分类号 _____ 编号_____

U D C_____ 密级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：拓扑数据分析和拓扑深度学习在图像数据中的应用

姓　　名：张海宇

学　　号：12011936

系　　别：数学系

专　　业：数学与应用数学

指导教师：朱一飞

2024 年 4 月 29 日

CLC _____ Number _____

UDC _____ Available for reference ☐Yes ☐No

SUSTech Southern University of Science and Technology

Undergraduate Thesis

Thesis Title: **Topological Data Analysis and Topological Deep Learning: with Applications to Image Data**

Student Name: **Haiyu Zhang**

Student ID: **12011936**

Department: **Department of Mathematics**

Program: **Mathematics and applied mathematics**

Thesis Advisor: **Yifei Zhu**

Date: April 29, 2024

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2. 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名:

_____ 年____ 月____ 日

COMMITMENT OF HONESTY

1. I solemnly promise that the paper presented comes from my independent research work under my supervisor's supervision. All statistics and images are real and reliable.

2. Except for the annotated reference, the paper contents no other published work or achievement by person or group. All people making important contributions to the study of the paper have been indicated clearly in the paper.

3. I promise that I did not plagiarize other people's research achievement or forge related data in the process of designing topic and research content.

4. If there is violation of any intellectual property right, I will take legal responsibility myself.

Signature:

Date:

拓扑数据分析和拓扑深度学习在图像数据中的应用

张海宇

（数学系　指导教师：朱一飞）

[摘要]：拓扑数据分析是一种通过对数据构建单纯复合体并运用持续同调方法来研究数据拓扑特征与结构的方法，近年来逐渐被应用于深度学习领域。本文首先回顾了 2008 年 Gunnar Carlsson 等人运用拓扑数据分析方法从自然图像局部区域像素空间中提取出具有克莱因瓶拓扑结构的子流形的相关研究，同时复现了拓扑数据分析应用于分析训练自然图像数据的卷积神经网络的权重空间的拓扑结构的相关工作。结果表明神经网络的卷积层在训练图像数据的过程学到了一些简单拓扑结构（圆，克莱因瓶等）并且该结构与自然图像像素空间的结构有一定相似性。这提高了卷积神经网络的可解释性，同时也对神经网络的优化提供了思路：将拓扑特征嵌入神经网络。本文解释了神经网络结构与数据特征分布的这种相似性并介绍了 2023 年 Ephy R. Love 等人基于该思路提出的拓扑卷积神经网络。本文从理论角度解释了拓扑卷积神经网络的构造原理，并从训练效果和泛化性质两个角度结合实验结果将其与传统神经网络进行对比，展示了拓扑卷积神经网络的优越性。

[关键词]：拓扑数据分析；持续同调；自然图像；卷积神经网络；可解释性

[**ABSTRACT**]: Topological data analysis is a method that studies the topological features and structures of data by constructing simplicial complexes and applying persistent homology methods, which has been increasingly applied in the field of deep learning in recent years. In this paper, we review the groundbreaking work of topological data analysis applied on the space of natural images which extract a submanifold with topology of the Klein bottle by Gunnar Carlsson et al. in 2008. Then we reproduce the analysis on weight space from convolutional neural networks trained on image data. The results indicates that while training, the weight space of convolutional neural network learns simple structures (circle or Klein bottle) consistent with feature space of image data, which enhance the interpretability of neural network and provide inspiration that we can embed topological features to neural networks to improve it. We interpret the consistence between structure of neural network and local patch distribution of natural images and introduce the topological convolutional neural network proposed by Ephy R. Love et al. in 2023. The construction principles of the topological convolutional neural network are theoretically explained in this paper, and its performance and generalization properties are compared with traditional neural networks based on experimental results to show the superiority of topological convolutional neural networks.

[**Key words**]: Topological Data Analysis, Persistent Homology, Natural Image, Convolutional Neural Network, Interpretability

Contents

1. Introduction

Topological data analysis (TDA) is a branch of geometric data analysis based on algebraic topology for imposing and analyzing the geometric structure of data. Concepts in algebraic topology, such as simplicial complex and homology, provide a theoretical foundation for TDA. Combining with some statistical and algorithmic methods, TDA enables remarkable progress in the field of geometric data analysis. The significant advantage of TDA, compared with other statistical techniques, is that it could capture nonlinear structures within the data using topological techniques. Currently, topological data analysis is being applied in multiple specific fields such as neuroscience[1] and material science[2].

Deep learning is a broader field of artificial intelligence (AI), which involves the use of neural networks to model and solve complex problems. These deep neural networks are inspired by the structure and function of the human brain, where connected neurons collaborate to analyze information. Deep learning has shown remarkable success in various applications, leading to breakthroughs in areas like computer vision[3], speech recognition, autonomous vehicles, and many others.

Convolutional neural net work (CNNs) is a widely applied type of neural network characterized by locality and weight sharing. Convolutional layers apply filters or kernels to input data, capturing local patterns and features. However, the development of Convolutional Neural Networks (CNNs) also faces some challenges. On one hand, the interpretability of neural networks is relatively low and we do not have a clear understanding of how they work. On the other hand, there is some technical challenges, for example, how to effectively control model complexity to prevent overfitting and to improve the training performance and generalization of the model.

Rencently, there has been a trend to put TDA on datasets and embed these topological structures into a deep learning scheme and include topological features in neural networks. Specific work can be found in Zhao et al.[4] and Oballe et al.[5].

This paper reviews the ground-breaking work about local behavior of the space of nat-

ural images[6] which apply topological data analysis on natural images and extract a high density 2-dimensional Klein bottle from the space of high-contrast image patches.Then we use persistent homology to reproduce the topological data analysis work[7] showing that CNN learns simple structures while training on image data. Based on these discoveries, there is an improvement on the structure of Convolutional Neural Networks (CNNs)[8] by redefining the architecture of convolutional layers and the design of kernels which aims to enhance the efficiency and effectiveness of feature extraction for image classification.

2. Theoretical Foundation and Techniques of TDA

In this chapter, We will review some basic definitions, models, theories[9] and techniques in topological data analysis.

2.1 Complexes for Point Cloud

In algebraic topology, simplicial complexes is a kind of specialized class of complexes consist of blocks called simplices.

Definition 2.1 ***(Simplex)*** *Given $k+1$ distinct points $\{v_i\}_{i=0}^k$ in $\mathbb{R}^n$, the set $\{v_i\}_{i=0}^k$ is* ***affinely independent*** *if the set $\{v_i - v_0\}_{i=1}^k$ is linearly independent. The* ***simplex*** *(plural:**simplices**) spanned by the affinely independent set $\{v_i\}_{i=0}^k$, denoted by $[v_0, ..., v_k]$, is the set*

$$[v_0, ..., v_k] = \left\{ \sum_{i=0}^{k} t_i v_i : t_i \geq 0 \ \& \sum_{i=0}^{k} t_i = 1 \right\}$$

Definition 2.2 ***(Simplicial Complex)*** *A simplicial complex is a locally finite collection K of simplices satisfying that every face of one simplex is in K and the intersection of two simplices is a face of each or empty.*

Data sets can be represented by a set of points, located by the coordinates of different features. The set of points is called a **point cloud**. The fundamental idea of TDA is to assign a simplicial complex to the point cloud and study the topology of the simplicial complex. However, we need to pay attention to the consistency that if the topological variants of the point cloud agree with the simplicial complex with high possibility. The following are three

natural methods to construct a simplicial complex inside a point cloud.

Definition 2.3 *(**Čech complex**) Given a point cloud $Z = \{x_\alpha\} \subseteq \mathbb{R}^n$, the **Čech complex** $\mathcal{C}_\epsilon$ is the abstract simplicial complex consisting of k-simplices whose vertices are unordered $(k+1)$-tuples of points $\{x_\alpha\}_0^k$ if and only if the closed ϵ-ball neighborhoods of these points have at least one point in common.*

Definition 2.4 *(**Vietoris-Rips complex**) Given a point cloud $Z = \{x_\alpha\} \subseteq \mathbb{R}^n$, the **Vietoris-Rips complex** $\mathcal{R}_\epsilon$ is the abstract simplicial complex consisting of k-simplices whose vertices are unordered $(k+1)$-tuples of points $\{x_\alpha\}_0^k$ if and only if*

$$d(z_i, z_j) \leq \epsilon$$

Definition 2.5 *(**Alpha complex**) Given a point cloud $Z = \{x_\alpha\} \subseteq \mathbb{R}^n$, the **Voronoi cell** of x_α is*

$$V(x_\alpha) = \{p \in \mathbb{R}^n | d(p, x_\alpha) \leq d(p, z), z \in Z\}$$

*the **alpha complex** $\mathcal{A}_\epsilon$ is the abstract simplicial complex consisting of k-simplices whose vertices are unordered $(k+1)$-tuples of points $\{x_\alpha\}_0^k$ if and only if the intersection of closed ϵ-ball neighborhood and closed Voronoi cell of these points have at least one point in common[10].*

Definition 2.6 *(**Witness complex**) Given a finite multiset (or bag) A of real numbers [see Hein 2003, Section 1.2.4, or Monro 1987 for a discussion of multisets], a submultiset $S \in A$ is ϵ- minimizing in A if for all $a \in A \setminus S$, we have $a + \epsilon \geq s$ for all $s \in S$.*

Given a point cloud $\mathcal{L} \subseteq \mathbb{R}^n$, the simplex $\sigma = \{l_0, \ldots, l_k\}$ exists if there is an $x \in \mathbb{R}^n$ such that the set $\{d(l_0, x), \ldots, d(l_k, x)\}$ is 0-minimizing in the set $\{d(l, x)\}_{l \in \mathcal{L}}$. The **witness complex** consists of all the faces of these simplexes.

More generally, given a metric space X and $\mathcal{L} \subseteq X$, for a submultiset $\sigma = \{l_{i_0}, \ldots, l_{i_s}\} \subseteq \mathcal{L}$, an element $x \in X$ is said to be an ϵ-witness for σ if $\{d(l_{i_0}, x) \ldots, d(l_{i_s}, x)\}$ is ϵ-minimizing in $\{d(l, x)\}_{l \in \mathcal{L}}$. The witness complex $W_\infty(X, \mathcal{L}, \epsilon)$ is defined as the simplicial

complex with vertex set $\mathcal{L}$ for which $\sigma \subseteq \mathcal{L}$ is a simplex if and only if σ and all its faces have ϵ-witnesses.

