— The Flood Complex —
Large-Scale Persistent Homology on Millions of Points

Florian Graf[1,†] ✉ **, Paolo Pellizzoni**[2,†] ✉ **, Martin Uray**[1,3]**, Stefan Huber**[3]**, Roland Kwitt**[1]
[1]University of Salzburg, Austria
[2]Max Planck Institute of Biochemistry, Germany
[3]Josef Ressel Centre for Intelligent and Secure Industrial Automation,
University of Applied Sciences, Salzburg, Austria

Abstract

We consider the problem of computing Persistent Homology (PH) for large-scale Euclidean point cloud data, aimed at downstream machine learning tasks, where the exponential growth of the most widely-used Vietoris-Rips complex imposes serious computational limitations. Although more scalable alternatives such as the Alpha complex or sparse Rips approximations exist, they often still result in a prohibitively large number of simplices. This poses challenges in the complex construction and in the subsequent PH computation, prohibiting their use on large-scale point clouds. To mitigate these issues, we introduce the *Flood complex*, inspired by the advantages of the Alpha and Witness complex constructions. Informally, at a given filtration value $r \geq 0$, the Flood complex contains all simplices from a Delaunay triangulation of a small subset of a point cloud X that are fully covered by the union of balls of radius r emanating from X, a process we call *flooding*. Our construction allows for efficient PH computation, possesses several desirable theoretical properties, and is amenable to GPU parallelization. Scaling experiments on 3D point cloud data show that we can compute PH of up to dimension 2 on several millions of points. Importantly, when evaluating object classification performance on real-world and synthetic data, we provide evidence that this scaling capability is needed, especially if objects are geometrically or topologically complex, yielding performance superior to other PH-based methods and neural networks for point cloud data. Source code and datasets are available on : `https://github.com/plus-rkwitt/flooder`.

1 Introduction

Throughout the past years, topological data analysis (TDA) tools and, in particular, persistent homology (PH) [1, 21, 42, 54], have found widespread application in machine learning [39]. Applications range from graph classification [10, 28, 29], and time series forecasting [14, 51] to studying representations [2, 38, 48] and generalization of neural networks [7, 18], and developing novel regularizers or loss functions [13, 30]. The central pillar of most of these works is the power of PH to reveal and concisely summarize topological and geometrical information from a finite sample of points. In particular, within this pipeline, one first constructs a *simplicial complex* together with a *filtration* that encodes the underlying topological structure across scales. Once built, one then computes a stable summary called a *persistence barcode* or *persistence diagram*.

Importantly, however, the unfavorable computational complexity of the algorithmic pipeline for PH computation has, so far, largely limited the scope of topological approaches to studying connectivity

[†] equal contribution

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

properties only. For this reason, efficient approaches to the computation of PH have been a goal for the field for some time, as emphasized in a recent position paper [39, Section 4].

When taking a closer look at how such simplicial complexes are typically built, we immediately identify potential scalability issues. In particular, when the input data is a point cloud, one often chooses the simplicial complex whose k-simplices are all subsets of cardinality $(k + 1)$. For example, the Vietoris-Rips and the Čech complex at filtration value r consist of all subsets of the point cloud with a diameter less than r and all subsets that can be enclosed in some ball of radius r, respectively. Hence, at high filtration values, their number of simplices becomes exponentially large, rendering memory consumption and persistent homology computation impractical for large point clouds. Simple mitigation strategies include computing the complex and homology only up to some degree, only up to some filtration value, or only on a *subsample* of the point cloud. More advanced methods, such as sparse approximations [44] and collapsing strategies [8], reduce the Vietoris-Rips complex to a smaller one with (approximately) equivalent topology. However, for (very) large point clouds, one may need to “sparsify” so aggressively that topological features and approximation guarantees no longer apply, while the resulting complex may *still* be too large for practical computation.

The natural way to avoid such scalability problems is to work with smaller complexes. Specifically, in low dimension (e.g., three as in our experiments), filtrations on the Delaunay triangulation of the point clouds are used, such as the Alpha complex [19], Delaunay-Čech complex [4] (both homotopy equivalent to Čech), or Delaunay-Rips [36] complex. However, in true large-scale settings, even the benign scaling of the Delaunay triangulation can be challenging, not in terms of memory consumption, but in terms of subsequent PH computation time of these filtered complexes, as the latter comes at worst-case cubic complexity (in the complex size) using the standard matrix reduction algorithm [21].

To further reduce the size of the complex, one can resort to *subsampling* strategies. In subsampling [9, 11], the key idea is to repeatedly draw small subsets of the full point cloud, execute the PH pipeline, and then aggregate the resulting persistence diagrams [9], or an appropriately vectorized representation of the latter [11]. Both approaches come with theoretical guarantees but, depending on the data, may require sufficiently large subsets, a large number of random draws, and a computationally expensive aggregation step (as in [9]). Crucially, once the subsamples from the point cloud are chosen, the complex construction is agnostic of the original data. Because of this, topological features that are smaller than the distance between samples are irreversibly lost.

The *Witness complex* construction [16], which is most similar to our approach, builds a simplicial complex in a data-driven manner on top of a reduced *landmark set*. In particular, simplices appear in the complex if they are witnessed, i.e., if there exists a single non-landmark point that satisfies a distance condition to the simplex. Similar to subsampling approaches, small topological features can be lost. Moreover, the construction of the Witness complex can be fragile, as it entails carefully controlling a distance cut-off to avoid truncating genuine topological features while taming the running time, which, in many cases, renders this type of construction impractical on large point clouds.

Contribution(s). We seek to mitigate the discussed scalability issues via the *Flood complex*, i.e., a new filtered simplicial complex for Euclidean point cloud data. Being built on top of a small subsample of the point cloud at hand, its size is orders of magnitude smaller not only than the Vietoris-Rips complex but also the Alpha complex, facilitating efficient PH computation. Moreover, as its simplices are endowed with filtration values that are tied to the *entire* point cloud, the topological approximation quality of the Flood complex is considerably better compared to subsampling strategies. The computation of the Flood complex is designed for execution on GPUs, enabling efficient exploitation of specialized hardware and software libraries frequently used in contemporary machine learning research. We provide (1) initial theoretical guarantees on the approximation quality of PH computed on the Flood complex, stability results, and guarantees for approximation steps. Further, we (2) demonstrate scalability to point cloud sizes that were previously impossible to process, and eventually (3) present strong empirical evidence that this scalability is needed and useful when seeking to classify geometrically complex 3D point cloud data.

2 Preliminaries

We study finite point sets X in Euclidean space $\mathbb{R}^d$ in terms of the topology of the union of balls $X_r := \bigcup_{x \in X} B_r(x)$ at varying radii $r \geq 0$, with $B_r(x) := \{y \in \mathbb{R}^d : d(x, y) \leq r\}$ and metric $d(\cdot, \cdot)$. The topology of X_r and, in particular, its homology, can be computed from a combinatorial object called a simplicial complex. An (abstract) simplicial complex Σ over a set S is a collection of non-empty, finite subsets $\sigma \subset S$ such that for every non-empty subset $\tau \subset \sigma$ also $\tau \in \Sigma$. Moreover, σ is called a k-simplex if it has cardinality $k+1$, and a simplex $\tau \subset \sigma$ is called a *face* of σ.

As we study X_r for different $r \geq 0$, we need a simplicial complex Σ_r at each radius r. Analogously to $t \leq r$ implying $X_t \subset X_r$, we want $\Sigma_t \subset \Sigma_r$. A sequence of simplicial complexes $\{\Sigma_t : t \geq 0\}$ satisfying this property is called a *filtration* or *filtered simplicial complex*. In particular, a function $f : \Sigma_\infty \to \mathbb{R}$ that fulfills certain consistency properties induces a filtration via $\Sigma_t = f^{-1}((-\infty, t])$.

Informally, when increasing r, the shape of X_r will gradually change from the points $X_0 = X$ themselves to $X_\infty = \mathbb{R}^d$. During this process, we can study X_r by means of the homology groups H_k of an associated simplicial complex Σ_r and their evolution over r. Specifically, H_0 encodes connectivity information, H_1 information about loops, H_2 information about 3-dimensional voids (and subsequent homology groups encode higher dimensional topological information). As r grows, these homological features appear and disappear, and we summarize this information in the form of a *persistence diagram* $\mathrm{dgm}_k(\Sigma)$, i.e., a multiset of tuples $(b, d) \in \mathbb{R}^2$, where b denotes the minimal value r such that a feature exists in $H_k(\Sigma_r)$ and d denotes the maximal such value. Informally, the feature is born at time b and dies at time d. Two persistence diagrams D_1 and D_2 can be compared via their *bottleneck distance* $d_B(D_1, D_2) := \inf_{\eta : D_1 \to D_2} \sup_{p \in D_1} \|p - \eta(p)\|_\infty$ where the infimum is taken over all bijections $\eta : D_1 \to D_2$, considering each diagonal point (b, b) with infinite multiplicity. Importantly, if for two filtered complexes Σ_r and Σ'_r, there exists $\epsilon > 0$ such that $\Sigma_{r-\epsilon} \subset \Sigma'_r \subset \Sigma_{r+\epsilon}$ for all $r \geq 0$, then $d_B(\mathrm{dgm}(\Sigma), \mathrm{dgm}(\Sigma')) \leq \epsilon$, see [15].

A particularly relevant family of simplicial complexes over point clouds X are Alpha complexes [19] $\mathrm{Alpha}_r(X)$, which form a filtration of the Delaunay complex $\mathrm{Del}(X)$ on X. They are naturally embedded in $\mathbb{R}^d$ with each simplex σ represented by its convex hull $\mathrm{conv}(\sigma) := \{\sum_{x \in \sigma} \lambda_x x : \sum_{x \in \sigma} \lambda_x = 1, \lambda_x \geq 0\}$. Denoting by $|\mathrm{Alpha}_r| := \bigcup_{\sigma \in \mathrm{Alpha}_r} \mathrm{conv}(\sigma) \subset \mathbb{R}^d$, we have that $|\mathrm{Alpha}_r| \subset X_r$ and that both are homotopy equivalent [19]. This relates back to the very first point, i.e., it guarantees that X_r can be studied in terms of $\mathrm{Alpha}_r(X)$.

3 The Flood complex

As discussed, given a finite point cloud $X \subset \mathbb{R}^d$, ideally we would want to compute the PH of its (filtered) Alpha complex. The latter can be computed from a filtration of the Delaunay complex $\mathrm{Del}(X)$ which has, for $|X| = n$, at most $O\left(n^{\lfloor d/2 \rfloor}\right)$ simplices (and thus much fewer than the $2^n - 1$ simplices of the Vietoris-Rips complex). However, the computation of PH will still be challenging if n becomes very large. While, of course, one could compute the Alpha complex on only a subset $L \subset X$, one may lose valuable information when doing so. We therefore propose a filtration on $\mathrm{Del}(L)$ that is *aware* of the entire point cloud X.

Informally, at a given filtration value $r \geq 0$, our novel complex contains all simplices $\mathrm{Del}(L)$ that are fully covered by balls of radius r emanating from X, a process we call *flooding*. We refer to the resulting complexes $\mathrm{Flood}_r(X, L) \subset \mathrm{Del}(L)$ as Flood complexes and, in alignment with the terminology of witness complexes, we will refer to L as landmarks. We often select L as a subset of X, but doing so is not necessary, and the theoretical results in Section 3.2 do not require this assumption.

Definition 1. *For $r \geq 0$, the Flood complex* $\mathrm{Flood}_r(X, L)$ *at flood radius r is the simplicial complex*

$$\mathrm{Flood}_r(X, L) = \left\{\sigma \in \mathrm{Del}(L) \colon \mathrm{conv}(\sigma) \subset \bigcup_{x \in X} B_r(x)\right\} . \tag{1}$$

Whenever $0 \leq r \leq t$, we have $\mathrm{Flood}_r(X, L) \subset \mathrm{Flood}_t(X, L)$ from $X_r \subset X_t$, which yields a genuine filtration, i.e., a nested family $\{\mathrm{Flood}_r(X, L)\}_{r \in [0,\infty)}$ to which we refer to as the *filtered Flood complex*. By construction, if $L \subset X$, then all 0-simplices are already in the complex at time $r = 0$ and vice versa. For brevity, we will refer to the entire nested family of Flood complexes as $\mathrm{Flood}(X, L)$.

3.1 Intuition

Figure 1 illustrates a point cloud X, a subset $L \subset X$, and the construction of the filtered complexes $\mathrm{Alpha}(L)$ and $\mathrm{Flood}(X, L)$. We argue that $\mathrm{Alpha}_r(L)$, which is homotopy equivalent to the union of balls of radius r centered on L, can be a poor representation of the topology of X_r, while the filtration function of the Flood complex makes it more aligned to the topology of the underlying data.

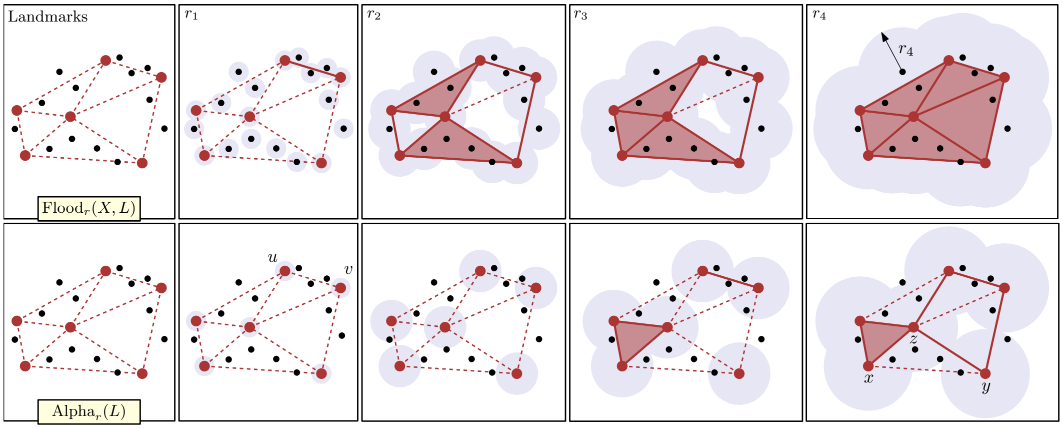


Figure 1: Schematic overview of the Flood complex $\mathrm{Flood}_r(X, L)$ (**top**), the Alpha complex on a subsample $\mathrm{Alpha}_r(L)$ (**bottom**), and their accordance with the union of balls X_r at different radii r. The point cloud X is marked by ●, the landmarks $\subset X$ by ●, and ● identify the balls of radius r.

For example, the points $u, v \in X$ are in the same connected component of X_{r_1}, as illustrated in the second column of Figure 1. This is faithfully encoded in $\mathrm{Flood}_{r_1}(X, L)$ which already contains the edge $\{u, v\}$, because $\mathrm{conv}(\{u, v\}) \subset X_{r_1}$. However, in the Alpha complex, u and v will be in the same connected component much later, i.e., only after r_3. Similarly, at radius r_2, there are two cycles in X_{r_2}, which are recognized by $\mathrm{Flood}_{r_2}(X, L)$ but not by $\mathrm{Alpha}_{r_2}(L)$. Specifically, the rightmost cycle in X_{r_2} is less persistent in the Alpha filtration, as it is born only much later in $\mathrm{Alpha}(L)$ at filtration value r_4 and dies shortly afterwards. For the leftmost cycle this effect is even more pronounced. Similar considerations apply to higher order simplices, such as, for example, the triangle $\{x, y, z\}$, which is missing from $\mathrm{Alpha}(L)$ even at filtration value r_4.

