PathletRL: Trajectory Pathlet Dictionary Construction using Reinforcement Learning

Gian Alix
York University
gcalix@eecs.yorku.ca

Manos Papagelis
York University
papaggel@eecs.yorku.ca

ABSTRACT

Sophisticated location and tracking technologies have led to the generation of vast amounts of trajectory data. Of interest is constructing a small set of basic building blocks that can represent a wide range of trajectories, known as a *trajectory pathlet dictionary*. This dictionary can be useful in various tasks and applications, such as trajectory compression, travel time estimation, route planning, and navigation services. Existing methods for constructing a pathlet dictionary use a top-down approach, which generates a large set of candidate pathlets and selects the most popular ones to form the dictionary. However, this approach is memory-intensive and leads to redundant storage due to the assumption that pathlets can overlap. To address these limitations, we propose a bottom-up approach for constructing a pathlet dictionary that significantly reduces memory storage needs of baseline methods by multiple orders of magnitude (by up to ~24K× better). The key idea is to initialize unit-length pathlets and iteratively merge them, while maximizing utility. The utility is defined using newly introduced metrics of *trajectory loss* and *representability*. A deep reinforcement learning method is proposed, PathletRL, that uses Deep Q Networks (DQN) to approximate the utility function. Experiments show that our method outperforms the current state-of-the-art, both on synthetic and real-world data. Our method can reduce the size of the constructed dictionary by up to 65.8% compared to other methods. It is also shown that only half of the pathlets in the dictionary is needed to reconstruct 85% of the original trajectory data.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies**;

KEYWORDS

mobility data, trajectory data mining, pathlet dictionary

ACM Reference Format:
Gian Alix and Manos Papagelis. 2023. PathletRL: Trajectory Pathlet Dictionary Construction using Reinforcement Learning. In *The 31st ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '23), November 13–16, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589132.3625622

SIGSPATIAL '23, November 13–16, 2023, Hamburg, Germany

ACM ISBN 979-8-4007-0168-9/23/11…$15.00
https://doi.org/10.1145/3589132.3625622

Figure 1: (a) Graph representation of a small area in Toronto[1]; (b) Example of various-length edge-disjoint pathlets in (a).

1 INTRODUCTION

Motivation & Problem of Interest. 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. Mining trajectory data to find interesting patterns is of increased research interest due to a broad range of useful applications, including analysis of transportation systems [30], human mobility [43], location-based services [59], spatiotemporal epidemics [3, 37, 38] and more. There are several technical problems in trajectory data mining that researchers and practitioners have focused on in recent years, including trajectory similarity [11], clustering [16], classification [4], prediction [57] and simplification [54].

A few comprehensive surveys on the topic can be found in Zheng [63], Alturi et al. [6], and Hamdi et al. [15]. In this research, we focus on the problem of constructing 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* [1, 8, 24, 35, 41, 61]. For consistency, we will use the term *pathlets* to denote these building blocks.

The Broader Impact. 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 [58], travel time prediction [17], personalized destination prediction [56], trajectory prediction [57], and trajectory compression [62] (see Appendix A for a supplementary discussion of these applications).

The State of the Art & Limitations. 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) [1, 20, 48]. A few works considered an integer programming formulation with constraints to solve the problem [8, 24]. Some works designed their pathlets based on a route "representativeness" criterion [35, 53]. Unfortunately, these existing works suffer some limitations. For example, Chen et al.

[1]Downtown Toronto maps taken from https://www.mapquest.com/

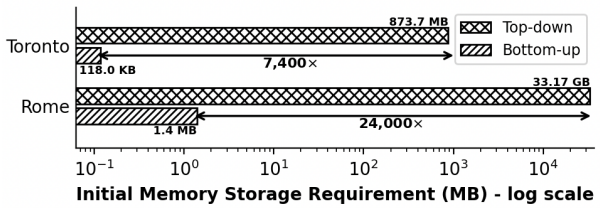

Figure 2: The memory required by top-down (existing) methods that use overlapping pathlets can be reduced by our proposed bottom-up solution that relies on edge-disjoint pathlets. Our approach demonstrates significant savings.

[8] assumes that the datasets used are noise-free. Zhou et al.'s [64] bag-of-segments method requires that trajectory segments are of fixed length. Van Krevald et al. [48] demands input trajectories to have the same start/endpoints. The cluster centroids in (sub)trajectory clustering methods [1, 20, 48, 51] do not necessarily reflect real roads in the road network. In addition, Wang et al. [53] demonstrated empirically that these clustering methods are computationally slow. In spite of runtime improvements, [53] 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 [20] 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, 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.

Our Approach & Contributions. To address these limitations, we propose a bottom-up approach for constructing a pathlet dictionary that complies with edge-disjoint pathlets (see Fig. 1) and reduces memory storage requirement. In Fig. 2 for instance, we illustrate how our proposed approach saves up to ~24K× less memory space than existing methods for storing the initial pathlets (see Experiment **(Q2)** for full details, with Appendix B presenting a more theoreotical proof). The key idea of our approach is to initialize unit-length pathlets & iteratively merge them to form longer, higher-order ones, while maximizing utility [2, 26]. Longer pathlets are preferred (over shorter ones) as they hold more spatiotemporal information, such as mobility patterns in trajectories [8]. A deep reinforcement learning method is proposed to approximate the utility function. A summary of our contributions is provided below:

- 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.

Symbol	Definition
τ and $\mathcal{T}$	A trajectory τ and a set of trajectories $\mathcal{T}$
$\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$	Road network with intersections $\mathcal{V}$ and segments $\mathcal{E}$
ρ and $\mathcal{P}$	A pathlet ρ and a pathlet set $\mathcal{P}$
$\mathcal{G}_p\langle\mathcal{V}_p,\mathcal{E}_p\rangle$	The pathlet graph representation of road network $\mathcal{G}$
$\Psi(\rho)$	The neighbors of pathlet ρ
$\Phi(\tau)$	The pathlet-based representation of a trajectory τ
$\Lambda(\rho)$	The trajectory traversal set of a pathlet ρ
$\mu(\tau)$	The trajectory representability of trajectory τ
L_{traj}	The trajectory loss
$\mathbb{S}$	The pathlet dictionary
ϕ	The avg # of pathlets representing each τ in $\mathcal{T}$

Table 1: Summary of notation used in this work

- 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 open-source our code to encourage reproducibility[2].

Paper Organization. The remainder of the paper is organized as follows. Section 2 presents preliminaries and a formal definition of the problem. Section 3 presents our methodology and the details of the proposed method. We describe the experimental setup, present the results, and make some insightful discussions in section 4. We review related work in section 5 and conclude in section 6.