Remark 2.7 *Let X be a metric space. When we consider a subset $Z \subseteq X$ and the Voronoi cell of each poins in Z, Z is the "landmark set" for building a simplicial complex that could reflect the geometric information of X.*

2.2 Persistent Homology

Definition 2.8 ***(Filtered Simplicial Complex)*** *Let $\{\Sigma_\epsilon\}$ be a family of simplicial complexes with the parameter ϵ, for example, the Vietoris-Rips complex with distance parameter ϵ. Σ_ϵ is functorial since if $\epsilon < \epsilon'$ then $\Sigma_\epsilon \subseteq \Sigma_{\epsilon'}$. Hence for some sufficiently dense set $t_1, ...t_n$ of parameters, a filtration of Σ_{t_n} is given by the sequence $\emptyset \subseteq \Sigma_{t_1} \subseteq \cdots \subseteq \Sigma_{t_n}$.*

Definition 2.9 ***(Persistent Chain Complex)*** *A persistence chain complex C is a sequence of chain complexes $\mathsf{C} = (C_*^i)$ together with chain maps $x : C_*^i \to C_*^{i+1}$.*

Definition 2.10 ***(Persistent Homology)*** *Given a sequence of filtered simplicial complex $\{\Sigma_{t_i}\}$, we can construct a persistent chain complex $C_*(\Sigma_*)$ by setting $C_*^i(\Sigma_*) = C_*(\Sigma_{t_i})$ and the chain map $x : C_*^i(\Sigma_*) \hookrightarrow C_*^{i+1}(\Sigma_*)$. For $i < j$, the (i, j)-persistent homology of C, denoted $H_*^{i\to j}(\mathsf{C})$, is defined to be the image of the induced homomorphism $x_* : H_*(C_*^i(\Sigma_*)) \to H_*(C_*^j(\Sigma_*))$.*

Theorem 2.11 *For a finite persistence module $\mathcal{C}$ with field F coefficients,*

$$H_*(\mathsf{C}; F) \cong \bigoplus_i x^i \cdot F[x] \oplus \left(\bigoplus_j x^{r_j} \cdot (F[x]/(x^{s_j} \cdot F[x])) \right).$$

Definition 2.12 ***(Betti Number)*** *Given a topological space X, for a field F, the dimension of $H_k(X; F)$, if it is finite, is defined as the k-th Betti number with coefficients in F, written as $\beta_k(X, F)$.*

Definition 2.13 ***(Barcode)*** *A barcode visually represents the persistent homology group $H_k(\mathsf{C}; F)$ as a set of horizontal line segments in a plane. In this representation, the horizontal axis corresponds to the parameter, while the vertical axis signifies an arbitrary ordering*

of homology generators.The dimension of the persistent homology group $H_k^{i\to j}(\mathsf{C}; F)$ *corresponds to the count of intervals within the barcode of* $H_k(\mathsf{C}; F)$ *that cover the parameter range* $[i, j]$. *Specifically,* $H_*(C_*^i; F)$ *equals the number of intervals that encompass the value* i.

2.3 Mapper

Mapper is a particular technique of TDA which aims to obtain higher level understanding of data structures based on certain features and construct corresponding complex. It combines dimensionality reduction, clustering and some graph networks techniuqes.

2.3.1 Process of Mapper

Given a datasets of points, the process of mapper contains four steps:

• **Dimensionality Reduction:** Use a filter function f based on PCA or other density-based methods to map the data into a lower-dimensional space.

• **Cover of Projected Space:** Construct a cover $(U_i)_{i\in I}$ of the projected space which satisfies that overlapping intervals have constant length.

• **Clustering:** For each interval U_i, cluster the points in $f^{-1}(U_i)$ into sets $S_{i,1}, \ldots \square S_{i,k_i}$

• **Graph Construction:** Construct the graph whose vertices are the cluster sets and if two clusters have some points in common, an edge between them exists.

2.3.2 Parameters of Coverings, Lens and Clustering

There are four core parameters during the construction of mapper complex.

The first two parameters are defined to construct the covering.

• **Resolution**: the number of open sets in the range.

• **Gain**: the amount of overlap of these intervals

• **Lens**: the filter function f for dimensionality reduction. There are many choices of lenses from different perspective of data based on different purposes . We could use some data driven features, statistics methods like mean, variance and density, geometry properties like centrality and curvature and machine learning algorithm like PCA.

• **Clustering**: The clustering algorithm depends on the metric of the data space. Common clustering methods include KNN clustering, DBSCAN clustering.

3. Algebraic Formalism of Neural Network Architectures

In this section, we introduce a mathematical formalism of neural network architectures based on metric and graph information of the feature space of a data set which is set up by Gunnar Carlsson[11]. It provides a interpretable, systematic and abstract way to describe and construct various neural networks which include metric information about the feature space. This section will provide corresponding examples for these concepts.

Definition 3.1 ***(Feed Forward System)***

*A **Feed Forward System** is represented as a directed graph Γ with a vertex set $V(\Gamma)$. It satisfies the following properties:*

1. *$V(\Gamma)$ is decomposed into layers as the disjoint union*

$$V(\Gamma) = V_0(\Gamma) \sqcup V_1(\Gamma) \sqcup \cdots \sqcup V_r(\Gamma).$$

2. *If $v \in V_i(\Gamma)$, then every edge (v, w) in Γ leads to a vertex $w \in V_{i+1}(\Gamma)$.*

3. *For any non-initial vertex $w \in V_i(\Gamma)$ with $i > 0$, there exists at least one vertex $v \in V_{i-1}(\Gamma)$ such that (v, w) is an edge in Γ.*

4. *For each vertex v of Γ , denote the set of all vertices w of Γ such that (v, w) (respectively (w, v)) is an edge of Γ by $\Gamma(v)$ (respectively $\Gamma^{-1}(v)$).*

*As for neural networks, the **Feed Forward Neural Networks (FFNN)** can be regarded as a feed forward system above. The nodes in $V(\Gamma)$ are collectively referred to as **nodes**. For each index i, $V_i(\Gamma)$ consists of the nodes in layer-i. The first layer $V_0(\Gamma)$ contains the **input nodes** of the neural network. The final layer $V_r(\Gamma)$ contains the **output nodes**.*

A **correspondence** between two sets X and Y, defined as a subset $C \subset X \times Y$, describes the connections between elements in the two set with

$$C(x) := \{y \in Y \mid (x, y) \in C\} \subset Y,$$
$$C^{-1}(y) := \{x \in X \mid (x, y) \in C\} \subset X.$$

Example 3.2 *Given an FFNN represented by a directed acyclic graph* Γ *with vertex set* $V(\Gamma)$*, the correspondence in the FFNN is called the* ***graphical correspondence****, defined by* $(v, v') \in C_\Gamma$ *if* (v, v') *is an edge in* Γ.

Example 3.3 *Given two sets* X *and* Y*, the* ***complete correspondence*** *is defined by* $C_c(X, Y) = X \times Y$*. If* X *and* Y *are two consecutive layers in an FFNN, denoted by* V_i *and* V_{i+1}*, then we call* V_{i+1} *a* ***fully connected layer****.*

Example 3.4 *Given a map of two sets* $f : X \to Y$*, the* ***functional correspondence*** $C_f : X \to Y$ *attached to f is defined to be* $\{(x, f(x))|x \in X\}$.

Example 3.5 *Given a metric space* X *with the distance function* d *and a threshold* $r \geq 0$*, we define the* ***metric correspondence with threshold*** r *to be* $C_d(r) : X \to X$ *such that* $C_d(r)(x) = \{x'|d(x, x') \geq r\}$.

Definition 3.6 ***(Generator)*** *Let* I_r *be the totally ordered set {0, 1,...,r} regarded as a category. A* ***generator*** *for an r-layer feed-forward system is a functor F from the category* I_r *to the category* Cor *of finite sets and the correspondences between them. The associated feed-forward system has* $\bigsqcup F(i)$ *as its set of vertices and there is a connection from* $v \in F(i)$ *to* $w \in F(i + 1)$ *if and only if* $(v, w) \in F(i \to i + 1)$.

Definition 3.7 ***(Activator)*** *An* ***activator*** *is a triple* (μ, S, f)*, where* μ *is a commutative semigroup structure (binary operation which is commutative and associative) on* $\mathbb{R}$*, S is a multiplicative semigroup of* $\mathbb{R}$ *and* f *is a function from* $\mathbb{R}$ *to* $\mathbb{R}$ *called* ***cutoff function****. For a feed forward structure* Γ*, an* ***activation system*** *is a selection of an activator* (μ_v, S_v, f_v) *for each non-initial vertex* v *of* Γ*. A* ***coefficient system*** *for a feed-forward system* Γ *and activation system* (μ_v, S_v, f_v) *is a choice of an element* $\lambda_{(u,v)} \in S_v$ *for each edge* (u, v) *of* Γ.

An FFNN equipped with activation system is called **a neural net**. In an FFNN, the semigroup structures μ is usually the additive structure which means that the value of one node on a layer is the sum of values of connected nodes on the former layer. Other options like the commutative operation $(x, y) \to max(x, y)$ can also be applied in some layers of neural networks, for example, the pooling layer. The multiplicative semigroup S is a set containing the weights in an FFNN and the cutoff function f maybe chosen to be identity, but uasually is a continuous function like ReLU.

Now we can construct functions on the input data and consider about the choice of loss function. Given an FFNN denoted by Γ, equipped with a coefficient system $\{\lambda_(u, v)\}$ and an activation system (μ_v, S_v, f_v). For $1 \leq i \leq r$, let W_i be the real vector space of functions mapping $V_i(\Gamma)$ to $\mathbb{R}$. We regard the input data of a layer V_{i-1} as a function $g : V_{i-1} \to \mathbb{R}$. Then we could define a function $\phi_i : W_{i-1} \to W_i$ by:

$$\phi_i(g)(v) = f_v \left(\sum_{u \in \Gamma^{-1}(v)} \lambda_{u,v} \cdot g(u) \right).$$

Thus the function Φ from the input set W_0 to the output set W_r is defined by the composite

$$\Phi = \phi_r \circ \phi_{r-1} \circ \ldots \circ \phi_1$$

Deep learning aims to find a function f which best approximates the function Φ. If Φ can be viewed as a continous function, then we could define the L_2 distance of f from Φ to be the loss function. If the output function is categorical, we could define a continous function on the region of a simplex spanned by its vertices which represents the categories. A typical example is predict probability by softmax function and measure the loss with the cross-entropy loss.

Definition 3.8 ***(Convolutional Structure)*** *Given a feed-forward system Γ and one layer $V_i(\Gamma)$, we define an equivalence relation by a pair $(\cong, \varphi)$ where $\cong$ is defined on the set*

of vertices of layer $V_i(\Gamma)$ and φ is a bijection for any $v \cong w$ in $V_i(\Gamma)$:

$$\varphi_{(v,w)} : \Gamma^{-1}(v) \to \Gamma^{-1}(w)$$

which satisfies $\varphi_{(v,w)} = \varphi^{-1}_{(w,v)}$ and $\varphi_{(w,v)} = \varphi_{(w,u)} \circ \varphi_{(u,v)}$. An activation system of Γ is ***adapted to the convolutional structure*** *on a layer $V_i(\Gamma)$ if $v \cong w$ indicates that $(\mu_v, S_v, f_v) = (\mu_w, S_w, f_w)$. Similarly, a coefficient system $\lambda_{(v,w)}$ of a neural net $(\Gamma, \mathcal{A})$ is* ***adapted to the convolutional structure*** *if it satisfies the compatibility requirement that if $v \cong w$, then*

$$\lambda_(u, v) = \lambda_{(\varphi_{(v,w)}(u),w)}$$

for all $u \in \Gamma^{-1}(v)$.

