Towards general-purpose representation learning of polygonal geometries

Gengchen Mai1,2,3,4, Chiyu Jiang5, Weiwei Sun6, Rui Zhu3,4,7, Yao Xuan8, Ling Cai3,4, Krzysztof Janowicz3,4,9, Stefano Ermon2,10, Ni Lao11

Received: 23 December 2021 / Revised: 19 September 2022 / Accepted: 29 September 2022 / Published online: 22 October 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

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 (Fig. 1a), Question ‘How far it is from Canada to US’ cannot be answered correctly even by state-of-the-art QA system1. As for the BPC task (Fig. 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 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 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 multipolygons) in an embedding space. Furthermore, we identify a few desirable properties for polygon encoders to guarantee their model generalizability: loop 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 subsequently be 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

\[\text{(a) Geographic Question Answering} \quad \text{(b) Building Pattern Classification}\]

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]

\(^2\) A simple polygon is a polygon that does not intersect itself and has no holes.
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 b 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 degradations. For the spatial relation prediction task, NUFTspec outperforms ResNet1D, as well as multiple deterministic 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 Sections 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 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 Fig. 3a and b 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 multipolygon**: Polygon $g_C$ and $g_D$ shown in Fig. 3c and d 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
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 Fig. 4a with real-world examples from OpenStreetMap. In the following, we discuss how these problems pose challenges to the above mentioned alternative approaches:

1. **Sliver polygon problem**: As shown in zoom-in window 1, 2, and 3 in Fig. 4a, those three tiny polygons yielded from the geometry difference between the red and blue polygon are called sliver polygons. 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 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.

2. **Scale problem**: Two polygons might have very different sizes and thus need to use different map scales for visualization (Fig. 4b). `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.

3. **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”.

---

3 In GIScience, sliver polygon is a technical term referring to the small unwanted polygons resulting from polygon intersection or difference.
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 Fig. 4c, \textit{dbr:Berkeley, California} (the blue polygon) sits in the north of \textit{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 $G = \{g_i\}$ be a set of polygonal geometries in a 2D Euclidean space $\mathbb{R}^2$ where $G$ is a union of a polygon set $P = \{p_i\}$ and a multipolygon set $Q = \{q_i\}$, a.k.a $G = P \cup Q$ and $P \cap Q = \emptyset$. We have $g_i \in P \vee g_i \in Q$.

**Definition 1** (Polygon) Each polygon $p_i$ can be represented as a tuple $(B_i, h_i = \{H_{i,j}\})$ where $B_i \in \mathbb{R}^{N_{b_i} \times 2}$ indicates a point coordinate matrix for the exterior of $p_i$ defined in a counterclockwise direction. $h_i = \{H_{i,j}\}$ is a set of holes for $p_i$ where each hole $H_{i,j} \in \mathbb{R}^{N_{h_{i,j}} \times 2}$ 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 $B_i$ are not the same and $B_i$ does not intersect with itself. Similar logic applies to each hole $H_{i,j}$ and $N_{h_{i,j}}$ is the number of unique points in the $j$th hole of $p_i$.

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

**Definition 3** (Polygonal Geometry) A polygonal geometry $g_i \in P \cup Q$ can be either a polygon or a multipolygon. $\{e_{in}\}_{n=1}^{N_{e_i}}$ 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. $N_{e_i}$ 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., $g_i = (B_i, h_i = \emptyset)$, 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_G,θ(g_i) : G^* → R^d which is parameterized by θ and maps any polygonal geometry g_i ∈ G^* in R^2 to a vector representation of d dimension⁴. Here G^* indicates the set of all possible polygonal geometries in R^2 and G ⊆ G^*.

⁴ We use Enc(g_i) to represent Enc_G,θ(g_i) in the following
Figure 5a illustrates a multipolygon \( q = \{ p_0, p_1 \} \) where \( p_0 = (B_0, h_0 = \{ H_0 \}) \) has one hole and \( p_1 = (B_1, h_1 = 0) \) has no hole. Our objective is to develop general-purpose polygon encoders, so model generalizability is the key consideration. To test the generalizability 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 \( p_0' = (B_0, 0) \) as a simple polygon made up from the exterior of \( p_0, B_0 \) can be written as \( B_0 = [x'_0, x'_1, \ldots, x'_{k-1}, x'_k, x'_0] \in \mathbb{R}^{6 \times k} \). Let \( B_0 = L_0B_0 \) as another representation of \( p_0' \) exterior where \( L_0 \in \mathbb{R}^{6 \times k} \) is a loop matrix that shifts the order of \( B_0 \) by \( s \). For example, \( B_0^{(s)} = L_0B_0 = [x'_0, x'_1, \ldots, x'_k, x'_0, x'_1, \ldots, x'_{k-1}] \). Conceptually, we have \( p_0' = (B_0, 0) = p_0^{(s)} = (L_0B_0, 0) \forall s \in \{1, 2, ..., 6\} \) where \( s \) 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 - \( A', B', C', D', E', F' \) - (Fig. 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 \( Enc(p_0') = Enc(p_0) \).

3. **Part permutation invariance (ParP):** The encoding result of a multipolygon \( q_i \) should not be invariant when permuting the feed-in order of its parts. For instance, the encoding result of \( Enc(q) \) (Fig. 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 Fig. 5a and c, \( q = \{ p_0, p_1 \} \) 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 polygon 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 \( p_0 \) (Fig. 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

 Springer
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 an 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 polygonal 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 Fig. 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. “#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 |

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.

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 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 Fig. 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 NUFTspec. Figure 2a and b 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 $g = (B, \emptyset)$ where $B = [x_0^T, x_1^T, \ldots, x_{g-1}^T] \in \mathbb{R}^{N \times 2}$, ResNet1D treats the exterior $B$ of $g$ as a 1D coordinate sequence. Before feeding $B$ into the 1D ResNet layer,
we first compute a point embedding \( l_m \in \mathbb{R}^{4t+2} \) for the \( m \)th point \( x_m \) by concatenating \( x_m \) with its spatial affinity with its neighboring \( 2t \) points:

\[
l_m = [x_m; x_{m-t} - x_m; \ldots; x_{m-1} - x_m; x_{m+1} - x_m; \ldots; x_{m+t} - x_m]
\]

(1)

We call Eq. 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 \( B \) represents a circle. The resulting embedding matrix \( \mathbf{L} = [l_0^T; l_1^T; \ldots; l_m^T; \ldots; l_N_g^T] \in \mathbb{R}^{N_g \times (4t+2)} \) 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 Eq. 2.

\[
Enc_{ResNet1D}(g) = \left[ \mathbf{L} \rightarrow \text{CNN1D}^{d,1,1}_{3 \times 1} \rightarrow \text{BN1D} \rightarrow \text{ReLU} \rightarrow \text{MP1D}^{2,0}_{2 \times 1} \rightarrow (\text{ResNet1D}_{cp})^{K}_{k=1} \rightarrow \text{GMP1D} \rightarrow \text{DP} \rightarrow \mathbf{p} \right]
\]

