Towards General-Purpose Representation Learning of Polygonal Geometries

Gengchen Mai[1,2,3,4*], Chiyu Jiang[5], Weiwei Sun[6], Rui Zhu[7,3,4], Yao Xuan[8], Ling Cai[3,4], Krzysztof Janowicz[3,4,9], Stefano Ermon[2,10] and Ni Lao[11+]

[1*]Spatially Explicit Artificial Intelligence Lab, Department of Geography, University of Georgia, Athens, 30602, Georgia, USA.
[2*]Department of Computer Science, Stanford University, Stanford, 94305, California, USA.
[3*]STKO Lab, University of California Santa Barbara, Santa Barbara, 93106, California, USA.
[4*]Center for Spatial Studies, University of California Santa Barbara, Santa Barbara, 93106, California, USA.
[5]Department of Mechanical Engineering, University of California Berkeley, Berkeley, 94720, California, USA.
[6]Department of Computer Science, University of British Columbia, Vancouver, V6T 1Z4, British Columbia, Canada.
[7]School of Geographical Sciences, University of Bristol, Bristol, BS8 1TH, United Kingdom.
[8]Department of Mathematics, University of California Santa Barbara, Santa Barbara, 93106, California, USA.
[9]Department of Geography and Regional Research, University of Vienna, Vienna, 1040, Austria .
[10]Chan Zuckerberg Biohub, San Francisco, 94158, California, USA .
[11]Google, Mountain View, 94043, California, USA
[+] Work done while working at mosaix.ai.

*Corresponding author(s). E-mail(s): gengchen.mai25@uga.edu; Contributing authors: maxjiang93@gmail.com; weiweis@cs.ubc.ca; rui.zhu@bristol.ac.uk; yxuan@math.ucsb.edu;

ling.cai@geog.ucsb.edu; jano@geog.ucsb.edu;
ermon@cs.stanford.edu; nlao@google.com;

Abstract

Neural network representation learning for spatial data (e.g., points, polylines, polygons, and networks) is a common need for geographic artificial intelligence (GeoAI) problems. In recent years, many advancements have been made in representation learning for points, polylines, and networks, whereas little progress has been made for polygons, especially complex polygonal geometries. In this work, we focus on developing a general-purpose polygon encoding model, which can encode a polygonal geometry (with or without holes, single or multipolygons) into an embedding space. The result embeddings can be leveraged directly (or finetuned) for downstream tasks such as shape classification, spatial relation prediction, building pattern classification, cartographic building generalization, and so on. To achieve model generalizability guarantees, we identify a few desirable properties that the encoder should satisfy: loop origin invariance, trivial vertex invariance, part permutation invariance, and topology awareness. We explore two different designs for the encoder: one derives all representations in the spatial domain and can naturally capture local structures of polygons; the other leverages spectral domain representations and can easily capture global structures of polygons. For the spatial domain approach we propose ResNet1D, a 1D CNN-based polygon encoder, which uses circular padding to achieve loop origin invariance on simple polygons. For the spectral domain approach we develop NUFTspec based on Non-Uniform Fourier Transformation (NUFT), which naturally satisfies all the desired properties. We conduct experiments on two different tasks: 1) polygon shape classification based on the commonly used MNIST dataset; 2) polygon-based spatial relation prediction based on two new datasets (DBSR-46K and DBSR-cplx46K) constructed from OpenStreetMap and DBpedia. Our results show that NUFTspec and ResNet1D outperform multiple existing baselines with significant margins. While ResNet1D suffers from model performance degradation after shape-invariance geometry modifications, NUFTspec is very robust to these modifications due to the nature of the NUFT representation. NUFTspec is able to jointly consider all parts of a multipolygon and their spatial relations during prediction while ResNet1D can recognize the shape details which are sometimes important for classification. This result points to a promising research direction of combining spatial and spectral representations.

Keywords: Polygon Encoding, Non-Uniform Fourier Transformation, Shape Classification, Spatial Relation Prediction, Spatially Explicit Artificial Intelligence

1 Introduction

Deep neural networks have shown great success for numerous tasks from computer vision, natural language processing, to audio analysis, in which the underlining data is usually in a regular structure such as grids (e.g., images) or sequences (e.g., sentences, audios) [1, 2]. These successes can be largely attributed to the fact that such regular data structures are natively supported by the neural networks [1]. For example, convolutional neural networks (CNN) are naturally suitable for image and video analysis. Recurrent neural networks (RNN) are suitable for data with sequence structures such as sentences and time series. However, it is hard to apply similar models on data with more complex structures. Recent years have witnessed increasing interests in geometric deep learning [1, 3], which focuses on developing deep models for non-Euclidean geometric data such as graphs [4–9], points [10–13], and manifolds [3, 14] that have rather irregular structures. In fact, deep learning models on irregularly structured data or non-Euclidean geometric data have various applications in different domains such as computational social science (e.g., social network [15, 16]), chemistry (e.g., organic molecules [17]), bioinformatics (e.g., gene regulatory network [18]), and geoscience (e.g., traffic network [19, 20], air quality sensor network [21], weather sensor networks [22], and species occurrences [12, 13]). This trend indicates that representing various types of spatial data in an embedding space for downstream neural network models is an important task for geographic artificial intelligence (GeoAI) research [2].

Recently, we have seen many research advancements in developing representation learning models for points [2, 12, 13, 23], polylines [24–26], and network [20, 27]. Compared with other non-Euclidean geometric data, few efforts have been taken to develop deep models on polygons, especially complex polygonal geometries (e.g., polygons with holes, multipolygons), despite the fact that they are widely utilized in multiple applications, especially (geo)spatial applications such as shape coding and classification [28, 29], building pattern classification (BPC) [30–32], building grouping [30, 33], cartographic building generalization [34], geographic question answering (GeoQA) [9, 35–39], and so on. Figure 1 demonstrates the importance of polygon data for two geospatial tasks - GeoQA and BPC. Without proper polygon representations of Canada and the US (Figure 1a), Question '*How far it is from Canada to US*' cannot be answered correctly even by state-of-the-art QA system[1]. As for the BPC task (Figure 1b), the shape and arrangement of building polygons in a neighborhood are indicative for its types, e.g., regular or irregular building groups [31].

A representation learning model on polygons is desired. In many previous GeoAI study, due to the lack of ways to directly encode polygons into the embedding space, researchers have to rely on feature engineering to convert polygon shapes into a set of predefined shape descriptors before feeding them into the neural networks. For building pattern classification, given a set of

---

[1]The answer to this brain teaser question should be 0 because Canada and the US are adjacent to each other. However, since Google utilizes geometric central points as the spatial representations for geographic entities, Google QA returns 2260 km as the answer as the distance between them.
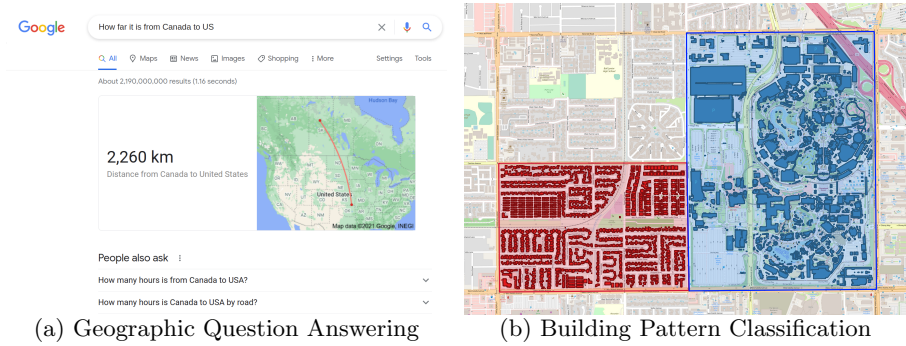
(a) Geographic Question Answering   (b) Building Pattern Classification

Fig. 1: Two geospatial tasks involving polygon data: (a) GeoQA: many geographic questions can only be answered correctly based on polygon representations of geographic entities. Otherwise, it will yield an incorrect answer (2260 km) such as '*How far it is from Canada to US*'. (b) BPC: the shape, scale, and arrangement of building footprints in a neighborhood are indicative for the neighborhood types. The blue neighborhood (Disneyland Park in Los Angles) shows irregular patterns whereas the red neighborhood (residential area) shows regular patterns [31].

building polygons, Yan et al. [31], He et al. [30], and Bei et al. [32] converted the polygon set into a graph in which each node represents a building polygon and edges represent the spatial adjacent relations among buildings. They compute a feature vector for each building polygon/node based on a set of predefined shape descriptors. These vectors are used as initial node embeddings for the following graph neural network for building pattern recognition. These feature engineering approaches have several disadvantages: 1) these shape descriptors can not fully capture the shape information polygons have which yield information loss; 2) lots of domain knowledge is needed to design these descriptors; 3) this practice lacks generalizability – it is hard to used the developed shape descriptors in other polygon tasks. In contrast, developing a general-purpose polygon encoder has several advantages: 1) it allows us to develop end-to-end neural architectures directly taking polygons as inputs which increases the model expressivity; 2) it eliminates the need of domain knowledge when handling polygon data; 3) this model is task agnostic and can benefit a wide range of GeoAI tasks.

For the object instance segmentation task, existing deep models decode a simple polygon[2] based on the object mask image [40–42]. These approaches can be seen as a reverse process of polygon encoding and they cannot decode complex polygonal geometries. In contrast, **we propose to develop general-purpose polygon representation learning models, which directly encode a polygonal geometry (with or without holes, single or multi-polygons) in an embedding space**. Furthermore, we identify a few desirable properties for polygon encoders to guarantee their model generalizability: *loop*

---

[2]A simple polygon is a polygon that does not intersect itself and has no holes.
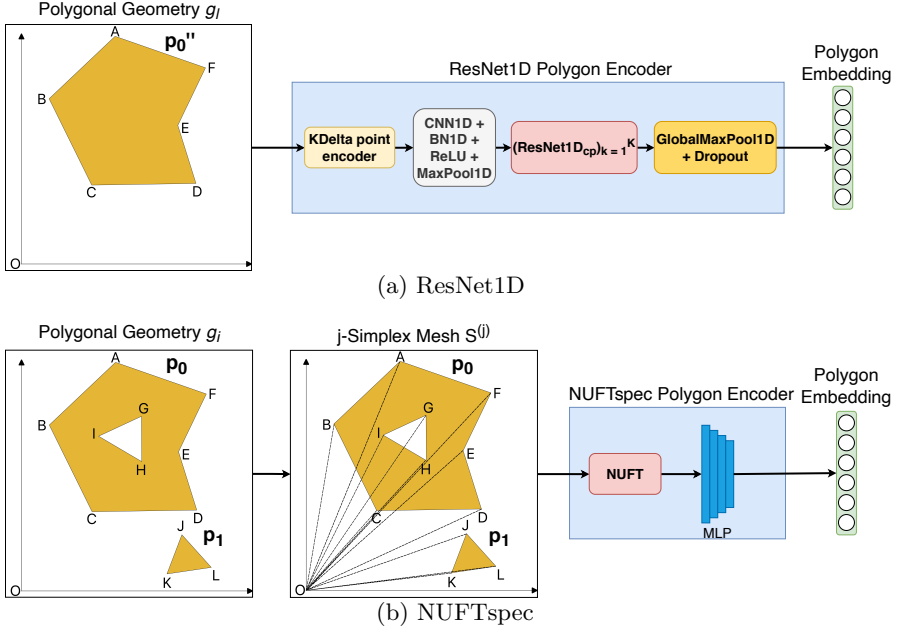
(a) ResNet1D

(b) NUFTspec

Fig. 2: Illustrations of the proposed polygon encoders. (a) ResNet1D: given a simple polygon (i.e., without hole) g_l, its exterior coordinate sequence can be encoded as a 1D point embedding sequence **L** by using KDelta point encoder. **L** is fed into an initial 1D CNN layerand 1D max pooling layer, followed by $\mathcal{K}$ 1D ResNet layers with circular padding. Eventually, a global max pooling layer produces the final polygon embedding. (b) NUFTspec: given a polygonal geometry g_i (i.e., a single polygon with/without holes or multipolygons), first it is converted into a j-simple mesh $\mathcal{S}^{(j)}$ (j = 2) based on *auxiliary node method* (See Section 5.2.1). After NUFT, the Fourier features are fed into a multi-layer perceptron (MLP) to obtain the final polygon embedding.

origin invariance, *trivial vertex invariance*, *part permutation invariance*, and *topology awareness*, which will be discussed in detail in Section 3. The resulting polygon embeddings can be subsequently utilized in multiple downstream tasks such as shape classification [28, 29, 43, 44], spatial relation prediction between geographic entities [45], GeoQA [36], and so on.

Existing polygon encoding approaches can be classified into two groups: spatial domain polygon encoders and spectral domain polygon encoders. Spatial domain polygon encoders such as VeerCNN [28], GCAE [29], directly learn polygon embeddings from polygon features in the spatial domain (e.g., vertex coordinate features). In contrast, spectral domain polygon encoders such as DDSL [46, 47] first convert a polygonal geometry into spectral domain features by using Fourier transformations and learn polygon embeddings based on these spectral features. Both practices have unique advantages and disadvantages. To study the pros and cons of these two approaches, we propose

two polygon encoders: *ResNet1D* and *NUFTspec*. ResNet1D directly takes the polygon features in the spatial domain, i.e., the polygon vertex coordinate sequences, and uses a 1D convolutional neural network (CNN) based architecture to produce polygon embeddings. In contrast, *NUFTspec* first transforms a polygonal geometry into the spectral domain by the Non-Uniform Fourier Transformation (NUFT) and then learns polygon embeddings from these spectral features using feed forward layers. Figure 2a and 2b illustrate the general model architectures of ResNet1D and NUFTspec polygon encoder respectively.

We compare the effectiveness of ResNet1D and NUFTspec with various deterministic or deep learning-based baselines on two types of tasks – 1) polygon shape classification and 2) polygon-based spatial relation prediction. For the first task, we show that both ResNet1D and NUFTspec are able to outperform multiple baselines with statistically significant margins whereas ResNet1D are better at capturing local features of the polygons, and NUFTspec better at capturing global features of the polygons. Our analysis shows that because of NUFT, NUFTspec is robust to multiple shape-invariant geometry modifications such as loop origin randomization, vertex upsampling, and part permutation whereas ResNet1D suffers from significant performance degredations. For the spatial relation prediction task, NUFTspec outperforms ResNet1D, as well as multiple determinstic and deep learning-based baselines on both DBSR-46K and DBSR-cplx46K datasets because it can learn robust polygon embeddings from the spectral domain derived from NUFT. In addition, experiments on both tasks show that compared with other NUFT-based methods such as DDSL[46], NUFTspec is more flexible in the choice of NUFT frequency maps. Designing appropriate NUFT frequency maps for NUFTspec can help learn more robust and effective polygon representations which is the key to its better performance. Our contribution can be summarized as follows:

1. We formally define the problem of representation learning on polygonal geometries (including simple polygons, polygons with holes, and multipolygons), and identify four desirable polygon encoding properties to test their model generalizability.
2. We propose two polygon encoders, ResNet1D and NUFTspec, which learn polygon embeddings from spatial and spectral feature domains respectively.

3. We compare the performance of the proposed polygon encoders as well as multiple baseline models on two representative tasks – shape classification and spatial relation prediction, and introduce three new datasets – MNIST-cplx70k, DBSR-46K, and DBSR-cplx46K.
4. We provide a detailed analysis of the invariance/awareness properties on these two models and discuss the pros and cons of polygon representation learning in the spatial or spectral domain. Our analysis points to interesting future research directions.

This paper is organized as follows: We discuss the motivation of polygon encoding in Section 2. Then, in Section 3, we define the problem of representation learning on polygons and discuss four expected polygon encoding properties. Related work are discussed in Section 4. We present ResNet1D and NUFTspec polygon encoder and compare their properties in Section 5. Experiments on shape classification and spatial relation prediction tasks are presented in Section 6 and 7, respectively. Finally, we conclude this work in Section 8.

2 Motivations

First of all, we discuss the challenges of representing polygons, especial complex polygonal geometries into an embedding space. Given a polygonal geometry, there are two pieces of important information we are especially interested in: its shape and spatial relations with other geometries. These two pieces of information correspond to two polygon-based tasks – shape classification and polygon-based spatial relation prediction. In the following, we motivate polygon encoding from these two aspects by using real-world examples.

2.1 Polygon Encoding for Shape Classification

Shape classification [48] (a.k.a. shape-based object recognition) aims at predicting the category label for a given shape represented by a polygon or multipolygon. Figure 3 shows shape examples from the MNIST dataset [49] illustrating the challenges of polygon encoding for shape classification, especially for complex polygonal geometries, which we summarize as the following:

1. **Automatic representation learning for polygonal geometries**: Traditional shape classification models are based on handcrafted shape descriptors which capture different geometric properties based on vector
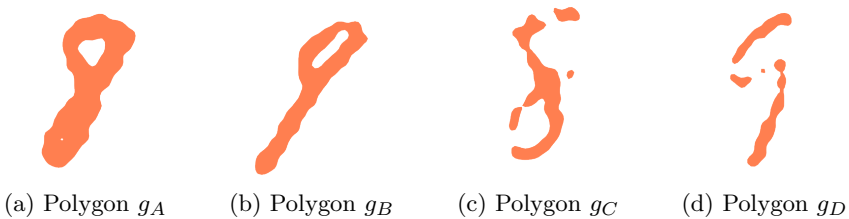

(a) Polygon g_A (b) Polygon g_B (c) Polygon g_C (d) Polygon g_D

Fig. 3: An illustration of the challenges of polygon encoding for shape classification task with examples from the MNIST-cplx70k dataset. Please refer to Section 6.1 for details of the data set construction. (a) and (b) show that it is important to encode polygon holes since only encoding the polygon exteriors are not sufficient for shape classification. (c) and (d) shows that it is also critical to encode all parts of a multipolygon at once in order to capture its global shape, since none of its subpolygons is sufficient for shape classification.

geometries. Examples of shape descriptors are center of gravity, circularity ratio, radial distance, fractality, and so on [31, 48]. For more advanced shape descriptors such as Bag of Contour Fragments (BCF) [44], the first step is also to *vectorize* a given image into vector geometries – polygons, so-called "shape contours". Then a feature extraction pipeline can be applied to produce shape representations. Polygon encoding can be seen as an alternative to these traditional shape encoding models by replacing the feature extraction pipeline with an end-to-end deep learning model.

2. **Encoding the topology (such as holes) in polygons**: Most existing work on polygon encoding [28, 29] focus on encoding simple polygons, i.e., single part polygons without holes. This is insufficient to capture the overall shapes for polygons with holes. For example, Polygon g_A and g_B shown in Figure 3a and 3b indicate two handwritten digits "8" and "9" from our MNIST-cplx70k dataset. If we ignore their holes but only encode the exteriors, Polygon g_A might be misinterpreted as "9" or "7" and Polygon g_B might be recognized as "1" or "7".

3. **Jointly encoding all sub-parts of a multpolygon**: Polygon g_C and g_D shown in Figure 3c and 3d indicate another two handwritten digits "8" and "9" from our MNIST-cplx70k dataset. They are represented as two multipolygons. A model can make the correct shape classification only if it jointly considers all sub-parts of a multipolygon, whereas none of the subpolygons of g_C and g_D is sufficient for shape classification.