Example 3.9 *Consider that a group G and the free group action on layer $V_{i-1}(\Gamma)$ and $V_{i-1}(\Gamma)$ and suppose for any $v \in V_{i-1}(\Gamma)$ and $w \in V_i(\Gamma)$, (v, w) is an edge in Γ IFF for all $g \in G$, (gv, gw) is an edge. We define an equivalence relation $\cong$ on $V_i(\Gamma)$ by letting $v \cong w$ IFF there exists an element $g \in G$ such that $gv = w$. Since the group action is free, v and w could determine a unique g. Also notice that the group preserves the structure in Γ, g will carry $\Gamma^{-1}(v)$ to $\Gamma^{-1}(w)$. The structure defined above is called* ***Cayley structures****.*

A **Convolutional layer** in an FFNN is a layer equipped with the convolutional structure which means that the weights exhibits translational invariance.

In this article, the main subject of our study is image data. For digital images, we model it as grids indexed by $\mathbb{Z}^2$. Then the grid of a grayscale image can be represented by triples $(x, y, i) : x, y \in \mathbb{Z}$ and $i \in [0, 1]$ is the intensity at the point (x, y). Equivalently, an image can be specified by a map $\mathbb{Z}^2 \to [0, 1]$. For videos, we model it by grids indexed by $\mathbb{Z}^3 = \mathbb{Z}^2 \times \mathbb{Z}$, where the third dimension denotes time. Generally, grids are indexed by $\mathbb{Z}^N$ for any positive integer N.

In convolutional neural networks, the nodes are filters with kernels of the same size. Let χ be a finite index set of the space of the filters in a convolutional layer and a grid denotes

kernels in the filter. The space of the nodes in a convolutional layer (also in pooling layer) is denoted by $V_i = \chi \times \mathbb{Z}^N$. To describe the layers in an FFNN in terms of generators, we denote the functor by $F = F^c \times F^s$ where F^c is a **complete generater** defined on the filter space and F^s is called a **structural generator** defined on the grids. For example, we can construct a convolutional layer with notations as follows:

$$\chi(1) \times \mathbb{Z}^2 \xrightarrow{C_c \times C_d(1)} \chi(64) \times \mathbb{Z}^2$$

where $F^c = C_c$ and $F^s = C_d(1)$ and the number of inputs channel is 1 and outputs channel is 64.

Usually, we denote the pooling layer by a generator $\pi^2(m, n, N)$ where N is the stride and $n - m + 1$ is the width. Here is an example:

$$\chi(64) \times \mathbb{Z}^2 \xrightarrow{C_c \times \pi^2(0,1,2)} \chi(64) \times \mathbb{Z}^2$$

4. Local Behavior of Feature Space in Natural Images

Natural images statistics is a widely researched field which can be applied in many areas such as computer vision and neuroscience. Features of natural images usually distribute on high contrast patches. TDA can be appiled to analyze the local behavior of feature space of natural images by searching for low-dimensional manifolds within the space of 3 by 3 high contrast local patches[6]. We will introduce the work and findings in this section.

4.1 Data Processing

The dataset consisting of 4.2×10^6 3 by 3 high contrast local patches is described by Lee Pederson and Mumford[12] which is obtained from a subset of the collection of still images gathered by van Hateren and van der Schaaf[13].For each image, extract 5000 3 by 3 patches randomly from the image and regard them as 9-vectors. Then apply Discrete Cosine Transform (DCT) to obtain patches with the top 20 percent contrast. The space of 9-vectors obtained from the steps above is denoted by $\mathcal{M}$.

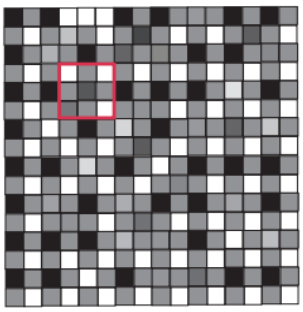


Figure 1 Extrating the local patch from the natural images

4.2 Three Circle Model

To capture the significant topological features, first we need to find the high-density regions, considered as the core set of the space. This technique is called density filtration. Define the density function with a parameter k:

$$\rho_k(x) = d(x, x_k)$$

where x_k is the k-th nearest neighbor of x. Intuitively, Smaller values of k reflect local density information (lower dimensional geometry), while larger k values reflect more global information (higher dimensional geometry).

Firstly, consider a space of local patches of size 5×10^4 denoted by X, which is much smaller than $\mathcal{M}$. 30 percent of densest points are chosen with smallest values of ρ_{15} from X[14]. Denote the space by $X(15, 30)$. By constructing witness complex and analyze the persistent homology, it is observed that first Betti number b^1 of the space equals 5 shown in Fig.2.

A "three circle" model was propose which satisfies $b^1 = 5$, denoted by C_3[14]. The data are clustered along the three circles in $\mathbb{R}^8$. In the following picture, Both the green and yellow circles intersect the black circle at two points each, without intersecting each other. To figure out the first betti number of C_3 is exactly 5, we could consider C_3 as a graph with 4 vertices, 8 edges and one connected component. Hence, using the formula we obtain $b_1 =$ #(arcs) −#(vertices)+#(connected components) = 5.

Experiments were also conducted for the case of k=300[14]. The space $X(300, 30)$ loses

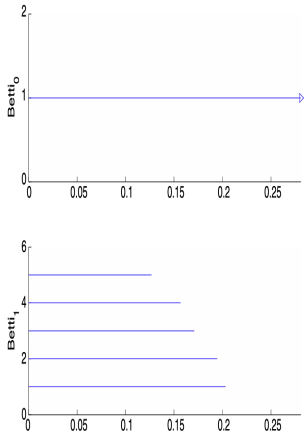
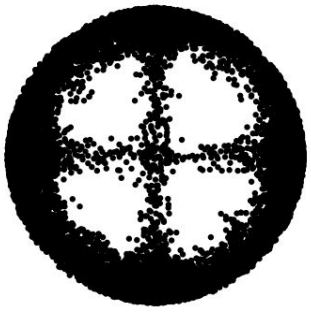
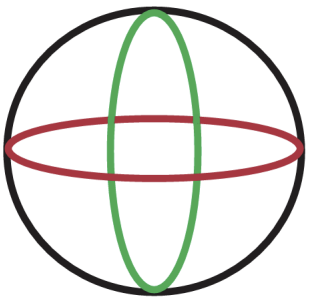


Figure 2 Barcode for $X(15, 30)$



a) Point cloud of $X(15, 30)$



b) Abstract model

Figure 3 Three circle model

two secondary circles as k becomes bigger. The results are shown in Fig.4.

Figure 4 Point cloud of $X(300, 30)$

There is an specific explanation for C_3 and what the high-density local patches truely looks like in[14]. We could combine the patches with a real-valued function of two variables by applying it on nine grid points of the unit disc. As we can see in Fig.5, the primary circle corresponds to linear gradients, parametrized by angle. These patches can be considered as edge features in a picture. The secondary circles connect functions that increase along a linear projection to quadratic functions known as "bump functions," which feature a local maximum within the same linear projection. Note that the patches in DCT basis appears in three circles, which indicates spatial relationship between them.

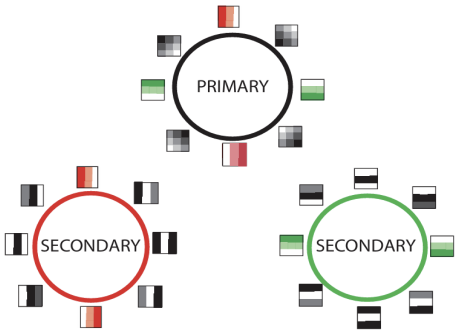


Figure 5 Data on the three circle

The results of $X(15, 30)$ indicates that there are two competing preferences of local behavior in natural images. One is a preference for linear intensity functions on local patches (primary circle) and the other is for vertical and horizontal directions in the secondary circle. If we put similar analysis on the local patches obtained from the images taken by the camera

holding at $45°$, the linear gradients is also rotated by the same angle while the quadratic gradients on secondary circles still have a strong bias in vertical and horizontal directions. It can be interpreted in two aspects: One is the nature has the bias since objects are most stable in horizontal and vertical positions. The other is the technology of camera makes the patches in favor of vertical and horizontal directions[15].

4.3 Klein Bottle

The three circle model is a relatively sparse skeleton which indicates the preferences of local behavior in natural images. To find a larger 2-dimensional manifold containing it with substantial density, we consider a natural embedding from the three circles to the manifold. Recall that the model embeds naturally in Klein bottle as shown in the following picture.

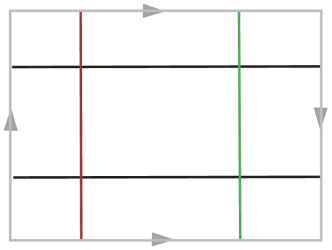


Figure 6 Three model embeds naturally in Klein bottle

The black horizontal segments represents the primary circle while the two vertical segments are secondary circles after identification. The embedding indicates the possibility that a bigger space of high contrast patches after a more relaxed density filtration may be homeomorphic to Klein bottle.

4.3.1 Theoretical Version of Klein Bottle in the Space of Polynomials

In this subsection, we show a theoretical proof of the existence of Klein bottle in the space of local patches by combining 3×3 local patches with quadratic functions of two variables[15].

Firstly, regard the 3×3 patches as the result vectors obtained by applying a smooth real-valued function of two variables on nine grid points of the unit disc. Weierstrass's theorem tells us that polynomials are dense in the space of continuous functions. Hence we consider

a subspace consists of all two variable polynomials of degree 2, denoted by $\mathcal{Q}$, i.e.

$$\mathcal{Q} = \left\{ f(x,y) = A + Bx + Cy + Dx^2 + Exy + Fy^2 \right\}$$

Obviously, $\mathcal{Q}$ is a 6-dimensional real vector space. Since the mean centering and normalization has been applied to the data in practical operations, the function here should satisfy two properties:

$$\int_D f^2 = 1 \ \& \int_D f = 0$$

Denote the space of functions in $\mathcal{Q}$ with two restrictions above by $\mathcal{P}$, which is a 4-dimensional ellipsoid in $\mathbb{R}^6$.

Now we consider a subspace of the space of smooth real-valued function , the set of all functions in the form

$$f(x,y) = q(\lambda x + \mu y)$$

where $\lambda^2 + \mu^2 = 1$ and q is a quadratic function with single variable.

