LEVERAGING DEEP LEARNING FOR TRAJECTORY SIMILARITY LEARNING AND TRAJECTORY PATHLET DICTIONARY CONSTRUCTION

GIAN CARLO ALIX

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

AUGUST 2023

Abstract

The rapid development of geospatial technologies and location-based devices have motivated the research community of trajectory data mining, due to numerous applications including route planning and navigation services. Of interest are similarity search tasks that several works addressed through representation learning. Our method St2Box offers refined representations by first representing trajectories as sets of roads, then adapting set-to-box architectures for learning *accurate*, *versatile* and *generalizable* set representations of trajectories for preserving similarity. Experimentally, St2Box outperforms baselines by up to ∼38%.

Another related problem involves constructing small sets of building blocks that can represent wide-ranging trajectories (*pathlet dictionaries*). However, currently-existing methods in constructing PDs are memory-intensive. Thus, we propose PathletRL for generating dictionaries that offer significant memory-savings. It initializes unit-length pathlets and iteratively merges them while maximizing utility – that is approximated using deep reinforcement learning-based method. Empirically, PathletRL can reduce its dictionary's size by up to 65.8% against state-of-the-art methods.

Acknowledgements

Firstly, I would like to express my sincerest appreciation to my supervisor, Dr. Manos Papagelis, for his guidance, expertise, and unwavering support throughout this research journey. His valuable insights, constructive feedback, and mentorship have been instrumental in the shaping of this thesis.

I would also like to acknowledge the examination committee members, Dr. Hina Tabassum and Dr. Mehdi Nourinejad, for taking the time to read my thesis and providing invaluable feedback that enriched the content and quality of my thesis.

Furthermore, I would like to express my deepest gratitude to my parents, Nurain Alix and Antonio Alix, for their unwavering love, endless support, and boundless encouragement throughout this academic pursuit.

In addition, I also would like to extend my thanks to my friends and my colleagues whose presence and camaraderie added immeasurable value to this thesis journey.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Definition
St2Box	Spatiotemporal Trajectories to Box Embeddings for Similarity Learning
Seg2Vec	Segments to Vector Representation
SegSrl	Segment-based Spatial Representation Learning
SegTrl	Segment-based Temporal Representation Learning
Seg2Box	Segments to Box Representation
Tp	Two-Phase Algorithm
Dita	Distributed In-Memory Trajectory Analytics
Lcrs	Longest Common Road Segments
NetErp	Network-aware Edit Distance with Real Penalty
Pd	Pathlet Dictionary
Dqn	Deep Q Network
Drl	Deep Reinforcement Learning
Mdp	Markov Decision Process
PathletRL	Pathlet Dictionary Construction using Trajectories with Reinforcement Learning
Poi	Point of Interest
SotA	State-of-the-art

Chapter 1

Introduction

The development of technology for gathering and tracking location data has led to the accumulation of vast amounts of trajectory data, consisting of spatial and temporal information of moving objects, such as persons or vehicles, that can be used in a variety of trajectory data analytics [153, 119]. Mining trajectory data to find interesting patterns is of increased research interest due to a broad range of practical applications relating to domains in transportation systems [92, 82], urban planning [55, 83], security [33, 47, 95], spatiotemporal epidemics [5, 4, 110], behavioral animal ecology [24, 117], among others. There are several technical problems in trajectory data mining that researchers and practitioners have focused on in recent years, including trajectory classification [108, 88, 73], clustering [81, 1, 52], prediction [192, 60, 183, 113, 168], simplification [161], recovery [32] and anomaly detection [114, 71]. A few comprehensive surveys on the topic can be found in Zheng [194], Alturi et al. [10], and Hamdi et al. [50].

In this research, we focus on the *trajectory similarity* problem, where given a trajectory set $\mathcal{T}$ and a query trajectory τ_q, we want to discover (and rank) the top k trajectories in $\mathcal{T}$ that are most similar to τ_q. Efficient methods for trajectory similarity are well sought-after due to their utility in several real-world applications, such as route planning [169], travel time estimation

[86, 160], and recommending trips [159] or points of interest (POIs) [151]. Moreover, we also address another problem of interest that is fairly new and is also related to trajectory data mining. It involves the construction of a small set of basic building blocks that can represent a wide range of trajectories, known as a *(trajectory) pathlet dictionary* (PD). The term *pathlet* appears in the literature by many names, such as *subtrajectories*, *trajectory segments*, or *fragments* [26, 1, 86, 192, 120, 108]. For consistency, we will use the term *pathlets* to denote these building blocks. Effectively constructing pathlet dictionaries is of increased research and practical interest due to a broad range of tasks and applications that can use it, such as route planning [26, 174], travel time prediction [54], personalized destination prediction [192, 167, 36], trajectory prediction [168, 60], and trajectory compression [26, 193]. For instance, storing massive trajectory datasets in smaller, limited resources such as mobile devices may require the need for compressing trajectories that preserves important spatiotemporal features while minimizing information loss; in such a case, a pathlet dictionary that could represent as many trajectories possible in the dataset could be useful. In another example, pathlet dictionaries can be beneficial in answering user queries such as path recommendations, particularly in cases for when navigation service apps are not readily available (e.g., lack of internet coverage, weak mobile signals, etc.). Appendix A provides an in-depth discussion of these applications.

1.1 The State of the Art and Limitations

To maintain focus on the two problems of interest we intend to address, we split this Chapter into two parts. The first part discusses the state-of-the-art models (and their limitations) for the trajectory similarity search problem, while the second covers the state-of-the-art methods for constructing pathlet dictionaries (PD), together with their inherent limitations.

1.1.1 SotA and Their Limitations for Similarity Search Tasks

Traditional metrics for measuring the similarity of two trajectories such as Dtw [173], Lcss [147], Hausdroff [9], and Erp [27] exist as *free-space* measures – that consider the geometric aspect of trajectories on the continuous (e.g., Euclidean) space. Others such as Tp [126], Dita [128], Lcrs [177] and NetERP [72] are called *network-aware* measures – that take into account the properties of the underlying road network (e.g., network structure, connectivity, etc.). The problem with these metrics lie on their computational overhead when measuring similarity; their expensive runtimes [53, 85] is due to their dependence on quadratic $\mathcal{O}(n^2)$ time pointwise matching schemes (for average trajectory length n), and can be a severe bottleneck when working with larger datasets [171, 185].

To address the concerns regarding computational complexity, several learning-based models [85, 171, 185, 170, 53] have been proposed that represent trajectories as embedding vectors while preserving their similarity relations in the latent space. These methods avoid operating directly on the original trajectories and have demonstrated significant computational savings. However, Yao et al. [172] and Li et al. [85] both treat their trajectories as ordinary sequence data and as a result are unable to capture the spatial semantic information of trajectories when represented as vectors. Yao et al.'s [171] NeuTraj focuses only on calculating similarity between two complete trajectories and thus ignores the rich information of sub-trajectories and the interrelationships among them. Zhang et al.'s [185] Traj2SimVec is designed mostly for learning similarity metrics on the Euclidean space only and thus fails to learn spatial information on the road network. Han et al.'s [53] Gts is only designed for Poi-based trajectory similarity computation and disregards actual travel paths on the road network. All these works moreover ignore the temporal aspect of trajectories and can be sub-optimal in contexts where real-time information is required (e.g., personalized trip recommendations). To address this limitation, Fu et al. [45] has proposed Trembr, an Rnn-based encoder-decoder that takes the temporal information into account. It considers timestamps in the decoding process, but is unable to capture the temporal regularities and periodic patterns in trajectories. Fang et al.'s [39] St2Vec learns the temporal features of trajectories by also capturing the periodic behaviors and the temporal dependence of the trajectories.

1.1.2 SotA and Their Limitations for Pd Construction

Many existing works frame the problem of analyzing and deriving pathlets as a (sub)trajectory clustering problem, where (sub)trajectory clusters represent popular paths (the pathlets) [81, 74, 1]. A few works considered an integer programming formulation with constraints to solve the problem [26, 86]. Some works designed their pathlets based on a route "representativeness" criterion [108, 156]. Unfortunately, these existing works suffer some limitations. For example, Chen et al. [26] assumes that the datasets used are noise-free. Zhou et al.'s [195] bag-of-segments method requires that trajectory segments are of fixed length. Van Krevald et al. [74] demands input trajectories to have the same start/endpoints. The cluster centroids in (sub)trajectory clustering methods [81, 74, 1, 155] do not necessarily reflect real roads in the road network. In addition, Wang et al. [156] demonstrated empirically that these clustering methods are computationally slow. In spite of runtime improvements, [156] also requires the user to provide some budget constraint B in the route representative discovery task, a domain-specific parameter that requires domain expert knowledge. Another related method is Traculus [81] that requires pathlets to be straight line segments, which is not always the case in real road maps. In addition, all these works do not constraint pathlets to be *edge-disjoint*; two pathlets are said to be edge-disjoint if they don't share any edge. Therefore, existing works allow pathlets in the dictionary to (partially) overlap. These methods, by design, follow a top-down approach in constructing a dictionary. This involves forming all possible pathlet candidates first, by considering pathlets of various configurations and sizes,

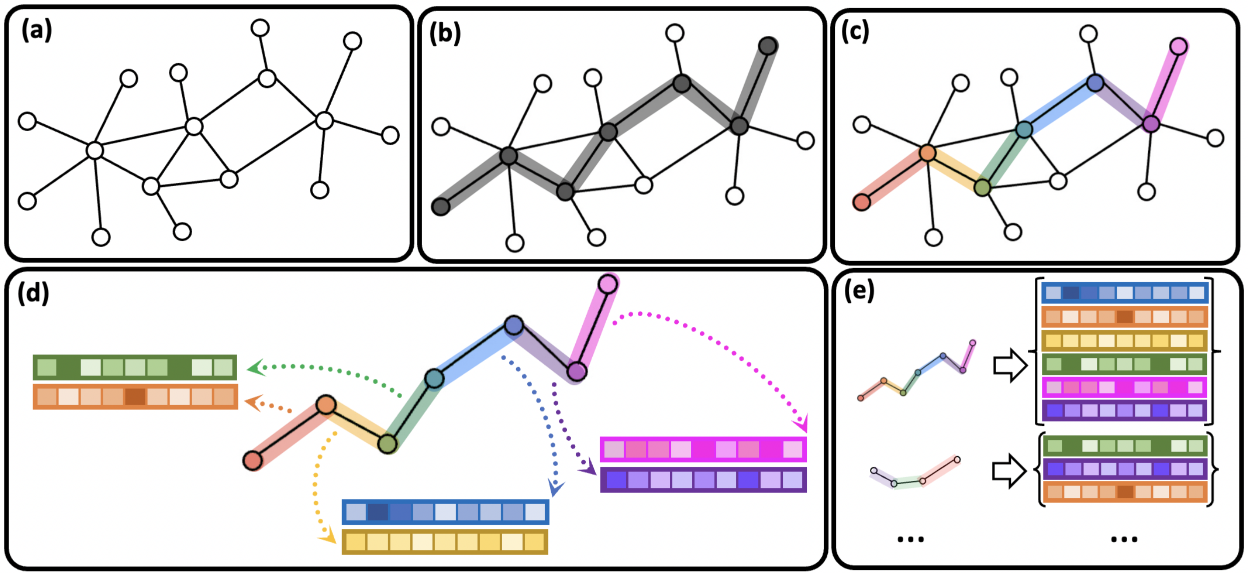


Figure 1.1: (a) A small road network; (b) The path of some trajectory τ; (c) The road-based representation of τ; (d) The spatiotemporal embedding vector for each road of τ; (e) All trajectories represented as the *set* of road embedding vectors.

and then eliminating candidates to form a smaller-sized dictionary that consists of only the most important ones (e.g., the most popular). While simple and intuitive, its main limitation is the need for a large memory to initially store the large number of pathlets, most of which are redundant. This also limits its applicability in real-world settings, particularly when dealing with large road networks and trajectory data, as the number of initial candidates can quickly become overwhelming.

1.2 The Proposed Approach

Similar to Chapter 1.1, we also split this Chapter into two parts. The first (second) part gives a brief overview of the proposed approach for addressing the trajectory similarity search task (pathlet dictionary construction) problem.

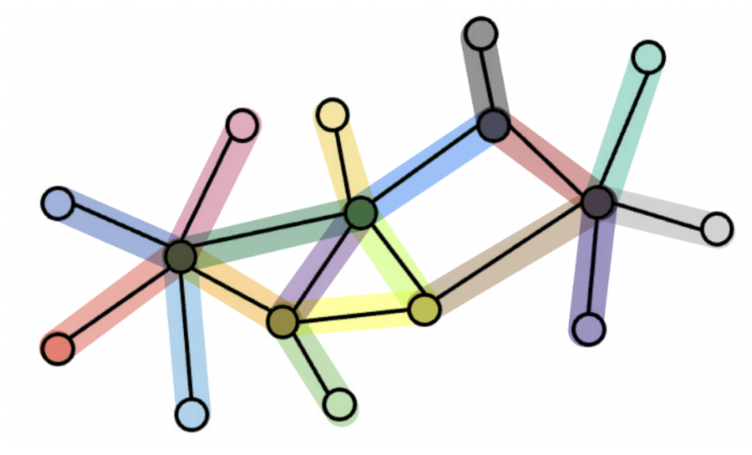


Figure 1.2: This illustrates a toy road network in the form of a graph and the *edge-disjoint* pathlets[1](different colors represent different pathlets).

1.2.1 The Proposed Approach for Similarity Search Tasks

Our work aims to address the limitation of works that ignore the temporal similarity. But even for those that do consider it, we also target to improve model performance. In this work on trajectory similarity, we leverage the set data structure – which to the best of our knowledge is the first to do so due to the approach being not too popular. Indeed, trajectories have a preserved inherent ordering and sets are permutation-invariant in nature. However, we demonstrate that our method that learns the spatiotemporal information of the road segments (as elements) traversed by these trajectories (as sets) suffices to ignore the ordering of the road segments. By expressing each trajectory as a set of (time-enabled) road segments (i.e., decomposing trajectories as sets of the traversed roads, with timestamps), learning the spatiotemporal embedding vectors of each road of the trajectory (see Figure 1.1), and then collectively feeding them to a neural network that respects various set properties, then we can learn effective spatiotemporal representations of trajectories while preserving similarity relations. Similarities of trajectories with these learned representations (called *box embeddings*)

[1]Note that in this specific toy example, all the pathlets here have pathlet length of 1 but in theory could be of longer lengths. See Definition 3.2.3 for more details.

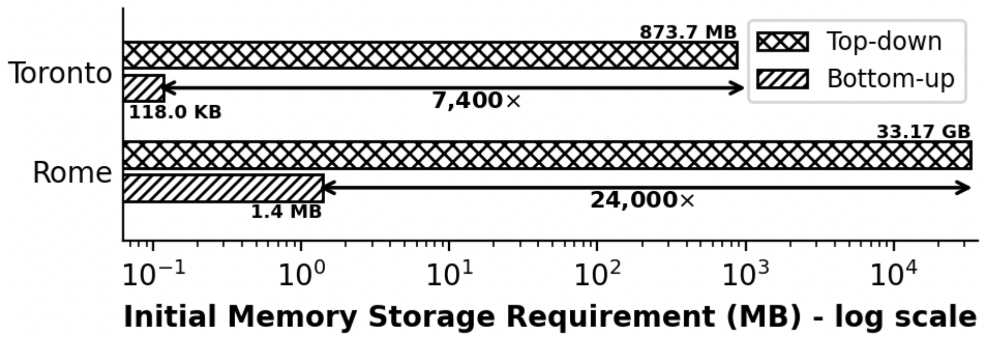


Figure 1.3: The memory required by top-down (existing) methods that use overlapping pathlets can be reduced by our proposed bottom-up solution that use edge-disjoint pathlets.

can be quickly computed, and then we return the top k trajectories in the trajectory set that are most similar to the query trajectory. In other words, the *(spatiotemoral) trajectory similarity problem* is framed as a *road segment-based set similarity problem.*

1.2.2 The Proposed Approach for PD Construction

To address these limitations, we propose a bottom-up approach for constructing a pathlet dictionary that complies with edge-disjoint pathlets (see Figure 1.2) and reduces memory storage requirement. In Figure 1.3 for instance, we illustrate how our proposed approach saves up to $\sim$24K$\times$ less memory space than existing methods for storing the initial pathlets (see Experiment **(RQ 2.2)** for full details, with Appendix B.2 presenting a more theoreotical proof). The key idea of our approach is to initialize unit-length pathlets and iteratively merge them to form longer, higher-order ones, while maximizing utility [3, 94]. Longer pathlets are preferred (over shorter ones) as they hold more spatiotemporal information, such as mobility patterns in trajectories [26]. A deep reinforcement learning method is proposed to approximate the utility function.

1.3 Contributions

We collectively summarize the list of all contributions this thesis paper has to offer:

- To the best of our knowledge, our work is the first to treat trajectories as sets that can be fed into neural networks for learning set similarity relations and supporting set operations in vector space. In particular, the trajectories are expressed as sets of road segments before passing them to the set-specific neural network. Thus, our work is the first to treat the *trajectory similarity problem* as *a set similarity problem*.

- We introduce a deep representation learning method SEG2VEC for projecting road segments onto the spatiotemporal embedding space. In particular, we adapt a similar framework as the work of Fang et al. [39] for learning latent representations of road segments instead of trajectories.

- We propose ST2BOX, for retrieving (and ranking) the top k trajectories in our dataset that are most similar to a query trajectory. The approach finds sets of road segments in the network that are most similar to the segment-based set representation of the query trajectory using a set-to-box architecture [79], specifically for trajectory's road segments.

- We demonstrate through experiments that our method is able to outperform SOTA baselines, by as much as $\sim$38%.

- We revisit the relatively new problem of interest, *pathlet dictionary construction*, where we introduce a more strict definition of a pathlet than in previous works to comply with edge-disjoint pathlets. This enables a bottom-up approach for constructing pathlet dictionaries that reduces memory storage needs.

- We introduce two novel metrics, namely *trajectory loss* and *trajectory representability*, which allow us to more comprehensively evaluate the utility of a pathlet and the overall quality of a constructed pathlet dictionary.

- We formulate the problem of *pathlet dictionary construction* as a utility maximization problem, where shorter pathlets are merged to form a set of longer ones with higher utility.

- We propose PathletRL, a deep reinforcement learning method that utilizes a Deep Q Network (Dqn) policy to approximate the utility function of constructing a pathlet dictionary. To the best of our knowledge, this is the first attempt to employ a deep learning method for the problem.

- We demonstrate empirically that the dictionary constructed by our PathletRL is of superior quality to those constructed by traditional non-learning-based methods. Our method reduces the size of the the dictionary by up to 65.8% compared to other methods. Moreover, using only half of the pathlets in the dictionary suffices to reconstruct 85% of the original trajectory data.

- We make our source code publicly available to encourage the reproducibility of our work (see Appendix F for more details).

1.4 Thesis Organization

For the remainder of this thesis, we first provide in Chapter 2 a thorough review of related works in the literature that are relevant to the study of interest. Then, in Chapter 3, we introduce some preliminary background, including the common notations and definitions throughout this thesis. In the same chapter, we also formalize the problems that we aim to address. Chapter 4 gives a more in-depth discussion on the proposed methods, and then evaluate the performances of these methods in Chapter 5. More specifically, we describe the design of the experimental study – including the datasets used, the evaluation metrics, the various baseline models, the results of the experiments, insightful discussions based on the observed results, and other details relevant to the experimental design. Finally, we end the thesis with some concluding remarks in Chapter 6.

Chapter 2

Literature Review

Research interest on mobility data and their applications has grown over the years, and we discuss in this chapter some of the most notable efforts towards this area. For instance, surveys from [194, 41, 10, 50] have compiled and highlighted the most significant and substantial contributions in the area of trajectory data mining. As deep learning have become hot these days, surveys [152, 154] emphasized works in trajectory data mining that have seen success in the use of deep learning methods.

We organize the following literature review as follows. We begin by conducting an extensive review on works related to trajectory similarity in Chapter 2.1. We then cover works relevant to mining pathlets in Chapter 2.2. Next, we provide some review on a related area of graph mining in Chapter 2.3, and finally cover some related works that are relevant to deep learning methods in general in Chapter 2.4.

2.1 Trajectory Similarity

A popular problem in trajectory data mining is known as the *trajectory similarity* problem; refer to [141, 135, 139] for some relevant surveys. As this thesis also aims to provide some contribution towards this problem of interest, we provide some discussion as well on some of the related existing works – both on traditional [44, 173, 147, 27, 9, 126, 128, 177, 72] and learning-based [172, 85, 171, 186, 45, 185, 170, 53, 39, 61] methods. We first review traditional-based measures and then go over the learning-based ones.

2.1.1 Traditional Methods

Traditional metrics for measuring the similarity of two trajectories such as the Dtw (Dynamic Time Warping) [173], Lcss (Longest Common Subsequence) [147], Hausdroff distance [9], and Erp (Edit Distance with Real Penalty) [27] exist as *free-space* measures. There are also those such as Tp (Two-Phase Algorithm) [126], Dita (Distributed In-Memory Trajectory Analytics) [128], Lcrs (Longest Common Road Segments) [177] and NetErp (Network-aware Edit Distance with Real Penalty) [72] are *network-based* measures. The problem with these similarity measures lies on their computational overhead when measuring similarity. Running times are computationally expensive due to their dependence on quadratic $\mathcal{O}(n^2)$ time pointwise matching schemes, where n is the average trajectory length. This can quickly become a severe bottleneck when working with larger trajectory datasets [171, 186].

2.1.2 Learning-based Methods

To address the main limitations of traditional-based methods, we turn our attention to learning-based models [172, 85, 171, 186, 45, 186, 170, 53, 39, 61], where some deep architecture learns representations of these trajectories in the low-dimensional embedding space while preserving the trajectories' similarity relations. Clearly, two similar trajectories are found proximate each other in the latent space; and moreover, dissimilar trajectories are found farther apart each other. These learning-based methods tend to avoid operating directly on the original trajectories, and as a result have demonstrated significant computational savings. For example, Yao et al. [172] used sliding windows to extract spatial and temporal characteristics that are invariant in trajectories, which provides some information on the trajectories' object movements. Li et al. [85] uses a recurrent neural network-based architecture to preserve the sequential feature in trajectories; they demonstrated that their technique is robust to noisy and non-uniform trajectory points while presenting a fast, efficient approach for similarity calculations in $\mathcal{O}(n + |\mathbf{v}|)$ time (with n as the average trajectory length and $|\mathbf{v}|$ as the embedding vector dimension).

Another important work was Yao et al.'s NeuTraj [171], that used a seed number to sample from a set of trajectories, and then followed by some pairwise-similarity guided loss enhanced by a neural network framework. This architecture is composed of a spatial attention module and a distance-weighted rank loss function to effectively extract information from the seed-sampled trajectories. Zhang et al. [186] proposed Traj2SimVec that is robust and scalable in trajectory similarity learning. The model utilizes a simple, fast trajectory simplification approach [31] with indexing to obtain triplet samples for training: $\langle I_a, I_n, I_f \rangle$ comprising of an anchor input I_a, a near input I_n and a far input I_f, which are randomly sampled from the k nearest neighbors of I_a. These are indexed by the k-d tree [14] and thus yields an efficient $\mathcal{O}(\log n)$ running time. Han et al. [53] also introduced a novel framework called Gts that is able to capture the spatial and topological features of trajectories on the road network using Graph Neural Networks to identify neighboring points of interests (Pois) that belong to the same trajectory. A final Lstm layer extracts the trajectory's sequential information. However, this state-of-the-art model is best designed mostly for Pois; but however, real world trajectories do not necessarily follow the path of Poi sequences. Moreover, all these works mentioned so far ignore the temporal features of trajectories and can be sub-optimal in contexts when the time dimension is essential, such as in personalized trip recommendation applications [151] when real-time information of trajectories is required.

Fu et al. [45] incorporates an encoder-decoder component for extracting the time-based features of trajectories (i.e., the timestamps), in addition to the ROAD2VEC framework for finding representations of road segments for modelling co-relationships between these segments in the road network. Their TREMBR model however does not take temporal dependence, regularities and patterns into account. As such, there is a need for a framework that can model these temporal features. For example, Fang et al. [39] proposed ST2VEC that consists of a novel temporal modeling module for representing the time component in trajectories. This also includes any periodic patterns and temporal regularities in trajectories. ST2VEC also uses a Unified Fusion scheme for allowing interactions between spatial and temporal representations in trajectories. It has been proven to be effective and efficient against state-of-the-art baselines. As we shall see later in our methods, we also adapt a similar architecture in our work to integrate the temporal dimension, but this we do for the road-based representation of trajectories instead of trajectories as a whole as a way of obtaining more refined levels of representations for trajectories. In another work, Jiang et al. [61] also introduced the START model for trajectory representation learning based on temporal regularities and travel semantics in trajectories. While it has demonstrated superior results, their model is trained under self-supervised means (does not utilize ground truth labels).

2.2 Pathlet Mining

There is a number of works related to pathlet mining. One of the original works in this direction is the work of Chen et al. [26], where they formulated the problem of pathlet dictionary construction as an integer programming problem with optimization constraints; they also provided a solution based on dynamic programming. Their constructed dictionary was also tested in various use cases such as trajectory compression and route planning. However, their work assumes that the datasets used are noise-free. Zhou et al. [195] designed a bag of segments representation for motion trajectories, where each trajectory segment is projected onto the codeword space and clustering using the *Expectation Maximization (*Em*)* algorithm is applied. Although, they have showed that their method is compact and expressive on trajectory classification and similarity search tasks, the algorithm requires that trajectory segments are of fixed size. Panagiotakis et al. [108] proposed a method for finding representative subtrajectories through global voting, segmentation and sampling methods. However, their methodology partitions trajectories into subtrajectories that is based on local density – without regard to the trajectories that contribute to the global density. Wang et al. [156] solves the problem of finding top k representative routes that cover as many trajectories as possible under some specific distance threshold. They also proposed three near-optimal solutions (*maximum-weight*, *coverage-first*, and *connect-first* algorithms) that can address the problem efficiently. Despite the efficiency their methods offer, they still require the user to provide some budget constraint B in the route representative discovery task, which is a domain-specific parameter that requires domain expert knowledge.

2.2.1 Subtrajectory Clustering

One common approach to pathlet mining is framed as a subtrajectory clustering problem. Lee et al. [81] proposed the Traculus algorithm that partitions trajectories into line segments, and then groups those partitions that lie in a similar dense region to form a cluster. In particular, they present a two-staged method: (1) the *partition phase* that is based on the minimum description length (Mdl) principle, and (2) the *grouping phase*, a clustering-based algorithm that relies on the density of line segments. However, Traculus assumes pathlets are straight line segments, which is not always the case in real road maps. Van Krevald et al. [74] designed a novel measure for mining median trajectories, similar to the method of Wang et al. [156], that serves as the cluster centroid of (sub)trajectories. More specifically, the methodology aims to find what is known as *majority medians* that is based on useful trajctory edges. The limitation however in their work assumes that input trajectories have the same starting and ending points – which is not always the case for real world trajectories. Agarwal et al. [1] addressed the problem of subtrajectory clustering using the *pathlet cover* method that is motivated from the popular set cover algorithm. In addition, they have demonstrated a theoretical proof for the NP-hardness of the problem and have proposed this approximate algorithm for fast, efficient solution. In all these subtrajectory clustering methods, the cluster centroids are seen as popular segments traversed by many trajectories and can alternatively be seen as the pathlets. However, Wang et al. [156] have demonstrated that the cluster centroids in these (sub)trajectory clustering methods [81, 74, 1, 155] do not necessarily reflect real roads in the road network, and moreover are computationally slow in general.

2.3 Graph Mining

This work is also related to graphs and mining patterns in graphs; surveys from [65, 13] have summarized and compiled together substantial contributions towards this effort. A plethora of works have been focused on this area of research, particularly in popular graph mining tasks such as ones related to frequent graph mining [143], link prediction [188], community detection [43], anomaly detection [2], recommendation systems [100], and mining dynamic temporal graphs [118, 166] among others. Closer to our work is on problems related to edge contraction [8, 48]. We provide some discussion on this topic.

2.3.1 Edge Contraction

Related to our method is edge contraction. In edge contraction in graphs, an edge in the graph is elected to be removed, and then consequently merge the two nodes that connect this edge; moreover, the neighbors of the merged node are updated accordingly [8, 48]. While our proposed method as one will see in Chapter 4 has a similar resemblance, it is not necessarily the same as edge contraction. Edge contraction deals with the removal of an edge in the graph to merge the two nodes connecting that removed edge. However in our method, two edges/pathlets are required for merging and they will not be removed in the network.

A related algorithm involving edge contraction goes as follows: (1) compute for the (edge) betweenness centrality of all edges in the pathlet graph, (2) remove (via edge contraction) the edge with the highest centrality to shorten paths of trajectories on the graph, and (3) repeat the procedure until some termination criterion has been reached (say when the graph has shrunk by at least 20% of the original for instance). While this in theory is a viable baseline model to compare with our methods, it suffers extensively from computational overhead. Calculating the betweenness centrality score is a computationally intensive task, with a $\mathcal{O}(|\mathcal{V}||\mathcal{E}|)$ time complexity under the Brandes' algorithm [19]. As there is a need to do so for all edges in the graph to identify the edge with the highest betweenness score, not to mention of dense large graph datasets with many edges, then the aforementioned scheme does not scale and may not even be feasible to implement.

2.3.2 Machine Learning with Graphs

With the prominence of graphs and their numerous applications, it is important to also gain some understanding on the different methodologies for working with graphs. More recently, learning-based methods has become a popular paradigm in problems involving various data types including but not limited to text [96, 21, 149, 66], images [77, 75, 57, 28, 15], audios [112, 142, 98] and videos [184, 17, 124]. As such, it is only logical that popular machine learning algorithms are also tried on graphs. See [165, 25] for some comprehensive surveys.

Graphs are considered to be unstructured data. Hence, one must perform some preprocessing on these graphs to be able to perform some task such as node classification [93], edge prediction [188], or graph clustering [150]. Usually, graphs are first represented using vector embeddings, and then use these as features to train a model of a particular task of interest. It is more common to learn the representations of nodes in a graph. If one requires learning edge representations, one can simply concatenate the embeddings of the two nodes that are adjacent to the edge. If one however requires representations of the sub/full graph, then one can simply perform some pooling or aggregation, such as AVERAGEPOOL or SUMPOOL, of all embeddings of the nodes to capture the graph's vector representation.

Walk-based methods for graphs have become one of the first learning-based approaches for learning graph representations. Take the PageRank method for example, where each node receives an "importance score" based on its connectivity with other nodes (usually evaluated based on the number of times the node has been visited throughout the random walk process) [20]. Sarma et al. [121] however exposed that traditional style matrix-vector multiplications in PageRank calculations do not adapt well in a distributed setting. Moreover, they first introduced a distributed random walk-based algorithm for calculating PageRank scores in $\mathcal{O}(\log n/\epsilon)$ rounds with high probability for directed graphs, and a faster $\mathcal{O}(\sqrt{\log n}/\epsilon)$ for undirected graphs – with n as the number of nodes in the graph and ϵ as the fixed reset probability constant. Grover et al. [49] also proposed a random-walk simulator among the nodes in the graph, which is then processed with a SKIP-GRAM – as we do in words of a sentence [96]. The proposed NODE2VEC architecture basically learns the structural information by capturing the co-occurrences of adjacent nodes in the graph; i.e., NODE2VEC approximates the conditional probability of nodes in each of its neighborhoods [49].

