GeoInformatica (2026) 30:32
https://doi.org/10.1007/s10707-026-00587-x

RESEARCH

On the applicability of artificial intelligence models to geospatial vector data analysis and exploration

Majid Saeedan[1] · **Alberto Belussi**[2] · **Sara Migliorini**[2] · **Ahmed Eldawy**[1]

Received: 25 March 2026 / Revised: 15 June 2026 / Accepted: 13 August 2026

Abstract

Data-driven prediction and decision-making increasingly depend on large-scale data lakes and machine learning (ML) and deep learning (DL) analytics across a wide range of application domains. Nevertheless, although geospatial vector data constitutes a substantial fraction of contemporary datasets, it has remained primarily managed within traditional GIS and database systems and has benefited only to a limited extent from recent advances in artificial intelligence (AI). This limitation has been largely attributable to the intrinsic characteristics of geospatial information, in particular when represented in vector format, like multidimensional spatial coordinates, scale and projection effects, variable-size geometries, and a broad spectrum of spatial operations, which are difficult to reconcile with the fixed-dimensional inputs typically required by DL models. This paper advanced DL-enabled geospatial analytics by enabling AI models and agents to provide native access, interpretation, and query processing over geospatial vector datasets. Rather than proposing novel architectures, we introduced a methodology for leveraging existing DL models by properly encoding both geospatial inputs and the spatial operations to be estimated. We investigated three representation/architecture families: (i) image-like, georeferenced histograms processed with ResNet and UNet, (ii) point- and graph-based variants of PointNet++, and (iii) fixed-length geometry embeddings obtained by extending Poly2Vec combined with a Transformer model. We instantiated these approaches on three representative geospatial operators, spatial synopsis, spatial clustering, and walkability estimation, and evaluated them using synthetic and real-world datasets. The results demonstrated that all three families could achieve high accuracy in specific configurations with distinct trade-offs: graph-based models were most effective for dense data and short-range interactions, whereas image- and vector-based approaches exhibited superior scalability due to fixed-size inputs, and the image-based approach was particularly effective for multi-dataset operations.

Keywords Spatial deep learning · Spatial data representation

Extended author information available on the last page of the article

Published online: 07 September 2026

1 Introduction

We are witnessing continuous growth in data-driven scientific methods that rely on abundant real data to inform scientific findings. Most of these applications are based on *data lakes* [1, 2] that store very large datasets and provide scalable analytics. Recent advances in machine learning (ML) and deep learning (DL) are transforming data analytics from human-driven query execution to model-driven data reasoning. Vector databases [3] now enable robust semantic search, powering applications like Retrieval Augmented Generation (RAG) [4], which allow AI agents to automatically retrieve relevant datasets for the desired task. However, retrieval performance depends on the quality of embeddings produced by models such as CLIP [5].

In this context, geospatial information and analytics are attracting increased attention, as studies show that about 60-80% of today's data contains a geospatial component [6]. Example datasets include temperature measurements [7], traffic prediction [8], and online text posts [9], with applications in urban planning [10], epidemiology [11, 12], and weather forecasting [13]. Meanwhile, geospatial data is still primarily supported in traditional data lakes, which creates challenges for its effective and efficient management. In particular, it is important to note that geospatial data are commonly represented using two fundamental approaches: raster and vector. Raster data represent geographic space as a regular grid of cells, where each cell stores the value of a given attribute. This representation is widely used for satellite imagery, remote sensing products, and spatial phenomena modeled as continuous fields. In contrast, vector data describe geographic space through a set of discrete spatial objects, such as points, lines, and polygons, each associated with geometric and semantic properties. For example, in a raster representation, the spatial distribution of hotels in downtown New York could be modeled by storing in each cell the number of hotels located within that area. In a vector representation, the same information would be stored as a collection of individual hotel objects, each associated with its geographic location, typically represented by point coordinates, together with additional descriptive attributes.

In this paper, we focus on vector geospatial data, as they have received considerably less attention in the literature than raster data and pose several unique challenges for AI-based analysis. These challenges stem from a number of distinctive characteristics.

First, geospatial data are inherently multidimensional. Unlike traditional data sources such as text or time series, spatial data are embedded in two- or three-dimensional space, making the identification of correlations and patterns among individual elements more complex. Second, vector data describe geographic entities through geometries and spatial relationships, requiring the management of issues such as scale, coordinate reference systems, map projections, and spatial dependencies associated with the Earth's surface. Third, geospatial applications rely on a rich set of spatial operations characterized by heterogeneous inputs and outputs. Such operations may involve one or multiple datasets and may produce either spatial objects or scalar values. Moreover, spatial objects are typically represented by a variable number of vertices, while spatial datasets may contain a variable number of objects. This intrinsic variability contrasts with the assumptions of most machine learning and deep learning models, which generally require fixed-size input representations.

Our goal is to move toward DL architectures that enable AI models and agents to natively access, interpret, and query vector geospatial data. Traditional GIS queries and domain-specific programs cannot easily express or evaluate semantic constraints, as they rely on

rigid schemas, explicit predicates, and expert-crafted spatial logic. In contrast, DL-based approaches can learn rich representations of geospatial and contextual properties, enabling AI agents to perform similarity- and embedding-based queries that better align with human intent. However, as discussed above, fundamental challenges remain in representing vector geospatial data for DL models and in evaluating how well these models capture spatial semantics and operations critical for reliable reasoning.

This paper explores how to enrich existing DL models to express a wide range of complex geospatial analytical functions working with vector data. The goal of this work is not to develop new models, but rather to provide a methodology for effectively exploiting existing models through suitable encodings of vector geospatial datasets. Specifically, we investigate how such models can be used to predict the outcome of selected spatial operations, chosen as representative of broader classes of geospatial analytical procedures. This work represents a first step toward DL-based geospatial query analytics, laying the foundation for AI agents that can efficiently search and reason over thousands of geospatial datasets using deep learning.

Figure 1 provides an overview of the proposed method. Traditional approaches require a different algorithm for each problem that processes the full dataset to compute an exact answer. Although these approaches can be accurate, implementing a separate algorithm for every task requires substantial development effort, and repeatedly processing large geospatial datasets can be computationally expensive. In contrast, the proposed method explores the use of AI-based models to produce fast approximate answers with minimal task-specific development. First, the input dataset is passed through a learned geospatial representation component, which transforms vector geospatial data into a form suitable for AI models. Then, for each target problem, we train a learned model to approximate the behavior of the corresponding algorithm and produce an answer efficiently. Each model is learned from sample input–output instances, rather than being manually implemented as a customized algorithm.

To achieve this goal, the proposed approach is organized into two main components, which are described in greater detail below:

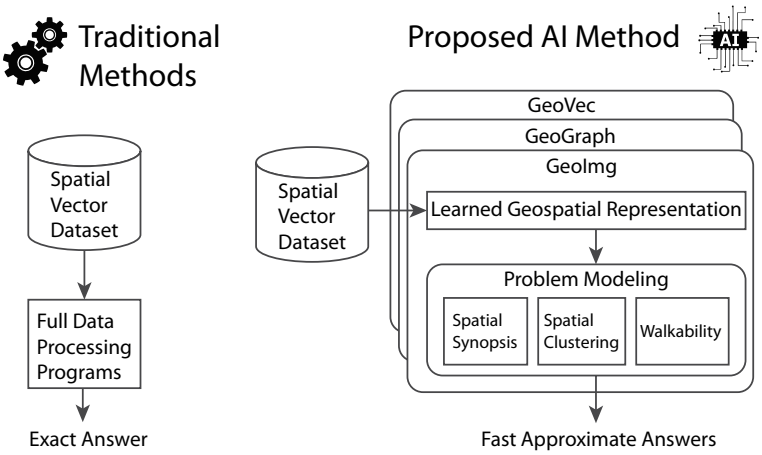

Fig. 1 Overview of the proposed AI-based method for geospatial vector data analysis

- **Spatial data representation**: First, we investigated how three popular deep learning architectures can be adapted to process vector geospatial data. The first paradigm is an image-based architecture, in which vector geospatial data are transformed into a fixed-size georeferenced histogram that serves as a raster representation of the original dataset. Each cell stores spatial and non-spatial information describing the geographic objects that intersect it. We adopted ResNet [14] and UNet [15] as representative image-based models. The second paradigm is a graph- and point-based architecture, where the input consists of a set of spatial points characterized by geographic coordinates and associated attributes. To support vector geospatial data, we designed and evaluated several adaptations of the PointNet++ [16] architecture. The third paradigm is an embedding-based architecture, in which geometries are encoded into fixed-length vector representations. In this context, the term vector refers to a numerical embedding that compactly captures the geometric and semantic characteristics of a spatial object. To evaluate this approach, we extended the Poly2Vec [17] encoder and integrated it with a Transformer architecture [18]. Our initial study showed that all three paradigms can effectively address a variety of geospatial learning tasks, although each exhibits distinct strengths and limitations.
- **Spatial problem modeling**: Second, we modeled three representative geospatial tasks using the three data representations and corresponding model architectures described above: spatial data synopsis, spatial clustering, and walkability estimation. These tasks were selected as representative instances of broader classes of geospatial queries, enabling us to evaluate the effectiveness of the proposed approach across different analytical scenarios. The selected tasks involve diverse spatial characteristics, including proximity relationships, spatial density, and the interaction among geographic entities. As a result, they provide a comprehensive testbed for assessing the ability of the proposed representations and learning architectures to capture different spatial phenomena while accelerating query processing. More generally, our objective was to investigate how the three representation paradigms support the estimation of query results for heterogeneous geospatial analysis tasks.

To evaluate the proposed architectures, we conducted an extensive experimental study using both synthetic and real-world datasets. The results demonstrated that all three approaches can effectively estimate the outcomes of geospatial analytical tasks, although each exhibits distinct strengths and limitations. In particular, the graph-based architecture achieved the best performance on dense datasets characterized by small distances among geometries, where local spatial relationships play a crucial role. In contrast, the image- and embedding-based architectures exhibited superior scalability because they rely on fixed-size input representations. Furthermore, the image-based approach proved particularly effective for tasks that involve integrating multiple datasets.

Overall, the contribution of the paper can be summarized as follows. (i) We propose a methodology for applying deep learning to vector geospatial data through suitable dataset encodings. (ii) We study and compare three representation paradigms: image-based, point-/graph-based, and embedding-based. (iii) We adapt and evaluate several state-of-the-art architectures, including ResNet, UNet, PointNet++, and a Transformer-enhanced Poly2Vec encoder. (iv) We validate the approach on three representative geospatial tasks and show that DL models can approximate spatial query results accurately and efficiently. (v) We pro-

vide empirical evidence of the trade-offs among the three paradigms and outline directions for future research on learned geospatial query analytics.

The remainder of this paper is structured as follows. Section 3 discusses the three representation methods that we considered. Section 4 illustrates the three geospatial problems, their definitions, data preparation, and model architectures. Section 5 details the extensive experimental evaluation and discusses the results. Following that, Section 2 presents the related work. Finally, Section 6 concludes the paper and discusses future directions.

2 Related work

Deep learning for geospatial raster data – The use of deep learning architectures with geospatial data is improving. Recently, libraries like TorchGeo [19] have been introduced, which include geospatial foundation models such as [20–23], among others; refer to [24] for a recent survey. More recent work has also explored multimodal and location-aware pretraining for geospatial imagery, including vision-location contrastive models such as SatCLIP [25] and image-to-location alignment models such as GeoCLIP [26]. However, all of these are based on vision models and tailored for geospatial raster data, a different modality from geospatial vector data. Since this work focuses on vector geospatial data, we restrict our comparison to approaches operating on vector representations. Methods designed for raster data are not considered, as they rely on different assumptions and are tailored to a fundamentally different representation of geographic information.

Deep learning for geospatial vector data – Although deep learning has been successfully applied to geospatial vector data, full support is still lacking. There have been several proposals for better representing geospatial vector data for deep learning architectures. These include [17, 27–29], which all focused on representing a single object, either a point or a polygon, and possibly encoding the context around the object. Related location-encoding approaches, such as SatCLIP [25] and GeoCLIP [26], further show the importance of learning reusable geographic embeddings, although they primarily encode individual coordinates or image-location pairs rather than complete vector datasets and spatial operations. However, this is very limiting, since many geospatial problems require analyzing large sets of geometries, as in the problems we consider in this work. We integrated Poly2Vec [17] with the GeoVec architecture, and it is a promising approach. The work in [30] provides a survey of different spatial representations, including models for point sets. To the best of our knowledge, this is the first work to study point set models for geospatial applications, analyzing their effectiveness across various spatial operations and identifying existing limitations.

Point-set architectures – Several architectures have been proposed that take a set of points as input, and they are mainly related to the points-of-cloud domain, where a set of points represents three-dimensional objects. PointNet [31] was among the first architectures to process unordered point sets directly by enforcing permutation invariance, while PointNet++ [16] extended this idea by hierarchically capturing local structures at multiple spatial scales. We chose to study PointNet++ [16] due to its suitability for the problems we con-

sider and its hierarchical, multi-scale approach. While we focused on this architecture, we kept our discussions general enough so that different layer types can be replaced, since this might produce higher-quality results. However, our goal is not to provide the best accuracy possible but to learn about the suitability of point-set-based approaches for problems that involve geospatial vector data. Other alternatives include graph-based point-cloud models, such as DGCNN [32], which dynamically construct neighborhood graphs in feature space, and attention-based architectures, such as Point Transformer [33]. Point Transformer's more recent version PTv3 [34] provides more efficient aggregation by using space-filling curves. Anyway, it does not support processing attributes associated with points, a requirement for all our problems. Also, these new aggregations are not yet integrated with PyTorch Geometric [35], which is the library of choice. There are other alternatives as well, and [30] provides a good comparison among them.

Data synopsis – The K_V metric is based on Ripley's K function, used in traffic flow, neighborhood accessibility, clustering, and hotspot detection [36–40]. To address scalability, Spark-, GPU-, and heuristic-based solutions exist [41–43]. Box counts [44] have been used in query optimization [45]. Other applications of synopsis include similarity search [46]. More broadly, learned data-management systems have investigated learned synopses and data-driven models for approximate query processing, cardinality estimation, and query optimization, such as DeepDB [47] and Neo [48]. These works are related in spirit because they learn compact representations of data or query behavior, but they do not target vector geospatial datasets or spatial operations. Metric selection is application-driven; e.g., [12] develops custom metrics, and we considered only a small subset.

Clustering – This operation has a wide range of applications [49–52], and there has been several work on deep learning-based clustering [53–56]. Representative examples include Deep Embedded Clustering [57], which jointly learns feature representations and cluster assignments, and DeepCluster [53], which uses clustering assignments as pseudo-labels for representation learning. It involves different architectures, custom clustering loss functions, and other steps. We focused on supervised learning to show how different representations can replicate the operations of existing algorithms. We leave unsupervised geospatial clustering for future work.

Walkability – We study walkability because it includes spatial joins. There are many ways to measure walkability [58], and often it involves OpenStreetMap data [59, 60]. Spatial joins are expensive [45, 61–63] and are characterized by the fact that they process two datasets at a time. Our aim is to evaluate the effectiveness of different representations for capturing spatial join operations rather than advancing walkability estimation.