Let A be the space of single variable quadratic function $q(t) = c_0 + c_1 t + c_2 t^2$ with two restrictions

$$\int_{-1}^{1} q(t)\, dt = 0 \ \& \int_{-1}^{1} q(t)^2\, dt = 1$$

Simply the equations above we get ‘

$$c_2 = -3c_0 \ \& \ \frac{2}{3}c_1^2 + \frac{8}{5}c_0^2 = 1,\ c_0,\ c_1,\ c_2 \in \mathbb{R}$$

It indicates that A is an ellipse which is homeomorphic to a circle.

For any $q \in A$ and any unit vector $\vec{v}$ in $\mathbb{R}^2$, define $q_{\vec{v}} : \mathbb{R}^2 \to \mathbb{R}$ by $q_{\vec{v}}\,(\vec{\omega}) = q(\vec{v} \cdot \vec{\omega})$ where $\vec{\omega}$ is a 2-dimensional variable. We have that

$$\int_D {q_{\vec{v}}}^2 \neq 0 \ \& \int_D q_{\vec{v}} = 0$$

Define a continuous map ϕ from $A \times S^1$ to P_0 by the formula

$$(q, \vec{v}) \mapsto \frac{q_{\vec{v}}}{\|q_{\vec{v}}\|_2}$$

Let (c_0, c_1) be the representative of q and it is easy to check that $((c_0, c_1) \times \vec{v}) \sim ((c_0, -c_1) \times (-\vec{v}))$ in ϕ, i.e.

$$\phi((c_0, c_1) \times \vec{v})) = \phi((c_0, -c_1) \times (-\vec{v})))$$

Notice that it is an 2-fold covering map and the orbit space is homeomorphic to a Klein bottle.

4.3.2 Experimental Version of Klein Bottle in the Data

The aforementioned theoretical proof can assist us in finding the Klein bottle (based on the three circle model) in the data. In fact, some direct experiments like changing the density threshold or enlarging the space will not generate the non-trivial second betti number b^2 which is a characteristic of Klein bottle. The failure of the density function to cut out the 2-dimensional manifold leads us to think about if there are some local patches that correspond to certain points on the theoretical Klein bottle, but they occur with such low frequency (low density) that are undetectable by the density threshold. Through the three circle model, we know that there are preferences for the linear polynomials and local patches that are lined up with horizontal and vertical directions. Hence, the least frequently appearing local patch may be the non-horizontal and non-vertical pure quadratic gradients.

Let $\bar{Q}$ be the set of patches with purely quadratic gradients corresponding to the polynomials $(ax + by)^2$ or $-(ax + by)^2$ $(a, b \neq 0)$. Then compute the distances between $\mathcal{M}$ and 30 points sampled from $\bar{Q}$. For each point sampled from $\bar{Q}$, choose the closest point of $\mathcal{M}$. Denote the obtained set as Q.

As we have discussed in the part of density filtration, smaller k will exhibit more details of data structure. Hence consider the space $\mathcal{M}(100, 10) \cup Q \subset \mathcal{M}$. Note that $k = 100$ for $\mathcal{M}$ corresponds to $k = 1.25$ for X. Construct the witness complex on it and the results of persistent homology are as follows. Note that the coefficients of homology is in $\mathbb{Z}_2$.

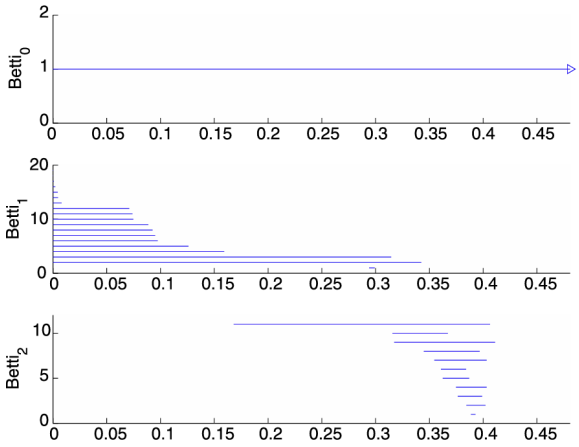


Figure 7 Barcode for $\mathcal{M}(100, 10) \cup Q$

Actually, there are two 2-dimensional manifolds whose $\mathbb{Z}_2$ homology group agrees with above results: torus and Klein bottle. However, the results of $\mathbb{Z}_3$ homology group indicate that the space is homeomorphic with Klein bottle.

In summary, the results in this subsection shows that as the density estimation parameter (k) decreases, the space of high contrast local patches with high density gradually fills out a 2-dimensional manifold: firstly the primary circle of linear gradients and then two secondary circles of quadratic gradients in horizontal and vertical directions. Finally, it contains polynomials in all intermidiate directions except the purely quadratic gradients in non-horizontal and non-vertical directions. This also indicates the importance of the choice of threshold in density analysis since different density filtration will reveal different levels of topological information and an appropriate threshold can help us exclude a few outliers and capture core features.

4.3.3 Embedding the Theoretical Klein Bottle into the Data

To test that the space of high contrast patches with high density is consistent with the theoretical Klein bottle defined by quadratic polynomials in a way, the author used the theoretical construction to embed the Klein bottle $\mathcal{K}_0$ into S^7 and then move each point towards closest and densest points of $\mathcal{M}$ while ensuring the topology of the new manifold remains invariant[6].

Here is a result showing the barcode of a subspace containing 60 percent of points of $\mathcal{M}$ with a topology of Klein bottle[6]. The author also use Monte Carlo method and find out the volume of $\mathcal{M}$ is 84% while the volume of the subspace is only 21%. It indicates that the subspace which is homeomorphic to the Klein bottle contains a dense cluster of points.

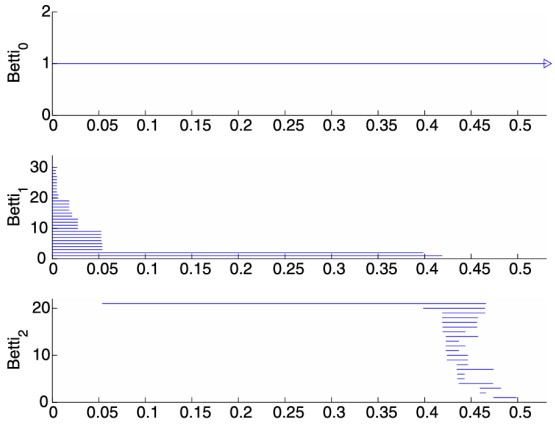


Figure 8 Barcode of a subspace containing 60 percent of points of $\mathcal{M}$

4.4 Summary

The study shows the topological structure of the high density space of high contrast local patches (also known as features), which is homeomorphic to a Klein bottle. This result can be applied in image compression by replacing the origin space with an embedded Klein bottle. Besides, it can be applied to study the functioning mechanism of the primary visual cortex and also provides guidance for research on bio-inspired neural networks. We will introduce the latter in the next two sections.

5. Topological Analysis on Weight Space of Trained CNNs

In a CNN, the weights are the coefficients of the transition formula, which reflects the relationship between adjacent layers. Like local patch in natural images, we could regard a 3×3 kernel in a CNN as a 9-dimensional vector, which is called the weight vector. The distribution space of weight vectors on a given convolutional layer is called the weight space.

The neurons within the primary visual cortex, as well as the filters in CNNs, reflects functions or responses on image patches. Hence a natural idea is to explore the structure of CNNs to see if it has any correspondence with the distribution characteristics of image

patches. This section will focus on the reproduction of topological analysis on weight sapce of CNNs[7].

5.1 Experiments on Different Datasets

We perform analyses on weight space of CNNs trained on MNIST and CIFAR-10 and find some simple structures over the course of training.

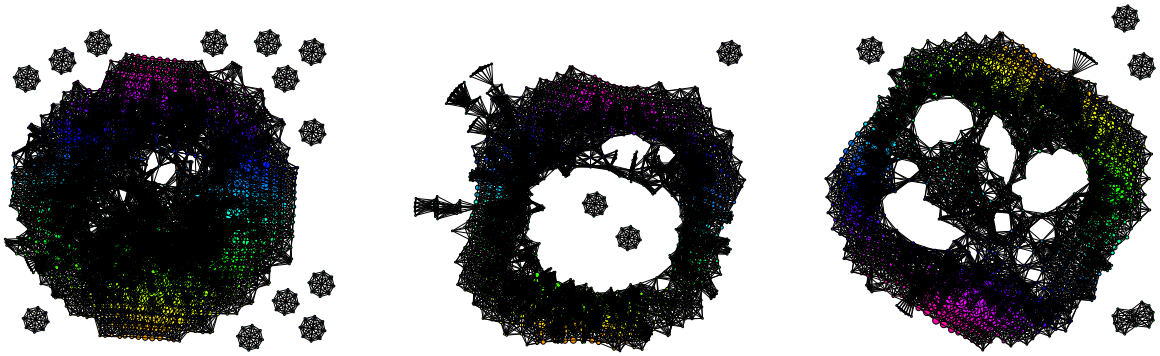
5.1.1 MNIST

We first implement analysis on the dataset of handwritten digits. MNIST was divided into two subsets with 60,000 training examples and 10,000 test examples. We train 100 CNNs with a batch size of 128 for 20 epochs (to a test accuracy about 0.99), obtaining 6400 9-dimensional vectors for the first layer filters. We use Cross Entropy as criterion and ADAM as optimizer. The detailed architecture of CNN is as follows:

$$\chi(1)\times\mathbb{Z}^2 \xrightarrow[ReLU]{C_c\times C_d(1)} \chi(64)\times\mathbb{Z}^2 \xrightarrow{C_c\times\pi^2(0,1,1)} \chi(64)\times\mathbb{Z}^2 \xrightarrow[ReLU]{C_c\times C_d(1)}$$
$$\chi(32)\times\mathbb{Z}^2 \xrightarrow{C_c\times\pi^2(0,1,1)} \chi(32)\times\mathbb{Z}^2 \xrightarrow[ReLU]{C_c} \chi(64)\times\mathbb{Z}^2 \xrightarrow[Dropout(0.5)]{C_c} \chi(10)$$

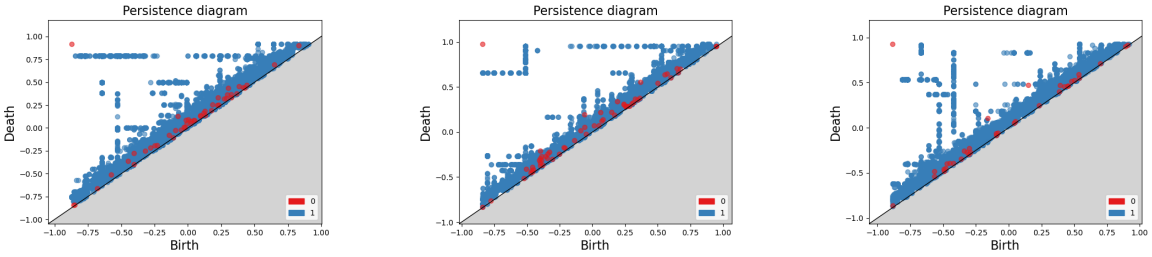
Then we standarize the weight vectors of the first layer and use KNN density filtration with $k = 200$ and $p = 0.3$ (choose the distance between one data point and the kth nearest neighborhood as an indicator of density and take the top p fraction of the densest points) to get 1920 points. Next, we apply Mapper with resolution = 30, gain=3 and lens= PCA(components=2) to obtain a Mapper complex. The visualization and the persistence diagrams of the weight space during the training process are shown in Fig.9a to Fig.10c. Note that the size of the point in the mapper complex represents the number of real data points contained in it and the color is decided by the clustering. The 'birth' 'death' time in the persistent diagram represent the maximum and minimum colors that form the topological features which is different from $\epsilon-$ threshold persistence of VR complex.

