Efflex: Efficient and Flexible Pipeline for Spatio-Temporal Trajectory Graph Modeling and Representation Learning

Ming Cheng[1*], Ziyi Zhou[1*], Bowen Zhang[2*], Ziyu Wang[3], Jiaqi Gan[1], Ziang Ren[1], Weiqi Feng[4], Yi Lyu[5], Hefan Zhang[1], Xingjian Diao[1†]
[1]Dartmouth College [2]Shanghai Jiao Tong University [3]University of California, Irvine
[4]Harvard University [5]Independent Researcher
{ming.cheng.gr, ziyi.zhou.gr, xingjian.diao.gr}@dartmouth.edu

Abstract

In the landscape of spatio-temporal data analytics, effective trajectory representation learning is paramount. To bridge the gap of learning accurate representations with efficient and flexible mechanisms, we introduce Efflex, a comprehensive pipeline for transformative graph modeling and representation learning of the large-volume spatio-temporal trajectories. Efflex pioneers the incorporation of a multi-scale k-nearest neighbors (KNN) algorithm with feature fusion for graph construction, marking a leap in dimensionality reduction techniques by preserving essential data features. Moreover, the groundbreaking graph construction mechanism and the high-performance lightweight GCN increase embedding extraction speed by up to 36 times faster. We further offer Efflex in two versions, Efflex-L for scenarios demanding high accuracy, and Efflex-B for environments requiring swift data processing. Comprehensive experimentation with the Porto and Geolife datasets validates our approach, positioning Efflex as the state-of-the-art in the domain. Such enhancements in speed and accuracy highlight the versatility of Efflex, underscoring its wide-ranging potential for deployment in time-sensitive and computationally constrained applications.

1. Introduction

The analysis of high-volume spatio-temporal data, which captures both the location of human activities and their movement over time [34], is becoming increasingly essential across various fields including cloud computing [45, 49, 50], recommender systems [18, 46], network management [41], smart healthcare [5, 35] and monitoring [36, 48, 52], social policy analysis [26], and localization-based services [16, 53]. However, spatio-temporal data generated from

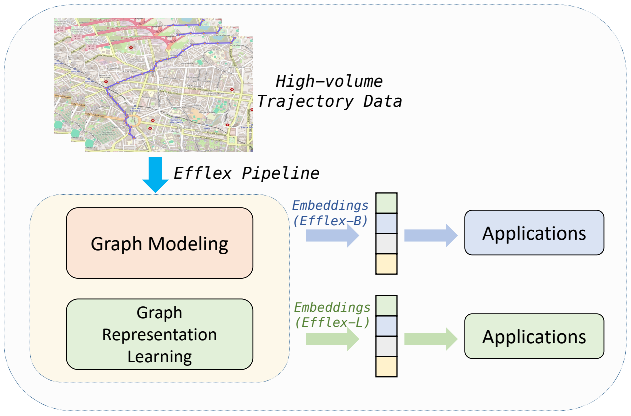


Figure 1. **The proposed Efflex pipeline.** We offer two models, Efflex-B and Efflex-L, to learn accurate embeddings from the original high-volume trajectory data. Efflex-B specializes in improving the speed while Efflex-L focuses on obtaining state-of-the-art performance, indicating different applications for each model.

real-world human activities often come in large volumes, diverse formats, and with a lot of non-useful information, leading to significant storage and analysis expenses [6]. These issues limit the data's practical use. Consequently, deriving meaningful insights from raw spatio-temporal data to create dense and informative representations has become a critical focus in the field of computer science. The essence of utilizing this data effectively lies in transforming it into embeddings – simplified representations that highlight underlying patterns, facilitating easier analysis and prediction by computational models.

While traditional approaches to spatio-temporal data analysis have largely relied on dimensionality reduction techniques [21, 33, 37] and neural network-based methodologies [4, 10, 28], these methods often grapple with limitation in processing speed, generalization to unseen data, and complex pre-processing overhead [2, 4]. To be spe-

*Equal contribution.
†Corresponding author.

cific, owing to the time-ordered characteristics of spatio-temporal data, many current techniques utilize neural architectures capable of sequence processing (including LSTM and RNN) for feature extraction and data representation learning [9, 23, 39]. Yet, these approaches often necessitate extensive preliminary data processing, which involves organizing spatial data into grids and using pre-arranged spatio-temporal datasets. Furthermore, the intricate design of these sequential neural networks contributes to significant training demands. Graph neural networks (GNNs) have emerged as a promising alternative, offering a powerful means of spatio-temporal representation learning [6, 15, 43]. ***However, there remains a significant gap: a specialized end-to-end pipeline that addresses the unique characteristics of spatio-temporal graph modeling with a focus on flexible and efficient learning mechanisms.***

In response to this gap, we introduce Efflex, an efficient and flexible pipeline designed specifically for spatio-temporal trajectory graph modeling and representation learning, as shown in Figure 1. Efflex leverages innovative techniques to construct graphs directly from raw trajectory data and learn from them efficiently, marking a significant departure from conventional methods. Our contributions through the development of Efflex are threefold:

- **Multi-scale graph construction.** To the best of our knowledge, we are the ***first*** to apply a multi-scale k-nearest neighbors (KNN) algorithm with feature fusion for graph construction, achieving nuanced dimensionality reduction while retaining essential trajectory data. Our innovation sets a new standard for capturing the complexity of spatio-temporal information.
- **State-of-the-art performance.** We develop a custom-built lightweight Graph Convolutional Network (GCN) that significantly enhances the model's efficiency. Compared to existing methodologies, our lightweight GCN improves embedding extraction speed up to **36 times faster** while maintaining competitive accuracy.
- **Generalized and flexible framework.** Efflex offers two versions tailored to diverse application needs. Efflex-L prioritizes precision with node2vec [13], while Efflex-B focuses on speed with our GCN, proving the framework's adaptability and broad real-world applicability.

2. Related Work

2.1. Matrix Factorization-Based Methods

Matrix factorization approaches are key in representation learning, typically reducing high-dimensional data into a more practical form while striving to maintain the integrity of the original data through matrix transformation [10, 27, 29].

