硕士学位论文

Transformer 在拓扑深度学习中的使用

THE APPLICATION OF TRANSFORMER IN TOPOLOGICAL DEEP LEARNING

研　　究　　生：丁泽洋
指　导　教　师：朱一飞

南方科技大学

二〇二五年六月

国内图书分类号：515.1　　学校代码：14325
国际图书分类号：O29　　密级：公开

理学硕士学位论文

Transformer 在拓扑深度学习中的使用

学 位 申 请 人：丁泽洋
指 导 教 师：朱一飞
学 科 名 称：数学
答 辩 日 期：2025 年 5 月
培 养 单 位：数学系
学位授予单位：南方科技大学

THE APPLICATION OF TRANSFORMER IN TOPOLOGICAL DEEP LEARNING

A dissertation submitted to
Southern University of Science and Technology
in partial fulfillment of the requirement
for the degree of
Master of Science
in
Mathematics

by
Ding Zeyang

Supervisor: Prof. Yifei Zhu

May, 2025

学位论文公开评阅人和答辩委员会名单

公开评阅人名单

匿名评阅

答辩委员会名单

主席	严质彬	教授	哈尔滨工业大学（深圳）
委员	朱一飞	助理教授	南方科技大学
	于天舒	助理教授	香港中文大学（深圳）
秘书	刘鲁一	教辅序列	南方科技大学

南方科技大学学位论文原创性声明和使用授权说明

南方科技大学学位论文原创性声明

本人郑重声明：所提交的学位论文是本人在导师指导下独立进行研究工作所取得的成果。除了特别加以标注和致谢的内容外，论文中不包含他人已发表或撰写过的研究成果。对本人的研究做出重要贡献的个人和集体，均已在文中作了明确的说明。本声明的法律结果由本人承担。

作者签名：　　日期：

南方科技大学学位论文使用授权书

本人完全了解南方科技大学有关收集、保留、使用学位论文的规定，即：

1. 按学校规定提交学位论文的电子版本。

2. 学校有权保留并向国家有关部门或机构送交学位论文的电子版，允许论文被查阅。

3. 在以教学与科研服务为目的前提下，学校可以将学位论文的全部或部分内容存储在有关数据库提供检索，并可采用数字化、云存储或其他存储手段保存本学位论文。

（1）在本论文提交当年，同意在校园网内提供查询及前十六页浏览服务。

（2）在本论文提交 □ 当年/□ __ 年以后，同意向全社会公开论文全文的在线浏览和下载。

4. 保密的学位论文在解密后适用本授权书。

作者签名：　　日期：

指导教师签名：　　日期：

摘 要

拓扑数据分析（Topological Data Analysis, TDA）自 21 世纪初在持续同调理论突破性进展的推动下，经过二十余年的发展，已成长为应用拓扑学中具有严格数学基础的研究体系，并在生物医学成像、材料微结构分析、复杂网络建模等跨学科领域展现出独特优势。当前该领域面临的核心挑战之一是如何从持续图（Persistence Diagrams）中提取适合深度学习架构的特征表示——作为刻画数据多尺度拓扑特征的代数结构，持续图本质上是 $\mathbb{R}^2$ 空间中的多重点集，其非规则几何特性导致其无法直接作为向量输入神经网络。针对这一关键问题，学界主要发展出两大向量化范式：基于拓扑特征预定义的“预定向量化方法”和结合神经网络参数优化的“可学习向量化方法”。拓扑数据分析领域一般将这类面向深度学习的拓扑特征提取研究统称为拓扑深度学习。

过去十年间，以自注意力机制为核心的 Transformer 架构彻底改变了深度学习的范式。该架构通过动态建立数据元素间的全局依赖关系，摒弃传统递归结构的同时实现了高效的并行计算，在自然语言处理、计算机视觉等领域持续刷新性能纪录。

在本篇文章中，我们介绍了拓扑数据分析的背景知识，从持续模这一代数对象，逐步过渡到由数据包含的拓扑信息汇总而成的持续图，并给出了持续图上的常用度量。我们列举了当前拓扑深度学习领域一些重要的工作，以帮助读者能够对该领域有一个基本了解。我们介绍了当前拓扑深度学习的预定向量化方法 Persistence Landscapes 和 Persistence Landscapes 相关的一些理论工作，以及最新的可学习向量化方法 PLLay 和 PersLay 以及相关的一些理论结果。

本文的创新之处体现在使用 Transformer 这一强大的深度学习架构对持续图进行向量化，并且取得了不错的效果，在经典的合成数据集和图数据集上，显著优于先前的拓扑深度学习架构。此外，我们证明了使用 Transformer 架构对持续图进行向量化满足一个通用逼近定理，这说明 Transformer 在处理持续图的过程中，保持了鲁棒性，这是使用拓扑方法进行数据分析的主要优点。最后，我们使用显著性图对使用 Transformer 向量化持续图这一过程进行了分析，并得到了一些结论，这是在拓扑数据分析领域进行神经网络可解释性的一次成功尝试。

关键词：拓扑数据分析；Transformer；持续图；拓扑深度学习

Abstract

Topological Data Analysis (TDA) has developed into a research system with a rigorous mathematical foundation in applied topology, driven by breakthrough advancements in persistent homology theory since the early 21st century. Over the past two decades, it has demonstrated unique advantages in interdisciplinary fields such as biomedical imaging, materials microstructure analysis, and complex network modeling. One of the core challenges currently faced by this field is extracting feature representations from persistence diagrams suitable for deep learning architectures. As an algebraic structure that characterizes the multi-scale topological features of data, persistence diagrams are essentially multi-point sets in $\mathbb{R}^2$ space, and their irregular geometric properties make them unsuitable for direct input into neural networks. The academic community has primarily developed two major vectorization paradigms to address this key issue: "predefined vectorization methods" based on topological features and "learnable vectorization methods" that combine neural network parameter optimization. We generally refer to research in topological feature extraction for deep learning in TDA as Topological deep Learning.

Over the past decade, the Transformer architecture, depending on the self-attention mechanism, has completely transformed the deep learning paradigm. This architecture dynamically establishes global dependencies among data elements, abandons traditional recursive structures, and achieves efficient parallel computation, continually setting new performance records in fields such as natural language processing and computer vision.

In this paper, we introduce the background of Topological Data Analysis, transitioning from the algebraic object of persistence modules to persistence diagrams, which summarize the topological information in the data and provide standard metrics on persistence diagrams. We list essential works in the field of topological deep learning to help the reader gain a basic understanding of this area. We introduce the current predefined vectorization method Persistence Landscapes and its related theoretical work, the latest learnable vectorization methods PLLay and PersLay, and some related theoretical results.

This paper's novelty lies in using the powerful deep learning architecture Transformer to vectorize persistence diagrams, achieving good results that significantly outperform previous topological deep learning architectures on classical synthetic and graph datasets. Furthermore, we prove that the vectorization of persistence diagrams using the

Transformer architecture satisfies a universal approximation theorem, indicating that the Transformer maintains robustness when processing persistence diagrams, which is a key advantage of using topological methods for data analysis. Finally, we analyze the process of using Transformer to vectorize persistence diagrams with Saliency Maps and draw some conclusions, representing a successful attempt at neural network interpretability in Topological Data Analysis.

Keywords: Topological Data Analysis; Transformer; Persistence Diagram; Topological Deep Learning

目 录

第 1 章　引言

拓扑数据分析（Topological Data Analysis, 以下简称 TDA）是数据科学中一个快速发展的领域，它将估计数据集拓扑结构的方法纳入深度学习。该领域中最常见的描述符是所谓的持续图，这些持续图进一步用于拓扑数据分析中的分类和回归任务；其应用范围广泛，涉及材料科学[31,35]、动力系统[46]、神经科学[40]、癌症生物学[3,25]、深度学习架构的理解[28,34]以及 COVID-19 传播的分析[19]。

拓扑深度学习则是拓扑数据分析背景下，考虑如何从持续图中提取可直接用于深度学习的特征的领域，关于该领域的发展，读者可以参考[48]。

持续图是 $\mathbb{R}^2$ 的子集，其点对应于数据集中的拓扑特征（如连通分支、循环、空洞等），坐标编码特征的大小概念。它们通常使用 Bottleneck 距离或 Wasserstein 距离进行比较。

当前，将持续图纳入深度学习的主要挑战有两个。首先，持续图的底层数据结构本质上是一个集合。因此，学习到的集合表示应当对点呈现顺序的不变性。其次，无论考虑哪种类型的距离（Bottleneck 或 Wasserstein），持续图空间不能等距嵌入希尔伯特空间。这是一个最重要的挑战，因为大多数深度学习算法设计用于希尔伯特空间中的向量操作。

为了克服上述问题，拓扑数据分析研究人员开发了几种向量化方法，以便将持续图集合与希尔伯特空间中的向量集合关联起来。这些方法主要有两种类型。一种是定义向量化映射，另一个是通过可训练的架构（如神经网络）学习它。

在过去十年中，神经网络架构中的一个主要变化是 Transformer 架构的引入，其构建包含自注意机制。简而言之，Transformer 模型将数据整体处理，利用数据元素之间的长距离关系，并完全避免递归。Transformer 架构的这些优势使得它在各种任务上实现了最先进的性能。在自然语言处理中，大型预训练语言模型如 BERT 和 GPT-4 在各种自然语言处理基准测试中取得了最先进的结果。在计算机视觉任务中，Vision Transformer 模型在包括 ImageNet 分类基准在内的几个基准测试中实现了最先进的结果。

在本篇文章中，我将持续图直接作为 Transformer 架构的输入，使 Transformer 的强大功能和多样性在拓扑深度学习中得以使用。在具体实验中，我会将上述持续图直接作为 Transformer 输入的架构与已经存在的用于处理持续图的神经网络架构（PersLay 和 PLLay）进行比较，最终的结果表明应用到拓扑深度学习中的

Transformer 网络在本文所使用的数据集上效果构会优于 PersLay 和 PLLay。

第 2 章　相关工作

正如引言中所述，拓扑数据分析的主要挑战之一在于，带有 Bottleneck 距离或 Wasserstein 距离的持续图空间不能等距嵌入希尔伯特空间。由于大多数深度学习方法假设输入数据集是希尔伯特空间的子集，因此它们不能直接应用于持续图的数据集。为了克服这个问题，拓扑数据分析研究人员做出了大量努力，定义了持续图空间的向量化，即定义一个希尔伯特空间 $\mathcal{H}$ 以及一个连续映射 $\phi : \mathcal{D} \to \mathcal{H}$，其中 $\mathcal{D}$ 是带有 Bottleneck 距离或 Wasserstein 距离的持续图空间。这些方法主要有两种类型。

预定向量化方法　该方法的常见做法是定义一个希尔伯特空间 $\mathcal{H}$ 和一个与尝试解决的深度学习任务无关的连续映射 $\phi : \mathcal{D} \to \mathcal{H}$。其中包括持续尺度空间核、Persistence Landscapes、加权高斯核或 Sliced Wasserstein Kernel 等。

可学习向量化方法　持续图向量化方法的另一种方法是，在一组向量化映射中为固定的数据分析任务学习“最佳”方法。更确切地说，假设我们正在考虑希尔伯特空间 $\mathcal{H}$ 上的一个固定学习任务（如监督分类）和一组向量化映射 $\phi_\theta : \mathcal{D} \to \mathcal{H}$。然后，这种方法包括根据学习任务提供的优化标准（通常是与 $\mathcal{H}$ 中的分类过程相关的损失函数）学习参数 θ 的最佳值。

第一篇介绍持续图可学习向量化方法的文章是 Hofer et al.[22]，其中 ϕ_θ 由对持续图点评估的二维高斯函数（其均值和标准差由 θ 定义）之和给出。在 Hofer et al.[23] 中，作者介绍了第一个能够接受持续图作为输入的神经网络架构。此外，作者还详细讨论了任何在 $\mathbb{R}^2$ 中包含恰好 n 个点的持续图集上的实值豪斯多夫连续函数都可以被任意逼近

$$L'(\{x_1, \ldots, x_n\}) := \rho\left(\sum_{i=1}^{n} \phi(x_i)\right),$$

其中 $\phi : \mathbb{R}^2 \to \mathbb{R}^p$ 和 $\rho : \mathbb{R}^p \to \mathbb{R}$ 是某些特定函数。他们为 ρ 和 ϕ 引入了特定类别的函数，这些函数随后在 Carrière et al.[10] 中通过 PersLay 架构得到了扩展。

本文的贡献在于将持续图直接作为 Transformer 架构的输入，使 Transformer 的强大功能和多样性在拓扑深度学习中得以使用，并且取得了不错的效果，在经典

的合成数据集和图数据集上，显著优于先前的拓扑深度学习架构。

第 3 章　背景知识

本章主要用于介绍拓扑数据分析的背景知识，包括拓扑数据分析主要的代数对象持续模，我们如何从数据中生成持续模；持续图的相关定义，以及定义在持续图上的度量；一个之后会用到的集合上的逼近函数的定理。

3.1　持续模

拓扑数据分析中主要的代数对象是持续模。

一个 *持续模* M 包含一个向量空间 M_a 对于所有 $a \in \mathbb{R}$，并且对于所有 $a \leqslant b$ 有线性映射 $M(a \leqslant b) : M_a \to M_b$，使得 $M(a \leqslant a)$ 是恒等映射，并且对于所有 $a \leqslant b \leqslant c$，有 $M(b \leqslant c) \circ M(a \leqslant b) = M(a \leqslant c)$。

拓扑数据分析工作者研究出了很多构造持续模的方法。其中一个笔者认为最自然的例子是从平面 $\mathbb{R}^2$ 中的点集 $X = \{x_1, \cdots, x_n\}$ 开始，如图 3-1 的左上方所示。我们“加厚”每个点，通过用一个固定半径 r 的圆盘 $B_x(r) = \{y \in M \mid \mathrm{d}(x, y) \leqslant r\}$ 替换每个点 x，其中圆盘的中心为 x。得到的联合体 $X_r = \bigcup_{i=1}^{n} B_r(x_i)$ 如图 3-1 中对于不同的 r 值所示。对于每个 r，我们可以计算 $H(X_r)$，即得到的圆盘联合体的同调。准确地说，$H(-)$ 表示 $H_k(-, \mathbb{F})$，是一个以域 $\mathbb{F}$ 为系数的 k 维的奇异同调函数。所以，$H(X_r)$ 是一个向量空间，是 k-循环与那些边界的商。随着 r 的增大，圆盘的联合体不断增大，相应的包含映射在同调群之间诱导了映射。更准确地说，如果 $r \leqslant s$，那么包含映射 $\iota_r^s : X_r \subseteq X_s$ 诱导了一个映射 $H(\iota_r^s) : H(X_r) \to H(X_s)$。这些映射的像就是 *持续同调群*。向量空间 $H(X_r)$ 和线性映射 $H(\iota_r^s)$ 的集合就是一个持续模。

注意，这个构造适用于任何在 $\mathbb{R}^n$ 中的点集，或者更一般地，适用于度量空间中的点集。

球的联合体 X_r 有一个很好的组合描述。*Čech 复形*，$\check{C}_r(X)$，是由球集 $\{B_{x_i}(r)\}$ 构成的单纯形复形，其顶点是点集 $\{x_i\}$，k-单纯形对应于有非空交集的 $k+1$ 个球（见图 3-1）。这也称为 *神经*。有一个基本结果是，如果我们所研究的空间是 $\mathbb{R}^n$，则 X_r 与它的 Čech 复形同伦等价。对该证明该兴趣的读者可以参考 Carlsson et al.[8]，这是一本非常好的拓扑数据分析入门书籍。因此，为了得到球的联合体的奇异同调，可以计算对应的 Čech 复形的单纯形同调。复形 $\{\check{C}_r(X)\}$ 和包含映射 $\check{C}_r(X) \subseteq \check{C}_s(X)$ 对于 $r \leqslant s$ 形成了一个过滤的单纯形复形。应用单纯形同调我们得

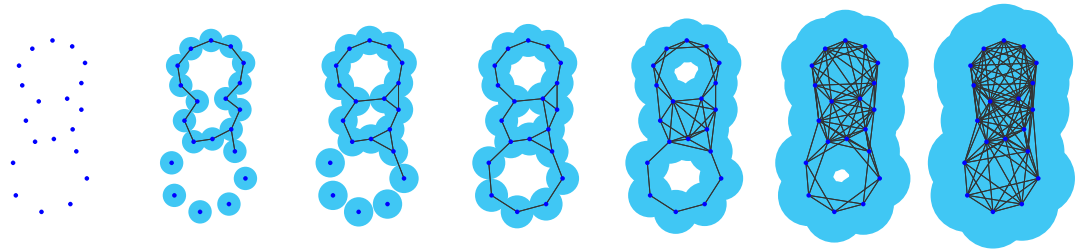

图 3-1　不断增大的球的联合体及其对应的 1-单纯形的 Čech 复形。随着半径的增大，特征——如连通分量和孔——出现并消失。在这里，复形展示了三个孔的出生和死亡，它们是度数为 1 的同调类。相应的出生-死亡对作为图 4-1 左上部分的一部分绘制。

到一个持续模。

Čech 复形通常计算开销较大，因此在计算拓扑中使用了许多变体。

一个较大但更简单的复形是 Rips 复形：

它的顶点是点集 x_i，k-单纯形对应于所有成对交集非空的 $k+1$ 个球。其他可能的复形包括 witness 复形[15]，图诱导复形[16]，以及使用核密度估计器和环境空间的三角剖分构建的复形[5]。

给定任何一个实值函数 $f: S \to \mathbb{R}$,定义其关联的持续模 $M(f)$,其中 $M(f)(a) = H(f^{-1}((\infty, a]))$，$M(f)(a \leqslant b)$ 由包含映射诱导。将 f 取为到一个有限点集 X 的最小距离，我们可以得到第一个持续模的例子。

