow{\scriptstyle \partial_{C'}^{d-2}} \\ C^f_{d-3} & \xrightarrow{a_{d-3,f}} & C'^f_{d-3} \end{array} \tag{25}$$

Thus, as a d-simplex consists of $d+1$ $(d-1)$-faces, the removal of one vertex leads to the removal of all faces that contain it, thus also of d $(d-1)$ faces. Hence, a d-chain c' is removed from C'^f_d and d $(d-1)$ chains are removed from C'^f_{d-1}. Thus neither $a_{d,f}$ nor $a_{d-1,f}$ are bijective. To satisfy the conditions of a chain map and thus guarantee commutativity of square 1 in Eq. (25), $a_{d,f}(c) = 0$, whereby c is the d-chain in C^f_d corresponding to c' before its removal. Also, the faces $\partial^d_C(c)$ need to be mapped to the zero chain in C'^f_{d-1} to guarantee commutativity of square 2, with one face c'' remaining. As the remaining $(d-1)$-face in C'^f_{d-1} is no longer part of the boundary of a d-simplex, it now itself has a nontrivial

$(d-2)$-boundary Δ. The chain map condition is still satisfied in square 3 in Eq. (25) if $a_{d-2,f}$ is not surjective. By the Fundamental Lemma of Homology, all changes induced in C'^{f}_{d-1} zero out in C'^{f}_{d-3}. Thus $a_{d-3,f}$ is bijective. □

Proposition 5.6. *Let $\mathcal{A}$ denote a trained autoencoder:*

$$\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^n. \tag{26}$$

Then a nonzero generalization error δ_{gen} does not induce a nonzero interleaving distance between the associated persistence modules on the input and output space.

Proof. ddddd ?

□

Example 5.2. *also for high generalization errors:*

Proposition 5.7. *A trained AE with nonzero generalization error not inducing a quasi-isomorphism at least produces an interleaving distance of XY.*

Autoencoders as (nonlinear) dimensionality reduction methods could provide a positive answer to Question (II). If the neuron operations described in Section 4.1 are unable to alter the torsion effects present in a data set, autoencoders could be used to transform high-dimensional data sets — which, as discussed in Section 4.1, cannot be computationally checked for torsion — into lower dimensions where such checks are feasible. We demonstrate the opposite with an easy example.

Example 5.3 (Neural Networks remove Torsion from Data)**.** *Already the linear operations in a neural network (see Eq. (9)) can remove torsion from a data set. This is demonstrated in Figure 4, where on the left the double loop point cloud $\in \mathbb{R}^3$ from [20], Fig. 4 (a) is shown. The following transformation matrix $\mathbf{T} : \mathbb{R}^3 \to \mathbb{R}^3$ is applied to the double loop point cloud:*

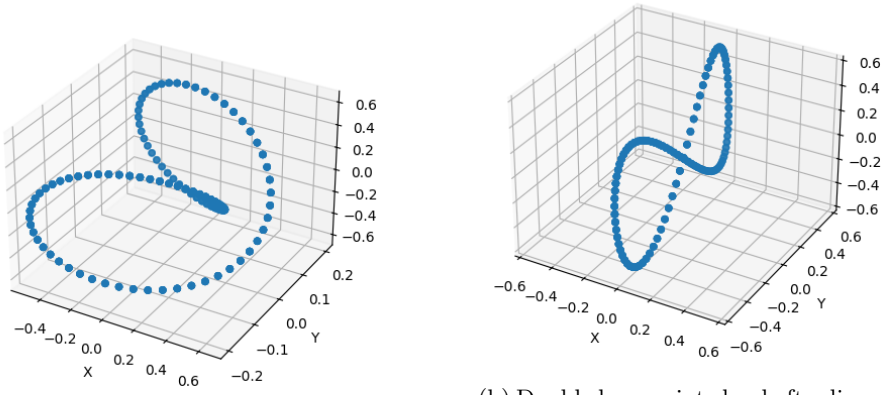
$$\mathbf{T} = \begin{pmatrix} 0.05 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This results in a point cloud without torsion as depicted in Figure 4b.

Conversely, applying $\mathbf{T}^{-1}$ to the dataset shown in Figure 4b illustrates that even through linear transformations, torsion can be introduced into datasets that initially do not exhibit it. This raises the following questions:

VI. From the perspective of torsional effects, what impact do the concatenation of weight matrices and nonlinear activation functions in (deep) neural networks have on the data?

VII. Can we estimate the probability, particularly with regard to the combinatorial and activation architecture of a neural network, that torsion is introduced into the data or, alternatively, that torsion is removed from the data?

