Connectivity-Optimized Representation Learning via Persistent Homology – Supplementary material –

Christoph D. Hofer [1] Roland Kwitt [1] Mandar Dixit [2] Marc Niethammer [3]

This supplementary material contains all proofs omitted in the main submission. For readability, all necessary definitions, theorems, lemmas and corollaries are restated (in dark blue) and the numbering matches the original numbering.

Additional (technical) lemmas are prefixed by the section letter, e.g., Lemma A.1.

A. Proofs for Section 3

First, we recall that the *connectivity loss* is defined as

$$\mathcal{L}_\eta(S) = \sum_{t \in \dagger(S)} |\eta - \varepsilon_t| \tag{4}$$

Definition 3. Let $S \subset \mathbb{R}^n$, $|S| = b$ and $z_i \in S$. We define the indicator function

$$\mathbf{1}_{i,j}(z_1, \dots, z_b) = \begin{cases} 1 & \exists t \in \dagger(S) : \varepsilon_t = ||z_i - z_j|| \\ 0 & \text{else} \end{cases} ,$$

where $\{i,j\} \subset [b]$ and $(\varepsilon_k)_{k=1}^M$ is the increasing sequence of all pairwise distance *values* of S.

Theorem 1. *Let $S \subset \mathbb{R}^n$, $|S| = b$, such that the pairwise distances are unique. Further, let $\mathcal{L}_\eta$ be defined as in Eq. (4) and $\mathbf{1}_{i,j}$ as in Definition 3. Then,*

$$\mathcal{L}_\eta(S) = \sum_{\{i,j\} \subset [b]} \big|\eta - \|z_i - z_j\|\big| \cdot \mathbf{1}_{i,j}(z_1, \dots, z_b) \ .$$

Proof. We have to show that

$$\sum_{t \in \dagger(S)} |\eta - \varepsilon_t|$$

from Eq. (4), denoted as A, equals the right-hand side of Theorem 1, denoted as B.

Part 1 ($A \leq B$). Let $t \in \dagger(S)$. Since the pairwise distances of S are unique, t is contained only once in the multi-set $\dagger(S)$ and we can treat $\dagger(S)$ as an *ordinary* set. Further, there is a (unique) $\{i_t, j_t\}$ such that $\varepsilon_t = \|z_{i_t} - z_{j_t}\|$ and hence $\mathbf{1}_{i_t,j_t}(z_1, \dots, z_b) = 1$. This means every summand in A is also present in B. As all summands are non-negative, $A \leq B$ follows.

Part 2 ($B \leq A$). Consider $\{i,j\} \subset [b]$ contributing to the sum, i.e., $\mathbf{1}_{i,j}(z_1, \dots, z_b) = 1$. By definition

$$\exists t \in \dagger(S) : \varepsilon_t = \|z_i - z_j\|$$

and therefore the summand corresponding to $\{i,j\}$ in B is present in A. Again, as all summands are non-negative $B \leq A$ follows, which concludes the proof. □

Lemma A.1. *Let $S \subset \mathbb{R}^n$, $|S| = b$, such that the pairwise distances are unique. Then, $\mathbf{1}_{i,j}(S)$ is locally constant in S. Formally, let $1 \leq u \leq b, 1 \leq v \leq n$, $h \in \mathbb{R}$ and*

$$S' = \{z_1, \dots, z_{u-1}, z_u + h \cdot e_v, z_{u+1}, \dots, z_b\}$$

where e_v is the v-th unit vector. Then,

$$\exists \xi > 0 : |h| < \xi \Rightarrow \mathbf{1}_{i,j}(S) = \mathbf{1}_{i,j}(S') \ .$$

Proof. $\mathbf{1}_{i,j}(X)$ is defined defined via $\dagger(X)$, which, in turn, is defined via the Vietoris-Rips filtration of X. Hence, it is sufficient to show that the corresponding Vietoris-Rips filtrations of S and S' are equal, which we will do next.