3.2　持续图

持续图是 TDA 研究人员开发的最常用描述符。它们主要有两种类型：普通持续图和扩展持续图。关于这一部分数学背景的详细介绍，感兴趣的读者可以在 Dey et al.[17]中找到。

普通持续图跟踪拓扑特征在嵌套的拓扑空间序列（或单纯复形）$(X_t)_{t\in\mathbb{R}}$ 中的演变，这些特征的维度为 i（$i=0$ 时为连通分量，$i=1$ 时为孔洞，$i=2$ 时为空腔等），其中 $X_a \subset X_b$ 当且仅当 $a \leqslant b$。变量 $t \in \mathbb{R}$ 称为过滤值，直观上对应于拓扑特征演变的时间。不同拓扑特征的出现或消失取决于 t，这被称为新拓扑特征的诞生或死亡。如果在过滤的时间 $b \in \mathbb{R}$ 时一个维度为 i 的拓扑特征诞生，并在时间 $d \in \mathbb{R} \cup \{+\infty\}$ 时死亡，那么它将在该过滤的第 i 个持续图中产生一个坐标为 (b, d) 的点。因此，普通持续图是点的多重集（元素可以具有重数的集合），这些点在 $\mathbb{R} \times \mathbb{R} \cup \{+\infty\}$ 的子集 $\{(b, d) \in \mathbb{R} \times \mathbb{R} \cup \{+\infty\} \mid b < d\}$

扩展持续图推广了普通持续图，并涵盖了连续映射 $f: X \to \mathbb{R}$ 的纤维的拓扑特征的大小和类型。在实践中，扩展持续图是为单纯复形的 0 单纯形上的实值函数定义的。对于每个维度 i，f 的纤维的第 i 维拓扑特征用诞生时间 $b \in \mathbb{R}$ 和死

亡时间 $d \in \mathbb{R}$ 编码，并且属于以下四种类型之一：Ordinary, Relative, Extended+ 或 Extended-[14]。扩展持续图是普通持续图的严格推广，因为后者包含在前者中。此外，它具有计算上的优势，因为它仅包含具有有限坐标的点。我们后面还会更详细的介绍扩展持续图，见 5.2.1。

为了简单起见，我们将持续图视为集合而不是多重集，即我们假设持续图中的所有点都是不相交的。设 $X \subset \{(b,d) \in \mathbb{R}^2 \mid b < d\}$。

定义 3.1：集合 X 上的持续图集定义为

$$PD(X) := \{D \subset X \mid \forall K \subset X \text{ 紧致}, D \cap K \text{ 有限}\}$$

给定 $n \in \mathbb{Z}_{>0}$，我们定义包含 n 个点的持续图集为：

$$PD_n(X) := \{D \in PD(X) \mid D \text{ 有 } n \text{ 个元素}\}$$

3.3　持续图上的度量

持续图可以通过各种距离来比较，这些距离都定义为两个持续图之间点的部分匹配问题的最小成本。主要有两类匹配规则。在第一种情况下，必须将第一个图中的所有点与第二个图中的点一一对应。在第二种情况下，只寻找持续图点之间的部分双射，未匹配的点与它们在对角线 $\Delta = \{(x,x) \mid x \in \mathbb{R}\}$ 上的投影相匹配。使用 *Giotto-tda*[43]、*Gudhi*[32]或 *Dionysus*[33]等软件可以高效计算持续图和距离。

对于 $p \in \mathbb{R}_{\geqslant 1}$，以及 $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$，我们用 $\|x\|_p$ 表示 x 的 p-范数，定义为 $\|x\|_p = (\sum_i x_i^p)^{\frac{1}{p}}$。对于 $p = \infty$，我们设 $\|x\|_\infty = \max_i \|x_i\|$。给定 $D, D' \in PD_n(X)$ 和 $\sigma : D \longrightarrow D'$ 一个双射，我们通过选择一个排序 $D = z_1, \ldots, z_n$，定义 $c(\sigma) \in \mathbb{R}^n$，并设 $c(\sigma) = (\|z_1 - \sigma(z_1)\|_\infty, \ldots, \|z_n - \sigma(z_n)\|_\infty)$。注意，我们对 $c(\sigma)$ 的使用与我们在 D 上选取的排序无关。

定义 3.2：令 $n \in \mathbb{Z}_{>0}$，$p \in \mathbb{R}_{\geqslant 1} \cup \{\infty\}$ 且 $D, D' \in PD_n(X)$。D 与 D' 之间的 p-Wasserstein 距离定义为：

$$W^p(D, D') := \min_{\sigma : D \tilde{\to} D'} \|c(\sigma)\|_p,$$

其中 σ 遍历所有从 D 到 D' 的双射。

W^∞ 通常称为 Hausdorff 距离。

命题 3.1：令 $n \in \mathbb{Z}_{>0}$ 且 $p, q \in \mathbb{R}_{\geqslant 1} \cup \{\infty\}$，存在两个严格非负常数 $m(p,q)$ 和 $M(p,q)$，使得对所有 $D, D' \in PD_n(X)$ 都有：

$$m(p,q)\cdot W^q(D,D')\leqslant W^p(D,D')\leqslant M(p,q)\cdot W^q(D,D').$$

证明：这是有限维实向量空间中所有范数等价性的直接结果。 ■

W^∞ 通常被称为豪斯多夫距离

因此，由 W^p 在 $PD_n(X)$ 上引起的拓扑与 p 无关。

给定 $x\in\mathbb{R}^2$，我们用 $\pi(x)$ 表示 x 到对角线 Δ 上的正交投影。对于 $D,D'\in PD_n(X)$，D 和 D' 之间的部分匹配是数据的两个可能为空的子集 $I\subset D$ 和 $I'\subset D'$，以及一个双射 $\sigma:I\longrightarrow I'$。我们将使用记号 $(\sigma,I,I'):D\longrightarrow D'$。给定 $(\sigma,I,I'):D\longrightarrow D'$，其中 I 和 I' 有 ℓ 个元素，我们选择一个排序 $D=z_1,\dots,z_n$，同样 $D'=z'_1,\dots,z'_n$，其中对于所有 $i\leqslant\ell$，$z_i\in I$ 和 $z'_i\in I'$。我们定义 $c(\sigma,I,I')\in\mathbb{R}^{2n-\ell}$ 如下：

$$c(\sigma,I,I')_i=\begin{cases}\|z_i-\sigma(z_i)\|_\infty \text{ if } i\leqslant\ell\\ \|z_i-\pi(z_i)\|_\infty \text{ if } \ell+1\leqslant i\leqslant n\\ \|z'_{i-(n-\ell)}-\pi(z'_{i-(n-\ell)})\|_\infty \text{ if } n+1\leqslant i\leqslant 2n-\ell\end{cases}.$$

定义 3.3：令 $n\in\mathbb{Z}_{>0}$，$p\in\mathbb{R}_{\geqslant 1}\cup\infty$ 以及 $D,D'\in PD_n(X)$。定义 D 和 D' 之间的对角-p-Wasserstein 距离为：

$$W_d^p(D,D'):=\min_{(\sigma,I,I'):D\to D'}\|c(\sigma,I,I')\|_p,$$

其中 (σ,I,I') 遍历 D 和 D' 之间的所有部分匹配。

W_d^∞ 通常称为 Bottleneck 距离[17]。

命题 3.2：令 $n\in\mathbb{Z}_{>0}$ 和 $p\in\mathbb{R}_{\geqslant 1}\cup\{\infty\}$。由 W^p 和 W_d^p 在 $PD_n(X)$ 上引起的拓扑是相同的。

证明：因为匹配是部分匹配，对于所有 $D,D'\in PD_n(X)$，有 $W_d^p(D,D')\leqslant W^p(D,D')$。因此，任何 W^p-开子集都是 W_d^p-开子集。

对于所有 $D\in PD_n(X)$，存在 $\varepsilon_D>0$，使得对于所有 $0<\varepsilon\leqslant\varepsilon_D$：

$$\left\{D'\in PD_n(X)\mid W_d^p(D,D')<\varepsilon\right\}\subseteq\{D'\in PD_n(X)\mid W^p(D,D')<\varepsilon\}.$$

令 $D\in PD_n(X)$。我们定义 $\varepsilon_D:=\min_{z\in D}\|z-\pi(z)\|_\infty>0$，并令 $0<\varepsilon\leqslant\varepsilon_D$。令 $D'\in PD_n(X)$ 满足 $W_d^p(D,D')<\varepsilon\leqslant\varepsilon_D$。令 $(\sigma,I,I'):D\longrightarrow D'$ 满足 $\|c(\sigma,I,I')\|_p=W_d^p(D,D')$。然后 I 必须等于 D，因为否则，我们必须将一个点 $z\in D$ 与其在对角线上的投影匹配，因此 $\varepsilon\leqslant\varepsilon_D\leqslant\|z-\pi(z)\|_\infty\leqslant\|c(\sigma,I,I')\|_p=W_d^p(D,D')$。因此，$\sigma$ 是定义在 D 上的双射，满足 $\|c(\sigma)\|_p<\varepsilon$，因此 $W^p(D,D')<\varepsilon$。这证明了所需

的包含关系。 ■

3.4　集合上的逼近函数

回顾一下关于集合上的函数的一个有用的近似结果。

定理 3.1 (Theorem 9 Zaheer et al.[47])：设 X 是 $\mathbb{R}^d$ 的紧子集，M 是一个正整数。令 $2_M^X \subset 2^X$ 表示具有正好 M 个元素的 X 的子集，配备豪斯多夫度量。

对于任何豪斯多夫连续函数 $L: 2_M^X \to \mathbb{R}$ 和 $\varepsilon > 0$，存在一个自然数 $\varepsilon > 0$ 和两个连续函数 $\phi: X \to \mathbb{R}^p$ 和 $\rho: \mathbb{R}^p \to \mathbb{R}$，使得：

$$\sup_{S \in 2_M^X} \left| \rho\left(\sum_{x \in S} \phi(x) \right) - L(S) \right| \leqslant \varepsilon.$$

注释 3.1：上述定理对于多重集也同样适用。

利用神经网络的通用逼近定理 Haykin[21]，定理3.1的结论可以扩展为，对于 $\mathbb{R}^2$ 中具有均匀有界的有限子集 M 的每个豪斯多夫连续函数 L，都可以通过以下方式任意逼近

$$L'(x_1, \ldots, x_n) := \rho\left(\sum_{i=1}^{n} \phi(x_i) \right) \tag{3-1}$$

其中 ϕ 和 ρ 是具有有限隐藏层数和每层包含有限数量神经元的神经网络，激活函数为非常数、有界且非递减的连续函数，如 ReLU。像 (3-1) 这样的神经网络称为 Deep Set[47]。

在不使用位置编码且具有多头注意力池化的 `Transformer` 架构（见 Vaswani et al.[45]中的 Section 6.1）的编码器部分，与一个全连接神经网络组合，至少与 Deep Sets 模型具有一样的效果，见 Lee et al.[30]。因此，这种架构满足 *set transformers* 的通用逼近定理。此外，正如 Lee et al.[30]中所指出的，自注意机制通过堆叠多个层，能够实现集合实例之间的显式交互以及高阶交互。

第 4 章　预定向量化方法的实例

本章用于介绍预定向量化方法实例 Persistence Landscapes 以及与 Persistence Landscapes 相关的一些理论。

4.1　Persistence Landscapes

在这一节中，我们将介绍 Persistence Landscapes。我们首先介绍一些从持续模中派生的函数。每个函数的示例见图 4-1。

令 M 是一个持续模，定义见3.1。对于 $a \leqslant b$，M 对应的 *Betti* 数是由对应的线性映射的像的维度给出的。即，

$$\beta^{a,b} = \dim(\mathrm{im}(M(a \leqslant b))). \tag{4-1}$$

引理 4.1：如果 $a \leqslant b \leqslant c \leqslant d$，则 $\beta^{b,c} \geqslant \beta^{a,d}$。

证明：由于 $M(a \leqslant d) = M(c \leqslant d) \circ M(b \leqslant c) \circ M(a \leqslant b)$，可以由 (4-1) 推导出来。 ■

最简单的函数，我们称之为 *rank 函数*，是函数 $\lambda : \mathbb{R}^2 \to \mathbb{R}$，定义如下：

$$\lambda(b,d) = \begin{cases} \beta^{b,d} & \text{如果} b \leqslant d \\ 0 & \text{其他情况.} \end{cases}$$

现在，让我们进行坐标变换，使得得到的函数分布在上半平面。设

$$m = \frac{b+d}{2}, \quad \text{和} \quad h = \frac{d-b}{2}. \tag{4-2}$$

*参数变换后的 rank 函数*是函数 $\lambda : \mathbb{R}^2 \to \mathbb{R}$，给出如下：

$$\lambda(m,h) = \begin{cases} \beta^{m-h,m+h} & \text{如果} h \geqslant 0 \\ 0 & \text{其他情况.} \end{cases}$$

本节的理论大部分将应用于这些简单的函数。以下版本，一般称之为 Persistence Landscapes，感兴趣的读者可参考 Bubenik[4]。

首先，让我们观察到对于固定的 $t \in \mathbb{R}$，$\beta^{t-\bullet,t+\bullet}$ 是一个递减函数。也就是说，

引理 4.2：对于 $0 \leqslant h_1 \leqslant h_2$，

$$\beta^{t-h_1,t+h_1} \geqslant \beta^{t-h_2,t+h_2}.$$

证明：由于 $t - h_2 \leqslant t - h_1 \leqslant t + h_1 \leqslant t + h_2$，根据 Lemma 4.1，$\beta^{t-h_2,t+h_2} \leqslant$

$\beta^{t-h_1,t+h_1}$。 ■

定义 4.1：Persistence Landscapes 是一个函数 $\lambda:\mathbb{N}\times\mathbb{R}\to\overline{\mathbb{R}}$，其中 $\overline{\mathbb{R}}$ 表示扩展实数 $[-\infty,\infty]$。或者，它可以被看作是一个函数序列 $\lambda_k:\mathbb{R}\to\overline{\mathbb{R}}$，其中 $\lambda_k(t)=\lambda(k,t)$。定义

$$\lambda_k(t)=\sup(m\geqslant 0\mid \beta^{t-m,t+m}\geqslant k).$$

Persistence Landscapes 具有以下特性：

引理 4.3：　（1）$\lambda_k(t)\geqslant 0$，

（2）$\lambda_k(t)\geqslant\lambda_{k+1}(t)$，和

（3）λ_k 是 1-Lipschitz 连续的。

前两个性质直接从定义中得到。我们给出第三个的证明：

证明 Lemma 4.3(（3）)：我们将证明 λ_k 是 1-Lipschitz 的。也就是说，对于所有 $s,t\in\mathbb{R}$ 有

$$|\lambda_k(t)-\lambda_k(s)|\leqslant|t-s|.$$

取任意 $s,t\in\mathbb{R}$。不失一般性，假设 $\lambda_k(t)\geqslant\lambda_k(s)\geqslant 0$。若 $\lambda_k(t)\leqslant|t-s|$，则有 $\lambda_k(t)-\lambda_k(s)\leqslant\lambda_k(t)\leqslant|t-s|$，此时命题成立。因此，假设 $\lambda_k(t)>|t-s|$。

令 $0<h<\lambda_k(t)-|t-s|$。则有 $t-\lambda_k(t)<s-h<s+h<t+\lambda_k(t)$。因此，根据 Lemma 4.1 和 Definition 4.1，有

$$\beta^{s-h,s+h}\geqslant k.$$

从而可以推出 $\lambda_k(s)\geqslant\lambda_k(t)-|t-s|$。因此，$\lambda_k(t)-\lambda_k(s)\leqslant|t-s|$。 ■

为了帮助可视化 $\lambda:\mathbb{N}\times\mathbb{R}\to\overline{\mathbb{R}}$ 的图形，我们可以将其扩展为一个函数 $\overline{\lambda}:\mathbb{R}^2\to\overline{\mathbb{R}}$，定义如下：

$$\overline{\lambda}(x,t)=\begin{cases}\lambda(\lceil x\rceil,t), & \text{如果}x>0,\\ 0, & \text{如果}x\leqslant 0.\end{cases}\tag{4-3}$$

我们注意到，持续模 M 的非持续 Betti 数 $\{\dim(M(t))\}$ 可以从 rank 函数的对角线、参数变换 rank 函数的 m 轴，以及 Persistence Landscapes 的支撑集中得到。

4.2　Persistence Landscapes 与条形码

在一个特定的持续模中，所有的信息完全包含在一个称为 *条形码*的区间多重集中。将每个区间映射到其端点，我们得到持续图，持续图性质可回顾3.2。

这些拓扑描述符与 Persistence Landscapes 函数之间存在双向映射。对于对应的持续图、条形码和 Persistence Landscapes 的示例，见图 4-1。使用非正式的语言描