2 PRELIMINARIES & PROBLEM DEFINITION

In this section, we briefly introduce some definitions and notations (see Table 1). Then we formally define the problem of interest.

2.1 Primary Definitions and Notations

Definition 2.1 (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: $\tau = \langle (x_1, y_1, t_1), ..., (x_{|\tau|}, y_{|\tau|}, t_{|\tau|}) \rangle$, where each x_i and y_i represents o's longitudinal and latitudinal coordinates at a specific time instance $t_i \in [0, T]$. We let $|\tau|$ be its length, or the # of time-enabled points for the trajectory of o. Moreover, we let *trajectory (data) set* $\mathcal{T}$ consist of all trajectories of all $o \in \mathbf{O}$: $\mathcal{T} = \bigcup_{\forall o \in \mathbf{O}} \mathcal{T}_o$, with $\mathcal{T}_o$ as the set of all o's trajectories.

[2] https://github.com/techGIAN/PathletRL

Definition 2.2 (Road Network). We denote by $\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$ the *road network* within map $\mathcal{M}$, where $\mathcal{V}$ and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ represents $\mathcal{G}$'s set of road intersections (nodes) and segments (edges) respectively.

Trajectory points (GPS traces) outside $\mathcal{M}$ are filtered out as a preprocessing step. The remaining trajectories also require to be map-matched (see Appendix C for details). With map-matched data, we can now formalize the fundamental building block in this work.

Definition 2.3 (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, s.t. 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 2.4 (Pathlet Length[3]). Denoted by ℓ, this represents pathlet ρ's 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 $\ell \leq k$, for some user-defined k. In this case, we say that $\mathcal{P}$ is a **k-order pathlet set**.

Definition 2.5 (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 $\mathcal{E}_p \subseteq \mathcal{E}$.

Fig. 1(a), for example, represents the pathlet graph representation of a small area in Toronto. In our work, we consider an initial pathlet graph where each pathlet has length $\ell = 1$.

Definition 2.6 (Pathlet Neighbors). Given a pathlet $\rho \in \mathcal{P}$, its neighbor pathlets, denoted by $\Psi(\rho)$, are all other pathlets $\rho' \in \mathcal{P}\setminus\{\rho\}$ who share the same start/endpoints with that of ρ:

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

For example, the grey pathlet in Fig. 1(b) has two neighbors: the orange & blue pathlets.

Definition 2.7 (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 $\Phi(\tau) = \{\rho^{(1)}, \rho^{(2)}, ..., \rho^{(|\mathcal{P}_{sub}|)}\}$, where $\rho^{(i)} \in \mathcal{P}_{sub}$ denotes the ith pathlet in the sequence that represents the pathlet-based representation for τ.

Based on this, it is also possible to define a trajectory's pathlet length, which we initially set before constructing the pathlet graph. Each trajectory $\tau \in \mathcal{T}$ has pathlet length equal to $\sum_{\forall\rho\in\Phi(\tau)} \ell(\rho)$, whose value remains static for the rest of our algorithm.

Definition 2.8 (Trajectory Traversal Set of a Pathlet). Let $\Lambda(\rho)$ be the set of all trajectories $\tau \in \mathcal{T}$ that *pass* or *traverse* pathlet $\rho \in \mathcal{P}$. This can also be written as $\Lambda(\rho) = \{\tau \,|\, \forall\tau \in \mathcal{T},\ \rho \in \Phi(\tau)\}$.

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 # 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.

[3]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.

2.2 Novel Trajectory Metrics

We can now introduce some novel metrics to allow us to more comprehensively evaluate our pathlets and pathlet dictionaries.

Definition 2.9 (Trajectory Representability[4]). The (trajectory) representability $\mu \in [0\%, 100\%]$ of a trajectory τ denotes the % of τ that can be represented using pathlets in pathlet set $\mathcal{P}$.

Clearly, the pathlet-based representation of τ is directly related to its representability, i.e., $\mu(\tau) = |\Phi(\tau)|/\sum_{\forall\rho\in\Phi(\tau)} \ell(\rho)$, for the unweighted case and $\mu(\tau) = \sum_{\forall\rho\in\Phi(\tau)} \omega(\rho)$, for the weighted version.

Definition 2.10 (Trajectory Loss). We define the *trajectory loss* L_{traj} to be the # of trajectories $\forall\tau \in \mathcal{T}$ that have representability value $\mu = 0\%$, i.e., $L_{traj} = |\{\tau|\tau \in \mathcal{T}, \mu(\tau) = 0\}|$. 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 %.

The relevance and impact of these two metrics will become clear as we go over the methodology in finer details.

2.3 Problem Definition

Before formalizing the problem, we first introduce the *pathlet dictionary* and some optimization definitions.

Definition 2.11 (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 Fig. 3 (the righthand boxes inside the orange & blue panels) for an illustrative example of a PD. We are interested in constructing a PD that aims to achieve one or a combination of the following:

- **(O1)** Minimal size of candidate pathlet set $\mathbb{S}$, or the candidate set with the least possible number of pathlets (i.e., $\min|\mathbb{S}|$)
- **(O2)** Minimal ϕ, or the avg # of pathlets representing each trajectory $\tau \in \mathcal{T}$ (i.e., $\min\phi = \min\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}|\Phi(\tau)|$)
- **(O3)** Minimal trajectory loss L_{traj} (i.e., $\min L_{traj}$)
- **(O4)** Maximal $\bar{\mu}$, or the average representability values of the remaining trajectories in $\mathcal{T}$ (i.e., $\max\bar{\mu} = \max\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\mu(\tau)$)

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

$$\min_{\sum_i\alpha_i=1}\left(\alpha_1|\mathbb{S}| + \alpha_2\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}|\Phi(\tau)| + \alpha_3 L_{traj} - \alpha_4\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\mu(\tau)\right) \quad (1)$$

where the α_i's are user-defined objective weights.

Problem 1 (Pathlet Dictionary Construction). Given a road network $\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$ of a specific map $\mathcal{M}$, a trajectory set $\mathcal{T}$, max pathlet length k, max trajectory loss M, and avg trajectory representability threshold $\hat{\mu}$, construct a k-order pathlet dictionary $\mathbb{S}$. The dictionary $\mathbb{S}$ consists of edge-disjoint pathlets with lengths of at most k, and achieves the max possible utility according to some utility function as depicted in Equation (1), such that $L_{traj} < M$ and $\bar{\mu} \geq \hat{\mu}$.

3 METHODOLOGY

We now describe our methods to address the problem of interest (see Fig. 3 for its architecture). In particular, we describe the components of the proposed PathletRL model (**Pathlet** dictionary construction using trajectories with **R**einforcement **L**earning). There are two

[4]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.