(2)

CNN1D\(^{d,1,1}_{3 \times 1}\) indicates a 1D CNN layer with 1 stride, 1 padding (circular padding) and \( d \) number of \( 3 \times 1 \) kernel (1D kernel). BN1D and ReLU indicate 1D batch normalization layer and a ReLU activation layer. MP1D\(^{2,0}_{2 \times 1}\) indicates a 1D Max Pooling layer with 2 stride, 0 padding, and kernel size 2. \((\text{ResNet1D}_{cp})^{K}_{k=1}\) indicates \( K \) 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 Fig. 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 \( S^j = \{S_n^j\}_{n=1}^{N_g} = (V, E, D) \) (here \( j = 2 \)) by adding one auxiliary node (the origin point \( x_O \)). A 2-simplex is simply a triangle in the 2D space. The polygon-to-\( j \)-simplex-mesh operation is illustrated in Fig. 6 and summarized as below:

1. As shown in Fig. 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 \( x_O = [0,0] \). 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 separately 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 \( x_O = [0, 0] \) as an auxiliary node which helps us to convert Polyonal Geometry \( g \) into \( j \)-simplex mesh \( S^{(j)} \) (See Fig. 6b).

3. We loop through all edges in \( \{e_n\}_{n=1}^{N_e} \) (See Definition 3) of \( g \). For the \( n \)th edge \( e_n \), we connect its two vertices to the auxiliary node \( x_O \) (origin) to construct a 2-simplex (triangle) \( S_n^{(j)} \). For instance, for Edge \( e_{AB} \), we construct a Triangle \( S_{ABO}^{(j)} = \triangle ABO \) (See Fig. 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 \( S^{(j)} = \{S_n^{(j)}\}_{n=1}^{N_g} \) (See Fig. 6d).

A 2-simplex mesh \( S^{(j)} = \{S_n^{(j)}\}_{n=1}^{N_g} = (V, E, D) \) is represented as three matrices - vertex matrix \( V \in \mathbb{R}^{(N_e+1) \times 2} \) for polygon/simplex vertex coordinates, edge matrix \( E \in \mathbb{N}^{N_e \times 3}_0 \) for 2-simplex connectivity, and density matrix \( D \in \mathbb{R}^{N_e \times d_d} \) for per-simplex density. \( V \) is a float matrix that contains the coordinates of all vertices of \( g \) as well as the origin \( x_O = [0, 0] \) (the last row). The edge matrix \( E \) contains non-negative integers. The \( n \)th row of \( E \) corresponds to the \( n \)th simplex \( S_n^{(j)} \) in \( S^{(j)} \) whose values indicate the indices of vertices of \( S_n^{(j)} \) in \( V \). For the density matrix \( D \in \mathbb{R}^{N_e \times d_d} \), the \( n \)th row indicates a \( d_d \)-dimensional density features associated with the \( n \)th edge/simplex (e.g., edge color, density, and etc.). In our case, we assume each edge \( e_n \) has a constant 1D density feature \(- [1] \), i.e., \( D = \mathbf{1} \in \mathbb{R}^{N_e \times 1} \), 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

![Fig. 6](image-url)
use the right-hand rule\(^5\) to infer the correct orientation of each edge \(e_n\). So the orientation of the boundary of the corresponding simplex \(S_n^{(i)}\) can be determined (e.g., counterclockwise or clockwise). Based on that, we can compute the signed content (area) of each simplex \(S_n^{(i)}\) 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 \(S^{(2)} = (V, E, D)\), we use the example of Multipolygon \(g\) shown in Fig. 2b. It can be converted into Simplex \(S^{(2)} = (V, E, D)\) where \(V = \{x_A^T, x_B^T, x_C^T, x_D^T, x_E^T, x_F^T, x_H^T, x_I^T, x_J^T, x_K^T, x_L^T, x_O^T\} \in \mathbb{R}^{13 \times 2}\), \(E = \{(0, 1, 12), (1, 2, 12), (2, 3, 12), (3, 4, 12),(4, 5, 12),(5, 0, 12),(6, 7, 12),(7, 8, 12), \{8, 6, 12\}, \{9, 10, 12\}, \{10, 11, 12\}, \{11, 9, 12\}\} \in \mathbb{N}_0^{12 \times 2}\), and \(D\) is a 12 \(\times\) 1 constant one matrix.

### 5.2.2 Non-uniform fourier transforms

Next, we perform NUFT on this \(j\)-simplex mesh \(S^{(j)} = \{S_n^{(j)}\}_{n=1}^{N_j}\) (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 \(n\)th \(j\)-simplex \(S_n^{(j)}\) in a \(j\)-simplex mesh \(S^{(j)} = \{S_n^{(j)}\}_{n=1}^{N_j} = (V, E, D)\), we define a density function \(f_n^{(j)}(x)\) on \(S_n^{(j)}\) as Eq. 3 in which \(\rho_n\) is the signal density defined on \(S_n^{(j)}\).

\[
f_n^{(j)}(x) = \begin{cases} 
\rho_n, & x \in S_n^{(j)} \\
0, & x \not\in S_n^{(j)} 
\end{cases}
\]  

(3)

**Definition 7** (Piecewise-Constant Function over a simplex mesh) The Piecewise-Constant Function (PCF) over a simplex mesh \(S^{(j)}\) is the superposition of the density function \(f_n^{(j)}(x)\) for each simplex \(S_n^{(j)}\).

\[
f_S^{(j)}(x) = \sum_{n=1}^{N_j} f_n^{(j)}(x)
\]  

(4)

**Definition 8** (NUFT of PCF \(f_S^{(j)}(x)\)) The NUFT of PCF \(f_S^{(j)}(x)\) over a \(j\)-simplex mesh \(S^{(j)} = \{S_n^{(j)}\}_{n=1}^{N_j} = (V, E, D)\) on a set of \(N_w\) Fourier base frequencies \(\mathcal{W} = \{w_k\}_{k=1}^{N_w}\) is a sequence of \(N_w\) complex numbers:

\[
F_S^{(j)}(x) = [F_{S,1}^{(j)}(x), F_{S,2}^{(j)}(x), ..., F_{S,k}^{(j)}(x), ..., F_{S,N_w}^{(j)}(x)]
\]  

(5)

where the NUFT of \(f_S^{(j)}(x)\) on each base frequency \(w_k \in \mathbb{R}^2\) can be written as the weighted sum of the Fourier transform on each \(j\)-simplex \(S_n^{(j)}\).