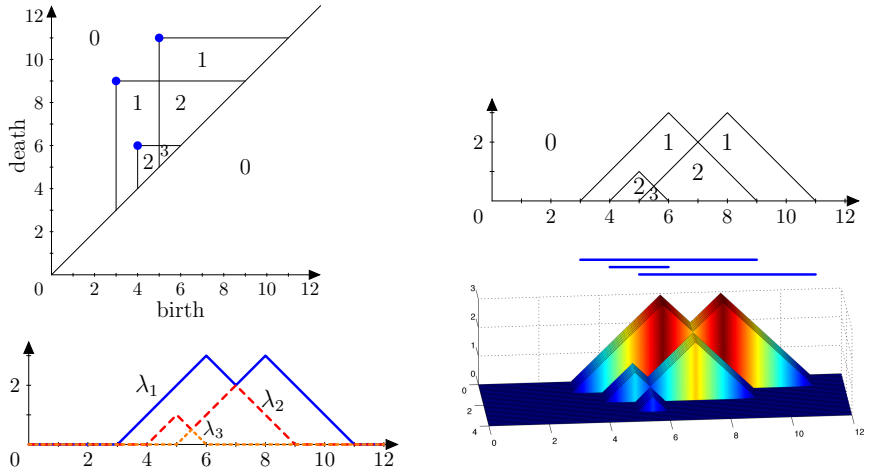

图 4-1　度数为 1 的同调的 Persistence Landscapes，图 3-1 中的示例。对于 rank 函数（左上）和参数变换 rank 函数（右上），给出了相应区域内的函数值。左上图还包含了相应持续图的三个点。右上图下方是对应的条形码。我们还展示了相应的 Persistence Landscapes（左下）及其三维版本（右下）。注意，λ_1 给出了每个过滤点的主导同调特征的度量。

述，持续图由 rank 函数中的“左上角”组成。反过来，$\lambda(b,d)$ 计算持续图中 (b,d) 的左上象限中的点的数量。使用非正式的语言描述，条形码由参数变换的 rank 函数中的“三角形的底边”组成，另一方向是通过“堆叠等腰三角形”来获得，其底边是条形码中的区间。

例如，给定一个持续图 $\{(b_i,d_i)\}_{i=1}^n$，

$$\lambda_k(t) = k\text{th largest value of } \min(t-b_i, d_i-t)_+,$$

其中 c_+ 表示 $\max(c,0)$。条形码是持续模的一个完整不变量，这一事实是这些等价关系的核心。

持续图空间缺少所谓的“几何结构”使得它很难处理。例如，持续图的集合不一定有唯一的（Fréchet）均值。相比之下，Persistence Landscapes 空间则非常好。因此，一组 Persistence Landscapes 具有唯一的均值。见图 4-2。

与持续图相比，条形码有额外的信息，说明区间的端点是否包含在内。此更精细的信息在 rank 函数和参数变换 rank 函数中可见，但在 Persistence Landscapes 中不可见。然而，当我们在第 4.3节中转到相应的 L^p 空间时，这些信息会消失。

4.3　Persistence Landscapes 的范数

回忆一下，对于一个测度空间 $(\mathcal{S},\mathcal{A},\mu)$，以及一个在 μ-几乎处处定义的函数 $f:\mathcal{S}\to\mathbb{R}$，对于 $1\leqslant p<\infty$，$\|f\|_p=\left[\int|f|^p d\mu\right]^{\frac{1}{p}}$，并且 $\|f\|_\infty=\operatorname{ess\,sup} f=\inf\{a\mid \mu\{s\in\mathcal{S}\mid f(s)>a\}=0\}$。对于 $1\leqslant p\leqslant\infty$，定义 $\mathcal{L}^p(\mathcal{S})=\{f:\mathcal{S}\to\mathbb{R}\mid\|f\|_p<\infty\}$，并定义 $L^p(\mathcal{S})=\mathcal{L}^p(\mathcal{S})/\sim$，其中 $f\sim g$ 如果 $\|f-g\|_p=0$。

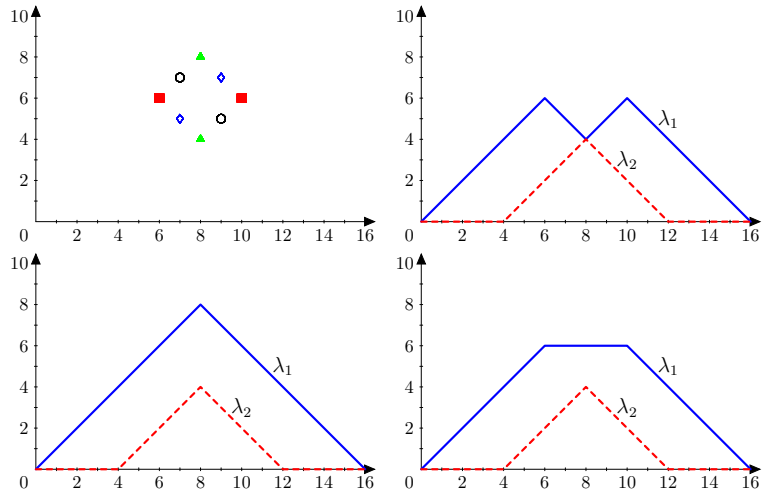

图 4-2　持续图与 Persistence Landscapes 的均值。左上：参数变换持续图 $\{(6,6),(10,6)\}$ 和 $\{(8,4),(8,8)\}$ 有两个（Fréchet）均值：$\{(7,5),(9,7)\}$ 和 $\{(7,7),(9,5)\}$。相比之下，它们对应的 Persistence Landscapes（右上和左下）具有唯一的均值（右下）。

在 $\mathbb{R}$ 和 $\mathbb{R}^2$ 上，我们将使用勒贝格测度。在 $\mathbb{N}\times\mathbb{R}$ 上，我们使用在 $\mathbb{N}$ 上的计数测度和在 $\mathbb{R}$ 上的勒贝格测度的乘积。对于 $1\leqslant p<\infty$ 和 $\lambda:\mathbb{N}\times\mathbb{R}\to\overline{\mathbb{R}}$，

$$\|\lambda\|_p^p=\sum_{k=1}^{\infty}\|\lambda_k\|_p^p,$$

其中 $\lambda_k(t)=\lambda(k,t)$。通过 Lemma 4.3(（2）)，$\|\lambda\|_\infty=\|\lambda_1\|_\infty$。如果我们将 f 扩展为 $\overline{\lambda}:\mathbb{R}^2\to\overline{\mathbb{R}}$，如同 (4-3) 中所示，我们有 $\|\lambda\|_p=\|\overline{\lambda}\|_p$，对于 $1\leqslant p\leqslant\infty$。

如果 λ 是与条形码对应的任何函数，它是有限区间的有限集合，那么 $\lambda\in\mathcal{L}^p(\mathcal{S})$ 对于 $1\leqslant p\leqslant\infty$，其中 $\mathcal{S}$ 等于 $\mathbb{N}\times\mathbb{R}$ 或 $\mathbb{R}^2$。

让 λ_{bd} 和 λ_{mh} 分别表示 Persistence Landscapes λ 对应的 rank 函数和参数变换的 rank 函数，并且让 D 是相应的持续图。让 $\mathrm{pers}_2(D)$ 表示条形码中区间长度的平方和，并且让 $\mathrm{pers}_\infty(D)$ 是最长区间的长度。

命题 4.1：　（1）$\|\lambda\|_1=\|\lambda_{mh}\|_1=\frac{1}{2}\|\lambda_{bd}\|_1=\frac{1}{4}\mathrm{pers}_2(D)$，和

（2）$\|\lambda\|_\infty=\|\lambda_1\|_\infty=\frac{1}{2}\mathrm{pers}_\infty(D)$。

证明：　（1）要看 $\|\lambda\|_1=\|\lambda_{mh}\|$，我们注意到它们的体积是相同的。坐标变换表明 $\|\lambda_{mh}\|_1=\frac{1}{2}\|\lambda_{bd}\|_1$。如果 $D=\{(b_i,d_i)\}$，则每个点 (b_i,d_i) 对体积 $\|\lambda_{mh}\|_1$ 贡献 h_i^2，其中 $h_i=\frac{d_i-b_i}{2}$。所以 $\|\lambda_{mh}\|_1=\sum_i h_i^2$。最后，$\mathrm{pers}_2(D)=\sum_i(2h_i)^2=4\sum_i h_i^2$。

（2）引理 4.3(（2）) 表明 $\|\lambda\|_\infty=\|\lambda_1\|_\infty$。如果 $D=\{(b_i,d_i)\}$，则 $\|\lambda\|_\infty=\sup_i\frac{d_i-b_i}{2}$。

■

第 5 章　可学习向量化方法实例

本章用于介绍可学习向量化方法实例 PLLay 和 PersLay。

5.1　PLLay

本节用于介绍拓扑深度学习近期成果之一的 PLLay 架构，感兴趣的读者可以参考 Kim et al.[26]

5.1.1　测度距离 (DTM) 函数

我们首先介绍 PLLay 架构必须要用到的测度距离 (DTM)。测度距离 (DTM)[12]是距离函数的一种鲁棒化版本。更精确地说，对于一个概率分布 μ，其参数 $m_0 \in (0,1)$ 且 $r \geqslant 1$，DTM 定义为 $d_{\mu,m_0}: \mathbb{R}^d \to \mathbb{R}$：

$$d_{\mu,m_0}(x) = \left(\frac{1}{m_0} \int_0^{m_0} (\delta_{\mu,m}(x))^r \, dm \right)^{1/r},$$

其中，当 $\mathbb{B}(x,t)$ 表示以 x 为中心、半径为 t 的开球时，有

$$\delta_{\mu,m}(x) = \inf\{t > 0 : \mu(\mathbb{B}(x,t)) > m\}.$$

其中 ϖ_i 为 μ 的权重。在这种情况下，我们定义 *经验 DTM 函数*为

$$\hat{d}_{m_0}(x) = d_{P_n,m_0}(x) = \left(\frac{\sum_{X_i \in N_k(x)} \varpi_i' \|X_i - x\|^r}{m_0 \sum_{i=1}^n \varpi_i} \right)^{1/r}, \tag{5-1}$$

其中 $N_k(x)$ 是 $\{X_1, \cdots, X_n\}$ 中距离 x 最近的 k 个邻居的子集，且 k 满足

$$\sum_{X_i \in N_{k-1}(x)} \varpi_i < m_0 \sum_{i=1}^n \varpi_i \leqslant \sum_{X_i \in N_k(x)} \varpi_i,$$

并且如果至少有一个 X_i 属于 $N_k(x)$，则定义

$$\varpi_i' = \sum_{X_j \in N_k(x)} \varpi_j - m_0 \sum_{j=1}^n \varpi_j,$$

否则 $\varpi_i' = \varpi_i$。因此，经验 DTM 的表现类似于 k-近邻距离，其中 $k = \lfloor m_0 n \rfloor$。对于独立同分布（independent and identically distributed 后面简称为 i.i.d. ）的情况，我们通常设 $\varpi_i = 1$，但这些权重可以通过数据驱动的方式灵活确定。参数 m_0 决定了从

局部或全局结构中应提取多少拓扑/几何信息。关于 DTM 参数选择可以在 Chazal et al.[12]）中找到。由于所得的持续图对输入扰动不太敏感且具有良好的稳定性，因此人们通常倾向于使用 DTM 作为过滤函数。

5.1.2　图的计算：$X \to \mathcal{D}_X$

本节用于介绍在 PLLay 架构中如何从输入数据计算持续图。为了从输入数据中计算持续图，首先需要定义过滤（filtration），这要求给出一个单纯形复形 K 以及一个函数 $f: K \to \mathbb{R}$。关于 K 和 f 有多种选择。使用哪种过滤方法在实际操作中往往依赖于具体问题，读者有兴趣可以参考 Otter et al.[36]。

正如在第 5.1.1 节中所述，f 的一个非常好的选择是 DTM 函数。由于其优良的性质，DTM 函数已被广泛应用于 TDA[2]，且在深度学习应用中具有良好的潜力。

- 如果将输入数据 X 视为经验数据点，则带权重 ϖ_i 的经验 DTM 如式 (5-1) 变为

$$\hat{d}_{m_0}(x) = \left(\frac{\sum_{X_i \in N_k(x)} \varpi_i' \left\| X_i - x \right\|^r}{m_0 \sum_{i=1}^n \varpi_i} \right)^{1/r}, \tag{5-2}$$

其中 k 以及 ϖ_i' 按照式 (5-1) 确定。

- 如果将输入数据 X 视为对应于固定点 $\{Y_1, \cdots, Y_n\}$ 的权重，则带有数据点 Y_i 和权重 X_i 的经验 DTM 如式 (5-1) 变为

$$\hat{d}_{m_0}(x) = \left(\frac{\sum_{X_i \in N_k(x)} X_i' \left\| Y_i - x \right\|^r}{m_0 \sum_{i=1}^n X_i} \right)^{1/r}, \tag{5-3}$$

其中 k 以及 ϖ_i' 同样按式 (5-1) 确定。

5.1.3　PLLay 的构造

PLLay 架构的拓扑层基于一个参数化映射，该映射利用 Persistence Landscapes 将持续图 $\mathcal{D}$ 投影到 $\mathbb{R}$ 上。与 Hofer et al.[22] 提出的特定变换过程相比，PLLay 架构的优势体现在避免了由人工弯曲（artificial bending）带来的影响，同时仍能保证持续图中的关键信息得到良好保留（见第 5.1.5 节）。对于持续度较低的不重要点，系统性地忽略它们不会引入额外的干扰参数[6]。

令 $\mathbb{R}^{+0}$ 表示 $[0, \infty)$。给定持续图 $\mathcal{D} \in \mathbb{D}$，我们计算第 k 阶 Persistence Landscapes $\lambda_k(t)$（参见4.1），其中 $k = 1, \cdots, K_{max}$。接着，我们计算加权均值：

$$\overline{\lambda}_{\boldsymbol{\omega}}(t) \coloneqq \sum_{k=1}^{K_{max}} \omega_k \lambda_k(t),$$

其中权重参数 $\boldsymbol{\omega} = \{\omega_k\}_k$，满足 $\omega_k > 0, \sum_k \omega_k = 1$。然后，我们设定一个区间

算法 5-1　实现 PLLay 的单个结构元素

输入: 持续图 $\mathcal{D} \in \mathbb{D}$

（1）计算 $\lambda_k(t)$ (4.1)，其中 $t \in [0,T]$，$k = 1,\cdots,K_{max}$

（2）计算加权均值 $\overline{\lambda}_{\boldsymbol{\omega}}(t) \coloneqq \sum_{k=1}^{K_{max}} \omega_k \lambda_k(t)$，其中 $\omega_k > 0, \sum_k \omega_k = 1$

（3）设定 $\nu \coloneqq \frac{T}{m-1}$，并计算 $\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}} = (\overline{\lambda}_{\boldsymbol{\omega}}(T_{\min}), \overline{\lambda}_{\boldsymbol{\omega}}(T_{\min}+\nu), ..., \overline{\lambda}_{\boldsymbol{\omega}}(T_{\max}))^{\top} \in \mathbb{R}^m$

（4）设定参数化可微映射 $g_{\boldsymbol{\theta}}: \mathbb{R}^m \to \mathbb{R}$，定义 $S_{\boldsymbol{\theta},\boldsymbol{\omega}} = g_{\boldsymbol{\theta}} \circ \overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}$

输出: $S_{\boldsymbol{\theta},\boldsymbol{\omega}}: \mathbb{D} \to \mathbb{R}$

$[T_{\min}, T_{\max}]$ 和分辨率 $\nu \coloneqq T/(m-1)$，并在 $[T_{\min}, T_{\max}]$ 中均匀采样 m 个点，得到：

$$\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}} = \left(\overline{\lambda}_{\boldsymbol{\omega}}(T_{\min}), \overline{\lambda}_{\boldsymbol{\omega}}(T_{\min}+\nu), ..., \overline{\lambda}_{\boldsymbol{\omega}}(T_{\max})\right)^{\top} \in \left(\mathbb{R}^{+0}\right)^m.$$

因此，我们定义了一个映射：

$$\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}: \mathbb{D} \to \left(\mathbb{R}^{+0}\right)^m,$$

它是一个加权 Persistence Landscapes 在分辨率 ν 处的（向量化的）有限样本近似。最后，我们考虑一个参数化的可微映射 $g_{\boldsymbol{\theta}}: \left(\mathbb{R}^{+0}\right)^m \to \mathbb{R}$，它接受输入 $\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}$，并且对 $\boldsymbol{\theta}$ 也是可微的。那么，以 $S_{\boldsymbol{\theta},\boldsymbol{\omega}} \coloneqq g_{\boldsymbol{\theta}} \circ \overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}$ 定义的映射，就是我们拓扑输入层的一个结构元素。算法总结如5-1

映射 $S_{\boldsymbol{\theta},\boldsymbol{\omega}}$ 在每个 $t \in [T_{\min}, T_{\max}]$ 处都是连续的。此外，它对 $\boldsymbol{\omega}$ 和 $\boldsymbol{\theta}$ 是可微的，与分辨率 ν 无关。PLLay 算法的实现有一些规律可循：

关于 $\boldsymbol{\omega}$ 权重参数 $\boldsymbol{\omega}$ 可以初始化为均匀分布，即 $\omega_k = 1/K_{max}$，并在训练过程中通过 softmax 层重新调整，使得携带重要信息的景观具有更高的权重。一般来说，低阶景观比高阶景观更重要，但最优权重可能因任务而异。

关于 $\boldsymbol{\theta}, \mathbf{g}_{\boldsymbol{\theta}}$: 某些出生-死亡对（birth-death pairs）在景观函数中包含的拓扑信息比其他对更重要。这可以理解为，某些景观中的“山峰”或“脊线”对结构更加关键。因此，$g_{\boldsymbol{\theta}}$ 应该能够反映这一点。一般来说，它可以通过线性变换加非线性归一化来实现。以下是两种可能的选择：

- 仿射变换（Affine transformation）：设比例参数 $\boldsymbol{\sigma}_i$ 和位移参数 $\boldsymbol{\mu}_i \in \mathbb{R}^m$，定义

$$g_{\boldsymbol{\theta}_i}(\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}) = \boldsymbol{\sigma}_i^{\top}(\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}} - \boldsymbol{\mu}_i),$$

其中 $\boldsymbol{\theta}_i = (\boldsymbol{\sigma}_i, \boldsymbol{\mu}_i)$。

- 对数变换（Logarithmic transformation）：设 $\boldsymbol{\theta}_i = (\sigma_i, \boldsymbol{\mu}_i)$，定义

$$g_{\boldsymbol{\theta}_i}(\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}) = \exp\left(-\sigma_i \|\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}} - \boldsymbol{\mu}_i\|_2\right)。$$