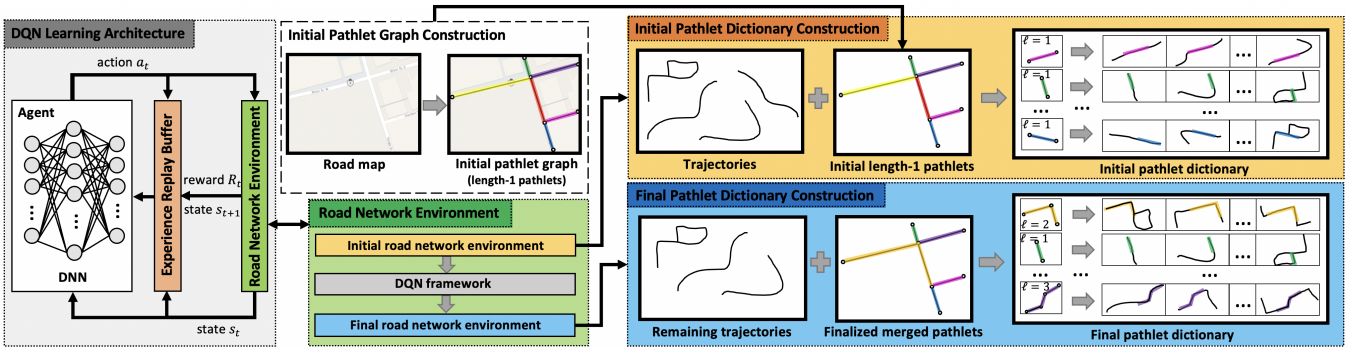

Figure 3: The overall architecture (including the constructed PDs) of our proposed PathletRL model

main components: (1) the method responsible for extracting the candidate pathlet sets through a merging-based process, and (2) a deep reinforcement learning-based architecture for approximating the utility function of the merging process of (1).

3.1 Extracting Candidate Pathlets

In this section, we describe the algorithmic details for merging our edge-disjoint pathlets. The high-level idea of the algorithm is based on the theory of maximal utility [2, 26], i.e., iteratively merging (neighboring) pathlets until this brings forth little to no improvement on the *utility* (details of the utility are given later). The algorithm takes in as input a road network $\mathcal{G}$, a trajectory set $\mathcal{T}$ operating within $\mathcal{G}$, the max threshold for the trajectory loss M, the trajectory representability threshold $\hat{\mu}$, and a positive integer k denoting the desired k-order pathlet graph. As output, it returns a pathlet dictionary (PD) that holds pathlet information as described in Definition 2.11. The extracted PD aims to satisfy the four objectives **(O1)-(O4)**. See Algorithm 1 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 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 $\ell > k$ (i.e., merging the two pathlets ρ and ρ^* would violate the k-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 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 avg 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 the next section.

3.2 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 [28, 44]. 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 gray panel of Fig. 3).

The Environment. RL models are designed for an agent to learn the most optimal actions, commonly in games [28, 46]. In this context, we consider the entire pathlet graph $\mathcal{G}_p$ to be the environment model; it is where our deep RL algorithm will be operating on.

Algorithm 1: Candidate Pathlet Set Extraction Algorithm

Input : The road network $\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$, the trajectory set $\mathcal{T}$, integer k, the maximum trajectory loss M and the average trajectory representability threshold $\hat{\mu}$.
Output : The k-order candidate pathlet set $\mathbb{S}$ of merged pathlets with a trajectory loss not exceeding M

/* Initialization */
$\mathcal{G}_p\langle\mathcal{V}_p,\mathcal{E}_p\rangle \leftarrow$ ExtractPathletGraph$(\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle)$
$\ell \leftarrow 1$ // Size of the initial length 1 pathlets
$\mathcal{T}^* \leftarrow \mathcal{T}$; $\mathbb{S} \leftarrow \varnothing$
$\phi \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}|\Phi(\tau)|$; $\bar{\mu} \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}\mu(\tau)$
/* Setup traj loss and utility dictionaries */
$T_D \leftarrow$ Dict(); $U_D \leftarrow$ Dict()
$\rho \leftarrow$ 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 */
while $\mathcal{E}_p \neq \mathbb{S}$ **or sum**$(T_D.\text{Values}()) < M$ **or** $\bar{\mu} \geq \hat{\mu}$ **do**
  /* Initially set $\mathcal{G}_p$'s utility associated to the curr pathlet ρ to be 0 */
  $U_D[\rho] = 0$
  /* For each of ρ's unprocessed neighbors */
  foreach $\hat{\rho}$ **in** $\Psi(\rho) \setminus \mathbb{S}$ **do**
    $U_D[\hat{\rho}] \leftarrow$ ComputeUtil(Merge$(\rho,\hat{\rho})$)
    $T_D[\hat{\rho}] \leftarrow$ GetAllTrajLost (Merge$(\rho,\hat{\rho}), \mathcal{T}^*$)
  /* Find the one with the highest utility */
  $\rho^* \leftarrow \underset{\rho\in key}{\text{argmax}}\, U_D[key]$
  if $\rho^* = \rho$ **or** $\ell > k$ **then** // Merge not necessary
    $\mathbb{S} \leftarrow \mathbb{S} \cup \{\rho\}$
    $\rho \leftarrow$ Rand$(\mathcal{E}_p \setminus \mathbb{S})$ // Pick new pathlet
    if $\rho = \varnothing$ **then** // All pathlets processed
      break
    $\ell \leftarrow 1$ // Reset pathlet length
  else // Merge recommended
    $\rho_{merged} \leftarrow$ Merge(ρ,ρ^*)
    $\mathcal{E}_p \leftarrow (\mathcal{E}_p \setminus \{\rho,\rho^*\}) \cup \{\rho_{merged}\}$
    $\mathcal{T}^* \leftarrow \mathcal{T}^* \setminus T_D[\rho^*]$
    $\phi \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}|\Phi(\tau)|$; $\bar{\mu} \leftarrow \frac{1}{|\mathcal{T}^*|}\sum_{\tau\in\mathcal{T}^*}\mu(\tau)$
    $\rho \leftarrow \rho_{merged}$; $\ell \leftarrow \ell + 1$
return $\mathbb{S}$

The Agent. RL is often designed for training robotic agents, or some AI [19]. 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) [49], 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 # of pathlets in the current pathlet graph, S_2 denotes the average # 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 $\mathcal{A} = \{\text{keep}, \text{merge}\}$. 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 1 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:

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

The Reward Function. We formulate our reward function R based on the optimization equation defined in Equation (1):

$$\max_{a_t} \mathbb{E}\left[\left(-\alpha_1|\mathbb{S}| - \alpha_2\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}|\Phi(\tau)| - \alpha_3 L_{traj} + \alpha_4\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\mu(\tau)\right)\right] \quad (2)$$

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 $\{r_t\}_{t=0}^{T}$:

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

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 (2). 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 max possible future expected reward in the environment for state-action pair (s_t, a_t):

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

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:

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha_{lr}\left[\gamma\max_{a_{t+1}} Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)\right] \quad (3)$$

where α_{lr} is the learning rate. In fact, this Q-learning paradigm has seen significant success in the reinforcement learning community [47]. 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 D for other reinforcement learning 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* [13]. 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 # of trajectories that cannot be represented by the pathlets in the pathlet set.