As a prior work, PCA reduces dimension by projecting data onto a hyperplane structured to capture maximum variance, thus ensuring a robust representation of the data's original structure [4, 37]. SVD follows a similar reduction principle but factorizes the data matrix into orthogonal components, which help isolate independent information sources within the data [1, 33]. Meanwhile, MDS focuses on dimensional reduction by striving to conserve the pairwise distances between data points, aiming to uphold the spatial relationships post-reduction [21, 32].

Although these methods are cornerstones of data analysis, their rigid mathematical underpinnings can lead to suboptimal performance on sizable or intricate datasets[20]. They often struggle to adapt to novel, unseen data, especially within the ever-changing contexts of real-world applications [2, 4].

2.2. Learning-Based Methods

In recent years, learning-based methods using neural networks such as LSTM and RNN have been pivotal for efficiently learning representations from spatio-temporal data, capitalizing on their sequential dynamics [17, 23, 42]. Pei *et al.* [31] proposed Siamese Recurrent Networks (SRNs) to model time series similarities through recurrent neural networks, offering a fresh perspective on embedding learning. Similarly, NEUTRAJ [42] introduces a seed-guided neural metric learning method to efficiently compute trajectory similarities, leveraging RNNs for scalable and effective analysis. T3S [40] combines RNNs with attention mechanisms for nuanced representation learning of trajectory data, enhancing the accuracy of similarity computations. These approaches highlight the adaptability and efficiency of learning-based models in capturing the complexities of data through advanced neural network techniques. However, while these approaches are effective in identifying temporal characteristics, their extensive resource requirements for training pose challenges for widespread application and generalization in real-world settings.

Meanwhile, the field of graph representation learning has also seen significant innovations [13, 14, 24]. GGSNN merges gated recurrent units and graph neural networks to dynamically refine node representations, thus improving the detection of complex relationships [24]. Concurrently, node2vec [13] utilizes sophisticated random walk strategies to define and explore node neighborhoods, thereby enhancing feature learning. Further, Hamilton *et al.* [14] proposed GraphSAGE, which creates node embeddings by aggregating features from local neighborhoods, facilitating learning from large-scale graphs. These methods represent crucial advancements in graph analysis, however, they mainly focus on static structures, leaving a gap in capturing the spatio-temporal dynamics inherent to many real-world scenarios, highlighting the ongoing need for models that effectively integrate the spatio-temporal aspects of data.

3. Method

The overview of the Efflex pipeline is shown in Figure 2, which involves two parts. The Multi-Scale Graph Construction Module specializes in constructing adjacent matrices representing edge connections within the graph, and the Graph Representation Learning Module learns accurate graph embeddings.

3.1. Multi-Scale Graph Construction

3.1.1 Graph Construction From Trajectories

To bridge the gap between trajectory similarity and graph topology, and convert the trajectory representation learning problem into the task of graph embedding learning, each trajectory is represented as a vertex in the graph $G(V, E, S)$, where V, E, and S represent the vertex set, edge set, and weighted adjacent matrix, respectively.

Formally, assume $\mathcal{T} = \{T_1, T_2, ..., T_n\}$ as the set of n trajectories, each vertex $v_i \in V$ represents each trajectory $T_i \in \mathcal{T}$. Inspired by [42], the connection between vertex is determined through the k-nearest neighbors (KNN) algorithm: If T_i and T_j are k-nearest neighbors, an edge $e_{ij} \in E$ exists. $S = (s_{ij})_{|V|\times|V|}$, as the weighted adjacent matrix, quantitatively reflexes the edge connection between vertex, which is computed through the equation below:

$$s_{ij} = \frac{e^{dist(T_i,T_j)}}{\sum_{T_k\in\mathcal{K}} e^{dist(T_i,T_k)}} \tag{1}$$

where $\mathcal{K}$ indicates the set of k-nearest neighbors of trajectory T_i, and $dist(\cdot,\cdot)$ represents the distance function (Fréchet [11], Hausdorff [3], and DTW [12]). In Equation 1, s_{ij} measures the weight of connections between vertex i and i in the graph, and the adjacent matrix S of graph G is computed for the certain k.

3.1.2 Multi-Scale Graph Construction and Fusion

Considering that different values of k significantly affect the measure of trajectory similarity, we compute multiple adjacent matrices based on different k values. Specifically, larger k values capture more global information and lead to a comprehensive understanding of the overall graph, while smaller k provide a detailed view of local connections, potentially revealing finer and localized patterns within the graph. Formally, for specific k_m, the corresponding adjacent matrix S_m is obtained following the process above. Therefore, for $K = \{k_1, k_2, ..., k_m\}$, we can get:

$$\mathcal{S} = \{S_1, S_2, ..., S_m\} \tag{2}$$

which represents a set of weighted adjacent matrices in multi-scale. $\mathcal{S}$ will then be fused together for the Graph Representation Learning Module to extract graph embeddings.

We employ a lightweight linear transformation-based attention mechanism, inspired by [7, 8], to facilitate the extraction of intricate patterns and dependencies among the adjacent matrix set $\mathcal{S}$. Formally, assume $\mathcal{S} = \{S_1, S_2, ..., S_m\}$ in Equation 2 as the set of adjacency matrix, the stacked matrix is applied to a sequence of learnable linear transformations followed by non-linear activation functions to compute the attention weights W, as expressed below:

$$W = Attn(Stack(\mathcal{S})), Attn(\cdot) = Seq[LT(\cdot)f(\cdot)] \tag{3}$$

where $Stack(\cdot)$ indicates the stack operation among all matrices, $Seq[\cdot]$ represents the sequential blocks, $LT(\cdot)$ and $f(\cdot)$ refer to the linear transformation and non-linear activation function (LeakyReLU), respectively. Afterward, the fused adjacent matrix S' can be expressed by:

$$S' = Norm(MatMul(W, Stack(\mathcal{S}))) \tag{4}$$

where $MatMul$ is the matrix multiplication, and $Norm(\cdot)$ indicates the normalization operation to map the connection weights in the adjacent matrix within $[0, 1]$.

