CLC _____ Number _____

UDC _____ Available for reference ☐Yes ☐No

SUSTech Southern University of Science and Technology

Undergraduate Thesis

Thesis Title: **Euler Characteristic Transform and Its Applications to Deep Learning**

Student Name: **Dongyuan Deng**

Student ID: **12212401**

Department: **Department of Mathematics**

Program: **Mathematics and Applied Mathematics**

Thesis Advisor: **Yifei Zhu**

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: 2026. 5. 28

Euler Characteristic Transform and Its Applications to Deep Learning

[ABSTRACT]: Topological Data Analysis (TDA) provides mathematical tools for obtaining the geometric and topological features of data. Among these tools, the Euler Characteristic Transform (ECT) is an efficient shape descriptor that scans an embedded object from multiple directions and collects the Euler characteristic of its sublevel sets. In this thesis, we study the Euler Characteristic Transform and its applications to deep learning from both experimental and theoretical perspectives.

In terms of theoretical framework, we first review relevant mathematical foundations, and then give a brief introduction to the injectivity of the ECT and a discussion on the stability of the ECT for a perturbed embedding.

Based on these, we focus on the work by Röell and Rieck, by introducing Differentiable Euler Characteristic Transform (DECT) and reproducing their experimental results. These experiments demonstrate the performance of DECT applied to point cloud classification and graph classification. Here ECT-based representations are implemented as differentiable feature layers whose output are fed into neural network architectures for supervised learning tasks. In the reproduction, we also examine an implementation approach adopted in the Röell and Rieck's original code, where image intensity information is incorporated into the DECT-based experiments.

Furthermore, we introduce the Weighted Euler Curve Transform (WECT) of Jiang et al., which extends the ECT by using intensity information through a weight function on the underlying complex embedded from the image. The acronym WECT appears with slightly different expansions in the literature. Qi-

tong Jiang et al. introduced it as the Weighted Euler Curve Transform, while the Röell and Rieck referred to it as the Weighted Euler Characteristic Transform. In this thesis, these two terms are used interchangeably without ambiguity.

Building on the reproduced DECT experiments, we construct differentiable WECT representations for image data and feed them into simple neural network architectures. Moreover, we combine both WECT and the adopted approach in the Röell and Rieck's original code that can preserve the intensity of the image, and evaluate their effect on the model. Finally, based on the work by George et al., we extend the stability discussion to the weighted setting.

[Keywords]: Topological Data Analysis; Euler Characteristic Transform; Weighted Euler Curve Transform; Topological Deep Learning

[摘要]：拓扑数据分析（Topological Data Analysis, TDA）为获取数据几何特征和拓扑特征提供了数学工具。在这些工具中，欧拉示性数变换（Euler Characteristic Transform, ECT）通过从多个方向扫描嵌入对象，并记录其下水平集的欧拉示性数，而成为一种高效的描述形状的特征。本文从实验和理论两个角度研究欧拉示性数变换及其在深度学习中的应用。

在理论角度，我们首先回顾了一些数学基础，然后简要介绍了欧拉示性数变换的单射性, 同时简要讨论了该变换在扰动嵌入下的稳定性。

在这些基础上，本文主要关注 Ernst Röell 和 Bastian Rieck 的工作。我们主要介绍可微欧拉示性数变换（Differentiable Euler Characteristic Transform, DECT），并复现其实验。复现的实验主要关注 DECT 的引入在点云分类和图分类任务的表现。这些实验将基于 ECT 的表示实现为可微特征层，随后作为输入接入神经网络结构，并用于监督学习任务。在复现过程中，我们根据作者的实验代码，还研究了一种能够将图像强度信息用于 DECT 实验的方法。

此外，本文介绍了 Qitong Jiang 等人提出的加权欧拉曲线变换（Weighted Euler Curve Transform, WECT）。该方法利用强度信息构造图像数据嵌入产生的底层复形上的权重函数，从而扩展了 ECT。缩写 WECT 在相关文献中的全称略有不同。Qitong Jiang 等人称其全名为 Weighted Euler Curve Transform（加权欧拉曲线变换），而 Ernst Röell 和 Bastian Rieck 则将其称为 Weighted Euler Characteristic Transform（加权欧拉示性数变换）。本文不区分这两个术语，并在不造成歧义的情况下交替使用。

基于复现的 DECT 实验，我们为图像数据构造了可微的 WECT 表示，并将其与简单的神经网络结构相结合。此外，由于 WECT 和 Ernst Röell 及 Bastian Rieck 实验代码中使用的方法都有保留强度信息的作用，所以我们结合并研究它们对模型的影响。最后，基于 Jasmine George 等人的工作，我们在 WECT 条件下拓展了关于扰动嵌入下 ECT 的稳定性的讨论。

[关键词]：拓扑数据分析；欧拉示性数变换；加权欧拉曲线变换；拓扑深度学习

Contents

1. Introduction

Topological Data Analysis provides mathematical methods for extracting topological and geometric structure from datasets. Persistent homology is one of the most widely used tools in TDA, which studies how topological features appear and disappear along a filtration and summarizes these informations by persistence diagrams. Persistence diagrams provide rich multi-scale information, when they are used in standard machine learning or deep learning models, additional vectorization, kernel-based functions are usually required[1]. These transformations, like PH itself, suffer from computational limitations that preclude its application to large-scale data sets[2].

The Euler Characteristic Transform introduced by Turner et al. is another particularly useful descriptor[3]. Given data embedded in Euclidean space, the ECT scans the shape from multiple directions. For each fixed direction, the ECT records the Euler characteristics of the sublevel sets of the corresponding height function as the threshold varies, thereby producing an Euler characteristic curve (ECC). Thus the collection of such curves over different directions forms the ECT. Moreover, Ghrist et al.[4] and Curry et al.[5] have proved some conclusions related to the injectivity of ECT. Furthermore, George et al. discussed the stability of ECT for a perturbed embedding[6].

Based on these theoretical foundations, there are some studies combine the ECT and machine learning by setting ECT ad its variants as static features that require specific hyperparameter choices[7]. By contrast, Röell and Rieck made the ECT end-to-end trainable, by introducing DECT, which provides an efficient and effective shape descriptor that can be integrated into deep learning models, such as multilayer perceptron (MLP) and convolutional neural network (CNN)[2].

To make use of the intensity information in image data, the Weighted Euler Curve Transform, a variant of ECT, was introduced and incorporated into a machine learning model, namely a support vector machine (SVM)[7].

This paper reviews the DECT framework and reproduces representative experiments

that combine DECT layers with deep learning models for point cloud and graph classification. Motivated by the role of intensity information in image-based data, we firstly adopt an engineering approach that incorporates intensity as an additional geometric coordinate[2]. Secondly, we implement differentiable WECT representations that encodes intensity through a weight function on the underlying complex. After evaluating these approaches on image classification tasks, we preliminarily extend the stability perspective to WECT, motivated by the work of George et al.[6].

2. Preliminaries

In this section, we will review some basic definitions

2.1 Euler Characteristic

Definition 2.1.1. Given $m+1$ distinct points $\{x_1, x_2, ..., x_m\} \subseteq \mathbb{R}^n$, the set $\{x_i\}_{i=0}^m$ is ***affinely independent*** if the set $\{x_i - x_0\}_{i=1}^m$ is linearly independent.

Definition 2.1.2. A ***simplex*** spanned by the affinely independent set $\{x_i\}_{i=0}^m$, denoted by $[x_0, \ldots, x_m]$, is the set

$$[x_0, \ldots, x_m] = \left\{ \sum_{i=0}^{m} t_i x_i : t_i \geq 0 \, , \sum_{i=0}^{m} t_i = 1 \right\}.$$

Definition 2.1.3. A finite ***simplicial complex*** K is a finite collection of simplices in some Euclidean space such that:

(i) if $s \in K$, then every face of s also belongs to K;

(ii) if $s, t \in K$, then $s \cap t$ is either empty or a common face of s and t.