4 EVALUATION

In this section, we present the details of our experimental setup for evaluating our proposed method. We aim to analyze and evaluate our models based on the following research questions:

(Q1) How does PathletRL compare with the SotA methods, in terms of the quality of the extracted PDs?
(Q2) How much memory does the bottom-up approach save compared to top-down methods?
(Q3) How much improvement and how much more effective is our PathletRL model against its ablation variations?
(Q4) What is the distribution of pathlet lengths in the obtained dictionary in our PathletRL model?
(Q5) How effective is the constructed PD in reconstructing the original trajectories?
(Q6) What is the sensitivity of the user-defined parameters [α_1, α_2, α_3, α_4] in the performance of our PathletRL model?

4.1 Datasets

We utilize two datasets that each depict a different map scenario (see Appendix E for its complete statistics, and Appendix F for a brief discussion about the data's privacy concerns). We used real world maps of two metropolitan cities, Toronto[5] and Rome through the OpenStreetMaps[6]. A realistic synthetic vehicular mobility datasets for the Toronto map was generated using the Sumo mobility app simulator[7] (3.7 hrs). Moreover, larger-scale, real-world taxi cab trajectories (first week of February 2014) were taken from Crawdad [7], 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.

4.2 Experimental Parameters

Refer to Appendix G for full details of the implementation. To implement the RL architecture, we used a deep neural network that consists of the following parameters. 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. We also use a 0.2 dropout in the network, together with the Huber loss function. More specific to the DQN's parameters, we have $m = 5$ 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. Our agent also uses a discount factor $\gamma = 0.99$. Moreover, we use $k = 10$ for the k-order candidate set, M = 25% maximum trajectory loss and $\hat{\mu}$ = 80% average representability threshold. We also set $\alpha_i = ¼$, $\forall i$, which denotes equal importance for each of the four objectives as depicted in Equation (2).

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

Table 2: Features of the proposed PathletRL models, alongisde its ablation baselines.

4.3 Baselines

We introduce the following baselines (see Appendix H for a discussion on baseline choice). The first two are SotA (top-down-based) baselines, while the last three are ablation versions of our proposed model. Table 2 depicts the features withheld in each ablation variation to demonstrate the feature's importance and effectiveness.

Chen et al. [8]. 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, we immediately discard trajectory τ.

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

4.4 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 avg # of pathlets that represent each trajectory [↓]
(3) L_{traj}, the # of trajectories discarded (expressed in %) [↓]
(4) $\bar{\mu}$, the average representability across the remaining trajectories (expressed in %) [↑]

[5]https://www.toronto.ca/city-government/data-research-maps/open-data/
[6]https://www.openstreetmap.org/
[7]Simulation of Urban MObility: https://www.eclipse.org/sumo/

		Baselines		Null	PathletRL			
		[8]	[1]	Sgt	Rnd	Nr	Unw	(ours)
Toronto	[↓] $\lvert\mathbb{S}\rvert$	13,886	7,982	2,563	2,454	1,896	1,801	**1,743**
	[↓] ϕ	7.02	5.97	4.76	3.77	**2.89**	3.98	3.75
	[↓] L_{traj}	N/A	N/A	0%	19.7%	17.6%	**15.1%**	15.2%
	[↑] $\bar{\mu}$	N/A	N/A	100%	79.9%	N/A	80.0%	**83.9%**
Rome	[↓] $\lvert\mathbb{S}\rvert$	59,396	31,017	15,465	9,718	7,003	5,804	**5,291**
	[↓] ϕ	202.91	188.33	230.15	173.04	158.18	146.39	**139.89**
	[↓] L_{traj}	N/A	N/A	0%	24.9%	21.1%	22.9%	**20.4%**
	[↑] $\bar{\mu}$	N/A	N/A	100%	82.7%	N/A	**86.2%**	85.6%

Table 3: Numerical results showing the attributes of the pathlet dictionaries extracted by each method for each dataset.

Note that the third and fourth metrics above do not apply to Chen et al.'s [8] 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.

4.5 Results and Discussion

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

(Q1) Quality of the Extracted PDs. We evaluate the pathlet dictionary extracted by our PathletRL algorithm against the PDs extracted by SotA baselines. See Table 3 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). Although the nature of the pathlet definition and the approaches are not necessarily the same, the algorithms of Chen et al. [8] 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, $\lvert\mathbb{S}\rvert$, 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 [8] 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 [8] and Agarwal et al.'s [1] PDs are larger than the initial # 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 $\lvert\mathbb{S}\rvert$, 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

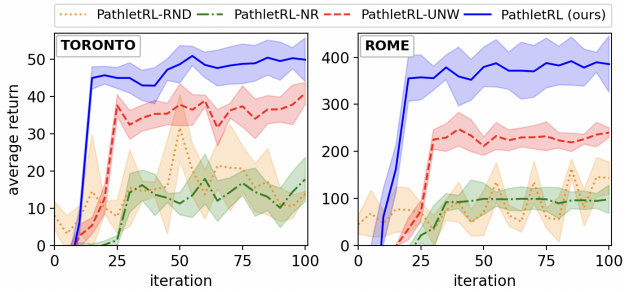

Figure 4: Performance evaluation of proposed and ablation PathletRL models, measured using the average return metric of $m = 5$ episodes across $n = 100$ iterations (run 10 times)

Toronto (Rome) dataset. Meanwhile, Chen et al.'s [8] 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 $\lvert\mathbb{S}\rvert$ 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.

(Q2) Memory Efficiency. Fig. 2 (in log scale) demonstrates how much more memory-efficient PathletRL is compared to the baselines [1, 8]. As can be seen in Fig. 2, 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 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.