\(^5\) https://mapster.me/right-hand-rule-geojson-fixer/
\[ F^{(j)}_{S,k}(\mathbf{x}) = \int \cdots \int_{-\infty}^{\infty} f^{(j)}_{S}(\mathbf{x}) e^{-i(w_j \cdot \mathbf{x})} d\mathbf{x} \]
\[ = \sum_{n=1}^{N_{x}} \rho_n \int \cdots \int_{-\infty}^{\infty} S^{(j)}_{n} e^{-i(w_j \cdot \mathbf{x})} d\mathbf{x} \]
\[ = \sum_{n=1}^{N_{x}} \rho_n F^{(j)}_{n,k}(\mathbf{x}) \]  

(6)

\[ F^{(j)}_{n,k}(\mathbf{x}) \] is the NUFT on the \(n\)th simplex \(S^{(j)}_{n}\) with base frequency \(w_k\) which can be defined as follow:

\[ F^{(j)}_{n,k}(\mathbf{x}) = i^n \mu(S^{(j)}_{n}) \gamma^{(j)}_{n} \sum_{i=1}^{j+1} \frac{e^{-i(w_i \cdot \mathbf{x})}}{\prod_{i=1}^{j+1}(e^{-i(w_i \cdot \mathbf{x})} - e^{-i(w_i \cdot \mathbf{x})})}, \]  

(7)

where \(\mu(\cdot) : S^{(j)} \rightarrow \{-1, 1\}\) is a sign function which determines the sign of the content (area) of \(S^{(j)}_{n}\), \(\gamma^{(j)}_{n}\) is the content distortion factor, which is the ratio of the unsigned content of \(S^{(j)}_{n} - C^{(j)}_{n}\) - and the content of the unit orthogonal \(j\)-simplex - \(\gamma^{(j)}_{n} = 1/j!\). \(\mu(S^{(j)}_{n}) \gamma^{(j)}_{n}\) is the signed content distortion factor of \(S^{(j)}_{n}\) which can be computed based on the determinant of the Jacobian matrix\(^6\) \(J\) of simplex \(S^{(j)}_{n}\). Let \(x_{n,1}, x_{n,2}, x_O\) the three vertices of \(S^{(j)}_{n}\), we have:

\[ \mu(S^{(j)}_{n}) \gamma^{(j)}_{n} = \frac{\mu(S^{(j)}_{n}) C^{(j)}_{n}}{C^{(j)}_{n}} = \frac{1/j! \det(J_{n})}{1/j!} \]
\[ = \det([x_{n,1} - x_O, x_{n,2} - x_O]) = \det([x_{n,1}, x_{n,2}]) \]  

(8)

The proof of Eqs. 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 \(\mathcal{W} = \{w_k\}_{k=1}^{N_{w}}\) where \(w_k \in \mathbb{R}^2\). We can either use a series of equally spaced frequencies along each dimension, denoted as \(\text{linear grid frequency map} \mathcal{W}^{(fg)}\), 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 \(\text{geometric grid frequency map} \mathcal{W}^{(gf)}\), 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 \(\mathcal{W}^{(fg)}\) is just simply the normal \textit{integer} Fourier frequency bases used by the Fast Fourier Transform (FFT). It is defined as a Cartesian product between the linear frequency sets along the X and Y axis – \(W_x^{(fg)}\) and \(W_y^{(fg)}\) – which contain \(N_{w_x}\) and \(N_{w_y}\) integer values respectively:

\[ \mathcal{W}^{(fg)} = W_x^{(fg)} \times W_y^{(fg)} \subseteq \mathbb{N}_{N_{w_x}} \times \mathbb{N}_{N_{w_y}} \]  

(9)

Here, \(N_w = N_{w_x} N_{w_y}\). \(W_x^{(fg)}\) and \(W_y^{(fg)}\) are defined as Eq. 10.

---

\[^6\] https://en.m.wikipedia.org/wiki/Simplex#Volume
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 $W_x^{(ff)}$ has roughly half of $W_y^{(ff)}$’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 $W_x^{(ft)}$. 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 $W_x^{(gmf)}$ is defined as a Cartesian product between the selected geometric series based frequency sets along the X and Y axis – $W_x^{(gmf)} \times W_y^{(gmf)}$:

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

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

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

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

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

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

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

A visualization of $W_x^{(ff)}$ and $W_x^{(gmf)}$ can be seen in Fig. 7a and b. 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 \( N_w = N_w x N_w y \) complex valued NUFT features each of which corresponds to one frequency item \( w_k \in \mathbb{R}^{2} \) in \( W^{\text{fft}} \) or \( W^{\text{gmf}} \). We compute the data variance\(^7\) of each \( w_k \) across all 60K training polygons in MNIST-cplx70k and visualize them in Fig. 7a and b. We can see that for linear grid frequency map \( W^{\text{fft}} \), 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 \( W^{\text{gmf}} \), more frequencies have higher data variance which is easier for the following MLP to learn from.

### 5.2.3 NUFT-based polygon encoding

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

\[
Enc_{\text{NUFTspec}}(g) = \text{MLP}_F \left( \Psi \left( F_{S}^{(j)}(x) \right) \right)
\]

(14)

Here, \( \text{MLP}_F(\cdot) \) 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 \( F_{S}^{(j)}(x) \) 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) \( \text{MLP}_{\text{shp}}(\cdot) \) and a softmax layer to predict its shape class \( y \):

\[
P(r \mid g) = \text{softmax}(\text{MLP}_{\text{shp}}(Enc(g))).
\]

(15)

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

### 5.4 Spatial relation prediction model

Given a polygon pair \((g_{\text{sub}}, g_{\text{obj}})\), 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

\[
P(r \mid g_{\text{sub}}, g_{\text{obj}}) = \text{softmax}(\text{MLP}_{\text{rel}}([Enc(g_{\text{sub}}); Enc(g_{\text{obj}})])),
\]

(16)

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

---

\(^7\) We only compute the data variance for the real value part for each NUFT complex feature.
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  |

5.5 Model property comparison

Theorem 1 ResNet1D is loop origin invariant for simple polygons \( g = (B, \emptyset) \).

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 Sections 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.

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

5.6.2 Proofs of Theorem 2

Proof of Theorem 2 (1) - Loop Origin Invariance

Given a simple polygon \( p = (B, \emptyset) \), we convert it to a 2-simplex mesh \( S^{(j)}_p = (U^{(j)}_0, U^{(j)}_1, U^{(j)}_2) \) similar to what is shown in Fig. 8a. Since \( \{S_n^{(j)}\}_{n=1}^{N_{2}} \) is an unordered set of 2-simplexes, for any loop matrix \( L_s \), Polygon \( p^{(i)} = (L_s B, \emptyset) \) will have exactly the same 2-simplex mesh as \( p - S^{(j)} \). 2-simplex mesh \( S^{(j)} \) is the input of our NUFTspec. So the output polygon embedding \( \text{Enc}_{\text{NUFTspec}}(p) \) is invariant to any loop.
transformation $L_s$ on Polygon $p$’s exterior $B$. 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 $g$.