Definition 2.1.4. ***Euler characteristic*** is defined as the alternating sum of the number of simplices in each dimension. For a simplicial complex K, we define the Euler characteristic χ as

$$\chi(K) = \sum_{n=0}^{\dim K} (-1)^n |K^n|, \tag{1}$$

where $|K^n|$ denotes the amount of the n-dimensional simplices in K. The Euler characteristic is a homotopy invariant. By the Euler–Poincaré formula, the Euler characteristic can also be

expressed as the alternating sum of the Betti numbers:

$$\chi(K) = \sum_{d=0}^{\dim K} (-1)^d \beta_d(K), \tag{2}$$

where $\beta_d(K)$ denotes the d-th Betti number of K.

2.2 Persistence Diagrams

The presentation here mainly follows George[6].

Definition 2.2.1. Fix a given simplicial complex K, a real-valued function $f : K \to \mathbb{R}$, and a sequence of values $a_1 < a_2 < \cdots < a_n$. A ***filtration*** is the collection of sublevel sets

$$K_{a_i} := f^{-1}((-\infty, a_i]).$$

Note that for all $a \leq b$, there are inclusions

$$K_a \subseteq K_b.$$

Additionally, we take $a_0 = -\infty$ in order to set

$$K_{a_0} = \emptyset.$$

The filtration induced by f is written as $\mathcal{F}_f$.

Definition 2.2.2. A function f on a simplicial complex K is **simplex-wise monotone** if for all subcomplexes $\tau \subset \sigma$, the function satisfies

$$f(\tau) \leq f(\sigma).$$

It is straightforward to prove that if f is simplex-wise monotone, then for any $t \in \mathbb{R}$, the sublevel set

$$K_t := f^{-1}((-\infty, t])$$

is a subcomplex of K. For the purposes of describing the filtration as a nested sequence of

subcomplexes, the actual values of a_i are immaterial. Only the induced ordering

$$K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K$$

is relevant.

Definition 2.2.3. Let

$$\mathcal{F}_f = \{K_i\}_{i \in \mathbb{N}}$$

be a filtration of a simplicial complex K. For a fixed dimension k, the ***k*-dimensional persistence diagram**, denoted by $\mathrm{Dgm}_k(\mathcal{F})$, is a multiset of points (b, d) in the extended plane

$$\mathbb{R}^{2+} := (\mathbb{R} \cup \{\infty\})^2,$$

where b is the filtration value at which a k-dimensional homology class appears and d is the filtration value at which it disappears. The value b is called the birth time, and d is called the death time. If a homology class never disappears during the filtration, then its death time is taken to be ∞. Thus, each point in $\mathrm{Dgm}_k(\mathcal{F})$ records the lifetime of one topological feature in dimension k.

Definition 2.2.4. Let Δ^∞ denote countably many copies of the diagonal

$$\Delta = \{(x, x) \mid x \in \mathbb{R}\}.$$

A ***matching*** between two persistence diagrams D_1 and D_2 is a bijection

$$\phi : D_1 \cup \Delta^\infty \longrightarrow D_2 \cup \Delta^\infty.$$

The copies of Δ allow off-diagonal points in one diagram to be matched to the diagonal when they have no corresponding off-diagonal point in the other diagram.

Definition 2.2.5. Let D_1 and D_2 be two persistence diagrams, and let matching between D_1 and D_2 is regarded as

$$\phi : D_1 \cup \Delta^\infty \longrightarrow D_2 \cup \Delta^\infty.$$

For $1 \leq p < \infty$ and $1 \leq q \leq \infty$, the (p, q)-***Wasserstein distance*** between D_1 and D_2 is defined by

$$W_{p,q}(D_1, D_2) = \inf_{\phi} \left(\sum_{x \in D_1 \cup \Delta^\infty} \|x - \phi(x)\|_q^p \right)^{1/p},$$

where the infimum is taken over all matchings ϕ.

3. Methods

The Euler characteristic provides a compact topological summary of a finite simplicial complex. However, as a single integer, it describes the object only at one scale and therefore loses many geometric informations of an embedding of a data, while persistent homology is infeasible to calculate for large data sets. To obtain a more informative descriptor at multi scale, Turner et al. proposed the Euler Characteristic Transform[3].

3.1 Euler Characteristic Curves

Let K be a finite simplicial complex and let $f : K \to \mathbb{R}$ be a simplex-wise monotone filtration function. ***The Euler Characteristic Curve*** induced by f is defined by

$$\mathrm{ECC}_{K,f}(t) = \chi(K_t).$$

Thus, instead of assigning one Euler characteristic to the whole complex, the ECC records how the Euler characteristic changes as simplices enter the filtration.

This construction can be understood as a multi-scale version of the Euler characteristic. At small filtration values, only a small part of the complex is included. As the threshold increases, vertices, edges, and higher-dimensional simplices are gradually added. Each newly added simplex changes the Euler characteristic according to its dimension: vertices contribute positively, edges negatively, triangles positively, and so on. Therefore, the ECC encodes the evolution of topological structure along the chosen filtration. Compared with persistent homology, the ECC is less expressive, because it keeps only the alternating sum of Betti numbers rather than the full birth-death information. Nevertheless, it is much cheaper to compute and easier to vectorize, which makes it attractive in machine learning settings.

3.2 Euler Characteristic Transform

The Euler Characteristic Curve depends on the choice of the filtration function. For an embedded simplicial complex, a family of filtrations can be obtained naturally from height functions. Let $K \subset \mathbb{R}^d$ be a finite geometric simplicial complex and let $v \in S^{d-1}$ be a unit direction. For a vertex, the 0-dimensional simplex $x \in K$, define the ***height function*** with the direction v by

$$h_v(x) = \langle x, v \rangle.$$

For a general simplex σ, this function is extended to be simplex-wise monotone by the maximum rule

$$h_v(\sigma) = \max_{x \in V(\sigma)} h_v(x),$$

where $V(\sigma)$ denotes the set of vertices of σ.This convention ensures that a subcomplex enters the filtration only after all of its faces have entered. Thus

$$K_{v,t} = \{\sigma \in K : h_v(\sigma) \leq t\}.$$

is a subcomplex for every $t \in \mathbb{R}$.

For a fixed direction v, ***the Euler Characteristic Transform*** is obtained by collecting these curves over all directions:

$$\mathrm{ECT}_K : S^{d-1} \times \mathbb{R} \to \mathbb{Z}, \qquad (v, t) \mapsto \chi(K_{v,t}).$$

$$\mathrm{ECT}_K(v, t) = \mathrm{ECC}_{K,h_v}(t) = \chi(K_{v,t}).$$

Intuitively, the ECT scans the embedded complex from many directions. Each direction gives one Euler characteristic curve, and the full transform is the collection of all such directional summaries.

This directional viewpoint is important because a single ECC may miss geometric information that is invisible from one direction. By contrast, the ECT combines multiple scans and therefore captures both topological and geometric features of the embedded complex.

The ECT provides a mathematically meaningful way to describe the shape of an embedded object. It combines geometric and topological information, coming from the embedding, the choice of scanning directions and the Euler characteristic of the induced sublevel complexes. However, in practice, one cannot evaluate the transform over all directions in S^{d-1} or over all thresholds in $\mathbb{R}$. Therefore, a computational implementation must replace the continuous transform by a finite approximation. This requires choosing a finite set of directions

$$V = \{v_1, \ldots, v_k\} \subset S^{d-1}.$$

and a finite set of threshold values

$$T = \{t_1, \ldots, t_l\} \subset \mathbb{R}.$$

This finite approximation is natural from a computational point of view, but it introduces several design choices. The number of directions, the way these directions are sampled, and the number of filtration thresholds all become hyperparameters.

