Towards Knowledge Enhanced Deep Time Series Analytics

Zhao, Kai

DOI (link to publication from Publisher):
10.54337/aau793105773

Publication date:
2025

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Zhao, K. (2025). *Towards Knowledge Enhanced Deep Time Series Analytics*. Aalborg University Open Publishing. https://doi.org/10.54337/aau793105773

TOWARDS KNOWLEDGE ENHANCED DEEP TIME SERIES ANALYTICS

BY
KAI ZHAO

PhD Thesis 2025

AALBORG
UNIVERSITY

Towards Knowledge Enhanced Deep Time Series Analytics

Ph.D. Dissertation

Kai Zhao

Ph.D. Thesis 2025
Aalborg University
Department of Computer Science
DK-9220 Aalborg

Submitted:	June 2025
Main Supervisor:	Associate Professor Chenjuan Guo Aalborg University
Assessment:	Associate Professor Álvaro Torralba (chair) Aalborg University, Denmark
	Associate Professor Daniel Romero University of Agder, Norway
	Professor Yang Yang Zhejiang University, China
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Computer Science

ISSN: 2446-1628
ISBN: 97887-7642-058-1

Published by: Aalborg University Open Publishing | www.open.aau.dk

Abstract

Data collected from real-world applications can usually be modeled as time-dependent observations that form multivariate time series, e.g., energy production in different locations and traffic flows on different roads. With these data, time series analytics plays an important role in ensuring the effective functionality of applications in different scenarios. For example, time series forecasting can help improve schedules in transport systems, and anomaly detection can help improve safety and reliability in computing systems. Even though recent works with deep learning technologies have demonstrated good performance in multivariate time series analytics, these approaches are usually based on data-driven models. They struggle to utilize real-world knowledge, which can enhance reliability to improve accuracy for data-driven models.

Specifically, previous time series analytics approaches have three major limitations: 1) existing methods struggle to learn dynamic relationships among different variates and causal temporal influence from different timestamps for time series forecasting; 2) different variates can vary continuously or discretely across continuous time, but existing methods cannot utilize continuity and discreteness knowledge for anomaly detection; and 3) existing methods usually detect anomalies after anomalies get severe but cannot predict future anomaly for timely maintenance of systems.

This Ph.D. project aims to contribute new technologies to improve time series analytics by incorporating real-world and scientific knowledge into data-driven deep learning models. Specifically, we focus on three types of functionality in this thesis: time series forecasting, anomaly detection, and anomaly prediction.

First, we incorporate dynamic graph modeling and causality knowledge into multivariate time series forecasting. Recent methods based on graph neural networks (GNN) learn the relation graph to capture correlations among historical time series variates. However, it is known that relations among variates can differ across time where future correlations may be different from the recent relation graphs. Thus, we propose the MTSF-DG model. MTSF-DG can learn historical relation graphs and predict future relation graphs to capture the dynamic relations, guided by causality knowledge. We further propose a reasoning network to learn temporal influences from historical timestamps and explicitly forecast each future timestamp. Extensive experiments on real-

world datasets from different scenarios show that MTSF-DG consistently outperforms state-of-the-art baselines.

Second, we incorporate continuity and discreteness knowledge into anomaly detection. While existing methods perform well for time series collected with regular time intervals, they struggle when having to learn both continuous and discrete dynamics for different variates across time. Further, they struggle to recognize the importance of variates that have different measurement units. To overcome these limitations, we propose TAD-UP that incorporates differential equation knowledge for Time series Anomaly Detection via Unified Probabilistic modeling. We propose two co-dependent branches of neural ordinary differential equations to learn both continuous and discrete dynamics for different variates. We also propose a unified joint probability distribution modeling. The resulting model is optimized using Maximum Likelihood Estimation on joint variates. We detect anomalies using joint probabilities, which take the marginal probabilities of different variates into account as their importance on final anomaly scores. Experiments on nine datasets from different domains offer evidence that TAD-UP is capable of state-of-the-art performance for real-world time series anomaly detection.

Third, we explore unsupervised time series anomaly prediction using Maximum Mean Discrepancy (MMD) knowledge. Existing anomaly prediction methods rely on labeled training data for achieving acceptable accuracy. However, such data may be difficult to obtain; and in real-time deployments, anomalies can occur that were not seen in labeled data, thus making them difficult to predict. We propose an Importance-based Generative Contrastive Learning method (IGCL) for unsupervised anomaly prediction. IGCL employs a diffusion module to produce anomaly precursor patterns, controlled by MMD knowledge. IGCL then can predict anomalies by identifying anomaly precursors that will evolve into future anomalies. To address challenges caused by potentially complex precursor combinations involving multiple variables, we propose a memory bank with importance scores that stores representative samples adaptively and generates more complex anomaly precursors. Extensive experiments on real-world datasets offer evidence that the proposed IGCL is able to outperform state-of-the-art baselines.

Resumé

Data indsamlet fra forskellige applikationer kan ofte modelleres som tidsafhængige observationer, der danner multivariate tidsserier, fx data om energiproduktion og forbrug og trafikstrømme og hastigheder i vejnetværk. Tilgængeligheden af sådanne data gør, at tidsserieanalyser spiller en vigtig rolle i applikationer i forskellige scenarier. Tidsserieprognoser kan fx hjælpe med at forbedre kørselsplaner i vejnetværk, og detektering af anomalier kan hjælpe med at forbedre sikkerheden og pålideligheden i computersystemer. Selvom vi har set fremskridt inden for deep learning til tidsserieanalyse, er eksisterende metoder ofte baseret på datadrevne modeller og har svært ved at udnytte yderligere domæneviden viden til at opnå forbedret nøjagtighed.

Specifikt har tidligere tidsserieanalysemetoder tre væsentlige begrænsninger: (i) eksisterende metoder har svært ved at lære dynamiske relationer mellem forskellige dimensioner i tidsserier og årsagsmæssige sammenhænge over tid til brug i tidsserieprognoser; (ii) til trods for at forskellige dimensioner rummer enten kontinuerte eller diskrete data, så eksisterende metoder kan ikke udnytte denne viden til identifikation af anomalier; og (iii) eksisterende metoder finder ofte først anomalier, når de er blevet alvorlige, og de kan ikke umiddelbart identificere begyndende anomalier med henblik på tidlig intervention.

Dette ph.d.-projekt sigter mod at bidrage med nye metoder til at forbedre tidsserieanalyser ved at indarbejde domæneviden og videnskabelig viden i datadrevne metoder baseret på deep learning. Specielt fokuserer projektet på på tre typer funktionalitet: tidsserieprognoser og detektion og forudsigelse af anomalier.

Først studerer vi tidsserieprognoser. Nye metoder baseret på graf-neurale netværk (GNN) lærer relations-grafer for at opfange korrelationer mellem dimensioner i historiske tidsserier. Det er velkendt, at korrelationer mellem værdier i forskellige dimensioner kan variere over tid, hvor fremtidige korrelationer kan afvige fra korrelationerne de seneste relations-grafer. Derfor foreslår vi MTSF-DG modellen, der indarbejder dynamisk grafmodellering i tidsserieprognoser. MTSF-DG kan lære historiske relations-grafer og kan forudsige fremtidige relations-grafer og kan opfange dynamiske relationer baseret på viden om kausalitet. Vi foreslår desuden teknikker, der gør det muligt at lære tidsmæssige påvirkninger fra historiske tidsstempler og at forudsige fremtidige værdier. Omfattende eksperimenter med virkelige data fra forskellige scenarier viser, at MTSF-DG konsekvent kan levere bedre performance end eksisterende alternativer.

For det andet studerer vi detektering af anomalier. Mens eksisterende metoder fungerer godt for tidsserier indsamlet med regelmæssige tidsintervaller, har de svært ved at lære samtidigt på tværs af kontinuerte og diskrete data i multivariate tidsserier. Desuden har de svært ved at håndtere data med forskellige måleenheder. For at overvinde disse begrænsninger foreslår vi TAD-UP, der udnytter viden modelleret ved hjælp af differentialligninger til af finde anomalier. Vi foreslår to indbyrdes afhængige grene af neurale almindelige differentialligninger for at lære både kontinuerlig og diskret dynamik i forskellige dimensioner. Vi foreslår også en samlet fælles sandsynlighedsfordelingsmodellering. Den resulterende model er optimeret ved hjælp af maximum likelihood estimation på tværs af dimensioner. Vi opdager anomalier ved hjælp af fælles sandsynligheder, som tager højde for de marginale sandsynligheder i forskellige dimensioner. Eksperimenter på ni datasæt fra forskellige domæner viser, at TAD-UP er i stand til at opnå state-of-the-art performance ved detektion af anomalier i tidsserier.

For det tredje studerer vi forudsigelse af anomalier i tidsserier. Eksisterende metoder til forudsigelse af anomalier er afhængige af træningsdata for at opnå acceptabel nøjagtighed. Sådanne data kan dog være vanskelige at skaffe, og i anvendelser, hvor data ankommer løbende, kan der forekomme anomalier, som ikke findes i træningsdata, hvilket gør dem vanskelige at forudsige. Vi foreslår en vigtighedsbaseret metode baseret på Generative Contrastive Learning (IGCL) til forudsigelse af anomalier uden brug af indsamlede træningsdata. IGCL anvender et diffusionsmodul til at producere mønstre, der udgør forstadier til anomalier, baseret på viden fra Maximal Mean Discrepancy teori. IGCL kan derefter forudsige anomalier ved at identificere anomali-forstadier, der vil udvikle sig til fremtidige anomalier. For at løse udfordringer forårsaget af potentielt komplekse kombinationer af forstadier på tværs af flere dimensioner, foreslår vi teknikker, der er i stand til løbende og adaptivt at generere mere komplekse anomaliforstadier. Omfattende eksperimenter med virkelige datasæt viser, at IGCL er i stand til at opnå bedre performance end eksisterende metoder.

Acknowledgements

I would like to express my deepest gratitude to all who have supported and helped me during my Ph.D. journey.

First and foremost, I would like to express my sincere thanks to my main supervisor, Prof. Chenjuan Guo, for her exceptional guidance, unwavering support, and invaluable insights throughout my doctoral studies and life. Her expertise, patience, and encouragement have been crucial to my academic development and the successful completion of this dissertation. I truly appreciate her dedication to my growth as a researcher and scholar, and it is my pleasure to be her Ph.D. student.

I would also like to extend my deepest gratitude to my co-supervisors, Prof. Christian S. Jensen, Assistant Prof. Miao Zhang, and Assistant Prof. Peng Han, for their thoughtful feedback, constructive criticism, and guidance at various stages of my research. Their contributions have been instrumental in shaping the direction of my work.

Special thanks and gratitude to Prof. Bin Yang for his collaborative spirit, insightful discussions and suggestions, and constant motivation. The camaraderie shared during these years has made this journey more rewarding.

Further, I would like to thank my co-authors, especially Zhihao Zhuang, Shiyan Hu, Qichao Shentu, Beibu Li, and Kasper Skytte Andersen, for their substantial support in terms of developing codes and writing scientific papers. It is a big pleasure to work with them and the camaraderie shared has made this journey more enjoyable.

I am also grateful to my colleagues and fellow researchers at the Data Engineering, Science and Systems group at Aalborg University (AAU) for creating a stimulating, warm and supportive academic environment. I specially thank Prof. Xiaokui Xiao who hosted me as a visiting Ph.D. student at National University of Singapore (NUS). In addition, I thank the administrative colleagues at AAU and NUS who helped my Ph.D. life smoother.

I would also like to acknowledge the financial support I received from Prof. Chenjuan Guo, the Department of Computer Science, Independent Research Fund Denmark, and the Villum Foundation, which made this research possible. Their generosity has been essential in providing the resources necessary for my work.

A heartfelt thanks to my family and friends for their unwavering love, understanding, and encouragement. To my parents, your belief in me has been a source of strength throughout my academic career. To my friends, thank you for your endless patience, love, and support, especially during the more challenging times. You have been my rock, and this accomplishment is as much yours as it is mine.

Finally, I would like to dedicate this work to all those who have supported me in ways both big and small. Your presence, encouragement, and love have been the foundation upon which this dissertation was built.

Thank you all.

Contents

Thesis Details

Thesis Title:	Causality and Scientific Knowledge Enhanced Reliable Deep Time Series Analytics
Ph.D. Student:	Kai Zhao
Supervisors:	Prof. Chenjuan Guo, Aalborg University
	Prof. Christian S. Jensen, Aalborg University
	Asst. Prof. Miao Zhang, Aalborg University
	Asst. Prof. Peng Han, Aalborg University

The main body of this thesis consist of the following papers.

[A] **Kai Zhao**, Chenjuan Guo, Yunyao Cheng, Peng Han, Miao Zhang, Bin Yang, "Multiple Time Series Forecasting with Dynamic Graph Modeling," *in Proc. VLDB Endow.*, pp. 753-765, 2023.

[B] **Kai Zhao**, Chenjuan Guo, Yuying Qiu, Christian S. Jensen, Yunyao Cheng, Bin Yang, "TAD-UP: Learning Continuous and Discrete Dynamics for Time Series Anomaly Detection via Unified Probabilistic Modeling," *submitted to VLDB,* 2025.

[C] **Kai Zhao**, Zhihao Zhuang, Chenjuan Guo, Hao Miao, Christian S. Jensen, Yunyao Cheng, Bin Yang, "Unsupervised Time Series Anomaly Prediction with Importance-based Generative Contrastive Learning," *in SIGKDD,* 2025.

In addition to the above papers, the following main publications have also been made during the Ph.D. studies, which are not included in this thesis.

[1] **Kai Zhao**, Zhihao Zhuang, Chenjuan Guo, Miao Zhang, Yang Shu,

Bin Yang, "Enhancing Diversity for Data-free Quantization," *in CVPR*, 2025.

[2] Kasper Skytte Andersen*, **Kai Zhao***, Alexander de Linde Agerskov, Christian Bro Sørensen, Trine Juhl Holmager, Marta Nierycho, Miriam Peces, Chenjuan Guo, Per Halkjær Nielsen, "Predicting microbial community structure and temporal dynamics by using graph neural network models," *Nature Communications (revision submitted)*, 2025.

[3] Qichao Shentu, Beibu Li, **Kai Zhao**, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, Chenjuan Guo, "Towards a General Time Series Anomaly Detector with Adaptive Bottlenecks and Dual Adversarial Decoders," *in ICLR*, 2025.

[4] Zhihao Zhuang, Yingying Zhang, **Kai Zhao**, Bin Yang, Chenjuan Guo, Lunting Fan, "Noise Matters: Cross Contrastive Learning for Flink Anomaly Detection," *in Proc. VLDB Endow.*, 2025.

[5] Yihang Wang, Yuying Qiu, Peng Chen, **Kai Zhao**, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, Chenjuan Guo , "Towards a General Time Series Forecasting Model with Unified Representation and Adaptive Transfer," *in ICML*, 2025.

This thesis has been submitted for assessment in partial fulfillment of the Ph.D. degree. The thesis is based on the submitted or published scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extended summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The permission for using the published and accepted articles in the thesis have been obtained from the corresponding publishers with the condition that they are cited and copyrights are placed prominently in the references. In reference to IEEE copyrighted material, which is used with permission in this thesis, the IEEE does not endorse any of Aalborg University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution,

Part I

Thesis Summary

Introduction

This thesis focuses on incorporating real-world and scientific knowledge into data-driven deep learning models to improve time series analytics on time series forecasting, anomaly detection, and anomaly prediction. In the next section, we introduce the background and motivation. Then, we introduce the thesis structure.

1 Background and Motivation

The advent of ubiquitous data collection in modern applications and pervasive sensing to monitor environment and runtime status has led to an exponential increase in multivariate time series data [9, 56, 87]. For example, computing systems can record different runtime indicators [1, 70] and environmental systems can record water and air quality across time [24, 50]. In domains ranging from energy and transportation to computing systems, the need to understand and forecast the behavior of dynamic systems has never been more critical. As exemplified in Fig. 1, effective time series analytics is important in many real-world applications. For example, time series forecasting can support urban computing and smart transportation [14, 22, 36, 37], and anomaly detection can support fault location and risk monitoring [10, 33, 58].

Recent works with deep learning technologies have demonstrated better performance in time series analytics than traditional methods that are based on statistical models [73, 88]. These deep learning methods [51, 53, 75] are powerful in learning complex temporal features and relations from the massively available multivariate time series data. However, existing traditional and deep learning methods for time series analytics are usually

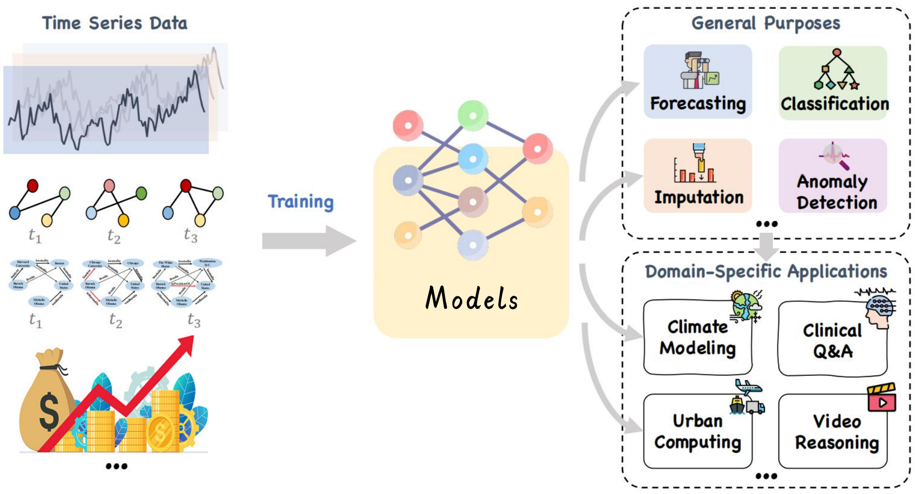
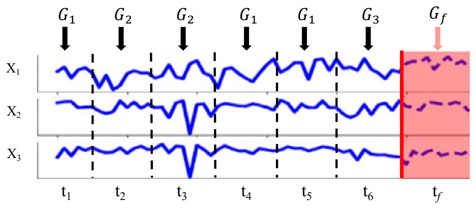


Fig. 1: Example multivariate time series data and applications [35].

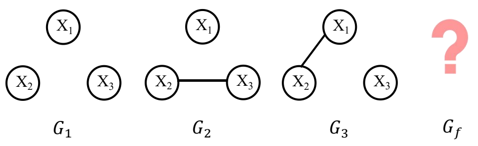
only based on data-driven models and are limited in the capacity to leverage real-world and scientific knowledge, which is essential for enhancing reliability to improve accuracy.

The first challenge lies in the real-world knowledge that the influence and relationships among various observed variates vary across time, as exemplified in Fig. 2. For instance, while existing graph neural network (GNN) [52] based methods [36, 79, 80] have shown promise in modeling the correlations among historical time series variates, they only use recent historical relationships that may not hold as the system evolves. Further, existing methods use RNN [49] or Transformer [15] based module to learn temporal dependencies. RNN based models only explicitly model the influence from one timestamp to the next timestamp, and Transformer based models only model the influence among historical timestamps. However, one historical timestamp may have different influences on future timestamps. In many real-world scenarios, the causal influence among variates changes over time [39], demanding models that can adapt to these shifts and forecast future observations more accurately.

Another pressing challenge is the knowledge of continuity and discrete-

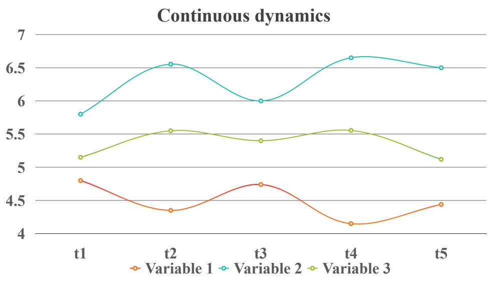


(a) An example of multivariate time series

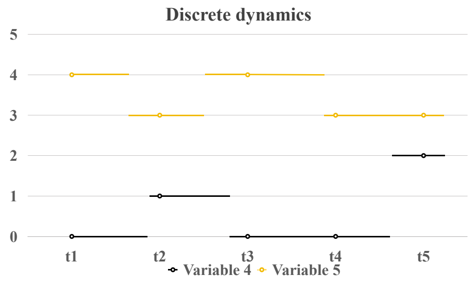


(b) The dynamic relationships

Fig. 2: The dynamic influence and relationships among variates



(a) Continuity inherent in variates.



(b) Discreteness inherent in variates.

Fig. 3: The knowledge of continuity and discreteness inherent in different variates.

ness inherent in different variates, as exemplified in Fig. 3. Continuous variates are represented by real numbers, such as latencies and temperatures. Their values change continuously over continuous time, as illustrated in Fig. 3(a), which are called the *continuous dynamics*. Discrete variates are represented by natural numbers, such as the encoding of categories, flags, and status. Their values change discretely over continuous time, as illustrated in Fig. 3(b), which is called the *discrete dynamics*. Existing methods work well for time series collected with regular time intervals but struggle to accommodate the continuous and discrete dynamics exhibited by different varieties across continuous time. Further, existing methods [21, 24, 33, 64, 67, 94, 96] struggle to recognize the importance of variates that have different measurement units. For example, the latencies are measured in seconds, and the category information is measured in natural number encoding. They simply sum up the reconstruction errors or contrastive errors from all variates to obtain the final anomaly score, whereas the reconstruction errors and contrastive errors are based on different measurement units that should not be treated indiscriminately for time series anomaly detection. Such knowledge is crucial in many real-world scenarios, demanding models that can utilize continuity and discreteness knowledge and distinctions of variates for more accurate anomaly detection.

Finally, while current anomaly detection methods can detect anomalies that deviate from expected behaviors, they frequently react only after anomalies have fully manifested, rather than identifying anomaly precursors to provide early warnings that could prevent severe future anomalies for better safety and reliability. Meanwhile, existing anomaly prediction methods rely on labeled training data for achieving acceptable accuracy, as exemplified in Fig. 4. However, such data may be difficult to obtain; and in real-time deployments, anomalies can occur that were not seen in labeled data, thus making them difficult to predict.

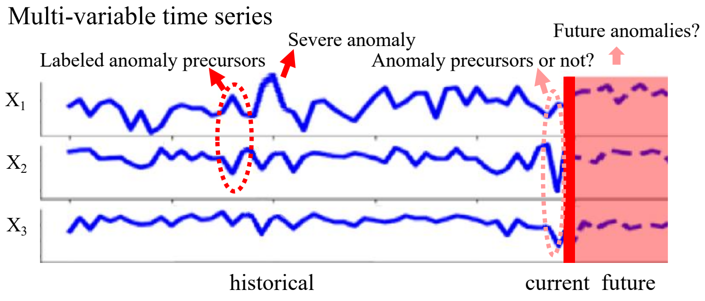

Multi-variable time series
Severe anomaly
Future anomalies?
Labeled anomaly precursors
Anomaly precursors or not?
X_1
X_2
X_3
historical
current
future

Fig. 4: Supervised anomaly prediction methods rely on labeled anomaly precursors.

The motivation for this Ph.D. thesis stems from the need to overcome these critical challenges by bridging the gap between data-driven methods and real-world and scientific knowledge. The proposed research tackles the limitations of existing methods on three tasks in time series analytics:

1. **Incorporate dynamic graph modeling and causality knowledge into multivariate time series forecasting.** We propose the MTSF-DG model that models dynamic relation graphs for multivariate time series forecasting. MTSF-DG can learn historical relation graphs and predict future relation graphs to capture the dynamic re-

lations, guided by causality knowledge and empirical covariance matrices. We further propose a reasoning network to learn temporal influences from historical timestamps and explicitly forecast each future timestamp.

2. **Incorporate differential equation knowledge for time series anomaly detection.** We propose the TAD-UP model with differential equation knowledge for time series anomaly detection via unified probabilistic modeling. We propose two co-dependent branches of neural ordinary differential equations to learn both continuous and discrete dynamics for different variates. We also propose a unified joint probability distribution modeling. The resulting model is optimized using Maximum Likelihood Estimation on joint variates. We detect anomalies using joint probabilities, which take the marginal probabilities of different variates into account as their importance on final anomaly scores.

3. **Incorporate Maximum Mean Discrepancy knowledge for unsupervised anomaly prediction.** We propose an Importance-based Generative Contrastive Learning method (IGCL) for unsupervised anomaly prediction. IGCL employs a diffusion module to produce anomaly precursor patterns, controlled by knowledge from Maximum Mean Discrepancy theory. IGCL then can predict anomalies by identifying anomaly precursors that will evolve into future anomalies. To address challenges caused by potentially complex precursor combinations involving multiple variates, we propose a memory bank with importance scores that stores representative samples adaptively and generates more complex anomaly precursors.

By addressing the above challenges with real-world and scientific knowledge, this research endeavors to enhance the performance and applicability of time series analytics in real-world applications, paving the way for more accurate and reliable analytical models.

2 Thesis Structure

Overall, three key challenges and tasks are addressed in this thesis.

First, in Chapter 1, we propose the MTSF-DG model that incorporates dynamic graph modeling into multivariate time series forecasting, which is the topic of our Paper A [90].

Second, in Chapter 2, We propose the TAD-UP model that incorporates differential equation knowledge for time series anomaly detection via unified probabilistic modeling, which is the topic of our Paper B [91].

Finally, in Chapter 3, we propose an importance-based generative contrastive learning method (IGCL) for unsupervised anomaly prediction, which is the topic of our Paper C [93].

Chapter 1

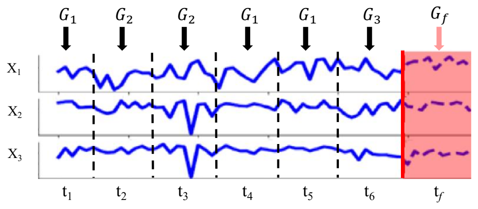
Multiple Time Series Forecasting with Dynamic Graph Modeling

This chapter provides an overall summary of Paper A [90], which proposes the MTSF-DG model that incorporates dynamic graph modeling and causality knowledge into multivariate time series forecasting. This chapter reuses content from Paper A [90] when that is considered clear, and it offers no new contributions beyond those reported in Paper A [90]. More details and experiments can be found in Paper A [90].

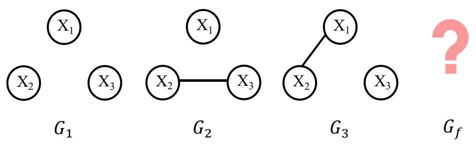
1 Motivation and Problem Statement

Real-world data often consists of time-dependent observations that form multiple time series [14, 57, 78]. For instance, in power grids, multivariate time series capture energy consumption and production [19]. Forecasting future observations based on historical data is essential for ensuring effective operations of various applications, such as predicting future patterns for improved scheduling [7, 9, 37, 41, 43, 60, 84].

Earlier approaches [49, 73] employed machine learning models that focused solely on learning *temporal dependencies* from time series, such as ARIMA [88] and Recurrent Neural Networks (RNN) [13]. Some stud-

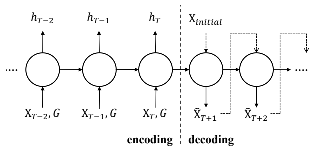


(a) An example of time series for traffic flows at three locations



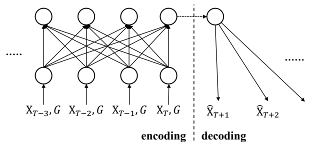
(b) The dynamic relation graphs

Fig. 1.1: Motivations



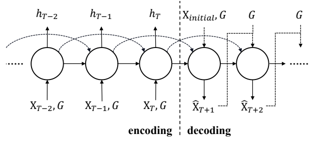
h_{T-2} h_{T-1} h_T $X_{initial}$
X_{T-2}, G X_{T-1}, G X_T, G $\hat{X}_{T+1}$ $\hat{X}_{T+2}$
encoding
decoding

(a) RNN based model



X_{T-3}, G X_{T-2}, G X_{T-1}, G X_T, G $\hat{X}_{T+1}$ $\hat{X}_{T+2}$
encoding
decoding

(b) Transformer based model



h_{T-2} h_{T-1} h_T $X_{initial}, G$ G G
X_{T-2}, G X_{T-1}, G X_T, G $\hat{X}_{T+1}$ $\hat{X}_{T+2}$
encoding
decoding

(c) Our reasoning network

Fig. 1.2: Model comparison

ies [53, 80] explored time series forecasting within the traffic domain using various Graph Neural Network (GNN) architectures [26, 47, 92], which learn correlations among time series from different locations based on their spatial proximity. Recently, a new direction has emerged involving graph learning [79] to construct a *relation graph* that models correlations among multiple time series without relying on spatial distance, enhancing forecasting accuracy. For example, DGSL [65] constructs a relation graph where time series are represented as nodes, and edges are drawn between two time series if their historical observations exhibit similarity.

Figure 1.1(a) illustrates three time series recording traffic flows in distinct districts: a commercial district $\mathbf{X}_1$, a residential district $\mathbf{X}_2$, and an industrial district $\mathbf{X}_3$. During workday peak hours ($\mathbf{t}_2$ to $\mathbf{t}_3$), traffic flows between $\mathbf{X}_2$ and $\mathbf{X}_3$ may be more closely correlated. In contrast, on weekends ($\mathbf{t}_6$), traffic flows between $\mathbf{X}_1$ and $\mathbf{X}_2$ may exhibit stronger correlations. As a result, multiple historical relation graphs, such as G_1, G_2, and G_3, are necessary to capture the dynamic correlations among time series. Additionally, correlations during a future period $\mathbf{t}_f$ may differ from historical patterns, which can be effectively represented by a future relation graph G_f, as depicted in Figure 1.1(b).

The knowledge of dynamic relation graphs can help time series forecasting. However, there are two main challenges.

Challenge 1: Existing works can only construct a static relation graph by learning the similarities or correlations among time series from the known recent data. However, none of the existing methods have demonstrated the ability to learn a future relation graph to predict observations in the future. In addition, it results in high time complexity to extract features from different relation graphs, since GNNs in previous time series forecasting methods [16, 86] can only extract features from one relation graph each time.

Challenge 2: Existing methods directly use RNN or Transformer [15] based modules to learn temporal dependencies. As shown in Figure 1.2(a) and 1.2(b), RNN based models can only explicitly model the temporal dependency from one timestamp to the following timestamp, and Transformer based models only model the temporal dependencies among historical timestamps. However, one historical timestamp may have different influence on different future timestamps since the changeable relation graphs appear at different times. The existing methods fail to model such different temporal dependencies explicitly.

We introduce MTSF-DG, a novel approach for multiple time series forecasting with dynamic graph modeling and causality knowledge, designed to address these challenges.

To tackle **Challenge 1**, we segment each time series sample into historical and future time-windows, constructing a relation graph distribution for each window. This approach not only learns the historical relation graph distribution but also predicts the future relation graph distribution by optimizing correlation coefficients derived from an empirical covariance matrix using a memory network. Unlike traditional GNNs that operate on a single graph at a time, we propose a causal GNN that effectively integrates observations from both historical and future relation graphs into a unified feature.

Addressing **Challenge 2**, we develop a reasoning network that employs logical operations and a symbolic reasoning process to explicitly learn how one historical timestamp may influence different future timestamps differently. Specifically, we derive features, such as h_{T-2}, h_{T-1}, and h_T, representing timestamps $T-2$, $T-1$, and T, respectively. As illustrated in Figure 1.2(c), we utilize a reasoning process where, for example, $h_{T-2} \rightarrow h_{T-1}$ and $(h_{T-2} \wedge h_{T-1}) \rightarrow h_T$ are evaluated as *TRUE* or *FALSE*. This approach enables the cognition ability [11, 62] to determine how a past timestamp $(T-2)$ may exert varying influences on future timestamps $(T-1$ and $T)$. Through this mechanism, we explicitly model diverse temporal dependencies.

To the best of our knowledge, this is the first work that uses both historical and future relation graphs in multiple time series forecasting. And we summarize contributions as follows.

- We propose to model dynamic relation graphs and propose a causal GNN to model the observations with both historical and future relation graphs into features efficiently.
- We propose a reasoning network to explicitly learn how historical timestamps have different influence on future timestamps.
- By evaluating on six real-world datasets from different domains, we show that our model consistently outperforms the state-of-the-art baselines.

2 Preliminaries

We formalize the multiple time series forecasting problem and introduce reasoning.

2.1 Problem definition

Multiple Time Series Forecasting. The multiple time series is represented as $\mathbf{X} \in \mathbb{R}^{N \times L}$, where N is the number of time series and each time series has observations during total L timestamps. We use $\mathbf{X}_t \in \mathbb{R}^N$ to indicate the observations of all time series at timestamp t, use $\mathbf{X}_{t:t+t_\Delta} \in \mathbb{R}^{N \times t_\Delta}$ to indicate the observations of all time series from timestamp t to timestamp $t + t_\Delta$, use $\mathbf{X}^i_{t:t+t_\Delta} \in \mathbb{R}^{t_\Delta}$ to indicate the observations of the i-th time series from timestamp t to timestamp $t + t_\Delta$, and use $\mathbf{X}_t^i \in \mathbb{R}^1$ to indicate the observations of the i-th time series at timestamp t, where $1 \leq i \leq N$ and $1 \leq t, t + t_\Delta \leq L$.

We formulate multiple time series forecasting problem as follows. Given a sub-sequence of historical p timestamps of observations from the multiple time series, *i.e.,* $\mathbf{X}_{T-p+1:T}$, the goal is to predict the values for the q future timestamps, *i.e.,* $\mathbf{X}_{T+1:T+q}$, where $q \geq 1$. Thus, we formulate the multiple time series forecasting problem as finding a mapping function $\mathcal{F}$ as follows:

$$\hat{\mathbf{X}}_{T+1:T+q} = \mathcal{F}_\theta(\mathbf{X}_{T-p+1:T}), \tag{1.1}$$

where θ is the parameters of $\mathcal{F}$, and $\hat{\mathbf{X}}$ denotes the predicted values of multiple time series.

Relation Graph. The latent correlations among time series at timestamp t are represented by a relation graph $G_t = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where: $\mathcal{V}$ is the set of nodes, with each node $TS_i \in \mathcal{V}$ representing a time series, such that $|\mathcal{V}| = N$. $\mathcal{E}$ is the set of edges, where each edge $e_{i,j} \in \mathcal{E}$ indicates that time series i and time series j are correlated. $\mathcal{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, where: $\mathcal{A}_{ij} = 0$ if $e_{i,j} \notin \mathcal{E}$ (i.e., no correlation exists between i and j). $\mathcal{A}_{ij} \neq 0$ if $e_{i,j} \in \mathcal{E}$, with $\mathcal{A}_{ij}$ representing the weight that measures the strength of the correlation between time series i and j. If $e_{i,j} \in \mathcal{E}$, then nodes i and j are considered first-hop neighbors of each other.

2.2 Reasoning

In this paper, we employ propositional logic, which includes fundamental operations such as conjunction (AND, $\wedge$), negation (NOT, $\neg$), and material implication ($\rightarrow$), to explicitly capture how historical timestamps differently influence future timestamps. For time series forecasting: Hidden states are variables, such as h_t, representing the states of multiple time series at timestamp t, similar to the hidden states used in RNNs. Expressions are operations performed over hidden states. For example, $(h_{t-1} \wedge h_t)$ signifies that multiple time series have experienced states h_{t-1} and h_t at timestamps $t-1$ and t, respectively. When an expression involves the material implication ($\rightarrow$) operation, it is referred to as a Horn clause. The reasoning outcome of a Horn clause like $(h_{t-2} \wedge h_{t-1}) \rightarrow h_t$ being $TRUE$ indicates that the historical states h_{t-2} and h_{t-1} significantly influence the

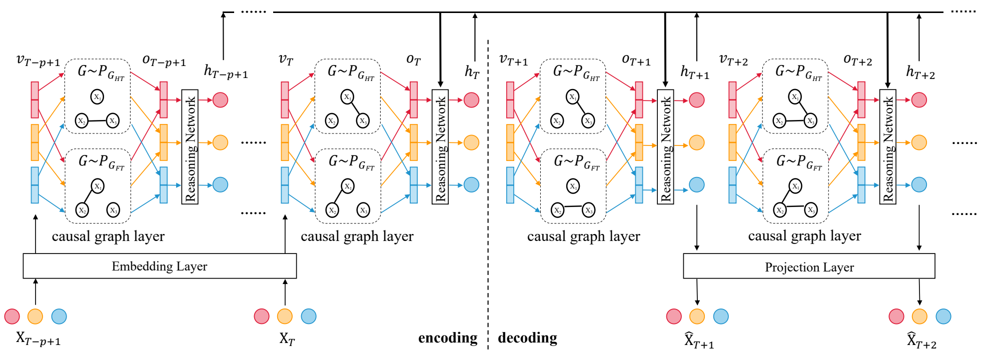


Fig. 1.3: The overall framework.

future state h_t. Conversely, if the result is $FALSE$, it implies that h_{t-2} and h_{t-1} have minimal impact on h_t.

3 Methodology

As illustrated in Figure 1.3, we present the overall framework of our method, built on an encoder-decoder architecture and comprising four key components: the embedding layer, the causal graph layer, the reasoning network, and the projection layer.

3.1 The causal graph layer

Modeling the dynamic relation graphs

We model dynamic relation graphs by learning a probability distribution over relation graphs.

For a given timestamp T, the probability distribution of relation graphs within the historical time-window (from timestamp $T - p + 1$ to T) is denoted as $G \sim P_{G_{HT}}$. Here, G represents a potential relation graph at timestamp $t \in [T - p + 1, T]$. Likewise, the probability distribution of relation graphs within the future time-window (from timestamp $T + 1$ to $T + q$) is denoted as $G \sim P_{G_{FT}}$, where G is a possible relation graph at timestamp $t' \in [T + 1, T + q]$.

We use a parameterized Bernoulli distribution [65] to model these probabilities. Specifically, for any pair of time series i and j, the probability distribution $P_{G_{HT}}$ is defined as:

$$P(\mathcal{A}_{ij} = 1) = P_h(i,j),\ P(\mathcal{A}_{ij} = 0) = 1 - P_h(i,j), \tag{1.2}$$

where $\mathcal{A}$ is the adjacency matrix of a relation graph G. Here, $P(\mathcal{A}_{ij} = 1)$ represents the probability that time series i and j are correlated, with $P_h(i,j)$ being a learnable parameter ranging from 0 to 1. A similar definition applies to $P_{G_{FT}}$.

Probability distribution of historical relation graphs. For observations $\mathbf{X}_{T-p+1:T}$ within the historical time-window, we construct a historical empirical covariance matrix $S_h \in \mathbb{R}^{N\times N}$ to measure the degree of co-variation between two time series:

$$S_h = \frac{1}{p}(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})^\top, \tag{1.3}$$

where $\overline{\mathbf{X}}_{T-p+1:T}$ represents the mean value of each time series over the p timestamps, and $\top$ denotes matrix transpose. The normalized historical correlation coefficient matrix $\rho_h \in \mathbb{R}^{N\times N}$ is then calculated as:

$$\rho_h(i,j) = \frac{S_h(i,j)}{\sqrt{S_h(i,i)S_h(j,j)}}, \ \text{for } 1 \le i,j \le N, \tag{1.4}$$

where each element $\rho_h(i,j) \in [-1,1]$ indicates the correlation between the i-th and j-th time series. A value closer to 1 (or -1) suggests a strong positive (or negative) correlation, while $\rho_h(i,j) = 0$ implies that the two time series are linearly independent. Therefore, the probability distribution of relation graphs within the historical time-window, $P_{G_{HT}}$, is obtained as follows:

$$\begin{aligned} P(\mathcal{A}_{ij} = 1) &= \begin{cases} P_h(i,j) = |\rho_h(i,j)| & \text{if } |\rho_h(i,j)| \ge \delta \\ 0 & \text{if } |\rho_h(i,j)| < \delta \end{cases} \\ P(\mathcal{A}_{ij} = 0) &= \begin{cases} 1 - P_h(i,j) = 1 - |\rho_h(i,j)| & \text{if } |\rho_h(i,j)| \ge \delta \\ 1 & \text{if } |\rho_h(i,j)| < \delta \end{cases}, \end{aligned} \tag{1.5}$$

where δ is a hyper-parameter controlling the sparsity of the relation graph, with $0 \le \delta \le 1$.

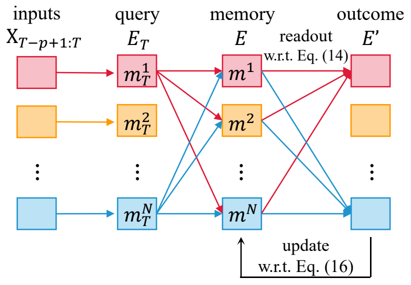


Fig. 1.4: The memory network

Probability distribution of future relation graphs. We utilize the features $E_T = (m_T^1, m_T^2, \cdots, m_T^N) \in \mathbb{R}^{N \times d^c}$, which are derived from data $\mathbf{X}_{T-p+1:T}$ and the timestamps information from $T-p+1$ to T, to predict the normalized correlation coefficient matrix $\hat{\rho}_f \in \mathbb{R}^{N \times N}$ for the future time-window. To preserve global features, we employ an additional memory unit $E \in \mathbb{R}^{N \times d^c}$, as illustrated in Figure 1.4. The primary idea is to use a memory unit to retain the correlations that have emerged among multiple time series over time.