其他关于 $g_{\boldsymbol{\theta}}, \boldsymbol{\theta}, \boldsymbol{\omega}$ 的构造也是可能的，只要它们满足上述条件。由于每个结构元素对应于层中的一个节点，我们可以组合多个这样的元素，每个元素具有不同

的参数，从而构建完整的拓扑层。

定义 5.1 (基于 Persistence Landscapes 的拓扑层（PLLay）)：设 $n_h \in \mathbb{N}$，令 $\boldsymbol{\eta}_i = (\boldsymbol{\theta}_i, \boldsymbol{\omega}_i)$ 表示第 i 个结构元素的参数集，并设 $\boldsymbol{\eta} = (\boldsymbol{\eta}_i)_{i=1}^{n_h}$。给定持续图 $\mathcal{D}$ 和分辨率 ν，定义 $PLLay$ 为

$$h_{\text{top}} : \mathcal{D} \to \left(S_{\boldsymbol{\eta}_i}(\mathcal{D}; \nu)\right)_{i=1}^{n_h} \quad (5\text{-}4)$$

5.1.4　可微性

本小节用于分析 PLLay 关于其输入（或前一层的输出）的微分行为，即计算导数 $\frac{\partial h_{\text{top}}}{\partial X}$。由于

$$\frac{\partial h_{\text{top}}}{\partial X} = \frac{\partial h_{\text{top}}}{\partial \mathcal{D}_X} \circ \frac{\partial \mathcal{D}_X}{\partial X},$$

这可以通过结合两个导数 $\frac{\partial \mathcal{D}_X}{\partial X}$ 和 $\frac{\partial h_{\text{top}}}{\partial \mathcal{D}_X}$ 来完成。PLLay 的工作在 Poulenard et al.[38] 的工作基础上进行了扩展。我们将在定理 1 中重述 PLLay 中的结果。

定理 1：令 f 为过滤函数。令 ξ 为一个将每个出生-死亡点 $(b_i, d_i) \in \mathcal{D}_X$ 映射到一对单纯形 (β_i, δ_i) 的映射。假设 ξ 在 X 处局部不变，且 $f(\beta_i)$ 和 $f(\delta_i)$ 关于 X_j 是可微的。那么，h_{top} 关于 X 是可微的，并且有

$$\frac{\partial h_{\text{top}}}{\partial X_j} = \sum_i \frac{\partial f(\beta_i)}{\partial X_j} \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial b_i} + \sum_i \frac{\partial f(\delta_i)}{\partial X_j} \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial d_i}.$$

证明：对于计算 $\frac{\partial h_{\text{top}}}{\partial X_j}$，注意它可以利用链式法则展开为

$$\frac{\partial h_{\text{top}}}{\partial X_j} = \sum_i \frac{\partial h_{\text{top}}}{\partial b_i} \frac{\partial b_i}{\partial X_j} + \sum_i \frac{\partial h_{\text{top}}}{\partial d_i} \frac{\partial d_i}{\partial X_j}, \quad (5\text{-}5)$$

因此，我们需要计算

$$\frac{\partial \mathcal{D}_X}{\partial X} = \left\{ \left(\frac{\partial b_i}{\partial X_j}, \frac{\partial d_i}{\partial X_j} \right) \right\}_{(b_i, d_i) \in \mathcal{D}_X, X_j \in X}$$

以及

$$\frac{\partial h_{\text{top}}}{\partial \mathcal{D}_X} = \left\{ \left(\frac{\partial h_{\text{top}}}{\partial b_i}, \frac{\partial h_{\text{top}}}{\partial d_i} \right) \right\}_{(b_i, d_i) \in \mathcal{D}_X},$$

以便计算 $\frac{\partial h_{\text{top}}}{\partial X_j}$。

我们首先计算 $\frac{\partial \mathcal{D}_X}{\partial X}$。令 K 为单纯形复形，并假设在过滤过程中所有单纯形按照 f 值非递减的顺序排列，即如果 ς 出现在 τ 之前，则有 $f(\varsigma) \leqslant f(\tau)$。注意，从每个出生-死亡点 $(b_i, d_i) \in \mathcal{D}_X$ 到一对单纯形 (β_i, δ_i) 的映射 ξ 就是标准持续图 Carlsson et al.[9] 返回的配对。设 γ 为对应于 (b_i, d_i) 的同调特征，则出生单纯形 β_i 是在

$K_{b_i} = f^{-1}(-\infty, b_i]$ 中形成 γ 的单纯形，而死亡单纯形 δ_i 则是在 $K_{d_i} = f^{-1}(-\infty, d_i]$ 中使得 γ 消失的单纯形。例如，如果 γ 是一个一维特征，则 β_i 是在 K_{b_i} 中形成该环的边，而 δ_i 则是 K_{d_i} 中使该环能够收缩的三角形。

现在，有 $f(\xi(b_i)) = f(\beta_i) = b_i$ 和 $f(\xi(d_i)) = f(\delta_i) = d_i$，且由于 ξ 在 X 处局部不变，

$$\frac{\partial b_i}{\partial X_j} = \frac{\partial f(\xi(b_i))}{\partial X_j} = \frac{\partial f(\beta_i)}{\partial X_j}, \quad \frac{\partial d_i}{\partial X_j} = \frac{\partial f(\xi(d_i))}{\partial X_j} = \frac{\partial f(\delta_i)}{\partial X_j}. \tag{5-6}$$

因此，出生值和死亡值的导数就是在相应单纯形处计算的过滤函数的导数。而

$$\frac{\partial \mathcal{D}_X}{\partial X} = \left\{\left(\frac{\partial b_i}{\partial X_j}, \frac{\partial d_i}{\partial X_j}\right)\right\}_{(b_i,d_i)\in\mathcal{D}_X, X_j\in X}$$

就是这些导数的集合，由 (5-6) 得到

$$\frac{\partial \mathcal{D}_X}{\partial X} = \left\{\left(\frac{\partial b_i}{\partial X_j}, \frac{\partial d_i}{\partial X_j}\right)\right\}_{(b_i,d_i)\in\mathcal{D}_X, X_j\in X} = \left\{\left(\frac{\partial f(\beta_i)}{\partial X_j}, \frac{\partial f(\delta_i)}{\partial X_j}\right)\right\}_{\xi^{-1}(\beta_i,\delta_i)\in\mathcal{D}_X, X_j\in X}. \tag{5-7}$$

接下来，我们计算

$$\frac{\partial h_{\text{top}}}{\partial \mathcal{D}_X} = \left\{\left(\frac{\partial h_{\text{top}}}{\partial b_i}, \frac{\partial h_{\text{top}}}{\partial d_i}\right)\right\}_{(b_i,d_i)\in\mathcal{D}_X}.$$

计算 $\frac{\partial h_{\text{top}}}{\partial b_i}$ 可以利用链式法则对 $h_{\text{top}} = S_{\boldsymbol{\theta},\boldsymbol{\omega}} = g_{\boldsymbol{\theta}} \circ \overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}$ 进行展开：

$$\frac{\partial h_{\text{top}}}{\partial b_i} = \frac{\partial S_{\boldsymbol{\theta},\boldsymbol{\omega}}}{\partial b_i} = \frac{\partial (g_{\boldsymbol{\theta}} \circ \overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}})}{\partial b_i} = \nabla g_{\boldsymbol{\theta}} \circ \frac{\partial \overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}}{\partial b_i} = \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \frac{\partial \overline{\lambda}_{\boldsymbol{\omega}}(lv)}{\partial b_i}, \tag{5-8}$$

其中 x_l 是 $g_{\boldsymbol{\theta}}$ 的输入的简记。接着，利用

$$\overline{\lambda}_{\boldsymbol{\omega}}(lv) = \sum_{k=1}^{K_{max}} \omega_k \lambda_k(lv),$$

将上式 (5-8) 展开得到

$$\frac{\partial h_{\text{top}}}{\partial b_i} = \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial b_i}. \tag{5-9}$$

类似地，$\frac{\partial h_{\text{top}}}{\partial d_i}$ 可计算为

$$\frac{\partial h_{\text{top}}}{\partial d_i} = \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial d_i}. \tag{5-10}$$

因此，$\frac{\partial h_{\text{top}}}{\partial \mathcal{D}_X}$ 就是由式 (5-9) 和 (5-10) 中的导数构成的集合，即

$$\frac{\partial h_{\text{top}}}{\partial \mathcal{D}_X} = \left\{ \left(\sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial b_i}, \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial d_i} \right) \right\}_{(b_i,d_i)\in \mathcal{D}_X} . \quad (5\text{-}11)$$

■

因此，可以利用式 (5-7) 和 (5-11) 对式 (5-5) 进行求和，得到

$$\begin{aligned} \frac{\partial h_{\text{top}}}{\partial X_j} &= \sum_i \frac{\partial h_{\text{top}}}{\partial b_i} \frac{\partial b_i}{\partial X_j} + \sum_i \frac{\partial h_{\text{top}}}{\partial d_i} \frac{\partial d_i}{\partial X_j} \\ &= \sum_i \frac{\partial f(\beta_i)}{\partial X_j} \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial b_i} + \sum_i \frac{\partial f(\delta_i)}{\partial X_j} \sum_{l=1}^{m} \frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l} \sum_{k=1}^{K_{\max}} \omega_k \frac{\partial \lambda_k(lv)}{\partial d_i}. \end{aligned}$$

注意，$\frac{\partial \lambda_k}{\partial b_i}$ 和 $\frac{\partial \lambda_k}{\partial d_i}$ 是分段常数，并且可以以显式形式容易地计算出来。同时，$\frac{\partial g_{\boldsymbol{\theta}}}{\partial x_l}$ 可通过自动微分框架（如 `tensorflow` 或 `pytorch`）轻松实现。因此，根据定义 5.1，PLLay 可以在网络中的任意位置通过反向传播进行训练。

5.1.5　稳定性分析

PLLay 的一个关键性质是稳定性，用于判别架构应对非系统性噪声或输入数据的扰动保持稳定的能力。在本节中，我们将给出 PLLay 层稳定性性质的理论结果。首先，我们在定理 2 中给出了对于每个结构元素关于持续图变化的稳定性。

定理 2：设 $g_{\boldsymbol{\theta}}$ 是 $\|\cdot\|_\infty$-Lipschitz 的，即存在常数 $L_g > 0$ 使得对于所有 $x, y \in \mathbb{R}^m$ 有

$$\left| g_{\boldsymbol{\theta}}(x) - g_{\boldsymbol{\theta}}(y) \right| \leqslant L_g \|x - y\|_\infty .$$

那么，对于两个持续图 $\mathcal{D}$ 和 $\mathcal{D}'$，有

$$\left| S_{\boldsymbol{\theta},\boldsymbol{\omega}}(\mathcal{D}; v) - S_{\boldsymbol{\theta},\boldsymbol{\omega}}(\mathcal{D}'; v) \right| \leqslant L_g \, d_B(\mathcal{D}, \mathcal{D}').$$

证明：详细证明感兴趣的读者可以参考 Kim et al.[26] ■

该定理表明，$S_{\boldsymbol{\theta},\boldsymbol{\omega}}$ 对于由 Bottleneck 距离3.3测量的持续图扰动具有稳定性。需要注意的是，仅需要 $g_{\boldsymbol{\theta}}$ 的 Lipschitz 连续性即可建立这一结果。

特别地，当我们使用式 (5-2) 和 (5-3) 中提出的基于 DTM 函数的过滤时，定理 2 可转化为关于输入 X 的以下稳定性结果。

定理 3：假设 DTM 函数采用 $r = 2$。给定一个可微函数 $g_{\boldsymbol{\theta}}$ 和分辨率 v，以及一个分布 P。对于 X_j 为数据点的情况，即当使用式 (5-2) 作为 X 的 DTM 函数时，令 P_n

为经验分布，定义为

$$P_n = \frac{\sum_{i=1}^{n} \varpi_i \delta_{X_i}}{\sum_{i=1}^{n} \varpi_i}.$$

对于 X_j 为权重的情况，即当使用式 (5-3) 作为 X 的 DTM 函数时，令 P_n 为经验分布，定义为

$$P_n = \frac{\sum_{i=1}^{n} X_i \delta_{Y_i}}{\sum_{i=1}^{n} X_i}.$$

令 $\mathcal{D}_P$ 为 P 的 DTM 过滤得到的持续图，$\mathcal{D}_X$ 为 X 的 DTM 过滤得到的持续图。则有

$$\left| S_{\boldsymbol{\theta},\boldsymbol{\omega}}(\mathcal{D}_X; v) - S_{\boldsymbol{\theta},\boldsymbol{\omega}}(\mathcal{D}_P; v) \right| \leqslant L_g \, m_0^{-1/2} W_2(P_n, P).$$

证明：详细的证明读者可以参考 Kim et al.[26] ■

定理 3 表明，如果由给定输入 X 诱导的经验分布 P_n 在 Wasserstein 距离下很好地逼近真实分布 P（即 $W_2(P_n, P)$ 很小），那么基于观测数据构造的 PLLay 与基于真实分布 P 构造的结果相近。

定理 2 与定理 3 表明，所提出层中嵌入的拓扑信息对小的噪声、数据干扰或异常值具有鲁棒性。

本节主要介绍 PLLay 架构的理论知识，关于实验部分感兴趣的读者可以参考 Kim et al.[26]。

5.2　PersLay

本节用于介绍另一个可学习向量化方法的实例 PersLay 架构，感兴趣的读者可以参考 Carrière et al.[10]

5.2.1　扩展持续图

通常，普通持续不能完全编码 X 的拓扑信息。例如，考虑一个图 $G = (V, E)$，其中 V 是顶点集，E 是（无向）边集。设 $f: V \to \mathbb{R}$ 是定义在顶点上的一个函数，并考虑 下水平图 $G_\alpha = (V_\alpha, E_\alpha)$，其中 $\alpha \in \mathbb{R}$，$V_\alpha = v \in V, :, f(v) \leqslant \alpha$，$E_\alpha = (v_1, v_2) \in E, :, v_1, v_2 \in V_\alpha$。

在这个序列 $(G_\alpha)_\alpha$ 中，环（loop）会永远持续存在，因为它们不会在下水平图序列中消失（它们不会被“填充”）。同样，整个图的连通分量也不会消失。此外，向上的分支（相对于由 f 给定的方向，参见图 5-1）不会被检测到，而向下的分支会被检测到，因为当它们在下水平图中出现时不会创建新的连通分量，使得普通持续无法检测到它们。

为了解决这个问题，扩展持续通过观察所谓的 上水平集 X^α =

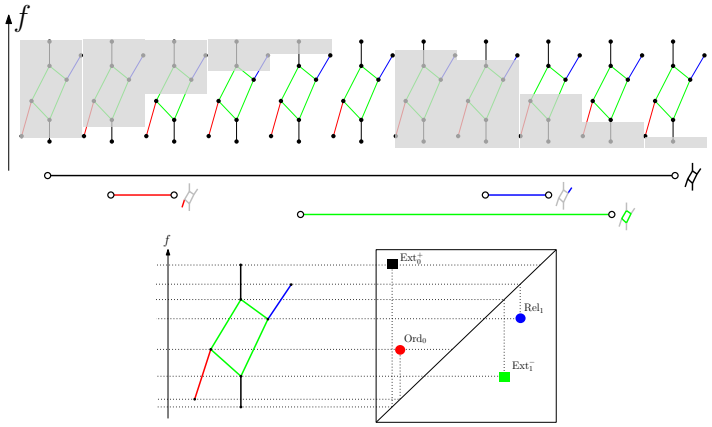

图 5-1　在图上计算的扩展持续图：图的拓扑特征在图左侧显示的下水平和上水平子图序列中被检测到。对应的区间显示在该序列下方：黑色区间表示图的连通分量，红色区间表示其向下的分支，蓝色区间表示其向上的分支，绿色区间表示其环。由这些区间给出的扩展持续图显示在下方。

$x \in X, :, f(x) \geqslant \alpha$ 来细化分析。

类似于普通持续对下水平集的处理，使 α 从 $+\infty$ 递减到 $-\infty$ 生成一个单调递增的子集序列，其中可以记录结构变化。

虽然扩展持续可以在一般度量空间中定义，但我们在此仅讨论 $X = G$ 为图的情况。

特别地，可以为环和整个连通分量定义死亡时间，即选取这些特征再次出现的上水平图，并使用对应的 α 值作为这些特征的死亡时间。在这种情况下，向上的分支可以在这个上水平图序列中被检测到，与下水平图中检测向下分支的方式完全相同。

参见图 5-1 以获得直观示例。

最后，将形如 $[\alpha_b, \alpha_d]$ 的区间族通过将区间端点作为坐标，转换为欧几里得平面 $\mathbb{R}^2$ 上的一个多重集。该多重集称为函数 f 的 *扩展持续图*，记作 $\mathrm{Dg}(G, f) \subset \mathbb{R}^2$。

由于图具有四种拓扑特征（参见图 5-1），即向上分支、向下分支、环和连通分量，因此扩展持续图中对应的点可以分为四种不同的类型。这些类型分别记为 Ord_0、Rel_1、Ext_0^+ 和 Ext_1^-，分别对应向下分支、向上分支、连通分量和环。

虽然扩展持续编码的信息比普通持续更多，但它确保了所有点均具有有限的坐标。相比之下，依赖于普通持续的方法必须设计专门的工具来处理具有无限坐标的点，或者干脆忽略它们，从而在此过程中丢失信息。因此，扩展持续允许使用通用架构而不受同调维数的限制。实验结果表明，相较于仅使用普通持续，扩展持续在性能上有显著改进。

