이학박사 학위논문

Topological Methods in Data Analysis
(데이터 분석에서의 위상수학적 방법)

2025년 2월

서울대학교 대학원

수리과학부

김세훈

Topological Methods in Data Analysis
(데이터 분석에서의 위상수학적 방법)

지도교수 **Otto van Koert**

이 논문을 이학박사 학위논문으로 제출함

2024년 10월

서울대학교 대학원

수리과학부

김세훈

김세훈의 이학박사 학위논문을 인준함

2024년 12월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Topological Methods in Data Analysis

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Sehun Kim

Dissertation Director : Professor Otto van Koert

Department of Mathematical Sciences
Seoul National University

February 2025

Abstract

Topological Methods in Data Analysis

In recent years, data analysis has seen an increasing integration of advanced mathematical techniques to better understand and interpret complex datasets. This doctoral thesis explores the topological methods in data analysis, focusing on the synergies Topological Data Analysis (TDA), and deep learning. A central theme of this work is the development and application of Persistent Homology, a tool from TDA, to capture the topological features of data across multiple scales. This work examines how deep learning architectures can be enriched by integrating these geometric and topological frameworks, leading to more robust and interpretable models for various data-driven applications. By combining these advanced mathematical tools, the thesis aims to provide new insights and methodologies that bridge the gap between theory and practical data analysis.

Key words: Data analysis, Topological Data Analysis, Persistent homology, Deep learning, Self-supervised learning, neural collapse, electrocardiography
Student Number: 2018-23771

Contents

Chapter 1

Introduction

The explosive growth of data across diverse domains, from healthcare and finance to social networks and scientific research, necessitates the development of robust methods for extracting meaningful insights. Machine learning, particularly deep learning, has demonstrated remarkable success in analyzing and interpreting complex datasets. However, the increasing complexity of data structures, such as high-dimensional spaces, temporal dynamics, and intricate geometric patterns, poses significant challenges to traditional machine learning approaches. To address these challenges, this thesis explores the integration of topological data analysis (TDA), deep learning, and self-supervised learning (SSL) to develop advanced methods for extracting and leveraging intrinsic data properties.

Topological data analysis has emerged as a powerful framework for understanding the shape and structure of data. By capturing global and local topological features, TDA provides insights that are invariant to deformations, noise, and changes in scale. Persistence diagrams, a fundamental representation in TDA, encode these features and have been successfully applied in a wide range of applications, including material science, biology, and image analysis. Despite its utility, integrating TDA with modern machine learning techniques remains a non-trivial task due to the inherently non-Euclidean nature of topological features.

Deep learning, with its ability to learn hierarchical representations from raw data, has transformed many fields. Yet, its potential is often limited by the availability of labeled data, which can be expensive or impractical to obtain in many scenarios. To overcome this limitation, self-supervised learning has gained prominence as a paradigm that learns meaningful representations without the need for extensive labeled datasets. By leveraging

pretext tasks that use intrinsic properties of the data, SSL models have demonstrated state-of-the-art performance in computer vision, natural language processing, and beyond.

This thesis aims to bridge the gap between TDA and modern deep learning frameworks by developing methods that effectively combine topological insights with the representational power of deep neural networks. The first part of this work investigates neural collapse through the lens of persistent homology, translating neural collapse equations into persistent homology-related formulas using the Wasserstein distance. A novel topological loss function is proposed to enhance the effect of neural collapse, offering new perspectives on the geometric and topological structure of neural network representations. In the second part, we develop an extended persistence diagram transformer architecture. Persistence diagrams (PDs), while rich in topological information, are often challenging to integrate into machine learning workflows due to preprocessing complexities and hyperparameter tuning. To address this, we streamline the preprocessing steps and minimize hyperparameter choices, enabling the use of extended persistence diagrams as inputs to the transformer architecture. Finally, this thesis introduces a self-supervised model for 12-lead ECG data, leveraging innovative techniques to address domain-specific challenges and enhance representation learning. These contributions collectively demonstrate the potential of integrating TDA, deep learning, and self-supervised learning for analyzing complex data structures.

The contributions of this thesis are threefold. First, we present a comprehensive study of the integration of TDA with deep learning, addressing challenges such as scalability, representation, and optimization. Second, we introduce novel self-supervised learning approaches tailored to topological and geometric data, focusing on tasks where labeled data is scarce or unavailable. Finally, we validate our methods through extensive experiments on real-world datasets, demonstrating their effectiveness in classification, clustering, and representation learning tasks.

By synthesizing the strengths of topological data analysis, deep learning, and self-supervised learning, this work aims to contribute to the advancement of machine learning methodologies and their application to complex data structures. The findings of this research have implications for a wide range of fields, from biosignal research to theoretical understanding of deep learning, where understanding the shape and structure of data is critical.

Chapter 2

Preliminaries

In this chapter, we explain minimal basic notions and concepts that are essential for understanding the subsequent chapters. We introduce the fundamental concepts of persistent homology, transformer architecture, and self-supervised learning. We also provide an overview of the mathematical tools and techniques used in this thesis. Unless otherwise stated, the poset T is considered to be $[0, \infty)$, and k denotes a vector space.

2.1 Persistent Homology

Definition 2.1.1 (Filtration)**.** A *filtration* is a functor $\mathcal{F} : T \to \mathrm{Top}$, where T is considered as a poset category, and Top is the category of topological spaces. For convenience, denote $\mathcal{F}_{t_1,t_2} = \mathcal{F}(t_1 \leq t_2)$.

Remark 2.1.2. *Functoriality of the filtration implies that if* $t_1, t_2, t_3 \in T$ *and* $t_1 \leq t_2 \leq t_3$*, then* $\mathcal{F}_{t_1,t_3} = \mathcal{F}_{t_2,t_3} \circ \mathcal{F}_{t_1,t_2}$*, and* $\mathcal{F}_{t_1,t_1} = id_{\mathcal{F}_{t_1}}$*.*

Definition 2.1.3 (Persistent Homology)**.** Let $\mathcal{F} : T \to \mathrm{Top}$ be a filtration, and $H_*(-; k) : \mathrm{Top} \to \mathrm{Vect}_k$ be the homology functor with coefficients in a field k. The *persistent homology* of the filtration $\mathcal{F}$ is defined as the composite functor

$$H_*(-; k) \circ \mathcal{F} : T \to \mathrm{Vect}_k.$$

A persistent homology is an example of a persistence module, which is a mathematical structure that captures the evolution of vector spaces over a poset T.

Definition 2.1.4 (Persistence Module)**.** A *persistence module* over k is a functor $\mathbb{V} : T \to \mathrm{Vect}_k$. The morphism $\mathbb{V}_s \to \mathbb{V}_t$ is denoted as $v_{s,t}$ for $s, t \in T$ with $s \leq t$. A morphism between two persistence modules $\mathbb{U}, \mathbb{V}$ is a natural transformation $\phi : \mathbb{U} \to \mathbb{V}$.

Definition 2.1.5 (Homomorphism)**.** A *homomorhism* between two persistence modules $\mathbb{U}, \mathbb{V}$ is a natural transformation $\phi : \mathbb{U} \to \mathbb{V}$. Two persistence modules $\mathbb{U}, \mathbb{V}$ are *isomorphic* if there exists a natural isomorphism $\phi : \mathbb{U} \to \mathbb{V}$. We write $\mathbb{U} \cong \mathbb{V}$ to denote that $\mathbb{U}$ and $\mathbb{V}$ are isomorphic.

Remark 2.1.6. *Naturality of $\phi : \mathbb{U} \to \mathbb{V}$ implies that the if $s, t \in T$ with $s \leq t$, then $\phi_t \circ u_{s,t} = v_{s,t} \circ \phi_s$.*

The simplest persistence module is the interval module.

Definition 2.1.7 (Interval Module)**.** A persistence module $\mathbb{I} : T \to \mathrm{Vect}_k$ is called an *interval module* if there is an interval I such that

$$\mathbb{I}_t = \begin{cases} k & \text{if } t \in I, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{and} \qquad i_{s,t} = \begin{cases} id & \text{if } s, t \in I \text{ with } s \leq t, \\ 0 & \text{otherwise.} \end{cases}$$

In such a case, we write $\mathbb{I} = \mathbb{I}_I$.

Definition 2.1.8 (Direct Sum)**.** The direct sum $\mathbb{W} = \mathbb{U} \oplus \mathbb{V}$ of two persistence modules $\mathbb{U}, \mathbb{V}$ is defined as follows:

$$\mathbb{W}_t = \mathbb{U}_t \oplus \mathbb{V}_t, \quad w_{s,t} = u_{s,t} \oplus v_{s,t} \quad \text{for all } s, t \in T \text{ with } s \leq t.$$

Interval modules are simple in the sense that they are indecomposable. A persistence module $\mathbb{W}$ is indecomposable if the only decompositions of $\mathbb{W}$ are trivial, i.e., $\mathbb{W} = \mathbb{U} \oplus \mathbb{V}$ with $\mathbb{U} = 0$ or $\mathbb{V} = 0$. Interval modules are building blocks for general persistence modules, as shown in the following theorem.

Theorem 2.1.9 (Gabriel, 1972)**.** *Let $\mathbb{V}$ be a persistence module such that dim$(\mathbb{V}_t) < \infty$. Then $\mathbb{V}$ can be decomposed as a direct sum of interval modules in a unique way.*

This theorem implies that any persistence module consists of finite vector spaces can be decomposed into a direct sum of interval modules, which are the simplest building blocks of persistence modules. This decomposition is known as the *interval decomposition* of a persistence module.

If a persistence module $\mathbb{V}$ can be decomposed as

$$\mathbb{V} \cong \bigoplus_i \mathbb{I}_{I_i},$$

then the collection of intervals $\{I_i\}$ is called the *barcode* of the persistence module $\mathbb{V}$. The barcode provides a concise representation of the topological features captured by the persistence module. A *persistence diagram* is a visual representation of the barcode, where each interval is represented as a point in the plane. The persistence diagram is a powerful tool for summarizing the topological features of a dataset across multiple scales.

2.2 Transformer Encoder Architecture

The transformer encoder, introduced by Vaswani et al. in their paper *"Attention is All You Need"* (2017), is a foundational building block in deep learning. It is designed to process sequential input data in parallel, making it highly efficient and capable of modeling long-range dependencies. This section details the encoder's structure, focusing on how it transforms input data into meaningful representations through self-attention and feedforward mechanisms.

2.2.1 Input Representation

The input to the encoder is a sequence of tokens, represented as $X = \{x_1, x_2, \ldots, x_N\}$, where N is the sequence length, and x_i is the i-th token's embedding. Each token embedding is a vector of size d_{model}, which represents the dimensionality of the model. We consider X as a matrix $X \in \mathbb{R}^{N \times d_{\text{model}}}$.

Since the self-attention mechanism processes tokens without regard to their positions in the sequence, *positional encodings* are added to the embeddings to encode order information. The resulting input to the encoder is:

$$X' = X + PE$$

where PE represents the positional encoding matrix. Each token x'_i in X' is then passed through the encoder layers.

2.2.2 Core Components of the Transformer Encoder

The encoder consists of a stack of identical layers, each comprising a multi-head self-attention mechanism, a feedforward network, residual connections, and layer normalization.

Construction of Q, K, and V from X

The self-attention mechanism operates on three matrices derived from the input X': Queries (Q), Keys (K), and Values (V). These matrices are constructed by linearly projecting the input embeddings using learned weight matrices:

$$Q = X'W^Q, \quad K = X'W^K, \quad V = X'W^V$$

where $W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, and $W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are the learnable projection matrices. Here, d_k and d_v denote the dimensions of the key and value vectors, respectively. These projections allow the model to focus on specific aspects of the input data during attention computation.

Self-Attention Mechanism

Once Q, K, and V are constructed, the self-attention mechanism computes the attention scores and outputs. The attention mechanism captures relationships between all tokens in the sequence. The attention scores are calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

The dot product $QK^\top$ measures the similarity between query and key vectors, and the scaling factor $\sqrt{d_k}$ prevents the scores from becoming too large. The `softmax` function normalizes the scores into probabilities, which are then used to weight the value vectors V.

Multi-Head Self-Attention

To improve the model's ability to capture diverse patterns, the encoder uses *multi-head self-attention*. Instead of using a single attention head, the input is split into multiple

subspaces (or *heads*). Each head computes attention independently:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where W_i^Q, W_i^K, and W_i^V are separate learnable projection matrices for the i-th head. The outputs of all heads are concatenated and linearly transformed:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

where W^O is a learned output projection matrix.

Feedforward Neural Network

Each encoder layer includes a position-wise feedforward network (FFN) that applies two fully connected layers with a non-linear activation function (e.g., ReLU):

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

This network processes each token independently and helps transform the embeddings into higher-level representations.

Residual Connections and Layer Normalization

Each sub-layer in the encoder employs *residual connections* and *layer normalization* to improve training stability and gradient flow. The output of each sub-layer is computed as:

$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$

Residual connections mitigate the vanishing gradient problem, while layer normalization stabilizes activations.

2.2.3 Encoder Workflow

The workflow of the transformer encoder can be summarized as follows:

1. The input sequence X is embedded into a continuous space and augmented with positional encodings to produce X'.

2. The embeddings are passed through a stack of encoder layers. Each layer:
   (a) Computes multi-head self-attention to capture token-to-token relationships.
   (b) Processes the attention outputs through a feedforward network.
   (c) Applies residual connections and layer normalization after each sub-layer.
3. The final encoder outputs are high-dimensional representations that encode both local and global dependencies in the input sequence.

2.2.4 Applications of the Transformer Encoder

The transformer encoder has become a cornerstone in modern deep learning, powering state-of-the-art models across various domains. In NLP, it forms the basis of pre-trained models such as BERT and RoBERTa. In other fields like computer vision and time-series analysis, encoder-based architectures like Vision Transformers (ViT) and time-series transformers leverage its ability to process sequences efficiently and extract meaningful features.

Chapter 3

Neural Collapse Through the Lens of Persistent Homology

The *Neural Collapse* (NC) phenomenon in deep neural network classifiers, first discovered by [40], indicates that feature vectors in the last hidden layer converge to a geometrically intriguing shape: a regular simplex, as the model enters the terminal phase of training. This convergence raises pivotal questions in deep learning: Does *NC* in the training set induce a similar collapse in the test set? Is it related to a model's generalization ability? And can we harness *NC* to enhance classifiers? And can we utilize *NC* to optimize classifiers? Given the inherent topological nature of a simplex, persistent homology—a cornerstone of topological data analysis—emerges as a natural tool to dissect *NC*'s geometric and topological properties. In this work, we present a detailed analysis of *NC* through the lens of persistent homology. Additionally, we leverage this methodology to affirmatively address the aforementioned questions and introduce a novel topological loss term. Topological loss function encourages a more pronounced *NC* in both training and test sets, resulting in superior model generalization.

3.1 Introduction

Deep learning, over the past decade, has emerged as a groundbreaking approach in a myriad of applications, from image recognition to natural language processing. While the successes of deep learning models are evident in various tasks, understanding the intricate dynamics of their training remains an active area of research. One such intriguing

phenomenon observed during the training of deep classifiers is the Neural Collapse (NC) [40].

The essence of NC lies in the behavioral pattern exhibited by the features in the final layer of deep learning classifiers. As we enter the terminal phase of training, these features demonstrate a tendency to converge towards their respective within-class means. Intriguingly, when the dataset is balanced, these mean vectors position themselves in a way that they are maximally separated, by being the vertices of a regular simplex. This geometric arrangement is not only captivating but also provides intuition to better understand deep learning.

In the realm of topology, a branch of mathematics that studies properties preserved under continuous deformations, the simplex is a fundamental object. This offers the opportunity to utilize topological data analysis tools, specifically persistent homology, to gain deeper insights into the NC phenomenon. Persistent homology, a cornerstone of topological data analysis, provides a robust framework to study the topological features of data across various scales.

The relationship between NC and generalization has been a focal point of multiple research endeavors, but these efforts have yielded divergent conclusions, likely attributed to varying experimental setups. [40] suggested that NC works in favor with generalization, but their assertions have been met with skepticism, particularly by studies like [24]. Another interesting quesiton is related to the imbalanced dataset. In an imbalanced setting, [18] has pointed out that the regular simplex structure does not appear. A natural follow-up question is is this irregularity favorable to generalization or is it an impediment to better generalization.

Our contributions To shed light on this ambiguity, we conduct experiments in a simple setting. Our research scrutinizes the NC phenomenon using the tools of topological data analysis. Specifically:

- We present a novel perspective by characterizing NC using persistent homology, marking a pioneering effort to meld these two domains.
- We introduce a topological loss function tailored to accentuate the NC effect in both training and test datasets.
- Through empirical studies, we demonstrate that a model fortified with NC indeed

showcases superior generalization capabilities compared to its conventional counterparts.

- Our empirical results substantiate that encouraging regularity in settings with imbalanced datasets enhances the generalization capability of the model, implying that any deviation from this regularity may hinder optimal model generalization.

3.1.1 Notation

Let $\mathcal{X}$ denote the input space and $\mathcal{Y} = \{1, 2, \ldots, K\} = [K]$ represent the labels. The dataset is defined as $X = \bigsqcup X_i \subset \mathcal{X}$, where X_i includes samples of label $i \in [K]$. The learned deep neural network is denoted by $f : \mathcal{X} \to \mathcal{Y}$ and the network mapping each sample to its final-layer feature is represented by $h : \mathcal{X} \to \mathbb{R}^m$. Therefore, $f(x) = W \cdot h(x) + b$, with W being a $K \times m$ matrix and $b \in \mathbb{R}^K$ as the bias vector. $H = h(X)$ is the collection of all final-layer features, and $H_i = h(X_i)$ is the set of features labeled i. The mean of the final-layer features labeled i is $\mu_i = \frac{1}{|H_i|} \sum_{h \in H_i} h$, and $\mu = \frac{1}{K} \sum_i \mu_i$ is the barycenter of each within-class mean. The normalized version of μ_i is defined as $\tilde{\mu_i} = (\mu_i - \mu)/\|\mu_i - \mu\|$. $\left(X^{(\text{batch})}, Y^{(\text{batch})}\right)$ is a batched data of (X, Y), and the corresponding notations are $H^{(\text{batch})}, \mu_i^{(\text{batch})}$, and $\mu^{(\text{batch})}$. We will sometimes call H, or $H^{(\text{batch})}$ to be clustered dataset.

3.2 Neural Collapse

Deep neural networks have underscored numerous breakthroughs in machine learning, especially in tasks with large and intricate datasets. These networks inherently formulate internal data representations within their hidden layers as they learn. These embeddings, capturing the latent patterns and structures, empower the network for precise predictions or classifications. Interestingly, on probing these internal representations, [40] have discovered certain captivating behaviors, notably the Neural Collapse phenomenon.

Neural Collapse, at its core, is marked by a distinctive convergence pattern during deep neural network training. As the network iteratively refines its weights, the embeddings of distinct data points—especially those of the same class—undergo a significant transformation. Instead of preserving individualized and dispersed representations, these embeddings tend to cluster, leading to a "collapse". Explicitly, within-class embeddings

shift towards their class means. Simultaneously, these class means in the last-layer feature space aim for maximal separation, resulting in a Simplex Equiangular Tight Frame (**ETF**)—indicating the class means as vertices of a regular simplex in the embedding realm.

Delving into this behavior's technical nuances:

NC1 **(Variability collapse):** For any $x \in X_i$, it's last-layer feature $h(x)$ converges to μ_i for all $i \in [K]$.

NC2 **(Simplex ETF structure):**

$$\left| \|\mu_i - \mu\| - \|\mu_j - \mu\| \right| \to 0, \forall i, j \in [k]$$
$$\langle \tilde{\mu_i}, \tilde{\mu_j} \rangle \to \frac{K}{K-1}\delta_{i,j} - \frac{1}{K-1}, \forall i, j \in [K]$$

By "convergence" in this context, we mean the behavior "as the number of training steps approaches infinity". At a cursory look, *NC2* might appear intricate. However, it essentially conveys that the mean vectors are equidistant from their barycenter and the angles between the normalized mean vectors remain a consistent $-1/(K-1)$. This arrangement positions the mean vectors precisely as the vertices of a regular simplex with K vertices.

This behavior of the last-layer features are very interesting, but at the same time, it raises several questions. Does the network, through this "collapse", find a streamlined understanding of data? Does this collapse also oberved in test set as well?Or might it be a limitation, an over-simplification potentially affecting the network's generalization to unseen data? In this paper, we endeavor to provide insights, uniquely employing tools from persistent homology.