We represent each time series i across all timestamps using a learnable embedding vector $m^i \in \mathbb{R}^{d^c}$. Consequently, the memory unit for all time series is represented as $E = (m^1, m^2, \cdots, m^N) \in \mathbb{R}^{N \times d^c}$. The recorded correlations $\rho(i,j)$ between time series i and j are obtained via the inner-product similarity between m^i and m^j, formulated as follows:

$$\rho(i,j) = m^i \cdot m^j \tag{1.6}$$

To predict the correlation coefficient matrix for the future time-window, we use the local representation matrix E_T as a query to derive the outcome feature matrix $E' \in \mathbb{R}^{N \times d^c}$ from the memory unit E as follows:

$$\begin{aligned} E' &= readout(\mathbf{Q}, \mathbf{K}, E) = \varphi(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d^c}})E \\ \mathbf{Q} &= E_t W_{\mathbf{Q}},\ \mathbf{K} = E\, W_{\mathbf{K}}, \end{aligned} \tag{1.7}$$

where $W_{\mathbf{Q}}, W_{\mathbf{K}} \in \mathbb{R}^{d^c \times d^c}$ are the parameters for extracting local and global features, $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N \times d^c}$ are the learned query and key to readout the features from the memory unit E, φ is the *softmax* function, and $E' \in \mathbb{R}^{N \times d^c}$ is the outcome feature matrix, which encompasses both local and global features. Thus, we predict the correlation coefficient matrix for the future time-window by calculating the inner-product similarity between the i-th and j-th elements of E' as follows:

$$\hat{\rho}_f(i,j) = E'(i) \cdot E'(j) \tag{1.8}$$

Finally, we update the memory unit E as follows:

$$E = \beta E + (1-\beta)E', \tag{1.9}$$

where $0 < \beta < 1$ controls the retention rate of existing correlations within the memory unit.

The memory network is optimized by minimizing the mean square error loss function given by:

$$\mathcal{L}_{graph} = \frac{1}{N \times N} \sum_{T} \sum_{1 \le i,j \le N} \Big(\hat{\rho}_f(i,j) - \rho_f(i,j) \Big)^2 \tag{1.10}$$

where:

$$\begin{aligned} S_f &= \frac{1}{p}(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})^\top, \\ \rho_f(i,j) &= \frac{S_f(i,j)}{\sqrt{S_f(i,i)S_f(j,j)}}, \quad \text{for } 1 \le i,j \le N \end{aligned} \tag{1.11}$$

Following the same approach as in Equation 1.5, we can derive the probability distribution of relation graphs for the future time-window, denoted as $P_{G_{FT}}$.

Causal graph neural network

To ensure efficiency, we sample one historical relation graph and one future relation graph at different timestamps. At each timestamp t, where $t \in [T-p+1, T]$, we sample a potential relation graph based on the probability distributions $P_{G_{HT}}$ and $P_{G_{FT}}$, respectively. The sampled relation graphs are denoted as G_{ht} and G_{ft}, with their adjacency matrices represented as $\mathcal{A}_{ht}$ and $\mathcal{A}_{ft}$. We then introduce a causal GNN to extract features from the

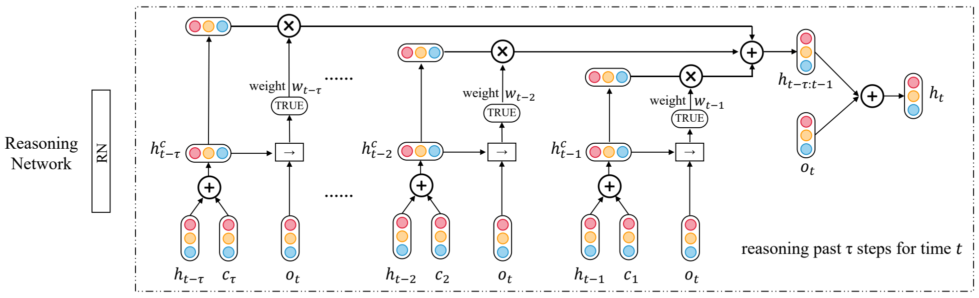


Fig. 1.5: The reasoning network will output the hidden state for multiple time series at each timestamp t using the previous states from past τ steps.

sampled historical and future relation graphs, converting the features v_t, which contains information from the observations, into the hidden states o_t.

First, we use the feature vector v^i_t from the embedding layer to learn two feature vectors: $v^i_{ht} = W_h v^i_t, \in \mathbb{R}^{\frac{d^e}{2}}$ and $v^i_{ft} = W_f v^i_t, \in \mathbb{R}^{\frac{d^e}{2}}$ for timestamp t. These features are then used to integrate the historical and future relation graphs, respectively.

Next, we use v_{ht} and v_{ft} as input features for the GNN on the historical relation graph G_{ht} and the future relation graph G_{ft}, respectively. The causal GNN on the relation graph G_{ht}, with input feature v_{ht}, denoted as $\star_{G_{ht}}(v_{ht})$, is as follows:

$$o_{ht} = \star_{G_{ht}}(v_{ht}) = \sum_{k=0}^{K} \frac{1}{k+1} \left\{ \left(\widetilde{\mathcal{A}_{ht}}\right)^k v_{ht} W_k \right\}, \tag{1.12}$$

where $\widetilde{\mathcal{A}_{ht}}$ is the adjacency matrix normalized by the diagonal degree matrix D:

$$D_{i,i} = \sum_{1 \leq j \leq N} \mathcal{A}_{ht}(i,j), \quad \widetilde{\mathcal{A}_{ht}} = D^{-1}\mathcal{A}_{ht} \tag{1.13}$$

Similarly to Equation 1.12, we obtain $o_{ft} = \star_{G_{ft}}(v_{ft})$. Finally, the hidden states $o_t \in \mathbb{R}^{N \times d^e}$ for all time series, which integrates the information from the historical observations along with both the historical and future relation graphs, is expressed as $o_t = (o_{ht} || o_{ft})$.

3.2 The reasoning network

Up to this point, we have obtained the hidden state o_t, which contains the features at each timestamp t individually. Next, we introduce a reasoning network as depicted in Figure 1.2(c), which learns a feature h_t not only from the current hidden state o_t but also from its previous τ timestamps' features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, for each timestamp t in a recurrent manner. Specifically, h_t captures how the past τ timestamps influence the current timestamp t, where $1 \le \tau \le p$ is a hyperparameter that governs the number of previous timestamps considered in the reasoning network, balancing efficiency and effectiveness. Since this is a recurrent process, when reasoning for timestamp t to obtain h_t, we have already computed h_{t-k} for previous timestamps, with $1 \le k \le \tau$.

As shown in Figure 1.5, the reasoning network first combines each feature h_{t-k} with a positional embedding ck, which encodes the relative time interval from each past timestamp $t-k$ to the current timestamp t, producing a feature h^c_{t-k}. Then, the reasoning network evaluates the importance of the previous timestamp $t-k$ on the current timestamp t by assessing the horn clause, i.e., $h^c_{t-k} \to o_t$, and outputs the evaluation result as the importance weight w_{t-k}. Finally, the reasoning network generates the feature $h_t \in \mathbb{R}^{N \times d^e}$ for the current timestamp t, which is used for forecasting the observations, by integrating all previous features h^c_{t-k} with their corresponding importance weights w_{t-k} as follows:

$$h_t = \sum_{1 \le k \le \tau} w_{t-k} \times h^c_{t-k} + o_t, \tag{1.14}$$

such that h_t captures the information of how the time series contribute to the current hidden state o_t based on the previous features $(h_{t-1}, h_{t-2}, \cdots, h_{t-\tau})$.

Now, we provide a more detailed explanation of our reasoning network's procedure.

First, it is important to note that, given the previous features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, all the time series contribute to the hidden state o_t for $t \le T$. As a result, we can state that $(h_{t-\tau} \wedge \cdots \wedge h_{t-2} \wedge h_{t-1}) \to o_t$ is TRUE and $(h_{t-\tau} \wedge \cdots \wedge h_{t-2} \wedge h_{t-1}) \to \neg o_t$ is FALSE. Here, the conjunction operation $\wedge$ combines the features from multiple time series at previous timestamps,

$\rightarrow o_t = \mathit{TRUE}$ means that all the previous features together will lead to the current hidden state o_t, $\neg o_t$ refers to the opposite of o_t, and $\rightarrow \neg o_t = \mathit{FALSE}$ means that, under the previous features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, the multiple time series will not contribute to the hidden state $\neg o_t$.

To capture the sequential nature of the time-dependent previous features, we introduce τ learnable matrices $c_1, c_2, \cdots, c\mathcal{T} \in \mathbb{R}^{N \times d^e}$, which serve as the positional embedding [75]. Each matrix c_k learns to model the relative time interval from the past timestamp $t-k$ to the current timestamp t. This allows us to derive time-aware features:

$$h^c_{t-k} = h_{t-k} + c_k, \text{ for } 1 \leq k \leq \tau, \tag{1.15}$$

and model the historical sequential information by:

$$\begin{aligned} (h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) &\rightarrow o_t \text{ is } \mathit{TRUE}, \\ (h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) &\rightarrow \neg o_t \text{ is } \mathit{FALSE}, \end{aligned} \tag{1.16}$$

where $t \leq T$.

Then, by evaluating whether each horn clause

$$h^c_{t-k} \rightarrow o_t \tag{1.17}$$

is TRUE, we can determine if the previous feature h^c_{t-k} contributes to the current hidden state o_t. Specifically, if $h^c_{t-k} \rightarrow o_t$ is TRUE, it indicates that the previous feature h^c_{t-k} is the cause for the multiple time series to enter the current state o_t. Conversely, if $h^c_{t-k} \rightarrow o_t$ is FALSE, it implies that the previous feature h^c_{t-k} does not contribute to the multiple time series entering the current state o_t.

Now, we model Equations (1.16) and (1.17) using our neural reasoning network, which performs the aforementioned logic operations, i.e., $\neg$, $\wedge$, and $\rightarrow$, through neural logic modules. First, we use h_a, h_b, and h_c to represent any features, such as h^c_{t-k}, and each logic operation is computed by a neural logic module with fully connected layers as follows:

$$\begin{aligned} \neg h_a &= -h_a \\ h_a \wedge h_b \wedge \cdots \wedge h_c &= \sigma\Big((h_a \odot h_b \odot \cdots \odot h_c) W_a\Big) \\ h_a \rightarrow h_b &= \sigma\Big((h_a || h_b) W_i\Big), \end{aligned} \tag{1.18}$$

where $h_a, h_b \in \mathbb{R}^{N \times d^e}$ represent the features used in the calculation, while $W_a \in \mathbb{R}^{d^e \times d^e}$ and $W_i \in \mathbb{R}^{2d^e \times d^e}$ are the learnable network parameters for performing the logic operations $\wedge$ and $\rightarrow$, respectively. Additionally, $\odot$ denotes the Hadamard product, which performs element-wise multiplication of matrices.

Thus, all logical expressions, like Equation 1.16, can be computed step by step using the neural modules based on Equations 1.18. For example, taking Equation 1.16, $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t$, we can compute it as follows:

$$\begin{aligned} \mathbf{Exp}_1 &= (h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \\ &= \sigma\Big((h^c_{t-\tau} \odot \cdots \odot h^c_{t-2} \odot h^c_{t-1}) W_a\Big) \\ \mathbf{Exp}_+ &= \mathbf{Exp}_1 \rightarrow o_t = \sigma\Big((\mathbf{Exp}_1 || o_t) W_i\Big), \end{aligned} \tag{1.19}$$

where $\mathbf{Exp}_+ \in \mathbb{R}^{N \times d^e}$ is the final outcome of logical expression $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t$. In the same way, we can get the final outcome of logical expression $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow \neg o_t$ as $\mathbf{Exp}_- \in \mathbb{R}^{N \times d^e}$.

Then, another neural module, denoted as $isT()$, is used to evaluate whether an expression $\mathbf{Exp}_a$ is *TRUE* or *FALSE* by:

$$isT(\mathbf{Exp}_a) = sigmoid(\mathbf{Exp}_a W_r), \tag{1.20}$$

where $\mathbf{Exp}_a \in \mathbb{R}^{N \times d^e}$ represents the outcome of a logical expression, such as $\mathbf{Exp}_+$, used for the *TRUE*/*FALSE* evaluation. $W_r \in \mathbb{R}^{d^e \times 1}$ is the learnable parameter for evaluating the logical expression, and $isT(\mathbf{Exp}_a)$ denotes the result of the evaluation. The *sigmoid* function ensures that the evaluation result is within the range of 0 to 1, where $isT(\mathbf{Exp}_a) = 0$ indicates that the logical expression is *FALSE*, and $isT(\mathbf{Exp}_a) = 1$ indicates that the logical expression is *TRUE*.

To ensure the truth as defined by Equation 1.16, we introduce the logical regularization, which is minimized by the following loss function:

$$\mathcal{L}_{reg} = 1 - isT(\mathbf{Exp}_+) + isT(\mathbf{Exp}_-) \tag{1.21}$$

Here, $isT(\mathbf{Exp}_+)$ being 1 indicates that with the previous features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, the multiple time series indeed transitioned into the hidden state o_t, and $isT(\mathbf{Exp}_-)$ being 0 indicates that the multiple time series did not transition into the hidden state $\neg o_t$.

Next, we evaluate whether the previous feature h^c_{t-k} contributes to the current hidden state o_t by using the following expression:

$$\mathbf{Exp}_{t-k} = (h^c_{t-k} \rightarrow o_t) = (h^c_{t-k}||o_t)W_i \tag{1.22}$$

$$w_{t-k} = isT(\mathbf{Exp}_{t-k}), \tag{1.23}$$

where w_{t-k} represents the importance weight. The closer w_{t-k} is to 1, the more likely it is that the previous feature h^c_{t-k} is the cause of the multiple time series transitioning into the current state o_t.

Finally, the reasoning network generates the feature $h_t \in \mathbb{R}^{N \times d^e}$ for the current timestamp t by integrating all previous features h^c_{t-k} with their corresponding importance weights w_{t-k} using Equation 1.14.

3.3 The projection layer

The projection layer outputs the forecasting observations $\hat{\mathbf{X}}_{T+1} \in \mathbb{R}^{N \times 1}$:

$$\hat{\mathbf{X}}_{T+1} = h_{T+1}W_p \tag{1.24}$$

We employ the objective function to facilitate model learning via gradient descent. The memory network is optimized by minimizing Eq. (1.10). The embedding layer, causal GNN, reasoning network, and projection layer are optimized using the mean absolute error (MAE):

$$\mathcal{L}_{MAE} = \frac{1}{N \times q}\left|\left|\hat{\mathbf{X}}_{T+1:T+q} - \mathbf{X}_{T+1:T+q}\right|\right|^1 \tag{1.25}$$

4 Experiments

4.1 Experimental settings

Datasets.

We use a collection of datasets from the traffic and energy domains to assess the performance of multi-step time series forecasting:

- METR-LA and PEMS-BAY: These are traffic speed time series datasets, released by Li et al. [53].

- PEMS04 and PEMS08: These are traffic flow time series collected from the Caltrans Performance Measurement System (PEMS), released by Bai et al. [4].
- Solar-Energy: A dataset focused on solar energy, released by Lai et al. [49].
- Electricity: The electricity consumption dataset, released by Lai et al. [49].

Evaluation Metrics.

We evaluate multi-step forecasting using mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE). For single-step forecasting, we use Root Relative Squared Error (RRSE) and Empirical Correlation Coefficient (CORR).

Baselines.

We compare MTSF-DG with the following baseline methods:

- **Methods without relation graphs.** VAR-MLP: An auto-regressive model using multilayer perceptron (MLP) [88]. GP: A Gaussian Process model for time series forecasting [73]. LSTNet: Combines convolutional neural networks (CNN) with RNN to learn temporal dependencies [49]. TPA: A basic Transformer model [68]. FEDFormer: A frequency-enhanced Transformer that extracts trend and periodic features [72]. Crossformer: A state-of-the-art Transformer-based model that uses cross-dimensional attention to capture historical correlations without learning relation graphs [89].
- **Methods with a pre-defined graph.** DCRNN: Proposes diffusion graph convolutions to model spatial dependencies [53]. GWave: Uses 1D dilated CNNs combined with gated diffusion graph convolutions [80]. AGCRN: An adaptive recurrent graph convolution network [4]. MSDR: Utilizes attention-based graph convolutions and multi-step RNN [55].
- **Methods that learn relation graphs.** MTGNN: Learns a static relation graph to model similarities among time series, using graph convolutions and CNN for forecasting [79]. DGTS: Builds a static relation

Table 1.1: Accuracy of traffic domain forecasting

Data	q	Metric	DCRNN	GWave	AGCRN	MTGNN	STFGNN	Cformer	FEDFormer	MSDR	ESG	MTSF-DG
METR-LA	3rd	MAE	2.77	2.69	2.83	2.69	2.70	2.69	2.89	2.71	2.68	**2.62**
		RMSE	5.38	5.15	5.45	5.18	5.35	5.17	5.51	5.18	5.15	**5.11**
		MAPE	7.30%	6.90%	7.56%	6.86%	7.21%	6.88%	7.63%	7.08%	6.93	**6.78%**
	6th	MAE	3.15	3.07	3.20	3.05	3.10	3.05	3.27	3.09	3.06	**2.98**
		RMSE	6.45	6.22	6.55	6.17	6.36	6.18	6.56	6.33	6.19	**6.13**
		MAPE	8.80%	8.37%	8.79%	8.19%	8.60%	8.21%	8.87%	8.57%	8.20%	**8.14%**
	12th	MAE	3.60	3.53	3.58	3.49	3.51	3.48	3.69	3.50	3.49	**3.39**
		RMSE	7.60	7.37	7.41	7.23	7.46	7.27	7.66	7.33	7.23	**7.16**
		MAPE	10.50%	10.01%	10.13%	9.87%	10.05%	9.86%	10.44%	9.98%	9.96%	**9.58%**
PEMS-BAY	3rd	MAE	1.38	1.30	1.35	1.32	1.34	1.31	1.47	1.32	1.31	**1.28**
		RMSE	2.95	2.74	2.83	2.79	2.81	2.75	3.02	2.84	2.74	**2.67**
		MAPE	2.90%	2.73%	2.87%	2.77%	2.84%	2.72%	2.96%	2.77%	2.76%	**2.63%**
	6th	MAE	1.74	1.63	1.69	1.65	1.66	1.63	1.81	1.64	1.63	**1.57**
		RMSE	3.97	3.70	3.81	3.74	3.76	3.69	4.01	3.78	3.71	**3.63**
		MAPE	3.90%	3.67%	3.84%	3.69%	3.83%	3.66%	3.99%	3.68%	3.69%	**3.56%**
	12th	MAE	2.07	1.95	1.96	1.94	1.98	1.93	2.06	1.94	1.92	**1.85**
		RMSE	4.74	4.52	4.52	4.49	4.52	4.45	4.78	4.51	4.42	**4.35**
		MAPE	4.90%	4.63%	4.67%	4.53%	4.73%	4.49%	4.88%	4.55%	4.52%	**4.40%**
PEMS 04	1∼12	MAE	24.70	19.16	19.83	19.32	19.83	19.50	23.48	19.29	19.47	**18.67**
		RMSE	38.12	30.46	32.26	31.57	31.88	32.00	37.27	31.54	31.66	**30.17**
		MAPE	17.12%	13.26%	12.97%	13.52%	13.02%	13.07%	15.44%	12.89%	13.30%	**12.64%**
PEMS 08	1∼12	MAE	17.86	15.13	15.95	15.71	16.64	15.88	17.24	15.11	15.70	**14.80**
		RMSE	27.83	24.07	25.22	24.62	26.22	25.07	26.93	24.42	24.81	**23.68**
		MAPE	11.45%	10.10%	10.09%	10.03%	10.60%	10.17%	11.21%	9.93%	10.07%	**9.54%**

graph based on Euler distances among time series and uses recurrent graph convolutions [65]. STFGNN: Constructs a static relation graph using Dynamic Time Warping similarities among time series [39, 52]. ESG: Divides historical observations into sub-time windows, learns a historical relation graph for each window, and uses RNN for forecasting [86].

4.2 Overall comparison.

Tables 1.1 and 1.2 present the accuracy of MTSF-DG and the baseline methods across all datasets. Each method is repeated randomly five times, and the average result is reported. The highest accuracy is marked in bold, while the second-best, significantly outperformed accuracy is underlined.

The key observations are as follows:

First, MTSF-DG consistently outperforms state-of-the-art baseline methods across all datasets, demonstrating that it effectively learns dynamic correlations among multiple time series and leverages these correlations to improve forecasting performance.

Second, as shown in Table 1.2, methods such as MTGNN, DGTS, ESG, and MTSF-DG, which can learn relation graphs for multiple time series,

Table 1.2: Accuracy of energy domain forecasting

Dataset		Solar-Energy		Electricity	
		q		q	
Method	Metric	3rd	12th	3rd	12th
VAR-MLP	RRSE	0.1922	0.4244	0.1393	0.1557
	CORR	0.9829	0.9058	0.8708	0.8192
GP	RRSE	0.2259	0.5200	0.1500	0.1621
	CORR	0.9751	0.8518	0.8670	0.8394
LSTNet	RRSE	0.1843	0.3254	0.0864	0.1007
	CORR	0.9843	0.9467	0.9283	0.9077
TPA	RRSE	0.1803	0.3234	0.0823	0.0964
	CORR	0.9850	0.9487	0.9439	0.9250
MTGNN	RRSE	0.1778	0.3109	0.0745	0.0916
	CORR	0.9852	0.9509	0.9474	0.9278
DGTS	RRSE	0.1791	0.3144	0.0767	0.0925
	CORR	0.9852	0.9501	0.9470	0.9275
Crossformer	RRSE	0.1772	0.3089	0.0741	0.0905
	CORR	0.9859	0.9511	0.9474	0.9291
FEDFormer	RRSE	0.1788	0.3141	0.0769	0.0924
	CORR	0.9852	0.9498	0.9465	0.9280
ESG	RRSE	0.1708	0.3073	0.0718	0.0898
	CORR	0.9865	0.9519	0.9494	0.9321
MTSF-DG	RRSE	**0.1692**	**0.3025**	**0.0701**	**0.0882**
	CORR	**0.9874**	**0.9533**	**0.9502**	**0.9339**

perform better than methods like VAR-MLP, GP, LSTNet, TPA, and FEDFormer, which cannot capture these relationships. Crossformer, which uses Cross-Transformer to learn inter-series correlations, also surpasses VAR-MLP, GP, LSTNet, TPA, and FEDFormer. This highlights the importance of capturing inter-series relationships for effective multiple time series forecasting.

Third, MTSF-DG shows superior performance compared to other graph-based methods that rely on a single relation graph or exclusively use historical relation graphs. This advantage stems from the baselines' inability to capture the dynamic correlations among multiple time series, which can change over time and differ in the future. Different future relation graphs can affect future observations in distinct ways, making reliance on a single or historical graph prone to bias. Although methods like GWave, MTGNN, STFGNN, MSDR, and ESG achieve second-best accuracy on certain datasets, no single baseline consistently outperforms others. This suggests that relying solely on a single relation graph or historical relation graphs is inadequate for robust multi-time series forecasting. In contrast, MTSF-DG dynamically learns both historical and future correlations, consistently outperforming the baselines.

Finally, MTSF-DG achieves the highest accuracy compared to Transformer-based models, indicating that the reasoning network effectively learns temporal dependencies by explicitly modeling how historical timestamps influence future timestamps. This capability allows MTSF-DG to outperform TPA, FEDFormer, and Crossformer.

Chapter 2

TAD-UP: Learning Continuous and Discrete Dynamics for Time Series Anomaly Detection via Unified Probabilistic Modeling

This chapter provides an overall summary of Paper B [91], which incorporates continuity and discreteness knowledge into anomaly detection. This chapter reuses content from Paper B [91] when that is considered clear, and it offers no new contributions beyond those reported in Paper B [91]. More details and experiments can be found in Paper B [91].

1 Motivation and Problem Statement

Cyber-physical systems employ sensors to monitor their environment and runtime status, resulting in sequences of timestamped multivariate time series [9, 14, 56, 87], as exemplified in Figure 2.1. Effectively detecting anomalies in such time series is essential in many real-world applications, as it supports fault analyses and facilitates risk monitoring to ensure normal operations of cyber-physical systems, thereby avoiding economic losses and

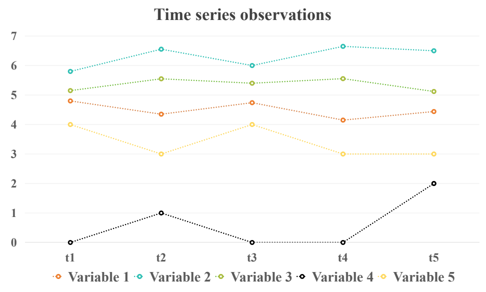
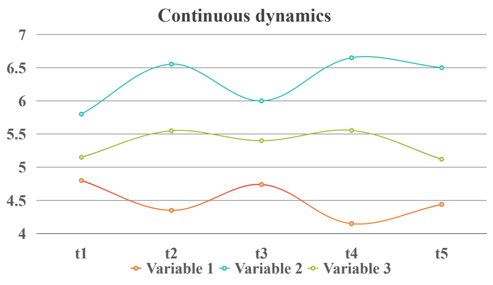
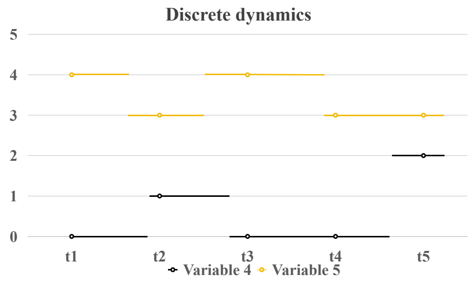


Fig. 2.1: Example timestamped multivariate time series.



(a) Continuous variates over continuous time.



(b) Discrete variates over continuous time.

Fig. 2.2: Our motivations.

health concerns [10, 29, 33]. For example, detecting anomalies can help locate and handle service faults and interruptions threats in computing systems [2, 29] and can warn against environmental [58], health [10], and financial risks [94].

The multiple variates in time series can be categorized into two types, *i.e.*, continuous variates and discrete variates, as shown in Figure 2.2. Continuous variates are exemplified by real numbers, such as latencies, CPU usage rates, and temperatures. Their values change continuously, as illustrated in Figure 2.2(a), which is called *continuous dynamics*. Discrete variates are exemplified by natural numbers. These include encodings of categories, flags, and status information. Their values change discretely, as illustrated in Figure 2.2(b), which is called *discrete dynamics*. Classic [6, 27, 56, 87] and deep learning-based [8, 21, 63, 96] anomaly detection methods that aim to learn the normal patterns from unlabeled data achieve good results when applied to time series with regular discrete timestamps. However, all existing methods [64, 94] remain unable to learn both continuous and discrete dynamics for time series variates along continuous time [28].

In addition, different variates have different physical and mathematical measurement units. For example, latencies are measured in seconds, CPU usage rates are measured in percentages, temperatures are measured in degrees Celsius, and category information is measured using natural number encoding. Deep learning-based methods [21, 24, 33, 64, 67, 94, 96] cannot recognize the importance of each variate to detect anomalies. Existing methods simply sum up reconstruction errors or contrastive errors from all variates to obtain the final anomaly score and determine timestamps with anomalous data, whereas the reconstruction errors and contrastive errors are based on different measurement units that should not be treated indiscriminately for time series anomaly detection.

Without learning continuous and discrete dynamics and recognizing the different importance of different variates, existing methods have limited modeling abilities and fall short in detecting anomalies in real-world scenarios. Overall, there are two challenges. **Challenge 1:** How to learn continuous and discrete dynamics that have completely different forms of behaviors along time? Further, continuous and discrete dynamics may also have correlations with each other as different variates may influence on each other. As exemplified in Figure 2.2, variate 1 and variate 5 have correlations. **Challenge 2:** Existing methods cannot learn the importance units of different variates on the final anomaly scores.

To contend with these challenges, we propose **TAD-UP** that features novel neural co-dependent ordinary differential equations (co-ODEs). To address **Challenge 1**, we propose two dependent branches of neural ordinary differential equations (NODEs) to learn the interactive continuous and discrete dynamics together from different time series variates. The first branch, *i.e.,* continuous co-ODE, learns continuous dynamics. We propose a Compound Poisson Process for the second branch, *i.e.,* discrete co-ODE, to learn discrete dynamics. Further, we propose a gate temporal convolution networks (TCNs) to learn correlations among discrete variates and continuous variates. To address **Challenge 2**, we introduce joint probability distribution modeling and obtain the final anomaly scores from a unified probabilistic space. Specifically, we use a multi-variate Gaussian distribution to model the observation probabilities of continuous variates and the softmax distribution to model the observation probabilities of discrete variates. We optimize the model with Maximum Likelihood Estimation in the unified probabilistic space, instead of using reconstruction losses or contrastive losses for variates in different semantic spaces.

We summarize our contributions as follows.

- We are the first to learn both continuous and discrete dynamics for multivariate time series anomaly detection.
- We propose two co-dependent branches of neural ordinary differential equations to learn interactive continuous and discrete dynamics effectively.
- We introduce joint probability distribution modeling and optimize our model with Maximum Likelihood Estimation, which helps obtain the anomaly scores that consider the importance of different variates.
- Experiments on nine benchmarks from different domains offer evidence that the method can achieve state-of-the-art performance, even compared with the large pre-trained model DADA.

2 Preliminaries

We first formalize the time series anomaly detection problem. Then, we introduce the background of neural differential equations.

2.1 Definitions

A Multi-variate Time Series is represented as $(\mathbf{C}, \mathbf{D})$, where $\mathbf{C} = \langle \mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_T \rangle \in \mathbb{R}^{N_C \times T}$ records the observations of the continuous variates by real numbers in $\mathbb{R}$, and $\mathbf{D} = \langle \mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_T \rangle \in \mathbb{N}^{N_D \times T}$ records the observations of the discrete variates by natural numbers in $\mathbb{N}$. N_C and N_D are the number of continuous and discrete variates observed at each timestamp, respectively, T is the total number of timestamps, $\mathbf{c}_i \in \mathbb{R}^{N_C}$

and $\mathbf{d}_i \in \mathbb{N}^{N_D}$ are the observations at timestamp t_i and $i \in \{1, 2, \cdots, T\}$. Taking Figure 2.1 as an example, we have $T = 5$, $N_C = 3$, and $N_D = 2$, and there are 3 continuous variates and 2 discrete variates observed at each timestamp. In the special case where $N_D = 0$, $\mathbf{C}$ is the time series as defined in previous works [8, 33, 67]. We use $\mathbf{C}^j \in \mathbb{R}^T$, $\mathbf{c}_i^j \in \mathbb{R}^1$, $\mathbf{D}^j \in \mathbb{N}^T$ and $\mathbf{d}_i^j \in \mathbb{N}^1$ to indicate the observations of the j-th variate with superscripts, respectively.

Time Series Anomaly Detection. Anomaly detection identifies which timestamp in the time series exists the anomaly, *i.e.,* a binary classification of the anomaly and normal for each timestamp. With sliding windows, anomaly detection takes as the input time series $\{\mathbf{C}, \mathbf{D}\}$, and outputs the binary vector $\mathbf{Y} = \langle y_1, y_2, \cdots, y_T \rangle$, where $y_i \in \{0, 1\}$ and $y_i = 1$ indicates an anomaly at timestamp t_i.

2.2 Neural differential equations

Chen *et al.* [12] first propose a new paradigm of learning continuous dynamics and modeling continuous time with ordinary differential equation-based neural networks, which then have been widely studied in time series forecasting recently [34, 36]. Unlike traditional time series neural network models, which only learn hidden states among discrete timestamps, *e.g.,* recurrent neural networks (RNNs) and TCNs, differential equations can learn the dynamics of the hidden state features $\mathbf{H}(t)$ by the integral along the continuous time with $\frac{d\mathbf{H}(t)}{dt}$. Neural ordinary differential equations (NODEs) directly use the integral to learn the hidden state feature $\mathbf{H}(t)$ at any continuous time t as follows:

$$\begin{aligned} \mathbf{H}(t) &= \mathbf{H}(0) + \int_0^t \frac{d\mathbf{H}(s)}{ds} ds \\ \frac{d\mathbf{H}(t)}{dt} &= f\Big(\mathbf{H}(t); \mathbf{\Theta}\Big) \end{aligned} \tag{2.1}$$

Here, $\mathbf{H}(0)$ is the hidden state feature that denotes the initial condition of NODEs where an embedding layer maps the input time series into the hidden state feature $\mathbf{H}(0)$, $f(\cdot; \mathbf{\Theta})$ denote the neural networks with parameters $\mathbf{\Theta}$ which estimates the continuous dynamical differentiation of $\mathbf{H}(t)$ along the continuous time t. Then, with the ODESolver($\cdot$) algorithm [34], we can obtain features $\mathbf{H}(t)$ at any continuous time t that could be used in downstream tasks:

$$\mathbf{H}_t = \text{ODESolver}\Big(\mathbf{H}(0), f, t; \mathbf{\Theta}\Big) \tag{2.2}$$

Recently, to improve the performance and robustness of the original NODEs, neural controlled differential equations (NCDEs) are proposed, which utilize the Riemann-Stieltjes integral [33, 40] to control the dynamics of hidden state feature $\mathbf{H}(t)$ at any continuous time t with the time series observations $\mathbf{C}$ as follows:

$$\begin{aligned} \mathbf{H}(t) &= \mathbf{H}(0) + \int_0^t \frac{d\mathbf{H}(s)}{ds} d\mathbf{C}(s) \\ &= \mathbf{H}(0) + \int_0^t f\Big(\mathbf{H}(s); \mathbf{\Theta}\Big) \frac{d\mathbf{C}(s)}{ds} ds \end{aligned} \tag{2.3}$$

$\mathbf{C}(t)$ is the continuous time series path across continuous time, which is interpolated from the original time series observations $\mathbf{C} = \langle \mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_T \rangle \in \mathbb{R}^{N_C \times T}$ recorded at discrete timestamps with the natural cubic spline interpolation pre-processing [33, 40]. Next, $\frac{d\mathbf{C}(s)}{ds}$ is the differential coefficient in the time series path.

The difference between NODEs and NCDEs is the continuous time series path $C(t)$, whose differential coefficient will control the dynamics. However, NCDEs cannot model long input time series effectively and efficiently, as NCDEs have a recurrent architecture as in RNNs and the natural cubic spline interpolation pre-processing will bring additional space and time complexity.

On the other hand, both NODEs and NCDEs cannot learn the discrete dynamics from the discrete variates, and they cannot model the interactions among different dynamics either.

In contrast, as shown in the following sections in this paper, we propose neural co-dependent ordinary differential equations, which are free from interpolation pre-processing. We further propose the Compound Poisson Process to learn the discrete dynamics and propose the dependent latent features to help learn the influence among dynamics of different variates.

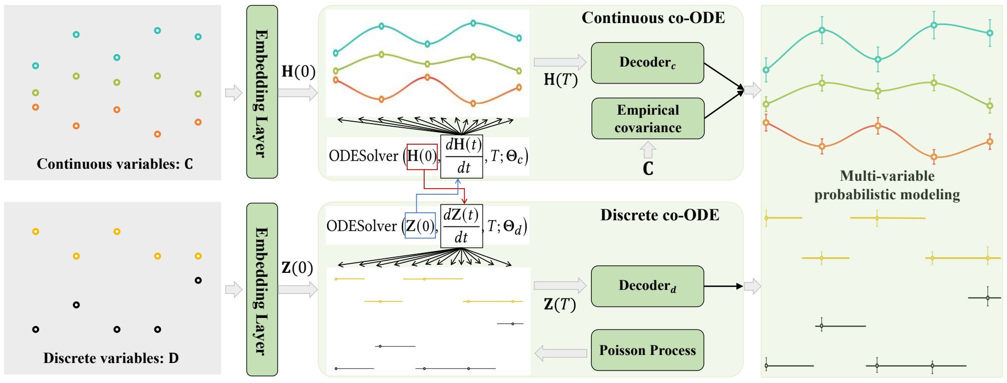


Fig. 2.3: The overall architecture.

3 Methodology

3.1 Overall architecture

We present the overall architecture in Figure 2.3, which consists of the embedding layers, continuous co-ODE, discrete co-ODE, and multivariate probabilistic modeling.

The embedding layers take as the input time series $\{\mathbf{C}, \mathbf{D}\}$, first apply Instance Normalization [74] on the observations of continuous variates, *i.e.*, each $\mathbf{C}^j$ being normalized, to learn stable features [44], and then map the observations $\mathbf{c}_i$ and $\mathbf{d}_i$ at each timestamp t_i into d-dimensional vectors $\mathbf{h}_i \in \mathbb{R}^d$ and $\mathbf{z}_i \in \mathbb{R}^d$ to extract the dense features for discrete timestamps. Note that the embedding layer for continuous variates is the fully connected feed-forward layer (FC):

$$\begin{aligned} \mathbf{h}_i &= \mathrm{FC}(\mathbf{c}_i), \text{ for } i \in \{1, 2, \cdots, T\} \\ \mathbf{H}(0) &= (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T), \end{aligned} \tag{2.4}$$

where $\mathbf{H}(0) \in \mathbb{R}^{d\times T}$ denotes the hidden state feature of the initial condition for our continuous co-ODE. The layer for discrete variates is the lookup embedding layer:

$$\begin{aligned}&\mathbf{d}^j = \text{one-hot}(\mathbf{d}^j),\ \text{for } j \in \{1, 2, \cdots, N_D\}\\&\mathbf{z}_i = \text{Embedding}(\mathbf{d}_i),\ \text{for } i \in \{1, 2, \cdots, T\}\\&\mathbf{Z}(0) = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T),\end{aligned} \tag{2.5}$$

where $\mathbf{Z}(0) \in \mathbb{R}^{d\times T}$ denotes the hidden state feature of the initial condition for our discrete co-ODE.

3.2 The continuous co-ODE

We introduce the composition of our continuous co-ODE, which can learn fine-grained and continuous temporal dynamics. Given the shortcomings of RNNs, like high time-complexity and gradient explosion [36], we choose temporal convolution networks as the function $f(\cdot;\mathbf{\Theta})$ to estimate the continuous dynamical differentiation of $\mathbf{H}(t)$ along the continuous time t:

$$\begin{aligned}&\mathbf{H}(t) = \mathbf{H}(0) + \int_0^t \frac{d\mathbf{H}(s)}{ds}ds\\&\frac{d\mathbf{H}(t)}{dt} = \tanh\Big(\text{TCN}\big(\mathbf{H}(t); r, k\big)\Big),\end{aligned} \tag{2.6}$$

where $\text{TCN}(\cdot; r, k)$ is the Conv1d temporal convolution, which is performed along the last dimension of $\mathbf{H}(t)$, *i.e.,* the time dimension; $\tanh(\cdot)$ is the hyperbolic tangent activation function; r and k are the dilation factor and the convolution kernel, respectively.

In order to learn the influence among different variates, we propose a gated TCN architecture as the attention mechanism to control the amount of information flows:

$$\begin{aligned}\frac{d\mathbf{H}(t)}{dt} =& \tanh\Big(\text{TCN}_1\big(\mathbf{H}(t)\big)\Big) \odot \sigma\Big(\text{TCN}_{\text{gate}_1}\big(\mathbf{H}(t)\big)\Big)\\&\odot \sigma\Big(\text{TCN}_{\text{gate}_2}\big(\mathbf{Z}(0)\big)\Big),\end{aligned} \tag{2.7}$$

where $\odot$ denotes the element-wise Hadamard product, $\text{TCN}_{\text{gate}_1}$ and $\text{TCN}_{\text{gate}_2}$ are the gate convolutions used for the continuous variates and discrete variates, respectively, and $\sigma(\cdot)$ is the sigmoid activation function to control the amount of information flows among different variates for the continuous dynamics branch. Then, with ODESolver$(\cdot)$ [34], we can obtain the final hidden state feature $\mathbf{H}(T)$ which learns co-dependent continuous dynamics:

$$\mathbf{H}_T = \text{ODESolver}\left(\mathbf{H}(0), \frac{d\mathbf{H}(t)}{dt}, T; \mathbf{\Theta}_c\right) \tag{2.8}$$

3.3 The discrete co-ODE

The NODEs and NCDEs used in the previous work PAD [33], as shown in Section 2.2, cannot learn the discrete dynamics where the hidden state features could jump along the continuous time. Thus, we propose to utilize a Compound Poisson Process [34] to learn the discrete dynamics. The Poisson Process is a time point generative model where the output contains a sequence of discrete eventual time points $\mathcal{H} = \{\tau_{t_i}\}_{i\in\{0,1,\cdots,T\}}$. The τ_{t_i} denotes there is a discrete eventual time point at timestamp t_i, and no eventual time points between timestamps t_{i-1} and t_i.

A Poisson Process can be defined by a counting function $\mathbf{N}(t)$ along continuous time t. $\mathbf{N}(t)$ records the number of eventual time point before time t, which is described as follows:

$$\mathbf{N}(t) = \sum_{i\in\{0,1,\cdots,T\}} He(t-t_i), \text{ where } He(t) = \begin{cases} 0 & t \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.9}$$

The $He(\cdot)$ is the Heaviside step function. The probability of the next eventual time point following τ_{t_i} usually depends on the previous contexts, known as the rate or intensity of the Poisson Process. Such temporal dependencies can be described by a conditional intensity function $\lambda(t)$. Let $\mathcal{H}_t$ denote the previous contexts of eventual time points up to but not including time t. Then $\lambda(t)$ defines the probability of observing the next eventual time point conditioned on the history:

$$\mathbb{P}\left\{\text{eventual time point in } [t, t+dt) \mid \mathcal{H}_t\right\} = \lambda\left(t\right) \cdot dt \tag{2.10}$$

With this Poisson Process, we propose our discrete co-ODE with Compound Poisson Process:

$$\begin{aligned}
&\lambda(t) = \mathrm{FC}_\lambda\Big(\mathbf{Z}(t)\Big) \\
&\mathbf{Z}(t) = \mathbf{Z}(0) + \int_0^t \frac{d\mathbf{Z}(s)}{ds} d\mathbf{N}(s), \\
&\frac{d\mathbf{Z}(t)}{dt} = \tanh\bigg(\mathrm{TCN}_2\Big(\mathbf{Z}(t)\Big)\bigg) \odot \sigma\bigg(\mathrm{TCN}_{\mathrm{gate}_3}\Big(\mathbf{Z}(t)\Big)\bigg) \\
&\qquad\qquad \odot \sigma\bigg(\mathrm{TCN}_{\mathrm{gate}_4}\Big(\mathbf{H}(0)\Big)\bigg),
\end{aligned} \tag{2.11}$$

where FC_λ denotes a fully connected feed-forward layer that learns $\lambda(t)$ from the contexts $\mathbf{Z}(t)$ to control the conditional intensity of the next eventual time point for different discrete variates, and $\mathrm{TCN}_{\mathrm{gate}_3}$ and $\mathrm{TCN}_{\mathrm{gate}_4}$ are gate convolutions used for the continuous variates and discrete variates in the discrete co-ODE, respectively, to control the amount of information flows among different variates for the discrete dynamics branch. Then, with ODESolver$(\cdot)$ [34], we can obtain the final hidden state feature $\mathbf{Z}(T)$ which learns co-dependent discrete dynamics:

$$\mathbf{Z}_T = \mathrm{ODESolver}\Big(\mathbf{Z}(0), \frac{d\mathbf{Z}(t)}{dt}, T; \boldsymbol{\Theta}_d\Big) \tag{2.12}$$

3.4 Multivariate probabilistic modeling

After we obtain the features $\mathbf{H}(T)$ and $\mathbf{Z}(T)$ with different dynamics, we use decoders to output the joint probabilities of all observations at the timestamp T.