Since the initial adjacent matrices are constructed with different k values, where larger values aim to capture broad and global relationships of the graph while smaller values focus on extracting localized patterns among nearby nodes, this fusion procedure instructs the model to selectively leverage features in multi-scale by dynamically assigning weights to each adjacency matrix. The qualitative visualization of edge connection weights is shown in Figure 3. Section 4.6 further proves the effectiveness of this design.

3.2. Graph Representation Learning Module

The Graph Representation Learning Module is mainly designed as a sequential lightweight Graph Convolutional Network (GCN) [19, 38], aiming to generate accurate graph embeddings based on input adjacent matrix and node features. Formally, given F_V and S' as the node features and adjacent matrix, the sequential GCN model $M(\theta, S', F_V)$ with trainable parameters θ can be represented as:

$$\begin{aligned} M(\theta, S', F_V) = Seq[&MatMul(S', \\ &MatMul(F_V, W(\theta))) + \delta(\theta)] \end{aligned} \tag{5}$$

where $Seq[\cdot]$ represents the sequential blocks, $MatMul$ indicates the matrix multiplication, $W(\theta)$ and $\delta(\theta)$ refer to the learnable weights and bias parameters within the GCN. Node features F_V for pipeline training are obtained through the adjacent matrix S' with normalization operation and self-loops (1 on the diagonal). The output of the model ($Em(\theta) \in \mathcal{R}^{N\times d}$) with parameters θ is the learned embedding of N trajectories, each as a $1 \times d$ embedding vector (d is the preset embedding dimension).

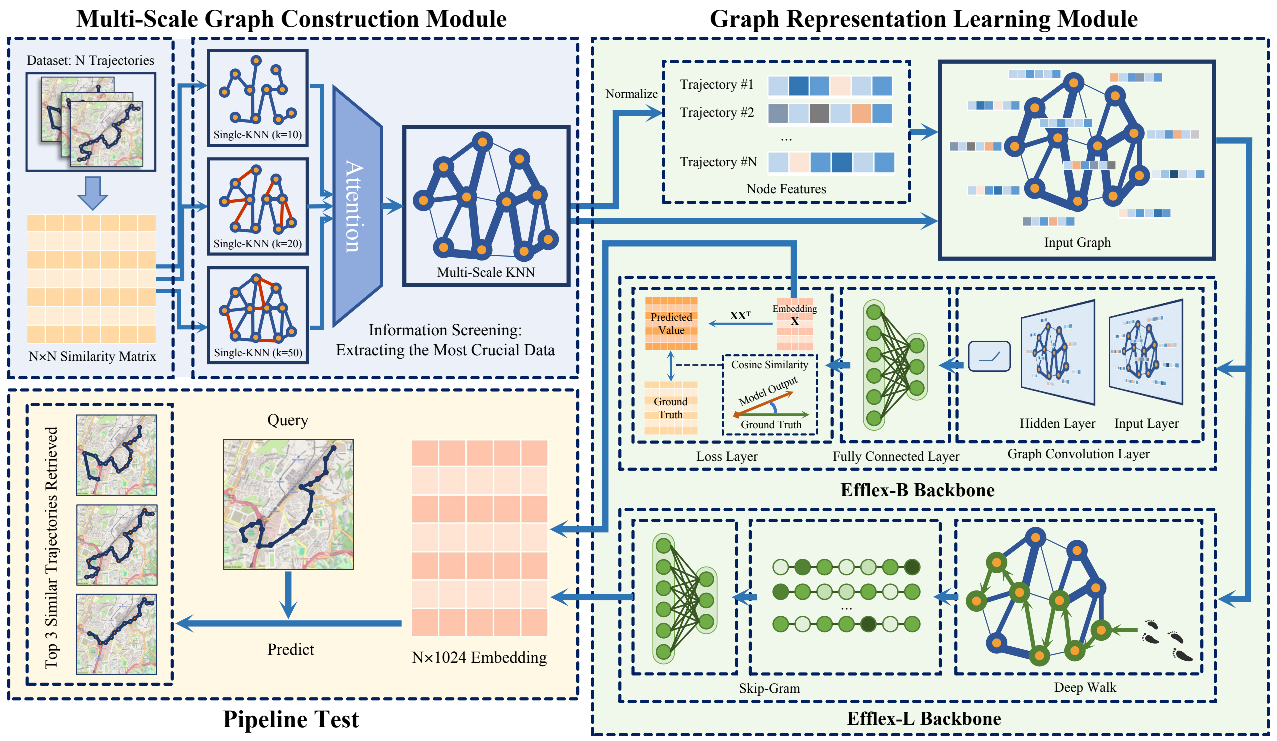


Figure 2. **Overview of the Efflex pipeline. Pipeline Train:** We build the graph from original trajectory data using multi-scale KNN algorithms with feature fusion by an attention module. The adjacent matrix and node features are then input into a lightweight GCN (Efflex-B) / node2vec [13] (Efflex-L) for accurate embedding learning. Efflex-B specializes in improving the speed significantly while Efflex-L embraces state-of-the-art performance. **Pipeline Test:** We conduct the *top-k* trajectory search experiment where given a query trajectory, the model outputs the *top-k* similar ones. The precise search results indicate Efflex's ability to learn high-quality representations of the original data.

To instruct the model to generate accurate graph embeddings, we employ cosine similarity distance [22, 44] as the loss function and AdamW [25] as the optimizer:

$$\theta = argmin_{\theta}(Cosine(GT, Em(\theta)Em(\theta)^T)) \tag{6}$$

where $Em(\theta) = M(\theta, S', F_V)$ is the embedding generated by the model, and GT indicates the ground truth by computing Euclidean distance (actual distance) between every two trajectories in the dataset. The learnable parameters θ will be optimized for each epoch. Ablation studies on other loss functions are shown in Section 4.6.

The designed lightweight structure allows our model to converge remarkably fast, which significantly reduces the training time compared to other existing graph representation learning models [13] while guaranteeing the generation of accurate embedding, as shown in Section 4.4.2.

