TrajCogn: Leveraging LLMs for Cognizing Movement Patterns and Travel Purposes from Trajectories

Zeyu Zhou[1,2*], **Yan Lin**[3], **Haomin Wen**[4], **Shengnan Guo**[1,2], **Jilin Hu**[5], **Youfang Lin**[1,2] and **Huaiyu Wan**[1,2*]

[1]School of Computer Science and Technology, Beijing Jiaotong University, China
[2]Beijing Key Laboratory of Traffic Data Mining and Embodied Intelligence, Beijing, China
[3]Department of Computer Science, Aalborg University, Denmark
[4]Carnegie Mellon University, Pittsburgh, USA
[5]School of Data Science and Engineering, East China Normal University, Shanghai, China
{zeyuzhou, guoshn, yflin, hywan}@bjtu.edu.cn, lyan@cs.aau.dk, jlhu@dase.ecnu.edu.cn, haominwe@andrew.cmu.edu

Abstract

Spatio-temporal trajectories are crucial for data mining tasks, requiring versatile learning methods that can accurately extract movement patterns and travel purposes. While large language models (LLMs) have shown remarkable versatility through training on extensive datasets, and trajectories share similarities with natural language, standard LLMs cannot directly handle spatio-temporal features or extract trajectory-specific information. We propose TrajCogn, a model that effectively adapts LLMs for trajectory learning. TrajCogn incorporates a novel trajectory semantic embedder to process spatio-temporal features and extract movement patterns and travel purposes, along with a trajectory prompt that integrates this information into LLMs for various downstream tasks. Experiments on three real-world datasets and four representative tasks demonstrate TrajCogn's effectiveness.

1 Introduction

A spatio-temporal (ST) trajectory is a sequence of timestamped locations, represented as $\mathcal{T} = \langle (l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n) \rangle$. It tracks the movements of an individual or object in a geographical space. With the widespread use of mobile phones, car navigation systems, location-based services, and online map services, ST trajectories are being recorded and collected from various sources [Zheng *et al.*2008]. They enable a wide range of spatio-temporal data mining tasks and applications, including trajectory prediction [Feng *et al.*2018, Kong and Wu2018], POI recommendation [Chen *et al.*2022a, Chen *et al.*2025], travel time estimation [Wang *et al.*2018, Shen *et al.*2025], trajectory similarity measurement [Yao *et al.*2019, Fang *et al.*2022a, Li *et al.*2018a, Yang *et al.*2021a], and trajectory-user linking [Zhou *et al.*2018, Miao *et al.*2020].

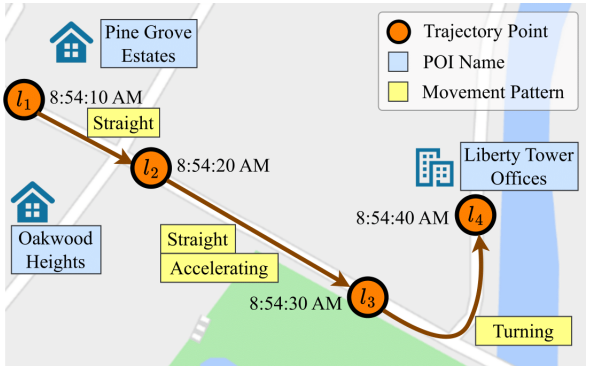

Figure 1: A trajectory of commuting to work.

To enhance the use of ST trajectories in tasks and applications, it is essential to develop a trajectory learning method that 1) effectively captures the information embedded in the trajectory, specifically, movement patterns that describe how the individual or object moves from one location to another and travel purposes that indicate the underlying reason or motivation for the movement; and 2) accurately performs a variety of downstream tasks, reducing the need for designing a separate method for each task and application. methods primarily use self-supervised learning [Dai and Le2015, Devlin *et al.*2019] to map trajectories into embedding vectors, training models from scratch [Yang *et al.*2023, Fu and Lee2020, Jiang *et al.*2023]. However, these models face limitations due to the complexity of trajectory information and constraints in model capacity, as well as the size and quality of available datasets.

On the other hand, versatile models have been highly successful in the domain of natural language processing (NLP), showcasing promising results on various downstream tasks [Radford *et al.*2019, Devlin *et al.*2019, Raffel *et al.*2020, Du *et al.*2022]. These models, often referred to as large language models (LLMs), benefit mainly from their large capacity, abundant large-scale corpus datasets, and well-thought-out prompt engineering [Brown *et al.*2020]. Given the simi-

*Corresponding author.

larities between trajectories and sentences in NLP, LLMs hold promise for enhancing trajectory learning models. Trajectory points share spatio-temporal correlations akin to word contexts, with movement patterns and travel purposes resembling word and sentence semantics. However, adapting LLMs for trajectory modeling presents challenges.

First, LLMs are incapable of processing the raw features in trajectories. LLMs are designed for discrete word sequences, but trajectories involve continuous and discrete spatio-temporal data like GPS coordinates and timestamps, which are difficult for LLMs to process. **Second, LLMs are unable to extract the movement patterns and travel purposes directly from trajectories.** For example, a moving object might go straight, accelerate, or turn, as shown in Figure 1. These movement patterns can be derived from changes in coordinates, timestamps, and velocities, while travel purposes are linked to the trajectory's origin and destination. The example trajectory which starts near residential buildings and ends near an office building suggests commuting. LLMs, however, focus on word semantics and are not equipped to interpret these spatio-temporal patterns or purposes.

To address these challenges and effectively leverage LLMs to construct a versatile trajectory learning model, we propose a novel approach named *Trajectory Cognition* (**TrajCogn**). TrajCogn employs a trajectory prompt to integrate movement patterns and travel purposes, and uses task-p-tuning to adapt to various tasks and make accurate predictions. It includes a trajectory semantic embedder to enable LLMs to process spatio-temporal features and extract movement patterns and travel purposes. Additionally, a cross-reconstruction pretext task based on self-supervised learning is implemented to enhance TrajCogn's ability to learn from trajectory data. Our contributions are summarized as follows:

- We introduce TrajCogn, a model that migrates LLMs to cognize movement patterns and travel purposes from trajectories, accurately performing different downstream tasks despite small dataset limitations.
- We develop a novel trajectory prompt to integrate movement patterns and travel semantics into LLMs, enhancing adaptability to various tasks.
- We propose a trajectory semantic embedder to process spatio-temporal features, allowing LLMs to extract movement patterns and travel semantics explainably.
- We conduct extensive experiments on three real-world trajectory datasets, and the results demonstrate TrajCogn's versatility and strong performance across different tasks.

