Efficient Learning-based Trajectory Generation for Predicting Future Road Network

by

Yunting Chen

A THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

Australian Artificial Intelligence Institute (AAII)
The School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney

November 2025

Certificate of Original Authorship

I, Yunting Chen, declare that this thesis is submitted in fulfillment of the requirements for the award of Master of Science, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship doi.org/10.82133/C42F-K220.

Signed:

Date: 2-Oct-2025

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Professor Ying Zhang and Professor Lu Qin, for their invaluable guidance, continuous support, and patience throughout the course of my master's study and research. Their profound knowledge and expertise in data mining, machine learning, and large-scale graph analysis have not only broadened my academic horizons but also provided me with the essential tools to pursue my research in time series generation and trajectory prediction. Their insightful advice and constructive feedback at every stage of my study have been instrumental in shaping both my academic progress and personal growth, and I am truly privileged to have had the opportunity to learn from them.

I am also sincerely grateful to the faculty members and staff of the Faculty of Engineering and Information Technology (FEIT) at the University of Technology Sydney (UTS) for providing a stimulating academic environment and for their kind assistance during my studies. The supportive atmosphere within FEIT, combined with the rich resources and research facilities available, has greatly enhanced my ability to explore deep learning frameworks such as Neural ODE, TimeGAN, and hybrid graph–language models, which form the core of my thesis work.

Special thanks go to my lab colleagues and friends, whose collaboration, discussions, and encouragement have made this journey both productive and enjoyable. Our many hours spent debating model architectures, debugging experiments, and refining ideas not only improved the quality of my research but also made the research process deeply rewarding.

I would also like to extend my appreciation to UTS for the financial support, which made my research possible and allowed me to fully dedicate myself to academic exploration.

Finally, I owe my heartfelt gratitude to my family for their unconditional love, encouragement, and understanding. Their constant support has been my greatest source of strength and motivation, enabling me to persevere through challenges and achieve this milestone in my academic journey.

Efficient Learning-based Trajectory Generation for Predicting Future Road Network

by

Yunting Chen

Abstract

Trajectory generation is one of the main application areas of spatio-temporal data mining in the current generative artificial intelligence era. Using generative algorithms, we can analyze and extract the inherent temporal and spatial laws of historical trajectory data, thereby achieving the goal of spatio-temporal road network prediction. In contemporary applications, trajectory simulation plays a vital role in domains like navigation systems and traffic route optimization. From a data modeling perspective, road trajectory information can be conceptualized as multi-dimensional time series incorporating temporal progression and spatial coordinates. Through an investigation of existing time series generation frameworks, we identified critical limitations and subsequently developed novel methodologies to address these gaps, thereby advancing the field of trajectory synthesis.

The first chapter is an introduction to the entire thesis. We elaborate on the background knowledge and motivation for studying trajectory generation based on deep learning. The existing models of time series generation and prediction in the past have application value, but lack a unified classification framework and evaluation metrics. After carefully studying various deep learning-based time series generation models, we found that different techniques have specific application defects. When processing data with different characteristics, the appropriate selection of model categories can achieve better performance. Then, we also introduced the outline of the rest parts and our contributions to coordinate the entire thesis.

The second chapter is an overview of generative time series models. We studied existing generative time series models and proposed a classification method based on the basic deep learning

framework. Under our proposed classification framework, generative time series models are categorized into four distinct types: those employing neural ordinary differential equations (Neural ODEs), those rooted in recurrent neural networks (RNNs), those based on variational autoencoders (VAEs), and those utilizing generative adversarial networks (GANs). This chapter provides a detailed analysis of the core features of each model type, with particular focus on their applicability to trajectory simulation tasks.

In Chapter 3, we introduced the generative time series model for anomaly detection. In this chapter, we first give a general summary of the content distribution of this chapter and then introduce the background knowledge and application areas of anomaly detection on time series data. In addition, we also introduced related works related to anomaly detection, mainly spoofing detection and dynamic graph learning for financial transactions. Next, we introduced the generative time series model and dynamic graph representation learning for anomaly detection and introduced the generative dynamic data encoding, pseudo-labeled graph generation, heterogeneous graph attention, and optimization objectives. In this chapter, we present the empirical evaluation of the proposed approach, encompassing components like experimental configurations and ablation analyses, to enhance comprehension of the model's overall functionality.

Chapter 4 introduces an efficient generative model for the simulation of trajectory data. In this chapter, we first introduce the background knowledge and application scenarios of trajectory data simulation. Then, we also introduce related works related to trajectory generation, including trajectory simulation methods and large language models. Next, we focus on a trajectory generation model we developed that combines a large language model and a graph neural network, and explain it from multiple perspectives, including model architecture, graph neural network encoder, prompt engineering, and model inference process. In addition, we also introduce the settings of the entire experiment and the comparison of experimental results.

In the last chapter, we summarize the previous chapters and discuss the contribution of this thesis to the field of trajectory generation and the prospects and possibilities for predicting future road networks. Starting from a data perspective, this thesis introduces the generative time series models proposed by predecessors and systematically classifies them. Then, combined with graph neural networks, the generative time series model is used as an important tool for anomaly detection models, which deepens the understanding of generative models. Finally, the large language model

with the ability to process sequential data is combined with the graph neural network model to creatively propose a trajectory generation model to predict future road networks. This thesis begins by examining existing research on time series generative models, with a focus on their practical applications in domains such as anomaly detection and urban traffic forecasting. Through this analysis, we highlight their significance while demonstrating their potential for real-world implementation in smart city contexts.

Publications

- Xiang S., Jiang Y., **Chen Y.**, Cheng D., Zhao G., Jiang C., "Generative Dynamic Graph Representation Learning for Conspiracy Spoofing Detection", *ACM The Web Conference (WWW 2025)* (**Chapter 3**)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, trajectory data has gained increasing significance across multiple domains, such as intelligent transportation systems, urban mobility planning, location-based services, and autonomous driving technologies. As the proliferation of GPS-enabled devices and mobile sensors continues to grow, the volume of spatio-temporal data capturing human and vehicle mobility has expanded dramatically [1]. This abundance of data presents new opportunities for predictive modeling and decision-making based on understanding and simulation of movement patterns across road networks.

A key challenge in this domain is the accurate and efficient generation of future trajectories over complex road infrastructures. Real-world trajectory generation involves capturing not only temporal dependencies, but also spatial constraints imposed by the underlying network topology. Conventional approaches, such as rule-based models or statistical forecasting methods, often fail to generalize to large-scale dynamic environments or capture the inherent uncertainty of real human mobility behavior [2]. Despite advances in deep learning technology, current models—though effective—are often limited by their inability to account for road network structures or face scalability challenges when deployed in dynamic environments [3].

To tackle these limitations, this research explores the design of a scalable deep learning framework tailored for trajectory generation, which effectively captures movement dynamics and forecasts path evolution within urban transportation environments. By integrating sophisticated generative

temporal modeling techniques, graph-structured representations for road networks, and an innovatively crafted architecture, this framework is designed to improve the fidelity, adaptability, and computational efficiency of trajectory simulation across real-time and batch processing scenarios.

This introductory section establishes the foundational context and significance of the research problem, thereby framing the subsequent discussion. Following this, later chapters will elaborate on the driving motivations behind the study, present an overview of the thesis organization, and highlight the key contributions introduced throughout this work.

1.1 Motivation

Predicting and generating future trajectories on real-world road networks is a core task in smart city infrastructure, with wide-ranging applications in transportation management, urban planning, and autonomous navigation. In rapidly developing metropolitan areas such as Chengdu, China, the ability to forecast urban mobility patterns is increasingly important for optimizing traffic systems, improving public safety, and improving service delivery [4]. However, designing a trajectory generation framework that is both accurate and efficient in such complex environments remains a challenging research problem.

This work is motivated by the following key challenges.

(1) Capturing Spatio-temporal Dependencies in Road Networks. Traditional time series models (e.g., ARIMA, HMM) and even many deep learning models (e.g., RNN, LSTM) focus primarily on temporal sequence patterns, often neglecting the spatial topology of road networks [5]. This results in trajectory predictions that may be temporally coherent but spatially infeasible, ignoring constraints such as road connectivity and directionality. In a real-world city like Chengdu, with thousands of intersections and complex road structures, modeling both spatial and temporal dependencies is crucial for realistic trajectory simulation.

(2) Enhancing Generalization and Adaptability for Wide Applicability in Different Traffic Scenarios. Many prior trajectory prediction models rely on fixed architectures tailored to specific urban settings, limiting their ability to generalize in diverse environments [6]. In reality, cities vary widely in terms of road network structures, traffic dynamics, and daily activity patterns, making it

difficult for a single rigid model to perform well in different scenarios. Enhancing generalization and adaptability is, therefore, critical for developing trajectory prediction systems that can be broadly applied in real-world applications, from dense metropolitan areas to more irregular or evolving urban layouts.

(3) Bridging the Gap Between Raw Data and Model Input for Improved Usability. Many existing trajectory prediction models rely on heavy preprocessing steps, such as one-hot encoding and matrix transformations, to make raw input data compatible with fixed model structures [7]. Although effective in controlled research settings, this approach reduces the interpretability of the model and increases the complexity of real-world deployment. The disconnect between raw trajectory data and model-ready input hinders seamless integration into interactive systems, where real-time data understanding and user feedback are crucial. Improving the readability and interaction of the model by aligning the input formats more closely with the representations of natural data is essential for practical and scalable trajectory prediction solutions.

These challenges underscore the necessity of an integrated approach that incorporates road network topology through graph neural networks while harnessing the analytical capabilities of large language models (LLMs). To resolve these issues, this study introduces an innovative framework that merges spatial reasoning via graph-aware representations with LLMs' generative potential, specifically optimized for complex, evolving urban environments.

1.2 Contributions

This thesis proposes a unified, scalable, and generalizable deep learning framework for trajectory generation on real-world road networks. The main contributions are as follows.

(1) A systematic and methodical framework for the classification of generative time series models. Based on research on existing time series generation models, this paper proposes a time series classification method based on deep learning and analyzes and summarizes various types of models, deepening the understanding of generative time series models [8]. In subsequent research on trajectory generation, researchers can look at past generative time series models from a new perspective and develop new models.

(2) A graph-aware generative framework to model spatio-temporal dependencies in urban road networks. To overcome the limitation of traditional sequence-based models that ignore spatial topology, we design a hybrid architecture that integrates graph neural networks (GNN) to capture the structure of the road network with large language models (LLM) for sequence generation [9]. This enables more spatially coherent and realistic trajectory prediction, especially in complex cities such as Chengdu.

(3) A flexible and adaptive model architecture for generalization in various urban traffic scenarios. Our method avoids rigid model structures and instead adopts topology-driven modular representations that can be adapted to different layouts and traffic patterns of the city [10]. This design significantly improves the generalizability and applicability of the model in various real-world settings.

(4) A usability-oriented design that bridges raw trajectory data with model inputs for practical deployment. To reduce the barrier between raw data and model compatibility, we streamline the data processing pipeline by minimizing preprocessing (e.g., avoiding excessive one-hot encoding) and aligning model inputs closely with natural data formats [11]. This improves both interpretability and interaction, enabling easier deployment in real-time or user-facing systems.

Chapter 2

An Overview of Generative Time Series Model

Generative time series models aim to learn the underlying temporal dynamics of observed sequences and generate new plausible sequences that follow similar patterns. In the context of trajectory prediction, these models serve as the core engine for simulating future mobility behaviors. This chapter reviews key developments in generative time series modeling, beginning with classical approaches and then focusing on recent deep learning-based methods that have advanced the field.

2.1 Classical Generative Time Series Models

This section introduces foundational time series models traditionally used for forecasting and simulation. Although these models laid the groundwork for temporal modeling, they often fall short when applied to complex, high-dimensional data such as urban trajectories.

2.1.1 Auto-regressive Models

Auto-regressive models have long served as foundational tools in time series forecasting. Among them, the **Autoregressive (AR)** [12], **Moving Average (MA)** [13], and **Autoregressive Integrated Moving Average (ARIMA)** [14] models are particularly well-established in both theory

and practice.

The **Autoregressive (AR)** model assumes that the current value of a time series can be expressed as a linear combination of its previous values. Mathematically, an AR(p) model is formulated as:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \varepsilon_t \tag{2.1}$$

where $\phi_1, \ldots, \phi_p$ are the model coefficients, and ε_t is a white noise error term. This model captures *temporal autocorrelation* through direct regression on past observations.

The **Moving Average (MA)** model, by contrast, represents the current observation as a function of past error terms. An MA(q) model is expressed as:

$$x_t = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{2.2}$$

Here, the parameters $\theta_1, \ldots, \theta_q$ weigh the contribution of past shocks or noise and μ is the mean of the series.

The **ARIMA** model generalizes these approaches by including a differencing step to handle non-stationary time series. An ARIMA(p, d, q) model first applies d-order differencing to the data, then fits an AR(p) and MA(q) model on the transformed series. This formulation enables ARIMA to model a broader class of time series with trends or other non-stationary behavior.

Despite their effectiveness and interpretability, these classical models rely heavily on assumptions of linearity and stationarity. They are best suited for time series with consistent statistical properties over time and limited long-range dependencies. As such, their application to complex domains such as trajectory prediction, which involve spatial constraints, multimodal behaviors, and abrupt transitions, is inherently limited.

In early mobility prediction tasks, ARIMA and its variants were applied to forecast vehicle or pedestrian locations based on time-indexed location data. However, these models struggle to capture nonlinear dependencies, contextual factors (e.g., road network structure or external events) and the structured output space typical in trajectory data. These limitations have motivated a shift

toward deep learning-based data-driven methods that can better model the complexity of human mobility patterns.

2.1.2 Hidden Markov Models

Hidden Markov Models (HMMs) [15, 16] are a class of probabilistic graphical models that have been widely used for sequential data modeling. An HMM is defined by a set of hidden states, a set of observable outputs, and a set of transition probabilities governing state evolution over time.