3.2.1 Measurements of *NC*

Metrics to measure the neural collapse degree are defined as follows:

For *NC1*:

$$\mathcal{NC}_1 = \frac{\mathbb{E}_{(x,y)\sim\mathcal{D}}[\|h(x) - \mu_y\|^2]}{\mathbb{E}_i[\|\mu_i - \mu\|^2]}$$

This metric, utilized in [24, 51], has the numerator representing the average squared distance from the last-layer feature to its respective center, and the denominator serves as a

normalizer, denoting the average squared distance from the mean vectors to the barycenter. A smaller value suggests features clustering near their class-mean vectors.

For *NC2*:

$$\mathcal{NC}_2 = \left\| \frac{MM^t}{\|MM^t\|_F} - \frac{1}{\sqrt{K-1}}(I_K - \frac{1}{K}1_K 1_K^t) \right\|_F$$

This metric is used in [28, 51]. Here, M is a $K \times m$ matrix whose i-th row is the normalized mean vector $\tilde{\mu}_i$.

We will measure $\mathcal{NC}_1$ and $\mathcal{NC}_2$ in a batched manner, but $\mathcal{NC}_2$ can be employed only if the batch contains samples of all labels. This is because the construction of M requires the mean vectors of all labels. Instead, we will define another metric $\mathcal{NC}_2^{PH}$ that makes uses of persistent homology. Persistent homology provides a natural way to compare two topological structures, and with this tool, we can measure how the shape of mean vectors deviate from a regular simplex. Through out this paper, we will use '*NC*' as a phenominon, whereas $\mathcal{NC}$ as a metric.

3.3 Background on Persistent Homology

Persistent homology is a prominent technique in topological data analysis (TDA), enabling the study of topological features of shapes or data. By capturing the evolution of topological features across varying scales, it provides a multi-scale topological perspective. This section briefly revisits persistent homology's foundational concepts, directing readers to [17] for an exhaustive treatment. Notably, all topological objects discussed herein are graphs, representing 1-dimensional simplicial complexes.

3.3.1 Simplicial Complex

A simplicial complex is a combinatorial object that serves as a versatile tool for the approximation and representation of a diverse range of topological spaces.

Extended from the concept of triangles to arbitrary dimensions, simplices can be understood as:

- 0-simplex: A point or vertex.
- 1-simplex: A line segment, spanned by its endpoints or 0-simplices.
- 2-simplex: A triangle, surrounded by three 1-simplices (edges) and three vertices.

- 3-simplex: A tetrahedron, comprising four 2-simplices (faces), six edges, and four vertices.
- Higher dimensions follow analogous patterns.

Within this framework, a face of a simplex denotes a subsimplex. For instance, a triangle's (2-simplex) faces include its edges (1-simplices) and vertices (0-simplices). A simplicial complex, $\mathcal{K}$, within space X, is a collection of simplices such that:

1. Every face of a simplex in $\mathcal{K}$ is also in $\mathcal{K}$.
2. The intersection of any two simplices in $\mathcal{K}$ is a face common to both.

For example, a graph $G = (V, E)$ is a 1-dimensional simplicial complex, with V denoting 0-simplices and E signifying 1-simplices.

When a deep learning model exhibits the NC property over a given dataset, the features in its last layer can be seen to mirror the form of a 1-dimensional simplicial complex. Specifically, within this topological representation, each within-class mean assumes the role of a 0-simplex, effectively acting as a vertex. Also, the relations between these mean vectors can be likened to 1-simplices, essentially serving as the line segments that bind the vertices together. See Figure 3.3 for visual representation.

3.3.2 Simplicial Homology

Central to simplicial homology are chain groups. For a simplicial complex $\mathcal{K}$, the n-th chain group, $C_n(\mathcal{K}; \mathbb{Z}_2)$, embodies all $\mathbb{Z}_2$-coefficients linear combinations of n-simplices. Boundary maps establish connections between successive chain groups. For an n-simplex $\sigma = [v_0, \ldots, v_n]$, the boundary map $\partial_n : C_n(\mathcal{K}; \mathbb{Z}_2) \to C_{n-1}(\mathcal{K}; \mathbb{Z}_2)$ is:

$$\partial_n(\sigma) = \sum_{i=0}^{n} [v_0, \ldots, \hat{v_i}, \ldots, v_n].$$

With $\mathbb{Z}_2$ coefficients, the addition is modulo 2.

The crux of simplicial homology is to discern cycles that are not bound by boundaries. Formally, the n-th simplicial homology group is:

$$H_n(\mathcal{K}; \mathbb{Z}_2) = \frac{\ker(\partial_n)}{im(\partial_{n+1})},$$

where $\ker(\partial_n)$ and $im(\partial_{n+1})$ are the kernel and image of the boundary maps, respectively. With $\mathbb{Z}_2$ coefficients, each $H_n(\mathcal{K};\mathbb{Z}_2)$ is a vector space. In this paper, we will use $\mathbb{Z}_2$ coefficients, and write $H_n(\mathcal{K}) = H_n(\mathcal{K};\mathbb{Z}_2)$. Denote $H_.(\mathcal{K}) = \bigoplus_{n=0}^{\infty} H_n(\mathcal{K})$ to be the direct sum of homology groups of all dimension. Since we deal with only finite simplicial complexes in this paper, the direct sum is actually a finite sum. An element of $H_.(\mathcal{K})$ is called a homology class, and informally, it is a topological feature.

3.3.3 Persistent Homology

The essence of persistent homology is capturing the topological features of a space across scales via a filtration. A *filtration* for a simplicial complex is a nested sequence of subcomplexes:

$$\mathcal{K}_0 \subset \mathcal{K}_1 \subset \cdots \subset \mathcal{K}_{m-1} \subset \mathcal{K}_m = \mathcal{K}.$$

As filtration progresses, topological features emerge, persist, and ultimately dissipate. Each feature is ascribed an interval $[b, d]$, marking its birth at stage b and death at d. In order to precisely measure the borns and deaths of topological features, we take homology at each level of filration to get

$$H_n(\mathcal{K}_0) \to H_n(\mathcal{K}_1) \to \cdots \to H_n(\mathcal{K}_{m-1}) \to H_n(\mathcal{K}_m),$$

which is called a *n-th persistent homology* of the filtration $\{\mathcal{K}_i\}_{i=0}^m$. Each map $H_n(\mathcal{K}_i) \to H_n(\mathcal{K}_{i+1})$ is induced from the inclusion map $\mathcal{K}_i \hookrightarrow \mathcal{K}_{i+1}$. Monitoring the evolution of homological features via these maps allows associating an interval of existence, typically $[b, d]$, where b and d are birth and death times, respectively. Beyond individual dimensions, persistent homology can also be comprehensively expressed as

$$H_.(\mathcal{K}_0) \to H_.(\mathcal{K}_1) \to \cdots \to H_.(\mathcal{K}_{m-1}) \to H_.(\mathcal{K}_m).$$

3.3.4 Persistence Diagram

A persistence diagram is a visual representation of the birth and death times of topological features obtained from persistent homology. Each feature is represented by a point in the plane, where the x-coordinate denotes the feature's birth time and the y-coordinate its death time.

Formally, let's consider a filtration of a simplicial complex $\mathcal{K}$ through which we've

applied persistent homology. For each topological feature that appears (i.e., is "born") at a certain stage of the filtration and subsequently disappears (i.e., "dies") at a later stage, we place a point (b, d) in the plane, where b is the birth time and d is the death time. Features that persist indefinitely (i.e., never die) can be represented by points lying on the line $y = \infty$. We will denote $PD_n(\mathcal{K})$ be the persistence diagram obtained from the persistent homology $\{H_n(\mathcal{K}_i)\}_{i\geq 0}$, and $PD_.(\mathcal{K}) := \bigsqcup_{n=0}^{\infty} PD_n(\mathcal{K})$ be the disjoint union of persistence diagrams of all dimension.

Noteworthy characteristics of the persistence diagram include:

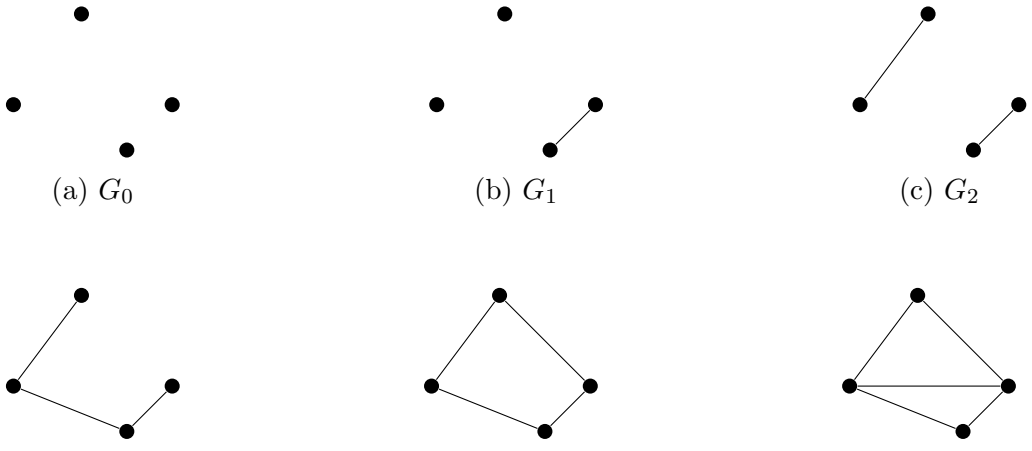
1. **Diagonal and Above:** All points in the persistence diagram lie on or above the diagonal line $y = x$. This is because a feature cannot die before it is born, ensuring that $d \geq b$ for all points (b, d).

2. **Stability:** One of the key properties of persistence diagrams is their stability. Small perturbations or changes in the input data will result in only minor changes in the persistence diagram. This stability ensures that the persistence diagram is a robust topological descriptor, resilient to noise in the data.

3. **Comparability:** Persistence diagrams provide a compact summary of the topological features of a dataset. They can be compared using various metrics, like the bottleneck or Wasserstein distances, to quantify the topological difference between two datasets.

Leveraging the comparability characteristic, we assess the topological divergence of the last-layer features from an ideal regular simplex. This assessment serves as a gauge for the intensity of the *NC* phenomenon.

Example Consider a weighted graph $G = (V, E)$, where each weight is assigned to an edge, which can be thought as a distance. Note that a graph is an example of a 1-dimensional simplicial complex. One can construct a filtration of the graph, $(G_\delta)_{\delta\geq 0}$, where $V(G_\delta) = V$ and $E(G_\delta) = \{e \in E : \text{weight}(e) \leq \delta\}$. Initially, this filtration presents the graph as isolated vertices. As the parameter δ increases, edges with weights less than or equal to δ are incorporated, progressively revealing the structure of G. See Figure 3.1



(a) G_0 (b) G_1 (c) G_2

(d) G_3 (e) G_4 (f) G_5

Figure 3.1: Graph filtration stages: (a) Discrete nodes. (b)-(d) Edges added, reducing connected components. (e)-(f) Edges create cycles.

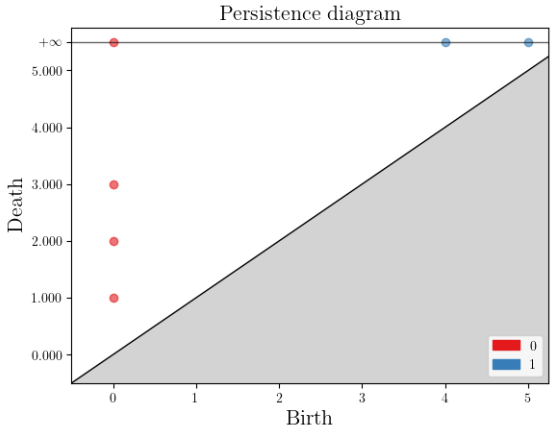


Figure 3.2: **Persistence diagram:** The diagram displays four blue dots (representing connected components) and two red dots (indicating 1-cycles). A red point at $(0, 1)$ signifies that a connected component merges with another one at $\delta = 1$. A blue point at $(4, 5)$ indicates the birth of a 1-cycle at $\delta = 4$. Every persistence diagram derived from a nonempty set includes a 0-dimensional feature at $(0, \infty)$, denoting the presence of at least one connected component.

The emergence of a new edge during the filtration can lead to one of two topological changes:

1. Merging two distinct connected components, effectively reducing the number of connected components. See Figure 3.1b, 3.1c, 3.1d. This phenomenon corresponds to a point $(0, \delta)$ in the 0-dimensional persistence diagram $PD_0(G)$. This point signifies a connected component's existence from the start of the filtration up to the moment it merges at δ.

2. Creating a cycle within the graph. See Figure 3.1e, 3.1f. This event is represented by a point (δ, ∞) in the 1-dimensional persistence diagram, $PD_1(G)$. It denotes a cycle that emerges at the δ threshold and persists indefinitely, as cycles, once formed, are enduring in this context.

Now, for conveience, suppose all edge weights in G are distinct. This distinction allows for the identification of a unique *minimal spanning tree* (MST) — a tree that connects all vertices in G while ensuring the total edge weight is minimized. See Figure 3.1d. Given that an MST contains exactly $(|V| - 1)$ edges, the 0-dimensional persistence diagram will have $(|V| - 1)$ points of the form $(0, \delta)$ and one point at $(0, \infty)$, where each δ corresponds to the weights of edges within the MST. Concurrently, the 1-dimensional persistence diagram, $PD_1(G)$, will feature points (δ', ∞) representing edge weights δ' not included in the MST, signifying cycles they induce. Persistence diagram from the graph filtration can be seen in Figure 3.2. Four red dots correspond to 0-dimensional features(connected components), and red dots correspond to 1-dimensional features(cycles).

3.3.5 Wasserstein Distance

The Wasserstein distance offers a robust metric for contrasting persistence diagrams, accounting for both point location and multiplicity. It provides a comprehensive comparison evaluating all the points in the diagrams.

For two persistence diagrams D and D' of the same dimension, the (p, q)-Wasserstein distance is:

$$W_{p,q}(D, D') = \left(\inf_{\text{matching } M} \sum_{(x,y) \in M} \|x - y\|_q^p \right)^{\frac{1}{p}},$$

Here, "matching" refers to a pairing of points between D and D'. A matching M assigns each point in D to a point in D' or to a point on the diagonal, with the objective

of minimizing the cumulative distance between matched points. The cost of matching a point to the diagonal is given by its ℓ_q distance from the diagonal, reflecting the lifespan of the feature. In this matching, points of the form (a, ∞) must be matched with (b, ∞), and their distance $\|(a, \infty) - (b, \infty)\|_q$ is defined to be $|a - b|$. The Wasserstein distance can be thought of as the minimal cost of matching two persistence diagrams.

If $D = \bigsqcup D_n$ and $D' = \bigsqcup D'_n$ are the disjoint union of persistence diagrams of all dimensions, then the Wasserstein distance is defined as the sum of each individual Wasserstein distance:

$$W_{p,q}(D, D') := \left(\sum_{n=0}^{\infty} W_{p,q}^p(D_n, D'_n) \right)^{\frac{1}{p}}.$$

The Wasserstein distance captures not only the differences in lifespans of matched features but also accounts for features present in one diagram but absent in the other. In this paper, we will specifically use $p = 2$ and $q = 1$. Henceforth, we'll represent $W_{(2,1)}$ simply as W.

3.4 Neural Collapse through Persistent Homology

The concept of a *NC1* suggests that the last-layer feature vectors aggregate around their mean vectors. In contrast, *NC2* states that these mean vectors should be maximally separated, inducing the shape of the vertices of a regular simplex. Essentially, as training approaches its final phase, the geometric and topological structure of the feature vectors increasingly aligns with that of a regular simplex. Persistent homology offers tools to contrast the geometry and topology of feature vectors with that of a regular simplex. In this section, after introducing some foundational concepts, the Wasserstein distance will be employed as the primary tool to investigate *NC*.

3.4.1 Simplicial Complex of *NC* features

When the model enters the terminal phase of training, the last-layer features $H = \bigsqcup H_i$ exhibits the *NC*. Within-class features H_i collapse to its center, whereas between-class features form the vertices of a regular simplex. This is a clustered dataset where each cluster corresponds to vertices of a regular simplex. We want to construct a simplicial complex that reflects the *NC* properties. That is, each cluster should be contractible

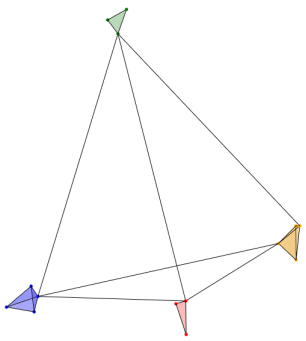


Figure 3.3: **Visualization of *NC*:** Within each class, features form a tightly clustered simplex. Each of these clusters, or small simplices, is interconnected by the shortest edges between them. Topologically, this structure corresponds to the 1-skeleton of a simplex. Each small colored simplex denotes features with distinct labels.

to its center, and the collection of edges between clusters should be similar to that of a regular simplex.

Definition 3.4.1 (**Simplicial Complex from Clustered Data**)**.** For a clustered dataset $H = \bigsqcup H_i$, we construct a simplicial complex $\mathcal{K}$ based on H to depict both intra- and inter-cluster relationships, defined as:

- The vertices of $\mathcal{K}$ is H.
- Each cluster H_i forms a simplex
- For each pair (H_i, H_j), there exists a unique edge between them; the shortest one

Figure 3.3 is an example simplicial complex of this type, where it consists of 4 clusters, and there is only one edge between each clusters. Since each cluster forms a simplex, it is contractible and it's topological structure is same as the 1-skeleton of a regular simplex with 4 vertices. Our goal is to force this simplicial complex to be similar to a 1-skeleton of a regular complex, with the help of persistent homology. By a 1-skeleton $\mathcal{L}^{(1)}$ of a simplicial complex $\mathcal{L}$, we mean the subcomplex of the $\mathcal{L}$ consisting of only vertices and edges. For example, the 1-skeleton of a simplex is a clique graph.

3.4.2 Filtration

We define a *pseudo-Rips* filtration $(\mathcal{K}_\delta)_{\delta\geq 0}$ on the induced simplicial complex $\mathcal{K}$ as follows. For $(m+1)$ vertices $\{h_0, \cdots, h_m\}$, they form a simplex in $\mathcal{K}_\delta$ if their pairwise distances are all less than or equal to δ. Note that if $[h_0, \cdots, h_m]$ is a simplex in $\mathcal{K}_\delta$ for $m \geq 2$, then $\{h_0, \cdots, h_m\}$ should belong to the same class. One important remark is that since there are only finitely many vertices in $\mathcal{K}$, the filtration $(\mathcal{K}_\delta)_{\delta\geq 0}$ is actually isomorphic to a filtration of discrete type $(\mathcal{K}_i)_{i=0}^n$ for some n even though it seems there are infinitely many distinct $\mathcal{K}_\delta$'s at first glance.

The pseudo-Rips filtration gives rise to a persistent homology $\{H_\cdot(\mathcal{K}_\delta)\}_{\delta\geq 0}$. Its persistence diagram, represented as $PD_\cdot(\mathcal{K}_\delta)$, simplifies the descriptions of *NC1* and *NC2*. Informally speaking, *NC1* and *NC2* correspond to the convergence of persistence diagrams from $PD_\cdot(\mathcal{K})$ to $PD_\cdot(\sigma^{(1)})$ in some sense, with σ being the regular simplex consisting of mean vectors and $\sigma^{(1)}$ is the 1-skeleton of σ. The reason we consider only up to 1-skeleton $\sigma^{(1)}$, not the full simplex σ is that we are not interested in higher-dimensioal topological features.

***NC* in terms of persistent homology** Let $H = \bigsqcup_{i=1}^K H_i$ be the clustered dataset such that

$$H_i \subset B_r(\mu_i), \|\mu_i - \mu_j\| \in (R-\epsilon, R+\epsilon), \text{ and } R > 4r,$$

where μ_i is the center of H_i, $B_r(\mu_i)$ is the ball at μ_i with radius $r > 0$. The fisrt condition corresponds to the Variability collapse(*NC1*), whereas the second condition means that each pairwise distance between centers are close to R(*NC2*), and the final condition means that each clusters are not too close. *NC* implies that $r, \epsilon \to 0$ as train step $t \to \infty$. Let G be the 1-skeleton of a regular simplex with $\mathcal{K}$ vertices with edge-length R. There is a natural graph filtration $(G_\delta)_{\delta\geq 0}$ where $G_\delta = V(G)$ if $\delta < R$ and $G_\delta = G$ if $\delta \geq R$.