2 Related Works

Trajectory Learning Models aim to extract information from trajectories and perform various related tasks. Compared to task-specific prediction models [Feng *et al.*2018, Yao *et al.*2019, Fang *et al.*2022a, Wang *et al.*2018, Chen *et al.*2022b, Li *et al.*2018b], which are end-to-end trained for one specific task, trajectory learning models are versatile and useful in modern intelligent transportation applications that usually involve multiple tasks.

Most existing efforts adhere to the self-supervised learning approach. Earlier research commonly used RNNs to reconstruct discrete locations [Li *et al.*2018a, Liu *et al.*2020, Fu and Lee2020] or continuous movement features [Yao *et al.*2017] of trajectories based on auto-encoding [Hinton and Salakhutdinov2006] and variational auto-encoders [Kingma and Welling2014]. Additionally, methods like CTLE [Lin *et al.*2021] and Toast [Chen *et al.*2021], based on transformers [Vaswani *et al.*2017] and Masked Language Model (MLM) tasks [Devlin *et al.*2019], treat trajectory points as tokens in a sentence. Furthermore, contrastive learning methods such as PIM [Yang *et al.*2021b], TrajCL [Chang *et al.*2023], and MMTEC [Lin *et al.*2023] implicitly model the travel purpose of a trajectory. More recently, methods combining multiple approaches have been developed. START [Jiang *et al.*2023] leverages both MLM tasks and SimCLR [Chen *et al.*2020], while LightPath [Yang *et al.*2023] incorporates a reconstruction task and a contrastive-style rational reasoning task.

Since these methods are self-supervised and trained from scratch, their performance heavily relies on the size and quality of the training datasets, which often have limitations. Despite the achievements of existing methods, further efforts are needed to enhance the performance of trajectory learning models.

Cross-domain Application of LLMs. The versatility and superior performance of large language models (LLMs) in the NLP domain have led to efforts to adopt LLMs in other fields to enhance performance. GPT4TS [Zhou *et al.*2023] uses LLMs by freezing the self-attention feed-forward layers. Time-LLM [Jin *et al.*2023] introduces a reprogramming framework. LM4VisualEncoding [Pang *et al.*2023] incorporates a frozen transformer block from an LLM as a general-purpose visual encoder layer. UrbanGPT [Li *et al.*2024a] employs spatio-temporal encoding combined with LLM instruction-tuning for generalized zero-shot prediction. FlashST [Li *et al.*2024b] introduces an innovative prompt-tuning framework designed to adapt pre-trained models for spatio-temporal prediction tasks.

Although these studies provide valuable insights, their methods cannot be directly applied to trajectory learning. Trajectory data has unique spatio-temporal features that require tailored approaches and considerations.

3 Preliminaries

3.1 Definition

Definition 1 (Road Network)**.** *A road network is represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is a set of $|\mathcal{V}|$ vertices, and each vertex $v_i \in \mathcal{V}$ represents an intersection between road segments or the end of a segment. $\mathcal{E}$ is a set of $|\mathcal{E}|$ segments, where each segment $s_i \in \mathcal{E}$ represents a road segment linking two vertices.*

Definition 2 (Trajectory)**.** *A trajectory $\mathcal{T}$ is a sequence of timestamped locations, represented as $\mathcal{T} = \langle (l_1, t_1), (l_2, t_2), \cdots, (l_n, t_n) \rangle$. Here, each location l_i is represented by its latitude and longitude coordinates, i.e., $l_i = (l_i^{\text{lat}}, l_i^{\text{lng}})$. The timestamp t_i indicates when l_i is visited. To simplify, we denote the i-th trajectory point (l_i, t_i) as τ_i.*

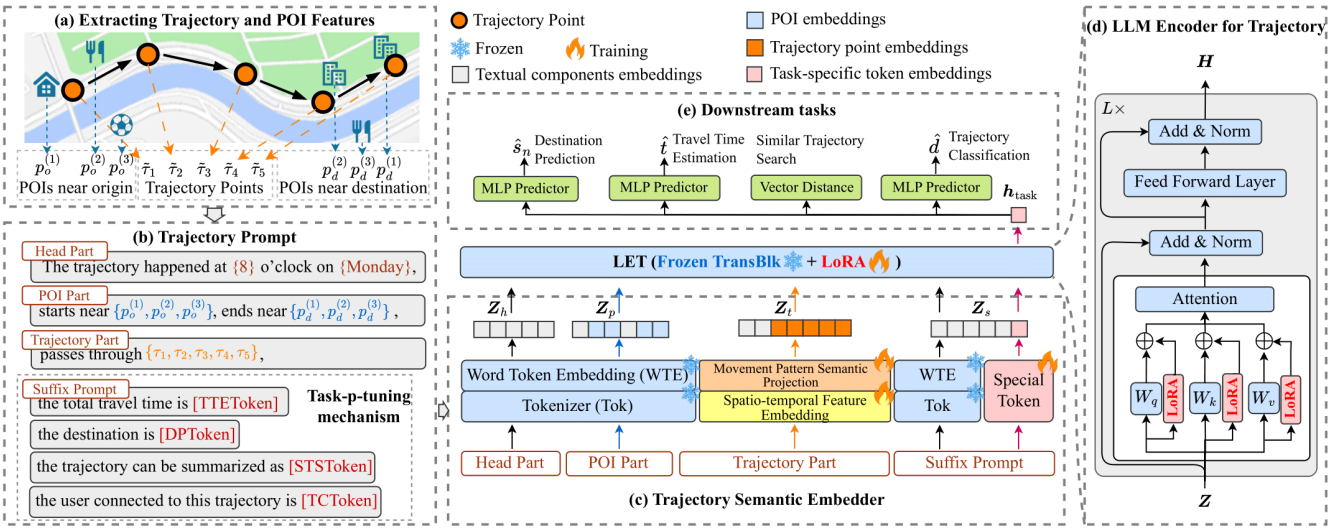

Figure 2: Overall framework of TrajCogn.

Definition 3 (Point of Interest, POI). *A POI is a particular location that individuals may find valuable or intriguing. It is denoted as $p = (l, n, a)$, where l represents its coordinates, n indicates its name, and a refers to its address.*

3.2 Problem Statement

Trajectory Learning. The objective is to develop a trajectory learning model f_Θ with a set of learnable parameters Θ. This model takes a trajectory $\mathcal{T}$ as input and extracts information from it. Subsequently, this model can adapt to various downstream tasks by accurately predicting the required outputs y for the task at hand, denoted as $\hat{y} = f_\Theta(\mathcal{T})$. For example, in travel time estimation, y and $\hat{y}$ represent the ground truth and the estimated travel time, respectively.