Formally, an HMM consists of:

- A finite set of hidden states $\{s_1, s_2, \ldots, s_N\}$
- A transition probability matrix $A = [a_{ij}]$, where $a_{ij} = P(s_{t+1} = j \mid s_t = i)$
- An emission probability distribution $B = P(x_t \mid s_t)$, specifying the likelihood of observation x_t given the current hidden state
- An initial state distribution $\pi = P(s_1)$

The model assumes that the hidden state sequence $\{s_t\}$ forms a first-order Markov chain, and that each observation x_t is conditionally independent of other observations and states, given the current state s_t.

HMMs have been successfully applied to various time series tasks, particularly in the context of activity recognition and trajectory mode inference. For instance, in mobility modeling, the hidden states can represent latent transportation modes (e.g., walking, cycling, driving), and the observable outputs can correspond to GPS features such as speed, heading, or location.

Despite their early success, Hidden Markov Models (HMMs) face significant limitations in modern trajectory analysis. Their reliance on discrete hidden states and simple emission distributions restricts their ability to capture complex or non-linear movement patterns. Furthermore, the low-resolution latent space often does not represent the high variability inherent in urban mobility. As trajectory data become more detailed and extended, HMMs also suffer from poor scalability due to their rigid structure and computational inefficiency.

These limitations have led researchers to explore more flexible and scalable alternatives, such as deep latent variable models and neural sequence architectures, which will be discussed in the following sections.

2.1.3 Gaussian Processes for Time Series Generation

Gaussian Processes (GPs) [17] are a class of non-parametric Bayesian models widely used for time series regression and stochastic function estimation. A GP defines a distribution over functions such that any finite set of time points follows a joint multivariate Gaussian distribution. This makes GPs particularly well-suited for interpolation and probabilistic forecasting tasks.

Formally, a GP is defined as:

$$f(t) \sim \mathcal{GP}(m(t), k(t, t')) \tag{2.3}$$

where $m(t)$ is the mean function and $k(t, t')$ is the covariance (or kernel) function that encodes the similarity between the time points t and t'. Given observed data $\mathcal{D} = \{(t_i, y_i)\}_{i=1}^{n}$, GPs can infer the posterior distribution over functions $f(t)$, allowing for both point predictions and uncertainty estimation at unobserved times.

This section highlights the notable strength of GPs in their capacity to assess prediction confidence, rendering them well-suited for critical domains where reliability is essential, such as autonomous navigation systems and healthcare surveillance. In addition, GPs offer smooth interpolation between observed time points and can model trajectories with a high degree of continuity and flexibility.

In the context of time-series generation, GPs have been used to simulate continuous spatio-temporal paths or to interpolate missing values in sparse or noisy sequences. Their nonparametric nature allows them to adapt flexibly to the data without requiring an explicit model structure.

Gaussian Processes (GPs), while powerful for modeling uncertainty, face several limitations that hinder their application in large-scale trajectory analysis. Their computational inefficiency, stemming from the need to invert large covariance matrices, makes exact inference impractical for

big datasets. Although approximate methods exist, they often trade off accuracy or require complex tuning. Furthermore, many standard kernels assume stationarity, which fails to capture the nonstationary and context-dependent nature of real-world trajectories.

As trajectory data in urban environments become increasingly large-scale, heterogeneous, and nonstationary, these limitations motivate the transition to deep learning-based generative models, which are more scalable and expressive in capturing complex spatiotemporal dependencies.

2.1.4 Summary of Classical Models

Classical time-series models such as ARIMA, Hidden Markov Models (HMMs), and Gaussian Processes (GPs) have provided foundational tools for sequence modeling and forecasting over the past decades. These methods offer strong theoretical underpinnings, interpretability, and well-understood inference procedures. In particular:

- AR, MA, and ARIMA models capture linear temporal dependencies and perform well on stationary time series with short-term dynamics.
- HMMs provide a probabilistic framework for modeling hidden state dynamics and have been widely used in activity recognition and mode-inference tasks.
- GPs offer flexible, nonparametric modeling with principled uncertainty estimates, suitable for smooth interpolation and regression.

Despite their strengths, these approaches face significant challenges when applied to modern trajectory generation tasks.

- **Linearity assumptions** limit their ability to capture non-linear dependencies inherent in human or vehicle mobility.
- **Poor scalability** makes them impractical for large-scale high-frequency datasets, such as GPS traces in urban environments.
- **Lack of structural awareness**, especially in spatial domains such as road networks, where trajectories must obey topological constraints.

As urban mobility data grows in complexity, richness, and volume, the need for more expressive and scalable generative models becomes apparent. These limitations have spurred an increasing interest in deep learning-based time series models, which can learn complex spatio-temporal patterns from raw data without strong parametric assumptions. The following section provides an overview of such approaches.

2.2 Deep Learning-based Generative Time Series Models

This section introduces data-driven generative models built on deep learning, which overcome many limitations of classical approaches. These models are capable of learning complex temporal patterns and can be extended to handle high-dimensional spatio-temporal data, such as urban trajectory sequences.

2.2.1 Neural ODEs and Their Extensions

Neural Ordinary Differential Equations (Neural ODEs) [18] represent a novel class of continuous-time deep learning models that learn the dynamics of hidden states as solutions to differential equations parameterized by neural networks. Unlike traditional recurrent models that evolve hidden states in discrete steps, Neural ODEs define the evolution via a continuous transformation governed by an ODE:

$$\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), t), \tag{2.4}$$

where $\mathbf{h}(t)$ is the hidden state at time t, and f_θ is a neural network parameterized by θ. Given an initial state $\mathbf{h}(t_0)$, the hidden state at any future time t_1 can be computed by solving the ODE:

$$\mathbf{h}(t_1) = \mathbf{h}(t_0) + \int_{t_0}^{t_1} f_\theta(\mathbf{h}(t), t)\, dt. \tag{2.5}$$

This continuous-time formulation provides several key advantages for time series modeling. It allows for flexible handling of irregularly sampled data, enables the representation of trajectories

as smooth and differentiable curves, and decouples the model architecture from rigid fixed-step update schemes, offering greater modeling flexibility and temporal resolution.

To further enhance modeling capacity, **Latent ODEs** [19] extend Neural ODEs into the generative modeling domain. In this framework, an encoder first maps observed sequences into a latent space, and then the latent dynamics are modeled using a Neural ODE. The generative process decodes latent states at arbitrary time points to reconstruct or generate future observations.

$$\mathbf{z}_0 \sim q(\mathbf{z}_0 \mid \mathbf{x}_{1:T}), \quad \frac{d\mathbf{z}(t)}{dt} = f_\theta(\mathbf{z}(t), t), \quad \mathbf{x}(t) = g_\phi(\mathbf{z}(t)), \tag{2.6}$$

where g_ϕ is the decoder network and $q(\cdot)$ is the variational posterior.

Neural ODEs and their extensions have shown promising results in continuous-time sequence modeling tasks, including human motion prediction, healthcare event modeling, and financial forecasting. However, they also come with several challenges.

- **Training instability**: ODE solvers require backpropagation through the integration process, which may lead to numerical instability or a high memory cost.
- **Computational inefficiency**: Solving ODEs can be slower than standard RNNs, especially for long or high-dimensional sequences.
- **Limited expressiveness**: Pure ODE dynamics may struggle to model sharp discontinuities or discrete state transitions.

Despite these limitations, Neural ODEs remain a powerful tool for modeling continuous and irregular time series data, and have laid the foundation for further research in time-aware generative modeling.

2.2.2 Variational Autoencoders and Their Extensions

Variational Autoencoders (VAEs) [20] are a class of probabilistic generative models that learn a latent representation of data through variational inference. VAEs are particularly attractive for

time series generation due to their ability to capture uncertainty in the latent space while providing an efficient sampling framework.

In the basic VAE framework, given observed data $\mathbf{x}$, the model assumes a latent variable $\mathbf{z}$ and defines a generative process:

$$\mathbf{z} \sim p(\mathbf{z}), \quad \mathbf{x} \sim p_\theta(\mathbf{x} \mid \mathbf{z}), \tag{2.7}$$

with an approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ learned using amortized variational inference. The model is trained by maximizing the Evidence Lower Bound (ELBO):

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] - \text{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})). \tag{2.8}$$

VRAE (Variational Recurrent Autoencoder) To model sequential data, the VAE framework has been extended to the Variational Recurrent Autoencoder (VRAE) [21]. VRAE integrates recurrent neural networks (typically LSTM or GRU) into both the encoder and the decoder:

- The encoder RNN maps the input sequence $\mathbf{x}_{1:T}$ to a global latent variable $\mathbf{z}$, capturing the overall dynamics of the sequence.
- The RNN decoder reconstructs the sequence from $\mathbf{z}$, treating it as a conditioning context for each time step.

VRAE enables global sequence-level generation and interpolation in the latent space. However, using a single global latent variable can limit its ability to model fine-grained temporal variation.

TimeVAE (Time-aware VAE) TimeVAE [22] is an advanced extension that explicitly models uncertainty in both time and latent dynamics. Introduces a time-conditioned latent representation, allowing for modeling of irregular sampling and temporal uncertainty. The encoder outputs not only a latent state $\mathbf{z}_t$, but also a time-dependent uncertainty σ_t, enabling more flexible trajectory generation.

TimeVAE also modifies the decoder to a continuous time condition t, such that:

$$\mathbf{x}_t \sim p_\theta(\mathbf{x}_t \mid \mathbf{z}_t, t), \tag{2.9}$$

This formulation makes TimeVAE suitable for applications such as continuous-time forecasting, missing data imputation, and irregular trajectory simulation.

Variational Autoencoders (VAEs) and their extensions offer a powerful probabilistic framework for sequence generation, combining several key strengths. They enable latent structure learning by capturing compact representations of complex sequences, support uncertainty quantification through probabilistic output, and scale efficiently to large datasets via amortized inference.

However, they also present some limitations:

- **Posterior collapse**: The decoder may ignore the latent code when it is too expressive.
- **Global latent bottleneck**: Basic VRAE uses a single latent variable for the entire sequence, limiting expressiveness.
- **Temporal structure modeling**: Requires careful design (e.g., hierarchical or time-aware latent spaces) to fully capture temporal dependencies.

Despite these limitations, VAE-based models form a foundational block in generative time series modeling and are often combined with more advanced architectures such as Neural ODEs and attention mechanisms to improve temporal resolution and flexibility.

2.2.3 GAN and Their Extensions

Generative Adversarial Networks (GANs) [23] have demonstrated remarkable success in generating high-quality data, particularly in computer vision. A standard GAN consists of two neural networks: a generator G that attempts to produce realistic samples, and a discriminator D that tries to distinguish between real and generated data. The two networks play a minimax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \tag{2.10}$$

In the context of time-series generation, applying GANs presents several challenges due to temporal dependencies, irregular sampling, and the need to preserve sequence dynamics. To address these issues, several GAN variants have been developed specifically for sequential data.

SeqGAN SeqGAN [24] is one of the first adaptations of GANs for sequence generation. Treats the generation process as a reinforcement learning (RL) task, where the generator is a stochastic policy that outputs sequences one token at a time. Because the output is discrete (e.g., symbols, locations), gradients from the discriminator cannot directly propagate through the generator.

SeqGAN tackles the challenge of sequence generation using reinforcement learning, specifically through policy gradient methods. Models the generator as a sequence model (such as an RNN) trained via the REINFORCE algorithm, while the discriminator provides reward signals based on whole sequences. To guide generation during training, Monte Carlo rollouts are employed to estimate expected rewards for incomplete sequences, enabling more effective learning from sparse feedback.

SeqGAN was originally designed for natural language generation, but has inspired similar techniques for trajectory and event sequence generation.

TimeGAN TimeGAN [25] is a state-of-the-art generative framework tailored for multivariate time series. Integrates adversarial training with supervised learning and latent dynamics modeling. The architecture is composed of four key components that work together to enable effective sequence generation and reconstruction. The embedding network maps observed sequences into a compact latent space, while the recovery network reconstructs sequences from these latent representations. A generator and a discriminator operate jointly in both the latent and observation spaces to guide the quality of the generation. Finally, the supervisor enforces temporal consistency by predicting the next step in the latent space, helping to preserve the sequential structure during training.

The overall objective combines reconstruction loss, supervised temporal loss, and two adversarial losses (in latent and real spaces). This hybrid design enables TimeGAN to preserve both temporal dynamics and distributional realism.

GAN-based time series models offer several compelling advantages for sequence generation. This approach facilitates high-quality generation, producing detailed and authentic sequences while avoiding reliance on predefined hypotheses about the distribution of underlying data. Additionally, models like TimeGAN enhance temporal structure awareness by incorporating supervised losses, allowing the network to better capture and preserve sequential dependencies during generation.

However, they also face significant challenges:

- **Training instability**: GAN training is known to be sensitive and difficult to stabilize.
- **Mode collapse**: Generators may fail to capture the full diversity of trajectories.
- **Complexity**: Architectures like TimeGAN are difficult to tune and require careful balancing of loss terms.

In summary, GAN-based models, especially TimeGAN, offer powerful frameworks for realistic time series generation, and are particularly suited for applications such as trajectory synthesis, anomaly detection, and simulation-based forecasting.

2.2.4 Sequence-to-Sequence RNN Models (LSTM, GRU)

Recurrent Neural Networks (RNNs) [26] form a fundamental category of models designed for sequential data analysis. These architectures retain an internal state that evolves dynamically, enabling the capture of temporal relationships within time-series inputs. Nevertheless, standard RNNs face challenges such as vanishing and exploding gradient phenomena, which hinder their capacity to effectively model long-range dependencies.

To overcome these limitations, enhanced versions like **Long-Short-Term Memory (LSTM)** [27] and **Gated Recurring Unit (GRU)** [28] have been introduced as improved alternatives.

LSTM and GRU. Both architectures incorporate gating structures that regulate the flow of information across temporal steps, thereby addressing the shortcomings of traditional RNNs.

- **LSTM**: Incorporates input, forget, and output gates along with a memory cell that enables learning of long-term dependencies.
- **GRU**: A simplified version with reset and update gates that offers comparable performance with fewer parameters.

These models have been widely used in mobility prediction, anomaly detection, and trajectory prediction tasks.