Specifically, for the continuous variates, the decoder uses fully connected feed-forward layers and outputs the estimated multivariate Gaussian mixture distribution $N(\hat{\mu}_T, \hat{\Sigma}_T)$:

$$\hat{\mu}_T, \hat{\sigma}_T = \mathrm{Decoder}_c\Big(\mathbf{H}(T)\Big), \tag{2.13}$$

where $\hat{\mu}_T = (\hat{\mu}_T^1, \hat{\mu}_T^2, \cdots, \hat{\mu}_T^{N_C}) \in \mathbb{R}^{N_C}$ denotes the estimated mean values and $\hat{\sigma}_T = (\hat{\sigma}_T^1, \hat{\sigma}_T^2, \cdots, \hat{\sigma}_T^{N_C}) \in \mathbb{R}^{N_C}$ denotes the estimated variance values as the diagonal elements of the covariance matrix $\hat{\Sigma}_T \in \mathbb{R}^{N_C \times N_C}$ for the N_C continuous variates, respectively. Then, we calculate the empirical covariance between the j_1-th and j_2-th continuous variates to estimate $\hat{\Sigma}_T(j_1, j_2)$, which can help to capture the more accurate marginal probability distribution, as follows:

$$\hat{\Sigma}_T(j_1, j_2) = \hat{\Sigma}_T(j_2, j_1) = \frac{1}{T-1}\sum_{i=1}^{T}(\mathbf{c}_i^{j_1} - \overline{\mathbf{c}}^{j_1})(\mathbf{c}_i^{j_2} - \overline{\mathbf{c}}^{j_2}), \tag{2.14}$$

where $\overline{\mathbf{c}}^{j_1}$ and $\overline{\mathbf{c}}^{j_1}$ are the mean values of the j_1-th and j_2-th continuous variates, respectively. Thus, our loss function based on the Maximum Likelihood Estimation in the probabilistic space for the continuous variates is as follows:

$$\mathcal{L}_c = -ln\Big[\frac{1}{\sqrt{2\pi\Big|\hat{\Sigma}_T\Big|}}exp\Big(-\frac{1}{2}(\mathbf{c}_T - \hat{\mu}_T)^\top \hat{\Sigma}_T^{-1}(\mathbf{c}_T - \hat{\mu}_T)\Big)\Big] \tag{2.15}$$

For the discrete variates, the decoder uses fully connected feed-forward layers and outputs the softmax distributions $\hat{\mathbf{d}}_T$:

$$\begin{aligned} &\hat{logits}_T^1, \hat{logits}_T^2, \cdots, \hat{logit}_T^{N_D} = \text{Decoder}_d\Big(\mathbf{Z}(T)\Big) \\ &\hat{\mathbf{d}}_T^j = \text{softmax}(\hat{logit}_T^j), \text{ for } j \in \{1, 2, \cdots, N_D\}, \end{aligned} \tag{2.16}$$

where $\hat{\mathbf{d}}_T^j$ denotes the probability distribution for the j-th discrete variate in the one-hot encoding manner. Thus, our loss function based on the Maximum Likelihood Estimation in the probabilistic space for the discrete variates is as follows:

$$\mathcal{L}_d = \text{CE}(\hat{\mathbf{d}}_T, \mathbf{d}_T) \tag{2.17}$$

Last, our final loss functions are:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_d, \tag{2.18}$$

where $\mathcal{L}$ optimizes the model on the embedding layers, $\mathcal{L}_c$ optimizes the model on the continuous co-ODE and Decoder$_c$, and $\mathcal{L}_d$ optimizes the model on the discrete co-ODE and Decoder$_d$.

During the inference stage, the lower joint probabilities of the observations of all variates at one timestamp T indicate a more likely anomaly.

Table 2.1: Details of benchmark datasets.

Dataset	Domain	N_C	N_D	#Training (Unlabeled)	#Validation (Unlabeled)	#Test (Labeled)	Anomaly Rate(%)
SMD	server machine	33	5	566,724	141,681	708,420	4.16
MSL	NASA space sensor	1	54	46,653	11,663	73,729	10.72
SMAP	NASA space sensor	1	24	108,146	27,036	427,617	13.13
SWaT	water treatment	25	26	396,000	99,000	449,919	12.14
SWAN	space solar weather	33	5	48,000	12,000	60,000	32.6
CICIDS	web server	58	14	68,092	17,023	85,116	1.28
PSM	application server	25	0	105,885	26,398	87,841	27.8
GECCO	water quality	9	0	55,408	13,852	69,261	1.1
Creditcard	finance	29	0	159,491	39,873	85,443	0.172

4 Experiments

4.1 Experimental settings

Datasets.

We validate the performance with real-world datasets from different domains: SMD [70], PSM [1], MSL [31], SMAP [31], SWAN [50], SWaT [83], GECCO [50] and Credit [94].

Baselines.

We compare TAD-UP with 21 baselines of different categories for comprehensive evaluations:

- Clustering-based: PCA [56] and OCSVM [63].
- Density estimation-based: HBOS [25] DAGMM [96], IForest [27], LODA [61], and LOF [6].
- Contrastive-based: DCdetector [85], AnomalyTransformer (A.T.) [83], and PAD [33].
- Autoregression and reconstruction-based: AE [21], LSTM [31], OmniAnomaly (Omni) [70], BeatGAN [18], CAE-Ensemble [8], D3R [76], GPT-4TS [95], ModernTCN [17], MEMTO [69], SensitiveHUE [23], and DADA [67].

Table 2.2: Experiments on five real-world datasets, presented in percentages.

Dataset	SMD			MSL			SMAP			SWaT			PSM		
Metric	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**
OCSVM	66.98	82.03	73.75	50.26	99.86	66.87	41.05	69.37	51.58	56.80	98.72	72.11	57.51	58.11	57.81
PCA	64.92	86.06	74.01	52.69	98.33	68.61	50.62	98.48	66.87	62.32	82.96	71.18	77.44	63.68	69.89
HBOS	60.34	64.11	62.17	59.25	83.32	69.25	41.54	66.17	51.04	54.49	91.35	68.26	78.45	29.82	43.21
LOF	57.69	99.10	72.92	49.89	72.18	59.00	47.92	82.86	60.72	53.20	96.73	68.65	53.90	99.91	70.02
IForest	71.94	94.27	81.61	53.87	94.58	68.65	41.12	68.91	51.51	53.03	99.95	69.30	69.66	88.79	78.07
LODA	66.09	84.37	74.12	57.79	95.65	72.05	51.51	100.00	68.00	56.30	70.34	62.54	62.22	87.38	72.69
DAGMM	63.57	70.83	67.00	54.07	92.11	68.14	50.75	96.38	66.49	59.42	92.36	72.32	68.22	70.50	69.34
A.T.	54.08	97.07	66.42	51.04	95.36	66.49	56.91	96.69	71.65	53.63	98.27	69.39	54.26	82.18	65.37
DCdetector	50.93	95.57	66.45	55.94	95.53	70.56	53.12	98.37	68.99	53.25	98.12	69.03	54.72	86.36	66.99
PAD	59.54	67.66	63.71	56.33	82.21	68.15	41.67	64.52	53.94	54.73	92.35	68.06	68.45	57.72	59.21
AE	69.22	98.48	81.30	55.75	96.66	70.72	39.42	70.31	50.52	54.92	98.20	70.45	60.67	98.24	75.01
LSTM	60.12	84.77	70.35	58.82	14.68	23.49	55.25	27.70	36.90	49.99	82.11	62.15	57.06	95.92	71.55
BeatGAN	74.11	81.64	77.69	55.74	98.94	71.30	54.04	98.30	69.74	61.89	83.46	71.08	58.81	99.08	73.81
Omni	79.09	75.77	77.40	51.23	99.40	67.61	52.74	98.51	68.70	62.76	82.82	71.41	69.20	80.79	74.55
CAE-Ensemble	73.05	83.61	77.97	54.99	93.93	69.37	62.32	64.72	63.50	62.10	82.90	71.01	73.17	73.66	73.42
D3R	64.87	97.93	78.02	66.85	90.83	77.02	61.76	92.55	74.09	60.14	97.57	74.39	73.32	88.71	80.29
GPT4TS	73.33	95.97	83.13	64.86	95.43	77.23	63.52	90.56	74.67	56.84	91.46	70.11	73.61	91.13	81.44
ModernTCN	74.07	94.79	83.16	65.94	93.00	77.17	69.50	65.45	67.41	59.14	89.22	71.13	73.47	86.83	79.59
MEMTO	49.69	98.05	65.96	52.73	97.34	68.40	50.12	99.10	66.57	56.47	98.02	71.66	52.69	83.94	64.74
SensitiveHUE	60.34	90.13	72.29	55.92	98.95	71.46	53.63	98.37	69.42	58.91	91.71	71.74	56.15	98.75	71.59
DADA	76.50	94.54	84.57	68.70	91.51	78.48	65.85	88.25	75.42	61.59	94.59	74.60	74.31	92.11	82.26
TAD-UP	77.11	94.55	**84.95**	68.54	92.67	**78.80**	66.37	89.12	**76.08**	62.48	94.86	**75.35**	73.75	93.13	**82.32**

Table 2.3: Experiments on four real-world datasets with more metrics, presented in percentages.

Dataset	CICIDS		Creditcard		GECCO		SWAN	
Metric	**Aff-F1**	**AUC-ROC**	**Aff-F1**	**AUC-ROC**	**Aff-F1**	**AUC-ROC**	**Aff-F1**	**AUC-ROC**
A.T.	34.71	49.00	65.14	52.55	64.27	51.60	33.67	44.74
DCdetector	40.02	53.95	58.28	42.36	66.18	45.38	14.42	43.48
ModernTCN	51.74	65.33	73.80	95.55	90.18	95.20	46.45	52.63
GPT4TS	54.00	67.91	72.88	95.58	88.11	90.21	47.27	51.93
DADA	73.49	69.33	**75.12**	**95.73**	90.20	93.44	71.93	53.29
TAD-UP	**75.88**	**72.01**	75.07	95.71	**90.21**	**95.37**	**71.95**	**57.98**

Evaluation Metrics.

Previous methods [33, 66, 70, 83] use point adjustments (PA) with test labels to adjust their outputs. However, recent works [24, 30, 59, 67, 71, 76, 94] have demonstrated that PA can lead to faulty performance evaluations, and it is known that PA can result in state-of-the-art performance even with random guessing [30, 76]. Thus, we use the Affiliation-based Precision (Aff-P), Recall (Aff-R), F1-score (Aff-F1) [30] following recent works [67, 76, 85], where larger values indicate higher model performance. We also use the ROC-AUC [59] metric following DADA [67], which enables evaluation without choosing a threshold.

4.2 Overall comparison and analysis

Table 2.2 and Table 2.3 show the overall anomaly prediction performance of different models on all datasets. To enable direct comparison with baselines [67, 83, 85], we report the F1-scores and AUC-ROC following their original papers. We randomly repeat our method 3 times and report the average result, and the best F1-score and AUC-ROC are highlighted in bold and outperform the underlined second-best ones. As DADA is pre-trained with very large multi-domain time series data, we also underline the second-best ones other than DADA.

Key observations are followed. First, TAD-UP can achieve state-of-the-art results on all datasets. This demonstrates that TAD-UP is able to learn both continuous and discrete dynamics to improve time series anomaly detection.

Second, we observe that the contrastive-based methods are unsatisfactory on most datasets. This is because their data augmentation, borrowed from CV or NLP, cannot fit well for time series data.

Third, TAD-UP achieves better accuracy compared to other reconstruction based methods, as they only add up all errors from different variates regardless of their measurement units. They cannot learn the importance units of different variates to obtain the final anomaly scores.

Last, TAD-UP outperforms DADA, only except for the Creditcard dataset, which may require a lot of finance knowledge that DADA can learn from pre-training on multi-domain data.

Ablation Studies

We conduct ablation studies to validate the effectiveness of our contributed components. Specifically, we compare TAD-UP with the following variants: (1) w/o continuous co-ODE: This variant directly learns features from the continuous variables without co-ODE. (2) w/o discrete co-ODE: This variant directly learns features from the discrete variables without co-ODE. (3) w/o probabilistic model: This variant directly uses the reconstruction loss.

Table 2.4 shows the results of different variants. From Table 2.4 we observe that: (1) Our continuous co-ODE and discrete co-ODE can learn the interactive dynamics effectively and contribute to more accurate re-

Table 2.4: Ablation study.

Dataset	SMD			MSL			SWaT			CICIDS		
Metric	Aff-P	Aff-R	Aff-F1	Aff-P	Aff-R	Aff-F1	Aff-P	Aff-R	Aff-F1	Aff-P	Aff-R	Aff-F1
w/o continuous co-ODE	74.96	93.20	83.09	67.98	91.34	77.95	61.94	94.67	74.88	64.55	90.26	75.27
w/o discrete co-ODE	75.63	92.47	83.21	67.10	91.49	77.41	61.44	93.75	74.23	61.86	90.13	73.37
w/o probabilistic mode	74.85	92.93	82.92	60.01	94.66	73.45	62.37	94.91	75.27	64.37	90.52	75.24
TAD-UP	77.11	94.55	**84.95**	68.54	92.67	**78.80**	62.48	94.86	**75.35**	64.89	91.33	**75.88**

Table 2.5: Anomaly detection for irregular time series

Dataset	SWaT											
Irregular ratio	0%				20%				40%			
Metric	Aff-P	Aff-R	Aff-F1	AUC-ROC	Aff-P	Aff-R	Aff-F1	AUC-ROC	Aff-P	Aff-	Aff-F1	AUC-ROC
PAD	54.73	92.35	68.06	63.86	52.71	92.13	67.05	63.91	49.73	90.64	64.22	56.84
TAD-UP	62.48	94.86	**75.35**	**83.01**	61.82	93.06	**74.28**	**80.92**	59.98	91.97	**72.60**	**72.90**

sults. (1) Our joint probabilistic modeling helps obtain the anomaly scores that consider the importance of different variables with Maximum Likelihood Estimation in the unified probabilistic space and thus can have more accurate results.

Irregular time series

In real-world applications, irregular timestamps occur when data points are collected or recorded at uneven intervals. Many real-world processes generate data irregularly. For example, sensor readings may be triggered by events rather than at fixed intervals, and manual collection may be sporadic as labor costs are very expensive in medical and biological fields. PAD [33] uses interpolation methods and NCDE to support anomaly detection on time series of irregular timestamps. We also show the experimental results on the irregular time series following PAD. From Table 2.5, we can see that TAD-UP still performs well for irregular time series with our co-ODEs to learn dynamics along continuous time.

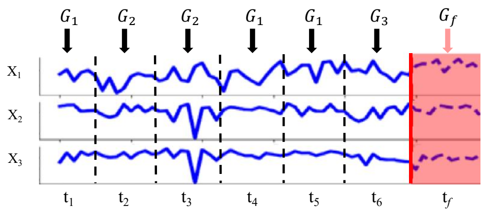
Chapter 3

Unsupervised Time Series Anomaly Prediction with Importance-based Generative Contrastive Learning

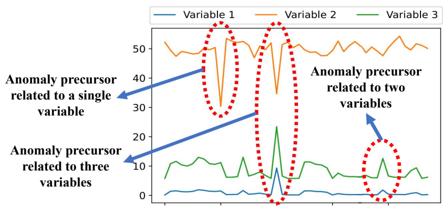
This chapter provides an overall summary of Paper C [93], which explores unsupervised time series anomaly prediction using Maximum Mean Discrepancy knowledge. This chapter reuses content from Paper C [93] when that is considered effective and clear, and it offers no new contributions beyond those reported in Paper C [93]. More details and experiments can be found in Paper C [93].

1 Motivation and Problem Statement

Much of the data collected from real-world applications can be represented as time-dependent observations, which form multi-variate time series [9, 56, 78, 87]. Based on the current time series data, anomaly prediction makes real-time decisions about whether anomaly signals are emerging, indicating that severe anomalies will occur in the future [5, 10, 29, 33], as illustrated in Figure 3.1(a). Anomaly prediction is crucial in scenarios where safety is a concern and minimizing the losses caused by anomalies



(a) Supervised anomaly prediction learned with labeled anomaly precursors.



(b) Different anomaly precursor combinations in multi-variate time series.

Fig. 3.1: Our motivation.

is a priority [2, 3, 58, 82], such as in healthcare [10] or drinking-water systems [33]. For instance, anomaly prediction can help identify potential pollution events or faults in advance [5, 29] or provide early warnings of extreme environmental conditions [58].

Only a few classification-based studies [2, 3, 58, 82] are related to time series anomaly prediction. Existing methods predict future anomalies by identifying whether current observations contain anomaly signals that mark the beginning of deviations from normal behaviors [5, 10, 29]. These early signals are referred to as anomaly *precursors* [33]. However, to achieve accurate results, existing methods heavily rely on supervised learning, where anomaly precursors are provided during training.

In many real-world applications, there is a lack of labeled training data for supervised learning because manual labeling is costly [64]. Moreover, unexpected anomalies [28, 77] may arise during real-time deployment that were not considered during training, making supervised methods fail in practice [24, 94]. Therefore, unsupervised time series anomaly prediction has gained significant interest. However, there are two main challenges.

Challenge 1: In the context of unsupervised anomaly prediction, there is no labeled anomaly precursor data to allow models to learn temporal dependency features from pairs of successive normal sub-sequences and anomaly precursors, where the data starts deviating from normal [10]. This is essential for enabling anomaly prediction [5, 58].

Challenge 2: We face the issue of potentially very complex anomaly precursor combinations in multi-variate time series, leading to high time and space complexity. As illustrated in Figure 3.1(b), for any n-variate time series, there can be as many as $O(2^n)$ potential anomaly precursor combinations [29], where different anomaly precursors may appear individually in each variate or in combinations of variates [5].

To address these challenges, we propose a novel anomaly prediction method called **IGCL**, which stands for **I**mportance-based **G**enerative **C**ontrastive **L**earning for unsupervised time series anomaly prediction.

To address **Challenge 1**, we employ a generative contrastive learning architecture. We introduce an anomaly precursor pattern generation module that utilizes a diffusion-based transformation and variance regularization to generate various potential precursor patterns labeled as negative data using Gaussian noise. Additionally, we propose an overlapping window-based temporal convolutional network (TCN) architecture to efficiently extract temporal dependencies and learn both positive and negative contextual representations. Specifically, the model distinguishes normal sub-sequences from anomaly precursors, where pairs of successive normal sub-sequences serve as positive samples, and pairs of a normal sub-sequence and an anomaly precursor, which exhibit temporal changes from normal to abnormal, serve as negative samples.

To address **Challenge 2**, we introduce a memory bank with importance-based scores to adaptively store representative anomaly precursors. This helps generate more complex anomaly precursor combinations related to different variates.

- We introduce a generative contrastive learning architecture that incorporates a diffusion-based transformation and variance regularization to generate diverse precursor patterns and capture temporal dependencies, enabling the distinction between normal and anomaly precursors.

- We present a memory bank with importance scores to store representative negative samples, facilitating the generation of more complex negative samples.

- Experimental results on benchmark datasets from various domains demonstrate that our method outperforms state-of-the-art baselines.

2 Preliminaries

2.1 Definitions

We formalize the unsupervised time series anomaly prediction problem.
A Multi-variate Time Series is represented as $\mathbf{X} = \langle \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T \rangle \in \mathbb{R}^{N \times T}$, where N is the number of variates collected at each timestamp, T is the total number of historical timestamps, $\mathbf{x}_t \in \mathbb{R}^N$ denotes the N-dimensional observations at timestamp t. We use $\mathbf{X}_{t+1:t+f} \in \mathbb{R}^{N \times f}$ to indicate the data during the time window $[t+1, t+f]$, use $\mathbf{X}^i_{t+1:t+f} \in \mathbb{R}^f$ to indicate the data of the i-th variate, and use $\mathbf{X}^i_t \in \mathbb{R}^1$ to indicate the data of the i-th variate at timestamp t. We also use T to denote the current timestamp.
A Sub-sequence $\mathbf{X}_{t_1:t_2}$ is a continuous subset of $\mathbf{X}$ from t_1 to t_2. The set of b most recent sub-sequences before the current timestamp T, each with sequence length of $h+1$, is denoted as $\mathbb{X}^b_{T,h} = \{\mathbf{X}_{T-h:T}, \mathbf{X}_{T-1-h:T-1}, \cdots, \mathbf{X}_{T-b+1-h:T-b+1}\}$.
An Anomaly Precursor is a sub-sequence $\mathbf{X}_{T-h:T}$ containing early signals that just start deviating from normal and followed by future anomalies in $\mathbf{X}_{T+1:T+f}$ [5, 29, 33], where h and f are hyper-parameters for the look-back and look-forward windows.
Unsupervised time series anomaly prediction. At any current timestamp T, given the historical observations $\mathbf{X}$, where we use a set of b most recent sub-sequences $\mathbb{X}^b_{T,h} = \{\mathbf{X}_{T-h:T}, \mathbf{X}_{T-1-h:T-1}, \cdots, \mathbf{X}_{T-b+1-h:T-b+1}\}$, the goal is to determine a real-time anomaly score $\hat{p}_T$ to indicate whether there are anomaly precursors currently and more anomalies in $\mathbf{X}_{T+1:T+f}$. The transformation of anomaly scores into binary labels is done by applying a threshold δ [33], *i.e.,* if $\hat{p}_T < \delta$, the future sub-sequence $\mathbf{X}_{T+1:T+f}$ is normal; and if $\hat{p}_T \geq \delta$, the future sub-sequence $\mathbf{X}_{t+1:t+f}$ is abnormal. Thus, we formulate the time series anomaly prediction problem as that of finding a model with a mapping function $\mathcal{F}$ and $\hat{p}_T = \mathcal{F}_\theta(\mathbb{X}^b_{T,h})$, where θ is the parameters of the model $\mathcal{F}$. For the unsupervised time series anomaly prediction problem, we do not have the true score p_T as there is no labeled data.

2.2 Theoretical analysis

We analyze unsupervised time series anomaly prediction using Maximum Mean Discrepancy (MMD) theory [32]. To model sequence-based probabilities, we leverage Markov Chains [48], which are commonly used in time series analytics [20]. We introduce random variates $\mathbf{H_{t-h-1}}$ and $\mathbf{H_t}$ to represent the observations that the sub-sequences $\mathbf{X}_{t-2h-1:t-h-1}$ and $\mathbf{X}_{t-h:t}$ might exhibit during two successive time-windows $[t-2h-1, t-h-1]$ and $[t-h, t]$, respectively. Then, we define

$$\mathcal{N} = P(\cdot|\mathbf{H_{t-h-1}} = \mathbf{X}_{t-2h-1:t-h-1}, \mathbf{H_t} \text{ is normal})$$

to denote the Markov conditional distribution of the normal sub-sequence observed during time-window $[t-h, t]$, given that its previous observations during time-window $[t-2h-1, t-h-1]$ are $\mathbf{X}_{t-2h-1:t-h-1}$. Similarly, we define

$$\mathcal{A} = P(\cdot|\mathbf{H_{t-h-1}} = \mathbf{X}_{t-2h-1:t-h-1}, \mathbf{H_t} \text{ is anomaly precursor})$$

to denote the Markov conditional distribution of the anomaly precursor observed during time-window $[t-h, t]$. We present the theorem here and its proof can be seen in our Paper C [93].

Theorem: In the unsupervised setting, the future anomaly is predictable if and only if we can identify its anomaly precursor, which follows a distribution different from the normal previous sub-sequences. $\square$

Thus, our model aims to distinguish normal distribution $\mathcal{N} = P(\cdot|\mathbf{H_{t-h-1}}$, $\mathbf{H_t}$ is normal) from any potential different distributions $\mathcal{A} = P(\cdot|\mathbf{H_{t-h-1}}$, $\mathbf{H_t}$ is anomaly precursor).

3 Methodology

We present the overall architecture in Figure 3.2, which includes the anomaly precursor pattern generation, positive and negative representations learning, contrastive loss, and the memory bank.

For any timestamp t, the model takes as input the set of b available historical sub-sequences $\mathbb{X}_{t,h}^b = \{\mathbf{X}_{t-h:t}, \mathbf{X}_{t-1-h:t-1}, \cdots\cdots, \mathbf{X}_{t-b+1-h:t-b+1}\}$. We first apply Instance Normalization [74] on the observation of each time series variate, *i.e.,* $\mathbf{X}_{t'}^i$ in the set $\mathbb{X}_{t,h}^b$, and output $\overline{\mathbb{X}}_{t,h}^b$, to learn stable components [44] from time series.

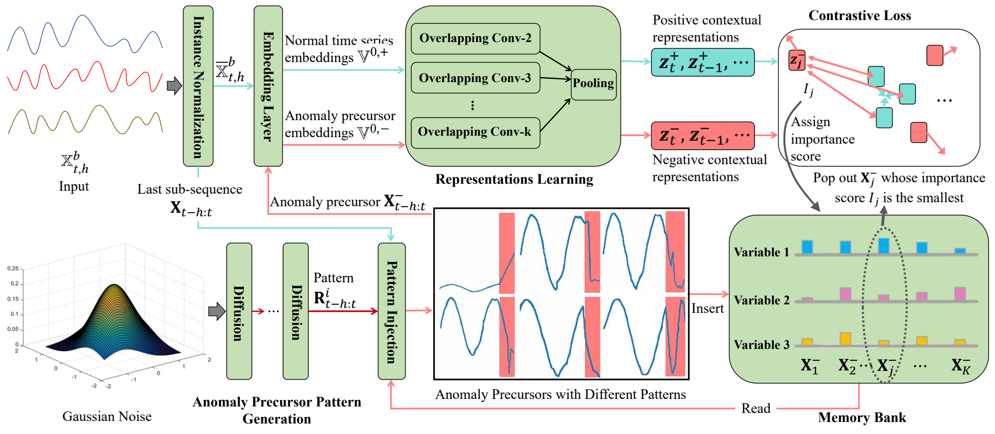

Input
Instance Normalization
Embedding Layer
Normal time series embeddings $\mathbb{V}^{0,+}$
Anomaly precursor embeddings $\mathbb{V}^{0,-}$
Overlapping Conv-2
Overlapping Conv-3
Overlapping Conv-k
Pooling
Representations Learning
Positive contextual representations
Negative contextual representations
Contrastive Loss
Assign importance score
Pop out $\mathbf{X}_j^-$ whose importance score I_j is the smallest
Last sub-sequence $\mathbf{X}_{t-h:t}$
Anomaly precursor $\mathbf{X}_{t-h:t}^-$
Gaussian Noise
Diffusion
Pattern $\mathbf{R}_{t-h:t}^i$
Pattern Injection
Anomaly Precursor Pattern Generation
Anomaly Precursors with Different Patterns
Insert
Read
Variable 1
Variable 2
Variable 3
Memory Bank

Fig. 3.2: The overall architecture.

The embedding layer takes the normalized input $\overline{\mathbb{X}}_{t,h}^b$ and maps the observations at each timestamp t', i.e., $\mathbf{X}_{t'} \in \mathbb{R}^N$, to d-dimensional vectors $\mathbf{v}_{t'} \in \mathbb{R}^d$. This process extracts dense features from the normal time series and outputs a set of feature vectors, i.e., $\mathbb{V}^{0,+}$.

3.1 Anomaly precursor pattern generation

As shown in experimental results, diffusion models have demonstrated superior performance compared to other generative models for time series [76, 81]. Building upon this, we introduce a diffusion-based transformation method that generates various potential anomaly precursor patterns starting from a simple Gaussian distribution, enhanced by our variance regularization, to simulate precursor patterns that begin deviating from normal behavior.

The denoising diffusion model is composed of two main processes: the pollution process with S steps and its reverse denoising diffusion process, where S represents the maximum number of diffusion steps. In this context, we use $x^0 = \mathbf{R}_{t-h:t}^i \in \mathbb{R}^h$ to represent any potential anomaly precursor pattern, which could be more complex than Gaussian patterns. The pollution process is detailed in our Paper C [93]. The anomaly precursor pattern generation module is based on the denoising diffusion process $p_\theta(x^{s-1}|x^s)$, which reverses random Gaussian noise x^S to produce more intricate potential anomaly precursor patterns.

$x^{s-1} \sim p_\theta(x^{s-1}|x^s) = \frac{1}{\sqrt{1-\beta^s}}\left(x^s - \sqrt{\beta^s}\varepsilon_\theta(x^s)\right)$, where $\varepsilon_\theta(x^s) \sim N(\mu_s, \sigma_s^2)$, and μ_s is the mean value parameter and σ_s is the variance parameter to be learned. We use Multi-layer Perceptron (MLP) layers to model with μ^s and σ^s:

$$\mu^s, \sigma^s = MLP\Big(concat(x^s, s, \mathbf{X}^i_{t-h:t})\Big),\ \forall s \in \{S, S-1, \cdots, 1\},$$

where x^S is a random Gaussian noise, x^s is the denoised state after each step, and i denotes this process aiming to generate anomaly precursor patterns for the i-th variate. We have $\sigma^s\varepsilon + \mu^s \sim N(\mu_s, \sigma_s^2)$, where $\varepsilon \sim N(0, I)$, and $\sigma^s\varepsilon+\mu^s$ is derivable with respective to μ^s and σ^s which are the outputs from the MLP layers. Thus, the final denoising diffusion process is

$$x^{s-1} = \frac{1}{\sqrt{1-\beta^s}}\left(x^s - \sqrt{\beta^s}(\sigma^s\varepsilon + \mu^s)\right),$$

where we randomly sample the Gaussian noise x^S, and apply the denoising diffusion process step by step to get the anomaly precursor pattern $\mathbf{R}^i_{t-h:t} = x^0$.

We also introduce a regularization loss on the parameters $N(\mu_s, \sigma_s^2)$, where the variance σ_s^2 governs the diversity of the generated anomaly precursor patterns, and both σ_s^2 and the mean μ_s control the extent of deviation from normal behavior. Firstly, the variance σ_s^2 should not be excessively large, as this could lead to the generation of severe anomalies instead of anomaly precursors that begin deviating from normal. Secondly, σ_s^2 should not be too small, as a smaller variance would produce less diverse precursor patterns [81]. Therefore, we suggest using Kullback-Leibler divergence as a regularization technique to ensure that $N(\mu_s, \sigma_s^2)$ remains close to $N(\mu_s, I)$, where σ_s^2 approximates the unit normal variance I, which is neither too large nor too small [46]. For any distribution $N(\mu, \sigma^2)$, the Kullback-Leibler divergence is computed as $KL\Big(N(\mu, \sigma^2)\Big\|N(\mu, I)\Big) = \frac{1}{2}\Big(-\log\sigma^2 + \sigma^2 - 1\Big)$. The detailed proof is provided in our Paper C [93].

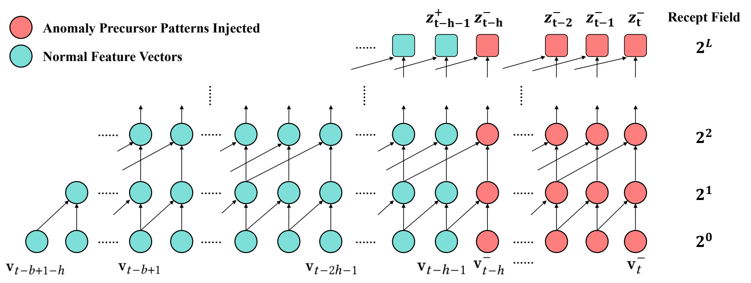


Fig. 3.3: The overlapping window-based temporal convolutional network with a kernel size k of 2.

Hence, the analytical solution for the regularization loss is as follows:

$$\mathcal{L}_r = \sum_{s=1}^{S} \frac{1}{2}\Big(- \log \sigma_s^2 + \sigma_s^2 - 1\Big) \tag{3.1}$$

3.2 Positive and negative representations

We introduce a temporal convolutional network (TCN) based on overlapping windows to learn contextual representations from all possible pairs of consecutive sub-sequences, as defined by the Markov Chain, enabling the extraction of both normal and abnormal temporal dependencies simultaneously.

Specifically, our embedding layer first transforms the normalized time series data, denoted as $\mathbf{X}_{t'} \in \mathbb{R}^N$ for $t' \in [t-b+1-h, t]$, which corresponds to the set $\mathbb{X}^b_{t,h}$, into d-dimensional feature vectors $\mathbf{v}_{t'} \in \mathbb{R}^d$. These vectors are designed to capture dense features from the time series at each timestamp. Additionally, the input to the embedding layer can be combined with supplementary attributes, such as timestamp encodings. These auxiliary attributes at timestamp t' are represented by $\mathbf{a}_{t'} \in \mathbb{R}^F$, where F is the total number of dimensions of these auxiliary features. Therefore, the embedding layer concatenates the original time series data $\mathbf{X}_{t'}$ and the auxiliary attributes

$$\mathbf{f}_{t'} = concat(\mathbf{X}_{t'}, \mathbf{a}_{t'}) \ \forall t' \in [t-b+1-h, t], \tag{3.2}$$

as its input, and maps it to the feature vector by:

$$\mathbf{v}_{t'} = \sigma(W_e \, \mathbf{f}_{t'}), \ \forall t' \in [t-b+1-h, t], \tag{3.3}$$

where $\mathbf{v}_{t'} \in \mathbb{R}^d$ is the learned feature vector. Following the same way, we can get the feature vector $\mathbf{v}^-_{t'} \in \mathbb{R}^d$, where $t' \in [t-h, t]$, from the generated anomaly precursor $\mathbf{X}^-_{t-h:t}$.

Next, our overlapping window-based TCN utilizes the feature vectors $\mathbf{v}_{t'}$ and $\mathbf{v}^-_{t'}$ to simultaneously capture temporal dependencies across all possible pairs of consecutive sub-sequences. The temporal convolution operation progresses across timestamps by applying a dilation factor that skips over certain feature vectors at different layers, as depicted in Figure 3.3. Specifically, this module processes all the feature vectors, i.e., $\mathbf{v}_{t'}$ for $t' \in [t-b+1-h, t]$ and $\mathbf{v}^-_{t'}$ for $t' \in [t-h, t]$, as inputs:

$$\begin{aligned} \mathbb{V}^{0,+} &= \Big\langle \mathbf{v}_{t-b+1-h}, \cdots, \mathbf{v}_{t-h}, \cdots, \mathbf{v}_t \Big\rangle \\ \mathbb{V}^{0,-} &= \Big\langle \mathbf{v}_{t-b+1-h}, \cdots, \mathbf{v}^-_{t-h}, \cdots, \mathbf{v}^-_t \Big\rangle, \end{aligned} \tag{3.4}$$

and uses the learnable convolution *filter* $\in \mathbb{R}^k$ with the kernel size of k to extract features and learn the temporal dependencies by:

$$\begin{aligned} \mathbb{V}^{l,+}(t') &= \sum_{i=0}^{k-1} \mathit{filter}(i)\mathbb{V}^{l-1,+}(t' - k^{l-1} \times i), \text{ for } 1 \le l \le L, \\ \mathbb{V}^{l,-}(t') &= \sum_{i=0}^{k-1} \mathit{filter}(i)\mathbb{V}^{l-1,-}(t' - k^{l-1} \times i), \text{ for } 1 \le l \le L, \end{aligned} \tag{3.5}$$

where k^{l-1} is the dilation factor in the layer $l-1$, and $L \sim O(log(h))$ is the max number of TCN layers whose recept field is large than the whole window size of the pair of successive sub-sequences. We employ various kernel sizes to extract features concurrently and combine them, allowing us to capture temporal dependencies at different scales. Finally, it outputs all contextual representations for all pairs of successive sub-sequences at once, *e.g.*, $z^+_t = \mathbb{V}^{L,+}(t)$, $z^+_{t-1} = \mathbb{V}^{L,+}(t-1)$, $z^-_t = \mathbb{V}^{L,-}(t)$ and $z^-_{t-1} = \mathbb{V}^{L,-}(t-1)$. In this way, our total complexity is reduced to $O\Big((b+h)log(h)\Big)$, rather than the complexity of all pairs of sub-sequences as $O\Big((b+h)^2\Big)$.

3.3 The objective function

We introduce a contrastive approach to differentiate between $\mathcal{N}$ and $\mathcal{A}$ by pushing z^-_t away from z^+_t. This method effectively groups positive representations, such as z_t^+ belonging to $\mathcal{N}$, and separates negative representations, such as z_t^- from $\mathcal{A}$. Each anchor z_t^+ represents the contextual embedding from the pair of successive normal sub-sequences prior to timestamp t, corresponding to $\mathcal{N} = P(\cdot|\mathbf{H_{t-h}}, \mathbf{H_t}$ is normal). Consequently, we aim to make the anchor z^+t similar to positive samples z_{t-j}^+, where $1 \leq j \leq P$, derived from the closest successive normal sub-sequences before timestamp t and following the normal data distribution. To maintain consistency with the inference phase and prevent data leakage, the positive samples for each anchor z_t^+ are always taken from earlier timestamps. At the same time, the anchor z_t^+ should be distinguishable from the negative representation z_t^-, which encodes the temporal dependencies from normal sub-sequences to generated anomaly precursors, representing the distribution of anomaly precursors $\mathcal{A}$. Hence, our contrastive loss function is:

$$\mathcal{L}_c = \sum_{i=t}^{t-h} -log \frac{\sum_{j=1}^{P} exp\Big(Sim(z_i^+, z_{i-j}^+)/\tau\Big)}{exp\Big(Sim(z_i^+, z_i^-)/\tau\Big) + \sum_{j=1}^{P} exp\Big(Sim(z_i^+, z_{i-j}^+)/\tau\Big)}, \tag{3.6}$$

where τ controls the strength for softmax, P is the number of positive samples used, and Sim is the similarity measurement to distinguish whether the contextual representations belong to the same distributions.

Finally, together with the regularization loss, we train the model using Adam gradient descent [45]:

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_r, \tag{3.7}$$

where λ controls the strength of the regularization.

3.4 Memory bank

We propose to use a memory bank to store previously generated anomaly precursors and combine them with the current anomaly precursor pattern $\mathbf{R}_{t-h:t}^i$, to accumulate more complex anomaly precursor combinations related to more than one variate.

To be more specific, the memory bank M has a fixed size of K and empirically $K \ll O(2^n)$ in our experiments:

$$\mathbf{M} = \Big\{\mathbf{X}_1^-, \mathbf{X}_2^-, \cdots, \mathbf{X}_K^-\Big\}, \tag{3.8}$$

where each $\mathbf{X}_j^-$ denotes a stored anomaly precursor. The currently generated anomaly precursor pattern $\mathbf{R}_{t-h:t}^i$ which is only related to the i-th variate, is also injected into the anomaly precursors in the memory bank:

$$\mathbf{X}_j^- \leftarrow \mathbf{X}_j^- + \mathbf{R}_{t-h:t}^i, \tag{3.9}$$

and thus we can get anomaly precursors that involve more than one variate. Then, similar to Equation 3.4, we can get more negative contextual representations, *e.g.,* $z_{t,j}^-$ and $z_{t-1,j}^-$, from the anomaly precursor $\mathbf{X}_j^-$. The contrastive loss $\mathcal{L}_c$ is updated to

$$\sum_{i=t}^{t-h} -log \frac{\sum_{j=1}^{P} exp\Big(Sim(z_i^+, z_{i-j}^+)/\tau\Big)}{\sum_{j=0}^{K} exp\Big(Sim(z_i^+, z_{i,j}^-)/\tau\Big) + \sum_{j=1}^{P} exp\Big(Sim(z_i^+, z_{i-j}^+)/\tau\Big)}, \tag{3.10}$$

where $z_{i,0}^- = z_i^-$ is the negative contextual representations from the current anomaly precursor $\mathbf{X}_{t-h:t}^-$, and $z_{i,j}^-$ is the negative contextual representations from the j-th anomaly precursor $\mathbf{X}_j^-$ stored in the memory bank.

Finally, the current anomaly precursor $\mathbf{X}_{t-h:t}^-$ is inserted into the memory bank which will have a size of $K+1$, *i.e.,* $\mathbf{X}_0^- = \mathbf{X}_{t-h:t}^-$. Instead of using a first-in-first-out strategy to keep a fixed size of K for the memory bank, we propose to pop out the anomaly precursor based on an importance score. The intuition is that the memory bank should store the hard and important negative samples for further model training, *i.e.,* the anomaly precursors that are not yet distinguished by the model, and should pop out not important samples, *i.e.,* the anomaly precursors that are already dissociated away from normal. The importance score I_j for the anomaly precursor $\mathbf{X}_j^-$ is calculated by:

$$I_j = \sum_{i=t}^{t-h} Sim(z_i^+, z_{i,j}^-) \tag{3.11}$$

Thus, we pop out the anomaly precursor with the smallest importance score after each iteration of model optimization.