2.2 Polygon Encoding for Spatial Relation Prediction

Next, we discuss the challenge of polygon encoding for predicting the correct spatial relations between two polygons, such as topological relations and relative cardinal directions. At the first glance, this task may seem trivial, and a GIScience expert may question the necessity of a polygon encoder for spatial relation prediction given that we have a set of well-defined deterministic spatial operators for spatial relation computation based on region connection calculus (RCC8) [50] or the dimensionally extended 9-intersection model (DE-9IM) [51]. A computer vision researcher might also question its necessity by suggesting an alternative approach that first rasterizes two polygons under consideration into two images that share the same bounding box so that a traditional computer vision model such as CNN can be applied for relation prediction. However, we will demonstrate three different problems related to the spatial relation prediction task - *sliver polygon problem*, *extreme scale problem*, and *semantic vagueness problem*, which are illustrated by Figure 4 with real-world examples from OpenStreetMap. In the following, we discuss how these problems pose challenges to the above mentioned alternative approaches:
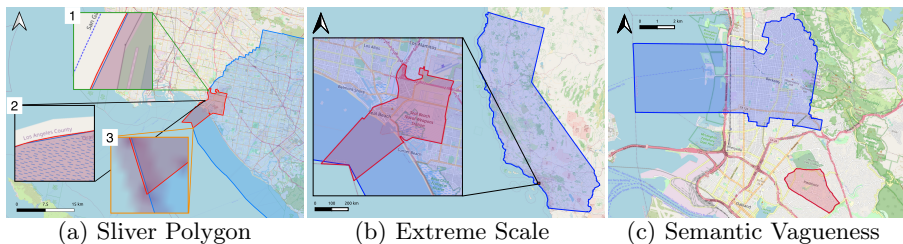
| (a) Sliver Polygon | (b) Extreme Scale | (c) Semantic Vagueness |

Fig. 4: Three real-world examples illustrating different challenges in predicting spatial relations between two polygons: **(a) Sliver Polygon Problem:** *dbr:Seal_Beach,_California* (the red polygon) should be tangential proper part (TPP) of *dbr:Orange_County,_California* (the blue polygon). However, because of those sliver polygons shown in the zoom-in window 1, 2, and 3, a deterministic spatial reasoner such as those used by PostGIS and GeoSPARQL-enabled triple stores will return *partially overlapping* as the result. **(b) Extreme Scale Problem:** *dbr:Seal_Beach,_California* (the red polygon) is extremely small compared with *dbr:California* (the blue polygon). On one hand, a deterministic reasoner will pay too much attention to the geometry detail and predict wrong relations because of the sliver polygons as shown in Figure (a). On the other hand, if we convert these polygons into two images (e.g., two 128×128 images), that share the same bounding box, the red polygon becomes too small and cannot cover even one pixel which will also affect the result. **(c) Semantic Vagueness Problem:** *dbr:Berkeley,_California* (the blue polygon) is in the north of *dbr:Piedmont,_California* (the red polygon) according to DBpedia and Wikipedia. However, based on their polygon representation, their cardinal direction is vague which can be "north" or "northwest".

1. **Sliver polygon problem**: As shown in zoom-in window 1, 2, and 3 in Figure 4a, those three tiny polygons yielded from the geometry difference between the red and blue polygon are called sliver polygons[3]. Here, *dbr:Seal_Beach,_California* (the red polygon) should be tangential proper part (TPP) of *dbr:Orange_County,_California* (the blue polygon). However, because of map digitization error, the boundary of the red polygon slightly stretches out of the boundary of the blue polygon. A deterministic spatial operator will return "intersect" instead of "part of" as their relation. Sliver polygons are very common in map data, hard to prevent, and require a lot of efforts to correct, while deterministic spatial operators are very sensitive to them. Please refer to a detailed analysis on the sliver polygon problem on our DBSR-46K and DBSR-cplx46K dataset in Section 7.6.

2. **Scale problem**: Two polygons might have very different sizes and thus need to use different map scales for visualization (Figure 4b). *dbr:Seal_Beach,_California* (the red polygon) is extremely small compared with *dbr:California* (the blue polygon). When using deterministic spatial

---

[3]In GIScience, sliver polygon is a technical term referring to the small unwanted polygons resulting from polygon intersection or difference.

operators, sliver polygons will lead to wrong answers while as for the rasterization method, the red polygon become too small to occupy even one pixel of the image.

3. **Semantic vagueness problem**: Some spatial relations such as cardinal direction relations are conceptually vague. While this vagueness can be well handled by neural networks through end-to-end learning from the labels, it is hard to design a deterministic method to predict them. As shown in Figure 4c, *dbr:Berkeley,_California* (the blue polygon) sits in the north of *dbr:Piedmont,_California* instead of northwest according to DBpedia. However, based on their polygon representations, both "north" and "northwest" seems to be true. In other words, their cardinal direction is vague.

Because of those challenges, designing a general-purpose neural network-based representation learning model for polygonal geometries is necessary and can benefit multiple downstream applications.

3 Problem Statement

Based on the Open Geospatial Consortium (OGC) standard, we first give the definition of polygons and multipolygons.

Let $\mathcal{G} = \{g_i\}$ be a set of polygonal geometries in a 2D Euclidean space $\mathbb{R}^2$ where $\mathcal{G}$ is a union of a polygon set $\mathcal{P} = \{p_i\}$ and a multipolygon set $\mathcal{Q} = \{q_i\}$, a.k.a $\mathcal{G} = \mathcal{P} \cup \mathcal{Q}$ and $\mathcal{P} \cap \mathcal{Q} = \emptyset$. We have $g_i \in \mathcal{P} \vee g_i \in \mathcal{Q}$.

Definition 1 (Polygon) Each *polygon* p_i can be represented as a tuple $(\mathbf{B}_i, h_i = \{\mathbf{H}_{ij}\})$ where $\mathbf{B}_i \in \mathbb{R}^{N_{b_i} \times 2}$ indicates a point coordinate matrix for the exterior of p_i defined in a *counterclockwise* direction. $h_i = \{\mathbf{H}_{ij}\}$ is a set of holes for p_i where each hole $\mathbf{H}_{ij} \in \mathbb{R}^{N_{h_{ij}} \times 2}$ is a point coordinate matrix for one interior linear ring of p_i defined in a *clockwise* direction. N_{b_i} indicates the number of unique points in p_i's exterior. The first and last point of $\mathbf{B}_i$ are not the same and $\mathbf{B}_i$ does not intersect with itself. Similar logic applies to each hole $\mathbf{H}_{ij}$ and $N_{h_{ij}}$ is the number of unique points in the jth hole of p_i.

Definition 2 (Multipolygon) A *multipolygon* $q_k \in \mathcal{Q}$ is a set of polygons $q_k = \{p_{ki}\}$ which represents one entity (e.g., Japan, United States, and Santa Barbara County).

Definition 3 (Polygonal Geometry) A *polygonal geometry* $g_i \in \mathcal{P} \cup \mathcal{Q}$ can be either a polygon or a multipolygon. $\{\varepsilon_{in}\}_{n=1}^{N_{g_i}}$ is defined as the set of all boundary segments/edges of the exterior(s) and interiors/holes of g_i or all its sub-polygons. N_{g_i} is the total number of edges in g_i which is equal to the total number of vertices of g_i.

Definition 4 (Simple Polygon and Complex Polygonal Geometry) If a polygonal geometry g_i is a single polygon without any holes, i.e., $g_i = (\mathbf{B}_i, h_i = \emptyset)$, we call it a *simple polygon*. Otherwise, we call it a complex polygonal geometry which might be a multipolygon or a polygon with holes.

Definition 5 (Distributed representation of polygonal geometries) *Distributed representation of polygonal geometries in the 2D Euclidean space $\mathbb{R}^2$ can be defined as*

a function $Enc_{\mathcal{G},\theta}(g_i) : \mathcal{G}^* \to \mathbb{R}^d$ which is parameterized by θ and maps any polygonal geometry $g_i \in \mathcal{G}^*$ in $\mathbb{R}^2$ to a vector representation of d dimension[4]. Here $\mathcal{G}^*$ indicates the set of all possible polygonal geometries in $\mathbb{R}^2$ and $\mathcal{G} \subseteq \mathcal{G}^*$.
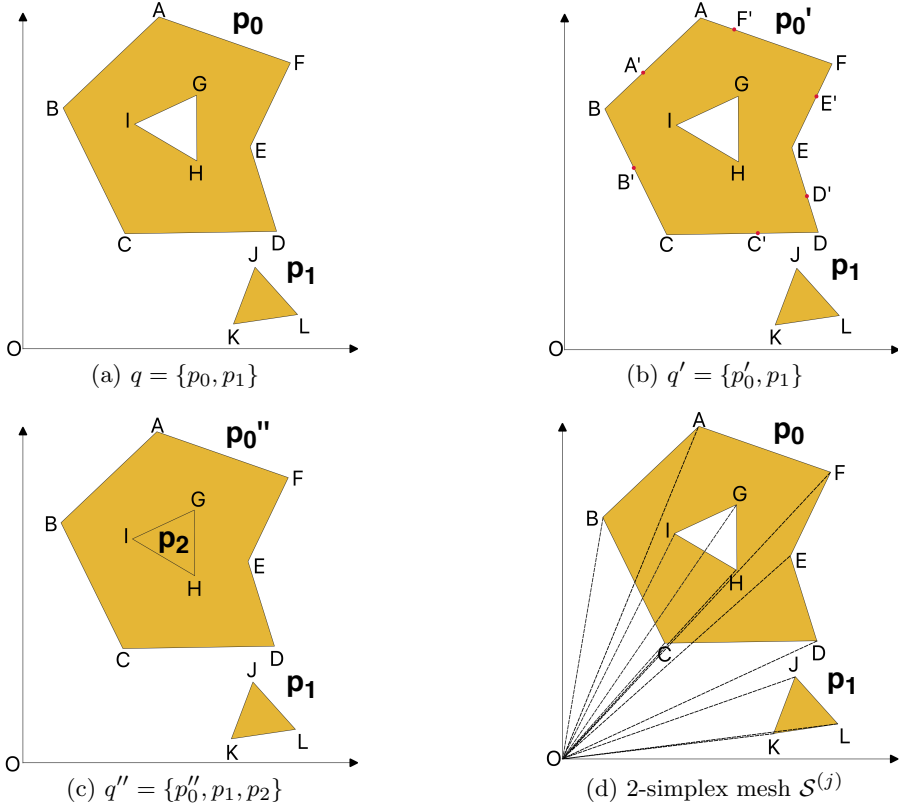

Fig. 5: Illustrations of the four polygon encoding properties and the auxiliary node method with a multipolygon. **(a)** A multipolygon $q = \{p_0, p_1\}$ with two parts. $p_0 = (\mathbf{B}_0, h_0 = \{\mathbf{H}_{00}\})$ has one hole and $p_1 = (\mathbf{B}_1, h_1 = \emptyset)$ has no hole. **(b)** A multipolygon $q' = \{p'_0, p_1\}$ where p'_0 has the same shape as p_0 but adding additional 6 trivial vertices (red dots) - A', B', C', D', E', F' - to its exterior. **(c)** A multipolygon $q'' = \{p''_0, p_1, p_2\}$ with three parts. $p''_0 = (\mathbf{B}_0, \emptyset)$ is a simple polygon made up from the exterior of p_0. p_2 is a simple polygon made up from the boundary of p_0's interior $\mathbf{H}_{00}$. **(d)** The auxiliary node method converts multipolygon q into a 2-simplex mesh $\mathcal{S}^{(j)}$ by adding the origin point $\mathbf{x}_O$ as another vertex (See Section 5.2.1).

Figure 5a illustrates a multipolygon $q = \{p_0, p_1\}$ where $p_0 = (\mathbf{B}_0, h_0 = \{\mathbf{H}_{00}\})$ has one hole and $p_1 = (\mathbf{B}_1, h_1 = \emptyset)$ has no hole. Our objective is to

---

[4]We use $Enc(g_i)$ to represent $Enc_{\mathcal{G},\theta}(g_i)$ in the following

develop general-purpose polygon encoders, so **model generalizability** is the key consideration. To test the generalizibility of a polygon encoder $Enc(g_i)$, we propose four properties:

1. **Loop origin invariance (Loop)**: The encoding result of a polygon p_i should be invariant when starting with different vertices to loop around its exterior/interior. Let consider $p_0'' = (\mathbf{B}_0, \emptyset)$ as a simple polygon made up from the exterior of p_0. $\mathbf{B}_0$ can be written as $\mathbf{B}_0 = [\mathbf{x}_A^T; \mathbf{x}_B^T; \mathbf{x}_C^T; \mathbf{x}_D^T; \mathbf{x}_E^T; \mathbf{x}_F^T] \in \mathbb{R}^{6\times2}$. Let $\mathbf{B}_0^{(s)} = \mathbf{L}_s \mathbf{B}_0$ as another representation of p_0'''s exterior where $\mathbf{L}_s \in \mathbb{R}^{6\times6}$ is a loop matrix that shifts the order of $\mathbf{B}_0$ by s. For example, $\mathbf{B}_0^{(3)} = \mathbf{L}_3 \mathbf{B}_0 = [\mathbf{x}_D^T; \mathbf{x}_E^T; \mathbf{x}_F^T; \mathbf{x}_A^T; \mathbf{x}_B^T; \mathbf{x}_C^T]$. Conceptually, we have $p_0'' = (\mathbf{B}_0, \emptyset) \doteq p_0^{(s)} = (\mathbf{L}_s \mathbf{B}_0, \emptyset) \; \forall s \in \{1, 2, ..., 6\}$ where $\doteq$ indicates two geometries represent equivalent shape information. Loop origin invariance expects $Enc(p_0'') = Enc(p_0^{(s)})$.

2. **Trivial vertex invariance (TriV)**: The encoding result of a polygonal geometry should be invariant when we add/delete trivial vertices to/from its exterior or interiors. Trivial vertices are unimportant vertices such that adding or deleting them from polygons' exteriors or interiors does not change their overall shape and topology. For example, 6 red vertices - A', B', C', D', E', F' - (Figure 5b) are trivial vertices of Polygon p_0' since deleting them yield Polygon p_0 which has the same shape as p_0'. We expect $Enc(p_0') = Enc(p_0)$. It is particularly difficult for 1D CNN- or RNN-based polygon encoders to achieve this since trivial vertices will significantly change the input polygon boundary coordinate sequences.

3. **Part permutation invariance (ParP)**: The encoding result of a multipolygon q_i should be invariant when permuting the feed-in order of its parts. For instance, the encoding result of $Enc(q)$ (Figure 5a) should not change when changing the feed-in order of p_0, p_1.

4. **Topology awareness (Topo)**: The polygon encoder $Enc(g_i)$ should be aware of the topology of the polygonal geometry g_i. $Enc(g_i)$ should not only encode the boundary information of g_i but also be aware of the exterior and interior relationships. For example, as shown in Figure 5a and 5c, $q = \{p_0, p_1\}$ and $q'' = \{p_0'', p_1, p_2\}$ are two multipolygons. p_0'' and p_2 are two simple polygons and p_2 is inside of p_0''. Although q and q'' have the same boundary information, the encoding results of them should be different given their different topological information.

As can be seen, these properties are unique requirements for encoding polygonal geometries. In certain scenarios, **translation invariance**, **scale invariance**, and **rotation invariance**, which require the encoding results of a polygon encoder unchanged when polygons are gone through translation/scale/rotation translations, are also expected in many shape related tasks such as shape classification, shape matching, shape retrieval [46]. However, in other tasks such as spatial relation prediction (including topological relations and cardinal direction relations) and GeoQA, translation/scale/rotation invariance are unwanted. For example, after a translation transformation on p_0 (Figure

5a), the cardinal direction between p_0 and p_1 changes. So in this work, we primarily focus on the above mentioned four properties.

4 Related Work

4.1 Object Instance Segmentation and Polygon Decoding

Most existing machine learning research involving polygons mainly focus on object instance segmentation and localization tasks. They aim at constructing a simple polygon as a localized object mask on an image. Most existing works adopt a polygon refinement approach. For example, both Zhang et al. [52] and Sun et al. [40] proposed to first detect the boundary fragments of polygons from images and then extract a simple polygon by finding the optimal circle linking these fragments into the object contours.

A recent deep learning approach, Polygon-RNN [41], first encodes a given image with a deep CNN structure (similar to VGG [53]) and then decodes the polygon mask of an object with a two-layer convolutional LSTM with skip connections. The RNN polygon decoder decodes one polygon vertex at each time step until the end-of-sequence is decoded which indicates that the polygon is closing. The first vertex is predicted with another CNN using a multi-task loss. Polygon-RNN++ [42] improves Polygon-RNN by adding a Gated Graph Neural Network (GGNN) [54] after the RNN polygon decoder to increase the spatial resolution of the output polygon. Since Polygon-RNN uses cross-entropy loss which over-penalizes the model and is different from the evaluation metric, Polygon-RNN++ changes the learning objective to reinforcement learning to directly optimize on the evaluation metric.

Similar to the polygon refinement idea, PolyTransform [55] first uses a instance initialization module to provide a good polygon initialization for each individual object. And then a feature extraction network is used to extract embeddings for each polygon vertex. Next, PolyTransform uses a self-attention Transformer network to encode the exterior of each polygon and deform the initial polygon. This deforming network predicts the offset for each vertex on the polygon exterior such that the resulting polygon snaps better to the ground truth polygon (the object mask).

Compared with our polygon encoding model, these polygon decoding models treat polygons as output instead of input to the model. In addition, although Polygon-RNN++ and PolyTransform have sub-network modules that take a polygon as the input and refine/deform it into a more fine-grained polygon shape, both their modules – GGNN for Polygon-RNN++ and Transformer for PolyTransform – can only handle simple polygons but not complex ploygonal geometries. Moreover, the GGNN for Polygon-RNN++ satisfies loop origin invariance but not other properties while the Transformer module of PolyTransform can not satisfy any of those four properties in Section 3.

4.2 2D Shape Classification

Stemming from image analysis, shape representation is a fundamental problem in computer vision [44], since shape and texture are two most important aspects for object analysis. Shape classification aims at classifying an object (e.g., animal, leaf) represented as either a polygon or an image into its corresponding class. For example, given an image of the silhouette of an animal, shape classification aims to decide the type of this animal [43]. Given a building footprint represented as a polygon, cartographers are interested in knowing which type of building it falls into such as E-type, Y-type [29]. Further more, given a neighborhood represented as a collection of building polygons as shown in Figure 1b, we would like to know the type/land use of this neighborhood.

Traditional shape classification models are based on handcrafted or learned shape descriptors. Wang et al. [44] proposed a Bag of Contour Fragment (BCF) method by decomposing one shape polygon into contour fragments each of which is described by a shape descriptor. Then a compact shape representation is max-pooled from them based on a spatial pyramid method. With the development of deep learning technology, many image-based shape classification models skip the image vectorization step and directly apply CNN on those input images for shape classification [56, 57]. Later on, instead of directly applying CNN to images, Hofer et al. [58] first converted 2D object shapes (images) into topological signatures and inputted them into a CNN-based model. However, several recent work showed that when using deep convolutional networks for object classification, surface texture plays a larger role than shape information. Although CNN models can access some local shape features such as local orientations, they have no sensitivity to the overall shape of objects [59]. This indicates that it is meaningful to develop a polygon encoding model that is sensitive to shape information for shape classification.

Table 1 provides statistic on multiple existing shape classification datasets. We can see that except for Yan et al's building dataset [29], most shape classification datasets provide images as shape samples and are open access. Yan et al [29] created a building shape classification dataset where each building is represented as one simple polygon, not image. But this dataset is not open sourced. In addition, most datasets are rather small (less than 5K training samples), which is very challenging for deep learning models. For example, according to Kurnianggoro et al. [48], BCF [44], a feature engineering model, is still the state-of-the-art model on MPEG-7 and outperforms all deep learning models. Our MNIST-cplx70k dataset is based on MNIST. See Section 6.1 for a detailed description, and Figure 3 for shape examples.

4.3 Polygon Encoding

Following Section 1, here, we review several existing work about spatial domain and spectral domain polygon encoders.

Spatial domain polygon encoders [28, 29] directly consume polygon vertex coordinates in the spatial domain for polygon encoding. Most of them only

Table 1: Statistics of shape classification datasets. "#C" and "#S/C" indicate the number of categories and the number of samples per category in each dataset. "∼" indicates an estimation of "#S/C" for datasets which are not balanced. "#Train" and "#Test" indicate the number of training and testing samples in each dataset. "-" indicates that there is no common agreement on train/test split on this dataset. "Topic" indicates the type of object each shape sample stands for. "OA" indicates whether this dataset is open access.