Given a time series $\mathbf{x}_{1:T}$, an LSTM or GRU processes the sequence iteratively:

$$\mathbf{h}_t = \text{RNNCell}(\mathbf{x}_t, \mathbf{h}_{t-1}), \tag{2.11}$$

where $\mathbf{h}_t$ is the hidden state at time t, summarizing all previous information.

Sequence-to-Sequence (Seq2Seq) Architecture. The Seq2Seq architecture extends RNNs to generate arbitrary length output sequences and is widely used for sequence generation tasks. It consists of two modules:

- **Encoder**: Consumes the input sequence and compresses it into a context vector (the final hidden state).
- **Decoder**: Generates the output sequence one step at a time, conditioned on the context vector and previously generated tokens.

For time series generation, the encoder learns a representation of past observations $\mathbf{x}_{1:T}$, and the decoder predicts future values $\mathbf{x}_{T+1:T+H}$.

Training and Generation Training is typically performed with teacher forcing, where the decoder receives the ground truth at each step. During inference, the model generates autoregressively by feeding back its own predictions.

$$\hat{\mathbf{x}}_{t+1} = \text{Decoder}(\hat{\mathbf{x}}_t, \mathbf{h}_t), \tag{2.12}$$

This process allows the model to generate multistep forecasts or simulate full-length trajectories.

RNN-based models are well-suited for time series modeling because of their ability to capture temporal dependencies in sequential data. They offer flexibility in handling variable-length inputs and outputs, making them adaptable to diverse tasks. Additionally, their relatively simple architecture makes them easier to train compared to more complex generative models, while still delivering strong performance in many sequence-learning applications.

However, they also present limitations:

- **Bottleneck effect**: The fixed-size context vector may limit capacity in long sequences.
- **Error accumulation**: Autoregressive decoding leads to compounding errors in long-horizon generation.
- **Limited global understanding**: Temporal patterns are captured locally unless enhanced with attention or memory mechanisms.

Overall, Seq2Seq RNNs provide a practical and widely adopted framework for time series generation, and serve as the basis for many hybrid and hierarchical models in more recent research.

2.2.5 Summary

In this section, we reviewed a range of deep learning models for time series generation, highlighting their core mechanisms and applicability to real-world sequence modeling tasks.

- **Neural ODEs and Latent ODEs** offer a principled framework for modeling continuous-time dynamics, making them particularly suitable for irregularly sampled time series. However, their training is computationally intensive and requires careful regularization.

- **Variational Autoencoders (VAEs)** provide a probabilistic latent space that enables the generation of diverse and structured time series. Extensions like VRAE and TimeVAE adapt this framework to sequential data but may produce blurry or overly smooth outputs due to variational averaging.
- **Generative Adversarial Networks (GANs)**—especially TimeGAN—achieve high-fidelity synthesis through adversarial learning and temporal supervision. Although powerful, these models suffer from training instability and require large datasets to generalize well.
- **Recurrent Neural Networks (RNNs), LSTM, and GRU** form the backbone of many sequence modeling systems. The Seq2Seq architecture allows flexible modeling of variable-length input-output sequences. However, RNNs tend to struggle with long-range dependencies and autoregressive error accumulation.

Each model class presents a different trade-off between expressiveness, scalability, and interpretability. Although traditional RNN-based models offer simplicity and fast training, latent variable and adversarial frameworks can generate more realistic and diverse sequences at the cost of higher complexity.

In the following chapter, we explore how these deep-generative models can be adapted and extended for the specific task of trajectory generation over road networks. We emphasize the integration of the model with spatial structures, long-horizon dependencies, and real-world constraints, paving the way for practical trajectory simulation and mobility prediction applications.

2.3 Time Series Generative Models for Trajectory Simulation

With the increasing demand for intelligent mobility systems and large-scale traffic forecasting, trajectory generation has become a central task in urban computing and transportation research. Unlike traditional time-series data, trajectories are structured spatiotemporal sequences that unfold over constrained physical spaces such as road networks. Generating future trajectories requires not only capturing complex temporal dependencies, but also adhering to topological constraints imposed by the underlying environment.

Recent advances in generative time series modeling, ranging from autoregressive recurrent networks to latent variable models and attention-based architectures, have significantly improved our ability to learn from sequential data. However, directly applying these models to trajectory generation presents new challenges. For example, most time-series models assume unconstrained output spaces, whereas valid trajectories must follow spatial paths and obey road connectivity. Moreover, trajectory data are often noisy, sparse, and multimodal, making reliable and diverse generation even more difficult.

This chapter explores how generative time-series models can be adapted to the trajectory generation task. We begin by formally defining the problem and the modeling objectives. Next, we analyze how deep generative models introduced in the previous chapter, such as VAEs, GANs, RNNs, and Transformers, can be repurposed or extended to generate plausible movement trajectories. We also highlight the key limitations of existing approaches when applied to real-world trajectory data, particularly in terms of spatial consistency, generative diversity, and efficiency. Finally, we discuss the design considerations that motivate the proposed framework introduced in the following chapter.

2.3.1 Trajectory Generation as a Time Series Problem

Trajectory data consist of sequences of spatial-temporal points representing the movement of objects over time. Formally, a trajectory can be represented as

$$\mathbf{T} = \{(x_t, y_t, t) \mid t = 1, 2, \ldots, T\}, \tag{2.13}$$

where (x_t, y_t) denotes the spatial coordinates at time t, and T is the trajectory length.

From a time series perspective, trajectory generation involves predicting future spatial positions given historical movement data. Unlike traditional univariate or multivariate time series, trajectory data have unique characteristics.

- **Spatial dependency**: Points are embedded in a road network or spatial environment, inducing structural constraints and dependencies.

- **Temporal dynamics**: Movements depend on speed, acceleration, and external factors that vary over time.
- **Nonlinear and multimodal behavior**: Trajectories may follow diverse routes with complex switching patterns.
- **Irregular sampling and missing data**: Real-world data often contain gaps and inconsistent sampling intervals.

These features pose challenges beyond standard time series generation, requiring models to integrate spatial structure and temporal evolution effectively.

Traditional time series models may fail to capture complex dependencies in trajectory data, while deep-generative models, particularly those combining graph neural networks with sequence models, offer promising solutions.

In this chapter, we review state-of-the-art approaches for time series generative modeling applied to trajectory data, focusing on methods that address spatial-temporal coupling and long-term forecasting accuracy.

2.3.2 Adapting Generative Models to Trajectory Data

Trajectory data pose unique challenges that require careful adaptation of existing generative time series models. While many generative models have demonstrated success in domains such as speech, finance, and healthcare, their direct application to trajectory generation is nontrivial due to the spatial-temporal complexity and structural constraints inherent in movement data. This section discusses key strategies and model adaptations to effectively leverage classical and deep generative frameworks for trajectory prediction.

Incorporating Spatial Structure A primary adaptation involves embedding spatial information into generative models originally designed for purely temporal data. For example, Graph Neural Networks (GNNs) have been integrated as spatial encoders to capture the underlying topology of road networks, which traditional sequence models lack. By learning node and edge representations

reflecting connectivity and road attributes, GNN-augmented generative models can better respect spatial constraints when predicting future positions.

Handling Non-Stationarity and Irregular Sampling Classical time series models often assume stationarity and regular time intervals, assumptions violated in real trajectory data due to varying traffic conditions and GPS sampling irregularities. Adaptations include incorporating time embeddings or continuous-time modeling approaches such as Neural Ordinary Differential Equations (Neural ODEs) to handle irregular timestamps and evolving dynamics. These approaches enable models to more naturally account for non-uniform temporal gaps and changing patterns in trajectories.

Modeling Multimodal Future Trajectories Movement behavior at decision points (for example, intersections) is inherently multimodal, with several plausible routes forward. To capture this uncertainty, generative models have been extended with probabilistic latent variables, as in Variational Autoencoders (VAEs), or adversarial training paradigms, such as Generative Adversarial Networks (GANs). These methods generate diverse trajectory samples that reflect the stochasticity of real-world navigation, improving robustness over deterministic predictions.

Integrating Temporal and Spatial Dependencies Hybrid architectures combining spatial encoders (e.g., GNNs) and temporal decoders (e.g., LSTMs, Transformers) have been proposed to jointly model the spatial-temporal dependencies in trajectory data. This integration requires a carefully designed information flow between graph-based spatial embeddings and sequence-based temporal generation, ensuring that the spatial context informs each step of the trajectory prediction.

Scaling to Large Urban Networks Scalability considerations have driven adaptations such as hierarchical graph modeling, sparse attention mechanisms, and mini-batch training strategies tailored for large-scale urban road networks and high-frequency trajectory streams. These modifications allow generative models to maintain computational efficiency while preserving accuracy.

In summary, adapting generative time series models to trajectory data demands the integration of spatial priors, the handling of temporal irregularities, and the modeling of complex, multi-modal movement patterns. The following chapters present specific applications of these adaptation principles in the detection of financial anomalies and simulation of urban trajectory.

2.4 Challenges in Trajectory Generation from Time Series Models

Trajectory generation as a time series modeling task involves unique challenges that differentiate it from traditional sequence generation problems. This section analyzes these challenges, emphasizing difficulties encountered when adapting and applying generative time series models to trajectory data.

Complex Spatial-Temporal Dependencies Unlike standard time series, trajectory data intrinsically couple spatial and temporal dimensions. The next position depends not only on the temporal sequence of past locations, but also on spatial constraints imposed by the topology of the road network. Traditional time series models, which mainly focus on temporal correlations, struggle to incorporate rich spatial information, leading to inaccurate or infeasible trajectory predictions.

Non-Stationarity and Heterogeneity Trajectory data often exhibit strong nonstationarity due to factors such as varying traffic conditions, weather, and human behavior changes. In addition, trajectories differ substantially between individuals, vehicles, or regions, resulting in heterogeneous data distributions. Time-series models assuming stationarity or homogeneous patterns fail to generalize well, requiring more flexible architectures that adapt to dynamic contexts.

Irregular Sampling and Missing Data Real-world trajectory data sets are rarely evenly sampled. GPS devices may record positions at irregular intervals or encounter signal loss, leading to missing data points. Time series models that assume fixed sampling rates or complete data struggle under these conditions. The handling of irregular temporal gaps and the imput of missing values pose significant challenges for model design and training.

Multimodality of Future Trajectories At many decision points, there are multiple plausible future paths. For example, a vehicle approaching an intersection may turn left, right, or continue straight. Modeling this inherent multimodality is difficult for deterministic generative models, which tend to produce averaged or unrealistic trajectories. Probabilistic or adversarial methods that capture uncertainty and generate diverse samples are necessary but introduce training complexity.

Long-Term Dependency and Error Accumulation Predicting long trajectories requires capturing dependencies over extended time horizons. Recurrent and autoregressive models are prone to error accumulation, where small prediction errors compound over steps, degrading long-term accuracy. Advanced architectures such as Transformers or Neural ODEs partially alleviate this but remain computationally intensive.

Scalability and Efficiency Urban-scale trajectory forecasting necessitates models capable of effectively managing extensive datasets characterized by intricate spatial networks and extended temporal sequences. Balancing model complexity, accuracy, and computational cost is a persistent challenge, especially for real-time or online applications.

Interpretability and Domain Constraints Trajectory predictions must often adhere to physical and traffic rules (e.g., no off-road movement, obeying traffic signals). Incorporating such domain knowledge explicitly into black-box deep generative models is difficult, limiting interpretability and trustworthiness in critical applications.

In conclusion, these challenges highlight the need for specialized time series generative models that jointly model spatial-temporal dynamics, handle data irregularities and multimodality, and scale efficiently while respecting domain constraints. Addressing these issues remains an active research area critical for advancing trajectory prediction technologies.

Chapter 3

Generative Time Series Model for Anomaly Detection

While previous chapters have focused on trajectory generation tasks, the modeling principles of generative time series models extend far beyond spatial data. One critical domain where these models can be particularly powerful is anomaly detection in financial transaction sequences. In such contexts, detecting unusual or fraudulent behavior is not only technically challenging due to the rarity and subtlety of anomalies but also highly consequential in practice.

Anomaly identification in financial systems demands frameworks capable of capturing normal temporal dynamics while detecting deviations from established patterns. Conventional approaches—such as rule-based systems or supervised classifiers—often struggle with cross-user generalization and adapting to evolving fraud tactics. In contrast, deep-generative methods like Variational Autoencoders (VAEs) and probabilistic recurrent architectures provide an unsupervised, data-centric approach that effectively models the underlying distribution of typical transaction behavior.

This chapter introduces a generative time series model designed to detect anomalies in financial transaction logs. Our method captures temporal patterns across sequences using a VAE-based architecture trained to reconstruct normal transaction trajectories. At inference time, anomalous events are flagged on the basis of a high reconstruction error or low likelihood under the learned distribution. The approach is validated on real-world financial datasets, where it successfully identifies both common and subtle forms of fraud with high precision and recall.

In doing so, this chapter demonstrates the flexibility of generative time series models in handling a different class of problems, those where deviations from learned patterns must be flagged while reusing many of the core architectural components discussed in earlier chapters. It also highlights how domain-specific constraints, such as temporal irregularity and financial semantics, can be integrated into a generative modeling pipeline.

3.1 Relative Works

3.1.1 Spoofing Detection

Since spoofing poses a major threat to the stability of financial markets, a variety of detection methodologies have been introduced in recent years to detect this issue. The early approaches relied mainly on statistical analysis of transaction behaviors to identify spoofing patterns. Machine learning-based methods, such as Linear Regression, Decision Tree and Multi-Layer Perceptron were broadly implemented in real-world spoofing detection systems. As spoof tactics evolved, researchers started to adopt deep learning techniques, such as recurrent neural networks (RNNs) and attention-based neural networks, to capture more complex trading patterns. Although these approaches offered advances, they still had limitations in capturing the intricate interactions within transactions. More recently, graph neural networks (GNNs) have gained attention for their effectiveness in spoof detection by leveraging interconnected behavior within transaction graphs. For example, RTG-Trans combines deep graph learning with temporal analysis to enhance spoof detection. However, many GNN models focus primarily on local context, relying on adjacent nodes, which restricts their ability to identify complex, coordinated spoofing activities that require a global perspective on transaction relationships. In addition to graph-based techniques, other approaches have also been developed to tackle spoofing detection from different angles. Rule-based algorithms, for example, set specific parameters to identify suspicious trading patterns. A good example of this can be found in studies that analyze transactions between stocks in indices such as the Ibovespa, as shown in. In addition, researchers have also explored spoofing from a microstructural perspective. They have introduced variables such as multilevel imbalances in price action and have delved into the optimization strategies that potential spoofers might employ, as described in. Despite these continuous advances, existing methods still struggle when it comes to capturing

the temporal relationships in trading behaviors. These relationships are inherently dynamic and do not fit well with traditional RNNs or transformer-based approaches.