3.5 Inference

For each real-time T in the inference stage, we output the probability score $\hat{p}_T$ to predict how likely there is an anomaly precursor currently and will

Table 3.1: Overall performance, presented in percentages.

Dataset	PSM			SMAP			SWAN			SWaT			GECCO			SMD			MSL		
Metric	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
DAGMM	39.27	36.14	37.64	11.91	22.00	15.51	52.73	39.46	45.14	82.14	61.72	70.48	47.50	27.64	34.95	14.74	15.37	15.05	13.94	17.04	15.33
IForest	39.39	38.82	39.10	11.79	18.67	14.45	53.85	39.07	45.28	75.39	62.22	68.17	42.75	29.89	35.18	15.25	18.28	16.63	13.82	17.11	15.29
K-means	38.89	39.04	38.96	13.05	21.37	16.20	54.04	42.30	47.45	82.13	62.50	70.98	47.94	29.18	36.28	15.77	17.40	16.54	14.23	15.99	15.06
Deep-SVDD	36.63	36.30	36.46	11.77	17.09	13.94	52.39	39.18	44.83	81.29	61.95	70.31	46.09	28.45	35.18	14.99	15.78	15.37	12.94	16.04	14.32
THOC	37.64	38.93	38.27	12.61	21.89	15.99	54.93	40.02	46.30	82.42	62.45	71.06	46.41	29.49	36.09	16.90	16.79	16.84	14.37	16.22	15.24
DCdetector	28.97	12.05	17.02	8.04	11.30	9.40	21.04	10.94	14.40	46.45	36.36	40.79	1.76	3.46	2.33	3.71	9.03	5.26	2.35	3.46	2.80
PAD	31.23	33.05	32.11	11.01	22.37	14.76	50.13	37.61	42.98	74.83	59.09	66.04	44.38	28.17	34.46	11.23	18.76	14.05	14.37	13.22	13.77
ATransformer	30.56	34.64	32.47	11.40	22.80	15.20	52.20	38.75	44.48	75.00	60.50	66.97	43.62	27.92	34.05	10.52	19.42	13.65	14.62	12.42	13.43
LSTM-VAE	39.20	40.31	40.02	11.68	22.65	15.41	54.42	40.88	46.69	66.76	68.36	67.55	46.67	30.92	37.20	14.21	16.42	15.24	16.18	15.28	15.72
Omni	39.45	40.88	40.15	11.90	23.39	15.77	54.83	41.00	46.92	85.73	58.57	69.59	50.55	29.65	37.38	16.22	18.19	17.15	13.66	21.66	16.75
GANomaly	37.02	43.06	39.81	12.18	23.41	16.02	54.44	40.91	46.72	83.99	59.77	69.84	50.06	29.23	36.91	15.06	21.39	17.68	15.36	17.30	16.27
CAE-Ensemble	40.18	41.99	41.07	15.88	26.68	<u>19.91</u>	55.98	41.52	47.68	88.61	59.90	<u>71.48</u>	51.67	29.92	37.90	18.41	19.51	<u>18.94</u>	20.35	18.27	19.26
PUAD	39.91	42.38	41.11	15.20	27.20	19.50	54.91	41.19	47.07	85.68	58.05	69.21	50.76	29.58	37.38	16.37	21.85	18.72	20.04	17.60	18.74
D3R	40.73	42.21	<u>41.46</u>	15.73	26.89	19.85	55.32	42.64	<u>48.16</u>	84.13	61.85	71.29	51.24	30.91	<u>38.56</u>	15.78	19.33	17.38	19.99	20.32	<u>20.15</u>
IGCL	42.31	46.97	**44.52**	17.48	28.02	**21.53**	59.20	44.94	**51.09**	84.98	63.54	**72.71**	52.10	32.77	**40.23**	17.39	25.90	**20.81**	21.97	22.81	**22.38**

be more future anomalies in $\mathbf{X}_{T+1:T+f}$:

$$\hat{p}_T = \sum_{j=1}^{K} Sim(z_T^+, z_{T,j}^-) - \sum_{j=1}^{P} Sim(z_T^+, z_{T-j}^+), \tag{3.12}$$

where K is the size of the memory bank with anomaly precursors as negative samples, and P is the number of positive samples.

4 Experiments

4.1 Experimental setup

Datasets.

We evaluate the performance of our approach using real-world datasets, including SMD [70], PSM [1], MSL [31], SMAP [31], SWAN [50], SWaT [83], and GECCO [50].

Baselines.

We compare our unsupervised anomaly prediction method against the following baselines as described in PAD [33]:

- Density-based methods: DAGMM [96] and IForest [38].
- Clustering-based methods: K-means [64], Deep-SVDD [63], and THOC [66].
- Contrastive-based methods: DCdetector [85] and PAD [33].

- Autoregression and reconstruction-based methods: LSTM-VAE [42], A-Transformer [83], Omni [70], GANomaly [18], CAE-Ensemble [8], PU-AD [54], and D3R [76].

Evaluation Metrics.

We assess the performance of the anomaly prediction models using Precision (P), Recall (R), and F1-score (F1), following the evaluation methodology suggested in [8, 33, 94].

4.2 Overall comparison and analysis

Table 3.1 presents the overall anomaly prediction performance of various models across all datasets. Each method is repeated three times, and we report the average result. The highest F1-scores are highlighted in bold, while the second-best results are underlined for comparison.

Key observations are as follows.

First, IGCL consistently outperforms all baseline methods across every dataset. This indicates that IGCL effectively learns normal temporal dependencies and detects abnormal temporal dependencies that transition from normal to abnormal through the generated anomaly precursors, leading to improved anomaly prediction accuracy.

Second, contrastive-based, density-based, and clustering-based methods generally perform poorly on most datasets. This is because these methods cannot generate anomaly precursors as negative samples to learn the abnormal temporal dependencies. They rely only on non-labeled data as positive samples, which limits their accuracy, as effective anomaly prediction requires learning both normal patterns and abnormal temporal dependencies.

Third, IGCL achieves the best performance when compared to autoregression and reconstruction-based methods. These methods focus solely on reconstructing normal time series sub-sequences within each time window and cannot explicitly learn the abnormal temporal dependencies that transition from normal to anomaly precursors. As a result, their accuracy is limited. The experiments highlight the importance of learning temporal dependencies from pairs of successive time series sub-sequences that transition from normal to anomaly precursors for effective anomaly prediction.

Chapter 4

Conclusion and Future Work

1 Conclusion

The motivation for this Ph.D. thesis stems from the need to overcome these critical challenges by bridging the gap between data-driven methods and real-world and scientific knowledge. By integrating more knowledge, we push the boundaries of time series analytics and enhance the accuracy of deep learning models. The proposed research tackles the limitations and problems of existing methods on three tasks in time series analytics. Each problem and task is addressed in one research paper included in this thesis. A conclusion of each research paper and this thesis is followed.

1. Paper A [90] proposes the MTSF-DG model that incorporates Dynamic Graph modeling into Multivariate Time Series Forecasting. MTSF-DG can learn historical relation graphs and predict future relation graphs to capture the dynamic relations, guided by causality knowledge and empirical covariance matrices. We further propose a reasoning network to learn temporal influences from historical timestamps and explicitly forecast each future timestamp.

2. Paper B [91] propose the TAD-UP model that incorporates differential equation knowledge for Time series Anomaly Detection via Unified Probabilistic modeling. We propose two co-dependent branches of neural ordinary differential equations to learn both continuous and discrete dynamics for different variates. We also propose a unified

joint probability distribution modeling. The resulting model is optimized using Maximum Likelihood Estimation on joint variates. We detect anomalies using joint probabilities, which take the marginal probabilities of different variates into account as their importance on final anomaly scores.

3. Paper C [93] We propose an Importance-based Generative Contrastive Learning method (IGCL) for unsupervised anomaly prediction. IGCL employs a diffusion module to produce anomaly precursor patterns, controlled by knowledge from Maximum Mean Discrepancy theory. IGCL then can predict anomalies by identifying anomaly precursors that will evolve into future anomalies. To address challenges caused by potentially complex precursor combinations involving multiple variates, we propose a memory bank with importance scores that stores representative samples adaptively and generates more complex anomaly precursors.

2 Future Work

In future works, there is significant potential to enhance knowledge enhanced time series analytics by training on large-scale time series datasets. By leveraging vast amounts of data across various domains, we can build more robust models that are capable of capturing diverse patterns, trends, and temporal dependencies that occur across different industries and applications. This approach will allow the model to learn a wide range of time-series behaviors, improving its generalizability and performance.

Furthermore, the development of a knowledge enhanced time series analytics foundation model would be a valuable step forward. A foundation model built on a vast corpus of diverse time series data can serve as a powerful starting point for a wide array of other downstream tasks, such as forecasting, anomaly detection, classification, and pattern recognition. This model could efficiently leverage pre-trained knowledge, enabling it to adapt and perform well across new applications with minimal task-specific fine-tuning. This type of model would be highly beneficial for industries such as finance, healthcare, energy, and transportation.

References

[1] A. Abdulaal, Z. Liu, and T. Lancewicki, "Practical approach to asynchronous multivariate time series anomaly detection and localization," in *SIGKDD*, 2021, pp. 2485–2494.

[2] Y. Ang, Q. Huang, A. Tung, and Z. Huang, "A stitch in time saves nine: Enabling early anomaly detection with correlation analysis," in *ICDE*, 2023, pp. 132–145.

[3] Y. Ang, Q. Huang, A. K. Tung, and Z. Huang, "Eads: An early anomaly detection system for sensor-based multivariate time series," in *ICDE*, 2024, pp. 5433–5436.

[4] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *NeurIPS*, 2020, pp. 11 465–11 475.

[5] P. Boniol, M. Meftah, E. Remy, B. Didier, and T. Palpanas, "dcnn/dcam: anomaly precursors discovery in multivariate time series with deep convolutional neural networks," *Data-Centric Engineering*, vol. 4, 2023.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *SIGMOD*, 2000.

[7] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen, "Unsupervised time series outlier detection with diversity-driven convolutional ensembles," *Proc. VLDB Endow.*, vol. 15, no. 3, pp. 611–623, 2022.

[8] ——, "Unsupervised time series outlier detection with diversity-driven convolutional ensembles," in *Proc. VLDB Endow.*, vol. 15, no. 3, 2022, pp. 611–623.

[9] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen, "LightTS: Lightweight time series classification with adaptive ensemble distillation," *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 171:1–171:27, 2023.

[10] V. Cerqueira, L. Torgo, and C. Soares, "Early anomaly detection in time series: a hierarchical approach for predicting critical health episodes," *Mach. Learn.*, vol. 112, no. 11, pp. 4409–4430, 2023.

[11] H. Chen, S. Shi, Y. Li, and Y. Zhang, "Neural collaborative reasoning," in *WWW*, 2021, pp. 1516–1527.

[12] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *NeurIPS*, 2018.

[13] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[14] R. Cirstea, C. Guo, and B. Yang, "Graph attention recurrent neural networks for correlated time series forecasting - full version," *CoRR*, vol. abs/2103.10760, 2021.

[15] R. Cirstea, C. Guo, B. Yang, T. Kieu, X. Dong, and S. Pan, "Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting," in *IJCAI*, 2022, pp. 1994–2001.

[16] R. Cirstea, T. Kieu, C. Guo, B. Yang, and S. J. Pan, "Enhancenet: Plugin neural networks for enhancing correlated time series forecasting," in *ICDE*, 2021, pp. 1739–1750.

[17] L. donghao and wang xue, “ModernTCN: A modern pure convolution structure for general time series analysis,” in *ICLR*, 2024.

[18] B. Du, X. Sun, J. Ye, K. Cheng, J. Wang, and L. Sun, “Gan-based anomaly detection for multivariate time series using polluted training set,” *Trans. Knowl. Data Eng.*, vol. 35, no. 12, 2023.

[19] S. Elmi and K. Tan, “Deepfec: Energy consumption prediction under real-world driving conditions for smart cities,” in *WWW*, 2021, pp. 1880–1890.

[20] N. Engelmann and H. Koeppl, “Forward-backward latent state inference for hidden continuous-time semi-markov chains,” in *NeurIPS*, 2022.

[21] Y. Fang, J. Xie, Y. Zhao, L. Chen, Y. Gao, and K. Zheng, “Temporal-frequency masked autoencoders for time series anomaly detection,” in *ICDE*, 2024, pp. 1228–1241.

[22] Z. Fang, Q. Long, G. Song, and K. Xie, “Spatial-temporal graph ode networks for traffic flow forecasting,” in *SIGKDD*, 2021, pp. 364–373.

[23] Y. Feng, W. Zhang, Y. Fu, W. Jiang, J. Zhu, and W. Ren, “Sensitivehue: Multivariate time series anomaly detection by enhancing the sensitivity to normal patterns,” in *SIGKDD*, 2024, pp. 782–793.

[24] R. Ghorbani, M. J. T. Reinders, and D. M. J. Tax, “PATE: proximity-aware time series anomaly evaluation,” in *SIGKDD*, 2024, pp. 872–883.

[25] M. Goldstein and A. Dengel, “Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm,” *KI-2012: poster and demo track*, 2012.

[26] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *NeurIPS*, 2017, pp. 1024–1034.

[27] S. Hariri, M. C. Kind, and R. J. Brunner, “Extended isolation forest,” *TKDE*, vol. 33, no. 4, pp. 1479–1489, 2019.

[28] X. He, Y. Li, J. Tan, B. Wu, and F. Li, “Oneshotstl: One-shot seasonal-trend decomposition for online time series anomaly detection and forecasting,” in *Proc. VLDB Endow.*, vol. 16, no. 5, 2023, pp. 700–712.

[29] H. Hojjati, M. Sadeghi, and N. Armanfard, “Multivariate time-series anomaly detection with temporal self-supervision and graphs: Application to vehicle failure prediction,” in *PKDD*, vol. 14175, 2023.

[30] A. Huet, J. M. Navarro, and D. Rossi, “Local evaluation of time series anomaly detection algorithms,” in *SIGKDD*, 2022, pp. 635–645.

[31] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding,” in *SIGKDD*, 2018, pp. 387–395.

[32] A. S. Iyer, J. S. Nath, and S. Sarawagi, “Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection,” in *ICML*, vol. 32, 2014, pp. 530–538.

[33] S. Y. Jhin, J. Lee, and N. Park, “Precursor-of-anomaly detection for irregular time series,” in *SIGKDD*, 2023, pp. 917–929.

[34] J. Jia and A. R. Benson, “Neural jump stochastic differential equations,” in *NeurIPS*, 2019.

[35] M. Jin, Q. Wen, Y. Liang, C. Zhang, S. Xue, X. Wang, J. Zhang, Y. Wang, H. Chen, X. Li, S. Pan, V. S. Tseng, Y. Zheng, L. Chen, and H. Xiong, “Large models for time series and spatio-temporal data: A survey and outlook,” *CoRR*, vol. abs/2310.10196, 2023.

[36] M. Jin, Y. Zheng, Y. Li, S. Chen, B. Yang, and S. Pan, “Multivariate time series forecasting with dynamic graph neural odes,” *Trans. Knowl. Data Eng.*, pp. 9168–9180, 2023.

[37] N. Jones, “How machine learning could help to improve climate forecasts,” *Nature*, vol. 548, p. 379, 2017.

[38] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al, “K-means-based isolation forest,” *Knowledge-based systems*, vol. 195, p. 105659, 2020.

[39] E. J. Keogh and M. J. Pazzani, “Derivative dynamic time warping,” in *SDM*, 2001, pp. 1–11.

[40] P. Kidger, J. Morrill, J. Foster, and T. J. Lyons, “Neural controlled differential equations for irregular time series,” in *NeurIPS*, 2020.

[41] T. Kieu, B. Yang, C. Guo, R. Cirstea, Y. Zhao, Y. Song, and C. S. Jensen, “Anomaly detection in time series with robust variational quasi-recurrent autoencoders,” in *ICDE*, 2022, pp. 1342–1354.

[42] ——, “Anomaly detection in time series with robust variational quasi-recurrent autoencoders,” in *ICDE*, 2022, pp. 1342–1354.

[43] T. Kieu, B. Yang, C. Guo, C. S. Jensen, Y. Zhao, F. Huang, and K. Zheng, “Robust and explainable autoencoders for unsupervised time series outlier detection,” in *ICDE*, 2022, pp. 3038–3050.

[44] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, “Reversible instance normalization for accurate time-series forecasting against distribution shift,” in *ICLR*, 2021.

[45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.

[46] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *ICLR*, 2014.

[47] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *ICLR*, 2017.

[48] L. Kohs, B. Alt, and H. Koeppl, “Markov chain monte carlo for continuous-time switching dynamical systems,” in *ICML*, 2022.

[49] G. Lai, W. Chang, Y. Yang, and H. Liu, “Modeling long- and short-term temporal patterns with deep neural networks,” in *SIGIR*, 2018, pp. 95–104.

[50] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, and X. Hu, “Revisiting time series outlier detection: Definitions and benchmarks,” in *NeurIPS*, 2021.

[51] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, “Temporal convolutional networks for action segmentation and detection,” in *CVPR*, 2017, pp. 1003–1012.

[52] M. Li and Z. Zhu, “Spatial-temporal fusion graph neural networks for traffic flow forecasting,” in *AAAI*, 2021, pp. 4189–4196.

[53] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *ICLR*, 2018.

[54] Y. Li, W. Chen, B. Chen, D. Wang, L. Tian, and M. Zhou, “Prototype-oriented unsupervised anomaly detection for multivariate time series,” in *ICML*, vol. 202, 2023, pp. 19 407–19 424.

[55] D. Liu, J. Wang, S. Shang, and P. Han, “MSDR: multi-step dependency relation networks for spatial temporal forecasting,” in *SIGKDD*, 2022, pp. 1042–1050.

[56] Q. Liu, P. Boniol, T. Palpanas, and J. Paparrizos, “Time-series anomaly detection: Overview and new trends,” *Proc. VLDB Endow.*, vol. 17, no. 12, pp. 4229–4232, 2024.

[57] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, “Predicting taxi-passenger demand using streaming data,” *TITS*, vol. 14, no. 3, pp. 1393–1402, 2013.

[58] A. Nina, “Analysis of VLF signal noise changes in the time domain and excitations/attenuations of short-period waves in the frequency domain as potential earthquake precursors,” *Remote. Sens.*, vol. 16, no. 2, 2024.

[59] J. Paparrizos, P. Boniol, T. Palpanas, R. Tsay, A. J. Elmore, and M. J. Franklin, “Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection,” *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2774–2787, 2022.

[60] S. A. Pedersen, B. Yang, and C. S. Jensen, “Anytime stochastic routing with hybrid learning,” *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1555–1567, 2020.

[61] T. Pevný, “Loda: Lightweight on-line detector of anomalies,” *Mach. Learn.*, 2016.

[62] M. Qu and J. Tang, “Probabilistic logic neural networks for reasoning,” in *NeurIPS*, 2019, pp. 7710–7720.

[63] L. Ruff, R. A. Vandermeulen, N. G"ornitz, L. Deecke, S. A. Siddiqui, A. Binder, E. M"uller, and M. Kloft, “Deep one-class classification,” in *ICML*, vol. 80, 2018, pp. 4393–4402.

[64] S. Schmidl, F. Naumann, and T. Papenbrock, “Autotsad: Unsupervised holistic anomaly detection for time series data,” in *Proc. VLDB Endow.*, vol. 17, no. 4, 2024, pp. 766–779.

[65] C. Shang, J. Chen, and J. Bi, “Discrete graph structure learning for forecasting multiple time series,” in *ICLR*, 2021.

[66] L. Shen, Z. Li, and J. Kwok, “Timeseries anomaly detection using temporal hierarchical one-class network,” *NeurIPS*, 2020.

[67] Q. Shentu, B. Li, K. Zhao, Y. Shu, Z. Rao, L. Pan, B. Yang, and C. Guo, “Towards a general time series anomaly detector with adaptive bottlenecks and dual adversarial decoders,” in *ICLR*, 2025.

[68] S. Shih, F. Sun, and H. Lee, “Temporal pattern attention for multivariate time series forecasting,” *Machine Learning*, vol. 108, no. 8-9, pp. 1421–1441, 2019.

[69] J. Song, K. Kim, J. Oh, and S. Cho, “Memto: Memory-guided transformer for multivariate time series anomaly detection,” *NeurIPS*, 2024.

[70] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, “Robust anomaly detection for multivariate time series through stochastic recurrent neural network,” in *SIGKDD*, 2019, pp. 2828–2837.

[71] Y. Sun, G. Pang, G. Ye, T. Chen, X. Hu, and H. Yin, “Unraveling the ’anomaly’ in time series anomaly detection: A self-supervised tri-domain solution,” in *ICDE*, 2024, pp. 981–994.

[72] Z. Tian, M. Ziqing, W. Qingsong, W. Xue, S. Liang, and J. Rong, “FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting,” in *ICML*, 2022, pp. 27 268–27 286.

[73] F. A. Tobar, T. D. Bui, and R. E. Turner, “Learning stationary time series using gaussian processes with nonparametric kernels,” in *NeurIPS*, 2015, pp. 3501–3509.

[74] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” *arXiv/1607.08022*, 2016.

[75] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NeurIPS*, 2017, pp. 5998–6008.

[76] C. Wang, Z. Zhuang, Q. Qi, J. Wang, X. Wang, H. Sun, and J. Liao, “Drift doesn’t matter: Dynamic decomposition with diffusion reconstruction for unstable multivariate time series anomaly detection,” in *NeurIPS*, 2023.

[77] R. Wang, X. Mou, R. Yang, K. Gao, P. Liu, C. Liu, T. Wo, and X. Liu, “Cutaddpaste: Time series anomaly detection by exploiting abnormal knowledge,” in *SIGKDD*, 2024, pp. 3176 – 3187.

[78] X. Wu, D. Zhang, M. Zhang, C. Guo, B. Yang, and C. S. Jensen, “AutoCTS+: Joint neural architecture and hyperparameter search for correlated time series forecasting,” in *Proc. ACM Manag. Data*, 2023.

[79] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the dots: Multivariate time series forecasting with graph neural networks,” in *SIGKDD*, 2020, pp. 753–763.

[80] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph wavenet for deep spatial-temporal graph modeling,” in *IJCAI*, 2019, pp. 1907–1913.

[81] C. Xiao, Z. Gou, W. Tai, K. Zhang, and F. Zhou, “Imputation-based time-series anomaly detection with conditional weight-incremental diffusion models,” in *SIGKDD*, 2023, pp. 2742–2751.

[82] D. Xu, W. Cheng, J. Ni, D. Luo, M. Natsumeda, D. Song, B. Zong, H. Chen, and X. Zhang, “Deep multi-instance contrastive learning with dual attention for anomaly precursor detection,” in *SIAM SDM*, 2021.

[83] J. Xu, H. Wu, J. Wang, and M. Long, “Anomaly transformer: Time series anomaly detection with association discrepancy,” in *ICLR*, 2022.

[84] S. B. Yang, C. Guo, and B. Yang, “Context-aware path ranking in road networks,” *TKDE*, vol. 34, no. 7, pp. 3153–3168, 2022.

[85] Y. Yang, C. Zhang, T. Zhou, Q. Wen, and L. Sun, “Dcdetector: Dual attention contrastive representation learning for time series anomaly detection,” in *SIGKDD*, 2023, pp. 3033–3045.

[86] J. Ye, Z. Liu, B. Du, L. Sun, W. Li, Y. Fu, and H. Xiong, "Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting," in *SIGKDD*, 2022, pp. 2296–2306.

[87] A. Zhang, S. Deng, D. Cui, Y. Yuan, and G. Wang, "An experimental evaluation of anomaly detection in time series," in *Proc. VLDB Endow.*, vol. 17, no. 3, 2023, pp. 483–496.

[88] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[89] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *ICLR*, 2023.

[90] K. Zhao, C. Guo, Y. Cheng, P. Han, M. Zhang, and B. Yang, "Multiple time series forecasting with dynamic graph modeling," in *Proc. VLDB Endow.*, vol. 17, no. 4, 2023, pp. 753–765.

[91] K. Zhao, C. Guo, Y. Qiu, C. S. Jensen, Y. Cheng, and B. Yang, "Tad-up: Learning continuous and discrete dynamics for time series anomaly detection via unified probabilistic modeling," in *Proc. VLDB Endow.*, under review.

[92] K. Zhao, Y. Zheng, T. Zhuang, X. Li, and X. Zeng, "Joint learning of e-commerce search and recommendation with a unified graph neural network," in *WSDM*, 2022, pp. 1461–1469.

[93] K. Zhao, Z. Zhuang, C. Guo, H. Miao, C. S. Jensen, Y. Cheng, and B. Yang, "Unsupervised time series anomaly prediction with importance-based generative contrastive learning," in *SIGKDD*, 2025, pp. 1–12.

[94] Y. Zhao, L. Deng, X. Chen, C. Guo, B. Yang, T. Kieu, F. Huang, T. B. Pedersen, K. Zheng, and C. S. Jensen, "A comparative study on unsupervised anomaly detection for time series: Experiments and analysis," *CoRR*, vol. abs/2209.04635, 2022.

[95] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all: Power general time series analysis by pretrained LM," in *NeurIPS*, 2023.

[96] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.

Part II

Papers

Paper A

Multiple Time Series Forecasting with Dynamic Graph Modeling

Kai Zhao, Chenjuan Guo, Yunyao Cheng, Peng Han, Miao Zhang, Bin Yang

The paper has been published in the
50th International Conference on Very Large Databases (VLDB) Vol. 17(4), pp. 753-765, 2023.

Abstract

Multiple time series forecasting plays an essential role in many applications. Solutions based on graph neural network (GNN) that deliver state-of-the-art forecasting performance use the relation graph which can capture historical correlations among time series. However, in real world, it is common that correlations among time series evolve across time, resulting in dynamic relation graph, where the future correlations may be different from those in history. To address this problem, we propose multiple time series forecasting with dynamic graph modeling (MTSF-DG) that is able to learn historical relation graphs and predicting future relation graphs to capture the dynamic correlations. We also propose a causal GNN to extract features from both kinds of relation graphs efficiently. Then we propose a reasoning network to explicitly learn the variant influence from historical timestamps to future timestamps for final forecasting. Extensive experiments on six benchmark datasets show that MTSF-DG consistently outperforms state-of-the-art baselines, and justify our design with dynamic relation graph modeling.

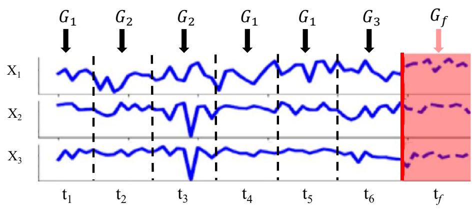
1 Introduction

Many real-world data sensed from cyber-physical systems (CPS) can be modeled as the recordings of time-dependent observations, which form the multiple time series [1–3]. For example, in power grid there exist multiple electric time series which record the energy consumption of different clients [4], and in transport network there exist multiple traffic time series which record the traffic flows and speeds at different locations across time [5, 6]. Based on the historical observations, forecasting the future observations plays an important role in ensuring effective functionality of CPS for various applications, such as spotting patterns [7–10] and predicting the future behaviors [11–13]. Thus, we aim at multiple time series forecasting problem in this paper.

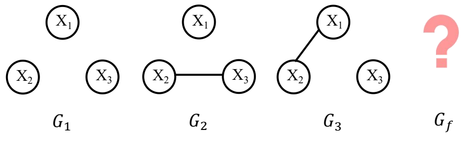
Early methods [14–16] forecast the future observations by applying machine learning models on the historical observations, which aim to learn the *temporal dependencies*. Some works [17, 18] study time series forecast-

Table A.1: The influence of dynamic relation graphs

Dataset	Metric	DCRNN	DCRNN-D	AGCRN	AGCRN-D
METR-LA	MAE	3.60	**3.55**	3.58	**3.51**
PEMS04		24.70	**21.03**	19.83	**19.51**



(a) An example of time series for traffic flows at three locations



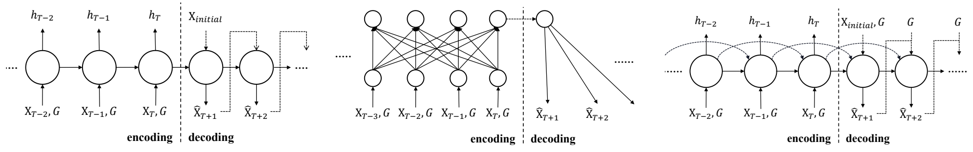
(b) The dynamic relation graphs

Fig. A.1: Motivations

ing in the traffic domain with different kinds of Graph Neural Network (GNN) [19–21] by learning the correlations among the time series of different locations based on their spatial distance. More recently, a new trend is to employ graph learning [22] to learn a *relation graph* that models correlations among multiple time series without requiring spatial distance to enable time series forecasting. For example, DGSL [23] constructs a relation graph, where time series are considered as nodes, and two time series are connected by an edge if their observations are similar in history.

Figure A.1(a) illustrates three time series that record traffic flows in three districts, *e.g.,* a commercial district $\mathbf{X}_1$, a residential district $\mathbf{X}_2$ and an industrial district $\mathbf{X}_3$. On the peak hours of workdays $\mathbf{t}_2{\sim}\mathbf{t}_3$, the traffic flows may be more correlated between $\mathbf{X}_2$ and $\mathbf{X}_3$. While on the weekends $\mathbf{t}_6$, the traffic flows may be more correlated between $\mathbf{X}_1$ and $\mathbf{X}_2$. Therefore, multiple historical relation graphs, *e.g.,* G_1, G_2, and G_3, are required to model the dynamic correlations among time series. Similarly, correlations in a future period $\mathbf{t}_f$ are different from those in history, and could be modeled by a future relation graph G_f, as shown in Figure A.1(b).

The information of dynamic relation graphs can help multiple time series forecasting, as shown in Table A.1 where two traffic datasets [17] are used. Specifically, DCRNN [17] has a static relation graph which is constructed by the distance among locations. AGCRN [24] learns a static relation



(a) RNN based model (b) Transformer based model (c) Our reasoning network

Fig. A.2: Model comparison

graph from historical time series data. However, both studies use the same, static relation graph for forecasting at different timestamps. DCRNN-D and AGCRN-D use dynamic relation graphs, which varies across time [25].

Although the state-of-the-art approaches [22, 26] can learn the relation graph to capture the correlations among multiple time series, they could only use the historical correlations. As the correlations change across time, the single, historical relation graph is insufficient to model the dynamic correlations and hinders the improvement of multiple time series forecasting. However, it is non-trivial to model the dynamic relation graphs, and to use historical observations to predict the future observations under the changeable relation graphs.

Challenge 1: It is challenging to learn the dynamic relation graphs and extract features from different relation graphs efficiently. Existing studies can construct a relation graph by learning the similarities among multiple time series from the known historical observations. It is possible for ESG [26] to cut the whole historical observations into sub time-windows, and learn a historical relation graph for each historical sub time-window using its historical observations. However, none of the existing methods have demonstrated the ability to learn a future relation graph for a future time-window, as the future observations are unknown. Besides, it is difficult to extract features from different relation graphs efficiently, as the GNN in these existing methods [26, 27] can only deal with one relation graph at one timestamp.

Challenge 2: It is challenging to predict the future observations under the changeable relation graphs. The existing methods use RNN or Transformer [28] based module to learn temporal dependencies. As shown in Figure A.2(a) and A.2(b), RNN based models only model the influence from one timestamp to the next timestamp explicitly, and Transformer based models only model the influence among historical timestamps. However, one historical timestamp may have different influence on future timestamps with regard to the changeable future relation graphs. The existing methods fail to model such different temporal dependencies explicitly.

Based on the above analysis, in this paper, we propose a novel approach MTSF-DG, multiple time series forecasting with dynamic graph modeling.

For **Challenge 1**, we cut each time series sample into historical and future time-windows, for each of which we build a relation graph distribution. It not only learns the historical relation graph distribution but also predicts the future relation graph distribution, by optimizing the correlation coefficients from an empirical covariance matrix using a memory network. Further, different from traditional GNN, which only models with one graph at a time, we propose a causal GNN, which models the observations with both historical and future relation graphs into a features efficiently.

For **Challenge 2**, we propose a reasoning network with the logical operations and symbolic reasoning procedure to explicitly learn how historical timestamps influence future timestamps. First, we learn features, say h_{T-2}, h_{T-1} and h_T, as the representations for timestamps $T-2$, $T-1$ and T. Next as shown in Figure A.2(c), We use a reasoning procedure, *e.g.,* $h_{T-2} \rightarrow h_{T-1}$ and $(h_{T-2} \wedge h_{T-1}) \rightarrow h_T$ being *TRUE* or *FALSE*, to achieve the cognition ability [29, 30] of how one historical timestamp $T-2$ may have different influence on future timestamps $T-1$ and T. In this way, we model the different temporal dependencies explicitly.

To the best of our knowledge, this is the first work that considers to predict future relation graphs, and use both historical and future relation graphs in multiple time series forecasting. And we summarize contributions as follows. First, we propose to model dynamic relation graphs and propose a causal GNN to model the observations with both historical and future relation graphs into features efficiently. Second, we propose a reasoning network to explicitly learn how historical timestamps have different influence on future timestamps. Third, by evaluating on six real-world benchmark datasets from different domains, we show that our model consistently outperforms the state-of-the-art baselines.

2 Preliminary

We formalize the multiple time series forecasting problem and introduce reasoning. The notations are summarized in Table A.2.

2.1 Problem definition

Multiple Time Series Forecasting. The multiple time series is represented as $\mathbf{X} \in \mathbb{R}^{N \times L}$, where N is the number of time series and each time series has observations during total L timestamps. We use $\mathbf{X}_t \in \mathbb{R}^N$ to indicate the observations of all time series at timestamp t, use $\mathbf{X}_{t:t+t_\Delta} \in \mathbb{R}^{N \times t_\Delta}$ to indicate the observations of all time series from timestamp t to timestamp $t + t_\Delta$, use $\mathbf{X}^i_{t:t+t_\Delta} \in \mathbb{R}^{t_\Delta}$ to indicate the observations of the i-th time series from timestamp t to timestamp $t + t_\Delta$, and use $\mathbf{X}^i_t \in \mathbb{R}^1$ to indicate the observations of the i-th time series at timestamp t, where $1 \leq i \leq N$ and $1 \leq t, t + t_\Delta \leq L$.

We formulate multiple time series forecasting problem as follows. Given a sub-sequence of historical p timestamps of observations from the multiple time series, *i.e.,* $\mathbf{X}_{T-p+1:T}$, the goal is to predict the values for the q future timestamps, *i.e.,* $\mathbf{X}_{T+1:T+q}$, where $q \geq 1$. Thus, we formulate the multiple time series forecasting problem as finding a mapping function $\mathcal{F}$ as follows:

$$\hat{\mathbf{X}}_{T+1:T+q} = \mathcal{F}_\theta(\mathbf{X}_{T-p+1:T}), \tag{A.1}$$

where θ is the parameters of $\mathcal{F}$, and $\hat{\mathbf{X}}$ denotes the predicted values of multiple time series.

Relation Graph. The latent correlations among time series at timestamp t is represented as a relation graph $G_t = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes and each node $TS_i \in \mathcal{V}$ denotes a time series so that $|\mathcal{V}| = N$, $\mathcal{E}$ is the set of edges and each edge $e_{i,j} \in \mathcal{E}$ denotes that time series i and time series j are correlated with each other, and $\mathcal{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. $\mathcal{A}_{ij} = 0$ if $e_{i,j} \notin \mathcal{E}$, $\mathcal{A}_{ij} \neq 0$ if $e_{i,j} \in \mathcal{E}$, and $\mathcal{A}_{ij}$ is the weight that denotes the degree of correlation between time series i and time series j. If $e_{i,j} \in \mathcal{E}$, node i and j are the first-hop neighbors for each other.

Table A.2: Notations

Notation	Explanation
G_t	A relation graph which presents the correlations among time series for timestamp t
v_t^i	The feature extracted from time series $\mathbf{X}_t^i$
τ	The max previous time steps used in reasoning
$\star_{G_t}$	The causal graph neural network using graph G_t
c_k	The learnable embedding matrix which denote the relative time interval with the k-th previous time step
h_t^i	The hidden state for i-th time series at timestamp t in the reasoning network

2.2 Reasoning

In this paper, we use *propositional logic*, which has basic operations, *i.e.,* conjunction (AND, $\wedge$), negation (NOT, $\neg$) and material implication ($\rightarrow$), to explicitly learn how historical timestamps have different influence on future timestamps. For time series forecasting: Each hidden state is a high-dimensional variable, such as h_t which represents the states of multiple time series at timestamp t, similar to the hidden state used in RNN. The operation over hidden states is called an *expression*, such as $(h_{t-1} \wedge h_t)$ which indicates that multiple time series have had the states h_{t-1} and h_t during two timestamps $t-1$ and t. When the expression has the material implication ($\rightarrow$) operation, it is called a *Horn clause*. The reasoning result of $(h_{t-2} \wedge h_{t-1}) \rightarrow h_t$ is $TRUE$ can show that the historical states h_{t-2} and h_{t-1} have significant influence on the future state h_t; otherwise the reasoning result is $FALSE$, the historical states h_{t-2} and h_{t-1} have little influence on the future state h_t.

3 The foundation

In this section, we theoretically analyze the limitation of existing methods and point out the importance of predicting the future relation graph distribution for dynamic graph modeling.

Specifically, we use a random variable $\mathbf{H}$ to denote a sequence of historical observations happened in the past p timestamps. Random variable

F denotes a sequence of future observations will happen in the future q timestamps. Random variable **G** denotes dynamic relation graphs at different timestamps. Then, we can model the relationship among **H**, **F**, and **G** using a structural causal model [31] as shown in Figure A.3(a), where an arrow indicates that there exists some influence. For example, the historical observations **H** influence the future observations **F**. Meanwhile, the dynamic relation graphs **G** influence the historical observations **H** and also the future observations **F**.

Existing methods extract the features from historical observations to forecast future observations. Following the probability theory, their forecasting process is learning the likelihood of conditional probability $P(\mathbf{F}|\mathbf{H})$ as:

$$\hat{\mathbf{X}}_{T+1:T+q} = \mathcal{F}_\theta(\mathbf{X}_{T-p+1:T}) = \underset{\mathbf{F}}{argmax}\, P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}). \tag{A.2}$$

By applying the Bayes rule, we can see that the dynamic relation graphs will influence the forecasting results of these existing methods. By the Bayes rule we can get:

$$\begin{aligned} &P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}) = \\ &\sum_{G_t} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, G_t) P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}), \end{aligned} \tag{A.3}$$

where G_t is the possible relation graphs at different timestamps. As the historical observations $\mathbf{X}_{T-p+1:T}$ are influenced by the historical relation graphs G_t where $t \in [T-p+1, T]$, we have:

$$\begin{aligned} P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}) \neq 0, \quad \forall t \in [T-p+1, T], \\ P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}) = 0, \quad \forall t \notin [T-p+1, T]. \end{aligned} \tag{A.4}$$

Then we can get $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ as:

$$\sum_{G_t} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, G_t) P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}). \tag{A.5}$$

From Equation (A.5), we can see that $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ used in existing methods which only learn from the historical relation graphs could lead to unsatisfactory forecasting performance once the distribution of future

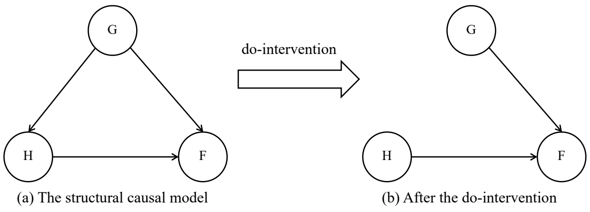

G
H
F
do-intervention
(a) The structural causal model
(b) After the do-intervention

Fig. A.3: The do-intervention for node **H**

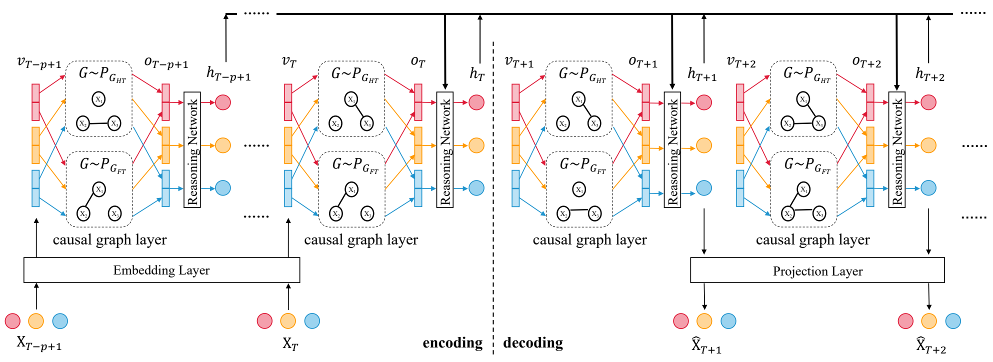


Fig. A.4: The overall framework

relation graphs, *i.e.*, G_t where $t \in [T+1, T+q]$, is changed and different from these in history.

To address this problem, we want to predict the future observations **F** based on the causal causes **H** directly. We apply the do-intervention [31] as shown in Figure A.3. We remove the arrow from **G** to **H** (the do-intervention) following Pearl's back-door criterion [32, 33], which enables us to learn the causal effect from **H** to **F** and from **G** to **F** for predicting the future observations. By this, our time series forecasting is:

$$P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T})) = \sum_{G_t} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, G_t)P(\mathbf{G} = G_t). \tag{A.6}$$

Following Equation (A.6), to forecast the future observations without the bias caused by the historical relation graphs, we should learn the probability distribution of relation graphs for both the historical and future timestamps, which will be presented in Section 4.1, and then combine the historical observations with all possible relation graphs to predict the future observations.