**Proof of Theorem 2 (2) - Trivial Vertex Invariance** We use Polygon $p_2$ and $p'_2$ shown in Fig. 8a and b 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: $\mathcal{S}^{(j)}_2 = \{S^{(j)}_{\text{GIO}}, S^{(j)}_{\text{HPO}}, S^{(j)}_{\text{HGO}}\}$ and $\mathcal{S}'^{(j)}_2 = \{S'^{(j)}_{\text{GIO}}, S'^{(j)}_{\text{HPO}}, S'^{(j)}_{\text{HGO}}\}$. The Piecewise-Constant Function (PCF) $f^{(j)}_n(x) = \sum_{n=1}^{N_g} f^{(j)}_n(x)$ defined on the simplex mesh $\mathcal{S}^{(j)}_2$ is essentially a summation over the individual density function $f^{(j)}_n(x)$ defined on each simplex of $\mathcal{S}^{(j)}_2$. 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 $\mathcal{S}^{(j)}_2 = \{S^{(j)}_n\}_{n=1}^{N_g}$ (similar to Fig. 5d) is an unordered set of signed 2-simplices/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 $p = (B, h = \{H_i\})$ with holes, its 2-simplex mesh $\mathcal{S}^{(j)}_2 = \{S^{(j)}_n\}_{n=1}^{N_g}$ is consist of oriented 2-simplices. 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 $p = (B, h = \{H_i\})$ 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 Eq. 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.
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

---

\(^8\) https://github.com/SPINlab/geometry-learning

---
NUFT features can be later used for IFFT, we can only use linear grid feature map \(W^{(fft)}\) (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 perceptron (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 \(W^{(fft)}\) (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 \(W^{(fft)}\) 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 \(W^{(gmf)}\) 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 Fig. 5a is represented as \([x^T_1, x^T_2, x^T_3, x^T_4, x^T_5, x^T_6, x^T_7, x^T_8, x^T_9, x^T_{10}, x^T_{11}, x^T_{12}]\) 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 1. The best hyperparameter combinations for different models are listed in Table 11. From Fig. 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 Fig. 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:

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 superiority 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 Fig. 9a and c), 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 subpolygons. 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

| #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) |
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 shown in Fig. 11. Figure 11a compares the PCA explained variance curves of three models when $N_{wx}=24$. Instead of fixing $N_{wx}$, Fig. 11b compares three

![PCA explained variance curve](image1)

(a) PCA explained variance curve ($N_{wx}=24$)

![Top $K_{PCA}$ PCA components for 90% data variance](image2)

(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.
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 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 Fig. 5a and c.

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 in Fig. 5a, the exterior of Polygon $p_0$, i.e., $B_0$, can be written as $B_0 = [x_A^T, x_B^T, x_C^T, x_D^T, x_E^T, x_F^T] \in \mathbb{R}^{6 \times 2}$ with $x_A$ as the start point. When we select $x_D$ as the start point, the exterior of Polygon $p_0$ becomes $B_0^{(3)} = L_3 B_0 = [x_D^T, x_E^T, x_F^T, x_A^T, x_B^T, x_C^T]$ which has an identical shape as $B_0$. 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 Fig. 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 Fig. 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 Fig. 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 Fig. 5a and c. 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 Fig. 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 Fig. 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 Fig. 13a, we can see that NUFTspec[gmf]+PCA+MLP sometimes fails because it cannot 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 the lower row of Fig. 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 Fig. 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 $E$ and triple set $T$ 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 $R = \{r_i\}^{N_r}_{i=1}$ 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 $\tilde{T} = \{(e_{sub}, r_i, e_{obj})\}$ forms a sub-graph of DBpedia with the entity set $\tilde{E}$.
4. For each entity $e \in \tilde{E}$, 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\(^9\).

\(^9\) https://wiki.openstreetmap.org/wiki/Overpass_API
The raw polygonal geometries from OpenStreetMap are very detailed and complex. For example, Keweenaw County, Michigan is represented as a multipolygon that consists 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-c show statistics about the complexity of these raw geometries. We simplify those polygonal geometries collected from OpenStreetmap and make two datasets: DBSR-46K and 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:northeast | 2063  | 863   | 300   | 900   |
| dbp:southeast | 2024  | 824   | 300   | 900   |
| dbp:southwest | 2000  | 904   | 274   | 822   |
| dbp:northwest | 1994  | 1175  | 205   | 614   |
| All           | 46578 | 36263 | 2579  | 7736  |
| ratio         | 100%  | 77.85%| 5.54% | 16.61%|

Fig. 14 A statistic of the complexity of the raw polygonal geometries as well as the simplified geometries in DBSR-cplx46K from OpenStreetMap. (a)-(c) indicate the statistics on the raw polygonal geometries retrieved from OpenStreetMap 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.

The raw polygonal geometries from OpenStreetMap are very detailed and complex. For example, Keweenaw County, Michigan is represented as a multipolygon that consists 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-c show statistics 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 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 an 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-e 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 10 in Appendix 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 \textit{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 \((e_{\text{sub}}, r, e_{\text{obj}})\), we first compute the shared minimal bounding box of subject polygon \(g_{\text{sub}}\) and object polygon \(g_{\text{obj}}\). Then we use this shared bounding box to normalize both \(g_{\text{sub}}\) and \(g_{\text{obj}}\) into \([-1, 1] \times [-1, 1]\) 2D unit space. Later on, VeerCNN and ResNet1D directly encode the normalized polygon geometries for spatial relation prediction. As for DDSL and NUFTspec, 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 Eq. 16 the same but vary the polygon encoders we use. Two exceptions are the Deterministic and DDSL+LeNet5. Deterministic is 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 \((e_{\text{sub}}, r, e_{\text{obj}})\), if the geometry of \(e_{\text{sub}}\) is inside of the geometry of \(e_{\text{obj}}\), Deterministic gives \(\text{dbo:isPartOf}\) 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 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 (Eq. 16).

3. **DDSL+LeNet5** [46]: We utilize the original DDSL implementation 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 Eq. 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 \(\mathcal{W}^{(f)}\) for NUFT followed by an MLP to encode the polygon into an embedding. The only difference is that it concatenates the embeddings of subject and object geometries and uses them in the spatial relation prediction model (Eq. 16).

---

10 [https://github.com/maxjiang93/DDSL](https://github.com/maxjiang93/DDSL)
7. NUFTspec[gmf]+MLP: Different from NUFTspec[fft]+MLP, this version of NUFTspec uses the geometric grid $W_{gmf}$.

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.

7.3 Main evaluation results

We evaluate 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 1 and the best hyperparameter combinations for different models on two datasets are listed in Table 12. 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, NUFTspec[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 $W_{gmf}$ 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 $W$. This allows us to design $W$ that can effectively capture irregular polygon representations.

### 7.4 Ablation study on ResNet1D

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 (Eq. 1) to enrich the point embedding with its neighborhood information.