Dataset	#C	#S/C	#Train	#Test	Data Format	Topic	OA
MPEG-7 [60]	70	20	-	-	Silhouette images	Various objects	Yes
Animal [43]	20	100	-	-	Silhouette images	Animals	Yes
Swedish leaf [61]	15	75	-	-	Colorful images	Leaves	Yes
ETH-80 [62]	8	410	3,239	41	Colorful images	Various objects	Yes
100 leaves [63]	100	16	-	-	Colorful images	Leaves	Yes
Kimia-216 [64]	18	12	-	-	Silhouette images	Objects/Animals	Yes
MNIST [49]	10	∼7000	60,000	10,000	Silhouette images	Handwritten digits	Yes
Yan et al [29]	10	∼775	5,000	2,751	Simple polygons	Building footprints	No

consider simple polygons . Veer et al. [28] proposed two spatial domain polygon encoders: a recurrent neural network (RNN) based model and a 1D CNN based model. The RNN model directly feeds the polygon exterior coordinate sequence into a bi-directional LSTM and takes the last state as the polygon embedding. The CNN model feeds the polygon exterior sequence into a series of 1D convolutional layers with zero padding followed by a global average pooling. Veer et al. [28] applied both models to three polygon-shape-based tasks: neighbourhood population prediction, building footprint classification, and archaeological ground feature classification. Results show that the CNN model is better than the RNN model on all three tasks. In this work, the CNN model denoted as VeerCNN is used as one of our baselines.

Another example of spatial domain polygon encoders is the Graph Convolutional AutoEncoder model (GCAE) [29]. GCAE learns a polygon embedding for each building footprint represented as a simple polygon in an unsupervised learning manner. The exterior of each building (a simple polygon) is converted to an undirected weighted graph in which exterior vertices are the graph nodes which are connected by exterior segments (graph edges). Each edge is weighted by its length. Each vertex (node) is associated with a node embedding initialized by some predefined local or regional shape descriptors. The GCAE follows a U-Net [65] like architecture which uses graph convolution layers and graph pooling [4] in the graph encoder and upscaling layers in the graph decoder. The intermediate representation between the encoder and decoder is the learned polygon embedding. The effectiveness of GCAE is demonstrated qualitatively and quantitatively on shape similarity and shape retrieval task.

Instead of encoding a polygonal geometry directly in the spatial domain, spectral domain polygon encoders first transform it into the spectral space by

using Fourier transformation and then design a model to consume these spectral features. One example is Jiang et al [47] which first perform Non-Uniform Fourier Transforms (NUFT) to transform a given polygon geometry (or more generally, 2-/3-simplex meshes) into the spectral domain. And then Jiang et al [47] perform a inverse Fast Fourier transformation (IFFT) to convert these spectral features into 2D images or 3D voxels. The result is an image of the polygonal geometry (or a 3D voxel for a 3D shape) which can be easily consumed by different CNN models such as LeNet5 [49], ResNet [66], and Deep Layer Aggregation (DLA) [67]. DDSL [46] further extends this NUFT-IFFT operation into a differentiable layer which is more flexible for shape optimization (through back propagation). The effectiveness of DDSL has been shown in shape classification task (MNIST), 3D shape retrieval task, and 3D surface reconstruction task. However, the NUFT-IFFT operation is essentially a polygon rasterization approach and sacrifice an information loss which depends on the pixel size. As shown in Figure 4b, when the pixel size is too large, the red polygon can not cover even one pixel which might lead to wrong prediction. On the other hand, when the pixel size is too small, the image become unnecessary large and lead to huge computation cost. Inspired by DDSL, our NUFTspec model adopts the NUFT idea. Instead of performing an IFFT, we directly learn the polygon embeddings in the spectral domain. Without the restriction of IFFT, we have more flexibility in terms of the choice of Fourier frequency map (See Section 5.2.2). So NUFTspec is expected to have lower information loss and a better performance. We will discuss the NUFT method in detail in Section 5.2.

5 Method

In this section, we first present two polygon encoders: ResNet1D and NUFT-spec. Figure 2a and 2b show the general model architectures of ResNet1D and NUFTspec respectively. Then we discuss how these encoders can be used to form shape classification models and spatial relation prediction models. We will compare the encoders with other baselines and discuss their properties in Section 5.5. Finally, we provide proofs for their key properties in Section 5.6.

5.1 ResNet1D Encoder

We first propose a spatial domain polygon encoder called ResNet1D which uses a modified 1D ResNet model with circular padding to encode the polygon exterior vertices.

Given a simple polygon $g = (\mathbf{B}, \emptyset)$ where $\mathbf{B} = [\mathbf{x}_0^T; \mathbf{x}_1^T; ...; \mathbf{x}_m^T; ...; \mathbf{x}_{N_g-1}^T] \in \mathbb{R}^{N_g \times 2}$, ResNet1D treats the exterior $\mathbf{B}$ of g as a 1D coordinate sequence. Before feeding $\mathbf{B}$ into the 1D ResNet layer, we first compute a point embedding $\mathbf{l}_m \in \mathbb{R}^{4t+2}$ for the mth point $\mathbf{x}_m$ by concatenating $\mathbf{x}_m$ with its spatial affinity with its neighboring $2t$ points:

$$\mathbf{l}_m = [\mathbf{x}_m; \mathbf{x}_{m-t} - \mathbf{x}_m; ...; \mathbf{x}_{m-1} - \mathbf{x}_m; \mathbf{x}_{m+1} - \mathbf{x}_m; ...; \mathbf{x}_{m+t} - \mathbf{x}_m] \quad (1)$$

We call Equation 1 **KDelta point encoder**, which adds neighborhood structure information into each point embedding and helps to reduce the need to train very deep encoders. Here, if $m-t < 0$ or $m+t \geq N_g$, we get its coordinates by *circular padding* given the fact that $\mathbf{B}$ represents a circle. The resulting embedding matrix $\mathbf{L} = [\mathbf{l}_0^T; \ \mathbf{l}_1^T; \ ...; \ \mathbf{l}_m^T; \ ...; \ \mathbf{l}_{N_g-1}^T] \in \mathbb{R}^{N_g \times (4t+2)}$ is the input of a modified 1D ResNet model which uses *circular padding* instead of zero padding in 1D CNN and max pooling layers to ensure loop origin invariance. The whole ResNet1D architecture is illustrated as Equation 2.

$$
\begin{aligned}
Enc_{ResNet1D}(g) = \Big[\mathbf{L} &\to \text{CNN1D}_{3 \times 1}^{d,1,1} \to \text{BN1D} \to \text{ReLU} \to \text{MP1D}_{2 \times 1}^{2,0} \\
&\to \left(\text{ResNet1D}_{cp} \right)_{k=1}^{\mathcal{K}} \to \text{GMP1D} \to \text{DP} \to \mathbf{p} \Big]
\end{aligned}
\tag{2}
$$

$\text{CNN1D}_{3 \times 1}^{d,1,1}$ indicates a 1D CNN layer with 1 stride, 1 padding (circular padding) and d number of 3×1 kernel (1D kernel). BN1D and ReLU indicate 1D batch normalization layer and a ReLU activation layer. $\text{MP1D}_{2 \times 1}^{2,0}$ indicates a 1D Max Pooling layer with 2 stride, 0 padding, and kernel size 2. $\left(\text{ResNet1D}_{cp} \right)_{k=1}^{\mathcal{K}}$ indicates $\mathcal{K}$ standard 1D ResNet layers with circular padding. GMP1D and DP are a global max pooling layer and dropout layer. The final output $Enc_{ResNet1D}(g) = \mathbf{p} \in \mathbb{R}^d$ is the polygon embedding of the simple polygon g.

5.2 NUFTspec Encoder

As shown in Figure 2b, NUFTspec first applies Non-Uniform Fourier Transforms (NUFT) to convert a polygonal geometry g into the spectral domain. Then it directly feeds these spectral features into a multi-layer perceptron to obtain the polygon embedding $\mathbf{p}$ of g. Polygonal geometry g can be either a polygon with/without holes, or a multipolygon.

5.2.1 Converting Polygon Geometries to *j*-Simplex Meshes

Following DDSL's *auxiliary node method* [46], we first convert a given polygonal geometry g into a j-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g} = (\mathbf{V}, \mathbf{E}, \mathbf{D})$ (here $j = 2$) by adding one auxiliary node (the origin point $\mathbf{x}_O$). A 2-simplex is simply a triangle in the 2D space. The polygon-to-j-simplex-mesh operation is illustrated in Figure 6 and summarized as below:

1. As shown in Figure 6a, we first apply a series of affine transformations including scaling and translation to transform g into a unit space. First, a translation operation is applied to move g such that the center of its bounding box is the origin point $\mathbf{x}_O = [0,0]$. Then, a scaling operator is applied to scale g into the unit space $[-1,1] \times [-1,1]$. Finally, another translation operation is used to move g into the space $[0,2] \times [0,2]$, since positive coordinates are required for NUFT. These transformation operations are used to allow each polygon lays in the same relative space which is critical for neural network learning. Polygon encoding mainly aims at encoding the shape

of each polygon. The global position of g (e.g., the real geographic coordinates of each vertex of California's polygon) should be encoded seperately through location encoding method [2, 13]. If we would like to do spatial relation prediction between a polygon pair, they should be transformed into the same unit space $[0, 2] \times [0, 2]$.

2. We add the origin point $\mathbf{x}_O = [0, 0]$ as an *auxiliary node* which helps us to convert Polygonal Geometry g into j-simplex mesh $\mathcal{S}^{(j)}$ (See Figure 6b).

3. We loop through all edges in $\{\varepsilon_n\}_{n=1}^{N_g}$ (See Definition 3) of g. For the nth edge ε_n, we connect its two vertices to the auxiliary node $\mathbf{x}_O$ (origin) to construct a 2-simplex (triangle) $\mathbf{S}_n^{(j)}$. For instance, for Edge ε_{AB}, we construction a Triangle $\mathbf{S}_{ABO}^{(j)} = \triangle_{ABO}$ (See Figure 6c).

4. For a polygonal geometry g with in total N_g vertices, we can get N_g 2-simplexes which form a 2-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g}$ (See Figure 6d).
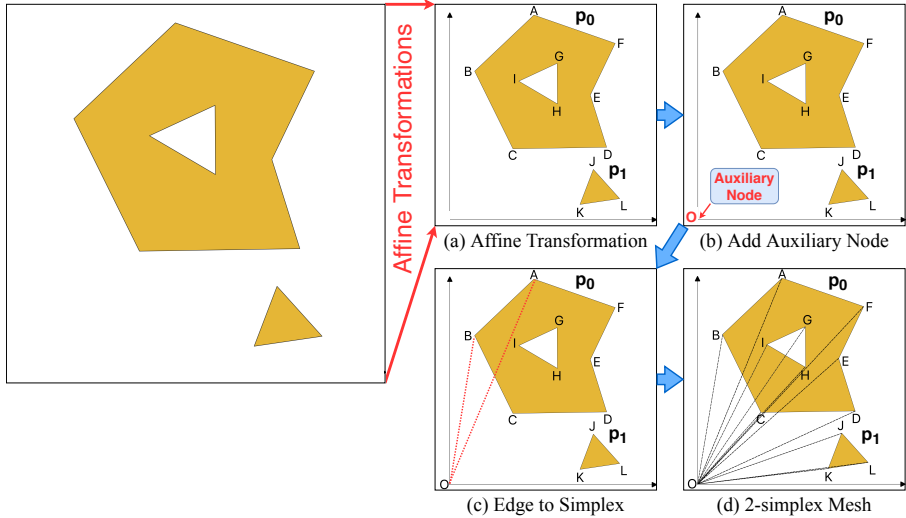

Fig. 6: An illustration of the auxiliary node method. (a) Applying a series of affine transformations to a polygonal geometry g; (b) Adding auxiliary node "O"; (c) Converting each polygon edge to a 2-simplex; (d) Constructing a 2-simplex mesh $\mathcal{S}^{(j)}$.

A 2-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g} = (\mathbf{V}, \mathbf{E}, \mathbf{D})$ is represented as three matrices - vertex matrix $\mathbf{V} \in \mathbb{R}^{(N_g+1) \times 2}$ for polygon/simplex vertex coordinates, edge matrix $\mathbf{E} \in \mathbb{N}_0^{N_g \times 3}$ for 2-simplex connectivity, and density matrix $\mathbf{D} \in \mathbb{R}^{N_g \times d_d}$ for per-simplex density. $\mathbf{V}$ is a float matrix that contains the coordinates of all vertices of g as well as the origin $\mathbf{x}_O = [0, 0]$ (the last row). The edge matrix $\mathbf{E}$ contains non-negative integers. The nth row of $\mathbf{E}$ corresponds to the nth simplex $\mathbf{S}_n^{(j)}$ in $\mathcal{S}^{(j)}$ whose values indicate the indices of vertices of $\mathbf{S}_n^{(j)}$ in $\mathbf{V}$. For the density matrix $\mathbf{D} \in \mathbb{R}^{N_g \times d_d}$, the nth row indicates a d_d-dimensional density features associated with the nth edge/simplex

(e.g., edge color, density, and etc.). In our case, we assume each edge ε_n has a constant 1D density feature – [1], i.e., $\mathbf{D} = \mathbf{1} \in \mathbb{R}^{N_g \times 1}$, a constant one matrix.

We should make sure that the boundary of g is oriented correctly by following Definition 1 - the exteriors of all sub-polygons should be oriented in a counterclockwise fashion while all interiors should be oriented in a clockwise fashion. This makes sure that we can use the *right-hand rule*[5] to infer the correct orientation of each edge ε_n. So the orientation of the boundary of the corresponding simplex $\mathbf{S}_n^{(j)}$ can be determined (e.g., counterclockwise or clockwise). Based on that, we can compute the signed content (area) of each simplex $\mathbf{S}_n^{(j)}$ so that the topology of g is preserved - *topology awareness*.

To concretely show how to convert a polygonal geometry g into a 2-simple mesh $\mathbf{S}^{(2)} = (\mathbf{V}, \mathbf{E}, \mathbf{D})$, we use the example of Multipolygon g shown in Figure 2b. It can be converted into Simplex $\mathbf{S}^{(2)} = (\mathbf{V}, \mathbf{E}, \mathbf{D})$ where $\mathbf{V} = [\mathbf{x}_A^T; \ \mathbf{x}_B^T; \ \mathbf{x}_C^T; \ \mathbf{x}_D^T; \ \mathbf{x}_E^T; \ \mathbf{x}_F^T; \ \mathbf{x}_G^T; \ \mathbf{x}_H^T; \ \mathbf{x}_I^T; \ \mathbf{x}_J^T; \ \mathbf{x}_K^T; \ \mathbf{x}_L^T; \ \mathbf{x}_O^T] \in \mathbb{R}^{13 \times 2}$, $\mathbf{E} = [[0,1,12],[1,2,12],[2,3,12],[3,4,12],[4,5,12],[5,0,12],[6,7,12],[7,8,12], [8,6,12],[9,10,12],[10,11,12],[11,9,12]] \in \mathbb{N}_0^{12 \times 3}$, and $\mathbf{D}$ is a 12×1 constant one matrix.

5.2.2 Non-Uniform Fourier Transforms

Next, we perform NUFT on this j-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g}$ (here $j = 2$). Compared with the conventional Discrete Fourier transform (DFT) whose input signal is sampled at equally spaced points or frequencies (or both), NUFT can deal with input signal sampled at non-equally spaced points or transform the input into non-equally spaced frequencies. This makes NUFT very suitable for irregular structured data such as point cloud, line meshes, polygonal geometries, and so on [46, 47]. In contrast, DFT is more suitable for regular structured data such as images, videos [68].

Definition 6 (Density Function on j-simplex) For the nth j-simplex $\mathbf{S}_n^{(j)}$ in a j-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g} = (\mathbf{V}, \mathbf{E}, \mathbf{D})$, we define a density function $f_n^{(j)}(\mathbf{x})$ on $\mathbf{S}_n^{(j)}$ as Equation 3 in which ρ_n is the signal density defined on $\mathbf{S}_n^{(j)}$.

$$f_n^{(j)}(\mathbf{x}) = \begin{cases} \rho_n, & \mathbf{x} \in \mathbf{S}_n^{(j)} \\ 0, & \mathbf{x} \notin \mathbf{S}_n^{(j)} \end{cases} \tag{3}$$

Definition 7 (Piecewise-Constant Function over a simplex mesh) The Piecewise-Constant Function (PCF) over a simplex mesh $\mathcal{S}^{(j)}$ is the superposition of the density function $f_n^{(j)}(\mathbf{x})$ for each simplex $\mathbf{S}_n^{(j)}$:

$$f_{\mathcal{S}}^{(j)}(\mathbf{x}) = \sum_{n=1}^{N_g} f_n^{(j)}(\mathbf{x}) \tag{4}$$

Definition 8 (NUFT of PCF $f_{\mathcal{S}}^{(j)}(\mathbf{x})$) The NUFT of PCF $f_{\mathcal{S}}^{(j)}(\mathbf{x})$ over a j-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g} = (\mathbf{V}, \mathbf{E}, \mathbf{D})$ on a set of N_w Fourier base frequencies $\mathcal{W} = \{\mathbf{w}_k\}_{k=1}^{N_w}$ is a sequence of N_w complex numbers:

---

[5]https://mapster.me/right-hand-rule-geojson-fixer/

$$F_{\mathcal{S}}^{(j)}(\mathbf{x}) = [F_{\mathcal{S},1}^{(j)}(\mathbf{x}), F_{\mathcal{S},2}^{(j)}(\mathbf{x}), ..., F_{\mathcal{S},k}^{(j)}(\mathbf{x}), ..., F_{\mathcal{S},N_w}^{(j)}(\mathbf{x})] \tag{5}$$

where the NUFT of $f_{\mathcal{S}}^{(j)}(\mathbf{x})$ on each base frequency $\mathbf{w}_k \in \mathbb{R}^2$ can be written as the weighted sum of the Fourier transform on each j-simplex $\mathbf{S}_n^{(j)}$:

$$\begin{aligned}
F_{\mathcal{S},k}^{(j)}(\mathbf{x}) &= \int \cdots \int_{-\infty}^{\infty} f_{\mathcal{S}}^{(j)}(\mathbf{x}) e^{-i\langle \mathbf{w}_k, \mathbf{x} \rangle} d\mathbf{x} \\
&= \sum_{n=1}^{N_g} \rho_n \int \cdots \int_{\mathbf{S}_n^{(j)}} e^{-i\langle \mathbf{w}_k, \mathbf{x} \rangle} d\mathbf{x} \\
&= \sum_{n=1}^{N_g} \rho_n F_{n,k}^{(j)}(\mathbf{x})
\end{aligned} \tag{6}$$

$F_{n,k}^{(j)}(\mathbf{x})$ is the NUFT on the nth simplex $\mathbf{S}_n^{(j)}$ with base frequency $\mathbf{w}_k$ which can be defined as follow:

$$F_{n,k}^{(j)}(\mathbf{x}) = i^j \mu(\mathbf{S}_n^{(j)}) \gamma_n^{(j)} \sum_{t=1}^{j+1} \frac{e^{-i\langle \mathbf{w}_k, \mathbf{x}_t \rangle}}{\prod_{l=1}^{j+1}(e^{-i\langle \mathbf{w}_k, \mathbf{x}_t \rangle} - e^{-i\langle \mathbf{w}_k, \mathbf{x}_l \rangle})}, \tag{7}$$

where $\mu(\cdot) : \mathcal{S}^{(j)} \to \{-1, 1\}$ is a sign function which determines the sign of the content (area) of $\mathbf{S}_n^{(j)}$. $\gamma_n^{(j)}$ is the content distortion factor, which is the ratio of the unsigned content of $\mathbf{S}_n^{(j)}$ - $C_n^{(j)}$ - and the content of the unit orthogonal j-simplex - $C_I^{(j)} = 1/j!$. $\mu(\mathbf{S}_n^{(j)})\gamma_n^{(j)}$ is the signed content distortion factor of $\mathbf{S}_n^{(j)}$ which can be computed based on the determinant of the Jacobian matrix[6] J_n of simplex $\mathbf{S}_n^{(j)}$. Let $\mathbf{x}_{n,1}, \mathbf{x}_{n,2}, \mathbf{x}_O$ the three vertices of $\mathbf{S}_n^{(j)}$, we have:

$$\begin{aligned}
\mu(\mathbf{S}_n^{(j)})\gamma_n^{(j)} &= \frac{\mu(\mathbf{S}_n^{(j)}) C_n^{(j)}}{C_I^{(j)}} = \frac{1/j! \det(J_n)}{1/j!} \\
&= \det([\mathbf{x}_{n,1} - \mathbf{x}_O, \mathbf{x}_{n,2} - \mathbf{x}_O]) = \det([\mathbf{x}_{n,1}, \mathbf{x}_{n,2}])
\end{aligned} \tag{8}$$

The proof of Equation 6, 7, and 8 can be found in [47].