3.3. Efflex with Flexibility

Benefiting from the generalized and flexible pipeline framework, Efflex offers two versions, Efflex-B and Efflex-L, with different models employed in the Graph Representation Learning Module. Specifically, Efflex-B uses the proposed lightweight GCN for representation learning, which achieves accurate and competitive accuracy while improving the training speed significantly ($\times$ 36 faster). Meanwhile, we replace the lightweight GCN with the deepwalk-based node2vec [13] model with massive parameters to learn graph embeddings, and regard the new version as Efflex-L. As shown in Section 4, Efflex-L reaches state-of-the-art performance under various evaluation metrics.

The two versions offered by the Efflex pipeline (Efflex-B/L) focus on diverse application needs. The base version specializes in applications requiring real-time modeling including wearable devices and embedded systems, while the large version can be used for environmental monitoring and smart city infrastructure management. Section 4.4.2 analyzes the performance of these two versions.

4. Experiments

4.1. Dataset

We conduct extensive experiments on commonly used trajectory datasets collected from real-world data points – Porto [30] and Geolife [51].

Methods	**Dataset**	**Hausdorff**			**Fréchet**			**DTW**		
		HR@10	**HR@50**	**R10@50**	**HR@10**	**HR@50**	**R10@50**	**HR@10**	**HR@50**	**R10@50**
PCA [37]	Porto	0.4850	0.5439	0.8454	0.4203	0.4909	0.8038	0.4751	0.5746	0.8534
SVD [33]	Porto	0.4839	0.5436	0.8445	0.4294	0.4977	0.8106	0.4689	0.5591	0.8362
MDS [21]	Porto	0.4839	0.6065	0.8770	0.4661	0.5874	0.8607	0.4762	0.5865	0.8673
Siamese [31]	Porto	0.3834	0.4999	0.7760	0.4740	0.5802	0.7970	0.3832	0.4804	0.7602
NEUTRAJ [42]	Porto	0.4372	0.5714	0.8089	0.5225	0.6351	0.8292	0.4370	0.5613	0.8396
T3S [40]	Porto	0.4672	0.5977	0.8344	0.5518	**0.6560**	0.8550	0.4345	0.5809	0.8350
Efflex-B (Ours)		**0.5510**	**0.6492**	**0.9817**	**0.5564**	0.6273	**0.9750**	**0.5760**	**0.5892**	**0.9080**
Efflex-L (Ours)		**0.5651**	**0.7126**	**0.9984**	**0.5705**	**0.7139**	**0.9984**	**0.6412**	**0.7195**	**0.9965**
PCA [37]	Geolife	0.4110	0.5562	0.8243	0.4336	0.5880	0.8446	0.4331	0.5481	0.8190
SVD [33]	Geolife	0.4081	0.5563	0.8248	0.4438	0.6041	0.8448	0.4481	0.5285	0.8133
MDS [21]	Geolife	0.3602	0.5472	0.8535	0.4793	0.6187	0.8716	0.4656	0.5347	0.8354
Siamese [31]	Geolife	0.3120	0.4236	0.6640	0.4631	0.6032	0.8121	0.2680	0.4582	0.6172
NEUTRAJ [42]	Geolife	0.3691	0.4870	0.7416	0.4947	**0.6786**	0.8403	0.3067	0.4832	0.6513
T3S [40]	Geolife	0.3807	0.5463	0.7690	0.5231	0.6732	0.8667	0.3208	0.4316	0.6601
Efflex-B (Ours)		**0.5621**	**0.6464**	**0.9694**	**0.5828**	0.6439	**0.9601**	**0.6034**	**0.6271**	**0.9165**
Efflex-L (Ours)		**0.6030**	**0.7303**	**0.9929**	**0.6163**	**0.7425**	**0.9947**	**0.6975**	**0.7706**	**0.9970**

Table 1. **Quantitative performance compared with state-of-the-art models on graph representation learning.** We evaluate under three distance functions (Fréchet, Hausdorff, and DTW) with multiple evaluation metrics (hitting ratio: HR@10, HR@50, recall: R10@50) employed for each distance under two datasets: Porto and Geolife. **Red** / **Blue** numbers: Highest/Second highest among all methods.

Methods	**Dataset**	**Hausdorff**		**Fréchet**		**DTW**	
		Recall	**Time / s**	**Recall**	**Time / s**	**Recall**	**Time / s**
Efflex-L	Porto	0.9984	1913.48	0.9984	1819.01	0.9965	1382.78
Efflex-B	Porto	0.9817	51.61	0.9750	51.60	0.9080	52.76
Diff (B vs. L)	Porto	↓**1.67%**	↑ ×**36**	↓**2.34%**	↑ ×**34**	↓**8.85%**	↑ ×**25**
Efflex-L	Geolife	0.9929	1309.80	0.9947	1307.65	0.9970	1263.46
Efflex-B	Geolife	0.9694	78.89	0.9601	77.94	0.9165	69.28
Diff (B vs. L)	Geolife	↓**2.35%**	↑ ×**16**	↓**3.46%**	↑ ×**16**	↓**8.05%**	↑ ×**17**

Table 2. **Efficiency analysis.** We compare the recall and time cost (CPU) for pipeline training between Efflex-B/L under three distance functions on Porto and Geolife datasets. Efflex-B (with GCN) significantly improves the speed (up to ×36 faster) while maintaining a competitive accuracy against Efflex-L (with node2vec [13]).

Porto [30] contains 1,704,759 taxi trajectories gathered between 2013 and 2014 within Porto, Portugal. The recorded data encompass longitude coordinates ranging from -8.74 to -8.16 and latitude coordinates spanning from 40.95 to 41.31. Similarly, the Geolife dataset [51] offers a rich collection of GPS trajectories, capturing the movements of 182 users over five years (from April 2007 to August 2012). This dataset includes over 24,876 trajectories, which amounts to more than 1.2 million kilometers and a cumulative duration exceeding 48,000 hours. The geographical scope of this data spans several cities in China, with a longitude ranging from 115.9 to 117.1 and a latitude ranging from 39.6 to 40.7.

Considering the complexity and variability observed in real-world traffic trajectories, the Porto and Geolife datasets are ideal for evaluating model performance.