In classical applications of the ECT, these choices are typically fixed before training. For instance, one may use uniformly sampled directions or directions chosen from a predefined grid. Such choices are reasonable as general-purpose approximations, but they are not necessarily optimal for a given data set or a given classification task, since the underlying structure of most data cannot be specified a priori or directly visualized. For example, symmetries of the underlying shape may reduce the number of independent directions needed to represent the ECT. For instance, if an embedded complex K is invariant under reflection across the equatorial plane, then each direction in the lower hemisphere has a reflected counterpart in the upper hemisphere that yields the same Euler characteristic curve. Hence, when this symmetry is known a priori, the ECT can be represented without loss of information by restricting the set of directions to a fundamental domain, such as the upper hemisphere. In particular, if the relevant geometric or topological differences between two classes are most visible from certain directions, a fixed uniform sampling may include many uninformative

directions while missing or under-emphasizing more discriminative ones.

In addition to the choice of directions, the filtration parameter also has to be discretized in practical computations. For a fixed direction v, the ECC is a function of a continuous threshold $t \in \mathbb{R}$. For each input complex K, the discretized ECT can then be represented as an $l \times k$ matrix, whose entry at (i, j) is given by

$$\chi(K_{i,j}) := \chi(K_{t_i,v_j}).$$

In this representation, the columns correspond to different directions, while the rows correspond to filtration thresholds. Since the same threshold set T is used for all directions, the ECCs are aligned across columns: the same row index always refers to the same filtration value.This matrix can be used as a numerical feature representation for a classifier.

An alternative strategy is to choose a separate threshold set T_v for each direction v. This can be useful when the effective range of filtration values varies significantly from one direction to another. Under this strategy, each column of the matrix still represents one ECC, but the same row index in different columns no longer necessarily corresponds to the same filtration value. Thus, the first strategy preserves a common filtration scale across all directions, whereas the second strategy normalizes or adapts the sampling range direction by direction. Following the discussion in Röell and Rieck[8], the common-threshold strategy is natural when the data are contained in, or normalized to, a common bounded region, while the direction-dependent strategy may be appropriate when relative shape variation within each ECC is more important than comparing absolute filtration values across different directions.

However, in this paper we focus on the strategy that aligne different directions with the same ordered set of thresholds.

This point is especially important in machine learning. A learning model is expected to adapt its representation to the data. However, if the ECT is computed only from a manually chosen set of directions, then the topological feature extraction step itself remains fixed. The classifier can learn how to use the resulting ECT features, but it cannot learn how the

ECT features should be produced. In other words, the directions are treated as external hyperparameters rather than trainable parameters. The resulting pipeline is therefore only partially learnable: the downstream classifier is trained from data, whereas the directions used to construct the ECT are selected manually.

The difficulty is not merely practical, but also analytic. The classical ECT can be expressed as an alternating sum of indicator functions. For a simplex $\sigma \in K$, let $h_v(\sigma)$ denote its filtration value in direction v. Then given a drection v and a threshold t, the ECT can be written in the form

$$\mathrm{ECT}_K(v,t) = \sum_{q=0}^{\dim K} (-1)^q \sum_{\sigma \in K^q} \mathbf{1}_{[h_v(\sigma),\infty)}(t).$$

This expression is exact and makes the contribution of each simplex explicit. A simplex contributes to the Euler characteristic once the threshold t reaches its filtration value. Nevertheless, the indicator function is discontinuous. As a result, the ECT is not differentiable with respect to the direction v, the threshold t, or the vertex coordinates that determine $h_w(\sigma)$. Consequently, gradient-based optimization cannot directly adjust the scanning directions or the input coordinates through the ECT computation.

3.3 Differentiable Euler Characteristic Transform

The Differentiable Euler Characteristic Transform addresses this issue by replacing the indicator function with a smooth approximation. Let

$$S(\lambda x) = \frac{1}{1+\exp(-\lambda x)}$$

be the sigmoid function, and let $\lambda > 0$ be a scale parameter. The differentiable approximation is defined by

$$\mathrm{DECT}_{K,\lambda}(v,t) = \sum_{q=0}^{\dim K} (-1)^q \sum_{\sigma \in K^q} S\big(\lambda(t - h_v(\sigma))\big).$$

The parameter λ controls the sharpness of the approximation. For large λ, the sigmoid becomes close to a step function, and the DECT approximates the ordinary ECT more tightly. For smaller λ, the approximation is smoother, which may improve gradient behavior during

optimization.

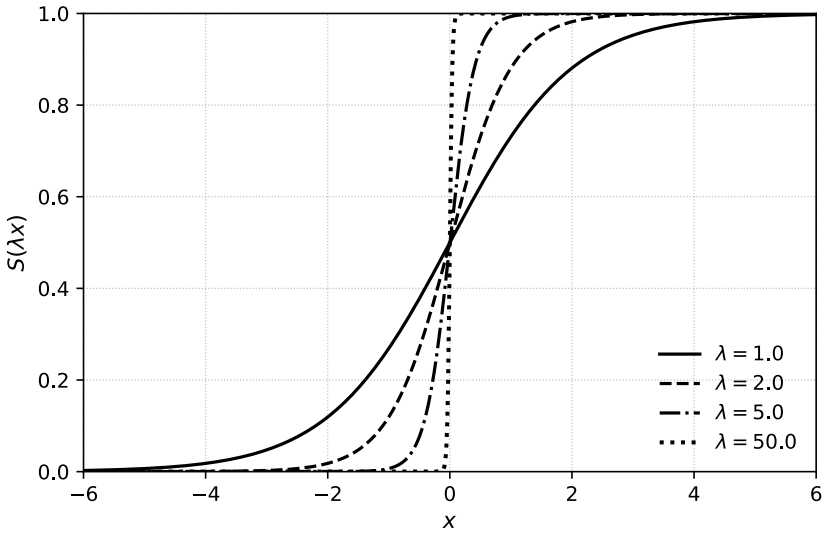

Figure 1. Sigmoid curves for different scale parameters λ. Larger values of λ yield a sharper approximation to the step function near the origin.

This modification changes the role of the ECT in a machine-learning pipeline. Instead of being only a static descriptor computed before training, the transform can be implemented as a differentiable layer. The directions $v_1, \ldots, v_k$ may be initialized, for example, by uniform sampling on the sphere, but they can subsequently be optimized together with the other parameters of the neural network. Thus, the model can learn which directions are useful for the supervised task.

It is important to emphasize that DECT does not remove all hyperparameters. One still has to choose the number of directions k, the number of thresholds l, and the scale parameter λ. However, the crucial difference is that the directions themselves can be treated as learnable quantities. This reduces the dependence on purely hand-designed ECT features and turns the transform into a trainable topological representation. In this sense, DECT can be viewed as a bridge between a theoretically motivated topological invariant and the end-to-end optimization framework of deep learning.

As discussed above, the ECT must be evaluated on a finite set of directions and thresholds in practical computations. The same discretized structure is retained in DECT. Consequently, the resulting matrix has the same image-like organization as the discretized ECT,

with directions indexed along one axis and filtration thresholds along the other, but its entries are real-valued differentiable approximations rather than exact integer-valued Euler characteristics.

The Figure2 to Figure4 below illustrate a finite sampled visualization of the DECT representation,[1] They should be interpreted as visual explanations of how the transform varies over sampled directions and filtration thresholds, rather than as the exact input tensors (matrices) used in the neural-network models. The vertical axis represents the sampled uniformly filtration thresholds $t \in [-1, 1]$, the horizontal axis represents sampled uniformly directions from S^1 and the color intensity indicates the corresponding $\mathrm{DECT}_{K,\lambda}(v, t)$.