Let $(\varepsilon_k)_{k=1}^M$ be the increasing sorted sequence of pairwise distance *values* of S. As all pairwise distances are unique, there is exactly one $\{i_k, j_k\}$ for each k such that

$$\varepsilon_k = \|z_{i_k} - z_{j_k}\| \ .$$

Further, let $S' = \{z'_1, \dots, z'_b\}$ be such that

$$z'_i = \begin{cases} z_i & i \neq u \\ z_u + h \cdot e_v & i = u \end{cases} ,$$

and $\varepsilon'_k = \|z'_{i_k} - z'_{j_k}\|$. We now show that $(\varepsilon'_k)_{k=1}^M$ is sorted and strictly increasing. First, let

$$\mu = \min_{1 \leq k < M} \varepsilon_{k+1} - \varepsilon_k \ . \tag{1}$$

[1]Department of Computer Science, University of Salzburg, Austria [2]Microsoft [3]UNC Chapel Hill. Correspondence to: Christoph D. Hofer <chr.dav.hofer@gmail.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

By construction, it follows that $\mu > 0$. Now, by the triangle inequality,

$$\big| \|z_i' - z_j'\| - \|z_i - z_j\| \big| \leq \big| \|z_i - z_j\| + |h| - \|z_i - z_j\| \big| = |h| \enspace ,$$

which is equivalent to

$$-|h| \leq \|z_i' - z_j'\| - \|z_i - z_j\| \leq |h| \enspace .$$

This yields

$$\|z_i' - z_j'\| \geq \|z_i - z_j\| - |h| \qquad (2)$$

and

$$-\|z_i' - z_j'\| \geq -\|z_i - z_j\| - |h| \enspace . \qquad (3)$$

Using Eqs. (2) and (3) we get

$$\begin{aligned} \varepsilon_{k+1}' - \varepsilon_k' &= \|z_{i_{k+1}}' - z_{j_{k+1}}'\| - \|z_{i_k}' - z_{j_k}'\| \\ &\geq \|z_{i_{k+1}} - z_{j_{k+1}}\| - |h| - \|z_{i_k} - z_{j_k}\| - |h| \\ &= \varepsilon_{k+1} - \varepsilon_k - 2|h| \\ &\overset{\text{by Eq. (1)}}{\geq} \mu - 2|h| \enspace . \end{aligned}$$

Overall, $(\varepsilon_k')_{k=1}^M$ is sorted and strictly increasing if

$$\mu - 2|h| > 0 \Leftrightarrow |h| < \frac{\mu}{2} \enspace .$$

It remains to show that the Vietoris-Rips filtration

$$\emptyset \subset \mathcal{V}_0(S) \subset \mathcal{V}_{\varepsilon_1/2}(S) \subset \cdots \subset \mathcal{V}_{\varepsilon_M/2}(S)$$

is equal to

$$\emptyset \subset \mathcal{V}_0(S') \subset \mathcal{V}_{\varepsilon_1'/2}(S') \subset \cdots \subset \mathcal{V}_{\varepsilon_M'/2}(S') \enspace .$$

For

$$\mathcal{V}_0(S) = \{\{1\}, \ldots, \{N\}\} = \mathcal{V}_0(S')$$

this is obvious. For $f_S, f_{S'}$, as in Definition 1 (main paper; Vietoris-Rips complex), we get

$$f_S^{-1}(\varepsilon_{k+1}/2) = \{\{i_k, j_k\}\} = f_{S'}^{-1}(\varepsilon_{k+1}'/2)$$

since

$$\varepsilon_k = \|z_{i_k} - z_{j_k}\| \text{ and } \varepsilon_k' = \|z_{i_k}' - z_{j_k}'\|$$

and the pairwise distances are unique. Now, by induction

$$\begin{aligned} \mathcal{V}_{\varepsilon_{k+1}'/2}(S') &= \mathcal{V}_{\varepsilon_k'/2}(S') \cup f_{S'}^{-1}(\varepsilon_{k+1}'/2) \\ &= \mathcal{V}_{\varepsilon_k/2}(S) \cup f_{S'}^{-1}(\varepsilon_{k+1}'/2) \\ &= \mathcal{V}_{\varepsilon_k/2}(S) \cup f_S^{-1}(\varepsilon_{k+1}/2) = \mathcal{V}_{\varepsilon_{k+1}/2}(S) \enspace . \end{aligned}$$