3.3 Pre-trained Language Model

In this work, a Large Language Model (LLM) refers to a Transformer-based language model pre-trained on corpus datasets. It consists of four essential functions. Formally,

$$\text{LLM} = \text{LMHead} \circ \text{TransBlk} \circ \text{WTE} \circ \text{Tok}(\cdot), \quad (1)$$

where $\circ$ represents the composition of functions. Specifically, a LLM consists of a tokenizer (Tok) to break down text into discrete tokens, a word token embedding layer (WTE) that converts the tokens into numerical vectors to capture their linguistic features, a transformer block (TransBlk) that further processes the vectors to capture their contextual relationships, and a prediction head (LMHead) that is respondible for making specific predictions, such as generating the next word in a sequence. In a LLM, the dimension of the word token embedding is denoted as d.

4 Methodology

4.1 Overview

Figure 2 shows the overall framework of TrajCogn. It is implemented in the following four steps:

1. **Trajectory and POI Feature Extraction**: Given a trajectory $\mathcal{T}$, we perform map-matching and calculate high-order features such as velocity, acceleration, and direction to expand its features, denoted as $\widetilde{\mathcal{T}}$. We also extract the address and name features of POIs near the trajectory's origin and destination.
2. **Trajectory Prompt Construction**: We integrate the extracted features into one sequence, called the trajectory prompt. This prompt also includes a task-p-tuning mechanism-based suffix to enable adaptation to various tasks.
3. **Trajectory Prompt Embedding**: We map the trajectory prompt into a sequence of d-dimensional embeddings with a trajectory semantic embedder. This embedder is designed to enable LLMs to process spatio-temporal features and effectively extract movement patterns and travel purposes with explainability.
4. **Model Training and Task Adaptation**: We process the embedding sequence with a *LLM Encoder for Trajectory* (LET). The last point of the output sequence of LET is used for performing downstream tasks. The learnable parameters in the model are refined by integrating a cross-reconstruction pretext task and further optimized with a dedicated objective function for each specific downstream task.

The following sections provide a detailed explanation of the steps in TrajCogn.

4.2 Trajectory Prompt

As illustrated in Figure 1, movement patterns in a trajectory can be represented by positions on the road network and variations in spatio-temporal features. Travel purposes can be inferred from the functionalities of locations near the OD points, and the address and name features of a POI indicate its functionalities.

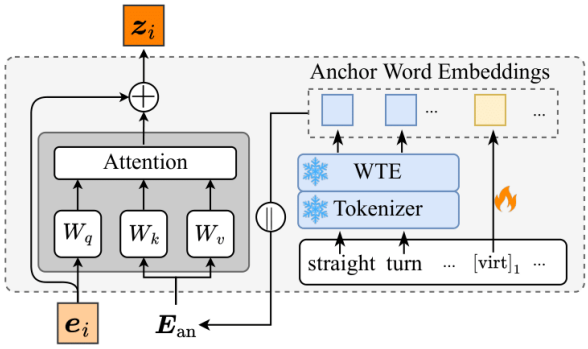

Figure 3: Movement pattern semantic projection.

To incorporate the movement patterns and travel purposes of a trajectory, we first extract spatio-temporal and POI features from the trajectory, as shown in Figure 2(a). To integrate these features into LLMs, we introduce a *Trajectory Prompt*, as illustrated in Figure 2(b). This prompt fuses natural language and the extracted features into a sequence. Furthermore, to adapt the model to different downstream tasks, we introduce a task-p-tuning mechanism, which provides a specific suffix for each task.

Trajectory and POI Feature Extraction

Given a trajectory $\mathcal{T} = \langle (l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n) \rangle$ and a road network $\mathcal{G}$, we apply the Leuven Map Matching (LMM) algorithm to map each trajectory point τ_i onto the road network, denoted as $\text{LMM}(\tau_i, \mathcal{G}) = (l_i, s_i, t_i)$, where s_i is the road segment containing l_i. We then calculate the velocity v_i, acceleration a_i, and direction θ_i of each point based on differences between consecutive points. The trajectory point $\tilde{\tau}_i = (l_i, s_i, t_i, v_i, a_i, \theta_i)$ is formed with these spatio-temporal features, setting the velocity and acceleration of the last point $\tilde{\tau_n}$ to 0. The final trajectory $\tilde{\mathcal{T}} = \langle \tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau_n} \rangle$ is obtained.

For POI feature extraction, we first identify the origin l_1 and destination l_n of the trajectory $\mathcal{T}$. Using the Ball Tree algorithm, we retrieve the closest N_{POI} POIs to l_1, denoted as $\mathcal{P}_O = \{p_o^{(1)}, \ldots, p_o^{(N_{\text{POI}})}\}$, ordered by distance from the origin. The same process is applied to retrieve POIs around l_n, denoted as $\mathcal{P}_D$. For each POI $p \in \mathcal{P}_O \cup \mathcal{P}_D$, we extract its address $p.a$ and name $p.n$ features, both represented as word lists.

Trajectory Prompt Construction

The trajectory prompt is made up of four parts. First, the ⟨Head Part⟩ introduces the context by stating *"The trajectory happened on* {*day-in-week*} *at* {*hour*} *o'clock."* Second, the ⟨POI Part⟩ provides details about the POIs near the origin and destination, saying *"starts near:* $\{p_o^{(1)}, p_o^{(2)}, \ldots, p_o^{(N_{\text{POI}})}\}$, *ends near:* $\{p_d^{(1)}, p_d^{(2)}, \ldots, p_d^{(N_{\text{POI}})}\}$*."* Third, the ⟨Trajectory Part⟩ includes the extracted features of the trajectory points, described as *"passes through* $\{\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_n\}$*."*

Finally, the ⟨Suffix Prompt⟩ is constructed using the task-p-tuning mechanism, which combines hard and soft components [Han *et al.*2022]. The hard component contains task-specific words, while the soft component is a task-specific token represented as [Token]. For example, for travel time estimation (TTE), the suffix prompt would read *"the total travel time is [TTEToken]."* For destination prediction (DP), it would be *"the destination is [DPToken]."* For trajectory classification, it would be *"the user connected to this trajectory is [TCToken]."*

Categories	Words
Driving Behaviors	straight, turn, u-turn, brake, accelerate, decelerate, stop, overtake, zigzag, swerve, detour, slide, cruise, glide, cautious, reckless, leisurely
Traveling Dynamics	steady, smooth, rough, constant, dynamic, fast, slow, rapid, rushed, erratic, agile, stationary, sluggish

Table 1: Words describing movement patterns.