Note that in Definition 8, we have not specified the way in which we select the set of N_w Fourier base frequencies $\mathcal{W} = \{\mathbf{w}_k\}_{k=1}^{N_w}$ where $\mathbf{w}_k \in \mathbb{R}^2$. We can either use a series of equally spaced frequencies along each dimension, denoted as *linear grid frequency map* $\mathcal{W}^{(fft)}$, or a series of non-equally spaced frequencies along each dimension. For the second option, in this work, we choose to use a geometric series as the Fourier frequencies in the X and Y dimension, denoted as *geometric grid frequency map* $\mathcal{W}^{(fft)}$, which has been widely used in many location encoding literature [2, 13, 69, 70] and Transformer architecture [71]. We formally define these two Fourier frequency maps as below.

Definition 9 (Linear Grid Frequency Map) The linear grid frequency map $\mathcal{W}^{(fft)}$ is just simply the normal *integer* Fourier frequency bases used by the Fast Fourier

---

[6]https://en.m.wikipedia.org/wiki/Simplex#Volume

Transform (FFT). It is defined as a Cartesian product between the linear frequency sets along the X and Y axis – $W_x^{(fft)}$ and $W_y^{(fft)}$ – which contain N_{wx} and N_{wy} integer values respectively:

$$\mathcal{W}^{(fft)} = W_x^{(fft)} \times W_y^{(fft)} \in \mathbb{N}^{N_{wx}} \times \mathbb{N}^{N_{wy}} \tag{9}$$

Here, $N_w = N_{wx} N_{wy}$. $W_x^{(fft)}$ and $W_y^{(fft)}$ are defined as Equation 10.

$$W_x^{(fft)} = \begin{cases} \{-U, ..., -1, 0, 1, U\}, & if \ N_{wx} = 2U + 1 \\ \{-U, ..., -1, 0, 1, U - 1\}, & if \ N_{wx} = 2U \end{cases}$$

$$W_y^{(fft)} = \begin{cases} \{0, 1, U\}, & if \ N_{wy} = U + 1 \\ \{0, 1, U - 1\}, & if \ N_{wy} = U \end{cases} \tag{10}$$

Where $U \in \mathbb{N}^+$ is half number of frequencies we use which decides N_{wx} and N_{wy}. Note that a normal practice of FFT is to use half frequency bases in the last dimension. So here $W_y^{(fft)}$ has roughly half of $W_x^{(fft)}$'s frequencies

Definition 10 (Geometric Grid Frequency Map) Since polygonal geometries are non-uniform signals, we do not need to use the normal integer FFT frequency map $\mathcal{W}^{(fft)}$. Instead, we can use real values as Fourier frequency bases to increase the data variance we can capture. By following this idea, the geometric grid frequency map $\mathcal{W}^{(gmf)}$ is defined as a Cartesian product between the selected geometric series based frequency sets along the X and Y axis – $W_x^{(gmf)}$ and $W_y^{(gmf)}$:

$$\mathcal{W}^{(gmf)} = W_x^{(gmf)} \times W_y^{(gmf)} \in \mathbb{R}^{N_{wx}} \times \mathbb{R}^{N_{wy}} \tag{11}$$

Here, $W_x^{(gmf)}$ and $W_y^{(gmf)}$ are defined as Equation 12.

$$W_x^{(gmf)} = \begin{cases} \{-w_u\}_{u=0}^{U-1} \cup \{0\} \cup \{w_u\}_{u=0}^{U-1}, & if \ N_{wx} = 2U + 1 \\ \{-w_u\}_{u=0}^{U-1} \cup \{0\} \cup \{w_u\}_{u=0}^{U-2}, & if \ N_{wx} = 2U \end{cases}$$

$$W_y^{(gmf)} = \begin{cases} \{0\} \cup \{w_u\}_{u=0}^{U-1}, & if \ N_{wy} = U + 1 \\ \{0\} \cup \{w_u\}_{u=0}^{U-2}, & if \ N_{wy} = U \end{cases} \tag{12}$$

Here, $W_y^{(gmf)}$ also has roughly a half of $W_x^{(gmf)}$'s frequencies. In Equation 12, $\{w_u\}_{u=0}^{U-1} = \{w_0 = w_{min}, w_1, ..., w_u, ..., w_{U-1} = w_{max}\}$ is defined as a geometric series, where

$$w_u = w_{min} \cdot g^{u/(U-1)}; \quad where \ g = \frac{w_{max}}{w_{min}}; \quad u \in \{0, 1, ..., U - 1\} \tag{13}$$

$w_{min}, w_{max} \in \mathbb{R}^+$ are the minimum and maximum frequency (hyperparameters).

A visualization of $\mathcal{W}^{(fft)}$ and $\mathcal{W}^{(gmf)}$ can be seen in Figure 7a and 7b. To investigate the effect of NUFT frequency base selection on the effectiveness of polygon embedding, we compute the data variance on each Fourier frequency base across all 60K training polygon samples in MNIST-cplx70k training set. More specifically, for each polygonal geometry g_i in the MNIST-cplx70k training set, we first perform NUFT. Each g_i yield $N_w = N_{wx} N_{wy}$ complex valued NUFT features each of which corresponds to one frequency item $\mathbf{w}_k \in \mathbb{R}^2$ in

$\mathcal{W}^{(fft)}$ or $\mathcal{W}^{(gmf)}$. We compute the data variance[7] of each $\mathbf{w}_k$ across all 60K training polygons in MNIST-cplx70k and visualize them in Figure 7a and 7b. We can see that for linear grid frequency map $\mathcal{W}^{(fft)}$, most of the data variance is captured by the low frequency components while the high frequencies are less informative. When we switch to the geometric grid frequency map $\mathcal{W}^{(gmf)}$, more frequencies have higher data variance which is easier for the following MLP to learn from.
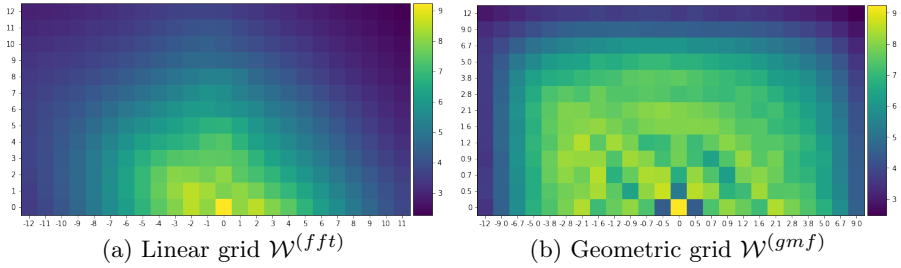

(a) Linear grid $\mathcal{W}^{(fft)}$ (b) Geometric grid $\mathcal{W}^{(gmf)}$

Fig. 7: An illustration of the full set of the linear grid frequency map $\mathcal{W}^{(fft)}$ and geometric grid frequency map $\mathcal{W}^{(gmf)}$.

5.2.3 NUFT-based Polygon Encoding

Finally, we can define the $NUFTspec$ polygon encoder $Enc_{NUFTspec}(g)$ as:

$$Enc_{NUFTspec}(g) = MLP_F(\Psi(F_{\mathcal{S}}^{(j)}(\mathbf{x}))) \qquad (14)$$

Here, $MLP_F(\cdot)$ is a K_F layer multi-layer perceptron in which each layer is a linear layer followed by a nonlinearity (e.g., ReLU), a skip connection, and a layer normalization layer [72]. $\Psi(\cdot)$ first extract a $2N_w$ dimension real value vector from $F_{\mathcal{S}}^{(j)}(\mathbf{x})$ and then normalize this spectral representation, e.g., L2, batch normalization.

5.3 Shape Classification Model

Given a polygonal geometry g, we encode it with a polygon encoder $Enc(\cdot)$ followed by a multilayer perceptron (MLP) $MLP_{shp}(\cdot)$ and a softmax layer to predict its shape class y:

$$P(r \mid g) = softmax(MLP_{shp}(Enc(g))). \qquad (15)$$

Here $Enc(\cdot)$ can be any baseline or our proposed encoders and $softmax(z_i) = e^{z_i} / \sum_{k=1}^{K} e^{z_k}$.

---

[7]We only compute the data variance for the real value part for each NUFT complex feature.

5.4 Spatial Relation Prediction Model

Given a polygon pair (g_{sub}, g_{obj}), the spatial relation prediction task aims at predicting a spatial relation between them such as topological relations, cardinal direction relations, and so on. In this work, we adopt a MLP-based spatial relation prediction model

$$P(r \mid g_{sub}, g_{obj}) = softmax(MLP_{rel}([Enc(g_{sub}); \ Enc(g_{obj})])), \qquad (16)$$

where $[Enc(g_{sub}); \ Enc(g_{obj})] \in \mathbb{R}^{2d}$ indicates the concatenation of the polygon embeddings of g_{sub} and g_{obj}. $MLP_{rel}(\cdot)$ takes this as input and outputs raw logits over all N_r possible spatial relations. $softmax(\cdot)$ normalizes it into a probability distribution $P(r \mid g_{sub}, g_{obj})$ over N_r relations.

5.5 Model Property Comparison

Theorem 1 *ResNet1D is loop origin invariant for simple polygons $g = (\mathbf{B}, \emptyset)$.*

Theorem 2 *NUFTspec is (1) loop origin invariant, (2) trivial vertex invariant, (3) part permutation invariant, and (4) topology aware for any polygonal geometry g.*

Now we compare different polygon encoders discussed in Section 4 as well as ResNet1D and NUFTspec based on: 1) their encoding capabilities – whether it can handle holes and multipolygons; 2) four polygon encoding properties (See Section 3).

By definition, VeerCNN, GCAE, and ResNet1D can not handle polygons with holes nor multipolygons. To allow these three models handle complex polygonal geometries, we need to concatenate the vertex sequences of different sub-polygons' exteriors and interiors. After doing that, the feed-in order of sub-polygons will affect the encoding results and the topological relations between exterior rings and interiors are lost. So none of them satisfy the part permutation invariance and topology awareness. Since VeerCNN consumes the polygon exterior as a coordinate sequence and encodes it with zero padding CNN layers, the origin vertex and the length of the sequence will affect its encoding results. So it is not loop origin invariant nor trivial vertex invariant. Both GCAE and ResNet1D are sensitive to trivial vertices which means they are not trivial vertex invariance. However, GCAE and ResNet1D are both loop origin invariant when encoding simple polygons. GCAE achieves this by representing the polygon exterior as an undirected graph where no origin is defined. ResNet1D achieves this by using circular padding in each 1D CNN and max pooling layer. However, this property can not be held for GCAE and ResNet1D if the input is complex polygonal geometries. We use "Yes*" in Table 2. Only DDSL [46, 47] and our NUFTspec can handle polygons with holes and multipolygons. They also satisfy all four properties. This is because the nature of NUFT. For ResNet1D and NUFTspec, we declare Theorem 1 and 2 whose proofs can be seen in Section 5.6.1 and 5.6.2. Table 2 shows the full comparison result.

Except for GCAE [29], the performance of all other polygon encoders listed in Table 2 are compared and analyzed on the shape classification (Section 6) and spatial relation prediction (Section 7) task. We do not include GCAE in our baseline models since its implementation is not open sourced.

Table 2: The comparison among different polygon encoders by properties such as whether it can handle polygon with holes or multipolygons as well as the four polygon encoding properties we discuss in Section 3. "Yes*" indicates that loop origin invariance only holds for GCAE and ResNet1D if the input is simple polygons.

Property	Type	Holes	Multipolygons	Loop	TriV	ParP	Topo
VeerCNN [28]	Spatial	No	No	No	No	No	No
GCAE [29]	Spatial	No	No	Yes*	No	No	No
DDSL [46, 47]	Spectral	Yes	Yes	Yes	Yes	Yes	Yes
ResNet1D	Spatial	No	No	Yes*	No	No	No
NUFTspec	Spectral	Yes	Yes	Yes	Yes	Yes	Yes

5.6 Theoretical Proofs of the Model Properties

5.6.1 Proofs of Theorem 1

In ResNet1D, circular padding is used in the convolution layers with stride 1. Given a polygon $p = (\mathbf{B}, h = \emptyset)$, circular padding wraps the vector $\mathbf{B}$ on one end around to the other end to provide the missing values in the convolution computations near the boundary. Thus, $\text{CNN1D}_{3\times1}^{d,1,1}(\mathbf{L}_s\mathbf{B}) = \mathbf{L}_s\text{CNN1D}_{3\times1}^{d,1,1}(\mathbf{B})$ for any input $\mathbf{B}$. Max pooling layer with stride 1 and circular padding has the similar property. $\text{MP1D}_{2\times1}^{1,1}(\mathbf{L}_s\mathbf{B}) = \mathbf{L}_s(\text{MP1D}_{2\times1}^{1,1}\mathbf{B})$. Trivially, $\text{BN1D}(\mathbf{L}_s\mathbf{B}) = \mathbf{L}_s\text{BN1D}(\mathbf{B})$ and $\text{ReLU}(\mathbf{L}_s\mathbf{B}) = \mathbf{L}_s\text{ReLU}(\mathbf{B})$. In the end, there is a global maxpooling $\text{GMP1D}(\mathbf{L}_s\mathbf{B}) = \text{GMP1D}(\mathbf{B})$. With these layers as components, ResNet1D would keep the loop origin invariance.

5.6.2 Proofs of Theorem 2

Proof of Theorem 2 (1) - Loop Origin Invariance.
Given a simple polygon $p = (\mathbf{B}, \emptyset)$, we convert it to a 2-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g}$ similar to what is shown in Figure 8a. Since $\{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g}$ is an *unordered set* of 2-simplexes, for any loop matrix $\mathbf{L}_s$, Polygon $p^{(s)} = (\mathbf{L}_s\mathbf{B}, \emptyset)$ will have exactly the same 2-simplex mesh as p - $\mathcal{S}^{(j)}$. 2-simplex mesh $\mathcal{S}^{(j)}$ is the input of our NUFTspec. So the output polygon embedding $Enc_{NUFTspec}(p)$ is invariant to any loop transformation $\mathbf{L}_s$ on Polygon p's exterior $\mathbf{B}$. Similarly, we can prove that the encoding results of NUFTspec is also invariant to a loop transformation on the boundary of one of holes of the complex polygon geometry g.
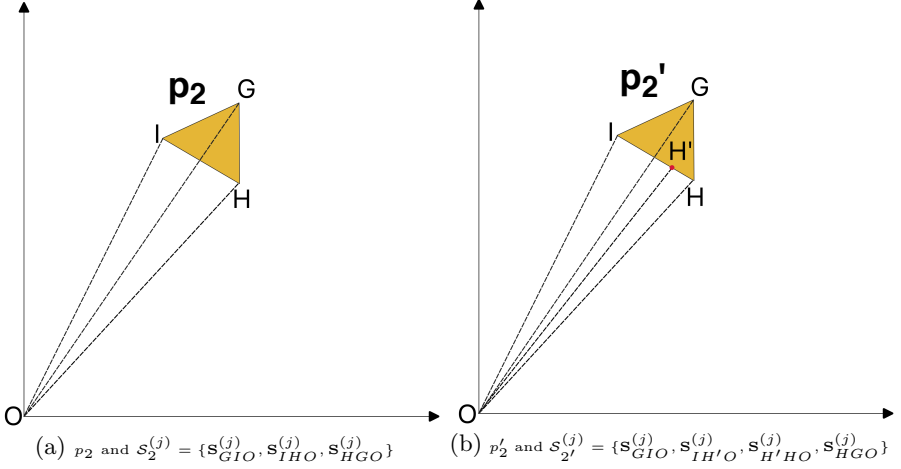
(a) p_2 and $\mathcal{S}_2^{(j)} = \{\mathbf{s}_{GIO}^{(j)}, \mathbf{s}_{IHO}^{(j)}, \mathbf{s}_{HGO}^{(j)}\}$ (b) p_2' and $\mathcal{S}_{2'}^{(j)} = \{\mathbf{s}_{GIO}^{(j)}, \mathbf{s}_{IH'O}^{(j)}, \mathbf{s}_{H'HO}^{(j)}, \mathbf{s}_{HGO}^{(j)}\}$

Fig. 8: A illustration to facilitate the proof of Theorem 2 . (a) Polygon p_2 is a simple polygon from Figure 5c. We convert it to a 2-simplex mesh $\mathcal{S}_2^{(j)} = \{\mathbf{S}_{GIO}^{(j)}, \mathbf{S}_{IHO}^{(j)}, \mathbf{S}_{HGO}^{(j)}\}$. (a) Polygon p_2' is also a simple polygon which has the same shape as p_2 but add an additional trivial vertex H'. We convert it to a 2-simplex mesh $\mathcal{S}_{2'}^{(j)} = \{\mathbf{S}_{GIO}^{(j)}, \mathbf{S}_{IH'O}^{(j)}, \mathbf{S}_{H'HO}^{(j)}, \mathbf{S}_{HGO}^{(j)}\}$.

Proof of Theorem 2 (2) - Trivial Vertex Invariance.

We use Polygon p_2 and p_2' shown in Figure 8a and 8b as an example to demonstrate the proof. The only difference between them is that p_2' has an additional trivial vertex H' while p_2 and p_2' have the same *shape*. They have different 2-simplex meshes: $\mathcal{S}_2^{(j)} = \{\mathbf{S}_{GIO}^{(j)}, \mathbf{S}_{IHO}^{(j)}, \mathbf{S}_{HGO}^{(j)}\}$ and $\mathcal{S}_{2'}^{(j)} = \{\mathbf{S}_{GIO}^{(j)}, \mathbf{S}_{IH'O}^{(j)}, \mathbf{S}_{H'HO}^{(j)}, \mathbf{S}_{HGO}^{(j)}\}$. The Piecewise-Constant Function (PCF) $f_{\mathcal{S}}^{(j)}(\mathbf{x}) = \sum_{n=1}^{N_g} f_n^{(j)}(\mathbf{x})$ defined on the simplex mesh $\mathcal{S}_2^{(j)}$ is essentially a *summation* over the individual density function $f_n^{(j)}(\mathbf{x})$ defined on each simplex of $\mathcal{S}_2^{(j)}$. Since p_2 and p_2' have the same *shape*, the PCF defined on them should be exactly the same. Since NUFTspec polygon embedding is derived from the NUFT of the PCF over a 2-simple mesh. So we can conclude that the NUFTspec polygon embeddings of p_2 and p_2' should be the same. In other words, NUFTspec is trivial vertex invariant.

Proof of Theorem 2 (3) - Part Permutation Invariance.

Given a multipolygon $q = \{p_i\}$, its 2-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g}$ (similar to Figure 5d) is an unordered set of signed 2-simplexes/triangles. Changing the feed-in order of polygon set $\{p_i\}$ will not affect the resulting 2-simplex mesh. So NUFTspec is part permutation invariant.

Proof of Theorem 2 (4) - Topology Awareness.

Given a polygon $p = (\mathbf{B}, h = \{\mathbf{H}_j\})$ with holes, its 2-simplex mesh $\mathcal{S}^{(j)} = \{\mathbf{S}_n^{(j)}\}_{n=1}^{N_g}$ is consist of oriented 2-simplexes. Since we require each polygonal geometry is oriented correctly, the right-hand rule can be used to compute the signed content of each 2-simplex. So the topology of polygon $p = (\mathbf{B}, h = \{\mathbf{H}_j\})$ is preserved during the polygon-simple mesh conversion. Thus, NUFTspec is aware of the topology of the input polygon geometry.

6 Shape Classification Experiments

Shape classification is an essential task for many computer vision [48] and cartographic applications [29, 31]. In this study we focus on a large scale polygon classification dataset MNIST-cplx70k, which is based on the commonly used MNIST dataset.

6.1 MNIST-cplx70k Dataset

According to Table 1, MNIST is the only large-scale open-sourced shape classification dataset with 60K training samples. The original MNIST data was originally designed for optical character recognition (OCR) which uses images to present shapes, not polygons. Jiang et al. [46] convert MNIST into a polygon shape dataset for shape classification purpose. Following their practice, we construct a polygon-based shape classification dataset – MNIST-cplx70k.

