Towards Scalable and Robust Filtration Learning for Point Clouds via Principal Persistence Measure

Anonymous authors
Paper under double-blind review

Abstract

Topological features in persistent homology extracted via a filtration process have been shown to enhance the performance of machine learning tasks on point clouds. The performance is highly related to the choice of filtration, thereby underscoring the critical significance of filtration learning. However, the current supervised filtration learning method for point clouds can not scale well. We identify that this shortcoming stems from the utilization of Persistence Diagrams (PD) for encoding topological features, such as connected components, rings or voids, etc. To address this issue, we propose to use Principal Persistence Measure (PPM), an existing statistical approximation of PD, as an alternative representation and adapt existing network for PPM-based filtration learning. Experimental results on point cloud classification tasks demonstrate that our PPM-based framework achieves comparable accuracy to the PD-based approach while offering better scalability and robustness against outliers.

1 Introduction

Topological information is shown to be effective in machine learning tasks on point clouds across many fields, such as biology (Kovacev-Nikolic et al., 2016; Liu et al., 2022; Meng et al., 2020; Xia et al., 2018; Xia & Wei, 2015) and chemistry (Hiraoka et al., 2016; Lee et al., 2017; Townsend et al., 2020). Topological features of a point cloud are built from a nested sequence of simplicial complexes (Salnikov et al., 2018) called *filtration*. The birth ($b_i \in \mathbb{R}$) and death ($d_i \in \mathbb{R}$) of a cycle, e.g., a ring (1-dimensional cycle) or a void (2-dimensional cycle), of the sequence are encoded in *Persistence Diagram* (PD) $\mathcal{D} = \{r_i = (b_i, d_i) \in \Omega | 1 \leq i \leq N(\mathcal{D})\}$, where **each $r \in \mathcal{D}$ is defined as a topological feature corresponding to the cycle**, $\Omega = \{(t_1, t_2) \in \mathbb{R}^2 | t_2 > t_1\}$ is an open half-plane and $N(\mathcal{D})$ is the number of topological features. PD can be transformed into a vector and fed into a machine learning model as input. A brief illustration of PD-based pipeline for point cloud [1] is shown in Figure 1(a,b,d,f).

In the pipeline, despite different learnable vectorization methods like Carrière et al. (2020); Kim et al. (2020); Reinauer et al. (2021), it is shown in Nishikawa et al. (2023) that the final performance is highly affected by the choice of the filtration. In addition, although there are different unsupervised filtration choices like Rips (Hausmann et al., 1995), DTM (Fasy et al., 2018) and Λ-filter (Zhang et al., 2023), supervised filtration learning (Nishikawa et al., 2023) for point cloud often produces better results. We will refer this PD-based pipeline with filtration learning as PD-FL for short.

Based on weighted filtration, PD-FL (Nishikawa et al., 2023) develops a neural network architecture with isometry-invariance to learn the weight in an end-to-end way. However, since the computation algorithm [2]

[1] Note that this pipeline can also be used for graph data, where the filtration usually employs unsupervised characteristics of a graph like degree (vertex-level), Ricci curvature (edge-level)(Ballester & Rieck, 2023; O'Bray et al., 2021; Southern et al., 2023) or outputs of supervised learnable networks (Hofer et al., 2020; Horn et al., 2022; Immonen et al., 2023; Mukherjee et al., 2024; Zhang et al., 2022). In this paper, we focus on the pipeline for point cloud.

[2] As pointed by Zomorodian & Carlsson (2004), time complexity for computing PD in the worst case is $O(m^3)$, where m is the number of simplices in the filtration. If we want topological feature corresponding to 1-dimensional cycle (ring), we need to consider up to 2-simplices in the complex. The number of simplices is $O(n^3)$ and the computational cost can be up to $O(n^9)$, where n is the size of the point cloud.

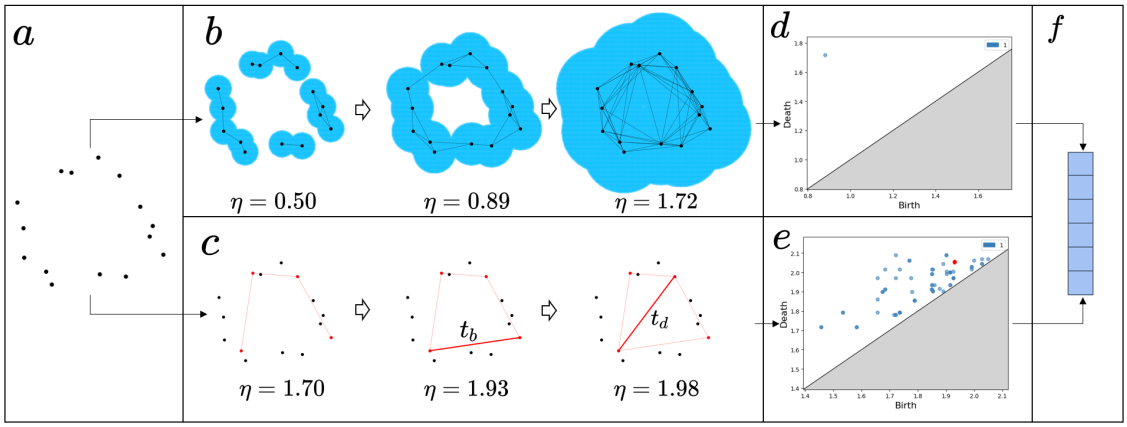

Figure 1: PD (a,b,d,f) and PPM-based (a,c,e,f) pipeline for point cloud. (a) Original point cloud X. Note that this can also be a distance matrix, since in the construction of simplicial complexes of the filtration, a pairwise connection is added if the pairwise distance is lower than a threshold η. This can be equivalently expressed that for balls centered at each point with radius of $\eta/2$, if two balls touch, an edge between two balls' centers is added. (b) An example filtration (Rips filtration) at values of η of X. The union of balls is intended to show the process of the Rips filtration: when two balls touch (pairwise distance $\leq \eta$), an edge is added. According to Nerve theorem (Mona & Hintz, 2023), $X_\eta = \cup_{x \in X} B(x, \eta/2)$ is actually homotopy equivalent to Cěch complex. While the Rips filtration serves as a practical approximation of the Cěch filtration, it does not strictly reflect the topology of the union of balls. (c) An example filtration (Rips filtration) at various values of η of one subset of X. The distances of the bold lines are t_b and t_d. (d) 1-dimensional Persistence Diagram of X. (e) 1-dimensional Principal Persistence Measure of X. **The red point corresponds to the topological feature obtained via the red subset in (c).** (f) Vectorization of PD or PPM.

of filtration-induced PD is highly nontrivial to parallelize and can only be conducted by either pure CPU implementations (Bauer, 2021; Pérez et al., 2021) or a CPU-GPU hybrid one (AIDOS-Lab; Zhang et al., 2020), this network suffers from high time cost and cannot scale for a large point cloud.

In order to address this issue, we propose to replace PD in the pipeline with Principal Persistence Measure (PPM) (Gómez & Mémoli, 2024; Tung et al., 2025), which is a statistical approximation of PD, for the following two reasons: 1) As shown in Tung et al. (2025), PPM can be computed entirely on GPU in a parallel way, making it ideal for a scalable filtration learning framework. 2) PPM can capture topological information in a point cloud. PPM has been used in Tung et al. (2025) for latent space matching in Generative Adversarial Networks (Goodfellow et al., 2020). Although PPM is a rather vague approximation of PD and may not demonstrate all the possible topological features in a point cloud because the sampling mechanism favors points in dense region, the experiments in Figure 2 of Tung et al. (2025) demonstrate that a smaller distance (measured by Persistence Weighted Gaussian Kernel (Kusano et al., 2016) based Maximum Mean Discrepancy) between two PPMs indicates a smaller Wasserstein distance between the PDs. This shows that topological information can be obtained through PPM.

The main contributions of this work are threefold:

1. Propose to use Principal Persistence Measure (PPM) in filtration learning framework (Nishikawa et al., 2023) to address the scalability limitation of PD-based approach and adapt existing framework for PPM-based Filtration Learning (PPM-FL).

2. Establish the theoretical guarantee on the robustness of PPM against outliers and perform corresponding experimental validation.

3. Demonstrate that PPM-FL achieves comparable accuracy to PD-FL while offering better scalability for large point clouds and improved robustness against outliers in point cloud classification tasks on public datasets.

2 Background

We provide the relevant information about Persistence Diagram and weighted filtration (refer to Chazal & Michel (2021) for a comprehensive introduction).

2.1 Persistence Diagram

Let $g : \mathcal{X} \to \mathbb{R}^+$ denote a function on space $\mathcal{X}$, where $\mathbb{R}^+$ stands for positive real numbers, $\mathcal{X}$ is a bounded and open subset of Euclidean space. Function g is computed based on a finite point cloud X sampled from manifold $\mathcal{M} \subset \mathcal{X}$. This manifold assumption means that the finite point cloud X is sampled from an underlying k-dimensional manifold $\mathcal{M}$, where $\mathcal{M} \subset \mathbb{R}^d$ is a smooth, low-dimensional subspace embedded in a higher-dimensional Euclidean space $\mathbb{R}^d$. At scale $\eta \geq 0$, the sublevel set $\mathcal{X}_\eta^g = \{x \in \mathcal{X} \mid g(x) \leq \eta\}$ encodes the topological information in $\mathcal{X}$. For $\gamma \leq \eta$, we can have the nested sublevel sets $\mathcal{X}_\gamma^g \subseteq \mathcal{X}_\eta^g$. By increasing scale η from 0, we obtain a filtration $\mathcal{K}(X) = \{\mathcal{X}_\eta^g\}_{0 \leq \eta < \infty}$, a nested sequence of topological spaces.

A cycle is considered 'born' at $b \in \mathbb{R}$ when it first emerges in $\mathcal{X}_b^g$, and it 'dies' at $d \in \mathbb{R}$ when it ceases to exist in $\mathcal{X}_p^g$ for any $p > d$. 0-dimensional cycles are connected components; 1-dimensional cycles are rings or loops; 2-dimensional cycles are voids, etc. This homology dimension is denoted as q. Topological feature $r = (b, d)$, which corresponds to a cycle, is presented in the form of PD $\mathcal{D} = \{r_i = (b_i, d_i) \in \Omega | 1 \leq i \leq N(\mathcal{D})\}$, where $N(\mathcal{D})$ is the number of topological features. An example using Rips filtration (Hausmann et al., 1995) $g(\cdot) = 2 \min_{x \in X} \ell(\cdot, x)$, where ℓ is Euclidean distance, and corresponding PD are provided in Figures 1(b) and 1(d). In practice, we usually set the maximum threshold for the filtration to be a finite value or just remove the point with infinite death time since it exists for all point clouds and has no practical value.