在实际中，可以使用 C++/Python 的 `Gudhi` 库 Maria et al.[32] 高效地计算扩展持续图。持续图通常采用所谓的 Bottleneck 距离进行比较，定义回顾 3.3。

5.2.2　热核特征函数

热核特征函数 (HKS) 是谱族特征函数的一个例子，也就是说，它们是从图拉普拉斯的谱分解中导出的函数，为图分析提供了有信息的特征。我们首先给出一些基本定义。给定一个图 G，其顶点集为 $V=\{v_1,\cdots,v_n\}$，图的邻接矩阵 A 定义为

$$A:=(\mathbf{1}_{(v_i,v_j)\in E})_{i,j},$$

其中 $\mathbf{1}_{(v_i,v_j)\in E}$ 为指示函数。图的度矩阵 D 是一个对角矩阵，其对角元定义为 $D_{i,i}=\sum_j A_{i,j}$。归一化图拉普拉斯 $L_w=L_w(G)$ 是作用在定义在 G 顶点上的函数空间上的线性算子，其矩阵表示为

$$L_w=I-D^{-\frac{1}{2}}AD^{-\frac{1}{2}}.$$

该矩阵拥有一组正交归一化的特征函数 $\Psi=\{\psi_1,\cdots,\psi_n\}$，其特征值满足

$$0\leqslant\lambda_1\leqslant\cdots\leqslant\lambda_n\leqslant 2.$$

由于正交归一化特征基 Ψ 不是唯一确定的，故不能直接利用特征函数 ψ_i 来比较图。为此，PersLay 架构考虑了 *热核特征函数* (HKS)：

热核特征函数已被用作图匹配问题的特征函数[24]，或用于构造谱描述符以比较图[44]。这些特征函数依赖于 HKS 取值的分布，而不依赖于其全局拓扑结构（全局拓扑结构编码在其扩展持续图中）。为了简洁起见，我们用 $\mathrm{Dg}(G,t)$ 表示利用扩散参数为 t 的 HKS 过滤所得到的图 G 的扩展持续图，即 $\mathrm{Dg}(G,\mathrm{hks}_{G,t})$。

下面的定理证明了这些扩展持续图对于 Bottleneck 距离 d_B 的稳定性。

定理 4：关于图扰动的稳定性. 设 $t\geqslant 0$，令 G 为具有 n 个顶点的图，其拉普拉斯矩阵为 L_w。令 G' 为另一具有 n 个顶点的图，其拉普拉斯矩阵为

$$\tilde{L}_w=L_w+W.$$

则存在一个常数 $C(G,t)>0$（仅依赖于 t 和 L_w 的谱），使得当 $\|W\|_F$（Frobenius 范数）足够小时，有

$$\mathrm{d}_B(\mathrm{Dg}(G,t),\mathrm{Dg}(G',t))\leqslant C(G,t)\|W\|_F. \tag{5-12}$$

证明：感兴趣的读者可以参考 Carrière et al.[10] ■

关于扩散参数 t 的影响. 在图 G 上基于热核特征函数构造扩展持续图需要选取一个特定的 t 值。特别地，理解扩散参数 t 的影响以及如何选择 t 对于统计和学习应用非常重要。首先，我们证明映射 $t\mapsto\mathrm{Dg}(G,t)$ 是 Lipschitz 连续的。

定理 5：关于参数 t 的稳定性. 设 G 为一个图，则映射 $t \mapsto \mathrm{Dg}(G,t)$ 是 2-Lipschitz 连续的，即对于任意 $t,t' \in \mathbb{R}$，

$$\mathrm{d}_B(\mathrm{Dg}(G,t),\mathrm{Dg}(G,t')) \leqslant 2|t-t'|. \tag{5-13}$$

证明：感兴趣的读者可以参考 Carrière et al.[10] ■

由定理 5 可知，持续图对于 t 的选取具有鲁棒性。

5.2.3　PersLay 的构造

在本节中，我们介绍 PersLay 的构造，并且简单陈述该模型的优势。

为了定义针对持续图的层，PersLay 通过定义并实现一系列新的置换不变层，对 Deep Set 架构 Zaheer et al.[47] 进行修改，从而能够恢复并泛化拓扑数据分析中使用的标准向量化方法。

$$\mathrm{PersLay}(\mathrm{Dg}) := \mathrm{op}\left(\{w(p)\cdot\phi(p)\}_{p\in\mathrm{Dg}}\right), \tag{5-14}$$

其中 op 表示任意置换不变的操作（例如最小值、最大值、求和、或第 k 大值等），$w:\mathbb{R}^2\to\mathbb{R}$ 是针对持续图点的权重函数，而 $\phi:\mathbb{R}^2\to\mathbb{R}^q$ 是我们称为“点变换”的表示函数，将持续图中的每个点 (α_b,α_d) 映射为一个向量。

在实际应用中，w 和 ϕ 分别取形式为 w_{θ_1} 和 ϕ_{θ_2}，其中映射 $\theta_1 \mapsto w_{\theta_1}$ 与 $\theta_2 \mapsto \phi_{\theta_2}$ 的梯度均已实现，以便进行反向传播，且参数 θ_1,θ_2 可在训练过程中进行优化。注意到，任何神经网络架构 ρ 都可以与 PersLay 组合，进而构造出适用于持续图的神经网络架构。

下面，我们介绍三种用于构造公式 (5-14) 中参数 ϕ 的点变换函数。

- **三角点变换** $\phi_\Lambda:\mathbb{R}^2\to\mathbb{R}^q$：对于每个点 p，定义

$$\phi_\Lambda(p)=\left[\Lambda_p(t_1),\Lambda_p(t_2),\cdots,\Lambda_p(t_q)\right]^T,$$

其中与点 $p=(x,y)\in\mathbb{R}^2$ 相关的三角函数 Λ_p 定义为

$$\Lambda_p:t\mapsto\max\{0,y-|t-x|\},$$

其中 $q\in\mathbb{N}$ 且 $t_1,\cdots,t_q\in\mathbb{R}$。

- **高斯点变换** $\phi_\Gamma:\mathbb{R}^2\to\mathbb{R}^q$：对于每个点 p，定义

$$\phi_\Gamma(p)=\left[\Gamma_p(t_1),\Gamma_p(t_2),\cdots,\Gamma_p(t_q)\right]^T,$$

其中与点 $p=(x,y)\in\mathbb{R}^2$ 相关的高斯函数 Γ_p 定义为

$$\Gamma_p:t\mapsto\exp\left(-\frac{\|p-t\|_2^2}{2\sigma^2}\right),$$

其中给定 $\sigma>0$，$q\in\mathbb{N}$ 且 $t_1,\cdots,t_q\in\mathbb{R}^2$。

• **直线点变换** $\phi_L\colon \mathbb{R}^2 \to \mathbb{R}^q$：对于每个点 p，定义

$$\phi_L(p) = \left[L_{\Delta_1}(p), L_{\Delta_2}(p), \cdots, L_{\Delta_q}(p)\right]^T,$$

其中与直线 Δ（具有方向向量 $e_\Delta \in \mathbb{R}^2$ 和偏置 $b_\Delta \in \mathbb{R}$）相关的直线函数 L_Δ 定义为

$$L_\Delta\colon p \mapsto \langle p, e_\Delta\rangle + b_\Delta,$$

其中 $q \in \mathbb{N}$ 且 $\Delta_1, \cdots, \Delta_q$ 是平面中的 q 条直线。

公式 (5-14) 的表述非常通用：尽管形式简单，它能够显著地编码大多数经典的持续图向量化方法，只需一小组点变换函数 ϕ，从而可以将 ϕ 的选择视为一种超参数。下面我们展示它如何与文献中流行的持续图向量化和核方法相连接。

• 当使用 $\phi = \phi_\Lambda$（取样点 $t_1, \cdots, t_q \in \mathbb{R}$）、op 取第 k 大值且权重函数 $w = 1$（常数权重函数）时，相当于在 $t_1, \cdots, t_q \in \mathbb{R}$ 上评估第 k 阶 Persistence Landscapes，见 4.1。

• 当使用 $\phi = \phi_\Lambda$（取样点 $t_1, \cdots, t_q \in \mathbb{R}$）、op 取求和且采用任意权重函数 w 时，相当于在 $t_1, \cdots, t_q \in \mathbb{R}$ 上评估加权的 *持续剪影* Chazal et al.[13]。

• 当使用 $\phi = \phi_\Gamma$（取样点 $t_1, \cdots, t_q \in \mathbb{R}^2$）、op 取求和且采用任意权重函数 w 时，相当于在 $t_1, \cdots, t_q \in \mathbb{R}^2$ 上评估加权的 *持续表面* Adams et al.[1]。此外，利用高斯函数来刻画持续图中的点，也是多种持续图核方法中提倡的做法 Le et al.[29]。

• 当使用 $\phi = \phi_{\tilde{\Gamma}}$，其中 $\tilde{\Gamma}$ 是对高斯点变换的修改版本，其定义为：对于任意 $p = (x, y) \in \mathbb{R}^2$，令 $\tilde{p} = p$ 当 $y \leqslant \nu$（对于某个 $\nu > 0$）时，否则取 $\tilde{p} = \left(x, \nu + \log\left(\frac{y}{\nu}\right)\right)$，且 op 取求和、权重函数 $w = 1$，则这对应于文献 Hofer et al.[22] 中提出的方法。

• 当使用 $\phi = \phi_L$（直线 $\Delta_1, \cdots, \Delta_q \in \mathbb{R}^2$）、op 取第 k 大值且权重函数 $w = 1$ 时，其方法与 Carrière et al.[11] 中提倡的方法类似，即先将点投影到直线上排序，再使用 $\|\cdot\|_1$ 范数比较排序结果，最后经过指数化构造出所谓的 Sliced Wasserstein Kernel。

5.2.4　稳定性分析

对于持续图向量化的连续性和稳定性问题，对拓扑数据分析的实践者来说非常重要。在 Hofer et al.[23, Remark 8] 中，作者观察到公式 (5-14)（当 $\mathrm{op} = \mathtt{sum}$ 时）在一般情况下并非连续（相对于常用的持续图度量）。实际上，Divol et al.[18, Prop. 5.1] 表明，对于所有 $s \geqslant 1$，映射

$$\mathrm{Dg} \mapsto \sum_{p \in \mathrm{Dg}} \phi(p)$$

相对于度量 d_s 是连续的，当且仅当 ϕ 的形式为

$$\phi(p) = \varphi(p)\|p - \Delta\|^s,$$

其中 $\|p - \Delta\|$ 表示点 $p \in \mathbb{R}^2$ 到对角线 $\Delta = \{(x, x) : x \in \mathbb{R}\}$ 的距离，且 φ 是一个连续且有界的函数。此外，当 $s = 1$ 且 φ 为 1-Lipschitz 连续时，可以证明该映射实际上是稳定的（参见 Hofer et al.[23, Thm. 12] 与 Divol et al.[18, Prop. 5.2]），具体表述为

$$\left\| \sum_{p \in \mathrm{Dg}_1} \phi(p) - \sum_{p' \in \mathrm{Dg}_2} \phi(p') \right\|_\infty \leqslant d_1(\mathrm{Dg}_1, \mathrm{Dg}_2).$$

特别地，这意味着要求学习到的向量化连续（如 Hofer et al.[23] 所做）就意味着必须约束权重函数对靠近对角线的点取较小值。然而，通常情况下并没有充分理由认为靠近对角线的点比其他点在某个学习任务中更不重要。

本节主要介绍 PersLay 架构理论方面的工作，关于具体的实验读者可以参考 Carrière et al.[10]和本文的章节6.5。

第 6 章　Transformer 的结构

这个章节专门用于介绍 Transformer 架构，感兴趣的读者可以阅读 Vaswani et al.[45]获得更细节的自注意力机制的介绍。

6.1　Transformer 架构的搭建

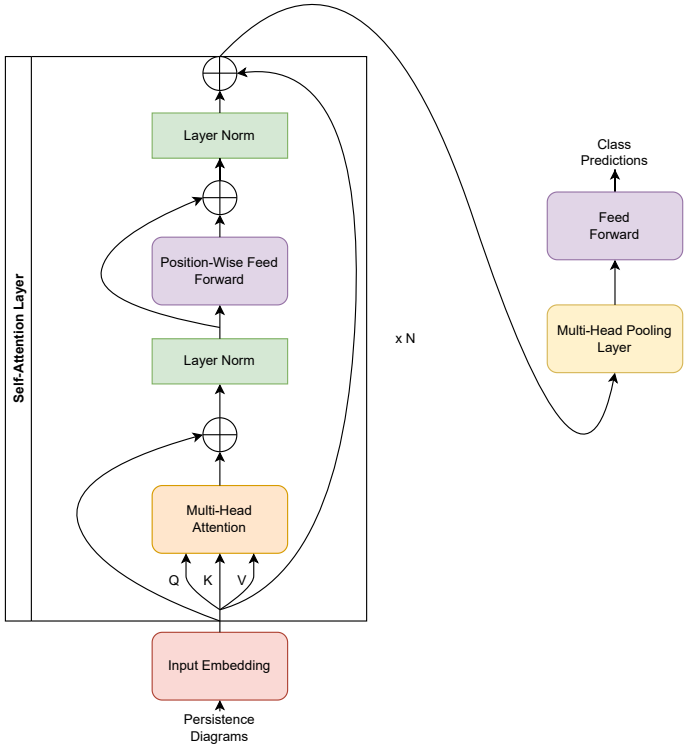

图 6-1　Transformer 架构的示意图

Transformer 架构如图 6-1 所示。它包括一个嵌入层，这是一个可训练的位置全连接层①$\mathbb{R}^2 \to \mathbb{R}^d$，后面是堆叠的自注意力层，包括多头自注意力块和全连接前馈层。最后，多头注意力池化层提供了持续图的向量表示，并且一个全连接神经网络计算最终的类别预测。架构的各个构建模块将在以下部分中详细解释。

自注意力块　自注意力机制旨在建模序列中元素的成对交互。为此，从长度为 N 的 d 维向量序列 $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ 中计算出三个向量族 $\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V} \in \mathbb{R}^{N\times d'}$。这些向量族称为 *query*、*key* 和 *value* 向量。它们是通过可训练矩阵 $\boldsymbol{W}_Q,\boldsymbol{W}_K,\boldsymbol{W}_V \in \mathbb{R}^{d\times d'}$ 对输入

① 位置全连接网络由一个具有两层的全连接神经网络组成，适用于序列中的所有向量。

序列 $\boldsymbol{X}$ 进行线性变换得到的。对于每个 query 向量 $\boldsymbol{Q}_i$，通过计算到所有 key 向量的标量积（除以 $1/\sqrt{d'}$），然后使用 softmax 函数对所有相似度得分进行归一化，以获得所谓的注意力分数

$$\text{AttentionScore}(\boldsymbol{Q}_i, \boldsymbol{K}) = \text{softmax}\left(\frac{\boldsymbol{Q}_i \boldsymbol{K}^T}{\sqrt{d'}}\right) \in \mathbb{R}^N.$$

自注意力块的输出 $\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ 是一个长度为 N 的序列，由 d' 维向量组成，其中第 i 个向量由 value 向量的凸组合给出

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})_i = \sum_{j=1}^{N} \text{AttentionScore}(\boldsymbol{Q}_i, \boldsymbol{K})_j \boldsymbol{V}_j.$$

多头注意力　多头注意力块结合了多个并行的自注意力块，能够联合关注输入序列的不同部分 Vaswani et al.[45]。

为此，query、key 和 value 向量被分成 H 个向量序列 $\boldsymbol{Q}^{(h)}, \boldsymbol{K}^{(h)}, \boldsymbol{V}^{(h)}$，其大小为 $\mathbb{R}^{d'/H}$，其中 H 是注意头的数量，$h = 1, \cdots, H$。这里我们假设 H 可以整除 d'。此外，query 和 key 向量的标量积乘以一个因子 $\sqrt{H/d'}$。对于每个头，计算注意力向量

$$\text{head}_\text{h} = \text{Attention}(\boldsymbol{Q}^{(h)}, \boldsymbol{K}^{(h)}, \boldsymbol{V}^{(h)}) \in \mathbb{R}^{N \times (d'/H)}$$

对于 $h = 1, \cdots, H$，然后组合成一个输出序列

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, \cdots, \text{head}_\text{H}) \boldsymbol{W}^O \in \mathbb{R}^{N \times d}$$

其中 $\boldsymbol{W}^O \in \mathbb{R}^{d' \times d}$ 是一个可训练的线性变换。

位置全连接网络　位置全连接网络是一个具有两层的全连接神经网络，应用于序列中的每个向量。

多头注意力池化　多头注意力池化层是多头注意力层的变体，具有单个可训练的 query 向量 $\boldsymbol{Q} \in \mathbb{R}^{1 \times d'}$ 和输入序列的 key 和 value 向量的线性变换 Lee et al.[30]。输出

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \in \mathbb{R}^d$$

是一个不依赖于输入序列顺序的单一向量。

残差连接　由于持续图中的点可能非常接近，编码器可能难以将它们分开。这种困难导致注意力块的梯度在开始时非常小，这使得训练该架构变得复杂，特别是在有大量自注意力层的情况下[20]。