In this situation, let's calculate the Wasserstein distance:

$$W^2(PD_\cdot(\mathcal{K}), PD_\cdot(G)) = W^2(PD_0(\mathcal{K}), PD_0(G)) + W^2(PD_1(\mathcal{K}), PD_1(G))$$

Theorem 3.4.2. *In the above setting,*

$$W^2(PD_\cdot(\mathcal{K}), PD_\cdot(G)) \to 0 \text{ if and only if } r, \epsilon \to 0$$

Proof. See the appendix 3.A.2 □

3.4.3 Proposed Loss Function

To simultaneously foster within-class feature collapse (*NC1*) and class separation (*NC2*) at the same time, the Wasserstein distance $W_{p,q}\left(PD_\cdot(\mathcal{K}), PD_\cdot(\sigma)\right)$ can be employed as a loss function. However, merely aiming for $PD_\cdot(\mathcal{K})$ to approach $PD_\cdot(\sigma)$ is impractical for various reasons. For instance, $\mathcal{K}$'s persistence diagram is highly susceptible to outliers. A single off-trend data point in X can significantly alter the persistence diagram $PD_0(\mathcal{K})$, and that is also reflected in the Wasserstein distance. Additionally, computing the persistence diagram $PD_\cdot(\mathcal{K})$ is computationally expensive for large $|\mathcal{K}|$. To overcome these challenges, we introduce a batched version of $PD_\cdot(\mathcal{K}) \to PD_\cdot(\sigma)$, aligning well with deep learning models usually trained on batched data. In order to do this, let's first define an ideal simplex for a given batch.

Definition 3.4.3 (Ideal Simplex for Batched Data)**.** Let $(H^{(\text{batch})}, Y^{(\text{batch})})$ be the batched data with $l \leq K$ distinct labels. Let $\mu_i^{(\text{batch})}$ be the mean vector of features whose label is i, i.e. $\mu_i^{(\text{batch})} = \text{mean}(H_i^{(\text{batch})})$ with $H_i^{(\text{batch})} = H^{(\text{batch})} \cap H_i$ for each i. Define:

1. $G^{(\text{batch})}$ as the 1-skeleton of a simplex with vertices $\{\mu_{i_1}^{(\text{batch})}, \ldots, \mu_{i_l}^{(\text{batch})}\}$.
2. $\tilde{G}^{(\text{batch})}$ as the 1-skeleton of a regular simplex with l vertices. The edge-length $\tilde{d}$ in $\tilde{G}^{(\text{batch})}$ is given by the average edge-length of $G^{(\text{batch})}$

$$\tilde{d} = \mathbb{E}_{(j,k)}\left[\left\|\mu_{i_j}^{(\text{batch})} - \mu_{i_k}^{(\text{batch})}\right\|\right].$$

Two aspects warrant further clarification:

1. Our primary focus is on the intrinsic shape, which makes the exact vertex positions in $\tilde{G}^{(\text{batch})}$ non-essential.
2. Although $G^{(\text{batch})}$ bears a resemblance to a regular simplex $\tilde{G}^{(\text{batch})}$ in a *NC* scenario, it doesn't perfectly mirror a regular simplex in real-world applications.

To foster enhanced *NC1* and *NC2* properties, it is imperative to:

1. Ensure that within-class features converge to their respective mean vectors.
2. Endeavor to make $G^{(\text{batch})}$ approximate $\tilde{G}^{(\text{batch})}$ as closely as possible.

Both objectives can be simultaneously achieved by minimizing the squared-Wasserstein distance between $\mathcal{K}^{(\text{batch})}$ and $\tilde{G}^{(\text{batch})}$, where $\mathcal{K}^{(\text{batch})}$ is the induced simplicial complex from clustered data $H^{(\text{batch})}$.

Definition 3.4.4 (Topological Loss)**.** The topological loss term is defined to be the squared $(2,1)$-Wasserstein distance

$$\begin{aligned}\mathcal{L}_T(X^{(batch)}, Y^{(batch)}) &:= W^2(PD_\cdot(\mathcal{K}^{(\text{batch})}), PD_\cdot(\tilde{G}^{(\text{batch})})) \\ &= W^2(PD_0(\mathcal{K}^{(\text{batch})}), PD_0(\tilde{G}^{(\text{batch})})) + W^2(PD_1(\mathcal{K}^{(\text{batch})}), PD_1(\tilde{G}^{(\text{batch})}))\end{aligned}$$

We will apply this loss function on a well-trained deep network classifier.

3.4.4 Analysis of Topological Loss

To commence our analysis, we introduce a set of notations. Let $\mathcal{K}_i^{(batch)}$ denote the simplex derived from $H_i^{(batch)}$, where $H_i^{(batch)}$ necessitates a clear definition. Similarly, $T_i^{(batch)}$ represents the minimal spanning tree of $\mathcal{K}_i^{(batch)}$. In a manner akin to the minimal spanning tree scenario, we select $l-1$ edges, denoted by $E = \{e_1, \cdots, e_{l-1}\}$, between each class $\mathcal{K}_i^{(batch)}$ ensuring that the union $\left(\bigsqcup \mathcal{K}_i^{(batch)}\right) \bigsqcup E$ is connected and the summation of edge lengths is minimized.

Let E' denote the set of $\binom{l-1}{2}$ edges between classes excluding $\{e_1, \cdots, e_{l-1}\}$. For a visual representation, see Figure 3.4. Thick edges between clusters correspond to E, whereas dashed edges are represented by E'.

0-dimensional distance Expanding on the previous observations, the computation of $PD_0(\mathcal{K}^{(batch)})$ becomes evident. Excluding the trivial point $(0,\infty) \in PD_0(\mathcal{K}^{(batch)})$, it comprises:

1. Intra-class: $\{(0, d_{intra}) : d_{intra}$ is the lengths of the edges in each $T_i^{(batch)}\}$
2. Inter-class : $\{(0, d_{inter}) : d_{inter}$ is the lengths of the edges of $E\}$

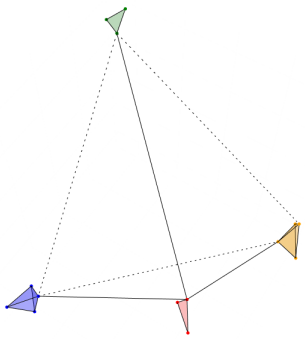


Figure 3.4: Edges between classes correspond to $\{e_1, \cdots, e_{l-1}\}$. Dashed edges represent the set E'.

Given that $PD_0(\tilde{G}^{(batch)})$ consists of $(0, \tilde{d})$ with a multiplicity of $l-1$, the matching between $PD_0(\mathcal{K}^{(batch)})$ and $PD_0(\tilde{G}^{(batch)})$ is evident. Intra-class features $(0, d_{intra})$ should align with the diagonal, while inter-class features $(0, d_{inter})$ should match with $(0, \tilde{d})$.

The cost of this matching is:

$$\left\|(0, d_{intra}) - \left(\frac{d_{intra}}{2}, \frac{d_{intra}}{2}\right)\right\|_1 = d_{intra},$$
$$\left\|(0, d_{inter}) - (0, \tilde{d})\right\|_1 = |d_{inter} - \tilde{d}|.$$

From the above, the Wasserstein distance is:

$$W^2(PD_0(\mathcal{K}^{(batch)}), PD_0(\tilde{G}^{(batch)})) = \sum d_{intra}^2 + \sum (d_{inter} - \tilde{d})^2.$$

1-dimensional distance Likewise, the one-dimensional persistence diagram $PD_1(\mathcal{K}^{(batch)})$ can be categorized into:

1. Intra-class : $\{(b, d) : \text{1-cycle in } \mathcal{K}_i^{(batch)} \text{ is born at } b \text{ and dies at } d\}$
2. Inter-class : $\{(d'_{inter}, \infty) : d'_{inter} \text{ is the length of the edges in } E'\}$

Once again, intra-class features should align with diagonal points, while inter-class

should match with $(0, \tilde{d})$. The cost of this matching is:

$$\left\|(b,d) - \left(\frac{d-b}{2}, \frac{d-b}{2}\right)\right\|_1 = d - b,$$
$$\left\|(d'_{inter}, \infty) - (\tilde{d}, \infty)\right\|_1 = |d'_{inter} - \tilde{d}|.$$

From this, the Wasserstein distance can be formulated as:

$$W^2(PD_1(\mathcal{K}^{(batch)}), PD_1(\tilde{G}^{(batch)})) = \sum (d-b)^2 + \sum (d'_{inter} - \tilde{d})^2,$$

where the first summation spans the set of all within-class 1-cycles.

Approximation From the prior analysis, we deduce:

$$W^2(PD_{\cdot}(\mathcal{K}^{(batch)}), PD_{\cdot}(\tilde{G}^{(batch)})) = \sum d^2_{intra} + \sum (d-b)^2 + \sum (d(H_i, H_j) - \tilde{d})^2$$

Here, $d(H_i, H_j)$ measures the distance between two clusters and can be either d_{inter} or d'_{inter}. The first and third summations span all clusters, while the second summation spans the set of all cluster pairs. The main challenge in computing the Wasserstein distance is the second term, which calculates the 1-dimensional persistence within each class. Given the distance matrix of $H^{(batch)}$, the first and third terms are not computationally intensive. The within-class deaths can be determined using Kruskal's algorithm, which has a worst-case complexity of $\mathcal{O}(n^2 \cdot \alpha(n^2))$, where n represents the number of within-class points, and α is the inverse Ackermann function, which grows slowly. The third term can be determined by examining the distance matrix. However, to compute the 1-dimensional persistence (the second term) for each cluster, we must construct a Rips complex, which is typically computationally intensive. Moreover, this term is substantially smaller than the other terms. Therefore, we approximate our Wasserstein distance using only the first and third terms.

3.4.5 Measurements of *NC2*

The limitations of employing $\mathcal{NC}_2$ with batched data are detailed in section 3.2.1. To address this, we introduce a persistent homology variant to gauge the strength of *NC2* via the Wasserstein distance, offering a more intuitive approach. The underlying principle

is that, given mean vectors of a batch $\left(H^{(\text{batch})}, Y^{(\text{batch})}\right)$, the 1-skeleton of the simplex formed by these vectors should tend towards regularity. Assessing the deviation from a regular simplex's 1-skeleton allows for an evaluation of the strength of *NC2*. However, this score is influenced by the number of classes and the scale of feature vectors, necessitating normalization factors. In an ideal *NC2* scenario where mean vectors are centered unit vectors, the regular simplex generated by them has an edge-length of $\sqrt{\frac{2K}{K-1}}$.

Definition 3.4.5 (Topological $NC2$ Metric $\mathcal{NC}_2^{PH}$)**.** Given a batched data $\left(H^{(\text{batch})}, Y^{(\text{batch})}\right)$ with $l = |Y^{(\text{batch})}|$ distinct labels, let $G^{(\text{batch})}$ and $\mu_i^{(\text{batch})}$ be as previously defined in definition 3.4.3. Define $\mu^{(\text{batch})}$ as the barycenter of $G^{(\text{batch})}$'s vertices and c as the mean distance from each $\mu_{i_j}^{(\text{batch})}$ to $\mu^{(\text{batch})}$:

$$c = \mathbb{E}_j \left[\left\| \mu_{i_j}^{(\text{batch})} - \mu^{(\text{batch})} \right\| \right].$$

Construct $G_{normal}^{(\text{batch})}$ as the 1-skeleton of the simplex with vertices:

$$\left\{ \frac{1}{c}\left(\mu_{i_1}^{(\text{batch})} - \mu^{(\text{batch})}\right), \cdots, \frac{1}{c}\left(\mu_{i_l}^{(\text{batch})} - \mu^{(\text{batch})}\right) \right\}.$$

The topological $NC2$ metric, denoted as $\mathcal{NC}_2^{PH}$, is then defined as:

$$\begin{aligned} \mathcal{NC}_2^{PH} &= \frac{1}{l} W^2 \left(PD_{\cdot}(G_{normal}^{(\text{batch})}), PD_{\cdot}(\sigma^{(1)}) \right) \\ &= \mathbb{E}_{(j,k)} \left[\left(\frac{1}{c} \left\| \mu_{i_j}^{(\text{batch})} - \mu_{i_k}^{(\text{batch})} \right\| - \sqrt{\frac{2K}{K-1}} \right)^2 \right], \end{aligned}$$

where σ represents a regular simplex with l vertices with edge-length $\sqrt{\frac{2K}{K-1}}$.

3.5 Empirical Results

Our primary objective in introducing the topological loss term at the end of the training process is to bolster the train-*NC*. To elaborate:

1. We initially train a base model for 200 epochs for each classification task.

2. Subsequently, we modify the loss function, transitioning from cross-entropy to topological loss, and continue training for an additional 5 epochs. We refer to this as the "topology-optimized model."

3. For a fair comparison, we also train the base model for an extra 5 epochs using the cross-entropy loss. This model is denoted as the "vanilla model."

It is essential to note that the exclusive use of the topology-optimization step can potentially degrade the model's performance on both the training and testing sets. This decline arises from eschewing the conventional cross-entropy in favor of solely relying on the topological loss term. Contrary to expectations, however, this modification enhances the model's performance. Our findings indicate the following:

- The topological loss fosters a more robust *NC* behavior in both the training and testing sets.
- With the increased feature collapse, the model's generalization performance improves.
- Pursuing regularity is advantageous even in an imbalanced setting, where the optimal *NC* is elusive.

3.5.1 Experimental Setup

We evaluated our method in both balanced and imbalanced settings.

Datasets. We use three popular datasets, MNIST, FashionMNIST, and CIFAR10. Their imbalanced version will be denoted as MNIST-I, FashionMNIST-I, and CIFAR10-I, respectively. For the balanced setting, we used 3000 samples per class. In the imbalanced setting, we utilized 3000 samples for the first 3 classes, 1000 samples for the next 4 classes, and 500 samples for the remaining 3 classes. For test sets, we just use the whole set.

Architectures. We employed a modified ResNet18 architecture.

Method. We train modified ResNet18 for MNIST, FashionMNIST, and CIFAR10 for both balanced and unbalanced setting. For each dataset, we train the model 40 different times for 200 epochs to measure the performance of the model more precisely.

We use Adam optimizer with cross-entropy loss for the base model. After 200 epochs, we train 5 more epochs with topological loss, and with the same cross-entropy for comparison.

3.5.2 Results

Topological Loss Induces Stronger *NC* We compare the *NC* metric after applying the topological loss. The experimental details can be found in Table 3.1 and 3.2. Experiments show that the topological loss function effectively intensify the *NC*.

Topological Loss Improves Model Performance As the above experiments show, topological loss effectively intensify the *NC*. The next question is, more importantly, whether enhanced *NC* leads to better generalized model. We compare the model's score on the test set.

3.A Supplementary Materials

3.A.1 Experiments

Details of additional experiments can be mentioned here.

	Train set		Test set	
Dataset	Vanilla	Topology optimized	Vanilla	Topology optimized
MNIST	0.059 ± 0.006	**0.025 ± 0.003**	0.069 ± 0.006	**0.034 ± 0.002**
FashionMNIST	0.143 ± 0.009	**0.070 ± 0.005**	0.232 ± 0.014	**0.143 ± 0.006**
CIFAR10	0.282 ± 0.012	**0.193± 0.011**	0.533 ± 0.019	**0.424 ± 0.015**
MNIST-I	0.044 ± 0.010	**0.025 ± 0.005**	0.075 ± 0.009	**0.052 ± 0.005**
FashionMNIST-I	0.112 ± 0.016	**0.070 ± 0.010**	0.243 ± 0.018	**0.192 ± 0.007**
CIFAR10-I	0.245 ± 0.042	**0.180± 0.023**	0.685 ± 0.038	**0.605 ± 0.020**

Table 3.1: Comparing $\mathcal{NC}_1$ on Vanilla model with topology-optimized model

	Train set		Test set	
Dataset	Vanilla	Topology optimized	Vanilla	Topology optimized
MNIST	0.087 ± 0.020	**0.058 ± 0.009**	0.094 ± 0.021	**0.064 ± 0.009**
FashionMNIST	0.156 ± 0.031	**0.147 ± 0.020**	0.265 ± 0.042	**0.256 ± 0.024**
CIFAR10	0.240 ± 0.032	**0.210± 0.017**	0.506 ± 0.051	**0.478 ± 0.024**
MNIST-I	0.178 ± 0.057	**0.106 ± 0.016**	0.173 ± 0.055	**0.111 ± 0.016**
FashionMNIST-I	0.245 ± 0.075	**0.157 ± 0.026**	0.385 ± 0.083	**0.293 ± 0.029**
CIFAR10-I	0.259 ± 0.058	**0.210± 0.042**	0.684 ± 0.081	**0.660 ± 0.063**

Table 3.2: Comparing $\mathcal{NC}_2$ on Vanilla model with topology-optimized model

3.A.2 Proof of Theorem 3.4.2

Proof. Since G is a graph with equal edge-lengths, the persistence diagram $PD_\cdot(G)$ is simple. For $PD_0(G)$ consists of a single $(0,\infty)$, and $(0,R)$ with multiplicity $K-1$. For $PD_1(G)$, there is a (R,∞) with multiplicity $\binom{K-1}{2} = \binom{K}{2} - (K-1)$.

In $PD_0(\mathcal{K})$, there are within-class topological features and between-class topological features. Note that each within-class features correspond to points of form $(0,d)$, where $d < 2r$. This is because each H_i in contained in a ball of radius r. These points should be matched with diagonal points, with matching cost

$$\left\| (0,d) - \left(\frac{d}{2}, \frac{d}{2}\right) \right\|_1 = d < 2r.$$

If $n_i = |H_i|$, then the number of within-class features in each H_i is $n_i - 1$. This implies that the matching cost of all within-class features are less than

$$\sum_{i=1}^{K} 2r(n_i - 1).$$

For between-class 0-dimensional features, they are $K-1$ points of the form $(0,R')$, where $|R - R'| < 2\epsilon$. The total matching cost will be less than

$$\sum_{i=1}^{K} 2r(n_i - 1) + 2\epsilon(K-1).$$

Finally

$$W^2(PD_0(\mathcal{K}), PD_0(G)) \leq \sum_{i=1}^{K} (2r)^2(n_i - 1) + (2\epsilon)^2(K - 1).$$

□

Chapter 4

xPerT: Extended Persistence Transformer

A persistence diagram provides a compact summary of persistent homology, which captures the topological features of a space at different scales. However, due to its nature as a set, incorporating it as a feature into a machine learning framework is challenging. Several methods have been proposed to use persistence diagrams as input for machine learning models, but they often require complex preprocessing steps and extensive hyperparameter tuning. In this paper, we propose a novel transformer architecture called the *Extended Persistence Transformer (xPerT)*, which is highly scalable than the compared to Persformer, an existing transformer for persistence diagrams. xPerT reduces GPU memory usage by over 90% and improves accuracy on multiple datasets. Additionally, xPerT does not require complicated preprocessing steps or extensive hyperparameter tuning, making it easy to use in practice.

4.1 Introduction

Topological Data Analysis (TDA) uses ideas from topology to explore the shape and structure of data, revealing patterns that traditional statistical methods may not fully grasp. TDA is not only an active area of mathematical research, but it also has broad practical applications across various fields. One of the key tools in TDA is *persistent homology*, which captures the multi-scale topological features of a dataset. A summary of persistent homology is provided by the *persistence diagram*, which is a multi-set of points

in the plane.

Persistent homology has been utilized in a wide range of areas, including biomolecules [35, 58], material science [29, 39], meteorology [47], and image analysis [45, 49]. However, the integration of TDA with machine learning models remains a challenge due to several reasons: (1) persistence diagram is an unordered set, which is not a natural input for machine learning models, (2) each persistence diagram has a different number of points, which complicates batch processing. Several methods have been proposed to address this issue by converting persistence diagrams into fixed-size feature vectors [1, 8, 43]. However, these methods often require extensive hyperparameter tuning and may fail to capture the full extent of topological information in the data. Another line of research explores using neural networks to directly process persistence diagrams, such as PersLay [9] and PLLay [26]. While these models have shown promising results, their application is limited due to complex hyperparameter choices and implementation difficulties.

The transformer architecture [55] has recently emerged as a powerful model in many domains, including natural language processing, computer vision, audio etc. Persistence diagrams have also been incorporated into transformer models, as demonstrated by *Persformer* (Reinauer et al., 2022). However, Persformer faces scalability issues, and its training process is often unstable in practice. In this work, we propose a novel transformer architecture for (extended) persistence diagrams called Extended Persistence Transformer (xPerT). The xPerT can directly process persistence diagrams, without requiring common preprocessing steps often needed in existing methods. The xPerT is highly scalable in terms of training time and GPU memory usage, and it does not require extensive hyperparameter tuning. We demonstrate the effectiveness of the xPerT model on classification tasks using two datasets: graph datasets and a dynamical system dataset.