The original MNIST pixel image is up-sampled using interpolation and contoured to get a polygonal representation of the digit by using the functions provided by Jiang et al. [46]. Then we simplify each geometry by: 1) making sure that each polygon sample contains 500 unique vertices in order to do mini-batch training; 2) dropping very small holes and sub-polygons while keeping large ones. The result is a shape classification dataset of 70k examples (see Table 1 for its detailed statistics). Figure 9 shows the histogram statistic of the number of sub-polygons and number of holes of each samples in the resulting MNIST-cplx70k training and testing set.

6.2 Baselines and Model Variations

We use the same shape classification module as shown in Equation 15 but vary the polygon encoders we used. We consider multiple polygon encoders for the shape classification task on MNIST-cplx70k dataset:

1. **VeerCNN** [28]: We strictly follow the TensorFlow implementation[8] of Veer et al. [28] and re-implement it in PyTorch. The polygon embedding is used for shape classification.
2. **ResNet1D**: We implement ResNet1D as described in Section 5.1 and use it for shape classification.

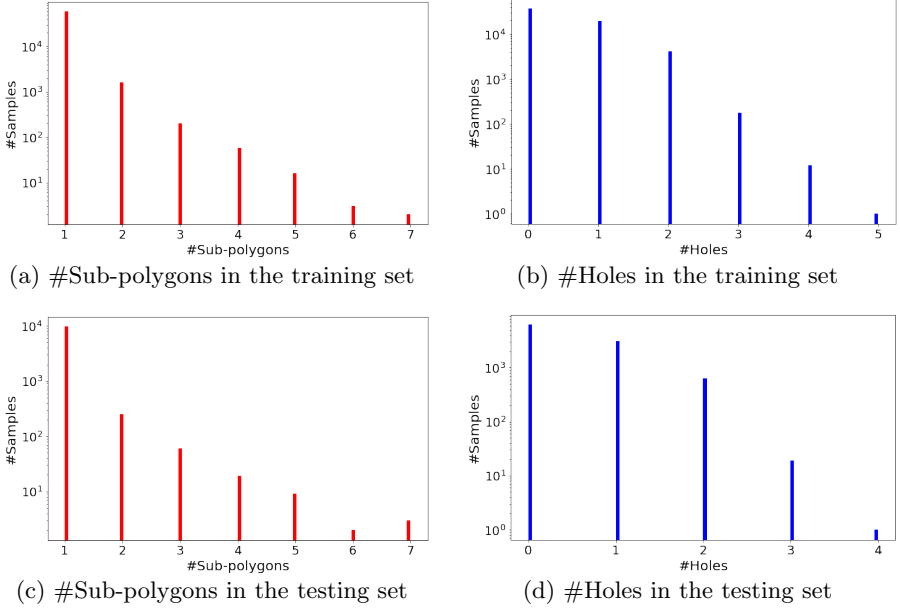
[8]https://github.com/SPINlab/geometry-learning

(a) #Sub-polygons in the training set

(b) #Holes in the training set

(c) #Sub-polygons in the testing set

(d) #Holes in the testing set

Fig. 9: The statistic of the complexity of polygonal geometries in the MNIST-cplx70k training and testing set.

3. **DDSL+MLP**: This is a model developed based on the DDSL model proposed by Jiang et al. [46]. Similar to NUFTspec, DDSL+MLP first transforms a polygonal geometry into the spectral space with NUFT. Instead of learning embedding directly on these NUFT features, it converts this NUFT representation back to the spatial space as an image by Inverse Fast Fourier Transform (IFFT). Note that in order to make sure the NUFT features can be later used for IFFT, we can only use linear grid feature map $\mathcal{W}^{(fft)}$ (See Definition 9) as the Fourier feature bases for NUFT in DDSL+MLP. This NUFT-IFFT transformation is essentially a vector-to-raster operation. DDSL+MLP converts the resulting 2D image into a 1D feature vector and directly applies a multi-layer percetron (MLP) on it to learn polygon embeddings.

4. **DDSL+PCA+MLP**: Directly applying an MLP on 1D image feature vector can lead to overfitting. In order to prevent overfitting, DDSL+PCA+MLP applies a Principal Component Analysis (PCA) on the 1D NUFT-IFFT image feature vectors across all training shapes in MNIST-cplx70k. The first K_{PCA} PCA dimensions (which accounts for 80+% of the data variances) of the NUFT-IFFT image feature vector are extracted and fed into an MLP.

5. **NUFTspec[fft]+MLP**: We first use NUFT to transform a polygonal geometry into the spectral space with linear grid $\mathcal{W}^{(fft)}$ (indicated by "[fft]"). Then an MLP is directly applied on the resulting NUFT features.

6. **NUFTspec[fft]+PCA+MLP**: Similarly, in order to prevent overfitting, we apply a projection of NUFT features down to the first K_{PCA} PCA dimensions (which account for 80+% of the data variance) before the MLP.

7. **NUFTspec[gmf]+MLP**: One advantage of NUFTspec is that since NUFTspec does not require IFFT, we do not need to use regularly spaced frequencies ($\mathcal{W}^{(fft)}$) but choose whatever frequency map works best for a given task. So compared with NUFTspec[fft]+MLP, we switch to the geometric grid frequency map $\mathcal{W}^{(gmf)}$ as proposed in Mai et al. [13].

8. **NUFTspec[gmf]+PCA+MLP**: Similar to NUFTspec[gmf]+MLP but we perform an extra PCA before feeding into the MLP.

Both VeerCNN and ResNet1D can only encode simple polygons by design while MNIST-cplx70k contains complex polygonal geometries. So given a complex polygon geometry, we concatenate the vertex coordinate sequences of all sub-polygons' exterior and interior rings before feeding it into VeerCNN or ResNet1D. We call this preprocess step as *boundary concatenation operation*. For example, Geometry $q = \{p_0, p_1\}$ shown in Figure 5a is represented as $[\mathbf{x}_A^T; \mathbf{x}_B^T; \mathbf{x}_C^T; \mathbf{x}_D^T; \mathbf{x}_E^T; \mathbf{x}_F^T; \mathbf{x}_G^T; \mathbf{x}_H^T; \mathbf{x}_I^T; \mathbf{x}_J^T; \mathbf{x}_K^T; \mathbf{x}_L^T]$ before feeding into VeerCNN and ResNet1D. After boundary concatenation operation, a complex polygonal geometry is converted to a single boundary vertex sequence. It is like using one stroke to draw a complex polygonal geometry [73] whereas all topology information is lost. In this situation, circular padding does not ensure loop origin invariance for ResNet1D anymore.

VeerCNN and ResNet1D are spatial domain polygon encoders while the last six models are spectral domain polygon encoders. The last six NUFT-based polygon encoders essentially provide a framework for the ablation study on the usability of NUFT features for polygon representation learning. We vary several components of the polygon encoder: 1) **DDSL v.s. NUFTspec** – whether to learn representation in the spectral domain (NUFTspec) or the spatial domains (DDSL); 2) **fft v.s. gmf** – whether to use linear grid or geometric grid frequency map; 3) **MLP v.s. PCA+MLP**: whether to apply PCA for feature projection before the MLP layers. Here, we keep the learning neural network component to be an MLP to make sure the model performance differences are all from those three model variations but not from different neural architectures.

The whole model architectures of ResNet1D, NUFTspec as well as all baselines are implemented in PyTorch. All models are trained and evaluated on one Linux machine with 256 GB memory, 56 CPU cores, and 2 GeForce GTX 1080 Ti CUDA GPU (12 GB memory each).

6.3 Main Evaluation Results

Figure 10 compares the shape classification accuracy of the eight models described in Section 6.2 on the MNIST-cplx70k testing set. Each bar indicates the performance of one model, where we mark the estimated standard deviations from 10 runs for each model setting. The hyperparameter tuning detailed

are discussed in Appendix A.2. The best hyperparameter combinations for different models are listed in Table A2. From Figure 10, we can see that:

1. VeerCNN performs the worse. Our ResNet1D achieves the best performance while our NUFTspec[gmf]+PCA+MLP comes as the second best.
2. PCA significantly improves the accuracy of all three settings. This is especially evident for NUFTspec[fft], which supports our intuition that a linear grid introduces too many higher frequency features which are noisy and make learning hard (See Figure 7a). For more analysis on the PCA features, please see Section 6.4.
3. When using the linear grid frequency map $\mathcal{W}^{(fft)}$, two DDSL models outperform their corresponding NUFTspec counterparts, i.e., DDSL+MLP > NUFTspec[fft]+MLP, and DDSL+PCA+MLP > NUFTspec[fft]+PCA+MLP.
4. However, when we switch to the geometric grid $\mathcal{W}^{(gmf)}$, NUFTspec outperforms DDSL, i.e., NUFTspec[gmf]+MLP > DDSL+MLP, and NUFTspec[gmf]+PCA+MLP > DDSL+PCA+MLP. These differences are statistically significant given the estimated standard deviations. Note that for the geometric grid $\mathcal{W}^{(gmf)}$, IFFT is no longer applicable so that we can not use DDSL logic to transform those spectral features into the spatial domain. This shows the flexibility of NUFTspec in terms of the choice of Fourier frequency maps which have a significant impact on the model performance, while DDSL is restricted to uniform frequency sampling.

In order to understand how different polygon encoders handle polygons with different complexity, we split the 10K polygon samples in the MNIST-cplx70k testing dataset into 6 groups using the number of sub-polygons of each sample, and compare the performances of VeerCNN, DDSL+PCA+MLP, ResNet1D, and NUFTspec[gmf]+PCA+MLP on each group. The numbers in the parenthesis are the estimated standard deviations. The results are shown in Table 3. We can see that:
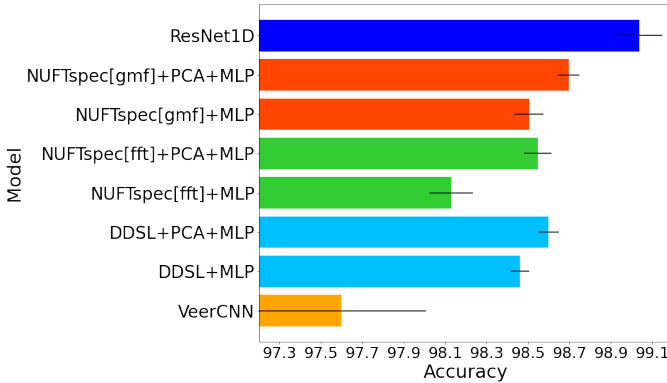

Fig. 10: Overall classification accuracy comparison among different polygon encoders on the MNIST-cplx70k testing set. We also plot the standard deviation of each model's performance as black lines on each bar based 10 runs of each model. We test multiple mode variations of DDSL and NUFTspec.

Table 3: Shape classification results on different polygon groups of the MNIST-cplx70k testing set. The "#Sub-Polygon" row indicates the number of sub-polygons each shape sample have in each group. "6+" indicates all shape samples who have at least 6 sub-polygons. "ALL" indicates the evaluation on the whole testing set. "#Samples" row shows the number of shape samples in each groups. "()" indicates the standard deviations of 10 runs of each model.

#Sub-Polygon	1	2	3	4	5	6+	ALL
#Samples	9,656	250	61	19	9	5	10,000
VeerCNN	98.06	93.60	88.52	68.42	55.56	60.00	97.60 (0.43)
DDSL+PCA+MLP	98.78	95.20	**96.72**	**84.21**	**77.78**	**80.00**	98.60 (0.05)
ResNet1D	**99.25**	**96.00**	**96.72**	63.16	66.67	40.00	**99.00** (0.05)
NUFTspec[gmf]+PCA+MLP	98.87	**96.00**	**96.72**	**84.21**	**77.78**	60.00	98.70 (0.05)

Table 4: Shape classification results on different polygon groups of the MNIST-cplx70k testing set (the same as Table 3) after training models on the MNIST-cplx70k+AUG dataset, which adds data augmentation for samples with more than 3 subpolygons.

#Sun-Polygon	1	2	3	4	5	6	ALL
#Samples	9,656	250	61	19	9	5	10,000
VeerCNN	97.41	87.60	83.61	73.68	66.67	60.00	96.99
DDSL+PCA+MLP	98.76	96.00	95.08	78.95	**88.89**	80.00	98.61
ResNet1D	**99.10**	**96.80**	96.72	68.42	77.78	60.00	**98.93**
NUFTspec[gmf]+PCA+MLP	98.81	**96.80**	**98.36**	**89.47**	**88.89**	80.00	98.76

1. Compared with DDSL+PCA+MLP and NUFTspec[gmf]+PCA+MLP, ResNet1D does poorly on multi-polygon samples, especially on samples with more than 3 subpolygons. This indicates that the superority of ResNet1D on MNIST-cplx70k dataset is mainly because most of samples (96.56%) in MNIST-cplx70k testing set are single polygons. Therefore, for shape classification tasks with a higher proportion of complex polygonal geometry samples, we expect NUFTspec[gmf]+PCA+MLP to perform better than ResNet1D.

2. Comparing DDSL+PCA+MLP and NUFTspec[gmf]+PCA+MLP, we see that they both can handle multi-polygon samples well due to the NUFT component. However, NUFTspec[gmf]+PCA+MLP performs better at 1 and 2 sub-polygon groups.

3. We find out that DDSL+PCA+MLP and NUFTspec[gmf]+PCA+MLP have the same performance on the 3, 4, and 5 sub-polygon groups. This indicates that both models fail on the same number of samples in each group. After looking into the actual predictions, we find out that they actually fail for different samples.

Since there are much less samples with more than 3 sub-polygons in MNIST-cplx70k training and testing dataset (see Figure 9a and 9c), it might be questionable to draw a conclusion that compared with DDSL+PCA+MLP

and NUFTspec[gmf]+PCA+MLP, ResNet1D does poorly on samples with more than 3 subpolygons. The lower performance of ResNet1D on samples with more than 3 subpolygons might be due to *the unbalanced training data with respect the number of samples with different sub-polygons.* To mitigate the unbalanced nature of the training data, we perform data augmentation to increase the number of multi-polygon samples in the training set. For each training sample with more than 3 subpolygons, in addition to the original sample, we generate 10 extra samples by adding random Gaussian noise to the vertices of current polygonal geometry. We denote this augmented training set as MNIST-cplx70k+AUG. We train four polygon encoders on MNIST-cplx70k+AUG dataset and evaluate them on the original MNIST-cplx70k testing set.

The evaluation results on different polygon groups are shown in Table 4. We can see that, after adding extra multipolygon samples, the overall accuracy of ResNet1D decrease whereas the over accuracy of NUFTspec[gmf]+PCA+MLP increase when we compare them with those in Table 3. Moreover, we can still observe that both DDSL+PCA+MLP and NUFTspec[gmf]+PCA+MLP can outperform ResNet1D on 4, 5, 6+ sub-polygon groups. This result demonstrates that ResNet1D has an intrinsic model bias to perform poorly on multi-polygon samples as opposed to model variance (lack of training examples).

6.4 Analysis of PCA-based Models

We also investigate the nature of the feature representations of those three PCA models – DDSL+PCA+MLP, NUFTspec[fft]+PCA+MLP and NUFTspec[gmf]+PCA+MLP. The results are show in Figure 11. Figure 11a compares the PCA explained variance curves of three models when $N_{wx} = 24$. Instead of fixing N_{wx}, Figure 11b compares three PCA models based on the number of PCA components needed to explain 90% of the data variance with different N_{wx}. We can see that for any given N_{wx}, these three models need different numbers of PCA components with NUFTspec[gmf]+PCA+MLP being the most compact and NUFTspec[fft]+PCA+MLP being the least compact. The curve of NUFTspec[gmf]+PCA+MLP is very flat indicating that geometric frequency introduces much less noise when N_{wx} increases.

6.5 Understanding the Four Polygon Encoding Properties

Section 6.3 shows that NUFTspec and ResNet1D are two best models on MNIST-cplx70k dataset. In order to deeply understand how those four polygon encoding properties discussed in Section 3 affect the performance of NUFTspec and ResNet1D, we conduct four follow-up experiments. To test the rotation invariance property of their proposed models, both Deng et al. [74] and Esteves et al. [75] kept the training dataset unchanged and modified the testing data by rotating each testing shape. They investigated how the testing shape rotations affect the model performance. Inspired by them, we also
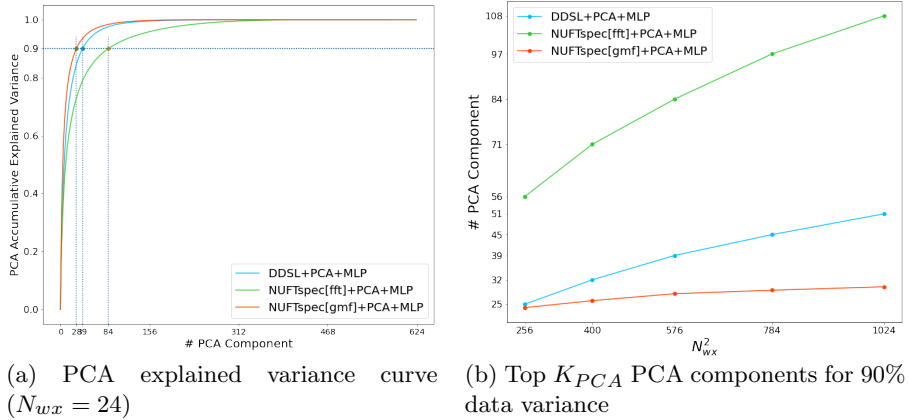
(a) PCA explained variance curve $(N_{wx} = 24)$

(b) Top K_{PCA} PCA components for 90% data variance

Fig. 11: The comparison of DDSL+PCA+MLP, NUFTspec[fft]+PCA+MLP and NUFTspec[gmf]+PCA+MLP on how their PCA components explain the data variance. (a) The PCA explained variance curves of three models when $N_{wx} = 24$. (b) The curve of the number of top K_{PCA} PCA components that can explain 90% of data variance for MNIST-cplx70k training set with different N_{wx}. We can see that compared with the other two models, NUFTspec[gmf]+PCA+MLP has a more compact set of PCA features.

keep the training dataset of MNIST-cplx70k unchanged and adopt four ways to modify the polygon representations in the MNIST-cplx70k testing dataset. We compare the performances of NUFTspec and ResNet1D on these modified datasets.

The first three properties are invariance properties. So what we expect for a polygon encoder is that when we 1) loop around the exterior/interiors by starting with different vertices, or 2) add/delete trivial vertices, or 3) permute the feed-in order of different sub-polygons, the resulting polygon embedding is invariant since the shape of the input polygon does not change. So for the first three properties, we modify the polygon representations in MNIST-cplx70k testing set while keeping their shape invariant. We call them *shape-invariant geometry modifications*.

The topology awareness property is not an invariance property. So our expectation is that when the topology of a polygon changes, the polygon embedding should be different even if they have the same vertex information such as $q = \{p_0, p_1\}$ and $q'' = \{p_0'', p_1, p_2\}$ shown in Figure 5a and 5c.

Figure 12 visualizes the evaluate results for each property.