4.2. Implementation Details

The datasets are preprocessed by excluding trajectories with fewer than 50 data points. Then we partition the dataset into 50m×50m grids, following the standard operation [42]. We set the initial learning rate as 0.001, and use StepLR as the learning rate scheduler, which decreases by a factor of 0.1 every 5 epochs. The total training epochs are 50. The model's parameters are updated through the AdamW optimizer. Our experiments are conducted on AMD EPYC 7313 16-Core CPU and NVIDIA RTX A6000 GPU. To ensure a comprehensive assessment of runtime efficiency, we benchmark the performance of all compared algorithms using a CPU in single-threaded mode.

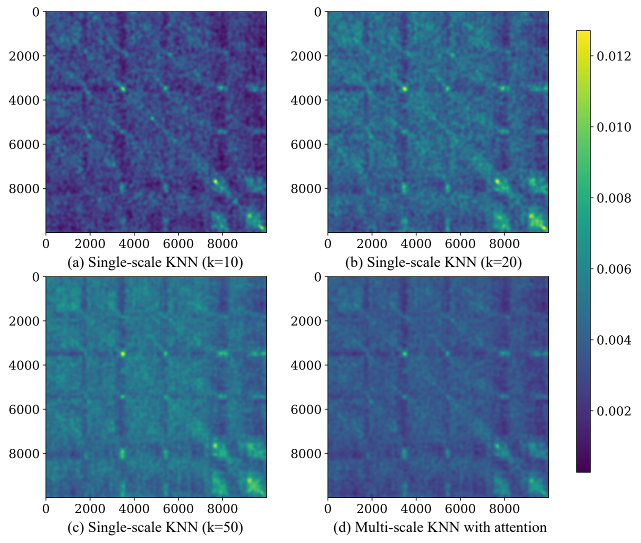


Figure 3. **Visualization of edge connection weights. Figure (a) - (c):** Edge connection weights obtained by single-scale KNN (k = 10, 20, 50). **Figure (d):** Connection weights fused by multi-scale KNN with the attention mechanism.

4.3. Evaluation Metrics

To conduct objective evaluation, we evaluate the model's performance on the top-N similarity search task problem, following the state-of-the-art models [40, 42]. Specifically, given a query trajectory, the model outputs its top-N similar trajectories based on the trajectory embeddings it learns under certain distance measurement function. The higher similarity search accuracy indicates the more accurate learned embeddings of the original trajectory dataset by the model.

Following the standard evaluation procedure [40, 42], two evaluation metrics are involved: hitting ratio (HR@10, HR@50) and recall (R10@50). We compare the results with both non-learning-based methods (PCA [37], SVD [33], MDS [21]) and state-of-the-art machine learning models (Siamese [31], NEUTRAJ [42], T3S [40]).

4.4. Quantitative Results

4.4.1 Representation Learning Performance

Table 1 shows the graph representation learning performance, where both our two solutions (base and large version) outperform all existing models and achieve the state-of-the-art. As mentioned previously, Efflex-B utilizes the lightweight GCN for graph representation learning, while Efflex-L employs node2vec [13] with massive trainable parameters for extracting graph embeddings.

Specifically, under Hausdorff distance, Efflex-B reaches the hitting ratio and recall of 55.10%, 64.92%, and 98.17%, while Efflex-L achieves a more accurate result: 56.51%, 71.26%, and 99.84%. In terms of the comparison with matrix factorization-based methods, Efflex-B/L demonstrates significant improvement over PCA ($+10.53/16.87\%$), SVD ($+10.56/16.90\%$), and MDS ($+4.27/10.61\%$) of HR@50. Similar results can be observed for other metrics (HR@10, R10@50). Similar results can be observed from Geolife.

Moreover, when compared with learning-based methods, Efflex-B/L consistently showcases its effectiveness, with a remarkable lead (HR@50 as an example) of $+14.93/21.27\%$, $+7.78/14.12\%$, and $+5.15/11.49\%$ against Siamese [31], NEUTRAJ [42], and T3S [40], respectively. Similar observations can be found in Geolife. Benefiting from the design of multi-scale KNN graph construction and fusion mentioned in Section 3.1, our model can selectively preserve the important global and local features when transforming the original trajectory dataset into the relatively low-dimension graph. In addition, the utilization of GCN allows our model to eventually converge for learning graph structures and capturing graph-level representations, leading to significant improvements compared with RNN [31] / LSTM [40, 42] based methods.

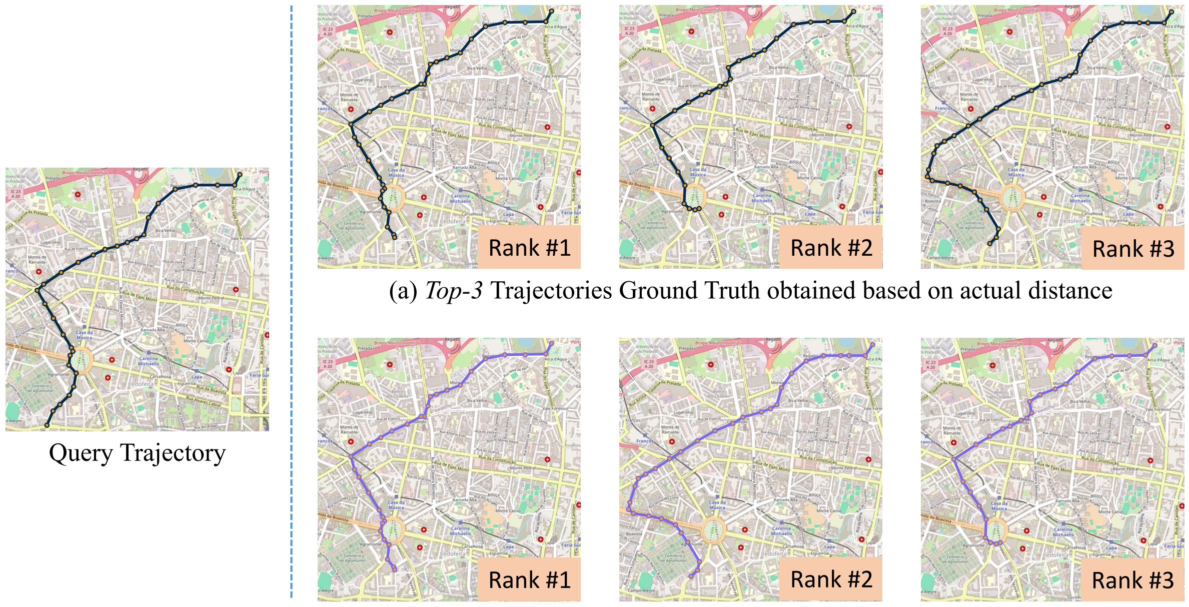
Similar observations can be concluded from Table 1 under either Fréchet or DTW measurement, proving the consistent effectiveness and robustness of our model across different evaluation metrics.