Fig.9 reveals the learning process of the first layer in the CNN: At the beginning, the distribution of the weight vector is scattered. Trained after 5 epochs, we can see a circular



a) untrained weights **b) 5-epochs trained weights** **c) 20-epochs trained weights**

Figure 9 Visualization of mapper complex



a) untrained weights **b) 5-epochs weights** **c) 20-epochs weights**

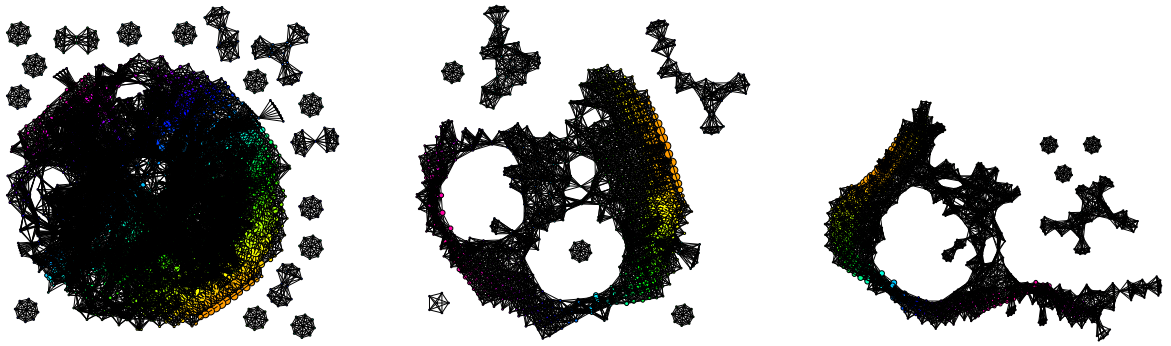
Figure 10 Persistence diagrams of mapper complex

shape. The visualization of mapper complex is roughly consistent with the primary circle in Fig.4 and the points on are filters parametrized by angle and used for edge detection. When it is trained for 20 epochs, the primary circle still exists but the structure seems more diffuse. The persistent diagram does not indicate the presence of a loop in that case. Specifically, the kernels distributed on the circle are primarily edge detection kernels parametrized with angle, and trained after 20 epochs, irregular kernels start to emerge in the central part of the circle.

5.1.2 CIFAR-10

We conduct similar experiments on a more complex dataset, CIFAR-10, which is divided into 50000 training examples and 10000 testing examples. Firstly, it is grayscaled by the weights (0.2989,0.5870,0.1140) and put into the neural network we used in previous section. We train 50 CNNs for 100 epochs and both the train accuracy and test accuracy

reach 0.66 after 50 epochs. However, the train accuracy exceeds 0.7 at 100 epochs while the test accuracy remains to be 0.65， indicating the occurrence of overfitting. We extract the 204800 weight vectors of second convolutional layer and use KNN density filtration with $k = 200$ and $p = 0.02$. Then construct the mapper complex with resolution $= 30$, gain $= 3$ and lens=PCA (components=2).



a) 5-epochs trained weights **b) 50-epochs trained weights** **c) 100-epochs trained weights**

Figure 11 Visualization of mapper complex

From fig.11 we can see that the undertrained weights has a scattered distribution and after 50 epochs, the well-trained weights exhibits a structure with a primary circle and two secondary circles inside. However, when the convolutional layer overfits at around 100 epochs, the overtrained weights lose the structure of primary circle and become chaotic again. It indicates that to some extent, the simple structure of the weight space reflects the training performance of a neural network. Both underfitting and overfitting can cause the disappearance of this simple structure.

5.2 Correlation between the Feature Space of Natural Images and the Weight Space of CNN

The analysis on weight space shows that CNN learns simple structures during the training process and the core topological features of well-trained weight space is consistent with the space of high contrast patches. Neural networks used to be regarded as a black box due to the lack of interpretability. The visualization of weight space gives us an insight of geometric information of CNNs while training, which helps us better understand the working

principles and learning process of neural networks.

What's more, the consistency of topological persistent features between natural images and CNNs indicates that we can view the mechanism of neural networks as the action on input data. Here is an reasonable interpretation that the weight vector of a kernel act on the vector of local patch as the inner product to extract the features. We know by previous sections the high contrast features of natural images mainly distributes on a primary circle and further the Klein bottle[6]. The weight vectors have similar distribution so that the result of inner product is larger than that of low contrast patches, which makes high-contrast patches have a greater impact in subsequent classification. This shows a concrete process of feature extraction which is known as one of the main functions of neural networks.

6. Topological Convolutional Neural Network

6.1 Construction of TCNN

Topological convolutional nueral networks (TCNN) are defined with topological features and put restrictions on convolutional layers based on topological embedding into image space and weight space of NNs[8].

6.1.1 Motivations

In section 4, it was found that there is a large portion of the space of patches which is topologically equivalent to the Klein bottle. Then section 5 shows that CNNs learn the weights on filters and the trained weights observed are exactly corresponding to the high-density image patches found in section 4 which distribute on the Klein bottle $\mathcal{K}$. Thus, it seems natural and reasonable that regarding the space of filters as discretizations of Klein bottle embedded in the convolutional layer and the filters can be interpreted as points on the Klein bottle. Notice that in section 4, there is a second embedded Klein bottle $\mathcal{K}^{alg}$ based on the theoretical proof of existence of Klein bottle in the space of local patches[15] and $\mathcal{K}^{alg}$ has simple algebraic description and lies close to $\mathcal{K}$, TCNNs are constructed based on symmetric properties of $\mathcal{K}^{alg}$ and paramatrized algebraically from $\mathcal{K}^{alg}$.

6.1.2 Two New Types of Convolutional Layers for 2D image Classification

In this section, we introduce two types of convolutional layers[8] which form TCNNs for 2D image classification: M Filters Layer and M One Layer where M represents manifolds like circle and Klein bottle. The Circle Filters (CF) layer and Klein Filters (KF) layer use normal architecture of convolutional layer but predefined filters on it. The Klein One Layer (KOL) restricts connections between filters within a convolutional layer based on the topological structure of the Klein bottle. Similarly, Circle One Layer (COL) is analogous layers based on the topology of a circle. The precise definition are introduced below.

Firstly, we specify the convolutional layer with image data as inputs.

Definition 6.1 *(**NOL**) Let V_{i+1} be a layer in a FFNN. We call V_{i+1} a **convolutional layer** or a **normal one layer (NOL)** if $V_i = \chi \times \mathbb{Z}^N$ and $V_{i+1} = \chi' \times \mathbb{Z}^N$ for some finite sets χ and χ' and a positive integer N, and if for some fixed threshold $s \geq 0$ the edge-defining correspondence $C \subset V_i \times V_{i+1}$ is of the form*

$$C = C_c \times C_{d,N}(s),$$

where $C_c = \chi \times \chi'$ is the fully connected correspondence and $C_{d,N}(s) \subset \mathbb{Z}^N \times \mathbb{Z}^N$ is the correspondence given by

$$C_{d,N}(s)^{-1}(\underline{x}') := \{\underline{x} \in \mathbb{Z}^N \mid d_{\mathbb{Z}^N}(\underline{x}, \underline{x}') \leq s\}$$

for all $\underline{x}' \in \mathbb{Z}^N$. Here, $d_{\mathbb{Z}^N}$ is the L^∞-metric on $\mathbb{Z}^N$ defined by

$$d_{\mathbb{Z}^N}(\underline{x}, \underline{x}') = \max\{|x_1 - x_1'|, \ldots, |x_N - x_N'|\}.$$

Next, we introduce a layer which localizes weights on the Klein bottle $\mathcal{K}$ inspired by the appearance of $\mathcal{K}$ in the weight space of CNNs observed in section 5. Recall the theoretical version of $\mathcal{K}$ proved in section 4, the points in $\mathcal{K}$ have correspondence with functions in the

following form

$$f(x,y) = q(\lambda x + \mu y)$$

where $\lambda^2 + \mu^2 = 1$ and q is a quadratic function with single variable satisfying that the quadratic coefficient is a multiple of the constant term. Thus, define $F_{\mathcal{K}}$ to be an embedding from $\mathcal{K}$ to the vector space of quadratic functions on the square $[-1,1]^2$:

$$F_{\mathcal{K}}(\theta_1,\theta_2)(x,y) = \sin(\theta_2)(\cos(\theta_1)x + \sin(\theta_1)y) + \cos(\theta_2)Q(\cos(\theta_1)x + \sin(\theta_1)y),$$

where $Q(t) = 2t^2 - 1$. As given, $F_{\mathcal{K}}$ is a function on the Klein bottle parameterized by the two angles θ_1 and θ_2 since it satisfies $F_{\mathcal{K}}(\theta_1,\theta_2) = F_{\mathcal{K}}(\theta_1 + 2k\pi, \theta_2 + 2l\pi)$ and $F_{\mathcal{K}}(\theta_1 + \pi, -\theta_2) = F_{\mathcal{K}}(\theta_1,\theta_2)$. The image space "embedded Klein bottle" $F_{\mathcal{K}}(\theta_1,\theta_2)$ consists of quadratic functions mentioned above and has a natural 'orientation' given by the angle θ_1.

To define $F_{\mathcal{S}^1}$, consider the composite of $F_{\mathcal{K}}$ and the inclusion map from $\mathcal{S}^1$ to $\mathcal{K}$ which maps θ to $(\theta, \pi/2)$:

$$F_{\mathcal{S}^1}(\theta)(x,y) := F_{\mathcal{K}}(\theta,\pi/2)(x,y) = \cos(\theta)x + \sin(\theta)y,$$

Circle Filters and Klein Filters are defined based on the function $F_{\mathcal{S}^1}$ and $F_{\mathcal{K}}$, respectively.