### 7.5 The impact of $N_{wx}$

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

**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      |

DBSR-46K and DBSR-cplx46K. We can see that NUFTspec[gmf]+MLP achieves the best performance on all these four datasets with any given $N_{wx}$ and show a clear advantage over the other three models.
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. 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 Fig. 4a. This clearly indicates the necessity of polygon encoding 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. 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 Fig. 4a. This clearly indicates the necessity of polygon encoding models.

![Figure 17](image1.png) One example in DBSR-cplx46K testing set to show how the sliver polygons can affect 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          | `dbr:northeast`     |
| DDSL+MLP               | `dbr:northeast`     |
| ResNet1D               | `dbo:isPartOf`      |
| NUTFspec[gmf]+MLP      | `dbo:isPartOf`      |
| Ground Truth           | `dbo:isPartOf`      |

![Figure 18](image2.png) Another example in DBSR-cplx46K testing set to demonstrate the affect of the sliver polygon problem similar to Fig. 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.

| Model                  | Prediction          |
|------------------------|---------------------|
| Deterministic          | `dbr:northwest`     |
| DDSL+MLP               | `dbr:northeast`     |
| ResNet1D               | `dbo:isPartOf`      |
| NUTFspec[gmf]+MLP      | `dbo:isPartOf`      |
| Ground Truth           | `dbo:isPartOf`      |

---

11 [https://shapely.readthedocs.io/en/stable/manual.html#binary-predicates](https://shapely.readthedocs.io/en/stable/manual.html#binary-predicates)
To qualitatively show how our proposed polygon encoders mitigate the sliver polygon problem, we show two examples in Figs. 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 Fig. 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 $A$ is computed as the ratio between the areas of the subject geometry and object geometry. $A$ shows the scale different between the subjects and objects. Based on $A$, DBSR-cplx46K testing set are divided into seven different mutually exclusive area ratio groups. Table 9 shows the evaluation results of different polygon encoders on different area ratio groups of DBSR-cplx46K testing set. Given a triple, the area ratio $A$ is the ratio between the areas of the subject geometry and object geometry. The table shows the predictions of different polygon encoders for this example.

![Fig. 19](image_url) 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.

| Model            | Area Ratio $A$          | # Triangles | 0.0, 0.1 | 0.1, 0.2 | 0.2, 0.3 | 0.3, 1   | 1, 1.1   | 1.1, 1.2 | 1.2, ∞   |
|------------------|-------------------------|-------------|----------|----------|----------|----------|----------|----------|----------|
| Deterministic    |                         | 1027        | 156      | 167      | 3152     | 675      | 411      | 2148     |
| VeerCNN          | 66.60                   | 50.00       | 62.87    | 74.62    | 81.19    | 74.70    | 411      | 2148     |
| DDSL+LeNet5      | 86.95                   | 51.92       | 50.30    | 75.44    | 82.67    | 76.89    | 76.96    |
| DDSL+MLP         | 87.93                   | 58.33       | 61.68    | 76.84    | 84.44    | 81.75    | 77.37    |
| ResNet1D         | 87.24                   | 54.49       | 63.47    | 79.09    | 84.44    | 80.78    | 78.68    |
| NUFTspec[ff]    +MLP | 87.05                 | 45.51       | 52.69    | 76.78    | 84.59    | 79.81    | 78.26    |
| NUFTspec[gmf] +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    | 83.94    | 83.94    | 79.24    |

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.
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[gm]+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 Figs. 19 and 20. In both cases, NUFTspec[fft]+MLP and Deterministic can correctly predict the spatial relation. However, two spatial domain polygon encoders – VeerCNN and ResNet1D – predict $\text{dbo:isPartOf}$, although the subject geometry is clearly not part of the object geometry. We guess the reason is that both VeerCNN 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 $\text{dbo:isPartOf}$ 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, 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 \( W^{gmf} \), 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 \texttt{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 \( W^{gmf} \) and \( W^{fft} \), can we let the neural network learn the optimal \( W \) 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 polygon 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.
Appendix A

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

Table 10.

| 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 $N_{wx} \in \{16, 20, 24, 28, 32, 36, 40, 44\}$ for the shape classification task while $N_{wx} \in \{16, 32, 64\}$ for the spatial relation prediction task. As for NUFTspec[gmf]-based models, we tune the $w_{min} \in \{0.2, 0.4, 0.5, 0.8, 1.0\}$ 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$_{cp}$ - $K \in \{1, 2, 3\}$. 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 $MLP_{P,\cdot}(\cdot)$ 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.

The best hyperparameter combinations for all models on MNIST-cplx70k are shown in Table 11. As for DBSR-46K and DBSR-cplx46K, each model’s best hyperparameter combinations are shown in Table 12.
Table 11  The best hyperparameter combinations for each model on MNIST-cplt70k dataset

| Model            | $lr$ | $d$   | $N_{wx}$ | $w_{min}$ | $w_{max}$ | $\sum_{PCA}$ | $K_{PCA}$ | $K$ | $2t$ |
|------------------|------|-------|----------|-----------|-----------|---------------|-----------|-----|-----|
| 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       | -         | -         | 80%           | 42        | -   | -   |
| NUFTspec[fft]+PCA+MLP | 0.0005 | 512 | 24     | 0.5       | 12        | 95%           | 46        | -   | -   |

Table 12  The best hyperparameter combinations for each model on DBSR-46K and DBSR-cplt46K dataset

| Dataset       | Model       | $lr$ | $d$   | $N_{wx}$ | $w_{min}$ | $w_{max}$ | $K$ | $2t$ |
|---------------|-------------|------|-------|----------|-----------|-----------|-----|-----|
| 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-cplt46K  | 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        | -   | -   |

Acknowledgements  This work is mainly funded by the National Science Foundation under Grant No. 2033521 A1 – KnowWhereGraph: Enriching and Linking Cross-Domain Knowledge Graphs using Spatially-Explicit AI Technologies and the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2021-201100004. Stefano Ermon acknowledges support from NSF (#1651565), AFOSR (FA95501910024), ARO (W911NF-21-1-0125), Sloan Fellowship, and CZ Biohub. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Data availability  DBpedia uses GNU General Public License and OpenStreetMap uses Open Database License. Both of them are open dataset for academic usage. They do not have personally identifiable information for the privacy protection purpose. Experimental data and the methods developed will be openly shared for reproducibility and replicability https://github.com/gengchenmai/polygon_encoder.

