Adaptive Topological Feature via Persistent Homology: Filtration Learning for Point Clouds

Naoki Nishikawa
The University of Tokyo
nishikawa-naoki259@g.ecc.u-tokyo.ac.jp

Yuichi Ike
Kyushu University
ike@imi.kyushu-u.ac.jp

Kenji Yamanishi
The University of Tokyo
yamanishi@mist.i.u-tokyo.ac.jp

Abstract

Machine learning for point clouds has been attracting much attention, with many applications in various fields, such as shape recognition and material science. For enhancing the accuracy of such machine learning methods, it is often effective to incorporate global topological features, which are typically extracted by persistent homology. In the calculation of persistent homology for a point cloud, we choose a filtration for the point cloud, an increasing sequence of spaces. Since the performance of machine learning methods combined with persistent homology is highly affected by the choice of a filtration, we need to tune it depending on data and tasks. In this paper, we propose a framework that learns a filtration adaptively with the use of neural networks. In order to make the resulting persistent homology isometry-invariant, we develop a neural network architecture with such invariance. Additionally, we show a theoretical result on a finite-dimensional approximation of filtration functions, which justifies the proposed network architecture. Experimental results demonstrated the efficacy of our framework in several classification tasks.

1 Introduction

Analysis of point clouds (finite point sets) has been increasing its importance, with many applications in various fields such as shape recognition, material science, and pharmacology. Despite their importance, point cloud data were difficult to deal with by machine learning, in particular, neural networks. However, thanks to the recent development, there have appeared several neural network architectures for point cloud data, such as DeepSet (Zaheer et al., 2017), PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), and PointMLP (Ma et al., 2022). These architectures have shown high accuracy in tasks such as shape classification and segmentation.

In point cloud analysis, topological global information, such as connectivity, the existence of holes and cavities, is known to be beneficial for many tasks (see, for example, (Hiraoka et al., 2016; Kovacev-Nikolic et al., 2016)). One way to incorporate topological information into machine learning is to use persistent homology, which is a central tool in topological data analysis and attracting much attention recently. By defining an increasing sequence of spaces, called a *filtration*, from a point cloud $X \subset \mathbb{R}^d$, we can compute its persistent homology, which reflects the topology of X. Topological features obtained through persistent homology are then combined with neural network methods and enhance the accuracy of many types of tasks, such as point cloud segmentation (Liu et al., 2022), surface classification (Zeppelzauer et al., 2016), and fingerprint classification (Giansiracusa et al., 2019).

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

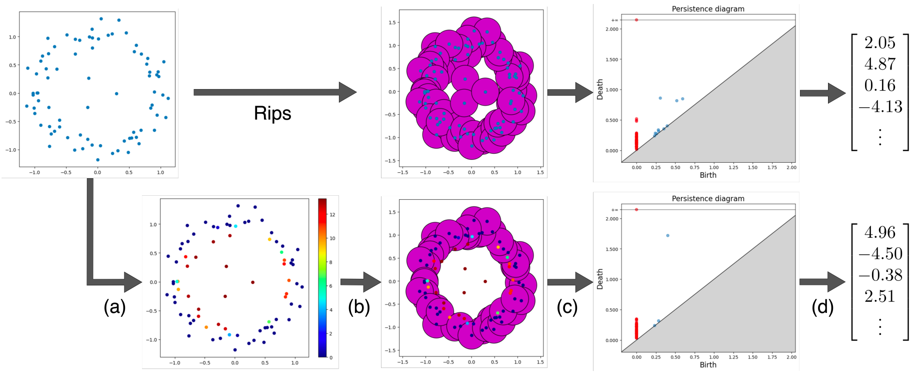

Figure 1: Procedure to utilize our framework to a classification task and its comparison to when we use Rips filtration. For a given set of points, define a value called “weight” for each point in (a), using a neural network method. In (b), a ball centered at each point is gradually expanded. The balls centered at points with larger weights expand later than the balls centered at points with smaller weights. In (c), we examine the topology of the sum of the balls as they grow larger, and aggregate the information into a diagram called a persistence diagram. In (d), the persistence diagram is converted to a vector, which can be used as input of machine learning models. When we use standard Rips filtration, the balls expand uniformly from all of the points, so that we cannot correctly capture the large global hole. On the other hand, when we use our method, the suitable filtration to solve a classification task is chosen, and the weights on the outliers get larger. As a result, we can capture the large global hole in the point cloud. Note that the resulting weights assigned to each point are *not pre-specified, but are fully learned from the data.*

The standard pipeline to use persistent homology in combination with machine learning is split into the following:

(i) define a filtration for data and compute the persistent homology;

(ii) vectorize the persistent homology;

(iii) input it into a machine learning model.

In (i), one needs to construct appropriate filtration depending on a given dataset and a task. A standard way to define a filtration is to consider the union of balls with centers at points in $X \subset \mathbb{R}^d$ with a uniform radius, i.e., take the union $S_t = \bigcup_{x \in X} B(x;t), B(x;t) := \{y \in \mathbb{R}^d \mid \|y - x\| \leq t\}$ and consider the increasing sequence $(S_t)_t$. Regarding (ii), one also has to choose a vectorization method depending on the dataset and the task. For example, one can use typical vectorization methods such as persistence landscape and persistence image. As for (iii), we can use any machine learning models such as linear models and neural networks, and the models can be tuned with the loss function depending on a specific task.

Recently, several studies have proposed ideas to learn not only (iii) but also (i) and (ii). Regarding (ii), Hofer et al. (2017) and Carrière et al. (2020), for example, have proposed methods to learn vectorization of persistence homology based on data and tasks. Furthermore, for (i), Hofer et al. (2020) have proposed a learnable filtration architecture for graphs, which gives a filtration adaptive to a given dataset. However, this method heavily depends on properties specific to graph data. To establish a method to learn filtration for point clouds, we need to develop a learnable filtration architecture for a point cloud incorporating the pairwise distance information. In this paper, we tackle this challenging problem to adaptively choose a filtration for point clouds.

1.1 Contributions

In this paper, we propose a novel framework to obtain topological features of point clouds based on persistent homology. To this end, we employ a *weighted filtration*, whose idea is to take the union of balls with different radii depending on points. Choosing a *weight function* $w\colon X \to \mathbb{R}$, we can define a filtration $(S_t)_t$ by

$$S_t = \bigcup_{x \in X} B(x; t - w(x)).$$

This type of filtration often extracts more informative topological features compared to non-weighted filtrations. Indeed, a weighted filtration plays an important role in various practical applications (Anai et al., 2020; Nakamura et al., 2015; Hiraoka et al., 2016; Obayashi et al., 2022). Then, we introduce a neural network architecture to learn a map that associates a weight function $f(X, \cdot)$ with a point cloud $X \subset \mathbb{R}^d$.

The desirable property for network f is *isometry-invariance*, i.e., $f(TX, Tx) = f(X, x)$ for any isometric transformation T of $\mathbb{R}^d$, so that the associated weighted filtration is isometry-invariant. To address this issue, we implement a function f with a network architecture based on distance matrices. Then, the resulting weight function satisfies the isometry-invariance *rigorously*. Additionally, we show a theoretical result about the approximation ability for a wide class of weight functions, which supports the validity of our architecture.

Our main contributions are summarized as follows:

1. We propose a novel framework to obtain adaptive topological features for point clouds based on persistent homology. To this end, we introduce an isometry-invariant network architecture for a weight function and propose a way to learn a weighted filtration.
2. We theoretically show that any continuous weight function can be approximated by the composite of two continuous functions which factor through a finite-dimensional space. This theoretical result motivates our architecture.
3. We conducted experiments on public datasets and show that the topological features obtained by our method improved accuracy in classification tasks.

Figure 1 presents one typical use of our framework for a classification task. Our method first calculates the weight of each point using a neural network model. The colored point clouds in the second row depict the assigned weights for each point. In the bottom one, our architecture assigns large weights on the outliers in the point cloud, which implies that our method can choose the filtration adaptively to data and tasks. Although the DTM filtration is also effective in the example shown in Figure 1, our method can learn a filtration appropriately for a given task and data in a supervised way. Therefore, our method would be effective for data and tasks with difficulties other than outliers. In §5, we will show that our method improves the classification accuracy compared to the Rips or DTM filtration in several experiments.

1.2 Related work

A neural network architecture for finite point sets was first proposed by Zaheer et al. (2017), as DeepSets. Later, there appeared several geometric neural networks for point clouds, such as PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), and PointMLP (Ma et al., 2022). These studies addressed the rotation invariance by approximating rotation matrices by a neural network. (Xu et al., 2021) used Gram matrices to implement an isometry-invariant neural network for point clouds. We instead use distance matrices for our isometry-invariant network architecture since the pairwise distance information is important in persistent homology.

Regarding vectorization of persistent homology, in the past, it has usually been chosen in unsupervised situations, as seen in Bubenik and Kim (2007); Chung et al. (2009); Bubenik et al. (2015); Adams et al. (2017); Kusano et al. (2017). However, recent studies such as Hofer et al. (2017) and Carrière et al. (2020) have proposed end-to-end vectorization methods using supervised learning. In particular, Carrière et al. (2020) used the DeepSets architecture Zaheer et al. (2017), whose input is a finite (multi)set, not a vector, for vectorization of persistent homology to propose PersLay. Although our method also depends on DeepSets, it should be distinguished from PersLay; our method uses it for learning *a weight for computing persistent homology* while PersLay learns *a vectorization method of persistence homology*.