2.2 Weighted Filtration

In Rips filtration, at scale η, the sublevel set is $\mathcal{X}_\eta = \bigcup_{x \in X} B(x, \eta/2)$, where $B(x, \eta/2)$ is the ball centered at x with radius $\eta/2$. Each ball has the same radius. Weighted filtration aims to put weight on each point's corresponding radius. For weighted filtration, at scale η, the sublevel set is $X_\eta = \bigcup_{x \in X} B(x, \eta - w(x))$, where $w(\cdot) : X \to \mathbb{R}$ is the weight function. Current work (Nishikawa et al., 2023), i.e., PD-FL, designed a network to learn this weight function w. In order to align with the setting in PD-FL and ensure a fair comparison, our work is also based on the weighted filtration.

3 Related Work

3.1 Filtration Learning

Supervised filtration learning is first introduced for graph data [3] by designing learnable vertex filter function (Hofer et al., 2020). It is then studied for more extensive and general purposes on graph data (Horn et al., 2022; Immonen et al., 2023; Mukherjee et al., 2024; Zhang et al., 2022). Filtration learning for point cloud is rarely developed. Existing work (Nishikawa et al., 2023) built a network via weighted filtration, i.e., given a point cloud $X \subset \mathbb{R}^d$, one can define the radius value $r_x(\eta)$ at scale η for $x \in X$ as $r_x(\eta) = \eta - w(x)$ if $\eta > w(x)$, otherwise $r_x(\eta) = -\infty$, where w is the weight function. DTM (Fasy et al., 2018) is a special case of this weighted filtration where w is distance-to-measure function.

It is required in Nishikawa et al. (2023) that for filtration learning, the weight function $w(\cdot) = f(X, \cdot) : \mathbb{R}^d \to \mathbb{R}$ needs to meet the following three conditions:

1. f should be determined by the whole point cloud X and does not depend on the order of points in X;

[3] Filtration Learning on graph data is scalable due to the simplicity of computing PD on graph. In the filtration, the addition of an edge either connects two connected components or creates a ring.

2. f should be isometry-invariant, i.e, for any isometric transformation T, any point cloud X, and $x \in X$, $f(TX, Tx) = f(X, x)$;

3. The output of the $f(X, x)$ should have both of global information X and pointwise information of x.

A network [4] that relies on deepsets (Zaheer et al., 2017) and takes distance matrix as input is designed to satisfy the conditions above. Once the filtration is determined by this network, PD of the entire point cloud is computed and then can be vectorized as input to a Multi-Layer Perceptron for classification task. Although this network outperforms unsupervised filtration like Rips and DTM, it relies on CPU for the computation of PD and the substantial time cost (Zomorodian & Carlsson, 2004) for computing PD prevents its scalability.

3.2 Principal Persistence Measure

One way to reduce the time cost of computing PD of the entire point cloud is to use statistical approximation: for multiple random subsets [5] (with fixed size) of the entire point cloud X, a PD $\mathcal{D}_i = \{r_j = (b_j, d_j) \in \Omega | 1 \leq j \leq N(\mathcal{D}_i)\}$ is obtained for each random subset $X_s^i \subset X$. The statistical approximation, i.e., Expected PD (EPD) (Chazal & Divol, 2018), takes a distributional view and represents each PD of subset X_s^i as a measure $\mu_i = \sum_{j=1}^{N(\mathcal{D}_i)} \delta_{r_j}$ supported on Ω where δ_{r_j} is Dirac point mass at r_j. The empirical EPD is then the average $\bar{\mu} = \frac{1}{M} \sum_{i=1}^{M} \mu_i$, where M is the number of random subsets.

Principal Persistence Measure (PPM) (Gómez & Mémoli, 2024) is a special case of Expected PD where each subset X_s^i has fixed size $2q + 2$ where q is the homology dimension ($q = 0$ for connected component, $q = 1$ for rings, etc.). It is guaranteed that each PD on $2q + 2$ points has at most one single topological feature $r_1 = (t_b, t_d)$ when $q \geq 1$, i.e., $|N(\mathcal{D}_i)| \leq 1$, which can be efficiently computed as follows: given any $x \in X_s^i$, let $x^{(1)}, x^{(2)} \in X_s^i$ be the points such that $d(x, x^{(1)}) \geq d(x, x^{(2)}) \geq d(x, a)$ for any $a \in X_s^i \setminus \{x^{(1)}, x^{(2)}\}$ where d is a distance function (filtration), then

$$r_1 = (\max_{x \in X_s^i} d(x, x^{(2)}), \min_{x \in X_s^i} d(x, x^{(1)})).$$

PPMs have stability with respect to Wasserstein distance, as shown in Theorem A.2. The computation of PPM can be easily implemented in a parallel way and conducted on GPU and the existence of r_1 is discussed in Theorem A.3. By considering random subsets, the computation of PPM costs less time than computing PD on the entire point cloud.

In order to improve scalability, we propose to use PPM, instead of PD, to encode topological information and adapt existing network (Nishikawa et al., 2023) for PPM-based filtration learning, which learns from multiple subsets.

4 Filtration Learning for PPM

We propose the filtration learning framework for PPM (PPM-FL) based on a weighted filtration. PPM-FL, which concentrates on learning from multiple subsets, is adapted from the network in Nishikawa et al. (2023) for learning from the entire point cloud.

Different from the three conditions mentioned in Nishikawa et al. (2023), here weight function w should be related to $\mathbb{X}_s = \cup_{i \in [M]} \{X_s^i\}$, with each subset $X_s^i \subset X$ of size $2q + 2$, where q is the homology dimension, instead of the whole point cloud X. This gives the following three adapted requirements for the weight function $w(\cdot) = \bar{f}(\mathbb{X}_s, \cdot)$:

1. The output of $\bar{f}(\mathbb{X}_s, x)$ should be determined by each subset X_s^i and does not depend on the order of the points. The closer a subset X_s^i is to x, the greater its impact on x should be.

[4] The full architecture of network (Nishikawa et al., 2023) is shown in Appendix A.5.
[5] Each point in the subset is i.i.d. sampled from the entire point cloud.

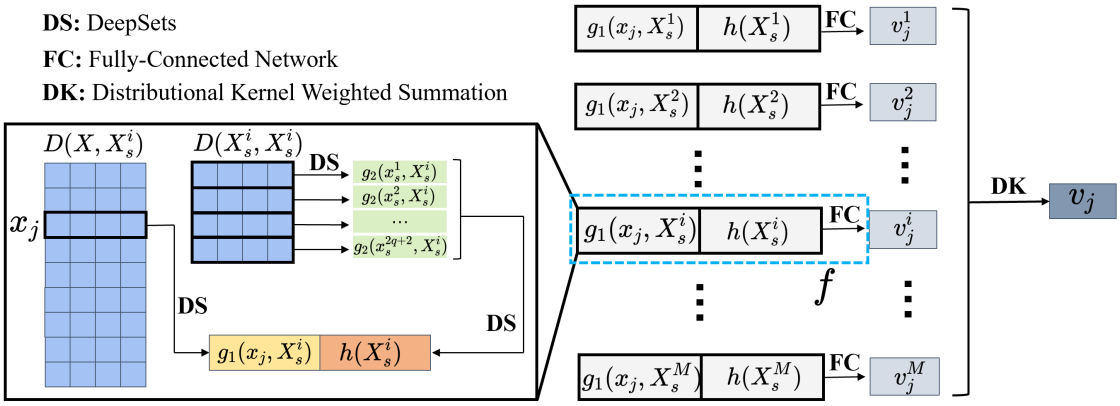

Figure 2: PPM-based Filtration Learning Framework. Value v_j is the output of $\bar{f}(\mathbb{X}_s, x_j)$. D represents a distance matrix for two point clouds X and Y, $D(X,Y) = (d(x_i, y_j))_{i=1,j=1}^{i=|X|,j=|Y|} \in \mathbb{R}^{|X|\times|Y|}$, where d is Euclidean distance.

2. $\bar{f}$ should be isometry-invariant, i.e., for any isometric transformation T, any point cloud X, $\mathbb{X}_s$ and $x \in X$, $\bar{f}(T\mathbb{X}_s, Tx) = \bar{f}(\mathbb{X}_s, x)$, where $T\mathbb{X}_s \triangleq \cup_{i\in[M]}\{TX_s^i\}$

3. The output of $\bar{f}(\mathbb{X}_s, x)$ should have both of global information $\mathbb{X}_s$ and pointwise information of x.

In order to deal with set $\mathbb{X}_s = \cup_{i\in[M]}\{X_s^i\}$, we model the weight function $\bar{f}$ in a simple summation form, i.e., $\bar{f}(\mathbb{X}_s, \cdot) = \sum_{i=1}^{M} K(X_s^i, \cdot) f(X_s^i, \cdot)$, where f is used to output the weight (v_j^i) of the j-th point x_j in X w.r.t. subset X_s^i, i.e., $v_j^i = f(X_s^i, x_j)$. Function K measures the similarity between the input and subset X_s^i to meet requirement 1: the closer X_s^i is to x, the greater its impact on x. The filtration learning framework for PPM, as shown in Figure 2, is then adapted from Nishikawa et al. (2023) to meet the three new requirements:

1. Following Nishikawa et al. (2023), the independence on the order of points is guaranteed by following DeepSets (Zaheer et al., 2017) architecture g_1, g_2 and h:

$$g_1(x_j, X_s^i) = \phi^{(2)}(\mathbf{op}(\{\phi^{(1)}(d(x, x_j)) | x \in X_s^i\}));$$

$$g_2(x_s^k, X_s^i) = \phi^{(4)}(\mathbf{op}(\{\phi^{(3)}(d(x, x_s^k)) | x \in X_s^i\}));$$

$$h(X_s^i) = \phi^{(5)}(\mathbf{op}(\{g_2(x, X_s^i) | x \in X_s^i\})),$$

   where x_s^k is the k-th [6] point in a subset X_s^i, d is Euclidean distance, $\phi^{(i)}$s are all fully-connected neural networks and **op** is the permutation invariant operator.