3 Learned geospatial data representation

Geospatial vector datasets consist of collections of records containing geometric shapes and non-geometric features. These data cannot be processed directly by existing DL architectures, and a conversion step is required to transform them into a more suitable representation. This step is not always straightforward, as it may, depending on data characteristics,

result in the loss of information that affects the quality of the predictions generated by the trained models. The remainder of this section introduces a set of preliminary definitions and then presents the three considered approaches, **GeoImg**, **GeoGraph**, and **GeoVec**, for properly representing geospatial data and for exploiting existing DL models to address a set of well-known geospatial problems. As will become evident in the following, applying these approaches requires specific preprocessing and transformation steps to obtain a representation suitable for geospatial tasks.

3.1 Preliminary definitions

This section presents a set of preliminary definitions that will be leveraged during the investigation of the generalized capabilities of a DL model in understanding and reasoning about geospatial data.

Definition 1 (Geospatial Dataset) A geospatial dataset D is a set of vector geometries with attributes defined as follows:

$$D = \{g_i \mid g_i = \langle \text{type}, \text{coords}, \text{attr} \rangle\} \tag{1}$$

where each geometry $g_i \in D$ is a tuple including the type of geometry, which could be, for instance, *"Point"*, *"LineString"*, or *"Polygon"*, an ordered list of coordinates, each one as a pair $(\text{longitude}, \text{latitude})$, representing together the position and shape of each object. Finally, each geometry could also be associated with a list of attributes *attr* from different domains, such as numerical or textual values.
In the following, the set of possible geospatial datasets represented in this form will be denoted as $\mathcal{D}$. As an example, a dataset $D \in \mathcal{D}$ containing a set of points, each associated with a temperature value, will be represented as a set $D = \{g_i \mid g_i = \langle \text{"Point"}, (\text{long}_i, \text{lat}_i), \text{temp}_i \rangle\}$.

Given the representation of a geospatial dataset, geospatial operations can be represented as analytical functions with one of the following generic signatures, involving at least one geospatial dataset D and a set of optional parameters, denoted below in square brackets, and producing an outcome that could consist in simply one or more scalar values, or in a new geospatial dataset, as formalized below.

Definition 2 (Geospatial operation) A generic geospatial operation can be represented as an analytical function with one of the following signatures:

- $f(D[, params]) \rightarrow m$: where m can be either a single real number, $m \in \mathbb{R}$, or a list of real numbers, $m \in \mathbb{R}^{d_1}$, or a k-dimensional grid of real numbers, $m \in \mathbb{R}^{d_1 \times \cdots \times d_k}$. This represents a function that processes a single spatial dataset and produces a simple outcome value.
- $f(D, D''[, params]) \rightarrow m$: where m can have one of the structures listed in the previous point. This represents a function that processes two spatial datasets and produces a simple outcome value.
- $f(D[, params]) \rightarrow D_T$: where D_T is an output geo-spatial dataset produced starting from the input ones. This represents a function that operates on two spatial datasets and produces a new spatial dataset as output.

- $f(D, D''[, params]) \rightarrow D_T$: where, as described in the previous point, D_T is an output geo-spatial dataset produced starting from the input ones, but in this case it is produced starting from the combination of two datasets D and D''.

With reference to the possible outcome produced by a geospatial operation, we can consider the cases where the target $m \in \mathbb{R}^2$ is the position of the point with the maximum average temperature, while D_T can be the input dataset D where to each geometry an additional attribute label is added that is produced by f through a clustering of D. Note that for some problems, the geometries in D_T can be the same as D, like in the mentioned clustering case, a subset of it, like in a spatial join operation, or a transformation of it, like in a reprojection operation. In all cases, the function f may also take additional parameters ([, *params*]), such as a distance threshold for clustering or a filter window for range query selection.

Our goal is to train a set of DL models that approximate analytical functions f involving complex geospatial operations and to assess their ability to understand the intrinsic characteristics of geospatial data in producing the desired results. The main objective is to investigate how a geospatial dataset can be effectively represented using one of three approaches: image-based, graph-based, or vector-based, to achieve this purpose. Indeed, current DL techniques cannot directly process a geospatial dataset D or generate target values for its geometries. For this reason, the next section discusses three possible approaches for representing a geospatial input dataset D. Notice that the choice of each of these representations also determines the design of the network architecture used to estimate the spatial operation.

3.2 GeoImg: image-based representation

This section describes our first representation approach: an image-based one. This representation is inspired by Convolutional Neural Networks (CNNs), which are commonly used for image processing. We chose this as our first approach because CNNs preserve the spatial structure of data and can learn local spatial patterns. This makes them versatile and potentially suitable for modeling a variety of spatial problems that rely on translation-invariant features. Representing our data as an image enables the use of existing CNN architectures. The simplest approach for converting our data to an image is to rasterize it. In graphics, rasterization means plotting the shapes as an image of a given resolution, similar to rendering them on a map. However, one drawback of this solution is that it loses subtle features at low resolutions, making it less versatile and better suited to smaller regions and sparse data. A common way to overcome the limitation described above and produce a rasterized version of a spatial vector dataset is to build a histogram of the dataset. Histograms work by dividing the input reference space, i.e., the geographical region, into a grid of fixed-size cells and storing a summary of the geometries that fall within each cell. Previous work used a similar representation to address specific geospatial problems, but it typically tailored the representation to a specific problem or a specific set of input datasets.

In this paper, we propose a methodology to enrich the histogram-based representation and generalize it to a variety of problems, including those involving multiple datasets and those involving thematic attributes associated with geospatial objects. More specifically, we introduce **GeoImg**, an approach for representing geospatial data as histograms and leveraging image-based DL models to address geospatial problems. The next section introduces the

characteristics of this type of representation by discussing suitable aggregate functions for summarizing the geospatial content of each cell. After defining the histograms, it becomes straightforward to utilize existing CNN architectures such as ResNet [14] and UNet [15] to encode functions in the forms described in Def. 2, representing geospatial problems.

Dealing with the **GeoImg** representation requires specifying five tasks, each one focused on the representation of (i) geometry positions, (ii) non-geometric features, (iii) an entire input dataset, (iv) multiple input datasets, and (v) target values.

3.2.1 Representing geometries

To represent geometries in a histogram, we first needed to transform the coordinates into a metric reference system, particularly when positions are represented as latitude and longitude, to ensure consistent distance computations between objects and intersection identification with grid cells. Once coordinates had been transformed into the desired metric reference system, we computed the Minimum Bounding Rectangle (MBR) of the whole dataset *D* to determine its reference space. Then, we selected a histogram size *s*, i.e., number of rows and columns, to compute the desired subdivision of the reference space into cells. Finally, we defined a spatial aggregate function $f_{sa}(D, c)$, which returned for each cell *c* an aggregate value for the geometries of *D* that intersect *c*.

Definition 3 (Spatial Aggregate Function) Given a spatial dataset $D \in \mathcal{D}$ projected into metric reference space and a grid cell $c \subset \mathbb{R}^2$, a spatial aggregate function $f_{sa} : \mathcal{D} \times \mathbb{R}^2 \rightarrow \mathbb{R}$ returns an aggregate value $v \in \mathbb{R}$ describing the geometries of *D* intersecting the cell *c*.

A possible aggregate function is $f_{Fcount}(D, c)$, which returns the number of geometries in *D* intersecting a grid cell *c* divided by the dataset cardinality $|D|$. In this case, the value of each cell in the histogram is the percentage of all geometries that intersect the range of coordinates the cell represents, and the sum is always 1.0 for points but can be larger for other geometries when some shapes intersect multiple cells. Other spatial aggregate functions could also be defined for geometries, depending on the problem the model needs to solve, particularly when the input geometries are not simple points but polygons or linestrings. Indeed, in the presence of more complex geometries, it would be useful to add information about the average area, length, or number of coordinates of the geometries of *D* intersecting the cell *c*.

Definition 4 (Positional Histogram) Given a spatial dataset $D \in \mathcal{D}$, a geometric aggregation function $f_{sa} : \mathcal{D} \times \mathbb{R}^2 \rightarrow \mathbb{R}$, and a histogram dimension $s \in \mathbb{N}$, the positional histogram

$$H_p(D) = Histogram(D, f_{sa}(), s) \tag{2}$$

is a grid covering the MBR of the dataset *D* in a metric reference space, having $s \times s$ cells each one populated by using the spatial aggregate function $f_{sa}()$.
Given the positional histogram $H_p(D)$ describing the geometries in *D*, we also needed to represent the non-geometric attributes characterizing the elements in *D*.

3.2.2 Representing non-geometric attributes

To represent the additional attributes associated with the geometries in *D*, namely, the *attr* part of the geometries in *D*, we built for each $a \in attr$ an additional histogram with the same size *s* identified for the positional histogram $H_p(D)$, but populated through a specific aggregate function. As a result, when $|attr| = j$ and we used one aggregation function per attribute, we ended up with *j* thematic histograms, which were summarized by a single stacked thematic histogram.

Definition 5 (Stacked thematic histogram) Given a spatial dataset $D \in \mathcal{D}$ where each geometry is characterized by a set of attributes $attrs = \{a_1, \ldots a_j\}$, and a set of aggregate functions $f_{a_i} : \mathbb{A} \rightarrow \mathbb{R}$ one for each attribute, where $\mathbb{A}$ denotes a generic domain for the thematic attribute values, the stacked thematic histogram

$$H_{attr}(D) = \begin{bmatrix} Histogram(D, f_{a_1}(), s) \\ \vdots \\ Histogram(D, f_{a_j}(), s) \end{bmatrix} \tag{3}$$

is obtained by stacking on top of each other a set of grids covering the MBR of the dataset *D* in a metric reference space, having $s \times s$ cells. At each level $i \in \{1, \ldots j\}$, each cell *c* is populated by applying the aggregate function $f_{a_i}()$ on the thematic attribute $a_i \in attrs$ of the geometries of *D* intersecting the cell *c*.

3.2.3 Representing a single dataset

Given the positional histogram and the stacked thematic histogram of a dataset *D*, the GeoImg representation of an entire dataset was obtained by stacking the thematic histogram on top of the positional histogram, yielding the *global histogram* of *D*.

Definition 6 (Global histogram) Given a dataset $D \in \mathcal{D}$ together with its positional histogram $H_p(D)$ and its stacked thematic histogram $H_{attr}(D)$, the global histogram *H*(*D*) of *D* is obtained by stacking $H_{attr}(D)$ on top of $H_p(D)$:

$$H(D) = \begin{bmatrix} H_{attr}(D) \\ \vdots \\ H_p(D) \end{bmatrix} \tag{4}$$

In this way, the final representation of an input dataset $D \in \mathcal{D}$ is a $(j+1)-$dimensional stacked histogram $H(D) \in \mathbb{R}^{(j+1)\times s \times s}$. This histogram could be represented as a tensor T_H with $shape(T_H) = (j+1, s, s)$, which could be directly given as input to the considered DL model.

3.2.4 Representing multiple datasets

In the case where the geospatial operation to be considered requires multiple input datasets $D_1, \ldots, D_d$, we first needed to create a global histogram for each dataset D_i based on the

global metric reference space, i.e., the MBR of the union $D_1 \cup \cdots \cup D_d$, and by using the same histogram size s. Given all these global histograms, they were combined into a unique global histogram $H(D_1, \ldots, D_d)$ by stacking them on top of each other. This operation was quite straightforward, since they were all at the same resolution and shared the same reference space.

Definition 7 (Global compound histogram) Given a set of datasets $D_1, \ldots, D_d$ together with their global histograms $H(D_1), \ldots, H(D_d)$ all computed with respect to the same global metric reference space and by using the same size s, the global compound histogram $H(D_1, \ldots, D_d)$ summarizing all of them is obtained by stacking them on top of each other:

$$H(D_1, \ldots, D_d) = \begin{bmatrix} H(D_1) \\ \vdots \\ H(D_d) \end{bmatrix} \tag{5}$$

For example, given two input datasets D_1 and D_2, each one characterized by geometries with j_1 and j_2 thematic attributes, respectively, and a histogram size equal to 64, we will have a global histogram $H_1 \in \mathbb{R}^{(j_1+1)\times 64\times 64}$ for D_1, and $H_2 \in \mathbb{R}^{(j_2+1)\times 64\times 64}$ for D_2. Given that, we can combine them into one global compound histogram $H(D_1, D_2) \in \mathbb{R}^{(j_1+j_2+2)\times 64\times 64}$, which can be represented as a tensor T_H with $shape(T_H) = (j_1 + j_2 + 2, 64, 64)$.

3.2.5 Representing target values

As discussed in Def. 2, the expected output of a spatial operation can be (i) a vector *m* of numbers (i.e., $m \in \mathbb{R}^{d_1}$), (ii) a *k*-dimensional grid *m* of numbers (i.e., $m \in \mathbb{R}^{d_1 \times \cdots \times d_k}$) or (iii) a target dataset of geo-spatial objects D_T with a maximum number of output values v_{max}.

In the first two cases, the output can be generated, as a resulting histogram, directly by the model itself, since a CNN typically produces a list of values as results. The output will be essentially presented as a tensor *T* with $shape(T) = (d_1)$, or $shape(T) = (d_1, \ldots, d_k)$, respectively.

However, in the third case, an output for each target geometry in D_T must be generated. Notice that the dimension of the output $|D_T|$ does not necessarily need to be the same as the input $|D|$, even though it can be the same for many applications. Suppose that $|D| = |D_T|$ and for each geometry in input the output is a vector of features $(v_1, \ldots, v_{max})$. In this case, the model will build a new output histogram for each feature, where each histogram cell *c* summarizes the output feature for the geometries of *D* that intersect *c*. The final result can then be represented by a tensor *T* with $shape(T) = (|D_T|, v_{max})$, obtained by computing the target values for the geometries in each cell.

3.2.6 Model architecture

As previously mentioned, we used CNN-based architectures for dealing with image-based representation. The specific architecture depends on the type of desired target. In particular, for problems with a fixed target size $m \in \mathbb{R}^{d_1}$ or $m \in \mathbb{R}^{d_1 \times \cdots \times d_k}$, we used a ResNet [14] architecture, like the one shown in Fig. 2. In this example, the network takes our histo-

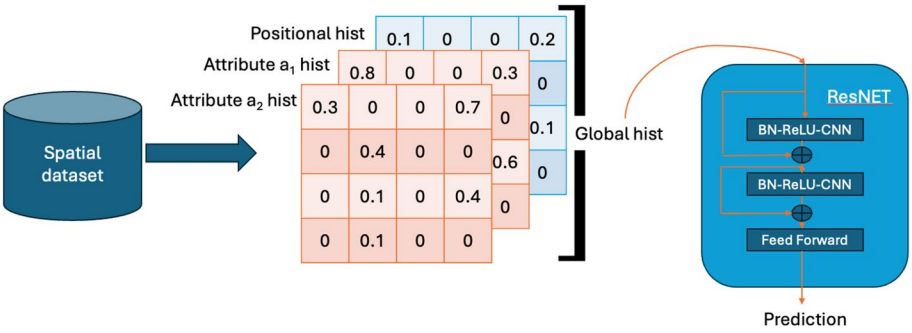

Fig. 2 Architecture of the GeoImg model (ResNET will be substituted by UNet when the output is a dataset of geospatial objects)

gram representation of the input dataset D and produces a fixed-size output. Conversely, for problems with variable output size, namely those that produce a target dataset of geospatial objects D_T, we used a UNet [15] architecture, and the target is represented, as discussed earlier. The first part of a UNet is similar to a ResNet, but it has additional layers that perform up-convolution to produce an image output. UNet architectures are typically used for problems such as image segmentation [15]; here, we used them to estimate quantities based on the geographic positions of geometries.