For graph data, Hofer et al. (2020) proposed the idea of filtration learning in supervised way, i.e., to learn the process in (i), and this idea was followed by Horn et al. (2021) and Zhang et al. (2022). They introduced a parametrized filtration with the use of graph neural networks and learned it according to a task. The persistent homology obtained by the learned filtration could capture the global structure of the graph, and in fact, they achieved high performance on the graph classification problem. However, filtrations for point clouds cannot be defined in the same way, as they are not equipped with any adjacency structure initially, but equipped with pairwise distances. Moreover, the filtration should be invariant with respect to isometric transformations of point clouds, and we need a different network architecture to that for graphs. While many previous studies, such as Bendich et al. (2007); Cang and Wei (2017); Cang et al. (2018); Meng et al. (2020), have tried to design special filtration in an unsupervised way based on data properties, we aim to learn a filtration in a *data-driven and supervised way*. To the best of our knowledge, this study is the first attempt to learn a filtration for persistent homology on point cloud data in a supervised way.

Another approach to obtaining topological features is to approximate the entire process from (i) to (iii) for computing vectorization of persistent homology by a neural network (Zhou et al. (2022) and de Surrel et al. (2022)). Note that similar methods are proposed for image data (Som et al., 2020) and graph data (Yan et al., 2022). Since these networks only approximate topological features by classical methods, they cannot extract information more than features by classical methods or extract topological information adaptively to data.

2 Background

2.1 Network Architectures for Point Clouds

There have been proposed several neural network architectures for dealing with point clouds for their input. Here, we briefly explain DeepSets (Zaheer et al., 2017), which we will use to implement our isometry-invariant structure. A more recent architecture, PointNet (Qi et al., 2017a) and PointMLP (Ma et al., 2022), will also be used for the comparison to our method in the experimental section.

The DeepSets architecture takes a finite (multi)set $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ of possibly varying size as input. It consists of the composition of two fully-connected neural networks $\phi_1 \colon \mathbb{R}^d \to \mathbb{R}^{d'}$ and $\phi_2 \colon \mathbb{R}^{d'} \to \mathbb{R}^{d''}$ with a permutation invariant operator $\mathbf{op}$ such as $\max, \min$ and summation:

$$\mathrm{DeepSets}(X) \coloneqq \phi_2(\mathbf{op}(\{\phi_1(x_i)\}_{i=1}^N)).$$

For each $x_i \in X$, the map ϕ_1 gives a representation $\phi_1(x_i)$. These pointwise representations are aggregated via the permutation invariant operator $\mathbf{op}$. Finally, the map ϕ_2 is applied to provide the network output. The output of DeepSets is permutation invariant thanks to the operator $\mathbf{op}$, from which we can regard the input of the network as a set. The parameters characterizing ϕ_1 and ϕ_2 are tuned through the training depending on the objective function.

2.2 Persistent Homology

Here, we briefly explain persistence diagrams, weight filtrations for point clouds, and vectorization methods of persistence diagrams.

Persistence Diagrams Let $S \colon \mathbb{R}^d \to \mathbb{R}$ be a function on $\mathbb{R}^d$. For $t \in \mathbb{R}$, the t-sublevel set of S is defined as $S_t \coloneqq \{p \in \mathbb{R}^d \mid S(p) \leq t\}$. Increasing t from $-\infty$ to ∞ gives an increasing sequence of sublevel sets of $\mathbb{R}^d$, called a filtration. *Persistent homology* keeps track of the value of t when topological features, such as connected components, loops, and cavities, appear or vanish in this sequence. For each topological feature α, one can find the value $b_\alpha < d_\alpha$ such that the feature α exists in S_t for $b_\alpha \leq t < d_\alpha$. The value b_α (resp. d_α) is called the birth (resp. death) time of the topological feature α. The collection (b_α, d_α) for topological features α is called the *persistence diagram* (PD) of f, which is a multiset of the half-plane $\{(b, d) \mid d > b\}$. The information of connected components, loops, and cavities are stored in the 0th, 1st, and 2nd persistence diagrams, respectively.

Given a point cloud $X \subset \mathbb{R}^d$, i.e., a finite point set, one can take S to be the distance to point cloud, defined by

$$S(z) = \min_{x \in X} \|z - x\|.$$

Then, the sublevel set S_t is equal to the union of balls with radius t centered at points of X: $\bigcup_{x \in X} B(x; t)$, where $B(x; t) := \{y \in \mathbb{R}^d \mid \|x - y\| \leq t\}$. In this way, one can extract the topological features of a point cloud X with the persistence diagram. The persistent homology of this filtration can be captured by the filtration called Čech or Vietoris-Rips filtration (Rips filtration for short). In this paper, we use Rips filtration for computational efficiency. See Appendix A for details. The Rips filtration can be computed only by the pairwise distance information.

Weighted Filtrations for Point Clouds While the Rips filtration considers a ball with the same radius for each point, one can consider the setting where the radius value depends on each point (see, for example, Edelsbrunner (1992)). For instance, Hiraoka et al. (2016); Nakamura et al. (2015); Obayashi et al. (2022), which use persistent homology to find the relationship between the topological nature of atomic arrangements in silica glass and whether the atomic arrangement is liquid or solid, use such a filtration. Given a point cloud $X \subset \mathbb{R}^d$, one can define the radius value $r_x(t)$ at time t for $x \in X$ as follows. We first choose a function $w\colon X \to \mathbb{R}$, which is called *weight*, and define

$$r_x(t) := \begin{cases} -\infty & \text{if } t < w(x), \\ t - w(x) & \text{otherwise.} \end{cases}$$

The associated weighted Rips filtration is denoted by $R[X, w]$. One choice as a weight function is the distance-to-measure (DTM) function. The resulting filtration is called a DTM-filtration, for which several theoretical results are shown in Anai et al. (2020). One of those results shows that a persistent homology calculated by the DTM-filtrations is robust to outliers in point clouds, which does not hold for the Rips filtrations.

Vectorization Methods of Persistence Diagrams Persistence diagrams are difficult to handle by machine learning since they are a multiset on a half-plane. For this reason, several vectorization methods of persistence diagrams have been proposed, such as persistence landscape (Bubenik et al., 2015) and persistence image (Adams et al., 2017). Carrière et al. (2020) proposed the method called PersLay, which vectorizes persistence diagrams in a data-driven way using the structure based on DeepSets. In this paper, we utilize this method as a vectorization method. Note that PersLay can be replaced with any vectorization method in our architecture.

Given a function transformation map ϕ_c with a learnable parameter c and permutation invariant operator $\mathbf{op}$, we can vectorize a persistence diagram D by PersLay with

$$\mathrm{PersLay}(D) := \mathbf{op}(\{\phi_c(q)\}_{q \in D}).$$

If we choose appropriate parametrized function ϕ_c, we can make PersLay similar to classical vectorization methods such as persistence landscape and persistence image. In this paper, we utilized the following map as ϕ_c, which makes PersLay similar to persistence image:

$$\phi_c(q) = \left[\exp\left(-\frac{\|q_1 - c_1\|^2}{2}\right), \ldots, \exp\left(-\frac{\|q_M - c_M\|^2}{2}\right)\right]^\top.$$

3 Proposed Framework

In this section, we describe our framework to obtain topological representations of given point clouds. Our idea is to use persistence homology for extracting topological information and model a weight function w in §2.2 by a neural network, which can be learned adaptively to data. Moreover, we propose a way to combine the topological features with deep neural networks.

3.1 Filtration Learning for Point Clouds

First, we introduce a structure to learn a filtration on point clouds end-to-end from data. For that purpose, we learn a weight function w for a weighted filtration, by modeling it by a neural network. We assume that the weight function depends on an input point cloud $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$, i.e., w is of the form $f(X, \cdot)\colon \mathbb{R}^d \to \mathbb{R}$. Designing such a function by a neural network has the following three difficulties:

1. The weight function $f(X, \cdot)$ should be determined by the whole point cloud *set* X and does not depend on the order of the points in X.

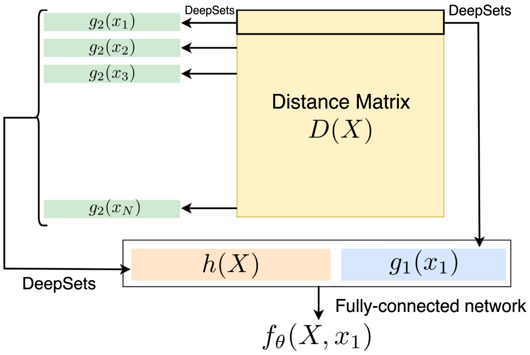

Figure 2: The architecture of the network. This figure is for the case of getting the weight of point x_1 in the point cloud X. Note that $d(X, x_i)$ is equal to (the transpose of) the i-th row of $D(X)$, i.e., $(d(x_i, x_j))_{j=1}^N$.

2. The resulting function should be *isometry-invariant*, i.e, for any isometric transformation T, any point cloud X, and $x \in X$, it holds that $f(TX, Tx) = f(X, x)$.
3. The output of the $f(X, x)$ should have both of global information X and pointwise information of x.

To address the problem 1, we use the DeepSets architecture (§2.1). To satisfy invariance for isometric transformations as in problem 2, we utilize the distance matrix $D(X) = (d(x_i, x_j))_{i,j=1}^N \in \mathbb{R}^{N \times N}$ and the relative distances $d(X, x) = (\|x - x_j\|)_{j=1}^N$ instead of coordinates of points, since $D(TX) = D(X)$ and $d(TX, Tx) = d(X, x)$ hold for any isometric transformation T. Finally, to deal with problem 3, we regard f as a function of the feature of $D(X)$ and the feature of $d(X, x)$. Below, we will explain our implementation in detail.

First, we implement a network to obtain a pointwise feature vector of a point $x \in \mathbb{R}^d$ by

$$g_1(x) \coloneqq \phi^{(2)}(\mathbf{op}(\{\phi^{(1)}(\|x - x_j\|)\}_{j=1}^N)), \tag{1}$$

where $\phi^{(1)}$ and $\phi^{(2)}$ are fully-connected neural networks. This structure (1) can be regarded as the DeepSets architecture whose input is a set of one-dimensional vectors, hence it is permutation invariant.