   For each $x_j \in X$, after the weight v_j^i w.r.t. the i-th subset X_s^i is obtained, the final weight value v_j is the distributional kernel weighted summation of each v_j^i, i.e., $v_j = \sum_{i=1}^{M} K(x_j, X_s^i) v_j^i$, where $K(x_j, X_s^i) = \frac{1}{|X_s^i|}\sum_{x\in X_s^i} \kappa(x, x_j)$ and $\kappa(x, x_j) = \exp(\frac{-\|x-x_j\|_2^2}{2\sigma^2})$.

2. The isometry-invariance is guaranteed by using the distance matrix as input.

3. The global and pairwise information are stored in h and g_1 respectively.

[6] This index k is used to differentiate two different point in subset X_s^i. We just assume a random order here since the framework is permutation invariant.

Once all the weights are obtained, the weighted filtration is used to compute the PPM of the point cloud [7]. The space cost of PPM-FL is $O((2q+2)nM + (2q+2)^2M)$, where q is the homology dimension, M is the number of subset in PPM and n is the point cloud size.

Proposition 4.1 (Time Complexity Comparison)**.** *Let n denote the point cloud size, M the number of random subsets, q the homology dimension, and t_ϕ the time for a single forward pass of a fully-connected network $\phi^{(i)}$. Then:*

1. *The per-sample time complexity of PPM-FL is $O(M(2q+2)(n+(2q+2))t_\phi + M(2q+2)^2)$. The first term accounts for the network computing M weight values for each of the n points (via distance matrices of size $(2q+2)\times n$ and $(2q+2)\times(2q+2)$), and the second term accounts for computing the PPM topological feature for each subset.*
2. *The per-sample time complexity of PD-FL is $O(n^2 t_\phi + T_{PD}(n,q))$, where $T_{PD}(n,q)$ is the cost of computing the q-dimensional PD. In the worst case, $T_{PD}(n,q) = O(n^{3(q+2)})$ (Zomorodian & Carlsson, 2004), because for homology dimension q, we need to consider the number of $q+1$-d simplices, which contains $q+2$ points.*

Since $2q+2$ is a small constant (e.g., 4 for $q=1$), the PPM-FL complexity is effectively $O(Mnt_\phi)$, which scales linearly in n for fixed M. In contrast, PD-FL scales at least as $O(n^2 t_\phi)$ for the network and up to $O(n^9)$ for computing 1-dimensional PD.

Remark 4.2 (End-to-End GPU Training). A practical advantage of PPM-FL beyond time complexity is that the entire forward and backward pass runs on GPU. In PD-FL, the PD computation and its gradient (AIDOS-Lab) require CPU–GPU data transfer at every training iteration, creating a communication bottleneck. In PPM-FL, the topological feature $r_1 = (t_b, t_d)$ of each subset is computed via differentiable max and min operations on pairwise distances, so gradients flow through the PPM computation natively on GPU without any CPU involvement.

As for the approximation ability, for weight function $\bar{f}(\mathbb{X}_s, \cdot) = \sum_{i=1}^{M} K(X_s^i, \cdot) f(X_s^i, \cdot)$, function f is able to approximate any continuous function, as shown in Theorem 4.1 of Nishikawa et al. (2023).

PPM can be then vectorized [8] by a supervised method PersLay (Carrière et al., 2020) for a potential task. For example, if the task is point cloud classification, we can input the resulting vector into MLP that employs a cross entropy loss function. Due to the parallel implementation of PPM, this PPM-based pipeline can be deployed on GPU to scale on a large point cloud.

5 Robustness of PPM against Outliers

Here we demonstrate the robustness of PPM against outliers. We start by considering the robustness of a general case of PPM, i.e., Expected Persistence Diagram (EPD), where the size of random subset X_s is denoted as n, and then extend the result to PPM. Let $\mathcal{M}$ denote the underlying manifold of point cloud X; P denote the density of the distribution on k-dimensional $\mathcal{M}$; and U denote the density of the distribution of outliers, such as a uniform distribution. Following the setting in Vishwanath et al. (2020); Cai et al. (2025), the outlier-contaminated distribution P_o is expressed as

$$P_o = (1-\epsilon)\cdot P + \epsilon \cdot U,$$

where ϵ is the percentage of outliers. X is a sample of P_o. Then we have the following lemma on the robustness of EPD under the assumption that filtration $\mathcal{K}$ meets K1-K5 requirements in Section 3 of Chazal & Divol (2018).

We have the following assumptions on P and U:

[7] Note that we could use the framework in Nishikawa et al. (2023), which uses the distance matrix of the entire point cloud to learn the weights. But the space cost would be $O(n^2)$, which is one order of magnitude higher than PPM-FL's $O((2q+2)nM + (2q+2)^2M)$.

[8] Despite many methods for vectorization for PD or PPM, including supervised (Carrière et al., 2020; Kim et al., 2020; Reinauer et al., 2021) and unsupervised ones (Adams et al., 2017; Bubenik et al., 2015; Chung & Lawson, 2022), we choose to use PersLay (Carrière et al., 2020) , a supervised vectorization that uses similar structure like Deepsets for simplicity.

1. We assume that P and U share the same support. Regarding the manifold's support, U is a (maybe uniform) distribution over the entire support of $\mathcal{M}$, including regions where the inlier density P are very low and points have little chance to be sampled from this low density area. An example is shown in Figure 16 in Gómez & Mémoli (2024). The support of P is the circular area and the low density region lies at the inner part of the ring.

2. P and U are both lower and upper bounded in the support, i.e. there exist constants $\bar{c} \geq \underline{c} > 0$ such that $\bar{c} \geq P(x) \geq \underline{c}$ and $\bar{c} \geq U(x) \geq \underline{c}$ for any x in the support. This assumption is identical to assumption 2(ii) in Cai et al. (2025), which shares the same contamination model $P_o = (1-\epsilon) \cdot P + \epsilon \cdot U$.

Lemma 5.1. *For EPD $\mathbb{E}_{X_s \sim P^n}[\nu(X_s)]$ with density μ_1 and $\mathbb{E}_{X_s \sim P_o^n}[\nu(X_s)]$ with density μ_2, where $\nu(X_s)$ is the measure form of PD $\mathcal{D}(\mathcal{K}(X_s))$, i.e., $\nu(X_s) = \sum_{r \in \mathcal{D}(\mathcal{K}(X_s))} \delta_r$, it holds that*

$$\|\mu_1 - \mu_2\|_1 \leq C_n H_k(\mathcal{M})^n p_{n-1}(\epsilon)\epsilon,$$

where C_n is the expected number of points in the PD built with the filtration $\mathcal{K}$ on n i.i.d. points on $\mathcal{M}$, $H_k(\mathcal{M})$ is k-dimensional Hausdorff measure [9] of $\mathcal{M}$ and $p_{n-1}(\epsilon)$ is a polynomial of order $n-1$ with bounded coefficients.

The proof is provided in Appendix B.

Theorem 5.2. *For PPM $\mathbb{E}_{X_s \sim P^{2q+2}}[\nu(X_s)]$ with density μ_1 and $\mathbb{E}_{X_s \sim P_o^{2q+2}}[\nu(X_s)]$ with density μ_2, it holds that*

$$\|\mu_1 - \mu_2\|_1 \leq H_k(\mathcal{M})^{2q+2} p_{2q+1}(\epsilon)\epsilon,$$

where $H_k(\mathcal{M})$ is k-dimensional Hausdorff measure of $\mathcal{M}$ and $p_{2q+1}(\epsilon)$ is a polynomial of order $2q+1$ with bounded coefficients.

Proof. Following Lemma 5.1, PPM is a special case of EPD where the size of each subset is $n = 2q+2$, q is homology dimension. Combined with the fact that each subset of size $2q+2$ has at most 1 topological feature, i.e., $C_n \leq 1$, we can have the result above. □

Theorem 5.2 indicates that a small number of outliers in the point cloud will not severely disturb PPM. The relation between the upper bound and outlier distribution U is discussed in Appendix B.3. The experimental demonstration of PPM's robustness with the learned filtration in point cloud classification task is provided in Section 6.2. We discuss the relation between Theorem 5.2 and existing results in Divol & Lacombe (2021b) in Appendix D.4.

6 Experiments

We compare PPM-FL with PD-FL [10] on the protein (Kovacev-Nikolic et al., 2016) and ModelNet10 (Wu et al., 2015) datasets used previously in similar evaluations (Nishikawa et al., 2023). We also conduct an ablation study to compare PPM-FL with the unsupervised Rips filtration to demonstrate the effectiveness of PPM-FL. PPM-based approach can not use DTM since there are only $2q+2$ points in each random subset. In addition, we validate the robustness and scalability of PPM-FL. **In summary, PPM-FL achieves comparable classification accuracy to PD-FL while offering better robustness to outliers and superior scalability for large point clouds.**

All the classification results are obtained through 3-fold cross validation. All the experiments are conducted on a Ubuntu 20.04 system with 2TB RAM, AMD EPYC 7763 64-Core 1500 MHZ CPU and NVIDIA A6000

[9] Hausdorff measure is a generalization of the traditional notions of area and volume to non-integer dimensions. Let k be a non-negative integer. For $A \subset \mathcal{M}$, and $\delta > 0$, consider $H_k^\delta(A) = \inf\{\sum_i \alpha(k)(\frac{\text{diam}(U_i)}{2})^k, A \subset \bigcup_i U_i \text{ and } \text{diam}(U_i) < \delta\}$, where $\alpha(k)$ is the volume of the k-dimensional unit ball and *diam* represents diameter. The k-dimensional Hausdorff measure on $\mathcal{M}$ of A is defined by $H_k(A) = \lim_{\delta \to 0} H_k^\delta(A)$.

[10] Here PD-FL means the PD-based Filtration Learning (Nishikawa et al., 2023). The experiments on comparing PD-based filtration learning (PD-FL) with Rips (PD-Rips) and DTM (PD-DTM) filtration for PD have already been conducted in Nishikawa et al. (2023). PD-Rips produces similar results to that of PD-DTM. And PD-FL outperforms PD-Rips and PD-DTM. Hence, we just use PD-FL as baseline in our work.