为了解决这个问题，自注意力层之间添加了残差连接。这种方法可以有效解决梯度消失问题，并允许在堆叠更多的自注意力层的同时显著加快训练速度。

残差连接（Residual Connections）是一种在深度神经网络中广泛使用的结构，其核心思想是通过添加“捷径”或“跳跃连接”，使得信息可以直接在网络层之间传递，从而缓解深层网络训练中的一些常见问题，如梯度消失和梯度爆炸。传统神经网络直接学习从输入到输出的映射 $H(x)$，而残差连接则让网络学习一个残差函数

$$F(x) = H(x) - x.$$

因此，网络的输出可以表示为

$$H(x) = F(x) + x.$$

这种结构假设最优解可能更接近于恒等映射，因此只需要学习输入与期望输出之间的差异。若残差 $F(x)$ 学习到零（即不改变输入），则输出将直接等于输入。

6.2　置换不变性

从全局角度上看，Transformer 架构由*编码器* (ϕ)、*池化层* (p) 和*解码器* (ρ) 组成，计算最终的类别预测。输入时，我们采用持续图中的点及其单热编码的同调维度。

编码器由堆叠的注意力层组成。每个注意力层将向量序列映射到向量序列，并具有置换等变性，即对于任意向量序列 $x_1, \cdots, x_n$ 和集合 $1, \cdots, n$ 的置换 σ，我们有

$$\phi(\{x_{\sigma(1)}, \cdots, x_{\sigma(n)}\}) = \sigma(\phi(\{x_1, \cdots, x_n\})),$$

其中 σ 置换序列 $\phi(x_1, \cdots, x_n)$ 中的向量顺序。注意力池化将向量序列映射到一个单一向量，以置换不变的方式，即输出向量不依赖于序列的顺序。结合编码器的置换等变性和注意力层的置换不变性，我们得到一个置换不变的映射 $\rho \circ p \circ \phi$，将向量序列映射到一个单一向量。

6.3　Transformer 的通用逼近定理

以下技术引理的证明是 Heine 关于紧支持连续函数的均匀连续性的定理的结果。

引理 6.1：设 $X \subset \mathbb{R}^a$ 为紧子集。对于 $k \geqslant 0$，令 $\phi_k, \phi : X \longrightarrow \mathbb{R}^b$ 为连续映射，使得 $K := \overline{\cup_k \phi_k(X) \cup \phi(X)}$ 为紧集，并且令 $\rho_k, \rho : K \longrightarrow \mathbb{R}^c$ 为连续映射。如果

$$\sup_{x \in X} \|\phi_k(x) - \phi(x)\| \underset{k \longrightarrow +\infty}{\longrightarrow} 0 \quad \text{and} \quad \sup_{x \in K} \|\rho_k(x) - \rho(x)\| \underset{k \longrightarrow +\infty}{\longrightarrow} 0,$$

那么对于每个 $n \geqslant 1$：

$$\sup_{(x_1,\ldots,x_n) \in X^n} \left\| \rho_k \left(\sum_{i=1}^{n} \phi_k(x_i) \right) - \rho \left(\sum_{i=1}^{n} \phi(x_i) \right) \right\| \underset{k \longrightarrow +\infty}{\longrightarrow} 0.$$

定理 6.1 (Transformer 的通用逼近定理)：令 $X \subset \mathbb{R}^2$ 为紧集，且令 $PD_n(X)$ 表示包含于 X 且最多含 n 个点的持久性图空间，赋予由任一距离 W^p 或 W_d^p（$p \in \mathbb{R}_{\geqslant 1}$）所诱导的拓扑。

则每个连续函数 $f : PD_n(X) \to \mathbb{R}$ 都可以被具有 ReLU 激活函数且编码器层固定隐含维度为 $2n + 1$ 的 Transformer 模型一致逼近。

证明：根据命题3.2，由距离 W^p 与 W_d^p 所诱导的拓扑一致。因此，我们可以假设 $f : PD_n(X) \to \mathbb{R}$ 是 W^∞（即 Hausdorff）连续的函数。令 $\varepsilon > 0$。则由定理 3.1，存在连续映射

$$\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^{2n+1}, \quad \rho : \mathbb{R}^{2n+1} \to \mathbb{R}$$

使得对任意 $x_1, \cdots, x_n \in X$，都有

$$\left| \rho\Big(\sum_{i=1}^{n} \phi(x_i) \Big) - f\big(\{x_1, \cdots, x_n\} \big) \right| \leqslant \frac{\varepsilon}{2}.$$

保持上一节的符号，我们现在定义一列 Transformer 模型

$$F_k : (\mathbb{R}^2)^n \to \mathbb{R}$$

形式如下：

$$F_k(x_1, \cdots, x_n) = \rho_k \circ p_k \circ \big(\phi_k(x_1), \cdots, \phi_k(x_n) \big).$$

需要注意的是，当注意力得分设为 0 时，注意力层等价于标准的全连接前馈层。因此，将输入嵌入层与自注意力模块及残差连接级联得到的 ϕ_k 在表达能力上严格优于每层具有残差连接和 $2n + 1$ 隐含神经元的前馈网络。根据文献[42]，我们

可以选取 ϕ_k 使得

$$\sup_{x\in X}\|\phi_k(x)-\phi(x)\| \xrightarrow[k\to+\infty]{} 0.$$

同理，当将查询向量设为 0 时，注意力池化计算 $\phi_k(x)$ 的平均值，我们可将其视为在前馈层 ρ_k 归一化之后对向量各坐标元素的求和。因此得到

$$F_k(x_1,\cdots,x_n)=\rho_k\Big(\sum_{i=1}^{n}\phi_k(x_i)\Big).$$

最后，根据全连接层的通用逼近定理[21]，我们可以选取 ρ_k 使得

$$\sup_{x\in X}\|\rho_k(x)-\rho(x)\| \xrightarrow[k\to+\infty]{} 0.$$

由引理 6.1，可得

$$\sup_{(x_1,\cdots,x_n)\in X^n}\Big\|F_k(x_1,\cdots,x_n)-\rho\Big(\sum_{i=1}^{n}\phi(x_i)\Big)\Big\| \xrightarrow[k\to+\infty]{} 0.$$

特别地，存在整数 N，使得对所有 $k\geqslant N$ 都有

$$\sup_{(x_1,\cdots,x_n)\in X^n}\Big\|F_k(x_1,\cdots,x_n)-\rho\Big(\sum_{i=1}^{n}\phi(x_i)\Big)\Big\|\leqslant\frac{\varepsilon}{2}.$$

因此

$$\sup_{(x_1,\cdots,x_n)\in X^n}\Big\|F_N(x_1,\cdots,x_n)-f\big(\{x_1,\cdots,x_n\}\big)\Big\|\leqslant\varepsilon.$$

■

6.4　Transformer 的训练

Training specification　与其他神经网络架构的优化不同，对于 Transformer 架构来说，学习率预热阶段阶段对于取得良好效果至关重要 Popel et al.[37]。实验证明，这一阶段使 Transformer 模型的测试准确率提高了大约 2 个百分点，本篇文章使用了 AdamW 优化器，其权重衰减设置为 1e-4，最大学习率为 1e-3，批量大小为 32，并采用带有硬重启的余弦学习率调度器，设置了 10 个预训练 epoch、3 个周期，总共 1,000 个 epoch。通过网格搜索，我们发现当 Transformer 的解码器部分使用 0.2 的 dropout，而编码器部分不使用 dropout 时，能够获得最佳效果。

Model specification　本篇文章使用了一个带有残差连接的 Transformer 模型，该模型包含 5 个编码器层，隐藏维度为 $d=128$，每层具有 8 个注意力头，采用可训练的层归一化、多头注意力池化以及 GELU 激活函数，同时还使用了一个解码器，

该解码器为一个全连接神经网络，其层结构为 [128,256,256,64,5].

6.5　数据集

6.5.1　ORBIT5k 数据集

为了展示模型的性能，本篇文章考虑了 ORBIT5k 数据集，该数据集是一个用于对持续图向量化方法进行基准测试的标准数据集 Adams et al.[1], Carrière et al.[10], Kim et al.[26]。该数据集由依赖于参数 $\rho > 0$ 的动力系统生成的单位正方形 $[0,1]^2$ 中大小为 1,000 的子集构成。为了生成点云，首先在 $[0,1]^2$ 中随机选择一个初始点 (x_0, y_0)，然后递归地生成一系列点 (x_n, y_n)，其中 $n = 0, 1, \cdots, 999$，其递推公式为

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

对于每个参数 $\rho = 2.5, 3.5, 4.0, 4.1$ 和 4.3，递推公式生成了包含 1,000 个轨道的数据集，最终获得了 5,000 个轨道。点云的形状在很大程度上依赖于参数 ρ。通过计算 0 维和 1 维的 alpha 复形过滤（alpha complex filtration）Carrière et al.[10]，将轨道转换为持续图。分类问题即是从给定的持续图中恢复出参数 $\rho \in 2.5, 3.5, 4.0, 4.1, 4.3$。

除了 ORBIT5k 数据集，本篇文章还考虑了 ORBIT100k 数据集，该数据集中每个参数包含 20,000 个轨道（而不是 1,000 个），总计 100,000 个轨道。两个数据集均按照 70:30 的比例划分为训练集和测试集。

除了传统的核方法之外，本篇文章还将 Transformer 模型的性能与当前最先进的模型——PersLay 模型和 PLLay 模型进行了比较。由于 PLLay 同时使用持续图和原始点云作为输入，本篇文章训练了一个类似于 Transformer 的模型，仅使用原始点云作为输入，获得了 99.1% 的测试准确率，比 PLLay 提高了 4.1 个百分点。本篇文章对表现最佳的架构重复了所有实验五次，并报告了在保留测试集上的平均性能及其标准差。所有模型的详细比较见表 6-2。

6.5.2　MUTAG 图数据集

本篇文章还使用 MUTAG 数据集来评估模型在图分类任务上的性能，并将其与 PersLay 模型进行比较。MUTAG 数据集包含 188 个化合物图，这些化合物被标记为诱变或非诱变。每个图包含 17 到 28 个节点。该数据集已被广泛应用于许多研究，是图分类任务的标准数据集。为了与最新的最先进方法进行直接比较，我们在实验中采用了与 Carrière et al.[10] 中相同的设置。

作为过滤函数，我们采用了定义在图 G 中节点 v 上的热核特征函数（*Heat*

表 6-1　在 10 折交叉验证中计算平均验证准确率及标准偏差。这两个模型仅在扩展持续图上进行训练。

Model	MUTAG
PersLay	85.8%(±1.3)
Transformer	**89.9**%(±2.1)

Kernel Signature）：

$$\mathrm{hks}_t(v) = \sum_{k=1}^{n} \exp(-t\lambda_k)\varphi_k(v)^2,$$

其中 $L = I_n - D^{-\frac{1}{2}}AD^{\frac{1}{2}}$ 为归一化图拉普拉斯矩阵（*normalized graph Laplacian*），其特征函数为 $\varphi_1,\cdots,\varphi_n$，特征值满足 $0 \leqslant \lambda_1 \leqslant \cdots \leqslant \lambda_n \leqslant 2$，$I_n$ 为单位矩阵，A 为图的邻接矩阵，性质可以回顾 5.2.2。使用扩散参数 $t = 10.0$ 以及扩展持续同调 $\mathrm{Ord}_0, \mathrm{Rel}_1, \mathrm{Ext}_0^+, \mathrm{Ext}_1^-$，具体定义可以回顾 5.2.1从而得到六维输入向量，其中前两维是出生时间和死亡时间，后四维为 one-hot 编码的同调类型。

用于 MUTAG 数据集的模型较小，仅有两层，每层大小为 $d = 32$ 且每层有四个注意力头。本篇文章采用 GELU 激活函数和可训练的层归一化，同时使用一个全连接神经网络作为解码器，其层结构为 [64,32,2]。通过实验我发现，将注意力池化与求和池化结合使用（而非单独使用注意力池化）能获得更好的模型性能。具体地，本篇文章对编码器最后一层同时计算了求和池化和注意力池化，并将两者结果拼接在一起。

本篇文章通过 10 折交叉验证对模型进行评估，并报告了在保留验证集上的平均准确率。将模型与仅在扩展持续图上训练的 PersLay 模型进行了比较，以检验模型从扩展持久性图中学习表达性特征的能力。实验结果见表 6-1。

表 6-2　Transformer 模型在以点云（pc），持续图（pd），点云 + 持续图（pc+pd）分别作为输入，ORBIT5k 和 ORBIT100k 分别作为数据集的情况下，取得了比当前最先进的模型更好的效果

Model	ORBIT5k			ORBIT100k
	pc	pd	pd+pc	pd
PointNet Qi et al.[39]	70.8%			
CNN Kim et al.[26]	91.5%			
PLLay + CNN Kim et al.[26]			94.5%	
PLLay + CNN Kim et al.[26]			95.0%	
Persistence Scale Space Kernel Reininghaus et al.[41]		72.38%		
Persistence Weighted Gaussian Kernel Kusano et al.[27]		76.63%		
Sliced Wasserstein Kernel Carrière et al.[11]		83.6%		
Persistence Fisher Kernel Le et al.[29]		85.9%		
PersLay Carrière et al.[10]		87.7%		89.2%
Transformer w/o layer-norm	97.8%	90.4%	98.2%	91.0%
Transformer w/ layer-norm	96.4%	–	96.1%	–
Transformer w/ layer-norm	**99.1**%	**91.2**%	**99.1**%	**92.0**%
+ residual connections	(± 0.3)	(± 0.8)	(± 0.3)	(± 0.4)

第 7 章　针对 Transformer 的可解释性方法

与先前的方法 Carrière et al.[10], Kim et al.[26]相比，本篇文章的模型在神经网络的第一层没有使用人为构造的持续图向量化方式，这使得 Transformer 在第 3.4 节的意义下满足通用逼近性质。这使得通过显著性图（Saliency Maps）这一深度学习领域常用的可解释性方法，可以识别出对分类任务重要的点。在本节中，将通过具体的数据集（ORBIT5k、ORBIT100k 以及一个曲率数据集 Bubenik et al.[7]）验证与通常的观点相反，“短条”（small bars），即在 Bottleneck 距离或 Wasserstein 距离下不稳定的持续图特征，在分类和回归问题中同样是重要的预测因子，这一点在 Bubenik et al.[7] 中已经被首先提出。

7.1　显著性图（Saliency Maps）

对于一个分类问题，Transformer 模型是一个几乎处处可微的函数 $F : \mathcal{D} \to \mathbb{R}^m$，其中 m 是类别的数量，$\mathcal{D}$ 是持续图的空间。它将持续图映射到类别概率的对数。设 d 为要考虑的最大同调维度，设 $x = (x_k)_{k\in 1,\cdots,n} \in (\mathbb{R}^{2+d})^n$ 为一个持续图，并定义 $i(x) = \operatorname{argmax}_j F(x)_j$。其中，$x_k \in \mathbb{R}^{2+d}$ 的前两个坐标是出生时间和死亡时间，最后 d 个坐标是 one-hot 编码的同调维度。显著性图（saliency map）在 x 上由 F 定义如下：

$$\mathcal{S}_F(x) := \left(\left\| \frac{\partial F_{i(x)}(x)}{\partial x_k} \right\|_2 \right)_{k\in\{1,\cdots,n\}} \in \mathbb{R}^n_{\geqslant 0}.$$

因此，$\mathcal{S}_F$ 为持续图中的每个点赋予一个实值，该值表示该点对分类任务的重要性。

7.2　实验

7.2.1　ORBIT5k 数据集

对于如 6.5 所述的 ORBIT5k 分类问题，我们观察到传统拓扑数据分析（TDA）的观点，即持续图中最重要的特征是最持久的特征。和我们的直觉相符，靠近对角线的点几乎具有零显著性值（Saliency value）；见图 7-1。

有趣的是，许多稳定的特征具有较高的显著性图得分，这解释了传统向量化方法的良好表现。然而，在持续图中也存在靠近对角线的点，其显著性图得分较

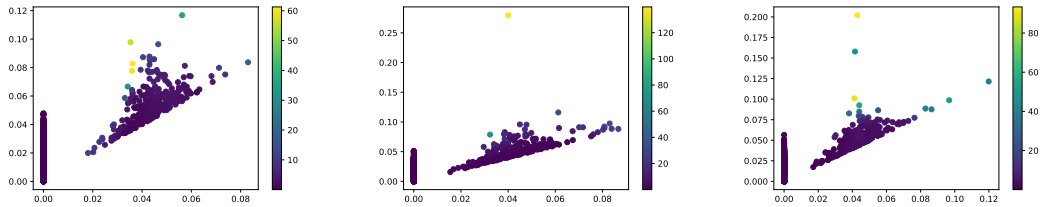

图 7-1　持续图的显著性图（Saliency map），来自 ORBIT5k 数据集，分别对应（从左到右）$\rho = 3.5, 4.1, 4.3$，其中包含 H_0 和 H_1 的特征。H_0 特征对应所有出生值为 0 的点，而 H_1 特征对应所有出生值大于 0 的点。颜色比例表示用于本文描述的分类问题的显著性图得分。

高，即它们对应于“短条”（short bars），但对类别预测影响很大。因此，这些点对于分类任务至关重要，这也解释了在持续图处理任务中为何 Transformer 模型优于以往的方法。此外，持续图的极端点具有较高的显著性图得分，而 H_0 特征的显著性图得分非常低。

为了验证显著性图得分对分类任务的高度相关性，本篇文章对持续图进行筛选，仅保留显著性图得分高于持续图中某一百分位数的点。然后，使用原始模型对筛选后的数据集进行评估。当仅保留显著性图得分高于 80 分位数的点时，在 ORBIT5k 数据集上获得的测试准确率为 91.1%，这一结果与使用所有持续图点的原始模型的测试准确率 91.2% 非常接近。这表明模型中考虑的所有特征对分类任务都很重要。此外，这也表明本篇文章的方法可以用于筛选持续图。例如，仅考虑显著性图得分高于 80 分位数的点，仍然可以在测试数据集上保持良好的结果。数据集 ORBIT5k 在不同阈值下的测试准确率显示在图 7-2 中。