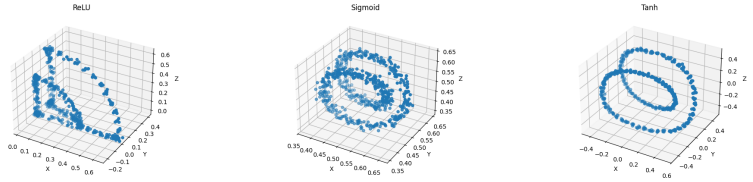


(a) Double loop point cloud.

(b) Double loop point cloud after linear transformation.

Figure 4: Torsion after linear transformation.

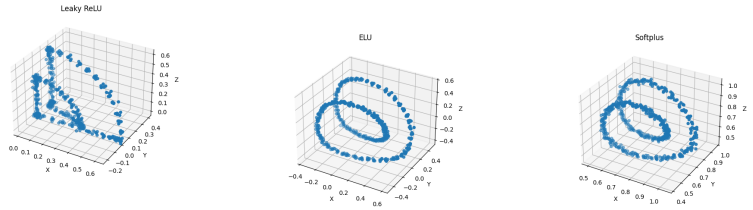
As a first small experiment with regard to answering Question (VI), we applied six common activation functions to the double loop point cloud; the result is shown in Figure 5.



(a) ReLU: No torsion.

(b) Sigmoid: Torsion.

(c) Tanh: Torsion.



(d) Leaky ReLU: No torsion.

(e) ELU: Torsion.

(f) Softplus: Torsion

Figure 5: Double loop point cloud after application of the denoted activation functions. Small noise was added to avoid violation of the general position.

The result suggests that the answer to Question (VI) is highly sensitive to the spatial distribution, scaling, and normalization of the data. This leads to follow-up questions subordinate to Question (VI):

VI (a) Even if torsion remains in the data after applying the neural operations (Eq. 9), to what extent does this affect their placement within the simplicial complex in terms of spatial localization and homology dimension?

VI (b) Can this be checked algorithmically?

6 3D Data in Autoencoders

For a data set embedded in a Euclidian space $\mathbb{R}^d$, integral homology groups may exhibit a torsion part for homology dimension $p \geq 3$ to $p \leq d-1$ [20]. Already in dimension 3, one can construct intuitive and instructive torsional examples, such as the double loop point cloud from Figure 4a. Yet, torsion seems to be rare in $\mathbb{R}^3$, at least in randomly generated datasets [20].

Proposition 6.1. *Under Assumption 1.1, an autoencoder transforming input data $X \subset \mathbb{R}^3$ into a two dimensional latent space L, equipped with a linear decoder activation architecture, cannot reconstruct torsion.*

Proof. We consider a filtered complex K_X whose vertices are the elements of the input data set X. The input data set X is transformed to a latent space $L \subset \mathbb{R}^2$ embedded in the two dimensional Euclidian space by Assumption 1.1. On the latent space a filtered complex K_L is constructed. By Corollary 1.7 in [20], persistence diagrams $D_p(K;\mathbb{F})$ for a filtered complex K with coefficients in a field $\mathbb{F}$ are independent of the choice of the field $\mathbb{F}$ for homology dimension $p = 0$. By Corollary 2.1, for a filtered simplicial complex K embedded in an Euclidian space $\mathbb{R}^M$, the persistent diagrams of homology dimension $p = M-1$ are independent of the choice of field $\mathbb{F}$. Thus, for the filtered simplicial complex $K_L \subset \mathbb{R}^2$, all possible persistence diagrams $D_0(K_L;\mathbb{F})$ and $D_1(K_L;\mathbb{F})$ are independent of the choice of the coefficient field $\mathbb{F}$. By the rank-nullity theorem, a linear decoder maps this two-dimensional latent space to a two-dimensional output space, which by Assumption 1.1 is $X' \subset \mathbb{R}^2$. Thus, an autoencoder with a two-dimensional latent space and a linear decoder cannot reconstruct torsion from three dimensional input data. □

7 Complex Data in Autoencoders

Proposition 7.1. *By transforming data from a d dimensional input space $X \subset \mathbb{R}^d$, $d > 3$ to a λ-dimensional latent space $\lambda < d$ with $\lambda > 2$ and under Assumption 1.1, all torsion effects in homology groups with homology dimension $\geq \lambda - 1$ are removed from the data.*

Proof. By definition, the integral simplicial homology groups derived from a filtered simplicial complex on the input space may exhibit a torsion part for homology dimension $1 \leq p \leq d-1$. We consider a filtered simplicial complex K_L whose vertices are the elements of the latent space L. By Assumption 1.1, the

latent space is a subset of a Euclidian space $L \subset \mathbb{R}^\lambda$. Thus for $\lambda > 2$, the integral homology groups constructed via application of the simplicial homology functor to K_L may exhibit a torsional part for homology dimensions $1 \leq p \leq \lambda - 1$. By Corollary 2.1, for a filtered simplicial complex embedded in a Euclidian space $\mathbb{R}^\lambda$, the persistence diagrams $D_{\lambda-1}(K;\mathbb{F})$ are independent of the choice of coefficient field $\mathbb{F}$. □