GPU. The network is implemented with PyTorch 2.0.1. The code implementation of PD-FL is from `https://github.com/git-westriver/FiltrationLearningForPointClouds`. The details of the experiment setting are provided in Appendix C. Code is provided at `https://anonymous.4open.science/r/PPM-FL-415C`.

6.1 Comparison with PD-FL

Following the same setting in Nishikawa et al. (2023) on the protein dataset, we employ the pipeline shown in Figure 1 with filtration learning, which uses the topological information only for classification. The results are shown in Table 1. The accuracies of PPM-FL over different homology dimensions are significantly higher than PPM-Rips [11] and the standard deviation of PPM-FL is lower. This comparison with PPM-Rips demonstrates the advantage of the supervised filtration learning over the unsupervised filtration like Rips.

Table 1: Accuracy of the binary classification task of protein structure. We compared our method PPM-FL with PPM-Rips and PD-FL. PPM-Rips denotes using the Rips filtration instead of the filtration learning for PPM. The PersLay vectorization is learned in an end-to-end way. Homology dimension q stands for the dimension of PDs (PPMs). $q = 0\&1$ means that we use both 0-dimensional and 1-dimensional PD or PPM, and the feature input to the MLP is the concatenation of the vectorization of 0-dimensional and 1-dimensional PDs (PPMs) via PersLay.

	$q = 0$	$q = 1$	$q = 0\&1$
PD-FL	65.80 ± 1.76	82.10 ± 1.66	81.70 ± 1.31
PPM-Rips	64.09 ± 7.78	65.40 ± 2.12	71.50 ± 1.31
PPM-FL	84.00 ± 1.39	80.20 ± 1.76	84.60 ± 1.22

When compared with PD-FL, PPM-FL produces better results in $q = 0$ and $q = 0\&1$, but slightly worse in $q = 1$. The large improvement of PPM-FL over PD-FL at $q = 0$ (84.00 vs. 65.80) suggests that for connected components, the statistical sampling in PPM may act as a regularizer that captures the dominant topological structure more robustly. In contrast, for $q = 1$ (rings), PD-FL retains an advantage, likely because the precise ring structure benefits from exact computation on the entire point cloud. Overall, both approaches are effective in extracting meaningful topological features from the protein dataset for classification, suggesting that the key aspect for achieving good performance in this context is not solely the precise topological information but rather the ability to appropriately integrate topological information with the classification model.

For the ModelNet10 dataset, we use the version which contains 10 classes, with 100 instances in each class. Each instance is a point cloud of shape 128×3. Here we employ the same two-phase process in Nishikawa et al. (2023) by combining topological embedding with a DNN-based method. For a point cloud X, let $\Psi_{\mathrm{topo}}(X) \in \mathbb{R}^{L_1}$ be the topological embeddings from PD-FL or PPM-FL, $\Psi_{\mathrm{DNN}}(X) \in \mathbb{R}^{L_2}$ be the feature from a DNN-based method. Let ℓ be the loss function and m be the number of classes. The two-phase training classifiers proposed by Nishikawa et al. (2023) are specified as follows:

1. Phase 1 classifier receives feature from Ψ_{DNN} as $C_1 : \mathbb{R}^{L_2} \to \mathbb{R}^m$, where m is the number of classes. C_1 and Ψ_{DNN} are jointly learned by minimizing $\sum_j \ell(C_1(\Psi_{\mathbf{DNN}}(X_j), y_j)$.

2. Phase 2 classifier $C_2 : \mathbb{R}^{L_1+L_2} \to \mathbb{R}^m$. We fix the parameters of the learned Ψ_{DNN} in the first phase and learn C_2 and Ψ_{topo} by minimizing $\sum_j \ell(C_2([\Psi_{\mathrm{DNN}}(X_j)^\intercal, \Psi_{\mathrm{topo}}(X_j)^\intercal]^\intercal), y_j)$.

The final classification is conducted through C_2 with the concatenated features from Ψ_{DNN} and Ψ_{topo} on the test set. For the choices of DNN method, we consider DeepSets (Zaheer et al., 2017), PointNet (Qi et al., 2017) and PointMLP (Ma et al., 2022). The results of the two-phase process are shown in Table 2.

[11] In the original PD-FL work (Nishikawa et al., 2023), it has been demonstrated that the Rips and DTM yield comparable results when applied to both the protein and the ModelNet10 dataset. For the sake of simplicity and to streamline our analysis, we will solely compare our proposed method with Rips.

Table 2: Accuracies for the classification task on the ModelNet10 dataset. Two-phase training process is utilized here. The first phase we use DeepSets and PointNet. The results of PointMLP are discussed in Appendix D.1. The first phase directly uses point cloud as input and does not have the notion of homology dimension since it does not involve topological summary like PD or PPM. So there is no result for each homology dimension in the first phase.

1st Phase	2nd Phase			
DeepSets		PD-FL	PPM-Rips	PPM-FL
	$q = 0$	67.40 ± 2.31	67.30 ± 2.41	67.90 ± 3.01
66.23 ± 3.19	$q = 1$	68.20 ± 2.37	67.50 ± 2.17	66.90 ± 2.17
	$q = 0\&1$	67.40 ± 0.82	67.10 ± 1.42	67.50 ± 2.88
PointNet		PD-FL	PPM-Rips	PPM-FL
	$q = 0$	68.60 ± 3.09	67.10 ± 2.21	68.70± 2.23
67.23 ± 1.80	$q = 1$	68.90 ± 2.64	67.10 ± 2.21	69.60 ± 1.33
	$q = 0\&1$	69.00 ± 5.57	67.30 ± 2.02	69.80 ± 0.77

When using DeepSets in the first phase, PD-FL, PPM-Rips and PPM-FL can all improve the classification accuracy in the second phase. Except for the case of $q = 1$ where PD-FL outperforms PPM-Rips and PPM-FL, PPM-FL achieves comparable results with PD-FL and PPM-Rips. Note that the differences between PPM-FL and PD-FL are within standard deviations in most settings. This demonstrates the benefit of topological information when combined with the DNN method, and suggests that PPM captures sufficient topological signal for classification despite being an approximation.

When using PointNet in the first phase, PPM-Rips makes no improvement on the classification accuracy. PPM-FL produces comparable results to that of PD-FL: both of them improve the accuracy in the second phase. The highest accuracy (69.80 ± 0.77) is obtained via PointNet + PPM-FL with homology dimensions $q = 0\&1$.

The additional results of PointMLP are discussed in Appendix D.1. The ablation study of the Gaussian weight is provided in Appendix D.3. Table 7 shows the effectiveness of our proposed Gaussian weight: by prioritizing nearer subsets, the Gaussian weight effectively amplifies meaningful topological features, thereby enhancing the method's performance in capturing the intrinsic structure of the data.

It is worth noting that using all the homology dimensions is a generally reliable choice to get high accuracy. Hence we will use all the homology dimensions in the following experiments.

Note on statistical significance. The accuracy differences between PPM-FL and PD-FL on ModelNet10 are generally within standard deviations (e.g., 67.50 ± 2.88 vs. 67.40 ± 0.82 for DeepSets with $q = 0\&1$), which is expected given the 3-fold cross-validation on a relatively small dataset (1000 instances). We emphasize that the primary contribution of PPM-FL is not to surpass PD-FL in accuracy but to achieve comparable classification performance while providing substantially better scalability for large point clouds and improved robustness against outliers.

6.2 Robustness

Here we show the robustness of PPM against outliers in the point cloud. For the ModelNet10 dataset, we use the model trained on outlier-free point clouds; and for each point cloud in the test set, we add outliers from uniform distribution at percentage ϵ. We choose the two-phase process where the first phase uses DeepSets and the second phase uses all the homology dimensions $q = 0\&1$, since in this case PPM-FL and PD-FL produces similar results when there is no outliers on the test set, ensuring a fair comparison.

The average accuracies of PPM-FL and PD-FL under different outlier percentages ϵ are shown in Figure 3. PPM-FL and PD-FL produce similar results when the number of outliers is small, i.e., $\epsilon \leq 5.0\%$. When the number of outliers increases, i.e., $\epsilon > 5.0\%$, the accuracy of PD-FL drops quickly while PPM-FL remains

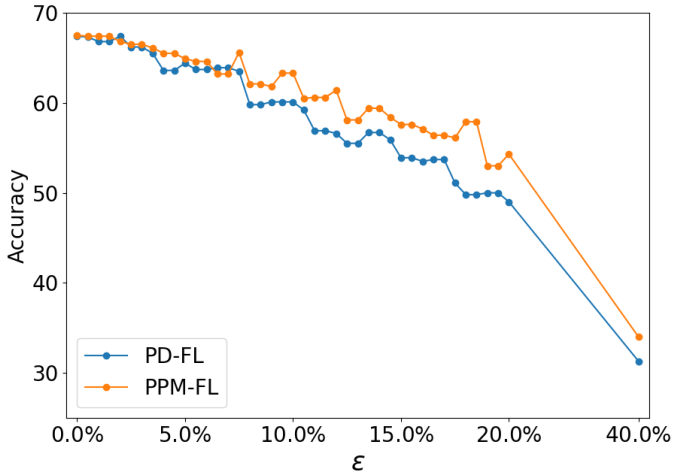

Figure 3: Average classification accuracy under different outlier percentages (ϵ). The actual accuracies, corresponding standard deviations and analysis are provided in Appendix D.5.

accuracy above 55% when $\epsilon \leq 18.5\%$. This result validates the robustness against outliers of PPM shown in Section 5.

6.3 Scalability w.r.t. Point Cloud Size

We demonstrate the scalability of PPM-FL w.r.t. the point cloud size n. The settings in each phase here are the same as that in last subsection on robustness. The number of the point clouds in the dataset is fixed. We consider both homology dimensions 0 and 1 in the second phase. We fix the batch size and report the average time of each epoch in the second phase in Table 3. In the first phase, we use the same pretrained model for PPM-FL and PD-FL.