While the representation of geospatial datasets through our model proposed in this section can capture more information than rasterizing the geometries alone, it still has drawbacks. One drawback is that it cannot capture small patterns that are smaller than the cell size. Also, all geometries within the same cell will have the same output features. For these reasons, we considered alternative approaches with other model types, i.e., GeoGraph and GeoVec, that avoid this type of data loss.

3.3 GeoGraph: graph-based representation

This section introduces a second representation approach that is tailored for graph-based DL architectures. Specifically, we focused on point-set-based approaches that build a nearest-neighbor graph at run-time.

This architecture operates directly on a set of points and is inspired by point-cloud-based models, e.g., PointNet++ [16]. A graph-based representation provides a richer description of the space and enables the model to capture local relationships among points more accurately than an image-based approach does. The same properties that enabled this approach to be successful in the point-cloud domain can also be beneficial in the geospatial domain. Additionally, one advantage of this approach over GeoImg is that it works directly with variable-sized datasets D, whereas an image-based approach represents any input dataset D using a fixed-size histogram, potentially leading to information loss. Therefore, this approach does not require a heavy pre-processing step. However, geospatial problems exhibit characteristics that differ significantly from those typically addressed by generic point-set learning approaches. These differences arise from the inherent relationship between geographic data and the Earth's surface, as well as from the richness and complexity of spatial representations. Indeed, in conventional point-set models, each node generally corresponds to a point

within a global spatial structure. In contrast, geospatial applications often require nodes to represent spatial entities with potentially complex geometries, embedded in a specific geographic reference system, while edges encode spatial relationships and interactions among such entities.

To the best of our knowledge, point-set-based architectures have not yet been systematically investigated for the joint representation of geospatial geometries and spatial operations. The remainder of this section introduces GeoGraph, a possible point-set-based representation framework for vector geospatial data.

3.3.1 Representing geometries

In this approach, a geospatial dataset D is represented as a graph, where each node corresponds to a geometry $g_i \in D$ and edges represent neighbor relationships. Clearly, as in the GeoImg approach, a transformation of the coordinates to a metric reference system was also needed, along with normalization of each coordinate to [0..1], to correctly and efficiently represent distance and position properties.

The original PointNet++ model has been tailored to represent a set of points, so when the dataset D contains only geometries of type "*Point*", the representation of D in this model is straightforward. Indeed, in the case $|D| = n$, nodes were overall represented as a Tensor $G \in \mathbb{R}^{n\times 2}$, where each point is given by a tuple whose first element is the longitude, and the second is the latitude, both transformed in the metric reference space and normalized between zero and one

However, geospatial datasets may contain other geometry types, and in this case, it is necessary to encode the input to make the use of the PointNet++ model feasible. In particular, in the proposed GeoGraph model, each geometry $g_i \in D$ was represented by its centroid point, to preserve location information, together with additional geometry descriptor attributes that captured the object's shape and extent. Examples of such geometry descriptors include the area, length, and width of a geometry, or, more generally, the coordinates of the Minimum Bounding Rectangle (MBR) of the geometry, as proposed in this paper. This made the representation of each geometry $g_i \in D$ as a tensor $G \in \mathbb{R}^{n\times 6}$, where the first two elements represented the centroid coordinates, while the last four elements described the coordinates of its MBR, all transformed and normalized.

Definition 8 (Geometry position tensor) Given a dataset $D \in \mathcal{D}$ containing *n* geometries whose coordinates have been translated into a metric reference space and normalized between 0 and 1, its content is represented by a geometry position tensor *G* of size $\mathbb{R}^{n\times 6}$

$$G = \{\langle long^i, lat^i, mbr^i_{x_0}, mbr^i_{y_0}, mbr^i_{x_1}, mbr^i_{y_1}\rangle\}^n_{i=1} \tag{6}$$

where for each row *i* the first two elements represent the centroid of the geometry $g_i \in D$ if the type of g_i is different from "Point", or the coordinates of g_i otherwise, while the other four elements represent the coordinates of the MBR of g_i.

Let us notice that with respect to the GeoImg, in this case, all geometries in *D* have a direct representation in the tensor *G*, whose dimension depends on the number *n* of geometries in *D*, while in GeoImg the size of the input does not depend on the number of input geometries but on the fixed and predefined histogram size *s*.

3.3.2 Representing non-geometric attributes

Given a dataset D, additional attributes associated with its geometries can be represented as a thematic tensor $A \in \mathbb{R}^{n \times j}$, where n is the cardinality of D and j is the number of thematic attributes, so each row contains the list of attribute values associated with the corresponding geometry g_i in G. Pre-processing can include normalizing features within each dataset D independently or using global statistics across a collection of datasets of interest.

Definition 9 (Attribute tensor) Given a dataset $D \in \mathcal{D}$ containing n geometries, each one characterized by j thematic attributes $attrs = \{a_1, \ldots, a_j\}$, the values of these attributes across the geometries $g_i \in D$ are represented by an attribute tensor A of size $\mathbb{R}^{n \times j}$:

$$A = \{\langle a_1^i, \ldots, a_j^i \rangle\}_{i=1}^{n} \tag{7}$$

3.3.3 Representing a single dataset

Given a geospatial dataset D together with its position tensor G and attribute tensor A, the GeoGraph representation of D is given by the pair of these two tensors, which together will represent the two inputs for a GeoGraph model.

Definition 10 (Graph dataset tensor) Given a dataset $D \in \mathcal{D}$, its graph dataset tensor is given by the pair: $\langle G, A \rangle$, where G is the geometry position tensor of D and A is its thematic attribute tensor.

3.3.4 Representing multiple datasets

In case the geospatial operation of interest requires multiple input datasets $D_1, \ldots, D_d$, GeoGraph simply requires generating a geometry position tensor G and an attribute tensor A for each one of the input datasets. However, in this case, the positions must all be normalized using the MBR of the union of all input datasets before generating the position and attribute tensors. The model will then accept multiple position and attribute tensors and join them appropriately, based on the considered operation, as we will describe in Sec. 3.3.6.

3.3.5 Representing target values

As for the previous model and as discussed in Def. 2, the target value m can be (i) a vector of numbers ($m \in \mathbb{R}^{d_1}$), (ii) a k-dimensional grid of numbers ($m \in \mathbb{R}^{d_1 \times \cdots \times d_k}$) or a target dataset of geo-spatial values D_T.

In the case of a single aggregate prediction for the entire dataset D, the target becomes a tensor m with a fixed shape. Depending on the range of values, we might need to normalize them to improve the model's performance. If we had a target set of geometries, D_T, we needed to represent them similarly to the positions tensor G for the input dataset. In particular, the target tensor M would have its first dimension equal to the cardinality of D_T, while the other dimensions would be fixed, similar to m, depending on the shape of the target values. As for the previous cases, the number of geometries represented in D_T can be

exactly the same as the number of geometries in D, but this is not always necessary, since the specific problem can generate values for geometries different from those in the input.

3.3.6 Model architecture

The model architecture is slightly different depending on the number of input datasets and the shape of the target value. An example of the architecture of this model for problems with a single input dataset and a fixed target size m is shown in Fig. 3, where to avoid cluttering the notation, a dataset composed of geometries of type "*Point*" is assumed, so no further information about the geometry's MBR is needed in the geometry position tensor G. The model architecture takes as input the position tensor G and the eventual attribute tensor A and consists of four main layers.

The core component of the model is the *Set Abstraction* (SA) layer, which hierarchically extracts local spatial features from the input point set. Each SA layer consists of three main steps.

(i) First, a sampling step selects a subset of points to be propagated to the next layer. Sampling is performed using Farthest Point Sampling (FPS), which iteratively selects points that maximize the distance from previously selected points, ensuring representative coverage of the spatial domain.

(ii) Second, a neighborhood grouping step identifies the local region associated with each sampled point. This operation is implemented using a range query parameterized by a search radius and a maximum number of neighbors, which collects nearby points around each centroid.

Third, the coordinates and attributes of the points belonging to each neighborhood are processed by a multilayer perceptron (MLP), which constitutes the only trainable component of the SA layer. The MLP learns a latent representation that captures local geometric and semantic patterns, and the resulting features are aggregated to produce a compact

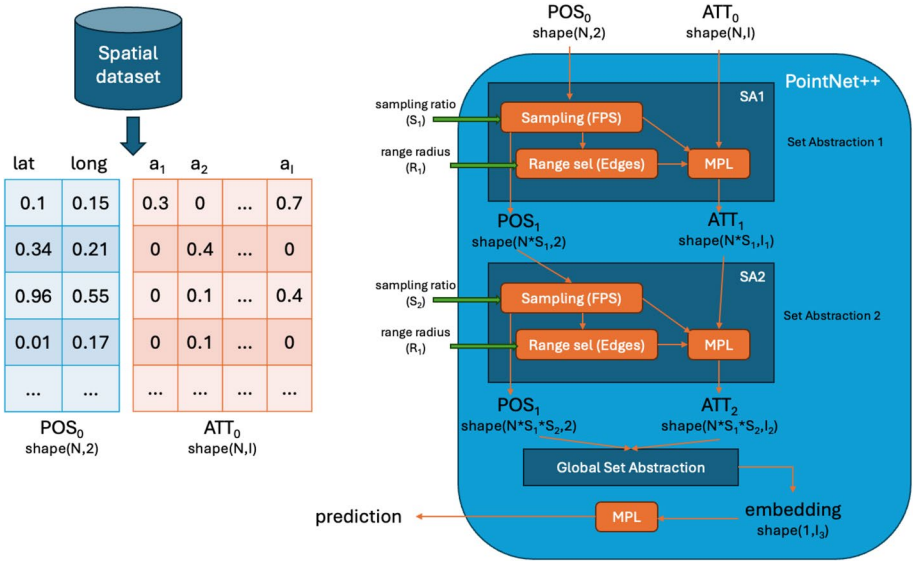

Fig. 3 Architecture of the GeoGraph model

description of each sampled neighborhood. The output of an SA layer consists of the sampled points together with their learned feature representations. Multiple SA layers can be stacked hierarchically to capture increasingly larger spatial contexts. The final SA layer is replaced by a *Global Set Abstraction* layer, which aggregates information from the entire input dataset into a single global representation. This representation is then fed to a final MLP that produces the target output m.

The "Set Abstraction" layers enable the model to work with variable-sized inputs. The first abstraction layer summarizes more localized information, while subsequent abstraction layers summarize information at a more global scale. What differentiates an SA layer from a traditional sampling operation is that the latter simply selects a set of representative points, whereas the former can identify the local structure of the sampled points.

To handle multiple input datasets, an initial SA layer was added to join them. The idea was to use one dataset as the reference points and the other as the sampled points. From this layer, we obtained a set of features for each point in the second dataset, based on its nearest points in the first dataset.

For problems with a target value represented by a dataset D_T, additional layers are used at the final stage. These layers are called "Feature Propagation Layers" and they work by reversing the process, generating new features for the larger point set in the previous layer from the features of the points in the lower layer, until producing one prediction for each point in the target dataset. The features are interpolated by finding the k-nearest neighbors among the points from the previous step and using an MLP to generate a new feature for the target point. The reader can refer to [16] for more details about this architecture.

3.4 GeoVec: vector-based representation

This section introduces the third considered approach, which is based on a Transformer architecture [18]. Transformers are mainly used for natural language processing (NLP) applications, but have been successfully applied to a variety of tasks, including geospatial data [64]. However, there are still challenges with its generalization to different types of geospatial problems, and it is not clear how it compares with other approaches. In general, a transformer-based approach requires less pre-processing and is more flexible than GeoImg. However, it is less flexible than GeoGraph, since it imposes a fixed upper bound on the number of geometries.

The remainder of this section introduces **GeoVec**, a Transformer-based architecture, where geo-spatial objects are represented as vectors. Like the other models, it is characterized by the components described below.

3.4.1 Representing geometries

In NLP, words are typically represented using word vectors and positional encoding, preserving relationships among words. Similarly, for geospatial objects, we had to use embeddings that preserve their spatial relationships. To achieve this, we used Poly2Vec [17], a recently proposed geospatial encoder that produces vector embeddings that ensure three important geospatial properties: shape preservation, direction preservation, and distance preservation, while maintaining task flexibility. The core idea of Poly2Vec is to represent a geometry g as a complex boundary signal: $g(t) = x(t) + iy(t)$ and compute its Fourier

coefficients $F_g(k)$ based on its discretized boundary (i.e., $k \in \{1, \ldots, n\}$ where n is the number of vertices of g). Each Fourier coefficient $F_g(k) \in \mathbb{C}$ corresponds to a harmonic frequency k and can be decomposed into two real-valued components: the magnitude z, which reflects the geometry size and the overall shape, and the phase $\boldsymbol{\phi}$, which encodes the positional and orientational information of the geometry [65]. At the same time, since any ML model typically requires a fixed-size upper bound e for the input sequence, only a subset of the frequencies $\mathcal{K}$ is retained, which can be different from the number of vertices of g. This operation can be interpreted as a form of spectral sampling, and this is one of the reasons Poly2Vec can produce fixed-dimensional embeddings regardless of the number of vertices.

Given this intuition, the geometry embedding can be computed as defined below.

Definition 11 (Geometry embedding) Given a geometry g represented as a sequence of 2D vertices $(x_1, y_1), \ldots, (x_n, y_n)$,

- g is modeled as a complex boundary signal: $g(t) = x(t) + iy(t)$ and its Fourier coefficients $\mathrm{F}_g = \{F_g(k)\}_{k=1}^{n}$ are computed.
- A subset $\mathcal{K} \subseteq \{1, \ldots, n\}$ of selected frequencies is identified as a fixed chosen hyperparameter, and F_g of g can be represented as a vector of $|\mathcal{K}|$ complex coefficients corresponding to selected frequencies.
- F_g can be decomposed into two real-valued vectors: $\mathrm{z} = \{z_k\}_{k \in \mathcal{K}}$, representing the magnitude, and $\boldsymbol{\phi} = \{\phi_k\}_{k \in \mathcal{K}}$, representing the phase.
- The two vectors z and $\boldsymbol{\phi}$ are concatenated and passed to a final MLP $h : \mathbb{R}^{2 \times |\mathcal{K}|} \rightarrow \mathbb{R}^e$ to produce the final geometry embedding with dimension e: $v = h([\mathrm{z}, \boldsymbol{\phi}]) \in \mathbb{R}^e$.

Notice that the final MLP acts as a learnable projection from the spectral feature space $\mathbb{R}^{2 \times |\mathcal{K}|}$ to a latent space $\mathbb{R}^e$, allowing dimensionality reduction or expansion depending on the chosen architecture. However, $e < 2|\mathcal{K}|$ is typically the case, since its objective is to learn a more compact representation that removes redundancy and optimizes downstream tasks. This is particularly true for complex geometries, such as polygons or linestrings, but does not hold for point datasets. In this way, Poly2Vec can handle variable-length vertex sequences and transform them into a fixed-dimensional embedding.

Poly2Vec is a geometry encoder, not a sequence model, so it processes one geometry at a time, and it does not model interactions between geometries. However, given the embedding of a single geometry, we can also consider the embedding of an entire dataset D. The idea is to combine the set of vector representations $v_i \in \mathbb{R}^e$ of each geometry g_i into an embedding matrix $M = \{v_i \in \mathbb{R}^e\}_{i=1}^{n}$ where n is the number of geometries in D. The embedding matrix M representing the overall spatial dataset will serve as input to a subsequent Transformer model that encodes the spatial operation. However, since a Transformer architecture has a fixed upper bound b for the input sequence, it was necessary to perform padding if $n < b$ or sampling if $n > b$. Therefore, the final dataset embedding passed to the Transformer model had a fixed size of $b \times e$.