Corollary 7.1. *Let $X \subset \mathbb{R}^d$ be the input set to an autoencoder. Let the encoder transform X to a $\lambda < d$ dimensional latent space L, which is a subset of a Euclidian space $\mathbb{R}^\lambda$ by Assumption 1.1. A decoder with a linear architecture cannot fully reconstruct the torsion in X.*

Proof. Torsion subgroups of the integral homology groups can potentially appear up to homology dimension $p = d - 1$ by definition. By the rank-nullity theorem, the output X' reconstructed from the latent space L with a linear decoder has dimension $\leq \lambda$ and is embedded in a Euclidian space $\mathbb{R}^\lambda$ by Assumption 1.1. Thus, by Proposition 7.1, torsional effects in dimensions $\geq \lambda - 1$ are not reconstructible. □

Corollary 7.2. *Let $f_{\text{dec}} := f_\lambda \circ f_{\lambda+(d-n_{l_1})} \circ ... \circ f_{\lambda+(d-n_{l_N})}$ denote the decoder of an autoencoder with nonlinear activation architecture and a λ-dimensional latent space, $\lambda < d$, consisting of N layers with natural numbers $\lambda \leq n_{l_i} \leq d-1$, $0 \leq i \leq N$. Let the subscript denote the dimension of the output space of the respective layer under consideration. Then torsion within homology groups of homology dimension $p < d - n_{l_{i+1}} - 1$ is reconstructible in one layer transition from $f_{\lambda+(d-n_{l_i})}$ to $f_{\lambda+(d-n_{l_{i+1}})}$ under the condition that the decoder architecture is able to reconstruct torsion.*

Corollary 7.2 immediately raises the following question:

VIII. If torsion is reconstructible within the decoder of an autoencoder with nonlinear activation architecture, how is it reconstructed?

The answer to Question (VII) could be that torsion with homology dimension $d - n_{l_i} < p < d - n_{l_{i+1}}$ is reconstructed layerwise for $f_{\lambda+(d-n_{l_i})} \to f_{\lambda+(d-n_{l_{i+1}})}$. Algorithmically, this can only be investigated when the answer to Question VI (b) is affirmative.

8 Notes: Question VII

9 Computational Experiments

9.1 3D Data

9.1.1 Reconstruction of Torsion

Vanilla Autoencoder

Topological Autoencoder

RTD Autoencoder

DIPOLE

9.2 Complex Data

Acknowledgments

References

[1] Manu Aggarwal and Vipul Periwal. "Dory: Computation of persistence diagrams up to dimension two for Vietoris–Rips filtrations of large data sets". In: *Journal of Computational Science* 79 (2024), p. 102290. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2024.102290. URL: https://www.sciencedirect.com/science/article/pii/S1877750324000838.

[2] Marco Antonio Armenta and Pierre-Marc Jodoin. *The Representation Theory of Neural Networks*. 2021. arXiv: 2007.12213 [cs.LG]. URL: https://arxiv.org/abs/2007.12213.

[3] Serguei Barannikov et al. *Representation Topology Divergence: A Method for Comparing Neural Network Representations*. 2022. arXiv: 2201.00058 [cs.LG]. URL: https://arxiv.org/abs/2201.00058.

[4] Ulrich Bauer. "Ripser: Efficient Computation of Vietoris-Rips Persistence Barcodes". In: *Journal of Applied and Computational Topology* 5.3 (2021), pp. 391–423. ISSN: 2367-1726. DOI: 10.1007/s41468-021-00071-5. URL: https://doi.org/10.1007/s41468-021-00071-5.

[5] Jean-Daniel Boissonnat and C. Maria. "Computing persistent homology with various coefficient fields in a single pass". In: *Journal of Applied and Computational Topology* 3 (2019), pp. 59–84. DOI: 10.1007/s41468-019-00025-y.

[6] Peter Bubenik, Vin de Silva, and John Scott. "Metrics for Generalized Persistence Modules". In: *Foundations of Computational Mathematics* 15.6 (2015), pp. 1501–1531. DOI: 10.1007/s10208-014-9229-5. URL: https://doi.org/10.1007/s10208-014-9229-5.

[7] Sebastian Damrich, Philipp Berens, and Dmitry Kobak. *Persistent Homology for High-dimensional Data Based on Spectral Methods*. 2024. arXiv: 2311.03087 [cs.LG]. URL: https://arxiv.org/abs/2311.03087.

