LightPath: Lightweight and Scalable Path Representation Learning

Sean Bin Yang
Aalborg University, Denmark
seany@cs.aau.dk

Jilin Hu
East China Normal University, China
Aalborg University, Denmark
jlhu@dase.ecnu.edu.cn

Chenjuan Guo
East China Normal University, China
Aalborg University, Denmark
cjguo@dase.ecnu.edu.cn

Bin Yang
East China Normal University, China
Aalborg University, Denmark
byang@dase.ecnu.edu.cn

Christian S. Jensen
Aalborg University, Denmark
csj@cs.aau.dk

ABSTRACT

Movement paths are used widely in intelligent transportation and smart city applications. To serve such applications, path representation learning aims to provide compact representations of paths that enable efficient and accurate operations when used for different downstream tasks such as path ranking and travel cost estimation. In many cases, it is attractive that the path representation learning is lightweight and scalable; in resource-limited environments and under green computing limitations, it is essential. Yet, existing path representation learning studies focus on accuracy and pay at most secondary attention to resource consumption and scalability. We propose a lightweight and scalable path representation learning framework, termed ***LightPath***, that aims to reduce resource consumption and achieve scalability without affecting accuracy, thus enabling broader applicability. More specifically, we first propose a sparse auto-encoder that ensures that the framework achieves good scalability with respect to path length. Next, we propose a relational reasoning framework to enable faster training of more robust sparse path encoders. We also propose global-local knowledge distillation to further reduce the size and improve the performance of sparse path encoders. Finally, we report extensive experiments on two real-world datasets to offer insight into the efficiency, scalability, and effectiveness of the proposed framework.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

Path representation learning, Lightweight, Self-supervised learning

Corresponding Author: Jilin Hu (jlhu@dase.ecnu.edu.cn).

ACM Reference Format:
Sean Bin Yang, Jilin Hu, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2023. LightPath: Lightweight and Scalable Path Representation Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599415

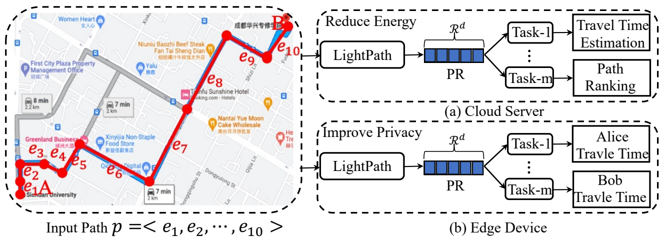
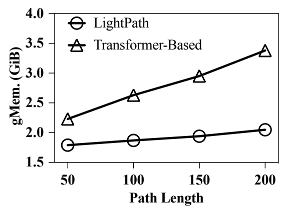


Figure 1: Intuition of the Lightweight Path Representation Learning Problem

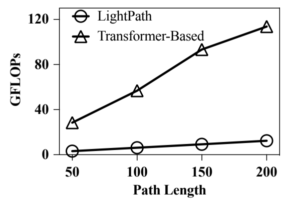
1 INTRODUCTION

Motivated in part by an increasing number of intelligent transportation and smart city services that operate on movement paths, path representation learning (PRL) has received remarkable attention [9, 12, 42]. In fact, a variety of intelligent transportation services involve paths, e.g., travel cost estimation [3, 6, 7, 19, 27, 33–35, 40, 41], trajectory analysis [13, 24, 25, 31, 32], and path ranking [21, 35, 37]. Path representations that are both accurate and compact, thus facilitating efficient operations, are in high demand as they hold the potential to significantly improve the services that use them. Indeed, recent path representation learning methods, in particular deep learning based methods, demonstrate impressive and state-of-the-art performance on a wide variety of downstream tasks.

However, existing path representation learning methods focus on accuracy improvement and pay at best secondary attention to scalability and resource usage. The resulting models often include large numbers of layers and parameters, driving up computational costs, power consumption, and memory consumption, especially for long paths. Although path encoders with many parameters may achieve good accuracy, they have two limitations. First, using large



(a) GPU Memory



(b) GFLOPs

Figure 2: Scalability w.r.t. Path Length.

path encoders in the cloud consumes substantial energy, which is not eco-friendly (cf. Fig 1(a)). Second, increasingly many users enjoy personalized services, e.g., personalized travel time estimation based on their own trajectories. Due to privacy concerns, such personalized services often require the path encoder to be deployed in resource-limited edge environments, such as on individual users' mobile phones (cf. Fig 1(b)), without having to upload their trajectories to the cloud. More generally, it is sensible to enable lightweight path representation learning that works in resource-limited environments. Next, existing path representation methods suffer from two limitations.

Poor scalability w.r.t. path length. Since a path is a sequence of road-network edges, path representation learning benefits from models that are good at capturing sequential relationships, such as the Transformer [29]. However, a Transformer-based method [4] employs a self-attention mechanism, where one edge attends to all other edges in a path in each attention, resulting in quadratic complexity, $O\left(N^2\right)$, in the path length N. This results in poor scalability to long paths with many edges. Figure 2 gives an example of the scalability w.r.t. path length N, covering both memory consumption and computational cost, in terms of GPU memory (gMem.) and Giga floating point operations per second (GFLOPs). We observe that when the path length N increases from 50 to 200 edges, the Transformer-based method performs poorly. A method that scales better w.r.t. N is desirable.

Very large model size. Many existing PRL models have large numbers of parameters, which restricts their use in resource-limited environments. For example, in a Transformer-based method [4], where the Transformer stacks L transformer layers, each layer employs multi-head (i.e., M heads) attentions. Thus, the Transformer functions like a *large cuboid*, with a complexity of $O\left(L \cdot M \cdot N^2\right)$, as shown in Figure 7a in the Appendix. Table 1 shows the numbers of parameters of Transformer-based path encoders when varying the number of layers among 12, 24, 48, and 96 while fixing the number heads at 8 per layer and the feature dimension of the encoder at 512. We observe that the model parameters grow dramatically when the number of encoder layers increase, preventing the models from being deployed in resource-limited environments. Moreover, models with large amounts of parameters also suffer from high storage and computational costs, which is not eco-friendly. More specifically, as shown in Figure 2a, for path length $N = 200$, the Transformer-based model consumes almost 3.4GiB GPU memory.

Table 1: Model Parameter Size with Varying Encoder Layers

Encoder Layers L	12	24	48	96
Parameters (Millions)	29.85	55.07	105.51	206.40

Proposed Solution. To tackle the above limitations, we propose *LightPath*, a lightweight and scalable path representation learning approach. To address the first limitation, we first propose a sparse auto-encoder targeting good scalability, w.r.t., path length. In particular, the introduction of sparseness reduces the path length from N to N' by removing edges and returning a sparse path of length N'. The sparse path is fed into a Transformer-based encoder, which reduces the complexity from $O\left(L \cdot M \cdot N^2\right)$ to $O\left(L \cdot M \cdot N'^2\right)$. As shown in Figure 7b, this reduces a huge cuboid to a *slimmer cuboid*. To avoid information loss due to the removed edges, we connect the encoder with a decoder, with the aim of reconstructing the full path. This enables scalable yet effective unsupervised training.

To further improve the training of the sparse encoder, we add an additional training scheme based on relational reasoning. In particular, for each path p_i, we construct two distinct sparse path views, denoted as p_i^1 and p_i^2, using different reduction ratios, e.g., removing 40% and 80%, respectively. Then, we propose a dual sparse path encoder, including the original main encoder, and an additional, auxiliary encoder. The dual sparse path encoder encodes the two path views. Thus, we achieve four path presentations PR_i^1, PR_i^2, $\hat{PR}_i^1$, and $\hat{PR}_i^2$ for path p_i according to the two path views and the two sparse path encoders, where PR and $\hat{PR}$ denote the representations from the main and the auxiliary encoders, respectively. Finally, given two path representations, we train a relational reasoning network to determine whether the two path representations are from the same "relation." If they are from the same path, we consider them as positive relations; otherwise, they are negative relations.

To address the second limitation, we propose a global-local knowledge distillation framework that aims to reduce the model size of the main path encoder, which not only enables use in resource-limited environments but also improves accuracy [26]. We consider the main path encoder as a teacher, and we create a lightweigth sparse encoder with fewer layers and one head as a student, further reducing a slim cuboid to a *slim rectangle* (cf. Figure (7c)). The global knowledge distillation tries to push the lightweight student to mimic the teacher from a global semantic level (i.e., path representation level), while the local knowledge distillation can push the lightweight student to mimic the edge correlations from the teacher, thus building a lightweight encoder while maintaining or even improving the accuracy of downstream tasks.