Definition 6.2 ***(M Filters (MF) Layer)*** *Let $M = S^1$ or $\mathcal{K}$. Let $\chi \subset M$ be a finite subset and $V_i = \mathbb{Z}^2$ and $V_{i+1} = X \times \mathbb{Z}^2$ be successive layers in a FFNN. Assume V_{i+1} is a convolutional layer with threshold $s \geq 0$. Then V_{i+1} is called a* ***M Filters (MF) layer*** *if the weights $\lambda_{-,(\kappa,-,-)}$ are given for $\kappa \in \chi$ by a convolution over V_i of the filter of size $(2s+1)\times(2s+1)$ with values*

$$Filter(\kappa)(n,m) = \int_{-1+\frac{2m}{2s+1}}^{-1+\frac{2(m+1)}{2s+1}} \int_{-1+\frac{2n}{2s+1}}^{-1+\frac{2(n+1)}{2s+1}} F_M(\kappa)(x,y)dxdy$$

for integers $0 \leq n, m \leq 2s$.

Notice that according to the theoretical proof of Klein bottle, if the filter is on the Klein bottle $\mathcal{K}$ paramatrized by (θ_1, θ_2), the values of the kernel should equal to the function value of $F_{\mathcal{K}}(\theta_1, \theta_2)$ at the grid points of the square $[-1, 1]^2$. The integration can be interpreted as dividing $[-1, 1]^2$ into $(2s+1) \times (2s+1)$ smaller squares, where the value at each grid point is the average of the function values within its corresponding small square.

Taking 3×3 kernels as an example, We can have a look at these predefined kernels. The detailed equation of KF after integration is as follows:

$$Filter(\kappa)(n, m) = \frac{8}{27}(\sin\theta_2)[(m-1)\cos\theta_1 + (n-1)\sin\theta_1] \tag{1}$$

$$+\frac{32}{81}(\cos\theta_2)[(m-1)\cos\theta_1 + (n-1)\sin\theta_1]^2 \tag{2}$$

$$-\frac{100}{243}\cos\theta_2 \tag{3}$$

There are three parts in the equation corresponding to terms of different orders. The kernel of part (1) is just CF since the primary circle S^1 is embedded to the Klein bottle $\mathcal{K}$ by setting $\theta_2 = \pi/2$. The simplified CF are as follows：

$$\begin{bmatrix} \cos\theta - \sin\theta & \cos\theta & \cos\theta + \sin\theta \\ -\sin\theta & 0 & \sin\theta \\ -\cos\theta - \sin\theta & -\cos\theta & -\cos\theta + \sin\theta \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \cos\theta & \cos\theta \\ 0 & 0 & 0 \\ -\cos\theta & -\cos\theta & -\cos\theta \end{bmatrix} + \begin{bmatrix} -\sin\theta & 0 & \sin\theta \\ -\sin\theta & 0 & \sin\theta \\ -\sin\theta & 0 & \sin\theta \end{bmatrix}$$

$$= \cos\theta \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} + \sin\theta \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

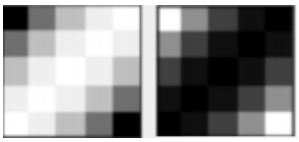
It can be seen that CF uses two kinds of Prewitt operator[16] (vertical and horizontal) as basis, which is an one-order partial derivative operator to extract edge by linear gradient and the angle θ represents the direction of edge extraction. [Ref]

The kernel of second-order term (2) is simplified below:

$$\begin{bmatrix} (\cos\theta_1 + \sin\theta_1)^2 & \sin^2(\theta_1) & (\cos\theta_1 - \sin\theta_1)^2 \\ \cos^2(\theta_1) & 0 & \cos^2(\theta_1) \\ (\cos\theta_1 - \sin\theta_1)^2 & \sin^2(\theta_1) & (\cos\theta_1 + \sin\theta_1)^2 \end{bmatrix}$$

It is a Laplacian operator[16], which is an second-order partial derivative extracts texture by computing second gradients.

a) Prewitt operator



b) Laplacian operator

Figure 12 First and second order operators

Next, we will introduce the other type of TCNN[8]: COL and KOL. These layers are designed with the idea of locality in the connections between filters. Locality in NNs is defined with a distance function as metric:

Definition 6.3 ***(Correspondence of Locality)*** *M is a manifold and let $\chi, \chi' \subset M$ be two discretizations of M. Let $V_i = \chi \times \mathbb{Z}^N$ and $V_{i+1} = \chi' \times \mathbb{Z}^N$ be successive layers in a FFNN. Fix a threshold $s \geq 0$. Let d be a metric on M. Define a correspondence $C(s) \subset \chi \times \chi'$ by*

$$C(s)^{-1}(\kappa') = \{\kappa \in \chi \mid d(\kappa, \kappa') \leq s\}$$

for all $\kappa' \in \chi'$. Together with another threshold $s' \geq 0$, this defines a correspondence $C \subset V_i \times V_{i+1}$ by

$$C = C(s) \times C_{d,N}(s'),$$

where $C_{d,N}(s')$ is the convolutional correspondence. This means that

$$\begin{aligned} C^{-1}(\kappa', \underline{x}') &= C(s)^{-1}(\kappa') \times C_{d,N}(s')^{-1}(\underline{x}') \\ &= \{(\kappa, \underline{x}) \in \chi \times \mathbb{Z}^N \mid d(\kappa, \kappa') \leq s \text{ and } d_{\mathbb{Z}^N}(\underline{x}, \underline{x}') \leq s'\} \end{aligned}$$

for all $(\kappa', \underline{x}') \in \chi' \times \mathbb{Z}^N$.

In COL and KOL, topological structures are added into the convolutional layer with corre-

sponding metric. In[8], the metric of S^1 and $\mathcal{K}$ are defined as follows:

$$d_{S^1}(\kappa, \kappa') = cos^{-1}(\kappa \cdot \kappa') \text{ for } \kappa, \kappa' \in S^1.$$

$$d_{\mathcal{K}}(\kappa, \kappa') = \left(\int_{[-1,1]^2} \left(F_{\mathcal{K}}(\kappa)(x,y) - F_{\mathcal{K}}(\kappa')(x,y) \right)^2 dxdy \right)^{\frac{1}{2}}$$

It is easy to understand that we could measure distance in COL by the difference of angles. As for $d_{\mathcal{K}}(\kappa, \kappa')$, we use the L^2 norm of the difference of two functions as the metric in KOL.

Definition 6.4 ***(Circle One Layer (COL) and Klein One Layer (KOL))*** *Let M be S^1 in definition 5.2.2 and the* ***circle correspondence*** *$C_S(s) \subset \chi \times \chi'$ is defined by*

$$C_{S^1}(s)^{-1}(\kappa') = \{\kappa \in \chi \mid d_{S^1}(\kappa, \kappa') \leq s\}$$

for all $\kappa' \in \chi'$.

We call V_{i+1} a circle one layer (COL) if, for some other threshold $s' \geq 0$, the edge-defining correspondence $C \subset V_i \times V_{i+1}$ is of the form

$$C = C_{S^1}(s) \times C_{d,2}(s'),$$

Let M be $\mathcal{K}$ and The ***Klein correspondence*** *$C_{\mathcal{K}}(s) \subset \chi \times \chi'$ is defined by*

$$C_{\mathcal{K}}(s)^{-1}(\kappa') = \{\kappa \in \chi \mid d_{\mathcal{K}}(\kappa, \kappa') \leq s\}$$

for all $\kappa' \in \chi'$. We call V_{i+1} a Klein one layer (KOL) if, for some other threshold $s' \geq 0$, the edge-defining correspondence $C \subset V_i \times V_{i+1}$ is of the form

$$C = C_{\mathcal{K}}(s) \times C_{d,2}(s'),$$

Here is an example of COL represented by a rectangle. The upper row represents an input slice or channel portrayed by the filter aligned with a point on the circle, while the left column mirrors an output slice similarly. Every input slice and output slice features the associated trained filter in a COL，which is the intersubsection of row and column. Note

that the black block means there is no connection between the two indices.

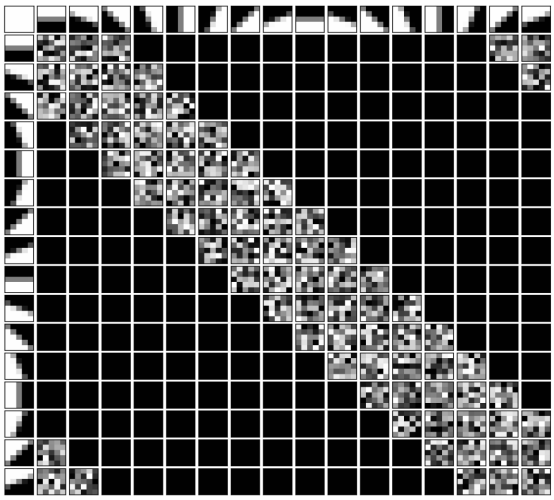


Figure 13 The trained weights in a COL.

6.2 Performance Test of TCNN

We conducted several experiments on different datasets to compare TCNN and CNN in two aspects: training performance and generalizability.

6.2.1 Training Performance

We train MNIST with similar neural networks architecture on CNN and TCNN. The control group is a traditional CNN consisting of two convolutional layers and fully connected layers. For the experimental group, we replaced the two convolutional layers with combinations of CF, KF, COL, KOL and NOL.

We outline two approaches for incorporating KF (Klein Filter) and KOL (Klein Oriented Learning) into neural networks: The first method involves uniformly distributing the convolution kernels on the Klein bottle (θ_1 and θ_2 are uniformly sampled); the second method entails a density-based distribution of the convolution kernels on the Klein bottle. The theoretical analysis in Section 4 reveals that the points of highest density in the Klein bottle are located on the three-circle model, whereas points of lowest density are generated by pure quadratic functions on non-vertical and non-horizontal directions. Hence we simulate the true Klein bottle distribution by placing a varying number of convolution kernels in different

regions. We call the fomer ”KF” and ”KOL” as usual and the latter is ”KF optimized” and ”KOL optimized”.

To enhance model robustness and prevent overfitting, Gaussian noise is added into the training set. We also add it to testing set to evaluate the generalizability of the model. The left half of the figure14 illustrates the training performance with noise added to the training set, while the right half shows the training performance with noise added to the test set.