Convolutional Neural Networks (CNNS) are powerful artificial feedforward networks designed to work well with large-scale image tasks [78]. However, unlike images, graphs do not have a fixed ordering and as such, CNNS do not do well when working with graph data. To handle graphs, a neural network capable of permutation-invariance (i.e., a graph and its permutations mapping to the same representation in the latent space) such as Graph Neural

Networks (GNNs) is required. For instance, Scarselli et al. [122] has shown some promising direction on the use of graph neural networks for computing PageRank scores. Scarselli et al. [123] on this work proposed supervised learning algorithms to learn the architecture's parameters, whilst demonstrating that GNN scales to larger datasets.

More specialized GNNs come with the form of playing around with the aggregation and transformation functions of its message-passing mechanisms. For example, Kipf et al.'s [67] used the MEAN function to aggregate the representations information from each node's neighborhood, and then a non-linear activation function (an MLP) for transforming this aggregation. Their proposed Graph Convolution Networks (GCN) uses a convolution architecture, inspired by a localization of first-order approximation of spectral graph convolutions. Being one of the most influential works in this area, it is able to scale to large graph datasets and outperform state of the art competitors. Hamilton et al. [51] improves on this by proposing GRAPHSAGE that samples neighbors at different hops and pools them together through a MAXPOOL aggregation. In addition, this architecture also learns from node features (e.g., node degrees, node text attributes, etc.) to easily generalize to unseen nodes. Transductive learning from GCNs are extended to inductive learning setting in GRAPHSAGE, much like in Veličković's GAT [145] method. This architecture uses a self-attention strategy and weighs neighbor nodes according to importance.

Meanwhile, graph transformers have also become popular methods. Yun et al.'s [181] GTNs for example extends from the popular GNNs by constructing new graphs from original ones, where it aims to learn useful relationships between linked nodes in the generated graph that would have been absent in the original graph. The algorithm can also learn effective node representations in the generated graph through multi-hop connections called *meta-paths*. Other popular transformer-based architectures specifically designed for graph data include Cai et al.'s [23] model for graph to sequence data, and Ying et al.'s [175] GRAPHORMER from Microsoft. Both these works had used datasets from the well-known OGB (Stanford's Open

Graph Benchmark)[2], a diverse collection of realistic, large-scale benchmark datasets, loaders and evaluators for graph machine learning.

2.4 Deep Learning

In the recent years, deep learning has become a prominent approach for many successful research works due to their effectiveness as demonstrated by their promising results, in various domains and applications in vision [29, 115, 176], natural language processing [12, 22, 69], healthcare [191, 116], recommendation systems [104, 91, 151, 159], and so forth (see Pouyanfar et al. [111] and Dargan et al. [34] for some relevant surveys). Of interest are deep learning methods geared towards trajectory data and spatiotemporal-based architectures. We begin this review with a brief discussion on some popular works on deep representation learning for spatiotemporal data. This is followed up by a literature review on deep reinforcement learning works for spatiotemporal data. And then finally, we end the literature review chapter with a brief discussion on deep learning for sets – that are much closer to this work.

2.4.1 Deep Representation Learning for Trajectories

In recent years, deep learning based methods have been proposed for learning representations of spatiotemporal data. These representations are a good fit for several trajectory data mining downstream tasks. A comprehensive review can be found in Wang et al. [152]. As we have mostly touched on works related to deep representation learning for trajectory similarity learning in Chapter 2.1.2, then we omit them here and focus instead on works that use deep representation learning on other tasks.

For example, Shrivastava et al. [130] have designed an Lstm architecture for analyzing the temporal dimension of trajectories to infer missing trajectory points. They optimized their

[2]`https://ogb.stanford.edu/`

method for faster speedups by applying clustering schemes on similarly-detected trajectories to reduce the search space – thus allowing scalability to larger datasets. Wang et al. [162] has also used a deep representation learning approach for detecting outlier trajectories that exemplify anomalous behaviors. The unsupervised model utilizes an auto-encoder model with deep feature fusion architecture to extract the spatiotemporal characteristics in trajectories. Finally, an unsupervised clustering method is used to identify any anomaly trajectories not belonging to any trajectory clusters. Meanwhile, Chen et al. [30] introduced MainTul that is composed of Rnns for trajectory-user linking tasks. This architecture, enhanced with a temporal-aware transformer, is used to encode trajectories with guidance from a mutual distillation of information. Liu et al. [89] also proposed a graph-based model for identifying individual travel activities based on Pois and spatiotemporal information from trajectories. Their proposed Gstp2Vec architecture utilizes a series of deep feedfoward networks that can generate accurate representations of Pois and trajectory features in the low-embedding space. More recently is Park et al.'s [109] method for future trajectory prediction in vehicles, that centralizes on the idea of *"two vehicles passing by in adjacent lanes are likely to interact at a future timepoint"*. In light of this, they utilized a Graph Convlution Neural Network with message passing mechanisms to approximate these probabilities and to predict future interactions in trajectories.

2.4.2 Deep Reinforcement Learning

There are several works dedicated to reinforcement learning [64, 138, 7, 137, 157]. For instance, some surveys have summarized and compiled together notable contributions in the area of reinforcement learning [64, 7, 157]. Mnih et al. [97] have developed a Q-learning-based convolutional neural network architecture to learn control policies taken from sensory inputs with the use of deep reinforcement learning. The novel methodology is applied to several

Atari 2600 games from the Arcade Learing Environment. Deep neural networks of AlphaGo have been trained on both human-expert expert games (supervised means) and from self-play games (reinforcement learning), as it uses value networks to assess board positions and policy networks to decide on its moves [131]. The novel search algorithm that is based on a combination of Monte Carlo simulations, value networks and policy networks, have demonstrated a 99.8% win rate against other Go programs and moreover can win against a human European champion on a 5-0 score. Silver et al. [132] takes it further by generalizing their algorithm and showcase its significant performance in other games such as Shoji and Chess. Other works focused on evaluating their reinforcement learning models on more sophisticated video games such as Minecraft, Dota2, and Candy Crush [62, 105, 42, 136, 11]. There are also certain works that directed their attention to tie deep reinforcement learning with game theory [129, 56]. Multi-agent reinforcement learning has also gained interest in the recent years, where the model takes into account multiple agents, instead of a single agent [187, 158].

Of particular interest are works related to the spatiotemporal domain. Now while reinforcement learning methods are often evaluated on agents playing a specific game, the research community on trajectory data mining and mobility data in the recent years has actually gained some interest in adapting RL methods and moreover has yielded great success. For example, deep reinforcement learning has been used in route planning [46]; more specifically, Geng et al. devised a dynamically adjusted route planning method called Darp that employs a dueling network that is based on deep Q learning policies to refrain from electing congested roads in learning the most optimal paths. The method was seen to have been able to save travel time by as much as 52% under road congested conditions in several test simulations. Wang et al. [161] addresses the *Min-Error* problem in trajectory simplification tasks, where the problem of interest is modelled as a sequential decision process that is based on a *Markov Decision Process (*Mdp*)*. Their proposed data-driven Rlts algorithm can effectively minimize the error function $\epsilon(\tau')$ while at the same time running efficiently in both online and offline settings, compared to existing works that are mostly heuristics-based in nature. Arasteh et al. [6] uses a multi-agent network aware reinforcement learning model to solve the adaptive vehicle navigation problem; the proposed model can adapt to real-time traffic conditions on the road network while achieving a 17.3% improvement on the average travel time compared to existing greedy *shortest path first (*SPF*)* methods.

2.4.3 Deep Learning for Sets

In this work where sets are deemed important data structures and set-based neural networks being crucial, we provide some literature review on deep learning methods that can specifically handle sets. One of the earliest works that trains a deep network that is of Zaheer et al.'s [182]. Their proposed model called DEEPSETS obeys both the permutation invariance and equivariance properties (to be discussed in more detail in Chapter 4.1.3), while offering flexibility to both regression and classification tasks. They demonstrate its effectiveness in a variety of supervised and unsupervised tasks including population statistics estimation, point cloud classification, text concept set retrieval, image tagging and anomaly detection.

Despite DEEPSETS outperforming its baseline competitors, it suffers in the inherent assumption that all sets in the dataset have a fixed input size, which is not the case in real world scenarios. To improve on this, Zhang et al. [189] proposes a general architecture for predicting sets given variable-sized input sets; it is based on the idea of the permutation equivariance property exhibited by the gradient of a set encoder with respect to its input set. As such, gradient descent was utilized in the decoder to find a set that encodes to the feature vector in the decoder. The method was seen to respect set structure, avoid discontinuity issues[3] altogether, and prove to be effective in various tasks such as point set auto-encoding,

[3]Discontinuity here refers to small changes in the input set space requires the need for a significant change in the neural network outputs [190].

state predictions, and bounding box predictions for object detection. Meanwhile, Skianis et al. [133] also introduced a novel neural network that can handle arbitrary-sized sets of vectors. In particular, the model computes correspondences between the input set and the hidden sets via a series of network flow problems, which then in turn is fed to a fully-connected layer to generate the set representation. Their proposed RepSet has been shown to demonstrate powerful results against baseline methods on text categorization and graph classification tasks. Moreover, Lee et al. [79] presents an accurate, concise, generalizable, versatile, and fast approach for embedding sets that can preserve various set similarity measures, including overlapping coefficient, cosine similarity, jaccard index, and the dice index. In particular, sets are represented as a d-dimensional hyper-rectangle (the box) whose objective is to approximate the volumes of boxes in relation to the set sizes in order to capture and preserve relations with other sets in the latent space. Integrating this method to our proposed model, these desirable characteristics are thereby inherited and makes our method effective.

On a theoretical perspective, Wagstaff et al. [148] demonstrated how arbitrary functions on sets on a finite latent space has some limitations. For instance, they have shown that under the continuity constraint, it suffices (and is necessary) for the dimension of a latent space to be at least as large the maximum size of the input set in order to universally represent functions on sets.

Chapter 3

Preliminaries

In this chapter, we introduce some definitions used throughout the work. See Table 3.1 for a table listing these notations. At the end of the chapter, we formalize the problems of interest.

3.1 Preliminary Definitions

A few of the most primary definitions include the *trajectory* and the *road network*, which serves as the most used *terminologies* in this work.

Definition 3.1.1 (Trajectory)**.** Let $\mathbf{O} = \{o_1, o_2, ..., o_{|\mathbf{O}|}\}$ be a set of moving objects in a certain geographic map $\mathcal{M} \subset \mathbb{R}^2$. A *trajectory* τ of a single object $o \in \mathbf{O}$ can be represented as a sequence of time-enabled geo-coordinate points:

$$\tau = \left\langle (x_1, y_1, t_1), ..., (x_{|\tau|}, y_{|\tau|}, t_{|\tau|}) \right\rangle \tag{3.1}$$

where each x_i and y_i represents o's longitudinal and latitudinal coordinates at a specific time instance $t_i \in [0, T]$. Here, $|\tau|$ denotes the length of a trajectory, or the number of time-enabled points for the trajectory of o.

Symbol	Definition
$\mathbf{O}$	Set of moving objects
o	A single object; i.e., $o \in \mathbf{O}$
$\mathcal{M}$	Geographic area of interest (map)
τ	The trajectory of a single object o; a sequence of time-enabled geocoordinates: $\tau = \left\langle (x_1, y_1, t_1), ..., (x_{\lvert\tau\rvert}, y_{\lvert\tau\rvert}, t_{\lvert\tau\rvert}) \right\rangle$
$\mathcal{T}$	Set of trajectories (trajectory data set)
r	A road segment
$\mathbf{R}$	A set of road segments
$\mathcal{G}\langle\mathcal{V}, \mathcal{E}\rangle$	Road network represented as a graph with node set $\mathcal{V}$ (road intersections) and edge set $\mathcal{E}$ (road segments)
$\mathfrak{R}(\tau)$	The road segment-based representation of trajectory τ
ρ	A pathlet
$\mathcal{P}$	A pathlet set; i.e., $\rho \in \mathcal{P}$
$\rho.s$, $\rho.e$	The starting and ending points of a pathlet ρ
$\ell(\rho)$	The pathlet length of ρ
χ	The maximum pathlet length in pathlet set $\mathcal{P}$; i.e., the χ-order pathlet set
$\mathcal{G}_p\langle\mathcal{V}_p, \mathcal{E}_p\rangle$	The pathlet graph with node set $\mathcal{V}_p$ (road intersections) and edge set $\mathcal{E}_p$ (road segments)
$\Phi(\tau)$	The pathlet-based representation of a trajectory τ
$\Lambda(\rho)$	The trajectory traversal set of a pathlet ρ
$\omega(\rho)$	The weight of pathlet ρ in the road network
$\mu(\tau)$	The trajectory representability of trajectory τ
$\bar{\mu}$	The average trajectory representability of all trajectories $\tau \in \mathcal{T}$
$\hat{\mu}$	The average trajectory representability threshold
L_{traj}	The trajectory loss
M	The maximum trajectory loss
$\mathbb{S}$	The extracted pathlet dictionary (PD); keys – candidate pathlets, values – the trajectory traversal set of the pathlets
ϕ	The average number of pathlets representing each trajectory in the trajectory set
α_i	Objective weights for the pathlet dictionary construction problem
γ	Discount rate factor
$\mathcal{S}$, $\mathcal{S}^{(s)}$, $\mathcal{S}^{(t)}$	The spatiotemporal, spatial, and temporal similarity functions
$\tau^{(s)}$ and $\tau^{(t)}$	The spatial and temporal aspects of trajectory τ
θ	The spatiotemporal weight parameter for $\mathcal{S}$
$\mathbf{x}$ and $\mathbf{z}$	Embedding representations for trajectory τ and pathlet ρ
k	The top k trajectory similarity search task
β	Box smoothing parameter

Table 3.1: Summary table of notations used in this work

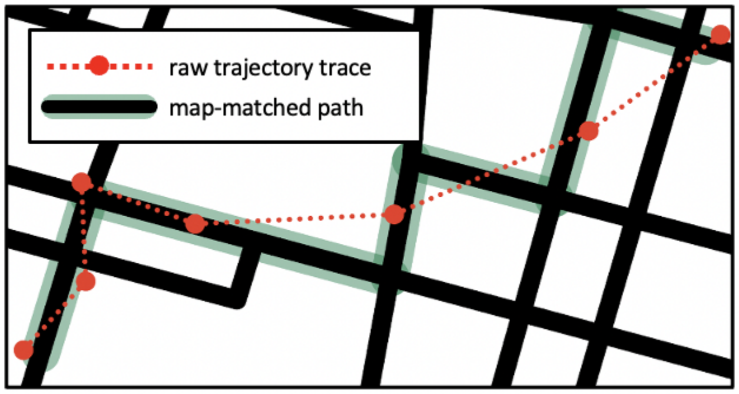


Figure 3.1: Toy example of map-matching on the road network

Definition 3.1.2 (Trajectory Set)**.** The *trajectory (data) set*, denoted as $\mathcal{T}$, consists of all the trajectories of all objects $o \in \mathbf{O}$, which is represented as:

$$\mathcal{T} = \bigcup_{\forall o \in \mathbf{O}} \mathcal{T}_o \tag{3.2}$$

with $\mathcal{T}_o$ as the set of all o's trajectories.

Definition 3.1.3 (Road Segment)**.** Let r denote a road segment[4] that connects two road intersections on the map $\mathcal{M}$, and denote the collection of all road segments to be $\mathbf{R}$.

Definition 3.1.4 (Road Network)**.** We denote by $\mathcal{G}\langle\mathcal{V}, \mathcal{E}\rangle$ the *road network* within the boundaries of some map $\mathcal{M}$, where $\mathcal{V}$ represents its set of road intersections (nodes) and $\mathcal{E} = \mathbf{R} \subseteq \mathcal{V} \times \mathcal{V}$ represents its set of road segments (edges).

Trajectory points (GPS traces) outside map $\mathcal{M}$ are automatically filtered out as a preprocessing step; in addition, the remaining trajectories require to be map-matched, a common preprocessing task that identifies the path on the road an object has taken given a sequence of GPS locations [87, 102] (see Figure 3.1 for an illustrative example). Ideally, highly accurate map-matched data from GPS trajectory traces are preferred; however, this

[4]We will refer to this object as either a *road*, a *segment*, or a *road segment* and use these terms interchangeably throughout the thesis.

task is itself very involved and is outside the scope of the thesis; thus, some existing methods [90, 180, 106, 63] have been utilized to handle map-matching. Provided the map-matched trajectories, we now describe a special representation for a trajectory in the context of a road network.

Definition 3.1.5 (Road Segment-based Representation of a Trajectory)**.** A trajectory $\tau \in \mathcal{T}$ can be represented as a set of road segments $\mathbf{R}_s \subseteq \mathbf{R}$, the elements of which can be concatenated in a sequence that forms the path τ on $\mathcal{G}$:

$$\mathfrak{R}(\tau) = \{r^{(1)}, ..., r^{(|\mathbf{R}_s|)}\}$$

where $r^{(i)} \in \mathbf{R}_s$ is the ith segment in the sequence that represents the *(road) segment-based representation* for τ.

3.2 Problem Statements

This thesis aims to solve the main problem, *trajectory similarity learning*, based on the context of a road network. We first describe what does similarity entail in trajectories, then formalize our main problem of interest in Chapter 3.2.1. In addition, we also attempt to address a related problem in trajectory data mining that is still a fairly new area called *trajectory pathlet dictionary construction*, which we formalize in the succeeding chapter.

3.2.1 Trajectory Similarity Learning

Definition 3.2.1 (Similarity)**.** The (spatiotemporal) similarity[5] $\mathcal{S}$ of two trajectories τ_i and τ_j can be measured based on the dot product of their (spatiotemporal) vector representations

[5]The terms *trajectory similarity* and *spatiotemporal similarity* are used interchangeably in this thesis.

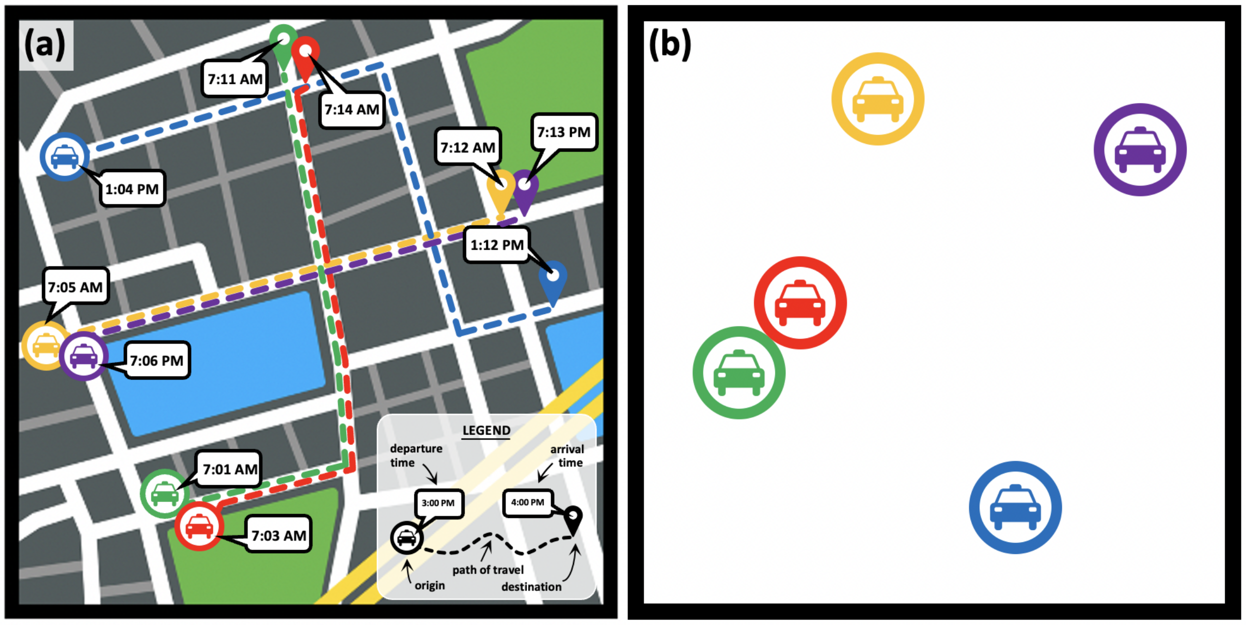


Figure 3.2: (a) Five different trajectories in the form of taxi trips in a small road network labelled with times of departure and arrival from their origins to their destinations; (b) A visualization of a possible 2D embedding space for the trajectories in (a) that preserves spatiotemporal similarity.

$\mathbf{x}_i$ and $\mathbf{x}_j$ in the spatiotemporal embedding space, i.e.,

$$\mathcal{S}(\tau_i, \tau_j) = \mathbf{x}_j^\top \mathbf{x}_i \tag{3.3}$$

Clearly, any two similar (dissimilar) trajectories have trajectory embeddings that are close (distant) in the spatiotemporal embedding space.

Example 3.2.1 Figure 3.2(b) provides a potential example of the 2D latent space that can preserve similarity in the taxi trajectories of Figure 3.2(a). As an example, consider the trajectories of the red and green taxis. They are temporally similar because they relatively departed/arrived their origins/destinations at the same time. Moreover, they are spatially similar because they both took the same route; as a result, it can be said that the red and green trajectories are spatiotemporally similar and that their embeddings appear closest in the embedding space. The orange and purple trajectories are only spatially similar, because while they share the same route, their trips are twelve hours apart; as such their embeddings' distance in the latent space is not as close as the red and green's. Meanwhile, the red and orange trajectories are only temporally similar; while their trip times are relatively close in time, they clearly do not have the same travel paths, origins and destinations. Finally, the trajectory of the blue taxi is the farthest because it is not spatially nor temporally similar with any of the other four trajectories.

Since this work focuses on *network-aware* spatiotemporal similarity of trajectories, it is natural to measure $\mathcal{S}(\cdot,\cdot)$ in the context of *network-aware evaluation metrics*, such as Tp [126], Dita [128], Lcrs [177], and NetErp [72]. These metrics serve as the various similarity measures that we would like our method to estimate accurately, or otherwise serve as the "*ground truth(s)*" of our problem. Note as well that while these measures are spatially-inherent in nature, they are still able to support temporal similarity [39]. As this work aims to provide support to both spatial and temporal domains, we make use of a spatiotemporal weight parameter $\theta \in [0, 1]$ to determine the importance of the spatial/temporal aspects in the similarity measures:

$$\mathcal{S}(\tau_i, \tau_j) = \theta \cdot \mathcal{S}^{(s)} \left(\tau_i^{(s)}, \tau_j^{(s)}\right) + (1-\theta) \cdot \mathcal{S}^{(t)} \left(\tau_i^{(t)}, \tau_j^{(t)}\right) \tag{3.4}$$

with superscript $*^{(s)}$ and $*^{(t)}$ denoting spatial and temporal aspects respectively.

Now, the similarity of two trajectories τ_i and τ_j can be calculated by aggregating the similarities of the road segments that represent these trajectories in the embedded space:

$$\mathcal{S}(\tau_i, \tau_j) = \text{AGG}\left(\{\mathcal{S}(r_i, r_j) \,|\, (r_i, r_j) \in \mathfrak{R}(\tau_i) \times \mathfrak{R}(\tau_j)\}\right) \tag{3.5}$$

This approach aligns with previous works that are based on point-wise similarity computation

[85, 53]. However, this approach is computationally expensive, where we expect a quadratic running time in the average number of road segments that represent each trajectory. To avoid this limitation, in our research, we treat the trajectory similarity problem as a *set similarity problem*. The set similarity occurs at the embedded spatiotemporal vector space and utilizes a method for *similarity-preserving set representations of trajectories*, as we will describe in Chapter 4.

We are now in position to formally define the problem of interest. Note that while there are many flavors for the problem of trajectory similarity computation, we instead focus on the top k trajectory similarity search task.

Problem 3.2.1 (Top-k Similarity Search) Given a trajectory set $\mathcal{T}$ in the road network $\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$, a query trajectory τ_q, and a positive integer $k \geq 1$, find the ranked list of the top k trajectories $\{\tau^{(1)}, \tau^{(2)}, ..., \tau^{(k)}\} \subseteq \mathcal{T}$ that are the most spatiotemporally similar to τ_q. In other words:

$$\mathcal{S}(\tau_q, \tau^{(k)}) \leq \mathcal{S}(\tau_q, \tau^{(k-1)}) \leq ... \leq \mathcal{S}(\tau_q, \tau^{(1)}) \tag{3.6}$$

where $\tau^{(i)}$ is the ith trajectory in $\mathcal{T}$ that is most similar to τ_q.

3.2.2 Trajectory Pathlet Dictionary Construction

While the main problem of interest in this thesis is the retrieval of similar trajectories to a given query based on learned spatiotemporal similarities, we also tackle another related trajectory data mining problem that involves the construction of a data structure that can widely represent trajectories in the dataset. This data structure, particularly a dictionary, consists of abstract, ubiquitous objects in the context of the road network, known as *(trajectory) pathlets*. These pathlets serve as the fundamental building blocks in this second problem of interest. We first formalize these definitions in brief before formally defining this fairly new problem of interest.

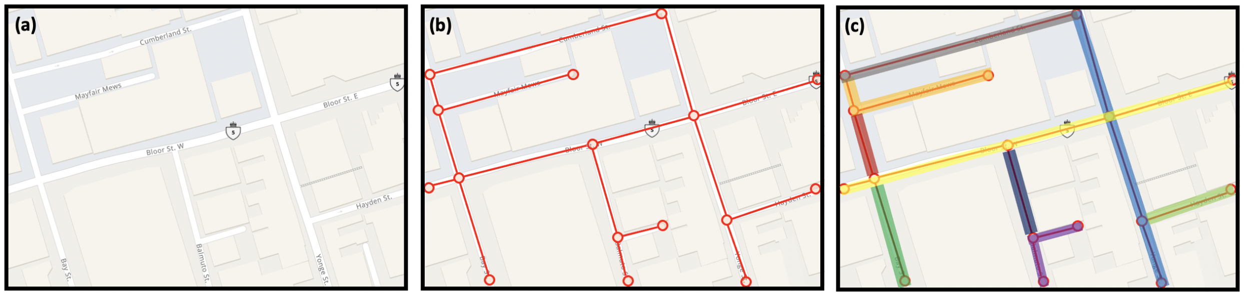


Figure 3.3: (a) A small area of the map in Downtown Toronto[7], (b) Graph representation of the road network of the map in (a), (c) Discrete pathlets of various lengths based on the graph representation in (b).

Definition 3.2.2 (Pathlet)**.** A *pathlet* ρ is defined as any sub-path in the road network $\mathcal{G}$, with $\mathcal{P}$ being the set of all such pathlets.

In our work, we consider *edge-disjoint* pathlets, such that no two $\rho_1, \rho_2 \in \mathcal{P}$ share any edge. For simplicity, we assume *discrete* pathlets – meaning they begin and end at an intersection (a node in the graph, with either start/endpoints at $\rho.s/\rho.e$), but the work can easily be generalized to include *continuous* pathlets that drop this restriction.

Definition 3.2.3 (Pathlet Length)**.** The *(pathlet) length*[6] of a pathlet $\rho \in \mathcal{P}$, denoted by ℓ, represents its path length in the road network.

The smallest unit of the pathlet has length $\ell = 1$. Moreover, we restrict all pathlets $\rho \in \mathcal{P}$ to be of length $\ell \leq \chi$, for some user-defined parameter χ. In this case, we say that $\mathcal{P}$ is an χ**-order pathlet set**. Now given these pathlets, one can well-define a *pathlet graph* $\mathcal{G}_p$ that is derived from a road network $\mathcal{G}$.

Definition 3.2.4 (Pathlet Graph)**.** The *pathlet graph* $\mathcal{G}_p\langle\mathcal{V}_p, \mathcal{E}_p\rangle$ of a road network $\mathcal{G}\langle\mathcal{V}, \mathcal{E}\rangle$ depicts the road network's pathlets, where the road intersections represent the nodes $\mathcal{V}_p \subseteq \mathcal{V}$ and the road segments connecting road intersections as the edges $\mathcal{E}_p \subseteq \mathcal{E}$.

[6]Not to confuse with a road segment's length that represents the measure depicting its actual physical distance, the pathlet length can be derived based on graph context.
[7]Map taken from `https://www.mapquest.com/`

To illustrate a simple example, see Figure 3.3 that depicts (a) a picture of a road network (a small area in Downtown Toronto), (b) its graph representation, and (c) some pathlets of various lengths. In this work, the pathlet graph was initialized to have each pathlet to be of the smallest possible unit, i.e., a pathlet length $\ell = 1$. In other words, all road segments are initially considered as the initial pathlets as they all have pathlet length $\ell = 1$. Now that we consider a pathlet graph, it may be useful to also properly define the neighbors of a pathlet.

Definition 3.2.5 (Pathlet Neighbors)**.** Given a pathlet $\rho \in \mathcal{P}$, its neighbor pathlets, denoted by set $\Psi(\rho)$, is the collection of all other pathlets $\rho' \in \mathcal{P} \setminus \{\rho\}$ who share the same starting/ending points with that of ρ:

$$\Psi(\rho) = \bigcup_{\substack{\rho' \in \mathcal{P} \setminus \{\rho\} \\ \rho.s \in \{\rho'.s, \rho'.e\} \vee \rho.e \in \{\rho'.s, \rho'.e\}}} \rho' \tag{3.7}$$

Example 3.2.2 To give an illustrative example of what it means for two pathlets to be neighboring, consider Figure 3.3(c). Here, the grey pathlet is a neighbor of the orange pathlet as they share the same start/end node. The same can be said for the grey and blue pathlet. Thus in this example, the grey pathlet has a total of two neighbors. The yellow and the blue pathlets, despite intersecting at some node or road intersection, are not neighboring pathlets because none of their starting/ending points coincide.

Similar to the road segment-based representation of a trajectory, a trajectory can also be expressed in terms of pathlets through the *pathlet-based representation of a trajectory* that follows a similar idea.

Definition 3.2.6 (Pathlet-based Representation of a Trajectory)**.** A trajectory $\tau \in \mathcal{T}$ can be represented based on some subset of pathlets $\mathcal{P}_{sub} \subseteq \mathcal{P}$. Moreover, the pathlets in $\mathcal{P}_{sub}$ can be concatenated in some sequence resulting into the path traced by τ on the road network $\mathcal{G}$. We denote this by:

$$\Phi(\tau) = \{\rho^{(1)}, \rho^{(2)}, ..., \rho^{(|\mathcal{P}_{sub}|)}\} \tag{3.8}$$

where $\rho^{(i)} \in \mathcal{P}_{sub}$ denotes the ith pathlet in the sequence that represents the pathlet-based representation for τ.

Now based on Defintion 3.2.6, it is also possible to define a trajectory's pathlet length. This is initially set up before constructing the pathlet graph. Each trajectory $\tau \in \mathcal{T}$ would have a pathlet length equal to

$$\ell(\tau) = \sum_{\forall \rho \in \Phi(\tau)} \ell(\rho) \tag{3.9}$$

whose value remains static for the rest of our algorithm, i.e., the trajectory's pathlet length is the sum of the pathlet lengths of each of the pathlet it traversed throughout the observation period $[0, T]$.

We now define the *trajectory traversal set* of a pathlet, which is a related concept to the pathlet-based representation of a trajectory.