Next, we use the DeepSets architecture to obtain a feature vector for the entire point cloud X as follows. We first implement pointwise feature vectors with the same architecture as g_1, i.e.,

$$g_2(x_i) \coloneqq \phi^{(4)}(\mathbf{op}(\{\phi^{(3)}(d(x_i, x_j))\}_{j=1}^N)),$$

where $\phi^{(3)}$ and $\phi^{(4)}$ are fully-connected neural networks. Then, we get a feature vector $h(X)$ for the entire point cloud X by applying the DeepSets architecture again:

$$h(X) = \phi^{(5)}(\mathbf{op}(\{g_2(x_i)\}_{i=1}^N)), \tag{2}$$

where $\phi^{(5)}$is a fully-connected neural network. Here we omit the first inner network for the DeepSets architecture since it can be combined with $\phi^{(4)}$ in g_2.

Finally, we implement f as a neural network whose input is the concatenated vector of $h(X)$ and $g_1(x)$. Specifically, we concatenate $h(X)$ with the feature vector $g_1(x)$ for $x \in \mathbb{R}$ to get a single vector, and then input it into a fully-connected neural network $\phi^{(6)}$:

$$f_\theta(X, x) = \phi^{(6)}([h(X), g_1(x)]^\top),$$

where $\theta \coloneqq (\theta_k)_{k=1}^6$ is the set of parameters that appear in network $\phi^{(k)}$ $(k = 1, \dots, 6)$. The overall neural network architecture is shown in Figure 2.

Remark 3.1. Since our implementation of the weight function depends on distance matrices, not on the coordinates of each point, we can apply our method to a dataset given in distance matrix format. In fact, we will apply our framework to such a dataset in Section 5.

Remark 3.2. If we want to consider features other than the position information assigned to each point in the calculation of the filtration weights, as in the silica glass studies, we vectorize those features and concatenate them together.

By using the weight function $f_\theta(X, \cdot)$, we can calculate the persistent homology of filtration $R[X, f_\theta(X, \cdot)]$. Then, we can obtain the vectorization using PersLay parametrized by c. Then, we can input this vector into a machine learning model $F_{\theta'}$ with parameter θ'. Given the dataset $\{(X_k, y_k)\}_{k=1}^M$, the objective function to be minimized can be written as

$$\mathcal{L}(\theta, \theta', c) := \frac{1}{M} \sum_{k=1}^{M} \ell\left(F_{\theta'}(\mathrm{PersLay}(R[X_k, f_\theta(X_k, \cdot)])), y_k\right),$$

where ℓ is an appropriate loss function depending on tasks. Typically, if the task is regression, ℓ is the mean square loss, and if the task is classification, ℓ is the cross entropy loss. The optimization problem

$$\min_{\theta, \theta', c} \mathcal{L}(\theta, \theta', c)$$

is solved by stochastic gradient descent. Note that the differentiability of the objective function and the convergence of the optimization algorithm is guaranteed by results in Carrière et al. (2021).

3.2 Combining Topological Features with a DNN-based method

While we can use the topological features alone, we can also concatenate them with the features computed by a deep neural network (DNN). In order to do this, we need to learn not only (a) networks in our framework but also (b) DNN. Although (a) and (b) can be learned simultaneously since the output of the resulting feature is differentiable with all parameters, the networks cannot be successfully optimized in our experiments. This would be because the loss function is not smooth with respect to the parameters in (a), which can also make the optimization of (b) unstable. To deal with this issue, we propose to learn (a) and (b) separately.

Let us describe the whole training procedure briefly here. Suppose X is a input point cloud. We write $\Psi_{\mathrm{topo}}(X) \in \mathbb{R}^{L_1}$ for the topological feature which comes from our method, and $\Psi_{\mathrm{DNN}}(X) \in \mathbb{R}^{L_2}$ for the feature that comes from a DNN-based method. Let ℓ be a loss function for the classification task and n be a number of classes. We propose the following two-phase training procedure:

1st Phase. Let $C_1 \colon \mathbb{R}^{L_2} \to \mathbb{R}^n$ be a classifier that receives feature from Ψ_{DNN}. Then, C_1 and Ψ_{DNN} are learned through the classification task, which is achieved by minimizing $\sum_{k=1}^M \ell(C_1(\Psi_{\mathrm{DNN}}(X_k)), y_k)$.

2nd Phase. Let $C_2 \colon \mathbb{R}^{L_1+L_2} \to \mathbb{R}^n$ be a classifier. *We discard the classifier C_1 and fix Ψ_{DNN}.* Then, C_2 and Ψ_{topo} are learned through the classification task, which is achieved by minimizing $\sum_{k=1}^M \ell(C_2([\Psi_{\mathrm{DNN}}(X_k)^\top, \Psi_{\mathrm{topo}}(X_k)^\top]^\top), y_k)$.

This training procedure is observed to make the optimization stable and to help the architecture to achieve higher accuracy. We show the experimental result in § 5.

4 Approximation Ability of Weight Function

In this section, we theoretically show that our architecture has a good expression power. We prove that approximation of any continuous map can be realized by our idea to concatenate two finite-dimensional feature vectors.

Consider the space $2^{[0,1]^m}$ of subsets in $[0,1]^m$ equipped with the Hausdorff distance. For a fixed $N \in \mathbb{N}$, we define the following subspace of $2^{[0,1]^m}$:

$$\mathcal{X} := \{X \subset [0,1]^m \mid |X| \leq N\}.$$

The following theorem states that we can approximate any continuous function on $\mathcal{X} \times [0,1]^m$ by concatenating the finite dimensional feature vectors of a point cloud and a point. The proof of the theorem is given in Appendix B.

Theorem 4.1. *Let* $f \colon \mathcal{X} \times [0,1]^m \to \mathbb{R}$ *be a continuous function. Then for any* $\epsilon > 0$*, there exist* $K \in \mathbb{N}$ *and continuous maps* $\psi_1 \colon \mathcal{X} \to \mathbb{R}^K, \psi_2 \colon \mathbb{R}^{K+m} \to \mathbb{R}$ *such that*

$$\sup_{X \in \mathcal{X}} \int_{[0,1]^m} \left(f(X,x) - \psi_2([\psi_1(X)^\top, x^\top]^\top)\right)^2 \mathrm{d}x < \epsilon.$$

5 Experiments

To investigate the performance of our method in classification tasks, we conducted numerical experiments on two types of public real-world datasets. In §5.1, we present a simple experiment on a protein dataset to demonstrate the efficacy of our method. Next, in §5.2, we show the result of a experiment on a 3D CAD dataset. The classification accuracy was used as the evaluation metric.

The first experiment was conducted to show the validity of our method where we only used the topological feature, without DNN features. In the second experiment, on the other hand, we combine the topological features and features from DNNs using the method described in §3.2.

All the code was implemented in Python 3.9.13 with PyTorch 1.10.2. The experiments were conducted on CentOS 8.1 with AMD EPYC 7713 2.0 GHz CPU and 512 GB memory. Persistent homology was calculated by Python interface of GUDHI[1]. All of the source codes we used for experiments is publically available [2]. More details of the experiments can be found in Appendix C.

5.1 Protein classification

Dataset. In this section, we utilize the protein dataset in Kovacev-Nikolic et al. (2016) consisting of dense configuration data of 14 types of proteins. The authors of the paper analyzed the maltose-binding protein (MBP), whose topological structure is important for investigating its biological role. They constructed a dynamic model of 370 essential points each for these 14 types of proteins and calculated the cross-correlation matrix C for each type[3]. They then define the distance matrix, called the *dynamical distance*, by $D_{ij} = 1 - |C_{ij}|$, and they use them to classify the proteins into two classes, "open" and "close" based on their topological structure. In this paper, we subsampled 60 points and used the distance matrices with the shape 60×60 for each instance. We subsampled 1,000 times, half of which were from open and the rest were from closed. The off-diagonal elements of the distance matrix were perturbed by adding noise from a normal distribution with a standard deviation of 0.1.

Architectures and comparison baselines. Given a distance matrix, we computed the proposed adaptive filtration and calculate the persistent homology. Then, the 1st persistence diagram of that filtration was vectorized by PersLay. The obtained feature was input into a classifier using a linear model, and all the models were trained with cross entropy loss. For comparison, we evaluate the accuracy when we replace our learnable filtration with a fixed filtration. As a fixed filtration, we used Rips or DTM filtration (Anai et al., 2020), the latter of which is robust to outliers. The hyperparameter for DTM filtration was set to maximize the classification accuracy. Additionally, we replaced our topological feature by the output of (2), which we call DistMatrixNet, and compared the classification accuracy. Note that we used DistMatrixNet not for the computation of filtration weights but for the direct computation of point clouds' feature. Since the protein dataset is given in the format of distance matrices, we did not utilize standard neural network methods for point clouds.

Results. We present the results of the binary classification task of proteins in Table 1. We can see that our framework overperformed the other methods in terms of the accuracy of the classification.

We describe some observations below: Firstly, the result of our method is better than those of Rips and DTM filtrations. This would be because our method learned a weighted filtration adaptively to data and tasks and could provide more informative topological features than the classical ones. Secondly, our method achieved higher accuracy compared to DistMatrixNet, which has a similar

---

[1] `https://gudhi.inria.fr/`

[2] `https://github.com/git-westriver/FiltrationLearningForPointClouds`

[3] The correlation matrix C can be found in `https://www.researchgate.net/publication/301543862_corr`.

Table 1: The accuracy of the binary classification task of protein structure. We compared our methods with DistMatrixNet and persistent homology with Rips/DTM filtration. We can see that our method performs better than other three methods.

DistMatrixNet	Rips	DTM	Ours
65.0 ± 12.0	79.9 ± 3.0	78.0 ± 1.6	**81.9 ± 2.1**

number of parameters to the proposed method. This suggests that persistent homology is essential in the proposed method rather than the expressive power of DistMatrixNet.