Table 3: Actual time cost (s) per-epoch under different numbers n of points in the point cloud and numbers of random subsets M in PPM. Our aim is to compare the real-world usage of these methods on the ModelNet10 dataset. Thus, PPM-FL is conducted on GPU and PD-FL is on pure CPU or CPU-GPU Hybrid. The computation of PD-FL (CPU) is through the GUDHI library (Maria et al., 2014). The computation of PD-FL (CPU-GPU Hybrid) is through the *torch-topological* library (AIDOS-Lab).

	PPM-FL (GPU)			PD-FL (CPU)	PD-FL (Hybrid)
	M=100	M=200	M=400		
n=64	56.61 ± 1.08	112.79 ± 1.41	210.17 ± 2.50	41.46 ± 2.55	5.59 ± 0.23
n=128	58.91 ± 3.11	118.79 ± 1.22	218.26 ± 1.35	187.56 ± 2.87	29.46 ± 2.42
n=256	65.85 ± 1.04	129.19 ± 1.15	239.42 ± 1.01	1101.85 ± 11.46	157.95 ± 1.95
n=512	77.95 ± 1.77	154.32 ± 1.58	273.47 ± 1.26	7556.32 ± 66.11	607.95 ± 13.20

For PPM-FL, when the number of random subsets M is fixed, the time cost increases very modestly as n grows. When the number of points in each point cloud n is fixed, the time cost is linear w.r.t. M.

Compared with PD-FL (CPU), PPM-FL's time cost is similar to that of PD-FL (CPU) when point cloud size n is very small. PD-FL (Hybrid) is one order of magnitude faster than PD-FL (CPU). However, the time cost of PD-FL, whether on CPU or hybrid, increases very rapidly when n grows. At $n = 512$, PPM-FL (GPU, $M = 400$) takes 273.47s while PD-FL (Hybrid) takes 607.95s, demonstrating PPM-FL's superior scalability for large point clouds.

When to prefer PPM-FL over PD-FL. We note that for small point clouds ($n \leq 128$), PD-FL (Hybrid) is faster than PPM-FL due to the overhead of processing M subsets. The crossover occurs around $n = 256$,

where PPM-FL (GPU, $M = 100$) reaches parity with PD-FL (Hybrid). For $n \geq 512$, PPM-FL is clearly faster. In practice, PPM-FL is recommended when point clouds are large or when GPU-only computation is desired. For small point clouds where a CPU-GPU hybrid setup is available, PD-FL (Hybrid) remains a competitive option.

Table 4: Accuracies of PPM-FL (GPU) under different Ms ($n = 128, q = 0\&1$) on the ModelNet10 dataset with the 1st phase model being DeepSets.

$M = 25$	$M = 50$	$M = 100$	$M = 200$	$M = 400$
67.00 ± 2.53	67.00 ± 1.43	67.60 ± 2.20	67.50 ± 2.88	67.40 ± 1.92

Table 5: Accuracies of PPM-FL (GPU) under different ns ($M = 100, q = 0\&1$) on the ModelNet10 dataset with the 1st phase model being DeepSets.

$n = 128$	$n = 256$	$n = 512$
67.60 ± 2.20	71.65 ± 1.84	69.40 ± 2.94

Some selected accuracy results corresponding to Table 3 are shown in Table 4 and 5. These results explicitly demonstrate that scalability of PPM-FL is achieved without sacrificing accuracy. And according to Table 4, empirically, if $M(2q + 2)$ is close to or smaller than n, i.e. only a small subset of the entire point cloud is sampled, the performance of PPM-FL will degrade. Hence, a default choice of M would be an integer larger than $n/(2q + 2)$.

6.4 Discussion

On absolute accuracy. We note that the absolute classification accuracies on ModelNet10 (∼67–70%) are modest compared to state-of-the-art 3D point cloud methods that use full-scale architectures. However, following the setting of Nishikawa et al. (2023), the experimental setup uses only 100 instances per class with 128 points each, which is a deliberately small-data regime designed to evaluate the contribution of topological features rather than compete with large-scale 3D recognition methods. In this regime, PPM-FL consistently matches or exceeds PD-FL while being more scalable.

On the strong protein $q = 0$ result. The large accuracy gap between PPM-FL (84.00) and PD-FL (65.80) at $q = 0$ on the protein dataset is noteworthy. We hypothesize that for connected components ($q = 0$), PPM's random subsampling acts as an implicit regularizer: by computing PDs on many small subsets rather than the entire point cloud, PPM captures the dominant connectivity patterns while being less sensitive to noise in individual pairwise distances. Since the protein dataset involves noisy distance matrices (with added uniform noise of standard deviation 0.1), this regularization effect may be particularly beneficial.

7 Conclusion

In this study, we propose to use PPM to replace PD in a filtration learning framework. PPM-based filtration learning (PPM-FL) addresses the scalability limitations of existing PD-based approach for point clouds. By leveraging PPM, which can be computed entirely on GPU in a parallel manner, we achieved a more efficient solution for encoding topological features.

Our theoretical analysis establishes the robustness of PPM against outliers, and we experimentally validate this property in the context of supervised filtration methods. The results show that PPM-FL maintains more stable performance than PD-FL when the test point cloud is contaminated with outliers.

Limitation. The approximation nature of PPM may lead to the loss of some fine-grained topological information. Future work could explore ways to enhance the representational power of PPM while maintaining its computational efficiency. In addition, Theorem 5.2 on robustness has practical limitations, as its dependence

on q makes it weak for higher homology dimension. This leads to the result that the fraction of noise has to be very small to control the measure errors. Our experimental results in Figure 3 align with this: while PPM-FL is not highly robust, it degrades more gradually than PD-FL as outliers increase (maintaining accuracy above 55% when $18.5\% \geq \epsilon > 5\%$), showing a modest improvement.

Broader Impact Statement

This work introduces a scalable and robust framework for filtration learning on point clouds using Principal Persistence Measure. By enabling efficient topological data analysis on larger datasets and demonstrating robustness to outliers, our method has potential applications in diverse fields ranging from bioinformatics (e.g., analyzing protein structures) to computer vision (e.g., 3D object classification). We are not aware of any specific negative societal impacts that would directly stem from this work.

References

Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

AIDOS-Lab. Pytorch-topological: A topological machine learning framework for pytorch. URL `https://github.com/aidos-lab/pytorch-topological?tab=readme-ov-file.`

Rubén Ballester and Bastian Rieck. On the expressivity of persistent homology in graph learning. *arXiv preprint arXiv:2302.09826*, 2023.

Ulrich Bauer. Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, 2021.

Mireille Boutin and Gregor Kemper. On reconstructing configurations of points in p 2 from a joint distribution of invariants. *Applicable Algebra in Engineering, Communication and Computing*, 15:361–391, 2005.

Peter Bubenik et al. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16(1):77–102, 2015.

Yuchao Cai, Hanfang Yang, Yuheng Ma, and Hanyuan Hang. Bagged regularized k-distances for anomaly detection. *Journal of Machine Learning Research*, 26(178):1–59, 2025.

Yueqi Cao and Anthea Monod. Approximating persistent homology for large datasets. *CoRR*, abs/2204.09155, 2022. URL `https://doi.org/10.48550/arXiv.2204.09155`.

Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pp. 2786–2796. PMLR, 2020.

Frédéric Chazal and Vincent Divol. The density of expected persistence diagrams and its kernel based estimation. In *SoCG 2018-Symposium of Computational Geometry*, 2018.

Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, 4:667963, 2021.

Yu-Min Chung and Austin Lawson. Persistence curves: A canonical framework for summarizing persistence diagrams. *Advances in Computational Mathematics*, 48(1):6, 2022.

Vincent Divol and Théo Lacombe. Estimation and quantization of expected persistence diagrams. In *International Conference on Machine Learning*, pp. 2760–2770. PMLR, 2021a.

Vincent Divol and Théo Lacombe. Understanding the topology and the geometry of the space of persistence diagrams via optimal partial transport. *Journal of Applied and Computational Topology*, 5(1):1–53, 2021b.

Brittany Fasy, Fabrizio Lecci, Larry Wasserman, et al. Robust topological inference: Distance to a measure and kernel distance. *Journal of Machine Learning Research*, 18(159):1–40, 2018.

Alessio Figalli. The optimal partial transport problem. *Archive for Rational Mechanics and Analysis*, 195 (2):533–560, 2010.

Mario Gómez and Facundo Mémoli. Curvature sets over persistence diagrams. *Discrete & Computational Geometry*, 72(1):91–180, 2024.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

Jean-Claude Hausmann et al. On the vietoris-rips complexes and a cohomology theory for metric spaces. *Annals of Mathematics Studies*, 138:175–188, 1995.

Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G. Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016. doi: 10.1073/pnas.1520877113. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1520877113`.

Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *International Conference on Machine Learning*, pp. 4314–4323. PMLR, 2020.

Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten Borgwardt. Topological graph neural networks. In *International Conference on Learning Representations*, 2022.

Johanna Immonen, Amauri Souza, and Vikas Garg. Going beyond persistent homology using persistent homology. *Advances in Neural Information Processing systems*, 36:63150–63173, 2023.

Kwangho Kim, Jisu Kim, Manzil Zaheer, Joon Kim, Frédéric Chazal, and Larry Wasserman. Pllay: Efficient topological layer based on persistent landscapes. *Advances in Neural Information Processing Systems*, 33: 15965–15977, 2020.

Violeta Kovacev-Nikolic, Peter Bubenik, Dragan Nikolić, and Giseon Heo. Using persistent homology and dynamical distances to analyze protein binding. *Statistical Applications in Genetics and Molecular Biology*, 15(1):19–38, 2016.

Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, pp. 2004–2013. PMLR, 2016.

Yongjin Lee, Senja D Barthel, Paweł Dłotko, S Mohamad Moosavi, Kathryn Hess, and Berend Smit. Quantifying similarity of pore-geometry in nanoporous materials. *Nature Communications*, 8(1):1–8, 2017.

Xiang Liu, Huitao Feng, Jie Wu, and Kelin Xia. Dowker complex based machine learning (dcml) models for protein-ligand binding affinity prediction. *PLOS Computational Biology*, 18(4):1 – 17, 2022.

Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In *International Conference on Learning Representations*, 2022.

Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *Mathematical Software–ICMS 2014: 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings 4*, pp. 167–174. Springer, 2014.

Zhenyu Meng, D Vijay Anand, Yunpeng Lu, Jie Wu, and Kelin Xia. Weighted persistent homology for biomolecular data analysis. *Scientific Reports*, 10(1):1–15, 2020.