It is worth mentioning that since $\mathrm{Flood}(X, L)$ is defined by a filtration on the Delaunay triangulation only on L instead of X, there are natural limitations to its resolution. Most notably, every k-simplex in $\mathrm{Del}(L)$ can detect at most one k-dimensional hole. Still, if the number of landmarks is sufficiently large and they are well-placed, then $\mathrm{Flood}_r(X, L)$ will be able to capture the relevant homological information of X_r, as shown in the next subsection.

3.2 Theoretical results

The Flood complex satisfies certain beneficial properties (proofs can be found in Appendix C). First, it is stable with respect to the point cloud X. This means that if the point cloud X is perturbed by a little, then the PH of $\mathrm{Flood}(X, L)$ changes only a bit, quantified as follows.

Theorem 2. *The Flood complex is bottleneck stable with respect to its first argument, i.e., given $L, X, X' \subset \mathbb{R}^d$, it holds that $\forall i \in \mathbb{N}$*

$$d_B\left(\mathrm{dgm}_i(\mathrm{Flood}(X, L)), \mathrm{dgm}_i(\mathrm{Flood}(X', L))\right) \leq d_H(X, X') \ . \tag{2}$$

A second desirable property satisfied by the Flood complex is that it recovers the topology of the union of balls X_r when the landmarks L are chosen as the whole point cloud X.

Theorem 3. *Let $X \subset \mathbb{R}^2$ be a finite subset of points in general position. Then, $|\mathrm{Flood}_r(X, X)|$ is homotopy equivalent to X_r for any $r \geq 0$.*

While, at the moment, our proof only covers the case $X \subset \mathbb{R}^2$, we conjecture that similar arguments work for higher dimensions, and we leave the proof for future work. Nevertheless, because homotopy equivalent spaces have the same homology, Theorem 3 directly implies that the persistent homology of $\mathrm{Flood}(X, X)$ equals that of $\mathrm{Alpha}(X)$. Moreover, in combination with Theorem 2 and the classic Hausdorff stability of Čech and Alpha complexes [15], we get the following approximation guarantee.

Corollary 4. *Let $L, X \subset \mathbb{R}^2$ be finite subsets of points in general position. Then, it holds that $\forall i \in \mathbb{N}$*

$$d_B\left(\mathrm{dgm}_i(\mathrm{Alpha}(X)), \mathrm{dgm}_i(\mathrm{Flood}(X, L))\right) \leq 2 d_H(X, L) \ . \tag{3}$$

Essentially, if we want $d_B(\mathrm{dgm}_i(\mathrm{Flood}(X, L)), \mathrm{dgm}_i(\mathrm{Alpha}(X))) \leq 2\epsilon$, it suffices to select L such that $d_H(X, L) \leq \epsilon$. In particular, if $L \subset X$, finding the optimal landmark positions is just the metric

k-center problem, and greedy selection strategies, such as farthest point sampling (FPS), achieve a factor 2-approximation of the optimal covering radius [26]. Thus, $d_H(X, L) \leq \epsilon$ is achieved once $|L| \geq O(\epsilon^{-D})$ landmarks are selected, where D is the intrinsic dimension of X. Similar guarantees hold with high probability if X is a random sample and its first $|L|$ points are used as landmarks [34].

Moreover, Corollary 4 implies that the persistence diagrams of the Flood complex are stable with respect to the landmarks L, albeit with a multiplicative constant of 4. Still, in case L is perturbed only slightly such that its Delaunay triangulation is preserved, then there is a direct proof that $d_B\left(\mathrm{dgm}_i(\mathrm{Flood}(X, L)), \mathrm{dgm}_i(\mathrm{Flood}(X, L'))\right) \leq d_H(L, L')$.

4 Computational aspects

We next discuss the computational aspects of constructing the Flood complex. Note that the set of simplices, i.e., a Delaunay triangulation of the landmark set L, can be computed in $O(|L|^{\lfloor d/2 \rfloor})$ [12], and efficient implementations can be found in libraries such as `CGAL` [45] or `qhull` [3].

4.1 Approximation

The definition of the Flood complex entails calculating the filtration value $f(\sigma)$ of each simplex $\sigma \in \mathrm{Del}(X)$. This filtration value $f(\sigma) = \max_{p \in \mathrm{conv}(\sigma)} \min_{x \in X} d(p, x)$ is the minimal radius r such that $\mathrm{conv}(\sigma) \subset X_r$, or equivalently, the directed Hausdorff distance between $\mathrm{conv}(\sigma)$ and X. Since computing the directed Hausdorff distance is a nonconvex problem, finding $f(\sigma)$ would be inefficient. Hence, in our implementation, we replace each convex hull $\mathrm{conv}(\sigma)$ by a finite subset $P_\sigma \subset \mathrm{conv}(\sigma)$, resulting in the simplicial complex

$$\mathrm{Flood}_r(X, L, P) = \{\sigma \in \mathrm{Del}(L) \colon P_\tau \subset X_r \; \forall \tau \subset \sigma\} \quad . \tag{4}$$

Specifically, $\sigma \in \mathrm{Flood}_r(X, L, P)$ iff P_σ is flooded and all its faces $\tau \subset \sigma$ are in $\mathrm{Flood}_r(X, L, P)$. By construction, $\mathrm{Flood}_r(X, L) \subset \mathrm{Flood}_r(X, L, P)$, because $\mathrm{conv}(\sigma)$ being flooded implies that P_σ is flooded. Moreover, if the (directed) Hausdorff distance between each pair P_σ and $\mathrm{conv}(\sigma)$ satisfies $d_H(P, \mathrm{conv}(\sigma)) < \epsilon$, then $\mathrm{Flood}_r(X, L, P) \subset \mathrm{Flood}_{r+\epsilon}(X, L)$. We get the following guarantee.

Theorem 5. *Let $X, L \in \mathbb{R}^d$ and let $P = \{P_\sigma \colon \sigma \in \mathrm{Del}(L)\}$ with $P_\sigma \subset \mathrm{conv}(\sigma)$ for all $\sigma \in \mathrm{Del}(L)$. Then, it holds that $\forall i \in \mathbb{N}$*

$$d_B\left(\mathrm{dgm}_i(\mathrm{Flood}(X, L)), \mathrm{dgm}_i(\mathrm{Flood}(X, L, P))\right) \leq \max_{\sigma \in \mathrm{Del}(L)} d_H(P_\sigma, \mathrm{conv}(\sigma)) \quad . \tag{5}$$

We can select the sets P_σ in various ways, e.g., by explicitly enforcing a small Hausdorff distance to $\mathrm{conv}(\sigma)$, taking a random sample, or based on an evenly spaced grid of barycentric coordinates.

Lemma 6. *Given $m \in \mathbb{N}$ and a k-simplex $\sigma = \{v_0, \ldots, v_k\}$, let $P_\sigma = \{p = \sum_{i=0}^k \lambda_i v_i \colon \sum_{i=0}^k \lambda_i = 1, m\lambda_i \in \mathbb{N}\} \subset \mathrm{conv}(\sigma)$ be a grid of simplex points induced by evenly spaced barycentric coordinates. Then, it holds that*

$$d_H(P_\sigma, \mathrm{conv}(\sigma)) \leq \frac{1}{m} \sqrt{\sum_{i<j} \|v_i - v_j\|^2} \quad .$$

Notably, the bound decays with $1/m$, where $m + 1$ is the number of grid points on each edge. The shape of the simplices enters in form of the square root term, which scales with $O(\mathrm{diam}(\sigma))$. A similar scaling behavior can be observed when P_σ is selected randomly, see Lemma 9.

Thanks to the finite set P_σ, the optimization problem can be solved directly by iterating over the points in P_σ and X. However, the number of points in P_σ required for a good approximation is large, so this approach does not scale to large point clouds if implemented naively. As the problem entails finding the nearest neighbor in X for each point in P_σ, it can be efficiently solved using a data structure such as a k-d tree [5] or a cover tree [6]. Building such a data structure can, however, incur significant computational overhead. To avoid this issue, we compute the directed Hausdorff distance using a custom GPU algorithm, paired with a principled *masking* procedure, which is presented below.

4.2 Masking and GPU acceleration for the flooding process

Since the filtration value $\max_{p \in P_\sigma} \min_{x \in X} d(p, x)$ of the simplex σ depends only on a single pair of points (p, x), we want to ignore points $x \in X$ that are far from σ. In fact, if $L \subset X$, then we can precompute for each simplex σ a subset of X that must be considered.

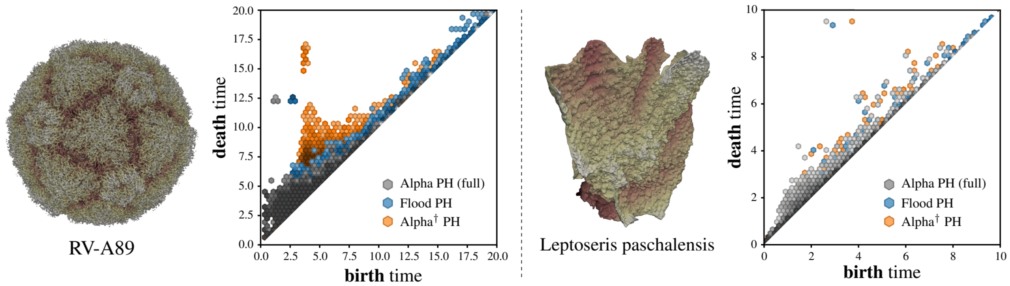


Figure 2: Exemplary hexbin plots of persistence diagrams of RV-A89 (**left**) and the Leptoseris paschalensis coral (**right**). Gray corresponds to Alpha PH of the full point cloud, blue to Flood PH with 10k landmarks and orange to Alpha† PH with 75k points. Point clouds are visualized via small spheres colored by distance to their bounding box center (for RV-A89) or by elevation (for the Leptoseris paschalensis coral). Best viewed in color.

Lemma 7. *Let $X \subset \mathbb{R}^d$ be a finite set in general position, and let $\sigma \subset X$ be a set of $k \leq d+1$ points. If $c \in \mathbb{R}^d$ and $r > 0$ are such that $\sigma \subset B_r(c)$, i.e., such that $B_r(c)$ is an enclosing ball of σ, then*

$$f(\sigma) = \max_{p \in \mathrm{conv}(\sigma)} \min\{d(p,x) : x \in X\} = \max_{p \in \mathrm{conv}(\sigma)} \min\left\{d(p,x) : x \in X \cap B_{\sqrt{2}r}(c)\right\} \quad . \tag{6}$$

In practice, we compute the enclosing balls $B_r(c)$ using a variation of Ritter's heuristic [41], i.e., we set c as the center of the longest edge of σ and use radius $\sqrt{2}\max_{i=1}^{k} d(c, v_i)$. This can be done in $O(d^2)$ time and is highly parallelizable on GPUs. In case σ is itself an edge, it suffices to set the radius to half its diameter, i.e., without the factor $\sqrt{2}$.

Once centers and radii have been computed for all simplices $\sigma \in \mathrm{Del}(L)$, checking whether the points in X belong to $B_{\sqrt{2}r}(c)$ can be done with $|\mathrm{Del}(L)| \cdot |X|$ distance evaluations, yielding a Boolean *masking* matrix of shape $|\mathrm{Del}(L)| \times |X|$. Since the mask can be computed independently for each (simplex, point) pair, computation of the masking matrix can be efficiently parallelized. Eventually, the masking matrix together with the points in X and the sets P_σ are fed to a custom Triton [47] kernel, which computes the directed Hausdorff distance; for specifics of this kernel, see Appendix A.1.

5 Experiments

We validate the applicability and relevance of the Flood complex as follows: in Section 5.1, we use it to compute PH on point clouds for which existing approaches require an impractical amount of computational resources; in Section 5.2, we study the scalability of the Flood complex, and, in Section 5.3, we show that PH computed on the Flood complex (in short, Flood PH) improves predictions in downstream machine learning tasks compared to simpler approaches such as subsampling.

In our experiments, we often compare the Flood complex and its PH to the Alpha complex on the entire point cloud and to the Alpha complex on a subset of size selected so that runtime is similar to Flood PH at the given number of landmarks and discretization level of simplices. Specifically, we discretize to 30 points per edge in Sections 5.1 and 5.2 and 20 points per edge in Section 5.3. We denote the former by Alpha PH (full) and the latter by Alpha† PH. For PH computation and the construction of Alpha complexes, we use `Gudhi` [46].

5.1 Persistent homology on large-scale point clouds

First, we showcase the approximation capabilities of the Flood complex on two exemplary real-world point clouds from different scientific disciplines, one (12M points) based on a cryo-electron microscopy of a rhinovirus (RV-A89) from [49], the other (10M points) based on a 3D-scan of a Leptoseris paschalensis coral from the *Smithsonian 3D Digitization* initiative. Additional information about the point clouds can be found in Appendix A.4. We compute zero-, one-, and two-dimensional PH using the Alpha complex, which can be thought of as the *ground truth*. Notably, this requires more than 15 minutes of runtime per point cloud and more than 90 GB of RAM. We then compare its PH to that of the Flood complex and to that of the Alpha complex computed on a subsample.

A visual inspection of the persistence diagrams, see Figure 2, shows that Flood PH mitigates the shift of (birth, death)-tuples along the diagonal (that is characteristic for subsampling methods), resulting

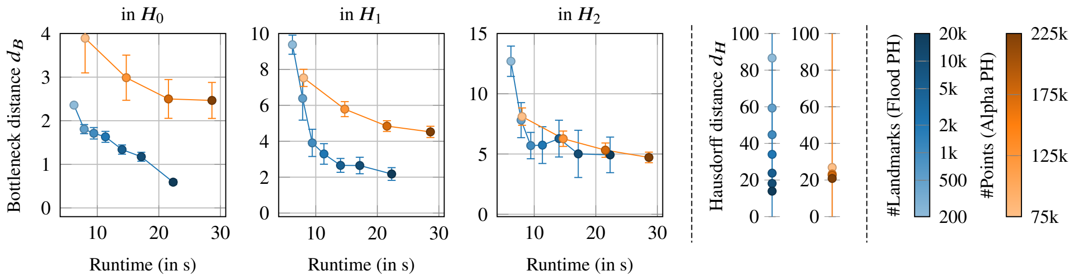


Figure 3: *Approximation quality* of Flood PH and Alpha PH on RV-A89. The **(left)** panel shows bottleneck distances to Alpha PH (full) in H_0, H_1 and H_2 when varying the number of landmarks for Flood PH and the subsample size for Alpha PH. The (**middle**) panel shows the Hausdorff distance between the full point cloud and the landmarks, resp., the points in the subsample. The (**right**) panel shows the color coding used in all the plots.

in a better approximation of birth and death times compared to Alpha PH. Moreover, the persistent H_1 features of the RV-A89 virus become scattered for Alpha† PH but remain close together for Flood PH.