(Q3) Ablation Study. Next, we perform an ablation experiment to see how well our proposed PathletRL model performs. Fig. 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 – 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

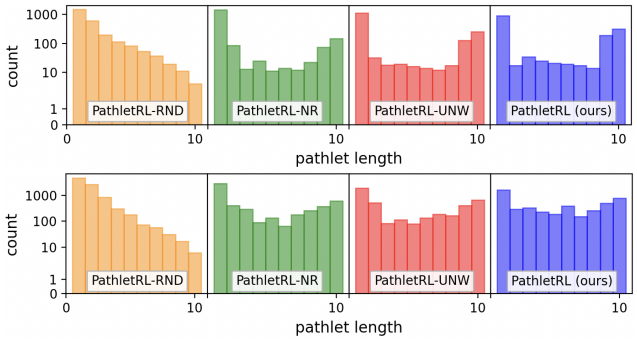

Figure 5: The pathlet length distribution of pathlet dictionaries obtained by PathletRL model and its ablation versions on the Toronto (top) and Rome (bottom) datasets.

Table 3 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. 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.

(Q4) Pathlet Length Distribution. We also analyze the length distribution of the pathlets in our dictionaries. The trend is similar for both datasets (Fig. 5 shows the pathlet length distribution, with the y-axis in log scale). The PathletRL-Rnd has a decreasing # for longer pathlets, which is intuitive as the random policy blindly keeps & 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 3. Meanwhile, we can still observe 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

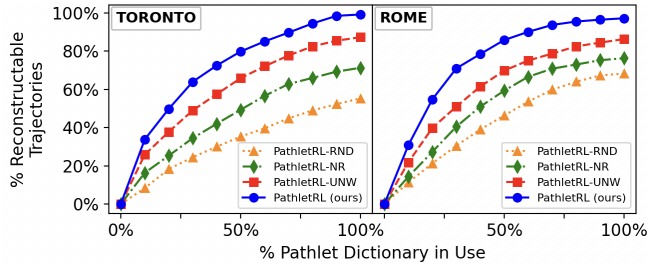

Figure 6: The % of evaluation trajectories reconstructable from a sample set taken from extracted pathlet dictionaries.

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.

(Q5) Partial Trajectory Reconstruction. See Fig. 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 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.

(Q6) 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[8]. There are four stacked plots, as seen in Fig. 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

[8]**Note**: the goal here is not to find an optimal value for α_i, but intends on showing the trend of how $|\mathbb{S}|, \phi, L_{traj}, \bar{\mu}$ changes with varying values of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ respectively.

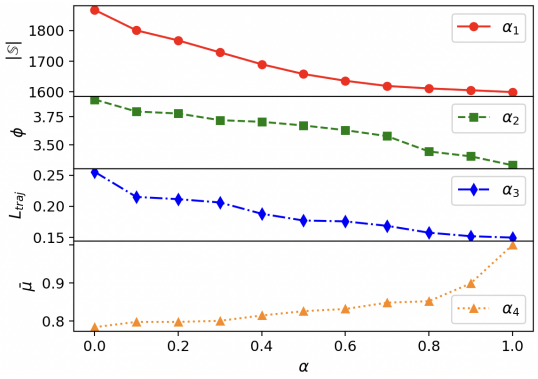

Figure 7: Parameter sensitivity experiment: the impact of α's on the quality of PathletRL's pathlet dictionary

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.

5 RELATED WORK

Our research is related to (i) *trajectory data mining*, (ii) *pathlet mining*, (iii) *subtrajectory clustering*, and (iv) *deep learning for spatiotemporal data*. We cover below some of the most significant efforts relevant to our work.

Trajectory Data Mining. Trajectory data mining has been an active research direction for a long time [6, 15, 63]. This high interest can largely be attributed to the rapid development and prominence of geospatial technologies [10], location-based smart devices [40], abundance of GPS-based applications [36], and generation of massive trajectory datasets [12]. The focus is on popular technical problems, including trajectory similarity [11] and trajectory clustering [16]. Modeling trajectory data using graphs to address complex trajectory mining tasks has also been a popular approach in recent years. Example tasks include, but are not limited to similarity search [11], recovery [9], node centrality computation [39], mining spatiotemporal interactions [42, 43], learning semantic relationships of geographic areas [27] and recommendations [50, 55].

Pathlet Mining. Pathlet mining focuses on finding patterns and extracting knowledge of a small set of basic building blocks that can represent a wide range of trajectories. One of the original works in this direction is the work of Chen et al. [8], 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. Zhou et al. [64] designed a bag of segments representation for motion trajectories and showed that the method is compact and expressive on trajectory classification and similarity search tasks. Panagiotakis et al. [35] proposed a method for finding representative subtrajectories through global voting, segmentation and sampling methods. Wang et al. [53] solves the problem of finding top k representative routes that cover as many trajectories as possible under some budget constraint and distance threshold; they also proposed three near-optimal solutions that can address the problem efficiently.

Subtrajectory Clustering. Pathlet mining is commonly framed as a subtrajectory clustering problem. Lee et al. [20] 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. Van Krevald et al. [48] designed a novel measure for mining median trajectories, similar to the method of route representative discovery, that serves as the cluster centroid of (sub)trajectories. Agarwal et al. [1] gains motivation in their pathlet cover method from the popular set cover algorithm. 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.

Deep Learning for Spatiotemporal Data. In recent years, deep learning methods have been proposed for learning representations of spatiotemporal data [52]. Of particular interest are deep reinforcement learning based methods that are often evaluated on agents playing a specific game [46]. These methods have successfully been adapted and shown promise in addressing several complex trajectory-related problems, such as route planning [14], trajectory simplification [54], and adaptive vehicle navigation problem [5].

6 CONCLUSIONS

Constructing trajectory pathlet dictionaries, or small sets of building blocks 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 ~24K× 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.