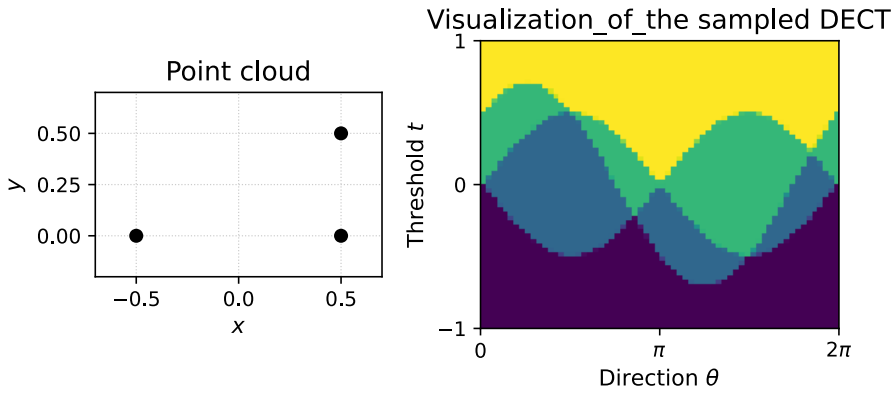

Figure 2. Point cloud and its ECT.

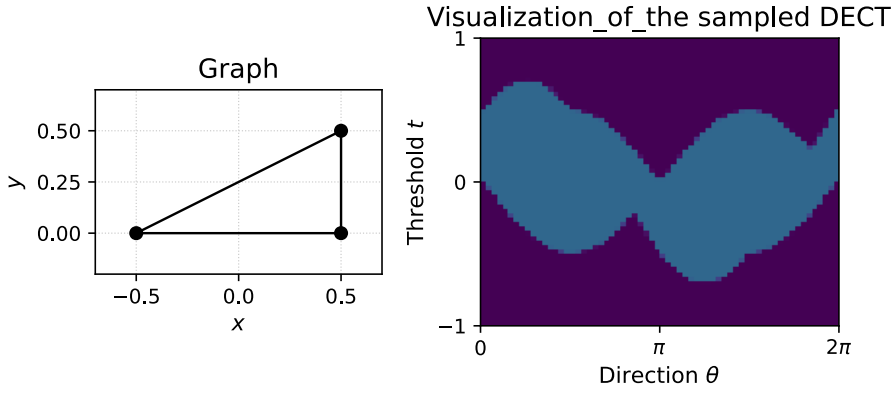

Figure 3. Graph and its ECT.

[1]The code used to generate this visualization is provided in the appendix.

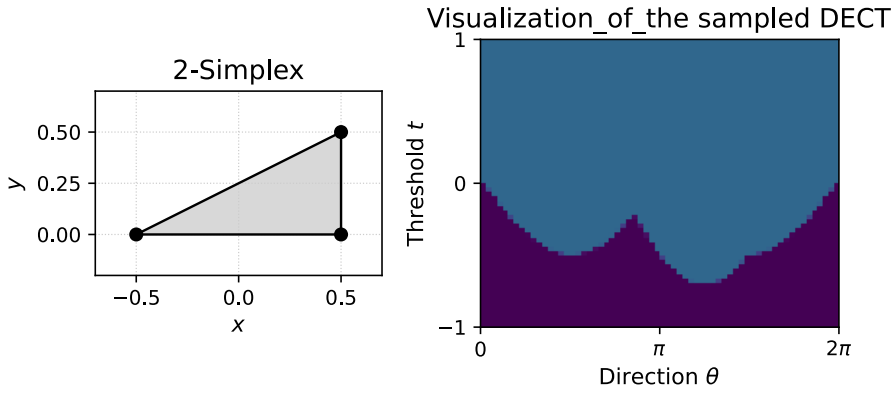

Figure 4. Simplicial complex and its ECT.

3.4 Weighted Euler Curve Transform

However, in many data sets, each vertex or even simplex may carry additional information. For instance, in image data, pixel intensity contains information that is not captured by the binary shape alone. To incorporate such scalar information, Jiang et al. introduced the Weighted Euler Curve Transform[7].

Let $K \subset \mathbb{R}^d$ be a finite embedded simplicial complex and let

$$g : K \to \mathbb{N}$$

be a ***weight function.*** The pair (K, g) is called a weighted simplicial complex. ***The Weighted Euler characteristic*** is defined by

$$\chi(K, g) = \sum_{q=0}^{\dim K} (-1)^q \sum_{\sigma \in K^q} g(\sigma).$$

If $g(\sigma) = 1$ for every simplex σ, then this definition reduces to the ordinary Euler characteristic. Thus, the weighted Euler characteristic is a direct extension of the classical one.

A weight function function $g\colon K \to \mathbb{N}$ which satisfy the consistency condition

$$g(\tau) = \max\{g(\sigma) \mid \tau < \sigma\}$$

is ***admissible***.

In this paper, we focus on admissible weight functions.

In the image setting, the weight function can be used to encode pixel intensities. The binary support of the image determines the shape, while the grayscale values provide additional scalar information on the simplices. In this sense, the WECT is designed to represent both geometry and intensity.

Given a simplex-wise monotone filtration function $f : K \to \mathbb{R}$, ***the Weighted Euler curve*** is defined by

$$\mathrm{WECC}_f^g(t) = \chi\big(f^{-1}((-\infty, t]), g\big),$$

where g is restricted to the subcomplex $f^{-1}((-\infty, t])$.

For a height function $h_v : K \to \mathbb{R}$ with the direction $v \in S^{d-1}$, and a threshold $t \in \mathbb{R}$, ***the Weighted Euler Transform*** is defined by

$$\mathrm{WECT}_{K,g}(v, t) = \mathrm{WECC}_{h_v,g}(t) = \chi\big(h_v^{-1}((-\infty, t]), g\big).$$

Equivalently, we can write it as

$$\mathrm{WECT}_{K,g}(v, t) = \sum_{q=0}^{\dim K} (-1)^q \sum_{\sigma \in K^q} g(\sigma) \mathbf{1}_{[h_v(\sigma),\infty)}(t).$$

This formula makes clear that the WECT differs from the ordinary ECT by weighting each simplex contribution. Instead of merely counting simplices with signs, the transform sums their weights with signs. Therefore, the WECT can distinguish cases where the underlying support is similar but the scalar field on the support is different.

Analogously, we define a differentiable approximation of the WECT by replacing the indicator functions with sigmoid functions:

$$\mathrm{DWECT}_{K,g}^{\lambda}(v, t) = \sum_{q=0}^{\dim K} (-1)^q \sum_{\sigma \in K^q} g(\sigma) S\big(\lambda(t - h_v(\sigma))\big),$$

where S denotes the sigmoid function and $\lambda > 0$ controls the sharpness of the approximation. We refer to this approximation as the Differentiable Weighted Euler Characteristic Transform

(DWECT).

Moreover, sampling finite many directions and filtration thresholds gives a practical discretization of DWECT.

4. Experiments

In the previous sections, we introduced the mathematical background of the ECT, its weighted variant, and the differentiable formulation used in neural network models. The goal of this section is to examine whether these topological descriptors can be used effectively in supervised learning tasks[2]. In particular, we first reproduce the point-cloud classification experiment from the paper[2], which we refer to it as Reproduction 5.1 and then modify the setting by incorporating intensity information of the original images, where we encode the differentiable WECT as a layer. We further study the graph classification experiment from the same paper by replacing the original graph dataset with MNISTSuperpixels and replacing the DECT layer with our differentiable WECT layer.To preserve the intensity information, we also adopt an implementation choice from the authors' public evaluation code,[3] where pixel intensity is treated as an additional geometric coordinate.

4.1 Reproducing experiment on MNIST

4.1.1 Preprocessing

At first we reproduces the experiment 5.1 in the paper by Röell and Rieck[2]. This experiment is designed as a simple test of whether the directions used in the DECT can be treated as trainable parameters in an end-to-end classification model.

We use the MNIST handwritten digit dataset We split data sets following an$80\%/20\%$ split. Each image is a grayscale image of size 28×28, together with a label in $\{0, \ldots, 9\}$. In this experiment, the image is not treated as a usual pixel grid. Instead, each non-zero pixel is regarded as a point in a two-dimensional point cloud. Thus, every digit image is converted

[2]The codes of all experimentsin this section are released on the repository:`https://github.com/DengG-T/ect-applications-deep-learning`.