Setting $\xi = \mu/2$ concludes the proof. □

Theorem 2. *Let $S \subset \mathbb{R}^n$, $|S| = b$, such that the pairwise distances are unique. Then, for $1 \leq u \leq b$ and $1 \leq v \leq n$, the partial (sub-)derivative of $\mathcal{L}_\eta(S)$ w.r.t. the v-th coordinate of z_u exists, i.e.,*

$$\frac{\partial \mathcal{L}_\eta(S)}{\partial z_{u,v}} = \sum_{\{i,j\} \subset [b]} \frac{\partial \big|\eta - \|z_i - z_j\|\big|}{\partial z_{u,v}} \cdot \mathbf{1}_{i,j}(z_1, \ldots, z_b) \enspace .$$

Proof. By Theorem 1, we can write

$$\mathcal{L}_\eta(S) = \sum_{\{i,j\} \subset [b]} \big|\eta - \|z_i - z_j\|\big| \cdot \mathbf{1}_{i,j}(z_1, \ldots, z_b) \enspace .$$

Further, from Lemma A.1, we know that $\mathbf{1}_{i,j}$ is locally constant for u, v. Consequently, the partial derivative w.r.t. $z_{u,v}$ exists and is zero. The rest follows from the product rule of differential calculus. □

B. Proofs for Section 4

Lemma 1. *Let $2 \leq b \leq m$ and $M \subset \mathbb{R}^n$ with $|M| = m$ such that for each $S \subset M$ with $|S| = b$, it holds that S is α-β-connected. Then, for $d = m - b$ and $z \in M$ arbitrary but fixed, we find $M_z \subset M$ with $|M_z| = d + 1$ and $M_z \subset B(z, \alpha, \beta)$.*

Proof. Let $z \in M$. Our strategy is to iteratively construct a set of points

$$\{z_1, \ldots, z_{d+1}\} \subset B(z, \alpha, \beta) \cap (M \setminus \{z\}) \enspace .$$

First, consider some $S^{(1)} \subset M$ with $z \in S^{(1)}$ and $|S^{(1)}| = b$. Since $S^{(1)}$ is α-β-connected (by assumption), there is $S^{(1)} \ni z_1 \in B(z, \alpha, \beta)$.

By repeatedly considering $S^{(i)} \subset M$ with $z_i \in S^{(i)}$ and $|S^{(i)}| = b$, we can construct $M_z^{(i)} = \{z_1, \ldots, z_i\}$ for $i \leq d = m - b$. It holds that

$$|M \setminus M_z^{(i)}| = m - i \geq m - d = m - (m - b) = b \enspace . \qquad (5)$$

Hence, we find $S^{(i+1)} \subset M \setminus M_z^{(i+1)}$ with $z \in S^{(i+1)}$ such that $|S^{(i+1)}| = b$. Again, as $S^{(i+1)}$ is α-β-connected, there is $S^{(i+1)} \ni z_{i+1} \in B(z, \alpha, \beta)$. Overall, this *specific procedure* allows constructing $d+1$ points, as for $i \geq d+1$, Eq. (5) is no longer fulfilled. □

Corollary 1. *Let $2 \leq b \leq m$ and $M \subset \mathbb{R}^n$ with $|M| = m$ such that for each $S \subset M$ with $|S| = b$, it holds that S is α-β-connected. Then M is $(m - b + 1)$-β-dense.*

Proof. By Lemma 1, we can construct $m - b + 1$ points, M_z, such that $M_z \subset B(z, \alpha, \beta)$. Conclusively,

$$y \in M_z \Rightarrow \|z - y\| \leq \beta \enspace .$$

□

Corollary 2. *Let $2 \leq b \leq m$ and $M \subset \mathbb{R}^n$ with $|M| = m$ such that for each $S \subset M$ with $|S| = b$, it holds that S is α-β-connected. Then, for $\varepsilon > 0$ and $m - b + 1 > \mathcal{E}^{\varepsilon,n}_{\alpha,\beta}$, it follows that M is not ε-separated.*

Proof. Choose some $z \in M$. By Lemma 1, we can construct $m - b + 1$ points, M_z, such that $M_z \subset B(z, \alpha, \beta)$. The distance induced by $\|\cdot\|$ is translation invariant, hence

$$\mathcal{E}^{\varepsilon,n}_{\alpha,\beta} = N_\varepsilon\big(B(z,\alpha,\beta)\big) \enspace .$$

If $m-b+1 > \mathcal{E}^{\varepsilon,n}_{\alpha,\beta}$, we conclude that M_z is not ε-separated and therefore M is not ε-separated. □