REFERENCES

[1] P. K. Agarwal, K. Fox, K. Munagala, A. Nath, J. Pan, and E. Taylor. 2018. Subtrajectory Clustering: Models and Algorithms. In *Proc. of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Sys. (PODS '18)*. 75–87.
[2] F. Aleskerov, D. Bouyssou, and B. Monjardet. 2007. *Utility Maximization, Choice and Preference.* Springer Berlin Heidelberg, Berlin, Heidelberg.
[3] G. Alix, N. Yanin, T. Pechlivanoglou, J. Li, F. Heidari, and M. Papagelis. 2022. A Mobility-based Recommendation System for Mitigating the Risk of Infection during Epidemics. In *2022 23rd IEEE Intl. Conf. on Mobile Data Mgt. (MDM)*. 292–5.
[4] M. Alsaeed, A. Agrawal, and M. Papagelis. 2023. Trajectory-User Linking using Higher-order Mobility Flow Representations. In *2023 24th IEEE International Conference on Mobile Data Management (MDM)*. 158–167.
[5] F. Arasteh, S. SheikhGarGar, and M. Papagelis. 2022. Network-Aware Multi-Agent Reinforcement Learning for the Vehicle Navigation Problem. In *Proc. of the 30th Intl. Conf. on Adv. in Geographic Info. Sys. (ACM SIGSPATIAL '22)*. Article 69.
[6] G. Atluri, A. Karpatne, and V. Kumar. Spatio-Temporal Data Mining: A Survey of Problems and Methods. *ACM Comp. Surveys* 51, 4, Article 83 (Aug 2018).
[7] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi. 2014. CRAWDAD Roma/taxi dataset. https://crawdad.org/roma/taxi/20140717.
[8] C. Chen, H. Su, Q. Huang, L. Zhang, and L. Guibas. 2013. Pathlet Learning for Compressing and Planning Trajectories. In *Proc. of the 21st ACM SIGSPATIAL Intl. Conf. on Adv. in Geographic Info. Sys. (ACM SIGSPATIAL'13)*. 392—-5.
[9] Y. Chen, H. Zhang, W. Sun, and B. Zheng. 2022. RNTrajRec: Road Network Enhanced Trajectory Recovery with Spatial-Temporal Transformer.

[10] A. Datta. Seizing The Future: Geospatial Industry Technology Trends and Directions. *GWPrime* (Feb 2022).

[11] Z. Fang, Y. Du, X. Zhu, D. Hu, L. Chen, Y. Gao, and C. S. Jensen. 2022. Spatio-Temporal Trajectory Similarity Learning in Road Networks. In *Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining (ACM SIGKDD '22)*. 347–56.

[12] A. Faraji, J. Li, G. Alix, M. Alsaeed, N. Yanin, A. Nadiri, and M. Papagelis. 2023. Point2Hex: Higher-order Mobility Flow Data and Resources. In *Proc. of the 31st Intl. Conf. on Adv. in Geographic Info. Sys. (SIGSPATIAL '23)*.

[13] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney. 2020. Revisiting Fundamentals of Experience Replay. In *Proc. of the 37th Intl. Conf. on Machine Learning (ICML'20)*. JMLR, Article 287.

[14] Y. Geng, E. Liu, R. Wang, and Y. Liu. Deep Reinforcement Learning Based Dynamic Route Planning for Minimizing Travel Time. *CoRR* (2020).

[15] A. Hamdi, K. Shaban, A. Erradi, A. Mohamed, S. K. Rumi, and F. D. Salim. Spatiotemporal data mining: a survey on challenges and open problems. *Artificial Intelligence Review* 55, 2 (Feb 2022), 1441–1488.

[16] N. Han, S. Qiao, K. Yue, J. Huang, Q. He, T. Tang, F. Huang, C. He, and C.-A. Yuan. Algorithms for Trajectory Points Clustering in Location-Based Social Networks. *ACM Trans. Intell. Syst. Technol.* 13, 3, Article 43 (mar 2022).

[17] Q. Han, Y. Lei, L. Zeng, G. He, L. Ye, and L. Qi. 2021. Research on Travel Time Prediction of Multiple Bus Trips Based on MDARNN. In *2021 IEEE Intl. Intelligent Transportation Sys. Conf. (ITSC)*. 3718–3725.

[18] Z. He, L. Tao, Z. Xie, and C. Xu. Discovering spatial interaction patterns of near repeat crime by spatial association rules mining. *Sci. Reports* 10, 1 (Oct 2020).

[19] J. Kober and J. Peters. 2014. *Reinforcement Learning in Robotics: A Survey*. Springer Intl. Publishing, Cham, 9–67.

[20] J.-G. Lee, J. Han, and K.-Y. Whang. 2007. Trajectory Clustering: A Partition-and-Group Framework. In *Proc. of the 2007 ACM SIGMOD Intl. Conf. on Mgt. of Data (ACM SIGMOD '07)*. 593–604.

[21] L. Li, R. Jiang, Z. He, X. M. Chen, and X. Zhou. Trajectory data-based traffic flow studies: A revisit. *Transportation Research Part C: Emerging Technologies* 114 (2020), 225–240.

[22] M. Li, P. Tong, M. Li, Z. Jin, J. Huang, and X.-S. Hua. Traffic Flow Prediction with Vehicle Trajectories. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 1 (May 2021), 294–302.

[23] T. Li, J. Gao, and X. Peng. 2021. Deep Learning for Spatiotemporal Modeling of Urbanization. In *Proc. of the 35th Conf. on Neural Info. Proc. Sys. (NeurIPS 2021)*.

[24] Y. Li, D. Gunopulos, C. Lu, and L. J. Guibas. Personalized Travel Time Prediction Using a Small Number of Probe Vehicles. *ACM Trans. Spatial Algorithms Syst.* 5, 1, Article 4 (may 2019).

[25] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. 2009. Map-Matching for Low-Sampling-Rate GPS Trajectories. In *Proc. of the 17th ACM SIGSPATIAL Intl. Conf. on Adv. in Geographic Info. Sys.* (Seattle, Washington) *(GIS '09)*. Assoc. for Comp. Machinery, New York, NY, USA, 352–361.

[26] K. McCormick. An Essay on the Origin of the Rational Utility Maximization Hypothesis and a Suggested Modification. *Eastern Eco. Journal* 23 ('97), 17–30.

[27] S. Mehmood and M. Papagelis. 2020. Learning Semantic Relationships of Geographical Areas based on Trajectories. In *2020 21st IEEE Intl. Conf. on Mobile Data Mgt. (MDM)*. 109–118.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. Playing Atari With Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*.

[29] J. Muckell, J.-H. Hwang, C. T. Lawson, and S. S. Ravi. 2010. Algorithms for Compressing GPS Trajectory Data: An Empirical Evaluation. In *Proc. of the 18th SIGSPATIAL Intl. Conf. on Adv. in Geographic Info. Sys. (ACM GIS '10)*. 402–405.

[30] A. Nematichari, T. Pechlivanoglou, and M. Papagelis. 2022. Evaluating and Forecasting the Operational Performance of Road Intersections. In *Proc. of the 30th Intl. Conf. on Adv. in Geographic Info. Sys. (ACM SIGSPATIAL '22)*. Article 31.

[31] M. E. J. Newman. 2018. *Networks* (second edition ed.). Oxford University Press, Oxford, United Kingdom; New York, NY, United States of America.

[32] P. Newson and J. Krumm. 2009. Hidden Markov Map Matching through Noise and Sparseness. In *Proc. of the 17th ACM SIGSPATIAL Intl. Conf. on Adv. in Geographic Info. Sys. (GIS '09)*. 336–343.