4.4.2 Efficiency Analysis

We compare the efficiency of the two versions of our model, Efflex-B/L, on Porto and Geolife datasets, as shown in Table 2, where both models are evaluated under the same CPU environment. We utilize recall (R10@50) as the embedding learning accuracy metric and compute the time cost under three distance functions. From Table 2, it is evidently observed that Efflex-B can reach a competitive accuracy while significantly reducing the time cost. Specifically, under Hausdorff distance, although Efflex-B slightly reduces the accuracy by 1.67%, it reaches the speed that is 36 times faster. For Fréchet distance, Efflex-B consistently showcases its effectiveness and efficiency, with an accuracy deducted by 2.34%, it also improves an extraction speed by 34 times faster. Similar results can be observed from DTW distance and in the Geolife dataset.

Benefiting from our lightweight but effective design of GCN, Efflex-B can effectively capture graph structure patterns for accurate representation learning while noticeably improving the training speed, compared with Efflex-L with node2vec [13] as the representation learning backbone. Since Efflex-B is a solely innovative design (without involving existing architecture [13] as Efflex-L), it strongly showcases our contributions to effectively learning graph representations on spatio-temporal data.

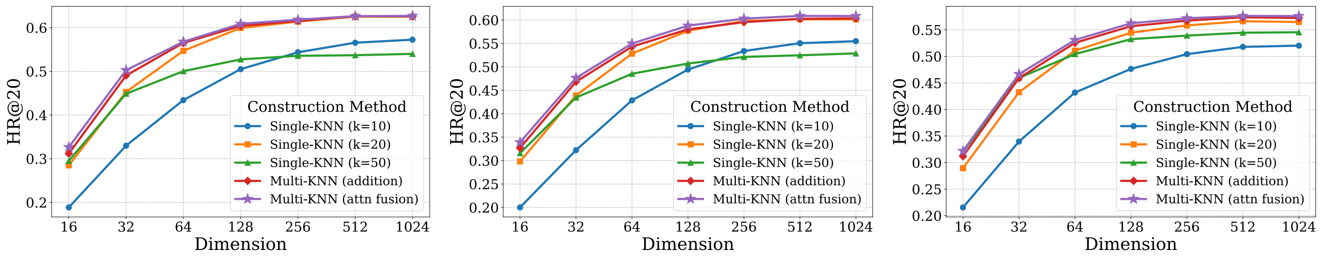
Moreover, the substantial improvement in modeling accuracy demonstrates the potential of Efflex-B for future ap-



(a) *Top-3* Trajectories Ground Truth obtained based on actual distance

(b) *Top-3* Trajectories output by Efflex-B

Figure 4. **Qualitative visualization of the trajectory similarity search task. Left:** Query trajectory. **Right (a):** *Top-3* similar ground truth trajectories. **Right (b):** *Top-3* similar retrieval results of our model (Efflex-B). Our retrieval results are consistent with ground truth.



(a) Hausdorff distance (b) Fréchet distance (c) DTW distance

Figure 5. **Ablation study on different structures and embedding dimensions under three distances on Porto.** We compare the results of single-scale KNN (k=10,20,50), multi-scale KNN (with different fusion strategies), and different output embedding dimensions.

plications on lightweight platforms such as mobile phones and wearable devices, and autonomous driving. As for application scenarios that do not require real-time modeling including basic environmental monitoring, our Efflex-L version can be widely used, considering its state-of-the-art learning accuracy.

4.5. Qualitative Results

The qualitative demonstration is illustrated in Figure 4, which shows the trajectory query results (Porto as an example). Specifically, given a query trajectory, the model will find the *top-3* similar trajectories based on the learned embeddings. The corresponding ground truth is retrieved by computing the actual distance in the dataset. From Figure 4, we can observe that the retrieval results of Efflex closely match the ground truth (i.e., Rank #1 in Ground Truth vs. Rank #1 in our results), proving the effectiveness of our model on the trajectory similarity search task. Meanwhile, the convincing results justify the theory of dimensionalizing the original spatio-temporal trajectories for graph construction, which provides an applicable solution to represent the massive spatio-temporal data efficiently. Additionally, the potential real-life application scenarios (i.e., finding alternative trajectories when the existing path cannot be traveled when navigating) show Efflex's application and generalization.

4.6. Ablation Studies

We conduct comprehensive ablation studies to verify the effectiveness of the model's key components and parameters, including attention for feature fusion, multi-scale graph construction, output embedding dimension, and loss terms.

Methods	Hausdorff	Fréchet	DTW
w/o Multi-KNN	0.9723	0.9447	0.8724
w/o Attn Fusion	0.9789	0.9733	0.9023
L1 Loss	0.9646	0.9632	0.8107
MSE Loss	0.9698	0.9658	0.8094
Efflex (Ours)	**0.9817**	**0.9750**	**0.9080**

Table 3. **Ablation studies on model structure and different losses under three distances on Porto.** Our final design showcases the most accurate embedding learning of the graph.

Attention for Feature Fusion. The ablation study on the proposed multi-scale KNN graph construction and fusion is shown in Figure 5, where we report the hitting ratio score against embedding dimensions under three distance functions. Specifically, we compare the results of the following: multi-scale graph construction and fusion with attention (final model), multi-scale construction and fusion through simple addition operation, and single-scale graph construction (k = 10, 20, and 50).