4.1.1 Related Works

The use of persistent homology in machine learning has been explored in various studies. One of the earliest approaches is the vectorization method, which converts a persistence diagram into a fixed-size feature vector. The *persistence landscape* [8] and *persistence image* [1] are two popular vectorization methods. The persistence image also transforms a persistence diagram, but directly into a fixed-size vector by placing Gaussian kernels at each point and summing the values over a predefined grid. Both methods require the selection of numerous hyperparameters, which play a crucial role in determining the

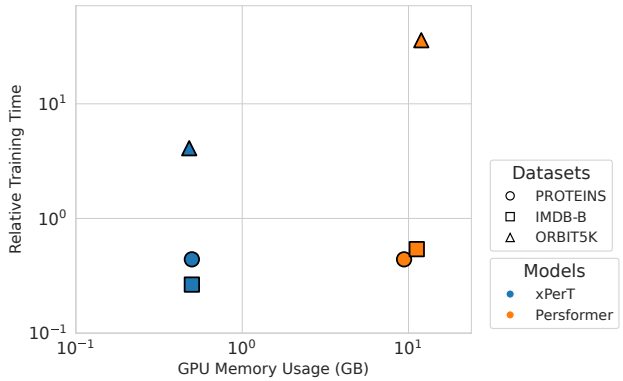


Figure 4.1: **Scaling.** Comparison of computational cost between persistence diagram transformer models in terms of training time and GPU memory usage (GB). The experiment was conducted using a batch size of 64 for the `PROTEINS` and IMDB-B datasets, and a batch size of 16 for the `ORBIT5K` dataset, as Persformer could not fit on our GPU (RTX 3090) with a batch size of 32.

quality of the resulting vector, making the cross-validation process complex. *ATOL* [43] offers an unsupervised vectorization approach by leveraging k-means clustering.

While most existing methods focus on vectorizing single-parameter persistent homology, extending these techniques to multi-parameter persistent homology is challenging due to the lack of a natural representation. However, some recent approaches have begun addressing these challenges, such as *GRIL* [59] and *HSM-MP-SW* [33].

Unlike vectorization methods, neural network-based approaches utilze persistence diagrams more directly. *PersLay* [9] maps a persistence diagram to a real number by:

$$\text{PersLay}(D) := op\left(\{w(p)\cdot\phi(p)\}_{p\in D}\right),$$

where D denotes a persistence diagram, $w : \mathbb{R}^2 \to \mathbb{R}$ is a learnable weight function, $\phi : \mathbb{R}^2 \to \mathbb{R}^d$ is a point transformation, and op is a permutation-invariant operator. While PersLay processes persistence diagrams directly, *PLLay* [26] first computes the persistence landscape and then applies a neural network to the resulting feature vector.

More recently, [41] introduced *Persformer*, a model that applies a transformer architecture to persistence diagrams. In this approach, each point in the diagram is treated as a token, and the transformer operates without positional encodings. In batch processing, Persformer pads dummy points during preprocessing to ensure uniformity in the number of points across diagrams. Though simple, this method involves processing numerous

tokens, leading to high computational costs and substantial GPU memory usage.

Contributions In this paper, we introduce the Extended Persistence Transformer (xPerT), a novel transformer architecture tailored for handling (extended) persistence diagrams in topological data analysis. Our main contributions are:

- **Persistence Diagram Transformer**: We propose xPerT, which bridges the gap between persistence diagrams and transformer models by discretizing the diagrams into pixelized representations suitable for tokenization and input into the transformer architecture.
- **Scalability through Sparsity**: By leveraging the inherent sparsity of persistence diagrams, xPerT achieves high scalability in training time and GPU memory usage, making it efficient for large-scale applications.
- **Practical Implementation**: Our method is easy to implement and requires minimal hyperparameter tuning, lowering the barrier for practitioners and facilitating quick adoption.

4.2 Background

4.2.1 Persistent Homology

Persistent homology studies the evolution of a space, capturing its topological features at different scales. This section briefly introduces the fundamental concepts of persistent homology, and see 4.B for more information. For readers unfamiliar with persistent homology, we recommend [17] for a comprehensive treatment.

Let $f : X \to \mathbb{R}$ be a continuous function from a topological space X. The *c-sublevel set* of f, defined as $X_c = \{x \in X : f(x) \leq c\}$ for $c \in \mathbb{R}$, is an essential object for the understanding the topology of X. These sublevel sets $(X_c)_{c\in\mathbb{R}}$ form an increasing sequence of spaces, known as a *sublevel set filtration* as described in Figure 4.2. In particular, sublevel sets are central in *Morse theory*, which analyzes the topology of X by studying the function f.

A sequence of homology groups $\{H_k(X_c)\}_{c\in\mathbb{R}}$ build upon the filtration $(X_c)_{c\in sR}$ is called a *k-dimensional persistent homology*. Note that there is a natural map $H_k(X_c) \to H_k(X_{c'})$

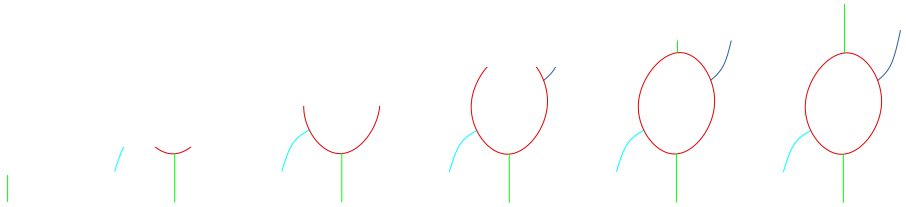


Figure 4.2: **Sublevel Set Filtration.** Six sublevel sets of the height function are shown. As c increases, the topology of X_c changes, which is captured by the ordinary persistence diagram. However, ordinary persistence cannot detect the appearance of the blue upright arm, which can instead be captured by the superlevel set filtration $(X^c)_{c\in\mathbb{R}}$ as c decreases.

induced by the inclusion $X_c \hookrightarrow X_{c'}$ for $c \leq c'$. The evolution of topological features through the maps $H_k(X_c) \to H_k(X_{c'})$ is encoded in the *persistence diagram*, which is a multiset of points in the extended plane $\mathbb{R}\times(\mathbb{R}\cup\{\infty\})$. If a topological feature appears at X_b and disappears at X_d for $b < d$, the point (b, d) is included in the persistence diagram. If a topological feature appears at X_b and persists indefinitely, the point (b, ∞) is added. See the supplementary material in 4.B for more details.

However, standard persistence diagrams capture only limited topological information since they focus solelyon the sublevel sets. To address this, *extended persistence* was introduced in [14], incorporating additional information by utilizing the *c-superlevel set* $X^c = \{x \in X : f(x) \geq c\}$. An extended persistence diagram E consists of four components: Ord_0, Rel_1, Ext_0^+, and Ext_1^-, capturing more topological information than ordinary persistence. As shown in Figure 4.3, the extended persistence diagram does not contain points at infinity, simplifying its use in machine learning models as well.

4.2.2 Wasserstein Distance

The collection of (ordinary) persistence diagrams forms a metric space with the *Wasserstein distance*, which provides a measure of dissimilarity between two persistence diagrams. The Wasserstein distance is defined to be the infimum of the cost between all possible matchings γ between two persistence diagrams.

Definition 4.2.1 (Wasserstein Distance)**.** Given a persistence diagram D, let $\mathrm{aug}(D)$ be the union of D and all points in the diagonal $\Delta = \{(x, x) : x \in \mathbb{R}\}$ with infinite multiplicity. For two persistence diagrams D and D', the (p, q)-Wasserstein distance

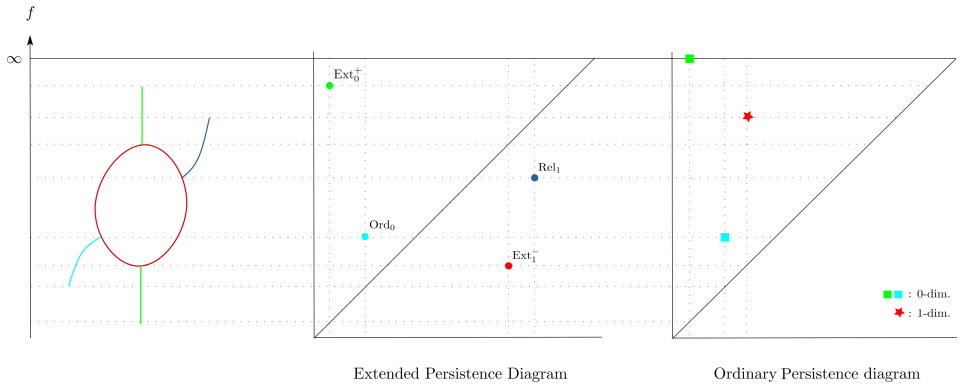


Figure 4.3: **Persistence Diagram.** The extended persistence diagram captures more information than the ordinary persistence diagram. In this figure, the extended persistence diagram includes information about the blue upright arm and the maximum value of f, which are not present in the ordinary persistence diagram. (Left) A topological space equipped with a height function f. (Center) Extended persistence diagram. (Right) Ordinary persistence diagram.

between D and D' is defined as

$$W_{p,q}(D, D^{'}) = \inf_{\gamma} \left(\sum_{u \in \text{aug}(D)} \|u - \gamma(u)\|_q^p \right)^{1/p},$$

where $\gamma : \text{aug}(D) \to \text{aug}(D^{'})$ ranges over all bijections.

Intuitively, the Wasserstein distance measures how much 'work' is required to match the points in one diagram to those in another, accounting for both the distance between points and the number of points involved. Throughout this paper, we will use the $(1, 2)$-Wasserstein distance $W = W_{1,2}$.

4.2.3 Heat Kernel Signature

The *Heat Kernel Signature (HKS)* is a feature descriptor that captures the intrinsic geometric properties of a shape. Originally defined for Riemannian manifolds [48], the discrete version of HKS can also be applied to graphs, allowing us to analyze their structural characteristics. We follow the approach in [9], using HKS values to generate extended persistence diagrams.

To define HKS, we first introduce the *graph Laplacian*, a fundamental tool in graph

analysis.

Definition 4.2.2 (Graph Laplacian)**.** Let $G = (V, E)$ be an undirected graph with n vertices. The *adjacency matrix* A of G is the $n \times n$ matrix defined as

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The *normalized Laplacian matrix* of G is given by

$$L = I - D^{-1/2} A D^{-1/2},$$

where I is the identity matrix, and D is the diagonal matrix with $D_{ii} = \sum_{j=1}^{n} A_{ij}$.

Definition 4.2.3 (Heat Kernel Signature)**.** Given a graph G with a diffusion parameter $t > 0$, the Heat Kernel Signature is the function $H_t : V \to \mathbb{R}$ defined at each node $v \in V$ by

$$H_t(v) = \sum_{i=1}^{n} \exp(-\lambda_i t) \langle \phi_i, v \rangle^2,$$

where λ_i are the eigenvalues and $\langle \phi_i, v \rangle$ is the value of the i-th eigenvector at node v.

For a fixed diffusion parameter $t > 0$, H_t assigns a real number to each node of the graph, encoding the intrinsic geometric properties of the graph. The diffusion parameter t controls the scale at which the graph's geometric features are captured, with smaller values of t focusing on local structures and larger values capturing global properties. By assigning a real number to each node, H_t encodes essential structural information of the graph.

4.3 Pixelized Persistence Diagram

In this section, we introduce the *Pixelized Persistence Diagram (PPD)*, an efficient representation of persistence diagrams designed for transformer inputs. The PPD is constructed by first applying instance normalization to handle varying scales and then projecting the normalized diagram onto a discrete grid. This approach ensures that diagrams of different scales are represented consistently, facilitating their use in machine learning

models while retaining the stability properties of the original diagrams. The process is straightforward and can be implemented in just a few lines of code.

The PPD shares similarities with the Persistence Image (PI) but offers specific advantages when integrated with transformer architectures: (1) PPD requires less or no hyperparameter tuning, and (2) more importantly, PPD contains many zero-value pixels, allowing only non-zero pixels to be utilized. This sparsity significantly reduces the number of tokens processed by the transformer, improving computational efficiency and scalability.

4.3.1 Projection of Persistence Diagrams

Given a rotated persistence diagram D_r, we aim to project it onto a discrete grid to create a pixelized representation. For a fixed grid size $\delta > 0$, we discretize the birth-persistence plane into grid cells:

$$I_{k,l} = [k\delta, (k+1)\delta) \times [l\delta, (l+1)\delta), \quad k, l \in \mathbb{N}.$$

Each point $(b, p) \in D_r$ is associated with the grid cell $I_{k,l}$ containing it.

Definition 4.3.1 (Projection of Persistence Diagram)**.** The projection map $\Pi_\delta : \mathbb{R}^2_{\geq 0} \to \mathbb{R}^2_{\geq 0}$ is defined by mapping each point $(b, p) \in D_r$ to the center of the grid cell containing it:

$$\Pi_\delta(b, p) = \left(\left(k + \tfrac{1}{2}\right)\delta, \left(l + \tfrac{1}{2}\right)\delta\right), \quad \text{where } (b, p) \in I_{k,l}.$$

The *projected persistence diagram* is then $\Pi_\delta(D_r) = \{\Pi_\delta(b, p) : (b, p) \in D_r\}$.

This projection maps each point to the center of its grid cell, effectively quantizing the diagram.

Stability of the Projected Persistence Diagram The projection operation Π_δ applied to any rotated persistence diagram preserves stability with respect to the Wasserstein distance.

Proposition 4.3.2. *Let D and D' be two persistence diagrams, and let $\delta < W(D, D')$. Then we have:*

$$W\left(\Pi_\delta(D_r), \Pi_\delta(D'_r)\right) \leq \left(\frac{\sqrt{|D|} + \sqrt{|D'|}}{\sqrt{2}} + \sqrt{3}\right) W(D, D').$$

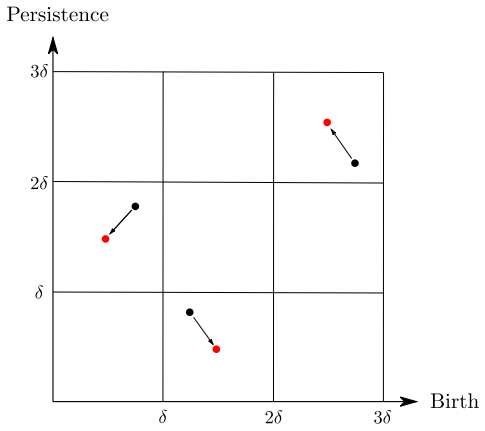


Figure 4.4: **Projection of Persistence Diagram.** Each point in D_r is projected onto the center of its corresponding grid cell using the projection map Π_δ. The arrows indicate the mapping.

Proof. See Appendix 4.A.1. □

4.3.2 Pixelized Persistence Diagram

The projected persistence diagram $\Pi_\delta(D_r)$ is a digital-image-like representation, where each point corresponds to the center of a grid cell. However, the scale of D_r can vary between different persistence diagrams, leading to inconsistencies in the representation. To address this, we apply instance normalization before projecting the diagram.

We define the *instance-nomalized diagram* as

$$\texttt{Norm}(D_r) = \left\{ \left(\frac{b}{b_{\max}}, \frac{p}{p_{\max}} \right) : (b,p) \in D_r \right\},$$

where $b_{\max} = \max\{b : (b,p) \in D_r\}$, $p_{\max} = \max\{p : (b,p) \in D_r\}$ are the maximum values of birth, and persistence in D_r, respectively. This maps the rotated diagram into the unit square.

Remark 4.3.3. *If $b_{max} = 0$ (e.g., in 0-dimensional diagrams where all birth times are zero), we set $b_{max} = 1$ to avoid division by zero. Since all b values are zero, this normalization leaves them unchanged.*

The *Pixelized Persistence Diagram (PPD)* is then the pixelized representation of

$\Pi_\delta\big(\texttt{Norm}(D_r)\big)$, with $\delta = 1/H$. Concretely, let $H > 0$ be a positive integer representing the grid resolution. We partition the unit square $[0,1] \times [0,1]$ into $H \times H$ grid cells:

$$I_{i,j} = \left[\frac{i-1}{H}, \frac{i}{H}\right) \times \left[\frac{j-1}{H}, \frac{j}{H}\right), \quad i,j = 1,2,\ldots,H.$$

Definition 4.3.4 (Pixelized Persistence Diagram)**.** The **Pixelized Persistence Diagram (PPD)** $\mathcal{P}_H(D) \in \mathbb{N}^{H\times H}$ is an integer matrix where each entry $\mathcal{P}_H(D)_{i,j}$ represents the number of points from $\texttt{Norm}(D_r)$ that fall into the (i,j)-th grid cell:

$$\mathcal{P}_H(D)_{i,j} = \left|\{(b,p) \in \texttt{Norm}(D_r) : (b,p) \in I_{i,j}\}\right|.$$

4.3.3 Extended Pixelized Persistence Diagram

Having defined the PPD for a single diagram, we now extend the definition to the extended persistence diagram. The extended pixelized persistence diagram $\mathcal{P}_H(E)$ of an extended persistence diagram $E = \{\mathrm{Ord}_0, \mathrm{Rel}_1, \mathrm{Ext}_0^+, \mathrm{Ext}_1^-\}$ is obtained discretizing each diagram:

1. Transpose the diagrams Rel_1 and Ext_1^- so that all four diagrams are positioned in the upper half-plane, above the diagonal.

2. Rotate the diagrams $\{\mathrm{Ord}_0, (\mathrm{Rel}_1)^T, \mathrm{Ext}_0^+, (\mathrm{Ext}_1^-)^T\}$ to obtain the set of rotated diagrams:
$$\{R_{\mathrm{ord}}, R_{\mathrm{rel}}, R_{\mathrm{ext}^+}, R_{\mathrm{ext}^-}\}.$$

3. Compute the pixelized persistence diagrams (PPDs) for each rotated diagram:
$$\mathcal{P}_H(E) = \{\mathcal{P}_H(R_{\mathrm{ord}}), \mathcal{P}_H(R_{\mathrm{rel}}), \mathcal{P}_H(R_{\mathrm{ext}^+}), \mathcal{P}_H(R_{\mathrm{ext}^-})\}^1.$$

4.4 xPerT

We now describe the Extended Persistence Transformer (xPerT), illustrated in Figure 4.5. The xPerT model processes a set of persistence diagrams by transforming them into sequences of tokens, which are then fed into a transformer model.

[1]When computing the PPDs for the extended persistence diagram, the same $b_{\max}$ and $p_{\max}$ values are applied to all four diagrams to ensure consistency.

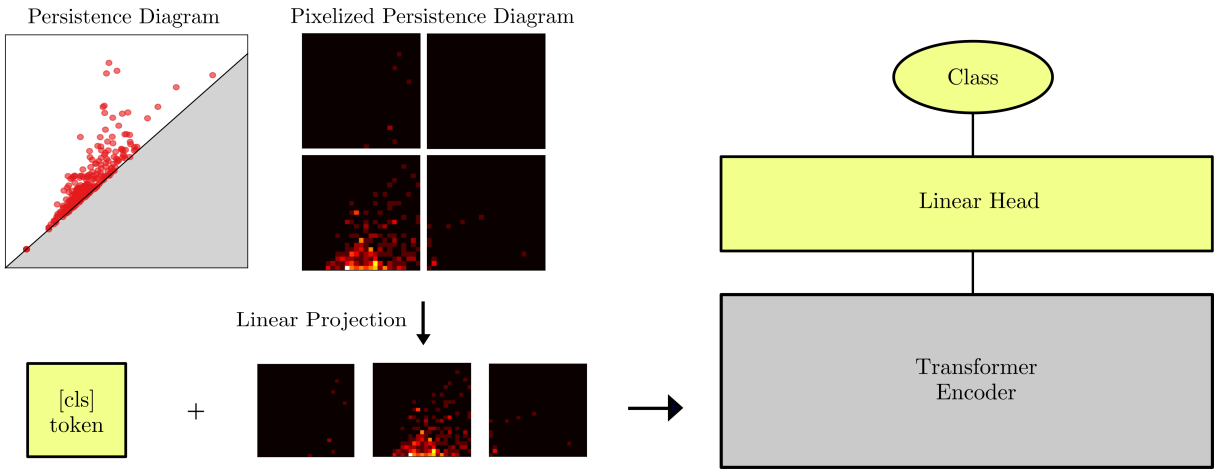


Figure 4.5: **xPerT Overview.** The persistence diagram is pixelized and split into fixed-size patches (left, top). These patches are linearly transformed into token vectors, with a `[cls]` token added (left, bottom). Empty patches are excluded from the transformer input.(Right) The token vectors are processed by the transformer model, and the output of the `[cls]` token is fed to the linear head for classification.