1. **Loop origin invariance (Loop)**: As for all training and testing polygons in original MNIST-cplx70k dataset, we always start looping around each polygon's exterior/interiors with its upper most vertex. To study the loop origin invariance property, We randomize the starting point of the coordinate sequence of each sub-polygon's exterior or interiors. We refer to this method as *loop origin randomization*. For example, as shown
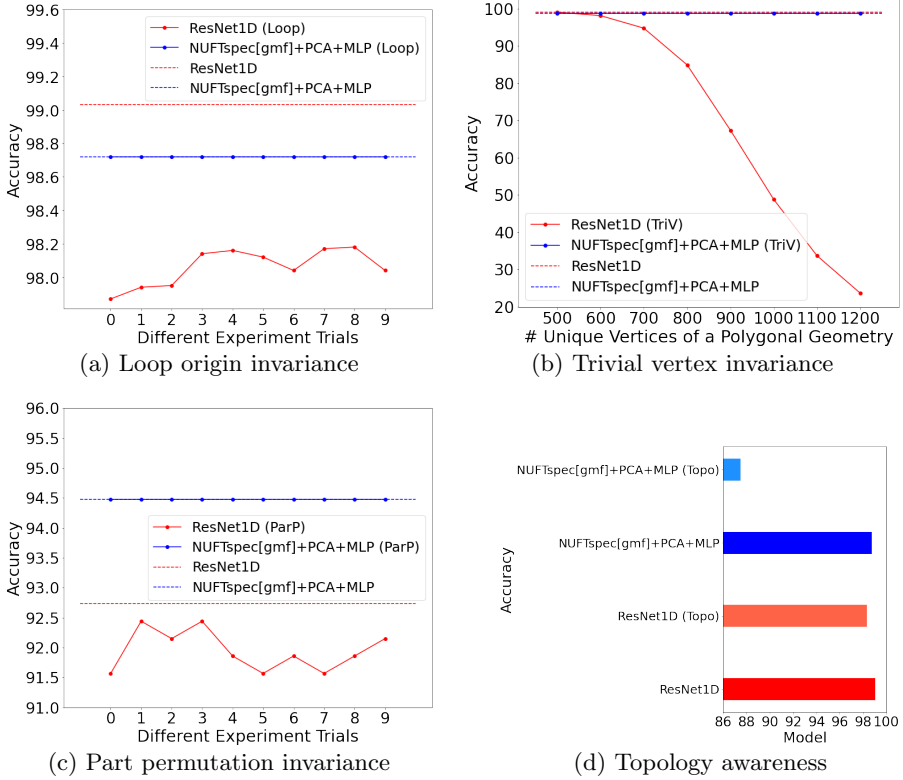
(a) Loop origin invariance

(b) Trivial vertex invariance

(c) Part permutation invariance

(d) Topology awareness

Fig. 12: The experiments on the four polygon encoding properties of NUFTspec[geometric]+PCA+MLP and ResNet1D. "ResNet1D" and "NUFT-spec[geometric]+PCA+MLP" indicate the evaluations of these two models on the original MNIST-cplx70k testing set while model names with "(*)" indicate their results on modified datasets. "(*)" indicates different modification methods for different properties: (a) Loop origin invariance: looping around the testing polygons' exterior/interiors with 10 different origins; (b) Trivial vertex invariance: adding different numbers of trivial vertices; (c) Part permutation invariance: permuting the feed-in order of different sub-polygons; (d) Topology awareness: converting polygon holes into subpolygons.

in Figure 5a, the exterior of Polygon p_0, i.e., $\mathbf{B}_0$, can be written as $\mathbf{B}_0 = [\mathbf{x}_A^T; \ \mathbf{x}_B^T; \ \mathbf{x}_C^T; \ \mathbf{x}_D^T; \ \mathbf{x}_E^T; \ \mathbf{x}_F^T] \in \mathbb{R}^{6 \times 2}$ with $\mathbf{x}_A$ as the start point. When we select $\mathbf{x}_D$ as the start point, the exterior of Polygon p_0 becomes $\mathbf{B}_0^{(3)} = \mathbf{L}_3 \mathbf{B}_0 = [\mathbf{x}_D^T; \ \mathbf{x}_E^T; \ \mathbf{x}_F^T; \ \mathbf{x}_A^T; \ \mathbf{x}_B^T; \ \mathbf{x}_C^T]$ which has an identical shape as $\mathbf{B}_0$. In total, we generate 10 independent testing datasets based on the MNIST-cplx70k original testing set by using the above method. We evaluate the trained ResNet1D and NUFTspec model on these different modified testing sets whose evaluation results are shown as two curves – "ResNet1D (Loop)" and "NUFTspec[geometric]+PCA+MLP (Loop)" in Figure 12a.

We also show their performance on the original testing set as the dotted lines. We can see that while NUFTspec's performance is unaffected under loop origin randomization, ResNet1D is affected with roughly absolute 1% performance decrease, which is statistically significant. Note that ResNet1D is loop origin invariance only if the polygon is a simple polygon. However, MNIST-cplx70k contains multiple complex polygonal geometries. In these cases, loop origin invariance property of ResNet1D does not hold. That is why we see this performance drop.

2. **Trivial vertex invariance (TriV)**: For each polygonal geometry in the MNIST-cplx70k testing dataset, we upsample its vertices to have more than the initial 500 points (600 - 1200) by adding trivial vertices. Figure 5b shows the example of trivial vertices (red points). We compare the performance of ResNet1D and NUFTspec on these upsampled testing datasets which indicate as "ResNet1D (TriV)" and "NUFTspec[geometric]+PCA+MLP (TriV)" in Figure 12b. We can see that while the performance of NUFTspec is unaffected no matter how many vertices we upsampled, the performance of ResNet1D decreases dramatically when the number of vertices increases.

3. **Part permutation invariance (ParP)**: The feed-in order of sub-polygons in each polygonal geometry sample in the original MNIST-cplx70k dataset follows the normal raster scan order – from up to down and from left to right. To test the part permutation invariance property, we do random part permutation for each multipolygon in MNIST-cplx70k testing set. Since this part permutation operation will only affect the multipolygon shape samples, we only compare the performance of ResNet1D and NUFTspec on this subset of the testing set which consists of 344 multipolygon samples. Similar to Figure 12a, we also do 10 different experiment trials. The performances of ResNet1D and NUFTspec on these permuted testing datasets are indicated as "ResNet1D (ParP)" and "NUFTspec[geometric]+PCA+MLP (ParP)". Dotted lines indicates their performance on the original dataset. We can see that while NUFTspec is unaffected, ResNet1D's performance decreased by 0.8%.

4. **Topology awareness (Topo)**: From a theoretical perspective, it is easy to see that NUFTspec is topology aware since NUFT knows which points are inside the polygon and which outside whereas ResNet1D is not as shown in Section 5.6.2. However, since topology awareness is not an invariance property, it is rather difficult to show with experiments. Nevertheless, to align with the experiments of other three properties, we did a similar experiment by modifying the polygon representations. More specifically, for each polygonal geometry with holes, we delete the holes and use the holes' coordinate sequence to construct a new sub-polygon for the current geometry. One example consists of the multipolygon $q = \{p_0, p_1\}$ and multipolygon $q'' = \{p_0'', p_1, p_2\}$ shown in Figure 5a and 5c. The hole of sub-polygon p_0 is deleted. Instead, it is instantiated as a new sub-polygon p_2. By doing that, we change the topology of a geometry while keeping the vertices

unchanged. Note that when a hole become the exterior of a sub-polygon, the coordinate sequence should switch from clockwise rotation to counterclockwise rotation. We evaluate ResNet1D and NUFTspec on the original and modified MNIST-cplx70k testing set as shown in Figure 12d. "ResNet1D (Topo)" and "NUFTspec[gmf]+PCA+MLP (Topo)" indicate the results on the modified dataset. We can see that when changing the polygon topology, the performances of both models are affected while NUFTspec is affected more severely. This is actually expected, or even desired since when the topology of a polygonal geometry changes, the shape also changes and the decision of shape classification should also change. Figure 12d shows that NUFTspec are more sensitive to topological changes of polygons.
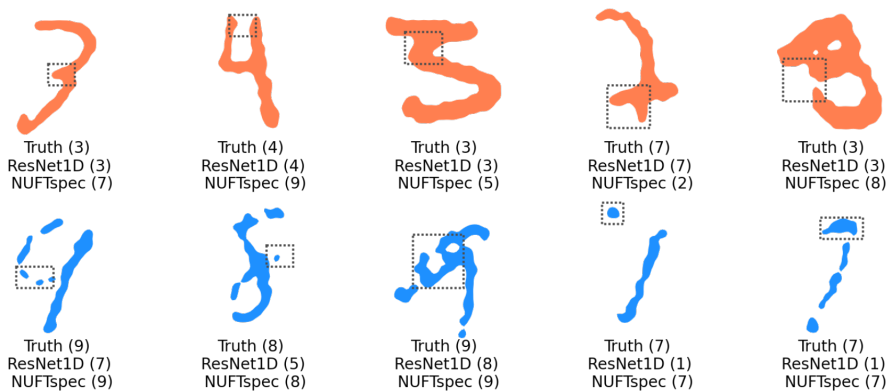
In conclusion, based on the above study, we can see that compared with ResNet1D, NUFTspec is *more robust to loop origin randomization, vertex upsampling, and part permutation operation whereas it is more sensitive to topological changes. In other words, NUFTspec retains identical performance when polygons undergo shape-invariant geometry modifications due to the invariance inherented from the NUFT representation, whereas models that directly utilize the polygon vertex features such as ResNet1D suffer significant performance degradations.*
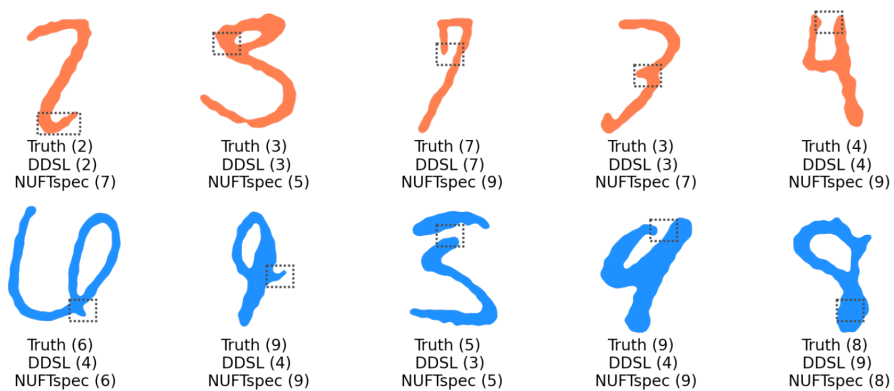
6.6 Qualitative Analysis

Figure 13 shows the qualitative analysis results of DDSL+PCA+MLP, ResNet1D, and NUFTspec[gmf]+PCA+MLP. We visualize some illustrated examples in which these models win or fail.

Figure 13a shows sampled winning and failing examples comparing NUFTspec[gmf]+PCA+MLP and ResNet1D. From the lower row of Figure 13a, we can see that NUFTspec[gmf]+PCA+MLP is able to jointly consider all sub-polygons of a multipolygon sample and the spatial relations among them during prediction while ResNet1D fails to do so. From the upper row of Figure 13a, we can see that NUFTspec[gmf]+PCA+MLP sometimes fails because it can not recognize the shape details which are sometimes important for classification (such as the 1st case – "7" vs "3").

Figure 13b shows sampled positive and negative examples comparing NUFTspec[gmf]+PCA+MLP and DDSL+PCA+MLP. From the model design choice, we expect NUFTspec works better than DDSL in certain cases because DDSL is essentially a vector-to-raster operation that may lose global information that is important for a given task. As shown in the first 2 examples in the lower row, DDSL+PCA+MLP may classify shapes as "4" because topologically the shapes do conform to "4". However, based on the overall shape NUFTspec[gmf]+PCA+MLP is able to determine that they are more likely "6" and "9". The small "local" protruding elements highlighted in dashed boxes in these examples are not intended by the writer, and have a relatively small impact on NUFTspec's features, since all Fourier features are global statistics. Similarly having no hole in the "8" example (the last example in

(a) ResNet1D v.s. NUFTspec[gmf]+PCA+MLP

(b) DDSL+PCA+MLP v.s. NUFTspec[gmf]+PCA+MLP

Fig. 13: A qualitative analysis on the performance of DDSL+PCA+MLP, ResNet1D, and NUFTspec[gmf]+PCA+MLP on the MNIST-cplx70k testing set. In each example's caption, "*" in "Truth (*)" indicates the ground truth labels while the other numbers in "()" indicates the predicted labels for different models. The dashed boxes highlight the parts which might lead to the wrong predictions of polygon encoders. (a) The upper row contains examples where ResNet1D predicts correctly while NUFTspec[gmf]+PCA+MLP fails. The lower row contains examples in which these two models perform otherwise. (b) The upper row contains examples where DDSL+PCA+MLP predicts correctly while NUFTspec[gmf]+PCA+MLP fails. The lower row contains examples in which these two models perform otherwise.

the lower row of Figure 13b) prevents DDSL+PCA+MLP from classifying it as "8", while NUFTspec[gmf]+PCA+MLP can well handle this topological abnormality. On the other hand, this insensitivity to local features may also hurt NUFTspec[gmf]+PCA+MLP when they are important. This is evident in the upper row in Figure 13b for number "2", "3", and "7" where a small

protruding structure, or a small gap indicates the topological changes that are intended by the writer.

Based on the above experiment results, we can see that compared with ResNet1D, both DDSL and NUFTspec are better at handling multi-polygon samples. While DDSL focuses on the localized features, NUFTspec pays attention to the global shape information. Moreover, since NUFTspec does not have the IFFT step, it is more flexible in terms of choosing the frequency maps.

7 Spatial Relation Prediction Experiments

The polygon-based spatial relation prediction is an important component for GeoQA [39]. To study the effectiveness of NUFTspec and ResNet1D on this task, we construct two real-world datasets DBSR-46K and DBSR-cplx46K for the evaluation purpose based on DBpedia and OpenStreetMap.

7.1 DBSR-46K and DBSR-cplx46K Dataset

Since there is no existing benchmark dataset available for this task, we construct two real-world datasets - DBSR-46K and DBSR-cplx46K- based on DBpedia Knowledge Graph as well as OpenStreetMap. DBSR-46K and DBSR-cplx46K use the same entity set $\mathcal{E}$ and triple set $\mathcal{T}$ and the only different is that DBSR-46K uses simple polygons as entities' spatial footprints while DBSR-cplx46K allows complex polygonal geometries. The dataset construction steps are described as below:

1. We first select a meaningful set of properties $R = \{r_i\}_{i=1}^{N_r}$ from DBpedia representing different spatial relations.
2. Then we collect all triples $\{(e_{sub}, r_i, e_{obj})\}$ from DBpedia whose relation $r_i \in R$ and e_{sub}, e_{obj} are geographic entities.
3. Next, we filter out triples whose subject e_{sub} or object e_{obj} is located outside the contiguous US. The resulting triple set $\mathcal{T} = \{(e_{sub}, r_i, e_{obj})\}$ forms a sub-graph of DBpedia with the entity set $\mathcal{E}$.
4. For each entity $e \in \mathcal{E}$, we obtain corresponding Wikidata ID by using *owl:sameas* links.
5. With each entity e's Wikidata ID, we can obtain its polygonal geometry from OpenStreetMap by using Overpass API[9].
6. The raw polygonal geometries from OpenStreetMap are very detail and complex. For example, Keweenaw County, Michigan is represented as a multipolygon that is consist of 462 sub-polygons. Lake Superior has in total 3130 holes and 206661 vertices in its polygon representation. United States has 128873 vertices. Figure 14a-14c show statistic about the complexity of these raw geometries. We simplify those polygonal geometries collected from OpenStreetmap and make two datasets: DBSR-46K and DBSR-cplx46K.
7. As for DBSR-46K, we delete all holes and only keep one single simple polygon with the largest area as the geometry for each geographic entity. We

---

[9]https://wiki.openstreetmap.org/wiki/Overpass_API

also simplify the exterior of each simple polygon by using the Douglas-Peucker algorithm such that they all have 300 unique vertices. For those polygons with less than 300 vertices, we do a equal distance interpolation on the exteriors to upsample the vertices to 300. The reason to do so is that same number of vertices make it possible for mini-batching.

8. As for DBSR-cplx46K, we also simplify the polygonal geometries but we keep holes and multipolygons if necessary. We delete holes if their area are less than 2.5% of the total area of their corresponding polygonal geometries. Figure 14d-14f show the statistics (number of holes, multipolygons, vertices) of the simplified geometries. Similarly, we make the simplified polygonal geometries to have 300 vertices, $N_g = 300$, for mini-batching. DBSR-46K and DBSR-cplx46K use the same entity set $\mathcal{E}$ and triple set $\mathcal{T}$ and the only difference between them is the polygonal geometry of each entity.

9. Table A1 in Appendix A.1 shows the number of entities with different place types in DBSR-46K and DBSR-cplx46K. Figure 15 shows the polygonal geometries of these geographic entities in these datasets.

10. We split $\mathcal{T}$ into training/validation/testing dataset by roughly 80:5:15. By following the traditional transductive knowledge graph embedding literature [9, 76], we make sure all entities in $\mathcal{E}$ appear in the training dataset. We construct balanced validation and testing dataset with respect to each spatial relation. Table 5 shows the statistic of the dataset split. In the training dataset, the balance of triples with different spatial relations can not be achieved at the same time with the need to include all entities in the graph. So in training dataset we have far more *dbo:isPartOf* triples than other spatial relations. This phenomenon is caused by the nature of DBpedia. So we need to keep this imbalance.

11. The spatial relation prediction task does not need the absolute position of each geographic entity but focuses on their relative spatial relation. So given a triple (e_{sub}, r_i, e_{obj}), we first compute the shared minimal bounding box of subject polygon g_{sub} and object polygon g_{obj}. Then we use this shared bounding box to normalize both g_{sub} and g_{obj} into $[-1, 1] \times [-1, 1]$ 2D unit space. Later on, VeerCNN and ResNet1D directly encodes the normalized polygon geometries for spatial relation prediction. As for DDSL and NUFT-spec, since NUFT prefers positive coordinate inputs, we translate those polygons into $[0, 2] \times [0, 2]$ space before feeding into the NUFT layer.

7.2 Baselines and Model Variations

We keep the spatial relation prediction module in Equation 16 the same but vary the polygon encoders we use. Two exceptions are the Deterministic and DDSL+LeNet5. Deterministicis based on deterministic (no-learning) spatial computation and DDSL+LeNet5 uses LeNet for spatial relation prediction. All models we used for the spatial relation prediction tasks are listed as below:

1. **Deterministic**: We implement a deterministic baseline based on RCC8 topological operators and cardinal direction operations. First, given a triple (e_{sub}, r_i, e_{obj}), if the geometry of e_{sub} is inside of the geometry of e_{obj},

Table 5: The training/validation/testing split of DBSR-46K/DBSR-cplx46K.

Relation	All	Train	Valid	Test
dbo:isPartOf	27364	26164	300	900
dbp:north	2807	1607	300	900
dbp:east	2797	1597	300	900
dbp:south	2770	1570	300	900
dbp:west	2759	1559	300	900
dbp:northwest	2063	863	300	900
dbp:southeast	2024	824	300	900
dbp:southwest	2000	904	274	822
dbp:northeast	1994	1175	205	614
All	46578	36263	2579	7736
ratio	100%	77.85%	5.54%	16.61%

(a) # Sub-polygons   (b) # Holes   (c) # Vertices

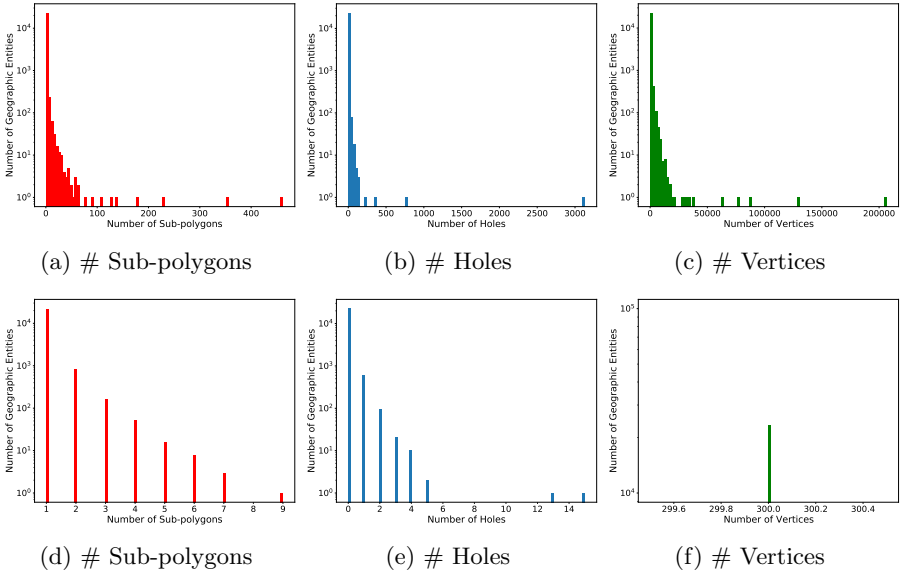(d) # Sub-polygons   (e) # Holes   (f) # Vertices

Fig. 14: A statistic of the complexity of the raw polygonal geometries as well as the simplfied geometries in DBSR-cplx46K from OpenStreetMap. (a)-(c) indicate the statistics on the raw polygonal geometries retrieved from Open-StreetMap while (d)-(f) are the same statistics on DBSR-cplx46K. (a) & (d) A histogram of the number of sub-polygons per geographic entities. (b) & (e) A histogram of the total number of holes per geographic entities. (c) & (f) A histogram of the total number of unique vertices on each polygonal geometry's exterior and interiors per geographic entities.