4.3 Trajectory Semantic Embedder

In order to equip LLMs with the ability to process the spatio-temporal features in the trajectory prompt, we propose the *Trajectory Semantic Embedder*, demonstrated in Figure 2(c).

Spatio-temporal Feature Embedding

The spatio-temporal features in the trajectory prompt are embedded into a d-dimensional space. For discrete road segments s_i, an index-fetching embedding module $\boldsymbol{E}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d}$ is used. Timestamps t_i are embedded using modules $\boldsymbol{E}_{\text{dw}} \in \mathbb{R}^{7 \times d}$ and $\boldsymbol{E}_{\text{h}} \in \mathbb{R}^{24 \times d}$ for day-in-week and hour features, respectively. Continuous features of trajectory points are embedded using a one-dimensional convolution, inspired by previous studies [Wang *et al.*2018, Liang *et al.*2022], to model movement patterns. The continuous embedding vector of τ_i is obtained as follows:

$$\boldsymbol{E}_{\text{con}}(i) = \text{Conv1D}(\tau^{\text{con}}_{i-\lfloor \frac{k}{2} \rfloor : i+\lfloor \frac{k}{2} \rfloor}), \tag{2}$$

where k denotes the kernel size, $\tau_i^{\text{con}} = (l_i^{\text{lat}}, l_i^{\text{lat}}, v_i, a_i, \theta_i, t_i)$ presents the continuous features.

Finally, the embedding vector $\boldsymbol{e}_i$ of the i-th trajectory point τ_i is derived as follows:

$$\boldsymbol{e}_i = \boldsymbol{E}_{\text{con}}(i) + \boldsymbol{E}_{\mathcal{E}}(s_i) + \boldsymbol{E}_{\text{dw}}(t_i) + \boldsymbol{E}_{\text{h}}(t_i) \tag{3}$$

Movement Pattern Semantic Projection

To improve the model's understanding and interpretability of movement patterns, each embedding vector $\boldsymbol{e}_i$ is projected onto a semantic-rich textual space, as shown in Figure 3. This space is defined by a set of descriptive words $\mathcal{M}$ and a set of vitual words $\mathcal{A}$. The descriptive words are listed in Table 1 and the virtual words are initialized randomly and trained end-to-end. These words form anchor words, whose embeddings are concatenated into $\boldsymbol{E}_{\text{an}}$. Using multi-head attention [Vaswani *et al.*2017], where $\boldsymbol{e}_i$ serves as the query and $\boldsymbol{E}_{\text{an}}$ acts as both the key and value, $\boldsymbol{e}_i$ is projected onto this space, resulting in $\widetilde{\boldsymbol{z}}_i$. The final embedding vector $\boldsymbol{z}_i$ is obtained by adding a residual connection through a two-layer MLP:

$$\boldsymbol{z}_i = \text{MLP}(\widetilde{\boldsymbol{z}}_i) + \boldsymbol{e}_i. \tag{4}$$

The sequence of these embeddings for the trajectory is denoted as $\boldsymbol{Z}_{\mathcal{T}} = \langle \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n \rangle$.

POI Feature Embedding

The travel purpose is inferred by analyzing the functionalities of POIs near the origin and destination points. To model these functionalities, we derive embeddings for POIs based on their address and name features. For the closest POIs to the origin or destination, $p_o^{(1)}$ or $p_d^{(1)}$, embeddings are obtained by concatenating their address and name:

$$\boldsymbol{E}_{\text{Tok}}(p) = \text{WTE} \circ \text{Tok}(p.a \| p.n), \tag{5}$$

where $\|$ denotes list concatenation. For other POIs, only the name is used for embedding: $\boldsymbol{E}_{\text{Tok}}(p) = \text{WTE} \circ \text{Tok}(p.n)$.

Sequence of Trajectory Prompt Embeddings

After obtaining embeddings of spatio-temporal and POI features, the remaining textual components in the trajectory prompt are embedded using $\text{WTE} \circ \text{Tok}$. Then, we concatenate the embeddings into a sequence in the same order as their raw features appear in the prompt. For example, the embeddings of the trajectory part are obtained as follows:

$$\boldsymbol{Z}_t = \boldsymbol{E}_{\text{Tok}}(\text{"passes through"}) \| \boldsymbol{Z}_{\mathcal{T}} \tag{6}$$

The embeddings of the ⟨Head Part⟩, ⟨POI Part⟩, and ⟨Suffix Prompt⟩ are denoted as $\boldsymbol{Z}_h$, $\boldsymbol{Z}_p$, and $\boldsymbol{Z}_s$, respectively. Finally, the sequence of trajectory prompt embeddings is gathered as follows:

$$\boldsymbol{Z} = \boldsymbol{Z}_h \| \boldsymbol{Z}_p \| \boldsymbol{Z}_t \| \boldsymbol{Z}_s \tag{7}$$

4.4 LLM Encoder for Trajectory

We use the transformer block TransBlk from an LLM as the backbone for the proposed *LLM Encoder for Trajectory* (**LET**). To better adapt the pre-trained TransBlk to trajectory learning, we employ the Low Rank Adaptation (LoRA) algorithm [Hu *et al.*2022], adding extra parameters to TransBlk.

Construction of LET

As illustrated in Figure 2(d), all parameters in the TransBlk are kept fixed, while we introduce a new learnable parameter matrix in every self-attention block by applying LoRA algorithm. The proposed LET can be expressed as follows:

$$\text{LET} = \text{LoRA}(\text{TransBlk}) \tag{8}$$

LET takes the embedding sequence $\boldsymbol{Z}$ from Equation 7 as input, and outputs a sequence of hidden vectors $\boldsymbol{H} = \text{LET}(\boldsymbol{Z}), \boldsymbol{H} \in \mathbb{R}^{L \times d}$, where L represents the length of $\boldsymbol{Z}$.

Adaptation to Downstream Tasks

LET adapts to different downstream tasks using the task-p-tuning mechanism described in Section 4.2. Specifically, the hidden vector corresponding to the task-specific token, i.e., the L-th hidden vector $\boldsymbol{h}_{\text{task}} \in \mathbb{R}^d$ in $\boldsymbol{H}$, can be utilized to perform downstream tasks.

In this study, we present *Travel Time Estimation* (**TTE**), *Destination Prediction* (**DP**), *Similar Trajectory Search* (**STS**) and *Trajectory Classification* (**TC**) tasks for evaluation, as shown in Figure 2(e).

