A Extended Topological Densification

In Section 2.3 we describe 0-th persistent homology in relation to topological densification [21]. We now present a concise definition of n-th persistent homology for $n \geq 0$, and elaborate on possible generalizations of the topological densification loss (Definition 2.4). For an introduction to persistent homology and topological data analysis we refer the reader to [8].

The persistent homology pipeline can be subdivided in steps:

From Data to Geometry. The first step consists in associating a nested sequence (i.e., a filtration) of simplicial complexes to the data. Simplicial complexes are simple geometric objects that generalize the notion of a graph and can be described combinatorially. If data is in the form of a finite metric space $\mathcal{D}$ the filtration of simplicial complexes can be for example constructed by computing the Vietoris-Rips complex, the Cech complex or the Witness complex at scale ε, for increasing values of the parameter $\varepsilon \in \mathbb{R}$.

From Geometry to Algebra. The topology of a simplicial complex and in particular its homology can be computed in an algorithmic way through simplicial homology. Fixed a natural number n and a coefficient field, the n-th homology of a simplicial complex is a vector space encoding connectivity of the simplicial complex. The 0-th homology describes connected components, 1-st homology describes cycles that are not bounded by faces, 2-nd homology describes voids and for $n \geq 2$ one can think of higher dimensional analogues. In the cases of a sequence of simplicial complexes, n-th homology can be applied to the whole sequence to obtain a sequence of vector spaces and linear maps. This is referred to n-th persistent homology module.

Representing Persistent Homology. Persistent homology modules are well known and simple algebraic objects, due for example to the fact that the sequence of vector spaces constituting a persistence module is indexed by $\mathbb{R}$ and that it is essentially discrete. Due to Gabriel's theorem, or equivalently the decomposition theorem of modules over a Principle Ideal Domain, persistent homology modules admit a barcode decomposition, they can be decomposed as sum of addends called bars. A bar is completely described by a pair of real numbers: the birth and the death of the bar. The collection of all birth and death pairs of the bars in a barcode decomposition of a persistence module is the so called persistence diagram. Persistence diagrams and their features are used both as data descriptors and for improving data representations in deep learning, for example through regularization.

As in Definition 2.4 we denote by $\mathcal{D}$ the dataset, by $\varphi\colon \mathcal{X} \to \mathcal{Z}$ an encoder mapping data to a latent space, and assume the data is partitioned into classes $\mathcal{D} = \mathcal{D}_1 \sqcup \cdots \sqcup \mathcal{D}_K$. The topological densification loss is defined as

$$\mathcal{R} = \sum_{i=1}^{K} \sum_{w \in \dagger(\varphi(\mathcal{D}_i))} |w - \beta|.$$

where $\dagger(\varphi(\mathcal{D}_i))$ is the set of death times in the persistence diagram of $\varphi(D_i)$. For 0-th persistence, death times correspond to lengths of bars in a barcode, since all bars start at parameter value 0. Consider now the case of n-th persistent homology of the $\varphi(\mathcal{D}_i)$ and the corresponding persistence diagrams $\ddagger(\varphi(\mathcal{D}_i))$. The formula for of the topological densification loss can be generalized as:

$$\mathcal{R} = \sum_{i=1}^{K} \sum_{(a,b) \in \ddagger(\varphi(\mathcal{D}_i))} |\ell(a,b) - \beta|.$$

where l is a real valued function assigning a weight to each point in the persistence diagram. Examples of such functions are the length of a bar $\ell(a,b) = b - a$ or the lifespan of a bar [1] – a generalization of length defined as $\ell_F(a,b) = F(b) - F(a)$, where $F\colon \mathbb{R} \to \mathbb{R}$ is an increasing bijection.

B Batch Construction

As mentioned in Section 3.2, the original work on topological densification [21] proposes the following batch construction. A batch consists of b sub-batches, where each sub-batch contains n samples from the same class.

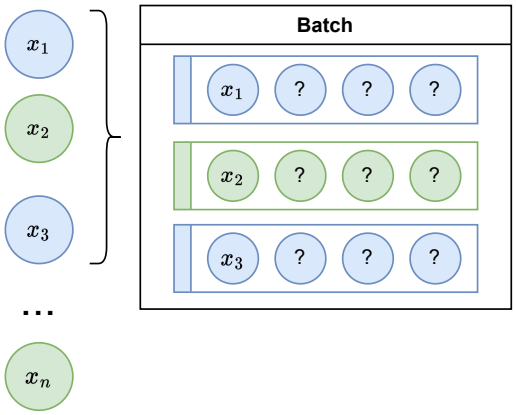

Figure 7: The original dataloader for topological densification.

More specifically, the dataloader in the original work is implemented as follows (see Figure 7 for an illustration):

1. A batch of size b is sampled from the original dataset.
2. For each datapoint x in the batch, a sub-batch is constructed by sampling with replacement $n-1$ more datapoints with the same class as x.

This approach comes with two major issues. Firstly, in the presence of significant class imbalance, classes are likely to be under-represented within batches. This can potentially lead to catastrophic forgetting and unstable training. Secondly, this method is computationally intensive, since training requires processing the original dataset n times in a single epoch.

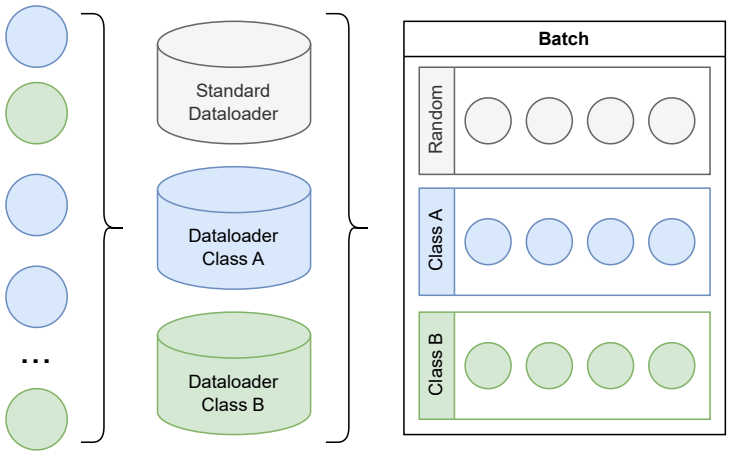

Figure 8: Our dataloader for topological densification.

We propose a new dataloader that follows the same high-level principle, but addresses the above concerns. Our dataloader is implemented as follows (see Figure 8 for an illustration):

1. For each class, we create a separate dataloader containing all the data from that class. Additionally, we create a standard dataloader containing all the data.
2. We aggregate n samples from each dataloader, resulting in a batch consisting of $K+1$ sub-batches, where K is the number of classes.

This new dataloader preserves the class structure necessary for applying the topological regularization loss. Its computational overhead is comparable to training directly with the standard dataloader. Moreover, it prevents class imbalance, since each class is guaranteed to be represented at least n times within a batch.

The introduction of the sub-batch from the standard dataloader is relevant for the normalization procedure involved in our robust relative transformation. Namely, we use the (encoding via φ of the) sub-batch from the standard dataloader as $\mathcal{B}$ (notation from Section 3.1) when fine-tuning robust relative representations. This is crucial, for example, when the original dataset exhibits class imbalance. In that case, the normalization should be performed with a set $\mathcal{B}$ that is representative of the original dataset – which is addressed by the sub-batch from the latter – while the topological loss still benefits from class-balanced data – which is addresses by the other class-specific sub-batches.