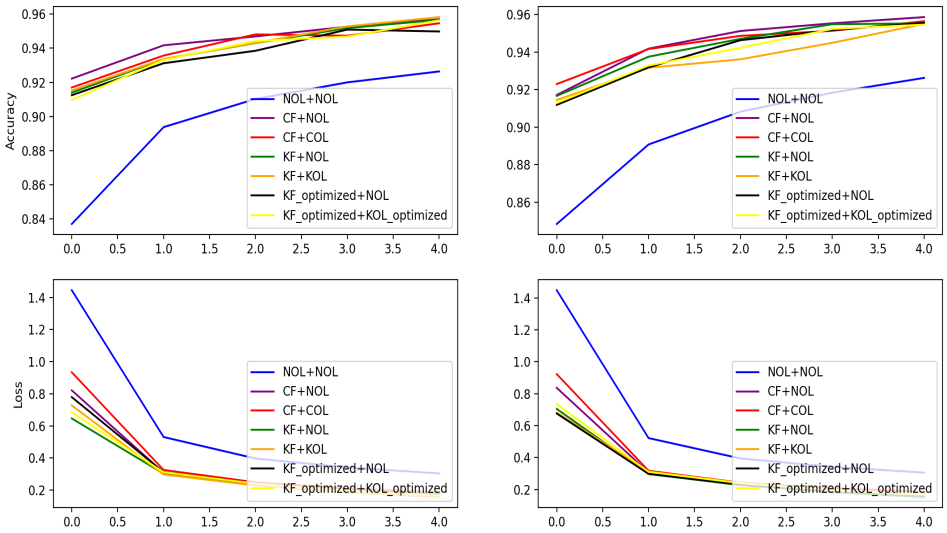


Figure 14 Training performance of TCNN and CNN

From the figure above, we can see that compared with traditional CNN, TCNN has a higher training accuracy when training for the same epoch and a faster training speed (the curve flattens out sooner). Among all types of TCNN, CF performs the best; The performance of KF optimized and KOL optimized is slightly better than that of KF and KOL.

6.2.2 Generalizability

MNIST and SVHN are two popular datasets which contain digits from 0 to 9. To compare the generalizability of TCNN and CNN, we trained MNIST on NOL, CF+COL and CF+NOL and test the model with SVHN. The test accuracy of each model are shown in the following figure.The interval on the horizontal axis represents training for two epochs.

It can be seen that the test accuracy of TCNN on SVHN is higher than that of traditional

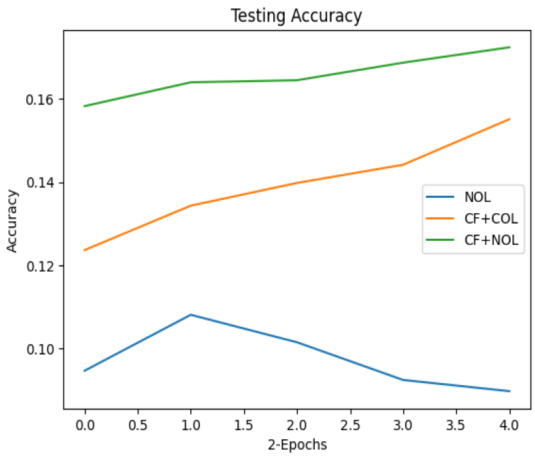


Figure 15 Gerneralizability of NOL, CF+COL and CF+NOL

CNN. The decreasing trend of the test accuracy curve of NOL as the number of epochs increases indicates that the model has to some extent overfit the data. Hence the generalizability of TCNN is better than traditional CNN.

6.3 TCNN vs CNN

In traditional CNN, the convolutional layer is a sparsification of a fully connected layer which enforced locality with homogeneity in the grids $\mathbb{Z}^N$. The two key properties of CNNs are locality and homogeneity (translation invariance), as discussed in[11], which originates from the use of kernels.

Based on the above two properties, CNNs, compared to the original fully connected neural networks, possess the characteristics of reducing model complexity and parameter space. It can enhance the speed of model training and prevent overfitting.

Compared to CNNs, TCNNs endow the convolutional layers with a topological structure. In CF and KF, we initialize the network with fixed weights (gradient=0) instead of forcing CNNs to learn these weights. This can bring the network high training efficiency and high accuracy and prevent overfitting. These layer can be thought of as a pretrained convolutional layer in CNNs. In COL and KOL, the filter space are parameterized by a discretization of a circle or Klein Bottle and during the entire training process, any weights connecting slices beyond a fixed threshold distance on the Klein bottle are maintained as zero. The locality conditions promote the network to learn local features both within the raw

pixels and in the tangential directions in the geometry of a circle or Klein bottle. There is an interpretation that what is learned in COL and KOL is second order local information, followed by the first order information obtained from the previous convolutional layer[8]. Also, symmetries in circle and Klein bottle enforce rotational invariance of the features as well as the invariance under black/white reversal.

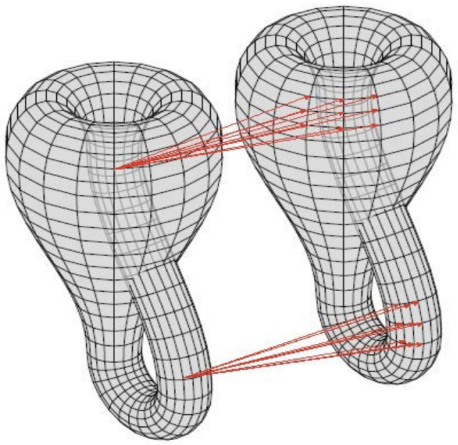


Figure 16 A graphical depiction of neurons and weights in a KOL with a threshold of s = 2. Each grid line intersubsection on the two Klein bottles represents a neuron. The non-zero weights associated with the specified input neuron are illustrated by the red arrows.

Both of these new types of TCNN can be regarded as a rigorous form of regularization, elucidating the TCNN's capacity to generalize more effectively to new data.

7. Summary

7.1 Research Contents and Results

This article roughly reviews the applications of topological data analysis in image data and the transition to topological deep learning, which begins with the analysis of geometric structures of natural images and neural networks and then used the obtained information to improve the architecture of neural networks. Some reproduced results are also presented in the article. The results are mainly three aspects: The core space of high contrast local patches of natural images has a topology of Klein bottle; CNNs trained on images learn simple structures which is consistent with the distribution of features on the images; Adding the topological information above to neural networks gains better training performance and

better generalizability.

7.2 Prior Judgements and Posterior Analytics in the Research Process

Prior judgements and posterior analytics are well used in the research process. In the study of feature space of natural images and weight space of CNNs, posterior analytics is used to find out the learning process of neural networks and then we use the ”pretrained” filters as a kind of prior judgements to improve the neural networks for image classification. Specifically, in the field of deep learning for image processing as we discussed in this paper, images and CNNs have an interactive relationship: the function of neural networks is to extract image features for classification; Conversely, if we know some features of the images, we can improve the neural network based on these features to enhance learning efficiency.

7.3 Future Work

Similar study using TDA can be conducted on other types of data, such as audio and video. We could also implement experiments on pooling layers and other architecture of neural networks to see the structures in it and use these geometric information to improve it.

Besides, inspired by the positive correlation between the persistence of topological features of CNN and the generalizability[7] and the phenomenon of overfitting we discussed in Section 5, we may consider the topological structures as a geometric feature of neural networks and further explore the correlation between it and other attributes (training performance, generalizability). This idea try to establish a connection between geometric properties and ”intrinsic” properties of neural networks, which makes the topological features a criterion to measure the performance of neural networks.

References

[1] SIZEMORE A E, PHILLIPS-CREMINS J E, GHRIST R, et al. The importance of the whole: Topological data analysis for the network neuroscientist[J/OL]. Network Neuroscience, 2019, 3(3): 656-673. eprint: https://direct.mit.edu/netn/article-pdf/3/3/656/1092395/netn_a_00073.pdf. https://doi.org/10.1162/netn%5C_a%5C_00073. DOI: 10.1162/netn_a_00073.

[2] KRAMAR M, GOULLET A, KONDIC L, et al. Persistence of Force Networks in Compressed Granular Media[J]. Physical Review E, 2013, 87(4): 042207. DOI: 10.1103/PhysRevE.87.042207.

[3] CHAI J, ZENG H, LI A, et al. Deep learning in computer vision: A critical review of emerging techniques and application scenarios[J/OL]. Machine Learning with Applications, 2021, 6: 100134. https://www.sciencedirect.com/science/article/pii/S2666827021000670. DOI: https://doi.org/10.1016/j.mlwa.2021.100134.

[4] ZHAO Q, YE Z, CHEN C, et al. Persistence Enhanced Graph Neural Network[C/OL]. in: CHIAPPA S, CALANDRA R. Proceedings of Machine Learning Research: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics: vol. 108. PMLR, 2020: 2896-2906. https://proceedings.mlr.press/v108/zhao20d.html.

[5] OBALLE C, BOOTHE D, FRANASZCZUK P J, et al. ToFU: Topology functional units for deep learning[J/OL]. Foundations of Data Science, 2022, 4(4): 641-665. https://www.aimsciences.org/article/id/724e5cd4-697f-440a-9495-c4607d5c5a30. DOI: 10.3934/fods.2021021.

[6] CARLSSON G, ISHKHANOV T, DE SILVA V, et al. On the Local Behavior of Spaces of Natural Images[J]. International Journal of Computer Vision, 2008, 76(1): 1-12. DOI: 10.1007/s11263-007-0056-x.

[7] BRÜEL GABRIELSSON R, CARLSSON G. Exposition and Interpretation of the Topology of Neural Networks[C]. in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). 2019: 1069-1076. DOI: 10.1109/ICMLA.2019.00180.

[8] LOVE E R, FILIPPENKO B, MAROULAS V, et al. Topological Convolutional Layers for Deep Learning[J]. Journal of Machine Learning Research, 2023, 24(59): 1-35.

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

[10] EDELSBRUNNER H, MUCKE E P. Three-Dimensional Alpha Shapes[J]. ACM Transactions on Graphics, 13:1, 1994: 43-72.

[11] CARLSSON G, GABRIELSSON R B. Topological Approaches to Deep Learning[C]. in: BAAS N A, CARLSSON G E, QUICK G, et al. Topological Data Analysis. Cham: Springer International Publishing, 2020: 119-146.

[12] LEE A, PEDERSEN K, MUMFORD D. The Nonlinear Statistics of High-Contrast Patches in Natural Images[J]. International Journal of Computer Vision, 2003, 54. DOI: 10.1023/A:1023705401078.

[13] VAN HATEREN J H, VAN DER SCHAAF A. Independent Component Filters of Natural Images Compared with Simple Cells in Primary Visual Cortex[J]. Proceedings of the Royal Society of London. Series B: Biological Sciences, 1998, 265(1394): 359-366. DOI: 10.1098/rspb.1998.0303.

[14] SILVA V D, CARLSSON G. Topological estimation using witness complexes[C]. in: GROSS M, PFISTER H, ALEXA M, et al. SPBG'04 Symposium on Point - Based Graphics 2004. The Eurographics Association, 2004. DOI: 10.2312/SPBG/SPBG04/157-166.

[15] CARLSSON G. Topology and Data[J]. Bulletin of the American Mathematical Society, 2009, 46(2): 255-308. DOI: 10.1090/S0273-0979-09-01249-X.