4 Methodology

As shown in Figure A.4, we first present the overall framework of our method, which is based on the encoder-decoder architecture and consists of four main components, *i.e.,* the embedding layer, the causal graph layer, the reasoning network and the projection layer.

The encoder takes as inputs the historical observations $\mathbf{X}_t$, with $t \in [T-p+1, T]$, and outputs hidden states h_t recurrently, which is passed to the decoder to predict future hidden states $h_{t'}$, with $t' \in [T+1, T+q]$, and then to project into the future observations $\hat{\mathbf{X}}_{t'}$.

Firstly, the embedding layer maps the original inputs, *i.e.,* the historical observations of each time series $\mathbf{X}_t$ to the high-dimensional representations v_t, with $t \in [T-p+1, T]$, which aim to extract the feature for the observation of each time series at each timestamp.

To address challenge 1, the causal graph layer contains a dynamic relation graph learning module and a causal graph neural network. Specifically, the former learns a historical relation graph distribution $P_{G_{HT}}$ to capture the correlations among time series in the p historical timestamps, and predicts a future relation graph distribution $P_{G_{FT}}$ to capture the possible correlations among observations in the q future timestamps. For each timestamp $t \in [T-p+1, T+q]$, the causal GNN samples a graph from $P_{G_{HT}}$ and $P_{G_{FT}}$, respectively, and combines them with the feature representation v_t into a hidden state o_t, such that o_t contains not only features of observations but also features of their historical and future correlations.

To address challenge 2, the reasoning network learns and outputs a feature h_t for every timestamp $t \in [T-p+1, T+q]$, using the current hidden state o_t and the previous τ features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$. The reasoning network is to identify in which way the past timestamps have different influence on the future timestamps.

The projection layer outputs the forecasting observations $\hat{\mathbf{X}}_{t'}$, based on the reasoned feature $h_{t'}$, with $t' \in [T+1, T+q]$.

4.1 The causal graph layer

We first introduce our probabilistic model which can present the distribution of dynamic relation graphs. It can not only learn the historical relation graphs but also predict the future relation graphs with a memory network. Lastly, we introduce our causal GNN, which combines the representation v_t at timestamp t with the dynamic relation graphs into a hidden state o_t efficiently.

Modeling the dynamic relation graphs

Based on the analysis in Section 3, we model the dynamic relation graphs by learning a probability distribution of relation graphs. As the historical and future relation graph distributions may be different, we model them separately.

Given the current timestamp T, we denote the probability distribution of relation graphs at **the historical time-window**, which ranges from timestamp $T-p+1$ to T, as $G \sim P_{G_{HT}}$, where G is a possible relation graph at timestamp $t \in [T-p+1, T]$. Similarly, we denote the probability distribution of relation graphs at **the future time-window**, which ranges from timestamp $T+1$ to $T+q$, as $G \sim P_{G_{FT}}$, where G is a possible relation graph at timestamp $t' \in [T+1, T+q]$.

In this work, we apply the parameterized Bernoulli distribution [23] to model the probability distribution of relation graphs, *i.e.,* $P_{G_{HT}}$ and $P_{G_{FT}}$. Specifically, $P_{G_{HT}}$ is defined as follows, for any two of the i-th and j-th time series:

$$P(\mathcal{A}_{ij} = 1) = P_h(i,j),\ P(\mathcal{A}_{ij} = 0) = 1 - P_h(i,j), \tag{A.7}$$

where $\mathcal{A}$ is the adjacency matrix for a relation graph G. Specifically, $P(\mathcal{A}_{ij} = 1)$ is the probability of the i-th and j-th time series being correlated with each other, $P(\mathcal{A}_{ij} = 0)$ is the probability of the i-th and j-th time series not correlated with each other, and $0 \leq P_h(i,j) \leq 1$ is the parameter for Bernoulli distribution, which models the correlation strength between the i-th and j-th time series in the historical time-window and needs be learned. Similarly, $P_{G_{FT}}$ is defined as follows:

$$P(\mathcal{A}_{ij} = 1) = P_f(i,j),\ P(\mathcal{A}_{ij} = 0) = 1 - P_f(i,j). \tag{A.8}$$

In the following parts, we introduce how to construct the probability distribution of relation graphs for the historical time-window and predict the probability distribution of relation graphs for the future time-window, by learning the parameters $P_h(i,j)$ and $P_f(i,j)$ from the correlation coefficients among the observations of multiple time series, which capture the degree to which two time series vary together.

Probability distribution of historical relation graphs. For the observations $\mathbf{X}_{T-p+1:T}$ in the historical time-window, we construct a historical empirical covariance matrix $S_h \in \mathbb{R}^{N\times N}$, to capture the degree to which two time series vary together, as follows:

$$S_h = \frac{1}{p}(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})^\top, \qquad \text{(A.9)}$$

where $\overline{\mathbf{X}}_{T-p+1:T}$ denotes the mean value for each time series across these p timestamps, and $\top$ is the matrix transpose operation. Then the normalized historical correlation coefficient matrix $\rho_h \in \mathbb{R}^{N\times N}$ can be obtained by:

$$\rho_h(i,j) = \frac{S_h(i,j)}{\sqrt{S_h(i,i)S_h(j,j)}}, \quad \text{for } 1 \le i,j \le N, \qquad \text{(A.10)}$$

where element $\rho_h(i,j) \in [-1,1]$ is the correlation coefficient between the i-th and j-th time series. If $\rho_h(i,j)$ is closer to 1 (or -1), the positive (or negative) correlation between i-th and j-th time series are more significant, and if $\rho_h(i,j) = 0$, the i-th and j-th time series are linearly independent with each other. Next, based on $\rho_h(i,j)$, we can learn the parameters $P_h(i,j)$. The intuition is as follows. The the bigger the correlation coefficient $|\rho_h(i,j)|$ between the i-th and j-th time series is, the more possible that the i-th and j-th time series are correlated with each other, so that the probability $P_h(i,j)$ is proportional to $|\rho_h(i,j)|$. Thus, we obtain the probability distribution of relation graphs at the historical time-window, *i.e.,* $P_{G_{HT}}$, as follows:

$$\begin{aligned} P(\mathcal{A}_{ij}=1) &= \begin{cases} P_h(i,j) = |\rho_h(i,j)| & \text{if } |\rho_h(i,j)| \ge \delta \\ 0 & \text{if } |\rho_h(i,j)| < \delta \end{cases}, \\ P(\mathcal{A}_{ij}=0) &= \begin{cases} 1 - P_h(i,j) = 1 - |\rho_h(i,j)| & \text{if } |\rho_h(i,j)| \ge \delta \\ 1 & \text{if } |\rho_h(i,j)| < \delta \end{cases}, \end{aligned} \qquad \text{(A.11)}$$

where threshold $0 \leq \delta \leq 1$ is a hyper-parameter which controls the sparsity of the relation graph. If the correlation coefficient between the i-th and j-th time series is smaller than δ, we set the i-th and j-th time series as not correlated.

Probability distribution of future relation graphs. After getting the probability distribution for the historical time-window, where observations are **available**, we proceed to predict the probability distribution for the future time-window. We use the features $E_T \in \mathbb{R}^{N \times d^c}$, which are extracted from the historical observations of multiple time series $\mathbf{X}_{T-p+1:T}$, to predict the normalized correlation coefficient matrix $\hat{\rho}_f \in \mathbb{R}^{N \times N}$ for the future time-window, and thus we learn the parameter $P_f(i,j)$ for the future relation graphs which is proportional to $|\hat{\rho}_f|$.

We could only utilize the local features E_T, which are extracted from this current historical time-window from $T-p+1$ to T, to predict the future correlation coefficient matrix $\hat{\rho}_f$. However, it prevents an accurate prediction of the future distribution, as the correlations in the future time-window may be different from those in this historical time-window [25]. Therefore, we use an additional memory unit $E \in \mathbb{R}^{N \times d^c}$ to preserve the global features, as shown in Figure A.5. The basic idea is that we use a memory unit to record the correlations that have appeared among multiple time series across all history, *i.e.,* including timestamps long before this time-window $[T-p+1, T]$, which can help to predict the relation graphs for this future time-window $[T+1, T+q]$, as correlations that occurred a long time ago may recur in the future.

Then, by querying the memory unit E with the representation matrix E_T, we learn an outcome feature matrix $E' \in \mathbb{R}^{N \times d^c}$, to predict the future correlation coefficient matrix $\hat{\rho}_f$. Thus, we are able to utilize both local and global features. Last, we update the memory unit E by adding the correlations newly learned from the outcome E'.

Specifically, we map the historical observations of each time series $\mathbf{X}^i_{T-p+1:T}$ to a high-dimensional representation $m^i_T \in \mathbb{R}^{d^c}$:

$$m^i_T = \sigma(W_c \mathbf{X}^i_{T-p+1:T}), \tag{A.12}$$

where σ is an activation function, and $W_c \in \mathbb{R}^{d^c \times p}$ is the parameters to extract the features for predicting the future correlations. We call m^i_T a

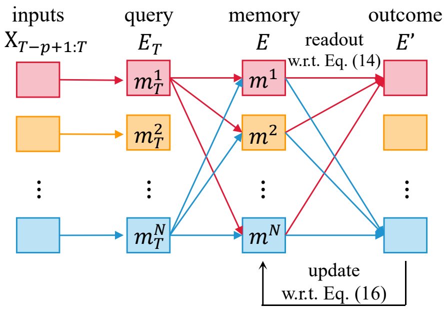


Fig. A.5: The memory network

local view, as it is time-dependent and is extracted from observations in the historical time-window. Thus, the local representation matrix for all time series are $E_T = (m_T^1, m_T^2, \cdots, m_T^N) \in \mathbb{R}^{N \times d^c}$.

Then, we use a learnable embedding $m^i \in \mathbb{R}^{d^c}$ to present each time series i across all timestamps. Thus, the memory unit for all time series are $E = (m^1, m^2, \cdots, m^N) \in \mathbb{R}^{N \times d^c}$, and the recorded correlations $\rho(i,j)$ between time series i and time series j can be obtained by the inner-product similarity between m^i and m^j as follows:

$$\rho(i,j) = m^i \cdot m^j. \tag{A.13}$$

Next, to predict the correlation coefficient matrix for the future time-window, we use the local representation matrix E_T as query to extract the outcome feature matrix $E' \in \mathbb{R}^{N \times d^c}$ from the memory unit E as follows:

$$\begin{aligned} E' &= \mathit{readout}(\mathbf{Q}, \mathbf{K}, E) = \varphi(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d^c}})E, \\ \mathbf{Q} &= E_t W_{\mathbf{Q}},\ \mathbf{K} = E\, W_{\mathbf{K}}, \end{aligned} \tag{A.14}$$

where $W_{\mathbf{Q}}, W_{\mathbf{K}} \in \mathbb{R}^{d^c \times d^c}$ are the parameters for extracting local and global features, $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N \times d^c}$ are the learned query and key to readout the features from the memory unit E, φ is the *softmax* function, and $E' \in \mathbb{R}^{N \times d^c}$ is the outcome feature matrix which has extracted both local and global features. Thus, we predict the correlation coefficient matrix for the future time-window, by using the inner-product similarity between the i-th

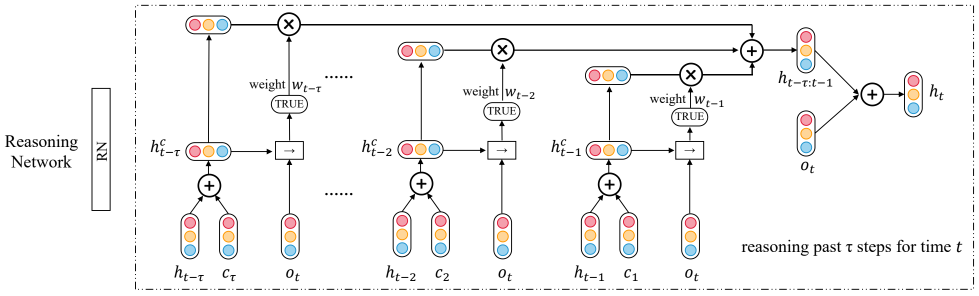


Fig. A.6: The reasoning network will output the hidden state for multiple time series at each timestamp t using the previous states from past τ steps.

element and j-th element of E' as follows:

$$\hat{\rho}_f(i,j) = E'(i) \cdot E'(j). \tag{A.15}$$

Last, we update the memory unit E as follows:

$$E = \beta E + (1-\beta)E', \tag{A.16}$$

where $0 < \beta < 1$ is a hyper-parameter which controls the percentage of existing correlations kept in the memory unit.

The memory network can be optimized by minimizing the mean square error loss function as follows:

$$\mathcal{L}_{graph} = \frac{1}{N \times N} \sum_{T} \sum_{1 \le i,j \le N} \Big(\hat{\rho}_f(i,j) - \rho_f(i,j) \Big)^2, \tag{A.17}$$

where:

$$\begin{aligned} S_f &= \frac{1}{p}(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})^\top, \\ \rho_f(i,j) &= \frac{S_f(i,j)}{\sqrt{S_f(i,i)S_f(j,j)}}, \quad \text{for } 1 \le i,j \le N. \end{aligned} \tag{A.18}$$

Similar to Equation (A.11), we can obtain the probability distribution of relation graphs at the future time-window, denoted as $P_{G_{FT}}$.

Causal graph neural network

Based on Equation (A.6), we should combine the observations together with the possible historical and future relation graphs to predict the future observations. Thus, at each timestamp t, with $t \in [T-p+1, T]$, we sample a possible relation graph according to the probability distribution $P_{G_{HT}}$ and $P_{G_{FT}}$, respectively. We denote the sampled relation graphs as G_{ht} and G_{ft}, and their adjacency matrix as $\mathcal{A}_{ht}$ and $\mathcal{A}_{ft}$. Then, we propose a causal GNN to extract feature from the sampled historical and future relation graphs, and transfer feature v_t, which contains information of the observations, into the hidden state o_t. By this, we efficiently integrate the historical observations with the possible historical and future relation graphs.

As the dynamic relation graphs can contain many possible relation graphs, the sampled relation graphs G_{ht} and G_{ft} may be different across time. It is difficult for existing GNN methods [21, 34] to learn with dynamic relation graphs, as they need to learn a set of different parameters $W_{t,k} \in \mathbb{R}^{d^e \times d^e}$, for different graphs at different timestamps as shown in Eq. (A.19):

$$o = \sum_{k=0}^{K} \left\{ \left(\mathcal{A}_t\right)^k v\, W_{t,k} \right\}, \forall t \in [T-p+1, T+q], \tag{A.19}$$

where K is a hyper-parameter which controls the max hop neighbors used in the graph convolution,

Therefore, we need to propose a causal GNN to deal with this problem. Firstly, using the feature v_t^i from the embedding layer, we learn a feature $v_{ht}^i = W_h v_t^i \in \mathbb{R}^{\frac{d^e}{2}}$ and a feature $v_{ft}^i = W_f v_t^i \in \mathbb{R}^{\frac{d^e}{2}}$ for timestamp t, which are then used to combine the historical and future relation graphs, respectively. $W_h, W_f \in \mathbb{R}^{\frac{d^e}{2} \times d^e}$ are the learnable parameters which capture the features for historical and future relation graphs, respectively.

Then, We use the $v_{ht} = (v_{ht}^1, v_{ht}^2, \cdots)$ and $v_{ft} = (v_{ft}^1, v_{ft}^2, \cdots)$ as the input features for GNN on the historical relation graph G_{ht} and the future relation graph G_{ft}, respectively. To be specific, the causal GNN on relation graph G_{ht} with input feature v_{ht}, *i.e.*, $\star_{G_{ht}}(v_{ht})$, is as follows:

$$o_{ht} = \star_{G_{ht}}(v_{ht}) = \sum_{k=0}^{K} \frac{1}{k+1} \Big\{ \Big(\widetilde{\mathcal{A}_{ht}}\Big)^{k} v_{ht} W_k \Big\}, \tag{A.20}$$

where $\widetilde{\mathcal{A}_{ht}}$ is the adjacency matrix normalized by the diagonal degree matrix D:

$$D_{i,i} = \sum_{1 \le j \le N} \mathcal{A}_{ht}(i,j), \;\; \widetilde{\mathcal{A}_{ht}} = D^{-1}\mathcal{A}_{ht}, \tag{A.21}$$

and $W_k \in \mathbb{R}^{\frac{d^e}{2} \times \frac{d^e}{2}}$, with $k \in [0, K]$, are the learnable parameters which extract the feature from k-th hop neighbors into the output $o_{ht} \in \mathbb{R}^{N \times \frac{d^e}{2}}$. The same as Equation (A.20), we can get $o_{ft} = \star_{G_{ft}}(v_{ft})$. And finally, the hidden state $o_t \in \mathbb{R}^{N \times d^e}$ for all time series, which combine the information of the historical observations with both historical and future relation graphs, is given by $o_t = (o_{ht} || o_{ft})$.

Thus, instead of learning a set of parameters $W_{t,k}$ for different relation graphs at different timestamps $t \in [T-p+1, T+q]$ with existing GNN, our causal GNN can use v_{ht} and v_{ft} to deal with the dynamics across time, and learn unified neural network parameters W_h, W_f and W_k to extract features from any relation graphs. There are two benefits: (1) It is difficult to train a set of parameters $W_{t,k}$, which can be seen in Section 5.3. (2) Our causal GNN is more efficient regarding time and space complexity, which can be seen in Section 5.5.

4.2 The reasoning network

Until now we have got the hidden state o_t, which only contains the features at each timestamp t separately. Now, we propose a reasoning network as shown in Figure A.2(c), which learns a feature h_t from not only the current hidden state o_t but also its previous τ timestamps features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, for each timestamp t recurrently. Specifically, h_t contains information of in which way its past τ timestamps influence the current timestamp t, where $1 \le \tau \le p$ is a hyper-parameter to controls the previous timestamps used in the reasoning network which make a trade off between efficiency and effectiveness. As this is a recurrent progress, when reasoning for the timestamp t to get h_t, we have obtained h_{t-k} in previous timestamps, with $1 \leq k \leq \tau$.

As shown in Figure A.6, the reasoning network firstly combines each feature h_{t-k} with a positional embedding c_k, which learns the feature to present a relative time interval from each past timestamp $t-k$ to the current timestamp t, into a feature h^c_{t-k}. Then, the reasoning network identifies the importance of the previous timestamp $t-k$ on the current timestamp t by evaluating the horn clause, *i.e.*, $h^c_{t-k} \rightarrow o_t$, and outputs the evaluating result as its importance weight w_{t-k}. Last, the reasoning network obtains the feature $h_t \in \mathbb{R}^{N \times d^e}$ for the current timestamp t, which is used for forecasting the observations, by integrating all previous features h^c_{t-k} with their importance weights w_{t-k} as follows:

$$h_t = \sum_{1 \leq k \leq \tau} w_{t-k} \times h^c_{t-k} + o_t, \tag{A.22}$$

such that h_t contains the feature of how multiple time series get into the current hidden state o_t based on the condition of previous features $(h_{t-1}, h_{t-2}, \cdots, h_{t-\tau})$.

Now we present the procedure of our reasoning network in more detail. Firstly, the fact is that under the previous features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$ all together, the multiple time series got into hidden state o_t for $t \leq T$. Therefore, we can have that $(h_{t-\tau} \wedge \cdots \wedge h_{t-2} \wedge h_{t-1}) \rightarrow o_t$ is $TRUE$ and $(h_{t-\tau} \wedge \cdots \wedge h_{t-2} \wedge h_{t-1}) \rightarrow \neg o_t$ is $FALSE$, where the conjunction operation $\wedge$ joins two features that multiple time series had in previous timestamps, $\rightarrow o_t = TRUE$ means that all the previous features will result in the current hidden state o_t, $\neg o_t$ denotes the opposite of the hidden state o_t, and $\rightarrow \neg o_t = FALSE$ means that under the previous features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, the multiple time series will not get into hidden state $\neg o_t$.

In order to capture the sequential information of the time dependent previous features, we introduce a total of τ learnable matrices $c_1, c_2, \cdots, c_\tau \in \mathbb{R}^{N \times d^e}$, which are used as the positional embedding [35]. Specifically, each c_k will learn to model the relative time interval from the past timestamp $t-k$ to the current timestamp t. Thus, we can get the time-aware features:

$$h^c_{t-k} = h_{t-k} + c_k, \text{ for } 1 \leq k \leq \tau, \tag{A.23}$$

and model the historical sequential information by:

$$\begin{aligned}(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t \textit{ is TRUE},\\ (h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow \neg o_t \textit{ is FALSE}\end{aligned} \tag{A.24}$$

where $t \leq T$.

Then, by evaluating whether each horn clause

$$h^c_{t-k} \rightarrow o_t \tag{A.25}$$

is *TRUE*, we can measure whether the previous feature h^c_{t-k} results in the current hidden state o_t. To be specific, if $h^c_{t-k} \rightarrow o_t$ is *TRUE*, we can get that previous feature h^c_{t-k} is the cause for the multiple time series to get into current state o_t. On the other hand, if $h^c_{t-k} \rightarrow o_t$ is *FALSE*, we can get that previous feature h^c_{t-k} is not the cause for the multiple time series to get into current state o_t.

Now, we model Equations (A.24) and (A.25) using our neural reasoning network, which achieves the above logic operations, *i.e.,* $\neg$, $\wedge$ and $\rightarrow$, by neural logic modules. First, We use h_a, h_b and h_c to denote any features, such as h^c_{t-k}, and each logic operation is calculated by a neural logic module with fully connected layers as follows:

$$\begin{aligned}\neg h_a &= -h_a\\ h_a \wedge h_b \wedge \cdots \wedge h_c &= \sigma\Big((h_a \odot h_b \odot \cdots \odot h_c)W_a\Big)\\ h_a \rightarrow h_b &= \sigma\Big((h_a || h_b)W_i\Big)\end{aligned} \tag{A.26}$$

where $h_a, h_b \in \mathbb{R}^{N \times d^e}$ denote the features for calculation, $W_a \in \mathbb{R}^{d^e \times d^e}$ and $W_i \in \mathbb{R}^{2d^e \times d^e}$ are the learnable network parameters to achieve the logic operations for $\wedge$ and $\rightarrow$, and $\odot$ is the Hadamard product which multiply matrix on element-wise.

Thus, all logical expressions, such as Equation (A.24), can be calculated by the neural modules using Equations (A.26), step by step. Taking Equation (A.24), $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t$, as an example, we can calculate it as follows:

$$\begin{aligned}\mathbf{Exp}_1 &= (h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1})\\ &= \sigma\Big((h^c_{t-\tau} \odot \cdots \odot h^c_{t-2} \odot h^c_{t-1})W_a\Big)\\ \mathbf{Exp}_+ &= \mathbf{Exp}_1 \rightarrow o_t = \sigma\Big((\mathbf{Exp}_1 || o_t)W_i\Big)\end{aligned} \tag{A.27}$$

where $\mathbf{Exp}_+ \in \mathbb{R}^{N \times d^e}$ is the final outcome of logical expression $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t$. In the same way, we can get the final outcome of logical expression $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow \neg o_t$ as $\mathbf{Exp}_- \in \mathbb{R}^{N \times d^e}$.

Then, another neural module, denoted as $isT()$, is used to evaluate whether an expression $\mathbf{Exp}_a$ is *TRUE* or *FALSE* by:

$$isT(\mathbf{Exp}_a) = sigmoid(\mathbf{Exp}_a W_r), \tag{A.28}$$

where $\mathbf{Exp}_a \in \mathbb{R}^{N \times d^e}$ denotes the outcome of a logical expression, such as $\mathbf{Exp}_+$, for the *TRUE/FALSE* evaluation, $W_r \in \mathbb{R}^{d^e \times 1}$ is the learnable parameter to evaluate the logical expression, and $isT(\mathbf{Exp}_a)$ is the evaluation result. The *sigmoid* function makes sure that the evaluation result is between 0 and 1, where $isT(\mathbf{Exp}_a) = 0$ means that the logical expression is *FALSE* and $isT(\mathbf{Exp}_a) = 1$ means that the logical expression is *TRUE*.

Thus, to satisfy the truth denotes by Equation (A.24), we have the logical regularization, which is achieved by minimizing the loss function as below:

$$\mathcal{L}_{reg} = \{1 - isT(\mathbf{Exp}_+)\} + isT(\mathbf{Exp}_-) \tag{A.29}$$

where $isT(\mathbf{Exp}_+)$ being 1 denotes that with the previous features $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$ the multiple time series actually got into hidden state o_t, and $isT(\mathbf{Exp}_-)$ being 0 denotes that the multiple time series did not get into hidden state $\neg o_t$.

Next, we can measure whether the previous feature h^c_{t-k} results in the current hidden state o_t, by:

$$\mathbf{Exp}_{t-k} = (h^c_{t-k} \rightarrow o_t) = (h^c_{t-k} || o_t) W_i, \tag{A.30}$$

$$w_{t-k} = isT(\mathbf{Exp}_{t-k}), \tag{A.31}$$

where w_{t-k} is the importance weight. The more the w_{t-k} is close to 1, the more possible the previous feature h^c_{t-k} is the cause for the multiple time series to get into current state o_t.

Lastly, the reasoning network get the feature $h_t \in \mathbb{R}^{N \times d^e}$ for the current timestamp t by integrating all previous features h^c_{t-k} with their importance weights w_{t-k} using Equation (A.22).

In this way, the reasoning network is able to explicitly learn how historical timestamps have different influence on future timestamps with different horn clauses.

4.3 The projection layer

We use the zero matrix $v_{T+1}, o_{T+1} \in \mathbb{R}^{N \times d^e}$ to present the initial hidden states of the multiple time series for the first future timestamp $T+1$, which denote the beginning of forecasting on the future observations. Then, after the reasoning network outputs the feature h_{T+1} based on the initial hidden state o_{T+1} and the past τ features $(h_{t-\tau+1}, \cdots, h_{t-1}, h_t)$, the projection layer outputs the forecasting observations $\hat{\mathbf{X}}_{T+1} \in \mathbb{R}^{N \times 1}$ as follows:

$$\hat{\mathbf{X}}_{T+1} = h_{T+1} W_p, \tag{A.32}$$

where $W_p \in \mathbb{R}^{d^e \times 1}$ is the network parameter mapping the features to the observations of time series. Next, the new feature v_{T+2} for next timestamp $T+2$ is obtained from the predicted observations $\hat{\mathbf{X}}_{T+1}$ as follows:

$$v_{T+2} = \hat{\mathbf{X}}_{T+1} W_n, \tag{A.33}$$

where $W_n \in \mathbb{R}^{1 \times d^e}$ is the learnable parameter to extract feature from the predicted observations. In this way, we can predict the observations for all the q future timestamps, $\hat{\mathbf{X}}_{T+1:T+q}$, recurrently.

4.4 The objective function

We use the objective function to enable model learning with gradient descent. Take the mean absolute error (MAE) as example:

$$\mathcal{L}_{MAE} = \frac{1}{N \times q} \Big|\Big| \hat{\mathbf{X}}_{T+1:T+q} - \mathbf{X}_{T+1:T+q} \Big|\Big|^1, \tag{A.34}$$

where $\Big|\Big|M\Big|\Big|^1 = \sum_{i,j} |M_{i,j}|$.

Overall, the memory network is optimized by minimizing Eq. (A.17), and the embedding layer, the causal GNN, the reasoning network and the projection layer is optimized by minimizing Eq. (A.35):

$$\mathcal{L} = \mathcal{L}_{MAE} + \lambda \mathcal{L}_{reg}, \tag{A.35}$$

Table A.3: The statistics of datasets

Dataset	N	L	Split	p	q
METR-LA	207	34,272	7:1:2	12	12
PEMS-BAY	325	52,116	7:1:2	12	12
PEMS04	307	16,992	6:2:2	12	12
PEMS08	170	17,856	6:2:2	12	12
Solar-Energy	137	52,560	6:2:2	168	1
Electricity	321	26,304	6:2:2	168	1

where λ is a hyper-parameter controls the importance of logical regularization.

5 Experiments

In this section, we empirically evaluate MTSF-DG on real-world benchmark datasets to justify our model. We introduce the multiple time series forecasting datasets, evaluation metrics and the competitor baselines first. Then we present the main results, and analyze our model with more details and ablation studies.

5.1 Experimental settings

Datasets.

We use a series of benchmark datasets from traffic and energy domains to evaluate the performance of multiple time series forecasting:

- METR-LA and PEMS-BAY: Both datasets are traffic speed time series datasets, released by Li et al. [17]. The METR-LA dataset contains the traffic speed measured by 207 sensors on the highways of Los Angeles County ranging from Mar. 2012 to Jun. 2012. The PEMS-BAY dataset contains the traffic speed measured by 325 sensors in the Bay Area ranging from Jan. 2017 to May 2017.
- PEMS04 and PEMS08: Both datasets are traffic flow time series collected from the Caltrans Performance Measurement System (PEMS), released by Bai et al. [24]. The PEMS04 dataset contains the traffic flow measured by 307 sensors in the San Francisco Bay Area ranging from Jan. 2018 to Feb. 2018. The PEMS08 dataset contains the traffic

Table A.4: Accuracy of traffic domain forecasting

Data	q	Metric	DCRNN	GWave	AGCRN	MTGNN	STFGNN	Cformer	FEDFormer	MSDR	ESG	MTSF-DG
METR -LA	3rd	MAE	2.77	2.69	2.83	2.69	2.70	2.69	2.89	2.71	2.68	**2.62**
		RMSE	5.38	5.15	5.45	5.18	5.35	5.17	5.51	5.18	5.15	**5.11**
		MAPE	7.30%	6.90%	7.56%	6.86%	7.21%	6.88%	7.63%	7.08%	6.93	**6.78%**
	6th	MAE	3.15	3.07	3.20	3.05	3.10	3.05	3.27	3.09	3.06	**2.98**
		RMSE	6.45	6.22	6.55	6.17	6.36	6.18	6.56	6.33	6.19	**6.13**
		MAPE	8.80%	8.37%	8.79%	8.19%	8.60%	8.21%	8.87%	8.57%	8.20%	**8.14%**
	12th	MAE	3.60	3.53	3.58	3.49	3.51	3.48	3.69	3.50	3.49	**3.39**
		RMSE	7.60	7.37	7.41	7.23	7.46	7.27	7.66	7.33	7.23	**7.16**
		MAPE	10.50%	10.01%	10.13%	9.87%	10.05%	9.86%	10.44%	9.98%	9.96%	**9.58%**
PEMS -BAY	3rd	MAE	1.38	1.30	1.35	1.32	1.34	1.31	1.47	1.32	1.31	**1.28**
		RMSE	2.95	2.74	2.83	2.79	2.81	2.75	3.02	2.84	2.74	**2.67**
		MAPE	2.90%	2.73%	2.87%	2.77%	2.84%	2.72%	2.96%	2.77%	2.76%	**2.63%**
	6th	MAE	1.74	1.63	1.69	1.65	1.66	1.63	1.81	1.64	1.63	**1.57**
		RMSE	3.97	3.70	3.81	3.74	3.76	3.69	4.01	3.78	3.71	**3.63**
		MAPE	3.90%	3.67%	3.84%	3.69%	3.83%	3.66%	3.99%	3.68%	3.69%	**3.56%**
	12th	MAE	2.07	1.95	1.96	1.94	1.98	1.93	2.06	1.94	1.92	**1.85**
		RMSE	4.74	4.52	4.52	4.49	4.52	4.45	4.78	4.51	4.42	**4.35**
		MAPE	4.90%	4.63%	4.67%	4.53%	4.73%	4.49%	4.88%	4.55%	4.52%	**4.40%**
PEMS 04	1∼12	MAE	24.70	19.16	19.83	19.32	19.83	19.50	23.48	19.29	19.47	**18.67**
		RMSE	38.12	30.46	32.26	31.57	31.88	32.00	37.27	31.54	31.66	**30.17**
		MAPE	17.12%	13.26%	12.97%	13.52%	13.02%	13.07%	15.44%	12.89%	13.30%	**12.64%**
PEMS 08	1∼12	MAE	17.86	15.13	15.95	15.71	16.64	15.88	17.24	15.11	15.70	**14.80**
		RMSE	27.83	24.07	25.22	24.62	26.22	25.07	26.93	24.42	24.81	**23.68**
		MAPE	11.45%	10.10%	10.09%	10.03%	10.60%	10.17%	11.21%	9.93%	10.07%	**9.54%**

flow measured by 170 sensors in the San Bernardino Area ranging from Jul. 2016 to Aug. 2016.

- Solar-Energy: The Solar-Energy dataset contains the solar power production records collected from 137 PV plants in the Alabama State in 2007, released by Lai et al. [16].
- Electricity: The Electricity energy dataset contains the electricity consumption records collected from 321 clients from 2012 to 2014, released by Lai et al. [16].

The detailed statistics of these datasets are shown in Table A.3, where N is the number of time series and L is the total number of timestamps. We follow the same train-validation-test splits as in the original papers [16, 17, 24], as shown in the "Split" column. More details about datasets can be seen in the Appendix.

Evaluation metrics.

Following the evaluation methodology in existing works [16, 17, 24], we use mean absolute error (MAE), root mean squared error (RMSE), mean absolute percentage error (MAPE) to evaluate the accuracy of multi-step forecasting, and use Root Relative Squared Error (RRSE) and Empirical Correlation Coefficient (CORR) to measure the accuracy of single-step forecasting. For MAE, RMSE, MAPE, and RRSE, lower values indicate higher accuracy, while larger CORR values indicate higher accuracy.

Baselines.

We compare MTSF-DG with baseline methods summarized as follows:

- **Methods without relation graphs.** VAR-MLP: It is an auto-regressive model using multilayer perception (MLP) [15]. GP: It uses Gaussian Process to model time series [14]. LSTNet: It combines convolutional neural network (CNN) with RNN to learn temporal dependencies [16]. TPA: It is a naive Transformer model [36]. FEDFormer: It uses frequency-enhanced Transformer to extract trend and periodic features [37]. Crossformer: It is the state-of-art Transformer based model, which uses a cross-dimension attention to learn the historical correlations among time series, without learning graphs [38].
- **Methods with a pre-defined graph.** DCRNN: It proposes diffusion graph convolutions to extract spatial dependencies [17]. GWave: It proposes 1D dilated CNN and combines with diffusion graph convolutions [18]. AGCRN: It proposes adaptive recurrent graph convolution network [24]. MSDR: It proposes attention based graph convolutions and multi-step RNN [39].
- **Methods that learn relation graphs.** MTGNN: It learns a static relation graph that models similarities among multiple time series, and uses graph convolutions and CNN for forecasting [22]. DGTS: It constructs a static relation graph based on the Euler distances among multiple time series, and uses recurrent graph convolutions for forecasting [23]. STFGNN: It constructs a static relation graph based on the Dynamic Time Warping similarities [40] among multiple time series [41]. ESG: It cuts the historical observations into sub time-windows, learns a historical relation graph for each time-window separately, and use RNN for forecasting [26].

We report results from the original papers if baselines conduct experiments on the dataset with the same setting. For the rest, we have carefully tuned the hyper-parameters based on the recommendations from their original papers.

5.2 Overall comparison.

Tables A.4 and A.5 present the accuracy of MTSF-DG and the baselines on all datasets. We randomly repeat each method 5 times and report the average result. We use bold to highlight the best accuracy, which significantly outperforms the underline second best accuracy.

Key observations are as follows. Firstly, MTSF-DG consistently outperforms the state-of-the-art baseline methods on all datasets. It demonstrates that MTSF-DG is able to learn the dynamic correlations among multiple time series and use them to improve the forecasting performance.

Secondly, from Table A.5 we observe that the MTGNN, DGTS, ESG and MTSF-DG methods, which can learn the relation graph(s) for multiple time series, perform better when comparing to VAR-MLP, GP, LSTNet, TPA and FEDFormer, which cannot capture the relations among multiple time series. Crossformer learns the correlations among time series with Cross-Transformer, which also performs better when comparing to VAR-MLP, GP, LSTNet, TPA and FEDFormer. It demonstrates that capturing the relations among multiple time series is important for multiple time series forecasting.

Thirdly, we observe that our MTSF-DG method is also superior when comparing to the other graph based methods, which use a single relation graph or only learn historical relation graphs, to enhance the forecasting accuracy. This is due to the fact that the baselines cannot capture the dynamic correlations among multiple time series which may change across time and be different in the future, where different future relation graphs may influence the future observations differently. A single relation graph or the historical relation graph will bias the forecasting. GWave, MTGNN, STFGNN, MSDR and ESG can only have the second best accuracy on some datasets. There does not exist a single baseline method that consistently outperforms others, which suggests that a single relation graph or the historical relation graph is insufficient for multiple time series forecasting. In contrast, our MTSF-DG can learn the historical and future correlations dynamically, and consistently outperforms baseline methods.

Table A.5: Accuracy of energy domain forecasting

Dataset		Solar-Energy		Electricity	
		q		q	
Method	Metric	3rd	12th	3rd	12th
VAR-MLP	RRSE	0.1922	0.4244	0.1393	0.1557
	CORR	0.9829	0.9058	0.8708	0.8192
GP	RRSE	0.2259	0.5200	0.1500	0.1621
	CORR	0.9751	0.8518	0.8670	0.8394
LSTNet	RRSE	0.1843	0.3254	0.0864	0.1007
	CORR	0.9843	0.9467	0.9283	0.9077
TPA	RRSE	0.1803	0.3234	0.0823	0.0964
	CORR	0.9850	0.9487	0.9439	0.9250
MTGNN	RRSE	0.1778	0.3109	0.0745	0.0916
	CORR	0.9852	0.9509	0.9474	0.9278
DGTS	RRSE	0.1791	0.3144	0.0767	0.0925
	CORR	0.9852	0.9501	0.9470	0.9275
Crossformer	RRSE	0.1772	0.3089	0.0741	0.0905
	CORR	0.9859	0.9511	0.9474	0.9291
FEDFormer	RRSE	0.1788	0.3141	0.0769	0.0924
	CORR	0.9852	0.9498	0.9465	0.9280
ESG	RRSE	<u>0.1708</u>	<u>0.3073</u>	<u>0.0718</u>	<u>0.0898</u>
	CORR	<u>0.9865</u>	<u>0.9519</u>	<u>0.9494</u>	<u>0.9321</u>
MTSF-DG	RRSE	**0.1692**	**0.3025**	**0.0701**	**0.0882**
	CORR	**0.9874**	**0.9533**	**0.9502**	**0.9339**

Lastly, MTSF-DG achieves the best accuracy compared to Transformer based models. This suggests that our reasoning network is also good at learning temporal dependencies by explicitly learning how historical timestamps have different influence on future timestamps. This enables MTSF-DG to get more high performance compared to TPA, FEDFormer and Crossformer.

5.3 Ablation studies.

We conduct ablation studies to validate the effectiveness of our key components that contribute to the improvements. In particular, we compare MTSF-DG with the following variants:

- w/o memory network: This variant does not use the memory network for predicting the future relation graph distribution. It directly uses the local features E_T.
- w/o causal GNN: This variant does not use the causal GNN. It use

Table A.6: Ablation studies

Dataset	PEMS04			PEMS08		
Method	MAE	RMSE	MAPE	MAE	RMSE	MAPE
MTSF-DG	**18.67**	**30.17**	**12.64%**	**14.80**	**23.68**	**9.54%**
w/o memory	18.68	30.27	12.79%	14.82	23.69	9.58%
w/o causal GNN	18.72	30.23	12.76%	14.85	23.73	9.59%
G_{ht} only	18.98	30.59	12.93%	15.07	23.97	9.85%
G_{ft} only	18.93	30.51	12.91%	14.93	23.95	9.77%
w Transformer	18.91	30.44	12.83%	14.98	23.93	9.69%

Table A.7: Parameter sensitivity

Dataset	PEMS04			PEMS08		
τ	MAE	RMSE	MAPE	MAE	RMSE	MAPE
2	19.83	31.21	12.91%	15.63	25.12	10.00%
3	19.20	30.77	12.73%	15.03	24.23	9.76%
4	18.67	30.17	12.64%	14.80	23.68	9.54%
5	18.69	30.22	12.65%	14.79	23.73	9.52%
12	18.72	30.25	12.69%	14.82	23.76	9.60%

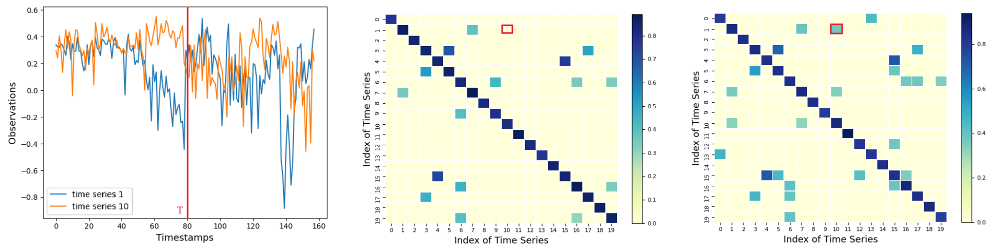
the existing GNN on the sampled historical relation graph and future relation graph at each timestamp with Eq. (A.19), and learn a set of parameters $W_{t,k}$.

- G_{ht} only: This variant does not use causal graph layer. It applies the existing GNN [18] on the sampled historical relation graph G_{ht} only.
- G_{ft} only: This variant does not use causal graph layer. It applies the existing GNN [18] on the sampled future relation graph G_{ft} only.
- w Transformer: This variant does not use reasoning network. It uses Transformer [35] to model with the hidden states.