5.2 3D CAD data classification

Dataset. In this section, we deal with the classification task on ModelNet10 (Wu et al., 2015). This is a dataset consisting of 3D-CAD data given by collections of polygons of 10 different kinds of objects, such as beds, chairs, and desks. We choose 1,000 point clouds from this dataset so that the resulting dataset includes 100 point clouds for each class. The corresponding point cloud was created with the sampling from each polygon with probability proportional to its area. Moreover, to evaluate the performance in the case where the number of points in each point cloud is small, we subsampled 128 points from each point cloud. The number of points is relatively small compared with the previous studies, but this setting would be natural from the viewpoint of practical applications. We added noise to the coordinates of each sampled point using a normal distribution with a standard deviation of 0.1.

Architecture and comparison baselines. We utilized the two-phase training procedure described in §3.2, where we concatenated a feature computed by DNN-based method and a topological feature. As DNN-based methods, which were trained through the 1st Phase, we utilized DeepSets (Zaheer et al., 2017), PointNet (Qi et al., 2017a), and PointMLP (Ma et al., 2022). A given point cloud was input into our adaptive filtration architecture, and the resulting 1st persistence diagram was vectorized by PersLay. The concatenated features were input into a classifier based on a linear model, and the parameters are tuned with the cross entropy loss.

We compared the result of 1st Phase, which is the accuracy of a DNN alone, and that of 2nd Phase to validate the efficacy of our learnable filtration. We also compared our method with a fixed filtration, Rips or DTM, instead of our learnable filtration. Furthermore, we replaced our topological feature replaced by the output of DistMatrixNet (2) and compared the classification accuracy.

Results. The results are shown in Table 2. Below, we present some observations for the results.

Firstly, our method outperformed Rips filtration, which is one of the most common filtrations. Moreover, the accuracy of our method is almost the same as or higher than that of DTM filtration. While DTM filtration has hyperparameters that need to be tuned, our method achieved such a result by learning an adaptive filtration automatically through the training. This result implies that our method is useful in efficiently choosing filtrations.

Secondly, our method again overperformed DistMatrixNet. This indicates that the combination of persistent homology and DistMatrixNet was crucial similarly to the protein dataset.

Thirdly, and most importantly, it can be observed that the classification accuracy is better when our method was concatenated with DeepSets/PointNet compared to using DeepSets/PointNet alone. Additionally, the accuracy when we combine our method and DeepSets/PointNet is competitive when using PointMLP alone. These observations indicate that concatenating the topological features obtained by our method yields positive effects. The accuracy when we combine our method and PointMLP is not higher than that of PointMLP. This would mean that PointMLP has already captured enough information including topological features during the 1st phase, so that the information obtained by persistent homology may be redundant, potentially negatively impacting the classification.

6 Conclusion

In this paper, we tackled the problem to obtain adaptive topological features of point clouds. To this end, we utilize a weighted filtration and train a neural network that generates a weight for each point.

Table 2: Results for the classification task of 3D CAD data. The row named "1st Phase" shows the results when we used the feature calculated by DNN only, and the rows named "2nd Phase" shows the results when we concatenated them with the feature calculated by DistMatrixNet or topological feature obtained by persistent homology. Compared to using the DeepSets/PointNet alone, we can achieve higher accuracy when we concatenate the topological feature obtained by our method. Additionally, regardless which DNN method we use, using topological feature computed by our method improves accuracy better than using topological feature computed by Rips filtration and feature computed by DistMatrixNet. Moreover, the accuracy when we combine our method with DeepSets/PointNet is competitive with using PointMLP alone.

		DeepSets	PointNet	PointMLP
# of parameters		813,488	3,472,500	3,524,386
1st Phase		65.7 ± 1.4	64.3 ± 4.4	**68.8 ± 6.3**
2nd Phase	DistMatrixNet	65.7 ± 4.8	55.7 ± 13.9	53.8 ± 7.4
	Rips	67.0 ± 2.6	68.4 ± 2.4	57.8 ± 12.4
	DTM	**68.0 ± 2.5**	68.7 ± 2.3	57.2 ± 6.8
	Ours	**67.5 ± 2.5**	**68.8 ± 2.0**	60.0 ± 6.3

Since the model is desired to be invariant with respect to isometric transformation, we proposed a network architecture that satisfies such invariance. We theoretically showed a finite-dimensional approximation property for a wide class of weight functions, which supports the validity of our architecture. Our numerical experiments demonstrated the effectiveness of our method in comparison with the existing methods.

Limitations and future work There are several limitations and remained future work for this study. First, while we concentrated on determining the weight function of the weighted Rips filtration in this paper, we can also consider other kinds of filtrations. For example, we can expand the balls from each point at different speeds. Second, we did not applied our method to larger datasets as we needed to compute persistent homology repeatedly in the training, which make the computational cost higher. In fact, it took about seven hours to train the neural networks in our method for each cross-validation. While we believe our method is meaningful since it sometimes improves classification accuracy, this is one of the largest limitations of our study. Addressing this issue is one of our future work. Third, since our method adaptively chooses a filtration, our framework is expected to work well even if a subsequent machine learning model has a simple architecture. We plan to validate the hypothesis experimentally and theoretically.

Acknowledgments

We appreciate Marc Glisse and Théo Lacombe for the helpful discussion. This work was partially supported by Grant-in-Aid for Transformative Research Areas (A) (22H05107) and JST KAKENHI JP19H01114.

References

H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18, 2017.

H. Anai, F. Chazal, M. Glisse, Y. Ike, H. Inakoshi, R. Tinarrage, and Y. Umeda. DTM-based filtrations. In *Topological Data Analysis*, pages 33–66. Springer, 2020.

P. Bendich, D. Cohen-Steiner, H. Edelsbrunner, J. Harer, and D. Morozov. Inferring local homology from sampled stratified spaces. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 536–546. IEEE, 2007.

M. Boutin and G. Kemper. On reconstructing n-point configurations from the distribution of distances or areas. *Advances in Applied Mathematics*, 32(4):709–735, 2004.

P. Bubenik and P. T. Kim. A statistical approach to persistent homology. *Homology, homotopy and Applications*, 9(2):337–362, 2007.

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

Z. Cang and G.-W. Wei. Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions. *PLoS computational biology*, 13(7):e1005690, 2017.

Z. Cang, L. Mu, and G.-W. Wei. Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening. *PLoS computational biology*, 14(1):e1005929, 2018.

M. Carrière, 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 *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796. PMLR, 2020.

M. Carrière, F. Chazal, M. Glisse, Y. Ike, H. Kannan, and Y. Umeda. Optimizing persistent homology based functions. In *International Conference on Machine Learning*, pages 1294–1303. PMLR, 2021.

M. K. Chung, P. Bubenik, and P. T. Kim. Persistence diagrams of cortical surface data. In *International Conference on Information Processing in Medical Imaging*, pages 386–397. Springer, 2009.

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. *arXiv preprint arXiv:2202.01725*, 2022.

H. Edelsbrunner. *Weighted alpha shapes*. University of Illinois at Urbana-Champaign, 1992.

N. Giansiracusa, R. Giansiracusa, and C. Moon. Persistent homology machine learning for fingerprint classification. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1219–1226. IEEE, 2019.

Y. Hiraoka, T. Nakamura, A. Hirata, E. G. Escolar, K. Matsue, and Y. Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016.

C. Hofer, R. Kwitt, M. Niethammer, and A. Uhl. Deep learning with topological signatures. *Advances in Neural Information Processing Systems*, 30, 2017.

C. Hofer, F. Graf, B. Rieck, M. Niethammer, and R. Kwitt. Graph filtration learning. In *International Conference on Machine Learning*, pages 4314–4323. PMLR, 2020.

M. Horn, E. De Brouwer, M. Moor, Y. Moreau, B. Rieck, and K. Borgwardt. Topological graph neural networks. In *International Conference on Learning Representations*, 2021.

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

G. Kusano, K. Fukumizu, and Y. Hiraoka. Kernel method for persistence diagrams via kernel embedding and weight factor. *The Journal of Machine Learning Research*, 18(1):6947–6987, 2017.

W. Liu, H. Guo, W. Zhang, Y. Zang, C. Wang, and J. Li. TopoSeg: Topology-aware segmentation for point clouds. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1201–1208. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/168. URL `https://doi.org/10.24963/ijcai.2022/168`. Main Track.

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 *International Conference on Learning Representations*, 2022.

Z. Meng, D. V. Anand, Y. Lu, J. Wu, and K. Xia. Weighted persistent homology for biomolecular data analysis. *Scientific reports*, 10(1):2079, 2020.

T. Nakamura, Y. Hiraoka, A. Hirata, E. G. Escolar, and Y. Nishiura. Persistent homology and many-body atomic structure for medium-range order in the glass. *Nanotechnology*, 26(30):304001, 2015.

I. Obayashi, T. Nakamura, and Y. Hiraoka. Persistent homology analysis for materials research and persistent homology software: HomCloud. *Journal of the Physical Society of Japan*, 91(9):091013, 2022.

N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6:1–38, 2017.

C. R. Qi, H. Su, K. Mo, and L. 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*, pages 652–660, 2017a.

C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017b.

A. Som, H. Choi, K. N. Ramamurthy, M. P. Buman, and P. Turaga. Pi-Net: A deep learning approach to extract topological persistence images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 834–835, 2020.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

D. Widdowson and V. Kurlin. Recognizing rigid patterns of unlabeled point clouds by complete and continuous isometry invariants with no false negatives and no false positives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1275–1284, 2023.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

J. Xu, X. Tang, Y. Zhu, J. Sun, and S. Pu. SGMNet: Learning rotation-invariant point cloud representations via sorted gram matrix. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10468–10477, 2021.