The TTE task aims to estimate travel time using spatial features and departure time, excluding time-related features. A two-layer MLP is used to predict travel time: $\hat{y}_{\text{TTE}} = \text{MLP}_{\text{TTE}}(\boldsymbol{h}_{\text{task}})$.

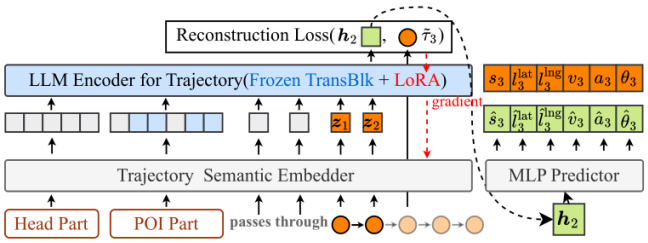

Figure 4: Reconstruction of trajectory points in cross-reconstruction pretext task.

The DP task aims to predict the destination road segment, excluding the last 5 trajectory points and nearby POIs to prevent data leakage. A two-layer MLP predicts the segment: $\hat{y}_{\text{DP}} = \text{argmax}_s(\hat{\boldsymbol{p}}), \hat{\boldsymbol{p}} = \text{Softmax}(\text{MLP}(\boldsymbol{h}_{\text{task}}))$.

The STS task aims to find the most similar trajectory using cosine similarity on $\boldsymbol{h}_{\text{task}}$. We construct ground truth by selecting 1,000 test trajectories, using odd-numbered points as queries $\mathcal{T}^q$ and even-numbered points as targets $\mathcal{T}^t$. We exclude the 10 closest trajectories and randomly select 5,000 others as negatives. Distances are calculated by downsampling to a uniform length and using mean square error, as per [Fang *et al.*2022b].

The TC task aims to classify trajectories by driver, using a two-layer MLP to predict the driver: $\hat{y}_{\text{TC}} = \text{argmax}_d(\hat{\boldsymbol{p}}), \hat{\boldsymbol{p}} = \text{Softmax}(\text{MLP}(\boldsymbol{h}_{\text{task}}))$

4.5 Model Training

We propose a cross-reconstruction pretext task to train the learnable parameters in the model, helping it adapt to trajectories. Before performing a specific task, the model can be further fine-tuned with supervision for that task.

Cross-reconstrution Pretext Task

The proposed pretext task involves reconstructing each trajectory point given ⟨Head Part⟩ and ⟨POI Part⟩, and reconstructing each POI given ⟨Head Part⟩ and ⟨Trajectory Part⟩.

As shown in Figure 4, for trajectory points, LET processes embeddings up to the $i-1$ step as $\boldsymbol{H}_{\text{traj},i-1} = \text{LET}(\boldsymbol{Z}_h \| \boldsymbol{Z}_p \| \boldsymbol{Z}_{t,:i-1})$, and predicting features with a two-layer MLP. The loss $\mathcal{L}_{\text{traj}}$ combines the cross-entropy loss of the predicted segments and the MSE loss of the predicted continuous features. For POIs, LET processes embeddings similarly as $\boldsymbol{H}_{\text{POI},i-1} = \text{LET}(\boldsymbol{Z}_h \| \boldsymbol{Z}_t \| \boldsymbol{Z}_{p,:i-1})$, predicting features with the LLM's LMHead. The loss $\mathcal{L}_{\text{POI}}$ is the cross-entropy of predicted features. The total pretext loss is

$$\mathcal{L}_{\text{pre}} = \mathcal{L}_{\text{traj}} + \mathcal{L}_{\text{POI}}. \tag{9}$$

Teacher-forcing is used to enhance training efficiency.

Task-specific Fine-tuning

When performing a specific task, the proposed model can be fine-tuned with the task's supervision to further improve prediction accuracy.

For the TTE task, the loss function is defined with mean square error (MSE) loss. For the DP and TC task, the loss function is defined with the cross-entropy loss. The STS task does not require fine-tuning, using the hidden state $\boldsymbol{h}_{\text{task}}$ from the pretext task directly.

Bold denotes the best result, and underline denotes the second-best result. ↑ means higher is better, and ↓ means lower is better.

Task		Travel Time Estimation			Destination Prediction			Similar Trajectory Search		
Datasets	Methods	RMSE (sec) ↓	MAE (sec) ↓	MAPE (%) ↓	ACC@1 (%) ↑	ACC@5 (%) ↑	Recall (%) ↑	Mean Rank ↓	ACC@1 (%) ↑	ACC@5 (%) ↑
Chengdu	Traj2vec	130.872 ± 2.013	59.993 ± 2.225	14.870 ± 0.698	43.074 ± 1.255	73.899 ± 1.568	14.760 ± 0.345	3.371 ± 0.156	83.325 ± 0.754	89.375 ± 0.459
	T2vec	128.508 ± 2.600	60.520 ± 2.575	15.224 ± 0.446	47.739 ± 0.239	73.509 ± 0.147	16.638 ± 0.108	3.345 ± 0.380	81.450 ± 0.778	93.700 ± 1.838
	TremBR	125.535 ± 2.849	57.965 ± 2.588	13.964 ± 0.860	48.987 ± 0.377	72.082 ± 0.289	17.010 ± 0.495	4.659 ± 1.010	83.980 ± 1.145	89.880 ± 0.303
	CTLE	132.636 ± 3.973	57.481 ± 1.144	13.153 ± 0.750	51.004 ± 0.683	79.434 ± 0.641	21.467 ± 0.704	9.429 ± 1.587	53.767 ± 7.414	69.200 ± 4.508
	Toast	128.793 ± 2.566	60.997 ± 3.537	14.883 ± 0.576	50.897 ± 0.495	79.664 ± 0.498	21.068 ± 0.383	5.944 ± 1.130	53.640 ± 2.244	71.600 ± 2.819
	TrajCL	120.211 ± 1.040	59.816 ± 1.841	14.741 ± 0.443	50.847 ± 0.249	79.693 ± 0.577	21.572 ± 0.324	1.198 ± 0.219	95.125 ± 5.022	98.875 ± 1.350
	START	122.205 ± 3.181	55.922 ± 2.397	12.717 ± 0.788	52.775 ± 0.311	80.423 ± 0.409	23.316 ± 0.310	1.089 ± 0.041	96.933 ± 2.060	99.900 ± 0.100
	LightPath	119.23 ± 2.367	55.614 ± 1.518	12.760 ± 0.854	49.154 ± 0.234	78.587 ± 0.583	20.660 ± 0.273	27.266 ± 3.544	74.267 ± 4.765	86.100 ± 3.874
	TrajCogn (ours)	**115.079 ± 1.608**	**51.973 ± 1.922**	**11.635 ± 0.587**	**59.594 ± 0.867**	**86.740 ± 0.294**	**30.184 ± 0.875**	**1.068 ± 0.044**	**99.240 ± 0.152**	**99.940 ± 0.060**