Definition 3.2.7 (Trajectory Traversal Set of a Pathlet)**.** Let $\Lambda(\rho)$ be the set of all trajectories $\tau \in \mathcal{T}$ that *pass* or *traverse* pathlet $\rho \in \mathcal{P}$, which can also be written as:

$$\Lambda(\rho) = \{\tau \,|\, \forall \tau \in \mathcal{T},\, \rho \in \Phi(\tau)\} \tag{3.10}$$

We can also assign weights ω to pathlets. In the unweighted case, all pathlets are weighed equally; while in the weighted version, pathlets are weighed equal to the number of trajectories traversing a pathlet, i.e., $\omega(\rho) = |\Lambda(\rho)|$, or $\frac{|\Lambda(\rho)|}{|\mathcal{T}|}$ when normalized. These weights indicate each pathlet's importance in the road network/pathlet graph.

Next, we introduce two novel metrics related to trajectories, namely the *trajectory representability* and the *trajectory loss*, allowing for a better and more comprehensive evaluation of our pathlets and pathlet dictionaries.

Definition 3.2.8 (Trajectory Representability)**.** The (trajectory) representability $\mu \in [0\%, 100\%]$ of a trajectory τ denotes the percentage of trajectory τ that can be represented using pathlets in pathlet set $\mathcal{P}$.

Now clearly, the pathlet-based representation of τ is directly related to its representability, i.e.,

$$\mu(\tau) = \frac{|\Phi(\tau)|}{\sum_{\forall \rho \in \Phi(\tau)} \ell(\rho)} = \frac{|\Phi(\tau)|}{\ell(\tau)} \tag{3.11}$$

However, in the weighted case, some pathlets are weighed more important than others. Thus, a trajectory's representability may need to be redefined based on those weights, i.e.,

$$\mu(\tau) = \sum_{\forall \rho \in \Phi(\tau)} \omega(\rho) \tag{3.12}$$

Definition 3.2.9 (Trajectory Loss)**.** We define the *trajectory loss* L_{traj} to be the number of trajectories $\forall \tau \in \mathcal{T}$ that have representability value $\mu = 0\%$, i.e.,

$$L_{traj} = |\{\tau | \tau \in \mathcal{T},\, \mu(\tau) = 0\}| \tag{3.13}$$

We can also describe these trajectories as "lost" or "discarded" from the given trajectory set $\mathcal{T}$, and we may also depict this number as a percentage.

See Example 4.2.1 of Chapter 4.2.1 for an example that depicts how trajectory representabilities and trajectory loss metrics are calculated. The relevance and impact of these two metrics will become clear as we go over the methodology in finer details.

[7]Not to be confused with *representativeness* that describes the capability of a trajectory to represent other similar nearby trajectories, *representability* depicts how much a trajectory can be reconstructed given our pathlets.

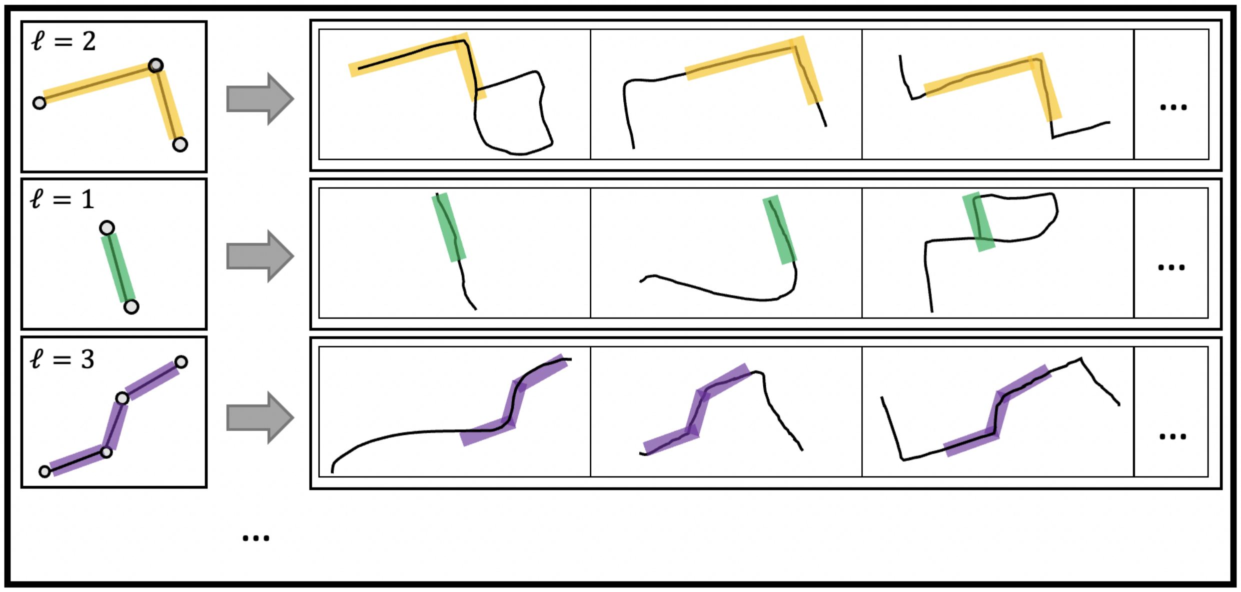


Figure 3.4: An illustrative example of a pathlet dictionary, with pathlets as keys and the trajectory traversal set as the values. The number on the upper left of each pathlet denotes its pathlet length.

Now that we have introduced the basic concepts of pathlets, we now formalize the problem of constructing trajectory pathlet dictionaries after describing this specialized data structure.

Definition 3.2.10 (Pathlet Dictionary)**.** A (trajectory) pathlet dictionary (PD) is a data structure that stores pathlets $\rho \in \mathcal{P}$ (keys), and their associated trajectory traversal set $\Lambda(\rho)$ (values).

See Figure 3.4 for an illustrative example of a pathlet dictionary. However, what we are interested in is the construction of a PD that aims to achieve one or a combination of the following objectives where their mathematical expressions are depicted in Table 3.2:

(O1) Minimal size of candidate pathlet set $\mathbb{S}$, or the candidate set with the least possible number of pathlets

(O2) Minimal ϕ, or the average number of pathlets representing each trajectory $\tau \in \mathcal{T}$

(O3) Minimal trajectory loss L_{traj}

Objective	**Mathematical Notation**	**Associated Weight**
(O1)	$\min \lvert\mathbb{S}\rvert$	α_1
(O2)	$\min \phi = \min \frac{1}{\lvert\mathcal{T}\rvert} \sum_{\tau \in \mathcal{T}} \lvert\Phi(\tau)\rvert$	α_2
(O3)	$\min L_{traj}$	α_3
(O4)	$\max \bar{\mu} = \max \frac{1}{\lvert\mathcal{T}\rvert} \sum_{\tau \in \mathcal{T}} \mu(\tau)$	α_4

Table 3.2: Mathematical notations and the associated weights for each of the terms in the objective function for the trajectory pathlet dictionary construction problem

(O4) Maximal $\bar{\mu}$, or the average representability values of the remaining trajectories in $\mathcal{T}$

In other words, the objective function that we aim to optimize is based on the four objectives above – which can be modelled by:

$$\min \left(\alpha_1 |\mathbb{S}| + \alpha_2 \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} |\Phi(\tau)| + \alpha_3 L_{traj} - \alpha_4 \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \mu(\tau) \right) \tag{3.14}$$

where $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$, are user-defined parameters depicting the weights of our four objectives, with $\sum_{i=1}^{4} \alpha_i = 1$.

Problem 3.2.2 (Pathlet Dictionary Construction) Given a road network $\mathcal{G}\langle\mathcal{V}, \mathcal{E}\rangle$ of a specific map $\mathcal{M}$, a trajectory set $\mathcal{T}$, the maximum pathlet length χ, the maximum trajectory loss M, and the average trajectory representability threshold $\hat{\mu}$, construct a χ-order pathlet dictionary $\mathbb{S}$. The dictionary $\mathbb{S}$ consists of edge-disjoint pathlets with lengths of at most χ, and achieves the maximum possible utility according to some utility function as depicted in Equation (3.14), such that the constraints on trajectory loss ($L_{traj} < M$) and trajectory representability ($\bar{\mu} \geq \hat{\mu}$) are satisfied.

Chapter 4

Methodology

In this chapter, we split the discussion into two parts: (1) the discussion for the **St2Box** model (**<u>S</u>**patiotemporal **<u>T</u>**rajectories to **<u>BOX</u>** embeddings for similarity learning), that captures the representations of trajectories for learning the (ranked) list of top k trajectories that are spatiotemporally similar with the query trajectory; and (2) the details of the proposed **PathletRL** architecture (**<u>Pathlet</u>** dictionary construction using trajectories with **<u>R</u>**einforcement **<u>L</u>**earning), that is responsible for learning trajectory pathlet dictionaries.

4.1 St2Box: Similarity Learning for Trajectories

To address the problem of interest for learning trajectory similarities, we propose **St2Box** (**<u>S</u>**patiotemporal **<u>T</u>**rajectories to **<u>BOX</u>** embeddings for similarity learning), that is designed for using edges of the road network (the road segments) in retrieving the (ranked) list of top k trajectories that are spatiotemporally similar with the query trajectory. More specifically, our algorithm (**1**) learns the spatiotemporal vector representation for each road segment in the network and expresses each trajectory as a set of these vectors, (**2**) uses box architecture to represent these sets of spatiotemporal vectors in a low-dimension embedding space that can

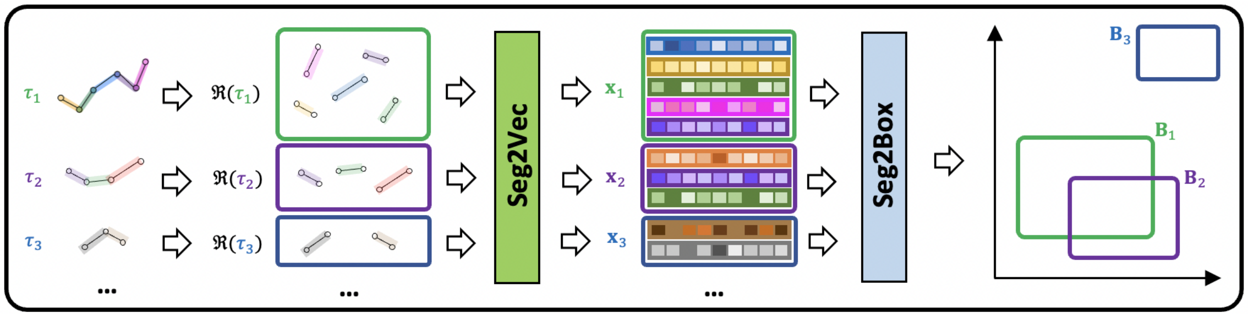


Figure 4.1: The overall architecture of St2Box.

preserve set similarity relations, and (**3**) returns the top k trajectories whose box embeddings possess the highest similarity scores with that of the query trajectory's embeddings. Figure 4.1 illustrates the architecture of St2Box, Algorithm 4.1 provides its pseudocode, and Chapter 4.1.1 for its algorithmic details.

Overall Picture. The method first extracts the road-based representation set $\mathfrak{R}(\tau)$ of each trajectory τ. Then it utilizes a method similar to St2Vec [39], but instead of learning spatiotemporal representations of whole trajectories, it learns representations of road segments. In other words, we learn *more refined spatiotemporal representations of trajectories* using their decompositions (the road segments). We call this architecture Seg2Vec as it learns road-segment based representations of trajectories. Once a trajectory is expressed as a set of learned representations of road segments, it is fed into Seg2Box. Seg2Box is an architecture that is based on Set2Box [79] – an architecture that can learn set representations that preserve set (similarity) relationships by using (smoothed) box embeddings. Seg2Box returns the box representations of trajectories that are used for similarity learning.

Example 4.1.1 In Figure 4.1 for example, we can see that trajectories τ_1 and τ_2 share some common road segments in their road-based representations and so their box representations $\mathbf{B}_1$ and $\mathbf{B}_2$ in the latent space are overlapping. However, τ_3 does not share any road segment with any of the other two trajectories; thus, its box embedding in the latent space is not overlapping with the box embeddings of the other trajectories.

Organization. We first provide the algorithmic details of St2Box in Chapter 4.1.1. We then discuss how the representations of the road segments traversed by trajectories can be learned in Chapter 4.1.2. Chapter 4.1.3 then provides a brief explanation on how trajectories can be viewed as sets – in preparation for Chapter 4.1.4, where we finally cover the details of the box architecture.

4.1.1 Algorithmic Details of St2Box

We now provide the algorithmic details of our method (see Algorithm 4.1). We first begin by initializing some empty dictionaries for easy lookups on demand: (1) D_{vec}, which will store for each road segment, its spatiotemporal vector representation; (2) S, which will keep sets of trajectories, and for each one of those the set of their segment-based representations (expressed as their spatiotemporal representations); (3) S_{score}, which will maintain the similarity scores of each trajectory and the query trajectory (`line 1`). Next, we adapt the method of Fang et al. [39] for representing road segments (instead of trajectories) as their spatiotemporal embeddings and store them in D_{vec} (`lines 2-4`). Following this, we express each trajectory in the trajectory dataset $\mathcal{T}$ as a set of road segments; more specifically, these segments are to be expressed in terms of their spatiotemporal representation obtained from `line 3` and are easily accessible from D_{vec}. Moreover, we store these in dictionary S for easy access on demand (`lines 5-9`). Then, we train a model $\mathbf{B}_{model}$ (see Chapter 4.1.4 for details on this model [79]), where the output of `lines 5-9` is taken as the input set of this model (`line 10`). We then use this trained $\mathbf{B}_{model}$ for finding a box embedding for the query trajectory τ_q using the set of its roads as input (`line 11`). Next, we can simply encode each of the trajectories in the box embedding space, and then compute the similarity score of such encoding with the box embedding of the query trajectory; the scores are stored in S_{score} (`lines 12-14`).

Algorithm 4.1: Spatiotemporal Trajectories to Box Embeddings for Similarity Learning (St2Box)

```
Input   : The road network $\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$, a trajectory set $\mathcal{T}$, a query trajectory $\tau_q$, and an
          integer $1 \leq k \leq |\mathcal{T}|$
Output  : A list of top $k$ trajectories in $\mathcal{T}$ ranked according to their spatiotemporal
          similarity scores with $\tau_q$
/* Initialization */
1  $D_{vec} \leftarrow$ Dict() ; $S \leftarrow$ Dict() ; $S_{score} \leftarrow$ Dict()
/* Learn the embeddings of each road segment in $\mathcal{E}$, and store these in the dict
   $D_{vec}$ */
2  foreach $r \in \mathcal{E}$ do
3      $\mathbf{z} \leftarrow$ Seg2Vec($r$)
4      $D_{vec}[r] \leftarrow \mathbf{z}$
/* Retrieve the embeddings of the roads that represent each traj $\tau$, and store
   them in a dict $S$ */
5  foreach $\tau \in \mathcal{T} \cup \{\tau_q\}$ do
6      $R \leftarrow \varnothing$
7      foreach $r \in \mathfrak{R}(\tau)$ do
8          $R \leftarrow R \cup \{D_{vec}[r]\}$
9      $S[\tau] \leftarrow R$
/* Train a neural network for representing sets as box embeddings */
10 $\mathbf{B}_{model} \leftarrow$ Seg2Box($S \setminus \{\tau_q\}$)
/* Encode the query traj's road-based set representation based on the trained
   $\mathbf{B}_{model}$ */
11 $\mathbf{q} \leftarrow$ encode($\mathbf{B}_{model}, S[\tau_q]$)
/* Encode all other $\tau \in S$ based on trained $\mathbf{B}_{model}$ for sim.  comp.  Store results
   in the dict $S_{score}$ */
12 foreach $\tau \in S \setminus \{\tau_q\}$ do
13     $\mathbf{t} \leftarrow$ encode($\mathbf{B}_{model}, S[\tau]$)
14     $S_{score}[\tau] \leftarrow \mathcal{S}(\mathbf{t}, \mathbf{q})$
/* Return the top $k$ trajectories with the highest similarity scores in the dict
   $S_{score}$ */
15 sort($S_{score}$.items(), key = lambda $\mathbf{x} : \mathbf{x}[1]$)
16 return list(zip($*S_{score}$))[0][$-k$ :]
```

Finally, we can simply return the top k trajectories in S_{score} that achieve the highest similarity with the query trajectory τ_q, as determined by the degree of their box overlaps in the box embedding space (`lines 15-16`).

4.1.2 Spatiotemporal Representation Learning of Road Segments

We introduce the Seg2Vec module, a deep learning framework for representing road segments as embedding vectors that capture their spatiotemporal characteristics. This component in St2Box follows a similar approach to Fang et al.'s St2Vec module [39], but operates at the level of road segments instead of trajectories as a whole. In other words, we split trajectories to road segments, and then learn the spatiotemporal representations of each of the road segments; each trajectory is formed by its set of associated road segment representations.

The Spatial Dimension. To capture the spatial features of road segments representing the trajectories, we introduce the (road) **S**egment-based **S**patial **R**epresentation **L**earning (SegSrl) module. SegSrl utilizes Node2Vec [49] to model co-occurrence of adjacent locations (roads) on the road network. More precisely, we can assign a "representative point" in each of these road segments (either one of the two endpoints/intersections of a road on the road network) to serve as the "location points" and then employ Node2Vec on each of those. After such transformation, we obtain the following location embedding: $r \rightarrow \mathbf{r}$, for all our road segments. Moreover, we apply a graph convolutional network (Gcn) [68] for local smoothing of these location embeddings:

$$\tilde{\mathbf{r}} = \textsc{Gcn}(\mathbf{r}) = \sigma\left(\left(\sum_{r' \in \mathcal{N}(r)} a \cdot W_d \mathbf{r}'\right) \otimes \mathbf{r}\right)$$

where $\sigma(\cdot)$ is a non-linear activation function, $a \in A$ is an adjacency weight value, $W_d \in \mathbb{R}^{d \times d}$ is a learnable weight matrix, $\otimes$ is the concatenator operator, and $\mathcal{N}(r)$ are the neighbors of r (i.e., adjacent edges/road segments of r). Finally, these embeddings are passed on to an LSTM [58] with self-attention [144], which serves as the spatial embedding of our road segments: $\tilde{\mathbf{r}} \rightarrow \mathbf{z}^{(s)}$. Note that this final step is necessary as different location points or roads in trajectories have different contributions towards similarity computations.

The Temporal Dimension. To capture the rich temporal features of trajectories based on their road-based representation (i.e., the time embedding of these trajectories traversing the road segments), we introduce the (road) **S**egment-based **T**emporal **R**epresentation **L**earning (SEGTRL). SEGTRL employs a BERT-based architecture [149] to model the periodic behaviors of the trajectories on the road network. More specifically, the model learns a time embedding vector $\hat{\mathbf{t}} \in \mathbb{R}^{(T+1)\times 1}$ for each timepoint $t \in [0, T]$:

$$\hat{\mathbf{t}}[i] = \begin{cases} \xi_i t + \zeta_i & \text{if } i = 0 \\ \cos(\xi_i t + \zeta_i) & \text{if } 1 \leq i \leq T \end{cases} \tag{4.1}$$

where $\{\xi_i\}_{i=0}^{T}$ and $\{\zeta_i\}_{i=0}^{T}$ are learnable parameters and $\cos(\cdot)$ is the periodic activation function. The output embeddings $\hat{\mathbf{t}}$ is passed onto an LSTM [58], where each step t of the LSTM layer represents a point in time from $[0, T]$. Its recurrent procedure at $\mathbf{h}_t$ takes the result of the last step $\mathbf{h}_{t-1}$, the current time embedding $\hat{\mathbf{t}}_t$, the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, output gate $\mathbf{o}_t$, and the memory cell $\mathbf{m}_t$, to produce the output of the current step as follows:

$$\mathbf{h}_t = \text{LSTM}\left(\hat{\mathbf{t}}_t, \mathbf{h}_{t-1}, \mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t, \mathbf{m}_t\right) \tag{4.2}$$

Now, the output of the final step $\mathbf{h}_T$ serves as the temporal representation of the trajectories' road segments. Moreover, as with SEGSRL, attention mechanisms [144] were utilized on this temporal representation (i.e., $\mathbf{h}_T \rightarrow \tilde{\mathbf{h}}_T = \mathbf{z}^{(t)}$) to capture correlations between trajectory points (roads) and further enhance its representation.

Spatiotemporal Fusion. The (road) **S**egment-based **S**patio-**T**emporal **F**usion (SegStf) module allows the interaction of the spatial and temporal features for fusing, resulting into a spatiotemporal representation. To be more precise, in this module, some weighted transformation is applied on the spatial and temporal vector representations ($\mathbf{z}^{(s)}$ and $\mathbf{z}^{(t)}$ respectively) of our roads separately: $\mathbf{Z}^{(s)} = W_F\mathbf{z}^{(s)}$ and $\mathbf{Z}^{(t)} = W_F\mathbf{z}^{(t)}$. Then, if we let:

$$w(i,j) = \frac{\exp\left(W_Q\mathbf{Z}^{(i)} \cdot W_K\mathbf{Z}^{(j)}\right)}{\sum_{*\in\{s,t\}} \exp\left(W_Q\mathbf{Z}^{(i)} \cdot W_K\mathbf{Z}^{(*)}\right)} \tag{4.3}$$

where i and j are either s and/or t (that denote the spatial and temporal dimensions), W_F is a feature matrix, W_Q and W_K are learnable matrices, and then apply a specialized spatiotemporal interaction on these representations to obtain enhanced spatial and temporal vector representations ($\hat{\mathbf{z}}^{(s)}$ and $\hat{\mathbf{z}}^{(t)}$ respectively):

$$\begin{aligned} \hat{\mathbf{z}}^{(s)} &= \textsc{Norm}\left(\textsc{Ffn}\left(w(s,t)\mathbf{Z}^{(t)} \otimes w(s,s)\mathbf{Z}^{(s)}\right) \otimes \mathbf{z}^{(s)}\right) \\ \hat{\mathbf{z}}^{(t)} &= \textsc{Norm}\left(\textsc{Ffn}\left(w(t,t)\mathbf{Z}^{(t)} \otimes w(t,s)\mathbf{Z}^{(s)}\right) \otimes \mathbf{z}^{(t)}\right) \end{aligned} \tag{4.4}$$

Finally, $\hat{\mathbf{z}}^{(s)}$ and $\hat{\mathbf{z}}^{(t)}$ are concatenated together and fed to an Lstm for the final spatiotemporal representation for each road segment at some specific point in time:

$$\mathbf{z} = \textsc{Lstm}\left(\hat{\mathbf{z}}^{(s)} \otimes \hat{\mathbf{z}}^{(t)}\right) \tag{4.5}$$

Given the spatiotemporal representations of road segments (along with a data structure to store them for easy lookups), we now turn into how the trajectory similarity problem can be addressed as a *set similarity problem.* More specifically, we describe how trajectories can be viewed as sets, with traversed road segments as elements.

4.1.3 From Trajectories to Sets

Many works focused on representing trajectories as a sequence of raw points [37], disjoint cells [85], and visited POIs [53]. In this work, we demonstrate how the ordering of these elements can be disregarded when searching for similar trajectories based on their representations on the latent space. In other words, trajectories in theory, can be represented instead as a set of such elements. And because trajectories can be well-represented by road segments as per Definition 3.1.5, we can instead view this road-based representation of a trajectory $\mathfrak{R}(\tau)$ as a set (instead of a sequence) of roads.

Intuition. Two similar trajectories will map to the same set representation as they both compose of the same sets of time-enabled road segments, without regard to ordering (of which road comes first in the trajectory's traversal). Whereas two dissimilar trajectories can be seen to have dissimilar set representations regardless of elements' permutation. What is then required is an architecture that respects these properties.

Neural Networks for Sets. Typical machine learning algorithms are designed mostly for datasets which have a fixed ordering. However, such models do not necessarily generalize when elements of the input set-like data are scrambled, i.e., permuting the input elements would result into an output that is completely different. This relates to what is known as the *responsibility problem* [189], where small changes in the input set space results in a large change in the neural network's output space [190], which can result in major discontinuity issues [189]. As such, there is a need for a *permutation-invariant* [182] architecture that can handle the unordered set property. Moreover, the said architecture must also be *permutation-equivariant* [182], which describes that the permutation applied to the input set space retains the same permutation on the neural network output. So then, for some function $f : \mathbf{X} \mapsto \mathbf{Y}$, $\boldsymbol{\chi} \in \mathbf{X}$ and some permutation $\tilde{\mathbf{p}} \in \mathbf{P}_n$, with $\mathbf{P}_n$ being the set of all permutations of indices

$\{1, ..., n\}$, these desired properties are summarized by:

$$\begin{cases} \text{①}\ f(\boldsymbol{\chi}) = f(\tilde{\mathbf{p}}\boldsymbol{\chi}) & \textit{permutation-invariance} \\ \text{②}\ f(\tilde{\mathbf{p}}\boldsymbol{\chi}) = \tilde{\mathbf{p}}f(\boldsymbol{\chi}) & \textit{permutation-equivariance} \end{cases} \tag{4.6}$$

Now, while there has been significant research efforts towards this direction of designing these desirable models for sets [182, 189, 190, 133], it is important to keep in mind that we aspire learning-based models that can represent sets that also preserve the similarity relation. In other words, two similar (dissimilar) sets A and B ought to have similar (dissimilar) representations in the latent space. In our case, we want our similar (dissimilar) trajectories to have similar (dissimilar) segment-based representations. Lee et al.'s model [79] known as SET2BOX can accurately preserve the similarity relations of sets by representing them as hyper-rectangles called *boxes*. Thus, we adapt a similar method, customized specifically for road elements – with the goal of preserving the similarity (dissimilarity) relations of our road segment-based set representations of trajectories.

4.1.4 The Box Architecture

The resulting spatiotemporal vectors obtained from SEG2VEC in Chapter 4.1.2 are then passed as set elements to SEG2BOX, a SET2BOX architecture that takes in road segments as elements and trajectories as sets. In general, this architecture based from Lee et al. [79] characterized for its *accurate*, *versatile*, and *generalizable* representations, specifically takes as input road-based representations of trajectories (the sets) and their road segments (the entities or elements of the sets). This architecture is able to represent the sets as *boxes*, while able to learn each of the sets, their structure and their relationships (i.e., similarity) with other sets. In particular, these box (lattices) are axis-aligned hyper-rectangles on a

d-dimensional space that can be represented by $\mathbf{B} = (\mathbf{c}, \boldsymbol{\varepsilon})$ [146]:

$$\mathbf{B} = \{\mathbf{p} \,|\, \mathbf{p} \in \mathbb{R}^d \textbf{ and } \mathbf{c} - \boldsymbol{\varepsilon} \preceq \mathbf{p} \preceq \mathbf{c} + \boldsymbol{\varepsilon}\} \tag{4.7}$$

for centre $\mathbf{c} \in \mathbb{R}^d$ and offset $\boldsymbol{\varepsilon} \in \mathbb{R}^d_+$. Now the intersection $\mathbf{B}_1 \cap \mathbf{B}_2$ of two boxes $\mathbf{B}_1 = (\mathbf{c}_1, \boldsymbol{\varepsilon}_1)$ and $\mathbf{B}_2 = (\mathbf{c}_2, \boldsymbol{\varepsilon}_2)$ can be represented:

$$\{\mathbf{p} \,|\, \mathbf{p} \in \mathbb{R}^d\,;\, \max\left(\mathbf{p}_1^{min}, \mathbf{p}_2^{min}\right) \preceq \mathbf{p} \preceq \min\left(\mathbf{p}_1^{max}, \mathbf{p}_2^{max}\right)\} \tag{4.8}$$

where $\mathbf{p}^{min} = \mathbf{c} - \boldsymbol{\varepsilon}$ and $\mathbf{p}^{max} = \mathbf{c} + \boldsymbol{\varepsilon}$ are the minimum and maximum vectors of $\mathbf{B}$ at each dimension respectively.

The Target. SEG2BOX aims to approximate the volume $\mathbb{V}$ of the box embedding $\mathbf{B}_X$ of some trajectory τ_X's segment-based representation set $\mathfrak{R}(\tau_X)$ to the relative size of $\mathfrak{R}(\tau_X)$, i.e., $\mathbb{V}(\mathbf{B}_X) \propto |\mathfrak{R}(\tau_X)|$; moreover, preserving similarity relations with other segment-based representation sets meant that SEG2BOX also targets to approximate the volume $\mathbb{V}$ of the box intersection $\mathbf{B}_X \cap \mathbf{B}_Y$ to the size of the intersection of two sets $\mathfrak{R}(\tau_X)$ and $\mathfrak{R}(\tau_Y)$, i.e., $\mathbb{V}(\mathbf{B}_X \cap \mathbf{B}_Y) \propto |\mathfrak{R}(\tau_X) \cap \mathfrak{R}(\tau_Y)|$. Note that the volume of a box $\mathbf{B}$ can be calculated by:

$$\mathbb{V}(\mathbf{B}) = \prod_{i=1}^{d} \mathrm{RELU}\left(\mathbf{p}^{max}[i] - \mathbf{p}^{min}[i]\right) \tag{4.9}$$

while the volume of the union of two boxes $\mathbf{B}_X$ and $\mathbf{B}_Y$ is $\mathbb{V}(\mathbf{B}_X) + \mathbb{V}(\mathbf{B}_Y) - \mathbb{V}(\mathbf{B}_X \cap \mathbf{B}_Y)$.

The Objective Function. To preserve elemental relations between (triplets of) trajectories (through their road-based representation set), then consider the cardinalities as seen in Table 4.1 for triplet set $\{\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z)\}$ for trajectories $\{\tau_X, \tau_Y, \tau_Z\}$ that contain singleton, pairwise, and triplet-wise set information [79]. Then we sample trajectory triplets ($\{\tau_X, \tau_Y, \tau_Z\}$) from our trajectory set $\mathcal{T}$: positive triplets $\mathcal{T}^+$ (trajectories that share some

Cardinalities	**Measures**
$C_1(\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z))$	$\lvert\mathfrak{R}(\tau_X)\rvert$
$C_2(\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z))$	$\lvert\mathfrak{R}(\tau_Y)\rvert$
$C_3(\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z))$	$\lvert\mathfrak{R}(\tau_Z)\rvert$
$C_4(\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z))$	$\lvert\mathfrak{R}(\tau_X) \cap \mathfrak{R}(\tau_Y)\rvert$
$C_5(\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z))$	$\lvert\mathfrak{R}(\tau_X) \cap \mathfrak{R}(\tau_Z)\rvert$
$C_6(\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z))$	$\lvert\mathfrak{R}(\tau_Y) \cap \mathfrak{R}(\tau_Z)\rvert$
$C_7(\mathfrak{R}(\tau_X), \mathfrak{R}(\tau_Y), \mathfrak{R}(\tau_Z))$	$\lvert\mathfrak{R}(\tau_X) \cap \mathfrak{R}(\tau_Y) \cap \mathfrak{R}(\tau_Z)\rvert$

Table 4.1: Cardinality measures for the objective function

common road segments) and negative triplets $\mathcal{T}^-$ (chosen uniformly at random), where each trajectory τ_i in each triplet is expressed as a set of road segments through its segment-based representation set $\mathfrak{R}(\tau_i)$. The goal is then to minimize the following objective function:

$$\sum_{\{\tau_X, \tau_Y, \tau_Z\} \in \mathcal{T}^+ \cup \mathcal{T}^-} \sum_{i=1}^{7} \left(\frac{C_i(\tau_X, \tau_Y, \tau_Z)}{\sum_{j=1}^{7} C_j(\tau_X, \tau_Y, \tau_Z)} - \frac{\hat{\mathbb{V}}_i(\mathbf{B}_X, \mathbf{B}_Y, \mathbf{B}_Z)}{\sum_{k=1}^{7} \hat{\mathbb{V}}_k(\mathbf{B}_X, \mathbf{B}_Y, \mathbf{B}_Z)} \right)^2 \tag{4.10}$$

where $\mathbf{B}_X$, $\mathbf{B}_Y$ and $\mathbf{B}_Z$ are the box embeddings for the segment-based representation sets of trajectory τ_X, τ_Y and τ_Z, and $\hat{\mathbb{V}}_i$ is the approximated volume of the boxes corresponding to C_i for $i = \{1, ..., 7\}$.