Table A.6 shows the accuracy of different variants on PEMS04 and PEMS08 datasets. For the other datasets, the results show similar trends. From Table A.6 we observe that: (1) MTSF-DG achieves better accuracy comparing to its variant w/o memory network. This demonstrates the effectiveness of the proposed memory network for predicting the future relation graph distribution. It can improve the forecasting accuracy by providing more accurate future correlations among multiple time series. (2) The information from different hops are not equally important. The near hop neighbors are more important to present the correlations among time series. (3) The existing GNN which learns a set of parameter will

Table A.8: Runtime and total parameters used

Dataset	PEMS04		PEMS08	
Method	Runtime (s/epoch)	Parameters (K)	Runtime (s/epoch)	Parameters (K)
DCRNN	226	371	159	371
DGTS	262	381	179	378
MSDR	618	1174	438	1174
ESG	378	1205	255	1205
MTSF-DG	217	803	151	799
w/o causal GNN	298	1270	194	1205



(a) Dynamic correlations between time series 1 and 10 (b) Heatmap for $P_{G_{HT}}$ before T (c) Heatmap for $P_{G_{FT}}$ after T

Fig. A.7: Case study

suffer from over-fitting and performs worse. (4) Simply using the existing GNN [17, 18] with a single historical relation graph or future relation graph will lower the performance significantly. This result is consistent with our analyses in Section 3, suggesting that our causal GNN, which can learn with historical relation graph and future relation graph jointly, is more effective in multiple time series forecasting. (4) MTSF-DG with reasoning network outperforms the variant with Transformer, which justifying that the Transformer, which only model the influence among historical timestamps, is insufficient to accurately forecast future observations under changeable future relation graphs. However, our reasoning network is able to do so, as it can explicitly learn how historical timestamps have different influence on future timestamps.

5.4 Parameter sensitivity

We evaluate the impact of the hyperparameter, *i.e.*, τ, which controls the max previous timestamps used in the reasoning network. The experimental

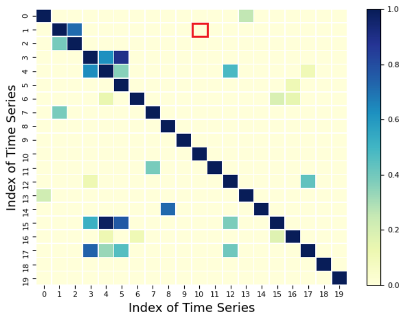


Fig. A.8: The static graph from DGTS

Table A.9: Comparisons between representative methods.

Method	Dynamic relation graphs in encoder	Dynamic relation graphs in decoder
[14–16, 28, 36–38, 42, 43]	$\times$, only use historical observations	$\times$
[22–24, 39, 41, 44–47]	$\times$, only a static relation graph	$\times$
[26, 27]	$\surd$, but only historical relation graphs	$\times$
MTSF-DG	$\surd$	$\surd$

results are shown in Table A.7. If we use more previous timestamps in the reasoning network up to 4 timestamps, the MTSF-DG model performs better. The reason is that for the traffic prediction task we need to learn the temporal influence for a long range. When the value of τ changes to 5 and 12, the accuracy results are relatively stable.

5.5 Runtime and total parameters used

For our dynamic graph learning, the time complexity is $O(N^2d)$. For causal GNN, the time complexity is $O(KN^2d)$. For reasoning network, the time complexity is $O((p+q)Nd)$. We also show the overall runtime and the number of total parameters used for different methods in Table A.8.

We can see that MTSF-DG is better than these baseline methods regarding the runtime. We can also see that the time and space complexity of MTSF-DG is smaller than MSDR, ESG and the variant w/o causal GNN. Our model is only worse than DCRNN and DGTS regarding the number of total parameters used. This is because DCRNN and DGTS use only one relation graph for all timestamps, and our model need more space to learn the dynamic relation graphs.

5.6 Case study

To show the superiority of our dynamic graph modeling, we give the case study on METR-LA dataset and visualize in Figure A.7. From Figure A.7(a), we can see that the time series 1 and 10 are more correlated to each other during the first 40 timestamps and the 80-120 timestamps, and are less correlated during the 40-80 timestamps. At timestamp T = 80, if we use the 12 historical observations to predict the 12 future observations, these dynamic correlations are captured by the probability distribution of historical relation graphs $P_{G_{HT}}$ and the probability distribution of future relation graphs $P_{G_{FT}}$, as shown in Figure A.7(b) and Figure A.7(c). However, the baseline DGTS can only learn a static, historical relation graph as shown in Figure A.8, which cannot capture the correlation between time series 1 and 10 that sometimes occurs. By learning such dynamic relation graphs for historical and future time-window, our MTSF-DG can improve the forecasting performance.

6 Related Work

We review existing works on time series forecasting and graph learning, and summary them in Table A.9. We also compare reasoning network with RNN and Transformer based models in Figure A.2.

6.1 Time series forecasting

Early methods try to utilize the statistical methods, *e.g.,* Auto-Regressive model (AR) [15] and Gaussian Process model (GP) [14], to forecast on time series, which model the future observations as the linear combination of the nearby historical observations, called the temporal dependencies. Some works [36, 42] proposed to utilize RNN [48], TCN [49] or attention network [35] for time series forecasting by modeling dependencies using more historical observations. LSTNet [16] employs 1D CNN and RNN to capture temporal dependencies. N-BEATS [50] uses fully connected MLP and residual blocks to predict the trend and periodicity. Timesnet [43] propose a 2D CNN to capture temporal dependencies and periodic frequency. Triformer [28] proposes a Transformer based time series forecasting model with linear complexity attention. FEDFormer [37] uses frequency-enhanced Transformer to extract trend and periodic features. Crossformer [38] uses a cross-dimension attention to implicitly learn the historical correlations among time series without graphs.

However, these methods only use historical observations to predict future observations, and cannot use a relation graph to capture correlations among time series explicitly.

Then, there have been a lot of GNN based methods for traffic time series forecasting [44–46]. One kind of correlations among multiple time series can be the spatial distance among different locations, which naturally form a spatial graph. The GNN based methods capture spatial dependencies by aggregating features [34, 51] from the neighbors on the spatial graph. DCRNN [17] proposes diffusion graph convolutions to extract spatial dependencies and use GRU for forecasting. AGCRN proposes adaptive recurrent graph convolution network to capture node-specific features for each time series [24]. Graph WaveNet [18] combines diffusion graph convolutions with gated dilated TCN.

However, these methods need a pre-defined graph to present correlations among time series in advance, and they cannot model the dynamic relation graphs.

6.2 Graph learning

Most recently, a new trend is to employ graph learning [25] to learn relation graphs that models correlations among multiple time series without requiring spatial distance in advance, to enable universal multiple time series forecasting. MTGNN [22], STFGNN [41] and DGSL [23] construct a static relation graph, where time series are considered as nodes, and two time series are connected by an edge if their observations are similar measured by the Cosine distances, Euler distances or Dynamic Time Warping distances [40]. AutoCTS [47] automatically search from RNN, TCN, GNN and attention network, to build a better deep neural network to learn a static relation graph and predict future observations.

EnhanceNet [27] and ESG [26] cut the historical observations into sub time-windows, and construct a historical relation graph which is used in each time-window separately.

However, EnhanceNet and ESG are incapable of learning the dynamic correlations for the future timestamps. Besides, they only use dynamic historical relation graphs in encoder to extract the invariant temporal dependencies from historical observations, which is used to predict the observations for all future timestamps simultaneously. They fail to use the dynamic relation graphs in decoder and cannot learn the different temporal dependencies. But our MTSF-DG can learn the dynamic correlations for the future timestamps using the memory network, and learn the different temporal dependencies using the reasoning network in both encoder and decoder.

7 Conclusion

We present MTSF-DG for multiple time series forecasting. We propose to learn historical relation graphs, and predict future relation graphs to capture the dynamic correlations with the memory network, by optimizing the relation graph distributions from an empirical covariance matrix. Then we propose a causal GNN to extract features from both historical and future relation graphs efficiently. Lastly, we propose a reasoning network to explicitly learn how historical timestamps have different influence on future timestamps with the logical operations and symbolic reasoning procedure, and predict the future observations based on reasoning the future feature vectors. Experiments on six benchmark datasets demonstrate the superiority of our method. In future work, it is of interest to extend MTSF-DG to other time series tasks, such as abnormality detection and prediction.

A Appendix

In this section, we provide more information for supplement. Our codes are at https://github.com/zhkai/MTSF-DG

B Dataset

The four traffic datasets have pre-defined graphs where each node represents a time series and the adjacency matrix represents the road network distances among time series. The two energy datasets have no pre-defined graphs.

Following existing works [16, 17, 22, 24, 39, 41], we adopt a multi-step forecasting setting for the four traffic datasets, where we use 12 historical timestamps to forecast the observations for the next all 12 timestamps. We adopt a single-step forecasting setting for the two energy datasets, where we use 168 historical timestamps to forecast the observations for the next 3rd timestamp and 12th timestamp, respectively.

To enable direct and fair comparisons with existing works [16, 17, 22, 24, 39, 41], for METR-LA and PEMS-BAY, we report the accuracy of the forecast on the future 3rd, 6th, and 12th timestamp, respectively; for PEMS04 and PEMS08, we report the average accuracy over the all future 12 timestamps; for Solar-Energy and Electricity, we report the accuracy of the forecast on the next 3rd and 12th timestamp, respectively.

C Experimental Details

We conduct grid search on the held-out validation set for each method and dataset to decide its hyper-parameters. Specifically, We optimize with Adam optimizer for a maximum of 200 epochs and use the early stop strategy with patience of 10. The learning rate is tuned from 0.0005, 0.001, 0.0015 and the batch size is tuned from 32, 64, 128. The dimension of hidden state is tuned from 32, 64, 128. The δ, which controls the sparsity of the relation graph, is tuned from 0.1, 0.3, 0.5. The K, which controls the max hop neighbors used in causal convolution, is tuned from 1, 2, 3. The τ, which controls the max previous timestamps used in the reasoning network, is tuned from 2, 3, 4, 5, 6, 12. The β used in the memory unit is set to 0.99, and the λ used in the loss function is set to 0.001.

D Implementation

All the model training experiments are conducted on a server running Ubuntu 18.04.5 LTS system, with Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz and NVIDIA Quadro RTX 8000 GPU. Baseline models in Python and Pytorch are open available from the original papers, and all the deep learning models are executed with Pytorch 1.2.0.

References

[1] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, “Predicting taxi-passenger demand using streaming data,” *TITS*, vol. 14, no. 3, pp. 1393–1402, 2013.

[2] X. Wu, D. Zhang, M. Zhang, C. Guo, B. Yang, and C. S. Jensen, “AutoCTS+: Joint neural architecture and hyperparameter search for correlated time series forecasting,” in *Proc. ACM Manag. Data*, 2023.

[3] R. Cirstea, C. Guo, and B. Yang, “Graph attention recurrent neural networks for correlated time series forecasting - full version,” *CoRR*, vol. abs/2103.10760, 2021.

[4] S. Elmi and K. Tan, “Deepfec: Energy consumption prediction under real-world driving conditions for smart cities,” in *WWW*, 2021, pp. 1880–1890.

[5] Y. Cheng, P. Chen, C. Guo, K. Zhao, Q. Wen, B. Yang, and C. S. Jensen, “Weakly guided adaptation for robust time series forecasting,” in *Proc. VLDB Endow.*, vol. 17, no. 4, 2024, pp. 766–779.

[6] H. Miao, Y. Zhao, C. Guo, B. Yang, Z. Kai, F. Huang, J. Xie, and C. S. Jensen, “A unified replay-based continuous learning framework for spatio-temporal prediction on streaming data,” *ICDE*, pp. 1050–1062, 2024.

[7] N. Jones, “How machine learning could help to improve climate forecasts,” *Nature*, vol. 548, p. 379, 2017.

[8] T. Kieu, B. Yang, C. Guo, R. Cirstea, Y. Zhao, Y. Song, and C. S. Jensen, “Anomaly detection in time series with robust variational quasi-recurrent autoencoders,” in *ICDE*, 2022, pp. 1342–1354.

[9] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen, “Unsupervised time series outlier detection with diversity-driven convolutional ensembles,” *Proc. VLDB Endow.*, vol. 15, no. 3, pp. 611–623, 2022.

[10] T. Kieu, B. Yang, C. Guo, C. S. Jensen, Y. Zhao, F. Huang, and K. Zheng, “Robust and explainable autoencoders for unsupervised time series outlier detection,” in *ICDE*, 2022, pp. 3038–3050.

[11] S. B. Yang, C. Guo, and B. Yang, “Context-aware path ranking in road networks,” *TKDE*, vol. 34, no. 7, pp. 3153–3168, 2022.

[12] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen, “LightTS: Lightweight time series classification with adaptive ensemble distillation,” *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 171:1–171:27, 2023.

[13] S. A. Pedersen, B. Yang, and C. S. Jensen, “Anytime stochastic routing with hybrid learning,” *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1555–1567, 2020.

[14] F. A. Tobar, T. D. Bui, and R. E. Turner, “Learning stationary time series using gaussian processes with nonparametric kernels,” in *NeurIPS*, 2015, pp. 3501–3509.

[15] G. P. Zhang, “Time series forecasting using a hybrid ARIMA and neural network model,” *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[16] G. Lai, W. Chang, Y. Yang, and H. Liu, “Modeling long- and short-term temporal patterns with deep neural networks,” in *SIGIR*, 2018, pp. 95–104.

[17] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *ICLR*, 2018.

[18] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph wavenet for deep spatial-temporal graph modeling,” in *IJCAI*, 2019, pp. 1907–1913.

[19] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *NeurIPS*, 2017, pp. 1024–1034.

[20] K. Zhao, Y. Zheng, T. Zhuang, X. Li, and X. Zeng, “Joint learning of e-commerce search and recommendation with a unified graph neural network,” in *WSDM*, 2022, pp. 1461–1469.

[21] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *ICLR*, 2017.

[22] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the dots: Multivariate time series forecasting with graph neural networks,” in *SIGKDD*, 2020, pp. 753–763.

[23] C. Shang, J. Chen, and J. Bi, “Discrete graph structure learning for forecasting multiple time series,” in *ICLR*, 2021.

[24] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, “Adaptive graph convolutional recurrent network for traffic forecasting,” in *NeurIPS*, 2020, pp. 11 465–11 475.

[25] D. Hallac, Y. Park, S. P. Boyd, and J. Leskovec, “Network inference via the time-varying graphical lasso,” in *SIGKDD*, 2017, pp. 205–213.

[26] J. Ye, Z. Liu, B. Du, L. Sun, W. Li, Y. Fu, and H. Xiong, “Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting,” in *SIGKDD*, 2022, pp. 2296–2306.

[27] R. Cirstea, T. Kieu, C. Guo, B. Yang, and S. J. Pan, “Enhancenet: Plugin neural networks for enhancing correlated time series forecasting,” in *ICDE*, 2021, pp. 1739–1750.

[28] R. Cirstea, C. Guo, B. Yang, T. Kieu, X. Dong, and S. Pan, “Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting,” in *IJCAI*, 2022, pp. 1994–2001.

[29] M. Qu and J. Tang, “Probabilistic logic neural networks for reasoning,” in *NeurIPS*, 2019, pp. 7710–7720.

[30] H. Chen, S. Shi, Y. Li, and Y. Zhang, “Neural collaborative reasoning,” in *WWW*, 2021, pp. 1516–1527.

[31] G. Madelyn, J. Pearl, and N. P. Jewell, *Causal inference in statistics: A primer.* John Wiley & Sons, 2016.

[32] Z. Yue, H. Zhang, Q. Sun, and X. Hua, “Interventional few-shot learning,” in *NeurIPS*, 2020.

[33] T. Wang, J. Huang, H. Zhang, and Q. Sun, “Visual commonsense R-CNN,” in *CVPR*, 2020, pp. 10 757–10 767.

[34] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *ICLR, 2018*, 2018.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NeurIPS*, 2017, pp. 5998–6008.

[36] S. Shih, F. Sun, and H. Lee, “Temporal pattern attention for multivariate time series forecasting,” *Machine Learning*, vol. 108, no. 8-9, pp. 1421–1441, 2019.

[37] Z. Tian, M. Ziqing, W. Qingsong, W. Xue, S. Liang, and J. Rong, “FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting,” in *ICML*, 2022, pp. 27 268–27 286.

[38] Y. Zhang and J. Yan, “Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting,” in *ICLR*, 2023.

[39] D. Liu, J. Wang, S. Shang, and P. Han, “MSDR: multi-step dependency relation networks for spatial temporal forecasting,” in *SIGKDD*, 2022, pp. 1042–1050.

[40] E. J. Keogh and M. J. Pazzani, “Derivative dynamic time warping,” in *SDM*, 2001, pp. 1–11.

[41] M. Li and Z. Zhu, “Spatial-temporal fusion graph neural networks for traffic flow forecasting,” in *AAAI*, 2021, pp. 4189–4196.

[42] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, “MICN: Multi-scale local and global context modeling for long-term series forecasting,” in *ICLR*, 2023.

[43] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “Timesnet: Temporal 2d-variation modeling for general time series analysis,” in *ICLR*, 2023.

[44] R. Cirstea, B. Yang, C. Guo, T. Kieu, and S. Pan, "Towards spatio-temporal aware traffic time series forecasting," in *ICDE*, 2022, pp. 2900–2913.

[45] Z. Fang, Q. Long, G. Song, and K. Xie, "Spatial-temporal graph ode networks for traffic flow forecasting," in *SIGKDD*, 2021, pp. 364–373.

[46] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *IJCAI*, 2018, pp. 3634–3640.

[47] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, and C. S. Jensen, "Autocts: Automated correlated time series forecasting," *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 971–983, 2022.

[48] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[49] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *CVPR*, 2017, pp. 1003–1012.

[50] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: neural basis expansion analysis for interpretable time series forecasting," in *ICLR*, 2020.

[51] K. Zhao, T. Bai, B. Wu, B. Wang, Y. Zhang, Y. Yang, and J. Nie, "Deep adversarial completion for sparse heterogeneous information network embedding," in *WWW*, 2020, pp. 508–518.

Paper B

TAD-UP: Learning Continuous and Discrete Dynamics for Time Series Anomaly Detection via Unified Probabilistic Modeling

Kai Zhao, Chenjuan Guo, Yuying Qiu, Christian S. Jensen, Yunyao Cheng, Bin Yang

The paper has been submitted for publication

Abstract

Anomaly detection for multivariate time series plays an important role in many real-world applications, enabling, e.g., fault analyses in cyber-physical systems and risk monitoring in environmental systems. While existing methods achieve good results on time series with regular timestamps, they struggle when having to learn both continuous and discrete dynamics for multiple variables across continuous time. Further, existing methods simply sum up reconstruction or contrastive errors from each variable to obtain final anomaly scores. They struggle to recognize the importance of variables with different measurement units. To overcome these limitations, we propose TAD-UP that learns both continuous and discrete dynamics for Time series Anomaly Detection via Unified Probabilistic modeling. First, we propose two co-dependent branches of neural ordinary differential equations using the Compound Poisson Process to learn both continuous and discrete dynamics for different time series variables and propose the gate mechanism to learn correlations among different dynamics. Second, we propose a unified joint probability distribution modeling. The resulting model is optimized using Maximum Likelihood Estimation on joint variables, instead of reconstruction or contrastive losses. We detect anomalies using joint probabilities, which take the marginal probabilities of different variables into account as their importance on final anomaly scores. Extensive experiments on nine real-world benchmark datasets from different domains offer evidence that TAD-UP is capable of state-of-the-art performance, making it a promising solution for real-world time series anomaly detection. All codes will be made available upon acceptance.

1 Introduction

Cyber-physical systems employ sensors to monitor their environment and runtime status, thereby producing multivariate time series data, consisting of sequences of timestamped recordings of multiple variables with different physical and mathematical meanings [1–4], as exemplified in Figure B.1. For example, computing systems may record a variety of runtime indicators [5, 6], and environmental systems may record weather and water information [7, 8].

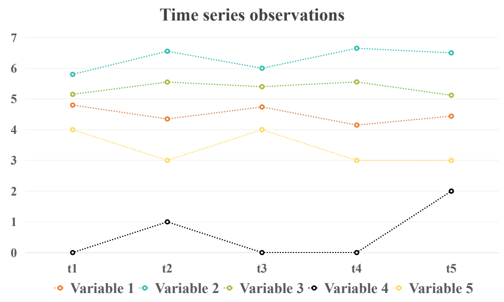
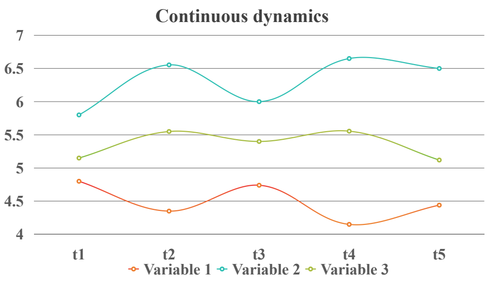


Fig. B.1: Example timestamped multivariate time series.

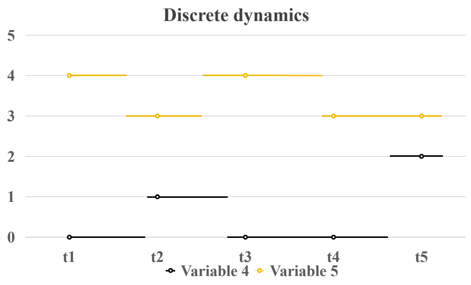
Effectively detecting anomalies in multivariate time series is essential in many real-world applications, as it supports fault analyses and facilitates risk monitoring to ensure the normal operation of cyber-physical systems, thereby avoiding economic losses and health concerns [9–11]. For example, detecting anomalies can help locate and handle service faults and interruptions threats in computing systems [11, 12] and can warn against environmental [13], health [10] and financial risks [14].

The multiple variables in time series can be categorized into two types, *i.e.,* the continuous variables and the discrete variables, as shown in Figure B.2. Continuous variables are represented by real numbers, such as latencies, CPU usage rates, and temperatures. Their values change continuously over continuous time, as illustrated in Figure B.2(a), which are called the *continuous dynamics*. Discrete variables are represented by natural numbers in encoding manners and are not amenable to mathematical calculations as adding or subtracting numerical values may not have meanings. These include the encoding for category, flag, and status information. Their values change discretely over continuous time, as illustrated in Figure B.2(b), which is called the *discrete dynamics*. Classic [3, 4, 15, 16] and deep learning-based [17–20] anomaly detection methods that aim to learn the normal patterns from unlabeled data achieve good results when applied to time series with regular discrete timestamps. However, all existing methods [14, 21] remain unable to learn both continuous and discrete dynamics for time series variables along the continuous time [22].

In addition, different variables have different physical and mathematical



(a) Continuous variables over continuous time.



(b) Discrete variables over continuous time.

Fig. B.2: Our motivations.

measurement units. For example, the latencies are measured in seconds, CPU usage rates are measured in percentages, temperatures are measured in Celsius degrees and the category information is measured in natural number encoding. Deep learning-based methods [8, 9, 14, 18, 20, 21, 23] cannot recognize the importance of each variable to detect the final anomaly at each timestamp in variable-specific level. Existing methods simply sum up the reconstruction errors or contrastive errors from all variables to obtain the final anomaly score and determine the anomalous timestamp, whereas the reconstruction errors and contrastive errors are based on different measurement units that should not be treated indiscriminately for time series anomaly detection.

Without learning continuous and discrete dynamics and recognizing different importance for the multiple variables, all existing methods have limited modeling ability on realistic meanings and fall short in detecting anomalies in real-world scenarios. However, the above problem has not been studied in depth nor analyzed theoretically due to its challenging nature, as outlined next.

Challenge 1: Continuous and discrete dynamics have completely different forms of behaviors along continuous time, and they may also have correlations with each other, which hinder existing models from learning the interactive continuous and discrete dynamics together from the different time series variables. Even though PAD [9] proposes to utilize the neural controlled differential equation (NCDE) [24] with interpolation methods to learn the continuous dynamics, they cannot learn the discrete dynamics. PAD and NCDE cannot model the correlations among different dynamics either, whereas different variables may have influences on each other as shown in Figure B.2 where variable 1 and variable 5 have correlations. In addition, PAD must also map the original time series of discrete timestamps into a continuous controlled path with interpolation methods, which brings additional processing steps and complexity. As a result, it is very challenging to learn the interactive continuous and discrete dynamics together from the multivariate time series effectively and efficiently.

Challenge 2: Existing methods cannot learn the importance units of different variables on the final anomaly scores without supervising signals. Anomaly detection often requires the unsupervised setting where labeled data is not feasible in real-world scenarios, because manual labeling is at high costs [3, 8, 21], and thus no supervising signals for existing methods to learn the importance units. Even though many recent works [3, 4, 19, 21, 23] propose to utilize Instance Normalization [25] to alleviate the difference of the measurement units, the reconstruction or contrastive errors from different variables are still in different semantic spaces and summing up all errors still ignore the importance of different variables. As a result, it is very challenging to learn the importance of different variables in the unsupervised setting.

To contend with these challenges, we propose **TAD-UP** to learn interactive continuous and discrete dynamics for multivariate **T**ime series **A**nomaly **D**etection via **U**nified **P**robabilistic modeling with novel neural co-dependent ordinary differential equations (co-ODEs). TAD-UP encompasses three main modules, as shown in Figure B.3: continuous co-ODE, discrete co-ODE, and multi-variable probabilistic modeling.

To address **Challenge 1**, we propose two co-dependent branches of neural ordinary differential equations (NODEs) to learn the interactive continuous and discrete dynamics together from different time series variables. Our first branch, *i.e.,* continuous co-ODE, learns the continuous dynamics. We propose the Compound Poisson Process for the second branch, *i.e.,* discrete co-ODE, to learn the discrete dynamics. Further, we propose the gate temporal convolution networks (TCNs) to learn correlations among discrete variables and continuous variables using co-dependent ordinary differential equations. Instead of NCDEs requiring complex interpolation methods, our co-ODEs learn the interactive continuous and discrete dynamics together effectively in the gate TCNs manner with efficient NODEs.

To address **Challenge 2**, we propose a joint probability distribution modeling and obtain the final anomaly scores from a unified probabilistic space. Specifically, we use the multi-variable Gaussian distribution to model the observation probabilities of continuous variables and the softmax distribution to model the observation probabilities of discrete variables. Then, we optimize our model with Maximum Likelihood Estimation (MLE) in the unified probabilistic space, instead of using reconstruction losses or contrastive losses for variables in different semantic spaces. Last, we detect anomalies using the joint probabilities that take the marginal probabilities of different variables into account to measure their importance on the final anomaly scores. The lower joint probability of all observations at a timestamp indicates a more likely anomaly. The lower marginal probability of one variable at a timestamp indicates that variable has more importance on the detected anomaly.

We conduct extensive experiments on nine real-world benchmark datasets from different domains to demonstrate that TAD-UP can achieve state-of-the-art performance, thereby offering a promising solution for real-world time series anomaly detection. We also conduct experiments on irregular time series where the discrete timestamps have irregular intervals, by dropping observations at random timestamps to create more challenging anomaly detection settings [9]. The ablation studies show the importance of our motivations and module designs.

We summarize our contributions in this work as follows.

- We are the first to learn both continuous and discrete dynamics for multivariate time series anomaly detection.
- We novelly propose two co-dependent branches of neural ordinary differential equations to learn interactive continuous and discrete dynamics effectively.
- We propose a joint probability distribution modeling and optimize our model with Maximum Likelihood Estimation, which helps obtain the anomaly scores that consider the importance of different variables.

- Experiments on nine benchmarks from different domains offer evidence that our method can achieve state-of-the-art performance even compared with large pre-training model DADA.

2 Preliminary

We first formalize the time series anomaly detection problem. Then, we introduce the background of neural differential equations.

2.1 Definitions

Multi-variable Time Series is represented as $\{\mathbf{C}, \mathbf{D}\}$, where $\mathbf{C} = \langle \mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_T \rangle \in \mathbb{R}^{N_C \times T}$ records the observations of the continuous variables by the real number $\mathbb{R}$, $\mathbf{D} = \langle \mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_T \rangle \in \mathbb{N}^{N_D \times T}$ records the observations of the discrete variables by the natural number $\mathbb{N}$, N_C and N_D are the number of continuous and discrete variables observed at each timestamp, respectively, T is the total number of timestamps, $\mathbf{c}_i \in \mathbb{R}^{N_C}$ and $\mathbf{d}_i \in \mathbb{N}^{N_D}$ are the observations at timestamp t_i and $i \in \{1, 2, \cdots, T\}$. Taking Figure B.1 as an example, we have $T = 5$, $N_C = 3$ and $N_D = 2$ and there are 3 continuous variables and 2 discrete variables observed at each timestamp. In the special case where $N_D = 0$, $\mathbf{C}$ is the time series as defined in previous works [9, 19, 23]. We use $\mathbf{C}^j \in \mathbb{R}^T$, $\mathbf{c}_i^j \in \mathbb{R}^1$, $\mathbf{D}^j \in \mathbb{N}^T$ and $\mathbf{d}_i^j \in \mathbb{N}^1$ to indicate the observations of the j-th variable with superscripts, respectively.

Time Series Anomaly Detection. Anomaly detection identifies which timestamp in the time series exists the anomaly, *i.e.*, a binary classification of the anomaly and normal for each timestamp. With sliding windows, anomaly detection takes as the input time series $\{\mathbf{C}, \mathbf{D}\}$, and outputs the binary vector $\mathbf{Y} = \langle y_1, y_2, \cdots, y_T \rangle$ where $y_i \in \{0, 1\}$ and $y_i = 1$ indicates an anomaly at timestamp t_i.

2.2 Neural differential equations

Chen *et al.* [26] first proposes a new paradigm of learning continuous dynamics and modeling continuous time with ordinary differential equation-based neural networks, which then have been widely studied in time series forecasting recently [27, 28]. Unlike traditional time series neural network models which only learn hidden states among discrete timestamps, *e.g.,* recurrent neural networks (RNNs) and TCNs, differential equations can learn the dynamics of the hidden state features $\mathbf{H}(t)$ by the integral along the continuous time with $\frac{d\mathbf{H}(t)}{dt}$. Neural ordinary differential equations (NODEs) directly use the integral to learn the hidden state feature $\mathbf{H}(t)$ at any continuous time t as follows:

$$\begin{aligned} \mathbf{H}(t) &= \mathbf{H}(0) + \int_0^t \frac{d\mathbf{H}(s)}{ds} ds, \\ \frac{d\mathbf{H}(t)}{dt} &= f\Big(\mathbf{H}(t); \mathbf{\Theta}\Big). \end{aligned} \tag{B.1}$$

$\mathbf{H}(0)$ is the hidden state feature that denotes the initial condition of NODEs where an embedding layer maps the input time series into the hidden state feature $\mathbf{H}(0)$, $f(\cdot; \mathbf{\Theta})$ denote the neural networks with parameters $\mathbf{\Theta}$ which estimates the continuous dynamical differentiation of $\mathbf{H}(t)$ along the continuous time t. Then, with the ODESolver $\Big(\cdot\Big)$ algorithm [28] we can obtain features $\mathbf{H}(t)$ at any continuous time t that could be used in downstream tasks:

$$\mathbf{H}_t = \text{ODESolver}\Big(\mathbf{H}(0), f, t; \mathbf{\Theta}\Big). \tag{B.2}$$

Recently, to improve the performance and robustness of the original NODEs, neural controlled differential equations (NCDEs) are proposed, which utilize the Riemann-Stieltjes integral [9, 24] to control the dynamics of hidden state feature $\mathbf{H}(t)$ at any continuous time t with the time series observations $\mathbf{C}$ as follows:

$$\begin{aligned} \mathbf{H}(t) &= \mathbf{H}(0) + \int_0^t \frac{d\mathbf{H}(s)}{ds} d\mathbf{C}(s) \\ &= \mathbf{H}(0) + \int_0^t f\Big(\mathbf{H}(s); \mathbf{\Theta}\Big) \frac{d\mathbf{C}(s)}{ds} ds. \end{aligned} \tag{B.3}$$

$\mathbf{C}(t)$ is the continuous time series path across continuous time, which is interpolated from the original time series observations $\mathbf{C} = \langle \mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_T \rangle \in \mathbb{R}^{N_C \times T}$ recorded at discrete timestamps with the natural cubic spline interpolation pre-processing [9, 24]. $\frac{d\mathbf{C}(s)}{ds}$ is the differential coefficient in the time series path.

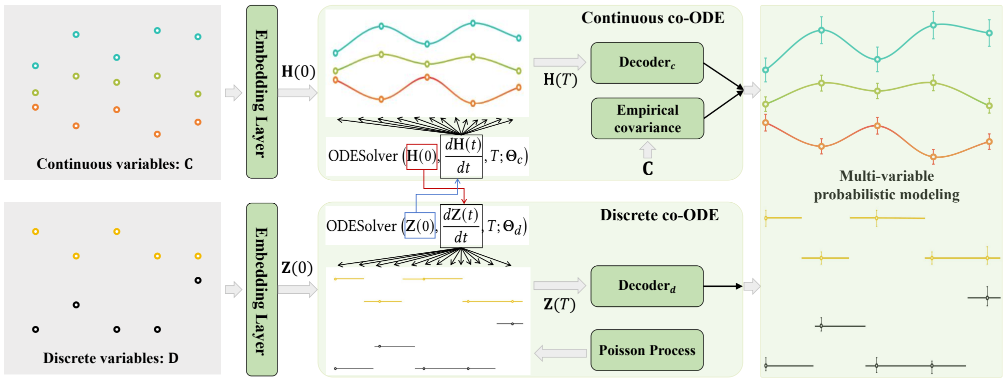


Fig. B.3: The overall architecture.

The difference between NODEs and NCDEs is the continuous time series path $C(t)$, whose differential coefficient will control the dynamics. However, NCDEs cannot model long input time series effectively and efficiently, as NCDEs have a recurrent architecture as in RNNs and the natural cubic spline interpolation pre-processing will bring additional space and time complexity.

On the other hand, both NODEs and NCDEs cannot learn the discrete dynamics from the discrete variables, and they cannot model the interactions among different dynamics either.

In contrast, as shown in the following sections in this paper, we propose neural co-dependent ordinary differential equations, which are free from interpolation pre-processing. We further propose the Compound Poisson Process to learn the discrete dynamics and propose the dependent latent features to help learn the influence among dynamics of different variables.

3 Methodology

3.1 Overall architecture

We present the overall architecture in Figure B.3, which consists of the embedding layers, continuous co-ODE, discrete co-ODE, and multi-variable probabilistic modeling.

The embedding layers take as the input time series $\{\mathbf{C}, \mathbf{D}\}$, first apply Instance Normalization [25] on the observations of continuous variables, *i.e.,* each $\mathbf{C}^j$ being normalized, to learn stable features [29], and then map the observations $\mathbf{c}_i$ and $\mathbf{d}_i$ at each timestamp t_i into d-dimensional vectors $\mathbf{h}_i \in \mathbb{R}^d$ and $\mathbf{z}_i \in \mathbb{R}^d$ to extract the dense features for discrete timestamps. Note that the embedding layer for continuous variables is the fully connected feed-forward layer (FC):

$$\begin{aligned} &\mathbf{h}_i = \text{FC}(\mathbf{c}_i),\ for\ i \in \{1, 2, \cdots, T\}, \\ &\mathbf{H}(0) = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T), \end{aligned} \tag{B.4}$$

where $\mathbf{H}(0) \in \mathbb{R}^{d \times T}$ denotes the hidden state feature of the initial condition for our continuous co-ODE. The layer for discrete variables is the lookup embedding layer:

$$\begin{aligned} &\mathbf{d}^j = \text{one-hot}(\mathbf{d}^j),\ for\ j \in \{1, 2, \cdots, N_D\}, \\ &\mathbf{z}_i = \text{Embedding}(\mathbf{d}_i),\ for\ i \in \{1, 2, \cdots, T\}, \\ &\mathbf{Z}(0) = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T), \end{aligned} \tag{B.5}$$

where $\mathbf{Z}(0) \in \mathbb{R}^{d \times T}$ denotes the hidden state feature of the initial condition for our discrete co-ODE.

The continuous co-ODE takes feature $\mathbf{H}(0)$ as the initial condition, uses gated TCNs architecture together with $\mathbf{Z}(0)$ to learn the influence flows from the discrete variables to the continuous variables, and output final hidden state feature $\mathbf{H}(T)$ which learns co-dependent continuous dynamics.

The discrete co-ODE takes feature $\mathbf{Z}(0)$ as the initial condition, uses gated TCNs architecture together with $\mathbf{H}(0)$ to learn the influence flows from the continuous variables to the discrete variables, and uses the Compound Poisson Process to output final hidden state feature $\mathbf{Z}(T)$ which learns co-dependent discrete dynamics.

The multi-variable probabilistic modeling takes as the inputs $\mathbf{H}(T)$ and $\mathbf{Z}(T)$, outputs the joint probabilities of all observations at the timestamp T where the multi-variable Gaussian distribution models the observation probabilities of continuous variables and the softmax distribution models the observation probabilities of discrete variables. Last, we optimize our model with Maximum Likelihood Estimation in the probabilistic space, and the lower joint probabilities of all observations indicate a more likely anomaly at the timestamp T.

3.2 The continuous co-ODE

We introduce the composition of our continuous co-ODE which can learn fine-grained and continuous temporal dynamics. Given the shortcomings of RNNs, like high time-complexity and gradient explosion [27], we choose temporal convolution networks as the function $f(\cdot;\mathbf{\Theta})$ to estimate the continuous dynamical differentiation of $\mathbf{H}(t)$ along the continuous time t:

$$\begin{aligned} \mathbf{H}(t) &= \mathbf{H}(0) + \int_0^t \frac{d\mathbf{H}(s)}{ds} ds, \\ \frac{d\mathbf{H}(t)}{dt} &= \tanh\Big(\mathrm{TCN}\big(\mathbf{H}(t); r, k\big)\Big), \end{aligned} \tag{B.6}$$

where $\mathrm{TCN}(\cdot; r, k)$ is the Conv1d temporal convolution which are performed along the last dimension of $\mathbf{H}(t)$, *i.e.,* the time dimension. $\tanh(\cdot)$ is the hyperbolic tangent activation function. r and k are the dilation factor and the convolution kernel, respectively. We omit r and k in the following paper.

In order to learn the influence among different variables, we propose a gated TCN architecture as the attention mechanism to control the amount of information flows:

$$\begin{aligned} \frac{d\mathbf{H}(t)}{dt} =& \tanh\Big(\mathrm{TCN}_1\big(\mathbf{H}(t)\big)\Big) \odot \sigma\Big(\mathrm{TCN}_{\mathrm{gate}_1}\big(\mathbf{H}(t)\big)\Big) \\ & \odot \sigma\Big(\mathrm{TCN}_{\mathrm{gate}_2}\big(\mathbf{Z}(0)\big)\Big), \end{aligned} \tag{B.7}$$

where $\odot$ denotes the element-wise Hadamard product, $\mathrm{TCN}_{\mathrm{gate}_1}$ and $\mathrm{TCN}_{\mathrm{gate}_2}$ are the gate convolutions used for the continuous variables and discrete variables, respectively, and $\sigma(\cdot)$ is the sigmoid activation function to control the amount of information flows among different variables for the continuous dynamics branch. Then, with the ODESolver$\big(\cdot\big)$ [28] we can obtain the final hidden state feature $\mathbf{H}(T)$ which learns co-dependent continuous dynamics:

$$\mathbf{H}_T = \mathrm{ODESolver}\Big(\mathbf{H}(0), \frac{d\mathbf{H}(t)}{dt}, T; \mathbf{\Theta}_c\Big). \tag{B.8}$$

3.3 The discrete co-ODE

The NODEs and NCDEs used in the previous work PAD [9], as shown in Section 2.2, cannot learn the discrete dynamics where the hidden state features could jump along the continuous time. Thus, we propose to utilize the Compound Poisson Process [28] to learn the discrete dynamics. The Poisson Process is a time point generative model where the output contains a sequence of discrete eventual time points $\mathcal{H} = \{\tau_{t_i}\}$. The τ_{t_i} denotes there is a discrete eventual time point at timestamp t_i and no eventual time points between timestamps t_{i-1} and t_i.

The Poisson Process can be formally defined by a counting function $\mathbf{N}(t)$ along the continuous time. $\mathbf{N}(t)$ records the number of eventual time point before time t, which is described as follows:

$$\mathbf{N}(t) = \sum_{\tau_{t_i} \in \mathcal{H}} He(t - t_i), \text{ where } He(t) = \begin{cases} 0 & t \leq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{B.9}$$

The $He(\cdot)$ is the Heaviside step function. The probability of the next eventual time point followed after τ_{t_i} usually depends on the previous contexts, known as the rate or intensity of the Poisson Process. Such temporal dependencies can be described by a conditional intensity function $\lambda(t)$. Let $\mathcal{H}_t$ denote the previous contexts of eventual time points up to but not including time t. Then $\lambda(t)$ defines the probability of observing the next eventual time point conditioned on the history:

$$\mathbb{P}\left\{\text{eventual time point in } [t, t+dt) \mid \mathcal{H}_t\right\} = \lambda\left(t\right) \cdot dt \tag{B.10}$$

With this Poisson Process, we propose our discrete co-ODE with Compound Poisson Process as defined by:

$$\begin{aligned} \lambda(t) &= \mathrm{FC}_\lambda\Big(\mathbf{Z}(t)\Big) \\ \mathbf{Z}(t) &= \mathbf{Z}(0) + \int_0^t \frac{d\mathbf{Z}(s)}{ds} d\mathbf{N}(s), \\ \frac{d\mathbf{Z}(t)}{dt} &= \tanh\Big(\mathrm{TCN}_2\Big(\mathbf{Z}(t)\Big)\Big) \odot \sigma\Big(\mathrm{TCN}_{\mathrm{gate}_3}\Big(\mathbf{Z}(t)\Big)\Big) \\ &\qquad \odot \sigma\Big(\mathrm{TCN}_{\mathrm{gate}_4}\Big(\mathbf{H}(0)\Big)\Big), \end{aligned} \tag{B.11}$$

where FC_λ denotes the fully connected feed-forward layer that learns $\lambda(t)$ from the contexts $\mathbf{Z}(t)$ to control the conditional intensity of the next eventual time point for different discrete variables, $\text{TCN}_{\text{gate}_3}$ and $\text{TCN}_{\text{gate}_4}$ are two another gate convolutions used for the continuous variables and discrete variables in the discrete co-ODE, respectively, to control the amount of information flows among different variables for the discrete dynamics branch. Then, with the ODESolver $\big(\cdot\big)$ [28] we can obtain the final hidden state feature $\mathbf{Z}(T)$ which learns co-dependent discrete dynamics:

$$\mathbf{Z}_T = \text{ODESolver}\Big(\mathbf{Z}(0), \frac{d\mathbf{Z}(t)}{dt}, T; \mathbf{\Theta}_d\Big). \tag{B.12}$$

3.4 Multi-variable probabilistic modeling

After we obtain the features $\mathbf{H}(T)$ and $\mathbf{Z}(T)$ with different dynamics, we use decoders to output the joint probabilities of all observations at the timestamp T.