Task		Trajectory Classification			Destination Prediction			Similar Trajectory Search		
Datasets	Methods	ACC@1 (%) ↑	ACC@5 (%) ↑	Recall (%) ↑	ACC@1 (%) ↑	ACC@5 (%) ↑	Recall (%) ↑	Mean Rank ↓	ACC@1 (%) ↑	ACC@5 (%) ↑
Porto	Traj2vec	51.469 ± 1.221	95.464 ± 0.135	45.225 ± 2.185	19.403 ± 0.754	42.086 ± 0.735	4.832 ± 0.578	2.365 ± 0.304	90.320 ± 1.492	93.838 ± 0.960
	T2vec	50.880 ± 1.964	96.248 ± 0.044	49.191 ± 1.933	20.168 ± 0.246	43.230 ± 0.881	5.324 ± 0.483	1.683 ± 0.108	91.640 ± 0.913	95.133 ± 0.404
	Trembr	55.300 ± 0.960	95.581 ± 0.113	50.321 ± 1.703	20.472 ± 0.762	43.974 ± 0.153	5.626 ± 0.392	2.051 ± 0.317	89.908 ± 1.016	93.504 ± 0.537
	CTLE	85.162 ± 4.111	97.426 ± 0.361	66.663 ± 3.049	18.775 ± 0.917	40.718 ± 1.093	2.796 ± 0.655	5.589 ± 0.621	64.675 ± 4.565	82.750 ± 4.778
	Toast	51.102 ± 4.921	82.088 ± 5.294	27.604 ± 4.898	18.731 ± 0.353	39.673 ± 0.688	2.412 ± 0.273	6.013 ± 1.097	61.533 ± 3.886	80.367 ± 2.602
	TrajCL	88.012 ± 2.085	96.308 ± 0.342	76.217 ± 2.541	20.601 ± 0.121	45.876 ± 0.204	4.590 ± 0.200	1.196 ± 0.0462	93.500 ± 0.765	98.933 ± 0.408
	START	89.464 ± 3.085	97.122 ± 0.612	73.913 ± 3.183	21.377 ± 0.129	46.777 ± 0.158	5.750 ± 0.030	1.127 ± 0.016	96.500 ± 0.141	99.300 ± 0.212
	LightPath	88.294 ± 0.398	93.272 ± 0.033	86.241 ± 0.671	22.092 ± 0.102	47.131 ± 0.092	5.728 ± 0.491	5.678 ± 0.309	83.067 ± 2.594	90.578 ± 1.034
	TrajCogn (Ours)	**92.021 ± 1.145**	**97.443 ± 0.302**	**87.101 ± 2.325**	**22.609 ± 0.524**	**48.827 ± 0.364**	**6.459 ± 0.092**	**1.034 ± 0.007**	**99.600 ± 0.216**	**99.900 ± 0.083**

Table 2: Overall performance of methods on Chengdu and Porto.

Task	Travel Time Estimation			Destination Prediction			Similar Trajectory Search		
Methods	RMSE (sec) ↓	MAE (sec) ↓	MAPE (%) ↓	ACC@1 (%) ↑	ACC@5 (%) ↑	Recall (%) ↑	Mean Rank ↓	ACC@1 (%) ↑	ACC@5 (%) ↑
w/o PT	120.737 ± 0.634	54.951 ± 2.632	12.087 ± 0.980	57.455 ± 0.723	85.331 ± 0.161	28.390 ± 1.512	3.914 ± 0.033	88.000 ± 0.566	94.600 ± 0.707
w/o POI	116.132 ± 2.131	52.941 ± 4.453	12.080 ± 0.924	58.711 ± 0.215	86.128 ± 0.118	29.372 ± 0.666	1.092 ± 0.065	98.200 ± 2.115	99.325 ± 0.754
w/o Conv	117.038 ± 2.237	53.402 ± 3.175	11.836 ± 1.175	59.078 ± 1.054	86.200 ± 0.673	29.521 ± 1.477	1.137 ± 0.050	96.733 ± 1.823	98.700 ± 0.781
w/o PSP	115.454 ± 5.551	53.003 ± 2.363	12.265 ± 0.856	58.797 ± 0.698	86.166 ± 0.460	29.503 ± 0.779	1.256 ± 0.256	96.667 ± 2.214	98.367 ± 1.037
w/o $\mathcal{M}$	115.233 ± 0.509	52.790 ± 3.297	11.891 ± 0.794	58.930 ± 0.220	86.668 ± 0.324	29.626 ± 0.287	1.069 ± 0.022	98.525 ± 0.551	99.350 ± 0.100
TrajCogn (full)	**115.079 ± 1.608**	**51.973 ± 1.922**	**11.635 ± 0.587**	**59.594 ± 0.867**	**86.740 ± 0.294**	**30.184 ± 0.875**	**1.068 ± 0.044**	**99.240 ± 0.152**	**99.940 ± 0.060**

Table 3: Performance of variants of TrajCogn.

5 Experiments

5.1 Datasets

In our experiments, we use three real-world datasets called Chengdu, Xi'an and Porto. Chengdu and Xi'an datasets were released by Didi[1] and consist of GPS trajectories recorded by taxis in Chengdu and Xi'an, China. Porto is an open-source dataset released for a Kaggle competition[2]. Trajectories with fewer than 6 points were excluded. Road networks from OpenStreetMap[3] were used for map-matching. An overview of the dataset statistics is shown in Appendix A.

5.2 Comparison Methods

We compare the proposed method with several state-of-the-art trajectory learning methods. **Traj2vec** [Yao *et al.*2017] calculates features with sliding windows and trains with an auto-regressive pretext task. **T2vec** [Li *et al.*2018a] pretrains the model using a denoising auto-encoder to reconstruct trajecotories. **TremBR** [Fu and Lee2020] constructs an RNN-based seq2seq model for recovering road segments and times. **CTLE** [Lin *et al.*2021] utilizes a bi-directional Transformer with MLM tasks for location and hour predictions. **Toast** [Chen *et al.*2021] employs a context-aware node2vec model with MLM and sequence discrimination tasks. **TrajCL** [Chang *et al.*2023] introduces a dual-feature, attention-based encoder trained with InfoNCE loss. **START** [Jiang *et al.*2023] incorporates a time-aware encoder and GAT, trained with MLM and SimCLR-based contrastive tasks. **LightPath** [Yang *et al.*2023] uses a sparse path encoder with path reconstruction and cross-view contrastive tasks.