The study makes four main contributions. We first propose a unified sparse auto-encoder framework that provides *LightPath* with good scalability. w.r.t. path length. Subsequently, we introduce relational reasoning to enable efficient sparse auto-encoder training. Specifically, we propose two types of relational reasoning objectives for accurate and efficient path representation learning. These two objectives regularize each other and yield a more effective path encoder. Next, we propose a novel global-local knowledge distillation framework that enables a lightweight student sparse encoder to mimic a larger teacher sparse encoder from global and local perspectives. Finally, we report on extensive experiments on two large-scale, real-world datasets with two downstream tasks. The results offer evidence of the efficiency and scalability of the proposed framework as compared with eleven baselines.

2 PRELIMINARIES

We first cover important concepts that underlie the paper's proposal and then state the problem addressed.

2.1 Definitions

Definition 1. ***Road Network.*** *A road network is defined as a graph* $\mathbf{G} = (V, E)$*, where* V *is a set of vertices* v_i *that represents road intersections and* $E \subseteq V \times V$ *represents a set of edges* $e_i = (v_j, v_k)$ *that denotes road segments.*

Definition 2. ***Path.*** *A path* $p = \langle e_1, e_2, e_3, \cdots, e_N \rangle$ *is a sequence of connected edges, where* $e_i \in E$ *denotes an edge in path and two adjacent edges share a vertex. Next,* $p.N$ *denotes the length of path, i.e., the number of edges in* p*. We let* $p.\Phi = [1, 2, 3, \cdots, N]$ *denote a sequence of orders of the edges in* p*.*

Definition 3. ***Sparse Path.*** *A sparse path* $p' = \langle e_i \rangle_{i \in p'.\Omega}$ *contains a subset of the edges in path* p*, where* $p'.\Omega$ *is a sub-sequence of* $p.\Phi$*.*

Example. Given a path $p = \langle e_1, e_3, e_4, e_6, e_7 \rangle$ and $p.\Phi = [1, 2, 3, 4, 5]$ then path $p' = \langle e_1, e_4, e_7 \rangle$, where $p'.\Omega = [1, 3, 5]$, is one of the sparse paths for p.

2.2 Problem Definition

Given a set of paths $\mathbb{P} = \{p_i\}_{i=1}^{|\mathbb{P}|}$ in a road network G, scalable and efficient path representation learning aims to learn a function $SPE_\theta(\cdot)$ that can generate a generic path representation for each path $p_i \in \mathbb{P}$ without relying on labels. It can be formulated as follows.

$$PR = SPE_\theta(p_i) : \mathbb{R}^{N \times d_k} \rightarrow \mathbb{R}^d , \tag{1}$$

where PR is learned path representation, θ represents the learnable parameters for the sparse path encoder, N is the path length and d_k and d are the feature dimensions for an edge and a final path representation, respectively.

3 SPARSE PATH ENCODER

Figure 3 illustrates the sparse path encoder framework, which includes a sparsity operation, a sparse path encoder, and a path reconstruction decoder.

3.1 Sparsity Operation

A path consists of a sequence of edges $p = \langle e_1, e_2, e_3, \cdots, e_N \rangle$, which are the basic processing units of different sequential models. The processing times of sequential models become longer when the path gets longer. Thus, we propose a sparsity operation, which is an approach to reduce the path length from N to N', where N' is much less than N. For simplicity, we conduct path reduction by randomly removing a subset of edges in a path based on a high reduction ratio γ (e.g., $\gamma = 0.6$). A high reduction ratio γ (the ratio for edge removal) can significantly reduce the length of each input path,

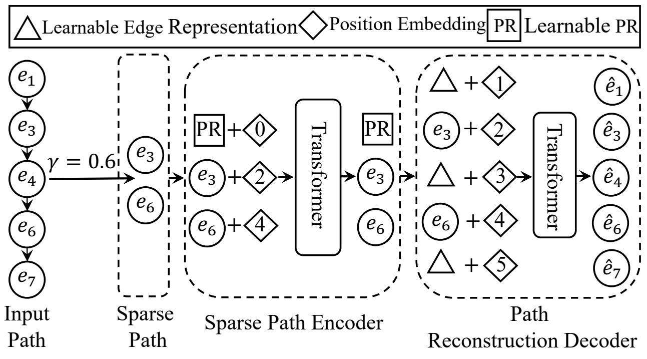


Figure 3: Sparse Auto-encoder.

thus enabling the scalability of the path [14, 18, 43]. Specifically, we construct the sparsity operation as:

$$p' = f(p, \gamma) = \langle e_j \rangle_{j \in \Omega} , \tag{2}$$

where p is input path. p' denotes the sparse path. For example, as shown in Figure 3, if we have a path $p = \langle e_1, e_3, e_4, e_6, e_7 \rangle$, then we conduct sparsity operation, which randomly removes a subset of edges in p, i.e., $\langle e_1, e_4, e_7 \rangle$ based on reduction ratio $\gamma = 0.6$ and achieve the sparse path $p' = \langle e_3, e_6 \rangle$ and $p'.\Omega = [2, 4]$. Thus, we can reduce path from N to N', i.e., from 5 to 2 in this example.

3.2 Learnable Path Representation

We use Transformer as our path encoder since it processes the input edges parallelly with respect to the self-attention mechanism. In contrast, the recurrent neural network (RNN) family is inefficient due to its recurrent nature. To avoid achieving path representation through extra aggregation function [35], we add a super extra learnable path representation representative PR in front of each sparse path, as shown in Figure 3. Moreover, PR is attached to position 0 for every path, thus enabling it to capture global information of the path during the training procedure. Thus, we update the p' as:

$$p' = \langle PR \rangle + \langle e_j \rangle_{j \in \Omega} = \langle e_k \rangle_{k \in \Omega'} , \tag{3}$$

where $e_0 = PR$ denotes a virtual edge and $\Omega' = [0, \Omega]$.

To preserve the sequential information of the path, we add learnable position embedding into the sparse path representations based on order information in Ω'. Specifically, we have:

$$\mathbf{X}'_p = \text{Concat}\langle x_k \rangle_{k \in \Omega'}, \text{ where } x_k = e_k + pos_k , \tag{4}$$

where pos_k represents the learnable position embedding for edges in the sparse path and $\mathbf{X}'_p$ represents the sparse path edge representation after concatenation.

3.3 Transformer Path Encoder

To achieve better performance, we usually stack multiple Transformer layers, each consisting of two sub-layers: multi-head attention and position-wise feed-forward network mentioned above. Motivated by [16], we employ a residual connection around each sub-layers, followed by layer normalization [1]. The stacked transformer model can be formulated as:

$$\begin{aligned} Z'_p &= \text{LayerNorm}(\mathbf{X}'_p + \text{ MultiHead } (\mathbf{X}'_p)) , \\ PR &= \text{LayerNorm}\left(Z'_p + \text{FFN}\left(Z'_p\right)\right) , \end{aligned} \tag{5}$$

where LayerNorm represents layer normalization and *PR* is learned path representation. Remarkably, our path encoder only takes as input a small subset of edges (e.g., 60%) of the full path edges, which means we ignore the removed edges and just consider unremoved edges during the encoder stage to enable the path scalability. Path scalability enables us to train our path encoders concerning different lengths of path effectively and reduce the corresponding computational cost and memory usage.

3.4 Path Reconstruction Decoder

To capture the global information of the full path, we further introduce a lightweight path decoder to reconstruct the removed edges in a path. As shown in Figure 3, we first complement the encoded path edges and path representation with a shared, learnable vector that represents the presence of a removed edge based on the original index of each edge in a path. Then, we add the position embedding vectors to all edge representation, which enables the learnable path representation vector to capture the global information of the entire path. Next, the path decoder takes as input the full set of representations, including (1) path representation, (2) encoded unremoved edges, and (3) removed edges. We select a more lightweight decoder structure, which has less number of Transformer layers. Since the path decoder is only used to perform path reconstruction, the architecture of our path decoder can be more flexible and independent of the path encoder. Thus, the decoder is much shallower than the encoder, e.g., one layer for the decoder and 12 layers for the encoder, which significantly reduces training time. We reconstruct the input path by predicting the removed edges to ensure the learned path representation contains complete information about the entire path. We employ mean squared error (MSE) as our reconstruction loss function and compute MSE between the reconstructed and initial edge representations in the edge level. We only employ MSE loss on removed edges, which can be formulated as follows:

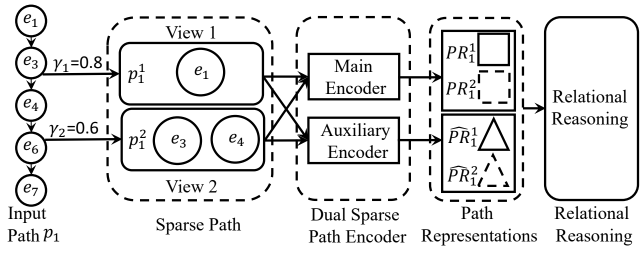
$$\mathcal{L}_{rec} = \frac{1}{N}\sum_{i=1}^{N}(e_i - \hat{e}_i)^2 \ , \tag{6}$$

where e_i and $\hat{e}_i$ are the initial and predicted masked edge representation, respectively. N represents the number of edges for each input path.