Z. Yan, T. Ma, L. Gao, Z. Tang, Y. Wang, and C. Chen. Neural approximation of extended persistent homology on graphs. *arXiv preprint arXiv:2201.12032*, 2022.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep Sets. *Advances in Neural Information Processing Systems*, 30, 2017.

M. Zeppelzauer, B. Zieliński, M. Juda, and M. Seidl. Topological descriptors for 3d surface analysis. In *Computational Topology in Image Context: 6th International Workshop, CTIC 2016, Marseille, France, June 15-17, 2016, Proceedings 6*, pages 77–87. Springer, 2016.

S. Zhang, S. Mukherjee, and T. K. Dey. Gefl: Extended filtration learning for graph classification. In *Learning on Graphs Conference*, pages 16–1. PMLR, 2022.

C. Zhou, Z. Dong, and H. Lin. Learning persistent homology of 3d point clouds. *Computers & Graphics*, 102:269–279, 2022.

—— Appendix ——

A Simplicial complexes and filtrations

In this appendix, we briefly recall the definitions of simplicial complexes and filtrations.

Definition A.1. Let V be a finite set. A (finite) *simplicial complex* whose vertex set is V is a collection K of subsets of V satisfying the following conditions.

(1) $\emptyset \notin K$;

(2) for any $v \in V$, $\{v\} \in K$;

(3) if $\sigma \in K$ and $\emptyset \neq \tau \subset \sigma$, then $\tau \in K$.

An element σ of K with the cardinality $\#\sigma = k + 1$ is called a k-simplex.

A *subcomplex* of a simplicial complex K is a subset K' of K that is a simplcial complex itself.

Definition A.2. A filtration of a simplicial complex K is an increasing family $(K_t)_{t\in\mathbb{R}}$ of subcomplexes of K.

Given a point cloud in $\mathbb{R}^d$, we can construct the Čech filtration as follows.

Definition A.3. Let $X \subset \mathbb{R}^d$ be a point cloud and $t \in \mathbb{R}$. One defines a simplicial complex $\check{C}[X]_t$ as follows:

$$\check{C}[X]_t := \left\{ \emptyset \neq \sigma \subset X \;\middle|\; \bigcap_{x\in\sigma} B(x;t) \neq \emptyset \right\}.$$

The family $(\check{C}[X]_t)_{t\in\mathbb{R}}$ forms a filtration, which is called the *Čech filtration* of X.

The Čech complex $\check{C}[X]_t$ is known to be homotopy equivalent to the union of balls $\bigcup_{x\in X} B(x,t)$. However, it takes much computational cost to compute the Čech complex. Therefore, we utilize the Rips filtration instead.

Definition A.4. Let $X \subset \mathbb{R}^d$ be a point cloud and $t \in \mathbb{R}$. One defines a simplicial complex $R[X]_t$ by

$$\{x_0, \dots, x_k\} \in R_p[X]_t :\Longleftrightarrow \|x_i - x_j\| \leq 2t \text{ for any } i, j \in \{0, \dots, k\}.$$

The family $(R[X]_t)_{t\in\mathbb{R}}$ forms a filtration, which is called the *Rips filtration* of X.

When a point cloud X is equipped with a weight function on X, we obtain the weighted Rips filtration as follows. This definition is totally same as that of Anai et al. (2020).

Definition A.5. Let $X \subset \mathbb{R}^d$ be a point cloud, $w\colon X \to \mathbb{R}$ a function. One defines a simplicial complex $R[X, w]_t$ by

$$\begin{aligned}&\{x_0, \dots, x_k\} \in R[X, w]_t \\ &\qquad :\Longleftrightarrow t \geq w(x_i), t \geq w(x_j), \|x_i - x_j\| \leq 2t - w(x_i) - w(x_j) \text{ for any } i, j \in \{0, \dots, k\}.\end{aligned}$$

The family $(R[X, w]_t)_{t\in\mathbb{R}}$ forms a filtration, which is called the *weighted Rips filtration* of X.

DTM filtration, which we mentioned in §2.2, is one type of weighted filtration with the weight function $w\colon \mathbb{R}^m \to \mathbb{R}$ defined by

$$w(x) = \left(\frac{1}{k_0} \sum_{k=1}^{k_0} \|x - p_k(x)\|^q \right)^{\frac{1}{q}}, \tag{3}$$

where $p_1(x), \dots, p_{k_0}(x)$ are choice of k_0-nearest neighbors of x in $\mathbb{R}^m$.

B Proofs of the theorem in Section 4

Recall that we define the space of point clouds $\mathcal{X}$ with

$$\mathcal{X} := \{X \subset [0,1]^m \mid |X| \leq N\},$$

where $N \in \mathbb{N}$ is a fixed natural number. Firstly, we prove the compactness of $\mathcal{X}$.

Lemma B.1. *The space $\mathcal{X}$ is compact.*

Proof. We prove this by induction on N. When $N = 0$, $\mathcal{X}$ is a singleton set, so the statement holds. Assume that the statement holds for all $N \leq N_0 - 1$ ($N_0 \geq 1$). We prove that $\mathcal{X}$ is sequentially compact for $N = N_0$.

Take an arbitrary sequence $(X_n)_{n\in\mathbb{N}}$ in $\mathcal{X}$. Then, we can choose $i \in \{1, \dots, N_0\}$ so that there exist an infinite number of $n \in \mathbb{N}$ such that $|X_n| = i$. Let $n(1), n(2), \dots$ be the sequence of n such that $|X_n| = i$, listed in increasing order. If $i < N_0$, by the assumption of induction, $(X_{n(k)})_{k\in\mathbb{N}}$ has a convergent subsequence, which means $(X_n)_{n\in\mathbb{N}}$ also has a convergent subsequence.

Suppose $i = N_0$. For each k, let x_k be the element of $X_{n(k)}$ that is the smallest in the lexicographic order, and let $Y_k := X_{n(k)} \setminus \{x_k\}$. By the compactness of $[0,1]^m$, the sequence $(x_k)_{k\in\mathbb{N}}$ has a convergent subsequence, which we denote by $(x_{k(l)})_{l\in\mathbb{N}}$. By the assumption of induction, the sequence $(Y_{k(l)})_{l\in\mathbb{N}}$ also has a convergent subsequence, which we denote by $(Y_{k'(l)})_{l\in\mathbb{N}}$. Then, the sequence $(x_{k'(l)})_{l\in\mathbb{N}}$ is also convergent. Let x^* and Y^* be the limits of $(x_{k'(l)})_{l\in\mathbb{N}}$ and $(Y_{k'(l)})_{l\in\mathbb{N}}$, respectively.

Take an arbitrary $\epsilon > 0$. Then we can choose $K_1, K_2 \in \mathbb{N}$ such that

$$\begin{aligned} l > K_1 &\Rightarrow \|x_{k'(l)} - x^*\| < \epsilon \\ l > K_2 &\Rightarrow d_H(Y_{k'(l)}, Y^*) < \epsilon. \end{aligned}$$

Here, for any $X, Y \subseteq \mathbb{R}^m$ and $x_0, y_0 \in \mathbb{R}^m$, we have

$$\begin{aligned} &d_H(X \cup \{x_0\}, Y \cup \{y_0\}) \\ &\quad = \max\{d(x_0, Y \cup \{y_0\}), \sup_{x\in X} d(x, Y \cup \{y_0\}), d(y_0, X \cup \{x_0\}), \sup_{y\in Y} d(y, X \cup \{x_0\})\} \\ &\quad \leq \max\{\|x_0 - y_0\|, \sup_{x\in X} d(x, Y), \|y_0 - x_0\|, \sup_{y\in Y} d(y, X)\} \\ &\quad \leq \max\{\|x_0 - y_0\|, d_H(X, Y)\} \end{aligned}$$

Therefore, for $K = \max\{K_1, K_2\}$ and $l > K$, we have

$$d_H(Y_{k'(l)} \cup \{x_{k'(l)}\}, Y^* \cup \{x^*\}) \leq \max\{d_H(Y_{k'(l)}, Y^*), \|x_{k'(l)} - x^*\|\} < \epsilon,$$

which implies that $(X_{n(k'(l))})_{l\in\mathbb{N}} = (Y_{k'(l)} \cup \{x_{k'(l)}\})_{l\in\mathbb{N}}$ converges to $Y^* \cup \{x^*\}$. Therefore, $(X_n)_{n\in\mathbb{N}}$ has a convergent subsequence. Thus, the claim is proven. ☐

In the following, assume that $L^2([0,1]^m)$ is the space of square-integrable functions defined on $[0,1]^m$.

Next, we show that any functions in a compact subspace of $L^2([0,1]^m)$ can be approximated by the linear combination of finite number of functions in $L^2([0,1]^m)$.

Lemma B.2. *Let $\mathcal{G}$ be a compact subspace of $L^2([0,1]^m)$. Then, for any $\epsilon > 0$, there exist $K \in \mathbb{N}$ and $e_1, \dots, e_K \in \mathcal{G}$ such that $\|e_1\|_{L_2} = \cdots = \|e_K\|_{L_2} = 1$ and*

$$\sup_{g\in\mathcal{G}} \int_{[0,1]^m} \left(g(x) - \sum_{k=1}^{K} \langle g, e_k \rangle \cdot e_k(x) \right) \mathrm{d}x < \epsilon.$$

Proof. Take $\epsilon > 0$ arbitrarily. Since $\mathcal{G}$ is compact, there exists $K' \in \mathbb{N}$ such that $g_1, \dots, g_{K'}$ such that $\min_{i=1,\dots,K'} \|g - g_i\|_{L_2} < \epsilon$ holds for any $g \in \mathcal{G}$. Let $\{e_1, \dots, e_K\}$ ($K \in \mathbb{N}, K \leq K'$) be a orthonormal basis of $\mathrm{span}\{g_1, \dots, g_{K'}\}$. Since $\sum_{k=1}^{K} \langle g, e_k \rangle \cdot e_k(x)$ is a projection of $g \in L_2([0,1]^m)$ on $\mathrm{span}\{g_1, \dots, g_{K'}\}$, for any $g \in \mathcal{G}$, we have

$$\left\| g - \sum_{k=1}^{K} \langle g, e_k \rangle \cdot e_k \right\|_{L_2} \leq \min_{i=1,\dots,K'} \|g - g_i\|_{L_2} < \epsilon.$$

This completes the proof. ☐

The following two lemmas show that the continuity of the map $X \mapsto f(X, \cdot)$, which is proved using the continuity of the map $(X, x) \mapsto f(X, x)$.