[16] BALOCHIAN S, BALOOCHIAN H. Edge detection on noisy images using Prewitt operator and fractional order differentiation[J]. Multimedia Tools and Applications, 2022, 81: 1-12. DOI: 10.1007/s11042-022-12011-1.

Appendix

Topological Data Analysis of Weight Space

weightsapceMNIST.py

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import torch.nn.functional as F
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
import gudhi as gd
from gudhi.cover_complex import MapperComplex
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from scipy.sparse.csgraph    import dijkstra,
    shortest_path, connected_components
from scipy.stats             import ks_2samp


transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

# load the dataset
train_dataset = datasets.MNIST(root='./data', train=
    True, transform=transform, download=True)
test_dataset = datasets.MNIST(root='./data', train=
    False, transform=transform, download=True)

batch_size = 128
train_loader = torch.utils.data.DataLoader(dataset=
    train_dataset, batch_size=batch_size, shuffle=True)
```

```
29 test_loader = torch.utils.data.DataLoader(dataset=
       test_dataset, batch_size=batch_size, shuffle=False)
30
31 # Define CNN architecture
32 class CNN(nn.Module):
33     def __init__(self):
34         super(CNN, self).__init__()
35         self.net= nn.Sequential(
36                 nn.Conv2d(1, 64, kernel_size=3, stride
                       =1, padding=1),
37                 nn.ReLU(),
38                 nn.MaxPool2d(kernel_size=2, stride=1),
39                 nn.Conv2d(64, 32, kernel_size=3, stride
                       =1, padding=1),
40                 nn.ReLU(),
41                 nn.MaxPool2d(kernel_size=2, stride=1),
42                 nn.Flatten(),
43                 nn.Linear(32 * 26* 26, 64),
44                 nn.ReLU(),
45                 nn.Dropout(0.5),
46                 nn.Linear(64, 10)
47         )
48     def forward(self, x):
49         out=self.net(x)
50         return out
51 # train the data to the test accuracy of about 99%
52 def train(model, train_loader, test_loader, optimizer,
       criterion, epochs):
53         for epoch in range(epochs):
54             model.train()
55             running_loss = 0.0
56             correct_train = 0
57             total_train = 0
58             for images, labels in train_loader:
59                 optimizer.zero_grad()
60                 outputs = model(images)
```

```
61                  loss = criterion(outputs, labels)
62                  #print(loss)
63                  loss.backward()
64
65                  optimizer.step()
66                  running_loss += loss.item()
67                  _, predicted = torch.max(outputs, 1)
68                  total_train += labels.size(0)
69                  correct_train += (predicted == labels).
                        sum().item()
70
71              train_accuracy = 100 * correct_train /
                    total_train
72              train_accuracy_history.append(
                    train_accuracy)
73
74              model.eval()
75              correct = 0
76              total = 0
77              with torch.no_grad():
78                  for images, labels in test_loader:
79                      outputs = model(images)
80                      _, predicted = torch.max(outputs.
                            data, 1)
81                      total += labels.size(0)
82                      correct += (predicted == labels).
                            sum().item()
83
84              test_accuracy = 100*correct / total
85              print(f"Epoch {epoch+1}: Train Accuracy = {
                    train_accuracy:.2f}%, Test Accuracy = {
                    test_accuracy:.2f}%")
86
87  # save the weight vectors
88  l=[]
89  epochs = 100
```

```
90
91  for i in range(1):
92      print('CNN'+str(i+1))
93      cnn = CNN()
94      criterion=nn.CrossEntropyLoss()
95      optimizer=torch.optim.SGD(cnn.parameters(),lr=0.01)
96      train_accuracy_history = []
97      test_accuracy_history = []
98      train(cnn, train_loader,test_loader, optimizer,
            criterion,epochs)
99      weights=cnn.net[0].state_dict()
100     numpy_weights = {key: value.numpy() for key, value
            in weights.items()}
101     l.extend(numpy_weights['weight'].tolist())
102
103 w=np.array(l)
104 np.save('weight_vectors.npy',w)
105
106 # load the weight vectors
107 file_path = 'weight_vectors.npy'
108 try:
109     w = np.load(file_path)
110 except FileNotFoundError:
111     print(f"File '{file_path}' not found.")
112 w = w.reshape(-1, w.shape[2], w.shape[3])
113 w = w.reshape(w.shape[0], -1)
114 #print(w.shape)
115
116 # standarize the data
117 b=np.mean(w, axis=1)
118 c=np.transpose(b)
119 d=np.expand_dims(c, axis=1)
120 w=w-np.tile(d,(1,w.shape[1]))
121 w =w / np.linalg.norm(w, axis=1, keepdims=True)
122
123 # KNN density filtration
```

```
124 r=[]
125 k=200
126 n=1920
127 n_neighbors = k + 1
128 nbrs = NearestNeighbors(n_neighbors=n_neighbors).fit(w)
129 distances, indices = nbrs.kneighbors(w)
130 r = distances[:, k]
131 min_id = np.argsort(r)[:n]
132 w_f= w[min_id]
133
134 # construct the mapper complex (See https://gudhi.inria
      .fr/python/latest/cover_complex_sklearn_isk_ref.html
      )
135 verbose = False
136 cover_complex = MapperComplex(
137     input_type='point␣cloud', min_points_per_node=0,
138     clustering=None, N=100, beta=0., C=10,
139     filter_bnds=None, resolutions=np.array([30,30]),
        gains=np.array([2/3,2/3]),
140     verbose=verbose)
141 pca = PCA(n_components=2)
142 lens = pca.fit_transform(w_f)
143 _ = cover_complex.fit(w_f, filters=lens, colors=lens)
144
145 # visualization of mapper complex
146 # method 1
147 cover_complex.save_to_dot(file_name="weight",
      color_name="color")
148 cover_complex.render('output/weight', format='png',
      width='800', height='600')
149 #neato -Tpdf weight.dot -o weight.pdf
150
151 # method 2
152 cover_complex.save_to_html(file_name="weight",
      data_name="weight", cover_name="PCA(compoments=2)",
      color_name="lens")
```

```
153
154 # extract topological features (See https://github.com/
      GUDHI/TDA-tutorial/blob/master/Tuto-GUDHI-cover-
      complex.ipynb)
155 dgm, bnd = compute_topological_features(cover_complex,
      threshold=0.)
156
157 # generate the barcode and persistence diagram
158 #barcode
159 gd.plot_persistence_barcode(persistence=dgm, alpha=0.6,
       max_intervals=20000, inf_delta=0.1, legend=None,
      colormap=None, axes=None, fontsize=16)
160
161 #Persistent diagram
162 gd.plot_persistence_diagram(persistence=dgm,
      persistence_file='', alpha=0.6, band=0.0,
      max_intervals=1000000, inf_delta=0.1, legend=None,
      colormap=None, axes=None, fontsize=16, greyblock=
      True)
```

Neural networks with CF and COL

TCNN.py

```
1  import numpy as np
2  import torch
3  from scipy import integrate
4  from torch import nn
5  import torch.nn.functional as F
6  import torch.nn as nn
7  import torch.nn.functional as F
8  i
9  # define circle filter
10 def initialize_circle_filters(eta, s):
11     filters_list = []
12
```

```
    for j in range(eta):
        theta = 2 * np.pi * j / eta
        filter_j = torch.zeros((2 * s + 1, 2 * s + 1))

        for n in range(2 * s + 1):
            for m in range(2 * s + 1):
                x_lower = -1 + 2 * n / (2 * s + 1)
                x_upper = -1 + 2 * (n+1) / (2 * s + 1)
                y_lower = -1 + 2 * m / (2 * s + 1)
                y_upper = -1 + 2 * (m+1) / (2 * s + 1)
                result, error = integrate.dblquad(
                    lambda x, y: x*np.sin(theta)+y*np.
                    cos(theta), x_lower, x_upper,
                    y_lower, y_upper)
                filter_j[n, m] =result

        filters_list.append(filter_j)

    filters=torch.stack(filters_list)
    circle_filters=torch.unsqueeze(filters, 1) #

    return circle_filters

# define convolutional one layer
def COL_regularization (eta1,eta2,d,s):
    mask = torch.zeros((eta2, eta1,2*s+1,2*s+1), dtype=
       torch.float32)
    for i in range(eta1):
            for j in range(eta2):
                if abs(i / eta1 - j / eta2)* 2 * np.pi
                   < d or abs(i / eta1 - j / eta2)* 2 *
                    np.pi > 2 * np.pi - d:
                    mask[j,i,:, :] = torch.ones((2*s
                       +1,2*s+1), dtype=torch.float32)
    return mask

```

```
42 #Construct a neural network
43 class CF_COL(nn.Module):
44     def __init__(self, eta1, s1, eta2, s2, d):
45             super().__init__()
46 
47             self.layer1_weight =
                   initialize_circle_filters(eta1, s1)
48             self.layer1 = nn.Conv2d(1, eta1,
                   kernel_size=2*s1+1, stride=1, padding=1,
                    bias=False)
49             self.layer1.weight = nn.Parameter(self.
                   layer1_weight)
50             self.layer1.weight.requires_grad = False
51             self.relu1 = nn.ReLU()
52             self.maxpool1 = nn.MaxPool2d(kernel_size=2,
                    stride=2)
53 
54 
55             self.mask = COL_regularization(eta1, eta2,
                   d, s2)
56             self.layer2 = nn.Conv2d(eta1, eta2,
                   kernel_size=2*s2+1, stride=1, padding=1)
57 
58             self.layer2.weight.data = self.layer2.
                   weight.data * self.mask
59             self.relu2 = nn.ReLU()
60             self.maxpool2 = nn.MaxPool2d(kernel_size=2,
                    stride=2)
61             self.fc_layers = nn.Sequential(
62                 nn.Flatten(),
63                 nn.Linear(64 * 7 * 7, 128),
64                 nn.ReLU(),
65                 nn.Linear(128, 10)
66             )
67 
68     def forward(self, x):
```

```
69          x = self.layer1(x)
70          x = self.relu1(x)
71          x = self.maxpool1(x)
72          x = self.layer2(x)
73          x = self.relu2(x)
74          x = self.maxpool2(x)
75          x = self.fc_layers(x)
76          return x
```

Acknowledgements

I would like to deliver my sincere gratitude to my advisor, Prof.Yifei Zhu, who provided guidance in my theoretical studies and project practicals. He not only offered me a wealth of academic resources, but also provided insightful and constructive feedback during communication.

Additionally, I wish to thank Zhiwang Yu and Qingrui Qu for their kind help during the research. When I encountered difficulties in understanding literature or writing code, they were always patient in helping me solve the problems.

Finally, I'm really grateful to my family and friends who have offered lots of support during my preparation for the paper. Their company gives me the courage to persistently overcome difficulties, and move forward with greater determination.