3.1.2 Graph Learning for Financial Transaction

Machine learning approaches leveraging graph structures have proven essential for assessing financial transactions and identifying fraudulent behaviors. These methodologies benefit from their established efficacy in domains such as computer vision, text analysis, and semantic network construction [29–33]. In financial applications, graph models have proven adept at addressing complex tasks such as credit risk assessment [34] and financial fraud detection [35]. For example, vulnerable nodes on the guarantee loan network can be detected by graph models [36]. A significant challenge in the assessment of credit risk for small and medium-sized enterprises is the limited sample size. To address this, Wang *et al.* [37] proposed an adaptive heterogeneous multiview graph learning model, integrating multiple data perspectives to aggregate heterogeneous information for a comprehensive evaluation of credit risks. Based on the idea of using diverse data sources, SemiGNN utilizes labeled and unlabeled data to capture dependencies between data views and neighboring nodes through a hierarchical attention mechanism [38]. Another major challenge is that fraudulent activities often involve sophisticated tactics, such as disguising features and relationships within the data. To address this, Dou *et al.* [39] introduced CARE-GNN, which incorporates a label-aware similarity measure and a reinforcement learning-driven neighbor selection strategy, dynamically focusing on relevant nodes based on label information to improve detection accuracy. In addition to relational data, temporal patterns play a crucial role in identifying anomalies [40]. GADBench offers a benchmark framework for sequence-based anomaly detection, capturing user behavior patterns over time, facilitating the detection of anomalies within temporal transaction data[41]. While these approaches address challenges in fraud detection broadly, they neglect the specific requirements of spoofing detection, particularly the need to distinguish between non-homophily graph structures. To date, no dedicated solutions have been proposed to model the heterogeneous graph relationships unique to spoofing scenarios.

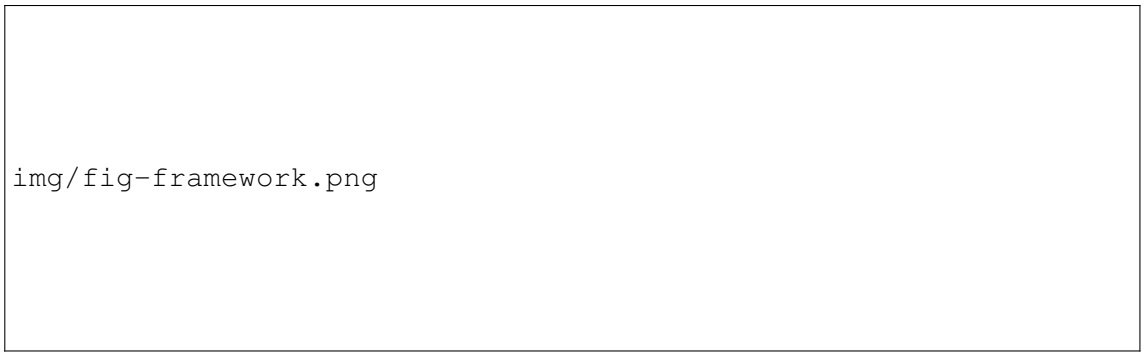

FIGURE 3.1: The proposed Generative Dynamic Graph Model (GDGM) architecture for Conspiracy Spoofing Detection. The first part is the historical transaction encoding of input time series data, which builds the embedding of irregular transaction series. The second part is the temporal graph attention. The third part is the heterogeneous graph attention layer, which aggregates the information for different types of neighbors. The fourth part is the classification layer, which is a multi-layer perception that gives the prediction of whether this transaction is spoofing.

TABLE 3.1: Notations used in this paper.

Symbol	Definition
n	the total number of nodes
m	the total number of edges
r	the number of relationships in graph $\tilde{G}$
d	the number of dimension
Z_i	the transformed representation using the i-th wavelet kernel
H	the comprehensive node representation aggregated from Z_0 to Z_C
p_i	the anomaly probability for node i
$X = \{x_1, x_2, ..., x_n\}$	the set of node features
$\hat{y}_i$	the pseudo-label for node i

3.2 Methodology

In this section, according to Figure 3.1, the proposed framework, GDGM, consists of four main components: (1) Generative Dynamic Data Encoding, (2) Pseudo-Labeled Graph Generation, (3) Heterogeneous Graph Attention, and (4) Classification. These components work together to capture generative dynamics data representations and heterogeneous graph relationships in financial transaction data, effectively modeling conspiracy spoofing behaviors.

3.2.1 Generative Dynamic Data Encoding

To effectively model the irregular temporal dynamics inherent in financial transaction data, especially in spoofing detection, we first explicitly leverage the timestamp of each transaction. Then, we employ a generative representation learning method, namely a Neural Ordinary Differential Equation-Based Recurrent Neural Network (ODERNN). This approach combines the continuous-time generative modeling capability of ODEs with the sequential representation power of GRU cells. The encoding process captures both temporal irregularities and sequential dependencies in the trading data.

Neural ODE Dynamics

We set the initial hidden state $h(0)$ as zero tensors. Then, the temporal evolution of the hidden state $h(t)$ in the ODE module is generated and governed by:

$$\frac{dh(t)}{dt} = f_\theta(h(t), t), \tag{3.1}$$

where f_θ is a neural network parameterized by θ that defines the dynamics of the system. The hidden state $h(t)$ at a future time t_1 is obtained by solving the initial value problem:

$$h(t_1) = h(t_0) + \int_{t_0}^{t_1} f_\theta(h(t), t)\, dt, \tag{3.2}$$

which is approximated numerically using the ODE solver:

$$h(t_1) = \text{ODEInt}(f_\theta, h(t_0), [t_0, t_1]), \tag{3.3}$$

where the function ODEInt computes the current hidden state, that is, the solution of the ODE over the time interval $[t_0, t_1]$.

GRU-based Sequential Updates

For each sequence, we initialize the hidden state as h_0. Then we iteratively update it using observations $\mathbf{x}_i$ at each time step with a gated recurrent unit (GRU) [42, 43]. The sequence update

combines ODE dynamics with a GRU to handle the final observations, which is formulated as:

$$h(t_j) = \text{ODEInt}(f_\theta, h(t_{j-1}), [t_{j-1}, t_j]), \tag{3.4}$$

$$h_j = \text{GRUCell}(\mathbf{x}_j, h(t_j)), \tag{3.5}$$

where $h(t_j)$ is the hidden state after solving the ODE, and h_j is the updated state that incorporates observation $\mathbf{x}_j$ through the GRU cell.

Mini-batch Encoding

Given a batch of transaction sequences, the model processes each transaction independently. The final hidden states $h_{i,T}$ for all transactions are aggregated into a batch representation, which is formulated as follows:

$$H_{\text{batch}} = \{h_{1,T}, h_{2,T}, \ldots, h_{N,T}\}, \tag{3.6}$$

where $h_{i,T}$ represents the last hidden state of the i-th sequence, and N is the batch size. The last hidden state will be used as input for the GNN-based classifier. In general, the neural ODE is responsible for modeling dynamics and generating candidates for hidden variables. Then the gated recurrent unit cells are leveraged to generate neural representations incorporated with temporal information.

3.2.2 Pseudo-Labeled Graph Generation

To effectively utilize unlabeled data, we propose a dynamic labeling mechanism integrated with the Pre-Trained Beta Wavelet Graph Neural Network (BWGNN) [44]. Leveraging its robust classification capabilities, BWGNN serves as the backbone for generating and refining pseudo labels, ensuring an effective and adaptive labeling process that enhances the model's robustness and generalization.

The process begins with the transformation of the node features X through a Multi-Layer Perceptron (MLP), which captures the intrinsic node characteristics. The raw node features X_{raw} were concatenated by the last hidden state H_{batch} of the encoding module, which is formulated as: $X = X_{\text{raw}}||H_{\text{batch}}$. These features are then processed by a set of Beta wavelet kernels $\mathcal{W}_{i,C-i}$,

each designed to extract spectral information at specific frequency scales, resulting in transformed representations Z_i:

$$Z_i = \mathcal{W}_{i,C-i}(\text{MLP}(X)), \quad i \in \{0, 1, \ldots, C\}. \tag{3.7}$$

The outputs from the wavelet kernels are aggregated into a comprehensive node representation H:

$$H = \text{AGG}([Z_0, Z_1, \ldots, Z_C]), \tag{3.8}$$

where AGG combines multi-scale spectral features. This representation is further processed by another MLP with a Sigmoid activation to compute anomaly probabilities p_i for each node:

$$p_i = \text{Sigmoid}(\text{MLP}(H)). \tag{3.9}$$

Based on these probabilities, pseudo labels are generated by applying a threshold z, which is formulated as follows:

$$\hat{y}_i = \begin{cases} 1 & \text{if } p_i > z \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

The transaction labeling mechanism provides a fully labeled graph for each batch of nodes to have graph data for heterogeneous graph attention during the training process.

3.2.3 Heterogeneous Graph Attention

In multi-relational graphs, redundant features can hinder the learning process by introducing noise. To address this challenge, we adopt a heterogeneous graph attention mechanism that consists of two main components: **intra-attention** and **inter-attention**. This approach efficiently handles the complexity of multi-relational transaction graphs by attending to information both within individual nodes and between different types of nodes, ensuring all meaningful relationships are captured.

Intra-Attention

The intra-attention mechanism is designed to consolidate features from neighboring nodes connected by the same relation type. This ensures that the model can choose important neighbors and

capture shared characteristics within each type of node. The process includes the following steps:

1. *Attention-based Neighbor Aggregation* within each relation:

$$h'^{,l}_{v,r,*} = \sum_{u \in \mathcal{N}_{r,*}(v)} \alpha_{vu,r} h^{l-1}_u \tag{3.11}$$

The attention weights $\alpha_{vu,r}$ are calculated using the softmax function to normalize the importance of neighboring nodes:

$$\alpha_{vu,r} = \frac{\exp\left(\text{LeakyReLU}\left(a_r^\top \cdot \left[W_r^l h_v^{l-1} \parallel W_r^l h_u^{l-1}\right]\right)\right)}{\sum_{k \in \mathcal{N}_{r,*}(v)} \exp\left(\text{LeakyReLU}\left(a_r^\top \cdot \left[W_r^l h_v^{l-1} \parallel W_r^l h_k^{l-1}\right]\right)\right)} \tag{3.12}$$

Here, a_r is a learnable vector specific to the relation r, W_r^l is a relation-specific weight matrix, and $\parallel$ denotes concatenation.

2. *Feature Update* using attended neighbor features:

$$h^l_{v,r,*} = \text{ReLU}\left(W^l_{\text{intra},r} \cdot \left(h_v^{l-1} \parallel h'^{,l}_{v,r,*}\right)\right) \tag{3.13}$$

where $W^l_{\text{intra},r}$ is the learnable weight matrix for intra-attention at layer l, h_v^{l-1} represents the central node's features at the previous layer, $h'^{,l}_{v,r,*}$ is the aggregated feature from the neighbors under relation r, and ReLU introduces non-linearity to the updated features.

The intra-attention mechanism focuses on leveraging homogeneous node relationships among the same type of nodes to refine the neighbor node feature representations.

Inter-Attention

After introducing intra-attention, the model proceeds to inter-attention, where it collects information across different types of nodes. The process is as follows: (1) Features within each type are aggregated using a mean operation, ensuring each type's features are combined based on their relationships; (2) The attended features from different types of nodes are then passed through an attention mechanism to compute the attention weights, allowing the model to focus on the most informative relationships between the central node and different types of relationships.

The inter-attention process is performed as follows:

1. *Mean Aggregation* within each relation:

$$h'^{,l}_{v,r,g} = \text{Agg}_{\text{mean}}(h^{l-1}_u), \quad \forall u \in \mathcal{N}_{r,g}(v) \tag{3.14}$$

where $\mathcal{N}_{r,g}(v)$ represents the set of neighboring nodes of v connected by relation r and group g, Agg_{mean} denotes the function to compute the mean values of neighboring node features h^{l-1}_u, and h^{l-1}_u is the feature of neighbor u at layer $l-1$.

2. *Inter-Attention* between different relations:

$$h^l_v = \sum_r \sum_g \alpha^l_{r,g} h^l_{v,r,g} \tag{3.15}$$

where h^l_v is the updated feature of the central node v at layer l, $h^l_{v,r,g}$ is the feature aggregated from relation r and group g, and $\alpha^l_{r,g}$ is the attention weight for relation r and group g. The attention weights $\alpha^l_{r,g}$ are computed using the softmax function:

$$\alpha^l_{r,g} = \frac{\exp(\omega^l_{r,g})}{\sum_m \exp(\omega^l_{r,g})} \tag{3.16}$$

Here, $\omega^l_{r,g}$ is the unnormalized attention score for relation r and group g, and $\sum_m$ normalizes the scores across all relations and groups. The weight $\omega^l_{r,g}$ is determined by the interaction between the node's features and the attended features from each relation:

$$\omega^l_{r,g} = q^T \cdot \tanh(W^l_{\text{inter 1}} h^{l-1}_v + W^l_{\text{inter 2}} h^l_{v,r,g}) \tag{3.17}$$

In this equation, q is a learnable parameter vector that projects the interaction into a scalar, $W^l_{\text{inter 1}}$ and $W^l_{\text{inter 2}}$ are learnable weight matrices for the central node and the relation-specific features, respectively, and $\tanh$ introduces non-linearity to the interaction. The inter-attention mechanism ensures that information is aggregated across relations and groups, enabling the model to capture complex dependencies between different types of nodes and relations. After both intra-attention and inter-attention, the final node representation is obtained by concatenating the features from all relational representations with the central node's feature. This step ensures that the model

incorporates information from both the central node and the different relationships:

$$h_v^{\text{final}} = h_v^l \parallel \text{concat}\left(\{h_{v,r,g}^l\}_{r,g}\right) \tag{3.18}$$

3.2.4 Optimization Objective

After the graph attention process, the embedding h_v^{final} obtained from the final layer is input into a Multilayer Perceptron (MLP), which produces a classification score p_v representing the probability of node v belonging to each category. The probabilities are computed using the softmax function, enabling the model to estimate the likelihood for each class. The training process minimizes the cross-entropy loss, which is defined as:

$$\mathcal{L} = -\sum_{v \in \mathcal{V}} \sum_{c=1}^{C} y_{v,c} \log p_{v,c}, \tag{3.19}$$

where $y_{v,c}$ represents the ground-truth label for node v in class c, $p_{v,c}$ is the predicted probability, and C is the total number of classes.