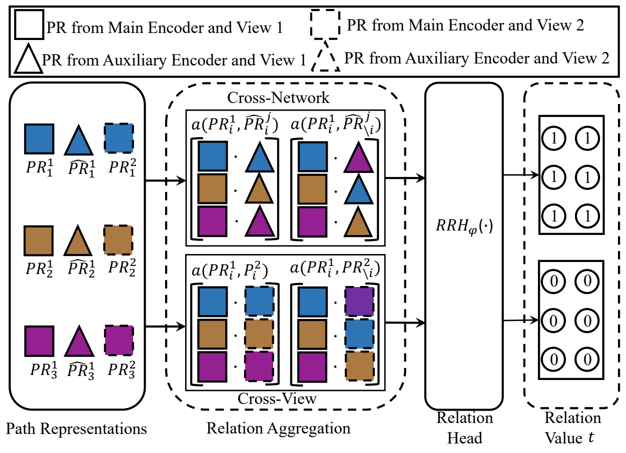
4 RELATIONAL REASONING PATH REPRESENTATION LEARNING

4.1 Overview

To further enhance sparse auto-encoder (cf. Section 3) training, we propose a novel self-supervised relational reasoning (RR) framework, as shown in Figure 4. The intuition behind this is that we train a relation head $RRH_\varphi(\cdot)$ to discriminate how path representations relate to themselves (same class) and other paths (different class). In particular, this framework consists of path representation construction (cf. Figure 4a) and relational reasoning (cf. Figure 4b), which includes cross-network relational reasoning and cross-view relational reasoning.



(a) Path Representation Construction Using Dual Sparse Path Encoders


PR from Main Encoder and View 1
PR from Main Encoder and View 2
PR from Auxiliary Encoder and View 1
PR from Auxiliary Encoder and View 2
Cross-Network
$a(PR_i^1, \widehat{PR}_i^j)$
$a(PR_i^1, \widehat{PR}_{\backslash i}^j)$
$a(PR_i^1, P_i^2)$
$a(PR_i^1, PR_{\backslash i}^2)$
Cross-View
$RRH_\varphi(\cdot)$
Path Representations
Relation Aggregation
Relation Head
Relation Value t

(b) Relational Reasoning

Figure 4: Illustration of RR Training: (a) Given an input path p_1, we construct two path views (i.e., p_1^1 and p_1^2) through two reduction ratios γ_1 and γ_2, based on which a main encoder and an auxiliary encoder are employed to generate path representations for each view (i.e., PR_1^1, PR_1^2, $\hat{PR}_1^1$, and $\hat{PR}_1^2$). (b) After getting corresponding path representations for paths in a minibatch, a relational reasoning path representation learning scheme, which utilizes both cross-network and cross-view relational reasoning modules, is deployed. In particular, for both modules, an aggregation function a joins positives (representations of the same paths, e.g., $a(PR_1^1, \hat{PR}_1^1)$, $a(PR_1^1, PR_1^2)$) and negatives (randomly paired representations, e.g., $a(PR_1^1, \hat{PR}_3^1)$, $a(PR_1^1, PR_3^2)$) through a commutative operator. Then relation head module $RRH_\varphi(\cdot)$ estimates the relation score y, which must be 1 for positive and 0 for negatives. Both cross-network and cross-view objectives are optimized minimizing the binary cross-entropy (BCE) between prediction and target relation value t. In this example, $i \in [1, 2, 3]$ denotes the number of paths in the minibatch and $j \in [1, 2]$ represents the number of views.

4.2 Dual Sparse Path Encoder

In this section, we introduce our dual sparse path encoder (SPE) that is employed to generate different path representations based on different path views. As shown in Figure 4a, given a path p_1, we first generate sparse paths in terms of two different reduction ratios γ_1 and γ_2. We consider them as different path views, i.e., path view 1 and path view 2. Then, our dual sparse path encoder, including a main encoder and an auxiliary encoder, takes as input two different path views (i.e., p_1^1 and p_1^2) and returns different path representations. Specifically, each encoder takes as input two different path views and returns two different path representations, where solid and dotted □ denote the path representations returned from main encoder based on path view 1 and path view 2, respectively, i.e., PR_1^1 and PR_1^2. In contrast, solid and dotted △ represent the path representations achieved from auxiliary encoder based on both path views, respectively, i.e,. $\hat{PR}_1^1$ and $\hat{PR}_1^2$. To this end, we construct four different path representations for a given path, which promote our design of cross-network relational reasoning and cross-view relational reasoning in turn. Finally, we formulate it as:

$$PR_i^j = SPE_\theta(p_i^j, \gamma) \ , \ \hat{PR}_i^j = SPE_{\hat{\theta}}(p_i^j, \gamma) \ , \tag{7}$$

where PR_i^j and $\hat{PR}_i^j$ are path representations obtained from the main encoder and the auxiliary encoder, respectively. p_i denotes the i-th path in the path set. $j \in [1, 2]$ denotes the path views. θ and $\hat{\theta}$ are the parameters for the main encoder and auxiliary encoder.

4.3 Relational Reasoning

4.3.1 Cross-Network Relational Reasoning. In *LightPath*, we employ a dual sparse path encoder, which includes main and auxilary encoder, as shown in Figure 4a. We first construct path representations through sparsity operation based on different reduction ratios γ_1 and γ_2. Given a set of path $\{p_1, p_2, \cdots, p_K\}$, we can have a set of path representations $\{PR_1^1, PR_2^1, \cdots, PR_K^1\}$ from main encoder and $\{\hat{PR}_1^1, \hat{PR}_2^1, \cdots, \hat{PR}_K^1\}$ or $\{\hat{PR}_1^2, \hat{PR}_2^2, \cdots, \hat{PR}_K^2\}$ from auxiliary encoder by using path representation construction. Then we employ relation aggregation $a(\cdot)$ that joins the positive path representation relations $\langle PR_i^1, \hat{PR}_i^1\rangle$ or $\langle PR_i^1, \hat{PR}_i^2\rangle$ and the negative path representation relations $\langle PR_i^1, \hat{PR}_{\backslash i}^1\rangle$, where i denotes the i-th path sample and $\backslash i \neq i$ represents randomly selected path representations in a minibatch. Take Figure 4b as an example, where $K = 3$. we join $\langle PR_1^1, \hat{PR}_1^1\rangle$ as a positive relation pair (representation from same path), and $\langle PR_1^1, \hat{PR}_2^1\rangle$ as a negative relation pair (representation from different paths) through aggregation function a. Next, the relational head $RRH_\varphi(\cdot)$, which is non-linear function approximator parameterized by φ, takes as input representation relation pairs of cross-network and returns a relation score y. Finally, we formulate the cross-network relational reasoning task as a binary classification task, where we use binary cross-entropy loss to train our sparse path encoder, which is given as follows.

$$\begin{aligned}\mathcal{L}_{cn} = \underset{\theta,\varphi}{\operatorname{argmin}} \sum_{i=1}^{K}\sum_{j=1}^{2}\sum_{k=1}^{2} \mathcal{L}\left(RRH_\varphi\left(a\left(PR_i^j, \hat{PR}_i^k\right)\right), t=1\right) + \\ \mathcal{L}\left(RRH_\varphi\left(a\left(PR_i^j, \hat{PR}_{\backslash i}^k\right)\right), t=0\right) \ ,\end{aligned} \tag{8}$$

where K is the the number of path samples in the minibatch. $a(\cdot,\cdot)$ is an aggregation function. $\mathcal{L}$ is a loss function between relation score and a target relation value. t is a target relation values.

The intuition behind this is to discriminate path presentations of same path and different paths, which are from different views across dual sparse path encoder and are able to distill the knowledge from historical observations, as well as stabilizing the main encoder training. To realize this, we adopt Siamese architecture for our dual sparse path encoder, where the auxiliary encoder does not directly receive the gradient during the training procedure. In contrast, we update its parameters by leveraging the momentum updating principle:

$$\hat{\theta}^t = m \times \hat{\theta}^{(t-1)} + (1-m) \times \theta^t, \tag{9}$$

where m is momentum parameters. θ and $\hat{\theta}$ are the parameters of the main encoder and the auxiliary encoder.

4.3.2 Cross-View Relational Reasoning. To enhance the learning ability of our *LightPath*, we further consider the ties between two views within main encoder, which acts as a strong regularization to enhance the learning ability of our methods. We do not have to include such relational reasoning within the auxiliary encoder because it will not directly compute gradient during training, and our goal is to train main encoder. Figure 4b shows the design of our cross-view relational reasoning, which contains two similar representations from two views based on γ_1 and γ_2. The intuition of cross-view relational reasoning is to preserve the relation between two views of the same path and discriminate them from the view of other paths.