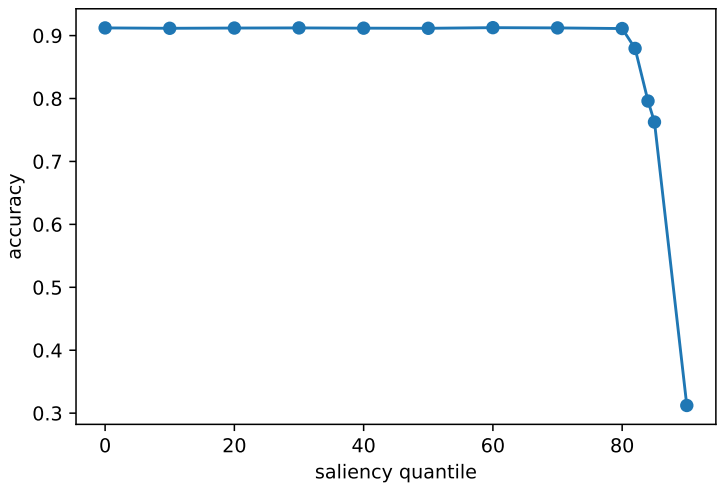

图 7-2　ORBIT5k 数据集在仅考虑显著性图得分高于某个百分位数阈值的点时的测试准确率。当阈值为 80 分位数时，已经能够有效筛选掉持久性图中最无用的点，同时仍然保持与在所有点上训练的模型相近的性能。

本篇文章还在筛选后的持续图（显著性图得分高于 80 分位数）上训练了一个 Transformer 模型，最终获得了 91.2% 的测试准确率。这表明，当滤除具有低显著

性图得分的点时，模型仍然能够与原始模型表现一致。同时，这提高了 Transformer 模型的计算效率，因为该模型的计算复杂度随持续图大小呈二次增长。

此外，本篇文章还训练了一个较小的模型，其自注意力层数、隐藏维度和注意力头数均为原始模型的一半，并提取了该模型的显著性图得分。然后，在使用较小模型得到的显著性图得分中，筛选 80 分位数以上的点，并使用原始模型在筛选后的持久性图上进行训练，最终获得了 91.0% 的测试准确率。这意味着，可以利用一个更简单的模型在训练前高效地筛选特征，同时仍然保持较好的性能。

7.2.2　曲率数据集

在 Bubenik et al.[7] 中，作者考虑了以下回归问题：在一个具有常数高斯曲率 K 且半径为 1 的圆盘上随机采样 1,000 个点，然后利用关于内在度量构建的 Vietoris-Rips 持续图预测 K。他们给出了数学证明，表明这些持续图中的“短条”应当是曲率的良好预测因子，并且能够训练出具有良好 R^2 分数的不同机器学习模型，例如支持向量回归。

本篇文章复现了 Bubenik et al.[7] 的数据集，并使用 H_1 持续图作为预测器，训练了一个回归 Transformer 模型，通过均方误差来预测点云的曲率。结果获得的 R^2 分数为 0.94，而使用支持向量回归得到的 R^2 分数为 0.78 Bubenik et al.[7]。

本篇文章按照前一个数据集中的方法计算了显著性图得分（Saliency map score），参见图 7-3。在持续图中，一些对应于“短条”的点具有非常高的显著性图得分，这验证了 Bubenik et al.[7] 的结果。一个有趣的观察是，显著性图得分似乎也随着出生时间的增加而提高。

在这种情况下，“短条”对于分类结果的重要性比在 ORBIT5k 数据集中更为显著。此外，甚至可以证明，显著性图得分随着与对角线距离的增加而降低，参见图 7-4 和图 7-5。

为了直观展示显著性图得分与与对角线距离之间的依赖关系，本篇文章将每个持续图中的寿命以及显著性图得分归一化到区间 $[0,1]$。然后，我们根据寿命将持久性图中的点分配到区间 $[0,0.1],\cdots,[0.9,1.0]$ 的各个子区间中，并分别计算每个子区间中所有显著性图得分的最大值和总和。接着，在整个测试数据集上对每个子区间的得分进行平均。正如图中所示，显著性图得分随着与对角线距离的增大而降低，这表明“短条”对于估计数据集的曲率具有重要意义。

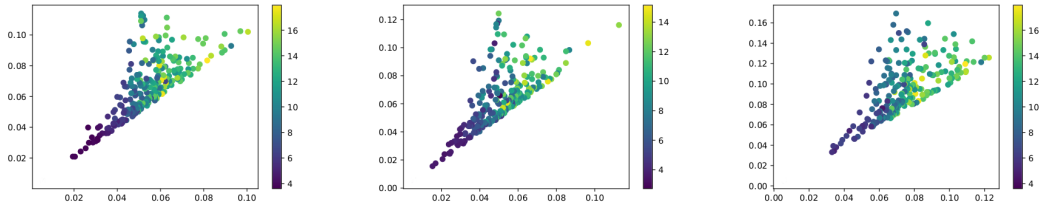

图 7-3　来自 Bubenik et al.[7] 曲率数据集的持续图显著性图（Saliency map），对应的曲率值（从左到右）为 $-1.50, -1.92,\ 0.54$ 和 H_1。

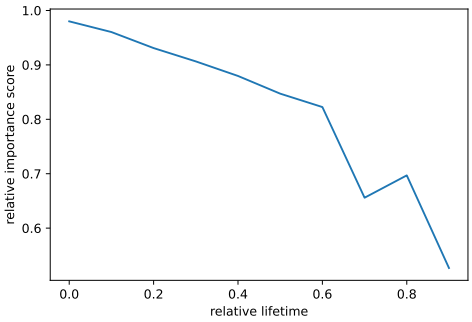

图 7-4　所有测试持续图中每个子区间的相对最大显著性图得分的平均值。

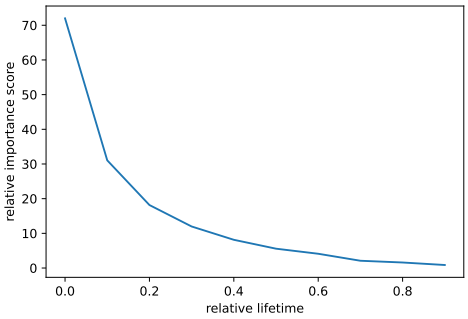

图 7-5　所有测试持续图中每个子区间的相对显著性图得分总和的平均值。

结 论

本文探讨了拓扑数据分析领域中的持续图向量化问题，并提出了基于 Transformer 架构的创新方法，成功地将持续图转化为深度学习模型可以接受的向量表示。我们首先回顾了持续图的基础知识以及其在拓扑数据分析中的应用，进而总结了目前在拓扑深度学习中的两大主要向量化范式——预定向量化方法与可学习向量化方法。

通过引入 Transformer 架构，我们不仅成功地提升了持续图向量化的效果，在经典合成数据集和图数据集上表现出了明显的优势，还证明了该方法在处理持续图时的鲁棒性。更为重要的是，我们提供了一个通用逼近定理，表明 Transformer 能够有效地保持持续图的拓扑信息，这一特性在拓扑数据分析中具有重要意义，能够充分发挥拓扑方法在数据分析中的优势。

此外，我们还对 Transformer 向量化过程进行了可解释性分析，使用显著性图对其进行了深度剖析，并得出了一些启示，为拓扑数据分析与神经网络的结合提供了新的思路。我们的研究表明，Transformer 架构不仅在传统任务中表现出色，而且在拓扑数据分析中同样具有巨大的潜力，开辟了一个全新的研究方向。

未来的工作可以进一步探索如何优化 Transformer 架构与持续图向量化的结合，特别是在处理更为复杂的高维数据时，探索更多可扩展的方法。此外，拓扑数据分析与深度学习的结合仍然是一个新兴的研究领域，未来可针对不同应用场景提出更加细化的特征提取和模型优化策略。

参考文献

[1] ADAMS H, EMERSON T, KIRBY M, et al. Persistence images: A stable vector representation of persistent homology[J/OL]. Journal of Machine Learning Research, 2017, 18: 1-35. arXiv: 1507.06217.

[2] ANAI H, CHAZAL F, GLISSE M, et al. Dtm-based Filtrations[C]//Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019). 2019.

[3] AUKERMAN A, CARRIÈRE M, CHEN C, et al. Persistent Homology Based Characterization of the Breast Cancer Immune Microenvironment: A Feasibility Study[C/OL]//CABELLO S, CHEN D Z. Leibniz International Proceedings in Informatics (LIPIcs): 36th International Symposium on Computational Geometry (SoCG 2020): vol. 164. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020: 11:1-11:20. https://drops.dagstuhl.de/opus/volltexte/2020/12169. DOI: 10.4230/LIPIcs.SoCG.2020.11.

[4] BUBENIK P. Statistical Topological Data Analysis using Persistence Landscapes[J]. Journal of Machine Learning Research, 2015, 16(1): 77-102.

[5] BUBENIK P, CARLSSON G, KIM P T, et al. Statistical Topology via Morse Theory Persistence and Nonparametric Estimation[G]//Contemp. Math. Algebraic Methods in Statistics and Probability II: vol. 516. Providence, RI: Amer. Math. Soc., 2010: 75-92.

[6] BUBENIK P, DŁOTKO P. A Persistence Landscapes Toolbox for Topological Statistics[J]. Journal of Symbolic Computation, 2017, 78: 91-114.

[7] BUBENIK P, HULL M, PATEL D, et al. Persistent homology detects curvature[J/OL]. Inverse Problems, 2020, 36(2): 1-22. arXiv: 1905.13196. DOI: 10.1088/1361-6420/ab4ac0.

[8] CARLSSON G, VEJDEMO-JOHANSSON M. Topological Data Analysis with Applications [M]. Cambridge: Cambridge University Press, 2021.

[9] CARLSSON G, ZOMORODIAN A, COLLINS A, et al. Persistence Barcodes for Shapes[J]. International Journal of Shape Modeling, 2005, 11(2): 149-187.

[10] CARRIÈRE M, CHAZAL F, IKE Y, et al. PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures[C/OL]//CHIAPPA S, CALANDRA R. Proceedings of Machine Learning Research: The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]: vol. 108. PMLR, 2020: 2786-2796. http://proceedings.mlr.press/v108/carriere20a.html.

[11] CARRIÈRE M, CUTURI M, OUDOT S. Sliced Wasserstein kernel for persistence diagrams [J/OL]. 34th International Conference on Machine Learning, ICML 2017, 2017, 2: 1092-1101. arXiv: 1706.03358.

[12] CHAZAL F, COHEN-STEINER D, MÉRIGOT Q. Geometric Inference for Probability Measures[J]. Foundations of Computational Mathematics, 2011, 11(6): 733-751.

[13] CHAZAL F, de SILVA V, GLISSE M, et al. The Structure and Stability of Persistence Modules [M/OL]. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-42545-0.

[14] COHEN-STEINER D, EDELSBRUNNER H, HARER J. Extending Persistence Using Poincaré and Lefschetz Duality[J/OL]. Foundations of Computational Mathematics, 2009, 9: 79-103. DOI: 10.1007/s10208-008-9027-z.

[15] De SILVA V, CARLSSON G. Topological Estimation Using Witness Complexes[C]//Eurographics Symposium on Point-Based Graphics. 2004.

[16] DEY T K, FAN F, WANG Y. Graph Induced Complex on Point Data[C]//Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry (SoCG '13). New York, NY, USA: ACM, 2013: 107-116.

[17] DEY T K, WANG Y. Computational Topology for Data Analysis[M]. Cambridge University Press, 2021.

[18] DIVOL V, LACOMBE T. Understanding the topology and the geometry of the persistence diagram space via optimal partial transport[J]. arXiv preprint arXiv:1901.03048, 2019.

[19] DLOTKO P, RUDKIN S. Visualising the Evolution of English Covid-19 Cases with Topological Data Analysis Ball Mapper[Z/OL]. 2020. arXiv: 2004.03282 [physics.soc-ph].

[20] GOODFELLOW I, BENGIO Y, COURVILLE A. Deep learning[M]. MIT press, 2016.

[21] HAYKIN S. Neural Networks and Learning Machines[M]. Pearson Education India, 2010.

[22] HOFER C, KWITT R, NIETHAMMER M, et al. Deep Learning with Topological Signatures [C/OL]//GUYON I, LUXBURG U V, BENGIO S, et al. Advances in Neural Information Processing Systems: vol. 30. Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/file/883e881bb4d22a7add958f2d6b052c9f-Paper.pdf.

[23] HOFER C D, KWITT R, NIETHAMMER M. Learning Representations of Persistence Barcodes[J/OL]. Journal of Machine Learning Research, 2019, 20(126): 1-45. http://jmlr.org/papers/v20/18-358.html.

[24] HU N, RUSTAMOV R, GUIBAS L. Stable and informative spectral signatures for graph matching[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 2305-2312.

[25] ISLAMBEKOV U, PATHIRANA H, KHORMALI O, et al. A Fast Topological Approach for Predicting Anomalies in Time-Varying Graphs[J/OL]. arXiv preprint arXiv:2305.06523, 2023. https://arxiv.org/abs/2305.06523.

[26] KIM K, KIM J, ZAHEER M, et al. PLLay: Efficient Topological Layer based on Persistent Landscapes[C/OL]//LAROCHELLE H, RANZATO M, HADSELL R, et al. Advances in Neural Information Processing Systems: vol. 33. Curran Associates, Inc., 2020: 15965-15977. https://proceedings.neurips.cc/paper/2020/file/b803a9254688e259cde2ec0361c8abe4-Paper.pdf.

[27] KUSANO G, FUKUMIZU K, HIRAOKA Y. Kernel method for persistence diagrams via kernel embedding and weight factor[J/OL]. Journal of Machine Learning Research, 2018, 18: 1-41. arXiv: 1706.03472.

[28] LACOMBE T, IKE Y, UMEDA Y. Topological Uncertainty: Monitoring trained neural networks through persistence of activation graphs[C]//IJCAI. 2021.

[29] LE T, YAMADA M. Persistence fisher kernel: A Riemannian manifold kernel for persistence diagrams[J/OL]. Advances in Neural Information Processing Systems, 2018, 2018-December(NeurIPS): 10007-10018. arXiv: 1802.03569.

[30] LEE J, LEE Y, KIM J, et al. Set transformer: A framework for attention-based permutation-invariant neural networks[C]//International Conference on Machine Learning. 2019: 3744-3753.

[31] LEE Y, BARTHEL S D, DŁOTKO P, et al. Quantifying similarity of pore-geometry in nanoporous materials[J/OL]. Nature Communications, 2017, 11: 48. https://www.nature.com/articles/ncomms15396/briefing/signup/.

[32] MARIA C, BOISSONNAT J D, GLISSE M, et al. The gudhi library: Simplicial complexes and persistent homology[C]//International congress on mathematical software. 2014: 167-174.

[33] MOROZOV D. Dionysus 2 : a software to compute persistent homology[Z]. Available at https://www.mrzv.org/software/dionysus2/.

[34] NAITZAT G, ZHITNIKOV A, LIM L H. Topology of Deep Neural Networks.[J]. J. Mach. Learn. Res., 2020, 21(184): 1-40.

[35] NAKAMURA T, HIRAOKA Y, HIRATA A, et al. Persistent homology and many-body atomic structure for medium-range order in the glass.[J]. Nanotechnology, 2015, 26 30: 304001.

[36] OTTER N, PORTER M A, TILLMANN U, et al. A Roadmap for the Computation of Persistent Homology[J/OL]. EPJ Data Science, 2017, 6(1): 17. DOI: 10.1140/epjds/s13688-017-0109-3.

[37] POPEL M, BOJAR O. Training Tips for the Transformer Model[J/OL]. The Prague Bulletin of Mathematical Linguistics, 2018, 110(1): 43-70. arXiv: 1804.00247. DOI: 10.2478/pralin-2018-0002.

[38] POULENARD A, SKRABA P, OVSJANIKOV M. Topological Function Optimization for Continuous Shape Matching[J/OL]. Computer Graphics Forum, 2018, 37(5): 13-25. https://doi.org/10.1111/cgf.13487. DOI: 10.1111/cgf.13487.

[39] QI C R, SU H, MO K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 652-660.

[40] REIMANN M W, NOLTE M, SCOLAMIERO M, et al. Cliques of Neurons Bound into Cavities Provide a Missing Link between Structure and Function[J/OL]. Frontiers in Computational Neuroscience, 2017, 11: 48. https://www.frontiersin.org/article/10.3389/fncom.2017.00048. DOI: 10.3389/fncom.2017.00048.

[41] REININGHAUS J, HUBER S, BAUER U, et al. A stable multi-scale kernel for topological machine learning[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 4741-4748.

[42] TABUADA P, GHARESIFARD B. Universal Approximation Power of Deep Neural Networks via Nonlinear Control Theory[J/OL]. CoRR, 2020, abs/2007.06007. arXiv: 2007.06007. https://arxiv.org/abs/2007.06007.

[43] TAUZIN G, LUPO U, TUNSTALL L, et al. giotto-tda: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration[Z/OL]. 2021. arXiv: 2004.02551 [cs.LG].

[44] TSITSULIN A, MOTTIN D, KARRAS P, et al. Netlsd: hearing the shape of a graph[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 2347-2356.

[45] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is All You Need[C]//NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, California, USA: Curran Associates Inc., 2017: 6000-6010.

[46] EL-YAAGOUBI A B, CHUNG M K, OMBAO H. Dynamic Topological Data Analysis: A Novel Fractal Dimension-Based Testing Framework with Application to Brain Signals[J/OL]. Frontiers in Neuroinformatics, 2024, 18: 1387400. DOI: 10.3389/fninf.2024.1387400.

[47] ZAHEER M, KOTTUR S, RAVANBHAKHSH S, et al. Deep sets[J/OL]. Advances in Neural Information Processing Systems, 2017, 2017-December(ii): 3392-3402. arXiv: 1703.06114.