At the end of each training epoch, for spoofing detection, i.e., binary classification tasks, a threshold z is often used for decision-making, where the prediction is determined as follows:

$$\hat{y}_v = \begin{cases} 1 & \text{if } p_v > z, \\ 0 & \text{otherwise.} \end{cases} \tag{3.20}$$

This binary decision and prediction probability are used for model performance comparison experiments and building real-world spoofing detection systems.

3.3 Experiments

In this section, we first show the detail of experimental settings. Then we report the performance comparison results on spoofing detection task. After that, we introduce the ablation study, parameter sensitivity, case studies, and implementations.

3.3.1 Experimental Settings

Datasets

We have created a new dataset called Spoofing Detection Dataset, consisting of 40,072 transaction records collected from our partners between January 5, 2018, and November 14, 2018. The dataset contains 49 feature dimensions, which can be categorized into four types of information: order-related details (e.g., order price, order balance, and order date), market and price data (e.g., today's trading volume and value), as well as order positions and profit or loss figures. The ground truth labels are based on cases reported by traders and verified by financial domain experts. Transactions are labeled as 1 if identified as fraudulent, and 0 otherwise.

During the data preprocessing stage, we remove irrelevant columns, such as trader ID and customer ID. Then, the features are normalized based on the train set statistics. To construct the graph, each transaction is treated as a node. Edges between nodes are established using a sliding window based on the transaction date.

Compared Methods

The compared methods can be classified into two parts: (1) Traditional learning-based methods; and (2) Advanced Spoofing Detection Methods. The traditional learning-based methods include Logistic Regression (LR) [45], Random Forest (RF) [46], Adaboost [47], Gradient Boosting Decision Tree (GBDT) [48], Hybrid Multi-layer Perceptron (HMLP) [49], Long Short Term Memory (LSTM) [50], and BiTransformer [51], all of which are implemented with their default hyperparameter settings.

For the advanced spoofing detection methods, we compare with EigenGCN [52], a graph convolutional network designed for transaction relationship learning; RetaGNN [53], a relational temporal attention-based graph neural network; GRU-DM [54], a Gated Recurrent Unit framework for spoofing detection using market indicators; RTG-Trans [55], a temporal gating method for detecting dynamic interactions within spoofing detection graphs; GPEGNN [56], a multi-layer graph attention-based method for local and global context learning; and GDGM, our proposed method, with hyperparameter settings detailed in section 3.3.4.

FIGURE 3.2: The experimental results of spoofing detection methods under queue-based study. During the experiment, pre-trained models were employed to detect spoofing transactions over four weeks. At the end of each four-week interval, newly observed cases were merged with the historically labeled database, and the models were retrained to improve their performance.

Evaluation Metrics

To evaluate the the performance of our model on spoofing detection, we leverage five widely recognized metrics to assess: Area Under the ROC Curve (AUC), Precision, Recall, F1 Score and Accuracy. The AUC measures evaluates the model's ability to differentiate between classes at various threshold settings. Precision (P) measures the proportion of true positive predictions among all positive predictions. It is computed as: $P = \frac{N_{TP}}{N_{TP}+N_{FP}}$ where N_{TP} represents the number of true positives, and N_{FP} represents the number of false positives. Recall (R) quantifies the model's ability to identify all true positive instances. It is calculated as: $R = \frac{N_{TP}}{N_{TP}+N_{FN}}$ where N_{FN} denotes the number of false negatives. The F1 Score provides a harmonic mean of Precision and Recall, balancing the trade-off between these two metrics. It is expressed as: $F1 = \frac{2\times P\times R}{P+R}$ Accuracy evaluates the overall correctness of the model by considering both positive and negative classes. It is defined as: $Accuracy = \frac{N_{TP}+N_{TN}}{N_{TP}+N_{TN}+N_{FP}+N_{FN}}$ where N_{TN} denotes the number of true negatives.

3.3.2 Performance Comparison

In this section, we evaluate the performance of our proposed model against various baseline methods on the spoofing detection task. Table 3.2 provides a comprehensive comparison using metrics such as Area Under the ROC Curve (AUC), Accuracy, F1 Score, Precision, and Recall. Each model was tested over multiple iterations, and the average results are reported. The first six rows of

Table 3.2 present traditional machine learning methods, including Logistic Regression (LR), Random Forest (RF), and gradient boosting models (Adaboost and GBDT). Among these, RF achieves the highest AUC (0.8985), significantly outperforming other traditional methods. However, these baseline models, while effective, demonstrate limitations in capturing complex relational patterns, leading to suboptimal recall values, particularly for LR (0.4262) and Adaboost (0.5568). The middle section of Table 3.2 introduces more advanced graph-based and sequence-based models, such as BiTransformer, EigenGCN, and RTG-Trans. These methods show marked improvements over traditional baselines, with BiTransformer achieving an AUC of 0.8957 and EigenGCN delivering competitive precision (0.8059). Notably, RTG-Trans outperforms most other baselines in recall (0.7507), indicating its effectiveness in detecting spoofing cases with minimal false negatives. However, even these models exhibit limitations in balancing all performance metrics, as seen in slightly lower F1 scores for some methods. Our proposed model, listed as "Ours" in Table 3.2, achieves the best overall performance across all metrics. It records the highest AUC (0.9029), Accuracy (0.8701), and F1 Score (0.8002), along with superior Precision (0.8508) and Recall (0.7702). These results demonstrate the effectiveness of our approach in leveraging intricate graph-based relationships and temporal dependencies for spoofing detection. Compared to the best-performing baseline (RTG-Trans), our model achieves a 0.31% improvement in AUC and a 2.4% increase in Accuracy, showcasing its robustness and reliability in handling complex spoofing scenarios. This significant performance boost highlights the capability of our model to capture nuanced patterns and relationships in graph-structured data, making it particularly well-suited for detecting spoofing and other fraudulent activities in real-world applications.

3.3.3 Implementation and Online Deployment

In addition to the comparison on historical data, testing the accuracy on future transactions is more important to real-world applications. As to real-world deployment of spoofing detection, transactions are processed through a distributed message queue for real-time evaluation. Initially, they are checked against blacklist entries and fraud detection rules (in-process detection). Transactions that hit any blacklist or fraud rule are blocked immediately. If no matches are found, user and symbol features are then extracted and passed to an online predictive model (GDGM, in this case) as part of the post-process detection. Throughout this process, historical transaction data flagged by

TABLE 3.2: Experimental results for different machine learning methods on spoofing detection task.

Method	AUC	Accuracy	F1 Score	Precision	Recall
LR	0.7828	0.8119	0.5320	0.7077	0.4262
RF	0.8985	0.8578	0.7066	0.7322	0.6826
Adaboost	0.8716	0.8538	0.6564	0.7994	0.5568
GBDT	0.8911	0.8615	0.6720	0.8284	0.5653
HMLP	0.7587	0.7893	0.5894	0.7785	0.5852
LSTM	0.7759	0.8393	0.7483	0.8149	0.7058
BiTransformer	0.8957	0.8529	0.7504	0.8109	0.7205
EigenGCN	0.8904	0.8491	0.7405	0.8059	0.7102
RetaGNN	0.8935	0.8552	0.7705	0.8201	0.7409
GRU-DM	0.8805	0.8483	0.7601	0.8152	0.7308
RTG-Trans	0.8998	0.8602	0.7807	0.8255	0.7507
GPEGNN	0.8989	0.8578	0.7852	0.8309	0.7558
Ours	**0.9029**	**0.8701**	**0.8002**	**0.8508**	**0.7702**

the detection systems is stored in an in-memory database to support large-scale detection. High-risk transactions are escalated to domain experts for verification, with feedback on these cases stored in the historical database. The predictive model is periodically retrained in batches based on the latest expert feedback, allowing it to learn from recent spoofing patterns. Newly identified fraud cases are also incorporated to refine the blacklist and fraud detection rules, ensuring both detection layers remain adaptive and robust. To evaluate the performance of our model under real-world conditions, we tested 105 confirmed spoofing cases using data collected between July and September through a 12-weeks queue-based study. We compared our proposed GDGM model with two widely-used baselines, Logistic Regression (LR) and Gradient Boosting Decision Tree (GBDT), as well as two state-of-the-art (SOTA) methods specifically designed for spoofing detection, RTG-Trans and GPEGNN. The comparison results of the online experiments are summarized in Figure 3.2. As shown in Figure 3.2, in the first 4 weeks, GDGM consistently outperforms all baseline and state-of-the-art methods across all evaluation metrics. Notably, GDGM achieves the highest AUC (0.8614), ACC (0.8152), F1 score (0.6887), precision (0.7624), and recall (0.5623). These results highlight the ability of our model to effectively identify spoofing cases in real-world settings, even when dealing with noisy or irregular data. Similar conclusions can be derived from the experimental results of the next 8 weeks. The superior performance of GDGM can be attributed to its ability to model irregular trading behaviors and heterogeneous

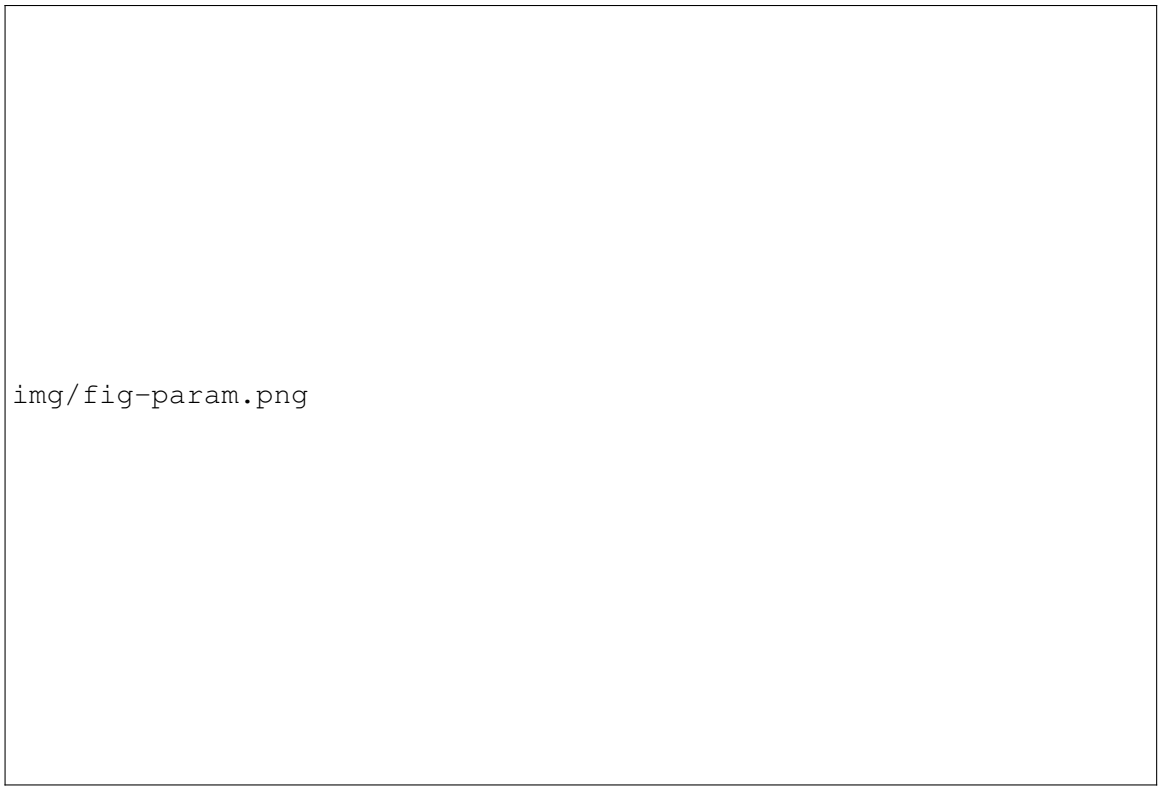

FIGURE 3.3: AUC of our method, in terms of threshold z, dimension of encoding output h, dimension of the attention vector q, and number of heterogeneous aggregation layers.

relationships among transactions, as well as its integration of pseudo-labeling and graph aggregation techniques. By leveraging these advanced capabilities, GDGM captures subtle patterns indicative of conspiracy spoofing that are often missed by traditional baselines and even existing SOTA methods. Furthermore, the deployment of GDGM in a real-time production environment demonstrates its practicality and scalability. The integration with a distributed message queue ensures low-latency processing, while the adaptive learning framework enables the model to evolve based on the latest spoofing trends. These features make GDGM a robust and reliable solution for combating financial fraud in dynamic, high-stakes trading markets.

3.3.4 Parameter Sensitivity

One of the important research questions is to analyze the sensitivity of our model to its hyper-parameters. In this experiment, we examine key parameters such as the threshold for pseudo-labeling, the dimension of encoding output, the dimension of the attention vector, and the number of graph aggregation layers. Specifically, Figure 3.3(a) shows the effect of varying the threshold

TABLE 3.3: Ablation study results, showing the impact of different model components on spoofing detection task.