Similar with cross-network, given a set of paths $\{p_1, p_2, \cdots, p_K\}$. We first achieve two set of path representations in terms of two path views based on main encoder, i.e., $\{PR_1^1, PR_2^1, \cdots, PR_k^1\}$ and $\{PR_1^2, PR_2^2, \cdots, PR_K^2\}$. Then, we join the positive relation pairs (e.g., $\langle PR_i^1, PR_i^2\rangle$) and negative relation pairs (e.g., $\langle PR_i^1, PR_{\backslash i}^2\rangle$) through aggregation function. For example, as shown in Figure 4b, there are 3 paths in the set. Thus, we can denote $\langle PR_1^1, PR_1^2\rangle$ as a positive pair and $\langle PR_1^1, PR_3^2\rangle$ as a negative pair. Then, we further employ relational head $RRH_\varphi(\cdot)$, which takes as input a positive pair and a negative pairs from different views, to achieve the corresponding relation score y for the cross-view relational reasoning. Last, we formulate the cross-view relational reasoning loss to discriminate how different views of a path is related to themselves and other paths. In this phase, the complete learning objective can be specified as:

$$\begin{aligned}\mathcal{L}_{cv} = \underset{\theta,\varphi}{\operatorname{argmin}} \sum_{i=1}^{K} \mathcal{L}\left(RRH_\varphi\left(a\left(PR_i^1, PR_i^2\right)\right), t=1\right) + \\ \mathcal{L}\left(RRH_\varphi\left(a\left(PR_i^1, PR_{\backslash i}^2\right)\right), t=0\right) \ ,\end{aligned} \tag{10}$$

where K is the the number of path samples in the minibatch.

4.3.3 Objective for RR. To train our dual path encoder end-to-end and efficient learn path representations for downstream tasks, we jointly leverage both the cross-network and cross-view relation reasoning loss. Specifically, the overall objective function is formulated as Eq. 11.

$$\min_{\theta,\varphi} \mathcal{L}_{RR} = \mathcal{L}_{cn} + \mathcal{L}_{cv} \tag{11}$$

4.4 LightPath Training

To train our sparse path encoder and learn path representations for downstream tasks, we jointly minimize the reconstruction and RR loss. Specifically, the overall objective function is defined as:

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{RR} \tag{12}$$

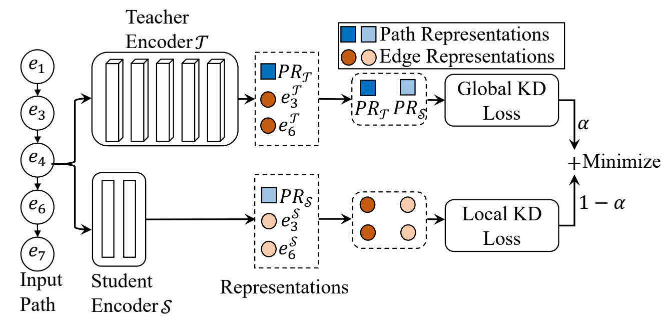


Figure 5: Illustration of GLKD.

5 GLOBAL LOCAL KNOWLEDGE DISTILLATION (GLKD)

So far, we realize our *LightPath* through sparse auto-encoder and relational reasoning and transform it from large cuboid (cf. Figure 7a to a slim cuboid (cf. Figure 7b). To enable the *LightPath* that can be deployed on resource-limited mobile devices, we introduce our global-local knowledge distillation (GLKD) to further reduce the size of the sparse auto-encoder, as shown in Figure 5. Specifically, GLKD constructs a local knowledge distillation by matching the representations of correlated edges. On such a basis, the global term distills the knowledge from teacher to student that enabling the informative and powerful path representation for the student model.

5.1 Global-path Representation Distillation

Given a path $p_i = \langle e_1, e_2, e_3, \cdots, e_N \rangle$, where N is the number of edges in a path. We define $PR^{\mathcal{T}}(p_i)$ and $PR^{\mathcal{S}}(p_i)$ represent the path representations achieved from teacher encoder $\mathcal{T}_\theta$ and student encoder $\mathcal{S}_\theta$. The intuition of global path representation knowledge distillation is to let the student encoder mimic the global properties captured by a large cuboid teacher encoder. And thus, the goal of global path representation knowledge distillation is to put closer the path representation from teacher encoder and student encoder in the latent space. We formalize this problem as minimizing a latent space distance representation pairs in terms of the large cuboid teacher encoder and the rectangle student encoder. The formulation of the objective function is given as follows:

$$\min_{\theta} \mathcal{L}_{global}\left(PR^{\mathcal{T}}(p_i), PR^{\mathcal{S}}(p_i)\right) = \left\| sp(PR^{\mathcal{T}}(p_i)/t) - sp(PR^{\mathcal{S}}(p_i)/t) \right\|^2, \tag{13}$$

where $sp(\cdot)$ is exponential function. t denotes the temperature.

5.2 Local-edge Correlation Distillation

The goal of local-edge structure distillation is to preserve the local similarity of the edge correlations in a path. In particular, it is expected that the representation of the same edge in a path represented by the teacher encoder and the student encoder should be close to each other. The intuition is that a rectangle student encoder mimics the edge correlations in a path captured by a large cuboid teacher encoder.

In specific, given a path $p = \langle e_1, e_2, e_3, \cdots e_N \rangle$ in a road network, where N is the number of edges in a path. Through applying an L-layers Transformer encoder (i.e., teacher encoder $\mathcal{T}_\theta$) and L'-layers Transformer encoder (i.e., student encoder $\mathcal{S}_\theta$) upon sparse path p', where $L \ll L'$, the edge representation that captures spatial dependencies are derived as follows.

$$F^{\mathcal{T}}(e_i)_{i=1}^{N'} = \mathcal{T}_\theta(p)\ , F^{\mathcal{S}}(e_i)_{i=1}^{N'} = \mathcal{S}_\theta(p)\ , \tag{14}$$

where $F^{\mathcal{T}}(e_i)_{i=1}^{N'}$ and $F^{\mathcal{S}}(e_i)_{i=1}^{N'}$ represent the edge representation with respect to the teacher and student encoder, respectively.

In this phase, the goal of learning is to reduce the latent space distance between same edge pair from the teacher and student encoder, respectively. To this end, we aim to minimize the following objective functions between edge representation pairs in terms of the parameters of the student encoder.

$$\min_{\theta} \mathcal{L}_{local}\left(\mathrm{F}^{\mathcal{T}}(e_i), \mathrm{F}^{\mathcal{S}}(e_i)\right) = \frac{1}{n}\sum_{i=1}^{n} \left\| sp(\mathrm{F}^{\mathcal{T}}(e_i)/t) - sp(\mathrm{F}^{\mathcal{S}}(e_i)/t) \right\|^2, \tag{15}$$

where $sp(\cdot)$ represents exponential function. t denotes the temperature.

5.3 Objective for *GLKD*

To train our global and local knowledge distillation in an end-to-end fashion, we jointly leverage both the global and local knowledge distillation loss. Specifically, the overall objective function to minimize is defined in Eq. 16.

$$\min_{\theta} \mathcal{L}_{GLKD} = \alpha * \mathcal{L}_{global} + (1-\alpha) * \mathcal{L}_{local}\ , \tag{16}$$

where α is balancing factor.

6 EXPERIMENTS

6.1 Experimental Setup

6.1.1 Datasets. We obtain two road network graph from OpenStreetMap[1]. The first is from Aalborg, Denmark, consisting of 10,017 nodes and 11,597 edges. The second is from Chengdu, China, consisting 6,632 nodes and 17,038 edges. We also achieve two substantial GPS trajectory datasets on two road networks. We consider 39,160 paths with length 50 in Aalborg dataset and 50,000 paths with lengths 50 in Chengdu dataset. Due to the lack of large amounts of long paths in the real-world datasets, we construct one synthetic dataset to verify the efficiency and scalability of *LightPath*. In particular, we first randomly pick 500 nodes in road network of Aalborg dataset, and then expand each node to a path by random walking until the path length reach the threshold (i.e., 100, 150, 200), which we refer as to generation process. Subsequently, we iterate the generation process 10 times to construct 5,000 paths for each path length.

6.1.2 Downstream Tasks. **Path Travel Time Estimation:** We obtain travel time (in seconds) for each path from the trajectory. We aim to utilize a regression model to predict the travel time based on the learned path representations. We employ Mean Absolute Error(**MAE**), Mean Absolute Relative Error(**MARE**), and Mean Absolute Percentage Error(**MAPE**) to evaluate the performance of travel time estimations.
Path Ranking: Each path is assigned a ranking score in the range [0, 1], which is obtained from historical trajectories by following the existing studies [37, 38]. To measure the path ranking, we apply

[1]https://www.openstreetmap.org

MAE, the Kendall rank correlation coefficient (τ), and the Spearman's rank correlation coefficient (ρ), which are widely used metrics in path ranking, to evaluate the effectiveness of path ranking.