For a quantitative comparison between Flood PH and Alpha PH on subsamples, we consider bottleneck distances to Alpha PH (full) on the RV-A89 point cloud. Results are presented in Figure 3; for results on the Leptoseris coral, we refer to Appendix B.2. As expected, the approximation quality of Flood PH improves with the number of landmarks. Notably, the randomness in the landmarks, caused by different starting points for FPS, mainly affects H_2. Beyond 500 landmarks, bottleneck distances are significantly smaller in dimensions 0 and 1, and similar in dimension 2, when comparing Flood PH to Alpha PH at the same runtime budget. A comparison between landmark selection using farthest point and uniform random sampling is reported in Appendix B.3.

5.2 Runtime and scalability

Table 1: Relative runtime breakdown (in %) of Flood PH on *one* point cloud from two datasets.

	`swisscheese`	RV-A89	
#Points $\lvert X\rvert$	1M	12M	12M
#Landmarks $\lvert L\rvert$	2k	2k	10k
Landmark select.	13.3	15.8	18.1
Delaunay triang.	6.0	0.8	2.9
Masking	3.7	3.2	7.7
Filtration	69.5	79.3	68.2
PH computation	3.0	0.2	1.0
Other	4.4	0.6	2.1

Breakdown of runtimes. In Table 1, we report the runtime share of different parts of Flood PH. Specifically, we consider one point cloud from the `swisscheese` dataset (1M points) and one from the RV-A89 (12M points), see Section 5.1. We compute filtration values for each simplex from a grid with 30 points per edge and use FPS for landmark selection (2k and 10k). Although the (relative) runtimes inevitably depend on the configuration of the Flood complex and the shape and size of the point clouds, several trends can be observed: the majority of time is spent computing filtration values, followed by landmark selection and masking; the runtime of the remaining parts, i.e., triangulating the landmarks, PH computation and other overhead (e.g., computing grid points and enclosing balls of simplices) mainly depends on the number of landmarks. On RV-A89 and $|L| = 2\text{k}$, this overhead is negligible.

Scalability. We assess the scaling behavior of the Flood complex on point clouds constructed in the same manner as the `swisscheese` dataset. Specifically, we study its scaling wrt. (a) the number $|X|$ of points in the point cloud, (b) the number $|L|$ of landmarks, and (c) the ambient dimension. We always compare to Alpha PH computed from subsamples of the same size as $|X|$. As can be seen in Figure 4 (a), both methods exhibit an increase in runtime with $|X|$. However, while for $|X| < 10^4$, the Flood PH runtime is similar to Alpha PH (likely due to the overhead of landmark selection), for larger point clouds, Flood PH requires consistently one to two orders of magnitude less time than Alpha PH. Figure 4 (b) presents the runtime of Flood PH as a function of $|L|$ (with $|X| = 1\text{M}$ fixed). In general, we observe a linear scaling behavior that is aligned with the typically linear growth [22] in the number of simplices in the Delaunay triangulation of L. Finally, Figure 4 (c) shows the impact of the ambient dimensionality $d \in \{2, 3, 4, 5\}$ when keeping $|X| = 1\text{M}$ and $|L| = 1\text{k}$ fixed. Here, runtimes increase with d for both methods, as the number of simplices in the Delaunay triangulations grows. Notably, Flood PH consistently remains at least one order of magnitude faster than Alpha PH.

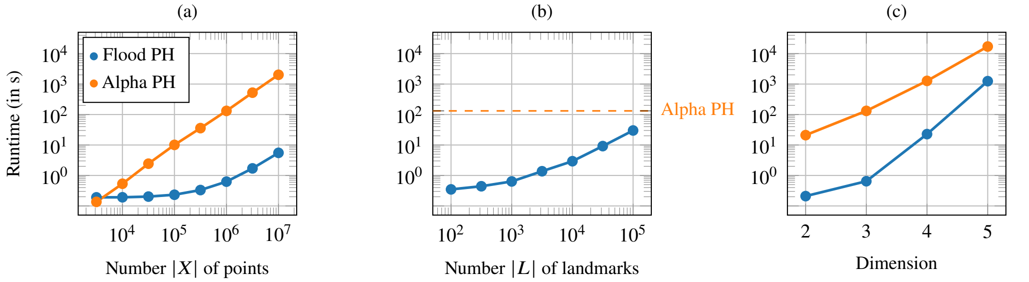


Figure 4: Runtime (in s) of Flood PH and Alpha PH for `swisscheese`-like point clouds: (a) in $\mathbb{R}^3$, varying the point cloud size $|X|$ with $|L| = 1$k landmarks; (b) in $\mathbb{R}^3$, varying the number of landmarks $|L|$ with $|X| = 1$M points; (c) varying the dimensionality with $|X| = 1$M and $|L| = 1$k.

5.3 Object classification

We run classification experiments on five datasets, including three real-world datasets, i.e., `modelnet10`, `mcb-c` and the self-curated `corals`, which come in the form of surface meshes with a varying number of vertices and varying object "complexity". In terms of the latter, the number of vertices and triangles is a good proxy (especially for the CAD-based models in `mcb-c`), as, e.g., representing a helical geared motor requires a finer mesh than representing a taper pin. To create point clouds, we uniformly sample $|X|$ points from the mesh surfaces (as, e.g., done in [17]), which mitigates artifacts from irregularly spaced vertices. We also create two topologically challenging 3D synthetic datasets, i.e., `swisscheese` and `rocks`. Dataset details are provided in Appendix A.4. We use *ten* random 80/20% training/testing splits, with 10% of the training data reserved for validation.

Parametrization of the Flood complex & classifier. Unless otherwise stated, we select $|L| = 2$k landmarks from $|X| = 1$M points using FPS. To compute filtration values, we discretize each simplex based on an equally spaced grid of barycentric coordinates with 20 points per edge (resulting in 210 points per triangle and 1540 points per tetrahedron); cf. Section 4.1. We vectorize persistence diagrams (H_0, H_1 and H_2) using [27] with (exponential) structure elements, parametrized as follows: *locations* are set to 64 k-means++ centers, computed from the (birth, death) tuples of all diagrams in the training data, *scales* are chosen as in ATOL [43, Eq. (2)], and the vectorization's *stretch* parameter is set to either the one, five, or ten percent lifetime quantile (based on the validation data); this yields 64-dim. vectorizations per diagram which, upon concatenation, are fed as 192-dim. feature vectors to an LGBM [31] classifier (except for `corals`, where we use ℓ_1-regularized logistic regression). Hyperparameters are tuned on the validation data using FLAML [50] and a time budget of 10 minutes.

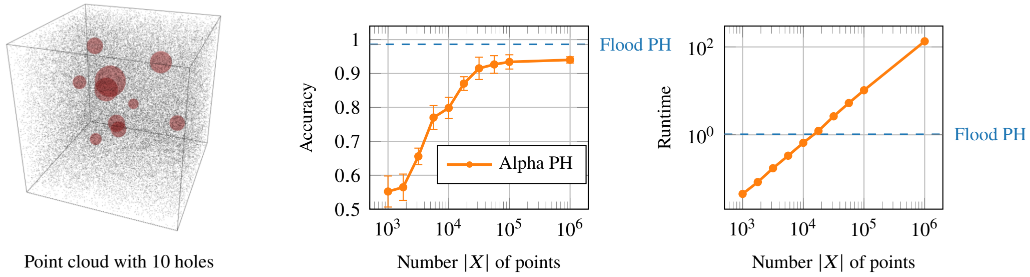
Baselines. Our primary TDA baseline is Alpha† PH, i.e., persistent homology computed from a subsample of size chosen such that the runtime is similar to Flood PH, i.e. 1% of the `rocks` point clouds and otherwise 20k points . Persistence diagrams are processed in the same manner as described above. Importantly, we do *not* account for the vectorization time when choosing the subsample size for Alpha† PH, although persistence diagrams obtained from Alpha complexes can become very large and require substantially more time for vectorization than those obtained from Flood PH. We also compare to *averaged* persistence diagram vectorizations [11], denoted as Alpha PH (avg.), obtained from Alpha PH of *five* point cloud subsamples of accordingly reduced size. Unfortunately, methods such as Witness complexes [16], sparse Rips filtrations [44], or adaptive approximations [25], which might at first glance seem natural competitors to the Flood complex, proved to be infeasible on large-scale point clouds. For example, computing a lazy Witness complex with 100 landmarks and 1M witnesses required more than 20 minutes on our hardware (using the `Gudhi` [46] implementation).

Moreover, we compare to three neural network methods, designed for point cloud data: PointNet++ [40], PointMLP [35], and PVT [52] (with network width and depth as in the reference implementations). We train all models on point clouds subsampled to 2k points by minimizing cross-entropy (or MSE for the regression task on `rocks`) over 200 epochs using Adam [33] with a cosine annealing schedule and batch size 64. We select the initial learning rate, weight decay, and early stopping period based on the validation data. On the real-world datasets, we use random scaling and shifts as data augmentation.

Evaluation metrics. For classification tasks, we report the mean balanced accuracy over training/testing splits ± 1 standard deviation on all datasets (to account for class imbalances). For the regression task on `rocks`, we report the mean of mean squared errors (MSE) over splits ± 1 standard deviation.

Table 2: Classification results on **synthetic** data. *Runtime* (in s) is given per point cloud (averaged over all point clouds in a dataset); *Acc* denotes balanced accuracy (averaged over all ten splits). We do not list runtimes for neural networks (bottom part), as they are not directly comparable. Further, there is only one *Runtime* column for `rocks`, as the classification and regression task use the same persistence diagrams. 🕒 indicates that results cannot be computed within a 48-hour time budget (runtime for `rocks` is estimated).

	`swisscheese` (2)		`rocks` (2)	`rocks` (reg)	
	Acc ↑	Runtime ↓	Acc ↑	MSE ↓	Runtime ↓
Flood PH	**0.98** ± 0.01	1.1 ± 0.1	**0.88** ± 0.03	(**0.5** ± 0.2) × 1e-3	7.0 ± 2.9
Alpha† PH	0.85 ± 0.04	1.8 ± 0.0	0.78 ± 0.03	(1.5 ± 0.3) × 1e-3	9.1 ± 4.2
Alpha† PH (avg.)	0.80 ± 0.02	0.9 ± 0.0	0.76 ± 0.04	(20.9 ± 1.3) × 1e-3	6.9 ± 3.1
Alpha PH (full)	0.94 ± 0.02	134.3 ± 15.8	🕒	🕒	1490 ± 680
PointNet++ [40]	0.49 ± 0.03	-	0.54 ± 0.01	(18.6 ± 2.8) × 1e-3	-
PointMLP [35]	0.51 ± 0.02	-	0.51 ± 0.03	(8.8 ± 1.0) × 1e-3	-
PVT [52]	0.50 ± 0.02	-	0.51 ± 0.04	(31.4 ± 3.4) × 1e-3	-



Point cloud with 10 holes

Figure 5: Comparison of classification of accuracy (on `swisscheese`) and runtime (in s) between Flood PH (2k landmarks) and Alpha PH when the latter has access to an increasing number of points in X. The leftmost panel shows an example of a `swisscheese` point cloud with 10 holes.

5.3.1 Synthetic data

Swiss cheese. We create a synthetic dataset of 3D point clouds by uniformly sampling 1M points in $[0, 5]^3$ and removing k disjoint balls with random radii (in $[0.1, 0.5]$) and centers. We set $k \in \{10, 20\}$ and seek to distinguish point clouds by their number of voids (i.e., a *binary* problem). An example (with $k = 10$ voids highlighted in red) is shown in Figure 5 (left). The motivation for creating this dataset is to demonstrate that any approach based solely on subsampling the data for computational reasons (either in the context of computing PH, or for training neural network models) will perform poorly, as the class-specific topological characteristics (i.e., the voids in H_2) will be difficult to identify.

Table 2 confirms that all neural network baselines fail on this problem. Moreover, the classification performance of Alpha† PH is worse than that of Flood PH, most likely because Alpha† PH provides less information. To examine the latter, we compare Flood PH with Alpha PH on increasingly larger subsampled point clouds in Figure 5. As expected, the classification accuracy increases with the subsample size. However, surprisingly, it never matches the accuracy of Flood PH, but reaches its maximum of 94% at 1M points, i.e., at the entire point cloud. We hypothesize that the remaining gap to Flood PH can be attributed to the very large size of the resulting persistence diagrams, making it difficult to represent the relevant information in the vectorization. Furthermore, we observe an approximate linear increase in runtime for Alpha PH with increasing point cloud size, resulting in more than 24 hours runtime for computing PH on the entire dataset when all 1M points are used.

Rocks. Extending the ideas underlying the `swisscheese` data, we generate a more difficult dataset that mimics *porous materials*. Its point clouds contain up to 16M points and are extracted from Boolean voxel grids of size 256^3, produced using two different generators (blobs and fractal noise) available as part of the `PoreSpy` library [23]. We evaluate classification accuracy wrt. the data generating method (i.e., a *binary* problem) and regression accuracy wrt. the surface area computed from the voxel grid. From Table 2, it is apparent that PH, and particularly Flood PH, excels on this dataset, achieving an average classification accuracy of 88%, whereas neural networks are only marginally better than random guessing. Similarly, on the regression task, Flood PH performs by far the best, with all neural networks yielding MSEs more than one magnitude larger than Flood PH.

Table 3: Classification results on **real world** data. *Runtime* (in s) is given per point cloud (averaged over all point clouds in a dataset); *Acc* denotes balanced accuracy (averaged over all ten splits). We do not list runtimes for neural networks (bottom part), as they are not directly comparable. 🕓 indicates that results cannot be computed within a 48-hour time budget.

	`corals` (2)		`mcb-c` (11)		`modelnet10` (10)	
	Acc ↑	Runtime ↓	Acc ↑	Runtime ↓	Acc ↑	Runtime ↓
Flood PH	**0.77** ± 0.09	1.8 ± 1.2	**0.74** ± 0.02	1.5 ± 0.5	0.72 ± 0.02	0.8 ± 0.2
Alpha† PH	0.58 ± 0.13	1.7 ± 0.3	0.66 ± 0.03	1.8 ± 0.2	0.64 ± 0.01	1.6 ± 0.1
Alpha† PH (avg.)	0.52 ± 0.08	1.4 ± 0.1	0.69 ± 0.03	1.4 ± 0.1	0.66 ± 0.02	1.3 ± 0.1
Alpha PH (full)	0.44 ± 0.07	153.4 ± 16.8	🕓	119.1 ± 7.6	🕓	107.3 ± 5.9
PointNet++ [40]	0.53 ± 0.10	-	**0.76** ± 0.02	-	**0.93** ± 0.01	-
PointMLP [35]	0.59 ± 0.15	-	**0.74** ± 0.03	-	**0.93** ± 0.01	-
PVT [52]	0.56 ± 0.11	-	**0.77** ± 0.03	-	**0.94** ± 0.01	-

5.3.2 Real-world data

ModelNet10. `modelnet10` is a subset of the larger ModelNet40 [53] corpus, containing geometrically rather *simple* objects with few characteristic topological features. Considering the latter, unsurprisingly, all neural network baselines perform much better (by more than 20 percentage points) than any purely PH-based approach, regardless of the simplicial complex. Nonetheless, we observe that all PH-based approaches yield accuracies notably higher than the ≈54% reported in [17] (on the same benchmark data) for a method that *predicts* vectorized persistence barcodes via a neural network.