[33] M. Nyhan, S. Grauwin, R. Britter, B. Misstear, A. McNabola, F. Laden, S. R. H. Barrett, and C. Ratti. "Exposure Track"—The Impact of Mobile-Device-Based Mobility Patterns on Quantifying Population Exposure to Air Pollution. *Environmental Science & Technology* 50, 17 (2016), 9671–9681.

[34] P. J. Olver and C. Shakiban. 2018. *Applied Linear Algebra* (2nd ed. 2018 ed.). Springer International Publishing: Imprint: Springer, Cham.

[35] C. Panagiotakis, N. Pelekis, I. Kopanakis, E. Ramasso, and Y. Theodoridis. Segmentation and Sampling of Moving Object Trajectories Based on Representativeness. *IEEE Trans. on Knowledge and Data Eng.* 24, 7 (2012), 1328–1343.

[36] M. K. Pandey, P. Srivastava, and G. Petropoulos. 2021. Chapter 21 - Future pathway for research and emerging applications in GPS/GNSS. In *GPS and GNSS Technology in Geosciences*. Elsevier, 429–438.

[37] T. Pechlivanoglou, G. Alix, N. Yanin, J. Li, F. Heidari, and M. Papagelis. 2022. Microscopic Modeling of Spatiotemporal Epidemic Dynamics. In *Proc. of the 3rd ACM SIGSPATIAL Intl. Workshop on Spatial Comp. for Epidemiology (SpatialEpi '22)*. 11–21.

[38] T. Pechlivanoglou, J. Li, J. Sun, F. Heidari, and M. Papagelis. Epidemic Spreading in Trajectory Networks. *Big Data Research* 27 (2022).

[39] T. Pechlivanoglou and M. Papagelis. 2018. Fast and Accurate Mining of Node Importance in Trajectory Networks. In *2018 IEEE Intl. Conf. on Big Data (Big Data)*. 781–790.

[40] T. Ruth. Why Location Services and IoT are Leading the 5G Trends of 2022. *Quuppa* (Sep 2022).

[41] S. Sankararaman, P. K. Agarwal, T. Mølhave, J. Pan, and A. P. Boedihardjo. 2013. Model-Driven Matching and Segmentation of Trajectories. In *Proc. of the 21st ACM SIGSPATIAL Intl. Conf. on Adv. in Geographic Info. Sys.* (Orlando, Florida) *(SIGSPATIAL'13)*. Assoc. for Comp. Machinery, New York, NY, USA, 234–43.

[42] A. Sawas, A. Abuolaim, M. Afifi, and M. Papagelis. 2018. Tensor Methods for Group Pattern Discovery of Pedestrian Trajectories. In *19th IEEE Intl. Conf. on Mobile Data Mgt. (MDM)*. 76–85.

[43] A. Sawas, A. Abuolaim, M. Afifi, and M. Papagelis. A versatile computational framework for group pattern mining of pedestrian trajectories. *GeoInformatica* 23, 4 (Oct 2019), 501–531.

[44] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[45] A. Strzelecki. The Apple Mobility Trends Data in Human Mobility Patterns during Restrictions and Prediction of COVID-19: A Systematic Review and Meta-Analysis. *Healthcare* 10, 12 (2022).

[46] S.-H. Sun, T.-L. Wu, and J. J. Lim. 2020. Program Guided Agent. In *Intl. Conf. on Learning Representations*.

[47] R. S. Sutton and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

[48] M. van Kreveld and L. Wiratma. 2011. Median Trajectories Using Well-Visited Regions and Shortest Paths. In *Proc. of the 19th ACM SIGSPATIAL Intl. Conf. on Adv. in Geographic Info. Sys. (GIS '11)*. 241–250.

[49] M. van Otterlo and M. Wiering. 2012. *Reinforcement Learning and Markov Decision Processes*. Springer Berlin Heidelberg, Berlin, Heidelberg, 3–42.

[50] J. Wang, N. Wu, and W. Zhao. Personalized Route Recommendation With Neural Network Enhanced Search Algorithm. *IEEE Transactions on Knowledge & Data Engineering* 34, 12 (Dec 2022), 5910–5924.

[51] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, and X. Qin. Fast Large-Scale Trajectory Clustering. *Proc. VLDB Endow.* 13, 1 (Sep 2019), 29–42.

[52] S. Wang, J. Cao, and P. Yu. Deep Learning for Spatio-Temporal Data Mining: A Survey. *IEEE Trans. on Knowledge & Data Eng.* 34, 08 (Aug 2022), 3681–700.

[53] T. Wang, S. Huang, Z. Bao, J. S. Culpepper, and R. Arablouei. 2022. Representative Routes Discovery from Massive Trajectories. In *Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Assoc. for Comp. Machinery, New York, NY, USA, 4059–69.

[54] Z. Wang, C. Long, and G. Cong. 2021. Trajectory Simplification with Reinforcement Learning. In *2021 IEEE 37th Intl. Conf. on Data Eng. (ICDE)*. 684–695.

[55] Z. Wang, Y. Zhu, Q. Zhang, H. Liu, C. Wang, and T. Liu. Graph-Enhanced Spatial-Temporal Network for Next POI Recommendation. *ACM Trans. Knowl. Discov. Data* 16, 6, Article 104 (Jul 2022).

[56] J. Xu, J. Zhao, R. Zhou, C. Liu, P. Zhao, and L. Zhao. Predicting Destinations by a Deep Learning based Approach. *IEEE Trans. on Knowledge and Data Eng.* 33, 2 (2021), 651–666.

[57] H. Xue, B. P. Voutharoja, and F. D. Salim. 2022. Leveraging Language Foundation Models for Human Mobility Forecasting. In *Proc. of the 30th Intl. Conf. on Adv. in Geographic Info. Sys. (ACM SIGSPATIAL '22)*. Article 90.

[58] R. K. Yadav, G. Kishor, Himanshu, and K. Kashyap. 2020. Comparative Analysis of Route Planning Algorithms on Road Networks. In *2020 5th Intl. Conf. on Comm. and Electronics Sys. (ICCES)*. 401–406.

[59] M. Yassin and E. Rachid. 2015. A survey of positioning techniques and location based services in wireless networks. In *2015 IEEE Intl. Conf. on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*. 1–5.

[60] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. 2010. An Interactive-Voting Based Map Matching Algorithm. In *2010 11th Intl. Conf. on Mobile Data Mgt.* 43–52.