It is evident that our design (multi-scale KNN with attention) outperforms other structures for different embedding dimensions under all distance functions, with the purple line consistently superior to others. Since the proposed lightweight attention allows the model to selectively and dynamically capture the important features and graph structure information with different scales (global and local levels), it contributes to more accurate embedding learning than other simple fusion strategies (addition operation), with the purple line is higher than the red one. The quantitative analysis on attention fusion is demonstrated in Table 3, where "w/o Attn Fusion" refers to utilizing the simple addition operation to fuse features.

Multi-scale vs. Single-scale Graph Construction. In Figure 5, both purple and red lines remain at the highest levels under different conditions, indicating the effectiveness of multi-scale graph construction over single-scale one. The employment of the KNN algorithm with different scales allows the model to capture board relationships and localized patterns at the same time, leading to a more comprehensive understanding of the graph structure compared with single-scale KNN (k = 10, 20, and 50). The quantitative result is shown in Table 3, where "w/o Multi-KNN" indicates the single-scale KNN algorithm (k = 20 as an example).

Output Embedding Dimension. Since the original trajectory is transferred into low-level embeddings, different embedding dimensions determine the representation quality. We conduct the ablation analysis of different embedding dimensions ranging from 16 to 1024, as shown in Figure 5. As the embedding dimension increases, the model's performance shows a significant increase followed by a gradually stable trend. Increasing the dimension of the embeddings allows the representation space to be closer to the original high-dimensional space (spatio-temporal trajectory data), assisting the model in better representing the original data.

Loss Terms. The result of different losses for pipeline training is shown in Table 3, where cosine similarity distance outperforms L1 Loss and MSE Loss. Considering the ability of cosine similarity distance to ignore the effect of data sparsity and dimensionality [44, 47], it is less sensitive to outliers and variations in the magnitude of the spatio-temporal graph representation learning. This indicates it to be the ideal loss term for stable pipeline training.

5. Future Work

In the future, we plan to focus on optimizing the framework for even greater scalability and under real-time scenario settings. This would be particularly beneficial for applications requiring immediate insights from vast amounts of spatio-temporal data, such as traffic management systems and real-time environmental monitoring.

Meanwhile, another interesting direction we are looking into involves integrating Large Language Models (LLMs) as encoders to process graph features within the Efflex pipeline, as LLM holds the potential to conveniently capture the complex semantics of spatio-temporal data, translating it into richer, context-aware embeddings. Such an integration could enhance the predictive accuracy and analytical depth of the Efflex framework, opening new avenues in spatio-temporal data analytics.

6. Conclusion

In this paper, we introduce a novel framework that addresses the challenges of effectively learning representations from large-volume spatio-temporal trajectory data. Our comprehensive pipeline, Efflex, integrates a multi-scale KNN algorithm with feature fusion for graph construction, achieving significant advancements in dimensionality reduction while preserving essential data features. Furthermore, our custom-built lightweight GCN enhances the model's efficiency, enhancing the embedding extraction speed by up to 36 times faster without compromising accuracy.

We demonstrate Efflex's superior performance through extensive experimens with the Porto and Geolife datasets, establishing new benchmarks in the domain. Efflex is presented in two versions, Efflex-B and Efflex-L, tailored to scenarios demanding high accuracy and environments requiring swift data processing, respectively. This dual-version approach highlights our framework's adaptability and broad applicability, underscoring its potential in time-sensitive and computationally constrained applications.

References

[1] Sami Abu-El-Haija, Hesham Mostafa, Marcel Nassar, Valentino Crespi, Greg Ver Steeg, and Aram Galstyan. Implicit svd for graph representation learning. *Advances in Neural Information Processing Systems*, 34:8419–8431, 2021. 2

[2] Marta Avalos, Richard Nock, Cheng Soon Ong, Julien Rouar, and Ke Sun. Representation learning of compositional data. *Advances in Neural Information Processing Systems*, 31, 2018. 1, 2

[3] E Belogay, C Cabrelli, U Molter, and R Shonkwiler. Calculating the hausdorff distance between curves. *Information Processing Letters*, 64(1), 1997. 3

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 1, 2

[5] Ming Cheng, Xingjian Diao, Shitong Cheng, and Wenjun Liu. Saic: Integration of speech anonymization and identity classification. *arXiv preprint arXiv:2312.15190*, 2023. 1

[6] Ming Cheng, Bowen Zhang, Ziyu Wang, Ziyi Zhou, Weiqi Feng, Yi Lyu, and Xingjian Diao. Vetrass: Vehicle trajectory similarity search through graph modeling and representation learning, 2024. 1, 2

[7] Xingjian Diao, Ming Cheng, Wayner Barrios, and SouYoung Jin. Ft2tf: First-person statement text-to-talking face generation. *arXiv preprint arXiv:2312.05430*, 2023. 3

[8] Xingjian Diao, Ming Cheng, and Shitong Cheng. Avmaskenhancer: Enhancing video representations through audio-visual masked autoencoder. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 354–360. IEEE, 2023. 3

[9] Jiaxin Ding, Shichuan Xi, Kailong Wu, Pan Liu, Xinbing Wang, and Chenghu Zhou. Analyzing sensitive information leakage in trajectory embedding models. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, pages 1–10, 2022. 2

[10] Gintare Karolina Dziugaite and Daniel M Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015. 1, 2

[11] Maurice Fréchet. Sur quelques points du calcul fonctionnel. 1906. 3

[12] Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *ACM Transactions on Algorithms (TALG)*, 14(4):1–17, 2018. 3

[13] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. 2, 4, 5, 6

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 2

[15] Peng Han, Jin Wang, Di Yao, Shuo Shang, and Xiangliang Zhang. A graph-based approach for trajectory similarity computation in spatial networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 556–564, 2021. 2

[16] Hanjiang Hu, Zhijian Qiao, Ming Cheng, Zhe Liu, and Hesheng Wang. Dasgil: Domain adaptation for semantic and geometric-aware image-based localization. *IEEE Transactions on Image Processing*, 30:1342–1353, 2020. 1