Mechanical Components Benchmark (MCB). As a slightly more challenging dataset, we selected a (11-class) *subset* of meshes, dubbed `mcb-c`, from the publicly available MCB corpus [32], focusing on objects with more geometrically and topologically interesting features, see Appendix A.4. Comparing the neural network results of Table 3 with the results in [32, Table 5] (mostly >90% on the *full* dataset of 68 classes), we see that the increased object complexity (of our 11-class subset) manifests as a drop in classification accuracy. Notably, Flood PH is competitive with the neural networks and achieves a significantly better balanced accuracy than Alpha† PH and Alpha† PH (avg.)

Coral mesh dataset. Finally, we curated a challenging dataset of 3D point clouds by uniformly sampling 1M points from surface meshes of *corals* from the *Smithsonian 3D Digitization* initiative. Our results on the *binary* classification problem of distinguishing corals by genus show that Flood PH yields (by far) the best result, suggesting that capturing fine details in the surface structure is mandatory for extracting discriminative information. Similar to Figure 5, we observed that the large number of points in the persistence diagrams of Alpha PH variants (compared to the leaner diagrams produced by Flood PH) tends to confound the persistence diagram vectorization technique, leading to drops in downstream performance. Moreover, all neural network baselines achieve only a balanced accuracy of less than 60% with a high variance across splits.

6 Discussion

We introduced the *Flood complex*, a novel simplicial complex designed to address the long-standing computational challenges of PH on large-scale point clouds. By combining careful subsampling and GPU parallelism, the Flood complex enables efficient computation of accurate PH approximation on point clouds with millions of points within seconds, offering speed increases of up to two orders of magnitude over the widely used Alpha complex. The Flood complex opens up several research questions for future work, including strengthening its theoretical guarantees and developing even more efficient algorithms. Specifically, we anticipate that the theoretical guarantees can be extended to Euclidean spaces of arbitrary dimension, and that more direct proofs, yielding tighter bounds on approximation quality and stability with respect to landmarks, are possible. Moreover, exploring additional real-world applications that require computing PH on large point clouds would further highlight the value of the Flood complex as a lightweight tool, and we envision that the Flood complex will facilitate a more widespread use of PH in machine learning applications that have, so far, been limited by computational constraints.

Acknowledgments

This work was supported by the Federal State of Salzburg within the EXDIGIT project 20204-WISS/263/6-6022 and projects 0102-F1901166- KZP, 20204-WISS/225/197-2019, the Austrian Science Fund (FWF) under project 10.55776/DFH4791124 (REVELATION). M. Uray and S. Huber were supported by the Christian Doppler Research Association (JRC ISIA), the Austrian Federal Ministry for Digital and Economic Affairs and the Federal State of Salzburg.

The authors also thank the anonymous reviewers for the valuable feedback during the review process, and Karsten Borgwardt and the Max Planck Institute of Biochemistry for providing access to the computing infrastructure.

References

[1] S. Barannikov. The Framed Morse complex and its invariants. *Adv. Sov. Math.*, 21:93–116, 1994. 1

[2] S. Barannikov, I. Trofimov, N. Balabin, and E. Burnaev. Representation topology divergence: A method for comparing neural network representations. In *ICML*, 2022. 1

[3] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, December 1996. ISSN 0098-3500. 5

[4] U. Bauer and H. Edelsbrunner. The Morse theory of čech and delaunay complexes. *Trans. Am. Math. Soc.*, 369(5):3741–3762, 2017. 2

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. 5, 14

[6] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *ICML*, 2006. 5

[7] T. Birdal, A. Lou, L. Guibas, and U. Şimşekli. Intrinsic dimension, persistent homology and generalization in neural networks. In *NeurIPS*, 2021. 1

[8] J.-D. Boissonnat, S. Pritam, and D. Pareek. Strong collapse and persistent homology. *J. Topol. Anal.*, 15(01):185–213, 2023. 2

[9] Y. Cao and A. Monod. Approximating persistent homology for large datasets, 2022. URL `https://arxiv.org/abs/2204.09155`. arXiv. 2

[10] M. Carriere, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *AISTATS*, 2020. 1

[11] F. Chazal, B.T. Fasy, F. Lecci, B. Michel, A. Rinaldo, and L. Wasserman. Subsampling methods for persistent homology. In *ICML*, 2015. 2, 8

[12] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10(4):377–409, 1993. 5

[13] C. Chen, Ni. X., Q. Bai, and Y. Wang. A topological regularizer for classifiers via persistent homology. In *AISTATS*, 2019. 1

[14] Y. Chen, I. Segovia-Dominguez, B. Coskunuzer, and Y. Gel. TAMP-s2GCNets: Coupling time-aware multipersistence knowledge representation with spatio-supra graph convolutional networks for time-series forecasting. In *ICLR*, 2022. 1

[15] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, 2007. ISSN 1432-0444. 3, 4, 19, 20

[16] V. de Silva and G. Carlsson. Topological estimation using witness complexes. In *Eurographics Symposium on Point-Based Graphics*, 2004. 2, 8

[17] T. de Surrel, F. Hensel, M. Carrière, T. Lacombe, Y. Ike, H. Kurihara, M. Glisse, and F. Chazal. RipsNet: a general architecture for fast and robust estimation of the persistent homology of point clouds. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. 8, 10

[18] N. Dupuis, G. Deligiannidis, and U. Şimşekli. Generalization bounds with data-dependent fractal dimensions. In *ICML*, 2023. 1

[19] H. Edelsbrunner. The union of balls and its dual shape. In *SCG*, 1993. 2, 3, 24

[20] H. Edelsbrunner and J. Harer. *Computational Topology - an Introduction.* American Mathematical Society, 2010. 22

[21] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2000. 1, 2

[22] J. Erickson. Nice point sets can have nasty Delaunay triangulations. *Discrete Comput. Geom.*, 30(1):109–132, May 2003. ISSN 1432-0444. 7

[23] J.T. Gostick, Z.A. Khan, T.G. Tranter, M.D. Kok, M. Agnaou, M. Sadeghi, and R. Jervis. Porespy: A python toolkit for quantitative analysis of porous media images. *J. Open Source Softw.*, 4(37):1296, 2019. 9, 16

[24] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. 21

[25] M. Herick, M. Joachim, and J. Vahrenhold. Adaptive approximation of persistent homology. *J. Appl. Comput. Topol.*, 8(8):2327–2366, Dec 2024. ISSN 2367-1734. 8

[26] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. ISSN 0364765X, 15265471. 5

[27] C. Hofer, R. Kwitt, and M. Niethammer. Learning representations of persistence barcodes. *JMLR*, 20(126):1–45, 2019. 8

[28] C. Hofer, F. Graf, B. Rieck, M. Niethammer, and R. Kwitt. Graph filtration learning. In *ICML*, 2020. 1

[29] M. Horn, E. De Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. Borgwardt. Topological graph neural networks. In *ICLR*, 2022. 1

[30] X. Hu, F. Li, D. Samaras, and C. Chen. Topology-preserving deep image segmentation. In *NeurIPS*, 2019. 1

[31] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*, 2017. 8

[32] S. Kim, H.-G. Chi, X. Hu, Q. Huang, and K. Ramani. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *ECCV*, 2020. 10, 16

[33] D.P. Kingma and L.J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 8

[34] S.R. Kulkarni and S.E. Posner. Rates of convergence of nearest neighbor estimation under arbitrary sampling. *IEEE Transactions on Information Theory*, 41(4):1028–1039, 1995. doi: 10.1109/18.391248. 5

[35] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *ICLR*, 2022. 8, 9, 10

[36] A. Mishra and F.C. Motta. Stability and machine learning applications of persistent homology using the Delaunay-Rips complex. *Front. Appl. Math. Stat.,*, 9:1–18, 2023. 2

[37] M. Mitzenmacher and E. Upfal. *Probability and Computing – Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. 20

[38] M. Moor, M. Horn, B. Rieck, and K. Borgwardt. Topological autoencoders. In *ICML*, 2020. 1

[39] T. Papamarkou, T. Birdal, M.M. Bronstein, G.E. Carlsson, J. Curry, Y. Gao, M Hajij, R. Kwitt, P. Lio, P. Di Lorenzo, V. Maroulas, N. Miolane, F. Nasrin, N.K. Ramamurthy, B. Rieck, S. Scardapane, M.T. Schaub, P. Veličković, B. Wang, Y. Wang, G. Wei, and G. Zamzmi. Position: Topological deep learning is the new frontier for relational learning. In *ICML*, 2024. 1, 2

[40] C.R. Qi, L. Yi, H. Su, and L.J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 8, 9, 10

[41] H. Ritter. *An efficient bounding sphere*, page 301–303. Academic Press Professional, Inc., USA, 1990. ISBN 0122861695. 6

[42] V. Robins. Towards computing homology from finite approximations. *Topology proceedings*, 24(1):503–532, 1999. 1

[43] M. Royer, F. Chazal, C. Levrard, Y. Umeda, and Y. Ike. Atol: Measure vectorization for automatic topologically-oriented learning. In *AISTATS*, 2021. 8

[44] D.R. Sheehy. Linear-size approximations to the vietoris–rips filtration. *Discrete Comput Geom*, 49(4):778–796, 2013. ISSN 1432-0444. 2, 8

[45] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0.1 edition, 2024. 5

[46] The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2015. URL `http://gudhi.gforge.inria.fr/doc/latest/`. 6, 8

[47] P. Tillet, H. T. Kung, and D. Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19. ACM, 2019. ISBN 978-1-4503-6719-6. 6, 14

[48] I. Trofimov, D. Cherniavskii, E. Tulchinskii, N. Balabin, E. Burnaev, and S. Barannikov. Learning topology-preserving data representations. In *ICLR*, 2023. 1

[49] J. Wald, N. Goessweiner-Mohr, A. Real-Hohn, D. Blaas, and T. C. Marlovits. DMSO might impact ligand binding, capsid stability, and RNA interaction in viral preparations. *Sci. Rep.*, 14 (1):30408, Dec 2024. 6, 16

[50] C. Wang, Q. Wu, X. Liu, and L. Quintanilla. Automated machine learning & tuning with FLAML. In *KDD*, 2022. 8

[51] S. Zeng, F. Graf, C. Hofer, and R. Kwitt. Topological attention for time series forecasting. In *NeurIPS*, 2021. 1

[52] C. Zhang, H. Wan, S. Liu, X. Shen, and Z. Wu. PVT: Point-voxel transformer for 3d deep learning, 2021. URL `https://arxiv.org/abs/2108.06076`. arXiv. 8, 9, 10

[53] W. Zhirong, S. Song, A. Khosla, Y. Fisher, Z. Linguang, T. Xiaoou, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015. 10, 16

[54] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33 (2):249–274, 2005. 1

Supplementary Material

In following supplementary parts to the main manuscript, we discuss implementation details & provide dataset descriptions (in Appendix A), some additional experiments (in Appendix B), and all proofs (in Appendix C) for our theoretical results.

A Additional details

A.1 Implementation details of masking and flooding procedures

In this section, we provide an algorithm for computing the *masking* and *(approximate) flooding process* tailored to modern GPU architectures.

First, recall that we compute for each simplex $\sigma \in \mathrm{Del}(L)$ its (approximate) filtration value $f(\sigma) = \max_{p \in P_\sigma} \min_{x \in X} d(p, x)$ with $P_\sigma \subset \mathrm{conv}(\sigma)$ a finite set of points on the convex hull of σ.

As described in Section 4, to avoid computing all $\sum_{\sigma \in \mathrm{Del}(L)} |P_\sigma| \cdot |X|$ pairwise distances $d(p, x)$, we identify for each simplex σ a subset of points from X over which the minimum $\min_{x \in X} d(p, x)$ is achieved. Specifically, this set is $B_{\sqrt{2}r_\sigma}(c_\sigma) \cap X$, where $B_{r_\sigma}(c_\sigma)$ is an enclosing ball of σ computed using a variation of Ritter's heuristic. Checking for all simplices $\sigma \in \mathrm{Del}(L)$ which points $x \in X$ are within $B_{\sqrt{2}r_\sigma}(c_\sigma)$ requires $|\mathrm{Del}(X)| \cdot |X|$ distance evaluations. In fact, by presorting X along one coordinate axis, we can compute, for each simplex σ, a slice $\tilde{X}_\sigma$ of X enclosing the ball $B_{r_\sigma}(c_\sigma)$ via two binary searches, leading to an $O(\sum_\sigma |\tilde{X}_\sigma| + |X| \log |X| + |\mathrm{Del}(L)| \log |X|)$ complexity, which is usually much faster than doing all $|\mathrm{Del}(L)| \cdot |X|$ distance evaluations. The distances $d(c_\sigma, x)$ can then be computed independently, and are therefore straightforward to parallelize, yielding a Boolean masking matrix of shape $|\mathrm{Del}(L)| \times |X|$.

To be more precise, in our implementation, we also presort the simplex centers c_σ along the same coordinate axis. This allows for efficient selection of a common bounding slice $\tilde{X}_B$ for an entire *batch* B of simplices to compute a masking matrix of shape $b \times \tilde{X}_B$, where $b = |B|$ is the batch size. The non-zero indices of the masking matrix are then, together with the points in $\tilde{X}_B$ and the sets $P_{\sigma_i}, i \in B$, input to a custom Triton [47] kernel, which efficiently computes the directed Hausdorff distances $f(\sigma_i)$.

We first initialize (with infinity) an array A of shape $b \times |P_\sigma|$ in shared memory which collects the minimal distances $\min_{x \in X} d(p, x)$ for all $p \in P_B := (P_{\sigma_1}, \ldots, P_{\sigma_B})$. The kernel is then launched over a grid of multiple independent programs, each scheduled to run on the GPU's streaming multiprocessors. Each program receives the points in the slice $\tilde{X}_B$ and a chunk of simplex points P_σ all from the same simplex σ. Moreover, it receives a chunk of non-zero (σ, x) index pairs of the masking matrix, where we ensure that, for one chunk, all pairs correspond to the simplex σ. It then computes the minimal distances $\min_x d(p, x)$, where the minimum is taken over all x in the chunk, and finally writes these values to A via an atomic min operation. Once all programs terminate, the filtration value $f(\sigma_i)$ of each simplex in the batch is computed by taking the maximum of a row of A.

In fact, when representing each simplex σ by a discrete set P_σ defined in terms of barycentric coordinates as described in Section 4 (or any other discrete set P_σ such that $P_\tau \subset P_\sigma$ for all faces $\tau \subset \sigma$), we can directly extract the filtration values of its faces by taking the maximum along the rows over appropriately selected column indices. It therefore suffices to perform the masking and distance computation only on batches of maximal simplices.