Generalizing to Unseen Trajectories. We first derive the box embedding of the road-based representation set of some trajectory via learnable embedding matrices $\mathbf{Q}^{(\mathbf{c})} \in \mathbb{R}^{|\mathbb{S}| \times d}$ and $\mathbf{Q}^{(\boldsymbol{\varepsilon})} \in \mathbb{R}_+^{|\mathbb{S}| \times d}$ that would represent the centre and offset for some entity/road $r \in \mathbf{R}$; from here, simply aggregate the centre and offset embeddings of the road segments in $\mathfrak{R}(\tau)$ to obtain $\mathbf{B}_\tau = (\mathbf{c}_\tau, \boldsymbol{\varepsilon}_\tau)$. And then use attention mechanisms [144] to emphasize on important entities (roads) for obtaining box centre and offset embeddings through a *termed set-context pooling* layer (SCP), a pooling defined by [79]:

$$\text{SCP}\left(\mathfrak{R}(\tau), \mathbf{Q}^{(*)}\right) = \sum_{r_i \in \mathfrak{R}(\tau)} \left[\frac{\exp\left(\mathbf{u}_{\mathfrak{R}(\tau)}^\top \mathbf{Q}_i^{(*)}\right)}{\sum_{r_j \in \mathfrak{R}(\tau)} \exp\left(\mathbf{u}_{\mathfrak{R}(\tau)}^\top \mathbf{Q}_j^{(*)}\right)} \right] \mathbf{Q}_i^{(*)} \tag{4.11}$$

for context vector $\mathbf{u}_{\mathfrak{R}(\tau)}$ containing information on $\mathfrak{R}(\tau)$:

$$\mathbf{u}_{\mathfrak{R}(\tau)} = \sum_{r_i \in \mathfrak{R}(\tau)} \left(\frac{\exp\left(\mathbf{U}^\top \mathbf{Q}_i^{(*)}\right)}{\sum_{r_j \in \mathfrak{R}(\tau)} \exp\left(\mathbf{U}^\top \mathbf{Q}_j^{(*)}\right)} \right) \mathbf{Q}_i^{(*)} \tag{4.12}$$

and global vector $\mathbf{U}$ that is shared by all trajectories. Note that $\mathbf{Q}^{(*)}$ here would represent either $\mathbf{Q}^{(\mathbf{c})}$ or $\mathbf{Q}^{(\boldsymbol{\varepsilon})}$.

Smoothing Boxes. The boxes we saw so far are *hard boxes*, and gradient-based optimization is not possible when these are disjoint; to fix this, *smoothing* using SoftPlus to approximate ReLU of the volume function may be needed [84]:

$$\mathbb{V}(\mathbf{B}) = \prod_{i=1}^{d} \textsc{SoftPlus}\left(\mathbf{p}^{max}[i] - \mathbf{p}^{min}[i]\right) \tag{4.13}$$

where $\textsc{SoftPlus}(x) = \frac{1}{\beta}\log(1+e^{\beta x})$ for box smoothing parameter $\beta > 0$, with $\textsc{SoftPlus}(x)$ approaching $\textsc{ReLU}(x)$ function as β increases.

Scalability. Seg2Box inherits this desired characteristic from Set2Box [79] as their model is generic to any kind of sets/elements. Their constant $\mathcal{O}(d)$ running time for computing pairwise similarities between sets (through their theoretical proof and conducted experiments) allows Seg2Box to also perform such computation between trajectories in a short amount of time, enabling for the method to scale.

Overall Desired Characteristics. Ideally, our St2Box method, as we will see later in the experimental section, is characterized by its (i) *accuracy* (i.e., similar trajectories that have similar road-based representation sets have embeddings that are close in the latent space), (ii) *versatility* (i.e., the representations can be used to approximate a wide variety of similarity measures and applicable to various trajectory set elements), (iii) *generalizability* (i.e., embeddings of unseen trajectories, specifically their road-based representation sets, are

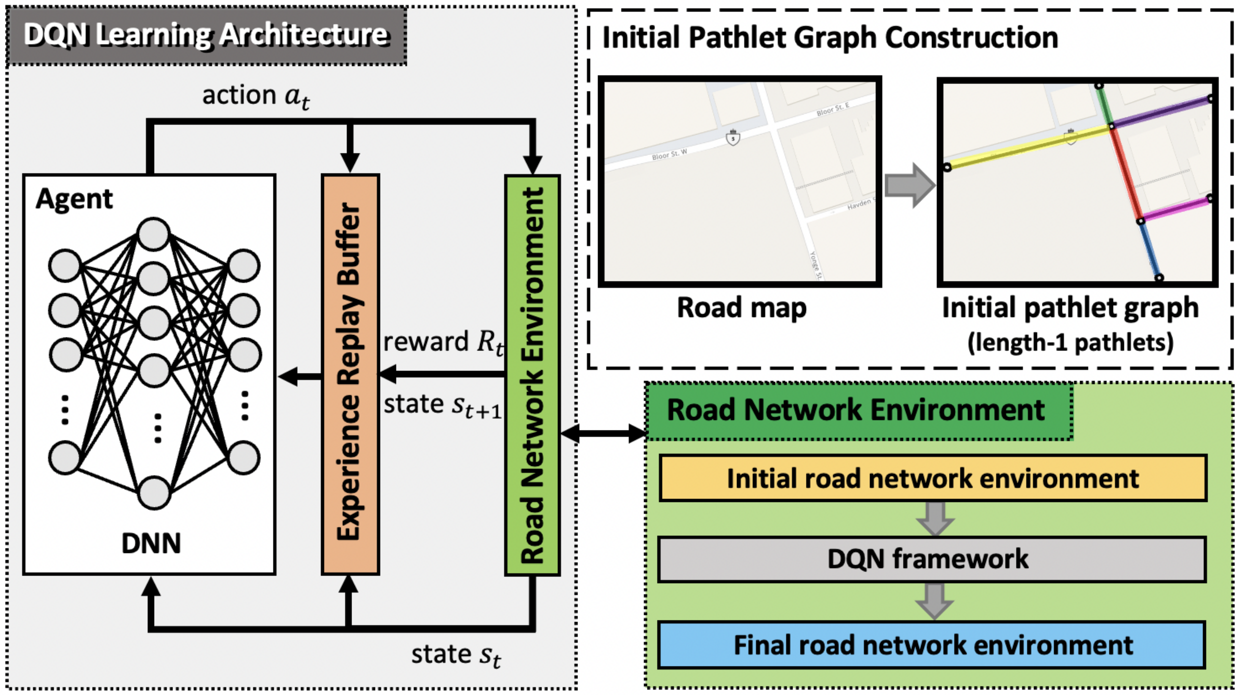


Figure 4.2: The overall architecture of the proposed PATHLETRL model

obtainable), (iv) *robustness* (i.e., in terms of spatiotemporal similarity that is based on any weight parameter θ), and (v) *speed* (i.e., able to estimate trajectory similarity quickly, and as result can therefore scale to larger datasets, as inherited from ST2BOX [79]).

4.2 PATHLETRL: A Solution for PD Construction

In the trajectory pathlet dictionary construction, the model consists of two main components: (1) the method responsible for extracting the candidate pathlet sets through a merging-based process (Chapter 4.2.1), and (2) a deep reinforcement learning-based architecture for approximating the utility function of the merging process of (1) (Chapter 4.2.2). Refer to Figure 4.2 for an illustration of the architecture.

4.2.1 Extracting Candidate Pathlets

In this chapter, we describe the algorithmic details for merging edge-disjoint pathlets. The high-level idea of the algorithm is based on the theory of maximal utility [94, 3], i.e., iteratively merging (neighboring) pathlets until this brings forth little to no improvement on the *utility* (details of the utility function are given later). The algorithm takes in as input a road network $\mathcal{G}$, a trajectory set $\mathcal{T}$ operating within $\mathcal{G}$, the maximum threshold for the trajectory loss M, the trajectory representability threshold $\hat{\mu}$, and a positive integer χ denoting the desired χ-order pathlet graph. As output, it returns a pathlet dictionary (Pd) that holds pathlet information as described in Definition 3.2.10. The extracted Pd aims to satisfy the four objectives **(O1)-(O4)** discussed in Chapter 3.2.2. See Algorithm 4.2 for the pseudocode.

Initialization. The algorithm first initializes the pathlet graph $\mathcal{G}_p$, extracted from $\mathcal{G}$ – and more importantly are the initial (length 1) pathlets $\mathcal{E}_p$ (`lines 1-2`). Then, we make a copy of all the input trajectories in $\mathcal{T}^*$, which keeps track of the current trajectories that we currently have; and further initialize an empty set for the candidate pathlet set we intend to build (`line 3`). We also set up other important variables such as the ϕ and the $\bar{\mu}$ as defined in the preliminaries, as well as empty dictionaries for the the trajectory loss and utility that will be useful for later (`lines 4-5`). Moreover, a pathlet from $\mathcal{E}_p$ is chosen uniformly at random (`line 6`).

An Iterative Algorithm. The basic idea behind the `while` loop is that we iteratively merge pathlets until merging brings little to no improvement on $\mathcal{G}_p$'s utility. Once a pathlet cannot be further merged with any of its neighbors, due to no further gain in utility, we add it to our candidate set and randomly select the next pathlet.

We set the utility of $\mathcal{G}_p$ associated with pathlet ρ to be 0, i.e., not merging ρ with any of its neighbors brings zero utility (`line 8`). We then consider each of the unprocessed neighbors $\hat{\rho}$ of pathlet ρ; compute the utility of merging ρ with each of its neighbors $\hat{\rho}$, and then also

Algorithm 4.2: Candidate Pathlet Set Extraction Algorithm

```
Input  : The road network $\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$, the trajectory set $\mathcal{T}$, integer $\chi$, the maximum
         trajectory loss $M$ and the average trajectory representability threshold $\hat{\mu}$.
Output : The $\chi$-order candidate pathlet set $\mathbb{S}$ of merged pathlets with a trajectory loss not
         exceeding $M$
   /* Initialization */
1  $\mathcal{G}_p\langle\mathcal{V}_p,\mathcal{E}_p\rangle \leftarrow$ ExtractPathletGraph($\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$)
2  $\ell \leftarrow 1$                                  // Size of the initial length 1 pathlets
3  $\mathcal{T}^* \leftarrow \mathcal{T}$; $\mathbb{S} \leftarrow \varnothing$
4  $\phi \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}|\Phi(\tau)|$ ; $\bar{\mu} \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}\mu(\tau)$
   /* Setup traj loss and utility dictionaries */
5  $T_D \leftarrow$ Dict(); $U_D \leftarrow$ Dict()
6  $\rho \leftarrow$ Rand($\mathcal{E}_p$)                        // Uniformly pick $\rho \in \mathcal{E}_p$ at random
   /* Repeat until all pathlets are processed */
   /* Or when traj loss exceeds the maximum */
7  while $\mathcal{E}_p \neq \mathbb{S}$ or sum($T_D$.Values()) $< M$ or $\bar{\mu} \geq \hat{\mu}$ do
       /* Initially set $\mathcal{G}_p$'s utility associated to the curr pathlet $\rho$ to be 0 */
8      $U_D[\rho] = 0$
       /* For each of $\rho$'s unprocessed neighbors */
9      foreach $\hat{\rho}$ in $\Psi(\rho) \setminus \mathbb{S}$ do
10         $U_D[\hat{\rho}] \leftarrow$ ComputeUtil(Merge($\rho,\hat{\rho}$))
11         $T_D[\hat{\rho}] \leftarrow$ GetAllTrajLost (Merge($\rho,\hat{\rho}$), $\mathcal{T}^*$)
       /* Find the one with the highest utility */
12     $\rho^* \leftarrow \operatorname{argmax}_{\rho\in key} U_D[key]$
13     if $\rho^* = \rho$ or $\ell > \chi$ then                        // Merge not necessary
14         $\mathbb{S} \leftarrow \mathbb{S} \cup \{\rho\}$
15         $\rho \leftarrow$ Rand($\mathcal{E}_p \setminus \mathbb{S}$)                       // Pick new pathlet
16         if $\rho = \varnothing$ then                       // All pathlets processed
17             break
18         $\ell \leftarrow 1$                                      // Reset pathlet length
19     else                                                  // Merge recommended
20         $\rho_{merged} \leftarrow$ Merge($\rho,\rho^*$)
21         $\mathcal{E}_p \leftarrow (\mathcal{E}_p \setminus \{\rho,\rho^*\}) \cup \{\rho_{merged}\}$
22         $\mathcal{T}^* \leftarrow \mathcal{T}^* \setminus T_D[\rho^*]$
23         $\phi \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}|\Phi(\tau)|$ ; $\bar{\mu} \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}\mu(\tau)$
24         $\rho \leftarrow \rho_{merged}$; $\ell \leftarrow \ell + 1$
25 return $\mathbb{S}$
```

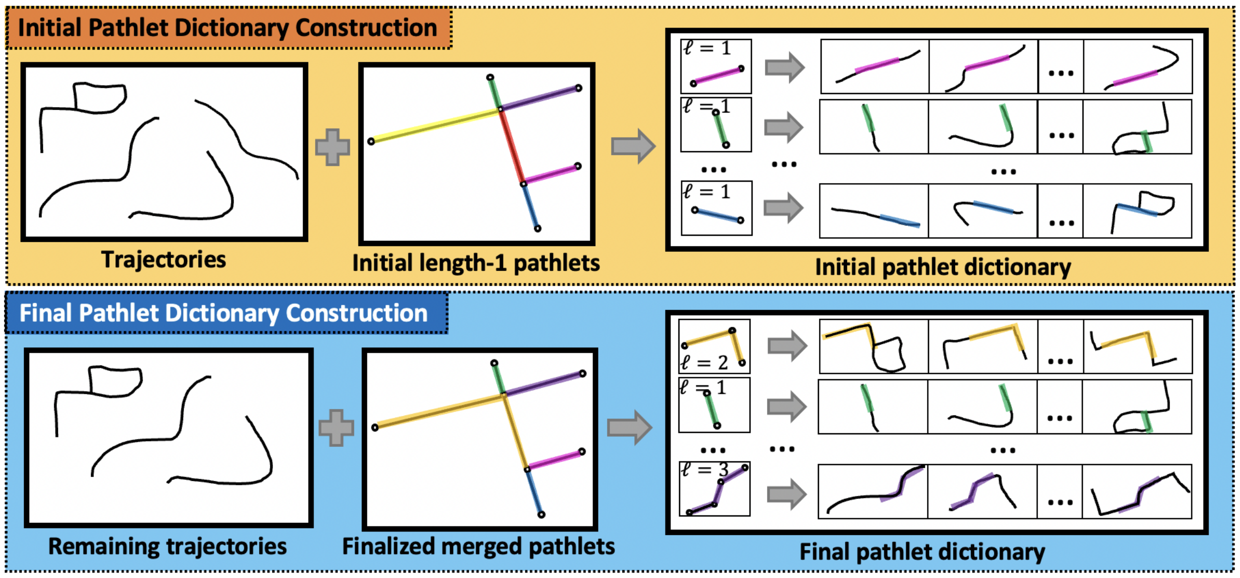

Initial Pathlet Dictionary Construction
Trajectories
Initial length-1 pathlets
Initial pathlet dictionary
$\ell = 1$
Final Pathlet Dictionary Construction
Remaining trajectories
Finalized merged pathlets
Final pathlet dictionary
$\ell = 2$
$\ell = 3$

Figure 4.3: Pathlet dictionary construction (initial – top; final – bottom) based on the initial and final road network environments of the deep reinforcement learning architecture (refer to the yellow and blue boxes in Figure 4.2).

the set of all trajectories that could be lost for when the pair of candidate pathlets do end up merging (`lines 9-11`). More specifically, these lines maintain a record of how much the merge of this candidate pair will impact the representabilities and losses of the trajectories. The algorithm then finds a pathlet ρ^* that is a candidate for merging with current pathlet ρ; this candidate achieves the highest utility when merged with the current pathlet (`line 12`). There are then two cases for where merging is not recommended (`line 13`): (1) when $\rho^* = \rho$ (i.e., the algorithm deems that merging with another neighboring pathlet contributes little to no improvement on the utility of $\mathcal{G}_p$), and (2) when $\ell > \chi$ (i.e., merging the two pathlets ρ and ρ^* would violate the χ-order constraint). In either of these cases, we add current pathlet ρ to our candidate pathlet set, and then randomly select another unprocessed pathlet in the pathlet graph; if all pathlets have already been processed, then we immediately end the loop and return the candidate pathlet set $\mathbb{S}$ (`lines 14-18`). Otherwise, we immediately take out the two pathlets that are candidate for merging and then add the newly merged pathlet to our current pathlet set $\mathcal{E}_p$ in our pathlet graph (`lines 20-21`). Moreover, we remove

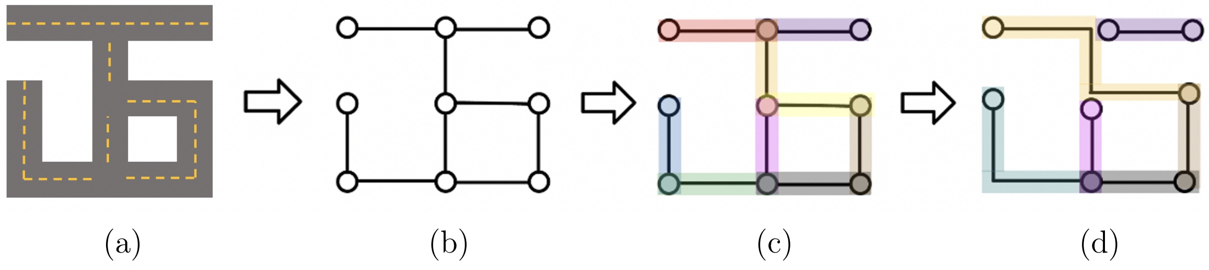


Figure 4.4: An illustrative example of Example 4.2.1: (a) A toy example of a simple road network; (b) A grid representation of (a); (c) The initial pathlet graph representation (of length-1 pathlets) for the road network in (a); (d) The final (merged) pathlet graph representation after the completion of the pathlet-merging algorithm in Algorithm 4.2.

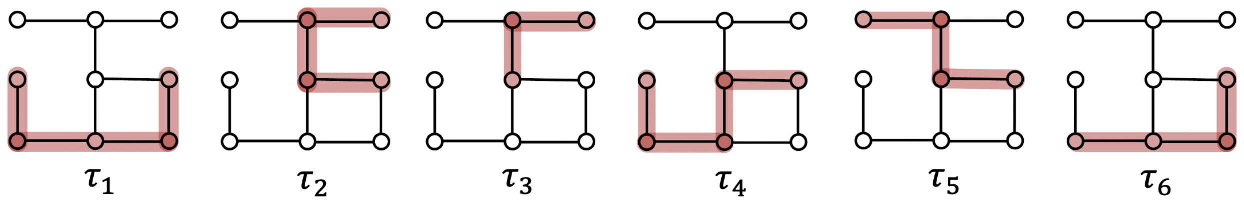


Figure 4.5: An illustrative example of the paths (road segments) traversed by six trajectories $\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$ (highlighted by maroon) from the road network of Example 4.2.1 as seen in Figure 4.4; see Table 4.3 that lists the sequential pathlet-based representations of each trajectory.

the collected lost trajectories from `line 11` from our current trajectory set $\mathcal{T}^*$ (`line 22`). The method also updates ϕ, $\bar{\mu}$, the current pathlet processed and its current length (`lines 23-24`). The iterative procedure ends when one of the following occurs: (1) all pathlets have been processed, (2) the trajectory loss has exceeded the threshold M, or (3) the average representability $\bar{\mu}$ falls below the threshold $\hat{\mu}$.

The Utility Function. To complete the description of the algorithm, we discuss the formulation of the utility function that we approximated using a learning-based method. In particular, a reinforcement learning method is utilized to learn the (sequence of) actions (i.e., merge or don't merge pathlets) that would yield the highest possible utility. Specifically, we frame the utility function as a *reward function* that we aim to optimize (i.e., maximize). We discuss the details of this design in Chapter 4.2.2.

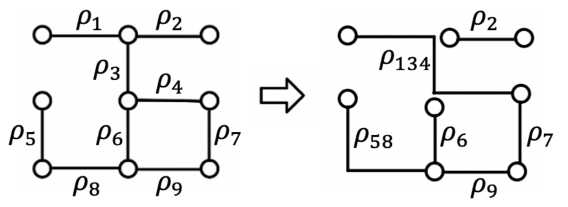


Figure 4.6: Pathlet labels for the initial and final pathlet graph representations

With the pathlet dictionary constructed following this process of pathlet merging based on the utility theory, we can attain something like the one depicted in Figure 4.3 (bottom figure). Now we provide a simple toy example of the algorithm.

Example 4.2.1 As edge-disjoint pathlets are non-overlapping, then a decision would have to be made when a pathlet is about to merge with one of its neighbor pathlets. As such, there is some advantages or gains by merging with a certain pathlet, but such action also comes with some cost. This is where trajectory representabilities and losses come in. More specifically, a trajectory's representability will drop when a portion of its trajectory cannot be represented due to a pathlet merging with one of its neighbors. To give a more concrete example, consider the snippet of a road network in Figure 4.4(a). Figure 4.4(b) illustrates its grid representation, and Figure 4.4(c) the (initial) pathlet graph representation, where we highlighted using various colors the nine pathlets in question. See the left image of Figure 4.6 for the labels of these pathlets and their color code in Table 4.2. Initially, our pathlet dictionary is composed of the following based on the six trajectories as highlighted by the maroon color on the grid graph in Figure 4.5 (their pathlet-based representations in Table 4.3):

$$\rho_1 : \{\tau_5\} \quad \rho_2 : \{\tau_2, \tau_3\} \quad \rho_3 : \{\tau_2, \tau_3, \tau_5\} \quad \rho_4 : \{\tau_2, \tau_4, \tau_5\} \quad \rho_5 : \{\tau_1, \tau_4\}$$
$$\rho_6 : \{\tau_4\} \quad \rho_7 : \{\tau_1, \tau_6\} \quad \rho_8 : \{\tau_1, \tau_4, \tau_6\} \quad \rho_9 : \{\tau_1, \tau_6\}$$

After the entire merging process, the algorithm returns the following final dictionary – with its illustration in Figure 4.4(d), with their labels in Figure 4.6, their color codes in Table 4.2 and the trajectories' updated representabilities and pathlet-based representations in Table 4.3:

Pathlet	**Color**
ρ_1	red
ρ_2	purple
ρ_3	orange
ρ_4	yellow
ρ_5	blue
ρ_6	pink
ρ_7	brown
ρ_8	green
ρ_9	grey
ρ_{134}	tangerine
ρ_{58}	aquamarine

Table 4.2: Color coding of the pathlets for the toy example in Example 4.2.1

Traj	**Pathlet-based representation set Φ (Representability μ)**	
	(Before merge)	**(After merge)**
τ_1	$\{\rho_5, \rho_8, \rho_9, \rho_7\}$ (100%)	$\{\rho_{58}, \rho_9, \rho_7\}$ (100%)
τ_2	$\{\rho_2, \rho_3, \rho_4\}$ (100%)	$\{\rho_2\}$ (33%)
τ_3	$\{\rho_3, \rho_2\}$ (100%)	$\{\rho_2\}$ (50%)
τ_4	$\{\rho_4, \rho_6, \rho_8, \rho_5\}$ (100%)	$\{\rho_6, \rho_{58}\}$ (75%)
τ_5	$\{\rho_4, \rho_3, \rho_1\}$ (100%)	$\{\rho_{134}\}$ (100%)
τ_6	$\{\rho_7, \rho_9, \rho_8\}$ (100%)	$\{\rho_7, \rho_9\}$ (67%)

Table 4.3: Pathlet-based representation set (and trajectory representabilities) of the provided toy example in Example 4.2.1

$$\rho_{134} : \{\tau_5\} \quad \rho_2 : \{\tau_2, \tau_3\} \quad \rho_{58} : \{\tau_1, \tau_4\} \quad \rho_6 : \{\tau_4\} \quad \rho_7 : \{\tau_1, \tau_6\} \quad \rho_9 : \{\tau_1, \tau_6\}$$

Note that pathlets ρ_1, ρ_3, and ρ_4 from initial pathlet graph merged to become pathlet ρ_{134} in the final pathlet graph (it is irrelevant in this example whether pathlet ρ_3 merged with ρ_1 or ρ_4 first). The same can be said for pathlets ρ_5 and ρ_8 to form pathlet ρ_{58}.

Initially, all trajectories have $\mu = 100\%$ representatbility values because all six trajectories can be represented by pathlets in the initial pathlet dictionary. However, after the entire merging process, we are left with some of the original six trajectories to have a lower μ value than the original representability value. Notice for example trajectory τ_1; its μ value did not drop because all the pathlets in its original pathlet-based representation Φ either have never merged, or have merged with a neighboring pathlet that also belongs to $\Phi(\tau_1)$. In other words, the entirety of trajectory τ_1 can still be represented by the pathlets in the final pathlet dictionary; this results into its representability being maintained at 100%. A similar story can be told for trajectory τ_5. For the rest of the trajectories, the representabilities are lower. Looking at trajectory τ_2 for example whose $\Phi(\tau_2) = \{\rho_2, \rho_3, \rho_4\}$. However, after the algorithm, we only have $\Phi(\tau_2) = \{\rho_2\}$ left; the reason being is that pathlets ρ_1, ρ_3, and ρ_4

have all merged together to form ρ_{134}. But note that since ρ_1 does not represent τ_2, then the merged ρ_{134} is not part of the $\Phi(\tau_2)$ after completing the iterative algorithm. As a result, it is expected for its trajectory representability to drop. The same story can be said for trajectories τ_3, τ_4 and τ_6.

Remark 4.2.1 One can imagine that the representabilities of the trajectories can potentially drop at each step of the iterative algorithm, until it drops to zero. In the event that a trajectory's representabilty reaches zero, it is removed from the trajectory set and counted as a *trajectory loss*. Now this is a soft version of what is considered to be a trajectory loss. There is a harder, stricter variation, where the notion of trajectory representability is eliminated from the picture, (i.e., a trajectory losing only a small portion of its representability due to a pathlet merge action meant that the entire trajectory is considered lost), entails throwing excessive trajectories and can be fatal to the performance of the model and the algorithm. Intuitively, one might not desire discarding an entire trajectory when (say only 1% of the trajectory) cannot be represented due to the pathlet merge. The experimental evaluations later will demonstrate the essence of this soft version that takes trajectory representability into account.

Theorem 4.2.1 (Trajectory Representability Theorem) At any step i of the iterative Algorithm 4.2, then the trajectory representability μ of some trajectory $\tau \in \mathcal{T}$ by the end of that iteration i is equal to:

$$\mu_i(\tau) = \frac{\sum_{\rho' \in \Phi_i(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} \tag{4.14}$$

where Φ_0 and Φ_i are the pathlet-based representation of trajectory τ in the initial (iteration 0) and iteration i of the iterative algorithm respectively.

The above theorem provides a formula for how to compute a trajectory's representability value μ at some iteration i of the algorithm. We refer the reader to Appendix B.1 where we provide a complete proof for this theorem.

4.2.2 Reinforcement Learning Framework

In reinforcement learning (RL), desirable actions lead to higher rewards while unfavorable actions result in punishment (lower-valued rewards) – a trend analogous to what we desire. RL methods have seen success in solving decision-based problems in an attempt to maximize rewards [97, 132]. As this aligns with our goal to maximize utility, we motivated the use of RL to merge pathlets. We briefly go over its components here (see the (left) gray panel of Figure 4.2).

The Environment. RL models are designed for an agent to learn the most optimal actions, commonly in games [97, 131, 132, 129, 136, 11]. In this context, we consider the entire pathlet graph $\mathcal{G}_p$ to be the environment; it is where our deep RL algorithm will be operating on.

The Agent. RL is often designed for training robotic agents, or some AI [70]. In our case, our agent is trained to learn which pathlets in the pathlet graph are to be merged/kept unmerged. In particular, we train the agent to learn the most optimal sequence of actions that would yield the highest possible utility (reward).

The States. The reinforcement learning paradigm is based on the Markov decision process (MDP) [107], that requires specification of states. In this case, the state $s_t \in \mathcal{S}$ is depicted by the current state of the pathlet graph environment. In particular, the pathlet graph's state can be represented as a 4-tuple (S_1, S_2, S_3, S_4), where S_1 denotes the number of pathlets in the current pathlet graph, S_2 denotes the average number of pathlets to represent the trajectories, S_3 is the trajectory loss and S_4 is the average trajectory representability.

The Actions. At each time t, the agent has a choice of two discrete actions on the currently processed pathlet ρ, as expressed by the action space $\mathcal{A} = \{\textsc{keep}, \textsc{merge}\}$. In other words, KEEP action suggests that the current pathlet ρ should be kept and not be merged with any one of its neighbors. As a result, Algorithm 4.2 puts the current pathlet ρ in the processed set and then selects a new pathlet to process, performing one of the two actions in the action space on that new pathlet. The MERGE action should however merge the current pathlet ρ with one of its $|\Psi(\rho)|$ neighbors. For that, the agent would need to decide on which neighbor in the set $\Psi(\rho)$ to merge with. Thus, the action space can succinctly be written as:

$$\mathcal{A} = \bigcup_{\forall \hat{\rho} \in \Psi(\rho)} \textsc{merge}(\rho, \hat{\rho}) \cup \{\textsc{keep}(\rho)\} \tag{4.15}$$

The Reward Function. We formulate our reward function R based on the optimization equation defined in Equation (3.14):

$$\max_{a_t} \mathbb{E} \left[\left(-\alpha_1 |\mathbb{S}| - \alpha_2 \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} |\Phi(\tau)| - \alpha_3 L_{traj} + \alpha_4 \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \mu(\tau) \right) \right] \tag{4.16}$$

Whenever the RL agent performs an action a_t in the pathlet graph environment, the environment provides back feedback to it in the form of instantaneous rewards $\{r_t\}_{t=0}^{T}$:

$$r_t = -\alpha_1 \Delta |\mathbb{S}| - \alpha_2 \Delta \phi - \alpha_3 \Delta L_{traj} + \alpha_4 \Delta \bar{\mu} \tag{4.17}$$

where $\Delta\odot$ represents the change of $\odot$'s value in the previous and current timesteps. In the end, the agent receives the total sum of these instantaneous rewards, plus the final reward as depicted in Equation (4.16). Note that in order for the agent to realize the importance of both immediate and long-term future rewards, a user-defined *discounted rate factor* $\gamma \in [0, 1]$ was introduced.