[17] Jianying Huang, Jinhui Li, Jeill Oh, and Hoon Kang. Lstm with spatiotemporal attention for iot-based wireless sensor collected hydrological time-series forecasting. *International Journal of Machine Learning and Cybernetics*, pages 1–16, 2023. 2

[18] Zheng Huang, Jing Ma, Yushun Dong, Natasha Zhang Foutz, and Jundong Li. Empowering next poi recommendation with multi-relational modeling. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2034–2038, 2022. 1

[19] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3

[20] Hamidreza Alikhani Koshkak, Ziyu Wang, Anil Kanduri, Pasi Liljeberg, Amir M. Rahmani, and Nikil Dutt. SEAL: Sensing Efficient Active Learning on Wearables through Context-awareness. In *Proceedings of the IEEE/ACM Design, Automation and Test in Europe Conference*, Spain, 2024. DATE'24. 2

[21] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. Number 11. Sage, 1978. 1, 2, 5, 6

[22] Marzena Kryszkiewicz. The cosine similarity in terms of the euclidean distance. In *Encyclopedia of Business Analytics and Optimization*, pages 2498–2508. IGI Global, 2014. 4

[23] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 617–628. IEEE, 2018. 2

[24] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 2

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 4

[26] Jing Ma, Yushun Dong, Zheng Huang, Daniel Mietchen, and Jundong Li. Assessing the causal impact of covid-19 related policies on outbreak dynamics: A case study in the us. In *Proceedings of the ACM Web Conference 2022*, pages 2678–2686, 2022. 1

[27] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19 (3):303–342, 1993. 2

[28] Jens Meiler, Michael Müller, Anita Zeidler, and Felix Schmäschke. Generation and evaluation of dimension-reduced amino acid parameter representations by artificial neural networks. *Molecular modeling annual*, 7(9):360–369, 2001. 1

[29] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007. 2

[30] Luís Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. Time-evolving od matrix estimation using high-speed gps data streams. *Expert systems with Applications*, 44:275–288, 2016. 4, 5

[31] Wenjie Pei, David MJ Tax, and Laurens van der Maaten. Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713*, 2016. 2, 5, 6

[32] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000. 2

[33] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer, 2003. 1, 2, 5, 6

[34] Senzhang Wang, Jiannong Cao, and S Yu Philip. Deep learning for spatio-temporal data mining: A survey. *IEEE transactions on knowledge and data engineering*, 34(8):3681–3700, 2020. 1

[35] Ziyu Wang, Nanqing Luo, and Pan Zhou. Guardhealth: Blockchain empowered secure data management and graph convolutional network enabled anomaly detection in smart healthcare. *Journal of Parallel and Distributed Computing*, 142:1–12, 2020. 1

[36] Ziyu Wang, Zhongqi Yang, Iman Azimi, and Amir M Rahmani. Differential private federated transfer learning for mental health monitoring in everyday settings: A case study on stress detection. *arXiv preprint arXiv:2402.10862*, 2024. 1

[37] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 1, 2, 5, 6

[38] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019. 3

[39] Chengcheng Yang, Lisi Chen, Hao Wang, and Shuo Shang. Towards efficient selection of activity trajectories based on diversity and coverage. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 689–696, 2021. 2

[40] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. T3s: Effective representation learning for trajectory similarity computation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2183–2188. IEEE, 2021. 2, 5, 6

[41] Xinyu Yang, Haoyuan Liu, Ziyu Wang, and Peng Gao. Zebra: Deeply integrating system-level provenance search and tracking for efficient attack investigation. *arXiv preprint arXiv:2211.05403*, 2022. 1

[42] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *2019 IEEE 35th international conference on data engineering (ICDE)*, pages 1358–1369. IEEE, 2019. 2, 3, 5, 6

[43] Di Yao, Haonan Hu, Lun Du, Gao Cong, Shi Han, and Jingping Bi. Trajgat: A graph-based long-term dependency modeling approach for trajectory similarity computation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 2275–2285, 2022. 2

[44] Huaxiong Yao, Yang Huang, Jiabei Hu, and Wenqi Xie. Cosine similarity distance pruning algorithm based on graph attention mechanism. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 3311–3318. IEEE, 2020. 4, 8

[45] Yuanfan Yao, Ziyu Wang, and Pan Zhou. Privacy-preserving and energy efficient task offloading for collaborative mobile computing in iot: An admm approach. *Computers & Security*, 96:101886, 2020. 1

[46] Hongzhi Yin and Bin Cui. *Spatio-temporal recommendation in social media*. Springer, 2016. 1

[47] Yue Yu, Tong Xia, Huandong Wang, Jie Feng, and Yong Li. Semantic-aware spatio-temporal app usage representation via graph convolutional network. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–24, 2020. 8

[48] Cao Zhang, Xiaohui Zhao, Ziyi Zhou, Xingyuan Liang, and Shuai Wang. Doseformer: Dynamic graph transformer for postoperative pain prediction. *Electronics*, 12(16):3507, 2023. 1

[49] Lu Zhang, Weiqi Feng, Chao Li, Xiaofeng Hou, Pengyu Wang, Jing Wang, and Minyi Guo. Tapping into nfv environment for opportunistic serverless edge function deployment. *IEEE Transactions on Computers*, 71(10):2698–2704, 2021. 1

[50] Lu Zhang, Chao Li, Xinkai Wang, Weiqi Feng, Zheng Yu, Quan Chen, Jingwen Leng, Minyi Guo, Pu Yang, and Shang Yue. First: Exploiting the multi-dimensional attributes of functions for power-aware serverless computing. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 864–874. IEEE, 2023. 1

[51] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010. 4, 5

[52] Ziyi Zhou, Baoshen Guo, and Cao Zhang. Doseguide: A graph-based dynamic time-aware prediction system for postoperative pain. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 474–481. IEEE, 2021. 1

[53] Xin Zhu, Shuai Wang, Baoshen Guo, Taiwei Ling, Ziyi Zhou, Lai Tu, and Tian He. Sparking: A win-win data-driven contract parking sharing system. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*, pages 596–604, 2020. 1