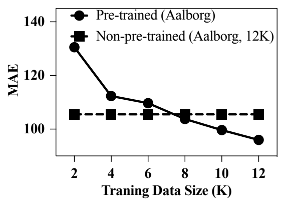
6.1.3 Models for Downstream Tasks. For all unsupervised learning methods, we first achieve the corresponding d dimensionality path representation and then we build a regression model that takes as input a path representation and output estimated the travel time and path ranking, respectively. In particular, we select ensemble model *Gradient Boosting Regressor*(GBR) [23] as our prediction model since they are regression problems.

6.1.4 Baselines. We compare *LightPath* with 11 baselines, including 6 unsupervised learning-based methods and 5 supervised learning-based methods. The unsupervised methods are: (1) **Node2vec** [11] is an unsupervised node representation model that learn node representation in a graph. (2) **MoCo** [15] is a momentum contrast for unsupervised visual representation learning. Here we use momentum contrast to learn path representations. (3) **Toast** [4], **t2vec** [20], and **NeuTraj** [39] are unsupervised trajectory representation learning framework. We use the same schema to learn path representations. (4) **PIM** [35] is an unsupervised path representation learning approach based on mutual information maximization. The supervised methods that take into account the labels from a specific downstream task to obtain path representations, which are: (1) **HMTRL** [21] enables unified route representation learning and exploits both spatio-temporal dependencies in road networks and the semantic coherence of historical routes. (2) **PathRank** [37] is a supervised path representation learning model based on GRUs. (3) **CompactETA** [10] aims to estimate travel time based on a real time predictor and an asynchronous updater. (4) **HierETA** [5] is a supervised multi-view trajectory representation method to estimate the travel time. (5) **LightPath-Sup** is a supervised version of our *LightPath*, where we train it in a supervised manner.

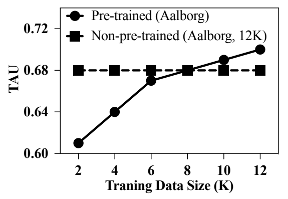
For all unsupervised learning methods, we first use unlabeled training data (e.g., 30K unlabeled Aalborg dataset and 50K unlabeled Chengdu dataset) to train path encoders to obtain path representations. Then a regression model takes as input path representations and returns the estimated travel time and path ranking score using a limited labeled training dataset, e.g., the 12K labeled Aalborg dataset. In comparison, for all supervised learning methods, we directly train path encoders using a limited labeled training dataset, e.g., the 12K labeled Aalborg dataset and Chengdu dataset.

6.2 Experimental Results

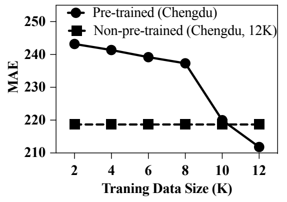
6.2.1 Overall Performance. Table 2 shows the overall performance of our *LightPath* and all the compared baselines on both datasets in terms of different evaluation metrics. Especially, we select 30K unlabeled paths on Aalborg and Chengdu datasets, respectively, but we only have 12K labeled paths for both datasets. Thus, we use 30K unlabeled paths to train path encoder for unsupervised-based methods. However, supervised approaches can only use the 12K labeled paths. Overall, *LightPath* outperforms all the baselines on these two tasks for both datasets, which demonstrates the advance of our model. Specifically, we can make the following observations. Graph representation learning based approach *Node2vec* is much worse than *LightPath*. This is because *Node2vec* fails to capture spatial dependencies in a path. In contrast, *LightPath* considers the spatial dependencies through the self-attention mechanism, thus achieving better performance.



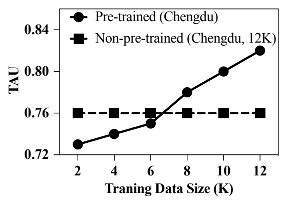
(a) Travel Time Estimation



(b) Path Ranking Estimation



(c) Travel Time Estimation



(d) Path Ranking Estimation

Figure 6: Effects of Pre-training.

Although *MoCo* considers the dependencies among edges in a path, this method still performs worse. The main reason is that *MoCo* can leverage the spatial dependencies, but it converges very slow since it needs large amounts of negative samples to enable training. *LightPath* also outperform *t2vec* and *NeuTraj*, which both are first design to learn trajectory representation for trajectory similarity computation. This suggests that random drops on some edges and not reconstruct these edges in a path resulting in spatial information missing, thus achieving the worse performance on downstream tasks. *PIM* consistently outperforms all other unsupervised baselines, which demonstrates the effectiveness of representation learning. The main reason is that *PIM* is designed for path representation learning. However, *PIM* is InfoNCE based method and has high computation complexity, making it hard to deploy on resource-limited edge devices. *HMTRL*, *PathRank*, *CompactETA*, *HierETA* and *LightPath-Sup* are five supervised learning methods that achieve relatively worse performance due to the lack of labeled training data. Since labeling data is very time-consuming and expensive. We consider a scenario where labeled data is limited in this paper.

6.2.2 Using LightPath *as Pre-training Methods.* In this experiment, we evaluate the effect of Pre-training. We employ *LightPath* as a pre-training method for the supervised method *LightPath-Sup*. Specifically, we first train *LightPath* in an unsupervised fashion, and then we use the learned transformer path encoder to initialize the transformer in *LightPath-Sup*. Here, it takes as input a sequence of edge representations and predicts the travel time and path ranking score. Figure 6 illustrates the performance of *LightPath-Sup* w. and w/o pre-training over two downstream tasks on both datasets. When employing non-pre-trained *LightPath-Sup*, we train it using 12K labeled training paths. We notice that (1) when employing pre-training, we can obtain the same performance with no-pre-trained *LightPath-Sup* using less labeled data. For example, *LightPath-Sup* w. pre-training only needs 8K, and 10K labeled training paths for

Table 2: Overall Accuracy on Travel Time Estimation and Ranking Score Estimation

Method	Aalborg						Chengdu					
	Travel Time Estimation			Path Ranking			Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ	MAE	MARE	MAPE	MAE	τ	ρ
Node2vec	154.07	0.20	25.22	0.24	0.59	0.64	267.28	0.23	26.30	0.15	0.74	0.77
MoCo	146.29	0.19	21.60	0.25	0.53	0.57	237.14	0.20	23.13	0.15	0.77	0.81
Toast	137.27	0.17	20.43	0.24	0.59	0.63	240.57	0.21	23.50	0.11	0.65	0.68
t2vec	147.24	0.19	22.13	0.25	0.52	0.56	242.96	0.21	23.65	0.14	0.77	0.82
NeuTraj	117.06	0.15	18.09	0.25	0.60	0.64	232.96	0.20	22.73	0.12	0.79	0.83
PIM	102.09	0.14	14.92	0.20	0.63	0.67	223.34	0.19	21.69	0.12	0.80	0.84
HMTRL	101.81	0.13	14.51	0.17	0.68	0.72	218.94	0.19	21.22	0.09	0.83	0.84
PathRank	115.37	0.15	16.41	0.21	0.64	0.68	229.85	0.20	22.53	0.11	0.81	0.82
CompactETA	106.47	0.15	16.22	0.17	0.67	0.70	236.28	0.20	23.13	0.11	0.79	0.80
HierETA	*88.95*	*0.12*	*14.23*	*0.15*	*0.71*	*0.74*	*215.39*	*0.19*	*21.66*	*0.09*	*0.84*	*0.85*
LightPath-Sup	105.51	0.15	16.35	0.14	0.68	0.72	218.67	0.19	21.36	0.13	0.76	0.79
LightPath	**85.76**	**0.11**	**12.12**	**0.13**	**0.73**	**0.77**	**212.61**	**0.18**	**20.75**	**0.07**	**0.87**	**0.88**

Table 3: Effect of Variants of *LightPath*

Method	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
w/o RR	94.90	0.12	13.85	0.17	0.66	0.70
w/o Rec.	103.45	0.14	15.76	0.15	0.65	0.69
w/o ME	91.57	0.12	13.09	0.16	0.68	0.72
w/o CN	93.17	0.12	13.35	0.15	0.68	0.73
w/o CV	89.84	0.12	13.51	0.15	0.68	0.72
LightPath	**85.76**	**0.11**	**12.12**	**0.13**	**0.73**	**0.77**

Table 4: Effect of KD, Global Loss and Local Loss

Method	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
w/o KD	87.77	0.11	12.94	0.14	0.70	0.74
w/o Global	90.24	0.12	13.31	0.18	0.67	0.71
w/o Local	89.23	0.12	12.78	0.16	0.69	0.73
LightPath	**85.76**	**0.11**	**12.12**	**0.13**	**0.73**	**0.77**

the Aalborg, Denmark and Chengdu, China, respectively, to achieve the same performance of *LightPath-Sup* w/o pre-training with 12k labeled samples on the task of travel time estimation. (2) *LightPath-Sup* w. pre-training achieves higher performance than *LightPath-Sup* w/o pre-training. We observe similar results on the task of path ranking, demonstrating that *LightPath* can be used as a pre-training method to enhance supervised methods.