[3]The implementation refers to the official DECT evaluation repository released by Röell and Rieck: `https://github.com/aidos-lab/dect-evaluation`.

into a finite subset of $\mathbb{R}^2$. More specifically, all positive-intensity pixels are mapped from image coordinates into the unit disc D^2, and the DECT layer is applied to this point cloud representation.

It should be emphasized that, in this first reproduction experiment, the grayscale intensity is only used to decide whether a pixel is included in the point cloud. The intensity value is not used as an additional coordinate and is not used as a weight. This is different from the later WECT-based experiment, where pixel intensity will be incorporated into the topological representation.

4.1.2 Model architecture

The model consists of a DECT layer followed by a ***multilayer perceptron (MLP)***. Instead of using a large number of directions to approximate the full ECT, the experiment deliberately restricts the model to only two directions sampled uniformly from S^1 and 16 thresholds sampled uniformly from $[-1, 1]$. This sparse setting makes the comparison between fixed and learnable directions more explicit. This output is a matrix of the size 2×16 then flattened into a vector of dimension 32.

The classifier is an MLP with 3 hidden layers. Each hidden layer has 25 hidden units and uses the ReLU activation function. The final linear layer maps the hidden representation to 10 output logits, corresponding to the ten MNIST classes. The model is trained with the categorical cross-entropy loss and we use the ADAM optimiser with a starting learning rate of 0.001.

4.1.3 Procedure and Results

Following the experimental setting in the paper[2], we trained each model for 10 epochs and repeat the experiment 10 times, and compared two variants: one with fixed directions and one with learnable directions, as the paper[2] says. This distinction does not change the definition of the ECT or the DECT computation itself; it only controls whether the sampled directions are treated as trainable parameters and updated by ***backpropagation***. The results

suggest that the directions used in DECT can be learned as model parameters through end-to-end training.

To further investigate the effect of the number of directions on classification performance, we increased the number of directions from 2 to 8. This gives a denser sampling of the unit circle and provides a more complete finite scan of each point cloud. However, in the experiment(Experiment 4.1.1) with 8 directions and 25 hidden units in each hidden layer, the fixed-direction model achieved higher accuracy than the learnable-direction model. To examine whether this phenomenon was caused by overfitting, we checked the validation accuracy during training. Since the validation accuracy did not show a clear deterioration, we did not observe strong evidence of overfitting. Then we changed the units in every hidden from 25 units to 128 units for a lager neural network, while keeping the other experimental settings unchanged(Experiment 4.1.2). Then the results suggested that increasing the number of directions can improve the classification performance. The full results are summarized in Table 1.

Table 1. Classification results of DECT-based models on MNIST under different direction settings.

Experiment	D	R	Units/layer	Directions	Accuracy(%)
Reproduction 5.1	2	16	25	Fixed	74.94 ± 0.53
	2	16	25	Learnable	82.05 ± 2.54
Experiment 4.1.1	8	16	25	Fixed	87.82 ± 1.39
	8	16	25	Learnable	85.11 ± 7.14
Experiment 4.1.2	8	16	128	Fixed	92.67 ± 0.36
	8	16	128	Learnable	92.94 ± 0.34

D denotes the number of directions and R denotes the number of filtration thresholds.

4.2 WECT-based Experiment on MNIST

Based on the paper by Jiang et al.[7], the authors have already conducted experiments that combine ECT and WECT representations with SVM. Their WECT representations are computed using the same strategy of discretization method discussed above, but they are used as fixed, non-differentiable features in a classical machine-learning pipeline.

In this experiment, we do not aim to reproduce the WECT-based machine-learning ex-

periment in[7]. Instead, inspired by the Algorithm1 in[7], which converts a grayscale image into a weighted complex, we use this construction to transform the MNIST dataset into weighted simplicial complexes, which are the input of a differentiable WECT layer. After that, we feed its output into a neural network for the classification task.

4.2.1 Preprocessing

When all non-zero pixels are treated as points in $\mathbb{R}^2$, each pixel is represented by its center point on a regular square grid. Horizontally or vertically adjacent pixel centers are separated by unit distance. We refer to these points as ***center points***, in order to distinguish them from the ***corner points*** introduced later.

Next, for each center point $c = (i, j)$, we add four corner points

$$c + \left(\pm\frac{1}{2}, \pm\frac{1}{2}\right),$$

which correspond to the four corners of the unit square pixel centered at c. Since adjacent pixels may share the same corner points, duplicate corner points are removed. The vertex set is then given by the union of the center points and the corner points.

We then construct faces by dividing each unit square pixel into four triangles, each of which contains the center point c and two adjacent corner points. The edge set is obtained as the collection of all one-dimensional faces of these triangles, with duplicate edges removed. Therefore, each non-zero pixel contributes four triangular faces, and the union of these faces over all non-zero pixels forms a two-dimensional simplicial complex, as shown in Figure 5.

Recall that the weight function is required to be ***admissible***. We define the weight function g on the constructed simplicial complex as follows. For each center point, its weight is given by the intensity of the corresponding original pixel. Each triangular face contains exactly one center point, so the weight of a face is defined as the weight of its corresponding center point. For an edge, its weight is defined as the maximum weight among all incident faces. Similarly, the weight of a corner point is defined as the maximum weight among all

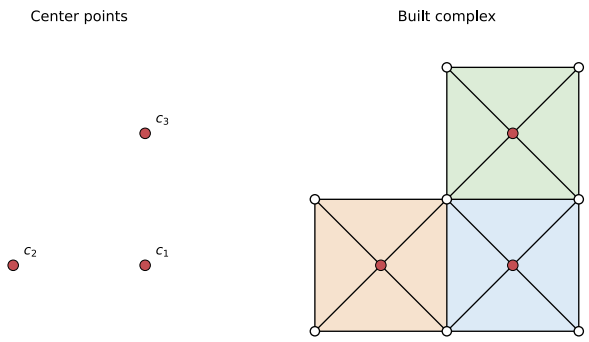

Figure 5. Construction of a weighted simplicial complex from three pixel centers. Each pixel center is augmented with its four corner points, and each pixel square is subdivided into four triangles.

incident faces containing it.

4.2.2 Model architecture

The model architecture follows the same general structure as that in Section 4.1.2. The main difference is that the DECT layer is replaced by the proposed DWECT layer. In addition, the number of hidden units in the neural network is set to 128, instead of 25 used in the previous experiment.

4.2.3 Procedure and Results

The experimental procedure is similar to that in Section 4.1.3. Since the fixed-direction setting has already been discussed in the previous experiment, we only consider learnable directions in this part and the left sections. Specifically, we conduct two groups of experiments, using 4 and 8 learnable directions, respectively. The results are shown in Table 2. The results are summarized in Table 2.

From Table 2, we observe that the DWECT-based neural-network models achieve higher classification accuracy than the WECT–SVM model reported in[7], while using a much smaller topological sampling size. In particular, the WECT–SVM model uses 25 directions and 50 filtration thresholds, corresponding to $25 \times 50 = 1250$ sampled values. By contrast, our DWECT-based models use only $4 \times 16 = 64$ and $8 \times 16 = 128$ sampled values, respectively, but still achieve higher accuracies of 95.20% and 96.91%. This suggests that combining a

Table 2. Comparison of ECT-, WECT-, DECT-, and DWECT-based models on MNIST.

Experiment	Representation	Model	D	R	Units/layer	Directions	Accuracy(%)
Jiang et al.[7]	ECT	SVM	25	50	–	Fixed	89.88 ± 1.66
Jiang et al.[7]	WECT	SVM	25	50	–	Fixed	94.68 ± 1.57
Experiment 4.1.2	DECT	MLP	8	16	128	Learnable	92.94 ± 0.34
This experiment	DWECT	MLP	4	16	128	Learnable	95.20 ± 0.16
	DWECT	MLP	8	16	128	Learnable	96.91 ± 0.17

D denotes the number of directions and R denotes the number of filtration thresholds. The symbol "–" means that the corresponding setting is not applicable to the SVM-based model.

differentiable WECT representation with a neural network can make more effective use of the sampled topological features than the classical SVM-based pipeline.