[8] Herbert Edelsbrunner and John L. Harer. *Computational Topology. An Introduction*. AMS, 2010.

[9] Herbert Edelsbrunner and János Pach. *Maximum Betti numbers of Čech complexes*. 2023. arXiv: 2310.14801 [math.CO]. URL: https://arxiv.org/abs/2310.14801.

[10] Patrizio Frosini. *Stable comparison of multidimensional persistent homology groups with torsion*. 2010. arXiv: 1012.4169 [math.AT]. URL: https://arxiv.org/abs/1012.4169.

[11] Joseph A. Gallian. *Contemporary Abstract Algebra*. 10th. Boston, MA: Cengage Learning, 2021. ISBN: 9780357692904.

[12] Robert Ghrist. "Barcodes: The persistent topology of data". In: *Bulletin of the American Mathematical Society* 45.1 (2008), pp. 61–75. DOI: 10.1090/S0273-0979-07-01191-3.

[13] Michael Goff. *Extremal Betti numbers of Rips complexes*. 2009. arXiv: 0910.0040 [math.CO]. URL: https://arxiv.org/abs/0910.0040.

[14] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[15] Allen Hatcher. *Algebraic Topology*. Cambridge, UK: Cambridge University Press, 2002. ISBN: 978-0521795401. URL: http://pi.math.cornell.edu/~hatcher/AT/ATpage.html.

[16] Yasuaki Hiraoka et al. *Curse of Dimensionality on Persistence Diagrams*. 2024. arXiv: 2404.18194 [math.ST]. URL: https://arxiv.org/abs/2404.18194.

[17] Nicholas O. Malott and Philip A. Wilsey. "Fast Computation of Persistent Homology with Data Reduction and Data Partitioning". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 880–889. DOI: 10.1109/BigData47090.2019.9006572.

[18] Michael Moor et al. *Topological Autoencoders*. 2021. arXiv: 1906.00722 [cs.LG]. URL: https://arxiv.org/abs/1906.00722.

[19] Dmitriy Morozov. *Omni Field Tutorial for Dionysus2*. https://mrzv.org/software/dionysus2/tutorial/omni-field.html. Accessed: [11.11.2024].

[20] I. Obayashi and M. Yoshiwaki. "Field Choice Problem in Persistent Homology". In: *Discrete and Computational Geometry* 70 (2023), pp. 645–670. DOI: 10.1007/s00454-023-00544-7.

[21] N. Otter et al. "A roadmap for the computation of persistent homology". In: *EPJ Data Science* 6 (2017), p. 17. DOI: 10.1140/epjds/s13688-017-0109-5. URL: https://doi.org/10.1140/epjds/s13688-017-0109-5.

[22] Raúl Rabadán et al. "Identification of Relevant Genetic Alterations in Cancer Using Topological Data Analysis". In: *Nature Communications* 11 (2020), p. 3808. DOI: 10.1038/s41467-020-17659-7. URL: https://doi.org/10.1038/s41467-020-17659-7.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning internal representations by error propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[25] Elchanan Solomon, Alexander Wagner, and Paul Bendich. *From Geometry to Topology: Inverse Theorems for Distributed Persistence*. 2022. arXiv: 2101.12288 [math.AT].

[26] Guillaume Tauzin et al. "Giotto-TDA: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration". In: (2020). DOI: 10.48550/ARXIV.2004.02551. URL: https://arxiv.org/abs/2004.02551.

[27] The GUDHI Project. *GUDHI User and Reference Manual*. 3.1.1. GUDHI Editorial Board, 2020. URL: https://gudhi.inria.fr/doc/3.1.1/.

[28] Dawson S. Thomas et al. "Topological Jet Tagging". In: *Machine Learning and the Physical Sciences Workshop at NeurIPS 2022*. 2022. URL: https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_176.pdf.

[29] Ilya Trofimov et al. *Learning Topology-Preserving Data Representations*. 2023. arXiv: 2302.00136 [cs.LG]. URL: https://arxiv.org/abs/2302.00136.

[30] Alexander Wagner, Elchanan Solomon, and Paul Bendich. *Improving Metric Dimensionality Reduction with Distributed Topology*. 2021. arXiv: 2106.07613 [cs.LG]. URL: https://arxiv.org/abs/2106.07613.

[31] Afra Zomorodian and Gunnar Carlsson. "Computing Persistent Homology". In: *Discrete and Computational Geometry* 33.2 (2005), pp. 249–274. DOI: 10.1007/s00454-004-1146-y. URL: https://doi.org/10.1007/s00454-004-1146-y.