We additionally support a CPU implementation based on a k-d tree [5] data structure. Avoiding the masking step, we directly build a k-d tree on the points X and compute the (global) minimal distance matrix A (of shape $|X| \times |P_\sigma|$) using nearest-neighbor queries for each $p \in \bigcup_{\sigma \in \mathrm{Del}(L)} P_\sigma$.

A.2 Source code

Flooder source code. We provide the full source code for constructing the Flood complex with subsequent PH computation at

https://github.com/plus-rkwitt/flooder

Furthermore, for easy use, practitioners can *install* (including all dependencies) our Python package `flooder`[2] (which is available on PyPi) via

```
pip install flooder
```

Below is a minimal working example (MWE) of how to compute Flood PH on 1M points sampled from the surface of a torus. Other examples, including timing experiments, can be found in the `examples` folder of the GitHub repository listed above.

```
from flooder import (
    flood_complex, generate_landmarks, generate_noisy_torus_points_3d
)

DEVICE = "cuda"
n_pts = 1_000_000  # Number of points to sample from torus
n_lms = 1_000      # Number of landmarks for Flood complex

pts = generate_noisy_torus_points_3d(n_pts).to(DEVICE)
lms = generate_landmarks(pts, n_lms)

stree = flood_complex(pts, lms, return_simplex_tree=True)
stree.compute_persistence()
ph_diags = [stree.persistence_intervals_in_dimension(i) for i in range(3)]
```

Datasets. In addition to the Flood complex implementation, the `flooder` package provides all point cloud datasets used for the object classification experiments in Section 5.3. These datasets are ready-to-use, with all pre-processing steps already applied and come with pre-defined splits to ensure reproducibility. These topologically challenging datasets can serve as a standardized benchmark for future research in topological data analysis and machine learning on point clouds in general. Below is a minimal working example (MWE) of how to load a dataset.

```
from flooder.datasets import (
    CoralDataset, MCBDataset, RocksDataset,
    ModelNet10Dataset, SwisscheeseDataset,
)

dataset = CoralDataset('./coral_dataset/')
for data in dataset:
    print(data.x.shape, data.y)
```

Experiments code. Experiments can be reproduced using the following repository:

https://github.com/plus-rkwitt/flooder-experiments

A.3 Computing infrastructure

The main experiments were run on an SUSE Linux Enterprise Server 15 SP6 system with AMD EPYC 9554 64-Core Processors, 1024 GB of main memory, and NVIDIA H100 80GB HBM3 GPUs.

[2] all experiments were run with `flooder (v1.0rc5)`

A.4 Dataset details

In the following, we provide a more detailed description of the used datasets and their properties.

Corals. We collected and curated a challenging 3D point cloud classification dataset by uniformly sampling 1M points from surface meshes of *corals* obtained from the *Smithsonian 3D Digitization* initiative[3]. In particular, this dataset is a *subset* of all coral meshes that are available under CC0 license, classified by *genus*. We label the corals according to their genus, and only use classes that have at least 30 instances. Overall, this yields a binary classification problem with a total of 83 available coral meshes. Specifically, there are 31 corals with the genus Acroporidae and 52 with Poritidae. On average, the meshes have ≈ 900k vertices (ranging from ≈ 29k to ≈ 10M with median ≈ 500k). A rendered example (with 1818450 vertices) is shown in Figure 6. In particular, the coral *Leptoseris paschalensis* (USNM 53156), which is the mesh with the most vertices in the collection, is also used to showcase the capabilities of the Flood complex in computing PH on large point clouds; see Section 5.1.

Figure 6: One example of a *Poritidae* coral.

Mechanical Components Benchmark (MCB). We used a *subset*, dubbed `mcb-c`, of the publicly available MCB-A dataset [32] to assess the classification performance on geometrically and topologically challenging objects. We filter MCB-A by (1) mesh size and (2) class size. In particular, we only take meshes with more than 10k vertices and then keep classes with more than 30 remaining instances. Here, the *vertex count* of a mesh serves as a proxy of *geometrical complexity*, with the intuition that object meshes with a larger number of vertices tend to be geometrically more complex. After filtering, 1,745 meshes split into 11 classes remain (out of 68 original classes), with ≈ 34.5k vertices on average. Finally, we uniformly sample 1M points from each mesh surface to obtain our training/testing point clouds.

ModelNet10. `modelnet10`[4] is a subset of the larger ModelNet40 benchmark [53] with 10 object categories (bathtub, toilet, table, etc.) distributed across 4,899 surface meshes. Originally, we have 3,591 training and 1,308 testing instances. We aggregate all meshes and then create ten random 80/20% training/testing splits. On average, we have ≈ 9.5k vertices per mesh. Overall, this dataset contains geometrically rather *simple* objects with few characteristic topological features. From each surface mesh we uniformly sample 250k points.

Rhinovirus. This point cloud is obtained from a cryo-electron-microscopy reconstruction [49] of the protein structure of RV-A89[5]. The raw (density) data is provided as voxels, which we first smooth out by applying a 3D average pooling (with kernel size 3, stride 1 and padding 1). We then transform it into a point cloud by taking the centers of the voxels that pass the provided density threshold (of 0.151). This results in $|X| \approx 12.3$M points.

Rocks. We created a point cloud dataset mimicking porous materials using the `PoreSpy` library [23]. In particular, we create 1k Boolean voxel grids of size 256^3, whose true values define the point clouds. In addition, we compute the *surface area* of a voxel grid as the sum over all voxels of the differences between the voxel grid and its shift by 1 along an axis. Following this strategy, we create 500 voxel grids using the *fractal noise* data generator and 500 using the *blobs* generator. In both cases, we select the porosity hyperparameter from a uniform distribution on $[0.05, 0.95]$, resulting in point cloud sizes between 800k and 16M. The hyperparameters that affect the surface area (i.e., frequency for fractal noise and "blobiness" for blobs) are empirically tuned so that a wide range of (porosity, surface) tuples is approximately uniformly covered, see Figure 7. For details, we refer to the source code.

B Additional experiments

B.1 Runtimes on different GPUs

We compare the runtime for computing Flood PH using different GPU architectures. Specifically, we report runtimes (in s) on an NVIDIA GeForce RTX 2080 Ti, a GeForce RTX 3090 and a H100 NVL

[3] available at `https://3d.si.edu/corals`

[4] available at `https://modelnet.cs.princeton.edu`

[5] available at `https://www.emdataresource.org/EMD-50844`

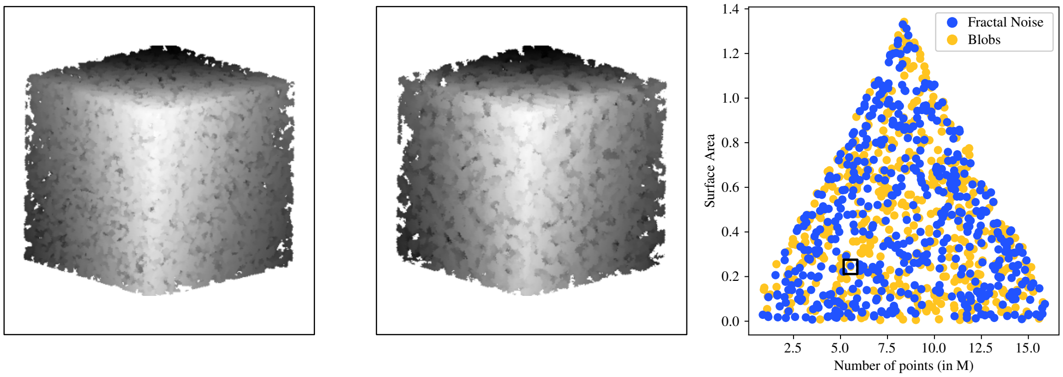


Figure 7: Two point clouds from the `rocks` dataset with similar porosity and surface area, one created using the blobs generator (**left**) and one using the fractal noise generator (**middle**). The **right** panel shows the number of points plotted against surface area (i.e., the regression target) for both generators. The black square □ indicates the location of the two examples.

card. As in Section 5.2, we evaluate on the *vertices* of the meshes used for generating the `corals` dataset and show results in Figure 8.

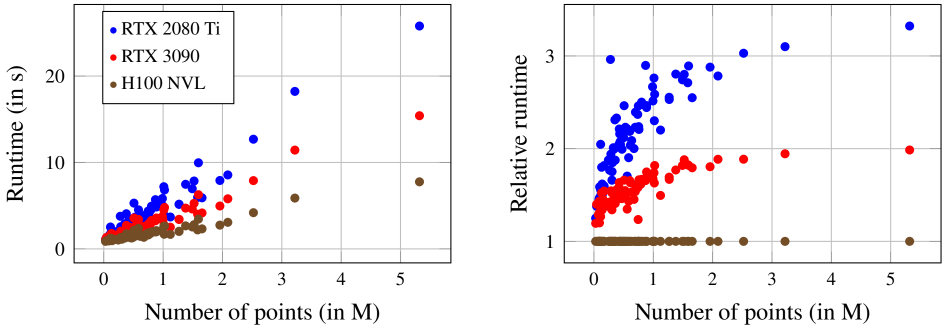


Figure 8: Absolute (**left**) and relative (**right**) runtimes of Flood complex construction plus persistent homology computation on the vertices of coral meshes on different GPUs. Hyperparameters were 2048 landmarks, 1024 random points and batch size 256.

B.2 Additional results on large-scale point clouds

We present additional results to Section 5.1. Figure 9 extends Figure 2 and shows *all* persistence diagrams from the RV-A89 virus and the Leptoseris paschalensis coral. Figure 10 shows bottleneck distances to Alpha PH on the full coral point cloud for varying numbers of landmarks. When comparing to Figure 2, one notices the much larger runtimes of Flood PH on the coral point cloud, which is increased by a factor of around 2 when many landmarks are used, e.g., 27 versus 14 seconds when using 5k landmarks and 35 versus 17 seconds when using 10k landmarks. On closer inspection, this is caused by the masking step of Flood PH being more successful for the latter, resulting in only around half of the distance comparisons that need to be done for computing the filtration values.

B.3 Effect of landmark selection method

In Table 4, we compare the bottleneck distances between Alpha PH on the full large-scale point clouds from Section 5.1 when landmarks are selected using either farthest point sampling (FPS) or uniform random sampling (abbr. as Rand). We report results obtained when using 1k landmarks and filtration values are computed from a grid with 30 points per edge. Specifically, we report mean and standard deviations from 25 runs, where the randomness in FPS comes from different initial points. As expected, FPS achieves a better coverage of the point clouds, resulting in lower Hausdorff distances between landmarks and the point cloud, and consequently lower bottleneck distances to the persistent homology of Alpha PH (full).

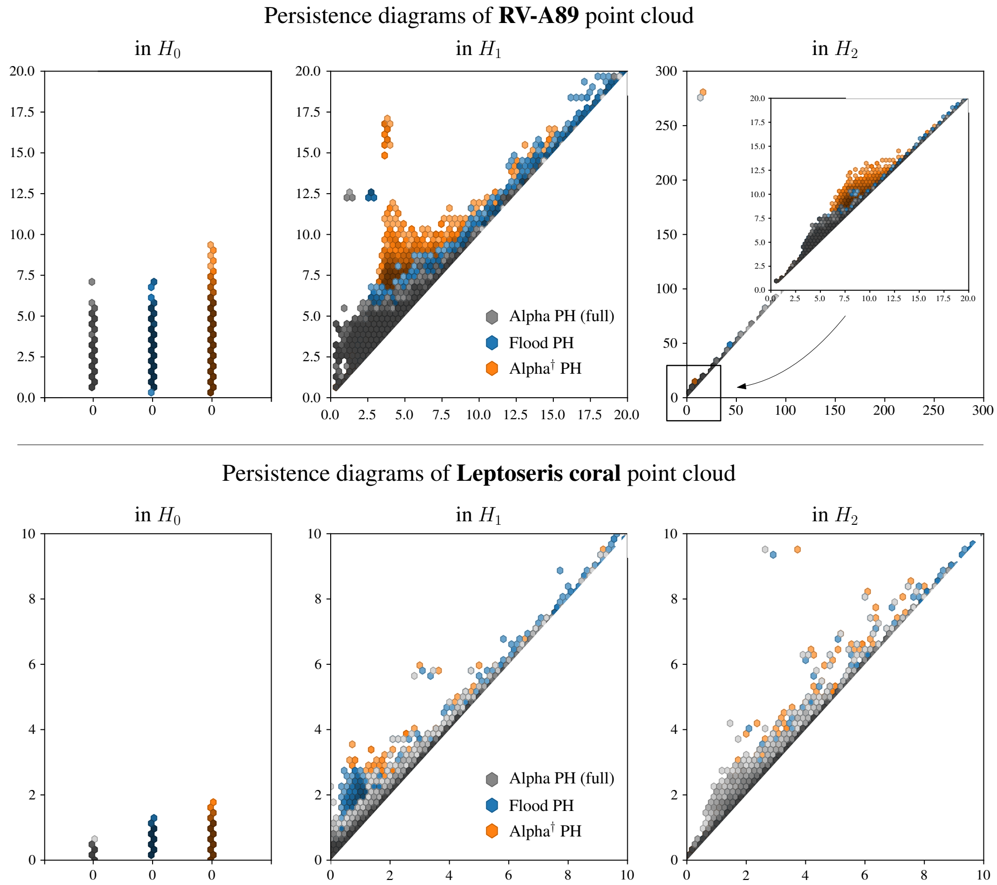


Figure 9: Hexbin plots of persistence diagrams of the RV-A89 and Leptoseris coral point clouds. Gray is Alpha PH on the full point cloud, blue is Flood PH with 10k landmarks, orange is Alpha PH on a subset of size 175k.

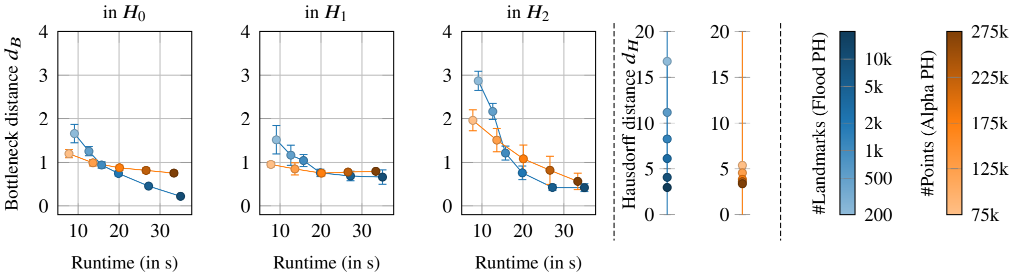


Figure 10: *Approximation quality* of Flood PH and Alpha PH on the Leptoseris paschalensis coral. The **(left)** panel shows bottleneck distances to Alpha PH (full) in H_0, H_1 and H_2 when varying the number of landmarks for Flood PH and the subsample size for Alpha PH. The (**middle**) panel shows the Hausdorff distance between the full point cloud and the landmarks, resp., the points in the subsample. The (**right**) panel shows the color coding used in all the plots.

Table 4: Comparison of bottleneck distances (d_B) between Alpha PH (full) and Flood PH when selecting landmarks using FPS and uniformly random (Rand) sampling on the large-scale point clouds from Section 5.1.