The Policy and DQN Networks. The policy π imposed on an agent is one that maximizes the future expected reward from the environment. A state-action pair at time t, denoted by (s_t, a_t) can be mapped to some quality index Q^π function represented as $Q^\pi(s_t, a_t)$. In other words, Q^π possess the maximum possible future expected reward in the environment for state-action pair (s_t, a_t):

$$Q^\pi(s_t, a_t) = \max\left[\mathbb{E}\left(R_t \mid s_t, a_t\right)\right] \tag{4.18}$$

Therefore, the agent's goal is to learn the most optimal policy π, through the selection of the action a_t while in state s_t that maximizes the Q-index. The idea of this Q-learning method is for the agent to record and keep track of all possible state-action (s_t, a_t) pairs and the Q-values they map to in a lookup table. In other words, it maintains a Q-table of values with $|\mathcal{S}|$ states and $|\mathcal{A}|$ actions. The Q-table is then updated at each timestep recursively:

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha_{lr}\left[\gamma \max_{a_{t+1}} Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)\right] \tag{4.19}$$

where α_{lr} is the learning rate. In fact, this Q-learning paradigm has seen significant success in the reinforcement learning community [137]. However, it can be observed that while our action space $\mathcal{A}$ is discrete, the state space $\mathcal{S}$ is continuous. As a result, the agent is unable to maintain large state-action spaces and therefore a nonlinear function approximator such as neural networks is necessary to estimate these Q-values. In particular, a deep reinforcement learning (DRL) architecture was employed, specifically a Deep Q-Network (DQN) algorithm was utilized as the proposed solution to this end (see Appendix C for other reinforcement learning policies and why DQN has been chosen over these other policies).

The Experience Replay Buffer. As there are no generated data for where the agent can learn the optimal actions, it would have to learn based on prior experience. More specifically, the agent (collects data of) keeps track of all state-action pairs and state-transitions it has had in the past so it can learn from them at a later time. The Epsilon-Greedy method was used to determine the most optimal action a_t while the agent collects the data; i.e., this Epsilon-Greedy policy is the data collection policy used by the agent and should not be confused with the Q-policy that the agent uses for evaluation and deployment. Moreover, the experience tuple records (s_t, a_t, r_t, s_{t+1}) are stored in a memory buffer called the *experience replay buffer* [40]. The agent samples a memory minibatch from this replay buffer and then calculates the (Huber) loss function. Note that this particular loss function is distinct from our proposed trajectory loss metric, where the former is calculated based on the agent's actions while the latter is based on the number of trajectories that cannot be represented by the pathlets in the pathlet set.

4.2.3 Space Complexity Analysis

One of the main motivations for why edge-disjoint pathlets have been used together with bottom-up approaches is due to the reduced memory storage requirements that is necessary to initialize pathlets, in contrast with previous works that utilize top-down schemes with overlapping, redundant pathlets. In fact, we provide a space complexity analysis of top-down approaches and compare this with the proposed bottom-up methods through the following theorem (see Appendix B.2 for the proof).

Theorem 4.2.2 (Initial Memory Storage Requirement Theorem) The memory space that is required by top-down methods for initializing a pathlet dictionary has a quadratic $\Theta(n^2)$ bound, with n as the number of segments of the road network. Bottom-up schemes on the other hand, such as the proposed PathletRL, requires only an initial $\Theta(n)$ amount of memory space, with n as the number of initial length-1 pathlets.

Chapter 5

Evaluation

In this chapter, we go over the experimental design, including information about the datasets, the evaluation metrics, the baseline methods, numerical results and insightful discussions. To keep the organization, as in Chapter 4, the entire chapter is split into two sections: Chapter 5.1 discusses the evaluation for the St2Box method, while Chapter 5.2 covers the evaluation for the proposed PathletRL.

5.1 Evaluating St2Box

5.1.1 Research Questions

To evaluate St2Box, we consider the following research questions:

(RQ 1.1) How does St2Box compare with the SotA methods, in terms of accuracy performance?

(RQ 1.2) What spatial element is best to use for expressing trajectories as sets (i.e., sets of points, cells or road segments)?

Feature	**T-Drive**	**Nyc**
# **nodes**	∼75K	∼78K
# **edges**	∼165K	∼121K
# **trajectories**	∼348K	∼634K
Observation period	1 week	1 month

Table 5.1: Attributes of the datasets used in the top k similarity search problem

(RQ 1.3) How robust is St2Box when varying the values of the spatiotemporal weight $\theta \in [0.0, 0.1, ..., 1.0]$?

(RQ 1.4) What impact does varying the values of the box smoothing parameter β bring to the performance of St2Box?

5.1.2 Datasets

To measure the effectiveness of St2Box, we utilize two popular taxi trajectory datasets on two large metropolitan urban cities of Beijing and New York. One is the T-Drive dataset[8] [179, 178] on the Beijing road network, while the other is Nyc[9] taxi dataset on the New York City road network. We initially preprocess the datasets (the procedure includes map-matching trajectories and filtering out those having less than 10 sampling points), which results into ∼348K trajectories in T-Drive and ∼634K trajectories for Nyc. See Table 5.1 for complete statistics of the datasets used.

5.1.3 Experimental Parameters

In comparing our methods with the baselines, we use the spatiotemporal weight parameter $\theta = 0.5$ (i.e., spatial and temporal aspects are equally important). The dimensions for the spatial and temporal embeddings are both set to 128. Moreover, the batch size is set to 512.

[8] `https://www.microsoft.com/en-us/research/publication/t-drive-trajectory/`
[9] `https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

The number of positive and negative training samples is both set to 10. Adam with learning rate of 0.001 was used for optimization. The box smoothing parameter β was set to 1.0.

5.1.4 Baselines

There are two main classifications of baseline methods that ST2BOX will be compared with: (1) *deep learning baselines*, that have some sort of learning component in their model architectures, and (2) *box-based architecture baselines*, that mainly use the SET2BOX [79] architecture with varying type of set element.

Deep Learning Baselines. We introduce the following baselines for evaluating ST2BOX. The first three deep learning models have been chosen, as they are popular methods that have yet to be compared with the state-of-the-art ST2VEC. Note that among all these deep learning baselines, only ST2VEC captures both the spatial and temporal features of trajectories; while all others capture the spatial aspect only.

- **CSSRNN** [164]. An extension to the traditional RNN that integrates topological constraints into the model.

- **TRAJ2VEC** [172]. Based on sliding windows and autoencoders to learn the fixed-length trajectory representations.

- **T2VEC** [85]. A sequence-based model for learning representations of trajectories that is robust to non-uniform, low sampling rates and noisy sample points.

- **ST2VEC** [39]. The current SOTA model that fuses spatial and temporal characteristics of trajectories.

Box-based Architecture Baselines. We also experiment on which type of element is best to pass onto SET2BOX [79] (only the spatial aspects are captured in the following baselines):

- **Pts2Box**. In this baseline, we pass geo-coordinate points of trajectories to the architecture for capturing similarity among trajectories based on trajectory points.

- **Hex2Box**. Instead of points, we express trajectories as a set of cells/hexagon blocks that the trajectories traverse[10].

- **Rds2Box**. Each trajectory is expressed as the set of all road segments from the road network traversed by that trajectory.

5.1.5 Evaluation Metrics

Ten percent of the preprocessed trajectory datasets $\mathcal{T}$ have been initially withheld as a query set $\mathcal{T}_q$. Then the remainder have been split into 50% training, 20% validation and 30% testing sets. Now to measure the effectiveness of the proposed method, we first pick a query trajectory τ_q selected uniformly at random from $\mathcal{T}_q$ and then choose the top $k = 100$ most similar trajectories to τ_q from the testing set $\mathcal{T}_{test}$ using the following similarity measures as ground truth(s), namely Tp [126], Dita [128], Lcrs [177], NetERP [72] and Fréchet [44] (while also ranking all other trajectories in $\mathcal{T}_{test}$ based on their similarity to τ_q). We refer the reader to Appendix D that provides more details on these five similarity measures. From the model's predictions and the ground truth(s), we consider the following evaluation metrics, where [$\downarrow$] and [$\uparrow$] denotes better performance with lower and higher values, respectively:

- **Hr** [$\uparrow$]. The top k *hitting ratio* captures the degree of overlap of the model's predicted top k most similar trajectories with that of the corresponding ground truth's.

- **Mae** [$\downarrow$]. The *mean absolute error* can be calculated by computing the average of all

[10]More formally, the map where the trajectories reside is first tessellated into regular disjoint hexagon blocks (called *polygon* objects in the context of computational geometry). Then the trajectory points are then map-matched onto the road network to form *linestrings* that represent the paths taken by the trajectories. And finally, trajectories are expressed as sets of hexagon blocks based on the intersection of these linestrings that they represent and the polygon (hexagons) objects from the tesselated map.

differences of the similarity rankings between the prediction and the ground truth. So if $\mathcal{R}(\tau)$ and $\hat{\mathcal{R}}(\tau)$ denote the function for the ground truth and predicted rankings of τ's similarity with τ_q, then the MAE can be computed as:

$$\frac{1}{|\mathcal{T}_{test}|} \sum_{\tau \in \mathcal{T}_{test}} |\mathcal{R}(\tau) - \hat{\mathcal{R}}(\tau)| \tag{5.1}$$

- **KT** [↑]. The *kendall-tau metric* measures the ordinal association of the predicted rankings versus the ranking of the ground truth. So then the KT measure is calculated as follows:

$$\frac{2}{|\mathcal{T}_{test}|(|\mathcal{T}_{test}| - 1)} \sum_{i<j} \operatorname{sgn}(\mathcal{R}(\tau_i) - \mathcal{R}(\tau_j)) \operatorname{sgn}(\hat{\mathcal{R}}(\tau_i) - \hat{\mathcal{R}}(\tau_j)) \tag{5.2}$$

5.1.6 Results and Discussion

In this Chapter, we go over each of the four research questions, present the experimental results and provide some discussion.

(RQ 1.1) Accuracy Performance. We evaluate the proposed method against four baselines, with ST2VEC [39] being the SOTA. Table 5.2 shows the numerical results. Notice two key observations. (i) Firstly, ST2VEC [39] and ST2BOX consistently outperform the three other baseline models (CSSRNN [164], TRAJ2VEC [172], and T2VEC [85]) in all five ground truth similarity measures for both T-DRIVE and NYC trajectory datasets. This is due to the fact that both ST2VEC [39] and ST2BOX take into account the temporal dimension in addition to the spatial aspect; while the other models only consider the spatial similarity. This demonstrates the importance of temporal features when learning the similarity in trajectories. (ii) Secondly, ST2BOX consistently outperforms ST2VEC [39] in terms of accuracy with the gain ranging between 0.6%-37.5% (T-DRIVE) and 1.2%-28% (NYC). Also, ST2BOX performs

	Model	**Tp**			**Dita**			**Lcrs**			**NetERP**			**Frechet**		
		Hr	Mae	Kt	Hr	Mae	Kt	Hr	Mae	Kt	Hr	Mae	Kt	Hr	Mae	Kt
T-Drive	CssRnn	0.618	1799	0.004	0.714	1667	0.228	0.642	2558	0.189	0.669	1746	0.107	0.705	1738	0.004
	Traj2Vec	0.651	1774	0.037	0.695	1642	0.272	0.636	2570	0.168	0.683	1732	0.111	0.679	1719	0.110
	T2Vec	0.640	1783	0.010	0.730	1650	0.259	0.629	2564	0.182	0.654	1760	0.046	0.728	1723	0.009
	ST2Vec	<u>0.745</u>	<u>1705</u>	<u>0.232</u>	<u>0.822</u>	<u>1565</u>	<u>0.413</u>	<u>0.781</u>	<u>2488</u>	<u>0.313</u>	<u>0.760</u>	<u>1683</u>	<u>0.210</u>	<u>0.826</u>	<u>1656</u>	<u>0.244</u>
	Pts2Box	0.679	1766	0.096	0.782	1624	0.320	0.656	2551	0.203	0.698	1719	0.128	0.753	1702	0.131
	Hex2Box	0.709	1744	0.157	0.803	1609	0.350	0.687	2548	0.245	0.711	1704	0.167	0.792	1689	0.165
	Rds2Box	0.718	1729	0.201	0.810	1598	0.387	0.722	2534	0.283	0.749	1697	0.195	0.813	1661	0.207
	ST2Box	**0.823**	**1682**	**0.319**	**0.862**	**1556**	**0.466**	**0.814**	**2465**	**0.350**	**0.786**	**1648**	**0.231**	**0.842**	**1635**	**0.269**
	% Impr.	10.5%	1.4%	37.5%	4.9%	0.6%	12.8%	4.2%	0.9%	11.8%	3.4%	2.1%	10.0%	1.9%	1.3%	10.2%
Nyc	CssRnn	0.649	2619	0.103	0.672	2491	0.027	0.621	2207	0.086	0.616	2069	0.101	0.679	1673	0.144
	Traj2Vec	0.611	2608	0.114	0.669	2470	0.039	0.659	2179	0.118	0.649	2090	0.089	0.720	1640	0.171
	T2Vec	0.678	2628	0.097	0.665	2453	0.060	0.640	2192	0.105	0.686	2041	0.122	0.707	1602	0.197
	ST2Vec	<u>0.842</u>	<u>2550</u>	<u>0.274</u>	<u>0.827</u>	<u>2348</u>	<u>0.207</u>	<u>0.749</u>	<u>2117</u>	<u>0.230</u>	<u>0.801</u>	<u>1936</u>	<u>0.243</u>	<u>0.820</u>	<u>1492</u>	<u>0.348</u>
	Pts2Box	0.725	2602	0.120	0.690	2417	0.108	0.682	2165	0.136	0.716	2009	0.166	0.748	1578	0.253
	Hex2Box	0.751	2584	0.198	0.741	2401	0.139	0.703	2146	0.188	0.741	1986	0.183	0.775	1561	0.286
	Rds2Box	0.760	2561	0.225	0.784	2367	0.174	0.716	2130	0.202	0.763	1958	0.207	0.791	1539	0.304
	ST2Box	**0.879**	**2498**	**0.333**	**0.883**	**2319**	**0.265**	**0.817**	**2085**	**0.283**	**0.835**	**1893**	**0.272**	**0.893**	**1463**	**0.399**
	% Impr.	4.4%	2.0%	21.5%	6.8%	1.2%	28.0%	9.1%	1.5%	23.0%	4.2%	2.2%	11.9%	8.9%	1.9%	14.7%

Table 5.2: Numerical results for similarity search task; bold/underlined numbers indicate best/second best method respectively. The last row shows the % improvement attained by St2Box from the state-of-the-art St2Vec.

significantly better in the Hr and Kt metrics. The proposed method, in addition to the SegStf for incorporating spatial and temporal dimensions, comes with an architecture for representing sets (road-based representations of trajectories) as boxes – which could explain its superiority. This special architecture is both *permutation-invariant* and *equivariant* for the road-based set representation of the trajectories, thus enhancing model performance; moreover, the box architecture is characterized by its *accuracy*, *versatility*, and *generalizability* [79] for representing sets while preserving similarity.

In addition, the improvement in performance offered by St2Box can be attributed to its ability to learn more refined levels of spatiotemporally-enriched information; first with Seg2Vec's architecture that is designed to capture the trajectory's road segments' spatiotemporal features. And then with Seg2Box that can maintain this information while preserving similarities among trajectories using box representations of road-based representation sets with desirable characteristics.

(RQ 1.2) Spatial Experiment. Next, we would like to know what the best spatial component to represent trajectories is (see Table 5.2 for the results). The most common is through raw GPS points (longitude/latitude) which are often collected by digital sensors, smart devices and other geo-tracking technologies. Another way to represent trajectories is through the use of block cells. In our case, we utilize hexagon-shaped grid tessellations[11] as they are known to be the most circular-shaped polygon that can minimally reduce bias in mobility data [16]. Lastly, trajectories can be represented as road segments. Each of these three representations is passed to the box architecture, and the output models are named as Pts2Box, Hex2Box, and Rds2Box, respectively. In all ground truth measures and in both datasets for all evaluation metrics, it can be observed how Rds2Box yields the best performance. Pts2Box does not seem to do well, likely due to raw Gps traces containing noisy data, many of which are not map-matched and do not align with the

[11]**Data source**: `https://github.com/alifa98/point2hex`

road network. Hex2Box is also outperformed by Rds2Box, likely because the hexagon-based representations of trajectories tend to underestimate and/or overestimate the exact paths/roads traversed by trajectories. In addition, the choice of resolution for hexagon tessellation on the map could also affect the overall results, and knowing the most suitable one can be challenging. In the end, we see that the best representation for these trajectories is that of the road segments. With St2Box, we improve the performance of Rds2Box by incorporating the temporal dimension, as explained in **(RQ 1.1)**.

(RQ 1.3) Robustness Study. We evaluate St2Box's robustness for varying values of the spatiotemporal weight $\theta \in [0.0, 0.1, ..., 1.0]$, where higher (lower) θ indicates more importance towards the spatial (temporal) aspect (refer back to Equation (3.4) for the spatiotemporal similarity equation). Figure 5.1 shows that the performance of St2Box remains stable on all three evaluation metrics for both datasets across all θ's, which indicates that it can provide support to a variety of applications as discussed in the Chapter 1. Only plots for $\theta \in [0.2, 0.4, 0.6, 0.8]$ are included but the same results hold for any other θ.

(RQ 1.4) Parameter Sensitivity. We also conduct a parameter sensitivity experiment on the box smoothing parameter β that can take any positive floating number. However, there appears to be no substantial changes to the results for values larger than 4.0 or smaller than 1.0. Hence, we only show the plots for $\beta \in [1.0, 2.0, 3.0, 4.0]$. Again, higher Hr and Kt values, as well as lower Mae numbers, indicate better performance. It can be observed that the best β parameter in both datasets for all ground truth measures based on these four values is $\beta = 1.0$. In fact, Figure 5.2 shows that for both datasets on the five ground truth measures, we observe higher Hr and Kt (similarly, seeing lower Mae values) by lowering β values. This is due to the fact that the β parameter controls the smoothing of the boxes, which also impacts the calculation for the boxes' volume. In particular, *harder boxes* (generated by higher β values) approximate the ReLU function (instead of the SoftPlus) when computing

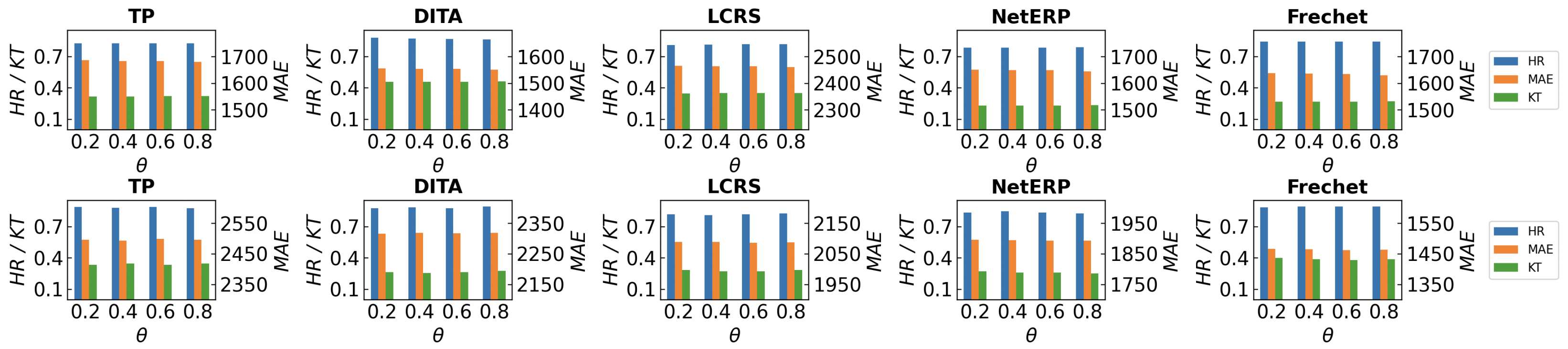


Figure 5.1: Robustness study of ST2BOX for varying values of the spatiotemporal weight $\theta \in [0.2, 0.4, 0.6, 0.8]$ for the five ground truth measures on the T-DRIVE (top) and NYC (bottom) datasets.

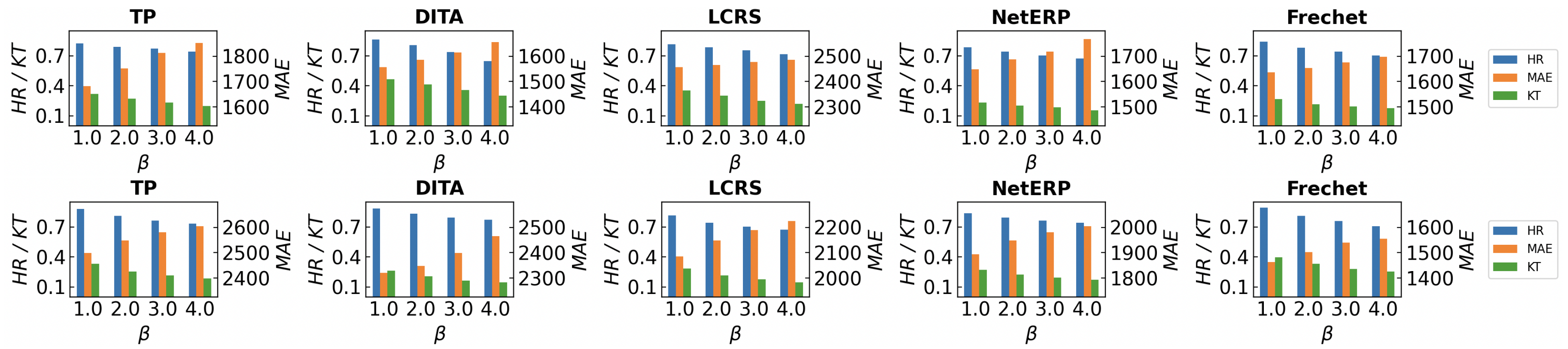


Figure 5.2: Parameter sensitivity analysis of the box smoothing parameter $\beta \in [1.0, 2.0, 3.0, 4.0]$ for the five ground truth measures on the T-DRIVE (top) and NYC (bottom) datasets.

box volumes. As a result, hard boxes pose a challenge in gradient-based optimization for disjoint boxes, as discussed in Chapter 4.1.4; this can explain St2Box's poorer performance with higher β values.

Summary of Key Observations. There are a few key insights that can be derived based on the results that can be observed from the experiments conducted in the four research questions above. The most important being, is that StBox outperforms all its baseline models. Indeed, while StVec outperforms other deep learning baselines due to its temporal learning and fusion modules, the proposed St2Box can offer a more enhanced improvement to the model thanks to its *highly-accurate*, *versatile* and *generalizable* box representations of the trajectories' road-based representation sets. It was also observed that the model is *robust* to spatiotemporal weight θ, and that the box smoothing parameter β tends to have better smoothing, and thus more improved results with lower parametric values. It was also shown how the road segments turned out to be the most effective spatial element in set-to-box representations, compared to points and hexagon cells. Computation of similarity values is moreover *fast* (as inherited from the Set2Box architecture [79]) and therefore is *scalable.*

5.2 Evaluating PathletRL

5.2.1 Research Questions

To evaluate PathletRL, we consider the following six research questions:

(RQ 2.1) How does PathletRL compare with the SotA methods, in terms of the quality of the extracted Pds?

(RQ 2.2) How much memory does the bottom-up approach save compared to top-down methods?

	Feature	TORONTO	ROME
roadmap	**# nodes**	∼1.9K	∼7.5K
	# edges / initial pathlets	∼2.5K	∼15.4K
trajectories	**Trajectory type**	realistic synthetic	real-world
	Object	cars	taxis
	# Total trajectories	∼169K	∼3.8M
	# Observation period	3.7 hours	1 week

Table 5.3: Attributes of the datasets used in the pathlet dicitonary construction problem

(RQ 2.3) How much improvement and how much more effective is the proposed PATHLETRL model against its ablation variations?

(RQ 2.4) What is the distribution of pathlet lengths in the obtained dictionary in our PATHLETRL model?

(RQ 2.5) How effective is the constructed PD in reconstructing the original trajectories?

(RQ 2.6) What is the sensitivity of the user-defined parameters $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ in the performance of our PATHLETRL model?

5.2.2 Datasets

Two datasets that each depict a different map scenario have been utilized to evaluate PATHLETRL (see Table 5.3 for its complete statistics)[12]. More specifically, we used real world maps of two metropolitan cities, Toronto[13] and Rome through the OpenStreetMaps[14]. A realistic synthetic vehicular mobility datasets for the TORONTO map was generated using the SUMO (**S**imulation of **U**rban **MO**bility) mobility app simulator[15] (3.7 hours). Moreover,

[12]The datasets used to evaluate ST2BOX can definitely be used to evaluate PATHLETRL, however we opted to use other datasets for variety purposes and for effective testing on both realistic synthetic and real-world datasets.
[13]`https://www.toronto.ca/city-government/data-research-maps/open-data/`
[14]`https://www.openstreetmap.org/`
[15]`https://www.eclipse.org/sumo/`

PathletRL Algorithm	Representability Measure	Weighted Networks	Deep Learning Policy
PathletRL-Nr	✗	✓	✓
PathletRL-Rnd	✓	✓	✗
PathletRL-Unw	✓	✗	✓
PathletRL (ours)	✓	✓	✓

Table 5.4: Features of the proposed PathletRL models, alongisde its ablation baselines.

larger-scale, real-world taxi cab trajectories (first week of February 2014) were taken from Crawdad[16] [18], an archive site for wireless network and mobile computing datasets, to form the Rome dataset. We split our trajectory sets into 70% training and 30% testing, where the training data was used to construct our pathlet dictionaries and the remainder for evaluation.

5.2.3 Experimental Parameters

To implement the RL architecture, a deep neural network that consists of the following parameters have been utilized. It comprises of three hidden fully-connected layers of 128, 64 and 32 hidden neurons. The ReLU activation function has been employed, optimized by Adam with a learning rate of 0.001. A 0.2 dropout in the network, together with the Huber loss function, have also been used.

More specific to the Dqn's parameters, there are a total of $m = 5$ episodes for each of the $n = 100$ iterations. The size of the experience replay buffer is 100,000 and the memory minibatch size is 64. The agent also uses a discount factor $\gamma = 0.99$. Moreover, the value $\chi = 10$ for the χ-order candidate set, $M = 25\%$ maximum trajectory loss and $\hat{\mu} = 80\%$ average representability threshold, were set. Then $\alpha_i = \frac{1}{4}, \forall i$, denoting equal importance for each of the four objectives as depicted in Equation (4.16).

[16] `https://crawdad.org/`

5.2.4 Baselines

The proposed PathletRL models will compete against the following baseline methods (see Appendix E as to why the following specific baselines are selected). The first two are SotA baselines that utilize top-down approaches, while the last three are ablation versions of our proposed model (see Table 5.4 for a summary of each ablation model and the features withheld in each version to demonstrate the importance and effectiveness of such features).

- **Chen et al.** [26]. The very first paper that introduces the notion of pathlets. This method frames the problem as an integer programming formulation, that is solvable using dynamic programming.

- **Agarwal et al.** [1]. Framing Pd extraction as a subtrajectory clustering problem, where subtrajectory clusters are treated as pathlets, they use *pathlet-cover* inspired from the popular *set-cover* algorithm.

- **Sgt**. The *Singleton* baseline considers all initial length-1 pathlets (the original road map), without merging any pathlets.

- **PathletRL-Rnd**. This version of PathletRL does not support Deep Q Networks and does not utilize a Dqn agent. Actions at each episodic timestep are taken uniformly at random.

- **PathletRL-Nr**. Trajectory representability is absent under this ablation. If a pathlet traversed by some trajectory τ merges with another pathlet that is not traversed by this τ, then there no longer exists a subset of pathlets in the pathlet set that can represent τ; as a result, immediately discard trajectory τ. Recall Remark 4.2.1 for details on a PathletRL model that withhelds the representability feature.

- **PathletRL-Unw**. This version of PathletRL is applied to a pathlet graph environment where all pathlets are equally weighted.

5.2.5 Evaluation Metrics

To evaluate model performance, we consider the following metrics that will measure the quality of the extracted pathlet dictionaries. Note that [↓] ([↑]) indicate that lower (higher) values are better.

(1) $|\mathbb{S}|$, the size of the pathlet dictionary [↓]

(2) ϕ, the average number of pathlets that represent each trajectory [↓]

(3) L_{traj}, the average number of trajectories discarded (expressed in percentage) [↓]

(4) $\bar{\mu}$, the average representability across the remaining trajectories (expressed in percentage) [↑]

Note that the third and fourth metrics above do not apply to Chen et al.'s [26] and Agarwal et al.'s [1] methods as such measures are only applicable to pathlet-merging methods. Moreover, the fourth metric does not apply to PathletRL-Nr. Under this model, all remaining trajectories in the dataset (and hence the average) are always 100% representable, which is not so interesting.

5.2.6 Results and Discussion

In this Chapter, we go over each of the six research questions, present the experimental results and provide some discussion.

(RQ 2.1) Quality of the Extracted PDs. We evaluate the pathlet dictionary extracted by our PathletRL algorithm against the PDs extracted by SotA baselines. See Table 5.5

		Baselines		NULL	PATHLETRL				**% Impr.**
		[26]	[1]	SGT	RND	NR	UNW	(OURS)	
TORONTO	$\|\mathbb{S}\|$	13,886	7,982	2,563	2,454	1,896	<u>1,801</u>	**1,743**	+3.22%
	ϕ	7.02	5.97	4.76	3.77	**2.89**	3.98	<u>3.75</u>	–22.9%
	L_{traj}	N/A	N/A	0%	19.7%	17.6%	**15.1%**	<u>15.2%</u>	–0.66%
	$\bar{\mu}$	N/A	N/A	100%	79.9%	N/A	<u>80.0%</u>	**83.9%**	+4.88%
ROME	$\|\mathbb{S}\|$	59,396	31,017	15,465	9,718	7,003	<u>5,804</u>	**5,291**	+8.84%
	ϕ	202.91	188.33	230.15	173.04	158.18	<u>146.39</u>	**139.89**	+4.44%
	L_{traj}	N/A	N/A	0%	24.9%	<u>21.1%</u>	22.9%	**20.4%**	+3.32%
	$\bar{\mu}$	N/A	N/A	100%	82.7%	N/A	**86.2%**	<u>85.6%</u>	–0.70%

Table 5.5: Numerical results showing the attributes of the pathlet dictionaries extracted by each method for each dataset.

for the numerical results, where the bold numbers indicate the result of the most superior model for the given PD metric and the underlined number is the result of the second-best performing model (note that we do not boldface or underline the numbers of SGT as such model serves as the null model where nothing is done to the pathlet graph). The last column of this table also highlights how much improvement the proposed model has on the quality of its extracted PD versus that of the best baseline model in the given evaluation metric; '+' indicates a better and '–' indicates lesser performance.