Lemma 2. *Let $\varepsilon < 2\alpha$ and $\alpha < \beta$. Then, in $(\mathbb{R}^n, \|\cdot\|_1)$, it holds that $\mathcal{E}^{\varepsilon,n}_{\alpha,\beta} \leq (2\beta/\varepsilon + 1)^n - (2\alpha/\varepsilon - 1)^n$.*

Proof. Let $M \subset B(0,\alpha,\beta)$ such that M is ε-separated. Then, the open balls $B^0(z, \varepsilon/2)$, $z \in M$, are pairwise disjointly contained in $B(0, \alpha - \varepsilon/2, \beta + \varepsilon/2)$. To see this, let $y \in B^0(z, \varepsilon/2)$. We get

$$\|y\| \leq \|y - z\| + \|z\| < \varepsilon/2 + \beta$$

and (by the reverse triangle inequality)

$$\begin{aligned}\|y\| = \|z - (z-y)\| &\geq \big|\|z\| - \|z-y\|\big| \\ &\geq \|z\| - \|z-y\| \geq \alpha - \varepsilon/2 \enspace .\end{aligned}$$

Hence, $y \in B(0,\alpha,\beta)$. The balls are pairwise disjoint as M is ε-separated and the radius of each ball is chosen as $\varepsilon/2$. Let λ denote the Lebesgue measure in $\mathbb{R}^n$. It holds that

$$\begin{aligned}|M| \cdot \lambda\big(B^0(0,\varepsilon/2)\big) &= \lambda\left(\bigcup_{z\in M} B^0(z,\varepsilon/2)\right) \\ &\leq \lambda\big(B(0,\alpha - \varepsilon/2, \beta + \varepsilon/2)\big)\end{aligned}$$

as λ is translation invariant and

$$\bigcup_{z\in M} B^0(z,\varepsilon/2) \subset B(0, \alpha - \varepsilon/2, \beta + \varepsilon/2) \enspace .$$

The volume of the $\|\cdot\|_1$-ball with radius r is

$$\lambda\big(B(0,r)\big) = \frac{2^n}{n!} r^n \enspace .$$

Hence, we get

$$|M| \cdot \frac{\varepsilon^n}{n!} \leq \frac{2^n}{n!}\left((\beta + \varepsilon/2)^n - (\alpha - \varepsilon/2)^n\right)$$

and thus

$$\begin{aligned}|M| &\leq \frac{2^n}{\varepsilon^n} \cdot \big((\beta + \varepsilon/2)^n - (\alpha - \varepsilon/2)^n\big) \\ &= \frac{2^n}{\varepsilon^n} \cdot \frac{\varepsilon^n}{2^n}\big((2\beta/\varepsilon + 1)^n - (2\alpha/\varepsilon - 1)^n\big) \\ &= (2\beta/\varepsilon + 1)^n - (2\alpha/\varepsilon - 1)^n \enspace .\end{aligned}$$

As the upper bound holds for any M, it specifically holds for the largest M, which bounds the metric entropy $\mathcal{E}^{\varepsilon,n}_{\alpha,\beta}$ and completes the proof. □

C. Parallel persistent homology computation

While there exist many libraries for computing persistent homology (`DIPHA` (Bauer et al., 2014a), `Dinoysus`[1], `JavaPlex`[2] (Tausz et al., 2014), `GUDHI`[3]) of a filtered simplicial complex, or fast (`RIPSER`[4]) and approximate (`SimBa`) (Dey et al., 2016) computation of Vietoris-Rips persistent homology, we are not aware of an available implementation that

(P1) fully operates on the GPU and

(P2) offers easy access to the persistence pairings.

As most deep learning platforms are optimized for GPU computations, (P1) is important to avoid efficiency bottlenecks caused by expensive data transfer operations between main memory and GPU memory; (P2) is required to enable the integration of persistent homology in an automatic differentiation framework, such as PyTorch.

Next, we present a straightforward (and not necessarily optimal) variant of the *standard* reduction algorithm to compute persistent homology, as introduced in (Edelsbrunner & Harer, 2010, p. 153), that offers both properties. While many improvements of our parallelization approach are possible, e.g., using *clearing* (Bauer et al., 2014b) or computing cohomology (de Silva et al., 2011) instead, we do not follow these directions here. We only present a simple parallel variant that is sufficient for the purpose of this work.