4.4.1 Tokenization

To feed persistence diagrams into a transformer, they must first be converted into sequences of tokens. Here, we describe the tokenization process for extended persistence diagrams. Due to the unique nature of 0-dimensional persistence diagrams, where points typically lie along the y-axis, a slightly different tokenization method is used, as detailed in Section 4.B.1 of the supplementary material.

Given an extended persistence diagram E, we first discretize it into an extended PPD $\mathcal{P}_H(E)$ (section 4.3.3), where $H > 0$ is the resolution. Let $\mathcal{P} \in \mathbb{N}^{4\times H\times H}$ be the tensor obtained by stacking individual PPD in $\mathcal{P}_H(E)$ along the channel dimension. This multi-channel representation is then divided into $N = (H/P)^2$ patches, where each patch is of size $4 \times P \times P$, and P is the patch size. Each patch is then flattened into a vector in $\mathbb{N}^{4P^2}$, resulting in the sequence $\{\mathcal{P}^1_{\text{patch}}, \ldots, \mathcal{P}^N_{\text{patch}}\}$.

Finally, we apply a linear transformation to each patch to obtain token embeddings:

$$\text{Tokens} = \left\{\mathbf{E}\mathcal{P}^1_{\text{patch}}, \ldots, \mathbf{E}\mathcal{P}^N_{\text{patch}}\right\}, \quad \mathbf{E}\mathcal{P}^i_{\text{patch}} \in \mathbb{R}^D,$$

where $\mathbf{E} \in \mathbb{R}^{D\times(4P^2)}$ is a learnable projection matrix, and D is the embedding dimension. Note that this tokenization approach closely parallels the process used in the Vision

Transformer [16], where images are similarly divided into patches and transformed into token embeddings.

After tokenization, we prepend a classification `[cls]` token to the sequence, which aggregates information for downstream tasks. To retain spatial relationships, we add standard 2D sinusoidal positional encodings to the token embeddings.

Sparsity of Patches. One of the key advantages of xPerT is its ability to leverage the inherent sparsity in persistence diagrams. Points in persistence diagrams are often distributed non-uniformly, resulting in many patches being zero vectors. The xPerT model takes advantage of this sparsity by processing only the non-zero patches, which significantly reduces the number of tokens and computational costs.

Classification Head. For classification, we use the `[cls]` token combined with max pooling over all token embeddings, which slightly improves performance compared to using the `[cls]` token alone. Specifically, we apply a single-layer linear head to the sum of the `[cls]` token and the pooled token embeddings, followed by a softmax layer to output the probability distribution over the classes.

4.5 Experiments

In this section, we present the experimental results of the xPerT model on graph and dynamical system classification tasks. We compare xPerT's performance with state-of-the-art methods related to persistent homology, including both single-parameter and multi-parameter approaches.

Architecture. We use the same xPerT architecture for all experiments to demonstrate that it performs well without extensive hyperparameter tuning.[2] The transformer model used in the experiments consists of 5 layers with 8 attention heads, with a token dimension set to 192. The resolution of the pixelized persistence diagram (PPD) is $H = 50$, with a patch size of $P = 5$, resulting in at most 100 patches per (extended) persistence diagram. However, due to the sparsity of the diagrams, the actual number of patches is often much smaller. For instance, the average number of non-zero patches in the `ORBIT5K` dataset is 8.4 for the 0-dimensional diagrams and 23.2 for the 1-dimensional diagrams.

[2]Further hyperparameter optimization could improve performance, as shown in the ablation study.

Table 4.1: **Graph classification.** The average classification accuracy across different datasets over 10-fold cross-validation. (Top) Methods leveraging single or multi-parameter persistent homology. (Bottom) Methods that combine persistent homology with Graph Isomorphism Network (GIN) models. The best results are highlighted in bold, and the second-best are underlined.

Method	IMDB-B	IMDB-M	MUTAG	PROTEINS	COX2	DHFR
PersLay†	71.2	48.8	<u>89.8</u>	<u>74.8</u>	<u>80.9</u>	80.3
ATOL	69.2 ± 4.1	42.9 ± 2.3	88.3 ± 3.9	72.6 ± 1.9	80.0 ± 7.6	81.2 ± 4.8
HSM-MP-SW†	**74.7 ± 5.0**	<u>50.3 ± 3.5</u>	86.8 ± 7.1	74.1 ± 2.0	77.9 ± 1.3	**82.8 ± 5.0**
GRIL†	65.2 ± 2.6	−[3]	87.8 ± 4.2	70.9 ± 3.1	79.8 ± 2.9	77.6 ± 2.5
Persformer	68.9 ± 8.8	**51.7 ± 3.3**	89.4 ± 4.0	72.0 ± 6.7	78.2 ±0.8 [4]	64.8 ± 2.3
xPerT	<u>72.6 ± 3.4</u>	50.0 ± 2.0	**91.0 ± 5.2**	**75.7 ± 3.8**	**84.4 ± 3.5**	<u>81.9 ± 2.9</u>
GIN	<u>75.3 ± 4.8</u>	**52.3 ± 3.6**	<u>94.1 ± 3.8</u>	76.6 ± 3.0	85.4 ± 3.4	**84.5 ± 3.6**
+ GRIL(sum)†	74.2 ± 2.8	−[3]	89.3 ± 4.8	71.9 ± 3.2	79.2 ± 4.9	78.5 ± 5.8
+ xPerT(sum)	**76.1 ± 2.7**	51.1 ± 1.9	93.7 ± 3.9	<u>77.6 ± 4.7</u>	**85.9 ± 2.9**	<u>83.5 ± 3.4</u>
+ xPerT(cat)	75.7 ± 2.3	<u>52.1 ± 2.0</u>	**94.7 ± 5.3**	**78.4 ± 5.1**	<u>85.7 ± 3.9</u>	81.9 ± 2.6

4.5.1 Classification on Graph Datasets

Given a graph, we compute the heat kernel signature (HKS) on the graph with diffusion parameter $t = 1.0$, which is used to generate the extended persistence diagram. For detailed hyperparameters, see the supplementary material in 4.A.2.

Graph Datasets. We evaluate xPerT on several widely used graph classification datasets: IMDB-BINARY , IMDB-MULTI, MUTAG, PROTEINS, COX2, and DHFR. The IMDB datasets consist of social network graphs, while the remaining datasets are derived from biological and medical domains (see [36] for details).

Baselines and Results. Table 4.1 summarizes the results of xPerT compared with state-of-the-art persistent homology-related methods, as described in Section 5.2.3. We compare xPerT with PersLay [9], ATOL [43], and Persformer [41], which, as previously discussed, use extended persistence diagrams generated from the HKS function. Addi-

[3]GRIL was not evaluated on the IMDB-MULTI dataset.

[4]The model consistently predicts the majority class, leading to inflated accuracy due to class imbalance.

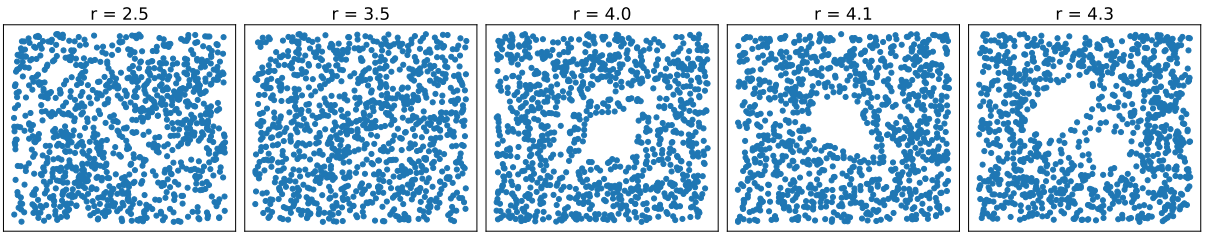


Figure 4.6: Examples of orbit datasets with different value of r.

tionally, we compare xPerT with GRIL [59] and HSM-MP-SW [33], which employ multi-parameter persistent homology without relying on persistence diagrams. This allows us to evaluate xPerT's performance relative to both single-parameter and multi-parameter persistent homology approaches. Detailed settings for each model are provided in Table 4.7 in the supplementary material.

In summary, xPerT demonstrates strong and consistent performance across various datasets, often surpassing or matching state-of-the-art methods. Furthermore, xPerT is flexible and can be seamlessly combined with models like GIN, making it adaptable for use with other deep learning architectures.

4.5.2 Dynamical System Dataset.

Table 4.2 shows the average classification accuracy of xPerT on the dynamical system datasets over 5 independent runs. We evaluate xPerT on two commonly used datasets in topological data analysis: `ORBIT5K` and `ORBIT100K`. These datasets consist of simulated orbits with distinct topological characteristics, generated using the recursive equations:

$$x_{n+1} = x_n + r y_n(1 - y_n) \mod 1$$
$$y_{n+1} = y_n + r x_{n+1}(1 - x_{n+1}) \mod 1$$

Each orbit is initialized with a random starting point $(x_0, y_0) \in [0, 1]^2$ and a parameter $r \in \{2.5, 3.5, 4.0, 4.1, 4.3\}$. The task is to predict the parameter r that generated each orbit, reflecting distinct topological behaviors in the system. The `ORBIT5K` dataset contains 5,000 point clouds, each with 1,000 points per orbit, while the larger `ORBIT100K` dataset consists of 100,000 point clouds, with 20,000 orbits for each r value. Both datasets are split into 70% training and 30% testing sets.

Table 4.2: **Orbit Classification.** Mean classification accuracy over 5 independent runs for the ORBIT5K and ORBIT100K datasets. Results marked with † are taken from the original papers.

Method	ORBIT5K	ORBIT100K
PersLay†	87.7 ± 1.0	89.2 ± 0.3
ATOL	72.2 ± 1.5	68.8 ± 8.0
Persformer†	**91.2 ± 0.8**	**92.0 ± 0.4**
Persformer[5]	28.2 ± 7.3	–
xPerT	87.0 ± 0.7	91.1 ± 0.1

Table 4.3: **Ablating patch size (Orbit).** Impact of varying patch sizes on classification performance for the ORBIT5K and ORBIT100K datasets. The results show that xPerT benefits from smaller patch sizes, with mean accuracy reported over 5 independent runs.

Patch Size	ORBIT5K	ORBIT100K
$P = 2$	**88.1 ± 0.4**	**90.8 ± 0.1**
$P = 5$	87.0 ± 0.7	90.2 ± 0.1
$P = 10$	80.5 ± 1.2	89.8 ± 0.3

4.6 Ablation Study

4.6.1 Patch Size.

Tables 4.3 and 4.4 show the effect of patch size on the dynamical system and graph datasets, respectively. Reducing the patch size increases the number of patches, providing a finer resolution for the pixelized persistence diagrams. We observe that the ORBIT5K dataset benefits significantly from smaller patch sizes, as they allow the model to capture more detailed topological information. However, the effect of decreasing the patch size is less pronounced on the ORBIT100K dataset, likely due to the increased scale of the dataset.

In contrast, the effect of patch size on the graph datasets is less evident. This may be due to the fact that persistence diagrams generated from graph datasets tend to have much fewer points compared to those from the dynamical system datasets. As a result, reducing the patch size may not provide significant additional information, since the smaller number of points limits the amount of detail that can be captured in the pixelized representation.

[5]Reproduced result. The model was not trainable in both ORBIT5K and ORBIT100K. We used the code from https://github.com/giotto-ai/giotto-deep.

Table 4.4: **Ablating patch size (Graph).** Average accuracies over 10-fold cross-validation. The effect of patch size is less clear in graph datasets.

Patch Size	IMDB-B	IMDB-M	MUTAG	PROTEINS	COX2	DHFR
$P = 2$	71.9 ± 3.8	48.1 ± 4.0	89.4 ± 5.7	**75.7 ± 3.0**	82.6 ± 4.6	80.7 ± 2.4
$P = 5$	72.6 ± 3.4	**50.0 ± 2.0**	**90.0 ± 5.2**	**75.7 ± 3.8**	**84.4 ± 3.5**	**81.9 ± 2.9**
$P = 10$	**74.5 ± 4.1**	49.2 ± 3.3	89.9 ± 6.1	75.5 ± 3.4	82.7 ± 3.6	81.6 ± 4.2

4.6.2 Model Size.

Tables 4.5 and 4.6 show the impact of model depth and width on classification performance across the orbit and graph datasets. While the effects of depth and width are less evident in the graph datasets, we observe that xPerT performs robustly across a range of model configurations in the ORBIT datasets. Models with 5 layers and a width of 192 showed solid performance, but overall, the model achieves consistent results across different depths and widths. This suggests that xPerT is flexible and performs well without being overly sensitive to model size.

Table 4.5: **Effect of depth.**

Depth	PROTEINS	COX2	ORBIT5K
2	75.0 ± 1.9	83.1 ± 3.4	86.2 ± 0.9
5	**75.7 ± 2.7**	83.3 ± 4.3	**87.0 ± 0.7**
8	75.1 ± 4.3	**83.9 ± 4.1**	86.3 ± 0.9

Table 4.6: **Effect of width.**

Width	PROTEINS	COX2	ORBIT5K
96	75.5 ± 4.9	83.1 ± 5.6	86.4 ± 1.0
192	**75.7 ± 2.7**	**83.3 ± 4.3**	**87.0 ± 0.7**
384	75.5 ± 2.1	82.9 ± 3.0	86.6 ± 1.0

4.7 Conclusion and Limitations

In this work, we introduced xPerT, a novel transformer architecture specifically designed for persistence diagrams, enabling efficient handling of topological information in data. xPerT's design allows for easy integration with other machine learning models while efficiently handling the sparse nature of persistence diagrams, significantly reducing computational complexity compared to previous transformer models for persistence diagrams.

We demonstrated the effectiveness of xPerT on both graph classification and dynamical system classification tasks, where it outperforms other methods that use (extended) persistence diagrams as machine learning features in several datasets. Furthermore, we conducted an ablation study to investigate the effects of patch size and model size on performance. Our results indicate that xPerT performs robustly across a wide range of hyperparameters and is resilient to changes in model size and grid resolution.

While xPerT shows great promise for topological data analysis, a key limitation of our study is that the model was tested on a limited number of datasets. In future work, we plan to evaluate xPerT on a more diverse set of datasets from different domains to better understand its generalizability and scalability.

In conclusion, xPerT offers a promising approach to utilizing topological data for machine learning tasks. Its ease of integration with other models, combined with its robust performance, makes xPerT a valuable tool for a wide range of applications, with the potential for further improvements through future research.

4.8 Reproducibility Statement

We are committed to ensuring that the experiments conducted in this paper can be fully reproduced. To this end, we provide the following resources and information:

- **Code Availability:** All code necessary to replicate our results, including the xPerT model implementation, data preprocessing scripts, and evaluation metrics, will be made publicly available on GitHub. The codebase is implemented in `[language/framework]`, and instructions for setting up the environment are included in the repository's README.

- **Datasets:** The datasets used in this work, including the graph datasets (IMDB-BINARY, IMDB-MULTI, MUTAG, PROTEINS, COX2, and DHFR) and the dy-

namical system datasets (ORBIT5K and ORBIT100K), are either publicly available or will be provided along with the code. Detailed instructions for downloading and preprocessing these datasets are included in the repository.

- **Hyperparameters and Model Configurations:** All hyperparameters, such as learning rates, batch sizes, and the number of epochs, are clearly specified in the code repository. Additionally, model configurations, including depth, width, patch size, and grid resolution, are explicitly documented to ensure consistent replication of our results. An ablation study exploring the effect of various hyperparameter settings is provided in the paper.
- **Computing Resources:** The experiments were conducted on a machine with an `NVIDIA RTX3090 GPU`.

We encourage others to use the provided resources to replicate our findings and build upon our work.

4.A Supplementary Material for xPerT

4.A.1 Proofs

Proof of Proposition 4.3.2

We first state and prove two lemmas that will be used in the proof of Proposition 4.3.2.

Lemma 4.A.1. *Let D_r be a rotated persistence diagram. Then*

$$W(\Pi_\delta(D_r), D_r) \leq \sqrt{\frac{|D|}{2}}\delta.$$

Proof. The cost of the matching $\Pi_\delta : D_r \to \Pi_\delta(D_r)$ is given by

$$\left(\sum_{u\in D_r} \|u - \Pi_\delta(u)\|_2^2\right)^{1/2} \leq \left(\sum_{u\in D_r} \frac{1}{2}\delta^2\right)^{1/2} = \sqrt{\frac{|D|}{2}}\delta.$$

□

Lemma 4.A.2. *Let D and D' be two persistence diagrams. The Wasserstein distance between their rotated versions D_r and D'_r satisfies the following inequality:*

$$W\left(D_r, D'_r\right) \leq \sqrt{3}W(D, D').$$

Proof. Let $\tilde{\gamma}$ be the optimal matching between D and D' that achieves the Wasserstein distance $W(D, D')$. We decompose the persistence diagrams into disjoint unions:

$$D = D_{\text{match}} \sqcup D_{\text{diag}}, \quad D' = D'_{\text{match}} \sqcup D'_{\text{diag}},$$

where D_{match} is the set of points matched to D'_{match}, and D_{diag}, D'_{diag} are the points matched to the diagonal. Let $D_{\text{match}} = \{u_1, \ldots, u_k\}$, $D'_{\text{match}} = \{v_1, \ldots, v_k\}$, and assume without loss of generality that $D_{\text{match}} \neq \varnothing$ and $v_i = \tilde{\gamma}(u_i)$ for $1 \leq i \leq k$. The Wasserstein distance between D and D' is given by

$$W^2(D, D') = \sum_{i=1}^{k} \|u_i - v_i\|_2^2 + \sum_{i=k+1}^{k+l} \|u_i - \pi(u_i)\|_2^2 + \sum_{i=k+1}^{k+m} \|v_i - \pi(v_i)\|_2^2, \tag{4.A.1}$$

where π denotes the projection onto the diagonal.

Now, consider the Wasserstein distance between the rotated diagrams D_r and D'_r. Using the matching $\gamma = R\tilde{\gamma}R^{-1}$ between D_r and D'_r, we have

$$W^2(D_r, D'_r) \leq \sum_{i=1}^{k} \|Ru_i - Rv_i\|_2^2 + \sum_{i=k+1}^{k+l} \|Ru_i - \pi_x(Ru_i)\|_2^2 + \sum_{i=k+1}^{k+m} \|Rv_i - \pi_x(Rv_i)\|_2^2,$$

where R is the rotation matrix and π_x is the projection in the birth-persistence plane.

Since rotation preserves distances, $\|Ru_i - Rv_i\|_2 = \|u_i - v_i\|_2$, and the Frobenius norm $\|R\|_F \leq \sqrt{3}$ implies

$$W^2(D_r, D'_r) \leq \sum_{i=1}^{k} 3\|u_i - v_i\|_2^2 + 2\sum_{i=k+1}^{k+l} \|u_i - \pi(u_i)\|_2^2 + 2\sum_{i=k+1}^{k+m} \|v_i - \pi(v_i)\|_2^2.$$

Thus, we obtain

$$W^2(D_r, D'_r) \leq 3W^2(D, D'),$$

which gives the desired inequality

$$W(D_r, D'_r) \leq \sqrt{3}W(D, D').$$

The same proof holds when $D_{\text{match}} = \varnothing$ if we use only the last two terms in the equation 4.A.1. □

Now, the proof of Proposition 4.3.2 follows directly from Lemmas 4.A.1 and 4.A.2.

Proposition 4.3.2. Let D and D' be two persistence diagrams, and suppose that $\delta < W(D, D')$. Then the following inequality holds:

$$W\Big(\pi_r(D), \pi_r(D')\Big) \leq \left(\frac{\sqrt{|D|} + \sqrt{|D'|}}{\sqrt{2}} + \sqrt{3}\right) W(D, D')$$

Proof. We begin by applying the triangle inequality in the Wasserstein metric space:

$$W\left(\pi_r(D), \pi_r(D')\right) \leq W\left(\pi_r(D), D_r\right) + W\left(D_r, D'_r\right) + W\left(D'_r, \pi_r(D')\right).$$

The terms $W\left(\pi_r(D), D_r\right)$ and $W\left(D'_r, \pi_r(D')\right)$ can be bounded using Lemma 4.A.1, which

gives

$$W\left(\pi_r(D), D_r\right) \leq \sqrt{\frac{|D|}{2}}\delta \quad \text{and} \quad W\left(D'_r, \pi_r(D')\right) \leq \sqrt{\frac{|D'|}{2}}\delta.$$

Next, using Lemma 4.A.2, we bound the middle term:

$$W\left(D_r, D'_r\right) \leq \sqrt{3}W(D, D').$$

Thus, combining these inequalities, we obtain

$$W\left(\pi_r(D), \pi_r(D')\right) \leq \sqrt{\frac{|D|}{2}}\delta + \sqrt{3}W(D, D') + \sqrt{\frac{|D'|}{2}}\delta,$$

which simplifies to

$$W\left(\pi_r(D), \pi_r(D')\right) \leq \left(\frac{\sqrt{|D|} + \sqrt{|D'|}}{\sqrt{2}} + \sqrt{3}\right) W(D, D').$$

This completes the proof. □

4.A.2 Experimental Details

Comparison of Baselines in Graph Classification

Table 4.7 shows the descriptions of the baseline models in graph classification in section 4.5.1. Note that even though the models in Table 4.1 are based on persistent homology related inputs, the specific inputs of each model are different.