Although the nature of the pathlet definition and the approaches are not necessarily the same, the algorithms of Chen et al. [26] and Agarwal et al. [1] are still comparable. We ultimately show that their top-down approaches are not as effective as our bottom-up strategies. First, we look at the size of the pathlet dictionary, $|\mathbb{S}|$, where the smaller the number the better is the result. Our PATHLETRL model was able to improve from SGT by ∼32.0% (∼65.8%) for the TORONTO (ROME) dataset. These numbers are an ∼87.4% (∼91.1%) improvement from Chen et al.'s [26] model on the TORONTO (ROME) dataset. Our model also improves by ∼78.2% (∼82.9%) from Agarwal et al.'s [1] method on the TORONTO (ROME) dataset. These two observations indicate how superior our method is against the state of the art; i.e., bottom-up methods being better than top-down schemes. Note as well that Chen et al.'s [26] and Agarwal et al.'s [1] PDs are larger than the initial number of length-1 pathlets, as their methods are top-down – which initially considers all possible pathlet sizes and configurations (including overlaps). Clearly, they do not exhibit ideal results, compared to our proposed PATHLETRL model, as well as in all of the ablation versions of PATHLETRL.

Then, we focus on the metric of the average pathlet number that represents each trajectory (which can go up or down at each step of the iterative algorithm). Similar to $|\mathbb{S}|$, a smaller ϕ indicates a more ideal dictionary. Our PATHLETRL model was able to extract dictionaries that improve from SGT by ∼21.2% (∼39.2%) on the TORONTO (ROME) dataset. Meanwhile, Chen et al.'s [26] PD has ϕ quality ∼47.4% higher than SGT for TORONTO dataset, and only ∼11.8% lower than SGT on ROME dataset. A similar trend can be seen in Agarwal et al.'s [1] dictionary, with ∼25.4% higher and ∼18.2% lower than the initial number in the TORONTO and ROME datasets respectively. Clearly, our proposed PATHLETRL (and its ablation variations) outperforms these SOTA baselines.

Our PATHLETRL also improves from SGT based on the $|\mathbb{S}|$ and ϕ metrics. Because no action is taken on the pathlet graph in SGT, only the original numbers are shown; thus, L_{traj} and $\bar{\mu}$ remains as 0% and 100% for both datasets. However, we can see the benefits of PATHLETRL by trading off these values to obtain smaller dictionaries with much less ϕ scores – as controlled by the α parameters.

(RQ 2.2) Memory Efficiency. Figure 5.3 (in log scale) demonstrates how much more memory-efficient PATHLETRL is compared to the baselines [26, 1]. As can be seen in Figure 5.3, our method gets as input a set of trajectories that requires ∼900 MB (∼30+ GB) to be stored in memory, and builds a trajectory pathlet dictionary that requires a mere ∼100 KB (∼1 MB) for the TORONTO (ROME) dataset. In fact, this represents a ∼7.4K× (∼24K×) saving. This considerable improvement can be attributed to the fact that our method uses

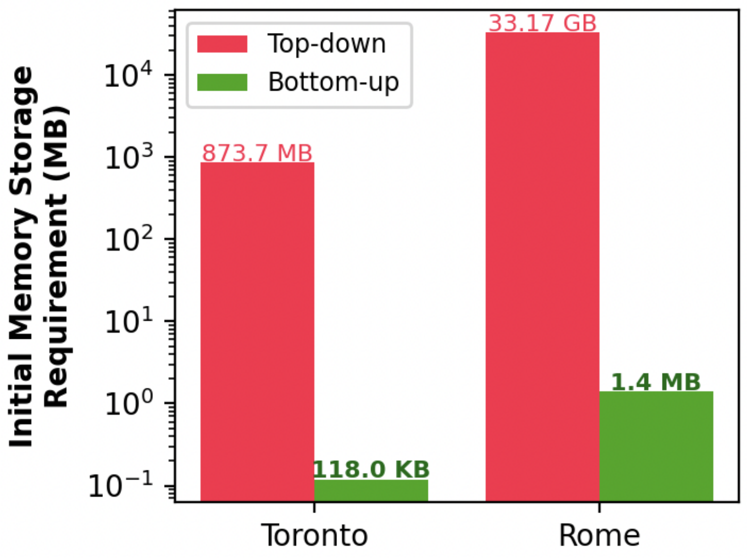


Figure 5.3: The initial memory required by top-down (existing) methods that use overlapping pathlets can be significantly reduced by the proposed bottom-up solution that use edge-disjoint pathlets, none of which are overlapping.

a bottom-up approach where only edge-disjoint pathlets are considered. In contrast, the current baselines follow a top-down approach, where the trajectory pathlet dictionary consists of overlapping pathlets of various sizes and configurations, most of which are redundant. The results of this empirical study well-supports the claim made in Theorem 4.2.2 of Chapter 4.2.3.

(RQ 2.3) Ablation Study. Next, we perform an ablation experiment to see how well our proposed PathletRL model performs. Figure 5.4 displays the average returns of PathletRL and its ablations across n iterations on the two datasets. We can observe similar trends on both datasets. Notice that PathletRL-Rnd has the poorest performance, exhibiting a random RL policy that shows no learning at all. Meanwhile, all other models demonstrate that their average return value converges after some iterations (for example, 15 and 20 iterations for PathletRL on the Toronto and Rome datasets), and then fluctuating slightly within a small range. PathletRL-Nr, while it demonstrates some level of learning due to the Dqn policy, does not perform well compared to PathletRL-Unw –

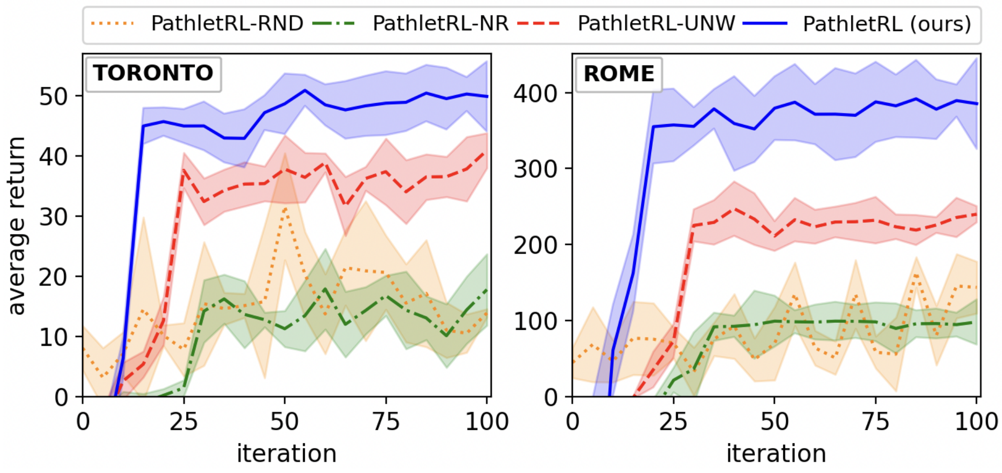


Figure 5.4: Performance evaluation of proposed and ablation PATHLETRL models, measured using the average return metric of $m = 5$ episodes across $n = 100$ iterations (run 10 times)

which suggests that representability is an essential component. This unweighted version of PATHLETRL can be seen as a runner up to our (weighted) proposed model, which indicates that there is some value to assigning pathlet weights than simply weighing all pathlets equally.

Besides comparing the trends of PATHLETRL models' performance evaluation, we can also look at the quality of their PDs (see Table 5.5 for the results). Generally speaking, our proposed PATHLETRL's PD is superior than the PDs extracted by its ablation variations (i.e., the $|\mathbb{S}|$ metrics for both the TORONTO and ROME datasets, the $\bar{\mu}$ metric for the TORONTO dataset, and the ϕ and L_{traj} metrics for the ROME dataset). In other cases that PATHLETRL did not rank first, it was a runner up but this can be explained. For instance, consider the ϕ metric in the TORONTO dataset. The reason for the higher quality of PATHLETRL-NR's PD than that of PATHLETRL's in terms of the ϕ metric is that because trajectory counts easily shrink faster in PATHLETRL-NR; the average ϕ can easily go down should the number of pathlets representing each trajectory also dwindle in number. The $\bar{\mu}$ metric for the PD of PATHLETRL-UNW is higher than that of the PD of PATHLETRL on the ROME dataset, which could be because PATHLETRL-UNW has fewer trajectories remaining in its trajectory set and that it just so happened that those that remained have high representability μ values.

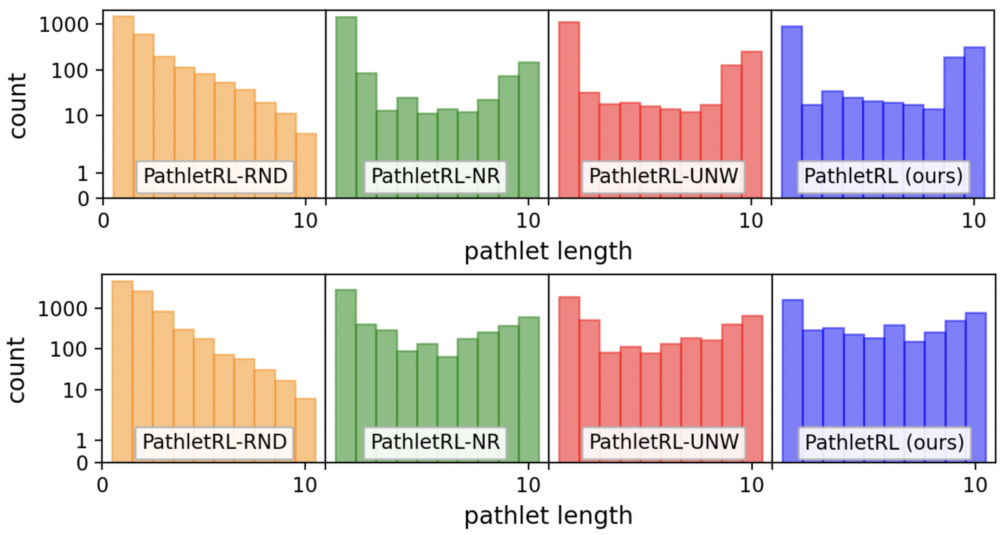


Figure 5.5: The pathlet length distribution of pathlet dictionaries obtained by PATHLETRL model and its ablation versions on the TORONTO (top) and ROME (bottom) datasets.

Regardless, the differences in numbers between the PDs of PATHLETRL and PATHLETRL-UNW in terms of the $\bar{\mu}$ metric is small and still comparable. The same can be said for the L_{traj} metric of the PDs of PATHLETRL-UNW and PATHLETRL on the TORONTO dataset, which differs by only a measly ∼0.1% – demonstrating that PATHLETRL is still competitive.

(RQ 2.4) Pathlet Length Distribution. We also analyze the length distribution of the pathlets in our dictionaries. The trend is similar for both datasets (Figure 5.5 shows the pathlet length distribution, with the y-axis in log scale). The PATHLETRL-RND has a decreasing number for longer pathlets, which is intuitive as the random policy blindly keeps and merges pathlets. It is harder to maintain longer pathlets in this random probabilistic manner as pathlets that terminate their growth are already considered "processed" and cannot grow further. As a result, it is more rare to see higher-ordered pathlets than shorter pathlets in PATHLETRL-RND's PD. The rest of the other RL models utilizing DQN policy have longer-length pathlets as expected (with more length-9 and 10 pathlets in the dictionary). Our proposed PD can capture more of these higher-ordered pathlets – indicating a smaller pathlet dictionary, as reflected by the results in Table 5.5. Meanwhile, we can still observe

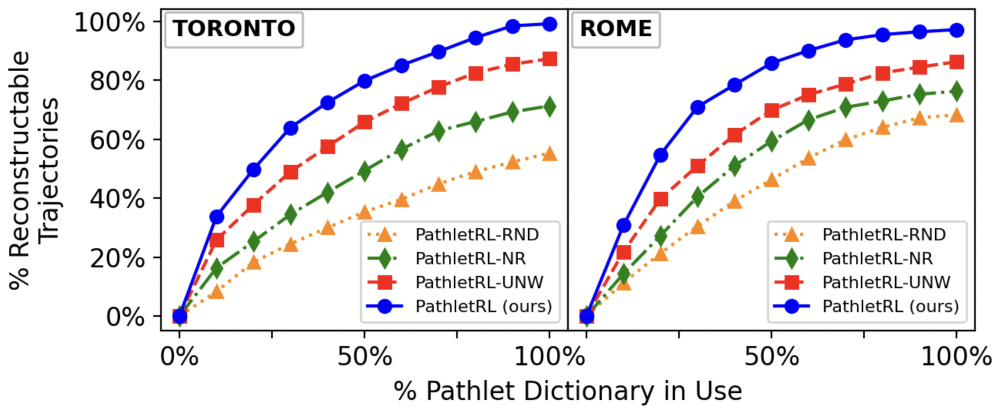


Figure 5.6: The percentage of evaluation trajectories reconstructable from a sample set taken from extracted pathlet dictionaries.

a large number of length-1 pathlets, which may be due to a number of reasons. One could be that some of the length-1 pathlets are still *unprocessed* (as a result of early stopping caused by the various termination criteria in our algorithm). It could also be that some of the length-1 processed pathlets are unmerged due to the algorithm's recommendation based on the utility, or perhaps based on pathlets losing all neighbor pathlets to merge with. The latter case depicts a scenario for where a length-1 processed pathlet ρ may have lost all its neighbor pathlets as a result of these neighbors merging amongst one another, leaving no way for ρ to merge with any of these formed merged pathlets.

(RQ 2.5) Partial Trajectory Reconstruction. See Figure 5.6 for a plot that displays the results of this experiment, where we determine how much of the dictionary is adequate enough to reconstruct most of our trajectories in our testing set. Here, we say that a trajectory is *reconstructable* if its representability value $\mu \geq 0.75$ (i.e., 75% of the trajectory can be represented by the PD). Anything less would mean that the trajectory is not reconstructable due to an excessive number of gaps. Ideally, we would like to take the top x% of the pathlets in the PD that are the most traversed by the trajectories in the training set. However, we can further remove the bias in the experiment by choosing instead a random sample of $x \in [10\%, 20\%, ..., 100\%]$ pathlets in the extracted PD, and measuring how much of the

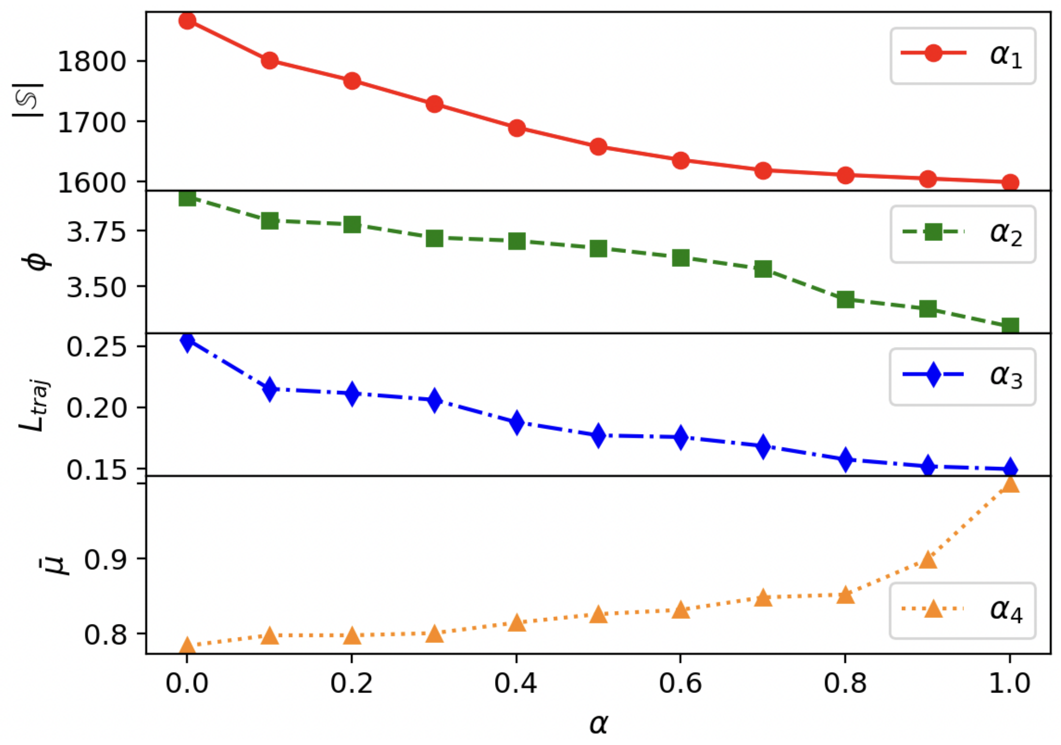


Figure 5.7: Parameter sensitivity experiment: the impact of α's on the quality of PathletRL's pathlet dictionary

trajectories in the testing set are reconstructable by this pathlet sample set. As shown in the results, by using around half of the pathlets in PathletRL's Pd, a good ∼80% (∼85%) of the trajectories in the Toronto (Rome) testing set can be reconstructed; and by using the whole dictionary, almost all trajectories in the set can be reconstructed. This shows that our proposed model is able to extract a high-quality pathlet dictionary. Comparing this with our ablation versions, they all follow a similar trend for both datasets – i.e., the amount of trajectories that the dictionaries of the ablation models can reconstruct is less than the amount that our proposed PathletRL's dictionary can do so. The worst being is PathletRL-Rnd's dictionary that can reconstruct up to only ∼50% (∼65%) of the trajectories in Toronto (Rome) given the entire dictionary.

(RQ 2.6) Parameter Sensitivity Analysis. We then discuss how the values of the four α values (α_1, α_2, α_3, α_4) affect the output Pd's quality in terms of its four attributes: $|\mathbb{S}|$,

ϕ, L_{traj}, $\bar{\mu}$ respectively[17]. There are four stacked plots, as seen in Figure 5.7. The first one depicts the changes to the PD's $|\mathbb{S}|$ as we vary $\alpha_1 = [0.0, 0.1, ..., 1.0]$ (while keeping the other α terms equal $\alpha_2 = \alpha_3 = \alpha_4 = \frac{1-\alpha_1}{3}$). We notice, as expected, a decreasing trend; larger α_1 values indicate higher importance to a smaller dictionary size. By varying α_2 and keeping the other terms equal, we can also observe a decreasing trend; higher α_2 values imply a stronger emphasis on lower ϕ scores – that is, lower average number of pathlets representing trajectories. Then when term α_3's value is varied, while keeping other α terms the same, it also shows a trend that decreases as we increases α_3. The greater the α_3 is, the more importance we put into keeping the trajectory loss low. Finally, varying α_4 while keeping the other α terms equal, shows an opposite trend than the other α terms. Here, we can observe that larger α_4 values are indicative of greater importance towards possessing higher representability values.

Summary of Key Observations. The results of the experiments as guided by the six research questions above have led us to a few key conclusions about the proposed methods. For one, the proposed PATHLETRL has demonstrated superior performance in terms of constructing a high-quality pathlet dictionary, in contrast with previous works. The deep reinforcement learning in its architecture has also helped in improving its performance, as existing works do not have a learning component in their method. The ablation experiments seem to also agree with this remark as withholding deep learning from PATHLETRL can cause poorer performance. In addition, the other PATHLETRL ablation versions seem to perform suboptimally compared to PATHLETRL, which indicates that trajectory representability and pathlet weights are important assets to PATHLETRL. Further proof that PATHLETRL is superior against its baselines in the trajectory reconstruction experiment, when PATHLETRL only requires half of the dictionary it generates to reconstruct a good amount ~85% of the

[17]**Note**: the goal here is not to find an optimal value for α_i, but intends on showing the trend of how $|\mathbb{S}|$, ϕ, L_{traj}, $\bar{\mu}$ changes with varying values of α_1, α_2, α_3, α_4 respectively.

trajectory set (that is in contrast with the random model that only reconstructs $\sim$65% of the trajectories but requires the entirety of the dictionary it produces). Moreover, it was shown empirically that the proposed method also have significant amount of memory savings compared to previous works.

Chapter 6

Conclusions

6.1 Summary and Contributions

The abundance of location-based tracking technologies brought forth huge amounts of mobility data, and as such mining interesting patterns in trajectories has become a hot area among researchers and practioners alike. Of particular interest is the search and retrieval of trajectories in a dataset that are most similar to a given query, i.e., the *top k trajectory similarity search task*. We first recognize that trajectories can be expressed as sets (of road segments), and then use this fact to represent these special kinds of sets as box representations (SEG2BOX). The spatiotemporally-enriched information, thanks to the architecture specifically for road segments known as SEG2VEC, have allowed enhancement to the proposed model performance. Overall, our proposed method ST2BOX is (i) *accurate*: achieving as much as ∼11%, ∼3% and a ∼38% improvement from the state-of-the-art baseline using the HR, MAE, and KT evaluation metrics respectively on two popular real world trajectory taxi datasets T-DRIVE and NYC, (ii) *versatile*: diverse to various similarity measures and applicable to various trajectory set elements (i.e., trajectory points, hexagon cells, road segments), (iii) *generalizable* to unseen trajectories as demonstrated by the improvement of model performance, (iv) *robust* to any spatiotemporal weight parameter $\theta \in [0, 1]$, and (v) *fast*: rapidly estimating trajectory similarity (as inherited from Set2Box [79]) and can therefore scale to larger datasets. The development of these more refined similarity search models can potentially offer greater improvements to various real world applications such as route planning, travel time prediction, and Poi recommendations.

We are also interested in this related, but fairly relatively new problem in the area of trajectory data mining, called the *trajectory pathlet dictionary construction* problem. The construction of these small sets of building blocks that are able to represent large numbers of trajectories, have become an important problem due to a number of applications such as route planning, travel time estimation, and trajectory compression. This work offers a deep (reinforcement) learning solution to the problem of interest, that can generate a dictionary that is 65.8% much smaller than the original dictionary, in contrast to non-learning-based methods. Further, only half of the pathlets in the proposed PathletRL's dictionary is necessary to reconstruct 85% of the original trajectory dataset; with baselines requiring the entirety of its dictionary to reconstruct only 65% the trajectories. Moreover, PathletRL also demonstrates a significant amount of memory savings by as much as $\sim$24K$\times$ in contrast with existing methods. This is due to the initial amount of memory required to store the initial pathlets of the dictionary, that was shown empirically and theoretically. The deep reinforcement learning component, representability and pathlet weights are all important assets to the PathletRL, as proven by the inferiority of its ablation variations.

6.2 Future Research Directions

Looking ahead, there are several directions for future work. This work introduced strong motivations as to how useful trajectory pathlet dictionaries are for representing several trajectories on the map, and moreover has presented a deep (reinforcement) learning-based method for constructing pathlet dictionaries. To bridge this with the work done related to trajectory similarity learning, we can use the dictionary generated by PathletRL to express trajectories as sets of pathlets (instead of points, hexagon cells or roads).

In addition, while our work has been mainly interested in finding and proposing solutions to trajectory similarity learning and trajectory pathlet dictionary construction, what has been left out are demonstrations of the importance in addressing these problems of interest. Therefore, another direction for future work involves conducting case studies, such a route planning or trajectory compression, using our proposed deep learning methods to emphasize the essence of these problems in the area of trajectory data mining.

Bibliography

[1] Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. “Subtrajectory Clustering: Models and Algorithms”. In: *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS ’18. Houston, TX, USA: Association for Computing Machinery, 2018, pp. 75–87. ISBN: 9781450347068.

[2] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. “Oddball: Spotting Anomalies in Weighted Graphs”. In: *Advances in Knowledge Discovery and Data Mining*. Vol. 6119. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 410–421. ISBN: 9783642136719.

[3] Fuad Aleskerov, Denis Bouyssou, and Bernard Monjardet. *Utility Maximization, Choice and Preference*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[4] Gian Alix, Nina Yanin, Tilemachos Pechlivanoglou, Jing Li, Farzaneh Heidari, and Manos Papagelis. “A Mobility-based Recommendation System for Mitigating the Risk of Infection during Epidemics”. In: *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*. 2022, pp. 292–295.

[5] Chrysovalantis Anastasiou, Constantinos Costa, Panos K. Chrysanthis, Cyrus Shahabi, and Demetrios Zeinalipour-Yazti. “ASTRO: Reducing COVID-19 Exposure through Contact Prediction and Avoidance”. In: 8.2 (Oct. 2022). ISSN: 2374-0353.

[6] Fazel Arasteh, Soroush SheikhGarGar, and Manos Papagelis. “Network-Aware Multi-Agent Reinforcement Learning for the Vehicle Navigation Problem”. In: *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL ’22. Seattle, Washington: Association for Computing Machinery, 2022. ISBN: 9781450395298.

[7] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. “Deep Reinforcement Learning: A Brief Survey”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[8] Takao Asano and Tomio Hirata. “Edge-contraction problems”. In: *Journal of Computer and System Sciences* 26.2 (1983), pp. 197–208. ISSN: 0022-0000.

[9] Stefan Atev, Grant Miller, and Nikolaos P. Papanikolopoulos. “Clustering of Vehicle Trajectories”. In: *IEEE Transactions on Intelligent Transportation Systems* 11.3 (2010), pp. 647–657.

[10] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. “Spatio-Temporal Data Mining: A Survey of Problems and Methods”. In: *ACM Computing Surveys* 51.4 (Aug. 2018). ISSN: 0360-0300.

[11] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. “Never Give Up: Learning Directed Exploration Strategies”. In: *International Conference on Learning Representations*. 2020.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: `1409.0473 [cs.CL]`.

[13] Hayat Dino Bedru, Shuo Yu, Xinru Xiao, Da Zhang, Liangtian Wan, He Guo, and Feng Xia. “Big networks: A survey”. In: *Computer Science Review* 37 (2020), p. 100247. ISSN: 1574-0137.

[14] Jon Louis Bentley. “Multidimensional Binary Search Trees Used for Associative Searching”. In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782.

[15] Ying Bi, Bing Xue, Pablo Mesejo, Stefano Cagnoni, and Mengjie Zhang. “A Survey on Evolutionary Computation for Computer Vision and Image Analysis: Past, Present, and Future Trends”. In: *IEEE Transactions on Evolutionary Computation* 27.1 (Feb. 2023), pp. 5–25.

[16] Colin PD Birch, Sander P Oom, and Jonathan A Beecham. “Rectangular and hexagonal grids used for observation, experiment and simulation in ecology”. In: *Ecological modelling* 206.3-4 (2007), pp. 347–359.

[17] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: `2004.10934 [cs.CV]`.

[18] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. *CRAWDAD dataset roma/taxi (v. 2014-07-17)*. Downloaded from `https://crawdad.org/roma/taxi/20140717`. July 2014.

[19] Ulrik Brandes. “A Faster Algorithm for Betweenness Centrality”. In: *Journal of Mathematical Sociology*. Vol. 25. 2001, pp. 163–177.

[20] Sergey Brin and Lawrence Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine”. In: *Computer Networks* 30 (1998), pp. 107–117.

[21] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya

Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[22] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[23] Deng Cai and Wai Lam. "Graph Transformer for Graph-to-Sequence Learning". In: *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 2020.

[24] Eren Cakmak, Manuel Plank, Daniel S. Calovi, Alex Jordan, and Daniel Keim. "Spatio-Temporal Clustering Benchmark for Collective Animal Behavior". In: *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Animal Movement Ecology and Human Mobility*. HANIMOB '21. Beijing, China: Association for Computing Machinery, 2021, pp. 5–8. ISBN: 9781450391221.

[25] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher RÃ©, and Kevin Murphy. "Machine Learning on Graphs: A Model and Comprehensive Taxonomy". In: *Journal of Machine Learning Research* 23.89 (2022), pp. 1–64.

[26] Chen Chen, Hao Su, Qixing Huang, Lin Zhang, and Leonidas Guibas. "Pathlet Learning for Compressing and Planning Trajectories". In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL'13. Orlando, Florida: Association for Computing Machinery, 2013, pp. 392–395. ISBN: 9781450325219.

[27] Lei Chen and Raymond Ng. "On the Marriage of Lp-Norms and Edit Distance". In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*. VLDB '04. Toronto, Canada: VLDB Endowment, 2004, pp. 792–803. ISBN: 0120884690.

[28] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. "Generative Pretraining From Pixels". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 1691–1703.

[29] Shu-Yu Chen, Wanchao Su, Lin Gao, Shihong Xia, and Hongbo Fu. "DeepFaceDrawing: Deep Generation of Face Images from Sketches". In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2020)* 39.4 (2020), 72:1–72:16.

[30] Wei Chen, Shuzhe Li, Chao Huang, Yanwei Yu, Yongguo Jiang, and Junyu Dong. "Mutual Distillation Learning Network for Trajectory-User Linking". In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. 2022.

[31] Yukun Chen, Kai Jiang, Yu Zheng, Chunping Li, and Nenghai Yu. "Trajectory Simplification Method for Location-Based Social Networking Services". In: *Proceedings of the 2009 International Workshop on Location Based Social Networks*. LBSN '09. Seattle, Washington: Association for Computing Machinery, 2009, pp. 33–40. ISBN: 9781605588605.

[32] Yuqi Chen, Hanyuan Zhang, Weiwei Sun, and Baihua Zheng. *RNTrajRec: Road Network Enhanced Trajectory Recovery with Spatial-Temporal Transformer*. 2022.

[33] Yu-Zhong Chen, Zi-Gang Huang, Shouhuai Xu, and Ying-Cheng Lai. "Spatiotemporal Patterns and Predictability of Cyberattacks". In: *PLOS ONE* 10.5 (May 2015). Ed. by Zhong-Ke Gao, e0124472. ISSN: 1932-6203.

[34] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning". en. In: *Archives of Computational Methods in Engineering* 27.4 (Sept. 2020), pp. 1071–1092. ISSN: 1134-3060, 1886-1784.

[35] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numer. Math.* 1.1 (Dec. 1959), pp. 269–271.

[36] Patrick Ebel, Ibrahim Emre Göl, Christoph Lingenfelder, and Andreas Vogelsang. *Destination Prediction Based on Partial Trajectory Data*. 2020.

[37] Ahmed Elragal and Nada El-Gendy. "Trajectory data mining: Integrating semantics". In: *Journal of Enterprise Information Management* 26.5 (2013), pp. 516–535. ISSN: 1758-7409.

[38] Michael R. Evans, Dev Oliver, Shashi Shekhar, and Francis Harvey. "Fast and Exact Network Trajectory Similarity Computation: A Case-Study on Bicycle Corridor Planning". In: *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*. UrbComp '13. Chicago, Illinois: Association for Computing Machinery, 2013.

[39] Ziquan Fang, Yuntao Du, Xinjun Zhu, Danlei Hu, Lu Chen, Yunjun Gao, and Christian S. Jensen. "Spatio-Temporal Trajectory Similarity Learning in Road Networks". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data*

Mining. KDD '22. Washington DC, USA: Association for Computing Machinery, 2022, pp. 347–356. ISBN: 9781450393850.

[40] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. "Revisiting Fundamentals of Experience Replay". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[41] Zhenni Feng and Yanmin Zhu. "A Survey on Trajectory Data Mining: Techniques and Applications". In: *IEEE Access* 4 (2016), pp. 2056–2067.

[42] Max Fischer. "Using Reinforcement Learning for Games with Nondeterministic State Transitions". MA thesis. 2019.

[43] Santo Fortunato. "Community detection in graphs". In: *Physics Reports* 486.3 (2010), pp. 75–174. ISSN: 0370-1573.

[44] M. Maurice Fréchet. "Sur quelques points du calcul fonctionnel". it. In: *Rendiconti del Circolo Matematico di Palermo* 22.1 (Dec. 1906), pp. 1–72. ISSN: 0009-725X, 1973-4409.