Lemma B.3. *Let* $f : \mathcal{X} \times [0,1]^m \to \mathbb{R}$ *be a continuous function and let* $X_0 \in \mathcal{X}$. *For any* $\epsilon > 0$, *there exists* $\delta > 0$ *such that*

$$d_H(X_0, X) < \delta \Rightarrow \sup_{x \in [0,1]^m} |f(X, x) - f(X_0, x)| < \epsilon$$

holds.

Proof. Since $[0,1]^m$ is compact, $\{X_0\} \times [0,1]^m$ is also compact in the product topology. Note that the product topology is equivalent to the topology induced by the metric $d_\infty((X, x), (Y, y)) := \max\{d_H(X, Y), \|x - y\|_2\}$.

Take an arbitrary $\epsilon > 0$. Since f is continuous, for any $x \in [0,1]^m$, we can choose $\delta_x > 0$ such that

$$y \in [0,1]^m, d_\infty((X_0, y), (X_0, x)) < \delta_x \Rightarrow |f(X_0, y) - f(X_0, x)| < \epsilon/2.$$

Let $\bigcup_x B_{\delta_x}(x)$ be an open cover of $\{X_0\} \times [0,1]^m$. Then, by the compactness of $\{X_0\} \times [0,1]^m$, there exist $n \in \mathbb{N}$ and $x_1, \dots, x_n \in [0,1]^m$ such that $\bigcup_i B_{\delta_{x_i}}(x_i)$ is also an open cover of $\{X_0\} \times [0,1]^m$.

Let $\delta := \min\{\delta_1, \dots, \delta_n\}/2$. Suppose $X \in \mathcal{X}$ satisfies $d_H(X_0, X) < \delta$, and let x be an arbitrary point in $[0,1]^m$. Then, there exists $i \in \{1, \dots, n\}$ such that

$$\|x - x_i\|_2 = d_\infty((X_0, x), (X_0, x_i)) < \delta_i$$

Moreover, we have

$$d_\infty((X, x), (X_0, x_i)) = \max\{d_H(X, X_0), \|x - x_i\|_2\} < \delta_i$$

which implies that

$$|f(X_0, x) - f(X_0, x_i)| < \epsilon/2, |f(X, x) - f(X_0, x_i)| < \epsilon/2.$$

Therefore, we obtain $|f(X, x) - f(X_0, x)| < \epsilon$. Since δ does not depend on x, the claim is proved. ☐

Lemma B.4. *Let* $f\colon \mathcal{X} \times [0,1]^m \to \mathbb{R}$ *is a continuous function. Then, the map* $h\colon \mathcal{X} \to L^2([0,1]^m), X \mapsto f(X, \cdot)$ *is continuous.*

Proof. Take $\epsilon > 0$ and $X_0 \in \mathcal{X}_0$ arbitrarily. By Lemma B.3, there exists $\delta > 0$ such that

$$d_H(X, X_0) < \delta \Rightarrow \sup_{x \in [0,1]^m} |f(X, x) - f(X_0, x)| < \epsilon.$$

Therefore, if $d_H(X, X_0) < \delta$, we have

$$\|f(X, \cdot) - f(X_0, \cdot)\|_{L_2} = \sqrt{\int_{[0,1]^m} \left(f(X, x) - f(X_0, x)\right)^2 \mathrm{d}x} < \epsilon,$$

which implies the map $X \mapsto f(X, \cdot)$ is continuous. ☐

Thanks to the compactness of $\mathcal{X}$ (Lemma B.1), the finite approximation result (Lemma B.2) and the continuity of the map $X \mapsto f(X, \cdot)$ (Lemma B.4), we can approximate $f(X, \cdot)$ with a linear combination of functions in $L^2([0,1]^m)$, for all $X \in \mathcal{X}$.

Finally, we prove the following theorem, which is same as Theorem 4.1, using the fact that the space of continuous functions defined on $[0,1]^m$ is dense in $L^2([0,1]^m)$.

Theorem B.5. *Let* $f\colon \mathcal{X} \times [0,1]^d \to \mathbb{R}$ *be a continuous function. Then for any* $\epsilon > 0$, *there exist* $K \in \mathbb{N}$ *and continuous maps* $\psi_1\colon \mathcal{X} \to \mathbb{R}^K, \psi_2\colon \mathbb{R}^{K+m} \to \mathbb{R}$ *such that*

$$\sup_{X \in \mathcal{X}} \int_{[0,1]^m} \left(f(X, x) - \psi_2([\psi_1(X)^\top, x^\top]^\top)\right)^2 \mathrm{d}x < \epsilon.$$

Proof. Since the image of a compact space under a continuous map is compact, by Lemma B.1 and Lemma B.4, $\{f(X,\cdot) \mid X \in \mathcal{X}\}$ is a compact set. By Lemma B.2, there exist $K \in \mathbb{N}$ and $e_1, \dots, e_K$ such that $\|e_1\|_{L_2} = \cdots = \|e_K\|_{L_2} = 1$ and

$$\sup_{X\in\mathcal{X}} \int_{[0,1]^m} \left(f(X,x) - \sum_{k=1}^{K} \langle f(X,\cdot), e_k\rangle \cdot e_k(x) \right)^2 \mathrm{d}x < \epsilon/4.$$

By Lemma B.4, $X \mapsto f(X,\cdot)$ is continuous. Therefore, for any $\tau > 0$, there exists $\delta > 0$ such that $d_H(X,Y) < \delta \Rightarrow \|f(X,\cdot) - f(Y,\cdot)\| < \tau$ for any $X, Y \in \mathcal{X}$. Then, if $X, Y \in \mathcal{X}$ satisfy $d_H(X,Y) < \delta$, we have

$$\begin{aligned} |\langle f(X,\cdot), e_k\rangle - \langle f(Y,\cdot), e_k\rangle| &= |\langle f(X,\cdot) - f(Y,\cdot), e_k\rangle| \\ &\le \|f(X,\cdot) - f(Y,\cdot)\|_{L_2} \\ &< \tau. \end{aligned}$$

This means that the function $h\colon \mathcal{X} \to \mathbb{R}$ such that $h(X) = \langle f(X,\cdot), e_k\rangle$ is continuous. Since $\mathcal{X}$ is a compact space, $h(X)$ is bounded, i.e., there exists $B > 0$ such that, for any k, $|\langle f(X,\cdot), e_k\rangle| < B$ holds.

Since the space of continuous functions with compact support is dense in $L^2([0,1]^m)$, there exist continuous functions $\hat{e}_1, \dots, \hat{e}_K$ such that $\|e_k - \hat{e}_k\|_{L_2} < \sqrt{\epsilon}/2BK$.

Let $\psi_1\colon \mathcal{X} \to \mathbb{R}^K$, $\psi_2\colon \mathbb{R}^{K+m} \to \mathbb{R}$ be continuous maps such that

$$\begin{aligned} &\psi_1(X) = [\langle f(X,\cdot), e_1\rangle, \dots, \langle f(X,\cdot), e_K\rangle]^\top \\ &\psi_2([x_1, \dots, x_K, y]^\top) = \sum_{k=1}^{K} x_k \cdot \hat{e}_k(y) \quad (x_1, \dots, x_k \in \mathbb{R}, y \in [0,1]^m). \end{aligned}$$

Then, for any $X \in \mathcal{X}$, we have

$$\begin{aligned} &\sqrt{\int_{[0,1]^m} \left(f(X,x) - \sum_{k=1}^{K} \psi_2([\psi_1(X)^\top, x^\top]^\top) \right)^2 \mathrm{d}x} \\ &\quad< \sqrt{\int_{[0,1]^m} \left(f(X,x) - \sum_{k=1}^{K} \langle f(X,\cdot), e_k\rangle \cdot e_k(x) \right)^2 \mathrm{d}x} \\ &\qquad\qquad + \sqrt{\int_{[0,1]^m} \left(\sum_{k=1}^{K} \langle f(X,\cdot), e_k\rangle \cdot e_k(x) - \psi_2([\psi_1(X)^\top, x^\top]^\top) \right)^2 \mathrm{d}x} \\ &\quad< \frac{\sqrt{\epsilon}}{2} + \sqrt{\int_{[0,1]^m} \left(\sum_{k=1}^{K} \langle f(X,\cdot), e_k\rangle \cdot e_k(x) - \sum_{k=1}^{K} \langle f(X,\cdot), e_k\rangle \cdot \hat{e}_k(x) \right)^2 \mathrm{d}x} \\ &\quad\le \frac{\sqrt{\epsilon}}{2} + \sqrt{\int_{[0,1]^m} \left(\sum_{k=1}^{K} \langle f(X,\cdot), e_k\rangle \cdot (e_k(x) - \hat{e}_k(x)) \right)^2 \mathrm{d}x} \\ &\quad\le \frac{\sqrt{\epsilon}}{2} + \sqrt{\int_{[0,1]^m} \left(\sum_{k=1}^{K} \langle f(X,\cdot), e_k\rangle^2 \right) \left(\sum_{k=1}^{K} (e_k(x) - \hat{e}_k(x))^2 \right) \mathrm{d}x} \\ &\quad\le \frac{\sqrt{\epsilon}}{2} + \sqrt{KB^2 \sum_{k=1}^{K} \int_{[0,1]^m} (e_k(x) - \hat{e}_k(x))^2 \mathrm{d}x} \\ &\quad< \frac{\sqrt{\epsilon}}{2} + \sqrt{KB^2 \cdot K \left(\frac{\sqrt{\epsilon}}{2KB} \right)^2} = \sqrt{\epsilon}. \end{aligned}$$

This completes the proof. □

C Details of Experiment Settings

C.1 Evaluation

In each experiment, 5-fold cross-validation was conducted, and average and standard deviation of the five results are shown in the result section.