5.3 Settings

For each dataset, we divide the trajectories into training, validation, and testing sets in an 8:1:1 ratio, ordered by departure time. Pre-training for the cross-reconstruction task and embedding methods lasts 20 epochs, with early stopping for downstream predictors based on validation performance.

All models are implemented using PyTorch [Paszke *et al.*2019]. We choose GPT2 [Radford *et al.*2019] as the foundation LLM to develop our model and obtain addresses and names of POIs using Amap APIs[4]. Our source codes are available at https://github.com/Zeru19/TrajCogn. Key hyperparameters for TrajCogn are $N_A = 15$, $K = 5$, $r = 8$, and $N_{\text{POI}} = 3$, optimized based on Acc@1 and Recall for destination prediction on Chengdu's validation set. The Adam optimizer is used with a learning rate of 1e-4 for our method and 0.001 for others. Experiments run on Ubuntu 22.04 with Intel(R) CPUs and nVidia(R) TITAN RTX GPUs, repeated 5 times to report mean and deviation of metrics.

5.4 Performance Comparison

Overall Performance

Table 2 present a comprehensive comparison of the performance of all task-adaptable trajectory learning methods

[1]https://gaia.didichuxing.com/

[2]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

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

[4]https://lbs.amap.com/

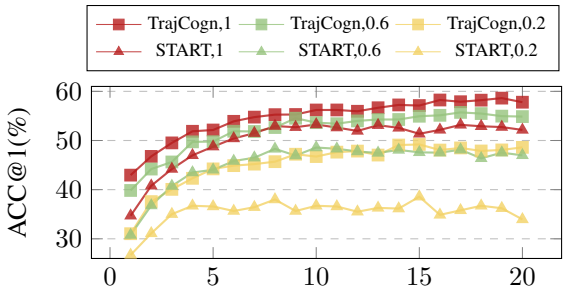

Figure 5: Scalability of fine-tuning on Chengdu.

Figure 6: Attention maps in the pattern semantic projection.

across four tasks and two datasets. The TTE task was excluded for Porto due to potential data leakage from equal-interval sampling, and trajectory classification was limited to Porto due to insufficient trajectories per driver in Chengdu. Our proposed method consistently outperforms the others and performs well across tasks, providing evidence that it is an advanced task-adaptable trajectory learning method.

Methods like Traj2vec, T2vec, and TremBR use RNN-based frameworks but lack crucial spatio-temporal features, leading to poor downstream performance. CTLE and Toast use bi-directional Transformers with MLM tasks [Devlin *et al.*2019] but miss essential continuous features and travel purposes, affecting their STS task performance. TrajCL, START, and LightPath use contrastive learning, aiding STS performance, but overlook POI functionalities and struggle with movement pattern extraction, resulting in unsatisfactory TTE and DP results. Our method utilizes LLMs for adaptable trajectory learning, effectively extracting movement patterns and incorporating POI functionalities and travel purposes through a trajectory prompt. These strengths lead to superior performance across tasks.

Scalability

To compare the scalability of the proposed model against START, one of the state-of-the-art models, we refine our model using varying proportions of the training data: 100%, 60%, and 20% for the destination prediction task on the Chengdu dataset. We use the START model as a reference point with an identical learning rate of 5×10^{-4} for comparison. The results are presented in Figure 5. It can be seen that our model demonstrates faster progress and achieves superior performance with less data compared to START. This shows that our model can be adapted to downstream tasks with lightweight finetuning.

5.5 Model Analysis

Effectiveness of Components

To evaluate components implemented in TrajCogn, we compared the performance of the complete model with the following variants. *w/o PT* excludes the cross-reconstruction pretext task, training directly on downstream tasks. *w/o POI* removes the ⟨POI Part⟩ from the trajectory prompt. *w/o Conv* replaces the convolution operator in the trajectory semantic embedder with a fully connected layer. *w/o PSP* excludes the pattern semantic projector, using only the trajectory point embedding $\boldsymbol{e}_i$; and *w/o* $\mathcal{M}$ omits the movement pattern vocabulary, relying solely on virtual anchor words.

We measured the performance of these variants on the Chengdu dataset, and the results are presented in Table 3. *w/o PT* shows performance degradation, proving the contribution of the cross-reconstruction pretext task to TrajCogn. The worse performance witnessed by *w/o POI* demonstrates the effectiveness of integrating POI information. *w/o Conv*, *w/o PSP*, and *w/o* $\mathcal{M}$ all have worse performance compared to *full*, showing that the removed components all contribute to TrajCogn's performance.

Attention Map Visualization

To demonstrate our model's ability to extract interpretable movement patterns, we visualize attention scores in the pattern semantic projector, as shown in Figure 6. Each subfigure shows the trajectory with marked points and subtrajectories on the left, and their attention maps on the right. We observed that trajectory points' movement patterns correspond to specific anchor words. Terms like "turn," "slow," and "steady" reveal the semantics of these patterns. For instance, turns increase attention for "turn," while "slow" and "stationary" suggest slow movement. However, these words do not always fully capture true movement semantics, highlighting the need for accurate labeled data for better alignment.

Additional Model Analysis

Further analysis of TrajCogn is detailed in the Supplementary Material. Appendix B covers its performance on the Xi'an dataset. Appendix C examines hyper-parameter effectiveness. Appendix D explores TrajCogn's efficiency. Appendix E discusses the impact of anchor words, while Appendix F looks at additional features. Appendix G analyzes the foundation LLM's effectiveness.

6 Conclusion

We propose TrajCogn, a novel trajectory learning model that leverages LLMs to model trajectories and accurately perform various trajectory-related tasks. TrajCogn introduces a trajectory prompt that combines movement patterns and travel purposes, enabling task adaptability. It also includes a trajectory semantic embedder for processing spatio-temporal features, allowing for effective and explainable extraction of movement patterns and travel purposes. Experiments on real-world datasets confirm TrajCogn's superior performance.

Ethical Statement

There are no ethical issues.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62272033).

Contribution Statement

Zeyu Zhou and Yan Lin contributed equally to this research.

References

[Brown *et al.*, 2020] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[Chang *et al.*, 2023] Yanchuan Chang, Jianzhong Qi, Yuxuan Liang, and Egemen Tanin. Contrastive trajectory similarity learning with dual-feature attention. In *ICDE*, pages 2933–2945, 2023.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, volume 119, pages 1597–1607, 2020.

[Chen *et al.*, 2021] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. Robust road network representation learning: When traffic patterns meet traveling semantics. In *CIKM*, pages 211–220, 2021.