Hyperparameters

Table 4.8 and 4.9 show the hyperparameters and the architecture of the xPerT and Persformer used in the experiments. The hyperparameters are chosen based on the performance of the models on the mean accuracy over 10-fold cross validations. The architecture of the xPerT is the same for all experiments, while the Persformer uses different architectures for graph and orbit datasets as in the original paper.

Method	MP	Filtration	Explanation
PersLay ATOL	×	- $HKS_{0.1}$ - $HKS_{10.}$	Each diagram in the extended persistence diagrams is transformed into a vector. Resulting 8 vectors in total are concatenated to generate a representation.
Persformer xPerT	×	- $HKS_{1.0}$	Extended diagram is transformed to a vector.
GRIL	∘	- $HKS_{1.0}$ + $HKS_{10.}$ - Ricci curvature	Persistence landscape is computed based on the generalized rank invariant.
HSM-MP-SW	∘	- $HKS_{10.}$ - Ricci curvature	Sliced Wasserstein kernel is computed based on the signed barcode.

Table 4.7: Summary of topological baselines in graph classification. MP indicates the model is based on multi-parameter persistent homology.

Table 4.8: Hyperparameters for the training of xPerT and Persformer on graph and orbit datasets.

config	**xPerT Graph**	**xPerT Orbit**	**Persformer Graph**	**Persformer Orbit**
optimizer	AdamW	AdamW	AdamW	AdamW
learning rate	1e-3	1e-4	1e-3	1e-3
weight decay	5e-2	5e-2	5e-2	5e-2
batch size	64	64	64	16
lr schedule	cosine decay	cosine decay	cosine decay	cosine decay
warmup epochs	50	50	50	50
epochs	300	300	300	300

Table 4.9: Transformer architecture for xPerT and Persformer used in the experiments. Note that the persformer uses different architectures for graph and orbit datasets as in the original paper.

	xPerT	**Persformer graph**	**Persformer orbit**
# layers	5	2	5
# heads	8	4	8
token dim.	192	32	128

4.B Persistent Homology

This section provides a brief introduction to persistent homology, which is a mathematical tool for studying the topological features of a space. A *filtration* is a key object in construction of persistent homology, which is an increasing sequence $(X)_{t>0}$ of subspaces of a space X. The *persistent homology of dimension* k is a sequence of vector spaces $\{H_k(X_t;\mathbb{F})\}_{t>0}$ and linear maps $\{H_k(X_t;\mathbb{F}) \to H_k(X_{t'};\mathbb{F})\}_{t\leq t'}$, which captures the topological features of the space as the filtration parameter t varies. Here, $\mathbb{F}$ is a field, which is usually taken to be the finite feild $\mathbb{Z}_2$, and H_k denotes the k-th homology group. Monitoring the evolution of homological features via linear maps $\{H_k(X_t;\mathbb{F}) \to H_k(X_{t'};\mathbb{F})\}_{t\leq t'}$ allows associating an interval $[b,d]$, where b and d are birth and death times, respectively. For a detailed introduction to homology and persistent homology, see [17].

Filtration on Graphs Given a graph $G=(V,E)$, lef $f: V \to \mathbb{R}$ be a function defined on the vertices of the graph. A sublevel set filtration $\big(G_t\big)_{t>0}$ is an increasing sequence of subgraphs of G defined as follows. For each $t>0$, the sublevel set G_t is a subgraph of G whose vertices are the vertices in V, and whose edges are the edges in E that connect the vertices whose function values are less than or equal to t. Formally,

$$V(G_t) = \{v \in V \mid f(v) \leq t\}, \quad E(G_t) = \{(u,v) \in E \mid f(u) \leq t, f(v) \leq t\}.$$

In this paper, we use the heat kernel signature (HKS) as the function f, which is a function defined on the vertices of the graph that provides the local geometric information of the graph. The usual sublevel set filtration is used for generate the ordinary persistent

homology.

For extended persistent homology, we use a filtration that combines the sublevel set and superlevel set filtrations. Given a value $t \in \mathbb{R}$, the superlevel set G^t of the graph G is defined as

$$V(G^t) = \{v \in V \mid f(v) \geq t\}, \quad E(G^t) = \{(u, v) \in E \mid f(u) \geq t, f(v) \geq t\}.$$

Without loss of generality, assume that the minimum value of f is 0. Then the extended filtration is given by

$$\tilde{G}_t = \begin{cases} G_t & \text{if } t \leq M, \\ G/G^{2M-t} & \text{otherwise,} \end{cases}$$

where M is the maximum value of the function f. Here G/G^{2M-t} is the quotient graph obtained by contracting the vertices in G^{2M-t} to a single vertex. Note that the extended filtration is not a true filtration, as it does not satisfy the monotonicity condition. Still, we can apply the homology to the sequence of graphs $\left(\tilde{G}_t\right)_{t>0}$ to compute the extended persistence diagram, which reflects the topological features of the graph at different scales.

Filtration on Point Cloud Given a point cloud $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, the *Rips filtration* $\left(R_t(X)\right)_{t>0}$ is an increasing sequence $\left(R_t(X)\right)_{t>0}$ of simplicial complexes defined as follows. For each $t > 0$, the Rips complex $R_t(X)$ is a simplicial complex whose vertices are the points in X, and whose simplices are the subsets of X that are pairwise within distance t. Formally, a simplex $\sigma \subset X$ is in $R_t(X)$ if the diameter of σ is less than or equal to t.

However, the computation of the Rips filtration can be quite costly even in a low-dimensional space. To reduce the computational cost, following [41][6], we use the *weak alpha complex*, which is a Rips complex defined on the Delaunay triangulation of the point cloud.

Persistence Diagram A persistence diagram is a collection of the pairs of birth and death times of topological features obtained from persistent homology. More concretely,

[6]The paper mentions that the authors have used the alpha filtration, but their code uses weak alpha filtration.

the persistence diagram is a multiset of points in the extended plane $\mathbb{R} \times (\mathbb{R} \cup \{\infty\})$, where each point (b, d) represents a topological feature that is born at time b and dies at time d.

4.B.1 Tokenization of Ordinary Persistence Diagrams

Given a point cloud X, let D_k be its k-dimensional persistence diagram, computed using one of the point cloud filtrations (e.g., Rips filtration or weak alpha filtration). For machine learning input, we typically use more than one diagram, $D = \{D_0, \ldots, D_k\}$. As with the extended persistence diagram, we could stack the corresponding PPDs. However, the points in the 0-dimensional persistence diagram D_0 typically lie only along the y-axis, while points from higher-dimensional diagrams are distributed in the 2D plane. This makes it unnatural to apply the channel-stacking tokenization method described in 4.4.1. Instead, we tokenize each diagram separately and combine them into a single sequence. More concretely, let $\{\mathcal{P}_H(D_0), \ldots, \mathcal{P}_H(D_k)\}$ be the PPDs of $D_0, \ldots, D_k$, respectively. Each PPD is treated as a single-channel image, and we apply the same tokenization method to each PPD as described in 4.4.1. Finally, we collect all tokens from each diagram into a single sequence.

Chapter 5

ECG-JEPA

Electrocardiogram (ECG) captures the heart's electrical signals, offering valuable information for diagnosing cardiac conditions. However, the scarcity of labeled data makes it challenging to fully leverage supervised learning in medical domain. Self-supervised learning (SSL) offers a promising solution, enabling models to learn from unlabeled data and uncover meaningful patterns. In this paper, we show that masked modeling in the latent space can be a powerful alternative to existing self-supervised methods in the ECG domain. We introduce ECG-JEPA, a SSL model for 12-lead ECG analysis that learns semantic representations of ECG data by predicting in the hidden latent space, bypassing the need to reconstruct raw signals. This approach offers several advantages in the ECG domain: (1) it avoids producing unnecessary details, such as noise, which is common in ECG; and (2) it addresses the limitations of naïve L2 loss between raw signals. Another key contribution is the introduction of Cross-Pattern Attention (CroPA), a specialized masked attention mechanism tailored for 12-lead ECG data. ECG-JEPA is trained on the union of several open ECG datasets, totaling approximately 180,000 samples, and achieves state-of-the-art performance in various downstream tasks including ECG classification and feature prediction.

5.1 Introduction

Electrocardiography is a non-invasive method to measure the electrical activity of the heart over time, serving as a crucial tool for diagnosing various cardiac conditions. While numerous supervised methods have been developed to detect heart diseases using ECG

data [21, 42, 46], these models often face significant performance degradation when applied to data distributions different from those on which they were trained. This challenge points to the need for more flexible approaches that can learn robust, transferable representations from ECG data.

Self-supervised learning (SSL) offers an alternative approach by learning general representations in diverse domains, such as natural language processing (NLP) [7, 15, 53], computer vision (CV) [2, 10, 22], and video analysis [5, 52]. Despite this promise, the application of SSL to ECG data presents unique challenges. For instance, data augmentation, which is essential in many SSL architectures, is more complex for ECG than for computer vision data. Simple transformations like rotation, scaling, and flipping, effective in CV, can distort the physiological meaning of ECG signals. Additionally, ECG recordings often contain artifacts and noise, which cause autoencoder-based SSL models to struggle with reconstructing raw signals. These architectures may also miss visually subtle but diagnostically critical features, such as P-waves and T-waves, which are imperative for diagnosing certain cardiac conditions.

In this work, we propose ECG Joint-Embedding Predictive Architecture (ECG-JEPA) tailored for 12-lead ECG data, effectively addressing the aforementioned challenges. ECG-JEPA utilizes a transformer architecture to capture the semantic meaning of the ECG. By masking several patches of the ECG, ECG-JEPA predicts abstract representations of the missing segments, indicating a high-level understanding of the data. Additionally, we develop a novel masked-attention for multi-lead ECG data, chich we call Cross-Pattern Attention (CroPA). CroPA incorporates clinical knowledge into the model as an inductive bias, guiding it to focus on clinically relevant patterns and relationships across leads.

Our contributions are as follows:

- ECG-JEPA achieves notable improvements in linear evaluation and fine-tuning on classification tasks compared to existing SSL methods without hand-crafted augmentations.
- CroPA introduces a specialized masked attention mechanism, allowing the model to focus on clinically relevant information in multi-lead ECG data, resulting in improved downstream task performance.
- ECG-JEPA can also recover important ECG features, including heart rate and QRS duration, which are classical indicators used in ECG evaluation. This is the

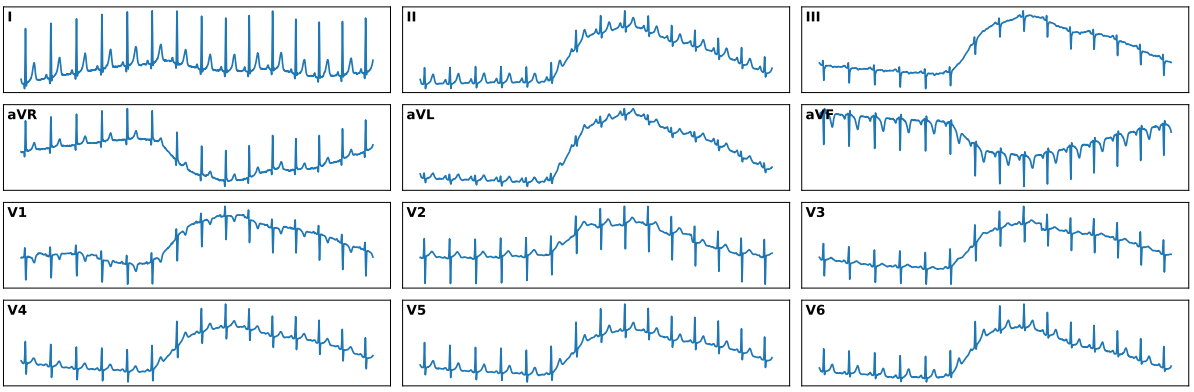


Figure 5.1: 12-lead ECG with baseline wander artifact.

first work to demonstrate that learned representations can effectively recover ECG features.

- ECG-JEPA is highly scalable, allowing efficient training on large datasets. For instance, ECG-JEPA is trained for only 100 epochs, yet outperforms other ECG SSL models on most downstream tasks, taking approximately 22 hours on a single RTX 3090 GPU.

In summary, ECG-JEPA introduces a robust SSL framework for 12-lead ECG analysis, overcoming traditional SSL limitations with clinically inspired design elements, scalable architecture, and demonstrated effectiveness on a wide range of tasks. Our code is available at `https://github.com/sehunfromdaegu/ECG_JEPA`.

5.2 Background

Self-Supervised Learning (SSL) facilitates learning abstract representations from input data without the need for labeled data, which is particularly beneficial in medical domains where labeled data is scarce and expensive. SSL leverages inherent data patterns to learn useful representations, allowing models to adapt to various downstream tasks with greater robustness to data imbalances [32]. We begin in Section 5.2.1 with an overview of the ECG and its key features, highlighting the critical characteristics essential for understanding ECG data. In Sections 5.2.2 and 5.2.3, we briefly explain key SSL techniques and their specific applications to ECG, respectively.

5.2.1 Electrocardiogram (ECG)

The electrocardiogram (ECG) is a non-invasive diagnostic method that records the heart's electrical activity over time using electrodes placed on the skin. The standard 12-lead ECG captures electrical activity of the heart from multiple angles. These 12 leads are categorized into limb leads (I, II, III), augmented limb leads (aVR, aVL, aVF), and chest leads (V1-V6). Each lead provides unique information about the heart's electrical activity, offering a comprehensive view that aids in diagnosing various cardiac conditions. Refer to Figure 5.1 for an illustration.

ECG features are specific characteristics of ECG signals that are critical for summarizing the overall signal. These features play an essential role in monitoring a patient's health status and are instrumental in the application of statistical machine learning models for diagnosing heart diseases. Key ECG features include heart rate, QRS duration, PR interval, QT interval, and ST segment. These features are identified by measuring specific time intervals or amplitude levels in the ECG waveform. For instance, heart rate is calculated using the formula $1000 \times (60/\text{RR interval})$ in beats per minute (bpm), where the RR interval is measured in milliseconds (ms). Refer to Figure 5.2 for a visual representation of these features.

In this work, we use only 8 leads (I, II, V1-V6) as the remaining 4 leads (III, aVR, aVL, aVF) can be derived from linear combinations of the 8 leads following the *Einthoven's law* [50]:

$$\text{III} = \text{II} - \text{I}$$

$$\text{aVR} = -(\text{I} + \text{II})/2$$

$$\text{aVL} = (\text{I} - \text{II})/2$$

$$\text{aVF} = (\text{II} - \text{I})/2.$$

This choice maintains the necessary diagnostic information while optimizing computational efficiency.

5.2.2 Self-Supervised Learning Architectures

Self-supervised learning can be broadly categorized into contrastive and non-contrastive methods. Non-contrastive methods can be further divided into generative and non-generative architectures. For a broader introduction to SSL, see [3].

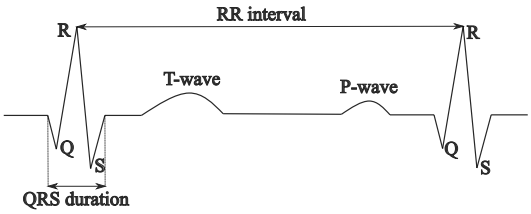


Figure 5.2: Key ECG Features.

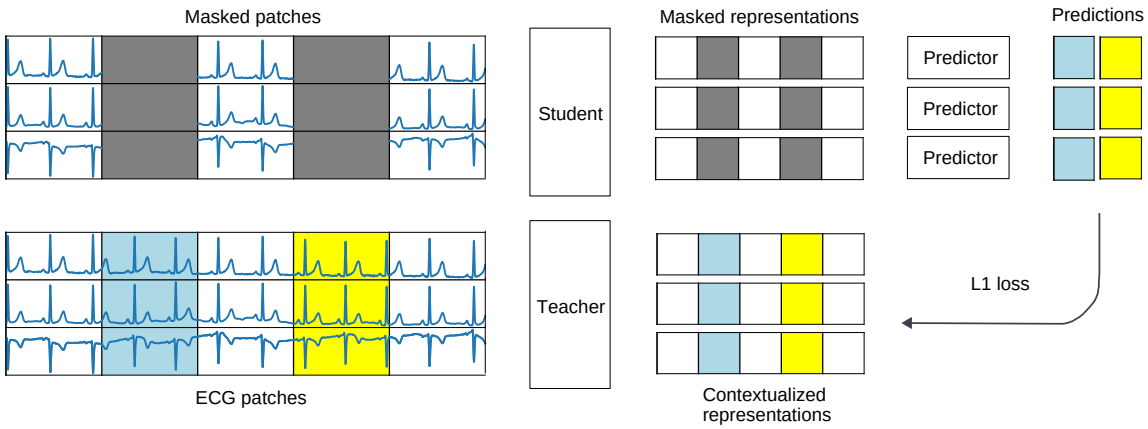


Figure 5.3: ECG-JEPA training overview. For illustration, we use $L = 3$, $N = 5$ subintervals and $Q = 3$ unmasked subintervals.

In *contrastive learning*, the model is encouraged to produce similar representations for semantically related inputs $x^{'}$ and $x^{''}$, while pushing apart the representations of unrelated inputs $x^{'}$ and $y^{'}$. *SimCLR* [10] is one of the most popular contrastive methods, using two different augmentations of a single input x to form semantically similar pairs $x^{'}$ and $x^{''}$.

Beyond contrastive methods, *generative architectures* have been particularly successful in recent large language models [7, 15, 53] and in computer vision [22]. Generative architectures involve reconstructing a sample x from its degraded version x' using an encoder-decoder framework. The premise is that reconstructing clean data from a corrupted version reflects the model's deep understanding of the underlying data structure. The encoder maps the perturbed input x' into a latent representation, which the decoder then uses to reconstruct the original input x [56]. Recently, the authors of [4] observed that generative architectures prioritize learning principal subspaces of the data, which may limit their capacity to capture semantic representations for perceptual tasks.

As an alternative, *non-generative methods* have shown promise across domains, includ-

ing computer vision [2,6,11,20] and video analysis [5]. Among these, the Joint-Embedding Predictive Architecture (JEPA) [30] processes an input pair x and its corrupted versions x' to obtain representations z and z' through encoders. Unlike generative architectures that make predictions in the input space, JEPA performs prediction in the latent space by reconstructing z from z'. This approach effectively avoids the challenge of predicting unpredictable details, a common issue in biological signals.

5.2.3 Related Works

Several studies have worked on capturing semantically meaningful representations of 12-lead ECG data. *Contrastive Multi-segment Coding (CMSC)* [27] splits an ECG into two segments, encouraging similar representations for compatible segments while separating incompatible ones. *Contrastive Predictive Coding (CPC)* [54], applied in [34], predicts future ECG representations in a contrastive manner, but its reliance on LSTM modules makes it inefficient for large datasets. More recently, [60] introduced masked autoencoders for ECG, proposing temporal and channel masking strategies, *Masked Time Autoencoder (MTAE)* and *Masked Lead Autoencoder (MLAE)*. Similarly, [38] proposed *ST-MEM*, which masks random time intervals for each lead. However, both MLAE and ST-MEM may struggle with the high correlations between ECG leads, potentially oversimplifying the prediction task.

5.3 Methodology

ECG-JEPA is trained by predicting masked representations of ECG data in the hidden representation space, using only a partial view of the input. The proposed architecture utilizes a student-teacher framework, as illustrated in Figure 5.3. We subdivide the multi-channel ECG into non-overlapping patches and sample a subset of these patches for masking. However, reconstructing the raw signals of masked patches can be particularly challenging in the ECG domain due to the prevalence of noise in biological signals. Instead, our model predicts the masked patches in the hidden representation space, where this challenge can be effectively addressed. We validate the quality of the learned representations through various downstream tasks, including linear probing, fine-tuning on classification tasks, and ECG feature extraction tasks.