	RV-A89		Leptoseris Paschalensis	
	FPS	Rand	FPS	Rand
Hausdorff distance $d_H(L, X)$	44.7 ± 0.2	79.1 ± 4.0	8.2 ± 0.1	20.0 ± 3.3
Bottleneck distance d_B in H_0	1.7 ± 0.1	3.2 ± 0.1	0.9 ± 0.1	0.9 ± 0.1
Bottleneck distance d_B in H_1	3.9 ± 0.8	3.3 ± 0.5	1.0 ± 0.1	1.2 ± 0.2
Bottleneck distance d_B in H_2	5.7 ± 1.1	5.4 ± 1.6	1.2 ± 0.2	2.6 ± 3.2

C Theory

C.1 Proofs

For $p \in \mathbb{R}^d$ and a bounded set $X \subset \mathbb{R}^d$, we define $d(p, X) = \min_{x \in X} d(p, x)$. We recall the definition of the *directed Hausdorff distance*

$$\vec{d}_H(X, Y) := \max_{x \in X} \min_{y \in Y} d(x, y) = \max_{x \in X} d(x, Y) \tag{7}$$

for bounded sets $X, Y \subset \mathbb{R}^d$.

For completeness, we include a proof that the directed Hausdorff distance satisfies a triangle inequality.

Lemma 8. *Let (X, d) be a metric space and let $A, B, C \subset X$ be compact. Then,*

$$\max_{a \in A} d(a, C) \leq \max_{a \in A} d(a, B) + \max_{b \in B} d(b, C) \quad . \tag{8}$$

Proof. Fix $a_0 \in A$. Since A, B, C are compact, there are $b_0 \in B$ such that $d(a_0, B) = d(a_0, b_0)$ and $c_0 \in C$ such that $d(b_0, C) = d(b_0, c_0)$. Hence,

$$\begin{aligned} d(a_0, C) &= \inf_{c \in C} d(a_0, c) \leq d(a_0, c_0) \\ &\leq d(a_0, b_0) + d(b_0, c_0) = d(a_0, B) + d(b_0, C) \\ &\leq \max_{a \in A} d(a, B) + \max_{b \in B} d(b, C) \quad . \end{aligned}$$

Since $a_0 \in A$ was arbitrary, we conclude

$$\max_{a \in A} d(a, C) \leq \max_{a \in A} d(a, B) + \max_{b \in B} d(b, C) \quad .$$

□

Proof of Theorem 2

Proof. Let $\sigma \in \mathrm{Del}(L)$ and denote by $f, f' : \mathrm{Del}(L) \to \mathbb{R}^+$ the filter functions that define $\mathrm{Flood}(X, L)$ and $\mathrm{Flood}(X', L)$, respectively. Observe, that f is the directed Hausdorff distance between $\mathrm{conv}(\sigma)$ and X, which satisfies the following triangle inequality (see Lemma 8),

$$f(\sigma) = \max_{p \in \mathrm{conv}(\sigma)} d(p, X) \leq \max_{p \in \mathrm{conv}(\sigma)} d(p, X') + \max_{x' \in X'} d(x', X) = f'(\sigma) + \max_{x' \in X'} d(x', X) \quad ,$$

and therefore $f(\sigma) - f'(\sigma) \leq \max_{x' \in X'} d(x', X)$. As, similarly, $f'(\sigma) - f(\sigma) \leq \max_{x \in X} d(x, X')$, we conclude

$$|f(\sigma) - f'(\sigma)| \leq \max \left(\max_{x \in X} d(x, X'), \max_{x' \in X'} d(x', X) \right) = d_H(X, X') \quad . \tag{9}$$

The main stability theorem from [15] then implies Eq. (2) which concludes the proof. □

Proof of Corollary 4

Proof. Upon combination of Theorems 2 and 3, and Hausdorff stability of Alpha complexes, we immediately get $\forall i \in \mathbb{N}$:

$$\begin{aligned} & d_B \left(\mathrm{dgm}_i(\mathrm{Alpha}(X)), \mathrm{dgm}_i(\mathrm{Flood}(X, L)) \right) \\ & \qquad \leq d_B \left(\mathrm{dgm}_i \,\mathrm{Alpha}(X)), \mathrm{dgm}_i(\mathrm{Alpha}(L)) \right) + d_B \left(\mathrm{dgm}_i(\mathrm{Alpha}(L)), \mathrm{dgm}_i(\mathrm{Flood}(X, L)) \right) \\ & \qquad \leq d_H(X, L) + d_H(X, L) = 2 d_H(X, L) \quad . \end{aligned}$$

□

Proof of Theorem 5

Proof. To prove the theorem, we need to show that $\mathrm{dgm}_i(\mathrm{Flood}(X, L))$ and $\mathrm{dgm}_i(\mathrm{Flood}(X, L, P))$ are $\max_{\sigma\in\mathrm{Del}(L)} d_H(P_\sigma, \mathrm{conv}(\sigma))$ interleaved. As already mentioned, $\mathrm{Flood}_r(X, L) \subset \mathrm{Flood}_r(X, L, P)$. Consequently, it suffices to show that $\mathrm{Flood}_r(X, L, P) \subset \mathrm{Flood}_{r+\epsilon}(X, L)$ for any $r \geq 0$ and $\epsilon < \max_{\sigma\in\mathrm{Del}(L)} d_H(P_\sigma, \mathrm{conv}(\sigma))$.

To this end, we assume that $\sigma \in \mathrm{Flood}_r(X, L, P)$, i.e., $P_\sigma \subset \bigcup_{x\in X} B_r(x)$, and let $\epsilon < \max_{\sigma\in\mathrm{Del}(L)} d_H(P_\sigma, \mathrm{conv}(\sigma))$. By assumption, for each $q \in \mathrm{conv}(\sigma)$, there is a point $p \in P_\sigma$ such that $d(q, p) \leq \epsilon$. Moreover, since $\sigma \in \mathrm{Flood}_r(X, L, P)$, for each $p \in P_\sigma$ there exists a point $x_p \in X$ with $d(p, x_p) \leq r$. Hence,

$$d(q, X) \leq d(q, x_p) \leq d(q, p) + d(p, x_p) \leq r + \epsilon \ , \tag{10}$$

and thus $\mathrm{Flood}_r(X, L, P) \subset \mathrm{Flood}_{r+\epsilon}(X, L)$. Eq. (5) then follows from [15]. □

Proof of Lemma 6

Proof. Let $\sigma = \{v_0, \dots, v_k\} \subset \mathbb{R}^d$ a k-simplex and let $P_\sigma = \{p = \sum_{i=0}^k \lambda_i v_i \in : \sum_{i=0}^k \lambda_i = 1, m\lambda_i \in \mathbb{N}\} \subset \mathrm{conv}(\sigma)$ be the grid points. For any $q := \sum_{i=0} \mu_i v_i \in \mathrm{conv}(\sigma)$, there exists a point $p := \sum_{i=0} \lambda_i v_i \in P_\sigma$ such that $|\mu_i - \lambda_i| \leq 1/m$ for every $i = 0, \dots, k$. Now, let $\delta_i := \mu_i - \lambda_i$ and observe that $\sum_{i=0}^k \delta_i = 0$. It follows that

$$\begin{aligned}
2\,\|p - q\|^2 = 2\left\|\sum_{i=0}^k \delta_i v_i\right\|^2 &= 2\sum_{i,j} \delta_i\delta_j \langle v_i, v_j\rangle \\
&= 2\sum_{i,j} \delta_i\delta_j \langle v_i, v_j\rangle - \sum_j \delta_j \sum_i \delta_i \|v_i\|^2 - \sum_i \delta_i \sum_j \delta_j \|v_j\|^2 \\
&= -\sum_{i,j} \delta_i\delta_j \left(\|v_i\|^2 + \|v_j\|^2 - 2\langle v_i, v_j\rangle\right) \\
&= -\sum_{i,j} \delta_i\delta_j \|v_i - v_j\|^2 \ ,
\end{aligned}$$

and therefore that

$$\|p - q\|^2 = -\sum_{i<j} \delta_i\delta_j \|v_i - v_j\|^2 \leq \frac{1}{m^2} \sum_{i<j} \|v_i - v_j\|^2 \ . \tag{11}$$

Since this holds for any $q \in \mathrm{conv}(\sigma)$, we conclude $d_H(P_{\mathrm{conv}(\sigma)}, \mathrm{conv}(\sigma)) \leq \frac{1}{m}\sqrt{\sum_{i<j} \|v_i - v_j\|^2}$. □

A similar result holds with high probability for *randomly sampled points* as shown next.

Lemma 9. *Let $P_\sigma \subset \mathbb{R}^d$ be a set of independently drawn points from a uniform distribution on* $\mathrm{conv}(\sigma)$ *of size $\tilde{O}(\epsilon^{-d} \ln(1/\delta))$. Then, with probability of at least $1 - \delta$ (over the choice of P_σ), we have*

$$d_H(P_\sigma, \mathrm{conv}(\sigma)) \leq \epsilon\,\mathrm{diam}(\sigma) \ . \tag{12}$$

Proof. Let $\epsilon > 0$, and let $Q_r = \{Q_1, \dots, Q_m\}$ be a partition of σ into sets of diameter at most $r := \epsilon\,\mathrm{diam}(\sigma)$, which can be obtained, e.g., by taking a covering of σ with balls B_j of radius $r/2$ and letting

$$Q_j = B_j \setminus \bigcup_{k=1}^{j-1} B_k \ .$$

Note that m is smaller than the covering number of $\mathrm{conv}(\sigma)$ for radius $r/2$, and therefore $m \leq (4\,\mathrm{diam}(\sigma)/r)^d = (4\epsilon)^d$. By a standard *balls in bins* argument [37], we obtain

$$\mathbb{P}\left[\max_{p\in\sigma} d(p, P_\sigma) > \epsilon\,\mathrm{diam}(\sigma)\right] = \mathbb{P}\left[\max_{p\in\sigma} d(p, P_\sigma) > r\right] \leq \mathbb{P}[\exists j : Q_j \cap P_\sigma = \emptyset] \leq m e^{-|P_\sigma|/m} \ .$$

Taking $|P_\sigma| \geq (4/\epsilon)^d \big(d \ln(4/\epsilon) + \ln(1/\delta)\big) \geq m \ln(m/\delta)$ yields $\mathbb{P}[\max_{x\in\sigma} d(x, P_\sigma) > \epsilon\mathrm{diam}(\sigma)] \leq \delta$ which establishes the claim. □

Proof of Lemma 7

Proof. Let $\sigma = \{v_0, \dots, v_k\}$ and let $p = \sum_{i=0}^k \lambda_i v_i \in \operatorname{conv}(\sigma)$, i.e., $\sum_{i=0}^k \lambda_i = 1$ with $\lambda_i > 0$. Since $\sigma \subset X$, $z = \arg\min_{x \in X} d(p, x)$ implies $d(z, p) \leq d(p, v_i)$ for each $i \in \{1, \dots, k\}$. It therefore suffices to minimize $d(p, x)$ over $x \in B_{\min_i d(p, v_i)} \cap X$.

Below, we will show that if $B_r(c)$ is an enclosing ball of σ, then for any $p \in \operatorname{conv}(\sigma)$ it holds that $B_{\min_i d(p,v_i)}(p) \subset B_{\sqrt{2}r}(c)$, and therefore that $\max_{p \in \operatorname{conv}(\sigma)} \min_{x \in X} d(p, x) = \max_{p \in \operatorname{conv}(\sigma)} \min_{x \in X \cap B_{\sqrt{2}r}(c)} d(p, x)$.

First, consider an arbitrary point $c \in \mathbb{R}^d$. Observe that

$$\|v_i - c\|^2 = \|v_i - p\|^2 + \|p - c\|^2 + 2\langle v_i - p, p - c\rangle$$

for each $v_i \in \sigma$, and therefore

$$\sum_{i=0}^k \lambda_i \|v_i - c\|^2 = \sum_{i=0}^k \lambda_i \|p - c\|^2 + \sum_{i=0}^k \lambda_i \|v_i - p\|^2 + 2 \sum_{i=0}^k \langle \lambda_i (v_i - p), p - c\rangle = \|p - c\|^2 + \sum_{i=0}^k \lambda_i \|v_i - p\|^2 \ ,$$

where the last equality follows from $\sum_{i=0}^k \lambda_i = 1$ and $\sum_{i=0}^k \lambda_i v_i = p$. In particular, if $B_r(c)$ is an enclosing ball of σ (and therefore $\operatorname{conv}(\sigma)$), then

$$\|p - c\|^2 = \sum_{i=0}^k \lambda_i \|v_i - c\|^2 - \sum_{i=0}^k \lambda_i \|v_i - p\|^2 \leq r^2 - \min_{i=0,\dots,k} \|v_i - p\|^2 \ .$$

Hence, if $z \in B_{\min_i d(p,v_i)}(p)$, i.e., $\|z - p\| \leq \min_{i=0,\dots,k} \|p - v_i\|$, then

$$\begin{aligned} \|z - c\| &\leq \min_i \|v_i - p\| + \|p - c\| \\ &\leq \min_i \|v_i - p\| + \sqrt{r^2 - \min_i \|v_i - p\|^2} \\ &\leq \max_{0 \leq s \leq r} s + \sqrt{r^2 - s^2} \\ &\leq \sqrt{2} r \ . \end{aligned}$$

□

C.2 Proof of Theorem 3

In this section, we cover the *homotopy equivalence* between $\text{Flood}_r(X, X)$ and the union of balls $\bigcup_{x \in X} B_r(x)$. Specifically, we show that $|\text{Flood}_r(X, X)|$ is homotopy equivalent to $|\text{Alpha}_r(X)|$, which, according to the nerve theorem [24, Corollary 4G.3], is homotopy equivalent to $\bigcup_{x \in X} B_r(x)$.

C.2.1 Background tools

Let us first collect some tools and background information on Voronoi diagrams and Delaunay triangulations which, in arbitrary dimensions, express themselves as a combinatorial complex.

Voronoi complex. Consider $X \subset \mathbb{R}^d$ to be a finite point set. Then, a Voronoi complex tessellates $\mathbb{R}^d$ into closest-neighbor cells around the points of X. That is, we define by $\text{VC}(x, X) = \{p \in \mathbb{R}^d : d(p, x) \leq d(p, X)\}$ the *Voronoi cell* of x w.r.t. X. Below are some facts regarding Voronoi cells:

- A Voronoi cell is a d-dimensional convex polyhedron, i.e., the intersection of half spaces.
- A Voronoi cell is unbounded if and only if x is on the convex hull of X; in other words, the Voronoi cell is a convex polytope if x is in the interior of the convex hull of X.
- As a convex polyhedron, the Voronoi cell itself possesses a combinatorial complex structure, i.e., its lower-dimensional faces are convex polyhedra again, and so are the non-empty intersections.

The *Voronoi diagram* $\text{Vor}(X)$ is usually defined geometrically as the union of the boundaries of the Voronoi cells. From a combinatorial perspective, it is more convenient to define the Voronoi diagram

$\mathrm{Vor}(X)$ as the union of the Voronoi cells as complexes, and $\mathrm{Vor}(X)$ is then itself a complex of convex polyhedra. To emphasize this, we speak of the Voronoi complex and refer to the components of the Voronoi complex, $\mathrm{Vor}(X)$, as *Voronoi faces*.