The comparison with the DECT-based model further shows the advantage of incorporating weights. Even with 4 directions and 16 thresholds, the DWECT-based model outperforms the DECT-based model with 8 directions and the same number of thresholds. However, this improvement should not be attributed only to the DWECT itself, since the model benefits from both the additional intensity information and the extra geometric structure introduced by the weighted complex construction in the DWECT preprocessing.

4.3 Experiment on GNNBenchmark MNIST

In this section, we reproduce the graph-classification part of Experiment 5.3 in the paper[2], and we refer to it as Reproduction 5.3. The experiment is conducted on the MNIST subset of the GNNBenchmark datasets, which represents handwritten digit images as superpixel graphs. Since our purpose is to reproduce and analyze the DECT-based graph model, we do not reimplement the GNN baselines. The baseline results are taken from the original benchmark and the DECT paper.

4.3.1 Preprocessing

The GNNBenchmark MNIST dataset converts each MNIST image into a graph whose nodes correspond to superpixels and whose edges encode adjacency relations between superpixels. Each graph is labeled by one of the ten digit classes. In particular, each node is associated with a two-dimensional position describing the location of the corresponding

superpixel and a scalar intensity value inherited from the original grayscale image.[9]

Following the implementation of experiment 5.3 in the paper[2], we examine an engineering choice in which intensity information of a image can be preserved. For each node, its two-dimensional position and its intensity value are concatenated into a three-dimensional coordinate vector. Therefore, for a node in that graph, the node-coordinate can be embedded into can be written as

$$X \in \mathbb{R}^3,$$

More precisely, in DECT or DWECT layer, we can sample a direction $v \in \mathbb{S}^2$ and the intensity thus can participate in height function.

4.3.2 Model Architecture

The overall architecture is The three-dimensional embedded graph, is first as the input to the DECT layer. The resulting finite representation is subsequently processed by a convolutional neural network, and the extracted features are flattened and passed to an MLP classifier to produce the final class logits.

The CNN module contains two convolutional blocks. The first block applies a two-dimensional convolution with one input channel, eight output channels, and kernel size 3, followed by a 2×2 max-pooling operation. The second block applies the same as the first block.

The MLP classifier consists of two hidden fully connected layers and one output layer. Both hidden layers have 100 units and are followed by ReLU activations. The output layer has 10 units, corresponding to the ten classes.

4.3.3 Procedure and Results

Reproduction 5.3 consists of five independent runs, with each run trained for 100 epochs. The comparison between Reproduction 5.3 and the result reported in the paper[2] is shown in Table 3.

Table 3. Comparison between Reproduction 5.3 and the result reported in paper[2].

Experiment	Accuracy (%)	Average epoch time (s)
Original result in paper[2]	93.00 ± 0.80	4.50
Reproduction 5.3	93.10 ± 0.35	4.86

4.4 Experiment on MNISTSuperpixels

The MNISTSuperpixels dataset has the same graph-based data format as the GNNBenchmark MNIST dataset. Each sample is represented as a superpixel graph with node positions, node intensity values, graph connectivity, and a digit label. Compared with GNNBenchmark MNIST, MNISTSuperpixels contains a larger number of nodes per graph, giving a finer graph representation of each image.

From the previous section, we have two ways to preserve the intensity information. The first method is DWECT, where intensity values are incorporated as weights in the weighted Euler characteristic. The second method is to treat intensity as an additional coordinate, so that each node is embedded into $\mathbb{R}^3$ and the intensity participates in the height function.

However, it is not clear in advance how these two methods interact. They may provide complementary information and improve the classification performance when used together. They may also be redundant, or one method may weaken the contribution of the other. Therefore, this section investigates the effects of these two methods, both separately and in combination.

4.4.1 Preprocessing

The preprocessing is similar to that in Section 4.3.1, but two additional points should be emphasized.

First, since MNISTSuperpixels already provides graph-structured data, no additional geometric graph structure is constructed. We only define weights on the existing graph. For each node, the weight is given by its intensity value. For each edge, the weight is defined as the minimum of the weights of its two incident nodes. This choice ensures that the edge weight is no larger than the weights of its endpoints, and is therefore compatible with the

superlevel-set construction used in the weighted Euler characteristic.

Second, the intensity value is not directly appended as the third coordinate. Instead, we introduce a hyperparameter, called the ***geometric intensity scale*** c. For a node with two-dimensional position $p_i \in \mathbb{R}^2$ and intensity value I_i, its geometric coordinate is defined as

$$x_i = (p_i, cI_i) \in \mathbb{R}^3.$$

Thus, the actual third coordinate used in the height function is cI_i. When $c = 0$, the third coordinate vanishes, and the embedding is equivalent to the original two-dimensional geometric embedding.

4.4.2 Model Architecture

The model architecture is similar to that in Section 4.3.3. The main difference is that the DECT layer is replaced by a differentiable WECT layer, so that the intensity information can be used as simplex weights in the Euler characteristic computation. The following CNN and MLP classifier remain the same as in the previous experiment.

In addition, we introduce the geometric intensity scale c, which controls how the intensity value is used in the geometric embedding.

To compare the effect of ECT representations, we consider two modes. In the WECT mode, the intensity-based weights defined in the preprocessing step are used in the weighted Euler characteristic computation. In the ECT mode, all simplex weights are set to 1, so the layer reduces to the ordinary ECT computation on the same embedded graph. This allows us to separate the effect of using intensity as a weight from the effect of using intensity as an additional geometric coordinate.

4.4.3 Procedure and Results

We compare three experimental settings: ECT mode with geometric intensity scale $c = 0$, ECT mode with $c = 1$, and WECT mode with $c = 1$. For each setting, we conduct five independent runs, with each run trained for 50 epochs.

The classification results are reported in Table 4.

Table 4. Classification results on MNISTSuperpixels under different settings.

Experiment	Accuracy (%)
ECT mode, $c = 0$	62.64 ± 13.03
ECT mode, $c = 1$	85.50 ± 1.25
WECT mode, $c = 1$	94.71 ± 0.22

The large standard deviation in the $c = 0$ setting indicates that the model performance is highly sensitive to different runs. This suggests that using only the two-dimensional node positions and graph connectivity may not provide a sufficiently stable representation for MNISTSuperpixels classification. In contrast, incorporating intensity information significantly improves both accuracy and stability.

After setting $c = 1$, the intensity value is used as the third coordinate in the geometric embedding. The accuracy increases substantially to $85.50\% \pm 1.25\%$, which suggests that incorporating intensity into the height function provides useful information for classification. Furthermore, when WECT mode is used with $c = 1$, the accuracy further improves to $94.71\% \pm 0.22\%$.

The result suggests that the two uses of intensity are complementary rather than conflicting. Using intensity as an additional geometric coordinate affects the height function and therefore affects the order in which vertices and edges enter the filtration. Using intensity as a weight affects the weighted Euler characteristic computation by changing the contribution of each simplex after it enters the filtration. These two mechanisms act at different stages of the transform, which may explain why their combination gives the best performance in this experiment.

Overall, the results suggest that the hybrid use of intensity, both as an additional geometric coordinate and as a weight, gives the best performance on MNISTSuperpixels classification tasks.

5. Injectivity and Stability

In this section, we briefly review the injectivity of the Euler Characteristic Transform. We then derive a natural stability extension for the Euler Characteristic Transform under perturbed embeddings.

5.1 Injectivity

The experiments above show that ECT-based representations and their variants are effective in classification tasks. A natural theoretical question is whether the ECT can, in principle, distinguish different shapes.

The injectivity of the ECT provides a theoretical explanation for why the transform is meaningful as a shape descriptor.

We briefly recall the main idea behind the injectivity result of Ghrist et al.[4] Their proof is based on Euler calculus and works in the setting of tame, or definable, sets. This restriction excludes pathological spaces and ensures that Euler characteristic behaves well under the operations used in the proof.