Definition 12 (Spatial dataset embedding) Given a dataset $D \in \mathcal{D}$ composed of n geometries $D = \{g_1, \ldots, g_n\}$, and a fixed upper bound $b \in \mathbb{N}$ for the input sequence, its encoding is an embedding matrix $M \in \mathbb{R}^{b \times e}$ where each row $i \in \{1, \ldots, b\}$ is the vector encoding $v_i \in \mathbb{R}^e$ of the i-the geometry in D:

$$M = \begin{bmatrix} v_1 = h([\mathrm{z}_1, \boldsymbol{\phi}_1]) \\ \vdots \\ v_b = h([\mathrm{z}_b, \boldsymbol{\phi}_b]) \end{bmatrix} \tag{8}$$

3.4.2 Representing non-geometric attributes

The thematic attributes are represented similarly as discussed in Section 3.3.2 for GeoGraph. However, this step only includes the attributes of the sampled objects. After this step, the attributes tensor is always $A \in \mathbb{R}^{b \times j}$, where j is the number of attributes and b is the number of sampled geometries. In addition, the values are normalized similarly to GeoGraph.

3.4.3 Representing a single dataset

Given a geospatial dataset D whose geometries are represented by a spatial dataset embedding M and their corresponding thematic attributes are given by a tensor A, the global spatial embedding was obtained by concatenating the two, so that the input to the Transformer model became a tensor T of shape $shape(T) = (b, e + j)$.

Definition 13 (Global spatial embedding) Given a dataset $D \in \mathcal{D}$ together with its spatial embedding $M \in \mathbb{R}^{b \times e}$ and its attribute tensor $A \in \mathbb{R}^{b \times j}$, the global spatial embedding $V \in \mathbb{R}^{b \times e+j}$ of D is obtained as:

$$V = M||A = \begin{bmatrix} v_1 = h([\mathrm{z}_1, \boldsymbol{\phi}_1]), a_1^1, \ldots, a_j^1 \\ \vdots \\ v_b = h([\mathrm{z}_b, \boldsymbol{\phi}_b]), a_1^b, \ldots, a_j^b \end{bmatrix} \tag{9}$$

This matrix was also padded with additional zero columns to ensure the final number of columns is a multiple of the number of attention heads, a requirement of this architecture.

3.4.4 Representing multiple datasets

For multiple input datasets $D_1, \ldots, D_d$, we first merged them into a single dataset and added, among its thematic attributes, a one-hot-encoded label indicating which dataset each record belongs to. Then, we performed stratified sampling based on the dataset labels, ensuring the dataset proportions were maintained and that the number of rows in the matrix embedding remained equal to b, while the number of columns was equal to the total number of attributes plus one to store the dataset label. Let us consider a geospatial operation involving two input datasets, D_1 and D_2, each with j_1 and j_2 attributes, respectively, encoded in the attribute tensors A_1 and A_2. Given that the cumulative attribute tensor A will contain an element for each attribute in A_1 and in A_2, plus an additional label representing the dataset identifier. It follows that $A \in \mathbb{R}^{b \times (j_1 + j_2 + 1)}$.

The final input to the model after encoding becomes of the form $\mathbb{R}^{b \times (e + j_1 + j_2 + 1)}$. The same consideration made above about the column padding also holds here.

3.4.5 Representing target values

As reported at the end of Sec. 3.1, the expected output m can be (i) a vector of numbers ($m \in \mathbb{R}^{d_1}$), (ii) a k-dimensional grid of numbers ($m \in \mathbb{R}^{d_1 \times \cdots \times d_k}$) or a target dataset of geo-spatial values D_T. In this approach, fixed targets can be generated directly. However, when a target set of geometries, D_T, is specified, this is not as straightforward, particularly when the global embedding V contains a subset of the input geometries, namely, its number of rows b is less than $|D|$.

In order to produce as a target value a set of geospatial objects D_T, we distinguished the case in which the geometries of D_T match those in D, from the case in which the two sets of geometries are different. In the first case, the procedure assigned to each geometry g_i in D_T the set of values $\{v_1, \ldots, v_{max}\}$ that the model has assigned to the nearest sampled geometries g_j passed to the Transformer, if the encoding of g_i is not contained in M, or exactly the values assigned by the model if g_i is part of the sampling. In the second case, the approach was quite similar, and the idea was to use the label of the nearest geometry passed to the Transformer. The only difference is that in this case, no exact matching could be identified.

3.4.6 Model architecture

In this architecture, first, the geometries were passed to the Poly2Vec encoder. Then, the embeddings were concatenated with the non-geometric attributes. This was then fed to a Transformer model. An example of this architecture is shown in Fig. 4. The Transformer model's architecture has been intentionally kept generic and can be tailored for each geo-spatial operation of interest.

4 Geo-spatial problems

This section discusses the three target geo-spatial problems that we considered. For each of them, we provided the problem definition and the input and output formats for each specific model. The goal was to show that the proposed models can be applied to a variety of spatial

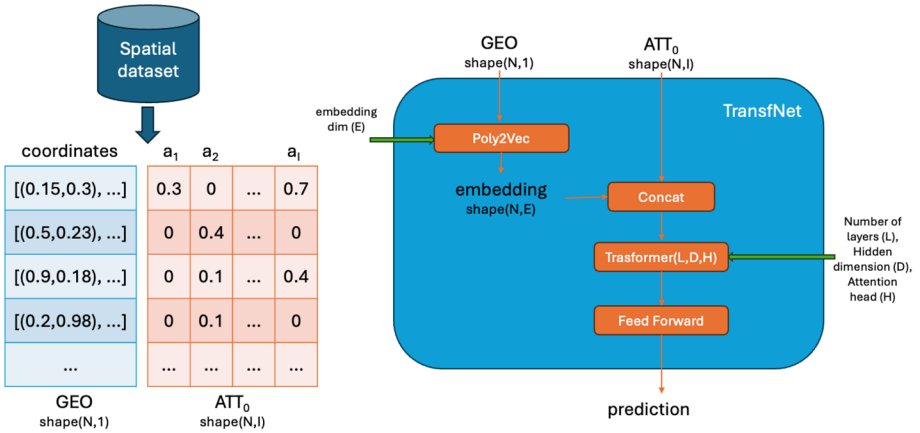

Fig. 4 GeoVec architecture

data analysis problems and can capture a wide range of spatial data characteristics, e.g., proximity and data density, producing approximate solutions faster than exact solutions. In particular, with reference to the classification made in Sect. 3.1, these three operations provide a meaningful example of each of them in terms of the number and types of both the input parameters and the target values.

Table 1 summarizes the three problems and the corresponding proposed models, as we further detail in the rest of this section.

4.1 Spatial data synopsis

Spatial analytics often require the computation of aggregate values on spatial data, called **synopsis**, that summarize different spatial features of a given dataset, e.g., skewness, symmetry, coverage of geometries, or density analysis for specific thematic attributes. However, as data sizes grow very large, as is typical for geospatial datasets, computing these characteristics in an exact way may become very expensive. This spatial operation can be considered a good representative of the first class of geospatial operation introduced in Def. 2, namely the ones that receive in input a dataset D together with some optional parameters and produces a vector of numbers $m \in \mathbb{R}^{d_1 \times \cdots \times d_k}$.

4.1.1 Problem definition

In this paper, we considered three categories of synopsis: (i) Finding extreme values and the regions where they are located (this can require a preliminary phase to calculate the k nearest neighbors for all geometries in the dataset and to compute the average of the thematic attribute among its neighbors). (ii) Computing the K value (K_V) of a dataset, which is a measure of the connectivity of a set of geometries, given a maximum distance r. This value is also very expensive to compute: when K_V is large, most of the geometries in the set are very close to each other. When it decreases, it means the geometries are farther apart, and we expect them to be sparsely distributed throughout the reference space. (iii) Computing the box-count values, $E_0(D)$ and $E_2(D)$, as defined in [44], that are good descriptors of the skewness of the data distribution in the spatial reference space. These three synopsis operations can be formally described as follows.

Definition 14 (Extreme values identification) Given a dataset $D \in \mathcal{D}$ of point locations characterized by a property value v, $D = \{g_i = \langle Point, p_i, v_i \rangle\}_{i=1}^{n}$, and a parameter k, the average value of the thematic attribute on the k-nearest neighbors of each geometry $g_i \in D$ is computed starting from the set:

$$avgAtt(D, k) = \{(p_i, avg(p_i))\}_{i=1}^{n}$$

where:

$$avg(p_i) = \frac{1}{k} \sum_{x \in kNN(D,\, p_i)} x.v_i$$

Table 1 Problems definition summary. Notice that the final output of the cluster problem, for GeoImg and GeoVec approaches, is obtained by applying a last conversion of the model output to the target setD_T

Approach	Model Name	Input Tensors	Output Shape
Problem Synopsis:$m = f(D, k, r)$			
GeoImg	GI-S	H,shape$(H) = (3, 64, 64)$	(9)
GeoGraph	GG-S	$G, shape(G) = (\|D\|, 2)$	
		$A, shape(A) = (\|D\|, 1)$	
GeoVec	GV-S	$V, shape(V) = (1024, 32 + 1)$	
Problem Clustering:$D_T = f(D, \varepsilon_1, \varepsilon_2, MP)$			
GeoImg	GI-C (GI-CP)	H,$shape(H) = (3, 64, 64)$	$(3, 64, 64) \rightarrow (\|D_T\|, C)$
GeoGraph	GG-C (GG-CP)	$G, shape(G) = (\|D\|, 2)$	$(\|D_T\|, C)$
		$A, shape(A) = (\|D\|, 1)$	
GeoVec	GV-C (GV-CP)	$V, shape(V) = (1024, 32 + 1)$	$(1024, C) \rightarrow (\|D_T\|, C)$
Problem: Walkability:$m = f(D, D'')$			
GeoImg	GI-W	H,$shape(H) = (11, 64, 64)$	(1)
GeoGraph	GG-W	$G(D_1), shape(G(D_1)) = (\|D_1\|, 2)$	
		$G(D_2), shape(G(D_2)) = (\|D_2\|, 2)$	
		$A(D_2), shape(A(D_2)) = (\|D_2\|, 9)$	
GeoVec	GV-W	$V, shape(V) = (1024, 32 + 10)$	

and $x \in kNN(D, p_i)$ identifies the k nearest neighbor of each p_i in D. Given *avgAtt*(D, k), the following four values are extracted: the maximum and minimum of avg_i and their respective location p_{max}, and p_{min}.

Definition 15 (K value computation) Given a dataset $D \in \mathcal{D}$ of point locations characterized by a property value v, $D = \{g_i = \langle Point, p_i, v_i \rangle\}_{i=1}^{n}$, the computation of K_V of D with radius ρ is as follows:

$$K_V(D, \rho) = \frac{1}{n \times (n-1)} \sum_{p_i \in D} |range_\rho(D, p_i)|$$

where the function $range_\rho(D, p_i)$ computes the range query for the point geometry p_i of g_i given a window ρ, and the $|\cdot|$ operator returns the cardinality of a set.

Definition 16 (Box counting) Given a dataset $D \in \mathcal{D}$ of point locations characterized by a property value v, $D = \{g_i = \langle Point, p_i, v_i \rangle\}_{i=1}^{n}$, the computation of $E_0(D)$ and $E_2(D)$ works by dividing the space into a grid with cells of side length r, and counts the number of cells in the grid that intersect at least one point in the set (for $E_0(D)$) and the number of points intersecting each cell (for $E_2(D)$). It keeps increasing the number of cells in the grid until the counts stop changing. The slope of the plot in log scale of the counts function of the side length r, can give an approximate measure of the skewness in the spatial distribution of geometries.

These problems help us evaluate whether the models can predict not only a numeric aggregate value but also properties that depend on the spatial distributions of the geometries and their relative positions and distances. Moreover, they are all expensive to compute and also capture complex spatial behaviors.

4.1.2 Target values

Considering the classification proposed in Section 3.1, for spatial synopsis, the target value was computed as an array m:

$$m = [(avg_i)_{max}, (avg_i)_{min}, p_{max}, p_{min}, K_V(D, r), E_0(D), E_2(D)] \tag{10}$$

These are a total of nine values since p_{max} and p_{min} are represented by two coordinates each. Therefore, the shape of the tensor representing the target m is (9), as shown in Table 1 and can be easily and directly produced by each model without any additional final transformation.

4.1.3 Input values

For this kind of problem, the input datasets containing 2D points were synthetically generated using the Spider Generator [66], producing datasets with various distributions and including values of a thematic attribute. Figure 5 shows some examples of the generated data, where the color associated with each point denotes a different value of the thematic

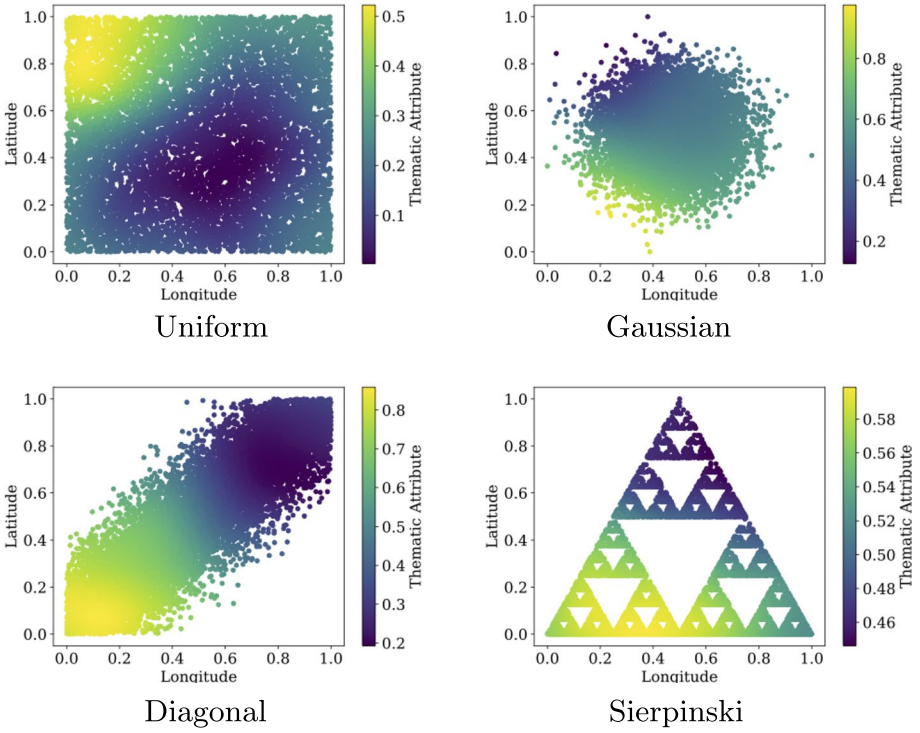

Fig. 5 Example of synthetic datasets with both different spatial distributions and distributions of thematic attributes

attribute. Given the generated datasets, we computed the target values described above by applying the algorithms that represent the exact solution, thereby building the ground truth for model training.