Deterministic gives *dbo:isPartOf* as the prediction. Otherwise, Deterministic computes the geometric center of the subject and object geometry and compute their cardinal direction. This is how the current SQL or
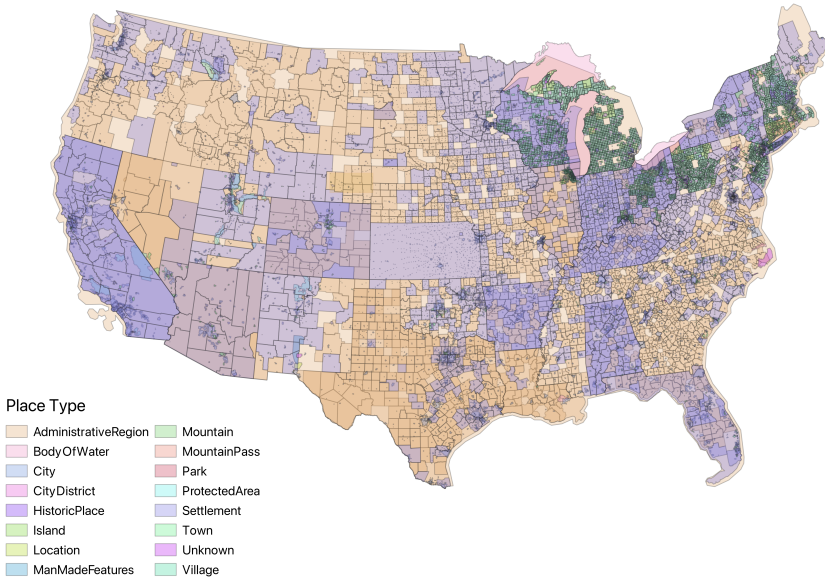
Place Type

AdministrativeRegion	Mountain
BodyOfWater	MountainPass
City	Park
CityDistrict	ProtectedArea
HistoricPlace	Settlement
Island	Town
Location	Unknown
ManMadeFeatures	Village

Fig. 15: A map of geographic entities in DBSR-46K dataset. Some geographic entities' spatial footprints overlays with each other.

GeoSPARQL-based geographic question answering models [36, 77] use to answer spatial relation questions.

2. **VeerCNN**: We use VeerCNN to encode both subject and object polygons and feed them into the spatial relation prediction model (Equation 16).

3. **DDSL+LeNet5** [46]: We utilize the original DDSL implementation[10] and modify it for mini-batch training. DDSL+LeNet5 does not include the spatial relation prediction model. Since the output of NUFT-IFFT of a polygonal geometry is an image, we concatenate the output images of the subject and object geometries in the channel dimension and feed it into the LeNet5, one type of 2D CNN, for spatial relation prediction.

4. **DDSL+MLP**: Unlike DDSL+LeNet5, given a polygonal geometry, DDSL+MLP first flattens the NUFT-IFFT output image into 1D and feeds it into an MLP to encode it into an embedding. The embeddings of the subject and object geometries are concatenated and fed into another MLP as described in Equation 16.

5. **ResNet1D**: We implement our ResNet1D as described in Section 5.1 and utilize it in the spatial relation prediction model.

6. **NUFTspec[fft]+MLP**: This version of NUFTspec is the same as what we described in Section 6.2. It uses a linear grid $\mathcal{W}^{(fft)}$ for NUFT followed by an MLP to encode the polygon into an embedding. The only difference is

---

[10]https://github.com/maxjiang93/DDSL

that it concatenates the embeddings of subject and object geometries and uses them in the spatial relation prediction model (Equation 16).

7. **NUFTspec[gmf]+MLP**: Different from NUFTspec[fft]+MLP, this version of NUFTspec uses the geometric grid $\mathcal{W}^{(gmf)}$.

Among these seven models, the first four models are baseline approaches while ResNet1D, NUFTspec[fft]+MLP, and NUFTspec[gmf]+MLP are polygon encoders we proposed. In order to allow VeerCNN and ResNet1D to handle complex polygonal geometries in the DBSR-cplx46K dataset, we perform the same boundary concatenation operation as described in Section 6.2.

Table 6: Spatial relation prediction accuracy on the DBSR-46K (simple polygon) and DBSR-cplx46K (complex polygon) datasets.

	DBSR-46K			DBSR-cplx46K		
	Train	Valid	Test	Train	Valid	Test
Deterministic	75.42	75.18	73.80	75.17	75.30	73.90
VeerCNN [28]	92.10	77.90	77.59	91.60	77.51	77.08
DDSL+LeNet5 [46, 47]	93.19	80.27	79.22	93.50	80.11	78.68
DDSL+MLP	**95.23**	78.32	78.63	92.89	79.64	79.78
ResNet1D	91.98	78.13	77.79	92.02	78.52	78.24
NUFTspec[fft]+MLP	93.02	80.85	79.20	93.77	79.33	79.12
NUFTspec[gmf]+MLP	93.41	**80.92**	**79.80**	**94.63**	**81.04**	**80.44**

7.3 Main Evaluation Results

We evaluates all models described in Section 7.2 on our DBSR-46K and DBSR-cplx46K dataset. Table 6 shows the overall performance (classification accuracy) of different models on the training, validation, and testing set of DBSR-46K as well as DBSR-cplx46K dataset. The hyperparameter tuning detailed can be seen in Appendix A.2 and the best hyperparameter combinations for different models on two datasets are listed in Table A3. From Table 6, we can see that:

1. NUFTspec[gmf]+MLP achieves the best performance on the validation and testing set of both DBSR-46K and DBSR-cplx46K. It outperforms NUFTspec[fft]+MLP, ResNet1D as well as all four baseline models. This demonstrates the effectiveness and robustness of our NUFTspec model.

2. On both datasets, ResNet1D outperforms Deterministic and VeerCNN, but still falls behind DDSL+LeNet5, DDSL+MLP, NUFTspec[fft]+MLP, and NUFTspec[gmf]+MLP. This might because that both VeerCNN and ResNet1D only encode the polygon boundary information but ignore the polygon topological information (i.e., distinguishing the exterior from the interiors of a polygon) which is very important for spatial relation prediction (e.g., dbo:isPartOf). Our qualitative analysis in Section 7.8 confirms this assumption.

3. Similar to the experiment results on MNIST-cplx70k, NUFT-spec[gmf]+MLP outperforms NUFTspec[fft]+MLP on both DBSR-46K and DBSR-cplx46K. This demonstrates that a good choice of the Fourier frequency map can help improve the model performance. Since DDSL requires an IFFT, the non-integer geometric grid frequency map $\mathcal{W}^{(gmf)}$ is no longer applicable. This also shows the flexibility of NUFTspec in terms of frequency choice.

4. The original DDSL+LeNet5 model [46] and DDSL+MLP share the same NUFT-IFFT layer while using different learnable components – LeNet5 v.s. MLP. We can see that DDSL+LeNet5 can outperform DDSL+MLP on DBSR-46K dataset but underperform DDSL+MLP on DBSR-cplx46K testing dataset. This shows that when predicting spatial relations between complex polygonal geometries, using more complex learnable model (e.g., LeNet5) might not be helpful.

5. NUFTspec[gmf]+MLP outperforms DDSL+MLP by 2.6% and 1.2% on the validation and testing set of DBSR-46K(1.4% and 0.7% for DBSR-cplx46K). Given the fact that they are using the same MLP-based spatial relation module, these performance improvements are mainly attributed to the advantages of NUFTspec: learning polygon representations directly from the NUFT spectral features enables us to have more flexible choices of NUFT frequency map $\mathcal{W}$. This allows us to design $\mathcal{W}$ that can effectively capture irregular polygon representations.

7.4 Ablation Study on ResNet1D

Table 7: Ablation study of ResNet1D on DBSR-46K dataset.

	DBSR-46K		
	Train	Valid	Test
ResNet1D	91.98	78.13	77.79
ResNet1D(zero padding)	90.67	74.49	73.07
ResNet1D(raw pt)	91.34	75.92	75.35

Despite the fact that ResNet1D is similar to VeerCNN in the sense that both of them use 1D CNN layers to model polygons' coordinate sequences, ResNet1D outperforms VeerCNN on both DBSR-46K and DBSR-cplx46K dataset. To understand the superiority of ResNet1D, we do an ablation study which is shown in Table 7. It shows that when we replace the circular padding with zero padding – ResNet1D(zero padding), the performance of ResNet1D drops 3.64% and 4.72% on the validation and testing set. This demonstrates that circular padding is critical for ResNet1D since it preserves the loop origin invariance property. A similar situation happens when we delete the KDelta point encoder component and direct feed the raw point coordinates to ResNet layers – ResNet1D(raw pt). This demonstrates

that KDelta is very helpful for polygon encoding and spatial relation prediction since it uses the spatial affinity features (Equation 1) to enrich the point embedding with its neighborhood information.
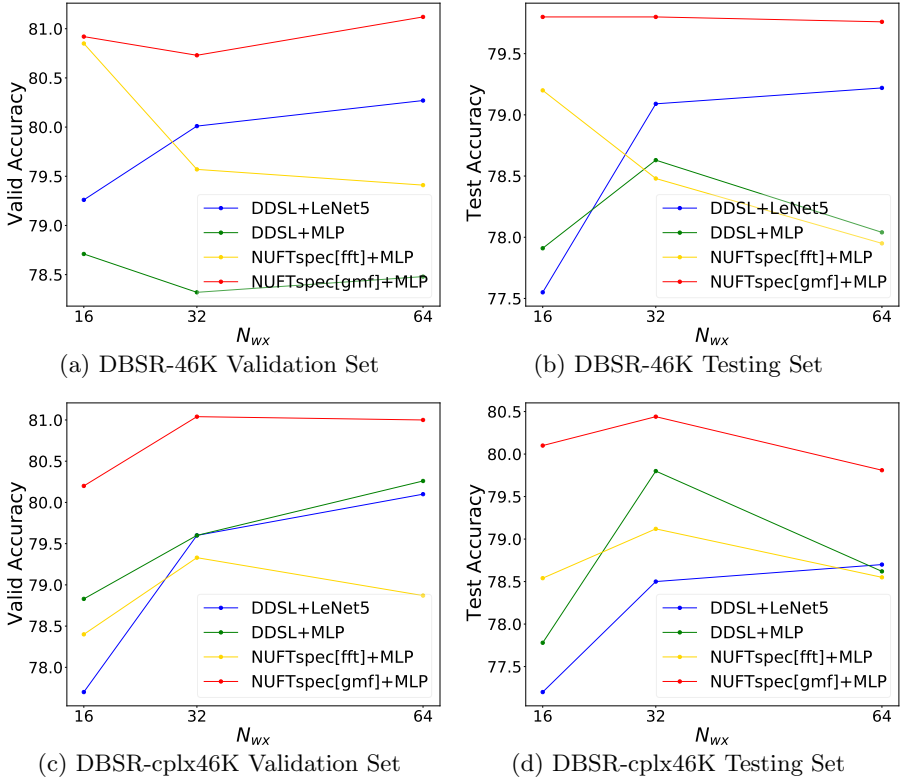
7.5 The Impact of N_{wx}

(a) DBSR-46K Validation Set   (b) DBSR-46K Testing Set

(c) DBSR-cplx46K Validation Set   (d) DBSR-cplx46K Testing Set

Fig. 16: Performance comparisons among four NUFT-based polygon encoder models on the validation and testing sets of DBSR-46K and DBSR-cplx46K with different numbers of NUFT frequencies used in X - N_{wx}. Here, $N_{wx} \propto N_{wy} \propto U \propto \sqrt{N_w}$.

Among the 7 models listed in Section 7.2, there are 4 NUFT-based models. In this section, we test the robustness of these 4 models on DBSR-46K and DBSR-cplx46K when we vary the number of frequencies we use in NUFT. Here, we use N_{wx} to indicate the used NUFT frequency numbers since $N_{wx} \propto N_{wy} \propto U \propto \sqrt{N_w}$. Figure 16 compares the model performance of them with different N_{wx} in the validation and testing dataset of DBSR-46K and DBSR-cplx46K. We can see that NUFTspec[gmf]+MLP achieves the best performance on all

Table 8: Relation classification result on DBSR-46K based on the deterministic RCC8 spatial operator.

Relation	Contains	Intersects	Touches	Disjoint
dbo:isPartOf	20,923	6,309	0	132
dbp:north	4	2,451	0	352
dbp:east	2	2,447	0	348
dbp:south	5	2,405	0	360
dbp:west	3	2,408	0	348
dbp:northwest	4	1,644	0	415
dbp:southeast	2	1,618	0	404
dbp:southwest	4	1,622	0	374
dbp:northeast	3	1,634	0	357

these four datasets with any given N_{wx} and show a clear advantage over the other three models.

7.6 Analysis of The Sliver Polygon Problem

To investigate the difficulty of polygon-based spatial relation prediction task and how the performance can be affected by sliver polygons, we compute the topological relations between the subject and object entity of each triple in DBSR-46K based on a deterministic RCC8-based spatial operator[11]. The statistics is shown in Table 8. We can see almost all triples with cardinal direction relations (the last 8 relations) have subject and object entities that intersect or are disjoint with each other. Interestingly, for *dbo:isPartOf* relation, in 76.5% of the 27,364 triples the subject polygons are inside the corresponding object polygons whereas in 6309 triples (23%+) the subject and object polygons intersect with each other when computing their relations deterministically. Based on manual inspection, most of those 'intersects' cases are caused by the *sliver polygon* problem shown in Figure 4a. This clearly indicates the necessity of polygon encoding models.

To qualitatively show how our proposed polygon encoders mitigate the sliver polygon problem, we show two examples in Figure 17 and 18. In both examples, the subjects indicated by the red polygonal geometries should be *part of* the objects, denoted by the blue polygonal geometries. However, because of the sliver polygons which are illustrated in the zoom-in windows, both Deterministic and DDSL+MLP fail to make the correct predictions while both ResNet1D and NUFTspec[gmf]+MLP can predict the correct relations.
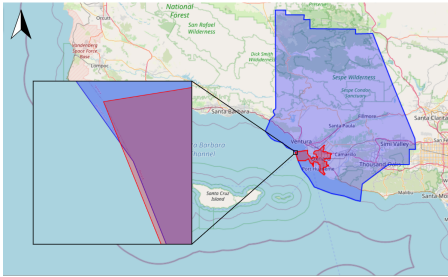
7.7 Analysis of the Scale Problem

To evaluate how well different polygon encoders can handle the scale problem shown in Figure 4b, we evaluate the performance of each models under different area ratio groups. For each triple in the testing set of DBSR-cplx46K, an area ratio R_A is computed as the ratio between the areas of the subject geometry and object geometry. R_A shows the scale different between the subjects

---

[11]https://shapely.readthedocs.io/en/stable/manual.html#binary-predicates

and objects. Based on R_A, DBSR-cplx46K testing set are divided into seven different mutually exclusive area ratio groups. Table 9 shows the performances of different polygon encoders in each area ratio groups. We can see that in all area ratio groups except $R_A \in [1, 1.1)$, NUFTspec[gmf]+MLP can outperform all other polygon encoders. This indicates that NUFTspec is flexible to handle spatial relation prediction task under different area ratio settings.
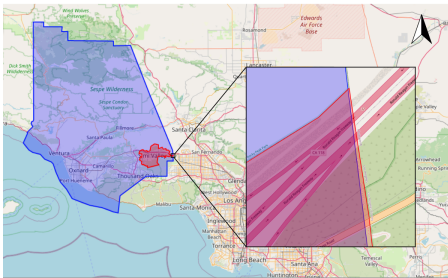
7.8 ResNet1D v.s. NUFTspec

Comparing the evaluation results from both tasks, we notice that ResNet1D can outperform NUFTspec on shape classification task but lose on spatial relation prediction. To understand the reason why ResNet1D fails on the spatial relation prediction task, we explore triples in DBSR-46K testing set on which NUFTspec[fft]+MLP can make the correct predictions while ResNet1D fails. Two representative examples are shown in Figure 19 and 20.

Model	Prediction
Deterministic	dbp:northeast
DDSL+MLP	dbp:northeast
ResNet1D	dbo:isPartOf
NUFTspec[gmf]+MLP	dbo:isPartOf
Ground Truth	dbo:isPartOf

Fig. 17: One example in DBSR-cplx46K testing set to show how the sliver polygons can affect the the prediction results of different polygon encoders. Red geometry (subject): *dbr:Oxnard,_California*; Blue geometry (object): *dbr:Ventura_County,_California*. The table shows the predictions of different polygon encoders on this example.
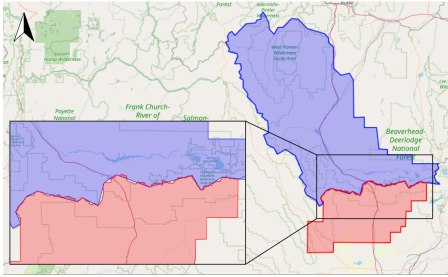

Model	Prediction
Deterministic	dbp:northwest
DDSL+MLP	dbp:northeast
ResNet1D	dbo:isPartOf
NUFTspec[gmf]+MLP	dbo:isPartOf
Ground Truth	dbo:isPartOf

Fig. 18: Another example in DBSR-cplx46K testing set to demonstrate the affect of the sliver polygon problem similar to Figure 17. Red geometry (subject): *dbr:Simi_Valley,_California*; Blue geometry (object): *dbr:Ventura_County,_California*. The table shows the predictions of different polygon encoders on this example.

Table 9: Evaluation results of different polygon encoders on different area ratio groups of DBSR-cplx46K testing set. Given a triple, the area ratio R_A is the ratio between the areas of the subject geometry and object geometry.

Model	Area Ratio R_A						
	$[0, 0.1)$	$[0.1, 0.2)$	$[0.2, 0.3)$	$[0.3, 1)$	$[1, 1.1)$	$[1.1, 1.2)$	$[1.2, \infty)$
# Triples	1027	156	167	3152	675	411	2148
Deterministic	66.60	50.00	62.87	74.62	81.19	74.70	76.49
VeerCNN	86.95	51.92	50.30	75.44	82.67	76.89	76.96
DDSL+LeNet5	87.93	58.33	61.68	76.84	84.44	81.75	77.37
DDSL+MLP	87.24	54.49	63.47	79.09	84.44	80.78	78.68
ResNet1D	87.05	45.51	52.69	76.78	84.59	79.81	78.26
NUFTspec[fft]+MLP	87.15	57.05	61.08	77.35	**85.63**	81.75	78.35
NUFTspec[gmf]+MLP	**87.93**	**59.62**	**64.07**	**79.22**	85.33	**83.94**	**79.24**

Model	Prediction
Deterministic	dbp:north
VeerCNN	dbo:isPartOf
DDSL+LeNet5	dbp:north
DDSL+MLP	dbp:north
ResNet1D	dbo:isPartOf
NUFTspec[fft]+MLP	dbp:north
Ground Truth	dbp:north

Fig. 19: An example in DBSR-46K relation prediction testing set to show the drawback of ResNet1D. Red geometry (subject): *dbr:Clark_County,_Idaho*; Blue geometry (object): *dbr:Beaverhead_County,_Montana*. The table shows the predictions of different polygon encoders for this example.

In both cases, NUFTspec[fft]+MLP and Deterministic can correctly predict the spatial relation. However, two spatial domain polygon encoders – VeerCNN and ResNet1D – predict *dbo:isPartOf*, although the subject geometry is clearly not part of the object geometry. We guess the reason is that both Veer-CNN and ResNet1D only encode vertex information but are not aware of the topology of the polygonal geometries. So it is hard for them to understand the concept of "interior" and "exterior" of polygons. Even there are sliver polygons between subject and object polygons, VeerCNN and ResNet1D still give *dbo:isPartOf* as predictions.