The core idea of the original reduction algorithm is to transform the boundary matrix of a filtered simplicial complex such that the "birth-death" times of its homological features can be easily read off. More precisely, the *boundary matrix* (Edelsbrunner & Harer, 2010) is transformed to its *reduced form* (see Definition C.1) via left-to-right column additions, defined in Algorithm 1.

First, we need to define what is meant by a *reduced form* of a boundary matrix B over $\mathbb{Z}_2^{m\times n}$.

Definition C.1. Let $B \in \mathbb{Z}_2^{m\times n}$ and $B[i]$, $B[\leq i]$ denote the i-th column and the sub-matrix of the first i columns, resp., of B. Then, for $B[j] \neq 0$, we define

$$\texttt{low}(B, i) = j$$

iff j is the row-index of the lowest 1 in $B[i]$. For convenience, we set

$$\texttt{low}(B, i) = -1$$

for $B[j] = 0$. We call B *reduced* iff for $1 \leq i < j \leq n$

$$B[i], B[j] \neq 0 \Rightarrow \texttt{low}(B, i) \neq \texttt{low}(B, j) \enspace .$$

[1] http://www.mrzv.org/software/dionysus2
[2] https://appliedtopology.github.io/javaplex/
[3] http://gudhi.gforge.inria.fr
[4] https://github.com/Ripser/ripser

Algorithm 1 Column addition

```
function ADD($B, i, j$):
    $B[j] \leftarrow B[j] + B[i]$        ▷ Addition in $\mathbb{Z}_2$
end function
```

Next, we restate the original (sequential) reduction algorithm. Let ∂ be the boundary matrix of a filtered simplicial complex.

Algorithm 2 Standard PH algorithm (Edelsbrunner & Harer, 2010, p. 153)

```
B $\leftarrow \partial$
for $i \leftarrow 1, n$ do
    while $\exists j_0 < j : \texttt{low}(B, j_0) = \texttt{low}(B, j)$ do
        ADD($B, j_0, j$)
    end while
end for
```

Algorithm 2 consists of two nested loops. We argue that in case column additions would be data-independent, we could easily perform these operations in parallel without conflicts. To formalize this idea, let us consider a set M of index pairs

$$M = \{(i_k, j_k)\}_k \subset \{1, \dots, b\} \times \{1, \dots, b\} \ .$$

If the conditions

(i) $\{i_k\}_k \cap \{j_k\}_k = \emptyset$, and

(ii) $\forall j_k : \exists! i_k : (i_k, j_k) \in M$

are satisfied, the ADD(B, i_k, j_k) operations from Algorithm 1 are data-independent. Informally, condition (i) ensures that no column is target *and* origin of a merge operation and condition (ii) ensures that each column is targeted by at most one merging operation. In the following definition, we construct two auxiliary operators that will allows us to construct M such that conditions (i) and (ii) are satisfied.

Definition C.2. Let $B \in \mathbb{Z}_2^{m \times n}$ and $1 \le j \le m$. We define

$$I(B, j) = \begin{cases} \emptyset & |\{i : \texttt{low}(B, i) = j\}| < 2 \\ \{i : \texttt{low}(i) = j\} & \text{else} \end{cases}$$

and

$$M(B, j) = \begin{cases} \emptyset & \text{if } I(B, j) = \emptyset \\ \mu(B, j) \times I(B, j) \setminus \mu(B, j) & \text{else} \end{cases}$$

where $\mu(B, j) = \{\min I(B, j)\}$. Finally, let

$$M(B) = \bigcup_{j=1}^{n} M(B, j) \ .$$

By construction, it holds that $M(B) = \emptyset$ iff B is reduced. We can now propose a parallel algorithm, i.e., Algorithm 3, that iterates until $M(B) = \emptyset$.

Algorithm 3 GPU PH algorithm

```
function ADD PARALLEL($B, M$):
    parallel for $(i, j) \in M$ do
        ADD($B, i, j$)
    end parallel for
end function

$B \leftarrow \partial$
$M \leftarrow M(B)$
while $M \neq \emptyset$ do
    ADD PARALLEL($B, M$)
    $M \leftarrow M(B)$
end while
```

Upon termination, $M(B) = \emptyset$, and hence B is reduced. It only remains to show that termination is achieved after a finite number of iterations.