We then processed the data using the approaches illustrated in Section 3 and built three models: GI-S for the GeoImg representation, GG-S for the GeoGraph representation, and GV-S for the GeoVec representation. The shapes of the input and target tensors depend on the model, as summarized in Table 1. In particular, for GI-S, assuming a histogram size s of 64 and the additional computation of two aggregation functions for the non-geometric thematic attribute, f_{Fmax} and f_{Fmin}, we obtained an input histogram $H \in \mathbb{R}^{3\times 64\times 64}$. Conversely, for GG-S, the coordinates of the points were represented by a tensor G with shape $(N, 2)$, while the thematic attributes were represented by a tensor A with shape $(N, 1)$, where N represents the cardinality of the dataset D. Finally, for GV-S, we assumed that an initial sampling (or padding) phase is applied to the Transformer model, with upper bound $b = 1024$, and that the embedding size was fixed to $e = 32$, obtaining an input tensor V with shape $(1024, 32 + 1)$.

4.2 Spatial clustering

Clustering is one of the most commonly used techniques for spatial analysis. It is used to group objects based on their proximity or density, and it allows the identification of hidden

structures or the detection of regions where specific events are concentrated. Besides the practical importance and popularity of this operation, another motivation for considering this problem is that it requires predicting all points in the input, rather than a single prediction for the entire input, so it is a good representative of spatial operations with signature $f(D, [params]) \rightarrow D_T$.

4.2.1 Problem definition

DBSCAN is one of the most popular clustering algorithms. It has two main parameters, a distance ε and a minimum number of points μ. It starts by selecting a random point p and determining if it is a *core point*, i.e., if it has at least μ neighbors within ε distance from it. If p is a core point, it is used to initialize a new cluster C_p, which is expanded until no new density-reachable points can be added. In particular, every point q that is in the ε-neighbor of p is added to C_p; if q is a core point, also the ε-neighbor of q is added to C_p. This process continues until no other points can be added to the cluster C_p, namely, the cluster contains all points density-reachable from p. The process repeats by creating new clusters until no more *core points* are found. Points that do not end up associated with any cluster are labeled as *noise points*, or outliers. Many variants of DBSCAN have been defined in the literature. In particular, ST-DBSCAN [67] is an extension tailored for dealing with spatio-temporal data. In this case, two distance thresholds, ε_1 and ε_2, are defined, one for the spatial and one for the time dimension. However, the second distance is not strictly required to be related to the time dimension but could be associated with any other attribute. In the original paper [67], the authors also conducted experiments in which the second distance corresponds to the temperature measured at each point, and the algorithm detected clusters based on spatial distance, as well as an additional attribute such as temperature or time. We defined this problem using this generic definition. The objective is to build models that closely match the clustering labels produced by this algorithm, given specific parameters, since our focus is on supervised learning.

4.2.2 Target values

For this problem, the input was a single dataset D together with three parameters, ε_1, ε_2, and μ, that governed the clustering operation. All geometries in D were assumed to be of type *Point* and to have an associated attribute representing, for example, the measured temperature. The target value in this case was a dataset D_T containing the same geometries of D but enriched with a set of values $\{v_1, \ldots, v_{|C|}\}$ representing the probability to belong to each cluster in C. Considering the formalization in Def. 2, for this problem, the target value was a dataset $D_T \in \mathbb{R}^{|D_T| \times |C|}$, where C is the maximum number of clusters, since the labels were one-hot-encoded during the training process. In other words, the function approximating the geospatial operation generated one prediction for each point in D_T, i.e., the probability to belong to each cluster in C.

Referring to the notation in Table 1, we can observe that while for the GeoGraph, the corresponding model was able to directly produce the target value with the desired shape, for the other two representation approaches it was necessary to apply a final transformation that, starting from the histogram (i.e., GeoImg) or the vector (i.e., GeoVec) representation, identified the corresponding values for each geometry in D. As already mentioned in Sect.

3.2.5, in the case of GeoImg representation, the value assigned to each geometry is the one predicted for the corresponding histogram cell. Conversely, as mentioned in Sect. 3.4.5, in the case of a GeoVec representation, the value was assigned based on the neighbor proximity with respect to the sampled geometry.

4.2.3 Input values

For this problem, we prepared a collection of datasets based on the Global Historical Climatology Network (GHCN)-Daily [7], which contains the average maximum temperature for each weather station worldwide for each month. The ground-truth values were generated using ST-DBSCAN. We then built six models for this problem. Two models for each representation, with one being parameterized, namely, approximating a function $f(D, \varepsilon_1, \varepsilon_2, \mu)$ where ε_1, ε_2, and μ are the clustering parameters and, and one trained on fixed parameters, namely, receiving only the dataset D, since the other parameters were predefined and fixed. The models were labeled GI-C, GG-C, and GV-C, for the ones with fixed parameters, and GI-CP, GG-CP and GV-CP for the corresponding parametrized versions. Therefore, the shape of the tensor representing the target depends on the model as summarized in Table 1, where for the GI-C* the shape of the input tensor (3,64,64) was given by the choice of a histogram size of 64 and the presence of two aggregate functions for non-geometric thematic attributes, $f_{F_{max}}$ and $f_{F_{min}}$. Conversely, for the GG-C*, we had two input tensors: one for the geometry G and one for the thematic attributes A, whose shapes depended on the dataset dimension $|D|$. Finally, for GV-C*, the input was again set to the upper bound of the sampling phase ($b = 1024$) and the embedding size ($e = 32$).

4.3 Walkability estimation

The last problem that we considered is for walkability estimation. The motivation for considering this problem was its complexity: it requires a spatial join across two datasets, followed by an aggregation step. Indeed, with reference to Def. 2, it is a good representation of a function $f(D, D''[, params]) \rightarrow m$, where m is the walkability score.

4.3.1 Problem definition

We defined walkability as the accessibility of various amenities and services from any location within a given geographic region. Each geographic region was given a score from 0 to 1, with 0 representing non-walkable regions. The calculation involves two different sets of points. The first set represented nodes in the road network, but only for pedestrian-friendly roads. The second set included points of interest (POIs) within the same region, from a fixed set of 9 categories, such as shopping, restaurants, etc. The main idea was that a point on the road network within very close proximity to at least one POI in each of these categories was considered to have a perfect walkability score. The score for a region was simply the average of all points within its road network.

The first step in computing the score was finding the nearest point from each POI category for every node in the road network and storing the distances. The score for a specific node in the network was computed based on these distances. A POI that is within a half-kilometer distance is given a score of 1, while those that are farther than two kilometers are

given zero. Then, the scores for each category were averaged to produce the score for each node. Finally, the scores for each node in the road network were averaged to yield the walkability score for the entire region. These steps are shown in Fig. 6.

4.3.2 Target values

This problem can be formalized as a function $m = f(D_1, D_2)$ where D_1 is the nodes dataset, and D_2 is for POIs. The value m represents the walkability score of the entire dataset. As for the synopsis problem, all three models can directly produce this value without any additional transformation.

4.3.3 Input values

For this problem, we extracted road networks using [68] and points of interest for several geographic regions from OpenStreetMap (OSM) across 9 categories. From that, we extracted several datasets by subdividing the overall area into smaller ones on which the exact walkability score could be calculated. We also applied some augmentation techniques to reduce the degree of unbalancing in the training set. We trained three models for each representation, GI-W, GG-W, and GV-W, shown in Table 1. As for the previous problem, the histogram size was set to 64, yielding an input tensor of shape (11, 64, 64), since the global compound histogram was obtained by stacking the global histogram $H(D_1) \in \mathbb{R}^{1\times64\times64}$ of D_1, containing only its positional histogram, on top of the global histogram $H(D_2) \in \mathbb{R}^{10\times64\times64}$, composed of the positional histogram of D_2 and its stacked thematic histogram. Similarly, for the GG-W model, there were three tensors: two for the geometries of the road network ($G(D_1)$) and the POIs ($G(D_2)$), respectively, and one for the 9 POI thematic attributes

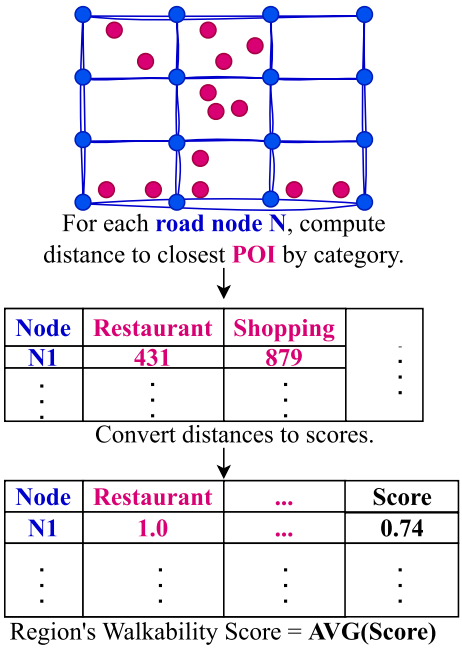

Fig. 6 Example of walkability computation

($A(D_2)$). Finally, for the GV-W model, the shape of the input tensor V depended on the sampling upper bound $b = 1024$ and the embedding size $e = 32$, in addition to the number of considered attribute categories.

5 Experiments

We conducted extensive experiments to evaluate the proposed representations across all three problems. The following subsections detail the experimental setup and present the corresponding results for each problem.

5.1 Experimental setup

Baselines We compared the three learning-based models against two baselines: (1) an optimized CPU-based algorithm that processes the entire dataset, and (2) a random baseline that generates guesses uniformly within the range of possible values. It is important to emphasize that the goal of this evaluation is not to outperform a specialized state-of-the-art method for each individual task, nor to establish a new best model for spatial synopsis, clustering, or walkability estimation. Rather, our objective is to compare the relative capabilities of three families of representations for geospatial vector data: image-based, graph-based, and vector-based representations. For this reason, we do not require a strong task-specific learning baseline for every problem. Such baselines would likely differ substantially across the three tasks and could obscure the study's main focus: the effect of the input representation on a deep learning model's ability to capture spatial structure and approximate geospatial operations. In this perspective, the optimized CPU-based algorithm provides the exact reference value for each task and therefore represents the upper bound on accuracy, although it requires processing the full dataset. The random baseline provides a simple lower bound on the error obtained without learning meaningful spatial patterns. Although the random baseline was not expected to be competitive, it showed the extent to which each representation model can capture meaningful information beyond random guessing. In this setting, the most relevant comparison is among the three proposed representation families, rather than a competition against highly specialized methods designed for each separate task.

Hardware For CPU-based processing, such as when computing ground truth values, the tasks were run on two *AMD EPYC 7713 64-Core Processor* with a total of 256 threads and memory of *128GB*. GPU-based tasks, e.g., training and inference, were run on a server with *AMD EPYC 7543 32-Core Processor* with a total of 64 threads, 256GB of memory, and *NVIDIA A100-SXM4-80GB* GPU.

Datasets The datasets used in the experiments were grouped into different collections, which have been detailed in Table 2 based on their type, i.e., synthetic or real, and the problem for which they have been considered. In particular, we have: (i) collections that contain synthetic datasets, i.e., type = syn, generated through the Spider Generator tool [66]. These collections are grouped based on the considered spatial distributions: uniform, diagonal, Gaussian, Sierpinski, bit, or parcel. The suffix "_large" was added to distinguish synthetic

Table 2 Synthetic and **real** datasets used for Spatial Data Synopsis (SDS), Spatial Clustering (CL), and Walkability (WK) evaluation

Collection	Type	Problem	# Sets	Min Card	Max Card
diagonal	syn	SDS	675	10000	20000
gaussian	syn	SDS	300	10000	20000
sierpinski	syn	SDS	300	10000	20000
uniform	syn	SDS	300	10000	20000
diagonal_large	syn	SDS	108	25000	100000
gaussian_large	syn	SDS	75	25000	75000
sierpinski_large	syn	SDS	100	25000	100000
uniform_large	syn	SDS	100	25000	100000
bit	syn	SDS	102	10000	20000
parcel	syn	SDS	108	10000	20000
weather	real	SDS, CL	1568	1025	11613
walkability	real	WK	20000	2	27999

datasets with a larger number of geometries. (ii) Collections that contain real datasets, i.e., type = real, namely *weather* and *walkability*, which were obtained from [7] and Open Street Map (OSM), respectively. As regards the operation column, SDS stands for spatial data synopsis, CL for clustering, and WK for walkability estimation. The last three columns indicate the number of datasets generated for each collection, and their respective minimum and maximum cardinality.

All of these datasets were encoded by using the three representations described in Sec. 3 and used as input for the corresponding model, depending on the specific application, as described in Sect. 4. Notice that some collections were used both in training and in the evaluation phase, by using a subset of 80% for training and a subset of 20% for testing, while other collections were used only for the evaluation, with the aim to capture the generalization capabilities of the models in making good predictions also for distributions belonging to collections they have never seen.

5.2 Spatial data synopsis

The models for this problem were trained to predict nine different values, given a dataset D, as discussed earlier in Sec. 4.1.1, i.e., the maximum and minimum average value (hotspots), the location of hotspots, the K_V, and the box count values E_0 and E_2. As mentioned above, we evaluated these models from a different perspective. First, we looked at how they perform across different collections of datasets, using some representatives during the training phase, and then we evaluated their ability to make predictions for collections they had never seen. For the first experiment, we separated 20% of the collections for using them in the evaluation stage, while the 80% were used for training.

We summarized the results obtained for models GI-S, GG-S, and GV-S in Table 3. The last column was added to compare how the models performed relative to a random prediction within the range of possible values. This verifies whether the models learn better than random guesses, which, as explained at the beginning of this section, represents the lower bound for prediction accuracy.

The metric used in this Table 3 to evaluate the prediction error is the average of the Weighted Mean Absolute Percentage Error (wMAPE) of all the model outputs, where

Table 3 Data Synopsis Summary by Collection (wMAPE)

Collection	GI-S	GG-S	GV-S	Random
In-distribution Results				
diagonal	0.1059	**0.0660**	0.0801	1.5825
gaussian	0.7096	**0.1377**	0.1660	2.3565
sierpinski	0.1271	**0.1133**	0.1235	2.7929
uniform	0.6664	**0.1932**	0.2632	10.2451
Out-of-distribution Results				
bit	0.3271	**0.1831**	0.3026	1.4528
parcel	0.2381	0.4653	**0.2261**	2.7672
weather	0.4323	**0.3326**	0.4937	0.8745
diagonal_large	0.1091	**0.0638**	0.0788	1.6031
sierpinski_large	0.1177	**0.1075**	0.1134	2.9641
gaussian_large	0.5805	**0.1243**	0.1633	2.2329
uniform_large	0.6529	**0.1905**	0.2979	9.7232

$\text{wMAPE} = (\sum_{i=1}^{n} |y_i - \hat{y}_i|)/(\sum_{i=1}^{n} |y_i|)$ and y_i is the actual value, while $\hat{y}_i$ is the predicted value. Note that wMAPE was computed for each output value separately, and then the obtained values were averaged to obtain the final value. The distributions in the first four collections are used for the 80% in training and the remaining 20% in testing.