Specifically, for the continuous variables, the decoder uses fully connected feed-forward layers and outputs the estimated multi-variable Gaussian mixture distribution $N(\hat{\mu}_T, \hat{\Sigma}_T)$:

$$\hat{\mu}_T, \hat{\sigma}_T = \text{Decoder}_c\Big(\mathbf{H}(T)\Big) \tag{B.13}$$

where $\hat{\mu}_T = (\hat{\mu}_T^1, \hat{\mu}_T^2, \cdots, \hat{\mu}_T^{N_C}) \in \mathbb{R}^{N_C}$ denotes the estimated mean values and $\hat{\sigma}_T = (\hat{\sigma}_T^1, \hat{\sigma}_T^2, \cdots, \hat{\sigma}_T^{N_C}) \in \mathbb{R}^{N_C}$ denotes the estimated variance values as the diagonal elements of the covariance matrix $\hat{\Sigma}_T \in \mathbb{R}^{N_C \times N_C}$ for the N_C continuous variables, respectively. Then, we calculate the empirical covariance between the j_1-th and j_2-th continuous variables to estimate $\hat{\Sigma}_T(j_1, j_2)$, which can help to capture the more accurate marginal probability distribution, as follows:

$$\hat{\Sigma}_T(j_1, j_2) = \hat{\Sigma}_T(j_2, j_1) = \frac{1}{T-1}\sum_{i=1}^{T}(\mathbf{c}_i^{j_1} - \overline{\mathbf{c}}^{j_1})(\mathbf{c}_i^{j_2} - \overline{\mathbf{c}}^{j_2}), \tag{B.14}$$

where $\overline{\mathbf{c}}^{j_1}$ and $\overline{\mathbf{c}}^{j_1}$ are the mean values of the j_1-th and j_2-th continuous variables, respectively. Thus, our loss function based on the Maximum

Likelihood Estimation in the probabilistic space for the continuous variables is as follows:

$$\mathcal{L}_c = -ln\Big[\frac{1}{\sqrt{2\pi\big|\hat{\Sigma}_T\big|}}exp\Big(-\frac{1}{2}(\mathbf{c}_T - \hat{\mu}_T)^\top\hat{\Sigma}_T^{-1}(\mathbf{c}_T - \hat{\mu}_T)\Big)\Big] \tag{B.15}$$

For the discrete variables, the decoder uses fully connected feed-forward layers and outputs the softmax distributions $\hat{\mathbf{d}}_T$:

$$\begin{aligned} &\hat{logits}_T^1, \hat{logits}_T^2, \cdots, \hat{logit}_T^{N_D} = \text{Decoder}_d\Big(\mathbf{Z}(T)\Big), \\ &\hat{\mathbf{d}}_T^j = \text{softmax}(\hat{logit}_T^j),\ for\ j \in \{1, 2, \cdots, N_D\}, \end{aligned} \tag{B.16}$$

where $\hat{\mathbf{d}}_T^j$ denotes the probability distribution for the j-th discrete variable in the one-hot encoding manner. Thus, our loss function based on the Maximum Likelihood Estimation in the probabilistic space for the discrete variables is as follows:

$$\mathcal{L}_d = \text{CE}(\hat{\mathbf{d}}_T, \mathbf{d}_T) \tag{B.17}$$

Last, our final loss functions are:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_d, \tag{B.18}$$

where $\mathcal{L}$ will optimize the model on the embedding layers, $\mathcal{L}_c$ will optimize the model on the continuous co-ODE and Decoder$_c$, and $\mathcal{L}_d$ will optimize the model on the discrete co-ODE and Decoder$_d$.

During the inference stage, the lower joint probabilities of the observations of all variables at one timestamp T indicate a more likely anomaly.

4 Experiments

In this section, we conduct extensive experiments to empirically evaluate the unsupervised time series anomaly detection on nine real-world benchmark datasets, to justify the effectiveness of our model and compare TAD-UP with state-of-the-art baselines. We present the experimental results in terms of overall comparison, ablation studies, parameter sensitivity, and irregular time series, and analyze our model in subsections.

Table B.1: Experiments on five real-world datasets, presented in percentages.

Dataset	**SMD**			**MSL**			**SMAP**			**SWaT**			**PSM**		
Metric	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**	**Aff-P**	**Aff-R**	**Aff-F1**
OCSVM	66.98	82.03	73.75	50.26	99.86	66.87	41.05	69.37	51.58	56.80	98.72	72.11	57.51	58.11	57.81
PCA	64.92	86.06	74.01	52.69	98.33	68.61	50.62	98.48	66.87	62.32	82.96	71.18	77.44	63.68	69.89
HBOS	60.34	64.11	62.17	59.25	83.32	69.25	41.54	66.17	51.04	54.49	91.35	68.26	78.45	29.82	43.21
LOF	57.69	99.10	72.92	49.89	72.18	59.00	47.92	82.86	60.72	53.20	96.73	68.65	53.90	99.91	70.02
IForest	71.94	94.27	81.61	53.87	94.58	68.65	41.12	68.91	51.51	53.03	99.95	69.30	69.66	88.79	78.07
LODA	66.09	84.37	74.12	57.79	95.65	72.05	51.51	100.00	68.00	56.30	70.34	62.54	62.22	87.38	72.69
DAGMM	63.57	70.83	67.00	54.07	92.11	68.14	50.75	96.38	66.49	59.42	92.36	72.32	68.22	70.50	69.34
A.T.	54.08	97.07	66.42	51.04	95.36	66.49	56.91	96.69	71.65	53.63	98.27	69.39	54.26	82.18	65.37
DCdetector	50.93	95.57	66.45	55.94	95.53	70.56	53.12	98.37	68.99	53.25	98.12	69.03	54.72	86.36	66.99
PAD	59.54	67.66	63.71	56.33	82.21	68.15	41.67	64.52	53.94	54.73	92.35	68.06	68.45	57.72	59.21
AE	69.22	98.48	81.30	55.75	96.66	70.72	39.42	70.31	50.52	54.92	98.20	70.45	60.67	98.24	75.01
LSTM	60.12	84.77	70.35	58.82	14.68	23.49	55.25	27.70	36.90	49.99	82.11	62.15	57.06	95.92	71.55
BeatGAN	74.11	81.64	77.69	55.74	98.94	71.30	54.04	98.30	69.74	61.89	83.46	71.08	58.81	99.08	73.81
Omni	79.09	75.77	77.40	51.23	99.40	67.61	52.74	98.51	68.70	62.76	82.82	71.41	69.20	80.79	74.55
CAE-Ensemble	73.05	83.61	77.97	54.99	93.93	69.37	62.32	64.72	63.50	62.10	82.90	71.01	73.17	73.66	73.42
D3R	64.87	97.93	78.02	66.85	90.83	77.02	61.76	92.55	74.09	60.14	97.57	74.39	73.32	88.71	80.29
GPT4TS	73.33	95.97	83.13	64.86	95.43	77.23	63.52	90.56	74.67	56.84	91.46	70.11	73.61	91.13	81.44
ModernTCN	74.07	94.79	83.16	65.94	93.00	77.17	69.50	65.45	67.41	59.14	89.22	71.13	73.47	86.83	79.59
MEMTO	49.69	98.05	65.96	52.73	97.34	68.40	50.12	99.10	66.57	56.47	98.02	71.66	52.69	83.94	64.74
SensitiveHUE	60.34	90.13	72.29	55.92	98.95	71.46	53.63	98.37	69.42	58.91	91.71	71.74	56.15	98.75	71.59
DADA	76.50	94.54	84.57	68.70	91.51	78.48	65.85	88.25	75.42	61.59	94.59	74.60	74.31	92.11	82.26
TAD-UP	77.11	94.55	**84.95**	68.54	92.67	**78.80**	66.37	89.12	**76.08**	62.48	94.86	**75.35**	73.75	93.13	**82.32**

4.1 Experimental settings

Datasets.

Following existing works [9, 19], we validate the performance of our method with widely used real-world datasets:

- SMD [6] (Server Machine Dataset) is collected from a computing cluster, where different server machines stack accessed traces of resource utilization with 38 dimensions.
- PSM [5] (Pooled Server Metrics) is a 25-dimension dataset collected by eBay that shows the performance of multiple application servers.
- MSL [30] (Mars Science Laboratory Rover) is from the spacecraft monitoring systems and collected by NASA. It shows the health check-up data of the Mars rover sensors.
- SMAP [30] (Soil Moisture Active Passive) is a 25-dimension dataset collected by NASA, which contains soil samples and telemetry data.
- SWaT [31] (Secure Water Treatment) is collected from sensors of the critical infrastructure systems under continuous operations for secure water

treatment.

- NeurIPS-TS [7] includes CICIDS, Creditcard, GECCO, and SWAN from different domains. CICIDS is collected from multiple web servers. GECCO is a 9-dimension dataset collected from the cyber-physical systems showing the drinking water quality. SWAN is extracted from solar photospheric vector magnetograms in the Spaceweather. Creditcard is from the finance scenario where the anomaly rate is only 0.172%.

More details of the datasets can be seen in Appendix A.1.

Baselines.

We compare our TAD-UP model with 21 baselines of different categories for comprehensive evaluations:

- Clustering-based: PCA [3], OCSVM [17].
- Density estimation-based: HBOS [32] DAGMM [20], IForest [16], LODA [33] and LOF [15].
- Contrastive-based: DCdetector [34], AnomalyTransformer (A.T.) [31] and PAD [9].
- Autoregression and reconstruction-based: AE [18], LSTM [30], OmniAnomaly (Omni) [6], BeatGAN [35], CAE-Ensemble [19], D3R [36], GPT-4TS [37], ModernTCN [38], MEMTO [39], SensitiveHUE [40] and DADA [23].

We introduce the details of baselines in Section 5, and note that DADA is a large time series anomaly detection model pre-trained with multi-domain open data. More details of settings are in Appendix A.2.

Evaluation metrics.

Previous methods [6, 9, 31, 41] use point adjustments (PA) with test labels to adjust their outputs. However, recent works [8, 14, 23, 36, 42–44] have demonstrated that PA can lead to faulty performance evaluations, and it is known that PA can result in state-of-the-art performance even with random guess [36, 42]. Thus, we use the Affiliation-based Precision (Aff-P), Recall (Aff-R), F1-score (Aff-F1) [42] following recent works [23, 34, 36], where

Table B.2: Experiments on four real-world datasets with more metrics, presented in percentages.

Dataset	**CICIDS**		**Creditcard**		**GECCO**		**SWAN**	
Metric	**Aff-F1**	**AUC-ROC**	**Aff-F1**	**AUC-ROC**	**Aff-F1**	**AUC-ROC**	**Aff-F1**	**AUC-ROC**
A.T.	34.71	49.00	65.14	52.55	64.27	51.60	33.67	44.74
DCdetector	40.02	53.95	58.28	42.36	66.18	45.38	14.42	43.48
ModernTCN	51.74	65.33	73.80	95.55	90.18	95.20	46.45	52.63
GPT4TS	54.00	67.91	72.88	95.58	88.11	90.21	47.27	51.93
DADA	73.49	69.33	**75.12**	**95.73**	90.20	93.44	71.93	53.29
TAD-UP	**75.88**	**72.01**	75.07	95.71	**90.21**	**95.37**	**71.95**	**57.98**

Table B.3: Ablation study.

Dataset	SMD			MSL			SWaT			CICIDS		
Metric	Aff-P	Aff-R	Aff-F1	Aff-P	Aff-R	Aff-F1	Aff-P	Aff-R	Aff-F1	Aff-P	Aff-R	Aff-F1
w/o continuous co-ODE	74.96	93.20	83.09	67.98	91.34	77.95	61.94	94.67	74.88	64.55	90.26	75.27
w/o discrete co-ODE	75.63	92.47	83.21	67.10	91.49	77.41	61.44	93.75	74.23	61.86	90.13	73.37
w/o probabilistic mode	74.85	92.93	82.92	60.01	94.66	73.45	62.37	94.91	75.27	64.37	90.52	75.24
TAD-UP	77.11	94.55	**84.95**	68.54	92.67	**78.80**	62.48	94.86	**75.35**	64.89	91.33	**75.88**

larger values indicate higher model performance. We also use the ROC-AUC [44] metric following DADA [23] which enables evaluation without choosing the threshold.

4.2 Overall comparison and analysis

Table B.1 and Table B.2 show the overall anomaly prediction performance of different models on all datasets. We randomly repeat our method 3 times and report the average result, where the best F1-score and AUC-ROC are highlighted in bold and outperform the underlined second-best ones. As DADA is pre-trained with very large multi-domain time series data, we also underline the second-best ones other than DADA.

Key observations are followed. First, our TAD-UP can achieve state-of-the-art results on all datasets. It demonstrates that TAD-UP is able to learn both continuous and discrete dynamics to improve time series anomaly detection.

Second, we observe that the contrastive-based methods are unsatisfactory on most datasets. This is because their data augmentation borrowed from CV or NLP cannot fit well for time series data.

Third, TAD-UP achieves better accuracy compared to other reconstruction based methods, as they only add up all errors from different variables regardless of their measurement units. They cannot explicitly learn the

Table B.4: PARAMETER SENSITIVITY

Dataset	SWaT				PSM				CICIDS			
window size	Aff-P	Aff-R	Aff-F1	AUC-ROC	Aff-P	Aff-R	Aff-F1	AUC-ROC	Aff-P	Aff-R	Aff-F1	AUC-ROC
8	62.03	94.15	74.79	82.33	73.45	92.41	81.84	60.67	64.41	91.00	75.43	71.85
16	62.19	94.36	74.97	82.72	73.75	93.13	82.32	61.33	64.89	91.33	75.88	72.01
32	62.48	94.86	75.35	83.01	73.44	92.97	82.06	60.70	64.22	91.06	75.32	70.03
64	62.36	94.47	75.13	82.93	72.58	91.76	81.05	60.19	64.15	90.53	75.43	70.57

Table B.5: Anomaly detection for irregular time series

Dataset	SWaT											
Dropping ratio	0%				20%				40%			
Metric	Aff-P	Aff-R	Aff-F1	AUC-ROC	Aff-P	Aff-R	Aff-F1	AUC-ROC	Aff-P	Aff-	Aff-F1	AUC-ROC
PAD	54.73	92.35	68.06	63.86	52.71	92.13	67.05	63.91	49.73	90.64	64.22	56.84
TAD-UP	62.48	94.86	**75.35**	**83.01**	61.82	93.06	**74.28**	**80.92**	59.98	91.97	**72.60**	**72.90**

importance units of different variables to obtain the final anomaly scores.

Last, TAD-UP outperforms DADA, only except for the Creditcard dataset which may require a lot of prior knowledge that DADA could learn from pre-training on multi-domain data.

4.3 Ablation studies

We conduct ablation studies to validate the effectiveness of our contributed components. Specifically, we compare TAD-UP with the following variants: (1) w/o continuous co-ODE: This variant directly learns features from the continuous variables without co-ODE. (2) w/o discrete co-ODE: This variant directly learns features from the discrete variables without co-ODE. (3) w/o probabilistic model: This variant directly uses the reconstruction loss.

Table B.3 shows the results of different variants. From Table B.3 we observe that: (1) Our continuous co-ODE and discrete co-ODE can learn the interactive dynamics effectively and contribute to more accurate results. (1) Our joint probabilistic modeling helps obtain the anomaly scores that consider the importance of different variables with Maximum Likelihood Estimation in the unified probabilistic space and thus can have more accurate results.

4.4 Paremeter sensitivity

We evaluate the impact of the sliding window sizes in Table B.4. We can see that TAD-UP is relatively stable to the sliding window sizes. We can also see that time series from different domains may require different sliding window sizes to learn the dynamics of optimal time lengths to get more accurate results.

4.5 Irregular time series

PAD [9] uses interpolation methods and NCDE to support anomaly detection on time series of irregular timestamps. We also show the experimental results on the irregular time series by dropping the observations at random timestamps following PAD. From Table B.5, we can see that TAD-UP still performs well for irregular time series with our co-ODEs to learn dynamics along continuous time.

5 Related Works

We review the relevant works on time series anomaly detection, neural differential equations, and multi-variable strategy.

5.1 Time series anomaly detection

There has been a lot of research on time series anomaly detection. Early works [14] utilize supervised classification-based methods to detect anomalies. Recent methods mainly focus on unsupervised settings based on the one-class classification theory [17]. They can be further categorized into four categories [3], including clustering-based methods [4], density estimation-based methods [15], autoregression and reconstruction-based methods [18] methods, and contrastive-based methods [34].

The clustering-based methods, such as K-means [21], PCA [3], and OCSVM [17], aim to cluster normal samples within a center. THOC [41] uses a hierarchical RNN to learn dynamic clustering centers for time series data.

The density estimation-based anomaly detection methods, such as LOF [15] and HBOS [32], assume abnormal samples lie in a low-density region. IForest [16] uses the tree model and DAGMM [20] and LODA [33] use the deep learning to measure the density.

Contrastive learning learns useful representations by contrasting positive pairs against negative pairs, such as DCdetector [34], where two view representations of a normal timestamp should be similar, and contrastive errors are used to detect anomalies. AnomalyTransformer [31] combines the reconstruction errors and contrastive errors together to detect anomalies.

LSTM [30] proposes forecasting-based approaches that forecast values and based on the forecasting errors detect whether the time points are anomalies. Autoencoder [45] is a classic architecture used in reconstruction, such as AE [18], GPT4TS [37], MEMTO [39], ModernTCN [38] and SensitiveHUE [40]. Omni [6] and InterFusion [46] utilize the Variational networks. BeatGAN [35] introduces a discriminator to help generate normal data. CAE-Ensemble [19] proposes to ensemble different AEs for reconstruction. DADA [23] pre-trains a time series anomaly detection model with large open data. The diffusion-based network is a generative model that generates data with denoising steps. ImDiffusion [47] and D3R [36] utilize diffusion-based networks to generate normal time series data and detect anomalies based on reconstruction errors.

5.2 Multi-variable strategy

There are mainly two strategies that consider multi-variable time series. The Channel-Independent (CI) strategy ignores the correlations among different variables, such as DCdetector [34] and DADA [23]. These CI-based methods use the same model on different variables in parallel. The Channel-Dependent (CD) strategy model the correlations among different variables, such as ModernTCN [38] and CAE-Ensemble [19]. These CD-based methods use the attention mechanism or graph neural networks to capture the correlations. However, they cannot learn the importance of different variables on the final anomalies without supervising signals.

5.3 Neural differential equations

Chen *et al.* [26] first proposes to learn continuous dynamics along the continuous time with NODEs, which then have been widely studied in

Table B.6: Details of benchmark datasets.

Dataset	Domain	N_C	N_D	#Training (Unlabeled)	#Validation (Unlabeled)	#Test (Labeled)	Anomaly Rate(%)
SMD	server machine	33	5	566,724	141,681	708,420	4.16
MSL	NASA space sensor	1	54	46,653	11,663	73,729	10.72
SMAP	NASA space sensor	1	24	108,146	27,036	427,617	13.13
SWaT	water treatment	25	26	396,000	99,000	449,919	12.14
SWAN	space solar weather	33	5	48,000	12,000	60,000	32.6
CICIDS	web server	58	14	68,092	17,023	85,116	1.28
PSM	application server	25	0	105,885	26,398	87,841	27.8
GECCO	water quality	9	0	55,408	13,852	69,261	1.1
Creditcard	finance	29	0	159,491	39,873	85,443	0.172

time series forecasting recently, such as traffic forecasting [27] and climate forecasting [28]. Recently, NCDEs have been proposed, which utilize the Riemann-Stieltjes integral [24] to control the continuous dynamics with the time series observations. PAD [9] proposes to utilize NCDEs and interpolation methods for irregular time series, using contrastive learning with data augmentation to detect anomalies. However, they cannot learn the interactive continuous and discrete dynamics.

6 Conclusion

We present TAD-UP, to learn continuous and discrete dynamics for Time series Anomaly Detection via Unified Probabilistic modeling. First, we propose two co-dependent branches of neural ordinary differential equations with the Compound Poisson Process to learn the continuous and discrete dynamics for different time series variables. Second, we propose a unified joint probability distribution modeling, and we optimize our model with the Maximum Likelihood Estimation instead of using reconstruction losses or contrastive losses. Last, we detect anomalies using the joint probabilities which take the importance of different variables into account by marginal distributions. Extensive experiments on nine real-world datasets offer evidence that TAD-UP is capable of state-of-the-art performance. In future works, it is worth improving TAD-UP with pre-training on the large-scale multi-domain time series data.

A Experimental details

A.1 Datasets

The datasets used in this work are representative open benchmarks for multi-variable time series anomaly detection evaluation. More statistical information and details can be seen in Table B.6, where N_C and N_C are the number of continuous variables and discrete variables observed at each timestamp. We follow the same training-validation-test splits as in the previous works [14, 23, 31] shown in each column.

A.2 Settings

Our codes will be available upon acceptance. We employ the official open-source implementations of baseline methods and have carefully tuned their hyper-parameters based on the recommendations from the original papers. The experiments are conducted on an Ubuntu 18.04.5 LTS system server, with Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz and NVIDIA Quadro RTX 8000 GPU. We use Adam optimizer with the default parameter and the learning rate is default 0.0001. The sliding window size is tuned from {8, 16, 32, 64}. The batch size is tuned from {32, 64}. We use three kernel sizes {2, 3, 5} for TCNs. Following the existing work [14], we use the same strategy to choose the threshold.

References

[1] R. Cirstea, C. Guo, and B. Yang, “Graph attention recurrent neural networks for correlated time series forecasting - full version,” *CoRR*, vol. abs/2103.10760, 2021.

[2] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen, “LightTS: Lightweight time series classification with adaptive ensemble distillation,” *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 171:1–171:27, 2023.

[3] Q. Liu, P. Boniol, T. Palpanas, and J. Paparrizos, “Time-series anomaly detection: Overview and new trends,” *Proc. VLDB Endow.*, vol. 17, no. 12, pp. 4229–4232, 2024.

[4] A. Zhang, S. Deng, D. Cui, Y. Yuan, and G. Wang, “An experimental evaluation of anomaly detection in time series,” in *Proc. VLDB Endow.*, vol. 17, no. 3, 2023, pp. 483–496.

[5] A. Abdulaal, Z. Liu, and T. Lancewicki, “Practical approach to asynchronous multivariate time series anomaly detection and localization,” in *SIGKDD*, 2021, pp. 2485–2494.

[6] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, “Robust anomaly detection for multivariate time series through stochastic recurrent neural network,” in *SIGKDD*, 2019, pp. 2828–2837.

[7] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, and X. Hu, “Revisiting time series outlier detection: Definitions and benchmarks,” in *NeurIPS*, 2021.

[8] R. Ghorbani, M. J. T. Reinders, and D. M. J. Tax, “PATE: proximity-aware time series anomaly evaluation,” in *SIGKDD*, 2024, pp. 872–883.

[9] S. Y. Jhin, J. Lee, and N. Park, “Precursor-of-anomaly detection for irregular time series,” in *SIGKDD*, 2023, pp. 917–929.

[10] V. Cerqueira, L. Torgo, and C. Soares, “Early anomaly detection in time series: a hierarchical approach for predicting critical health episodes,” *Mach. Learn.*, vol. 112, no. 11, pp. 4409–4430, 2023.

[11] H. Hojjati, M. Sadeghi, and N. Armanfard, “Multivariate time-series anomaly detection with temporal self-supervision and graphs: Application to vehicle failure prediction,” in *PKDD*, vol. 14175, 2023.

[12] Y. Ang, Q. Huang, A. Tung, and Z. Huang, “A stitch in time saves nine: Enabling early anomaly detection with correlation analysis,” in *ICDE*, 2023, pp. 132–145.

[13] A. Nina, “Analysis of VLF signal noise changes in the time domain and excitations/attenuations of short-period waves in the frequency

domain as potential earthquake precursors," *Remote. Sens.*, vol. 16, no. 2, 2024.

[14] Y. Zhao, L. Deng, X. Chen, C. Guo, B. Yang, T. Kieu, F. Huang, T. B. Pedersen, K. Zheng, and C. S. Jensen, "A comparative study on unsupervised anomaly detection for time series: Experiments and analysis," *CoRR*, vol. abs/2209.04635, 2022.

[15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *SIGMOD*, 2000.

[16] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *TKDE*, vol. 33, no. 4, pp. 1479–1489, 2019.

[17] L. Ruff, R. A. Vandermeulen, N. G"ornitz, L. Deecke, S. A. Siddiqui, A. Binder, E. M"uller, and M. Kloft, "Deep one-class classification," in *ICML*, vol. 80, 2018, pp. 4393–4402.

[18] Y. Fang, J. Xie, Y. Zhao, L. Chen, Y. Gao, and K. Zheng, "Temporal-frequency masked autoencoders for time series anomaly detection," in *ICDE*, 2024, pp. 1228–1241.

[19] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen, "Unsupervised time series outlier detection with diversity-driven convolutional ensembles," in *Proc. VLDB Endow.*, vol. 15, no. 3, 2022, pp. 611–623.

[20] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.

[21] S. Schmidl, F. Naumann, and T. Papenbrock, "Autotsad: Unsupervised holistic anomaly detection for time series data," in *Proc. VLDB Endow.*, vol. 17, no. 4, 2024, pp. 766–779.

[22] X. He, Y. Li, J. Tan, B. Wu, and F. Li, "Oneshotstl: One-shot seasonal-trend decomposition for online time series anomaly detection and forecasting," in *Proc. VLDB Endow.*, vol. 16, no. 5, 2023, pp. 700–712.

[23] Q. Shentu, B. Li, K. Zhao, Y. Shu, Z. Rao, L. Pan, B. Yang, and C. Guo, "Towards a general time series anomaly detector with adaptive bottlenecks and dual adversarial decoders," in *ICLR*, 2025.

[24] P. Kidger, J. Morrill, J. Foster, and T. J. Lyons, "Neural controlled differential equations for irregular time series," in *NeurIPS*, 2020.

[25] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv/1607.08022*, 2016.

[26] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *NeurIPS*, 2018.

[27] M. Jin, Y. Zheng, Y. Li, S. Chen, B. Yang, and S. Pan, "Multivariate time series forecasting with dynamic graph neural odes," *Trans. Knowl. Data Eng.*, pp. 9168–9180, 2023.

[28] J. Jia and A. R. Benson, "Neural jump stochastic differential equations," in *NeurIPS*, 2019.

[29] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *ICLR*, 2021.

[30] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *SIGKDD*, 2018, pp. 387–395.

[31] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *ICLR*, 2022.

[32] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: poster and demo track*, 2012.

[33] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Mach. Learn.*, 2016.

[34] Y. Yang, C. Zhang, T. Zhou, Q. Wen, and L. Sun, "Dcdetector: Dual attention contrastive representation learning for time series anomaly detection," in *SIGKDD*, 2023, pp. 3033–3045.

[35] B. Du, X. Sun, J. Ye, K. Cheng, J. Wang, and L. Sun, "Gan-based anomaly detection for multivariate time series using polluted training set," *Trans. Knowl. Data Eng.*, vol. 35, no. 12, 2023.

[36] C. Wang, Z. Zhuang, Q. Qi, J. Wang, X. Wang, H. Sun, and J. Liao, "Drift doesn't matter: Dynamic decomposition with diffusion reconstruction for unstable multivariate time series anomaly detection," in *NeurIPS*, 2023.

[37] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all: Power general time series analysis by pretrained LM," in *NeurIPS*, 2023.

[38] L. donghao and wang xue, "ModernTCN: A modern pure convolution structure for general time series analysis," in *ICLR*, 2024.

[39] J. Song, K. Kim, J. Oh, and S. Cho, "Memto: Memory-guided transformer for multivariate time series anomaly detection," *NeurIPS*, 2024.

[40] Y. Feng, W. Zhang, Y. Fu, W. Jiang, J. Zhu, and W. Ren, "Sensitivehue: Multivariate time series anomaly detection by enhancing the sensitivity to normal patterns," in *SIGKDD*, 2024, pp. 782–793.

[41] L. Shen, Z. Li, and J. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," *NeurIPS*, 2020.

[42] A. Huet, J. M. Navarro, and D. Rossi, "Local evaluation of time series anomaly detection algorithms," in *SIGKDD*, 2022, pp. 635–645.

[43] Y. Sun, G. Pang, G. Ye, T. Chen, X. Hu, and H. Yin, "Unraveling the 'anomaly' in time series anomaly detection: A self-supervised tri-domain solution," in *ICDE*, 2024, pp. 981–994.

[44] J. Paparrizos, P. Boniol, T. Palpanas, R. Tsay, A. J. Elmore, and M. J. Franklin, "Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection," *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2774–2787, 2022.

[45] X. Hao, Y. Chen, C. Yang, Z. Du, C. Ma, C. Wu, and X. Meng, "From chaos to clarity: Time series anomaly detection in astronomical observations," in *ICDE*, 2024, pp. 570–583.

[46] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, and D. Pei, “Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding,” in *SIGKDD*, 2021, pp. 3220–3230.

[47] Y. Chen, C. Zhang, M. Ma, Y. Liu, R. Ding, B. Li, S. He, S. Rajmohan, Q. Lin, and D. Zhang, “Imdiffusion: Imputed diffusion models for multivariate time series anomaly detection,” in *Proc. VLDB Endow.*, vol. 16, no. 4, 2023, pp. 766–779.

Paper C

Unsupervised Time Series Anomaly Prediction with Importance-based Generative Contrastive Learning

Kai Zhao, Zhihao Zhuang, Chenjuan Guo, Hao Miao, Christian S. Jensen, Yunyao Cheng, Bin Yang

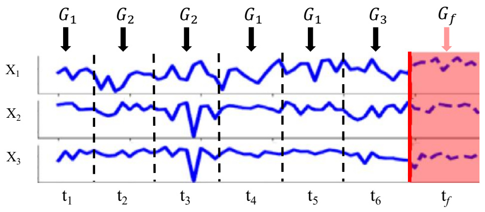
The paper has been submitted for publication

Abstract

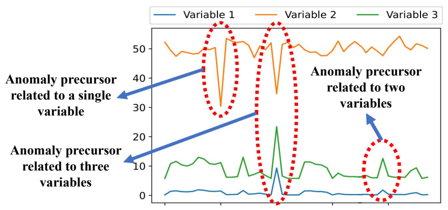
We study the problem of time series anomaly prediction, which is relevant to a range of real-world applications. Existing anomaly prediction methods rely on labeled training data for achieving acceptable accuracy. However, such data may be difficult to obtain; and in real-time deployments, anomalies can occur that were not seen in labeled data, thus making them difficult to predict. We provide a theoretical analysis and propose an Importance-based Generative Contrastive Learning method (IGCL) for unsupervised anomaly prediction. IGCL employs a controlled diffusion module to produce anomaly precursor patterns. Next, ICGL learns contextual representations to extract temporal dependencies from pairs of normal time series and anomaly precursors. IGCL is then able to predict anomalies by identifying anomaly precursors that will evolve into future anomalies. To address challenges caused by potentially complex precursor combinations involving multiple variables, we propose a memory bank with importance scores that stores representative samples adaptively and generates more complex anomaly precursors. Extensive experiments on nine benchmark datasets offer evidence that the proposed method is able to outperform state-of-the-art baselines.

1 Introduction

Much data collected by real-world applications can be modeled as time-dependent observations that form multi-variable time series [1–4]. For example, computing systems record runtime indicators [5, 6] and environmental systems record water quality [7–9]. Based on the current time series data, anomaly prediction makes real-time judgments on whether anomaly signals start to occur that indicate that severe anomalies will occur after the current time [10–13], as exemplified in Figure C.1(a). Anomaly prediction plays an important role in scenarios that preserve some form of safety and prioritize losses caused by anomalies [14–17], *e.g.,* in health care [11] or drinking-water systems [10]. For example, anomaly prediction can help address impending pollution or faults in advance [12, 13] or can warn of extreme environmental conditions [17].



(a) Supervised anomaly prediction learned with labeled anomaly precursors.



(b) Different anomaly precursor combinations in multi-variable time series.

Fig. C.1: Our motivation.

Only a few classification-based studies [14–17] exist related to the time series anomaly prediction. Existing methods predict future anomalies by determining whether current observations contain anomaly signals that are the start of deviations from normal behaviors [11–13]. Such early signals are called the anomaly *precursors* [10]. However, to obtain accurate results, existing methods heavily rely on supervised learning, where anomaly precursors are given during training.

Yet, in many real-world applications, there is insufficient labeled training data to enable supervised learning because manual labeling is costly [18]. Further, unexpected anomalies [19, 20] that did not appear prior to occurring during real-time deployment, and thus were not considered during training and make supervised methods fail in practice [9, 21]. Therefore, unsupervised time series anomaly prediction is of high interest. However, this has not been studied in depth nor analyzed theoretically due to its challenging nature, as outlined next.

Challenge 1: In the setting of unsupervised anomaly prediction, there is no labeled anomaly precursor data to enable models to learn temporal dependencies. Although PAD [10] proposes to use unsupervised anomaly detection methods [22–24] with a specific threshold to identify the precursors, their performance is limited when compared to supervised learning methods [13]. Without labeled data, existing methods cannot learn temporal dependency features from pairs of a successive normal sub-sequence and precursor, where the data just begins to deviate from normal [11], which is crucial to enable anomaly prediction [12, 17]. As a result, existing unsupervised methods do not perform well in anomaly prediction.

Challenge 2: We are facing potentially very complex anomaly precursor combinations in multi-variable time series, which cause very high time and space complexity. As shown in Figure C.1(b), for any n-variable time series, there can be as many as $O(2^n)$ potential anomaly precursor combinations [13], where different anomaly precursors may appear in each individual variable or in variable combinations [12]. It is very challenging to learn temporal dependencies comprehensively from this many anomaly precursor combinations.

We propose a novel anomaly prediction method called **IGCL**, which is short for **I**mportance-based **G**enerative **C**ontrastive **L**earn-ing for unsupervised time series anomaly prediction. To address **Challenge 1**, we employ a generative contrastive learning architecture. We propose an anomaly precursor pattern generation module that employs a diffusion-based transformation and variance regularization, to generate different kinds of potential precursor patterns labeled as negative data using Gaussian noise. We also propose an overlapping window-based temporal convolutional network (TCN) architecture to efficiently extract temporal dependencies and learn positive and negative contextual representations. More specifically, the model distinguishes normal sub-sequences from anomaly precursors, where pairs of successive normal sub-sequences are positive samples and pairs of a normal sub-sequence and an anomaly precursor, which contain temporal changes from normal to abnormal, are negative samples.

To address **Challenge 2**, we propose a memory bank with importance-based scores to adaptively store representative anomaly precursors, which helps generate more complex anomaly precursor combinations related to different variables. First, anomaly precursor patterns can be generated efficiently as only one variable is considered during each iteration. Second, the memory bank, with its fixed size $K \ll O(2^n)$, is initialized with generated anomaly precursors related to one variable. Then, other one-variable anomaly precursor patterns are generated iteratively and are injected into the previously generated anomaly precursors stored in the memory bank, thereby accumulating complex anomaly precursor combinations involving more variables. The memory bank only stores representative anomaly precursors and drops anomaly precursors that are already distinguished. We summarize our contributions as follows.

- We propose a generative contrastive learning architecture with a diffusion-based transformation and variance regularization to generate different precursor patterns and learn temporal dependencies to distinguish between normal and anomaly precursors.
- We propose a memory bank with importance scores to store representative negative samples, which helps to generate more complex negative samples.
- Experiments on nine benchmarks from different domains show our method outperforms the state-of-the-art baselines.

2 Preliminary

2.1 Definitions

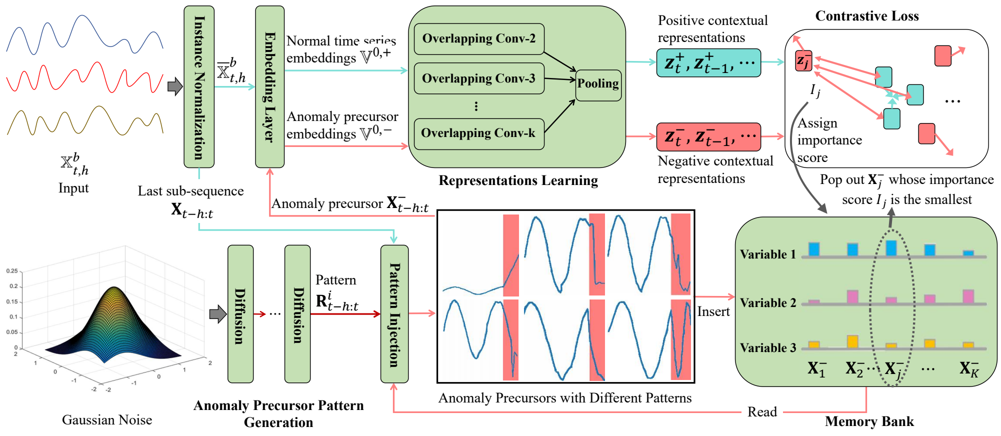
We first formalize the unsupervised time series anomaly prediction problem. Frequently used notation is summarized in Appendix A.

A Multi-variable Time Series is represented as $\mathbf{X} = \langle \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T \rangle \in \mathbb{R}^{N \times T}$, where N is the number of variables collected at each timestamp, T is the total number of historical timestamps, $\mathbf{x}_t \in \mathbb{R}^N$ denotes the N-dimensional observations at timestamp t. We use $\mathbf{X}_{t+1:t+f} \in \mathbb{R}^{N \times f}$ to indicate the data during the time window $[t+1, t+f]$, use $\mathbf{X}^i_{t+1:t+f} \in \mathbb{R}^f$ to indicate the data of the i-th variable, and use $\mathbf{X}^i_t \in \mathbb{R}^1$ to indicate the data of the i-th variable at timestamp t. We also use T to denote the current timestamp.

A Sub-sequence $\mathbf{X}_{t_1:t_2}$ is a continuous subset of $\mathbf{X}$ from t_1 to t_2. The set of b most recent sub-sequences before the current timestamp T, each with sequence length of $h+1$, is denoted as $\mathbb{X}^b_{T,h} = \{\mathbf{X}_{T-h:T}, \mathbf{X}_{T-1-h:T-1}, \cdots, \mathbf{X}_{T-b+1-h:T-b+1}\}$.

An Anomaly Precursor is a sub-sequence $\mathbf{X}_{T-h:T}$ containing early signals that just start deviating from normal and followed by future anomalies in $\mathbf{X}_{T+1:T+f}$ [10, 12, 13], where h and f are hyper-parameters for the look-back and look-forward windows.

Unsupervised time series anomaly prediction. At any current timestamp T, given the historical observations $\mathbf{X}$, where we use a set of b most



$\mathbb{X}^b_{t,h}$
Input
Instance Normalization
$\overline{\mathbb{X}}^b_{t,h}$
Embedding Layer
Normal time series embeddings $\mathbb{V}^{0,+}$
Anomaly precursor embeddings $\mathbb{V}^{0,-}$
Overlapping Conv-2
Overlapping Conv-3
Overlapping Conv-k
Pooling
Representations Learning
Positive contextual representations
$z^+_t, z^+_{t-1}, \cdots$
Negative contextual representations
$z^-_t, z^-_{t-1}, \cdots$
Contrastive Loss
z_j
I_j
Assign importance score
Pop out $\mathbf{X}^-_j$ whose importance score I_j is the smallest
Last sub-sequence $\mathbf{X}_{t-h:t}$
Anomaly precursor $\mathbf{X}^-_{t-h:t}$
Gaussian Noise
Diffusion
Diffusion
Pattern $\mathbf{R}^i_{t-h:t}$
Pattern Injection
Anomaly Precursor Pattern Generation
Anomaly Precursors with Different Patterns
Insert
Read
Variable 1
Variable 2
Variable 3
$\mathbf{X}^-_1$ $\mathbf{X}^-_2$ $\mathbf{X}^-_j$ ⋯ $\mathbf{X}^-_K$
Memory Bank

Fig. C.2: The overall architecture.

recent sub-sequences $\mathbb{X}^b_{T,h} = \{\mathbf{X}_{T-h:T}, \mathbf{X}_{T-1-h:T-1}, \cdots, \mathbf{X}_{T-b+1-h:T-b+1}\}$, the goal is to determine a real-time anomaly score $\hat{p}_T$ to indicate whether there are anomaly precursors currently and more anomalies in $\mathbf{X}_{T+1:T+f}$. The transformation of anomaly scores into binary labels is done by applying a threshold δ [10], *i.e.*, if $\hat{p}_T < \delta$, the future sub-sequence $\mathbf{X}_{T+1:T+f}$ is normal; and if $\hat{p}_T \geq \delta$, the future sub-sequence $\mathbf{X}_{t+1:t+f}$ is abnormal. Thus, we formulate the time series anomaly prediction problem as that of finding a model with a mapping function $\mathcal{F}$ and $\hat{p}_T = \mathcal{F}_\theta(\mathbb{X}^b_{T,h})$, where θ is the parameters of the model $\mathcal{F}$. For the unsupervised time series anomaly prediction problem, we do not have the true score p_T as there is no labeled data.