Lemma C.1. *For $B \in \mathbb{Z}_2^{m \times n}$, Algorithm 3 terminates after finitely many iterations.*

Proof. Let $B^{(k)}$ be the state of B in the k-th iteration. For $1 \le l \le n$ it holds that $M(B^{(k)}[\le l]) = \emptyset$ if $B^{(k)}[\le l]$ is reduced. Conclusively, for $k' > k$

$$B^{(k)}[\le l] \text{ is reduced } \Rightarrow B^{(k')}[\le l] \text{ is reduced}$$

as $B[\le l]$ does not change any more after the k-th iteration. Hence we can inductively show that the algorithm terminates after finitely many iterations.
First, note that $B^{(k)}[\le 1]$ is reduced. Now assume $B^{(k)}[\le l]$ is reduced and consider $B^{(k)}[\le l+1]$. If $B^{(k)}[\le l+1]$ is not reduced

$$M\left(B^{(k)}[\le l+1]\right) \subset \{1, \dots, l\} \times \{l+1\}$$

as $B^{(k)}[\le l]$ is already reduced. Thus, if the algorithm continues to the $k+1$-th iteration the lowest 1 of $B^{(k)}[l+1]$ is eliminated and therefore

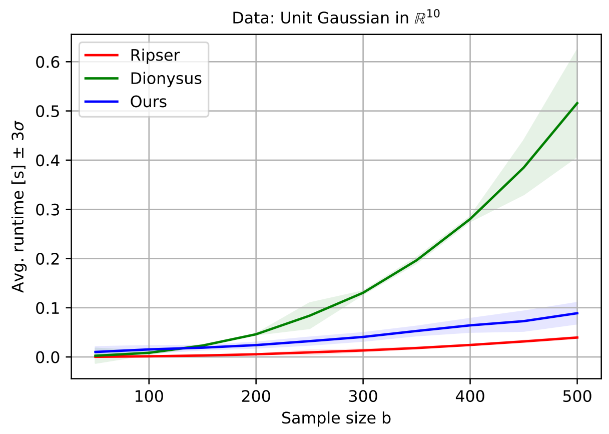
$$\texttt{low}(B^{(k+1)}, l+1) < \texttt{low}(B^{(k)}, l+1) \ .$$

Hence, after $d \le \texttt{low}(B^{(k)}, l+1)$ iterations $B^{(k+d)}[\le l+1]$ is reduced as either $B^{(k+d)}[l+1] = 0$ or there is no $j \le l$ such that

$$\texttt{low}\left(B^{(k+d)}[\le l], j\right) = \texttt{low}\left(B^{(k+d)}[\le l+1], l+1\right) \ .$$

In consequence $B^{(k_0)}[\le n] = B^{(k_0)}$ is reduced for $k_0 < \infty$ which concludes the proof. □

Figure 1. Runtime comparison of `Ripser` & `Dionysus` (both CPU) vs. our parallel GPU variant. Runtime (in seconds) is reported for 0-dimensional VR persistent homology, computed from random samples of size b drawn from a unit multivariate Gaussian in $\mathbb{R}^{10}$.



Runtime study. We conducted a simple runtime comparison to `Ripser` and `Dionysus` (which both run on the CPU). Both implementations are available through Python wrappers[5]. `Dionysus` implements persistent cohomology computation (de Silva et al., 2011), while `Ripser` implements multiple recent algorithmic improvements, such as the aforementioned clearing optimization as well as computing cohomology. Rips complexes are built using $\|\cdot\|_1$, up to the enclosing radius of the point cloud.

Specifically, we compute 0-dimensional features on samples of varying size (b), drawn from a unit multivariate Gaussian in $\mathbb{R}^{10}$. Runtime is measured on a system with ten Intel(R) Core(TM) i9-7900X CPUs (3.30GHz), 64 GB of RAM and a Nvidia GTX 1080 Ti GPU. Figure 1 shows runtime in seconds, averaged over 50 runs. Note that in this experiment, runtime includes construction of the Rips complex as well.