Declarations

Conflicts of interest  This paper has been approved by all co-authors. The authors have no competing interests to declare that are relevant to the content of this article.
References

1. Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P (2017) Geometric deep learning: going beyond Euclidean data. IEEE Signal Process Mag 34(4):18–42
2. Mai G, Janowicz K, Hu Y, Gao S, Yan B, Zhu R, Cai L, Lao N (2021) A review of location encoding for GeoAI: methods and applications. Int J Geogr Inf Sci
3. Monti F, Boscaini D, Masci J, Rodolà E, Svoboda J, Bronstein MM (2017) Geometric deep learning on graphs and manifolds using mixture model CNNs. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE Xplore, Honolulu, pp 5115–5124
4. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. NIPS
5. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations, May 2-4, 2016. OpenReview, San Juan, Puerto Rico
6. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp 1025–11035. LongBeach. NeurIPS Proceedings
7. Schlichtkrull M, Kipf TN, Bloem P, Van Den Berg R, Titov I, Welling M (2018) Modeling relational data with graph convolutional networks. In: European semantic web conference, Heraklion, Crete. Greece, Springer, pp 593–607
8. Cai L, Yan B, Mai G, Janowicz K, Zhu R (2019) TransGCN: Coupling transformation assumptions with graph convolutional networks for link prediction. In: Proceedings of the 10th International Conference on Knowledge Capture. ACM Proceeding, Marina Del Rey, pp 131–138
9. Mai G, Janowicz K, Cai L, Zhu R, Regalia B, Yan B, Shi M, Lao N (2020) SE-KGE: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting. Trans GIS. https://doi.org/10.1111/tgis.12629
10. Qi CR, Su H, Mo K, Guibas LJ (2017) Pointnet: Deep learning on point sets for 3d classification and segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, In, pp 652–660
11. Li Y, Bu R, Sun M, Wu W, Di X, Chen B (2018) Pointcnn: Convolution on x-transformed points. Adv Neural Inf Process Syst 31:820–830
12. Mac Aodha O, Cole E, Perona P (2019) Presence-only geographical priors for fine-grained image classification. In: Proceedings of the IEEE International Conference on Computer Vision. IEEE Xplore, Seoul, pp 9596–9606
13. Mai G, Janowicz K, Yan B, Zhu R, Cai L, Lao N (2020) Multi-scale representation learning for spatial feature distributions using grid cells. In: The Eighth International Conference on Learning Representations. OpenReview, Addis Ababa
14. Masci J, Boscaini D, Bronstein M, Vandergheynst P (2015) Geodesic convolutional neural networks on riemannian manifolds. In: Proceedings of the IEEE international conference on computer vision workshops. Santiago, IEEE Xplore, pp 37–45
15. Lazer D, Pentland AS, Adamic L, Aral S, Barabasi AL, Brewer D, Christakis N, Contractor N, Fowler J, Gutmann M et al (2009) Life in the network: the coming age of computational social science. Science (New York, NY) 323(5915):721
16. Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D (2019) Graph neural networks for social recommendation. In: The world wide web conference. IEEE Xplore, San Francisco, pp 417–426
17. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: ICML. Proceedings of Machine Learning Research, Sydney
18. Davidson EH, Rast JP, Oliveri P, Ransick A, Celestani C, Yuh CH, Minokawa T, Amore G, Hinman V, Arenas-Mena C et al (2002) A genomic regulatory network for development. Science 295(5560):1669–1678
19. Li Y, Yu R, Shahabi C, Liu Y (2019) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representations, OpenReview, New Orleans
20. Cai L, Janowicz K, Mai G, Yan B, Zhu R (2020) Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. Trans GIS 24(3):736–755
21. Lin Y, Mago N, Gao Y, Li Y, Chiang YY, Shahabi C, Ambite JL (2018) 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. ACM Proceeding, Seattle, pp 359–368
22. Appleby G, Liu L, Liu LP (2020) Kriging convolutional networks. In: Proceedings of AAAI 2020. AAAI Digital Library Conference Proceedings, New York
23. Wu Y, Zhuang D, Labbe A, Sun L (2021) Inductive graph neural networks for spatiotemporal kriging. Proceedings of AAAI 35:4478–4485
24. Xu Y, Piao Z, Gao S (2018) Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In: CVPR 2018. IEEE Xplore, Salt Lake City, pp 5275–5284
25. Zhang P, Ouyang W, Zhang P, Xue J, Zheng N (2019) Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach. IEEE Xplore, USA, pp 12085–12094
26. Rao J, Gao S, Kang Y, Huang Q (2020) LSTM-TrajGAN: A deep learning approach to trajectory privacy protection. In: GIScience 2020. Leibniz International Proceedings in Informatics series, Poznań, pp 12:1–12:17
27. Li Y, Yu R, Shahabi C, Liu Y (2018) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: ICLR 2018. OpenReview, Vancouver
28. Rv V, Bloem P, Folmer E (2018) Deep learning for classification tasks on geospatial vector polygons. arXiv preprint arXiv:1806.03857
29. Yan X, Ai T, Yang M, Tong X (2021) Graph convolutional autoencoder model for the shape coding and cognition of buildings in maps. Int J Geogr Inf Sci 35(3):490–512
30. He X, Zhang X, Xin Q (2018) Recognition of building group patterns in topographic maps based on graph partitioning and random forest. ISPRS J Photogramm Remote Sens 136:26–40
31. Yan X, Ai T, Yang M, Yin H (2019) A graph convolutional neural network for classification of building patterns using spatial vector data. ISPRS J Photogramm Remote Sens 150:259–273
32. Bei W, Guo M, Huang Y (2019) A spatial adaptive algorithm framework for building pattern recognition using graph convolutional networks. Sensors 19(24):5518
33. Yan X, Ai T, Yang M, Tong X, Liu Q (2020) A graph deep learning approach for urban building grouping. Geocarto International 37(10):2944–2966 Taylor & Francis
34. Feng Y, Thiemann F, Sester M (2019) Learning cartographic building generalization with deep convolutional neural networks. ISPRS Int J Geo-Inf 8(6):258
35. Zelle JM, Mooney RJ (1996) Learning to parse database queries using inductive logic programming. In: Proceedings of the national conference on artificial intelligence, Portland. AAAI Proceedings, pp 1050–1055
36. Punjani D, Singh K, Both A, Koubarakis M, Angelidis I, Bereta K, Beris T, Bilidas D, Ioannidis T, Karalis N et al (2018) Template based question answering over linked geospatial data. In: Proceedings of the 12th Workshop on Geographic Information Retrieval. ACM Proceedings, Seattle, pp 1–10
37. Scheider S, Nyamsuren E, Kruiger H, Xu H (2021) Geo-analytical question-answering with gis. Int J Digit Earth 14(1):1–14
38. Mai G, Yan B, Janowicz K, Zhu R (2019) Relaxing unanswerable geographic questions using a spatially explicit knowledge graph embedding model. In: AGILE. Springer, Limassol, pp 21–39
39. Mai G, Janowicz K, Zhu R, Cai L, Lao N (2021) Geographic question answering: Challenges, uniqueness, classification, and future directions. AGILE GIScience Ser 2:1–21
40. Sun X, Christoudias CM, Fua P (2014) Free-shape polygonal object localization. In: European Conference on Computer Vision. Springer, pp 317–332. Zurich, Springer