C.2 Optimization

We set the batch size as 40. As an optimization algorithm, we used Adam. We used the following learning rate scheduler based on the method proposed in Vaswani et al. (2017):

$$\eta = \eta_{\max} \cdot \min\left\{ \frac{1}{\mathrm{epoch}^{\frac{1}{2}}}, \frac{\mathrm{epoch}}{N_{\mathrm{warmup}}^{\frac{3}{2}}} \right\}.$$

We set $N_{\mathrm{warmup}} = 40$. Additionally, we set $\eta_{\max} = 2.0 \times 10^{-2}$ when we train DeepSets or PointNet in the 1st phase and $\eta_{\max} = 1.0 \times 10^{-2}$ in other training (PointMLP in the 1st phase, and all in the 2nd phase). For the protein dataset, the number of epochs was 200. For ModelNet10, the number of epochs in the 1st phase was 1000, and that of the 2nd phase was 200.

C.3 Settings related to persistent homology

For PersLay, the dimension of the parameter c was 2×32, i.e., $c \in \mathbb{R}^{2\times 32}$. Additionally, we set all of the permutation invariant operator $\mathbf{op}$ which appears in PersLay as summation. The dimension of the vectors computed by PersLay is 32. We convert this vector to the feature vector whose dimension is 16 using a linear model.

For the weighted function in the DTM filtration (see Equation (3)), we fixed the value of q at 2, and we chose the hyperparameter k_0 from among 2, 3, 5, 8, and 10 to maximize the classification accuracy through cross-validation. See Appendix E.2 for the results with other values of k_0.

C.4 Networks

We set all of the permutation invariant operator $\mathbf{op}$ which appears in our method and DeepSets as summation. The dimension of the feature vectors obtained by PersLay, DeepSets, PointNet, PointMLP was set as 16.

Next, we describe the architectures of DNN-based methods. In the following, we denote multi-layer parceptrons whose dimension of input is d_{in}, that of output is d_{out} and the number of neurons in the middle layers $d_1, d_2, \ldots$ by $\mathrm{MLP}(d_{in}, d_1, d_2, \ldots, d_{out})$. Batch normalization was performed after each layer. Regarding the proposed method, we set $\phi^{(1)} = \mathrm{MLP}(1, 64, 128, 256)$, $\phi^{(2)} = \mathrm{MLP}(256, 128, 64, 8)$, $\phi^{(3)} = \mathrm{MLP}(1, 64, 128, 256)$, $\phi^{(4)} = \mathrm{MLP}(256, 256, 256, 256)$, $\phi^{(5)} = \mathrm{MLP}(256, 128, 64, 16)$ and $\phi^{(6)} = \mathrm{MLP}(24, 256, 512, 256, 1)$. Dropout was performed for the last two layers of $\phi^{(2)}$ and $\phi^{(5)}$. As for DeepSets, which is defined by $\mathrm{DeepSets}(X) := \phi_2(\mathbf{op}(\{\phi_1(x_i)\}_{i=1}^N))$, we set $\phi_1 = \mathrm{MLP}(\texttt{input\_dim}, 64, 64, 64, 128, 1024)$ and $\phi_2 = \mathrm{MLP}(1024, 512, 256, 16)$. Dropout was performed for the last two layers of ϕ_2. For PointNet, we used the same architecture as that of used in Qi et al. (2017a), except for the dimension of the output, which we set 16. Regarding PointMLP, we set $\texttt{dim\_expansion} = [2, 2, 2], \texttt{pre\_blocks} = [2, 2, 2], \texttt{pos\_blocks} = [2, 2, 2], \texttt{k\_neighbors} = [24, 24, 24], \texttt{reducers} = [2, 2, 2]$.

Finally, we describe the initialization of the network parameters. In the experiment on protein dataset, we initialized the parameters in the network to compute the weight in our method with normal distribution with a mean of 0 and a standard deviation of 1.0×10^{-4}, while other parameters was initialized with the default settings of PyTorch. For the experiment on ModelNet, all of the parameters was initialized with the default settings of PyTorch.

C.5 Computational cost of persistent homology

First, we estimate the computational cost of our experiments. The computational complexity of persistent homology is known to be cubic in the number of simplices in the worst case. This fact can be found in, for example, the last sentence of Section 5.3.1 in Otter et al. (2017). Since we consider 1-dimensional persistent homology, we need to consider up to 2-simplices in the Rips complex, so the number of simplices is

$$\binom{N}{1} + \binom{N}{2} + \binom{N}{3} = \frac{1}{6}N(N^2 + 5),$$

where N is the number of points. For the protein dataset, the number of points is 60, so the number of simplices is 36,050. For the 3D CAD dataset, the number of points is 128, so the number of simplices is 349,632. Since the number of simplices is $O(N^3)$, the computational cost can be upper-bounded by $O(N^9)$ in our case.

D More Observation on the Approximation Ability of the Weight Function

In §4, we provided the theoretical result that demonstrates the validity of the concatenation finite dimensional vectors. While this result partially justifies our network architecture, we did not show that the continuous maps ψ_1 and ψ_2 can be approximated by the networks. Instead of providing further theoretical results, we show that our network can recover the DTM function. We trained our network $f_\theta(X, x)$ for the regression task to approximate DTM functions (3). We show the approximation error in Table 3. We can observe that the approximation error is small enough. This result means that our method can choose filtrations from the space including Rips and DTM filtration if trained appropriately.

Table 3: The approximation error of DTM functions by the proposed network $f_\theta(X, x)$ for various values of k_0. In this experiment, the hyperparameter q of DTM function was fixed at 2. We can observe that DTM function can be approximated by our network $f_\theta(X, x)$ with small error.

value of k_0	error
0 (Rips)	0.0000 ± 0.0000
2	0.0015 ± 0.0000
3	0.0016 ± 0.0000
4	0.0017 ± 0.0000
5	0.0018 ± 0.0000
10	0.0022 ± 0.0001

We leave it as future work to provide stronger theoretical results for our network or to propose a new architecture with a complete approximation guarantee. There are several previous studies that could help solve this task. For example, the study by Boutin and Kemper (2004) shows that in most cases, the arrangement of points can be uniquely determined from the distribution of distances between pairs of points. Additionally, Widdowson and Kurlin (2023) showed that Simplexwise Centered Distribution (SCD) is complete isometry invariant and continuous. These results would help us to obtain theoretical guarantees of our network or to propose a new network architecture with stronger theoretical guarantees.

E More Observations on Numerical Experiments

E.1 Visualization of weight function and the difference of persistent diagrams

In this section, we visualize some of the weight functions that are learned by the proposed method. Additionally, we draw the persistence diagrams for Rips filtration and the learned filtration.

For the protein dataset, we visualize the resulting weight function in Figure 3, and show the persistence diagram with/without the weight function in Figure 4. We can see that our method gives different weights depending on the points and the point clouds. However, we cannot observe large differences between the persistence diagrams. Developing the methods that can learn much interpretable weight function is our future work.

For the 3D CAD dataset, we show the weight functions and the persistence diagram in Figure 5 and 6, respectively. We can observe that the middle and the rightmost point clouds have non-trivial weight functions, which may lead to improving the classification accuracy. On the other hand, we can observe that the leftmoft point cloud has a trivial weight function, which means that our method does not have any advantages in this case.

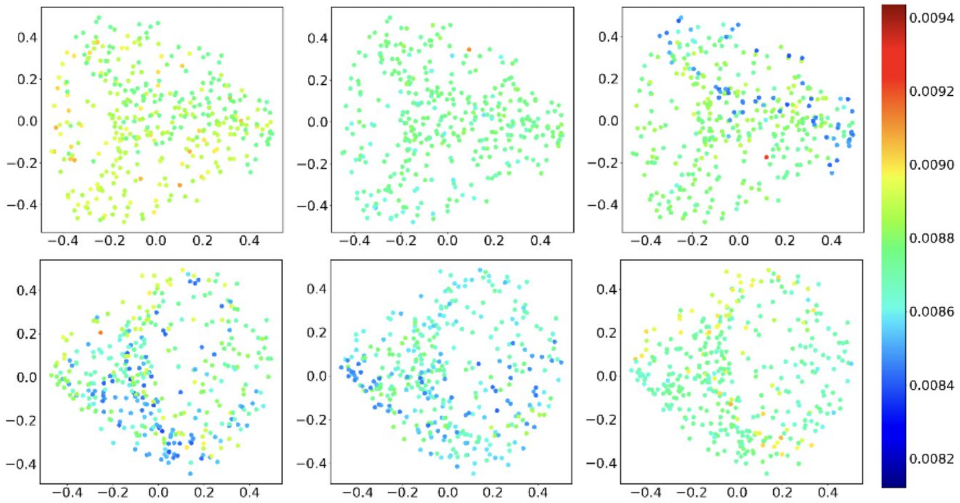

Figure 3: Some examples of the weight functions in the experiment for the protein dataset. These six point clouds correspond to different proteins. The point clouds in the first row belongs to *open*, and those in the second row belongs to *close*. Since the original dataset are given by the distance matrices, we visualize the point clouds using the method called multi-dimensional scaling. We visualize the weights for all of the 370 points, which include the 60 points used for training. We can see that the weight depends on each point. Additionally, we can observe that different weight functions are learned for different point clouds.