In this framework, a shape is represented by a constructible function. The ECT can then be interpreted as a topological Radon transform: for each direction and threshold, it records the Euler integral of the shape over a corresponding half-space. Since topological Radon transforms admit inversion results under suitable conditions, the ECT is injective on the class of tame compact sets. Equivalently, two such shapes with the same ECT must be identical.

Curry et al.[5] further studied the question of how many directions are needed to determine a shape. Their work shows that, under suitable geometric assumptions on a class of PL embedded shapes, finitely many directions are sufficient to determine the shape.

This result is important for applications, because practical implementations can only sample finitely many directions. However, it should not be interpreted as saying that an arbitrary small number of sampled directions is always sufficient. Rather, the required number of directions depends on the geometric complexity and regularity assumptions of the under-

lying shape.

5.2 Stability

Motivated by the stability result for the Euler Characteristic Transform under perturbed embeddings studied by George et al.[6], we use the admissibility of the weight function to obtain a natural extension to the weighted setting. Since the complex K is finite, the weight functions considered below take only finitely many values. Without loss of generality, we assume that these values are contained in $\{1, \ldots, m\}$.

Recall that an admissible function $g : K \to \mathbb{R}$ satisfies the condition that, for every simplex $\sigma \in K$ and every face $\tau < \sigma$,

$$g(\tau) \geq g(\sigma).$$

Therefore, for each $z \in \mathbb{R}$, the superlevel set

$$K_z := g^{-1}([z, \infty))$$

is a subcomplex of K. This is analogous to the fact that the sublevel sets of a simplex-wise monotone filtration function form subcomplexes.

Then, we have

$$\begin{aligned}
\chi(K, g) &= \sum_{d=0}^{\dim(K)} (-1)^d \sum_{\sigma \in K^d} g(\sigma) && \text{(3a)} \\
&= \sum_{d=0}^{\dim(K)} (-1)^d \sum_{\sigma \in K^d} \sum_{z \in \mathbb{N}} \mathbf{1}_{g(\sigma) \geq z} && \text{(3b)} \\
&= \sum_{d=0}^{\dim(K)} (-1)^d \sum_{z \in \mathbb{N}} \#\{\sigma \in K^d \mid g(\sigma) \geq z\} && \text{(3c)} \\
&= \sum_{z \in \mathbb{N}} \chi\big(g^{-1}([z, \infty))\big). && \text{(3d)}
\end{aligned}$$

Given a weighted simplicial complex (K, g), let

$$f, \phi : K \to \mathbb{R}^d$$

be two different geometric embeddings of the same abstract complex K. They induce two embedded weighted complexes, denoted by (K_a, g) and (K_b, g), where

$$K_a = f(K), \qquad K_b = \phi(K).$$

The weight function g is assumed to be the same in both cases, so the difference between (K_a, g) and (K_b, g) only comes from the embedding.

Then we consider the

$$\text{WECC}_v^{K_a,g}(t) = \chi\big(v_a^{-1}((-\infty, t]), g\big) = \sum_{z=1}^{m} \chi\big(g^{-1}([z, \infty)) \cap v_a^{-1}((-\infty, t])\big),$$

for any $t \in \mathbb{R}$.

By the paper[6], we have

$$\|\text{ECC}_\nu(f(K)) - \text{ECC}_\nu(\phi(K))\|_1 \leq 2W_{1,\infty}\left(\text{Dgm}(h_\nu^f(K)), \text{Dgm}(h_\nu^\phi(K))\right).$$

From the decomposition above, we have

$$\text{WECC}_\nu^{K_a,g}(t) = \sum_{z=1}^{m} \text{ECC}_\nu^{f(K_z)}(t),$$

and similarly,

$$\text{WECC}_\nu^{K_b,g}(t) = \sum_{z=1}^{m} \text{ECC}_\nu^{\phi(K_z)}(t).$$

Therefore,

$$\begin{aligned}\left\|\mathrm{WECC}_{\nu}^{K_a,g}-\mathrm{WECC}_{\nu}^{K_b,g}\right\|_1 &= \left\|\sum_{z=1}^{m}\left(\mathrm{ECC}_{\nu}^{f(K_z)}-\mathrm{ECC}_{\nu}^{\phi(K_z)}\right)\right\|_1 \\ &\leq \sum_{z=1}^{m}\left\|\mathrm{ECC}_{\nu}^{f(K_z)}-\mathrm{ECC}_{\nu}^{\phi(K_z)}\right\|_1 \\ &\leq 2\sum_{z=1}^{m} W_{1,\infty}\left(\mathrm{Dgm}(h_{\nu}^{f}(K_z)),\mathrm{Dgm}(h_{\nu}^{\phi}(K_z))\right).\end{aligned}$$

The first inequality follows from the triangle inequality, and the second inequality follows by applying the ECC stability bound to each superlevel subcomplex K_z.

Thus, the stability bound for ECC under perturbed embeddings naturally extends to the weighted case:

$$\left\|\mathrm{WECC}_{\nu}^{K_a,g}-\mathrm{WECC}_{\nu}^{K_b,g}\right\|_1 \leq 2\sum_{z=1}^{m} W_{1,\infty}\left(\mathrm{Dgm}(h_{\nu}^{f}(K_z)),\mathrm{Dgm}(h_{\nu}^{\phi}(K_z))\right).$$

Since the WECT is obtained by collecting the WECCs over all directions, the above pointwise bound for each fixed direction ν can be integrated over the sphere of directions. Therefore, the stability extension for WECC naturally induces the corresponding stability statement for WECT, and the argument follows directly from the treatment in the paper[6].

6. Summary

6.1 Conclusion

This thesis studied the Euler Characteristic Transform and its applications to deep learning. After reviewing the basic definitions of ECC, ECT, DECT, and WECT, we reproduced several DECT-based classification experiments on MNIST and graph-based MNIST data. The reproduced results show that ECT-based representations can be effectively used as differentiable feature layers in neural-network models.

The main experimental contribution of this thesis is the construction of differentiable WECT-style representations for image data. By incorporating pixel intensity as simplex weights, the WECT representation is made compatible with neural-network training.

The results show that the DWECT-based models outperform the corresponding DECT-based model in our experiments, suggesting that intensity information can improve the expressive power of ECT-based representations.

We also investigated two different ways of preserving intensity information: using intensity as an additional geometric coordinate and using intensity as a weight in the weighted Euler characteristic computation. The MNISTSuperpixels experiment suggests that these two mechanisms are complementary. Using intensity in the geometric embedding affects the height filtration, while using intensity as a weight changes the contribution of simplices to the Euler characteristic.

Finally, motivated by the stability result for ECT under perturbed embeddings, we gave a simple extension to the weighted invariant. Using the admissibility of the weight function, the weighted Euler characteristic can be decomposed into a sum of ordinary Euler characteristics over the subcomplexes induced by superlevel sets. This allows the stability argument for ECC to be naturally extended to WECC, and hence to WECT by integrating over directions.

Overall, this thesis shows that WECT can be combined with differentiable neural-network pipelines, and that intensity information is important for improving ECT-based representations in image classification tasks.

6.2 Future Work

The differentiable WECT construction in this thesis is still preliminary. A more systematic study of weight construction, weight normalization, and the geometric intensity scale would help clarify how intensity information should be used in ECT-based neural networks.

More generally, future work may consider combining ECT-based features with other topological descriptors, or selecting task-specific features that carry both geometric structure and topological meaning.

However, the experiments are mainly based on MNIST-type datasets. Future work could test DECT and DWECT on more complex datasets such as medical imaging datasets,

where topological and geometric information are important.

Finally, practical implementations only use finitely many directions and thresholds. It would be useful to further study how these sampling choices affect classification performance, computational cost, and stability.

References

[1] J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4741–4748, 2015. DOI: 10.1109/CVPR.2015.7299106.