Mohnhaupt Mona and S Kališnik Hintz. *The nerve theorem and its applications in topological data analysis*. PhD thesis, Bachelor's thesis, Swiss Federal Institute of Technology (ETH) Zurich, 2023.

Soham Mukherjee, Shreyas N Samaga, Cheng Xin, Steve Oudot, and Tamal K Dey. D-gril: End-to-end topological learning with 2-parameter persistence. *arXiv preprint arXiv:2406.07100*, 2024.

Naoki Nishikawa, Yuichi Ike, and Kenji Yamanishi. Adaptive topological feature via persistent homology: Filtration learning for point clouds. In *Advances in Neural Information Processing Systems*, 2023.

Leslie O'Bray, Bastian Rieck, and Karsten Borgwardt. Filtration curves for graph representation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1267–1275, 2021.

Julián Burella Pérez, Sydney Hauke, Umberto Lupo, Matteo Caorsi, and Alberto Dassatti. giotto-ph: a python library for high-performance computation of persistent homology of vietoris-rips filtrations. *arXiv preprint arXiv:2107.05412*, 2021.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer vision and Pattern Recognition*, pp. 652–660, 2017.

Raphael Reinauer, Matteo Caorsi, and Nicolas Berkouk. Persformer: A transformer architecture for topological machine learning. *arXiv preprint arXiv:2112.15210*, 2021.

Vsevolod Salnikov, Daniele Cassese, and Renaud Lambiotte. Simplicial complexes and complex systems. *European Journal of Physics*, 40(1):014001, 2018.

Joshua Southern, Jeremy Wayland, Michael Bronstein, and Bastian Rieck. Curvature filtrations for graph generative model evaluation. *Advances in Neural Information Processing Systems*, 36:63036–63061, 2023.

Jacob Townsend, Cassie Putman Micucci, John H Hymel, Vasileios Maroulas, and Konstantinos D Vogiatzis. Representation of molecular structures with persistent homology for machine learning applications in chemistry. *Nature Communications*, 11(1):1–9, 2020.

Wong Hiu Tung, Darrick Lee, and Hong Yan. Towards scalable topological regularizers. In *The Thirteenth International Conference on Learning Representations*, 2025.

Cédric Villani. The wasserstein distances. *Optimal transport: old and new*, pp. 93–111, 2009.

Siddharth Vishwanath, Kenji Fukumizu, Satoshi Kuriki, and Bharath K Sriperumbudur. Robust persistence diagrams using reproducing kernels. *Advances in Neural Information Processing Systems*, 33:21900–21911, 2020.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920, 2015.

Kelin Xia and Guo-Wei Wei. Multidimensional persistence in biomolecular data. *Journal of Computational Chemistry*, 36(20):1502–1520, 2015.

Kelin Xia, Zhiming Li, and Lin Mu. Multiscale persistent functions for biomolecular structure characterization. *Bulletin of Mathematical Biology*, 80:1–31, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in Neural Information Processing Systems*, 30, 2017.

Hang Zhang, Kaifeng Zhang, Kai Ming Ting, and Ye Zhu. Towards a persistence diagram that is robust to noise and varied densities. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Simon Zhang, Mengbai Xiao, and Hao Wang. Gpu-accelerated computation of vietoris-rips persistence barcodes. *arXiv preprint arXiv:2003.07989*, 2020.

Simon Zhang, Soham Mukherjee, and Tamal K Dey. Gefl: Extended filtration learning for graph classification. In *Learning on Graphs Conference*, pp. 16–1. PMLR, 2022.

Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. In *Proceedings of the twentieth Annual Symposium on Computational Geometry*, pp. 347–356, 2004.

A Background

A.1 Expected Persistence Diagram

A PD $\mathcal{D} = \{r_i = (b_i, d_i) \in \Omega | 1 \leq i \leq N(\mathcal{D})\}$ can be equivalently represented as a counting measure μ on Ω given by $A \in \mathcal{B} \to \mu(A) = \sum_{i=1}^{N(\mathcal{D})} \delta_{r_i}(A)$,

where $\mathcal{B}$ is the class of all Borel subsets of Ω and δ_r denotes the Dirac point mass at $r \in \Omega$. When each sampled PD is a random draw from a distribution P, its EPD, denoted as $\mathbb{E}[\mu]$, is defined as $A \in \mathcal{B} \to \mathbb{E}[\mu](A) = \mathbb{E}[\mu(A)]$ (Chazal & Divol, 2018).

Given a finite set $\{\mu_1, \mu_2, ..., \mu_n\}$, consisting of sampled PDs from P, the empirical EPD is defined as $\bar{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mu_i$. The support of $\bar{\mu}$ is $\mathcal{S}_{\bar{\mu}} = \cup_{i=1}^{n} \mathcal{D}_i$, where $\mathcal{D}_i = \{r_j = (b_j, d_j) \in \Omega | 1 \leq j \leq N(\mathcal{D}_i)\}$ is the support of the sampled PD μ_i. EPD can be viewed as a distribution (Chazal & Divol, 2018) of topological features supported on the open half plane Ω. Quantization method (Divol & Lacombe, 2021a) has been developed to reduce the support size of EPD.

An inherent metric applicable to the space of EPDs (Divol & Lacombe, 2021a) with the same mass, is Wasserstein distance (Villani, 2009). In a space of EPDs with different masses, the Optimal Partial Transport metric (OT_p) (Figalli, 2010) is used to allow any mass transportation from or to the diagonal $\partial\Omega$.

For the approximation error of EPD, we have the following theorem in Cao & Monod (2022):

Theorem A.1 (Cao & Monod (2022))**.** *Let $X \subset \mathbb{R}^m$ be a finite set of points, and π be a probability measure on X satisfying the (a, b, r_0)-standard assumption. Suppose $X_s^1, \ldots, X_s^M$ are M i.i.d. samples from the distribution $\pi^{\otimes n}$ with $|X_s^i| = n$. Let $\bar{\mu}$ be the empirical persistence measure, i.e., EPD; denote $\beta := \frac{p}{b} - 1$. Then the empirical persistence measure approaches the true persistence measure $\mathcal{D}(X)$ (here we omit the symbol filtration choice and $\mathcal{D}(X)$ is in measure form: $\mathcal{D}(X) = \sum_r \delta_r$) of X in expectation at the following rates:*

$$\mathbb{E}[\mathrm{OT}_p^p(\bar{\mu}, D(X))] \leq \begin{cases} O(M^{-1/2}) + O(1) + O\big(n^{-\beta}\big) & \text{if } p > b; \\ O(M^{-1/2}) + O(1) + O\left(\left(\frac{\log n}{n}\right)^{1/b}\right) & \text{if } p \leq b, r_0 < \left(\frac{\log n}{an}\right)^{1/b}; \\ O(M^{-1/2}) + O(1) + O\left(\left(\frac{\log n}{n}\right)^{p/b} \frac{1}{(\log n)^2}\right) & \text{if } p \leq b, r_0 \geq \left(\frac{\log n}{an}\right)^{1/b}. \end{cases}$$

A.2 Principal Persistence Measure

As a special case of Expected Persistence Diagram, the approximation error and convergence analysis of the empirical measure to Principal Persistence Measure is theoretically studied in Theorem 3.20 in Gómez & Mémoli (2024): the empirical measure converges to PPM almost surely as the number of subsets $M \to \infty$.

For the stability of PPM, we have the following Theorem A.2 from Tung et al. (2025).

Theorem A.2 (Tung et al. (2025))**.** *Let $p \geq 1$, and let W_p denote the p-Wasserstein metric on $\mathbb{R}^d$ and Ω'. A key property shown in Gómez & Mémoli (2024) Theorem 3.8, Theorem 4.11 is that PPMs are stable:*

$$W_p(PPM_q(\mu), PPM_q(\nu)) \leq C_q W_q(\mu, \nu),$$

for all $\mu, \nu \in \mathcal{P}_c(\mathbb{R}^d)$, where $C_q > 0$ is a constant which depends on homology dimension q and $\mathcal{P}_c(\mathbb{R}^d)$ is the Borel probability measure with compact support on $\mathbb{R}^d$.

The $\mathrm{PPM}_q(\mu)$ here means the Principal Persistence Measure with homology dimension q computed from points sampled from measure μ supported on $\mathbb{R}^n$ and the PPM is transformed from the (birth, death) to (birth, persistence) space $\Omega' = \{(b,l) \in \mathbb{R}^2\} \setminus \{l = 0\}$, where persistence=death-birth ≥ 0. This is a slightly rotated version of the PPM we considered in the $\Omega = \{(t_1, t_2) \in \mathbb{R}^2 | t_2 > t_1\}$.

For the computation of PPM, we have the following theorem:

Theorem A.3 (Gómez & Mémoli (2024))**.** *Let (X, d_X) be a metric space with n points. Then:*

1. *For all homology dimension larger than $\frac{n}{2} - 1$, the PD obtained via Rips filtration is empty.*

2. *If n is even and homology dimension equals to $\frac{n}{2} - 1$, then the PD obtained via Rips filtration consists of a single point $r_1 = (t_b, t_d)$ if and only if $t_b < t_d$, and is empty otherwise.*

A.3 Notes on Death time

Death times are finite when the homology dimension is zero and this results a point $r = (0, \infty)$ in the PD, representing the final connected component. In practice, for vectorization, we usually set the maximum threshold for the filtration to be a finite value or just remove the point with infinite death time since it exists for all point clouds and has no practical value. So here we focus on the practice aspect and assume the death time is always finite. We ignore the topological feature with infinite death time.

A.4 Vectorization via PersLay

PersLay (Carrière et al., 2020) is a supervised vectorization method for PD, a multiset $\mathcal{D} = \{r_i = (b_i, d_i) \in \Omega | 1 \leq i \leq N(\mathcal{D})\}$ on the half plane $\Omega = \{(t_1, t_2) \in \mathbb{R}^2 | t_2 > t_1\}$. PersLay is adapted from the DeepSets structure (Zaheer et al., 2017) and expressed as follows:

$$\mathrm{PersLay}(\mathcal{D}) = \mathbf{op}(\{w(r) \cdot \phi(r)\}_{r \in \mathcal{D}}),$$

where **op** is any permutation invariant operation (such as minimum, maximum, sum, kth largest value...), $w : \mathbb{R}^2 \to \mathbb{R}$ is a weight function for the points in PD, and $\phi : \mathbb{R}^2 \to \mathbb{R}^q$ is a representation function called point transformation, mapping each point (b_i, d_i) of a PD to a vector.