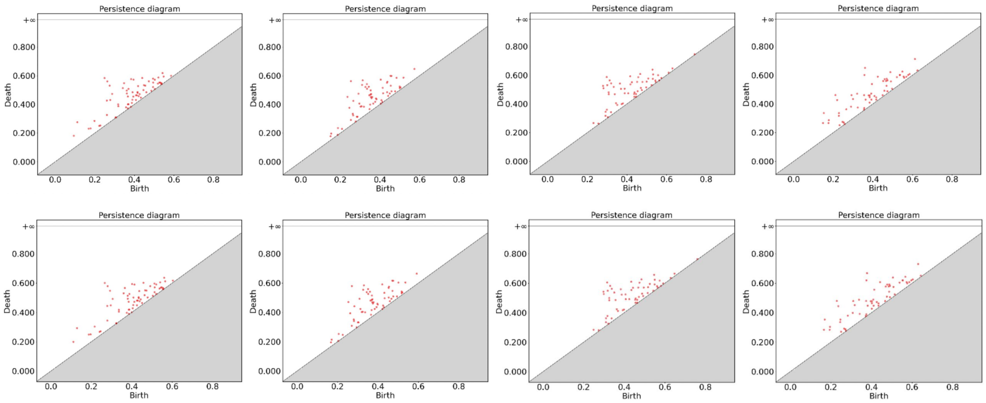

Figure 4: The persistence diagrams for the Rips filtration and the weighted filtration learned by our method (protein dataset). The first row corresponds to the Rips filtration, and the second row corresponds to the weighted filtration. The persistence diagrams in the first and second columns are associated with the point clouds with the label *open*, and those in the third and fourth columns are associated with the point clouds with the label *close*.

E.2 Results for DTM with various hyperparameters

As mentioned in §5.2, in our two experiments, our adaptive filtration achieved classification accuracy equal to or higher than the DTM filtration with the optimal value of k. In this chapter, we compare our method with DTM filtration using different values of the parameter k, while keeping q fixed at 2. The experimental results are presented in Table 4 and 5.

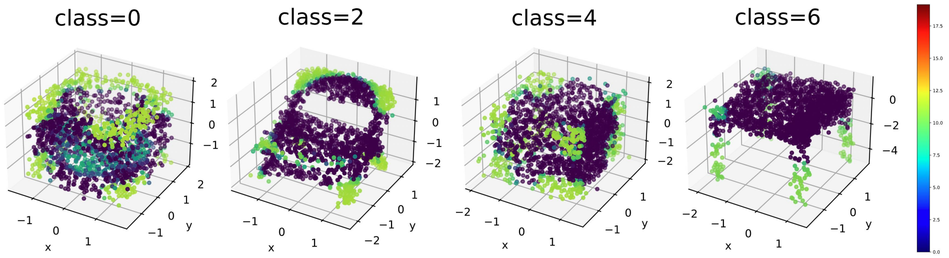

Figure 5: Some examples of the weight functions in the experiment for the 3D CAD dataset. Different point clouds in this figure represent different labels. We visualize the weights for 1920 points, which include the 128 points used for training. We can see that our method assigns small weight values to points near the edges of the objects and points corresponding to the legs of the desks. This would help improving the classification accuracy.

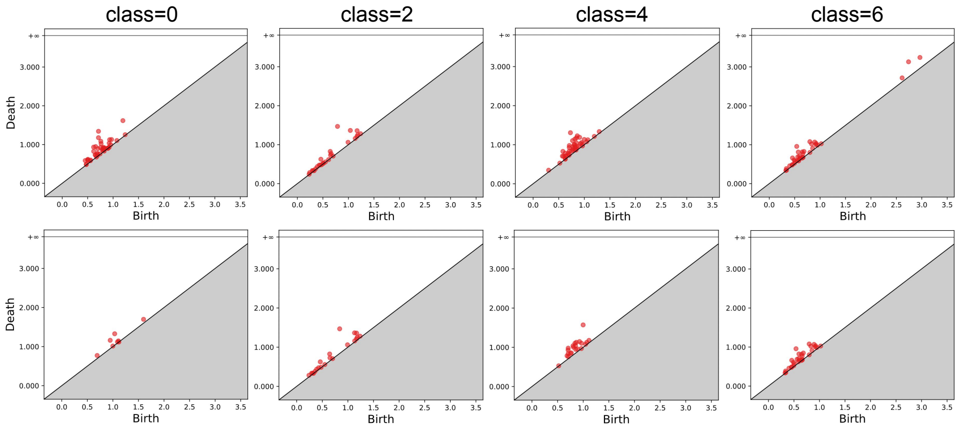

Figure 6: The persistence diagrams for Rips filtration and the weighted filtration learned by our method (3D CAD dataset). The first row corresponds to Rips filtration, and the second row corresponds to our method. The four columns in this figure correspond to the point clouds in Figure 5. We can see that the persistence diagrams for the weighted filtration learned by our method are different from those for the Rips filtration.

Table 4: The results for the protein dataset including DTM filtration with various hyperparameter settings.

DTM($k = 2$)	DTM($k = 3$)	DTM($k = 5$)	DTM($k = 8$)	DTM($k = 10$)	Ours
73.0 ± 2.9	71.1 ± 3.4	74.2 ± 1.5	77.3 ± 1.9	78.0 ± 1.6	**81.9 ± 2.1**

Table 5: The results for ModelNet10 including DTM filtration with various hyperparameter settings.

		DeepSets	PointNet	PointMLP
1st Phase		65.7 ± 1.4	64.3 ± 4.4	**68.8 ± 6.3**
2nd Phase	Rips	67.0 ± 2.6	68.4 ± 2.4	57.8 ± 12.4
	DTM ($k = 2$)	**67.7 ± 2.6**	68.3 ± 2.5	53.0 ± 9.6
	DTM ($k = 3$)	**68.0 ± 2.5**	**68.7 ± 2.3**	51.8 ± 9.4
	DTM ($k = 5$)	**67.6 ± 2.5**	67.9 ± 1.5	56.3 ± 7.6
	DTM ($k = 8$)	66.9 ± 3.1	67.5 ± 2.1	57.2 ± 9.8
	DTM ($k = 10$)	**67.5 ± 2.5**	67.7 ± 1.8	57.2 ± 6.8
	Ours	**67.5 ± 2.5**	**68.8 ± 2.0**	**60.0 ± 6.3**

E.3 Ablation studies for the proposed network

In our method, we used the networks with a huge number of parameters, which can cause overfitting. To address this concern, we conduct experiments with the reduced number of parameters to observe the changes in classification accuracy, especially for the protein dataset. In addition to the experimental results presented in the main text, which we call (R0), we conducted four experiments, (R1), (R2), (R3), and (R4), with reduction of the number of parameters. Table 6 shows the number of the architectures in the networks $\phi^{(1)}, \ldots, \phi^{(6)}$ used in each experiment. In (R1), we reduced the number of parameters in the networks, $\phi^{(1)}, \phi^{(2)}, \phi^{(3)}, \phi^{(4)}, \phi^{(5)}$, which are used to extract features from the entire point cloud and each point. Furthermore, in (R2), we decreased the number of parameters in the network $\phi^{(6)}$, which calculates weights using the extracted features. In (R3), we adopted both reductions, and in (R4), we further reduced the number of the intermediate layers from three to one in $\phi^{(4)}$.

We present the result in Table 7. We can see that the accuracy of (R1) and (R3) is lower than that of the original experiment (R1). This would mean that the parameter reduction in the networks $\phi^{(1)}, \phi^{(2)}, \phi^{(3)}, \phi^{(4)}, \phi^{(5)}$, which extract global/local features, has negative impact on the accuracy. On the other hand, the classification accuracy of (R2) and (R4) is a little better than that of (R0). This might mean that we can improve the classification accuracy by decreasing the number of parameters in $\phi^{(6)}$.

Since the difference between the accuracy of (R0) and that of (R2) is small, we believe that overfitting due to an excessive number of parameters would not be a significant issue for this experiment.

Table 6: Network layers in the ablation studies. For each experiment, the number of neurons in each layer of the MLP used in the networks $\phi^{(1)}, \ldots, \phi^{(6)}$ is specified in each element of the table.

	$\phi^{(1)}$	$\phi^{(2)}$	$\phi^{(3)}$	$\phi^{(4)}$	$\phi^{(5)}$	$\phi^{(6)}$
(R0)	1, 64, 128, 256	256, 128, 64, 8	1, 64, 128, 256	256, 256, 256, 256	256, 128, 64, 16	24, 256, 512, 256, 1
(R1)	1, 64, 128	128, 64, 8	1, 64, 128	128, 128, 128, 128	128, 64, 16	24, 256, 512, 256, 1
(R2)	1, 64, 128, 256	256, 128, 64, 8	1, 64, 128, 256	256, 256, 256, 256	256, 128, 64, 16	24, 128, 128, 1
(R3)	1, 64, 128	128, 64, 8	1, 64, 128	128, 128, 128, 128	128, 64, 16	24, 128, 128, 1
(R4)	1, 64, 128	128, 64, 8	1, 64, 128	128, 128	128, 64, 16	24, 128, 128, 1

Table 7: The accuracy of the binary classification task including ablation studies (R1), (R2), (R3), and (R4). The classification accuracy of (R1) and (R3) is worse than that of (R0), while that of (R2) and (R4) is a little better than that of (R0). This would show that decreasing the number of parameters of $\phi^{(6)}$ improve the accuracy and that decreasing the number of parameters in the networks $\phi^{(1)}, \phi^{(2)}, \phi^{(3)}, \phi^{(4)}, \phi^{(5)}$ does not.

(R0)	(R1)	(R2)	(R3)	(R4)
81.9 ± 2.1	80.2 ± 2.9	**82.1 ± 3.2**	79.4 ± 0.9	82.0 ± 3.4