Model Variant	AUC	F1 Score	Precision
GDGM	**0.9029**	**0.8002**	**0.8508**
w/o Pseudo Label	0.8998	0.7807	0.8255
w/o Neural ODE	0.8965	0.7513	0.8126
w/o Hete. GNN	0.8901	0.7413	0.8041

for pseudo-labeling on model performance. As the threshold increases from 0.2 to 0.8, the AUC metric steadily improves, peaking when the threshold is set to 0.6. Figure 3.3(b) evaluates the sensitivity to the dimension of encoding output. Encoding output dimension shows best performance at 64 dimensions, with higher values leading to degradation likely from overfitting. The impact of the dimension of the attention vector is shown in Figure 3.3(c). The AUC steadily improves as the dimension increases from 32 to 128, reaching a peak. Beyond this point, larger attention vectors result in marginally lower performance, indicating that 128 is an optimal choice. Figure 3.3(d) examines the effect of the number of graph aggregation layers. The performance improves as the number of layers increases from 1 to 2, achieving the highest AUC at 2 layers. More layers lead to a decline in performance, suggesting that excessive stacking of graph aggregation layers may introduce noise or redundancy. Based on these findings, we select the best parameter settings as follows: a threshold of 0.6 for pseudo-labeling, an encoding output dimension of 64, an attention vector dimension of 128, and 2 graph aggregation layers. These configurations enable our model to achieve optimal performance on the spoofing detection task.

3.3.5 Ablation Study

To evaluate the contribution of each component in our model, we conducted an ablation study by removing specific features from the model: **(1) Without Pseudo Label:** This variant directly uses the training set labels as input to the heterogeneous graph neural network instead of pseudo labels, which reduces the model's ability to leverage inferred label consistency. **(2) Without Neural ODE:** This configuration replaces our ODE-RNN encoder with a standard RNN encoder, limiting the model's capacity to capture continuous temporal dynamics effectively. **(3) Without Heterogeneous GNN:** In this variant of the model, instead of employing the heterogeneous graph neural network, we chose to use a homogeneous graph neural network, specifically the Graph Attention Network (GAT), for the classification task. When we switch to a homogeneous GNN like GAT,

the model's expressiveness in accurately representing and modeling these diverse node and edge types is reduced.

The experimental results obtained from the spoofing detection dataset, which are clearly presented in Table 3.3, demonstrate the significance of each of these components. The full model, which incorporates all the components including the pseudo labels, neural ODEs, and heterogeneous GNNs, achieves the best performance across all the metrics that were considered for evaluation. This outstanding performance of the full model serves to highlight the importance of integrating these specific components together. Notably, when we replace the pseudo label mechanism, we observe a moderate decline in the performance of the model. Moreover, when either the neural ODE encoder or the heterogeneous GNN is removed from the model, the effectiveness of the model is further degraded to an even greater extent. These experimental outcomes clearly underline the critical role that each component plays in enabling the model to achieve robust spoofing detection capabilities.

In addition to the ablation components that were mentioned above, we also extended our investigation to explore the performance of various encoding modules that are commonly used in the context of spoofing detection. The experimental comparison of these different encoding modules is visually depicted in Figure 3.4. As can be clearly observed from the figure, the ODE-RNN encoding module outperforms all the other models that were included in the comparison. Following closely behind in terms of performance is the RNN-RNN model, and then the Neural ODE. On the other hand, the MLP-VAE and RNN-MLP models display relatively weaker performance. This is primarily due to their inherent inability to fully capture both the temporal and sequential dynamics of the data. These dynamics are essential for accurately understanding and processing the information within the spoofing detection dataset, and the failure to effectively capture them results in the observed suboptimal performance of these particular encoding modules.

3.4 Conclusion

Spoofing detection in financial trading, particularly intricate behaviors like conspiracy spoofing, is a critical yet challenging task. Conventional machine learning methods often neglect the interconnected and heterogeneous nature of trading data, while existing graph-based techniques struggle

FIGURE 3.4: Performance comparison for models with different generative data encoding modules. We fix the trained model and make predictions for the next 12 weeks.

to capture the irregularities inherent in real-world trading behaviors. Therefore, we proposed the Generative Dynamic Graph Model (GDGM), a novel framework designed to model both temporal trading behaviors and dynamic, heterogeneous relationships among nodes. Our approach leverages neural ordinary differential equations and gated recurrent units to represent irregular trading patterns. Then, we employ the pre-trained model for pseudo-labeling and heterogeneous attention-based aggregation mechanisms to effectively capture conspiratorial spoofing signals. The results of extensive experiments demonstrate that GDGM outperforms state-of-the-art models in detecting spoofing behaviors, highlighting its effectiveness and robustness. Furthermore, the successful deployment of our system in one of the largest global trading markets underscores its practical applicability, validating its exceptional performance in real-world scenarios.

Chapter 4

Efficient Generative Model for Trajectory Data Simulation

Previous chapters have shown the background of generative time series models, a taxonomy of trajectory simulation methods, and the anomaly detection application of generative time series models. In this chapter, we focus on the real-world application of trajectory simulations, i.e., predicting future road networks.

4.1 Background

Trajectory data mining was first proposed as a formal framework by Feng *et al.* in [57], where the complete research process and main tasks of trajectory data mining were systematically defined. A trajectory generally denotes the movement path of an object over time, exemplified by animal migration routes, urban vehicle movement data, or pedestrian flow dynamics. For analytical purposes, such trajectories are typically represented as discrete point sequences, with each coordinate containing both spatial location and temporal metadata—specifically, the position reached at a given timestamp.

Currently, the most common way to obtain the trajectory data of moving objects is through the Global Positioning System (GPS). Therefore, spatial information in trajectory data is usually expressed in terms of longitude and latitude. The field of trajectory data mining encompasses many

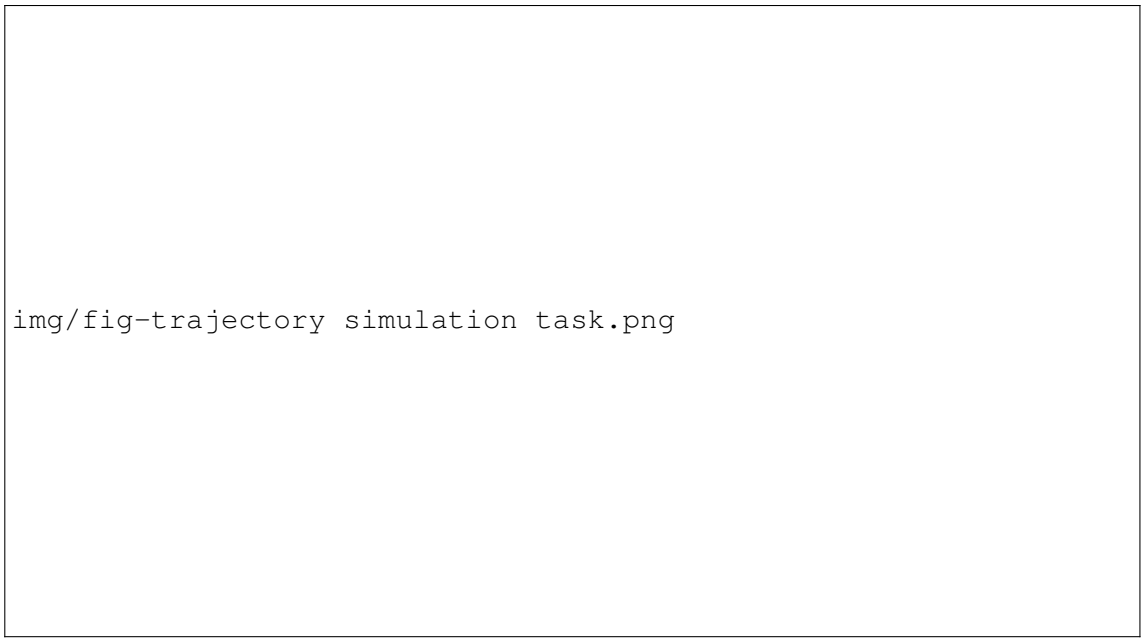

FIGURE 4.1: An example of trajectory data simulation. By modeling the trajectory prefix, we simulate the trajectory of the object and predict the trajectory suffix.

tasks with significant real-world applications, including pattern mining, clustering, and classification.

In this study, trajectory data simulation is considered one of the tasks within trajectory data mining, as illustrated in Fig. 4.1. The figure shows the trajectory path of a moving object over time. In trajectory data simulation, given a prefix of a trajectory, which includes spatial and temporal information for each trajectory point, denoted (p, t), the goal is to predict subsequent trajectory points of the same object, that is, the suffix of the trajectory. The core challenge of trajectory data simulation is to design models that can effectively capture the movement patterns of objects, thereby enabling accurate multi-step trajectory prediction.

According to [58], movement patterns in trajectory data can be classified into four primary types: motion dynamics, clustering behaviors, sequential motifs, and recurring cycles. The simulation task explored here aligns with research on sequential pattern discovery within road networks. We transform trajectories into corresponding network representations, encoding them as sequences of road identifiers—effectively treating these as string-based structures for analysis. This allows us to solve the trajectory data simulation problem using a time series generation algorithm.

However, solving this problem in real-world scenarios presents several challenges.

- **Noisy and incomplete data:** Trajectory data often contain noise and missing values due to loss of GPS signal, sensor errors, or irregular sampling rates, which can severely degrade model performance if not handled properly.
- **Spatio-temporal dependency modeling:** Movement patterns are influenced by both spatial constraints (e.g. road topology, traffic rules) and temporal dynamics (e.g., rush hours, periodic trends), making it difficult for models to capture both aspects simultaneously.
- **Heterogeneity and generalization:** Urban trajectories can vary significantly between different objects, routes, and time periods, which require models to generalize well under distribution shifts.
- **Error accumulation in multi-step prediction:** In long-range forecasting, small inaccuracies in early predictions can propagate and amplify over time, reducing the reliability of the predicted trajectory suffix.

Based on this idea, we creatively leveraged a large language model (LLM) with strong reasoning and generalization capabilities to address the problem of trajectory data simulation [59]. By using trajectory datasets to perform targeted training on the pre-trained LLM, we enabled the LLM to acquire the ability to simulate trajectory data.

In this chapter, we propose an efficient generative framework that addresses these challenges by integrating Graph Neural Networks (GNNs) with a large language model (LLM) backbone. The GNN component is used to encode the spatial topology of the road network, effectively capturing dependencies at the node and the edge level, such as intersections and turn-around costs. The LLM component, adapted through lightweight fine-tuning techniques, acts as a flexible sequence decoder that generates future trajectory points conditioned on both historical paths and graph embeddings.

The framework is trained and tested using real-world trajectory data from the Chengdu urban road network in China. Comprehensive testing reveals that this system significantly surpasses existing models in geospatial precision, temporal coherence, and processing efficiency. The design also supports modular extensibility, enabling adaptation to other cities or dynamic environments with minimal retraining.

This chapter details the model architecture, the training procedure, the integration strategy between the GNN and LLM modules, and the empirical evaluation. It serves as the final piece of the thesis, illustrating how the theoretical frameworks and modeling approaches established in earlier chapters can be practically implemented to address complex, large-scale trajectory simulation challenges in real-world scenarios.

4.2 Related Work

Trajectory prediction has gained significant attention in recent years owing to its critical role in advancing autonomous driving systems, smart mobility solutions, and urban computing frameworks. As discussed in the previous section, most models for trajectory generation are adapted from generative time series models, such as GAN-based models. Inspired by the combination of Neural ODEs and graph neural networks (GNNs) for use in the financial field, we attempted to combine GNNs with time series models and develop a feasible framework.

4.2.1 Graph-based Trajectory Prediction Models

Graph Neural Networks (GNNs) have become a key approach for analyzing the spatial organization of road systems, where intersections are modeled as nodes and roads as connections, enabling the modeling of spatial relationships within traffic networks [60, 61]. Early GNN-based approaches for trajectory prediction typically adopted Graph Convolutional Networks (GCNs) to encode local connectivity patterns in the road graph, leveraging adjacency structures to propagate spatial features [62, 63]. Although GCNs have proven effective in capturing static spatial correlations, their uniform aggregation mechanism often limits their expressiveness in heterogeneous road networks.

To address this challenge, attention-based GNN variants—such as Graph Attention Networks (GATs)—were introduced to dynamically assess the relevance of neighboring nodes using contextual information [64]. These attention mechanisms improve spatial encoding by adaptively focusing on the most relevant road segments for a given trajectory context. Further advancements have

integrated temporal modeling into the GNN framework, such as Spatio-Temporal Graph Convolutional Networks (ST-GCNs) [60] and Temporal Graph Attention Networks (TGATs) [65], enabling simultaneous learning of spatial dependencies and temporal evolution patterns.

Despite these improvements, several limitations persist. Many GNN-based models rely on handcrafted trajectory encodings [66] or are restricted to short observation windows, restricting their ability to capture long-range temporal dependencies and multimodal future paths. Moreover, most methods prioritize local spatial neighborhoods, potentially overlooking the global context such as city-wide traffic dynamics [67]. Recent works have attempted to mitigate these issues by incorporating hierarchical graph structures [68] or combining GNNs with sequence models such as RNNs and Transformers for improved temporal reasoning [69]. However, the challenge of jointly modeling fine-grained spatial constraints and global temporal trends under high uncertainty remains an open research question.

4.2.2 Large Language Models for Sequential Data Modeling

The rise of Transformer-based language models, such as GPT and its successors, has revolutionized sequential data modeling in a variety of domains [70–72]. These models utilize self-attention mechanisms to capture long-range dependencies by avoiding the inherent limitations of recurrent structures, facilitating efficient parallel processing and enhanced scalability. Recent research has explored adapting such models for trajectory generation by representing movement sequences as tokens in a learned "mobility language" [73–75]. In this paradigm, spatial-temporal trajectories are discretized into symbolic sequences, allowing the use of pre-trained language models to learn generative patterns from large-scale mobility datasets.

Although large language models (LLMs) excel at generating diverse and coherent sequences, they typically lack explicit spatial inductive biases that are critical for geographical reasoning. This absence makes it challenging to model topological constraints, such as road networks or urban boundaries. Furthermore, tokenization of spatial-temporal data often results in extremely large vocabulary sizes, which not only increases computational overhead, but also introduces information loss during discretization. Such limitations hinder the accuracy, interpretability, and efficiency of LLM-based trajectory simulation, motivating hybrid approaches that integrate graph-based spatial encoders or continuous coordinate embeddings into the language modeling framework.