6.2.3 Ablation Studies. To verify the effectiveness of different components in *LightPath*, we conduct ablation studies on *LightPath*: a) effect of variants of *LightPath*, specifically reconstruction (Rec) loss, relational reasoning (RR) loss, cross-network loss and cross-view loss; b) effect of global-local knowledge distillation.

a) *Effect of variants of* LightPath, we consider five variants of *LightPath*: 1) *w/o RR*; 2) *w/o Rec.*; 3) *w/o ME*; 4) *w/o CN*; 5) *w/o CV*. In *w/o RR*, we only consider path reconstruction loss and use main encoder; In *w/o Rec.*, we only consider relational reasoning loss; In *w/o ME*, we consider both path reconstruction and relational reasoning losses, but we do not consider Siamese architectures in dual path encoder; In *w/o CN*, we remove the cross-network loss in RR; And in *w/o CV*, we remove cross-view loss in RR. Table 3 reports the results on Aalborg. We can observe that (1) *LightPath w/o Rec.* achieves the worst performance because the learned *PR* only capture information from sparse path while ignoring the removed edges, which verifies the importance of path reconstruction decoder; (2) *LightPath w/o RR* also achieves the poor performance, which implies the effectiveness of self-supervised relational reasoning. (3) We observe that the performance of *LightPath* degrades without cross-network and cross-view loss, which further demonstrates the effectiveness of our relational reasoning loss. (4) We notice that *LightPath* achieves the best performance. This result implies that all the proposed modules contribute positively to the final performance, which validates that *LightPath* takes advantage of all designed components.

b) *Effect of KD, global KD loss, local KD loss*: We further study the effect of global-local knowledge distillation. We compared our framework with three variants: 1) *w/o KD*, which denotes the performance of the teacher model; 2) *w/o global KD loss*, which removes global loss from global-local knowledge distillation; and 3) *w/o local KD loss*, which removes local loss from global-local knowledge distillation. As shown in Table 4, compared with *KD*, *LightPath* achieves a better performance, which verifies that the teacher model can improve the performance of the student model. Both global and local loss can improve the performance of the learned path representation of the student model. In specific, global loss makes more contributions to the learned path representations. As a result, removing global loss degrades performance significantly.

6.2.4 Parameter Sensitivity Analysis. We proceed to study three important hyper-parameters, including 1) model scalability comparison, 2) Effect of Reduction Ratio γ.

Model Scalability. In the sequel, we explore the model scalability in terms of reduction ratio and path length based on the synthetic dataset. Table 5 depicts the results for both *LightPath* and its teacher

Table 5: Model Scalability vs. Reduction Ratio (γ) and Path Length (N)(Aalborg)

N	***LightPath***												
	$\gamma = 0$		$\gamma = 0.1$		$\gamma = 0.3$		$\gamma = 0.5$		$\gamma = 0.7$		$\gamma = 0.9$		Para.
	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	
50	8.01	1.47	7.48	1.39	6.44	1.37	5.39	1.36	4.34	1.34	3.18	1.33	1.570
100	15.77	1.60	14.72	1.50	12.62	1.48	10.52	1.46	8.43	1.44	6.23	1.43	1.570
150	23.53	1.72	21.95	1.64	18.81	1.60	15.66	1.58	12.52	1.55	9.27	1.52	1.570
200	31.29	1.90	29.19	1.81	24.99	1.77	20.80	1.73	16.61	1.68	12.31	1.65	1.570
	LightPath w/o KD												
	$\gamma = 0$		$\gamma = 0.1$		$\gamma = 0.3$		$\gamma = 0.5$		$\gamma = 0.7$		$\gamma = 0.9$		Para.
	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	GFLOPs	gMem.	
50	33.68	1.78	30.64	1.70	22.55	1.61	18.47	1.53	12.39	1.47	5.70	1.41	5.525
100	66.60	2.53	60.52	2.37	48.36	2.11	36.19	1.86	24.03	1.72	11.26	1.58	5.525
150	99.53	3.44	90.41	3.23	72.16	2.76	53.91	2.35	35.65	2.03	16.82	1.82	5.525
200	132.54	4.74	120.29	4.30	95.96	3.53	71.64	2.94	47.31	2.43	22.37	2.10	5.525

model, with varying $\gamma = 0, 0.1, 0.3, 0.5, 0.7, 0.9$. $\gamma = 0$ denotes we do not conduct sparsity operation for the input path, i.e., using a classic Transformer based encoder. We can observe that the GFLOPs and gMem. (GiB) decrease with the increase in the reduction ratio. It is because the higher value of γ is, the more edges we can remove. Second, *LightPath* has significantly reduced model complexity, w.r.t., GFLOPs and gMem.. For example, we can reduce the training GFLOPs by 2.54× for the *LightPath* by increasing the reduction ratio γ from 0 to 0.9 in terms of path length 200. Moreover, *LightPath* also shows better performance (i.e., GFLOPs and gMem.) over teacher model, e.g., 1.79× GFLOPs speedup with reduction ratio $\gamma = 0.9$. Third, the parameters (Para. (Millions)) of teacher model is at least 3.5× of *LightPath*, which implies the effectiveness of our proposed framework. Overall, *LightPath* shows potential of scalability to support path representation learning for long paths.

7 RELATED WORK

7.1 Path Representation Learning

Path Representation Learning (PRL) aims to learn effective and informative path representations in road network that can be applied to various downstream tasks, i.e., travel cost estimation, and path ranking. Existing studies can be categorised as supervised learning (SL) based [21, 37, 38], unsupervised learning (UL) based [35], and weakly supervised learning (WSL) based [36] approaches. SL based methods aim at learning a task-specific path representation with the availability of large amounts of labeled training data [21, 37, 38], which has a poor generality for other tasks. UL methods are to learn general path representation that does not need labeled training data and generalizes well to multiple downstream tasks [35, 36]. In contrast, WSL methods try to learn a generic temporal path representation by introducing meaningful weak labels, e.g., traffic congestion indices, that are easy and inexpensive to obtain, and are relevant to different tasks [36]. However, we aim to learn generic path representations instead of temporal path representations in this paper. Thus, we do not select WSL method as baseline method. In particular, these methods are computationally expensive and hard to deploy in resource-limited environments.

7.2 Self-supervised Learning

State-of-the-art self-supervised learning can be classified into contrastive learning-based and relation reasoning-based methods. Contrastive learning-based methods [2, 17, 28, 30, 35], especially for InfoNCE loss-based, commonly generate different views of same input data through different augmentation strategies, and then discriminate positive and negative samples. However, these methods suffer from their quadratic complexity, w.r.t. the number of data samples, given that it needs a large number of negative samples to guarantee that the mutual information lower bound is tight enough [17]. In contrast, relation reasoning-based methods [8, 22] aim to learn relation reasoning head that discriminates how entities relate to themselves and other entities, which results in linear complexity. However, existing studies construct relation reasoning between different views from the same encoder, ignoring the effect of different views between different encoders, i.e., main encoder and auxiliary encoder in Siamese encoder architecture.

8 CONCLUSION

We design a lightweight and scalable framework called *LightPath* for unsupervised path representation learning. In this framework, we first propose sparse auto-encoder that is able to reduce path length N to N', where N is much larger than N', which in turn reduces the computation complexity of the model. Then, we use path reconstruction decoder to reconstruct the input path to ensure no edges information missing. Next, we propose a novel self-supervised relational reasoning approach, which contains cross-network relational reasoning and cross-view relational reasoning loss, to enable efficient unsupervised training. After that, we introduce global-local knowledge distillation to further reduce the size of sparse path encoder and improve the performance. Finally, extensive experiments on two real-world datasets verify the efficiency, scalability, and effectiveness of *LightPath*.

9 ACKNOWLEDGMENTS

This work was supported in part by Independent Research Fund Denmark under agreements 8022-00246B and 8048-00038B, the VILLUM FONDEN under agreements 34328 and 40567.

REFERENCES

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016).

[2] Dmitrii Babaev, Nikita Ovsov, Ivan Kireev, Mariya Ivanova, Gleb Gusev, Ivan Nazarov, and Alexander Tuzhilin. 2022. CoLES: Contrastive Learning for Event Sequences with Self-Supervision. In *SIGMOD*. 1190–1199.