41. Castrejon L, Kundu K, Urtasun R, Fidler S (2017) Annotating object instances with a Polygon-RNN. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE Xplore, Honolulu, pp 5230–5238
42. Acuna D, Ling H, Kar A, Fidler S (2018) Efficient interactive annotation of segmentation datasets with Polygon-RNN++. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. Salt Lake City, IEEE Xplore, pp 859–868
43. Bai X, Liu W, Tu Z (2009) Integrating contour and skeleton for shape classification. In: 2009 IEEE 12th international conference on computer vision workshops, ICCV workshops, Kyoto, IEEE Xplore, pp 360–367
44. Wang X, Feng B, Bai X, Liu W, Latecki LJ (2014) Bag of contour fragments for robust shape classification. Pattern Recogn 47(6):2116–2125
45. Regalia B, Janowicz K, McKenzie G (2019) Computing and querying strict, approximate, and metrically refined topological relations in linked geographic data. Trans GIS 23(3):601–619
46. Jiang C, Lansigan D, Marcus P, Niefner M et al (2019) DDSL: Deep differentiable simplex layer for learning geometric signals. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. IEEE Xplore, Seoul, pp 8769–8778
47. Jiang CM, Wang D, Huang J, Marcus P, Niessner M (2019) Convolutional neural networks on non-uniform geometrical signals using euclidean spectral transformation. In: International Conference on Learning Representations. OpenReview, New Orleans
48. Kurniawanggoro L, Jo KH et al (2018) A survey of 2d shape representation: Methods, evaluations, and future research directions. Neurocomputing 300:1–16
49. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
50. Randell DA, Cui Z, Cohn AG (1992) A spatial logic based on regions and connection. In: 3rd International Conference on Knowledge Representation and Reasoning. AAAI Proceedings, Haifa, pp 165–176
51. Egenhofer MJ, Franzosa RD (1991) Point-set topological spatial relations. Int J Geogr Inf Syst 5(2):161–174
52. Zhang Z, Fidler S, Waggoner J, Cao Y, Dickinson S, Siskind JM, Wang S (2012) Sup eredge grouping for object localization by combining appearance and shape information. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp 3266–3273. Rhode Island. IEEE Xplore
53. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
54. Li Y, Tarlow D, Brockschmidt M, Zemel R (2016) Gated graph neural networks. In: ICLR 2016. OpenReview, New Orleans
55. Liang J, Homayounfar N, Ma WC, Xiong Y, Hu R, Urtasun R (2020) Polytransform: Deep polygon transformer for instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE Xplore, Seattle, pp 9131–9140
56. Atabay HA (2016) Binary shape classification using convolutional neural networks. IIOAB J 7(5):332–336
57. Atabay HA (2016) A convolutional neural network with a new architecture applied on leaf classification. IIOAB J 7(5):226–331
58. Hofer C, K witt R, Niethammer M, Uhl A (2017) Deep learning with topological signatures. In: NIPS 2017. NeurIPS Proceedings, Long Beach
59. Baker N, Lu H, Erli khman G, Kellman PJ (2018) Deep convolutional networks do not classify based on global object shape. PloS Comput Biol 14(12):1006613
60. Lakecki LJ, Lakamper R, Eckhardt T (2000) 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. IEEE Xplore, Hilton Head, pp 424–429
61. Söderkvist O (2001) Computer vision classification of leaves from Swedish trees. PhD thesis
62. Leibe B, Schiele B (2003) Analyzing appearance and contour based methods for object categorization. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings, vol 2. IEEE Xplore, Madison, pp II–409
63. Mallah C, Cope J, Orwell J et al (2013) Plant leaf classification using probabilistic integration of shape, texture and margin features. Signal Process Patt Recogn Appl 5(1):45–54
64. Sebastian TB, Kimia BB (2005) Curves vs. skeletons in object recognition. Signal Process 85(2):247–263
65. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Munich. Springer, pp 234–241
66. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE Xplore, Las Vegas, pp 770–778
67. Yu F, Wang D, Shel hammer E, Darrell T (2018) Deep layer aggregation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE Xplore, Salt Lake City, pp 2403–2412
68. Rippel O, Snoek J, Adams RP (2015) Spectral representations for convolutional neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, vol 2. NeurIPS Proceedings, pp 2449–2457
69. Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R (2020) Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. Springer, Glasgow, pp 405–421
70. Tancik M, Srinivasan PP, Mildenhall B, Fridovich-Keil S, Raghavan N, Singhal U, Ramamoorthi R, Barron JT, Ng R (2020) Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems. Virtual-only. NeurIPS Proceedings, vol 33, pp 7537–7547
71. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems. NeurIPS Proceedings, Long Beach, pp 5998–6008
72. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. arXiv preprint arXiv:1607.06450
73. Ha D, Eck D (2018) A neural representation of sketch drawings. In: International Conference on Learning Representations. OpenReview, Vancouver
74. Deng C, Litany O, Duan Y, Pουlenard A, Tagliasacchi A, Guibas LJ (2021) Vector neurons: A general framework for so (3)-equivariant networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 12200–12209. Montreal, IEEE Xplore
75. Esteves C, Allen-Blanchette C, Makadia A, Daniilidis K (2018) Learning so (3) equivariant representations with spherical CNNs. In: Proceedings of the European Conference on Computer Vision (ECCV). Springer, Munich, pp 52–68
76. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Neural Information Processing Systems (NIPS). NeurIPS Proceedings, Lake Tahoe, pp 1–
77. Chen W (2014) Parameterized spatial SQL translation for geographic question answering. In: 2014 IEEE International Conference on Semantic Computing. IEEE Xplore, Newport Beach, pp 23–27
78. Yan B, Janowicz K, Mai G, Gao S (2017) From ITDL to place2vec: Reasoning about place type similarity and relatedness by learning embeddings from augmented spatial contexts. In: Proceedings of the 25th ACM SIGSPATIAL international conference on advances in geographic information systems. ACM Proceedings, Redondo Beach, pp 1–10
79. Yan B, Janowicz K, Mai G, Zhu R (2019) A spatially explicit reinforcement learning model for geographic knowledge graph summarization. Trans GIS 23(3):620–640
80. Janowicz K, Gao S, McKenzie G, Hu Y, Bhaduri B (2020) GeoAI: spatially explicit artificial intelligence techniques for geographic knowledge discovery and beyond. Int J Geogr Inf Sci 34(4):625–636 Taylor & Francis
81. Li W, Hsu CY, Hu M (2021) Tobler’s first law in geoai: A spatially explicit deep learning model for terrain feature detection under weak supervision. Ann Assoc Geogr 111(7):1887–1905
82. Mai GM, Cundy C, Choi K, Hu Y, Lao N, Ermon S (2022) Towards a foundation model for geospatial artificial intelligence. In: Proceedings of the 30th SIGSPATIAL international conference on advances in geographic information systems. https://doi.org/10.1145/3557915.3561043