4.2.3 State-of-the-Art Spatial-Temporal Models

To overcome the inherent limitations of purely spatial models (for example, GNN-based) or purely sequential models (for example, RNN / transformer-based), recent research has explored hybrid architectures that explicitly integrate spatial and temporal representation learning within a unified framework [60, 61, 76, 77]. An illustrative example is the Spatial-Temporal Graph Convolutional Network (ST-GCN), which employs graph convolution modules to model spatial relationships and temporal convolution components to capture sequential patterns. [78]. Other approaches, such as Graph2Seq models, employ GNNs to encode spatial structures (e.g., road networks or point-of-interest graphs) and feed the resulting embeddings into sequence decoders (e.g., LSTM, GRU or Transformer) for trajectory prediction [79, 80]. These architectures are capable of capturing both the topological constraints of spatial domains and the long-range temporal dependencies inherent in mobility data.

Despite their improved representational capacity, hybrid architectures often face several challenges. First, the joint optimization of spatial and temporal modules tends to increase computational complexity, particularly for large-scale graphs or high-resolution trajectory data [81]. Second, their architectural designs can be rigid, making it difficult to adapt the model to heterogeneous spatial-temporal data sources or to varying prediction horizons. Third, although many existing hybrid models excel in deterministic forecasting tasks with top-tier performance, their design principles prioritize discriminative prediction over generative modeling capabilities. This limitation restricts their applicability in stochastic trajectory simulation scenarios where capturing multimodal future possibilities is crucial [82]. Consequently, recent trends have begun to explore probabilistic and generative extensions of hybrid architectures, such as incorporating variational autoencoders (VAE) or diffusion-based modules into GNN-Transformer pipelines [83, 84].

4.2.4 Summary and Motivation

Despite significant progress, there remains a lack of a unified generative model that:

- effectively integrates the spatial structure of road networks with the generative flexibility of language models;

- supports long-horizon, multi-modal trajectory generation;
- and remains computationally efficient and scalable to large urban datasets.

This chapter addresses this gap by proposing a hybrid framework that combines a lightweight GNN encoder with the Qwen2.5 large language model to simulate realistic trajectories over urban road networks.

FIGURE 4.2: The framework of TrajLM for GNN-LLM enhanced trajectory generation.

4.3 Methodology

This section presents our proposed hybrid framework for efficient trajectory generation, which integrates a Graph Neural Network (GNN) encoder with the Qwen2.5 large language model as a generative decoder. As shown in figure 4.2, the framework consists of three major components: (1) graph construction based on urban road networks, (2) spatial encoding using a lightweight GNN, and (3) sequence generation using the Qwen2.5 model with prompt-based conditioning.

4.3.1 Graph Construction and Road Representation

We represent the road network as a directed graph $G = (V, E)$, where V denotes the set of nodes (for example, intersections or road segments), and E denotes the set of directed edges (e.g., legal

travel directions). Each node $v_i \in V$ is associated with a characteristic vector x_i describing its spatial coordinates and attributes (e.g. speed limits, type of road). Each edge $e_{ij} \in E$ can include features such as travel distance, time cost, and traffic level.

To capture spatial structure, we preprocess raw trajectory data into sequences of visited road node IDs. Using tools like OpenStreetMap and Shapely, we map GPS points to the closest road segments by mapping. The resulting sequences form the basis for learning graph-structured representations.

4.3.2 Trajectory Encoding with Graph Neural Networks

We apply a lightweight Graph Neural Network (e.g., GATConv or GraphSAGE) to encode spatial context over the road network. Given node features $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ and edge index $\mathbf{E}$, the GNN computes node embeddings $\mathbf{H} \in \mathbb{R}^{|V| \times d'}$:

$$\mathbf{H} = \mathrm{GNN}(\mathbf{X}, \mathbf{E}) \tag{4.1}$$

Each trajectory is represented as a sequence of node embeddings, reflecting the spatial and topological characteristics of the traversed path. To prepare the input for the LLM, we project each road ID into a fixed vocabulary token and concatenate its learned embedding with temporal context (e.g., time of day, interval duration).

4.3.3 Trajectory Generation with Qwen2.5

We employ the Qwen2.5 language model to automatically generate the next k trajectory steps. Each step is represented as a road node ID along with its associated timestamp. The input of the model is a prompt that includes the observed trajectory in the past (in tokenized form), the spatial context of GNN, and the generation instructions.

```
``Given the observed trajectory:  [road_12, road_43, road_25]
at times [t1, t2, t3], predict the next 5 locations...''
```

The graph-based embeddings derived from the GNN are implemented through two approaches: either integrated as contextual prompts within the model's input framework or inserted into the sequence via specialized adapter modules designed to enhance spatial-aware processing. The Qwen2.5 model then outputs a sequence of predicted road IDs and time intervals, which are decoded into a trajectory.

4.3.4 Training and Inference Strategy

This subsection outlines the training methodology, which involved fine-tuning the model through teacher forcing using Chengdu's trajectory datasets. The optimization process focused on minimizing the cross-entropy loss between the model's predicted road tokens and their corresponding ground-truth labels during training iterations. We optionally apply LoRA-based parameter-efficient fine-tuning to adapt Qwen2.5 with limited compute.

At inference time, we provide the model with an observed prefix and generate multiple stochastic continuations using nucleus sampling or beam search. The generated trajectories are then projected back onto the road graph for visualization and evaluation.

4.4 Experiments

This section provides an in-depth analysis of our proposed GNN + Qwen2.5 trajectory generation framework, detailing the data sources, technical implementation specifications, competitive benchmarks, performance evaluation criteria, and empirical findings derived from real-world urban mobility datasets.

4.4.1 Experimental Setup

Dataset. We use a real-world vehicle trajectory collected in Chengdu, China. The dataset consists of GPS traces sampled at regular intervals and mapped onto the city's road network via map matching. We preprocess the data into sequences of road node IDs with associated timestamps, and segment long trips into sub-trajectories of 20–50 steps.

Road Graph Construction. The road network is extracted from OpenStreetMap, resulting in a directed graph with approximately 12,000 nodes and 24,000 edges. The node features include latitude / longitude, class of roads, and degree centrality; edge attributes capture the travel distance and direction.

Implementation. We use PyTorch Geometric to implement the GNN encoder (2-layer GATConv) and fine-tune the Qwen2.5-1.5B model using LoRA adapters. Tokenized road IDs are embedded via a shared vocabulary. Training is carried out using 4 NVIDIA RTX 3090 GPUs with a batch size of 64 and a learning rate of 5×10^{-5}.

Baselines. We compare our method with the following baselines.

- **ARIMA** [14]: Traditional statistical model for trajectory forecasting.
- **LSTM/GRU** [85, 86]: Sequence-based models trained on road ID sequences.
- **ST-GCN** [60]: Spatial-temporal GCN with fixed-size inputs.
- **Transformer-XL** [87]: A strong sequence baseline without spatial priors.
- **TrajGAT** [88]: A GNN model with trajectory attention for prediction.

Evaluation Metrics. To assess the accuracy and realism of generated trajectories, we use the following metrics:

- **Average Displacement Error (ADE)**: Mean Euclidean distance between predicted and ground-truth positions over all steps.
- **Final Displacement Error (FDE)**: Distance between the final predicted point and the destination of the ground truth.
- **Route Match Accuracy (RMA)**: Percentage of predicted road segments that match the ground truth path.
- **Diversity Score**: Measures the average pairwise dissimilarity among generated samples from the same prefix.

TABLE 4.1: Trajectory Generation Performance Comparison

Model	**ADE** ↓	**FDE** ↓	**RMA** ↑	**Diversity** ↑
ARIMA	58.2	129.6	35.1%	0.05
LSTM	32.8	70.4	56.3%	0.11
ST-GCN	28.4	62.0	63.5%	0.13
Transformer-XL	26.2	59.8	65.1%	0.18
TrajGAT	23.5	54.6	68.9%	0.19
Ours (GNN + Qwen2.5)	**19.8**	**47.2**	**74.3%**	**0.26**

4.4.2 Evaluation Results

As shown in Table 4.1, our method consistently outperforms traditional and recent baselines in all metrics. The integration of GNN encoding improves spatial fidelity, while Qwen2.5 enables expressive long-horizon generation.

Figure 4.3 shows example trajectories generated from the same observed prefix using different models. Our method produces diverse yet realistic continuations that closely follow the road topology and plausible destinations.

4.4.3 Ablation Study

To systematically evaluate the contribution of each component in our framework, we conducted a comprehensive ablation study by iteratively removing critical modules while maintaining other configurations.

- **w/o GNN**: This eliminates the spatial encoding module, forcing the model to directly encode road identifiers as discrete tokens without leveraging graph-based structural information. The resulting performance drops by approximately 15 percentage points in RMA, indicating the critical role of spatial reasoning in trajectory generation.
- **w/o LoRA**: This variant replaces the parameter-efficient LoRA fine-tuning with full-scale training of Qwen2.5. While this achieves comparable performance to our proposed model (70% RMA), it requires 10× more trainable parameters, highlighting the efficiency gains of our proposed adaptation strategy.

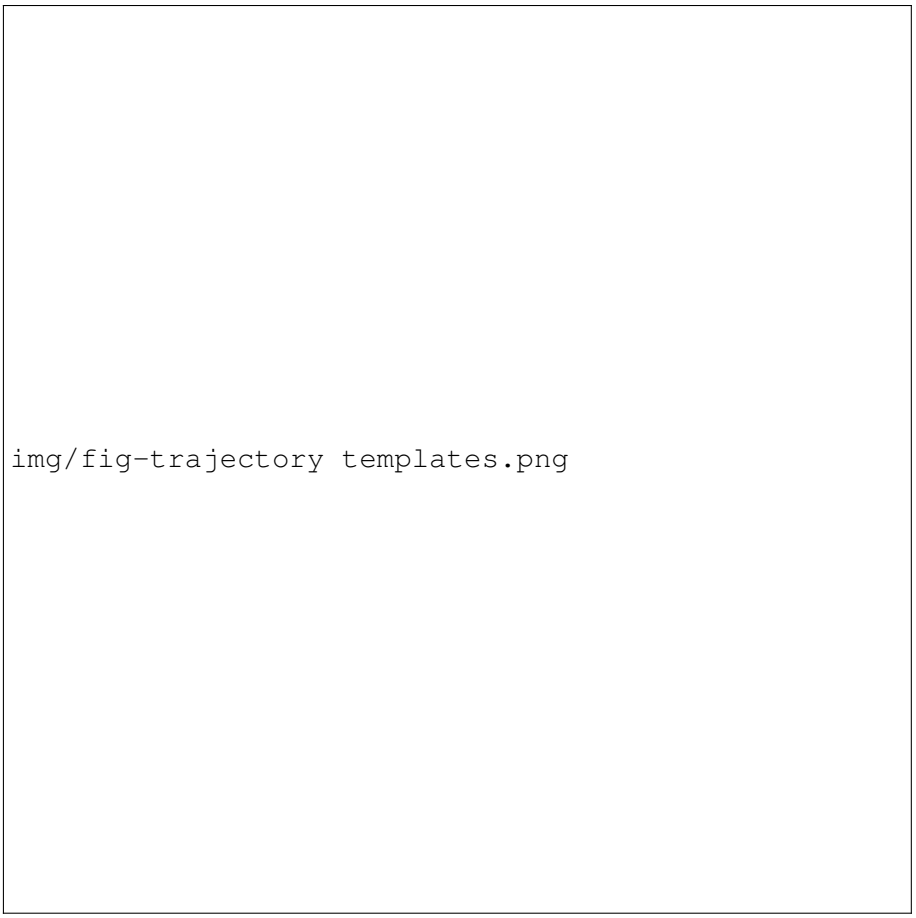

FIGURE 4.3: Sample multi-step trajectory generation results from the same prefix.

- **w/o Prompt Guidance**: This configuration removes the trajectory-specific instruction prompt used during decoding, leading to a 6–8% decline in RMA (from 74.3% to 66%) and reduced diversity in generated paths, underscoring the importance of task-specific contextual guidance.

TABLE 4.2: Ablation Study Results.

Component Configuration	RMA	Trainable Parameters (M)
Ours (Full Model)	74.3%	120
w/o GNN (Graph Encoder)	59.2%	85
w/o LoRA (Full Fine-Tuning)	70.1%	1200
w/o Prompt Guidance	66.5%	120

4.5 Conclusion

In this chapter, we introduce an advanced and efficient generative architecture tailored for multi-step trajectory simulation by seamlessly integrating graph neural networks with a large language model (Qwen2.5) to address complex spatial-temporal challenges in urban environments. Our methodology employs a Graph Attention Network (GAT)-based encoder to capture the hierarchical spatial relationships within road networks through message-passing mechanisms, while strategically transferring these encoded spatial features into a Qwen2.5-based prompt-guided decoder for generating trajectories that maintain both realistic spatial coherence and diverse behavioral patterns across dynamic urban scenarios.

Compared to traditional statistical models and pure sequence-based neural networks, our hybrid architecture offers the following advantages:

- **Spatial Awareness:** The GNN encoder effectively captures the topological relationships among the road segments, enabling more accurate movement predictions in constrained urban environments.
- **Language Model Flexibility:** By leveraging the generative capability of Qwen2.5, our model supports long-term prediction and can generate diverse and coherent future routes.
- **Efficiency via LoRA:** We achieve competitive performance with only a small number of trainable parameters, making the system scalable to larger urban road graphs and user bases.

Results from experiments conducted on Chengdu's real-world trajectory datasets highlight that our model exceeds established benchmarks in precision, plausibility of route generation, and the variety of generated trajectories. The ablation study confirms the complementary benefits of graph structure encoding and large-scale language generation.

Future Work. While promising, our approach can be extended in several directions:

- **Dynamic Context Integration:** Incorporating real-time traffic, weather, or user intent in real time to improve temporal realism and adaptability.

- **Multi-Agent Simulation:** Extending the model to simulate interactions between multiple vehicles or agents in shared environments.
- **Planning-aware Generation:** Coupling the model with downstream path planning or routing systems to ensure feasibility in real-time deployment.