[3] David Campos, Miao Zhang, Bin Yang, Tung Kieu, Chenjuan Guo, and Christian S. Jensen. 2023. LightTS: Lightweight Time Series Classification with Adaptive Ensemble Distillation. *SIGMOD* (2023).

[4] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. 2021. Robust Road Network Representation Learning: When Traffic Patterns Meet Traveling Semantics. In *CIKM*. 211–220.

[5] Zebin Chen, Xiaolin Xiao, Yue-Jiao Gong, Jun Fang, Nan Ma, Hua Chai, and Zhiguang Cao. 2022. Interpreting Trajectories from Multiple Views: A Hierarchical Self-Attention Network for Estimating the Time of Arrival. In *KDD*. 2771–2779.

[6] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. 2022. Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting. In *IJCAI*, Luc De Raedt (Ed.). 1994–2001.

[7] Razvan-Gabriel Cirstea, Bin Yang, Chenjuan Guo, Tung Kieu, and Shirui Pan. 2022. Towards Spatio- Temporal Aware Traffic Time Series Forecasting. In *ICDE*. 2900–2913.

[8] Haoyi Fan, Fengbin Zhang, and Yue Gao. 2020. Self-Supervised Time Series Representation Learning by Inter-Intra Relational Reasoning. *CoRR* abs/2011.13548 (2020).

[9] Ziquan Fang, Lu Pan, Lu Chen, Yuntao Du, and Yunjun Gao. 2021. MDTP: A Multi-source Deep Traffic Prediction Framework over Spatio-Temporal Trajectory Data. *Proc. VLDB Endow.* 14, 8 (2021), 1289–1297.

[10] Kun Fu, Fanlin Meng, Jieping Ye, and Zheng Wang. 2020. CompactETA: A Fast Inference System for Travel Time Prediction. In *KDD*. 3337–3345.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD*. 855–864.

[12] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian S. Jensen. 2018. Learning to Route with Sparse Trajectory Sets. In *ICDE*. 1073–1084.

[13] Xiaolin Han, Reynold Cheng, Chenhao Ma, and Tobias Grubenmann. 2022. DeepTEA: Effective and Efficient Online Time-dependent Trajectory Outlier Detection. *Proc. VLDB Endow.* 15, 7 (2022), 1493–1505.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. 2022. Masked Autoencoders Are Scalable Vision Learners. In *CVPR*. 15979–15988.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[17] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *CoRR* abs/1808.06670 (2018).

[18] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *KDD*. 594–604.

[19] Jilin Hu, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2019. Stochastic Weight Completion for Road Networks Using Graph Convolutional Networks. In *ICDE*. 1274–1285.

[20] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *ICDE*. 617–628.

[21] Hao Liu, Jindong Han, Yanjie Fu, Jingbo Zhou, Xinjiang Lu, and Hui Xiong. 2020. Multi-Modal Transportation Recommendation with Unified Route Representation Learning. *Proc. VLDB Endow.* 14, 3 (2020), 342–350.

[22] Massimiliano Patacchiola and Amos J. Storkey. 2020. Self-Supervised Relational Reasoning for Representation Learning. In *NeurIPS*.

[23] Sven Peter, Ferran Diego, Fred A. Hamprecht, and Boaz Nadler. 2017. Cost efficient gradient boosting. In *NIPS*. 1551–1561.

[24] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: A Distributed In-Memory Trajectory Analytics System. In *SIGMOD*. 1681–1684.

[25] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed In-Memory Trajectory Analytics. In *SIGMOD*. 725–740.

[26] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*, Vol. 139. 10347–10357.

[27] Luan V. Tran, Minyoung Mun, Matthew Lim, Jonah Yamato, Nathan Huh, and Cyrus Shahabi. 2020. DeepTRANS: A Deep Learning System for Public Bus Travel Time Estimation using Traffic Forecasting. *Proc. VLDB Endow.* 13, 12 (2020), 2957–2960.

[28] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018).

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[30] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR*.

[31] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast Large-Scale Trajectory Clustering. *Proc. VLDB Endow.* 13, 1 (2019), 29–42.

[32] Zheng Wang, Cheng Long, Gao Cong, and Yiding Liu. 2020. Efficient and Effective Similar Subtrajectory Search with Deep Reinforcement Learning. *Proc. VLDB Endow.* 13, 11 (2020), 2312–2325.

[33] Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S. Jensen. 2021. AutoCTS: Automated Correlated Time Series Forecasting. *Proc. VLDB Endow.* 15, 4 (2021), 971–983.

[34] Xinle Wu, Dalin Zhang, Miao Zhang, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2023. AutoCTS+: Joint Neural Architecture and Hyperparameter Search for Correlated Time Series Forecasting. *SIGMOD* (2023).

[35] Sean Bin Yang, Chenjuan Guo, Jilin Hu, Jian Tang, and Bin Yang. 2021. Unsupervised Path Representation Learning with Curriculum Negative Sampling. In *IJCAI*. 3286–3292.

[36] Sean Bin Yang, Chenjuan Guo, Jilin Hu, Bin Yang, Jian Tang, and Christian S. Jensen. 2022. Weakly-supervised Temporal Path Representation Learning with Contrastive Curriculum Learning - Extended Version. *CoRR* abs/2203.16110 (2022).

[37] Sean Bin Yang, Chenjuan Guo, and Bin Yang. 2022. Context-Aware Path Ranking in Road Networks. *IEEE Trans. Knowl. Data Eng.* 34, 7 (2022), 3153–3168.

[38] Sean Bin Yang and Bin Yang. 2020. Learning to Rank Paths in Spatial Networks. In *ICDE*. 2006–2009.

[39] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *ICDE*. 1358–1369.

[40] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective Travel Time Estimation: When Historical Trajectories over Road Networks Matter. In *SIGMOD*. 2135–2149.

[41] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2021. An Effective Joint Prediction Model for Travel Demands and Traffic Flows. In *ICDE*. 348–359.

[42] Junbo Zhang, Yu Zheng, Junkai Sun, and Dekang Qi. 2020. Flow Prediction in Spatio-Temporal Networks Based on Multitask Deep Learning. *IEEE Trans. Knowl. Data Eng.* 32, 3 (2020), 468–478.

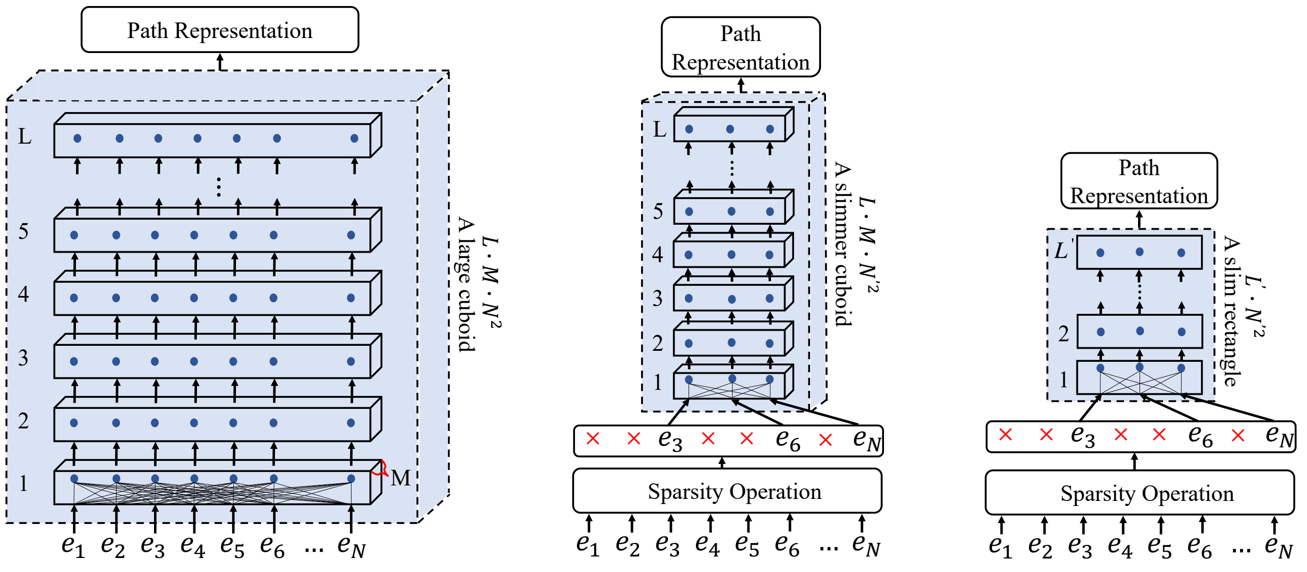
[43] Yizhen Zheng, Shirui Pan, Vincent C. S. Lee, Yu Zheng, and Philip S. Yu. 2022. Rethinking and Scaling Up Graph Contrastive Learning: An Extremely Efficient Approach with Group Discrimination. In *NeurIPS*.