Certain choices of w, ϕ and **op** can transform PersLay into specific methods like Persistence Image (Adams et al., 2017) and Persistence Landscape (Bubenik et al., 2015). In our work, we follow the setting in Nishikawa et al. (2023) and choose $w(\cdot) = 1$ and ϕ as

$$\phi(r) = [\exp(-\frac{\|p - c_1\|^2}{2}), \exp(-\frac{\|p - c_2\|^2}{2}), ..., \exp(-\frac{\|p - c_m\|^2}{2})]^\intercal,$$

where $r = (b, d)$, $p = (b, d - b)$ and all the c_is are the parameters to be learned.

Non-DL vectorization. While our current work is more centered on the scalability of our proposed method within the DL-based context, we recognize the importance of the non-DL perspective. Regarding the non-DL based vectorization method, the scalability issue of Expected Persistence Diagram (EPD), a more general form of PPM, has been discussed in detail in Bubenik et al. (2015) and Section 4.3.1 in Gómez & Mémoli (2024).

A.5 Persistence Diagram-based Filtration Learning Framework

The PD-based filtration learning framework (PD-FL) (Nishikawa et al., 2023) is shown in Figure 4. PD-FL tries to learn the weight function $w(\cdot) = f_\theta(X, \cdot)$ from the entire point cloud X for weighted filtration. Once the weight is learned, weighted filtration is used to compute PD. Then, PD is vectorized by PersLay and used in machine learning task like point cloud classification.

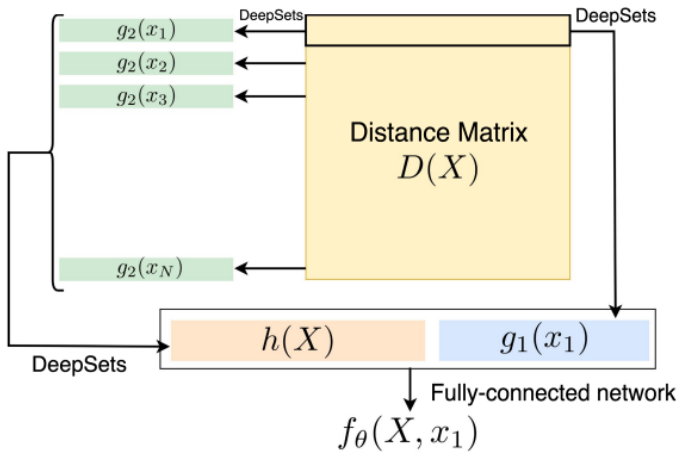

Figure 4: Persistence Diagram-based Filtration Learning Framework. Functions g_1, g_2 and h are the same as those we use in Section 4. This is a direct reuse of Figure 2 in Nishikawa et al. (2023).

B Proof and Related Analysis

B.1 On the assumption of P and U in Lemma 5.1 & Theorem 5.2

We have the following two assumptions on P and U.

1. We assume that P and U share the same support. Regarding the manifold's support, U is a (maybe uniform) distribution over the entire support of $\mathcal{M}$, including regions where the inlier density P are very low and points have little chance to be sampled from this low density area. An example is shown in Figure 16 in Gómez & Mémoli (2024). The support of P is the circular area and the low density region lies at the inner part of the ring.

2. P and U are both lower and upper bounded in the support, i.e. there exist constants $\bar{c} \geq \underline{c} > 0$ such that $\bar{c} \geq P(x) \geq \underline{c}$ and $\bar{c} \geq U(x) \geq \underline{c}$ for any x in the support.

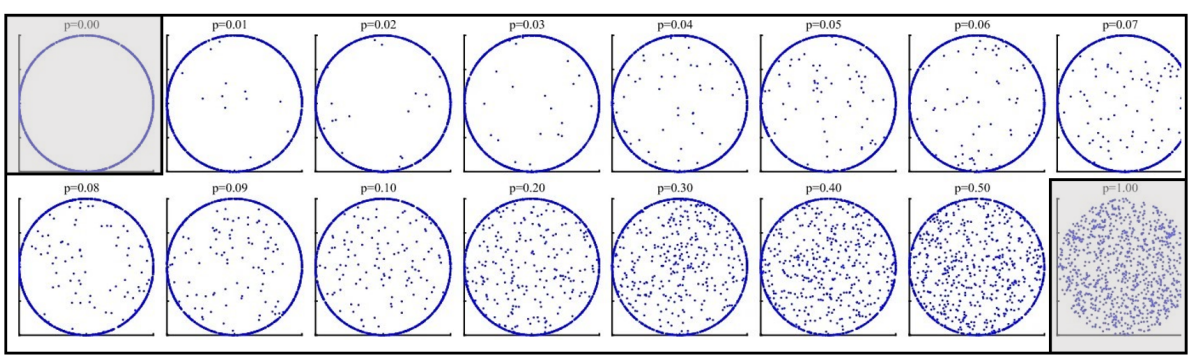

Figure 5: Part of Figure 16 from Gómez & Mémoli (2024).

The caption of Figure 16 in Gómez & Mémoli (2024) claims that "Given $0 \leq p \leq 1$, we sample $X \subset D^2$ with 1000 points so that each point is uniformly distributed in the interior of D^2 with probability p or on its boundary $\mathbb{S}^1_E$ with probability $1 - p$". We can consider P is such a distribution with a p that is close to 0, but it is not 0. And consider U is a distribution with p that is close to 1, but it is not 1. So both the supports of P and U are $D^2 \cup \mathbb{S}^1_E$ and the value of their pdf is upper and lower bounded. And it would not

cause the pdf being either 0 or infinite, as shown in the area not shaded in Figure 5. This aligns with our assumptions above.

B.2 Proof of Lemma 5.1

Lemma 5.1. For EPD $\mathbb{E}_{X_s \sim P^n}[\nu(X_s)]$ with density μ_1 and $\mathbb{E}_{X_s \sim P_o^n}[\nu(X_s)]$ with density μ_2, where $\nu(X_s)$ is the measure form of PD $\mathcal{D}(\mathcal{K}(X_s)$, i.e., $\nu(X_s) = \sum_{r \in \mathcal{D}(\mathcal{K}(X_s)} \delta_r$, it holds that

$$\|\mu_1 - \mu_2\|_1 \leq C_n H_k(\mathcal{M})^n p_{n-1}(\epsilon)\epsilon,$$

where C_n is the expected number of points in the PD built with the filtration $\mathcal{K}$ on n i.i.d. uniform points on $\mathcal{M}$, $H_k(\mathcal{M})$ is k-dimensional Hausdorff measure of $\mathcal{M}$ and $p_{n-1}(\epsilon)$ is a polynomial of order $n-1$ with bounded coefficients.

Proof. For Theorem 7.1 in Chazal & Divol (2018), we have that

$$\|\mu_1 - \mu_2\|_1 \leq C_n H_k(\mathcal{M})^n \|q_1 - q_2\|_\infty,$$

where $q_1(q_2)$ is the density with respect to the Hausdorff measure H_k, $q_1(x) = \Pi_{i=1}^n P(x_i)$ and $q_2(x) = \Pi_{i=1}^n P_o(x_i)$, where $x = [x_1^\intercal, ..., x_n^\intercal]^\intercal$. For the $\|q_1 - q_2\|_\infty$ term, since each x_i is i.i.d. sampled, for $x = \text{argmax}|q_1 - q_2|$, it holds that

$$\begin{aligned}(q_1 - q_2)(x) &= \Pi_i P(x_i) - \Pi_i P_o(x_i) \\ &= \Pi_i P(x_i)[1 - \frac{\Pi_i P_o(x_i)}{\Pi_i P(x_i)}],\end{aligned}$$

where under the assumption that each x_i is in the support (with positive density) of P and $1 > \frac{U(x_i)}{P(x_i)} \geq \Delta > 0$, we have that

$$\begin{aligned}\frac{\Pi_i P_o(x_i)}{\Pi_i P(x_i)} &= \Pi_i[1 - (1 - \frac{U(x_i)}{P(x_i)})\epsilon] \\ &\geq \Pi_i[1 - (1 - \Delta)\epsilon]\end{aligned}.$$

So it holds that

$$\begin{aligned}\|q_1 - q_2\|_\infty &= |q_1(x) - q_2(x)| \\ &\leq \Pi_i P(x_i)[1 - (1 - (1 - \Delta)\epsilon)^n] \\ &= p_{n-1}(\epsilon)\epsilon.\end{aligned}$$

Finally, we obtain that

$$\|\mu_1 - \mu_2\|_1 \leq C_n H_k(\mathcal{M})^n p_{n-1}(\epsilon)\epsilon.$$

□

B.3 The dependence on outlier distribution U

The upper bound in Lemma 5.1 is dependent on U. This dependence is reflected in the coefficients of the polynomial $p_{n-1}(\epsilon)$. As shown in the proof above, the polynomial's coefficients are derived from $\Delta = \inf \frac{U(x)}{P(x)}$ (the minimum ratio of outlier to inlier densities over $\mathcal{M}$), which is dependent on noise U.

For the area where P is dense and U is sparse, a sparser U will result a small Δ. This will lead to smaller absolute values of the coefficients in the polynomial $p_{n-1}(\epsilon)$, i.e. a smaller upper bound. This corresponds to the intuition that EPD (or PPM) is more robust to a sparser outlier distribution.

C Experiment Settings

Optimization. We set the batch size as 128 for protein datasets and 40 for ModelNet10 dataset. For optimizer, we use Adam. Learning rate is 0.1 for ModelNet and 0.001 for protein dataset. For

scheduler, we use the TransformerLRScheduler implemented in pytorch (`https://github.com/sooftware/pytorch-lr-scheduler/tree/main`) and the number of warm-up epoch is set to be 40. For ModelNet10, the number of epochs in the first phase is 1500. For protein dataset and the second phase of ModelNet dataset, we use the EarlyStopping handler (`https://pytorch.org/ignite/generated/ignite.handlers.early_stopping.EarlyStopping.html`) with patience=20 and min_delta = 0.002 for the loss on validation set of size 200.