8 Conclusion

In this work, we formally discuss the problem of polygon encoding – a general-purpose representation learning model for polygonal geometries that can be used in various polygon-based tasks including shape classification, spatial relation prediction, building pattern classification, geographic question answering,

Model	Prediction
Deterministic	dbp:west
VeerCNN	dbo:isPartOf
DDSL+LeNet5	dbp:northwest
DDSL+MLP	dbp:northwest
ResNet1D	dbo:isPartOf
NUFTspec[fft]+MLP	dbp:west
Ground Truth	dbp:west

Fig. 20: An example in DBSR-46K relation prediction testing set to show the drawback of ResNet1D. Red geometry (subject): *dbr:Arlington,_Texas*; Blue geometry (object): *dbr:Fort_Worth,_Texas*. The table shows the predictions of different polygon encoders for this example.

and so on. However, polygon encoding is not an easy task given the fact that polygonal geometries can have rather irregular structure, containing holes or multiple sub-polygons which cannot be easily handled by existing neural network architectures. To highlight the uniqueness of this problem, we point out four important polygon encoding properties including loop origin invariance, trivial vertex invariance, part permutation invariance, and topology awareness.

To design a polygon encoder that can handle complex polygonal geometries (including polygons with holes and multipolygons) and at the same time satisfy those four properties, we propose the NUFTspec polygon encoder which utilizes Non-uniform Fourier transformation (NUFT) to transform a polygonal geometry into the spectral space and then learn the polygon embedding from these spectral features. We also propose another 1D CNN-based polygon encoder called ResNet1D as a representative model of spatial domain polygon encoders. ResNet1D utilizes circular padding to achieve loop origin invariance on simple polygons but fail to satisfy other three properties.

To investigate the effectiveness and robustness of these two polygon encoders, we evaluate them and multiple existing baselines on two representative polygon-based tasks – shape classification and spatial relation prediction.

For the shape classification task, we construct a polygon-based shape classification dataset, MNIST-cplx70k, based on the well-known MNIST dataset. Experiment results show that both ResNet1D and NUFTspec can outperform all baselines with statistical significant margins. Moreover, NUFTspec is robust to many shape-invariant geometry modifications including loop origin randomization, vertex upsampling, and part permutation and is more sensitive to topological changes due to the invariance inherented from the NUFT representation. In contrast, models that directly utilize the polygon vertex features such as ResNet1D suffer performance degradations under these geometry modifications.

For the spatial relation prediction task, we construct two real-world datasets - DBSR-46K and DBSR-cplx46K based on DBpedia Knowledge

Graph and OpenStreetMap. Evaluation results show that NUFTspec outperforms all baselines on both datasets and is very robust when we vary the number of sampled frequencies in NUFT.

In addition, compared to other NUFT-based methods such as DDSL, NUFTspec does not use the Inverse Fast Fourier Trasnform (IFFT) so it is more flexible for the choice of the NUFT frequency maps. Both experiments show that by using non-integer frequency maps such as geometric grid $\mathcal{W}^{(gmf)}$, NUFTspec can outperform DDSL on both tasks.

Despite these success, several issues still remain to be solved. First, the training dataset of DBSR-46K and DBSR-cplx46K is very unbalanced since there are more *dbo:isPartOf* triples than triples with other relations. This imbalance causes an overfitting issue for all current models. How to design spatial relation prediction model which is more robust for dataset imbalance is a very interesting research direction. Second, how to effectively utilize NUFT features in the spectral domain is also an interesting future research direction. Moreover, instead of using predefined Fourier frequency maps such as $\mathcal{W}^{(gmf)}$ and $\mathcal{W}^{(fft)}$, can we let the neural network learn the optimal $\mathcal{W}$ based on network backpropagation? Finally, another interesting future direction is to combine NUFTspec and ResNet1D for their different strengths.

We believe that a general-purpose representation learning model for polygonal geometries will be a critical component of so-called *spatially explicit artificial intelligence* [9, 13, 38, 78–81]. It can also serve as an important building block to develop a foundation model for geospatial artificial intelligence [82] in general.

Statements and Declarations

References

[1] Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)

[2] Mai, G., Janowicz, K., Hu, Y., Gao, S., Yan, B., Zhu, R., Cai, L., Lao, N.: A Review of Location Encoding for GeoAI: Methods and Applications. International Journal of Geographical Information Science (2021)

[3] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5115–5124 (2017)

[4] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS (2016)

[5] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)

[6] Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035 (2017)

[7] Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference, pp. 593–607 (2018). Springer

[8] Cai, L., Yan, B., Mai, G., Janowicz, K., Zhu, R.: Transgcn: Coupling transformation assumptions with graph convolutional networks for link prediction. In: Proceedings of the 10th International Conference on Knowledge Capture, pp. 131–138 (2019)

[9] Mai, G., Janowicz, K., Cai, L., Zhu, R., Regalia, B., Yan, B., Shi, M., Lao, N.: SE-KGE: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting. Transactions in GIS (2020). https://doi.org/10.1111/tgis.12629

[10] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point

sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)

[11] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems **31**, 820–830 (2018)

[12] Mac Aodha, O., Cole, E., Perona, P.: Presence-only geographical priors for fine-grained image classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 9596–9606 (2019)

[13] Mai, G., Janowicz, K., Yan, B., Zhu, R., Cai, L., Lao, N.: Multi-scale representation learning for spatial feature distributions using grid cells. In: The Eighth International Conference on Learning Representations (2020). openreview

[14] Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 37–45 (2015)

[15] Lazer, D., Pentland, A.S., Adamic, L., Aral, S., Barabasi, A.L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., *et al.*: Life in the network: the coming age of computational social science. Science (New York, NY) **323**(5915), 721 (2009)

[16] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: The World Wide Web Conference, pp. 417–426 (2019)

[17] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML (2017)

[18] Davidson, E.H., Rast, J.P., Oliveri, P., Ransick, A., Calestani, C., Yuh, C.-H., Minokawa, T., Amore, G., Hinman, V., Arenas-Mena, C., *et al.*: A genomic regulatory network for development. science **295**(5560), 1669–1678 (2002)

[19] Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representations (2019)

[20] Cai, L., Janowicz, K., Mai, G., Yan, B., Zhu, R.: Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. Transactions in GIS **24**(3), 736–755 (2020)

[21] Lin, Y., Mago, N., Gao, Y., Li, Y., Chiang, Y.-Y., Shahabi, C., Ambite, J.L.: Exploiting spatiotemporal patterns for accurate air quality forecasting using deep learning. In: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 359–368 (2018)

[22] Appleby, G., Liu, L., Liu, L.-P.: Kriging convolutional networks. In: Proceedinngs of AAAI 2020 (2020)

[23] Wu, Y., Zhuang, D., Labbe, A., Sun, L.: Inductive graph neural networks for spatiotemporal kriging. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4478–4485 (2021)

[24] Xu, Y., Piao, Z., Gao, S.: Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In: CVPR 2018, pp. 5275–5284 (2018)

[25] Zhang, P., Ouyang, W., Zhang, P., Xue, J., Zheng, N.: Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12085–12094 (2019)

[26] Rao, J., Gao, S., Kang, Y., Huang, Q.: LSTM-TrajGAN: A deep learning approach to trajectory privacy protection. In: GIScience 2020, pp. 12–11217 (2020)

[27] Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: ICLR 2018 (2018)

[28] Veer, R.v., Bloem, P., Folmer, E.: Deep learning for classification tasks on geospatial vector polygons. arXiv preprint arXiv:1806.03857 (2018)

[29] Yan, X., Ai, T., Yang, M., Tong, X.: Graph convolutional autoencoder model for the shape coding and cognition of buildings in maps. International Journal of Geographical Information Science **35**(3), 490–512 (2021)

[30] He, X., Zhang, X., Xin, Q.: Recognition of building group patterns in topographic maps based on graph partitioning and random forest. ISPRS Journal of Photogrammetry and Remote Sensing **136**, 26–40 (2018)

[31] Yan, X., Ai, T., Yang, M., Yin, H.: A graph convolutional neural network for classification of building patterns using spatial vector data. ISPRS journal of photogrammetry and remote sensing **150**, 259–273 (2019)

[32] Bei, W., Guo, M., Huang, Y.: A spatial adaptive algorithm framework for building pattern recognition using graph convolutional networks. Sensors

19(24), 5518 (2019)

[33] Yan, X., Ai, T., Yang, M., Tong, X., Liu, Q.: A graph deep learning approach for urban building grouping. Geocarto International, 1–24 (2020)

[34] Feng, Y., Thiemann, F., Sester, M.: Learning cartographic building generalization with deep convolutional neural networks. ISPRS International Journal of Geo-Information **8**(6), 258 (2019)

[35] Zelle, J.M., Mooney, R.J.: Learning to parse database queries using inductive logic programming. In: Proceedings of the National Conference on Artificial Intelligence, pp. 1050–1055 (1996)

[36] Punjani, D., Singh, K., Both, A., Koubarakis, M., Angelidis, I., Bereta, K., Beris, T., Bilidas, D., Ioannidis, T., Karalis, N., *et al.*: Template-based question answering over linked geospatial data. In: Proceedings of the 12th Workshop on Geographic Information Retrieval, pp. 1–10 (2018)

[37] Scheider, S., Nyamsuren, E., Kruiger, H., Xu, H.: Geo-analytical question-answering with gis. International Journal of Digital Earth **14**(1), 1–14 (2021)

[38] Mai, G., Yan, B., Janowicz, K., Zhu, R.: Relaxing unanswerable geographic questions using a spatially explicit knowledge graph embedding model. In: AGILE, pp. 21–39 (2019). Springer

[39] Mai, G., Janowicz, K., Zhu, R., Cai, L., Lao, N.: Geographic question answering: Challenges, uniqueness, classification, and future directions. AGILE: GIScience Series **2**, 1–21 (2021)

[40] Sun, X., Christoudias, C.M., Fua, P.: Free-shape polygonal object localization. In: European Conference on Computer Vision, pp. 317–332 (2014). Springer

[41] Castrejon, L., Kundu, K., Urtasun, R., Fidler, S.: Annotating object instances with a Polygon-RNN. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5230–5238 (2017)

[42] Acuna, D., Ling, H., Kar, A., Fidler, S.: Efficient interactive annotation of segmentation datasets with Polygon-RNN++. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 859–868 (2018)

[43] Bai, X., Liu, W., Tu, Z.: Integrating contour and skeleton for shape classification. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 360–367 (2009). IEEE

[44] Wang, X., Feng, B., Bai, X., Liu, W., Latecki, L.J.: Bag of contour fragments for robust shape classification. Pattern Recognition **47**(6), 2116–2125 (2014)

[45] Regalia, B., Janowicz, K., McKenzie, G.: Computing and querying strict, approximate, and metrically refined topological relations in linked geographic data. Transactions in GIS **23**(3), 601–619 (2019)

[46] Jiang, C., Lansigan, D., Marcus, P., Nießner, M., *et al.*: DDSL: Deep differentiable simplex layer for learning geometric signals. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8769–8778 (2019)

[47] Jiang, C.M., Wang, D., Huang, J., Marcus, P., Niessner, M.: Convolutional neural networks on non-uniform geometrical signals using euclidean spectral transformation. In: International Conference on Learning Representations (2019)

[48] Kurnianggoro, L., Jo, K.-H., *et al.*: A survey of 2d shape representation: Methods, evaluations, and future research directions. Neurocomputing **300**, 1–16 (2018)

[49] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[50] Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: 3rd International Conference on Knowledge Representation and Reasoning, pp. 165–176 (1992)

[51] Egenhofer, M.J., Franzosa, R.D.: Point-set topological spatial relations. International Journal of Geographical Information System **5**(2), 161–174 (1991)

[52] Zhang, Z., Fidler, S., Waggoner, J., Cao, Y., Dickinson, S., Siskind, J.M., Wang, S.: Superedge grouping for object localization by combining appearance and shape information. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3266–3273 (2012). IEEE

[53] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[54] Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. In: ICLR 2016 (2016)

[55] Liang, J., Homayounfar, N., Ma, W.-C., Xiong, Y., Hu, R., Urtasun, R.: Polytransform: Deep polygon transformer for instance segmentation.

In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9131–9140 (2020)

[56] Atabay, H.A.: Binary shape classification using convolutional neural networks. IIOAB J **7**(5), 332–336 (2016)

[57] Atabay, H.A.: A convolutional neural network with a new architecture applied on leaf classification. IIOAB J **7**(5), 226–331 (2016)

[58] Hofer, C., Kwitt, R., Niethammer, M., Uhl, A.: Deep learning with topological signatures. In: NIPS (2017)

[59] Baker, N., Lu, H., Erlikhman, G., Kellman, P.J.: Deep convolutional networks do not classify based on global object shape. PLoS computational biology **14**(12), 1006613 (2018)

[60] Latecki, L.J., Lakamper, R., Eckhardt, T.: Shape descriptors for non-rigid shapes with a single closed contour. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662), vol. 1, pp. 424–429 (2000). IEEE

[61] Söderkvist, O.: Computer vision classification of leaves from swedish trees. PhD thesis (2001)

[62] Leibe, B., Schiele, B.: Analyzing appearance and contour based methods for object categorization. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., vol. 2, p. 409 (2003). IEEE

[63] Mallah, C., Cope, J., Orwell, J., et al.: Plant leaf classification using probabilistic integration of shape, texture and margin features. Signal Processing, Pattern Recognition and Applications **5**(1) (2013)

[64] Sebastian, T.B., Kimia, B.B.: Curves vs. skeletons in object recognition. Signal processing **85**(2), 247–263 (2005)

[65] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 234–241 (2015). Springer

[66] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[67] Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern

Recognition, pp. 2403–2412 (2018)

[68] Rippel, O., Snoek, J., Adams, R.P.: Spectral representations for convolutional neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2, pp. 2449–2457 (2015)

[69] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision, pp. 405–421 (2020). Springer

[70] Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint arXiv:2006.10739 (2020)

[71] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

[72] Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)

[73] Ha, D., Eck, D.: A neural representation of sketch drawings. In: International Conference on Learning Representations (2018)

[74] Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., Guibas, L.J.: Vector neurons: A general framework for so (3)-equivariant networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 12200–12209 (2021)

[75] Esteves, C., Allen-Blanchette, C., Makadia, A., Daniilidis, K.: Learning so (3) equivariant representations with spherical cnns. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 52–68 (2018)

[76] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Neural Information Processing Systems (NIPS), pp. 1–9 (2013)

[77] Chen, W.: Parameterized spatial sql translation for geographic question answering. In: 2014 IEEE International Conference on Semantic Computing, pp. 23–27 (2014). IEEE

[78] Yan, B., Janowicz, K., Mai, G., Gao, S.: From itdl to place2vec: Reasoning about place type similarity and relatedness by learning embeddings from

augmented spatial contexts. In: Proceedings of the 25th ACM SIGSPA-
TIAL International Conference on Advances in Geographic Information
Systems, pp. 1–10 (2017)

[79] Yan, B., Janowicz, K., Mai, G., Zhu, R.: A spatially explicit reinforce-
ment learning model for geographic knowledge graph summarization.
Transactions in GIS **23**(3), 620–640 (2019)

[80] Janowicz, K., Gao, S., McKenzie, G., Hu, Y., Bhaduri, B.: GeoAI: spa-
tially explicit artificial intelligence techniques for geographic knowledge
discovery and beyond. Taylor & Francis (2020)

[81] Li, W., Hsu, C.-Y., Hu, M.: Tobler's first law in geoai: A spatially explicit
deep learning model for terrain feature detection under weak supervision.
Annals of the American Association of Geographers **111**(7), 1887–1905
(2021)

[82] Mai, G.M., Cundy, C., Choi, K., Hu, Y., Lao, N., Ermon, S.: Towards
a foundation model for geospatial artificial intelligence. In: Proceedings
of the 30th SIGSPATIAL International Conference on Advances in Geo-
graphic Information Systems (2022). https://doi.org/10.1145/3557915.
3561043

Appendix A   Appendix

A.1   The Type Statistic of Polygonal Geometries in DBSR-cplx46K

Table A1: The place type statistic of geographic entities in DBSR-46K and DBSR-cplx46K dataset.

Place Type	Entity Count
City	7887
Town	6668
Settlement	3688
Village	2502
AdministrativeRegion	1420
CityDistrict	980
Unknown	57
ProtectedArea	20
BodyOfWater	12
ManMadeFeatures	12
Park	9
HistoricPlace	3
Island	2
Location	2
MountainPass	1
Mountain	1

A.2   Model Hyperparameter Tuning

We use grid search for hyperparameter tuning. For all polygon encoders on both tasks, we tune the learning rate lr over $\{0.02, 0.01, 0.005, 0.002, 0.001\}$, the polygon embedding dimension over $d \in \{256, 512, 1024\}$. As for all DDSL and NUFTspec-based models, we tune the frequency number $N_{wx} = \{16, 20, 24, 28, 32, 36, 40, 44\}$ for the shape classification task while $N_{wx} = \{16, 32, 64\}$ for the spatial relation prediction task. As for NUFTspec[gmf]-based models, we tune the $w_{min} = \{0.2, 0.4, 0.5, 0.8, 1.0\}$ and we tune w_{max} around $N_{wx}/2$. For all PCA models, we vary K_{PCA} such that the top K_{PCA} PCA components can account for different data variance $\sum_{PCA} = \{80\%, 85\%, 90\%, 95\%\}$. As for ResNet1D, we tune the KDelta point encoder's neighbor size $2t \in \{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$ and tune the number of ResNet1D$_{cp}$ - $\mathcal{K} \in \{1, 2, 3\}$. For DDSL+LeNet5, we tune the hidden dimension of LeNet5 over $\{128, 256, 512, 1024\}$. As to NUFTspec-based models, DDSL+MLP, and DDSL+PCA+MLP, we tune the number of hidden layers h and the number of hidden dimension o in $MLP_F(\cdot)$ over $h = \{1, 2, 3\}$, $o = \{512, 1024\}$. We also try different NUFT spectral feature normalization method $\Psi(\cdot)$ such as no normalization, L2 normalization, and batch normalization. We find out no normalization usually leads to the best performance on all three datasets.

Table A2: The best hyperparameter combinations for each model on MNIST-cplx70k dataset.

Model	lr	d	N_{wx}	w_{min}	w_{max}	$\sum_{PCA}$	K_{PCA}	$\mathcal{K}$	$2t$
VeerCNN	0.01	1024	-	-	-	-	-	-	-
ResNet1D	0.01	512	-	-	-	-	-	3	12
DDSL+MLP	0.001	512	24	-	-	-	-	-	-
DDSL+PCA+MLP	0.0005	512	24	-	-	90%	39	-	-
NUFTspec[fft]+MLP	0.0005	512	24	-	-	-	-	-	-
NUFTspec[fft]+PCA+MLP	0.0005	512	24	-	-	80%	42	-	-
NUFTspec[gmf]+MLP	0.0005	512	24	0.5	12	-	-	-	-
NUFTspec[gmf]+PCA+MLP	0.0005	512	24	0.5	12	95%	46	-	-

Table A3: The best hyperparameter combinations for each model on DBSR-46K and DBSR-cplx46K dataset.

Dataset	Model	lr	d	N_{wx}	w_{min}	w_{max}	$\mathcal{K}$	$2t$
DBSR-46K	VeerCNN	0.01	1024	-	-	-	-	-
	DDSL+LeNet5	0.01	512	32	-	-	-	-
	DDSL+MLP	0.001	512	32	-	-	-	-
	ResNet1D	0.01	512	-	-	-	1	4
	NUFTspec[fft]+MLP	0.01	512	16	-	-	-	-
	NUFTspec[gmf]+MLP	0.002	512	32	0.8	16	-	-
DBSR-cplx46K	VeerCNN	0.02	512	-	-	-	-	-
	DDSL+LeNet5	0.01	512	32	-	-	-	-
	DDSL+MLP	0.001	512	32	-	-	-	-
	ResNet1D	0.02	512	-	-	-	1	4
	NUFTspec[fft]+MLP	0.01	512	32	-	-	-	-
	NUFTspec[gmf]+MLP	0.002	512	32	0.5	20	-	-

The best hyperparameter combinations for all models on MNIST-cplx70k are shown in Table A2. As for DBSR-46K and DBSR-cplx46K, each model's best hyperparameter combinations are shown in Table A3.