The GeoGraph model (GG-S) performed better on the seen datasets and their scaled versions, such as *uniform* and *uniform_large*. There was only one case where it performed slightly worse than the other approaches. For the *parcel* distribution, GG-S performed much better for the hotspot-related values, but worse for some of the K_V and the box counts. However, we noted that the actual K_V where GG-S performed worse are very close to zero, and small deviations from the actual value result in larger errors. Anyway, all models had much smaller errors than the random predictions. In general, GG-S actually produced much better results for the K_V with the smallest radius. These results gave us confidence that the models actually learned the functions they were trained to predict, especially since some distributions were not seen in training and most of the evaluation sets have hotspot positions that were not seen in training. The results on the larger datasets showed that the quality remains consistent even when we made inferences on data sizes a few times larger than those seen in training.

We also performed a detailed evaluation of the results by considering every single component of the target output. We summarized these detailed results in Table 4. For this evaluation, we also used wMAPE, but this time it was computed for each output value across all datasets and collections. The table shows that GG-S performed better for hotspot prediction, performed similarly for K_V, and performed worse for box counts. It is not surprising that the image-based model performed better for box counts, since the first step in computing them was creating a histogram, which was then provided as input to the model. However, the GI-S model performed noticeably worse when predicting K_V for the smallest radius. This was most likely because each pixel in the input histogram spans a region that is multiple times the radius, which was consistent with our expectations.

Moreover, we performed experiments on real-world datasets, specifically, the *weather* collection. For this collection, we evaluated six models: three models trained only on synthetic data, GI-S, GG-S, and GV-S, and three models trained on 80% of the real-world weather data, GI-S-W, GG-S-W, and GV-S-W. The results are shown in Table 5. Clearly, the models trained on specific real-world datasets perform better, but those trained only on syn-

Table 4 Data Synopsis Summary by Output (wMAPE)

Attribute value	Params	GI-S	GG-S	GV-S	Random
hotspots min-value	$k = 16$	0.20	**0.12**	0.28	0.84
hotspots x_{min}	$k = 16$	0.25	**0.12**	0.13	0.60
hotspots y_{min}	$k = 16$	0.33	**0.16**	0.21	0.68
hotspots max-value	$k = 16$	0.05	**0.04**	0.11	0.31
hotspots x_{max}	$k = 16$	0.25	**0.18**	0.25	0.74
hotspots y_{max}	$k = 16$	0.19	**0.14**	0.18	0.72
hotspots min-value	$k = 32$	0.21	**0.13**	0.21	0.80
hotspots x_{min}	$k = 32$	0.25	**0.13**	**0.13**	0.59
hotspots y_{min}	$k = 32$	0.32	0.18	**0.17**	0.67
hotspots max-value	$k = 32$	0.05	**0.03**	0.12	0.31
hotspots x_{max}	$k = 32$	0.25	**0.18**	0.26	0.78
hotspots y_{max}	$k = 32$	0.17	**0.14**	0.16	0.71
hotspots min-value	$k = 64$	0.21	**0.13**	0.18	0.78
hotspots x_{min}	$k = 64$	0.25	**0.12**	0.15	0.60
hotspots y_{min}	$k = 64$	0.32	**0.18**	**0.18**	0.69
hotspots max-value	$k = 64$	0.04	**0.04**	0.12	0.32
hotspots x_{max}	$k = 64$	0.25	**0.17**	0.30	0.75
hotspots y_{max}	$k = 64$	0.16	**0.15**	0.17	0.69
K_V	$r = 0.025$	1.88	0.91	**0.76**	16.30
K_V	$r = 0.05$	**0.56**	0.60	0.68	5.30
K_V	$r = 0.10$	**0.36**	0.50	0.62	1.69
K_V	$r = 0.25$	0.34	**0.15**	0.42	0.60
box counts E_0	–	**0.03**	0.05	0.07	0.22
box counts E_2	–	**0.11**	0.20	0.33	0.28

thetic data showed acceptable errors, demonstrating the models' ability to understand geospatial concepts. The results are consistent with our previous observations, as both GG-S-* models consistently performed better than or equal to their counterparts.

Finally, we evaluated the running time of estimating these values using our models relative to that of traditional algorithms. This was summarized in Table 6. The prediction time of the GI-S model remained small even at large sizes, since the input size was fixed; however, preparing the histogram increased with data size, though it still scaled well. Similarly, GV-S scaled well, since its input was bounded in size. Conversely, the running time of the GG-S model can increase significantly with increasing dataset size, but it was still much lower than the time required to compute the target values with traditional algorithms. Since GG-S is the only model that did not have a fixed input size, we also evaluated how GG-S scales with increasing the dataset size. Some results are shown in Fig. 7. We started with 10, 000 points and kept doubling the size by monitoring the increment of the prediction time. We encountered a memory access error at $2^8 \times 10^4$ points, indicating that GPU memory demand exceeded available capacity. Memory consumption increases linearly, while inference time increases sub-quadratically. The main effect was due to the number of edges required per message-passing step. Another observation is that the GG-S model converged in fewer epochs than the others, as shown in Fig. 8.

Table 5 Data Synopsis for Weather Data Outputs (wMAPE)

Attribute value	Params	GI-S	GG-S	GV-S	GI-S-W	GG-S-W	GV-S-W	Rand
hotspots min-value	$k = 16$	0.40	**0.22**	0.52	**0.07**	**0.07**	0.12	0.55
hotspots x_{min}	$k = 16$	0.46	0.22	**0.15**	0.10	**0.08**	0.14	0.45
hotspots y_{min}	$k = 16$	0.72	**0.33**	0.41	0.16	**0.12**	0.26	0.72
hotspots max-value	$k = 16$	0.07	**0.05**	0.25	**0.02**	**0.02**	0.03	0.28
hotspots x_{max}	$k = 16$	0.61	**0.46**	0.60	0.27	**0.24**	0.34	0.70
hotspots y_{max}	$k = 16$	0.27	0.27	**0.24**	0.17	**0.15**	0.20	0.52
hotspots min-value	$k = 32$	0.40	**0.24**	0.34	**0.04**	**0.04**	0.07	0.54
hotspots x_{min}	$k = 32$	0.47	0.23	**0.14**	**0.10**	**0.09**	0.14	0.48
hotspots y_{min}	$k = 32$	0.70	0.39	**0.34**	0.14	**0.09**	0.23	0.67
hotspots max-value	$k = 32$	0.07	**0.04**	0.32	**0.01**	**0.01**	0.02	0.26
hotspots x_{max}	$k = 32$	0.66	**0.48**	0.72	0.27	**0.25**	0.31	0.76
hotspots y_{max}	$k = 32$	**0.21**	0.25	0.26	0.12	**0.12**	0.16	0.52
hotspots min-value	$k = 64$	0.41	**0.24**	0.28	**0.07**	0.08	**0.07**	0.51
hotspots x_{min}	$k = 64$	0.48	0.23	**0.17**	0.11	**0.09**	0.14	0.48
hotspots y_{min}	$k = 64$	0.67	0.39	**0.35**	0.14	**0.11**	0.20	0.66
hotspots max-value	$k = 64$	0.08	**0.07**	0.33	**0.02**	**0.02**	**0.02**	0.26
hotspots x_{max}	$k = 64$	0.67	**0.49**	0.95	0.28	**0.24**	0.29	0.79
hotspots y_{max}	$k = 64$	**0.18**	0.30	0.32	**0.11**	**0.11**	0.15	0.45
K_V	$r = 0.025$	1.24	**0.60**	0.85	2.47	**0.46**	1.16	8.05
K_V	$r = 0.05$	**0.30**	0.76	0.96	0.43	**0.18**	0.30	2.14
K_V	$r = 0.10$	**0.49**	0.74	0.95	0.23	**0.17**	0.28	0.57
K_V	$r = 0.25$	0.44	**0.24**	0.91	0.17	**0.05**	0.09	0.47
box counts E_0	–	**0.05**	0.12	0.19	**0.02**	**0.02**	0.03	0.20
box counts E_2	–	**0.30**	0.64	1.32	**0.01**	**0.01**	0.02	0.49

Table 6 Data synopsis execution time in seconds

Collection	GI-S		GG-S	GV-S	Exact value
	Inference	+Histogram			
uniform	0.02	1.61	2.37	0.23	830.91
diagonal	0.02	3.53	5.02	0.49	1625.47
gaussian	0.01	1.55	2.38	0.22	900.77
sierpinski	0.01	1.61	3.02	0.22	824.41
weather	0.04	4.49	3.39	1.16	2368.87
bit	0.02	2.75	4.25	0.38	1185.32
parcel	0.01	2.83	4.35	0.39	1253.81
uniform_large	0.01	11.25	204.86	0.37	6694.03
gaussian_large	0.01	10.93	202.96	0.37	11582.96
diagonal_large	0.01	11.83	219.14	0.39	7262.92
sierpinski_large	0.01	10.63	205.47	0.36	6611.57

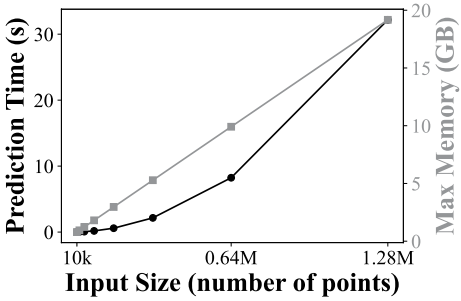

Fig. 7 GG-S time and memory vs. input size

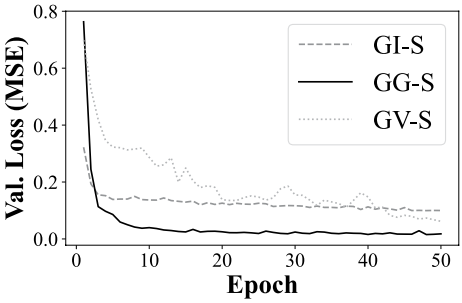

Fig. 8 Data Synthesis Validation Loss by Epoch

5.3 Clustering

For the clustering operation, we compared the clusters estimated by the three models with those discovered by ST-DBSCAN, which the model was trained to predict. As reported in Table 2, for the clustering operation, we used the real-world weather datasets, with 80% of the collection used for training and the remaining for testing. We used three scores for this evaluation: (i) a perfect *homogeneity* score of 1.0 indicates that all points within a predicted cluster are within the same cluster in the ground truth. (ii) *Completeness* has a perfect score of 1.0 if all points in a single cluster in the ground truth are predicted to be within the same cluster. (iii) The *V-measure* is an average of the other two scores. For more information on these scores, see [69]. The results of this evaluation are shown in Table 7.

The best clustering was produced by GG-C, the non-parametrized GeoGraph model. This outcome resulted from the observation that clustering leverages connections and neighbor relationships between geometric objects, which are better represented by a graph-based representation. The parametrized variant offers greater flexibility as it can work with any set of parameters which makes it more practical at the cost of slightly lower accuracy. The GI-C family of models built on GeoImg also performed well, as they captured the data's density, which is important for density-based clustering. Finally, the vector-based models built on GeoVec performed significantly lower since their encoding is centered on each geometry, which captures the relationship across geometries less well. Generally, the models performed well in all cases except when $\mu = 5$, which led to a large number of clusters with only a few points each. These become harder to capture by the proposed models.

Table 7 Evaluation of estimated clusters. Columns 3, 4, and 5 show homogeneity, completeness, and V-Measure, respectively. Higher values are better

Model	Parameters ($\varepsilon_1, \varepsilon_2, \mu$)	Homog.	Compl.	V-Meas.
GI-C	(0.05, 200.0, 50.0)	0.86	0.94	0.90
GI-CP	(0.02, 200.0, 5.0)	0.85	0.67	0.74
GI-CP	(0.03, 50.0, 5.0)	0.81	0.69	0.74
GI-CP	(0.03, 200.0, 20.0)	0.83	0.79	0.81
GI-CP	(0.05, 100.0, 50.0)	0.83	0.87	0.85
GI-CP	(0.05, 200.0, 50.0)	0.85	0.89	0.87
GI-CP	Average	0.83	0.78	0.80
GG-C	(0.05, 200.0, 50.0)	0.95	0.95	0.95
GG-CP	(0.02, 200.0, 5.0)	0.79	0.71	0.74
GG-CP	(0.03, 50.0, 5.0)	0.73	0.68	0.70
GG-CP	(0.03, 200.0, 20.0)	0.80	0.81	0.80
GG-CP	(0.05, 100.0, 50.0)	0.85	0.87	0.86
GG-CP	(0.05, 200.0, 50.0)	0.85	0.88	0.86
GG-CP	Average	0.80	0.79	0.79
GV-C	(0.05, 200.0, 50.0)	0.90	0.92	0.90
GV-CP	(0.02, 200.0, 5.0)	0.55	0.4	0.45
GV-CP	(0.03, 50.0, 5.0)	0.51	0.35	0.4
GV-CP	(0.03, 200.0, 20.0)	0.75	0.67	0.7
GV-CP	(0.05, 100.0, 50.0)	0.82	0.81	0.81
GV-CP	(0.05, 200.0, 50.0)	0.83	0.81	0.82
GV-CP	Average	0.69	0.61	0.64

5.4 Walkability estimation

We summarized the walkability results in Table 8. The table shows, for each walkability range, the percentage of scores that are predicted within 0.1 of their actual values. From the table, GI-W seems to work much better than the other two, which show similar accuracy. We think this can be attributed to three aspects related to each method. First, in this data set, the data was divided into $5km \times 5km$ regions, resulting in sparser histograms and less overlap of points within each histogram pixel. Second, the decrease in accuracy in GG-W could be due to the way the dataset join is performed. While we experimented with several join variations, there is still room for improvement. Third, in GV-W, the significant decrease in accuracy could be due to the sampling. However, all three cases still produce reasonably good results and perform much faster than computing the exact result.

5.5 Discussion

The experimental evaluation reported in the previous section provides evidence that existing DL architectures can approximate several classes of geospatial vector operations when the input data are encoded through suitable representations. At the same time, the results also show that the choice of representation has a substantial impact on model behavior. The three representation families considered in this work encode different inductive biases: GeoImg summarizes the input space through fixed-resolution histograms, GeoGraph preserves object-level structure and local neighborhoods through point-set processing, and GeoVec represents individual geometries as fixed-length embeddings processed by a Transformer. These differences explain most of the trends observed across the three tasks.

Table 8 Summary of accuracy for walkability results. The first column contains the dataset (DS) label: CA stands for California, FL for Florida, IT for Italy, NY for New York, and UK for the United Kingdom. Note: each column represents regions within a walkability range, and scores represent the percentage of regions that received highly accurate estimates

DS	Model	[0.0,0.1]	(0.1,0.2]	(0.2,0.3]	(0.3,0.4]	(0.4,0.5]	(0.5,0.6]	(0.6,0.7]	(0.7,0.8]	(0.8,0.9]	(0.9,1.0]	Avg
CA	GI-W	86%	89%	97%	95%	87%	85%	81%	79%	100%	–	88%
	GG-W	92%	88%	69%	68%	67%	66%	78%	79%	68%	–	77%
	GV-W	94%	86%	85%	84%	64%	65%	78%	61%	56%	–	75%
FL	GI-W	100%	75%	97%	97%	78%	74%	100%	–	–	–	89%
	GG-W	95%	98%	81%	91%	57%	45%	57%	–	–	–	80%
	GV-W	97%	70%	84%	88%	61%	55%	43%	–	–	–	71%
IT	GI-W	96%	94%	94%	90%	83%	80%	79%	83%	99%	–	89%
	GG-W	88%	86%	82%	82%	77%	76%	72%	78%	69%	–	78%
	GV-W	97%	89%	87%	84%	51%	68%	72%	67%	67%	–	76%
NY	GI-W	100%	100%	95%	95%	82%	100%	0%	83%	100%	100%	96%
	GG-W	92%	81%	91%	71%	82%	71%	50%	88%	94%	12%	51%
	GV-W	98%	93%	64%	86%	71%	100%	0%	51%	0%	96%	66%
UK	GI-W	100%	95%	93%	92%	85%	80%	91%	98%	100%	100%	93%
	GG-W	89%	93%	88%	91%	72%	59%	69%	88%	80%	21%	72%
	GV-W	96%	95%	84%	84%	56%	62%	75%	46%	47%	99%	74%

Spatial data synopsis The spatial synopsis task is the most heterogeneous among the considered problems because the target vector includes both local and global quantities: hotspot values and locations, K_V values at different radii, and box-count descriptors. The results in Tables 3 and 4 show that GeoGraph generally achieves the best accuracy for quantities that depend on fine-grained spatial relationships, especially hotspot localization and small-radius neighborhood statistics. This behavior is expected because GeoGraph operates directly on the input points and constructs neighborhoods from their metric positions. As a result, it can preserve local interactions that may be lost when data are aggregated into histogram cells or sampled into fixed-length sequences.