[2] E. Röell and B. Rieck. Differentiable euler characteristic transforms for shape classification. In *International Conference on Learning Representations*, 2024.

[3] K. Turner, S. Mukherjee, and D. M. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, Dec. 2014. DOI: 10.1093/imaiai/iau011.

[4] R. Ghrist, R. Levanger, and H. Mai. Persistent homology and euler integral transforms. *Journal of Applied and Computational Topology*, 2(1):55–60, 2018. DOI: 10.1007/s41468-018-0017-1.

[5] J. Curry, S. Mukherjee, and K. Turner. How many directions determine a shape and other sufficiency results for two topological transforms. *Transactions of the American Mathematical Society, Series B*, 9(32):1006–1043, 2022. DOI: 10.1090/btran/122.

[6] J. George, O. Lledo Osborn, E. Munch, M. Ridgley II, and E. X. Wang. On the stability of the euler characteristic transform for a perturbed embedding, 2025. DOI: 10.48550/arXiv.2506.19991. arXiv: 2506.19991 [cs.CG].

[7] Q. Jiang, S. Kurtek, and T. Needham. The weighted euler curve transform for shape and image analysis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3685–3694, June 2020. DOI: 10.1109/CVPRW50498.2020.00430.

[8] B. Rieck. Topology meets machine learning: an introduction using the euler characteristic transform. *Notices of the American Mathematical Society*, 72(7):719–727, 2025. DOI: 10.1090/noti3193.

[9] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023. URL: https://jmlr.org/papers/v24/22-0567.html.

Appendix

Reference Code for Figures 2–4

visualization-code

```python
import math

import torch
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon

from dect.directions import generate_2d_directions
from dect.ect import compute_ect
from dect.ect_fn import scaled_sigmoid


plt.rcParams.update({
    "font.size": 11,
    "axes.labelsize": 11,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10,
    "axes.linewidth": 0.8,
})


num_directions = 64
num_resolutions = 64
RADIUS = 1.0
SCALE = 500

POINTS = torch.tensor([[0.5, 0.0], [-0.5, 0.0], [0.5,
    0.5]], dtype=torch.float32)
EDGE_INDEX = torch.tensor([[0, 1, 2], [1, 2, 0]], dtype
    =torch.long)
FACE_INDEX = torch.tensor([[0], [1], [2]], dtype=torch.
```

```
    long)
DIRECTIONS = generate_2d_directions(num_thetas=
    num_directions)


def compute_dataset_ect(points_coordinates, edge_index=
    None, face_index=None):
    args = [points_coordinates]
    if edge_index is not None:
        args.append(edge_index)
    if face_index is not None:
        args.append(face_index)

    return compute_ect(
        *args,
        v=DIRECTIONS,
        radius=RADIUS,
        resolution=num_resolutions,
        scale=SCALE,
        ect_fn=scaled_sigmoid,
    )


def draw_data(ax, points_coordinates, edge_index=None,
    face_index=None):
    points = points_coordinates.detach().cpu().numpy()

    if face_index is not None:
        for face in face_index.detach().cpu().numpy().T
            :
            ax.add_patch(
                Polygon(
                    points[face],
                    closed=True,
                    facecolor="0.85",
                    edgecolor="none",
```

```
                    zorder=1,
                )
            )

    if edge_index is not None:
        for start_idx, end_idx in edge_index.detach().
            cpu().numpy().T:
            edge_points = points[[start_idx, end_idx]]
            ax.plot(
                edge_points[:, 0],
                edge_points[:, 1],
                color="black",
                linewidth=1.2,
                zorder=2,
            )

    ax.scatter(points[:, 0], points[:, 1], color="black
        ", s=36, zorder=3)

    x_min, y_min = points.min(axis=0)
    x_max, y_max = points.max(axis=0)
    x_margin = max(0.2, 0.15 * max(x_max - x_min, 1.0))
    y_margin = max(0.2, 0.15 * max(y_max - y_min, 1.0))

    ax.set_xlim(x_min - x_margin, x_max + x_margin)
    ax.set_ylim(y_min - y_margin, y_max + y_margin)
    ax.set_aspect("equal", adjustable="box")
    ax.set_xlabel(r"$x$")
    ax.set_ylabel(r"$y$")
    ax.grid(True, linestyle=":", linewidth=0.6, color="
        0.8")


def draw_ect(ax, ect, vmin, vmax):
    ect_image = ect.detach().squeeze().cpu().numpy().T

```

```
95      ax.imshow(
96          ect_image,
97          origin="lower",
98          cmap="viridis",
99          aspect="auto",
100         extent=(0.0, 2.0 * math.pi, -RADIUS, RADIUS),
101         vmin=vmin,
102         vmax=vmax,
103     )
104     ax.set_xlabel(r"Direction␣$\theta$")
105     ax.set_ylabel(r"Threshold␣$t$")
106     ax.set_xticks([0.0, math.pi, 2.0 * math.pi])
107     ax.set_xticklabels(["0", r"$\pi$", r"$2\pi$"])
108     ax.set_yticks([-1.0, 0.0, 1.0])
109
110
111 def save_figure(filename, data_title,
    points_coordinates, ect, vmin, vmax, edge_index=None
    , face_index=None):
112     fig, axes = plt.subplots(1, 2, figsize=(6.2, 3.0),
        gridspec_kw={"width_ratios": [1.0, 1.25]})
113
114     draw_data(axes[0], points_coordinates, edge_index=
        edge_index, face_index=face_index)
115     axes[0].set_title(data_title)
116
117     draw_ect(axes[1], ect, vmin=vmin, vmax=vmax)
118     axes[1].set_title("Visualization_of_the_sampled␣
        DECT")
119
120     fig.tight_layout()
121     fig.savefig(f"{filename}.pdf", bbox_inches="tight")
122     fig.savefig(f"{filename}.png", dpi=300, bbox_inches
        ="tight")
123     plt.close(fig)
124
```

```
125     print(f"Generated: {filename}.pdf")
126     print(f"Generated: {filename}.png")
127
128
129 def main():
130     datasets = [
131         {
132             "filename": "figure2",
133             "title": "Point cloud",
134             "points": POINTS,
135         },
136         {
137             "filename": "figure3",
138             "title": "Graph",
139             "points": POINTS,
140             "edge_index": EDGE_INDEX,
141         },
142         {
143             "filename": "figure4",
144             "title": "2-Simplex",
145             "points": POINTS,
146             "edge_index": EDGE_INDEX,
147             "face_index": FACE_INDEX,
148         },
149     ]
150
151     for dataset in datasets:
152         dataset["ect"] = compute_dataset_ect(
153             dataset["points"],
154             edge_index=dataset.get("edge_index"),
155             face_index=dataset.get("face_index"),
156         )
157
158     vmin = min(float(dataset["ect"].min()) for dataset
        in datasets)
159     vmax = max(float(dataset["ect"].max()) for dataset
```

```
        in datasets)

    for dataset in datasets:
        save_figure(
            dataset["filename"],
            dataset["title"],
            dataset["points"],
            dataset["ect"],
            vmin=vmin,
            vmax=vmax,
            edge_index=dataset.get("edge_index"),
            face_index=dataset.get("face_index"),
        )


if __name__ == "__main__":
    main()
```

Acknowledgments

I would like to express my sincere gratitude to my advisor, Prof. Yifei Zhu, for his guidance and support throughout this thesis. His course on Applied and Computational Topology was the starting point of my interest in this topic. I am also grateful for his Applied and Computational Topology Seminar, which exposed me to recent research topics and helped me develop a broader understanding of the connection between topology, computation, and data analysis. The knowledge and perspective I gained from both the course and the seminar laid an important foundation for this thesis.

I would also like to thank Haiyu Zhang for her guidance in helping me start coding. When my computer was unavailable due to unexpected hardware problems, She also generously provided me with access to a workplace, which allowed me to continue my experiments. Her patience and kindness helped me a lot.

Finally, I would like to thank my family and friends for their solid support. Their encouragement helped me complete this thesis.