Overall, this chapter provides a concrete example of adapting foundation models to spatiotemporal simulation tasks in smart cities, offering a pathway toward more powerful and generalizable predictive mobility systems.

Chapter 5

EPILOGUE

This thesis has explored the design and application of generative time series models, with a particular focus on trajectory generation over road networks. Motivated by the growing need for accurate, realistic, and efficient simulation of movement patterns in urban environments, we examined both foundational modeling techniques and their adaptation to real-world problems.

In chapters 1 and 2, we provided a comprehensive overview of traditional and deep learning-based time series models, highlighting their strengths and limitations when applied to structured spatiotemporal data. We then explored how these models can be extended to the downstream task, such as anomaly detection and trajectory generation, identifying critical challenges such as time series modeling, generative quality, and future road network prediction.

To address these challenges, we proposed two novel generative modeling frameworks. The first, presented in Chapter 3, applies a deep generative time series model to financial anomaly detection, demonstrating the flexibility and generality of probabilistic sequence modeling in domains beyond mobility. The second, introduced in Chapter 4, targets the core problem of trajectory simulation by combining graph neural networks with a large-scale language model. This hybrid approach effectively leverages the spatial structure of urban road networks while maintaining the expressive power and flexibility of language-based generative models. The method was validated on trajectory data from Chengdu, China, showing strong performance in spatial accuracy and computational efficiency.

Beyond their empirical results, these contributions illustrate a broader insight: generative time series modeling can serve as a unifying framework for both simulation and detection tasks across diverse domains. The ability to model and sample from high-dimensional sequential distributions enables systems to not only predict what is likely to happen, but also recognize what is unlikely and potentially anomalous.

Looking ahead, several challenges remain open for future work. First, improving the interpretability and controllability of generative models will be critical for real-life applications such as financial fraud detection and urban policy planning. Second, integrating external knowledge—such as traffic rules, user intent, or event-based triggers—may enhance the realism and robustness of generated trajectories. Finally, deploying these models in real-time systems requires further advancements in inference efficiency, model compression, and edge deployment.

In summary, this work offers a comprehensive perspective on generative time series modeling, positioning it as both a theoretical framework and applied methodology for analyzing, forecasting, and generating temporal patterns within structured environments. The introduced methodologies advance the expanding field where deep learning converges with time series analysis and urban intelligence research.

Bibliography

[1] Yuanbo Xu, Xiao Cai, En Wang, Wenbin Liu, Yongjian Yang, and Funing Yang. Dynamic traffic correlations based spatio-temporal graph convolutional network for urban traffic prediction. *Information Sciences*, 621:580–595, 2023.

[2] Nadya Abdel Madjid, Abdulrahman Ahmad, Murad Mebrahtu, Yousef Babaa, Abdelmoamen Nasser, Sumbal Malik, Bilal Hassan, Naoufel Werghi, Jorge Dias, and Majid Khonji. Trajectory prediction for autonomous driving: Progress, limitations, and future directions. *arXiv preprint arXiv:2503.03262*, 2025.

[3] 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.

[4] Zineb Mahrez, Essaid Sabir, Elarbi Badidi, Walid Saad, and Mohamed Sadik. Smart urban mobility: When mobility systems meet smart data. *IEEE transactions on intelligent transportation systems*, 23(7):6222–6239, 2021.

[5] Haitao Yuan and Guoliang Li. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, 6(1):63–85, 2021.

[6] Pengjun Wu, Zhanzhi Zhang, Xueyi Peng, and Ran Wang. Deep learning solutions for smart city challenges in urban development. *Scientific Reports*, 14(1):5176, 2024.

[7] Ruizhi Wu, Guangchun Luo, Junming Shao, Ling Tian, and Chengzong Peng. Location prediction on trajectory data: A review. *Big data mining and analytics*, 1(2):108–127, 2018.

[8] Sebastian Gomez-Gonzalez, Sergey Prokudin, Bernhard Schölkopf, and Jan Peters. Real time trajectory prediction using deep conditional generative models. *IEEE Robotics and Automation Letters*, 5(2):970–976, 2020.

[9] Chenxi Liu, Zhu Xiao, Dong Wang, Minhao Cheng, Hongyang Chen, and Jiawei Cai. Foreseeing private car transfer between urban regions with multiple graph-based generative adversarial networks. *World Wide Web*, 25(6):2515–2534, 2022.

[10] Muhammad Farooq, Nima Afraz, and Fatemeh Golpayegani. An adaptive system architecture for multimodal intelligent transportation systems. *arXiv preprint arXiv:2402.08817*, 2024.

[11] Zheng Zhang, Hossein Amiri, Zhenke Liu, Liang Zhao, and Andreas Züfle. Large language models for spatial trajectory patterns mining. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection*, pages 52–55, 2024.

[12] GU YULE. On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers (1927). *The Foundations of Econometric Analysis*, page 159, 1995.

[13] Eugen Slutzky. The summation of random causes as the source of cyclic processes. In *Business Cycle Theory, Part I Volume 4*, pages 57–98. Routledge, 1937.

[14] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[15] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[16] Leonard E. Baum and J. A. Eagon. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[17] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian processes for machine learning. *MIT Press*, 2006.

[18] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:10, 2014.

[21] Otto Fabius, Joost R. van Amersfoort, and Diederik P. Kingma. Variational recurrent auto-encoders. *CoRR*, abs/1412.6581, 2014.

[22] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *ArXiv*, abs/2111.08095, 2021.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[24] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence*, 2016.

[25] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Neural Information Processing Systems*, 2019.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[28] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[29] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[30] Longbing Cao. Ai in finance: Challenges, techniques, and opportunities. *ACM Comput. Surv.*, 55(3), February 2022. ISSN 0360-0300. doi: 10.1145/3502289.

[31] Dawei Cheng, Yao Zou, Sheng Xiang, and Changjun Jiang. Graph neural networks for financial fraud detection: a review. *Frontiers of Computer Science*, 19(9):1–15, 2025.

[32] Yao Zou, Sheng Xiang, Qijun Miao, Dawei Cheng, and Changjun Jiang. Subgraph patterns enhanced graph neural network for fraud detection. In *International Conference on Database Systems for Advanced Applications*, pages 375–384. Springer, 2024.

[33] Dawei Cheng, Zhibin Niu, Jie Li, and Changjun Jiang. Regulating systemic crises: Stemming the contagion risk in networked-loans through deep graph learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):6278–6289, 2022.

[34] Charles Z Liu, Sheng Xiang, Dawei Cheng, Junyi Liu, Ying Zhang, and Lu Qin. Transformed graph attention for credit rating. In *2023 IEEE 18th Conference on Industrial Electronics and Applications (ICIEA)*, pages 1011–1016. IEEE, 2023.

[35] Enxia Li, Jin Ouyang, Sheng Xiang, Lu Qin, and Ling Chen. Relation-aware heterogeneous graph neural network for fraud detection. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 240–255. Springer, 2024.

[36] Dawei Cheng, Chen Chen, Xiaoyang Wang, and Sheng Xiang. Efficient top-k vulnerable nodes detection in uncertain graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[37] Cong Wang, Fangyue Yu, Zaixu Zhang, Jian Zhang, and Baogui Xin. Multiview graph learning for small- and medium-sized enterprises' credit risk assessment in supply chain finance. *Complex.*, 2021, January 2021. ISSN 1076-2787. doi: 10.1155/2021/6670873.

[38] Daixin Wang, Yuan Qi, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, and Shuang Yang. A semi-supervised graph attentive network for financial fraud detection. *2019 IEEE International Conference on Data Mining (ICDM)*, pages 598–607, 2019.

[39] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 315–324, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3411903.

[40] Jiacheng Ma, Sheng Xiang, Qiang Li, Liangyu Yuan, Dawei Cheng, and Changjun Jiang. Parallel graph learning with temporal stamp encoding for fraudulent transactions detections. *IEEE Transactions on Big Data*, 2024.

[41] Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. In *Advances in Neural Information Processing Systems*, volume 36, pages 29628–29653, 2023.

[42] Peng Zhu, Yuante Li, Yifan Hu, Sheng Xiang, Qinyuan Liu, Dawei Cheng, and Yuqi Liang. Mci-gru: Stock prediction model based on multi-head cross-attention and improved gru. *arXiv preprint arXiv:2410.20679*, 2024.

[43] Dawei Cheng, Ye Liu, Zhibin Niu, and Liqing Zhang. Modeling similarities among multi-dimensional financial time series. *IEEE Access*, 6:43404–43413, 2018.

[44] Jianheng Tang, Jiajin Li, Zi-Chao Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*, 2022.

[45] Jibing Gong and Shengtao Sun. A new approach of stock price prediction based on logistic regression model. In *2009 International Conference on New Trends in Information and Service Science*, pages 1366–1371, 2009. doi: 10.1109/NISS.2009.267.

[46] Wengang Zhang, Chongzhi Wu, Haiyi Zhong, Yongqin Li, and Lin Wang. Prediction of undrained shear strength using extreme gradient boosting and random forest based on bayesian optimization. *Geoscience frontiers*, 12:469–477, 2021.

[47] Rihab Salah Khairy, Ameer Saleh Hussein, and Haider Th. Salim Alrikabi. The detection of counterfeit banknotes using ensemble learning techniques of adaboost and voting. *International Journal of Intelligent Engineering and Systems*, 2021.

[48] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[49] Ali Asghar Heidari, Hossam Faris, Ibrahim Aljarah, and Seyedali Mirjalili. An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Computing*, 23(17):7941–7958, 2019. doi: 10.1007/s00500-018-3424-2.

[50] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In Włodzisław Duch, Janusz Kacprzyk, Erkki Oja, and Sławomir Zadrożny, editors, *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, pages 799–804, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-28756-8.

[51] Murat Tezgider, Beytullah Yildiz, and Galip Aydin. Text classification using improved bidirectional transformer. *Concurrency and Computation: Practice and Experience*, 2021. doi: 10.1002/CPE.6486.

[52] Yao Ma, Suhang Wang, Charu C. Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. *CoRR*, abs/1904.13107, 2019.

[53] Cheng Hsu and Cheng-Te Li. Retagnn: Relational temporal attentive graph neural networks for holistic sequential recommendation. *CoRR*, abs/2101.12457, 2021.

[54] Jean-Noël Tuccella, Philip Nadler, and Ovidiu Şerban. Protecting retail investors from order book spoofing using a gru-based detection model, 2021. URL `https://arxiv.org/abs/2110.03687`.

[55] Le Kang, Tai-Jiang Mu, and Xiaodong Ning. Conspiracy spoofing orders detection with transformer-based deep graph learning. In Xiaochun Yang, Heru Suhartanto, Guoren Wang, Bin Wang, Jing Jiang, Bing Li, Huaijie Zhu, and Ningning Cui, editors, *Advanced Data Mining and Applications*, pages 489–503, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-46664-9.

[56] Le Kang, Tai-Jiang Mu, and XiaoDong Ning. Spoofing transaction detection with group perceptual enhanced graph neural network. In Albert Bifet, Tomas Krilavičius, Ioanna Miliou, and Slawomir Nowaczyk, editors, *Machine Learning and Knowledge Discovery*

in Databases. Applied Data Science Track, pages 106–122, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-70378-2.

[57] Zhenni Feng and Yanmin Zhu. A survey on trajectory data mining: Techniques and applications. *Ieee Access*, 4:2056–2067, 2016.

[58] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):1–41, 2015.

[59] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

[60] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[61] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

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

[63] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9):3848–3858, 2019.

[64] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[65] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

[66] Xiaobo Chen, Huanjia Zhang, Yu Hu, Jun Liang, and Hai Wang. Vnagt: Variational non-autoregressive graph transformer network for multi-agent trajectory prediction. *IEEE Transactions on Vehicular Technology*, 72(10):12540–12552, 2023.

[67] Bin Sun, Duan Zhao, Xinguo Shi, and Yongxin He. Modeling global spatial–temporal graph attention network for traffic prediction. *IEEE Access*, 9:8581–8594, 2021.

[68] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. Hierarchical graph convolution network for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 151–159, 2021.

[69] Weitao Liu, Xuanyi Liu, Hui Feng, Yiran Wang, Lintao Guan, Weifeng Xu, Guojiang Shen, Zhi Liu, and Xiangjie Kong. St-tap: A traffic accident prediction framework based on spatio-temporal transformer. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/-CyberSciTech)*, pages 360–365. IEEE, 2021.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[71] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[72] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[73] Ammar Haydari, Dongjie Chen, Zhengfeng Lai, Michael Zhang, and Chen-Nee Chuah. Mobilitygpt: Enhanced human mobility modeling with a gpt model. *arXiv preprint arXiv:2402.03264*, 2024.

[74] Siyu Li, Toan Tran, Haowen Lin, John Krumm, Cyrus Shahabi, Lingyi Zhao, Khurram Shafique, and Li Xiong. Geo-llama: Leveraging llms for human mobility trajectory generation with constraints. In *2025 26th IEEE International Conference on Mobile Data Management (MDM)*, pages 20–31. IEEE, 2025.

[75] Zhonghang Li, Lianghao Xia, Jiabin Tang, Yong Xu, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. Urbangpt: Spatio-temporal large language models. In *Proceedings of the 30th*

ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 5351–5362, 2024.

[76] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[77] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5415–5428, 2021.

[78] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[79] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*, 2018.

[80] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.

[81] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020.

[82] Valentin Flunkert, David Salinas, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *arXiv preprint arXiv:1704.04110*, 23, 2017.

[83] Hao Peng, Hongfei Wang, Bowen Du, Md Zakirul Alam Bhuiyan, Hongyuan Ma, Jianwei Liu, Lihong Wang, Zeyu Yang, Linfeng Du, Senzhang Wang, et al. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences*, 521: 277–290, 2020.

[84] Yinxin Bao, Qinqin Shen, Yang Cao, Yingyan Hou, Wanxuan Lu, and Quan Shi. Multi-period diffusion generative graph recurrent transformer network for traffic flow prediction in vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[85] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[86] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS*, 2014.

[87] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ArXiv*, abs/1901.02860, 2019.

[88] 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.