[Chen *et al.*, 2022a] Wei Chen, Huaiyu Wan, Shengnan Guo, Haoyu Huang, Shaojie Zheng, Jiamu Li, Shuohao Lin, and Youfang Lin. Building and exploiting spatial–temporal knowledge graph for next poi recommendation. *Knowledge-Based Systems*, 258:109951, 2022.

[Chen *et al.*, 2022b] Zebin Chen, Xiaolin Xiao, Yue-Jiao Gong, Jun Fang, Nan Ma, Hua Chai, and Zhiguang Cao. Interpreting trajectories from multiple views: A hierarchical self-attention network for estimating the time of arrival. In *KDD*, pages 2771–2779, 2022.

[Chen *et al.*, 2025] Wei Chen, Haoyu Huang, Zhiyu Zhang, Tianyi Wang, Youfang Lin, Liang Chang, and Huaiyu Wan. Next-poi recommendation via spatial-temporal knowledge graph contrastive learning and trajectory prompt. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[Dai and Le, 2015] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *NeurIPS*, pages 3079–3087, 2015.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019.

[Du *et al.*, 2022] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. GLM: general language model pretraining with autoregressive blank infilling. In *ACL*, pages 320–335, 2022.

[Fang *et al.*, 2022a] Ziquan Fang, Yuntao Du, Xinjun Zhu, Danlei Hu, Lu Chen, Yunjun Gao, and Christian S. Jensen. Spatio-temporal trajectory similarity learning in road networks. In *KDD*, pages 347–356, 2022.

[Fang *et al.*, 2022b] Ziquan Fang, Yuntao Du, Xinjun Zhu, Danlei Hu, Lu Chen, Yunjun Gao, and Christian S. Jensen. Spatio-temporal trajectory similarity learning in road networks. In *KDD*, pages 347–356, 2022.

[Feng *et al.*, 2018] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *WWW*, pages 1459–1468, 2018.

[Fu and Lee, 2020] Tao-Yang Fu and Wang-Chien Lee. Trembr: Exploring road networks for trajectory representation learning. *ACM Trans. Intell. Syst. Technol.*, 11(1):10:1–10:25, 2020.

[Han *et al.*, 2022] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. PTR: prompt tuning with rules for text classification. *AI Open*, 3:182–192, 2022.

[Hinton and Salakhutdinov, 2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[Hu *et al.*, 2022] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[Jiang *et al.*, 2023] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *ICDE*, pages 843–855, 2023.

[Jin *et al.*, 2023] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[Kong and Wu, 2018] Dejiang Kong and Fei Wu. HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction. In *IJCAI*, pages 2341–2347, 2018.

[Li *et al.*, 2018a] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *ICDE*, pages 617–628, 2018.

[Li *et al.*, 2018b] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. Multi-task representation learning for travel time estimation. In *KDD*, pages 1695–1704, 2018.

[Li *et al.*, 2024a] Zhonghang Li, Lianghao Xia, Jiabin Tang, Yong Xu, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. Urbangpt: Spatio-temporal large language models. In *KDD*, pages 5351–5362. ACM, 2024.

[Li *et al.*, 2024b] Zhonghang Li, Lianghao Xia, Yong Xu, and Chao Huang. Flashst: A simple and universal prompt-tuning framework for traffic prediction. In *ICML*, 2024.

[Liang *et al.*, 2022] Yuxuan Liang, Kun Ouyang, Yiwei Wang, Xu Liu, Hongyang Chen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. Trajformer: Efficient trajectory classification with transformers. In *CIKM*, pages 1229–1237, 2022.

[Lin *et al.*, 2021] Yan Lin, Huaiyu Wan, Shengnan Guo, and Youfang Lin. Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. In *AAAI*, pages 4241–4248, 2021.

[Lin *et al.*, 2023] Yan Lin, Huaiyu Wan, Shengnan Guo, Jilin Hu, Christian S Jensen, and Youfang Lin. Pre-training general trajectory embeddings with maximum multi-view entropy coding. *IEEE Trans. Knowl. Data Eng.*, 2023.

[Liu *et al.*, 2020] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. Online anomalous trajectory detection with deep generative sequence modeling. In *ICDE*, pages 949–960, 2020.

[Miao *et al.*, 2020] Congcong Miao, Jilong Wang, Heng Yu, Weichen Zhang, and Yinyao Qi. Trajectory-user linking with attentive recurrent network. In *AAMAS*, pages 878–886, 2020.

[Pang *et al.*, 2023] Ziqi Pang, Ziyang Xie, Yunze Man, and Yu-Xiong Wang. Frozen transformers in language models are effective visual encoder layers. *arXiv preprint arXiv:2310.12973*, 2023.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[Shen *et al.*, 2025] Zekai Shen, Haitao Yuan, Xiaowei Mao, Congkang Lv, Shengnan Guo, Youfang Lin, and Huaiyu Wan. Towards an efficient and effective en route travel time estimation framework. *arXiv preprint arXiv:2504.04086*, 2025.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2018] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? estimating travel time based on deep neural networks. In *AAAI*, pages 2500–2507, 2018.

[Yang *et al.*, 2021a] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. T3S: effective representation learning for trajectory similarity computation. In *ICDE*, pages 2183–2188, 2021.

[Yang *et al.*, 2021b] Sean Bin Yang, Chenjuan Guo, Jilin Hu, Jian Tang, and Bin Yang. Unsupervised path representation learning with curriculum negative sampling. In *IJCAI*, pages 3286–3292, 2021.

[Yang *et al.*, 2023] Sean Bin Yang, Jilin Hu, Chenjuan Guo, Bin Yang, and Christian S. Jensen. Lightpath: Lightweight and scalable path representation learning. In *KDD*, pages 2999–3010, 2023.

[Yao *et al.*, 2017] Di Yao, Chao Zhang, Zhihua Zhu, Jian-Hui Huang, and Jingping Bi. Trajectory clustering via deep representation learning. In *IJCNN*, pages 3880–3887, 2017.

[Yao *et al.*, 2019] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *ICDE*, pages 1358–1369, 2019.

[Zheng *et al.*, 2008] Yu Zheng, Longhao Wang, Ruochi Zhang, Xing Xie, and Wei-Ying Ma. Geolife: Managing and understanding your past life over maps. In *MDM*, pages 211–212, 2008.

[Zhou *et al.*, 2018] Fan Zhou, Qiang Gao, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. Trajectory-user linking via variational autoencoder. In *IJCAI*, pages 3212–3218, 2018.

[Zhou *et al.*, 2023] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *NeurIPS*, 2023.