[45] Tao-Yang Fu and Wang-Chien Lee. "Trembr: Exploring Road Networks for Trajectory Representation Learning". In: *ACM Trans. Intell. Syst. Technol.* 11.1 (Feb. 2020). ISSN: 2157-6904.

[46] Yuanzhe Geng, Erwu Liu, Rui Wang, and Yiming Liu. "Deep Reinforcement Learning Based Dynamic Route Planning for Minimizing Travel Time". In: *CoRR* (2020). arXiv: 2011.01771.

[47] Balakrishna Gokaraju, Rajeev Agrawal, Daniel Adrian Doss, and Sambit Bhattacharya. "Identification of Spatio- Temporal Patterns in Cyber Security for Detecting the Signature Identity of Hacker". In: *SoutheastCon 2018*. 2018, pp. 1–5.

[48] Jonathan L. Gross, Jay Yellen, and Mark Anderson. *Graph Theory and Its Applications*. 3rd ed. Chapman and Hall/CRC, Nov. 2018. ISBN: 9780429425134.

[49] Aditya Grover and Jure Leskovec. "Node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 855–864. ISBN: 9781450342322.

[50] Ali Hamdi, Khaled Shaban, Abdelkarim Erradi, Amr Mohamed, Shakila Khan Rumi, and Flora D. Salim. "Spatiotemporal data mining: a survey on challenges and open problems". In: *Artificial Intelligence Review* 55.2 (Feb. 2022), pp. 1441–1488. ISSN: 0269-2821, 1573-7462.

[51] William L. Hamilton, Rex Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 1025–1035. ISBN: 9781510860964.

[52] Nan Han, Shaojie Qiao, Kun Yue, Jianbin Huang, Qiang He, Tingting Tang, Faliang Huang, Chunlin He, and Chang-An Yuan. "Algorithms for Trajectory Points Clustering in Location-Based Social Networks". In: *ACM Trans. Intell. Syst. Technol.* 13.3 (Mar. 2022). ISSN: 2157-6904.

[53] Peng Han, Jin Wang, Di Yao, Shuo Shang, and Xiangliang Zhang. "A Graph-Based Approach for Trajectory Similarity Computation in Spatial Networks". In: *Proc. of the 27th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining*. KDD '21. Virtual Event, Singapore: Assoc. for Comp. Machinery, 2021, pp. 556–564. ISBN: 9781450383325.

[54] Qingwen Han, Yu Lei, Lingqiu Zeng, Guangyan He, Lei Ye, and Lingfeng Qi. "Research on Travel Time Prediction of Multiple Bus Trips Based on MDARNN". In: *2021 IEEE*

International Intelligent Transportation Systems Conference (ITSC). 2021, pp. 3718–3725.

[55] Marwa Ibrahim Hassan and Sami Mustafa M. Elhassan. “Modelling of Urban Growth and Planning: A Critical Review”. In: *Journal of Building Construction and Planning Research* 08.04 (2020), pp. 245–262. ISSN: 2328-4889, 2328-4897.

[56] Tanmoy Hazra and Kushal Anjaria. “Applications of game theory in deep learning: a survey”. In: *Multimedia Tools and Applications* 81.6 (Mar. 2022), pp. 8963–8994. ISSN: 1380-7501, 1573-7721.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[58] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667.

[59] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. “Searching for Similar Trajectories on Road Networks Using Spatio-Temporal Similarity”. In: ADBIS’06. Thessaloniki, Greece: Springer-Verlag, 2006, pp. 282–295. ISBN: 3540378995.

[60] Huatao Jiang, Lin Chang, Qing Li, and Dapeng Chen. “Trajectory Prediction of Vehicles Based on Deep Learning”. In: *2019 4th International Conference on Intelligent Transportation Engineering (ICITE)*. 2019, pp. 190–195.

[61] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. “Self-supervised Trajectory Representation Learning with Temporal Regularities and Travel Semantics”. In: *2023 IEEE 39th international conference on data engineering (ICDE)*. IEEE. 2023.

[62] Peter Jin, Kurt Keutzer, and Sergey Levine. "Regret Minimization for Partially Observable Deep Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 2342–2351.

[63] Zhixiong Jin, Jiwon Kim, Hwasoo Yeo, and Seongjin Choi. "Transformer-based map-matching model with limited labeled data using transfer-learning approach". In: *Transportation Research Part C: Emerging Technologies* 140 (2022), p. 103668. ISSN: 0968-090X.

[64] L. P. Kaelbling, M. L. Littman, and A. W. Moore. "Reinforcement Learning: A Survey". In: *Journal of Artificial Intelligence Research* 4 (May 1996), pp. 237–285. ISSN: 1076-9757.

[65] D. Kavitha, B.V. Manikyala Rao, and V.Kishore Babu. "A Survey on Assorted Approaches to Graph Data Mining". In: *International Journal of Computer Applications* 14.1 (Jan. 2011), pp. 43–46. ISSN: 09758887.

[66] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. "Natural language processing: state of the art, current trends and challenges". en. In: *Multimedia Tools and Applications* 82.3 (Jan. 2023), pp. 3713–3744. ISSN: 1380-7501, 1573-7721.

[67] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[68] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. 2017.

[69] Nikita Kitaev, Thomas Lu, and Dan Klein. "Learned Incremental Representations for Parsing". In: *Proceedings of the 60th Annual Meeting of the Association for Com-*

putational Linguistics (Volume 1: Long Papers). Dublin, Ireland: Association for Computational Linguistics, May 2022.

[70] Jens Kober and Jan Peters. "Reinforcement Learning in Robotics: A Survey". In: *Learning Motor Skills: From Algorithms to Robot Experiments*. Cham: Springer International Publishing, 2014, pp. 9–67. ISBN: 978-3-319-03194-1.

[71] Christian Koetsier, Jelena Fiosina, Jan N. Gremmel, Jörg P. Müller, David M. Woisetschläger, and Monika Sester. "Detection of anomalous vehicle trajectories using federated learning". In: *ISPRS Open Journal of Photogrammetry and Remote Sensing* 4 (2022), p. 100013. ISSN: 2667-3932.

[72] Satoshi Koide, Chuan Xiao, and Yoshiharu Ishikawa. "Fast Subtrajectory Similarity Search in Road Networks under Weighted Edit Distance Constraints". In: *Proc. VLDB Endow.* 13.12 (July 2020), pp. 2188–2201. ISSN: 2150-8097.

[73] Ioannis Kontopoulos, Antonios Makris, and Konstantinos Tserpes. "TraClets: A trajectory representation and classification library". In: *SoftwareX* 21 (2023), p. 101306. ISSN: 2352-7110.

[74] Marc van Kreveld and Lionov Wiratma. "Median Trajectories Using Well-Visited Regions and Shortest Paths". In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '11. Chicago, Illinois: Association for Computing Machinery, 2011, pp. 241–250. ISBN: 9781450310314.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[76] Doi Thi Lan and Seokhoon Yoon. "Trajectory Clustering-Based Anomaly Detection in Indoor Human Movement". In: *Sensors* 23.6 (2023).

[77] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[78] Yann LeCun. "Generalization and network design strategies". In: 1989.

[79] Geon Lee, Chanyoung Park, and Kijung Shin. "Set2Box: Similarity Preserving Representation Learning for Sets". In: *2022 IEEE International Conference on Data Mining (ICDM)*. 2022, pp. 1023–1028.

[80] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. "Trajectory Outlier Detection: A Partition-and-Detect Framework". In: *2008 IEEE 24th International Conference on Data Engineering*. 2008, pp. 140–149. DOI: `10.1109/ICDE.2008.4497422`.

[81] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. "Trajectory Clustering: A Partition-and-Group Framework". In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. SIGMOD '07. Beijing, China: Association for Computing Machinery, 2007, pp. 593–604. ISBN: 9781595936868.

[82] Li Li, Rui Jiang, Zhengbing He, Xiqun (Michael) Chen, and Xuesong Zhou. "Trajectory data-based traffic flow studies: A revisit". In: *Transportation Research Part C: Emerging Technologies* 114 (2020), pp. 225–240. ISSN: 0968-090X.

[83] Tang Li, Jing Gao, and Xi Peng. "Deep Learning for Spatiotemporal Modeling of Urbanization". In: *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021)*. Sydney, Australia, 2021.

[84] Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. "Smoothing the Geometry of Probabilistic Box Embeddings". In: *International Conference on Learning Representations*. 2019.

[85] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. “Deep Representation Learning for Trajectory Similarity Computation”. In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 2018, pp. 617–628. DOI: 10.1109/ICDE.2018.00062.

[86] Yang Li, Dimitrios Gunopulos, Cewu Lu, and Leonidas J. Guibas. “Personalized Travel Time Prediction Using a Small Number of Probe Vehicles”. In: *ACM Trans. Spatial Algorithms Syst.* 5.1 (May 2019). ISSN: 2374-0353.

[87] Yang Li, Qixing Huang, Michael Kerber, Lin Zhang, and Leonidas Guibas. “Large-Scale Joint Map Matching of GPS Traces”. In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL’13. Orlando, Florida: Association for Computing Machinery, 2013, pp. 214–223. ISBN: 9781450325219.

[88] Yuxuan Liang, Kun Ouyang, Yiwei Wang, Xu Liu, Hongyang Chen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. “TrajFormer: Efficient Trajectory Classification with Transformers”. In: *Proceedings of the 31st ACM International Conference on Information amp; Knowledge Management*. CIKM ’22. Atlanta, GA, USA: Association for Computing Machinery, 2022, pp. 1229–1237. ISBN: 9781450392365.

[89] Xinyi Liu, Meiliu Wu, Bo Peng, and Qunying Huang. “Graph-based representation for identifying individual travel activities with spatiotemporal trajectories and POI data”. en. In: *Scientific Reports* 12.1 (Sept. 2022), p. 15769. ISSN: 2045-2322.

[90] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. “Map-Matching for Low-Sampling-Rate GPS Trajectories”. In: *the Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS ’09. Seattle, Washington: Association for Computing Machinery, 2009, pp. 352–361. ISBN: 9781605586496.

[91] Jeffrey Lund and Yiu-Kai Ng. "Movie Recommendations Using the Deep Learning Approach". In: *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. 2018, pp. 47–54.

[92] Nikola Marković, Przemyław Sekuła, Zachary Vander Laan, Gennady Andrienko, and Natalia Andrienko. "Applications of Trajectory Data From the Perspective of a Road Transportation Agency: Literature Review and Maryland Case Study". In: *IEEE Transactions on Intelligent Transportation Systems* 20.5 (2019), pp. 1858–1869.

[93] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. "Simplifying approach to node classification in Graph Neural Networks". In: *Journal of Computational Science* 62 (2022), p. 101695. ISSN: 1877-7503.

[94] Ken McCormick. "An Essay on the Origin of the Rational Utility Maximization Hypothesis and a Suggested Modification". In: *Eastern Economic Journal* 23.1 (1997), pp. 17–30. ISSN: 00945056, 19394632. (Visited on 12/16/2022).

[95] Thibaud Mérien, Xavier Bellekens, David Brosset, and Christophe Claramunt. "A Spatio-Temporal Entropy-Based Approach for the Analysis of Cyber Attacks (Demo Paper)". In: SIGSPATIAL '18. Seattle, Washington: Association for Computing Machinery, 2018, pp. 564–567. ISBN: 9781450358897.

[96] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2013.

[97] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari With Deep Reinforcement Learning". In: *NIPS Deep Learning Workshop*. 2013.

[98] Mashrur M. Morshed, Ahmad Omar Ahsan, Hasan Mahmud, and Md. Kamrul Hasan. *Learning Audio Representations with MLPs*. 2022. arXiv: `2203.08490 [cs.SD]`.

[99] Jonathan Muckell, Jeong-Hyon Hwang, Catherine T. Lawson, and S. S. Ravi. "Algorithms for Compressing GPS Trajectory Data: An Empirical Evaluation". In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '10. San Jose, California: Assoc. for Comp. Machinery, 2010, pp. 402–405.

[100] M Viswa Murali, T G Vishnu, and Nancy Victor. "A Collaborative Filtering based Recommender System for Suggesting New Trends in Any Domain of Research". In: *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. 2019, pp. 550–553.

[101] Mark E. J. Newman. *Networks*. Second edition. Oxford, United Kingdom; New York, NY, United States of America: Oxford University Press, 2018.

[102] Paul Newson and John Krumm. "Hidden Markov Map Matching through Noise and Sparseness". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '09. Seattle, Washington: Association for Computing Machinery, 2009, pp. 336–343. ISBN: 9781605586496.

[103] Peter J. Olver and Chehrzad Shakiban. *Applied Linear Algebra*. 2nd ed. 2018. Undergraduate Texts in Mathematics. Cham: Springer International Publishing: Imprint: Springer, 2018. ISBN: 9783319910413.

[104] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. "Deep content-based music recommendation". In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013.

[105] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. "Dota 2 with Large Scale Deep Reinforcement Learning". In: (2019).

[106] Takayuki Osogami and Rudy Raymond. "Map Matching with Inverse Reinforcement Learning". In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. IJCAI '13. Beijing, China: AAAI Press, 2013, pp. 2547–2553. ISBN: 9781577356332.

[107] Martijn van Otterlo and Marco Wiering. "Reinforcement Learning and Markov Decision Processes". In: *Reinforcement Learning: State-of-the-Art*. Ed. by Marco Wiering and Martijn van Otterlo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–42. ISBN: 978-3-642-27645-3.

[108] Costas Panagiotakis, Nikos Pelekis, Ioannis Kopanakis, Emmanuel Ramasso, and Yannis Theodoridis. "Segmentation and Sampling of Moving Object Trajectories Based on Representativeness". In: *IEEE Transactions on Knowledge and Data Engineering* 24.7 (2012), pp. 1328–1343.

[109] Daehee Park, Hobin Ryu, Yunseo Yang, Jegyeong Cho, Jiwon Kim, and Kuk-Jin Yoon. "Leveraging Future Relationship Reasoning for Vehicle Trajectory Prediction". In: *The Eleventh International Conference on Learning Representations*. 2023.

[110] Tilemachos Pechlivanoglou, Gian Alix, Nina Yanin, Jing Li, Farzaneh Heidari, and Manos Papagelis. "Microscopic Modeling of Spatiotemporal Epidemic Dynamics". In: SpatialEpi '22. Seattle, Washington: Association for Computing Machinery, 2022, pp. 11–21. ISBN: 9781450395434.

[111] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. “A Survey on Deep Learning: Algorithms, Techniques, and Applications”. In: *ACM Comput. Surv.* 51.5 (Sept. 2018). ISSN: 0360-0300.

[112] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. “Deep Learning for Audio Signal Processing”. In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019), pp. 206–219.

[113] Wanting Qin, Jun Tang, and Songyang Lao. “DeepFR: A trajectory prediction model based on deep feature representation”. In: *Information Sciences* 604 (2022), pp. 226–248. ISSN: 0020-0255.

[114] Gerar F. Quispe-Torres, Germain Garcia-Zanabria, Harley Vera-Olivera, and Lauro Enciso-Rodas. “Trajectory Anomaly Detection based on Similarity Analysis”. In: *2021 XLVII Latin American Computing Conference (CLEI)*. 2021, pp. 1–10.

[115] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. “Zero-Shot Text-to-Image Generation”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8821–8831.

[116] S. A. Rizvi, R. Tang, X. Jiang, X. Ma, and X. Hu. *Local Contrastive Learning for Medical Image Recognition*. 2023. arXiv: `2303.14153 [cs.CV]`.

[117] Benjamin Robira, Andrea Corradini, Federico Ossi, and Francesca Cagnacci. “Bridging Human Mobility to Animal Activity: When Humans Are Away, Bears Will Play”. In: *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Animal Movement Ecology and Human Mobility*. HANIMOB ’22. Seattle, Washington: Association for Computing Machinery, 2022, pp. 1–8. ISBN: 9781450395342.

[118] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. *Temporal Graph Networks for Deep Learning on Dynamic Graphs*. 2020.

[119] Mahmoud Sakr, Cyril Ray, and Chiara Renso. "Big mobility data analytics: recent advances and open problems". en. In: *GeoInformatica* 26.4 (Oct. 2022), pp. 541–549. ISSN: 1384-6175, 1573-7624.

[120] Swaminathan Sankararaman, Pankaj K. Agarwal, Thomas Mølhave, Jiangwei Pan, and Arnold P. Boedihardjo. "Model-Driven Matching and Segmentation of Trajectories". In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL'13. Orlando, Florida: Association for Computing Machinery, 2013, pp. 234–243. ISBN: 9781450325219.

[121] Atish Das Sarma, Anisur Rahaman Molla, Gopal Pandurangan, and Eli Upfal. "Fast distributed PageRank computation". In: *Theoretical Computer Science* 561 (2015). Special Issue on Distributed Computing and Networking, pp. 113–121. ISSN: 0304-3975.

[122] F. Scarselli, Sweah Liang Yong, M. Gori, M. Hagenbuchner, Ah Chung Tsoi, and M. Maggini. "Graph neural networks for ranking Web pages". In: *The 2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI'05)*. 2005, pp. 666–672.

[123] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80.

[124] Madeline C. Schiappa, Yogesh S. Rawat, and Mubarak Shah. "Self-Supervised Learning for Videos: A Survey". In: *ACM Comput. Surv.* (Dec. 2022). ISSN: 0360-0300.

[125] Peter Schulam and Raman Arora. “Disease Trajectory Maps”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[126] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Kai Zheng, and Panos Kalnis. “Trajectory Similarity Join in Spatial Networks”. In: *Proc. VLDB Endow.* 10.11 (Aug. 2017), pp. 1178–1189. ISSN: 2150-8097.

[127] Shuo Shang, Ruogu Ding, Kai Zheng, Christian S. Jensen, Panos Kalnis, and Xiaofang Zhou. “Personalized Trajectory Matching in Spatial Networks”. In: *The VLDB Journal* 23.3 (June 2014), pp. 449–468. ISSN: 1066-8888.

[128] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. “DITA: Distributed In-Memory Trajectory Analytics”. In: *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD ’18. Houston, TX, USA: Association for Computing Machinery, 2018, pp. 725–740. ISBN: 9781450347037.

[129] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. *A Survey of Deep Reinforcement Learning in Video Games*. 2019.

[130] Aditya Shrivastava, Jai Prakash V Verma, Swati Jain, and Sanjay Garg. “A deep learning based approach for trajectory estimation using geographically clustered data”. en. In: *SN Applied Sciences* 3.6 (June 2021), p. 597. ISSN: 2523-3963, 2523-3971.

[131] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 0028-0836, 1476-4687.

[132] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.

[133] Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. “Rep the Set: Neural Networks for Learning Set Representations”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pp. 1410–1420.

[134] Artur Strzelecki. “The Apple Mobility Trends Data in Human Mobility Patterns during Restrictions and Prediction of COVID-19: A Systematic Review and Meta-Analysis”. In: *Healthcare* 10.12 (2022).

[135] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. “A survey of trajectory distance measures and performance evaluation”. en. In: *The VLDB Journal* 29.1 (Jan. 2020), pp. 3–32. ISSN: 1066-8888, 0949-877X.

[136] Shao-Hua Sun, Te-Lin Wu, and Joseph J. Lim. “Program Guided Agent”. In: *International Conference on Learning Representations*. 2020.

[137] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[138] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Cham: Springer International Publishing, 2010. ISBN: 9783031004230.

[139] Yaguang Tao, Alan Both, Rodrigo I. Silveira, Kevin Buchin, Stef Sijben, Ross S. Purves, Patrick Laube, Dongliang Peng, Kevin Toohey, and Matt Duckham. “A comparative analysis of trajectory similarity measures”. In: *GIScience & Remote Sensing* 58.5 (2021), pp. 643–669.

[140] E. Tiakas, A.N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. “Searching for similar trajectories in spatial networks”. In: *Journal of Systems and Software* 82.5 (2009), pp. 772–788.

[141] Kevin Toohey and Matt Duckham. “Trajectory Similarity Measures”. In: *SIGSPATIAL Special* 7.1 (May 2015), pp. 43–50.

[142] Joseph Turian, Björn W. Schuller, Dorien Herremans, Katrin Kirchoff, Paola Garcia Perera, and Philippe Esling, eds. *HEAR: Holistic Evaluation of Audio Representations (NeurIPS 2021 Competition).* Vol. 166. Proceedings of Machine Learning Research. PMLR.

[143] Saif Ur Rehman, Kexing Liu, Tariq Ali, Asif Nawaz, and Simon James Fong. “A Graph Mining Approach for Ranking and Discovering the Interesting Frequent Subgraph Patterns”. In: *International Journal of Computational Intelligence Systems* 14.1 (Dec. 2021), p. 152. ISSN: 1875-6883.

[144] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems.* Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[145] Petar Veličkovi, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. “Graph Attention Networks”. In: *International Conference on Learning Representations.* 2018.

[146] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. "Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 263–272.

[147] M. Vlachos, G. Kollios, and D. Gunopulos. "Discovering similar multidimensional trajectories". In: *Proceedings 18th International Conference on Data Engineering*. 2002, pp. 673–684.

[148] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. "On the Limitations of Representing Functions on Sets". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6487–6494.

[149] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. "On Position Embeddings in BERT". In: *International Conference on Learning Representations*. 2021.

[150] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. "Attributed Graph Clustering: A Deep Attentional Embedding Approach". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 3670–3676.

[151] Jingyuan Wang, Ning Wu, and Wayne Zhao. "Personalized Route Recommendation With Neural Network Enhanced Search Algorithm". In: *IEEE Transactions on Knowledge & Data Engineering* 34.12 (Dec. 2022), pp. 5910–5924. ISSN: 1558-2191.

[152] Senzhang Wang, Jiannong Cao, and Philip S. Yu. "Deep Learning for Spatio-Temporal Data Mining: A Survey". In: *IEEE Transactions on Knowledge & Data Engineering* 34.08 (Aug. 2022), pp. 3681–3700. ISSN: 1558-2191.

[153] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. "A Survey on Trajectory Data Management, Analytics, and Learning". In: *ACM Comput. Surv.* 54.2 (Mar. 2021). ISSN: 0360-0300.

[154] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. "A Survey on Trajectory Data Management, Analytics, and Learning". In: *ACM Computing Survey* 54.2 (Mar. 2021). ISSN: 0360-0300.

[155] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Xiaolin Qin. "Fast Large-Scale Trajectory Clustering". In: *Proc. VLDB Endow.* 13.1 (Sept. 2019), pp. 29–42. ISSN: 2150-8097.

[156] Tingting Wang, Shixun Huang, Zhifeng Bao, J. Shane Culpepper, and Reza Arablouei. "Representative Routes Discovery from Massive Trajectories". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '22. Washington DC, USA: Association for Computing Machinery, 2022, pp. 4059–4069. ISBN: 9781450393850.

[157] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. "Deep Reinforcement Learning: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–15.

[158] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong. "STMARL: A Spatio-Temporal Multi-Agent Reinforcement Learning Approach for Cooperative Traffic Light Control". In: *IEEE Transactions on Mobile Computing* 21.06 (June 2022), pp. 2228–2242. ISSN: 1558-0660.

[159] Zhaobo Wang, Yanmin Zhu, Qiaomei Zhang, Haobing Liu, Chunyang Wang, and Tong Liu. “Graph-Enhanced Spatial-Temporal Network for Next POI Recommendation”. In: *ACM Trans. Knowl. Discov. Data* 16.6 (July 2022). ISSN: 1556-4681.

[160] Zheng Wang, Kun Fu, and Jieping Ye. “Learning to Estimate the Travel Time”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 858–866. ISBN: 9781450355520.

[161] Zheng Wang, Cheng Long, and Gao Cong. “Trajectory Simplification with Reinforcement Learning”. In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 2021, pp. 684–695.

[162] Zhongqiu Wang, Guan Yuan, Haoran Pei, Yanmei Zhang, and Xiao Liu. “Unsupervised learning trajectory anomaly detection algorithm based on deep representation”. In: *International Journal of Distributed Sensor Networks* 16.12 (2020), p. 1550147720971504.

[163] Julian Wiederer, Arij Bouazizi, Marco Troina, Ulrich Kressel, and Vasileios Belagiannis. “Anomaly Detection in Multi-Agent Trajectories for Automated Driving”. In: *Proceedings of the 5th Conference on Robot Learning*. Vol. 164. Proceedings of Machine Learning Research. PMLR, Nov. 2022, pp. 1223–1233.

[164] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. “Modeling Trajectories with Recurrent Neural Networks”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 3083–3090.

[165] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. “Graph Learning: A Survey”. In: *IEEE Transactions on Artificial Intelligence* 2.2 (2021), pp. 109–127.

[166] Wenwen Xia, Yuchen Li, Jianwei Tian, and Shenghong Li. “Forecasting Interaction Order on Temporal Graphs”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD ’21. Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 1884–1893. ISBN: 9781450383325.

[167] Jiajie Xu, Jing Zhao, Rui Zhou, Chengfei Liu, Pengpeng Zhao, and Lei Zhao. “Predicting Destinations by a Deep Learning based Approach”. In: *IEEE Transactions on Knowledge and Data Engineering* 33.2 (2021), pp. 651–666.

[168] Hao Xue, Bhanu Prakash Voutharoja, and Flora D. Salim. “Leveraging Language Foundation Models for Human Mobility Forecasting”. In: *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL ’22. Seattle, Washington: Association for Computing Machinery, 2022. ISBN: 9781450395298.

[169] Raj Kapoor Yadav, Giriraj Kishor, Himanshu, and Kishan Kashyap. “Comparative Analysis of Route Planning Algorithms on Road Networks”. In: *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. 2020, pp. 401–406.

[170] P. Yang, H. Wang, Y. Zhang, L. Qin, W. Zhang, and X. Lin. “T3S: Effective Representation Learning for Trajectory Similarity Computation”. In: *2021 IEEE 37th Intl. Conf. on Data Eng. (ICDE)*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2021, pp. 2183–2188.

[171] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. “Comp. Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach”. In: *2019 IEEE 35th Intl. Conf. on Data Eng. (ICDE)*. 2019, pp. 1358–1369.

[172] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. “Trajectory clustering via deep representation learning”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 3880–3887. DOI: 10.1109/IJCNN.2017.7966345.

[173] Byoung-Kee Yi, H.V. Jagadish, and C. Faloutsos. “Efficient retrieval of similar time sequences under time warping”. In: *Proceedings 14th International Conference on Data Engineering*. 1998, pp. 201–208.

[174] Jiang Yi and Zhang Zhenhao. “Route planning based on improved A algorithm”. In: *2017 Chinese Automation Congress (CAC)*. 2017, pp. 6939–6942.

[175] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. “Do Transformers Really Perform Badly for Graph Representation?” In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021.

[176] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. “Point-BERT: Pre-Training 3D Point Cloud Transformers with Masked Point Modeling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[177] Haitao Yuan and Guoliang Li. “Distributed In-memory Trajectory Similarity Search and Join on Road Network”. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, pp. 1262–1273.

[178] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. “Driving with Knowledge from the Physical World”. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’11. San Diego, California, USA: Association for Computing Machinery, 2011, pp. 316–324.

[179] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. “T-Drive: Driving Directions Based on Taxi Trajectories”. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS ’10. San Jose, California: Association for Computing Machinery, 2010, pp. 99–108.

[180] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guang-Zhong Sun. “An Interactive-Voting Based Map Matching Algorithm”. In: *2010 Eleventh International Conference on Mobile Data Management*. 2010, pp. 43–52.

[181] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. “Graph Transformer Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[182] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J Smola. “Deep Sets”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 3394–3404. ISBN: 9781510860964.

[183] Simone Zamboni, Zekarias Tilahun Kefato, Sarunas Girdzijauskas, Christoffer Norén, and Laura Dal Col. “Pedestrian trajectory prediction with convolutional neural networks”. In: *Pattern Recognition* 121 (2022), p. 108252. ISSN: 0031-3203.

[184] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. “Learning Joint Spatial-Temporal Transformations for Video Inpainting”. In: *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, pp. 528–543. ISBN: 978-3-030-58517-4.

[185] Hanyuan Zhang, Xingyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. “Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching”. In: *Proc. of the Twenty-Ninth Intl. Joint Conf. on Artificial Intelligence*. IJCAI’20. Yokohama, Yokohama, Japan, 2021. ISBN: 9780999241165.

[186] Hanyuan Zhang, Xingyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. “Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching”. In: *Proceedings of the Twenty-Ninth International Joint*

Conference on Artificial Intelligence. IJCAI'20. Yokohama, Yokohama, Japan, 2021. ISBN: 9780999241165.

[187] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. 2019.

[188] Muhan Zhang and Yixin Chen. "Link Prediction Based on Graph Neural Networks". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 5171–5181.

[189] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. "Deep Set Prediction Networks". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[190] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. "FSPool: Learning Set Representations with Featurewise Sort Pooling". In: *International Conference on Learning Representations*. 2020.

[191] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. "Contrastive Learning of Medical Visual Representations from Paired Images and Text". In: *Proceedings of the 7th Machine Learning for Healthcare Conference*. Ed. by Zachary Lipton, Rajesh Ranganath, Mark Sendak, Michael Sjoding, and Serena Yeung. Vol. 182. Proceedings of Machine Learning Research. PMLR, May 2022, pp. 2–25.

[192] Jing Zhao, Jiajie Xu, Rui Zhou, Pengpeng Zhao, Chengfei Liu, and Feng Zhu. "On Prediction of User Destination by Sub-Trajectory Understanding: A Deep Learning Based Approach". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. Torino, Italy: Association for Computing Machinery, 2018, pp. 1413–1422. ISBN: 9781450360142.

[193] Yan Zhao, Shuo Shang, Yu Wang, Bolong Zheng, Quoc Viet Hung Nguyen, and Kai Zheng. “REST: A Reference-Based Framework for Spatio-Temporal Trajectory Compression”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 2797–2806. ISBN: 9781450355520.

[194] Yu Zheng. “Trajectory Data Mining: An Overview”. In: *ACM Trans. Intell. Syst. Technol.* 6.3 (May 2015). ISSN: 2157-6904.

[195] Yue Zhou and Thomas Huang. “‘Bag of segments’ for motion trajectory analysis”. In: *2008 15th IEEE International Conference on Image Processing*. 2008, pp. 757–760.

Appendix A

The Broader Impact: Applications to Pathlet Dictionaries

Pathlets and pathlet dictionaries are found to be useful in a variety of practical applications and thus serves a broader impact to many reseachers and practioners alike. We discuss a few of their applications in more detail.

A.1 Trajectory Compression

Compression is the process of reducing the size of a trajectory while preserving its important spatiotemporal features [99]; it is more commonly useful when there is a need of transmitting or storing large trajectory datasets in much smaller, more limited resources (e.g., mobile devices, wireless networks, etc.). A common technique for trajectory compression involves sampling points in trajectories that carry the most significant amount of spatiotemporal information. With a pathlet dictionary, one is able to capture the pathlets in a trajectory's pathlet-based representation set that are most useful and could represent a large number of trajectories in the dataset.

A.2 Route Planning

While navigation services and route planning apps such as Google Maps[18] and Apple Maps[19] exist, they are not necessarily accessible to users when online (internet) connection is not available (e.g., network outages, poor wifi signals in remote areas, etc.). As such, there is a need for accessing maps and loading mobility data offline while at the same time efficiently answering majority of user queries (for example, path recommendation from Point A to B). Pathlet dictionaries can be useful in this case.