5.3.1 Patch Masking

Let $x \in \mathbb{R}^{L \times T}$ represent a multi-lead ECG of length T with L channels. We divide the interval $[0, T]$ into N non-overlapping subintervals of length t. Each subinterval in each channel constitutes a patch of x, resulting in $L \times N$ patches. The masking strategy in multi-lead ECG must be carefully chosen because patches in different leads at the same temporal position are highly correlated, potentially making the prediction task too easy. To address this, we mask all patches across different leads in the same temporal space. With this in mind, we employ two masking strategies: *random masking* and *multi-block masking*.

In random masking, we randomly select a percentage of subintervals to mask, while in multi-block masking, we select multiple consecutive subintervals to mask. Note that we allow these consecutive subintervals to overlap, which requires the model to predict much longer sequences of representations. In this paper, we use both masking strategies to evaluate the effectiveness of ECG-JEPA, with a random masking ratio of $(0.6, 0.7)$ and a multi-block masking ratio of $(0.175, 0.225)$ with a frequency of 4. The unmasked patches serve as the contextual input for the student networks, while the masked patches are the ones for which we aim to predict the representations.

The patches are converted into sequences of token vectors using a linear layer, and augmented with positional embeddings. We employ the conventional 2-dimensional sinusoidal positional embeddings for the student and teacher networks, while we use 1-dimensional sinusoidal positional embeddings for the predictor network.

5.3.2 Teacher, Student, and Predictor

ECG-JEPA consists of three main components: the teacher network, the student network, and the predictor network. Both the teacher and student networks are based on standard transformer architectures. The weights of the teacher network are updated using an exponential moving average (EMA) of the student network, as detailed in 5.A.2. The predictor network, a smaller transformer, operates on single-channel representations, which still encode information from all leads due to the self-attention mechanism.

The teacher network handles the entire $L \times N$ patches, generating fully contextualized $L \times N$ representations. The student network, however, processes only $L \times Q$ visible (unmasked) patches, where $Q < N$ represents the number of visible time intervals. These $L \times Q$ representations from the student are then concatenated with the (learnable) mask

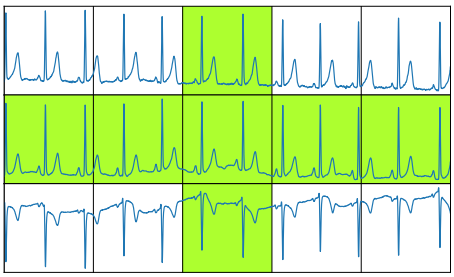


Figure 5.4: Cross-Pattern Attention (CroPA). The patch in the middle attends only to the colored patches.

tokens, resulting in $L \times N$ representations. Subsequently, each lead's representations are passed to the predictor, which processes single-channel representations. The predictor's output, the predicted representations of the target patches, is compared with the target representations using a smooth L1 loss function.

5.3.3 Cross-Pattern Attention (CroPA)

Multi-lead ECG signals require careful analysis of patterns that are often consistent across different leads, which is crucial for identifying potential cardiac abnormalities. This demands attention mechanisms that prioritize relationships within the same lead and within relevant time windows.

To incorporate this structural insight, we introduce Cross-Pattern Attention (CroPA), a masked self-attention mechanism designed for multi-lead ECG data. CroPA imposes an inductive bias by allowing each patch to attend only to patches within the same lead and temporal space (Figure 5.4). This aligns with the way ECG signals are typically interpreted, where intra-lead and temporally adjacent signals hold the most significance.

By incorporating this inductive bias, CroPA helps the model focus on relevant intra-lead relationships, reducing interference from unrelated signals across different channels and time points. Compared to the standard self-attention mechanism, which treat all patches equally, CroPA reflects a structured approach that mirrors the process of multi-lead signal interpretation, leading to improved performance in downstream tasks.

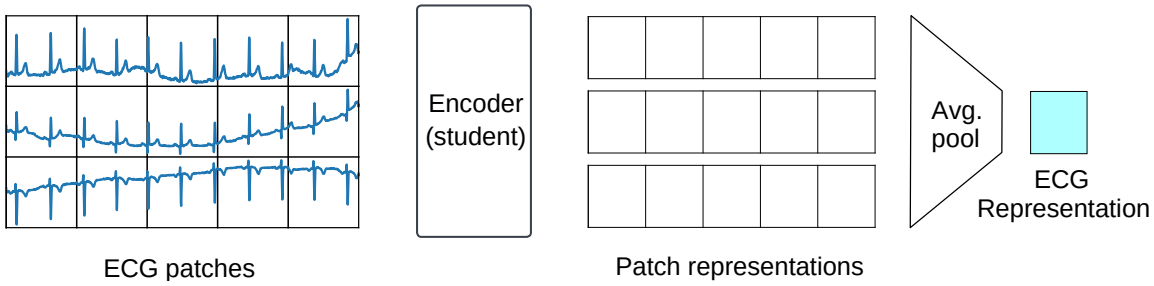


Figure 5.5: Squares following the encoder represent the representations of ECG patches. The representations are subsequently averaged through a pooling layer, with the resulting vector (highlighted in cyan) serving as an abstract representation of the ECG data.

5.3.4 ECG representation

After training, we use only the student network as the encoder. The encoder outputs are average-pooled to obtain the final ECG representation, which serves as the feature vector for downstream tasks. See Figure 5.5 for an illustration.

5.4 Experimental Settings

In all experiments, 10-second multi-lead ECG signals were resampled to 250Hz, yielding $T = 2500$ time points. We divided the interval $[0, T]$ into $N = 50$ non-overlapping subintervals, each of length $t = 50$. The model was trained for 100 epochs without data augmentation, and the final checkpoint was used for downstream tasks. Additional experimental details are provided in Appendix 5.A.1.

5.4.1 Pretraining Datasets

Training SSL models with large datasets is crucial for developing generalized representations. However, most previous works have used relatively small datasets, with the exception of [38], where an SSL model was trained with a large number of 12-lead ECGs. Following [38], we use the *Chapman* [62], *Ningbo* [61], and *CODE-15* [12] datasets for pretraining ECG-JEPA. The Chapman and Ningbo datasets collectively consist of 45,152 10-second 12-lead ECGs at 500Hz. CODE-15 includes 345,779 12-lead ECGs from 233,770 patients at 400Hz, with 143,328 being 10-second recordings. After excluding recordings with missing values, we have 43,240 ECGs from Chapman and Ningbo and 130,900 ECGs

from CODE-15.

5.4.2 Downstream Datasets

We use the *PTB-XL* [57] and *CPSC2018* [31] datasets to evaluate the performance of ECG-JEPA on downstream tasks. *PTB-XL* contains 21,837 clinical 10-second 12-lead ECG records from 18,885 patients, recorded at 500Hz and annotated with 71 diagnostic labels, which are aggregated into five superclasses. We use these superclass labels for our experiments. The *CPSC2018* dataset includes 6,877 12-lead ECG recordings with nine annotated cardiac conditions. These datasets are multi-label in nature, where each recording can have multiple labels simultaneously. The details of the datasets are provided in Appendix 5.A.1.

5.4.3 Architecture

Our model employs transformer encoder architectures for the student, teacher, and predictor networks. Both the teacher and student networks consist of 12 layers with 16 attention heads and a hidden dimension of 768. The predictor network, designed as a smaller transformer encoder, comprises 6 layers with 12 attention heads and a hidden dimension of 384. While the teacher and student networks process the multi-lead ECG data holistically, the predictor operates on each lead independently to reconstruct the masked representations. Importantly, this does not imply that the predictor relies solely on single-lead information for the reconstruction task; due to the self-attention mechanism, the input representations for each lead still encapsulate information from all leads.

5.4.4 Downstream Tasks

We conduct extensive experiments to show that ECG-JEPA effectively captures semantic representations. Its performance is evaluated on classification tasks using linear probing and fine-tuning. Furthermore, we assess its capability in low-shot learning settings, as well as under reduced-lead conditions where the downstream dataset is limited to single or two leads. Reduced-lead configurations are common in clinical practice, especially in scenarios like wearable devices or remote monitoring, where using the full 12-lead ECG setup is impractical.

Table 5.1: Linear evaluation on multi-label and multi-class tasks. Our proposed method outperforms all baselines, achieving the highest AUC and F1 scores across both tasks and datasets.

Method	Epochs	**Multi-label Task**				**Multi-class Task**			
		PTB-XL		CPSC2018		PTB-XL		CPSC2018	
		AUC	F1	AUC	F1	AUC	F1	AUC	F1
ST-MEM	800	0.896	0.662	0.964	0.752	0.888	0.566	<u>0.973</u>	<u>0.805</u>
SimCLR	300	0.866	0.624	0.890	0.523	0.842	0.496	0.918	0.624
CMSC	300	0.802	0.472	0.767	0.206	0.796	0.442	0.787	0.391
CPC	100	0.620	0.167	0.687	0.091	0.600	0.201	0.672	0.210
MoCo v3[1]	800	-	-	-	-	0.739	0.142	0.712	0.080
MTAE[1]	800	-	-	-	-	0.807	0.437	0.818	0.349
MLAE[1]	800	-	-	-	-	0.779	0.382	0.794	0.263
ECG-JEPA$_{rb}$	100	<u>0.906</u>	<u>0.690</u>	<u>0.969</u>	<u>0.769</u>	<u>0.894</u>	<u>0.616</u>	**0.974**	<u>0.805</u>
ECG-JEPA$_{mb}$	100	**0.912**	**0.712**	**0.971**	**0.789**	**0.896**	**0.628**	<u>0.973</u>	**0.819**

[1] Scores reported in [38]; results for multi-label tasks were not available.

To validate the expressiveness of the learned representations, we predict key ECG features such as heart rate and QRS duration. Notably, this work is the first to show that these learned representations can recover a variety of ECG features. The ability to predict these features highlights the informativeness of the representations and their potential to capture clinically relevant characteristics, which is crucial for reliable ECG analysis.

ECG datasets, such as *PTB-XL* and *CPSC2018*, often include multiple simultaneous labels for a single recording, making them multi-label tasks. However, many prior studies have simplified this into a multi-class classification problem by focusing on single-label subsets of the data. To ensure a fair comparison, we pretrain competing methods using publicly available code and evaluate them on the multi-label classification task. In cases where the code is unavailable, we will convert our task into a multi-class problem to align with the reported performance in the literature.

5.5 Experiments

In this section, we evaluate the performance of the learned representations across various downstream tasks to demonstrate their generalizability and ability to capture essential ECG features. ECG-JEPA is compared against several state-of-the-art self-supervised learning (SSL) methods.

For classification tasks, we use AUC (Area Under the ROC Curve) and F1 scores as evaluation metrics. AUC provides a comprehensive measure of discriminative ability by considering performance across all classification thresholds, making it more robust to variations in decision boundaries. In contrast, the F1 score balances precision and recall at a fixed threshold, offering insights into the model's performance when a specific decision boundary is chosen.

In multi-label classification, we compute AUC by averaging the scores from binary classification for each label, while for multi-class classification, AUC is calculated using the one-vs-rest approach. For both tasks, F1 scores are macro-averaged across all classes to ensure equal weighting of each class in the final score.

In most cases, ECG-JEPA consistently outperforms other SSL methods that rely on hand-crafted augmentations, highlighting its effectiveness in learning generalizable representations. In our experiments, ECG-JEPA$_{rb}$ and ECG-JEPA$_{mb}$ refer to ECG-JEPA models trained using random masking and multi-block masking strategies, respectively.

5.5.1 Linear Evaluation

Table 5.1 present the results of our linear evaluation on the *PTB-XL* and *CPSC2018* datasets. We train a linear classifier on top of the frozen representations for 10 epochs and evaluate its performance on downstream tasks. Further training beyond 10 epochs does not lead to any significant improvement in performance. As shown in the tables, ECG-JEPA consistently outperforms other SSL methods, demonstrating superior efficiency and effectiveness with substantially reduced computational resources.

5.5.2 Fine-tuning

Fine-tuning is another method to evaluate the quality of learned representations, as it tests the model's ability to adapt its pre-trained features to new tasks. We add a linear classification head at the end of the encoder and train the entire network for 10 epochs.

Similar to linear evaluation, training for 10 epochs is sufficient, as further training does not lead to additional performance gains. Fine-tuning can potentially enhance performance beyond what is achieved with linear evaluation alone.

Table 5.2 presents the results of fine-tuning on the *PTB-XL* and *CPSC2018* datasets. ECG-JEPA is compared with other SSL methods as well as supervised methods in a multi-class classification setting, where the student network is trained directly from the scratch. The results indicate that ECG-JEPA achieves the highest AUC and F1 scores on *PTB-XL* and the highest AUC on *CPSC2018*.

5.5.3 Low-shot Linear Evaluation

Table 5.3 presents the performance comparison on the low-shot task. Low-shot learning is particularly challenging, as models must generalize effectively with limited labeled data. Given the difficulty and resource-intensive nature of obtaining labeled data in medical research, low-shot learning represents a realistic and critical scenario in the medical field. In this experiment, we evaluate the performance of ECG-SSL models on the *PTB-XL* multi-label task with only 1% and 10% of the training set, while keeping the test set fixed. As shown in the table, ECG-JEPA demonstrates a clear advantage over other SSL methods, with its effectiveness becoming particularly evident in low-shot learning tasks. This suggests that ECG-JEPA can be particularly well-suited for transfer learning where labeled data is scarce.

5.5.4 Reduced Lead Evaluation

Since transformer architectures can handle variable input lengths, we evaluated ECG-JEPA's performance with reduced leads. In this experiment, we conducted a linear evaluation on the *PTB-XL* multi-label task using only a single lead (Lead II) and two leads (Lead II and V1), training linear classifiers on the learned representations for 10 epochs[1]. Table 5.4 presents the results. Notably, ECG-JEPA maintains strong performance even with fewer leads, which is valuable for practical applications in mobile health monitoring, where most devices typically output only one or two leads.

[1]We compare only with ST-MEM, as it is a transformer-based model whose pretrained weights are publicly available.

Table 5.2: **Fine-tuning on multi-class task.**

Method	Epochs	PTB-XL		CPSC2018	
		AUC	F1	AUC	F1
Supervised	100	0.887	0.608	0.893	0.566
MoCo v3[1]	800	0.913	0.644	0.967	**0.838**
MTAE[1]	800	0.910	0.613	0.961	0.769
MLAE[1]	800	0.915	0.625	0.973	0.816
CMSC[1]	800	0.877	0.510	0.938	0.717
ST-MEM	800	0.929	0.668	0.977	0.820
SimCLR	300	0.905	0.650	0.934	0.693
CPC[2]	100	-	-	-	-
ECG-JEPA$_{rb}$	100	**0.944**	**0.710**	0.980	0.821
ECG-JEPA$_{mb}$	100	0.937	0.680	**0.983**	0.799

[1] Scores reported in [38].

[2] We did not fine-tune CPC due to its slow training process.

Table 5.3: Low-shot linear evaluation on the multi-label PTB-XL. The mean and standard deviation of macro AUCs are reported for 1% (192 samples) and 10% (1923 samples) of the training set, selected three times independently.

		PTB-XL	
Method	Epochs	1%	10%
ST-MEM	800	0.807 ± 0.005	0.872 ± 0.001
SimCLR	300	0.803 ± 0.002	0.843 ± 0.001
CMSC	300	0.750 ± 0.008	0.792 ± 0.001
CPC	100	0.523 ± 0.006	0.560 ± 0.005
ECG-JEPA$_{rb}$	100	0.836 ± 0.006	0.887 ± 0.000
ECG-JEPA$_{mb}$	100	**0.843 ± 0.004**	**0.894 ± 0.003**

Table 5.4: Reduced lead evaluation. Linear evaluation of PTB-XL multi-label classification in single-leade (II) and dual-lead (II and V1).

	1-Lead		2-Lead	
Method	AUC	F1	AUC	F1
ST-MEM	0.832	0.571	0.848	0.597
ECG-JEPA$_{rb}$	<u>0.846</u>	**0.596**	<u>0.877</u>	<u>0.647</u>
ECG-JEPA$_{mb}$	**0.849**	0.593	**0.880**	**0.657**

5.5.5 ECG Feature Extraction

Extracting ECG features is crucial for diagnosing and monitoring cardiac conditions. In this experiment, we assess the model's ability to extract key features such as heart rate and QRS duration from the learned representations of the *PTB-XL* dataset. Unlike classification tasks, which focus on perceptual patterns, ECG features are directly tied to the signal's morphology.

Various methods exist for segmenting ECG signals [13, 25, 37, 44], which can be used to extract ECG features. For this experiment, we utilized a publicly available segmentation model [25] to generate ground truth labels for heart rate and QRS duration from the PTB-XL dataset. We then trained a linear regression model on the learned representations to predict these features, using mean squared error (MSE) as the loss function.

Table 5.5 shows the performance comparison, reporting the means and standard deviations of the absolute differences between the predicted and extracted values for the heart rate and QRS duration across the PTB-XL test set.

Interestingly, although the model's representations are designed to capture high-level features, they retain the capacity to recover low-level ECG features. This dual ability to encode both high-level semantics and low-level morphology underscores the versatility of ECG-JEPA, highlighting its potential in both diagnostic and real-world applications.

5.6 Ablation Study

5.6.1 Effect of CroPA

Table 5.6 presents the results of our evaluation of the effectiveness of CroPA. CroPA introduces a "human-like" inductive bias, enabling the model to be trained more efficiently

Table 5.5: ECG feature prediction results on PTB-XL multi-lable test set. The mean heart rate and QRS duration in the test set are 70.01 BPM (±17.65) and 90.48 ms (±17.02), respectively.

	Mean Absolute Error	
Method	Heart Rate (BPM)	QRS Dur. (ms)
ST-MEM	**1.35 ± 2.38**	4.60 ± 4.16
SimCLR	1.87 ± 2.81	6.14 ± 5.80
CMSC	7.20 ± 7.43	10.12 ± 9.98
CPC	11.40 ± 11.04	11.55 ± 11.55
ECG-JEPA$_{rb}$	1.54 ± 2.62	4.81 ± 4.29
ECG-JEPA$_{mb}$	1.45 ± 2.44	**4.41 ± 4.08**

Table 5.6: Effect of CroPA. Linear evaluation (*lin*) and fine-tuning (*ft*) results on PTB-XL multi-class task.

			lin	*ft*
Mask	CroPA	Epochs	AUC	AUC
Random	x	100	0.888	0.930
Random	x	200	0.887	0.927
Random	o	100	**0.894**	**0.944**
Multi-block	x	100	0.872	0.924
Multi-block	x	200	0.886	0.914
Multi-block	o	100	**0.896**	**0.937**

on multi-lead ECG data. Without CroPA, models may require more epochs to converge. For a fair comparison, we trained ECG-JEPA with and without CroPA for 100 and 200 epochs and compared their performance on the PTB-XL multi-class task. The results show that CroPA improves the model's performance, demonstrating its effectiveness in capturing inter-lead relationships and enhancing the model's ability to learn meaningful representations.

5.6.2 Masking Ratio

Table 5.7 presents the performance of ECG-JEPA in linear evaluation with different masking ratios and strategies. The results indicate that the model benefits from a high masking

Table 5.7: Effect of masking strategy. Linear evaluation results on PTB-XL multi-label task using different masking ratios and strategies.

Mask	Ratio	Freq.	AUC	F1
Random	(0.3, 0.4)	1	0.884	0.652
Random	(0.4, 0.5)	1	0.904	0.698
Random	(0.5, 0.6)	1	0.906	0.697
Random	(0.6, 0.7)	1	0.906	0.690
Random	(0.7, 0.8)	1	**0.909**	**0.706**
Multi-block	(0.10, 0.15)	4	0.904	0.678
Multi-block	(0.15, 0.20)	4	0.905	0.687
Multi-block	(0.175, 0.225)	4	**0.912**	**0.712**

Table 5.8: Comparison of 8-Lead and 12-Lead Models on PTB-XL multi-label.

Model	epochs	AUC	F1
8-Lead	100	0.906	0.690
12-Lead	100	0.905	0.699

ratio. Notably, multi-block masking is advantageous for linear evaluation, while random masking is more effective for fine-tuning, as indicated in Table 5.2. Although random masking with a ratio of (0.7, 0.8) achieves better performance in the PTB-XL multi-label task, a masking ratio of (0.6, 0.7) performs better in other tasks. Therefore, we chose the latter for our main experiments.