Any k-dimensional Voronoi face f from $\mathrm{Vor}(X)$ is characterized by $d-k$ points $x_1, \ldots, x_{d-k}$ of X as follows: any point $x \in f$ is equidistant to $x_1, \ldots, x_{d-k}$, $d(x, x_i) \leq d(x, X)$ and, if x is from the relative interior $\mathrm{relint} f$ of f, then we even have $d(x, x_i) < d(x, X \setminus \{x_1, \ldots, x_{d-k}\})$. We call $x_1, \ldots, x_{d-k}$ the *defining points* of f.

Delaunay complex. We can naturally dualize a k-dimensional Voronoi face f by the simplex σ formed by its defining points $x_1, \ldots, x_{d-k}$. If we do this for the entire Voronoi diagram $\mathrm{Vor}(X)$, we obtain the Delaunay complex $\mathrm{Del}(X)$. We distinguish between the combinatorial simplex $\sigma = \{x_1, \ldots, x_{d-k}\}$ and its geometric realization $|\sigma| = \mathrm{conv}(\sigma)$. It will be convenient to denote f as $|\sigma|^{\dagger}$.

Note that $|\sigma|^{\dagger}$ and $|\sigma|$ are orthogonal, however $|\sigma|^{\dagger}$ might not intersect $|\sigma|$. So, let us denote by $\mathrm{aff}\,|\sigma|^{\dagger}$ the affine hull of $|\sigma|^{\dagger}$, i.e., the smallest affine subspace supported by $|\sigma|^{\dagger}$ (e.g., in case of $|\sigma|^{\dagger}$ being a line segment, $\mathrm{aff}\,|\sigma|^{\dagger}$ would be an infinite line). Then, if $\dim |\sigma| < d$, the point $|\sigma| \cap \mathrm{aff}\,|\sigma|^{\dagger}$ is the closest equidistant point to the defining points of σ. Also note that the distance function $d(\cdot, \sigma)$ has no further local minima on $\mathrm{aff}\,|\sigma|^{\dagger}$ and, in fact, is convex on $\mathrm{aff}\,|\sigma|^{\dagger}$ (the Euclidean distance from a given point is convex on any affine subspace). In particular, if $\mathrm{aff}\,|\sigma|^{\dagger}$ is of dimension 1, then walking along this line away from $|\sigma| \cap \mathrm{aff}\,|\sigma|^{\dagger}$ *increases* the distance to σ, and walking towards $|\sigma| \cap \mathrm{aff}\,|\sigma|^{\dagger}$ *decreases* this distance.

Simplicial collapses. Given an abstract simplicial complex Σ, it is often possible to simplify its structure while preserving the homotopy type of its geometric realization $|\Sigma|$ using simplicial collapses. Specifically, if $(\tau, \sigma) \in \Sigma$ are a pair of simplices such that $\tau \subset \sigma$ is a face of σ, $\dim \sigma = \dim \tau + 1$ and τ has no other cofaces, then the removal of the pair (τ, σ) from Σ is called an elementary collapse. Similarly, if $\dim \sigma > \dim \tau$ and all cofaces of $\nu \supset \tau$ of τ are faces $\nu \subset \sigma$ of σ, then (starting from the top) there is a sequence of elementary collapses removing all such simplices ν, including τ and σ. Such a sequence is called a (simplicial) collapse, and a simplex τ satisfying the assumption is called a *free face* of Σ. Importantly, simplicial collapses induce a deformation retraction between the geometric realizations of the complexes before and after the collapse, and therefore preserves the homotopy type. Consider a filtered complex $\{\Sigma_r : r \geq 0\}$ that stays constant between filtration values $i < i+1$ but changes at $r = i+1$ when a simplex τ is added. If, at $r = i+1$, a simplex τ is added, resulting in a change of the homotopy type between Σ_i and Σ_{i+1}, then we call the addition of τ a *critical event*. If instead multiple simplices are added at $i+1$ but their removal is a simplicial collapse from Σ_{i+1} to Σ_i, then we call their addition a *regular event*.

Alpha complexes. The Alpha complex $\mathrm{Alpha}_r(X)$ with radius r on a set $X \in \mathbb{X}^d$ is defined as the nerve of $\{\mathrm{VC}(x, X) \cap B_r(x) \colon x \in X\}$, i.e., $\sigma \subset X$ is a simplex of $\mathrm{Alpha}_r(X)$ if and only if $\bigcap_{x \in \sigma} (\mathrm{VC}(x, X) \cap B_r(x)) \neq \emptyset$. By the nerve theorem, $|\mathrm{Alpha}_r(X)|$ is homotopy equivalent to $\bigcup_{x \in X} B_r(x)$. Moreover, as a subcomplex of the Delaunay triangulation $\mathrm{Del}(X)$, it is naturally embedded in $\mathbb{R}^d$. An equivalent definition of Alpha complexes is given by the filtration function

$$f_{\mathrm{Alpha}} : \mathrm{Del}(X) \to \mathbb{R}, \quad \sigma \mapsto \inf \left\{ r \geq 0 \colon \bigcap_{x \in \sigma} (\mathrm{VC}(x, X) \cap B_r(x)) \neq \emptyset \right\} . \tag{13}$$

Lemma 10. *Let $X \subset \mathbb{R}^d$ be in general position. A simplex $\tau \in \mathrm{Del}(X)$ is a free face of* $\mathrm{Alpha}_{f_{\mathrm{Alpha}}(\tau)}$ *if and only if $|\tau| \cap \mathrm{relint}|\tau|^{\dagger} = \emptyset$.*

Proof. Assume that $|\tau| \cap \mathrm{relint}|\tau|^{\dagger} = \emptyset$. Then, the point $q \in |\tau|^{\dagger}$ that is closest to τ is on the boundary $\partial|\tau|^{\dagger}$. Hence, q is also closest to other points $x \in X \setminus \tau$, i.e., there are $\sigma \supset \tau$ with $q \in |\sigma|^{\dagger}$ and $f_{\mathrm{Alpha}}(\sigma) = f_{\mathrm{Alpha}}(\tau) = d(q, \tau)$. Sorting these cofaces in descending order by their degree, we get a sequence of elementary collapses which will eventually remove τ. For details, we refer to standard textbooks such as [20].

Conversely, assume $\{q\} = |\tau| \cap \mathrm{relint}|\tau|^{\dagger}$. Then q is the closest equidistant point to τ and therefore $f_{\mathrm{Alpha}}(\tau) = d(q, \tau)$. Moreover, since $q \in \mathrm{relint}|\tau|^{\dagger}$, there is no $\sigma \supset \tau$ with $q \in |\sigma|^{\dagger}$, and hence, no coface of τ is added at filtration value $f_{\mathrm{Alpha}}(\tau)$. Hence, τ is not a free face of $\mathrm{Alpha}_{f_{\mathrm{Alpha}}(\tau)}$. □

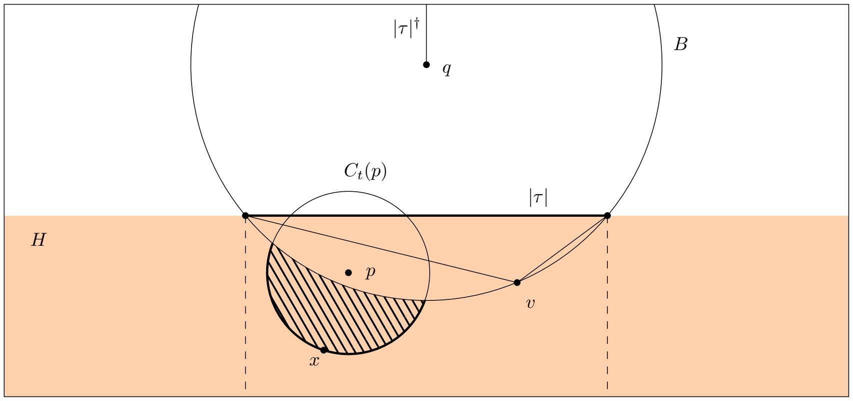


Figure 11: The existence of $p \in \mathrm{VC}(x, X) \cap (B \cap H)$ implies that x is on the circular arc that bounds the tiled region, and therefore $x \in H$.

C.2.2 Properties of the Flood filtration

Throughout this and the subsequent section, we will only consider Flood complexes whose landmark set L is the entire point set X. In this situation, the filtration function is given by

$$f_{\mathrm{Flood}} : \mathrm{Del}(X) \to \mathbb{R}, \quad \sigma \mapsto \max\{d(p, X) \colon p \in |\sigma|\} \ . \tag{14}$$

It is worth mentioning that the point $p \in |\sigma|$ that realizes $f_{\mathrm{Flood}}(\sigma) = d(p, X)$ cannot be in the interior of a Voronoi region $\mathrm{VC}(x, X)$ of some point $x \in X$.

Lemma 11. *Let $X \subset \mathbb{R}^2$ be in general position. A simplex $\tau \in \mathrm{Del}(X)$ is a free face of* $\mathrm{Flood}_{f_{\mathrm{Flood}(\tau)}}$ *if and only if $|\tau| \cap \mathrm{relint}|\tau|^\dagger = \emptyset$.*

Proof. Let $\{q\} = |\tau| \cap \mathrm{aff}|\tau|^\dagger$. Since q is the unique point on $|\tau|$ that is equidistant to τ, it is the point of $|\tau|$ that is contained in X_r the latest, i.e., only after the largest radius r. Consequently, $q \in |\tau|$ implies $f_{\mathrm{Flood}}(\tau) = d(q, \tau)$. Moreover, $q \in \mathrm{relint}|\tau|$ implies that every coface $\sigma \supset \tau$ contains a point $q' \in |\sigma| \cap |\tau|^\dagger$ with $d(q', \sigma) > d(q, \tau)$. Therefore, $f_{\mathrm{Flood}}(\sigma) > f_{\mathrm{Flood}}(\tau)$, and τ is not a free face.

Assume that $|\tau| \cap \mathrm{relint}|\tau|^\dagger = \emptyset$. Then, the point $q \in |\tau|^\dagger$ that is closest to τ is on the boundary $\partial|\tau|^\dagger$. Hence, q is also closest to other points $x \in X \setminus \tau$, i.e., there are $\sigma \supset \tau$ with $q \in |\sigma|^\dagger$. In fact, since $X \in \mathbb{R}^2$, $|\tau|$ must be an edge and $|\sigma| = |\tau \cup \{v\}|$ must be a triangle with additional vertex v. To show that τ is a free face, we need to show that σ has the same filtration value, i.e., that the point $o \in |\sigma|$ that realizes $f_{\mathrm{Flood}}(\sigma) = d(o, X)$ satisfies $o \in |\tau|$. To this end, denote by H the closed half-space bounded by $\mathrm{aff}|\tau|$ that contains $|\sigma|$, and denote by $B := B_{d(q,\sigma)}(q)$ the ball whose boundary is the circumcircle of $|\sigma|$. Observe that o must be on a Voronoi edge, and thus, we need to show that $d(\cdot, X)|_{|\sigma|}$ increases along the Voronoi edges towards $|\tau|$. The latter is true, if every 1-simplex $\gamma \in \mathrm{Del}(X)$ whose Voronoi edge $|\gamma|^\dagger$ intersects $|\sigma|$ satisfies (i) $\gamma \subset H$, and (ii) the point of $\mathrm{aff}|\gamma|^\dagger$ with minimal distance to γ is outside the interior $\mathrm{relint}|\sigma|$ (because $d(\cdot, \gamma)$ is convex on $|\gamma|^\dagger$) .

We first show $\gamma \subset H$. In fact, we show the (slightly) more general result that for any $x \in X$, $\mathrm{VC}(x, X) \cap (B \cap H) \neq \emptyset$ implies $x \in H$. Obviously, the latter is true, if $x \in \tau$. Hence, from now on we assume $x \in X \setminus \tau$, as illustrated in Figure 11. By assumption, there is a point $p \in \mathrm{VC}(x, X) \cap (B \cap H)$. Since $p \in \mathrm{VC}(x, X)$, it holds that $t := d(p, x) < d(p, \tau)$. Further, $p \in B \cap H$, i.e., p is in the minor circular segment of B defined by $|\tau|$. In particular, p is contained in the subset of H that orthogonally projects onto $|\tau|$. Hence, the circle $C_t(p)$ with center p and radius t intersects the affine hull $\mathrm{aff}|\tau|$ only in a subset of the *segment* $|\tau|$. As $|\tau|$ is a chord of B, this implies that $C_t(p) \setminus B \subset H$. Since also $x \in C_t(p) \setminus B$, we conclude $x \in H$.

To finish the proof, we need to show that the point $c \in \mathrm{aff}|\gamma|^\dagger$ that minimizes the distance to γ is not in the interior of $|\sigma|$. But this point c is simply the midpoint of $|\gamma|$, and therefore $c \in |\gamma|$. Hence, $c \in \mathrm{relint}|\sigma|$ would imply that $|\gamma|$ intersects $\mathrm{relint}|\sigma|$, which contradicts that $|\mathrm{Del}(X)|$ is a well-defined geometric realization, i.e., that realizations of simplices only intersect in a common face. □

Lemma 12. *Let $X \subset \mathbb{R}^d$ in general position and let $\sigma \in \mathrm{Del}(X)$. Then, $f_{\mathrm{Flood}}(\sigma) \leq f_{\mathrm{Alpha}}(\sigma)$ with equality if and only if $|\sigma| \cap |\sigma|^\dagger \neq \emptyset$.*

Proof. Denote by c the center of the minimal enclosing ball of σ, observe that $c \in |\sigma|$, and that $f_{\mathrm{Flood}}(\sigma) \leq d(c, \sigma)$. Assume that $f_{\mathrm{Alpha}}(\sigma) = r$. Then, $|\sigma|^\dagger \cap \bigcap_{x \in \sigma} B_r(x) \neq \emptyset$, and therefore $c \in \bigcap_{x\in\sigma} B_r(x) \neq \emptyset$. In particular, $c \in \bigcap_{x \in \sigma} B_r(x)$, and thus $f_{\mathrm{Flood}}(\sigma) \leq d(c, \sigma) \leq r = f_{\mathrm{Alpha}}(\sigma)$. Herein, equality holds if and only if $|\sigma|^\dagger \cap \bigcap_{x\in\sigma} B_r(x) = \{c\}$. The lemma then follows because $c \in |\sigma|$ and $|\sigma| \cap |\sigma|^\dagger$ is either empty or a singleton $\{c\}$. □

C.2.3 Proof of the theorem

Lemma 13. *Let $X \subset \mathbb{R}^2$ be in general position and let $\sigma \in \mathrm{Del}(X)$. The addition of σ to* $\mathrm{Alpha}(X)$ *is a critical event if and only if the addition of σ to* $\mathrm{Flood}(X)$ *is a critical event. Moreover, if both are true, then, $f_{\mathrm{Flood}}(\sigma) = f_{\mathrm{Alpha}}(\sigma)$.*

Proof. Let $\sigma \in \mathrm{Del}(X)$. The addition of σ to $\mathrm{Flood}(X, X)$ or $\mathrm{Alpha}(X)$ is a regular event if and only if σ is a free face or a coface of a free face in $\mathrm{Flood}_{f_{\mathrm{Flood}}(\sigma)}$ or $\mathrm{Alpha}_{f_{\mathrm{Alpha}}(\sigma)}$, respectively. By definition, the addition of σ is a critical event if it is not regular. The lemma then follows directly from Lemmas 10 to 12. □

Theorem 14. *Let $X \subset \mathbb{R}^2$ be in general position. Then,* $\mathrm{Flood}_r(X, X)$, $\mathrm{Alpha}_r(X)$ *and* $\bigcup_{x\in X} B_r(x)$ *have the same homotopy type for any $r \geq 0$.*

Proof. The homotopy equivalence between $\mathrm{Alpha}_r(X)$ and $\bigcup_{x \in X} B_r(x)$ follows directly from the nerve theorem, see also [19] for the construction of a particular deformation retraction. Hence, it suffices to show the homotopy equivalence between $\mathrm{Flood}_r(X, X)$ and $\mathrm{Alpha}_r(X)$. To this end, we will construct a sequence of simplicial collapses from $\mathrm{Flood}_r(X, X)$ onto $\mathrm{Alpha}_r(X)$ for each r where each collapse is an elementary collapse of an (edge, triangle) pair.