A.3 Trajectory Prediction

Prediction and forecasting of trajectories is important in several domains including transportation [92, 82], urban planning [55, 83], and healthcare [125, 110, 4, 134]. Pathlets could be useful by expressing trajectories as sequences of pathlets, predicting the next pathlet(s) in the sequence and then concatenating these predicted pathlets to form the predicted trajectories.

A.4 Anomaly Detection

Spotting outliers has been an active research direction [80, 163, 76]. One could use pathlets to detect trajectories that move or behave anomalously by expressing each trajectory as their pathlet-based representation set and then identifying which of those trajectories has a pathlet-based representation set that deviates from other trajectories.

[18] `http://maps.google.com/`
[19] `https://www.apple.com/maps/`

Appendix B

Proofs to Theorems

B.1 Proof of Theorem 4.2.1

Theorem 4.2.1 (Trajectory Representability Theorem). At any step i of the iterative Algorithm 4.2, then the trajectory representability μ of some trajectory $\tau \in \mathcal{T}$ by the end of that iteration i is equal to:

$$\mu_i(\tau) = \frac{\sum_{\rho' \in \Phi_i(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} \tag{4.14}$$

where Φ_0 and Φ_i are the pathlet-based representation of trajectory τ in the initial (iteration 0) and iteration i of the iterative algorithm respectively.

Before proceeding with the proof, we first introduce two facts that will prove useful in our main proof for the theorem. The first fact relates to pathlet lengths.

Fact 1. The pathlet length of some pathlet $\rho_A \in \mathcal{P}$, plus the pathlet length of a neighboring pathlet $\rho_B \in \mathcal{P}$ is equal to the pathlet length of ρ_{AB}, where ρ_{AB} is the pathlet formed when pathlets ρ_A and ρ_B are merged. In other words:

$$\ell(\rho_A) + \ell(\rho_B) = \ell(\rho_{AB}) \tag{B.1}$$

This fact is fairly intuitive and does not require further explanation. As a remark, note that either one of ρ_A or ρ_B (or both) has length 1. This is because we would never find two higher-ordered (i.e., a pathlet with length greater than 1) pathlets merging together. The current pathlet in process can only merge with a neighboring pathlet that have never been processed in the algorithm (they all have length 1) and any neighboring high-ordered pathlets have already been preprocessed and added to the dictionary.

Fact 2. The sequence of trajectory representabilities $\{\mu_i\}$ for some trajectory $\tau \in \mathcal{T}$ is monotonically non-increasing. In other words, the trajectory representability of τ at some iteration j of the algorithm, for some $i < j$ is less than or equal to its representability at iteration i:

$$\mu_i(\tau) \geq \mu_j(\tau) \tag{B.2}$$

This claim is straightforward and intuitive as it simply stems from the nature of Algorithm 4.2. As a remark, if $j = i + 1$, then $\mu_i(\tau) \geq \mu_{i+1}(\tau)$. This then implies that:

$$\mu_i(\tau) = \mu_{i+1}(\tau) + \epsilon \tag{B.3}$$

for some non-negative $\epsilon \geq 0$. Clearly, if there are no changes to the trajectory representability between iterations i and $(i + 1)$, then $\epsilon = 0$. Otherwise, if its representability decreased at this iteration, then $\epsilon > 0$.

Proof of Theorem 4.2.1.

We are then ready to provide proof to this theorem by means of induction. First, we claim that the theorem holds at the initial base case, i.e., when $i = 0$:

$$\mu_0(\tau) = \frac{\sum_{\rho' \in \Phi_0(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} = 1 = 100\%$$

which checks out since the trajectory representabilities of all trajectories $\tau \in \mathcal{T}$ have this representability value at the beginning of the algorithm. Now, assume that the theorem holds at some iteration $i = n \geq 0$:

$$\mu_n(\tau) = \frac{\sum_{\rho' \in \Phi_n(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)}$$

All it remains to complete the proof is to show that the theorem holds for when $i = n + 1$, i.e.,

$$\mu_{n+1}(\tau) = \frac{\sum_{\rho' \in \Phi_{n+1}(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)}$$

Now let current pathlet ρ_A be the pathlet that merges with its neighbor pathlet ρ_B to form ρ_{AB} at the iteration $(n + 1)$. At this point, there are three cases to consider.

Case 1. $\rho_A \notin \Phi_n(\tau)$ and $\rho_B \notin \Phi_n(\tau)$

This is the case when both these two pathlets that are candidate for merging are not in the pathlet-based representation set $\Phi_n(\tau)$. This means that there is no change to the pathlet-based representation of τ from iteration n to $(n + 1)$; i.e., $\Phi_{n+1}(\tau) = \Phi_n(\tau)$, as well as their representabilities at iteration $(n + 1)$, i.e., $\mu_{n+1}(\tau) = \mu_n(\tau)$.

$$\begin{aligned}
\mu_{n+1}(\tau) &= \mu_n(\tau) \\
&= \frac{\sum_{\rho' \in \Phi_n(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(by the Inductive Hypothesis)} \\
&= \frac{\sum_{\rho' \in \Phi_{n+1}(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && (\Phi_{n+1}(\tau) = \Phi_n(\tau) \text{ for Case 1})
\end{aligned}$$

Case 2. $\rho_A \in \Phi_n(\tau)$ and $\rho_B \in \Phi_n(\tau)$

In this case, both ρ_A and ρ_B are in $\Phi_n(\tau)$ and therefore their merged pathlet ρ_{AB} will be in the pathlet-based representation of τ at the next iteration $(n+1)$, i.e., $\rho_{AB} \in \Phi_{n+1}(\tau)$. Note that that both ρ_A and ρ_B will no longer be in $\Phi_{n+1}(\tau)$. Moreover in this case, it is clear as well that their representabilities of τ also do not change between iterations n and $(n+1)$. Thus we have:

$$\begin{aligned}
\mu_{n+1}(\tau) &= \mu_n(\tau) \\
&= \frac{\sum_{\rho' \in \Phi_n(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(by the Inductive Hypothesis)} \\
&= \frac{\sum_{\rho' \in \Phi_n(\tau) \setminus \{\rho_A, \rho_B\}} \ell(\rho') + \ell(\rho_A) + \ell(\rho_B)}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(separate } \rho_A \text{ and } \rho_B\text{)} \\
&= \frac{\sum_{\rho' \in \Phi_n(\tau) \setminus \{\rho_A, \rho_B\}} \ell(\rho') + \ell(\rho_{AB})}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(by Fact 1)} \\
&= \frac{\sum_{\rho' \in \Phi_n(\tau) \setminus \{\rho_A, \rho_B\} \cup \{\rho_{AB}\}} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(inclusion of the new merged pathlet } \rho_{AB}\text{)} \\
&= \frac{\sum_{\rho' \in \Phi_{n+1}(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(as } \Phi_{n+1}(\tau) = \Phi_n(\tau) \setminus \{\rho_A, \rho_B\} \cup \{\rho_{AB}\}\text{)}
\end{aligned}$$

Note that in the last step, to get $\Phi_{n+1}(\tau)$, this is simply the same as the previous $\Phi_n(\tau)$ except we take out the two pathlets that are candidate for merging $\{\rho_A, \rho_B\}$ and then add back the newly formed merged pathlet $\{\rho_{AB}\}$.

<u>Case 3</u>. $\rho_A \in \Phi_n(\tau)$ and $\rho_B \notin \Phi_n(\tau)$

Without loss of generality, this case should suffice for when only one of $\{\rho_A, \rho_B\}$ is in $\Phi_n(\tau)$. Unlike Cases 1 and 2 where the representability of τ did not change, here in Case 3 τ's representability will decrease. Following from Fact 2, there exists some positive $\epsilon > 0$ that satisfies the equality in Equation (B.3). This equation can be rewritten as:

$$\mu_{n+1}(\tau) = \mu_n(\tau) - \epsilon \tag{$*$}$$

(where i's are expressed as as n's). From here, it is not difficult to see that the ϵ we are after is basically the percentage of the trajectory that can no longer be represented by the pathlets in the dictionary at the next iteration $(n+1)$. Since we know that ρ_A, which was part of $\Phi_n(\tau)$ will no longer be part of $\Phi_{n+1}(\tau)$ due to its merge with ρ_B to form ρ_{AB} (and moreover, ρ_{AB} is not part of $\Phi_{n+1}(\tau)$ as $\rho_B \notin \Phi_n(\tau)$), then it must be that the portion covered by ρ_A in τ is the loss and is the ϵ that we want:

$$\epsilon = \frac{\ell(\rho_A)}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} > 0 \qquad (**)$$

Note that this value is positive as both $\ell(\rho_A) > 0$ and $\sum_{\rho \in \Phi_0(\tau)} \ell(\rho) > 0$. So now we have:

$$\begin{aligned}
\mu_{n+1}(\tau) &= \mu_n(\tau) - \epsilon && \text{(by } (*)\text{)}\\
&= \frac{\sum_{\rho' \in \Phi_n(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} - \epsilon && \text{(by the Inductive Hypothesis)}\\
&= \frac{\sum_{\rho' \in \Phi_n(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} - \frac{\ell(\rho_A)}{\sum_{\rho \in \Phi_0(\tau)}} && \text{(by } (**)\text{)}\\
&= \frac{\sum_{\rho' \in \Phi_n(\tau)} \ell(\rho') - \ell(\rho_A)}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)}\\
&= \frac{\sum_{\rho' \in \Phi_n(\tau) \setminus \{\rho_A\}} \ell(\rho') + \ell(\rho_A) - \ell(\rho_A)}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(separate } \rho_A\text{)}\\
&= \frac{\sum_{\rho' \in \Phi_n(\tau) \setminus \{\rho_A\}} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)}\\
&= \frac{\sum_{\rho' \in \Phi_{n+1}(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)} && \text{(as } \Phi_{n+1}(\tau) = \Phi_n(\tau) \setminus \{\rho_A\}\text{)}
\end{aligned}$$

Thus, on the basis of the inductive hypothesis, we just have shown that:

$$\mu_{n+1}(\tau) = \frac{\sum_{\rho' \in \Phi_{n+1}(\tau)} \ell(\rho')}{\sum_{\rho \in \Phi_0(\tau)} \ell(\rho)}$$

Clearly, the theorem holds true for all $i \geq 0$, which completes the proof. $\square$

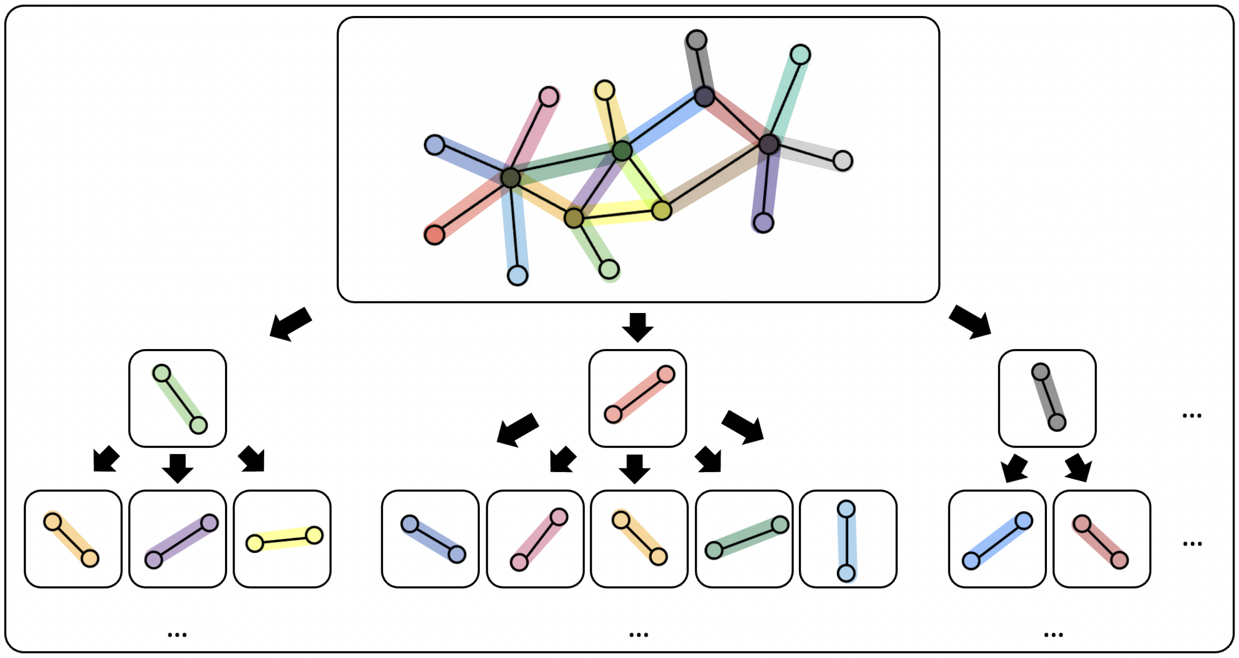


Figure B.1: An illustration of a pathlet tree. The root contains the road network and its children are the length-1 pathlets. Each of these length-1 pathlets have children which are their pathlet neighbors, and so forth.

B.2 Proof of Theorem 4.2.2

Theorem 4.2.2 (Initial Memory Storage Requirement Theorem). The memory space that is required by top-down methods for initializing a pathlet dictionary has a quadratic $\Theta(n^2)$ bound, with n as the number of segments of the road network. Bottom-up schemes on the other hand, such as the proposed PathletRL, requires only an initial $\Theta(n)$ amount of memory space, with n as the number of initial length-1 pathlets.

To prove that the proposed methods can significantly reduce the amount of memory space needed to store the initial pathlets in contrast to existing works, we first analyze the space complexity for the baseline top-down methods. Recall that there are n road segments in some arbitrary road network. Because existing models consider overlapping pathlets, as well as different varities, sizes and configurations of the pathlets (i.e., length-1, length-2, etc.), then we can simply count how many pathlets can be derived for each pathlet length:

- For length-1, there are clearly n such pathlets.
- For length-2, such pathlets can be formed as follows. Choose one of the n pathlets calling it ρ; and then affix a pathlet neighboring to ρ (called $\rho' \in \Psi(\rho)$); see the *pathlet tree* in Figure B.1 for an illustrative depiction. Clearly, there are $|\Psi(\rho)|$ choices. With n pathlets, then there are:

  $$|\Psi(\rho_1)| \times |\Psi(\rho_2)| \times ... \times |\Psi(\rho_n)| = \prod_{i=1}^{n} |\Psi(\rho_i)| \tag{B.4}$$

  total number of length-2 pathlets. However, this is not precisely correct since there are multiple pathlets that are counted twice. To fix this, it is enough to count the number using *combinatorics*, where we "choose" two road segments out of the n that we have in the road network (i.e., $\binom{n}{2}$ length-2 pathlets).
- In fact, for any pathlet of length-ℓ, then there should be $\binom{n}{\ell}$ such pathlets. In total, there should be $\sum_{i=1}^{\ell} \binom{n}{i}$ pathlets. Letting $\ell = n$ at the worst case, then we can derive an upper bound for the memory space complexity in terms of the total number of pathlets of the top-down approach:

  $$\sum_{i=1}^{n} \binom{n}{i} = \mathcal{O}\left(\binom{n}{1}\right) + \mathcal{O}\left(\binom{n}{2}\right) + ... + \mathcal{O}\left(\binom{n}{n}\right) = \mathcal{O}\left(2^n\right) \tag{B.5}$$

The exponential upper bound analysis for the space complexity of the top-down schemes can however still be further improved. For example, this bound also takes into account all such combinations of pathlets that are non-neighboring and as such should be excluded from the total count. This specific worst case also assumes that the road network is a complete graph (i.e., all the intersections of the road network are connected to each other), that is not very realistic. Moreover, we can also find a more tighter bound for the space complexity (involving both the lower and upper bounds). We first consider the following two facts that will be useful in the later proof.

Fact 1. Let a_{ij} be the entry situated at the ith row and jth column of an $m \times m$ square matrix A. Moreover, let $\mathbf{x}$ be a column vector of ones that has dimension[20] $\dim(A) \times 1$. Then the sum of all entries of A is:

$$\sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} = \mathbf{x}^\top A\mathbf{x} \tag{B.6}$$

Here is a short proof and it is quite easy to show:

$$\mathbf{x}^\top A\mathbf{x} = \begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}^\top \begin{bmatrix}a_{11} & a_{12} & \dots & a_{1m}\\ a_{21} & a_{22} & \dots & a_{2m}\\ \vdots & \vdots & \ddots & \vdots\\ a_{m1} & a_{m2} & \dots & a_{mm}\end{bmatrix}\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix} = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}\begin{bmatrix}a_{11}+a_{12}+\dots+a_{1m}\\ a_{21}+a_{22}+\dots+a_{2m}\\ \vdots\\ a_{m1}+a_{m2}+\dots+a_{mm}\end{bmatrix}$$

$$= \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}\begin{bmatrix}\sum_{j=1}^{m} a_{1j}\\ \sum_{j=1}^{m} a_{2j}\\ \vdots\\ \sum_{j=1}^{m} a_{mj}\end{bmatrix} = \sum_{j=1}^{m} a_{1j} + \sum_{j=1}^{m} a_{2j} + \dots + \sum_{j=1}^{m} a_{mj} = \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}$$

This shows a way to represent the sum of the entries of some matrix, which should prove useful at a later time. Now, here is another fact that bounds the above *quadratic form*:

Fact 2. For a square symmetric matrix A (i.e., $A = A^\top$) with λ_{min} and λ_{max} as its smallest and largest eigenvalues respectively, then the quadratic form $\mathbf{x}^\top A\mathbf{x}$ is tightly bounded by the

[20]The $\dim(A)$ of a square matrix A is defined to be equal to the number of rows it has, and this is how we will define the dimension of a square matrix throughout this manuscript. In this case, $\mathbf{x}$ has dimension $m \times 1$.

squared L_2-norm of $\mathbf{x}$, for all $\forall \mathbf{x} \in \mathbb{R}^{\dim(A) \times 1}$:

$$\lambda_{min}||\mathbf{x}||_2^2 \leq \mathbf{x}^\top A \mathbf{x} \leq \lambda_{max}||\mathbf{x}||_2^2 \tag{B.7}$$

In other words, $\mathbf{x}^\top A \mathbf{x} = \Theta(||\mathbf{x}||_2^2)$. In order to keep the discussion focused, I omit the proof of this well-known fact and direct the reader to linear algebra books [103] if interested in reading its complete proof.

For a more careful consideration of counting the pathlets, we first focus on the nodes (road intersections) of the road network. If one was to construct an adjacency matrix A where entry $a_{ij} \in A$ equals 1 if there is an edge (or road segment) from i to j (or j to i as our road network is undirected by assumption) and 0 otherwise, then $a_{ij} \in A^\ell$ determines the number of paths of length ℓ to traverse node i to j (or vice-versa) on the road network represented by adjacency matrix A [101]. Assuming the absence of self-loops for simplicity, then the total number of paths of length ℓ for any pair of nodes in the road network is the sum of all entries in the upper diagonal (excluding the main diagonal) of A^ℓ. We express it as follows. Let $Q_\ell = A^\ell - \mathcal{D}(A^\ell)$, where $\mathcal{D}(A)$ is the matrix that contains the diagonal entries of A along its main diagonal and zeros elsewhere. And then the sum of the upper diagonal entries of Q_ℓ is simply $\frac{1}{2}\mathbf{x}^\top Q_\ell \mathbf{x}$, with $\mathbf{x}$ as $\dim(Q_\ell) \times 1$ column of ones (the sum of entries is in line with Fact 1 above in Equation (B.6)). And then the half here is to avoid double counting (i.e., the undirected graph from i to j is the same as j to i as a result of symmetry).

So clearly, if we want all the pathlets of at least length-1, then we can write it as follows:

$$\sum_{\substack{\ell \in \mathbb{Z}^+ \\ \mathbf{x}^\top Q_\ell \mathbf{x} \neq 0}} \frac{1}{2}\mathbf{x}^\top Q_\ell \mathbf{x} \tag{B.8}$$

To yield a lower bound on this summation with respect to the number of road segments in the road network (instead of the nodes/road intersections), consider the smallest possible number of nodes for n edges in the road network first. Each edge connects together two nodes; as a result, there can be at least $|\mathcal{V}| = n$ nodes. Thus, in the Equation (B.8), it turns out that the $\dim(Q_\ell) = n$ and $\dim(\mathbf{x}) = n \times 1$ as a result of $\dim(A) = n$.

Now, each of the terms in Equation (B.8) is lower-bounded by $\Omega(||\mathbf{x}||_2^2) = \Omega(n)$ as a consequence of the Fact 2 in Equation (B.7). To see why $\Omega(||\mathbf{x}||_2^2)$ reduces to $\Omega(n)$, first note that $\mathbf{x}$ is a column vector of ones with dimension $n \times 1$. Therefore, its squared L_2-norm is:

$$\left(\sqrt{\underbrace{1^2 + 1^2 + ... + 1^2}_{n}}\right)^2 = \left(\sqrt{n}\right)^2 = n \tag{B.9}$$

Note however that the eigenvalue λ_{min} should not be relevant towards the space complexity expressed in terms of the number of road segments. However, such space analysis is only for one of the n terms in the summation of Equation (B.8)[21]; thus the lower bound for the number of pathlets of the top-down scheme is $n \times \Omega(n) = \Omega(n^2)$.

A similar argument can be used to show that the upper bound on this is also quadratic in the number of edges in the road network[22], i.e., $\mathcal{O}(n^2)$. As a result, we can yield a tighter bound of the memory space complexity of the top-down methods to be $\Theta(n^2)$, which is a much better improvement on our earlier analysis.

The proposed bottom-up method PATHLETRL, which consumes less memory space to initially store the pathlets, is more efficient with $\Theta(n)$ memory space complexity. It is quite easier to analyze. Since we only consider length-1 edge-disjoint pathlets, it is easy to see how ours simply relies exactly on the number of segments in the road network – which happens to be n. □

[21]It can be noted that at the worst case, the longest pathlet length in the road network is n; so that is why the summation can have up to n terms

[22]Note that here, we have $2n$ nodes at most instead of just n nodes, but the result should still hold for the upper bound, which is $\mathcal{O}(n^2)$, as we only scaled the number of nodes by a factor of 2.

Appendix C

Other Deep Reinforcement Learning Policies

The choice of DQN method over others such as Actor-Critic (A2C) and Policy-gradient is due to a number of reasons. For one, DQNs are more sample efficient than Policy-gradient methods because learning happens using an experience replay buffer rather than learning from data collected with the current policy. Moreover, with small action spaces, DQN's are more stable and more efficient than A2C methods; a separate neural network computes (approximates) the target Q-values, which can reduce the variance in these estimates. In addition, implementing DQNs is more straightforward compared to Actor-critic and Policy-gradient methods that tend to be more complex in terms of architectures and hyperparameter tuning. This makes them easier to employ in real-world settings.

Appendix D

Ground Truth Similarity Measures

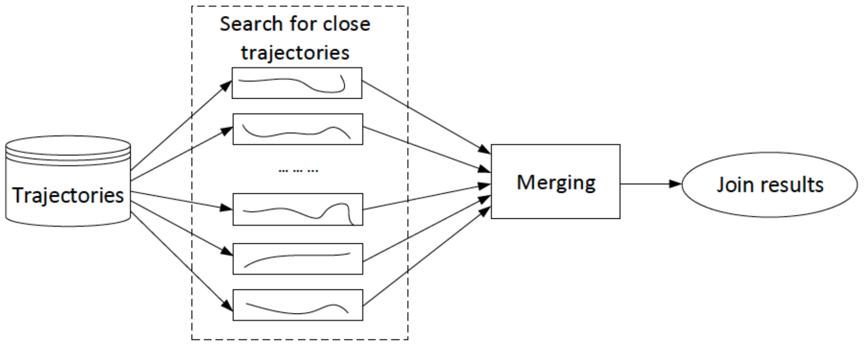
In the Top k Similarity Search Problem, five similarity measures have been utilized as ground truths. We provide some in-depth description of these ground truth measures in this appendix.

D.1 Two-Phase (Tp)

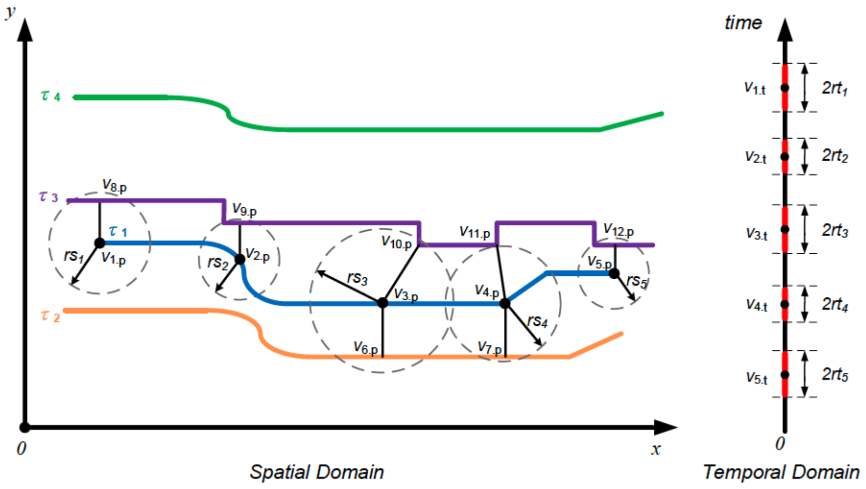
Tp [126], or the Two-Phase algorithm, is a divide-and-conquer strategy that comprises of two main steps: (1) the *trajectory-search phase*, that is responsible for concurrently exploring the spatial and temporal domains (the network expansion [35] and timestamp search for each trajectory τ respectively); and (2) the *merging phase* that combines together all the computed results for all the trajectories' searches. The illustration in Figure D.1 should give some idea of what the method is.

D.2 Distributed In-Memory Trajectory Analytics (Dita)

Dita [128], or Distributed In-Memory Trajectory Analytics, is a user-friendly distributed analytics system on Spark that supports trajectory similarity search under the following basic



(a) Parallel mechanism



(b) Trajectory search

Figure D.1: An example of the Tp Method; illustration taken from Shang et al. [126]

framework. Firstly, it utilizes global indexing to calculate relevant partitions that consists of trajectories similar to the query trajectory τ_q, and then sends it to the corresponding workers of these partitions. Next, for each of the partitions, the local indexing is used to generate candidate trajectories and then verify their similarity with τ_q. Finally, the results are collected and returned to the user.

D.3 Longest Common Road Segment (Lcrs)

The Longest Common Road Segment (Lcrs) is a new similarity function defined by Yuan et al. [177] according to the following formula. For two trajectories τ_i an τ_j:

$$\mathcal{S}_{\text{Lcrs}} = \frac{\varphi(\tau_i, \tau_j)}{|\tau_i| + |\tau_j| - \varphi(\tau_i, \tau_j)} \tag{D.1}$$

where we remind the reader that $|\tau|$ denotes the length of trajectory τ on the road network as per the preliminary chapters. The $\varphi(\tau_i, \tau_j)$ of $\tau_i = \langle r_i^{(1)}, ..., r_i^{(|\tau_i|)} \rangle$ and $\tau_j = \langle r_j^{(1)}, ..., r_j^{(|\tau_j|)} \rangle$ (the road segments of trajectories τ_i and τ_j can then be calculated recursively as follows:

$$\varphi(\tau_i, \tau_j) = \begin{cases} 0 & \text{if } |\tau_i| = 0 \vee |\tau_j| = 0 \\ |\tau_i[|\tau_i|]| + \varphi(\tau_i[1..(|\tau_i|-1)], \tau_i[1..(|\tau_j|-1])) & \text{if } \tau_i[|\tau_i|] = \tau_j[|\tau_j|] \\ \max\left\{\varphi\left(\tau_i\left[1..\left(|\tau_i|-1\right)\right], \tau_j\right), \varphi\left(\tau_i, \tau_i\left[1..\left(|\tau_j|-1\right)\right]\right)\right\} & \text{otherwise} \end{cases} \tag{D.2}$$

Note that $\tau_i[m]$ refers to the mth road segment of τ_i, i.e., $\tau_i[m] = \langle \tau_i^{(m)} \rangle$. And also $\tau_i[1..m]$ denotes the 1st to the mth road segments of τ_i, i.e., $\tau_i[1..m] = \langle \tau_i^{(1)}, ..., \tau_i^{(m)} \rangle$.

D.4 Network-aware Erp (NetERP)

The Network-aware Edit Distance with Real Penalty (NetERP) [72] extends from the Edit Distance with Real Penalty (Erp) [27] by replacing the Euclidean distance in Erp with the shortest path distance. This was mostly motivated from previous works that employ shortest path distance [38, 59, 126, 127, 140] between two nodes u and v in the road network.

D.5 Fréchet Distance (Fréchet)

The Fréchet distance [44] of two trajectories $\tau_a = \langle a_1, ..., a_m \rangle$ $\tau_b = \langle b_1, ..., b_n \rangle$ (where each trajectory is expressed as a sequence of points) is calculated recursively as follows:

$$\mathcal{S}_F(\tau_a, \tau_b) = \begin{cases} \max\limits_i d(a_i, b_1) & \text{if } n = 1 \\ \max\limits_j d(a_1, b_j) & \text{if } m = 1 \\ \max\Big\{ d(a_m, b_n), & \\ \quad \min\Big\{ \mathcal{S}_F(\tau_a[1..(m-1)], \tau_b[1..(n-1)]), & \\ \quad\quad \mathcal{S}_F(\tau_a[1..(m-1)], \tau_b), & \text{otherwise} \\ \quad\quad \mathcal{S}_F(\tau_a, \tau_b[1..(n-1)]) \Big\} & \\ \Big\} & \end{cases} \tag{D.3}$$

Appendix E

The Choice of Baselines against PathletRL

While there have been a number of considerable baseline methods [156, 108, 195] that can be used to compare PathletRL with, there are some reasons we decided to not do so. Firstly, Wang et al. [156] focuses on route representativeness discovery which is a completely different task than ours. Theirs, including Panagiotakis et al. [108] use a representativeness criterion that is not applicable to our case. Zhou et al. [195] moreover focuses on motion analysis which is a substantially different setting and task. This is in contrast to Chen et al. [26] and Agarwal et al. [1], where we have opted for their baselines methods due to a number of reasons. The former is the original and most representative work on pathlet dictionary construction, while the latter is the most representative/newest method on subtrajectory clustering that frames the pathlet dictionary construction as a subtrajectory clustering task. Both of these baselines also do not rely on learning-based methods, compared to our proposed model where we show the importance of deep learning.

As an aside, it is important to note that the technical problem of trajectory pathlet dictionary construction is novel and state-of-the-art methods are originating from the original work of Chen et al. [26]. We revisit the problem with a RL-based approach and consider additional constraints (such as edge-disjointness in pathlets and the χ-order constraint).

Appendix F

Reproducibility

I refer the reader to Chapters 5.1.3 and 5.2.3 for the experimental paramaters used for St2Box and PathletRL respectively. Moreover, the models have been implemented using `Python 3.10` using an `Intel Core i7-9700, 3.00GHz CPU, 62GB RAM` and a `GeForce RTX 2800 8GB GPU`. PathletRL was based on `Tensorflow`'s `tf-agents`[23] (version `0.15.0` at the time of development), while St2Box uses `PyTorch 1.13`. Their Github repositories are also available for public access[24,25].

[23]`https://www.tensorflow.org/agents`
[24]`https://github.com/techGIAN/PathletRL`
[25]`https://github.com/techGIAN/ST2Box`