[61] J. Zhao, J. Xu, R. Zhou, P. Zhao, C. Liu, and F. Zhu. 2018. On Prediction of User Destination by Sub-Trajectory Understanding: A Deep Learning Based Approach. In *Proc. of the 27th ACM Intl. Conf. on Info. and Knowledge Mgt. (ACM CIKM 2018)*. 1413–1422.

[62] Y. Zhao, S. Shang, Y. Wang, B. Zheng, Q. V. H. Nguyen, and K. Zheng. 2018. REST: A Reference-Based Framework for Spatio-Temporal Trajectory Compression. In *Proc. of the 24th ACM SIGKDD Intl. Conf. on Knowledge Discovery Data Mining (ACM SIGKDD '18)*. 2797–2806.

[63] Y. Zheng. Trajectory Data Mining: An Overview. *ACM Trans. Intell. Syst. Technol.* 6, 3, Article 29 (May 2015).

[64] Y. Zhou and T. S. Huang. 2008. 'Bag of segments' for motion trajectory analysis. In *2008 15th IEEE Intl. Conf. on Image Processing*. 757–760.

APPENDIX

A PATHLET DICTIONARY'S APPLICATIONS

Trajectory Compression. Compression is the process of reducing the size of a trajectory while preserving its important spatiotemporal features [29]; 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, we are 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.

Route Planning. While navigation services and route planning apps such as Google Maps[9] and Apple Maps[10] 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.

Trajectory Prediction. Prediction and forecasting of trajectories is important in several domains including transportation [21], urban planning [23], and healthcare [37, 45]. 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.

Anomaly Detection. Spotting outliers has been an active research direction. 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.

Trajectory Similarity Search. The trajectory similarity search task is an important problem of interest due to its numerous practical applications in domains such as traffic analysis [22], public safety [18], and environmental conservation [33]. With pathlets, the similarity of two trajectories can be calculated based on the cosine similarity (or some other similarity metric) of their pathlet-based representations.

B SPACE COMPLEXITY ANALYSIS

To analyze the space complexity expressed in terms of the n number of road segments on the road network, we can simply provide a lower bound on this number. First, we consider the analysis for top-down approaches. The following two facts would be useful in this proof (we will bypass the proofs of these facts as they are trivial; refer to linear algebra books for their complete proofs [34]).

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 $m \times 1$. Then the sum of all entries of A is:

$$\sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} = \mathbf{x}^\top A\mathbf{x} \tag{4}$$

Fact 2. For an $m \times m$ square symmetric matrix A (i.e., $A = A^\top$) with λ_{min} as its smallest eigenvalue, then the quadratic form $\mathbf{x}^\top A\mathbf{x}$ is lower bounded by the squared L_2-norm of $\mathbf{x}$, for all $\forall \mathbf{x} \in \mathbb{R}^{m\times 1}$:

$$\mathbf{x}^\top A\mathbf{x} \geq \lambda_{min}||\mathbf{x}||_2^2 \tag{5}$$

In other words, $\mathbf{x}^\top A\mathbf{x} \in \Omega(||\mathbf{x}||_2^2)$

Space Complexity Analysis. So now we focus firstly 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 [31]. Assuming the absence of self-loops, 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 $\dim(A) \times 1$ column of ones (the sum of entries is in line with Fact 1 above in Equation (4)). 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:

$$\sum_{\substack{\ell\in\mathbb{Z}^+ \\ \mathbf{x}^\top Q_\ell \mathbf{x}\neq 0}} \frac{1}{2}\mathbf{x}^\top Q_\ell \mathbf{x} \tag{6}$$

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 (i.e., $\dim(A) = n$). Thus, in Equation (6), (assuming we have this least possible number of nodes) then 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 the summation of Equation (6) is lower-bounded by $\Omega(||\mathbf{x}||_2^2) = \Omega(n)$ as a consequence of the Fact 2 in Equation (5). 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: $\left(\sqrt{1^2 + 1^2 + ... + 1^2}\right)^2 = \left(\sqrt{n}\right)^2 = n$. 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 (6)[11]; thus the lower bound for the number of pathlets of the top-down scheme is $n \times \Omega(n) = \Omega(n^2)$.

On the other hand, 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

[9] http://maps.google.com/
[10] https://www.apple.com/maps/

[11] 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

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, and is therefore more space efficient compared to the top-down approaches the requires at least $\Omega(n^2)$ amount of memory space. □

C MAP-MATCHING

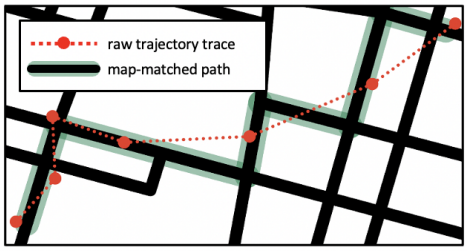

Figure 8: An example of the map-matching procedure.

Map-matching is a common task that identifies the path on the road an object has taken given a sequence of GPS locations [32] (Fig. 8 provides an illustrative example). Ideally, we prefer highly accurate map-matched data from GPS trajectory traces; however, this task is itself involved and is outside the paper's scope. Thus, we rely on existing methods [25, 60] to handle map-matching.

D 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.

E DATASET STATISTICS

	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

Table 4: Dataset attributes

F PRIVACY AND ETHICS

Our proposed methodology makes use of real-world trajectory datasets in the experiments that could potentially raise concerns about the individual (vehicle taxi cab) privacy and the potential for re-identification of such individuals. Therefore, to ensure protection of privacy and ethical considerations, all datasets used in our evaluation have been anonymized. Such datasets are derived from publicly-available sources, are publicly available, and are free of use for the intention of research purposes as outlined by their curators. In addition, all terms and conditions of use have been followed, with proper attribution and citation of works.

G IMPLEMENTATION DETAILS

Machine Specifications. The models have been implemented using Python 3.10 and an Intel Core i7-9700, 3.00GHz CPU, 62GB RAM and a GeForce RTX 2800 8GB GPU.

The RL Implementation. We implemented our deep reinforcement learning architecture using the `tf-agents`[12] package, a Tensorflow-based library designed specifically for implementing RL methods.

H THE CHOICE OF BASELINES

While there have been a number of considerable baseline methods [35, 53, 64] that can be used to compare PathletRL with, there are some reasons we decided to not do so. Firstly, Wang et al. [53] focuses on route representativeness discovery which is a completely different task than ours. Theirs, including Panagiotakis et al. [35] use a representativeness criterion that is not applicable to our case. Zhou et al. [64] moreover focuses on motion analysis which is a substantially different setting and task. This is in contrast to Chen et al. [8] 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. [8]. We revisit the problem with a RL-based approach and consider additional constraints (such as edge-disjointness in pathlets and the k-order constraint).

[12] https://www.tensorflow.org/agents