Let $0 = t_0 < t_1 < \cdots < t_\infty = \infty$ be the ordered list of filtration values for which a simplex $\sigma \in \mathrm{Del}(X)$ exists with $f_{\mathrm{Flood}}(\sigma) = t_i$ or $f_{\mathrm{Alpha}}(\sigma) = t_i$. We prove by induction on the t_i.

Induction hypothesis. For any t_i, there exists a sequence of simplicial collapses from $\mathrm{Flood}_r(X, X)$ to $\mathrm{Alpha}_r(X)$.

Induction start. At filtration value $t_0 = 0$, it holds that $\mathrm{Flood}_0(X, X) = X = \mathrm{Alpha}_0(X)$.

Induction step. Assume that there exists a sequence of collapses from $\mathrm{Flood}_{t_i}(X, X)$ onto $\mathrm{Alpha}_{t_i}(X)$. We will modify this sequence to get a sequence of collapses from $\mathrm{Flood}_{t_{i+1}}(X, X)$ onto $\mathrm{Alpha}_{t_{i+1}}(X)$. *For brevity, we will denote $F_t := \mathrm{Flood}_t(X, X)$ and $A_t := \mathrm{Alpha}_t(X)$.*

Given a filtration value t_{i+1} there are three non-exclusive possibilities:

(i) There is $\sigma \in \mathrm{Del}(X)$ such that $f_{\mathrm{Flood}}(\sigma) = t_{i+1}$ but $f_{\mathrm{Alpha}}(\sigma) > t_{i+1}$.

(ii) There is $\sigma \in \mathrm{Del}(X)$ such that $f_{\mathrm{Alpha}}(\sigma) = t_{i+1}$ but $f_{\mathrm{Flood}}(\sigma) < t_{i+1}$.

(iii) There is $\sigma \in \mathrm{Del}(X)$ such that $f_{\mathrm{Flood}}(\sigma) = t_{i+1} = f_{\mathrm{Alpha}}(\sigma)$.

For now, assume that at each filtration value t_{i+1} only one case is true.

Case (i). By Lemma 13, the addition of each such σ to $\mathrm{Flood}_{t_i}(X, X)$ is a regular event. Specifically, $F_{t_{i+1}} = F_{t_i} \cup f^{-1}_{\mathrm{Flood}}(\{t_{i+1}\})$, and the removal of $f^{-1}_{\mathrm{Flood}}(\{t_{i+1}\})$ from $F_{t_{i+1}}$ is a simplicial collapse. Thus we can simply prepend this collapse to the existing sequence of collapses given by the induction hypothesis.

Case (ii). Similarly as above, Lemma 13 implies that the addition of σ to A_{t_i} is regular event and does not change its homotopy type. Specifically, $A_{t_{i+1}} = A_{t_i} \cup f^{-1}_{\mathrm{Alpha}}(\{t_{i+1}\})$, and the removal of $f^{-1}_{\mathrm{Alpha}}(\{t_{i+1}\})$ from $A_{t_{i+1}}$ is a simplicial collapse. Since $F_r \subset A_r$ for any $r \geq 0$, the pair $f^{-1}_{\mathrm{Alpha}}(\{t_{i+1}\})$ was already added to Flood_r at a previous filtration time, thus (see (i)), its removal must be in our sequence of simplicial collapses from F_{t_i} onto A_{t_i}. Thus, the obvious candidate for a sequence from

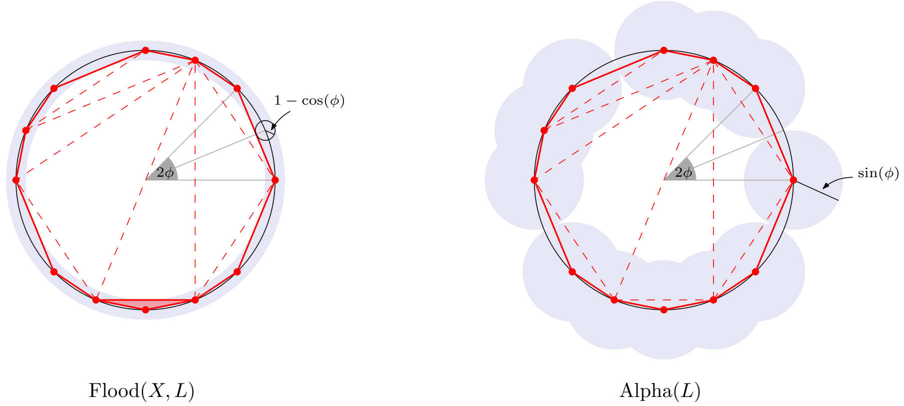
$F_{t_{i+1}}$ onto $A_{t_{i+1}}$ is to take the existing sequence of collapses but to omit collapsing $f^{-1}_{\text{Alpha}}(\{t_{i+1}\})$. Note that collapsing $f^{-1}_{\text{Alpha}}(\{t_{i+1}\})$ can only create free faces that are in A_{t_i}, so the collapses in our sequence from F_{t_i} onto A_{t_i} are not affected by skipping the former. Thus, we have constructed a well-defined sequence of collapses from $F_{t_{i+1}}$ onto $A_{t_{i+1}}$.

Case (iii). The last case is $F_{t_{i+1}} \setminus F_{t_i} = A_{t_{i+1}} \setminus A_{t_i} =: \{\sigma_1 \dots, \sigma_m\}$. The candidate sequence of simplicial collapses from $F_{t_{i+1}}$ onto $A_{t_{i+1}}$, is the sequence F_{t_i} onto A_{t_i} given by the induction hypothesis but applied to $F_{t_{i+1}}$. This is well defined, if any free face of F_{t_i} whose removal is in the existing sequence of collapses is still free after the addition of $f^{-1}_{\text{Flood}}(\{t_{i+1}\})$ to F_{t_i}. However, $\tau \in F_{t_i}$ can only be such a free face if $\tau \notin A_{t_i}$, so by the assumption that $F_{t_{i+1}} \setminus F_{t_i} = A_{t_{i+1}} \setminus A_{t_i}$ no coface of τ is added to F_{t_i} at filtration value t_{i+1}.

To finish the proof, we need to discuss the case when the filtration values are non-unique. By slightly changing the filtration values of the regular events, we get two new filtered complexes $\text{Flood}', \text{Alpha}' \subset \text{Del}(X)$ with homotopy equivalences $\text{Flood}'_r(X, X) \simeq \text{Flood}_r$ and $\text{Alpha}'_r \simeq \text{Alpha}_r$ for all $r \geq 0$ that satisfy the previously made assumption that cases (i) – (iii) are exclusive. □

C.3 Example: the Flood complex of a circle

Herein, we will calculate $\text{Flood}(X, L)$ when $X := \{x \in \mathbb{R}^2 : \|x\| = 1\}$, i.e., the entire unit circle, and L is selected using farthest point sampling (FPS). We will see that the deviation from the (persistent homology of) the thickenings X_r depends only on *curvature* via the sagittas of circular arcs. In contrast, for an Alpha complex computed on L, the deviation depends on the *slope* via their chord lengths. This geometric difference manifests in the different decay of the approximation error in bottleneck distances: *linear* in the number of L for the Alpha complex, but *quadratic* for the Flood complex, enabled by the additional information it can extract from X.



Flood(X, L)

Alpha(L)

Figure 12: Flood(X, L) (**left**) and Alpha(L) (**right**) for X the unit circle and $|L| = 12$ landmarks (●) selected using FPS. The Flood complex matches the underlying topology already at lower filtration values.

Landmark selection. Without loss of generality, we assume that the initial point for FPS is at $(1, 0)$ corresponding to the angular coordinate of 0. FPS then proceeds in stages, where in the k-th stage all points with angular coordinate $2\pi/2^k$ are selected (in arbitrary order), if they have not been selected at an earlier stage. Once all $|L|$ landmarks are selected, the resulting Delaunay triangulation on L is not unique. More specifically, it always contains $|L|$ edges along the circle connecting each landmark to the landmark with next lower and next higher angular coordinates, but the remaining $|L| - 3$ edges (across the circle) are arbitrary.

Structure of persistence diagrams. The persistent homology of Flood(X, L) does not depend on the choice of Delaunay triangulation. In fact, the lifetimes in H_0 are just the lengths of the edges along the circle, which are always in Del(L). Similarly, the only feature in H_1 is born as soon as all these edges are included in Flood$_r(X, L)$, and dies as soon as the entire convex hull of L is flooded, which always occurs at filtration value 1.

Filtration values of edges. We will compute the filtration values of the edges that are relevant for persistent homology, i.e., of those along the circle. Their filtration values are just the distance from an edge's midpoint to the circle X, i.e., the sagitta of the corresponding arc. Denoting the arc length by φ, this is just $1 - \cos(\varphi/2)$. Finally, we observe that there are exactly $m := 2(|L| - 2^{\lfloor \log_2(|L|) \rfloor})$ short arcs of length $\phi := \pi/2^{\lfloor \log_2(|L|) \rfloor}$ and $|L| - m$ long arcs of length 2ϕ. This is because if $|L|$ is a power of 2, then all $|L|$ arcs have length $2\pi/L$, and adding any further landmark splits one long arc and into two short ones, see Figure 12. We conclude that

$$\operatorname{dgm}_0(\mathrm{Flood}_r(X, L)) = \left\{\!\!\left\{ \begin{array}{ll} (0, 1 - \cos(\phi/2)) & m\text{-times} \\ (0, 1 - \cos(\phi)) & (|L| - m)\text{-times} \\ (0, \infty) & \text{once} \end{array} \right\}\!\!\right\} , \tag{15}$$

$$\operatorname{dgm}_1(\mathrm{Flood}_r(X, L)) = \{\!\{(1 - \cos(\phi), 1)\}\!\} . \tag{16}$$

Bottleneck distances. We compare the persistent homology of the Flood complex with that of the thickenings $\{X_r\}_{r \in [0,\infty)}$, which (by a slight abuse of notation) we denote as $\operatorname{dgm}_i(X)$. It is straightforward to see that

$$\operatorname{dgm}_0(X) = \{\!\{(0, \infty)\}\!\} \text{ , and} \tag{17}$$

$$\operatorname{dgm}_1(X) = \{\!\{(0, 1)\}\!\} . \tag{18}$$

Hence, bottleneck distances are just

$$d_B\big(\operatorname{dgm}_0(X), \operatorname{dgm}_0(\mathrm{Flood}_r(X, L))\big) = \frac{1}{2}(1 - \cos(\phi)) \qquad \text{(by matching to the diagonal)} , \tag{19}$$

$$d_B\big(\operatorname{dgm}_1(X), \operatorname{dgm}_1(\mathrm{Flood}_r(X, L))\big) = (1 - \cos(\phi)) . \tag{20}$$

In particular, since $\phi < 2\pi/|L|$, we conclude that all bottleneck distances are smaller ϵ if $2\pi/|L| < \arccos(1 - \epsilon)$, i.e., if we use $|L| > 2\pi/\arccos(1 - \epsilon) \sim 2\pi/\sqrt{2\epsilon}$ landmarks.

Comparison with Alpha complex. We discard the additional information available through X and simply compute an Alpha complex on L. Specifically, the arguments regarding the structure of persistence diagrams still apply, but now filtration values are not the sagitta of an arc, but half the chord length. Hence,

$$d_B\big(\operatorname{dgm}_0(X), \operatorname{dgm}_0(\mathrm{Alpha}_r(L))\big) = \frac{1}{2}\sin(\phi) , \tag{21}$$

$$d_B\big(\operatorname{dgm}_1(X), \operatorname{dgm}_1(\mathrm{Alpha}_r(L))\big) = \sin(\phi) , \tag{22}$$

and $|L| > 2\pi/\arcsin(\epsilon) \sim 2\pi/\epsilon$ are necessary.

D Changelog

D.1 Camera ready revisions / changes from arXiv version v1 to v2

We make the point cloud datasets used for the object classification experiments in Section 5.3 available as part of the `flooder` package. These datasets differ from the previous versions only (except for `mcb-c` and `modelnet10`, see below) in that they use different seeds for sampling the point clouds and also different training/validation/test splits. For reproducibility, we reran all experiments on the now public datasets and updated Tables 2 and 3 accordingly. On the `swisscheese`, `rocks` and `corals` datasets, the new results are (up to statistical uncertainty) in line with our initial results.

In the previous version of the manuscript, we selected a subset of `mcb` by joining MCB-A and MCB-B. This introduced duplicate point clouds (up to scaling and translation) in the dataset, however, with distinct labels. We now use only a subset of MCB-A. For `modelnet10`, we reduced the point clouds from 1M to 250k points to reduce the file size and facilitate sharing the data, and normalized point clouds to have coordinates in $[-1, 1]$. Moreover, we used different hardware for training the LGBM classification model, which uses the given training-time budget of 10 minutes more effectively. As a result, on `mcb-c` and `modelnet10`, balanced accuracies of the PH methods improved significantly while the neural network baselines are still consistent.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide, as claimed, both theoretical results (Section 3) and an experimental validation (Section 5).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the conclusions and in a separate section in the appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The proof of all lemmas and theorems are reported in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all details for the implementation of the complexes and training of the models. Moreover, we provide our source code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code will be publicly available on GitHub, and we provide it to reviewers.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Hyperparameters and optimizers are described in the main part of the paper. Further details can be found in the appendix and the provided source code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For all classification experiments and ablation studies, we report averages with one standard deviation error bars and specify that they are with respect to cross-validation runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computing infrastructure is detailed in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no direct societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Creators or owners of assets are properly credited in the main part of the manuscript. License terms are specified in the supplementary material and license terms are always respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All newly introduced assets are carefully documented, i.e., details of the method are described in Sections 4 and 5 of the manuscript. Furthermore, source code is attached to the submission and will be released publicly in case of acceptance (including datasets used in this work as well as a documentation).

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.