The advantage of GeoGraph is particularly visible for hotspot-related outputs. Hotspot estimation requires not only identifying extreme thematic values but also associating them with their correct spatial locations, taking local neighborhoods into account. Since GeoGraph maintains one representation per geometry and aggregates information through local set-abstraction layers, it can more accurately preserve the association between each point, its attributes, and its surrounding points. In contrast, GeoImg assigns all geometries falling in the same cell to the same spatial bin, which can blur local variation when multiple points with different attribute values fall in the same histogram cell. GeoVec also performs reasonably well, but its fixed sampling bound can remove some objects before the Transformer receives the input, which makes local extrema more difficult to preserve in dense datasets.

The behavior of the K_V metric further illustrates this trade-off. For small radii, the relevant signal is determined by short-range interactions among nearby points. This favors GeoGraph, which explicitly builds neighborhoods in the original metric space. GeoImg performs worse for the smallest radius because the histogram resolution imposes a minimum spatial granularity: if the radius is substantially smaller than a cell, points that should be distinguished may be aggregated into the same cell, while points in adjacent cells may not be represented with sufficient precision. As the radius increases, the operation becomes less sensitive to exact point-level distances and more dependent on the global distribution of the data. In those cases, the gap among the models decreases, and the histogram representation becomes more competitive.

Box-count values show the opposite pattern. GeoImg performs particularly well for these outputs because box counting is itself based on repeatedly partitioning the space into grids and measuring occupancy. Therefore, the histogram representation is naturally aligned with the structure of the target operation. This result is important because it shows that loss of point-level precision is not always detrimental: when the operation depends on global spatial occupancy or density summaries, a fixed-grid representation can encode the relevant signal compactly and efficiently. GeoGraph is less effective for this component because its architecture is optimized for local point interactions rather than for directly estimating multiscale grid occupancy.

Overall, all models remain substantially better than the random baseline, including on distributions not seen during training, suggesting they are not merely memorizing dataset-specific patterns. However, the degree of generalization varies across representations and targets. GeoGraph generalizes well when the unseen data still require local geometric reasoning similar to the training distributions. GeoImg generalizes well when the unseen distribution can be summarized by spatial density patterns at the chosen histogram resolution.

GeoVec is more variable, likely because its performance depends both on the quality of the geometry embedding and on the representativeness of the sampled sequence passed to the Transformer. These results confirm that learned geospatial representations can capture reusable spatial properties, but also that their generalization is constrained by the information retained during encoding.

The timing results in Table 6 highlight an additional trade-off. GeoImg and GeoVec have fixed-size inputs, so their inference time is relatively stable with respect to dataset cardinality, although GeoImg still requires histogram construction. GeoGraph is more expensive because its input size grows with the number of geometries and because neighborhood construction and message passing depend on the number of points and edges. Nevertheless, even GeoGraph remains much faster than exact computation for the evaluated synopsis operations. This suggests that learning-based approximations are most attractive when exact spatial analytics are repeatedly evaluated over many datasets or parameter settings, especially when small errors are acceptable.

Spatial clustering The clustering task evaluates a different kind of capability: instead of predicting a fixed-size summary for the entire dataset, the models must approximate a spatial operation that assigns a label to each geometry. The results in Table 7 show that all three representations can capture clustering structure to some extent, but their behavior differs significantly.

The best result is obtained by the non-parametric GeoGraph clustering model. This is consistent with the nature of density-based clustering. ST-DBSCAN assigns clusters based on neighborhood connectivity, local density, and reachability. These are precisely the types of relationships that a point-set architecture is designed to model. By maintaining point-level representations and propagating information across local neighborhoods, GeoGraph can better approximate the expansion behavior of density-based clusters. This explains the high homogeneity and completeness achieved by the fixed-parameter model.

The GeoImg models also perform well, especially when clusters correspond to spatially coherent dense regions that can be represented at the histogram resolution. The histogram representation captures density patterns effectively, which is useful for density-based clustering. However, it introduces a limitation: all points falling in the same cell share the same predicted output before the final conversion back to geometries. This makes it harder to represent cluster boundaries precisely, especially when a cell contains points from different clusters or when small clusters occupy only a few cells. This explains why GeoImg is competitive but generally less accurate than GeoGraph.

The GeoVec clustering models show the weakest average performance, especially in the parametrized setting. This suggests that encoding each geometry independently and then processing a sampled sequence is less effective for reproducing density-reachability relationships. Clustering depends on the relative arrangement of many nearby points, and this information may be weakened by sampling, padding, and nearest-neighbor assignment during the conversion from Transformer outputs back to the full target dataset. The degradation is especially visible when the parameters produce many small clusters. In such cases, missing or underrepresenting even a small number of points can significantly affect both homogeneity and completeness.

The comparison between fixed-parameter and parametrized models is also important. Fixed-parameter models need to learn only the behavior of a single clustering configuration, whereas parametrized models must learn how the output changes as ϵ_1, ϵ_2, and μ vary. This is a substantially harder learning problem because small changes in clustering parameters can lead to discontinuous shifts in cluster assignments. For example, increasing the distance threshold may merge two clusters, while increasing the minimum-points parameter may convert small clusters into noise. The lower scores of the parametrized models, therefore, do not necessarily indicate that the representations fail to encode spatial structure; rather, they show that learning an entire family of density-based clustering behaviors is more difficult than approximating a single fixed operation.

Walkability estimation The walkability task differs from the previous two because it requires reasoning across two datasets: pedestrian network nodes and points of interest. The target score depends on nearest-neighbor relationships between the two datasets, category-specific accessibility, and aggregation over the road network. Therefore, this task evaluates whether each representation can capture cross-dataset spatial interactions.

The results in Table 8 show that GeoImg performs best on this task. This may appear surprising because GeoImg is the most aggregated representation, but it is consistent with the structure of the walkability computation and the spatial scale of the data. The regions used in this task are relatively small, and the histogram representation aligns the road network and POI layers on a shared reference grid. Stacking the two datasets as channels allows the CNN to learn local co-occurrence patterns between road nodes and nearby POIs. In other words, the convolutional filters can detect whether amenities of different categories are spatially close to the pedestrian network at the resolution required for the final regional score. Since the output is a single aggregate walkability value rather than a point-level prediction, the loss of exact object identity is less harmful than in clustering.

The weaker performance of GeoGraph on walkability highlights a current limitation of the graph-based representation used in this paper. While GeoGraph is effective for single-dataset local reasoning, cross-dataset reasoning requires a suitable mechanism for joining or aligning two point sets. The adopted architecture joins datasets through neighborhood-based aggregation, but this step may not fully preserve all category-specific nearest-neighbor distances that determine the walkability score. In particular, walkability depends on the closest POI in each category for each road node, followed by a nonlinear distance-to-score transformation and an average over the region. If the join representation loses the closest object for a category, or aggregates several POIs before preserving the minimum-distance signal, the predicted score can degrade. This suggests that future graph-based models for this type of task should include more explicit cross-attention, category-aware nearest-neighbor aggregation, or differentiable spatial-join operators.

The GeoVec model performs similarly to or slightly worse than GeoGraph in most walkability cases. This behavior is likely due to the fixed sampling bound and the difficulty of representing two heterogeneous datasets in a single sequence. Although the dataset label and POI category attributes are included, the Transformer must infer the spatial relationship between sampled road nodes and sampled POIs from the embedded sequence. If relevant POIs or road nodes are not included in the sample, or if the sampling process changes the local category distribution, the walkability signal becomes noisy. This limitation is particu-

larly important for accessibility tasks because the score can depend on the nearest object in each category, not merely on the overall density of POIs.

Overall, walkability shows that the best representation is not necessarily the one that preserves the most detailed object-level information. For aggregate multi-dataset operations over moderately sized regions, a shared raster-like spatial reference can be highly effective because it naturally aligns heterogeneous inputs. However, the same representation may be less suitable for tasks that require object-level outputs or sub-cell precision. This reinforces the central message of the paper: different geospatial operations require different representation biases.

Cross-task comparison Taken together, the three tasks reveal a consistent pattern. GeoGraph is most effective when the operation depends on local point-level relationships within a single dataset, as in hotspot estimation, small-radius K_V, and fixed-parameter clustering. GeoImg is strongest when the operation can be expressed as a spatial density or co-occurrence pattern over a shared reference space, as in box counts and walkability estimation. GeoVec provides a flexible fixed-size representation and can be efficient at inference time, but its effectiveness depends heavily on the sampling strategy and on whether the target operation can tolerate the loss of unsampled geometries.

These findings also clarify the scalability-accuracy trade-off. GeoImg and GeoVec scale well because they impose fixed-size inputs, but this comes at the cost of discretization or sampling. GeoGraph avoids these two sources of information loss and can therefore preserve fine-grained spatial structure, but its computational and memory cost increases with dataset cardinality. Thus, GeoGraph is preferable when accuracy on local spatial interactions is the main objective and the dataset size remains manageable, whereas GeoImg and GeoVec are more appropriate when scalability, batching, and fixed input sizes are essential.

Another important observation is that the models approximate spatial operations rather than replace their exact definitions. Their predictions are useful for fast estimation, ranking, exploration, or pre-filtering of large collections of geospatial datasets. They are less appropriate when exact answers are required, such as in safety-critical settings. In this sense, the proposed approach should be viewed as complementary to traditional GIS and spatial database processing: learned models can quickly identify promising datasets, approximate expensive operations, or guide query optimization, while exact algorithms can still be applied when precise results are needed.

Finally, the results suggest several directions for improving learned geospatial analytics. First, hybrid representations could combine the strengths of the three approaches, for example by using histograms for global context and graph layers for local refinement. Second, cross-dataset operations such as walkability would benefit from architectures that explicitly model spatial joins, nearest-neighbor relationships, and category-aware aggregation. Third, GeoVec could be improved through more informed sampling, spatial partitioning, or hierarchical aggregation before the Transformer stage. Fourth, pretraining on large collections of vector geospatial datasets may improve transferability across regions, distributions, and spatial operations. These directions are important because the present results show that deep learning models can capture meaningful geospatial structure, but also that representation design remains the key factor determining their accuracy, scalability, and generalization.

6 Conclusion

This paper addressed a key limitation in the current landscape of geospatial analytics: although geospatial data constitute a major share of contemporary data ecosystems, existing deep learning methods still provide limited native support for directly understanding and processing vector geospatial datasets. To bridge this gap, we investigated how existing DL architectures can be adapted to geospatial settings by leveraging suitable representations of both input data and spatial operations, without requiring the design of entirely new models. In this sense, the research helped address an important knowledge gap identified in the literature, namely the lack of general methodologies for enabling DL models to reason over geospatial objects, their spatial characteristics, and their thematic attributes across different classes of geospatial problems.

The main contribution of this work to the geospatial community lies in the definition and comparative investigation of a representation-centered methodology for DL-enabled geospatial analytics. Rather than focusing on a single task or a single model family, we introduced a general framework for translating vector geospatial datasets and spatial operations into forms that can be processed by established DL architectures. By studying image-based, graph-based, and embedding-based representations under a unified perspective, the paper contributed a broader view of how geospatial structure can be encoded for learning.

The experimental findings provided several insights for readers. First, they showed that general-purpose DL architectures can, in fact, be leveraged to approximate a variety of geospatial operations when supported by suitable representations. Second, they suggested that image-based representations are particularly effective for multi-dataset problems and spatial estimation tasks in which spatial distributions can be meaningfully summarized over a shared reference space. Third, they showed that graph- and embedding-based approaches remain promising alternatives, especially in settings where preserving object-level structure is important, although they currently appear to be more sensitive to geometric complexity, scalability constraints, or the way in which inter-object spatial relations are encoded. More broadly, the results indicate that DL-based geospatial analytics is feasible, but that no single representation is universally best; instead, effectiveness depends on the interplay between data characteristics, task requirements, and model inductive biases.

At the same time, this research highlighted several limitations in existing DL representations and model architectures. Indeed, the considered representations, while general, may still lose information or introduce constraints: histogram-based encodings may smooth fine-grained geometric detail, graph-based methods may face scalability challenges for large datasets, and embedding-based approaches may depend heavily on the quality of the learned geometry encoders. These limitations open several directions for future research. A first direction is to investigate hybrid representations that combine the strengths of histograms, graphs, and learned embeddings.

Indeed, beyond vector-only representations, a promising research direction is to combine vector geospatial data with raster imagery and social-sensing data using multimodal learning frameworks. Investigating such heterogeneous representations is outside the scope of this paper and is left as an interesting direction for future work.

A second direction is the development of pretraining strategies and foundation-model-like approaches tailored to geospatial data to improve transferability across tasks and regions.

Finally, future work should also explore tighter integration among DL models, geospatial databases, and AI agents, enabling systems that not only predict from spatial data but also support richer forms of geospatial understanding, explanation, and decision-making.

Author Contributions All authors contributed equally to the conception, design, analysis, and writing of this manuscript.

Funding Open access funding provided by Università degli Studi di Verona within the CRUI-CARE Agreement. The author declares received no specific funding for this work.

Data Availability The data and the models supporting the findings of this study are available in a public GitHub repository https://github.com/smigliorini/geolearn

Declarations

Competing Interests Sara Migliorini is a member of the journal's Editorial Board as Action Editor, and she was not involved in the review of, or decisions related to, this manuscript. The author declares no other competing interests for this work.

Ethics Approval and Consent to Participate Not applicable, as this study did not involve human participants or animals.