A APPENDIX

In this section, we detail encoder architecture comparison in Figure 7, implementation details, additional parameter sensitivity analysis, and model efficiency.

A.1 Implementation Details

We employ an asymmetrical sparse auto-encoder architecture and randomly initialize all learnable parameters with uniform distributions. In particular, we adopt Siamese architecture, where we update the parameters of the auxiliary encoder based on the momentum updating principle based on the main encoder and we set the momentum parameter $m = 0.99$. We employ node2vec [11] to embed each edge to 128-dimensional vectors and set the dimension for path representation to 128. For a fair comparison, we set the path representation dimensionality of all baseline methods as 128. We select concatenate as the relation aggregation function $a(\cdot, \cdot)$. We use the AdamW optimizer with a cosine decay learning rate schedule over 400 epochs, with a warm-up period of 40 epochs. We set the base learning rate to 1e-3 and betas as $(0.9, 0.95)$. We vary γ from 0.1,0.3,0.5,0.7,0.9 to study the effect of path scalability and efficiency for the *LightPath*. In addition, we consider four different path lengths, i.e., 50, 100, 150, and 200, to study the effectiveness, efficiency, and scalability of the *LightPath*. We then evaluate our *LightPath* as well as all baselines on a powerful Linux server with 40



(a) Traditional Transformer Encoder (b) Sparse Transformer Encoder (c) LightPath

Figure 7: Encoder Architectures: (a) A traditional transformer encoder with L layers and M heads, takes as input a path (N-Length) and has complexity $O(L \cdot M \cdot N^2)$; (b) A sparse transformer encoder takes as input a sparse path (i.e., reducing path length from N to N'), resulting in $O(L \cdot M \cdot N'^2)$ complexity; (c) *LightPath* further compresses the traditional transformer in terms of layers and heads, yielding complexity $O(L' \cdot N'^2)$, making it more scalable and lightweight than a traditional transformer encoder.

Table 6: Effect of Temperature t in KD

t	**Aalborg**					
	Travel Time Estimation			**Path Ranking**		
	MAE	**MARE**	**MAPE**	**MAE**	τ	ρ
1	92.01	0.12	13.30	0.16	0.65	0.69
3	94.11	0.12	13.54	0.15	0.68	0.72
5	90.39	0.12	12.85	0.15	0.66	0.70
7	89.64	0.12	12.76	0.15	0.70	0.74
9	**85.76**	**0.11**	**12.12**	**0.13**	**0.73**	**0.77**
11	87.15	0.12	12.43	0.14	0.70	0.74

Table 7: Effect of Reduction Ratio γ

γ	**Aalborg**					
	Travel Time Estimation			**Path Ranking**		
	MAE	**MARE**	**MAPE**	**MAE**	τ	ρ
0.1	**82.79**	**0.11**	11.95	**0.12**	**0.74**	**0.77**
0.3	84.75	0.11	12.14	0.13	0.73	0.77
0.5	84.81	0.11	**11.86**	0.14	0.72	0.76
0.7	85.91	0.11	12.49	0.14	0.71	0.75
0.9	85.76	0.11	12.12	0.13	0.73	0.77

Intel(R) Xeon(R) W-2155 CPU @ 3.30GHz and two TITAN RTX GPU cards. Finally, all algorithms are implemented in PyTorch 1.11.0.

A.2 Parameter Sensitivity Analysis

We proceed to study three important hyper-parameters, including 1) effect of reduction ratio γ, 1) the parameter of temperature for global-local knowledge distillation, and 3) effect of balancing factor α.

Effect of Temperature t of Knowledge Distillation. To study the effect of the temperature t, we conduct a parameter study on Aalborg, which is reported in Table 6. We can observe that the performance of *LightPath* varies with different temperatures. It can be figured out that the best temperature t is 9, which indicates warm temperature can mitigate the peakiness of the teacher model and results in better performance.

Effect of Reduction Ratio γ. To study the impact of reduction ratio γ in the final performance, we conduct an experiment by varying the γ from 0.1 to 0.9 on Aalborg dataset, which is shown in Table 7. We can observe that the overall performance in both downstream tasks degrades a little when γ increases, which is reasonable as the the model has more input information. However, we can also observe the performance differences are not so significant, which suggests the effectiveness of our proposed framework. Even when a high reduction ratio is applied, the performance does not does not go down too much, which is led by the best same teacher model. Therefore, our proposed method can achieve good scalability while ensuring accuracy.

Effect of Balancing Factor α. To study the effect of the balancing factor of global-local knowledge distillation, we conduct a parameter study on Aalborg dataset. Based on the results reported in Table 8, we observe that the performance of our model changes when varying α. We can observe that the optimal α is 0.6, which means

Table 8: Effect of Balancing Factor α

α	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
0	90.24	0.12	12.78	0.16	0.69	0.73
0.2	89.35	0.12	12.85	0.14	0.69	0.73
0.4	91.57	0.12	13.17	0.15	0.69	0.73
0.6	**85.76**	**0.11**	**12.12**	**0.13**	**0.73**	**0.77**
0.8	87.44	0.12	12.76	0.14	0.70	0.75
1	89.23	0.12	12.78	0.16	0.69	0.73

Table 9: Comparison with Whole Path Input

	Aalborg		
	Travel Time Estimation		
	MAE	MARE	MAPE
LightPath w/o RR ($\gamma = 0.1$)	91.97	0.12	13.53
LightPath w/o RR ($\gamma = 0.1$)	90.85	0.12	13.39
LightPath ($\gamma = 0.1$)	82.79	0.11	11.95

that global and local knowledge distillation loss can contribute to the *LightPath*'s performance. When $\alpha = 0$, the global knowledge distillation loss is ignored, which yields poor performance. When $\alpha = 1.0$, the local knowledge distillation loss is ignored, and the performance also performs poorly. This confirms our conjecture that the two proposed global-local knowledge distillation losses can regularize each other and achieve better results than only optimizing one of them (i.e., $\alpha = 0.0$ or $\alpha = 1.0$).

A.3 Comparison with Whole Path Input

Compared with whole path input, we consider a variant "LightPath w/o RR" where only reconstruction loss is used and the relational reasoning (RR) loss is disabled. In this setting, we can see in the table 9 that LightPath w/o RR ($\gamma = 0$), i.e., using whole paths, outperforms LightPath w/o RR ($\gamma = 0.1$), i.e., using partial paths. This means that, when only using the reconstruction loss, using whole paths are indeed better than using partial paths. However, when using both losses, LightPath ($\gamma = 0.1$) outperforms LightPath w/o RR ($\gamma = 0$). This demonstrates that the relational reasoning loss, which employs partial paths to create different views, is indeed effective.*LigthPath*,

A.4 Comparison with Fixed Interval Strategy

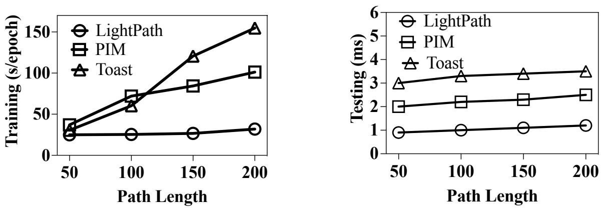
We then conducted additional experiments where we removed edges at fixed intervals strategy. Specifically, we set the fixed interval to 10 and removed n edges out of every 10 (for instance, we deleted 1 edge out of every 10 edges, i.e., corresponding to a removal ratio of $\gamma = 0.1$). The results on travel time estimation in Aalborg shown in Table 10, which indicate that the fixed interval edge removal strategy (ref. as to first three rows) achieves the worse performance compared with the random edge removal strategy (ref. as to last row).

A.5 Model Efficiency

We finally evaluate the model efficiency, including training and inference phases. Figure 8 illustrates the corresponding results. The first observation is that *LightPath* outperforms *PIM* and *Toast* in both training and inference phases. In the training phase, *LightPath* is more than 3× faster than *PIM* and almost 5× faster than *Toast* when path length is 200. In the testing phase, we measure the running time for each path sample. As observed, *LightPath* achieves up to at least 100% and almost 200% performance improvement compared with *PIM* and *Toast* when path length is 200.

Table 10: Comparison with Fixed Interval Strategy

Strategy	γ	Aalborg		
		Travel Time Estimation		
		MAE	MARE	MAPE
LightPath-Fixed	1/10	87.77	0.12	12.95
LightPath-Fixed	5/10	89.63	0.12	12.86
LightPath-Fixed	9/10	92.87	0.12	12.23
LightPath-Random	0.9	85.76	0.11	12.12



(a) Training (b) Testing

Figure 8: Model Efficiency Evaluation