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
Gengchen Mai  Dr. Gengchen Mai is currently a Tenure-Track Assistant Professor at the Department of Geography, University of Georgia, as a member of the UGA Environmental Artificial Intelligence Faculty Cluster. He is also the director of Spatially Explicit Artificial Intelligence Lab at UGA and an Affiliated Professor of the UGA Institute for Artificial Intelligence. Before he came to UGA, he was a Post-doctoral Scholar at Stanford AI Lab, Stanford University, working with Prof. Stefano Ermon on several geospatial machine learning and computation sustainability projects. He was also affiliated with the Stanford Sustainability and AI Lab. Gengchen obtained his PhD degree in Geographic Information Science from the Department of Geography, University of California, Santa Barbara in 2021 and B.S. degree in GIS from Wuhan University in 2015. His research mainly focuses on spatially explicit artificial intelligence, geospatial knowledge graph, and geographic question answering.

Chiyu Max Jiang  Dr. Chiyu Max Jiang is a Research Scientist at Waymo. His primary focus is building machine learning algorithms with autonomous vehicle applications, while his research interests extend to 3D perception, long-tail learning, and deep generative models. Prior to Waymo, he worked in various industrial and government-funded research labs, including Google Research, Cruise and Lawrence Berkeley National Laboratory. He holds a Ph.D. in Computer Vision and Physics-Informed Machine Learning from UC Berkeley, and a B.S. in Biological Engineering from both Cornell University and Zhejiang University. This work was done while he was a PhD student at UC Berkeley.

Weiwei Sun  Weiwei Sun is a Ph.D. student in Computer science of the University of British Columbia. His research focus is on 3D computer vision and geometry processing. He received his master degree from University of Calgary and B.Sc. from Wuhan University.
Rui Zhu  Dr. Rui Zhu is currently a Lecturer (Assistant Professor), at the School of Geographical Sciences, University of Bristol. Before joining University of Bristol, Rui was a Postdoctoral Scholar at the Center for Spatial Studies, University of California, Santa Barbara. He obtained a PhD in Geography from the University of California, Santa Barbara. He also holds a master degree in Information Sciences from the University of Pittsburgh and a bachelor degree in Information Management and Information Systems from Shanxi University of Finance and Economics. Rui has expertise in spatial statistics, geospatial semantics, knowledge graphs, and GeoAI. His work has been applied to urban planning, global health, environmental intelligence, as well as humanitarian relief.

Yao Xuan  Dr. Yao Xuan is a research scientist at Meta (Facebook). Yao got his Ph.D. from the Department of Mathematics at the University of California, Santa Barbara. His research focuses on combining machine learning methods and traditional computational techniques to solve problems in computational mathematics, such as accelerating computations of self-consistent field theory, solving high dimensional stochastic games, modeling time series by transformers, etc. This work was done while he was a PhD student at UC Santa Barbara.

Ling Cai  Ling Cai is a PhD Candidate at the Space and Time Knowledge Organization Lab, Department of Geography, University of California, Santa Barbara. She obtained her M.S. degree in Geographical Information Science from Chinese Academy of Sciences and B.S. degree from Wuhan University. Her research interests include qualitative spatial temporal reasoning, temporal knowledge graph, neuro-symbolic AI, and urban computing.
Krzysztof Janowicz Dr. Krzysztof Janowicz is a Professor for Geoinformatics at the University of California, Santa Barbara and director of the Center for Spatial Studies. He is also a Professor at the Department of Geography and Regional Research, University of Vienna. His research focuses on how humans conceptualize the space around them based on their behavior, focusing particularly on regional and cultural differences with the ultimate goal of assisting machines to better understand the information needs of an increasingly diverse user base. Janowicz’s expertise is in knowledge representation and reasoning as they apply to spatial and geographic data, e.g. in the form of knowledge graphs.

Stefano Ermon Dr. Stefano Ermon is an Associate Professor in the Department of Computer Science at Stanford University, where he is affiliated with the Artificial Intelligence Laboratory and a fellow of the Woods Institute for the Environment. The goal of his research is to enable innovative solutions to problems of broad societal relevance through advances in probabilistic modeling, learning and inference. Stefano develop new foundational methods motivated by concrete real-world applications, focusing on a new area that bridges computer science with other disciplines to address core questions in sustainability, including poverty mitigation, food security, and renewable energy.

Ni Lao Dr. Ni Lao is a research and product lead at Google. His expertise include Knowledge Graph (KG), Natural Language Understanding (NLU) and Machine Learning (ML). He was the Chief scientist and co-founder of Mosaix.ai, a voice AI startup.
Authors and Affiliations

Gengchen Mai\textsuperscript{1,2,3,4} \cdot Chiyu Jiang\textsuperscript{5} \cdot Weiwei Sun\textsuperscript{6} \cdot Rui Zhu\textsuperscript{3,4,7} \cdot Yao Xuan\textsuperscript{8} \cdot Ling Cai\textsuperscript{3,4} \cdot Krzysztof Janowicz\textsuperscript{3,4,9} \cdot Stefano Ermon\textsuperscript{2,10} \cdot Ni Lao\textsuperscript{11}

\begin{itemize}
\item Gengchen Mai
gengchen.mai25@uga.edu
\item Chiyu Jiang
maxjiang93@gmail.com
\item Weiwei Sun
weiweis@cs.ubc.ca
\item Rui Zhu
rui.zhu@bristol.ac.uk
\item Yao Xuan
yxuan@math.ucsb.edu
\item Ling Cai
ling.cai@geog.ucsb.edu
\item Krzysztof Janowicz
jano@geog.ucsb.edu
\item Stefano Ermon
ermon@cs.stanford.edu
\item Ni Lao
nlao@google.com
\end{itemize}

\textsuperscript{1} Spatially Explicit Artificial Intelligence Lab, Department of Geography, University of Georgia, Athens 30602, Georgia, USA
\textsuperscript{2} Department of Computer Science, Stanford University, Stanford 94305, California, USA
\textsuperscript{3} STKO Lab, University of California Santa Barbara, Santa Barbara 93106, California, USA
\textsuperscript{4} Center for Spatial Studies, University of California Santa Barbara, Santa Barbara 93106, California, USA
\textsuperscript{5} Department of Mechanical Engineering, University of California Berkeley, Berkeley 94720, California, USA
\textsuperscript{6} Department of Computer Science, University of British Columbia, Vancouver V6T 1Z4, British Columbia, Canada
\textsuperscript{7} School of Geographical Sciences, University of Bristol, Bristol BS8 1TH, UK
\textsuperscript{8} Department of Mathematics, University of California Santa Barbara, Santa Barbara 93106, California, USA
\textsuperscript{9} Department of Geography and Regional Research, University of Vienna, Vienna 1040, Austria
\textsuperscript{10} Chan Zuckerberg Biohub, San Francisco 94158, California, USA
\textsuperscript{11} Google, Mountain View 94043, California, USA