While `Ripser` is, on average, slightly faster than our implementation, we note that for mini-batch sizes customary in training neural networks (e.g., 32, 64, 128), the runtime difference is negligible, especially compared to the overall cost of backpropagation. Importantly, our method integrates well into existing deep learning frameworks, such as PyTorch, and thus facilitates to easily experiment with new loss functions, such as the proposed connectivity loss.

D. Supplementary figures

Fig. 2 shows a second variant of Fig. 6 from the main paper, only that we replace CIFAR-10 with TinyImage-Net (testing portion). The autoencoder was trained on the training portion of CIFAR-100.

[5]For `Ripser`, see `https://scikit-tda.org/`

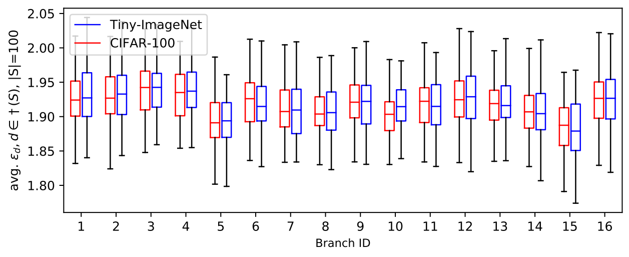


Figure 2. Average $\varepsilon_d, d \in \dagger(S)$, per branch, computed from batches, S, of size 100 over CIFAR-100 (test split) and Tiny-ImageNet (test split); f_θ is learned from the training portion of CIFAR-100 with $\eta = 2$.

E. Algorithmic summary

Algorithm 4 provides a high-level description of the workflow to apply the presented method for one-class learning.

Algorithm 4 Summary of training steps

```
Parameters: $\eta > 0$ (scaling parameter for $\mathcal{L}_\eta$); $\lambda > 0$ (weighting for $\mathcal{L}_\eta$); $B \geq 1$ (number of branches); $D \geq 1$ (branch dimensionality); $b$ (mini-batch size);
Remark: These are all global parameters.
  function SLICE($z, j$):
      return $z[D \cdot (j-1) : D \cdot j]$
  end function
Step 1: Autoencoder training
  Train $g_\phi$ and $f_\theta$ using an auxiliary unlabled dataset $\{a_1, \ldots, a_M\}$, minimizing (over batches of size $b$)
      $\frac{1}{b}\sum_{i=1}^{b} \|a_i - g_\phi \circ f_\theta(a_i)\|_1 + \lambda \sum_{j=1}^{B} \mathcal{L}_\eta(\{z_1^j, \ldots, z_b^j\})$
  where $z_i = f_\theta(a_i)$ with $z_i^j = \text{SLICE}(z_i, j)$.
  Remark: This autoencoder can be re-used. That is, if we already have $f_\theta$ trained on $\{a_1, \ldots, a_M\}$ (e.g., from another one-class scenario) using the same $\eta, B, D$ parameter choices, autoencoder training can be omitted.
Step 2: Create one-class model
  For one-class samples $\{x_1, \ldots, x_m\}$, compute and store $z_i^j = \text{SLICE}(f_\theta(x_i), j)$.
Step 3: Evaluate one-class model
  For each new sample $y_*$, obtain $y_*^j = \text{SLICE}(f_\theta(y_*), j)$ and compute the one-class score
      $s(y_*) = \sum_{j=1}^{B} \left| \left\{ z_i^j : \|z_*^j - z_i^j\| \leq \eta, 1 \leq i \leq m \right\} \right|$ .
  using the stored $z_i^j$ from Step 2.
```

References

Bauer, U., Kerber, M., and Reininghaus, J. Distributed computation of persistent homology. In *ALENEX*, 2014a.

Bauer, U., Kerber, M., and Reininghaus, J. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III*, pp. 103–117. Springer, 2014b.

de Silva, V., Morozov, D., and Vejdemo-Johansson, M. Dualities in persistent (co)homology. *Inverse Problems*, 27 (12):124003, 2011.

Dey, T., Shi, D., and Wang, Y. SimBa: An efficient tool for approximating Rips-filtration persistence via simplicial batch-collapse. In *ESA*, 2016.

Edelsbrunner, H. and Harer, J. L. *Computational Topology : An Introduction*. American Mathematical Society, 2010.

Tausz, A., Vejdemo-Johansson, M., and Adams, H. JavaPlex: A research software package for persistent (co)homology. In *ICMS*, 2014.