[48] ZIA A, KHAMIS A, NICHOLS J, et al. Topological Deep Learning: A Review of an Emerging Paradigm[J/OL]. Artificial Intelligence Review, 2024, 57(4): 77. DOI: 10.1007/s10462-024-10710-9.

附录 A　代码

下面是使用 Transformer 处理持续图的代码实现：

```
from typing import Callable, Dict

from gdeep.utility.enum_types import AttentionType

from .attention import AttentionBase, InducedAttention,
   ScaledDotProductAttention
from .transformer_config import transformerConfig


class AttentionFactory:
    """
    Factory for creating attention modules.
    """

    attention_modules: Dict[
        AttentionType, Callable[[transformerConfig],
           AttentionBase]
    ] = {}

    def __init__(self):
        # Register the attention layers here:
        self.register_attention__builder(
            AttentionType.DOT_PRODUCT, lambda config:
               ScaledDotProductAttention(config)
        )
        self.register_attention__builder(
            AttentionType.INDUCED_ATTENTION, lambda config:
               InducedAttention(config)
        )

    def register_attention__builder(
        self,
```

```
29          attention_type: AttentionType,
30          attention_module_builder: Callable[[transformerConfig],
                AttentionBase],
31      ) -> None:
32          """
33          Register an attention module.
34          """
35          self.attention_modules[attention_type] =
               attention_module_builder
36
37      def build(self, config: transformerConfig) -> AttentionBase
           :
38          """
39          Create an attention module.
40          """
41          return self.attention_modules[config.attention_type](
               config)
42
43          from typing import Optional
44
45  import torch
46  from torch.nn import Dropout, Linear, Module, ModuleList,
       Sequential
47
48  from .attention_factory import AttentionFactory
49  from .transformer_block import transformerBlock
50  from .transformer_config import transformerConfig
51  from .utility import get_activation_function, get_pooling_layer
52
53  # Type aliases
54  from gdeep.utility.custom_types import Tensor
55
56
57  class transformer(Module):
58      """transformer model as described in the paper: https://
           arxiv.org/abs/2112.15210
59
60      Examples::
```

```
61
62          from gdeep.topology_layers import transformerConfig,
               transformerModel
63          # Initialize the configuration object
64          config = transformerConfig()
65          # Initialize the model
66          model = transformer(config)
67
68      """
69
70      config: transformerConfig
71      embedding_layer: Module
72      transformer_blocks: ModuleList
73      classifier_layer: Module
74      pooling_layer: Module
75
76      def __init__(self, config: transformerConfig):
77          super().__init__()
78          self.config = config
79          self.build_model()
80
81      def build_model(self):
82          """
83          Build the model.
84          """
85          self.embedding_layer = self._get_embedding_layer()
86          self.transformer_blocks = ModuleList(
87              [
88                  self._get_transformer_block()
89                  for _ in range(self.config.num_attention_layers
                       )
90              ]
91          )
92          self.pooling_layer = self._get_pooling_layer()
93          self.classifier_layer = self._get_classifier_layer()
94
95      def _get_embedding_layer(self) -> Module:
96          return Sequential(
```

```
97              Linear(self.config.input_size, self.config.
                   hidden_size),
98              get_activation_function(self.config.hidden_act),
99          )
100
101     def _get_classifier_layer(self) -> Module:
102         return Sequential(
103             Linear(self.config.hidden_size, self.config.
                   hidden_size),
104             get_activation_function(self.config.hidden_act),
105             Dropout(self.config.classifier_dropout_prob),
106             Linear(self.config.hidden_size, self.config.
                   output_size),
107         )
108
109     def _get_transformer_block(self) -> Module:
110         return transformerBlock(self.config)  # type: ignore
111
112     def _get_pooling_layer(self) -> Module:
113         return get_pooling_layer(self.config)
114
115     def forward(
116         self, input_batch: Tensor, attention_mask: Optional[
               Tensor] = None
117     ) -> Tensor:
118         """
119         Forward pass of the model.
120
121         Args:
122             input_batch: The input batch. Of shape (batch_size,
                    sequence_length, 2 + num_homology_types)
123             attention_mask: The attention mask. Of shape (
                   batch_size, sequence_length)
124
125         Returns:
126             The logits of the model. Of shape (batch_size,
                   sequence_length, 1)
127         """
```

```
128         # Initialize the output tensor
129         output = input_batch
130         # Apply the embedding layer
131         output = self.embedding_layer(output)
132         # Apply the attention layers
133         for transformer_block in self.transformer_blocks:
134             output = transformer_block(output, attention_mask)
135         output = self.pooling_layer(output, attention_mask)
136         # Apply the classifier layer
137         output = self.classifier_layer(output)
138         return output
139 
140         from typing import Optional
141 
142 import torch
143 from gdeep.utility.enum_types import LayerNormStyle
144 from torch.nn import LayerNorm, Module, ModuleList
145 
146 from .transformer_config import transformerConfig
147 from .utility import get_attention_layer,
       get_feed_forward_layer
148 
149 # Type aliases
150 from gdeep.utility.custom_types import Tensor
151 
152 
153 class transformerBlock(Module):
154     """
155     A transformer block.
156     """
157 
158     config: transformerConfig
159     attention_layer: Module
160     feed_forward_layer: Module
161     dropout_layer: Module
162     layer_norms: Optional[ModuleList]
163 
164     def __init__(self, config: transformerConfig):
```

```
165         super().__init__()
166         self.config = config
167         self.build_model()
168 
169     def build_model(self):
170         """
171         Build the model.
172         """
173         self.attention_layer = get_attention_layer(self.config)
174         if not self.config.use_attention_only:
175             self.feed_forward_layer = get_feed_forward_layer(
                self.config)
176 
177         if (
178             self.config.layer_norm_style == LayerNormStyle.
                PRE_LAYER_NORMALIZATION
179             or self.config.layer_norm_style == LayerNormStyle.
                POST_LAYER_NORMALIZATION
180         ):
181             self.layer_norms = ModuleList(
182                 [
183                     LayerNorm(self.config.hidden_size, eps=self
                        .config.layer_norm_eps)
184                     for _ in range(2)
185                 ]
186             )
187 
188     def forward(
189         self,  # type: ignore
190         input_batch: Tensor,
191         attention_mask: Optional[Tensor] = None,
192     ) -> Tensor:
193         """
194         Forward pass of the model. We implement different layer
            normalizations in the forward pass,
195         see the paper https://arxiv.org/pdf/2002.04745.pdf for
            details.
196 
```

```
197         Args:
198             input_batch:
199                 The input batch. Of shape (batch_size,
                        sequence_length, 2 + num_homology_types)
200             attention_mask:
201                 The attention mask. Of shape (batch_size,
                        sequence_length)
202 
203         Returns:
204             The logits of the model. Of shape (batch_size,
                    sequence_length, 1)
205         """
206         if self.config.layer_norm_style == LayerNormStyle.
                NO_LAYER_NORMALIZATION:
207             return self._forward_no_layer_norm(input_batch,
                        attention_mask)
208         elif self.config.layer_norm_style == LayerNormStyle.
                PRE_LAYER_NORMALIZATION:
209             return self._forward_pre_layer_norm(input_batch,
                        attention_mask)
210         elif self.config.layer_norm_style == LayerNormStyle.
                POST_LAYER_NORMALIZATION:
211             return self._forward_post_layer_norm(input_batch,
                        attention_mask)
212         else:
213             raise ValueError(f"Unknown layer norm style {self.
                        config.layer_norm_style}")
214 
215     def _forward_no_layer_norm(
216         self, input_batch: Tensor, attention_mask: Optional[
                Tensor] = None
217     ) -> Tensor:
218         """
219         Forward pass of the model without layer normalization.
220 
221         Args:
222             input_batch:
223                 The input batch. Of shape (batch_size,
```

```
                    sequence_length, 2 + num_homology_types)
            attention_mask:
                The attention mask. Of shape (batch_size,
                    sequence_length)

        Returns:
            The logits of the model. Of shape (batch_size,
                sequence_length, 1)
        """
        output = self.attention_layer(input_batch,
            attention_mask) + input_batch  # type: ignore
        if not self.config.use_attention_only:
            output = self.feed_forward_layer(output) + output
        return output

    def _forward_pre_layer_norm(
        self, input_batch: Tensor, attention_mask: Optional[
            Tensor] = None
    ) -> Tensor:
        """
        Forward pass of the model with pre-layer normalization.

        Args:
            input_batch:
                The input batch. Of shape (batch_size,
                    sequence_length, 2 + num_homology_types)
            attention_mask:
                The attention mask. Of shape (batch_size,
                    sequence_length)

        Returns:
            The logits of the model. Of shape (batch_size,
                sequence_length, 1)
        """
        assert self.layer_norms is not None
        normalized = self.layer_norms[0](input_batch)
        output = self.attention_layer(normalized,
            attention_mask) + input_batch
```

```
253         if not self.config.use_attention_only:
254             normalized = self.layer_norms[1](output)
255             output = self.feed_forward_layer(normalized) +
                    output
256         return output
257 
258     def _forward_post_layer_norm(
259         self, input_batch: Tensor, attention_mask: Optional[
                Tensor] = None
260     ) -> Tensor:
261         """
262         Forward pass of the model with post-layer normalization
                .
263 
264         Args:
265             input_batch:
266                 The input batch. Of shape (batch_size,
                        sequence_length, 2 + num_homology_types)
267             attention_mask:
268                 The attention mask. Of shape (batch_size,
                        sequence_length)
269 
270         Returns:
271             The logits of the model. Of shape (batch_size,
                    sequence_length, 1)
272         """
273         assert self.layer_norms is not None
274         output = self.attention_layer(input_batch,
                attention_mask) + input_batch
275         output = self.layer_norms[0](output)
276         if not self.config.use_attention_only:
277             output = self.feed_forward_layer(output) + output
278             output = self.layer_norms[1](output)
279         return output
280 
281         from abc import ABC, abstractmethod
282 from typing import Callable, Dict, Optional
283 from transformers.configuration_utils import PretrainedConfig
```

```
import torch
import torch.nn as nn
from torch.nn import (
    Module,
    MultiheadAttention,
    Linear,
    Sequential,
    LayerNorm,
    Dropout,
    ModuleList,
)

from gdeep.utility.enum_types import (
    ActivationFunction,
    PoolerType,
    AttentionType,
    LayerNormStyle,
)

# Type aliases
from gdeep.utility.custom_types import Tensor


class transformerConfig(PretrainedConfig):
    """
    Configuration class to define a transformer model.

    Examples::

        from gdeep.topological_layers import transformerConfig,
            transformerModel
        # Initialize the configuration object
        config = transformerConfig()
        # Initialize the model
        model = transformer(config)
        # Access the configuration object
        config = model.config

```

```
    """

    input_size: int  # input size of the model
    output_size: int
    hidden_size: int
    num_attention_layers: int
    num_attention_heads: int
    intermediate_size: int
    hidden_act: ActivationFunction
    hidden_dropout_prob: float
    attention_probs_dropout_prob: float
    layer_norm_eps: float
    classifier_dropout_prob: float
    layer_norm_style: LayerNormStyle
    attention_type: AttentionType
    activation_fn: ActivationFunction
    pooler_type: PoolerType
    use_attention_only: bool
    use_skip_connections_for_transformer_blocks: bool

    def __init__(
        self,
        input_size: int = 2 + 4,
        output_size: int = 2,
        hidden_size: int = 32,
        num_attention_layers: int = 2,
        num_attention_heads: int = 4,
        intermediate_size: int = 32,
        hidden_act: ActivationFunction = ActivationFunction.
            GELU,
        hidden_dropout_prob: float = 0.1,
        attention_probs_dropout_prob: float = 0.1,
        layer_norm_eps: float = 1e-12,
        classifier_dropout_prob: float = 0.1,
        use_layer_norm: LayerNormStyle = LayerNormStyle.
            NO_LAYER_NORMALIZATION,
        attention_type: AttentionType = AttentionType.
            DOT_PRODUCT,
```

```
356         pooler_type: PoolerType = PoolerType.ATTENTION,
357         use_attention_only: bool = False,
358         use_skip_connections_for_transformer_blocks=False,
359         **kwargs
360     ):
361         super().__init__(**kwargs)  # type: ignore
362         self.input_size = input_size
363         self.output_size = output_size
364         self.hidden_size = hidden_size
365         self.num_attention_layers = num_attention_layers
366         self.num_attention_heads = num_attention_heads
367         self.intermediate_size = intermediate_size
368         self.hidden_act = hidden_act
369         self.hidden_dropout_prob = hidden_dropout_prob
370         self.attention_probs_dropout_prob =
              attention_probs_dropout_prob
371         self.layer_norm_eps = layer_norm_eps
372         self.classifier_dropout_prob = classifier_dropout_prob
373         self.layer_norm_style = use_layer_norm
374         self.attention_type = attention_type
375         self.pooler_type = pooler_type
376         self.use_attention_only = use_attention_only
377         self.use_skip_connections_for_transformer_blocks = (
378             use_skip_connections_for_transformer_blocks
379         )
380
381         from typing import Optional
382
383 import torch
384 from gdeep.utility.enum_types import (
385     ActivationFunction,
386     AttentionType,
387     LayerNormStyle,
388     PoolerType,
389 )
390 from torch.nn import Module
391
392 from .transformer import transformer
```

```
393 from .transformer_config import transformerConfig
394
395 from gdeep.utility.custom_types import Tensor
396
397
398 class transformerWrapper(Module):
399     """The wrapper for transformer to allow compatibility
400     with the HPO classes.
401     """
402
403     config: transformerConfig
404     model: Module
405
406     def __init__(self, **kwargs):
407         super().__init__()
408         self.config = transformerConfig(**kwargs)
409         self.model = transformer(self.config)
410
411     def forward(self, input: Tensor, attention_mask: Optional[
        Tensor] = None):
412         return self.model(input, attention_mask)
413
414         import torch.nn as nn
415 from torch.nn import Dropout, Linear, Module, Sequential
416
417 from gdeep.topology_layers.pooling_layers import (
418     AttentionPoolingLayer,
419     MaxPoolingLayer,
420     MeanPoolingLayer,
421     SumPoolingLayer,
422 )
423 from gdeep.utility.enum_types import ActivationFunction,
    PoolerType
424 from .attention_factory import AttentionFactory
425 from .transformer_config import transformerConfig
426
427
428 def get_pooling_layer(config: transformerConfig) -> Module:
```

```
429     """
430     Get the pooling layer.
431 
432     Args:
433         config: The configuration of the model.
434 
435     Returns:
436         The pooling layer.
437     """
438     if config.pooler_type is PoolerType.ATTENTION:
439         return AttentionPoolingLayer(config)  # type: ignore
440     elif config.pooler_type is PoolerType.MAX:
441         return MaxPoolingLayer(config)  # type: ignore
442     elif config.pooler_type is PoolerType.MEAN:
443         return MeanPoolingLayer(config)  # type: ignore
444     elif config.pooler_type is PoolerType.SUM:
445         return SumPoolingLayer(config)  # type: ignore
446     else:
447         raise ValueError(f"Pooler type {config.pooler_type} is
            not supported.")
448 
449 
450 def get_feed_forward_layer(config: transformerConfig) -> Module
    :
451     """
452     Get the feed forward layer.
453     """
454     feed_forward_layer = Sequential()
455     feed_forward_layer.add_module(
456         "intermediate", Linear(config.hidden_size, config.
            intermediate_size)
457     )
458     feed_forward_layer.add_module(
459         "activation", get_activation_function(config.hidden_act
            )
460     )
461     feed_forward_layer.add_module("dropout", Dropout(config.
        hidden_dropout_prob))
```

```

    feed_forward_layer.add_module(
        "output", Linear(config.intermediate_size, config.
           hidden_size)
    )
    feed_forward_layer.add_module("dropout", Dropout(config.
       hidden_dropout_prob))
    return feed_forward_layer


def get_attention_layer(config: transformerConfig) -> Module:
    """
    Get the attention layer.
    """
    return AttentionFactory().build(config)  # type: ignore


def get_activation_function(activation_function:
   ActivationFunction) -> Module:
    """
    Get the activation function.
    """
    if activation_function is ActivationFunction.RELU:
        return nn.ReLU()
    elif activation_function is ActivationFunction.GELU:
        return nn.GELU()
    elif activation_function is ActivationFunction.SELU:
        return nn.SELU()
    elif activation_function is ActivationFunction.MISH:
        return nn.Mish()
    else:
        raise ValueError("Unknown activation function.")
```

代码 A-1　Python 代码示例

致　谢

首先，我想要感谢我的导师，朱一飞教授，他在我的学习过程中提供了很多指导，使得我攻克了很多难关，并且也是朱老师向我系统介绍了拓扑数据分析领域的发展情况，使得我有机会在该领域进行深入学习和研究。我也非常感谢课题组的同学给予的帮助，与他们在讨论班上的交流使我受益匪浅，交流是数学学习中不可或缺的一步，我在与他们的交流中逐步成长，最终有了成果。

其次，我要感谢我的家人，他们一直以来给予的精神和物质支持，使得我能够完成学业，成为一个对社会有帮助的人。我想要感谢一直以来愿意倾听我的困难倾诉，愿意分享我的快乐的朋友们，你们的支持是我前进的动力。

最后，我想要感谢我的母校——南方科技大学，在这里的六年我学会了很多东西，明白了很多道理，真正做好了迈入社会的准备。

个人简历、在学期间完成的相关学术成果

个人简历

2001 年 01 月 01 日出生于江苏扬州。

2019 年 09 月考入南方科技大学数学系数学专业，2023 年 06 月本科毕业并获得理学学士学位。

2023 年 09 月——2025 年 06 月，在南方科技大学理学院数学系学习并攻读理学硕士学位。