Generative Artificial Intelligence During the preparation of this work, the authors used Grammarly's generative AI writing feature in order to check grammar, improve writing, and fix passive verbs to active verbs. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

1. Nargesian F, Zhu E, Miller RJ, Pu KQ, Arocena PC (2019) Data lake management: challenges and opportunities. Proc VLDB Endow 12(12):1986–1989. https://doi.org/10.14778/3352063.3352116
2. Armbrust M, Das T, Sun L, Yavuz B, Zhu S, Murthy M, Torres J, Hovell H, Ionescu A, Łuszczak A et al (2020) Delta lake: high-performance acid table storage over cloud object stores. Proceedings of the VLDB Endowment 13(12):3411–3424
3. Wang J, Hanson E, Li G, Papakonstantinou Y, Simhadri H, Xie C (2024) Vector databases: What's really new and what's next? (vldb 2024 panel). Proc VLDB Endow 17(12):4505–4506. https://doi.org/10.14778/3685800.3685911
4. Li G, Zhou X, Zhao X (2024) Llm for data management. Proceedings of the VLDB Endowment 17(12):4213–4216
5. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I (2021) Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, vol abs/2103.00020. arxiv.org/abs/2103.00020
6. Henke N, Bughin J, Chui M, Manyika J, Saleh T, Wiseman B, Sethupathy G (2016) The Age of Analytics: Competing in a Data-driven World. Technical report, McKinsey Global Institute

7. Menne MJ, Durre I, Vose RS, Gleason BE, Houston TG (2012) An Overview of the Global Historical Climatology Network-Daily Database. J Atmos Oceanic Tech 29(7):897–910. https://doi.org/10.1175/JTECH-D-11-00103.1
8. Tedjopurnomo DA, Bao Z, Zheng B, Choudhury FM, Qin AK (2023) A survey on modern deep neural network for traffic prediction: Trends, methods and challenges (extended abstract). In: 39th IEEE International Conference on Data Engineering. ICDE. IEEE, pp 3795–3796. https://doi.org/10.1109/ICDE55515.2023.00324
9. Mahmood AR, Aref WG (2017) Query processing techniques for big spatial-keyword data. In: Proceedings of the 2017 ACM International Conference on Management of Data. SIGMOD. ACM, pp 1777–1782. https://doi.org/10.1145/3035918.3054773
10. Doraiswamy H, Zacharatou ET, Miranda F, Lage M, Ailamaki A, Silva CT, Freire J (2018) Interactive visual exploration of spatio-temporal urban data sets using urbane. In: Proceedings of the 2018 International Conference on Management of Data. SIGMOD. ACM, pp 1693–1696. https://doi.org/10.1145/3183713.3193559
11. Shen T, Li Y, Moura J (2023) Forecasting COVID-19 dynamics: Clustering, generalized spatiotemporal attention, and impacts of mobility and geographic proximity. In: 39th IEEE International Conference on Data Engineering. ICDE. IEEE, pp 2892–2904. https://doi.org/10.1109/ICDE55515.2023.00221
12. Scarpone C, Brinkmann ST, Große T, Sonnenwald D, Fuchs M, Walker BB (2020) A multimethod approach for county-scale geospatial analysis of emerging infectious diseases: a cross-sectional case study of COVID-19 incidence in Germany. Int J Health Geogr 19(1):32. https://doi.org/10.1186/s12942-020-00225-1
13. Ma M, Xie P, Teng F, Wang B, Ji S, Zhang J, Li T (2023) HiSTGNN: Hierarchical spatio-temporal graph neural network for weather forecasting. Inf Sci 648(C). https://doi.org/10.1016/j.ins.2023.119580
14. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp 770–778. https://doi.org/10.1109/CVPR.2016.90
15. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. 18th Int.l Conf. on Medical Image Computing and Computer-Assisted Intervention. MICCAI 2015:234–241
16. Qi CR, Yi L, Su H, Guibas LJ (2017) Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, pp 5105–5114. Curran Associates Inc
17. Siampou MD, Li J, Krumm J, Shahabi C, Lu H (2025) Poly2Vec: Polymorphic Fourier-based encoding of geospatial objects for GeoAI applications. In: Singh A, Fazel M, Hsu D, Lacoste-Julien S, Berkenkamp F, Maharaj T, Wagstaff K, Zhu J (eds) Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol 267, pp 55511–55532. PMLR. https://proceedings.mlr.press/v267/siampou25a.html
18. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, pp 6000–6010
19. Stewart AJ, Robinson C, Corley IA, Ortiz A, Ferres J, Banerjee A (2022) Torchgeo: deep learning with geospatial data. In: Proceedings of the 30th Int Conf on Advances in Geographic Information Systems. SIGSPATIAL '22. https://doi.org/10.1145/3557915.3560953
20. Mendieta M, Han B, Shi X, Zhu Y, Chen C (2023) Towards geospatial foundation models via continual pretraining. In: IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, pp 16760–16770. IEEE. https://doi.org/10.1109/ICCV51070.2023.01541
21. IBM-NASA Prithvi Models Family (2026) ibm-nasa-geospatial (Hugging Face organization page). Accessed 13 Feb 2026. https://huggingface.co/ibm-nasa-geospatial
22. Tsaris A, Dias PA, Potnis A, Yin J, Wang F, Lunga D (2024) Pretraining Billion-scale Geospatial Foundational Models on Frontier. https://doi.org/10.48550/arXiv.2404.11706, arxiv.org/abs/2404.11706
23. Sun X, Wang P, Lu W, Zhu Z, Lu X, He Q, Li J, Rong X, Yang Z, Chang H, He Q, Yang G, Wang R, Fu K, Yuan J (2023) RingMo: A remote sensing foundation model with masked image modeling. IEEE Trans Geosci Remote Sens 61:1–22. https://doi.org/10.1109/TGRS.2022.3194732
24. Zhou Y, Feng L, Ke Y, Jiang X, Yan J, Yang X, Zhang W (2024) Towards Vision-Language Geo-Foundation Model: A Survey. arxiv.org/abs/2406.09385
25. Klemmer K, Robinson RE, Mackey C, Rußwurm L, M (2025) SatCLIP: Global, general-purpose location embeddings with satellite imagery. In: Proceedings of the AAAI conference on artificial intelligence
26. Cepeda VV, Nayak GK, Shah M (2023) GeoCLIP: Clip-inspired alignment between locations and images for effective worldwide geo-localization. In: Advances in neural information processing systems
27. Mai G, Janowicz K, Yan B, Zhu R, Cai L, Lao N (2020) Multi-scale representation learning for spatial feature distributions using grid cells. CoRR arxiv.org/abs/2003.00824

28. 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. SIGSPATIAL '17. https://doi.org/10.1145/3139958.3140054
29. Mai G, Jiang C, Sun W, Zhu R, Xuan Y, Cai L, Janowicz K, Ermon S, Lao N (2022) Towards general-purpose representation learning of polygonal geometries. GeoInformatica 27(2):289–340. https://doi.org/10.1007/s10707-022-00481-2
30. Mai G, Janowicz K, Hu Y, Gao S, Yan B, Zhu R, Cai L, Lao N (2022) A review of location encoding for geoai: methods and applications. Int J Geogr Inf Sci 36(4):639–673. https://doi.org/10.1080/13658816.2021.2004602
31. Qi CR, Su H, Mo K, Guibas LJ (2017) PointNet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 652–660
32. Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM (2019) Dynamic graph CNN for learning on point clouds. ACM Trans Graphics 38(5). https://doi.org/10.1145/3326362
33. Zhao H, Jiang L, Jia J, Torr P, Koltun V (2021) Point transformer. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp 16239–16248. https://doi.org/10.1109/ICCV48922.2021.01595
34. Wu X, Jiang L, Wang P-S, Liu Z, Liu X, Qiao Y, Ouyang W, He T, Zhao H (2024) Point transformer v3: Simpler, faster, stronger. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 4840–4851. https://doi.org/10.1109/CVPR52733.2024.00463
35. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch Geometric. In: ICLR workshop on representation learning on graphs and manifolds. arxiv.org/abs/1903.02428
36. Kan Z, Kwan M-P, Tang L (2022) Ripley's k-function for network-constrained flow data. Geogr Anal 54(4):769–788. https://doi.org/10.1111/gean.12300
37. Liao Z, Zhang L (2022) Spatial distribution characteristics and accessibility analysis of characteristic towns in guangdong province based on ripley's k function. J Math 1:2873707. https://doi.org/10.1155/2022/2873707
38. Tao R, Thill J-C, Yamada I (2015) Detecting clustering scales with the incremental k-function: Comparison tests on actual and simulated geospatial datasets. In: Popovich, V., Claramunt, C., Schrenk, M., Korolenko, K., Gensel, J. (eds.) Information Fusion and Geographic Information Systems (IF&GIS' 2015). Lecture Notes in Geoinformation and Cartography, pp 93–107. Springer, Cham. https://doi.org/10.1007/978-3-319-16667-4_6
39. Hohl A, Zheng M, Tang W, Delmelle E, Casas I (2017) Spatiotemporal Point Pattern Analysis Using Ripley's K Function. CRC Press, Geospatial Data Science Techniques and Applications
40. Liu Y, Kang YMahmood A, Magdy A (2023) Scalable evaluation of local k-function for radius-accurate hotspot detection in spatial networks. In: Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems. SIGSPATIAL '23. https://doi.org/10.1145/3589132.3625646
41. Wang Y, Gui Z, Wu H, Peng D, Wu J, Cui Z (2020) Optimizing and accelerating space-time Ripley 's K function based on Apache Spark for distributed spatiotemporal point pattern analysis. Futur Gener Comput Syst 105:96–118. https://doi.org/10.1016/j.future.2019.11.036
42. Tang W, Feng W, Jia M (2015) Massively parallel spatial point pattern analysis: Ripley's K function accelerated using graphics processing units. Int J Geogr Inf Sci 29(3):412–439. https://doi.org/10.1080/13658816.2014.976569
43. Chan TN, Leong Hou U, Peng Y, Choi B, Xu J (2022) Fast network k-function-based spatial analysis. Proc VLDB Endow 15(11):2853–2866. https://doi.org/10.14778/3551793.3551836
44. Belussi A, Migliorini S, Eldawy A (2018) Detecting skewness of big spatial data in spatialhadoop. In: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. SIGSPATIAL '18, pp 432–435. https://doi.org/10.1145/3274895.3274923
45. Vu T, Belussi A, Migliorini S, Eldawy A (2024) A learning-based framework for spatial join processing: estimation, optimization and tuning. VLDB J 33(4):1155–1177. https://doi.org/10.1007/s00778-024-00836-1
46. Yang W, Wang S, Sun Y, Peng Z (2022) Fast dataset search with earth mover's distance. Proc VLDB Endow 15(11):2517–2529. https://doi.org/10.14778/3551793.3551811
47. Hilprecht B, Schmidt A, Kulessa M, Molina A, Kersting K, Binnig C (2020) DeepDB: Learn from data, not from queries! Proceedings of the VLDB Endowment 13(7):992–1005
48. Marcus R, Negi P, Mao H, Zhang C, Alizadeh M, Kraska T, Papaemmanouil O, Tatbul N (2019) Neo: A learned query optimizer. Proceedings of the VLDB Endowment 12(11):1705–1718. https://doi.org/10.14778/3342263.3342644

49. Al Jawarneh IM, Bellavista P, Corradi A, Foschini L, Montanari R (2020) Locality-preserving spatial partitioning for geo big data analytics in main memory frameworks. In: 2020 IEEE Global Communications Conference, pp 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322544
50. Cesario E, Lindia P, Vinci A (2024) A scalable multi-density clustering approach to detect city hotspots in a smart city. Futur Gener Comput Syst 157:226–236. https://doi.org/10.1016/j.future.2024.03.042
51. Chattopadhyay A, Hassanzadeh P, Pasha S (2020) Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data. Sci Rep 10(1):1317. https://doi.org/10.1038/s41598-020-57897-9
52. Shobha N, Asha T (2017) Monitoring weather based meteorological data: Clustering approach for analysis. In: 2017 Int Conf on Innovative Mechanisms for Industry Applications (ICIMIA), pp 75–81. https://doi.org/10.1109/ICIMIA.2017.7975575
53. Tian K, Zhou S, Guan J (2017) DeepCluster: a general clustering framework based on deep learning. In: Machine learning and knowledge discovery in databases, pp 809–825. https://doi.org/10.1007/978-3-319-71246-8_49
54. Aljalbout E, Golkov V, Siddiqui Y, Strobel M, Cremers D (2018) Clustering with Deep Learning: Taxonomy and New Methods. https://doi.org/10.48550/arXiv.1801.07648
55. Min E, Guo X, Liu Q, Zhang G, Cui J, Long J (2018) A survey of clustering with deep learning: from the perspective of network architecture. IEEE Access 6:39501–39514. https://doi.org/10.1109/ACCESS.2018.2855437
56. Karim MR, Beyan O, Zappa A, Costa IG, Rebholz-Schuhmann D, Cochez M, Decker S (2021) Deep learning-based clustering approaches for bioinformatics. Brief Bioinform 22(1):393–415. https://doi.org/10.1093/bib/bbz170
57. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: Proceedings of the 33rd international conference on machine learning, pp 478–487
58. Venerandi A, Mellen H, Romice O, Porta S (2024) Walkability indices—the state of the art and future directions: A systematic review. Sustainability 16(16). https://doi.org/10.3390/su16166730
59. Gu P, Han Z, Cao Z, Chen Y, Jiang Y (2018) Using open source data to measure street walkability and bikeability in China: a case of four cities. Transportation Research Record Journal of the Transportation Research Board 2672(31):63–75. https://doi.org/10.1177/0361198118758652
60. Li Y, Yabuki N, Fukuda T (2023) Integrating GIS, deep learning, and environmental sensors for multicriteria evaluation of urban street walkability. Landsc Urban Plan 230:104603. https://doi.org/10.1016/j.landurbplan.2022.104603
61. Zhao X, Zhang J, Qin X (2018) k nn-dp: Handling data skewness in knn joins using mapreduce. IEEE Trans Parallel Distrib Syst 29(3):600–613. https://doi.org/10.1109/TPDS.2017.2767596
62. Francisco García-García LI, Corral A, Vassilakopoulos M (2023) Efficient distributed algorithms for distance join queries in spark-based spatial analytics systems. Int J Gen Syst 52(3):206–250. https://doi.org/10.1080/03081079.2023.2173750
63. Whitman RT, Marsh BG, Park MB, Hoel EG (2019) Distributed spatial and spatio-temporal join on apache spark. ACM Trans Spatial Algorithms Syst 5(1). https://doi.org/10.1145/3325135
64. Li X, Han K, Hu H, Zhang Y, Wang J (2023) Trajectorybert: Pre-training spatial trajectories with metric learning. In: Proceedings of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), pp 12657–12667
65. Zahn CT, Roskies RZ (1972) Fourier descriptors for plane closed curves. IEEE Transactions on Computers C 21(3):269–281. https://doi.org/10.1109/TC.1972.5008949
66. Katiyar P, Vu T, Eldawy A, Migliorini S, Belussi A (2020) Spiderweb: A spatial data generator on the web. In: Proceedings of the 28th International Conference on Advances in Geographic Information Systems. SIGSPATIAL '20, pp 465–468. https://doi.org/10.1145/3397536.3422351
67. Birant D, Kut A (2007) ST-DBSCAN: An algorithm for clustering spatial-temporal data. Data Knowl Eng 60(1):208–221. https://doi.org/10.1016/j.datak.2006.01.013
68. Boeing G (2024) Modeling and Analyzing Urban Networks and Amenities with OSMnx. https://geoffboeing.com/publications/osmnx-paper/. Accessed 25 Feb 2025
69. Rosenberg A, Hirschberg J (2007) V-measure: A conditional entropy-based external cluster evaluation measure. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp 410–420

Authors and Affiliations

Majid Saeedan[1] · Alberto Belussi[2] · Sara Migliorini[2] · Ahmed Eldawy[1]

✉ Sara Migliorini
sara.migliorini@univr.it

Majid Saeedan
msaee007@ucr.edu

Alberto Belussi
alberto.belussi@univr.it

Ahmed Eldawy
eldawy@ucr.edu

[1] Department of Computer Science and Engineering, University of California, Riverside, Riverside, California, USA

[2] Department of Computer Science, University of Verona, Verona, Italy