Principal Persistence Measure. For the computation of PPM, we set $M = 200$ for both protein and ModelNet10 datasets. For the vectorization method PersLay, the length of vectorization m is set to be 32 and the permutation invariant operation **op** is summation.

Networks. For the DNN-based methods (DeepSets, PointNet and PointMLP) in the first phase, we use the same structure as those in Nishikawa et al. (2023). We set all of the permutation invariant operators **op** that appear in PPM-FL and PD-FL are all summation. The dimension of the feature vectors obtained by PersLay is set as 16, except that when using both homology dimensions, it is 32. The DeepSets-like structures $\phi^{(1)} - \phi^{(5)}$ and fully connected network are the same as those in Nishikawa et al. (2023). We initialized the parameters in PPM-FL with normal distribution with a mean of 0 and a standard deviation of 1.0. Other parameters were initialized with the default settings of PyTorch.

Datasets. The protein dataset (Kovacev-Nikolic et al., 2016) does not provide a point cloud. Instead, a cross-correlation matrix $\mathbb{C}$ is provided for each protein. Then the dynamic distance matrix $\mathbb{D}$, where $\mathbb{D}_{i,j} = 1 - |\mathbb{C}_{i,j}|$, is used to compute PD or PPM. We use a version of this dataset (Nishikawa et al., 2023), which contains two classes of protein, with 500 instances in each class. Each instance is a distance matrix of shape 60×60 and has noise from a uniform distribution with standard deviation of 0.1 for the off-diagonal elements. For the ModelNet10 dataset, we use the version which contains 10 classes, with 100 instances in each class. Each instance is a point cloud of shape 128×3.

D Additional Results and Discussions

D.1 Results of two-phase training (PointMLP + PPM-FL)

For PointMLP, it is claimed in Nishikawa et al. (2023) that PD-FL reduces the accuracy of 68.80 to below or around 60 in the first phase, because PointMLP has already captured information including topology in the first phase and PD-FL brings in redundant information.

Table 6: Accuracie for the classification task of ModelNet10 dataset when the first phase is PointMLP.

1st Phase	PointMLP	70.10 ± 4.70		
		$q = 0$	$q = 1$	$q = 0\&1$
2nd Phase	PPM-Rips	54.20 ± 9.56	57.30 ± 13.37	49.29 ± 11.82
	PPM-FL	52.70 ± 8.58	53.90 ± 8.23	56.59 ± 13.01

The results of the two-phase process where the first phase uses PointMLP are shown in Table 6. Despite the choice of homology dimension, the accuracy of PPM-FL in the second phase is significantly lower than that in the first phase. This confirms the claim in Nishikawa et al. (2023) that PointMLP has already captured information including topology in the first phase and PD-FL brings in redundant information.

At a higher level, the topological information here is specific pairwise distance that is topologically meaningful, since the birth and death of a topological feature in PD or PPM are actually pairwise distances. In PointNet and DeepSets, no pairwise information is needed because both of them use a network to transform point feature and then use a permutation invariant operator to aggregate these features as point cloud-level feature. But PointMLP needs pairwise info to define a neighborhood in order to learn the feature of a local region. This process involves pairwise distances. This could explain why PPM or PD can only work for PointNet and DeepSets, rather than PointMLP.

D.2 Relation between PPM and the distribution of pairwise distances.

It is worth mentioning that when homology dimension $q = 0$ and we use the unsupervised Rips filtration instead of the filtration learning, PPM represents the distribution of pairwise distances, which is shown in Boutin & Kemper (2005) to almost solve the isometry classification problem for point clouds. PPM of higher homology dimension can be viewed as a conditional distribution of pairwise distances that are topologically meaningful for high dimensional cycles.

D.3 Ablation study on weight choice: Gaussian or Uniform

We present supplementary experimental results of PPM-FL with Gaussian ($\sum_{i=1}^{M} K(X_s^i) f(X_s^i, \cdot)$) or Uniform ($\sum_{i=1}^{M} f(X_s^i, \cdot)$) weight on the ModelNet10 dataset, under the same setting as that in Table 2. In the first phase, we use DeepSets. The results are shown in Table 7. For the choice of K , any differentiable distributional kernel would be fine. For simplicity, we choose to use Gaussian Distribution Kernel. The hyperparameter σ is set to be the median of all the pairwise distances. Our current results demonstrate effectiveness with this default setting.

Table 7: Accuracies of PPM-FL on ModelNet10 dataset under different weight functions.

	$q = 0$	$q = 1$	$q = 0\&1$
Gaussian	67.90 ± 3.01	66.90 ± 2.17	67.50 ± 2.88
Uniform	67.60 ± 1.97	67.00 ± 2.21	66.80 ± 1.87

In our pipeline, the first phase uses DeepSets, while the second phase employs PPM-FL with either a Gaussian or Uniform weight function. As demonstrated in the table below, across different homology dimensions, PPM-FL utilizing the Gaussian weight either outperforms or achieves similar results with the Uniform weight variant. This outcome validates the effectiveness of the Gaussian weight function, aligning with the design principle that subsets in closer proximity should have a more substantial influence. While the performance differences are subtle given the nature of the dataset, Table 7 shows the effectiveness of our proposed Gaussian weighting.

D.4 Relation between Theorem 5.2 and existing results in Divol & Lacombe (2021b).

Here we discuss the difference between our theoretical results (Theorem 5.2) with the two Propositions (5.4 & 5.5) in Section 5.3 of Divol & Lacombe (2021b).

- In Proposition 5.4, the support of distribution P and P' is $\mathcal{M}^p$, the space of PDs (measures) with finite persistence. Proposition 5.4 demonstrates that the expectation of P, i.e. EPD, is stable with respect to the distortion (P') of P. While our result is about the stability of EPD with respect to the addition of outliers of distribution P, which is supported on $\mathbb{R}^d$ (instead of $\mathcal{M}^p$), the space where we sample the subsets.
- Proposition 5.5 demonstrates the stability of EPD with respect to the distribution ξ supported on $\mathbb{R}^d$. This is similar to our result in Lemma 5.1, with $P(P_o)$ corresponding to $\xi(\xi')$. Compared with Proposition 5.5, our result (Lemma 5.1) takes a specific form of $P_o = (1 - \epsilon) \cdot P + \epsilon \cdot U$ (a mixture of the original distribution P and outlier distribution U) and links the upper bound to the mixture proportion ϵ, while Proposition 5.5 generally gives the upper bound as the bottleneck distance $W_\infty(\xi, \xi')$). It could be argued that our result is a specific case of Proposition 5.5 of Divol & Lacombe (2021b).

D.5 Results under different outlier percentages

We report accuracies and standard deviations corresponding to Figure 3 in Table 8. Around $\epsilon = 0$, the accuracies of PPM-FL and PD-FL drop with a similar rate. When $\epsilon \leq 1.5\%$, the accuracy of PPM-FL

Table 8: Accuracies and standard deviations under different outlier percentages (ϵs). The first phase uses DeepSets. The second phase uses both homology dimensions $q = 0\&1$.

ϵ	**PPM-FL**	**PD-FL**	ϵ	**PPM-FL**	**PD-FL**
0.0%	67.50 ± 2.88	67.40 ± 0.82	10.5%	60.50 ± 1.16	59.20 ± 2.81
0.5%	67.40 ± 2.94	67.30 ± 0.29	11.0%	60.60 ± 0.09	56.90 ± 1.65
1.0%	67.40 ± 2.13	66.80 ± 1.84	11.5%	60.60 ± 0.09	56.90 ± 1.65
1.5%	67.40 ± 2.13	66.80 ± 1.84	12.0%	61.40 ± 1.42	56.60 ± 1.42
2.0%	66.80 ± 2.55	67.40 ± 1.07	12.5%	58.10 ± 1.45	55.50 ± 2.16
2.5%	66.50 ± 1.25	66.20 ± 0.76	13.0%	58.10 ± 1.45	55.50 ± 2.16
3.0%	66.50 ± 1.25	66.20 ± 0.76	13.5%	59.40 ± 2.02	56.70 ± 1.59
3.5%	66.10 ± 1.76	65.50 ± 2.05	14.0%	59.40 ± 2.02	56.70 ± 1.59
4.0%	65.50 ± 2.22	63.60 ± 1.53	14.5%	58.40 ± 1.49	55.90 ± 2.53
4.5%	65.50 ± 2.22	63.60 ± 1.53	15.0%	57.60 ± 1.11	53.90 ± 2.66
5.0%	64.90 ± 1.16	64.40 ± 1.91	15.5%	57.60 ± 1.11	53.90 ± 2.66
5.5%	64.60 ± 0.90	63.70 ± 0.49	16.0%	57.10 ± 1.62	53.50 ± 3.36
6.0%	64.60 ± 0.90	63.70 ± 0.49	16.5%	56.40 ± 1.96	53.70 ± 2.02
6.5%	63.20 ± 0.61	63.90 ± 2.75	17.0%	56.40 ± 1.96	53.70 ± 2.02
7.0%	63.20 ± 0.61	63.90 ± 2.75	17.5%	56.10 ± 2.59	51.10 ± 2.94
7.5%	65.60 ± 1.03	63.50 ± 2.03	18.0%	57.90 ± 0.91	49.80 ± 4.12
8.0%	62.10 ± 0.57	59.80 ± 0.61	18.5%	57.90 ± 0.91	49.80 ± 4.12
8.5%	62.10 ± 0.57	59.80 ± 0.61	19.0%	53.00 ± 1.09	50.00 ± 3.09
9.0%	61.80 ± 2.09	60.10 ± 1.52	19.5%	53.00 ± 1.09	50.00 ± 3.09
9.5%	63.30 ± 1.15	60.10 ± 2.00	20.0%	54.30 ± 1.32	49.00 ± 3.79
10.0%	63.30 ± 1.15	60.10 ± 2.00	40.0%	34.00 ± 2.11	31.30 ± 5.91

almost remains the same while PD-FL drops from 67.40 to 66.80. This demonstrates that a few outliers have more impact on PD-FL than PPM-FL. PD-FL does not ignore the first outliers while PPM-FL does. The reason behind this may be that when the number of outliers are small, the outliers have very little chance to be selected in a small subset. When $\epsilon = 40.0\%$, both PPM-FL and PD-FL have very bad performance.