5.6.3 Comparison with 12-Lead Model

We now investigate the practical sufficiency of using 8 leads for ECG-JEPA pretraining. To evaluate the impact of this reduction, we trained models using both 8 leads and 12 leads and compared their performance on the linear evaluation of a multi-label task for PTB-XL.

Table 5.8 presents the results of this comparison using ECG-JEPA$_{rb}$. As expected, the performance difference between the 8-lead and 12-lead models is minimal, indicating that using 8 leads is sufficient for effective pretraining without significant loss of information.

5.7 Discussion

5.7.1 Insights and Interpretations

The results demonstrate that ECG-JEPA effectively captures high-quality representations from 12-lead ECG signals, as evidenced by its superior performance across various downstream tasks, including classification, low-shot learning, and feature extraction. The model's ability to maintain robust performance under reduced lead configurations underscores its practical applicability in resource-constrained scenarios, such as wearable devices and remote health monitoring.

Moreover, the proposed Cross-Pattern Attention (CroPA) mechanism introduces a clinically inspired inductive bias, aligning with the physiological patterns of multi-lead ECG signals. This targeted attention contributes to enhanced model performance, particularly in tasks requiring inter-lead correlations. The findings validate the importance of incorporating domain-specific design elements into self-supervised learning frameworks for medical data.

Compared to previous SSL approaches, ECG-JEPA offers significant advancements. While several methods rely on extensive augmentations or manual feature engineering, ECG-JEPA bypasses these requirements by learning semantic representations directly in the latent space.

To the best of our knowledge, we are the first to demonstrate that ECG representations learned through self-supervised learning can successfully recover key ECG features such as heart rate and QRS duration. This finding highlights the dual capability of ECG-JEPA to encode both high-level semantic information and low-level morphological details, making it versatile for various diagnostic and monitoring tasks. These results pave the way for further exploration of self-supervised learning methods in uncovering clinically meaningful patterns in physiological signals.

5.7.2 Limitations and Challenges

While ECG-JEPA achieves state-of-the-art performance, certain limitations remain. One notable limitation is the lack of inherent explainability in the model's learned representations. Although ECG-JEPA effectively captures semantic and morphological features, it provides limited insights into how these features are utilized for specific predictions, which can be crucial in medical applications. The absence of a clear interpretability mechanism

may hinder its adoption in clinical settings, where understanding the decision-making process is often as important as the results themselves.

5.7.3 Broader Implications

The implications of this work extend beyond ECG analysis. The principles underlying ECG-JEPA, particularly the combination of latent-space prediction and domain-specific attention mechanisms, could inspire advancements in other multivariate physiological signal domains, such as EEG and EMG. By leveraging these principles, researchers could develop models capable of extracting meaningful representations from diverse biomedical data, potentially accelerating progress in multimodal diagnostic systems.

5.7.4 Future Directions

Looking ahead, integrating ECG-JEPA with complementary diagnostic modalities, such as chest X-rays or echocardiograms, could provide a more holistic understanding of cardiac health. This multi-modal approach has the potential to improve diagnostic accuracy by leveraging the strengths of different data types, enabling a richer representation of patient conditions.

One significant challenge in pursuing these extensions is the scarcity of large-scale datasets in other modalities. Addressing this limitation is crucial for advancing the multi-model foundation model.

5.8 Conclusion

We proposed ECG-JEPA, a novel SSL method tailored for 12-lead ECG data. By utilizing a JEPA coupled with the innovative relative positional encoding method, CroPA, ECG-JEPA effectively learns meaningful representations of ECG signals. This approach addresses the challenges posed by noise and artifacts in ECG data, demonstrating substantial improvements over existing SSL methods in various downstream tasks, with the added benefit of significantly faster convergence.

Our extensive experimental evaluations reveal that ECG-JEPA outperforms state-of-the-art SSL methods across several tasks, including linear evaluation, fine-tuning, low-shot learning, and ECG feature extraction. Moreover, our investigation into the use of 8

leads, as opposed to the full 12-lead ECG, indicates that this reduction does not compromise performance while optimizing computational efficiency. This finding is particularly significant for applications constrained by limited computational resources.

5.A Supplementary Materials

5.A.1 Experimental Details

Downstream Datasets Details

Table 5.9, and 5.10 show the distribution of the PTB-XL and CPSC2018 datasets, respectively. Note that the sum of samples in each class exceeds the total number of ECG recordings in multi-label task.

The PTB-XL dataset is stratified into ten folds, where the first eight folds are used for training, the ninth fold for validation, and the tenth fold for testing. In our experiments, we used the first nine folds for training and the tenth fold for testing, as we did not observe overfitting during linear evaluation and fine-tuning.

For the CPSC2018 dataset, only the training set is publicly available, which is stratified into seven folds. We used the first six folds for training and the seventh fold for testing, omitting the validation set. The original CPSC2018 dataset consists of 6,877 ECG recordings, but we excluded recordings with a length of less than 10 seconds, resulting in 6,867 ECG recordings.

Table 5.9: PTB-XL Distribution.

Type	Set	# ECG	Norm	MI	STTC	CD	HYP
Multi-label	Total	21799	9514	5469	5235	4898	2649
	Train	19230	8551	4919	4714	4402	2387
	Test	2158	963	550	521	496	262
Multi-class	Total	16244	9069	2532	2400	1708	535
	Train	14594	8157	2276	2158	1524	479
	Test	1650	912	256	242	184	56

Table 5.10: CPSC2018 Distribution.

Type	Set	# ECG	Norm	PVC	AF	LBBB	STE	1AVB	PAC	STD	RBBB
Multi-label	Total	6867	918	1220	235	220	721	614	699	868	1854
	Train	5989	805	1059	206	197	632	534	615	742	1616
	Test	878	113	161	29	23	89	80	84	126	238
Multi-class	Total	6391	918	975	178	185	685	531	606	783	1530
	Train	5577	805	849	159	169	600	459	534	671	1331
	Test	814	113	126	19	16	85	72	72	112	199

Hyperparameters for ECG-JEPA

Hyperparameters for ECG-JEPA pretraining, linear evaluation, and fine-tuning are provided in Tables 5.11, 5.12, and 5.13, respectively. In ECG-JEPA$_{mb}$, the number of visible patches in ECG-JEPA$_{mb}$ varies more than in ECG-JEPA$_{rb}$, resulting in higher GPU memory usage. Consequently, we reduced the batch size to 64 to fit the model on a single NVIDIA RTX 3090 GPU. Interestingly, ECG-JEPA$_{mb}$ benefits from larger learning rates, even with the halved batch size.

For fine-tuning process, the actual learning rate is calculated as

$$lr = base\_lr \times batchsize/256,$$

following the heuristic by [19].

Table 5.11: Pretraining Settings for ECG-JEPA.

config	ECG-JEPA$_{\text{rb}}$	ECG-JEPA$_{\text{mb}}$
optimizer	AdamW	AdamW
learning rate	2.5e-5	5e-5
weight decay	0.05	0.05
batch size	128	64
learning rate schedule	cosine decay	cosine decay
warmup epochs	5	5
epochs	100	100
drop path	0.1	0.1

Table 5.12: Linear Evaluation Settings

config	value
optimizer	AdamW
learning rate	5e-4
weight decay	0.05
batch size	32
learning rate schedule	cosine decay
warmup epochs	3
epochs	10

Table 5.13: Fine-tuning Settings.

config	value
optimizer	AdamW
base learning rate	1.0e-4
weight decay	0.05
batch size	16
learning rate schedule	cosine decay
warmup epochs	3
epochs	10

Hyperparameters for Other Pretrained Models

Besides pretraining ECG-JEPA, we also pretrained other models, including CMSC [27], CPC [54], and SimCLR [10] using the same datasets as ECG-JEPA.

For CMSC and CPC, we adhered to the original architecture and hyperparameters. SimCLR utilized a ResNet50 [23] encoder with an output dimension of 2048. CMSC and SimCLR were pretrained for 300 epochs, selecting the best checkpoint at 100, 200, or 300 epochs based on linear evaluation performance on the PTB-XL multi-label setting. Due to the slow training process, CPC was pretrained for only 100 epochs, taking approximately 9 days on a single NVIDIA RTX 3090 GPU due to the LSTM module in the model. For ST-MEM [38], we employed the publicly available checkpoint pretrained for 800 epochs.

Given SimCLR's sensitivity to data augmentations, we applied several that work well empirically: baseline shift (adding a constant to all leads), baseline wander (low-frequency noise), Gaussian noise (random noise), powerline noise (50Hz noise), channel resize, ran-

dom crop, and jump noise (sudden jumps). These augmentations aimed to enhance the robustness of the model to various signal distortions.

5.A.2 Exponential Moving Average

The teacher network is initialized as a copy of the student network and is updated using an exponential moving average (EMA) of the student's weights. The EMA is computed as follows:

$$\theta^i_{\text{teacher}} = \beta_i \theta^{i-1}_{\text{teacher}} + (1 - \beta_i)\theta^i_{\text{student}}$$

where i denotes the current training iteration, and β_i is a momentum parameter that evolves during training. The momentum parameter β_i is computed as:

$$\beta_i = \text{ema}_0 + \frac{i \cdot (\text{ema}_1 - \text{ema}_0)}{\text{iterations\_per\_epoch} \cdot \text{epochs}}$$

Here, ema_0 and ema_1 represent the initial and final values of the momentum parameter, respectively. For our implementation, $\text{ema}_0 = 0.996$ and $\text{ema}_1 = 1.0$.

Bibliography

[1] 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** (2017), no. 8, 1–35.

[2] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas, *Self-supervised learning from images with a joint-embedding predictive architecture*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2023, pp. 15619–15629.

[3] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum, *A cookbook of self-supervised learning*, 2023.

[4] Randall Balestriero and Yann LeCun, *Learning by reconstruction produces uninformative features for perception*, 2024.

[5] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas, *Revisiting feature prediction for learning visual representations from video*, 2024.

[6] Adrien Bardes, Jean Ponce, and Yann LeCun, *Vicreg: Variance-invariance-covariance regularization for self-supervised learning*, 2022.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., *Language models are few-shot learners*, Advances in neural information processing systems **33** (2020), 1877–1901.

[8] Peter Bubenik et al., *Statistical topological data analysis using persistence landscapes.*, J. Mach. Learn. Res. **16** (2015), no. 1, 77–102.

[9] 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*, International conference on artificial intelligence and statistics, 2020, pp. 2786–2796.

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton, *A simple framework for contrastive learning of visual representations*, International conference on machine learning, 2020, pp. 1597–1607.

[11] Xinlei Chen and Kaiming He, *Exploring simple siamese representation learning*, 2020.

[12] Yu-Jhen Chen, Chien-Liang Liu, Vincent S Tseng, Yu-Feng Hu, and Shih-Ann Chen, *Large-scale classification of 12-lead ecg with deep learning*, 2019 ieee embs international conference on biomedical & health informatics (bhi), 2019, pp. 1–4.

[13] Zhenqin Chen, Mengying Wang, Meiyu Zhang, Wei Huang, Hanjie Gu, and Jinshan Xu, *Post-processing refined ecg delineation based on 1d-unet*, Biomedical Signal Processing and Control **79** (2023), 104106.

[14] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer, *Extending persistence using poincaré and lefschetz duality*, Foundations of Computational Mathematics **9** (2009), no. 1, 79–103.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021.

[17] Herbert Edelsbrunner and John L Harer, *Computational topology: an introduction*, American Mathematical Society, 2022.

[18] Cong Fang, Hangfeng He, Qi Long, and Weijie J Su, *Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training*, Proceedings of the National Academy of Sciences **118** (2021), no. 43, e2103091118.

[19] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He, *Accurate, large minibatch sgd: Training imagenet in 1 hour*, 2018.

[20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al., *Bootstrap your own latent-a new approach to self-supervised learning*, Advances in neural information processing systems **33** (2020), 21271–21284.

[21] Awni Y Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H Tison, Codie Bourn, Mintu P Turakhia, and Andrew Y Ng, *Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network*, Nature medicine **25** (2019), no. 1, 65–69.

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick, *Masked autoencoders are scalable vision learners*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2022, pp. 16000–16009.

BIBLIOGRAPHY

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, Proceedings of the ieee conference on computer vision and pattern recognition, 2016, pp. 770–778.

[24] Like Hui, Mikhail Belkin, and Preetum Nakkiran, *Limitations of neural collapse for understanding generalization in deep learning*, 2022.

[25] Chankyu Joung, Mijin Kim, Taejin Paik, Seong-Ho Kong, Seung-Young Oh, Won Kyeong Jeon, Jaehu Jeon, Joong-Sik Hong, Wan-Joong Kim, Woong Kook, et al., *Deep learning based ecg segmentation for delineation of diverse arrhythmias*, PloS one **19** (2024), no. 6, e0303178.

[26] 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** (2020), 15965–15977.

[27] Dani Kiyasseh, Tingting Zhu, and David A Clifton, *Clocs: Contrastive learning of cardiac signals across space, time, and patients*, International conference on machine learning, 2021, pp. 5606–5615.

[28] Vignesh Kothapalli, Tom Tirer, and Joan Bruna, *A neural collapse perspective on feature evolution in graph neural networks*, Advances in Neural Information Processing Systems **36** (2024).

[29] Aditi S Krishnapriyan, Joseph Montoya, Maciej Haranczyk, Jens Hummelshøj, and Dmitriy Morozov, *Machine learning with persistent homology and chemical word embeddings improves prediction accuracy and interpretability in metal-organic frameworks*, Scientific reports **11** (2021), no. 1, 8888.

[30] Yann LeCun, *A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27*, Vol. 62, 2022. Accessed: 2024-06-01.

[31] Feifei Liu, Chengyu Liu, Lina Zhao, Xiangyu Zhang, Xiaoling Wu, Xiaoyan Xu, Yulin Liu, Caiyun Ma, Shoushui Wei, Zhiqiang He, et al., *An open access database for evaluating the algorithms of electrocardiogram rhythm and morphology abnormality detection*, Journal of Medical Imaging and Health Informatics **8** (2018), no. 7, 1368–1373.

[32] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma, *Self-supervised learning is more robust to dataset imbalance*, 2022.

[33] David Loiseaux, Luis Scoccola, Mathieu Carrière, Magnus Bakke Botnan, and Steve Oudot, *Stable vectorization of multiparameter persistent homology using signed barcodes as measures*, Advances in Neural Information Processing Systems **36** (2024).

[34] Temesgen Mehari and Nils Strodthoff, *Self-supervised representation learning from 12-lead ecg data*, Computers in biology and medicine **141** (2022), 105114.

[35] Zhenyu Meng, D Vijay Anand, Yunpeng Lu, Jie Wu, and Kelin Xia, *Weighted persistent homology for biomolecular data analysis*, Scientific reports **10** (2020), no. 1, 2079.

[36] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann, *Tudataset: A collection of benchmark datasets for learning with graphs*, arXiv preprint arXiv:2007.08663 (2020).

BIBLIOGRAPHY

[37] Viktor Moskalenko, Nikolai Zolotykh, and Grigory Osipov, *Deep learning for ecg segmentation*, Advances in neural computation, machine learning, and cognitive research iii: Selected papers from the xxi international conference on neuroinformatics, october 7-11, 2019, dolgoprudny, moscow region, russia, 2020, pp. 246–254.

[38] Yeongyeon Na, Minje Park, Yunwon Tae, and Sunghoon Joo, *Guiding masked representation learning to capture spatio-temporal relationship of electrocardiogram*, 2024.

[39] Ippei Obayashi, Takenobu Nakamura, and Yasuaki Hiraoka, *Persistent homology analysis for materials research and persistent homology software: Homcloud*, journal of the physical society of japan **91** (2022), no. 9, 091013.

[40] Vardan Papyan, XY Han, and David L Donoho, *Prevalence of neural collapse during the terminal phase of deep learning training*, Proceedings of the National Academy of Sciences **117** (2020), no. 40, 24652–24663.

[41] Raphael Reinauer, Matteo Caorsi, and Nicolas Berkouk, *Persformer: A transformer architecture for topological machine learning*, 2022.

[42] Antonio H. Ribeiro, Manoel Horta Ribeiro, Gabriela M. M. Paixao, Derick M. Oliveira, Paulo R. Gomes, Jessica A. Canazart, Milton P. S. Ferreira, Carl R. Andersson, Peter W. Macfarlane, Meira Wagner Jr., Thomas B. Schon, and Antonio Luiz P. Ribeiro, *Automatic diagnosis of the 12-lead ecg using a deep neural network*, Nature communications **11** (2020), no. 1, 1760.

[43] Martin Royer, Frédéric Chazal, Clément Levrard, Yuhei Umeda, and Yuichi Ike, *Atol: measure vectorization for automatic topologically-oriented learning*, International conference on artificial intelligence and statistics, 2021, pp. 1000–1008.

[44] Iana Sereda, Sergey Alekseev, Aleksandra Koneva, Roman Kataev, and Grigory Osipov, *Ecg segmentation by neural networks: Errors and correction*, 2019 international joint conference on neural networks (ijcnn), 2019, pp. 1–7.

[45] Yashbir Singh, Colleen M Farrelly, Quincy A Hathaway, Tim Leiner, Jaidip Jagtap, Gunnar E Carlsson, and Bradley J Erickson, *Topological data analysis in medical imaging: current state of the art*, Insights into Imaging **14** (2023), no. 1, 58.

[46] Konstantinos C Siontis, Peter A Noseworthy, Zachi I Attia, and Paul A Friedman, *Artificial intelligence-enhanced electrocardiography in cardiovascular disease management*, Nature Reviews Cardiology **18** (2021), no. 7, 465–478.

[47] Kristian Strommen, Matthew Chantry, Joshua Dorrington, and Nina Otter, *A topological perspective on weather regimes*, Climate Dynamics **60** (2023), no. 5, 1415–1445.

[48] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas, *A concise and provably informative multi-scale signature based on heat diffusion*, Computer graphics forum, 2009, pp. 1383–1392.

[49] Anna Suzuki, Miyuki Miyazawa, James M Minto, Takeshi Tsuji, Ippei Obayashi, Yasuaki Hiraoka, and Takatoshi Ito, *Flow estimation solely from image data through persistent homology analysis*, Scientific reports **11** (2021), no. 1, 17948.

[50] Malcolm S Thaler, *The only ekg book you'll ever need*, Lippincott Williams & Wilkins, 2021.

[51] Tom Tirer, Haoxiang Huang, and Jonathan Niles-Weed, *Perturbation analysis of neural collapse*, Proceedings of the 40th international conference on machine learning, 202323, pp. 34301–34329.

[52] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang, *Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training*, Advances in neural information processing systems **35** (2022), 10078–10093.

[53] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample, *Llama: Open and efficient foundation language models*, 2023.

[54] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, *Representation learning with contrastive predictive coding*, 2019.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems, 2017, pp. 5998–6008.

[56] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, *Extracting and composing robust features with denoising autoencoders*, Proceedings of the 25th international conference on machine learning, 2008, pp. 1096–1103.

[57] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek, and Tobias Schaeffter, *Ptb-xl, a large publicly available electrocardiography dataset*, Scientific data **7** (2020), no. 1, 1–15.

[58] Kelin Xia and Guo-Wei Wei, *Persistent homology analysis of protein structure, flexibility, and folding*, International journal for numerical methods in biomedical engineering **30** (2014), no. 8, 814–844.

[59] Cheng Xin, Soham Mukherjee, Shreyas N. Samaga, and Tamal K. Dey, *Gril: A 2-parameter persistence based vectorization for machine learning*, Proceedings of 2nd annual workshop on topology, algebra, and geometry in machine learning (tag-ml), 202328 Jul, pp. 313–333.

[60] Huaicheng Zhang, Wenhan Liu, Jiguang Shi, Sheng Chang, Hao Wang, Jin He, and Qijun Huang, *Maefe: Masked autoencoders family of electrocardiogram for self-supervised pretraining and transfer learning*, IEEE Transactions on Instrumentation and Measurement **72** (2022), 1–15.

[61] Jianwei Zheng, Huimin Chu, Daniele Struppa, Jianming Zhang, Sir Magdi Yacoub, Hesham El-Askary, Anthony Chang, Louis Ehwerhemuepha, Islam Abudayyeh, Alexander Barrett, et al., *Optimal multi-stage arrhythmia classification approach*, Scientific reports **10** (2020), no. 1, 2898.

[62] Jianwei Zheng, Jianming Zhang, Sidy Danioko, Hai Yao, Hangyuan Guo, and Cyril Rakovski, *A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients*, Scientific data **7** (2020), no. 1, 48.

국문초록

주요어휘: 데이터 분석, 위상학적 데이터 분석, 지속 호몰로지, 딥러닝, 자기지도학습, 신경붕괴, 심전도
학번: 2018-23771