2.2 Theoretical analysis

We analyze unsupervised time series anomaly prediction using Maximum Mean Discrepancy (MMD) theory [25]. We utilize Markov Chains [26], widely used in time series analytics [27], to model sequence-based probabilities. We use random variables $\mathbf{H_{t-h-1}}$ and $\mathbf{H_t}$ to denote what observations the sub-sequences $\mathbf{X}_{t-2h-1:t-h-1}$ and $\mathbf{X}_{t-h:t}$ may be observed during two successive time-windows $[t-2h-1, t-h-1]$ and $[t-h, t]$, respectively. Then, we define

$$\mathcal{N} = P(\cdot|\mathbf{H_{t-h-1}} = \mathbf{X}_{t-2h-1:t-h-1}, \mathbf{H_t} \text{ is normal})$$

to denote the Markov conditional distribution of the normal sub-sequence observed during time-window $[t-h, t]$, given that its previous observations during time-window $[t-2h-1, t-h-1]$ are $\mathbf{X}_{t-2h-1:t-h-1}$. Similarly, we define

$\mathcal{A} = P(\cdot|\mathbf{H_{t-h-1}} = \mathbf{X}_{t-2h-1:t-h-1}, \mathbf{H_t}$ is anomaly precursor)

to denote the Markov conditional distribution of the anomaly precursor observed during time-window $[t-h, t]$. We present the theorem here and its proof in Appendix B.

Theorem: Under the unsupervised setting, the future anomaly is predictable if and only if we can identify its anomaly precursor that follows a distribution different from the normal previous sub-sequences. □

Thus, our model aims to distinguish normal distribution $\mathcal{N} = P(\cdot|\mathbf{H_{t-h-1}}, \mathbf{H_t}$ is normal) from any potential different distributions $\mathcal{A} = P(\cdot|\mathbf{H_{t-h-1}}, \mathbf{H_t}$ is anomaly precursor).

3 Methodology

We present the overall architecture in Figure C.2, which consists of anomaly precursor pattern generation, positive and negative representations learning, contrastive loss, and the memory bank.

For any timestamp t, the model takes as input the set of b available historical sub-sequences $\mathbb{X}_{t,h}^b = \{\mathbf{X}_{t-h:t}, \mathbf{X}_{t-1-h:t-1}, \cdots\cdots,$ $\mathbf{X}_{t-b+1-h:t-b+1}\}$. We first apply Instance Normalization [28] on the observation of each time series variable, *i.e.,* $\mathbf{X}_{t'}^i$ in the set $\mathbb{X}_{t,h}^b$, and output $\overline{\mathbb{X}}_{t,h}^b$, to learn stable components [29] from time series.

Embedding layer takes as input the normalized $\overline{\mathbb{X}}_{t,h}^b$, maps the observations at each timestamp t', *i.e.,* $\mathbf{X}_{t'} \in \mathbb{R}^N$, to a d-dimensional vectors $\mathbf{v}_{t'} \in \mathbb{R}^d$, which extracts the dense feature from normal time series, and outputs a set of feature vectors, *i.e.,* $\mathbb{V}^{0,+}$.

Anomaly precursor pattern generation takes as input a randomly generated Gaussian noise $x \sim N(0, I)$, and iteratively outputs any potential anomaly precursor pattern, *i.e.,* $\mathbf{R}_{t-h:t}^i$, for a randomly selected i-th time series variable.

Pattern injection module injects $\mathbf{R}_{t-h:t}^i$ into the last sub-sequence $\mathbf{X}_{t-h:t}$ in the selected i-th time series variable, and outputs a time series anomaly precursor, *i.e.,* $\mathbf{X}^{-}_{t-h:t} = \mathbf{X}^{i}_{t-h:t} + \mathbf{R}^{i}_{t-h:t}$, which is fed into the embedding layer to get precursor feature vectors $\mathbb{V}^{0,-}$.

Positive and negative representations learning takes as input all possible pairs of successive sub-sequences from $\mathbb{V}^{0,+}$ and $\mathbb{V}^{0,-}$. For example, this module takes as input a pair of normal sub-sequences $\mathbf{X}_{t-2h-1:t-h-1}$ and $\mathbf{X}_{t-h:t}$ from $\mathbb{V}^{0,+}$, and outputs a positive contextual representation z^{+}_{t} that learns their normal temporal dependencies. This module also takes as input a pair of a normal sub-sequence $\mathbf{X}_{t-2h-1:t-h-1}$ from $\mathbb{V}^{0,+}$ and an anomaly precursor $\mathbf{X}^{-}_{t-h:t}$ from $\mathbb{V}^{0,-}$, and outputs a negative contextual representation z^{-}_{t} that learns their abnormal temporal dependencies.

With **contrastive loss**, the model distinguishes $\mathcal{N} = P(\cdot|\ \mathbf{H_{t-h-1}},\ \mathbf{H_t}$ is normal) and $\mathcal{A} = P(\cdot|\mathbf{H_{t-h-1}}, \mathbf{H_t}$ is anomaly precursor) by pushing z^{-}_{t} away from z^{+}_{t}. Intuitively, it clusters positive representations, *e.g.,* z^{+}_{t}, that belong to $\mathcal{N}$ together and dissociates negative representations, *e.g.,* z^{-}_{t}, that belong to $\mathcal{A}$ away.

The memory bank stores the representative anomaly precursors. The generated anomaly precursor patterns are iteratively injected into existing anomaly precursors in the memory bank, to accumulate more complex anomaly precursor combinations. The importance score I_j for the anomaly precursor $\mathbf{X}^{-}_{j}$ is calculated by its similarity with the normal samples. Finally, the memory bank pops out the anomaly precursor with the smallest importance score, which the model can easily distinguish from normal already.

3.1 Anomaly precursor pattern generation

As experimental results show diffusion models outperform other generative models for time series [30, 31], we propose diffusion-based transformation to generate diverse kinds of potential anomaly precursors from a simple Gaussian distribution, with our variance regularization to simulate precursor patterns that start deviating from normal.

The denoising diffusion model consists of two processes, *i.e.,* the pollution process with S steps and its reverse denoising diffusion process, where S is the max diffusion steps. Specifically, we use $x^0 = \mathbf{R}^{i}_{t-h:t} \in \mathbb{R}^h$ to denote any potential anomaly precursor pattern that could be more complicated than Gaussian patterns. The pollution process is shown in Appendix C.

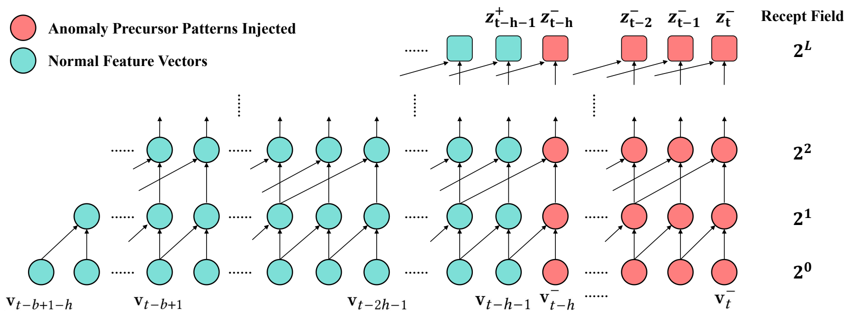


Fig. C.3: The overlapping window-based temporal convolutional network with a kernel size k of 2.

Our anomaly precursor pattern generation module is based on denoising diffusion process $p_\theta(x^{s-1}|x^s)$, reversing random Gaussian noise x^S to produce more complicated potential anomaly precursor patterns:

$$x^{s-1} \sim p_\theta(x^{s-1}|x^s) = \frac{1}{\sqrt{1-\beta^s}}\left(x^s - \sqrt{\beta^s}\varepsilon_\theta(x^s)\right),$$

where $\varepsilon_\theta(x^s) \sim N(\mu_s, {\sigma_s}^2)$, and μ_s is the mean value parameter and σ_s is the variance parameter to be learned. We use Multi-layer Perceptron (MLP) layers to model with μ^s and σ^s:

$$\mu^s, \sigma^s = MLP\Big(concat(x^s, s, \mathbf{X}^i_{t-h:t})\Big),\ \forall s \in \{S, S-1, \cdots, 1\},$$

where x^S is a random Gaussian noise, x^s is the denoised state after each step, and i denotes this process aiming to generate anomaly precursor patterns for the i-th variable. We have $\sigma^s\varepsilon + \mu^s \sim N(\mu_s, {\sigma_s}^2)$, where $\varepsilon \sim N(0, I)$, and $\sigma^s\varepsilon + \mu^s$ is derivable with respective to μ^s and σ^s which are the outputs from the MLP layers. Thus, the final denoising diffusion process is

$$x^{s-1} = \frac{1}{\sqrt{1-\beta^s}}\left(x^s - \sqrt{\beta^s}(\sigma^s\varepsilon + \mu^s)\right),$$

where we randomly sample the Gaussian noise x^S, and apply the denoising diffusion process step by step to get the anomaly precursor pattern $\mathbf{R}^i_{t-h:t} = x^0$.

We further propose regularization loss on parameters $N(\mu_s, {\sigma_s}^2)$, where the variance ${\sigma_s}^2$ controls the diversity of generated anomaly precursor patterns, and ${\sigma_s}^2$ and the mean value μ_s controls the degree of deviation from normal. First, the variance ${\sigma_s}^2$ should not be too large, as a large variance will produce severe anomalies rather than anomaly precursors with starting deviation from normal. Second, ${\sigma_s}^2$ should not be too small, as a small variance will generate less diverse [30] precursor patterns. Therefore, we propose to use the Kullback-Leibler divergence as regularization to make $N(\mu_s, {\sigma_s}^2)$ be close to $N(\mu_s, I)$ where ${\sigma_s}^2$ should be close to the unit normal variance I which is neither too large nor too small [32]. For any $N(\mu, \sigma^2)$, $KL\Big(N(\mu, \sigma^2)\Big\|N(\mu, I)\Big) = \frac{1}{2}\Big(-\log\sigma^2 + \sigma^2 - 1\Big)$. The proof is in Appendix D. Thus, our analytical solution of the regularization loss is:

$$\mathcal{L}_r = \sum_{s=1}^{S} \frac{1}{2}\Big(-\log\sigma_s^2 + \sigma_s^2 - 1\Big) \tag{C.1}$$

3.2 Positive and negative representations

We propose an overlapping window-based TCN to learn contextual representations from the all possible pairs of successive sub-sequences followed by the Markov Chain and extract the normal and abnormal temporal dependencies at once.

To be more specific, our embedding layer firstly maps the normalized time series data, *i.e.*, $\mathbf{X}_{t'} \in \mathbb{R}^N$ with $t' \in [t-b+1-h, t]$, equivalent to the set $\mathbb{X}^b_{t,h}$, into the d-dimensional feature vectors $\mathbf{v}_{t'} \in \mathbb{R}^d$, which aim to extract the dense features from time series for each timestamp. Further, the input to the embedding layer can also be coupled with other auxiliary attributes, such as the encoding of timestamps. The auxiliary attributes at timestamp t' are represented as $\mathbf{a}_{t'} \in \mathbb{R}^F$, where F is the total dimensions of the auxiliary attributes. Thus, the embedding layer concatenates the original time series data $\mathbf{X}_{t'}$ and the auxiliary attributes

$$\mathbf{f}_{t'} = concat(\mathbf{X}_{t'}, \mathbf{a}_{t'})\ \forall t' \in [t-b+1-h, t], \tag{C.2}$$

as its input, and maps it to the feature vector by:

$$\mathbf{v}_{t'} = \sigma(W_e\, \mathbf{f}_{t'}),\ \forall t' \in [t-b+1-h, t], \tag{C.3}$$

where $\mathbf{v}_{t'} \in \mathbb{R}^d$ is the learned feature vector, σ is an activation function, and $W_e \in \mathbb{R}^{d\times(N+F)}$ is the learnable neural network parameters which extract the feature from the time series data and the auxiliary attributes at each timestamp. Following the same way, we can also get the feature vector $\mathbf{v}^{-}_{t'} \in \mathbb{R}^d$, where $t' \in [t-h, t]$, from the generated anomaly precursor $\mathbf{X}^{-}_{t-h:t}$.

Then, our overlapping window-based TCN uses the feature vectors $\mathbf{v}_{t'}$ and $\mathbf{v}^{-}_{t'}$ to learn the temporal dependencies for all possible pairs of successive sub-sequences at once. The temporal convolution operation slides over timestamps by skipping feature vectors with a certain dilation factor in different layers, as illustrated in Figure C.3. More specifically, this module takes as input all the feature vectors, *i.e.,* $\mathbf{v}_{t'}$ with $t' \in [t-b+1-h, t]$ and $\mathbf{v}^{-}_{t'}$ with $t' \in [t-h, t]$:

$$\begin{aligned}\mathbb{V}^{0,+} &= \Big\langle \mathbf{v}_{t-b+1-h}, \cdots, \mathbf{v}_{t-h}, \cdots, \mathbf{v}_t \Big\rangle, \\ \mathbb{V}^{0,-} &= \Big\langle \mathbf{v}_{t-b+1-h}, \cdots, \mathbf{v}^{-}_{t-h}, \cdots, \mathbf{v}^{-}_t \Big\rangle,\end{aligned} \tag{C.4}$$

and uses the learnable convolution $filter \in \mathbb{R}^k$ with the kernel size of k to extract features and learn the temporal dependencies by:

$$\begin{aligned}\mathbb{V}^{l,+}(t') &= \sum_{i=0}^{k-1} filter(i)\mathbb{V}^{l-1,+}(t' - k^{l-1} \times i), \text{ for } 1 \leq l \leq L, \\ \mathbb{V}^{l,-}(t') &= \sum_{i=0}^{k-1} filter(i)\mathbb{V}^{l-1,-}(t' - k^{l-1} \times i), \text{ for } 1 \leq l \leq L,\end{aligned} \tag{C.5}$$

where k^{l-1} is the dilation factor in the layer $l-1$, $\mathbb{V}^{l,+}(t')$ and $\mathbb{V}^{l,-}(t')$ is the learned dependency features in the layer l for timestamp t', and $L \sim O(log(h))$ is the max number of TCN layers whose recept field is large than the whole window size of the pair of successive sub-sequences. We use different kernel sizes to extract features in parallel and pool them together, to capture different kinds of temporal dependencies with different kernel scales. Finally, it outputs all contextual representations for all pairs of successive sub-sequences at once, *e.g.,* $z^{+}_{t} = \mathbb{V}^{L,+}(t)$, $z^{+}_{t-1} = \mathbb{V}^{L,+}(t-1)$, $z^{-}_{t} = \mathbb{V}^{L,-}(t)$ and $z^{-}_{t-1} = \mathbb{V}^{L,-}(t-1)$. In this way, our total complexity is reduced to $O\Big((b+h)log(h)\Big)$, rather than the complexity of all pairs of sub-sequences as $O\Big((b+h)^2\Big)$.

3.3 The objective function

We propose a contrastive strategy to distinguish $\mathcal{N}$ and $\mathcal{A}$ by pushing z_t^- away from z_t^+. Intuitively, it clusters positive representations, *e.g.,* z_t^+, that belong to $\mathcal{N}$ together and dissociates negative representations, *e.g.,* z_t^-, that belong to $\mathcal{A}$ away. Each anchor z_t^+ is the contextual representation from the pair of the successive normal sub-sequences before timestamp t, which represents for $\mathcal{N} = P(\cdot|\mathbf{H_{t-h}}, \mathbf{H_t}$ is normal). Thus, we make the anchor z_t^+ similar to the positive representations z_{t-j}^+ with $1 \leq j \leq P$, which are from the most nearby successive normal sub-sequences before timestamp t and they follow the normal data distribution. In order to avoid data leakage and be consistent with the inference stage, for each anchor z_t^+ the positive samples are always from the previous timestamps. Meanwhile, the anchor z_t^+ should be different from the negative representation z_t^-, which learns the temporal dependencies from normal to the generated anomaly precursors and represents for anomaly precursors distribution $\mathcal{A}$. Thus, our contrastive loss function is:

$$\mathcal{L}_c = \sum_{i=t}^{t-h} -log\frac{\sum_{j=1}^{P} exp\Big(Sim(z_i^+, z_{i-j}^+)/\tau\Big)}{exp\Big(Sim(z_i^+, z_i^-)/\tau\Big) + \sum_{j=1}^{P} exp\Big(Sim(z_i^+, z_{i-j}^+)/\tau\Big)}, \quad \text{(C.6)}$$

where τ is the temperature parameter to control the softmax strength, P is the hyper-parameter for the number of positive samples used, and Sim is the similarity measurement, *e.g.,* Cosine Similarity, to distinguish whether the contextual representations belong to the same distributions.

Finally, together with the regularization loss, we train the model using Adam gradient descent [33]:

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_r, \quad \text{(C.7)}$$

where λ is to control the strength of our diffusion regularization.

3.4 Memory bank

We propose to use a memory bank to store previously generated anomaly precursors and combine them with the current anomaly precursor pattern

$\mathbf{R}^i_{t-h:t}$, to accumulate more complex anomaly precursor combinations related to more than one variable.

To be more specific, the memory bank M has a fixed size of K and empirically $K \ll O(2^n)$ in our experiments:

$$\mathbf{M} = \Big\{\mathbf{X}^-_1, \mathbf{X}^-_2, \cdots, \mathbf{X}^-_K\Big\}, \tag{C.8}$$

where each $\mathbf{X}^-_j$ denotes a stored anomaly precursor. The currently generated anomaly precursor pattern $\mathbf{R}^i_{t-h:t}$ which is only related to the i-th variable, is also injected into the anomaly precursors in the memory bank:

$$\mathbf{X}^-_j \leftarrow \mathbf{X}^-_j + \mathbf{R}^i_{t-h:t}, \tag{C.9}$$

and thus we can get anomaly precursors that involve more than one variable. Then, similar to Equation (C.4), we can get more negative contextual representations, *e.g.,* $z^-_{t,j}$ and $z^-_{t-1,j}$, from the anomaly precursor $\mathbf{X}^-_j$. The contrastive loss $\mathcal{L}_c$ is updated to

$$\sum_{i=t}^{t-h} -log \frac{\sum_{j=1}^{P} exp\Big(Sim(z^+_i, z^+_{i-j})/\tau\Big)}{\sum_{j=0}^{K} exp\Big(Sim(z^+_i, z^-_{i,j})/\tau\Big) + \sum_{j=1}^{P} exp\Big(Sim(z^+_i, z^+_{i-j})/\tau\Big)}, \tag{C.10}$$

where $z^-_{i,0} = z^-_i$ is the negative contextual representations from the current anomaly precursor $\mathbf{X}^-_{t-h:t}$, and $z^-_{i,j}$ is the negative contextual representations from the j-th anomaly precursor $\mathbf{X}^-_j$ stored in the memory bank.

Finally, the current anomaly precursor $\mathbf{X}^-_{t-h:t}$ is inserted into the memory bank which will have a size of $K+1$, *i.e.,* $\mathbf{X}^-_0 = \mathbf{X}^-_{t-h:t}$. Instead of using a first-in-first-out strategy to maintain the memory bank with a fixed size of K, we propose to pop out the anomaly precursor based on an importance score. The intuition is that the memory bank should store the hard and important negative samples for further model training, *i.e.,* the anomaly precursors that are not yet distinguished by the model, and should pop out not important samples, *i.e.,* the anomaly precursors that are already dissociated away from normal. The importance score I_j for the anomaly precursor $\mathbf{X}^-_j$ is calculated by:

$$I_j = \sum_{i=t}^{t-h} Sim(z^+_i, z^-_{i,j}). \tag{C.11}$$

Table C.1: Overall performance. We report Precision, Recall, and F1-score results here, presented in percentages.

Dataset	PSM			SMAP			SWAN			SWaT			GECCO			SMD			MSL		
Metric	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
DAGMM	39.27	36.14	37.64	11.91	22.00	15.51	52.73	39.46	45.14	82.14	61.72	70.48	47.50	27.64	34.95	14.74	15.37	15.05	13.94	17.04	15.33
IForest	39.39	38.82	39.10	11.79	18.67	14.45	53.85	39.07	45.28	75.39	62.22	68.17	42.75	29.89	35.18	15.25	18.28	16.63	13.82	17.11	15.29
K-means	38.89	39.04	38.96	13.05	21.37	16.20	54.04	42.30	47.45	82.13	62.50	70.98	47.94	29.18	36.28	15.77	17.40	16.54	14.23	15.99	15.06
Deep-SVDD	36.63	36.30	36.46	11.77	17.09	13.94	52.39	39.18	44.83	81.29	61.95	70.31	46.09	28.45	35.18	14.99	15.78	15.37	12.94	16.04	14.32
THOC	37.64	38.93	38.27	12.61	21.89	15.99	54.93	40.02	46.30	82.42	62.45	71.06	46.41	29.49	36.09	16.90	16.79	16.84	14.37	16.22	15.24
DCdetector	28.97	12.05	17.02	8.04	11.30	9.40	21.04	10.94	14.40	46.45	36.36	40.79	1.76	3.46	2.33	3.71	9.03	5.26	2.35	3.46	2.80
PAD	31.23	33.05	32.11	11.01	22.37	14.76	50.13	37.61	42.98	74.83	59.09	66.04	44.38	28.17	34.46	11.23	18.76	14.05	14.37	13.22	13.77
ATransformer	30.56	34.64	32.47	11.40	22.80	15.20	52.20	38.75	44.48	75.00	60.50	66.97	43.62	27.92	34.05	10.52	19.42	13.65	14.62	12.42	13.43
LSTM-VAE	39.20	40.31	40.02	11.68	22.65	15.41	54.42	40.88	46.69	66.76	68.36	67.55	46.67	30.92	37.20	14.21	16.42	15.24	16.18	15.28	15.72
Omni	39.45	40.88	40.15	11.90	23.39	15.77	54.83	41.00	46.92	85.73	58.57	69.59	50.55	29.65	37.38	16.22	18.19	17.15	13.66	21.66	16.75
GANomaly	37.02	43.06	39.81	12.18	23.41	16.02	54.44	40.91	46.72	83.99	59.77	69.84	50.06	29.23	36.91	15.06	21.39	17.68	15.36	17.30	16.27
CAE-Ensemble	40.18	41.99	41.07	15.88	26.68	<u>19.91</u>	55.98	41.52	47.68	88.61	59.90	<u>71.48</u>	51.67	29.92	37.90	18.41	19.51	<u>18.94</u>	20.35	18.27	19.26
PUAD	39.91	42.38	41.11	15.20	27.20	19.50	54.91	41.19	47.07	85.68	58.05	69.21	50.76	29.58	37.38	16.37	21.85	18.72	20.04	17.60	18.74
D3R	40.73	42.21	<u>41.46</u>	15.73	26.89	19.85	55.32	42.64	<u>48.16</u>	84.13	61.85	71.29	51.24	30.91	<u>38.56</u>	15.78	19.33	17.38	19.99	20.32	<u>20.15</u>
IGCL	42.31	46.97	**44.52**	17.48	28.02	**21.53**	59.20	44.94	**51.09**	84.98	63.54	**72.71**	52.10	32.77	**40.23**	17.39	25.90	**20.81**	21.97	22.81	**22.38**

Thus, we pop out the anomaly precursor with the smallest importance score after each iteration of model optimization.

3.5 Inference

For each real-time T in the inference stage, we output the probability score $\hat{p}_T$ to predict how likely there is an anomaly precursor currently and will be more future anomalies in $\mathbf{X}_{T+1,T+f}$:

$$\hat{p}_T = \sum_{j=1}^{K} Sim(z_T^+, z_{T,j}^-) - \sum_{j=1}^{P} Sim(z_T^+, z_{T-j}^+), \tag{C.12}$$

where K is the size of the memory bank with anomaly precursors as negative samples, and P is the number of positive samples.

4 Experiments

4.1 Experimental settings

Datasets.

Following existing works [10, 22, 34], we validate the performance of our method with nine real-world benchmark datasets: SMD [6], PSM [5], MSL [35], SMAP [35], SWAN [7], SWaT [36], GECCO [7], UCR [34] and Credit [21]. More details of the datasets can be seen in Appendix E.1.

Baselines.

We compare the unsupervised anomaly prediction method PAD [10]. We also modify more state-of-the-art unsupervised anomaly detection methods into anomaly prediction following PAD:

- Density-based: DAGMM [37] and IForest [38].
- Clustering-based: K-means [18], Deep-SVDD [39] and THOC [40].
- Contrastive-based: DCdetector [41] and PAD [10].
- Autoregression and reconstruction-based: LSTM-VAE [42], ATransformer [36], Omni [6], GANomaly [43], CAE-Ensemble [22], PUAD [24] and D3R [31].

We introduce the details of baselines in Section 5. More details of settings are in Appendix E.2.

Evaluation metrics.

Following the evaluation methodology in existing works [10, 21, 22], we validated the performance of time series anomaly prediction models with Precision (P), Recall (R), F1-score (F1), and ROC-AUC [44], where larger values indicate higher model performance and ROC-AUC enables evaluation without choosing the threshold. We do not use the point adjustment (PA) metrics, as many recent works have demonstrated that PA can lead to faulty performance evaluations [21, 31, 44–46]. PA uses true labels to adjust the outputs of models, and it is known that using PA can result in state-of-the-art performance even with random guess [45, 46].

4.2 Overall comparison and analysis

Table C.1 and Table C.10 in Appendix E.3 show the overall anomaly prediction performance of different models on all datasets. We randomly repeat each method 3 times and report the average result, where the best F1-score and AUC-ROC are highlighted in bold and significantly outperform the underlined second-best ones.

Key observations are followed. First, IGCL consistently outperforms the state-of-the-art baseline methods on all datasets. It demonstrates that IGCL is able to learn the normal temporal dependencies and identify the

Table C.2: Parameter sensitivity

Dataset	SWaT			MSL		
K	P	R	F1	P	R	F1
4	84.67	61.93	71.54	20.01	20.17	20.09
8	84.23	62.48	71.74	20.38	21.42	20.89
12	84.52	62.97	72.17	20.50	21.51	20.99
16	84.21	63.19	72.20	20.43	21.69	21.04
20	84.09	63.25	72.19	20.33	21.65	20.97
24	84.98	63.54	**72.71**	21.97	22.81	**22.38**

Table C.3: Larger look-forward windows

Dataset	SWaT					
f	4		8		12	
Metric	F1	AUC	F1	AUC	F1	AUC
CAE-Ensemble	71.48	81.71	68.62	74.92	62.70	73.32
D3R	71.29	80.62	68.51	75.94	62.43	71.76
IGCL	**72.71**	**82.39**	**70.02**	**77.75**	**64.56**	**74.88**

abnormal temporal dependencies that change from the normal to the abnormal with our generated anomaly precursors, which finally improves the accuracy of anomaly prediction. Second, we observe that the contrastive-based, density-based, and clustering-based methods are unsatisfactory on most datasets. This is because they cannot generate anomaly precursors as negative samples to learn the abnormal temporal dependencies that change from normal to abnormal. They only learn with the non-labeled data as positive samples. Thus, their accuracy is limited as anomaly prediction requires learning both normal patterns and abnormal temporal dependencies. Third, IGCL achieves the best accuracy compared to the autoregression and reconstruction-based methods, as they only learn to reconstruct each normal time series sub-sequence within each time window. They cannot explicitly learn the abnormal temporal dependencies from normal to anomaly precursors and thus have limited accuracy. The experiments demonstrate that learning the temporal dependencies from a pair of successive time series sub-sequences that change from the normal to the anomaly precursor is crucial to the anomaly prediction.

Table C.4: Parameter sensitivity

Dataset	SWaT				PSM				SWAN			
h	P	R	F1	AUC-ROC	P	R	F1	AUC-ROC	P	R	F1	AUC-ROC
8	83.13	62.75	71.52	81.33	41.56	46.83	44.04	65.67	58.13	43.11	49.51	65.99
16	83.59	62.86	71.76	81.72	42.31	46.97	44.52	65.75	59.20	44.94	51.09	68.92
32	84.98	63.54	72.71	82.39	42.14	46.61	44.26	65.70	59.31	44.07	50.57	66.91
64	87.67	61.36	72.19	82.33	42.59	45.10	43.81	65.18	59.85	44.03	50.73	68.57

Table C.5: Ablation study.

Dataset	PSM			SWAN			SWaT			GECCO		
Metric	P	R	F1	P	R	F1	P	R	F1	P	R	F1
w/o APPGM	41.58	40.51	41.04	55.82	41.03	47.30	84.19	60.28	70.26	51.88	29.03	37.23
w/o regularization loss	41.39	43.70	42.51	57.14	41.81	48.29	84.45	63.01	72.17	52.03	31.90	39.55
w/o memory bank	41.42	42.86	42.13	56.24	41.50	47.76	84.59	61.25	71.05	52.27	31.24	39.11
w Transformer	41.07	44.23	42.59	57.08	42.63	48.80	84.26	62.86	72.00	52.20	32.12	39.77
IGCL	42.31	46.97	**44.52**	59.20	44.94	**51.09**	84.98	63.54	**72.71**	52.10	32.77	**40.23**

4.3 Paremeter sensitivity

We evaluate the impact of K, the size of the memory bank. The experimental results are shown in Table C.2. If we set a bigger size for the memory bank to store more representative anomaly precursors, our model will have better results. When K increases to 24, our results are significantly better than the baselines, and $K = 24$ is far smaller than $O(2^n)$, where n, *i.e.*, the total number of variables, is 51 and 55 for SWaT and MSL datasets, respectively.

We evaluate the anomaly prediction with larger look-forward windows f in Table C.3. We can see that IGCL still performs better than state-of-the-art baselines, even though the farther future anomalies are more difficult to predict.

The impact of the size of look-back windows h is in Table C.4. We can see that IGCL is relatively stable to h and time series from different domains require different look-back windows to better identify the anomaly precursors.

4.4 Ablation studies

We conduct ablation studies to validate the effectiveness of our key components. In particular, we compare IGCL with the following variants:

- w/o anomaly precursor pattern generation module (APPGM): This variant directly uses random Gaussian noises as anomaly precursor patterns

Table C.6: Runtime and total parameters used

Dataset	SWaT		MSL	
Method	Runtime (s)	Parameters (K)	Runtime (s)	Parameters (K)
PAD	2376	204	248	204
GANomaly	993	1693	81	1695
CAE-Ensemble	787	236	62	236
D3R	1443	380	133	382
w Transformer	1286	471	108	472
IGCL	760	172	61	173

Table C.7: Comparisons between representative methods that can be applied to unsupervised time series anomaly prediction.

Methods	For time series	Unsupervised	Precursor Generation	Precursor Dependencies
[38, 39, 47]	✗	✓	✗	✗
[15–17]	✓	✗	✗	✗
[11–14]	✓	✗	✗	✓
[6, 18, 22, 23, 40, 43, 48–52] [10, 24, 30, 31, 36, 37, 41, 42]	✓	✓	✗	✗
Our IGCL	✓	✓	✓	✓

without the diffusion process.

- w/o regularization loss: This variant does not have regularization loss to control the diffusion process.
- w/o memory bank: This variant does not use the memory bank, and thus precursors only appear in each individual variable.
- w Transformer: This variant does not use the overlapping window-based TCN. It uses naive Transformer [53] to learn representations.

Table C.5 shows the accuracy of different variants. From Table C.5 we observe that: (1) As our anomaly precursor pattern generation module, regularization loss, and memory bank all help generate more complicated and representative anomaly precursors as negative samples, our model can have more accurate results. (2) Our overlapping window-based TCN can effectively learn the temporal dependencies from pairs of successive subsequences, and it is also more efficient than naive Transformer as shown in Table C.6.

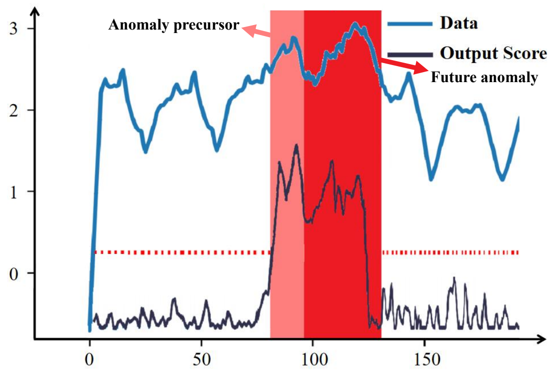


Fig. C.4: IGCL can output larger scores on the precursors that start to deviate from the normal ahead of the severer anomalies.

4.5 Runtime and total parameters used

We show the overall runtime and the number of total parameters used for different state-of-the-art methods in Table C.6. It can be observed that our IGCL model is better than these baseline methods regarding the runtime and the number of parameters used, as our memory bank can help efficiently generate complicated anomaly precursors related to different variables, and our overlapping window-based TCN can learn the contextual representations for all possible pairs of successive sub-sequences at once. Besides, CAE-Ensemble which uses ensemble convolutional networks also has good performance regarding accuracy and complexity. However, CAE-Ensemble is still worse than our model, which validates the efficiency and effectiveness of our model design.

4.6 Case study and visualization

We visualize the anomaly prediction of our IGCL method on the SMAP dataset, as shown in Figure C.4. By identifying the anomaly precursor that just begins to deviate from the normal, our model can report a warning at timestamp 80, ahead of the severe anomalies after timestamp 100.

5 Related Works

We review the relevant works on time series anomaly prediction, anomaly detection, contrastive learning, and generative learning. We also compare our method with related works in Table C.7.

5.1 Time series anomaly prediction

Based on the current time series data, anomaly prediction aims to make the real-time judgment on whether there are starting anomaly precursors where data begin to deviate from normal to future anomalies. Anomaly prediction plays an important role in many scenarios that prioritize the safety or economic influence of anomalies, However, there are a few studies [10–17] related to this problem. FEP [17] extracts statistical information from manually labeled frequency-based features as the starting anomaly precursors to predict future anomalies. EADS [15, 16] and VFP [13] use graphs to extract correlations among variables to help classify the starting anomaly precursors related to different variables. MCDA [11] proposes multi-instance attention, dCNN [12] proposes deep convolutional neural networks and CHE [14] proposes hierarchical windows, to extract temporal dependency features. They learn temporal dependencies from the pair of the normal time series sub-sequences and the starting anomaly precursors which help better identify starting abnormal changes for anomaly prediction.

However, these existing methods heavily rely on supervised learning of the temporal dependencies to get accurate anomaly prediction results. They are limited and fall short in the unsupervised time series anomaly prediction task for practical applications.

5.2 Time series anomaly detection

There has been a lot of research on time series anomaly detection. Early works citeanomlyreview utilize classification-based methods to detect anomalies in the historical time series data. Recent methods mainly focus on unsupervised settings based on the one-class classification theory [39]. They can be further categorized into four categories [3], including density estimation-based methods [47], clustering-based methods [4], autoregression and reconstruction based methods [6] methods, and contrastive-based methods [41].

The clustering-based methods, such as K-means [18], Deep-SVDD [39], and ITAD [49], aim to cluster normal samples within a center. THOC [40] uses a hierarchical RNN to learn dynamic clustering centers for time series data. The density estimation-based anomaly detection methods, such as LOF [47], assume abnormal samples lie in a low-density region. IForest [38] uses the tree model and DAGMM [37] uses the encoding probability model to measure the density or distribution. LSTM [18] proposes forecasting-based approaches that forecast values and based on the forecasting errors detect whether the time points are anomalies. Autoencoder [23, 50, 51] (AE) is a classic architecture used in reconstruction. Omni [6], InterFusion [52] and LSTM-VAE [42] utilize the Variational AE and use reconstruction probability to detect anomalies. GANomaly [43] introduces a discriminator to generate normal data and detect anomalies based on reconstruction errors. CAE-Ensemble [22] proposes to ensemble different AEs for reconstruction. PUAD [24] uses meta-learning-oriented AE to reconstruct typical time series.

Generative Learning. The diffusion-based network is a generative model that generates data with denoising steps [30]. ImDiffusion [48], DiffAD [30] and D3R [31] utilize diffusion-based networks to generate normal time series data and detect anomalies based on reconstruction errors. However, these methods only learn the underlying normal data distribution, but cannot generate anomaly precursors.

Contrastive Learning. Contrastive learning learns useful representations by contrasting positive pairs against negative pairs with data augmentation [54]. Recently, contrastive learning has been proposed for anomaly detection, such as DCdetector [41], where two view representations of a normal timestamp should be similar, and contrastive errors are used to detect anomalies. ATransformer [36] combines the reconstruction errors and contrastive errors together to detect anomalies.

PAD [10] proposes to utilize unsupervised anomaly detection methods and adjust specific thresholds on the anomaly scores to identify the anomaly precursors for anomaly prediction. However, these methods do not have negative samples and only reconstruct or contrast positive samples. They cannot learn the temporal precursor dependencies from the pair of normal time series and the anomaly precursor. Their performance is limited compared to supervised learning with labeled anomaly precursors as negative samples [11–14].

6 Conclusion

We present an importance-based generative contrastive learning model for a novel problem of unsupervised time series anomaly prediction. We propose anomaly precursor pattern generation, with diffusion-based transformation and variance regularization, to generate diverse anomaly precursors as labeled negative data. Then, we propose an overlapping window-based contrastive learning to distinguish anomaly precursors from the normal efficiently. Last, we propose a memory bank with importance scores to store representative anomaly precursors to generate complicated precursors efficiently. Experiments on seven benchmark datasets demonstrate the superiority of our method. In future works, it is worth improving IGCL with pre-training on large-scale time series data.

A Notation

Frequently used notation is summarized in Table C.8.

B Proof

Theorem 1: Under the unsupervised setting, the future time series anomaly is unpredictable if we cannot identify the anomaly precursor in the current observation that follows the same distribution as the normal previous sub-sequences do.

Proof 1: As the current observations follow the same conditional distribution as the normal sub-sequence follows, we have $\mathcal{N} = \mathcal{A}$. For any model $\mathcal{F}$ we have:

$$\underset{\mathbf{H_t} \sim \mathcal{A}}{P}\left[\mathcal{F}(\mathbb{X}^b_{t,h}) = \hat{p}_t\right] - \underset{\mathbf{H_t} \sim \mathcal{N}}{P}\left[\mathcal{F}(\mathbb{X}^b_{t,h}) = \hat{p}_t\right] = 0. \tag{C.13}$$

Thus, the predicted probability score for the anomaly precursor in the current observation is equal to that for the normal sub-sequences. Therefore, no unsupervised model can predict the future anomaly if we cannot identify the anomaly precursors in the current observations when the anomaly precursor follows the same distribution as the normal sub-sequences do. □

Theorem 2: Under the unsupervised setting, the future time series anomaly is predictable if we can identify its anomaly precursor in the current observation that follow a distribution different from the normal sub-sequences.

Proof 2: As the anomaly precursor follows a data distribution different from the normal sub-sequences, we have $\mathcal{N} \neq \mathcal{A}$. According to the Maximum Mean Discrepancy (MMD) theory [25], there exists at least one mapping function $\mathcal{F}$ satisfying:

$$\mathop{\mathbb{E}}_{\mathbf{H_t} \sim \mathcal{A}} \left[\mathcal{F}(\mathbb{X}^b_{t,h})\right] - \mathop{\mathbb{E}}_{\mathbf{H_t} \sim \mathcal{N}} \left[\mathcal{F}(\mathbb{X}^b_{t,h})\right] > 0 \tag{C.14}$$

Thus, the expectation of the predicted probability score for the anomaly precursor is larger than that for the normal sub-sequence. Therefore, there exists at least one model $\mathcal{F}$ that can predict a larger probability score to identify that there is the starting anomaly precursor and the future time series is more like an anomaly when the anomaly precursor follows the data distribution different from the normal sub-sequences. □

Based on the above analysis, we can only predict the future time series anomalies whose anomaly precursors just begin to deviate from the nearest previous normal sub-sequences under the unsupervised setting. However, we do not have labeled anomaly precursor data for the model to learn the temporal dependencies to estimate the Markov conditional distributions for $\mathcal{A} = P(\cdot|\mathbf{H_{t-h-1}}, \mathbf{H_t}$ is anomaly precursor) and $\mathcal{N} = P(\cdot|\mathbf{H_{t-h-1}}, \mathbf{H_t}$ is normal). Thus, the best potential model is that we generate different kinds of potential anomaly precursors to simulate the beginning of deviation from normal and then learn the temporal dependency features.

Table C.8: Notation

Notation	Explanation
$\mathbf{x}_t$	The observed data at timestamp t from the N-variable time series $\mathbf{X}$
$\mathbf{X}_{t+1:t+f}$	The sub-sequence of the time series $\mathbf{X}$ from time $t+1$ to $t+f$, with sequence length of f
$\mathbf{X}^i_{t-h:t}$	The data of the i-th variable starting from $t-h$ to t
$\mathbb{X}^b_{T,h}$	The set of most recent b sub-sequences with length $h+1$ before the current timestamp T $\mathbb{X}^b_{T,h} = \{\mathbf{X}_{T-h:T}, \cdots, \mathbf{X}_{T-b+1-h:T-b+1}\}$
$N(\mu, \sigma^2)$	The multi-variable Gaussian distribution with mean vector μ and covariance matrix σ^2
$\mathbf{R}^i_{t-h:t}$	The anomaly precursor patterns for the i-th variable
S	The max steps for diffusion-based transformation
$\mathbf{v}_t$	The feature vector extracted from time series at timestamp t
z^+_t	The representations from $(\mathbf{X}_{t-2h-1:t-h-1}, \mathbf{X}_{t-h:t})$ to estimate $\mathcal{N} = P(\cdot\|\mathbf{H_{t-h}}, \mathbf{H_t}$ is normal$)$
z^-_t	The representations from $(\mathbf{X}_{t-2h-1:t-h-1}, \mathbf{X}^-_{t-h:t})$ to estimate $\mathcal{A} = P(\cdot\|\mathbf{H_{t-h}}, \mathbf{H_t}$ is an anomaly precursor$)$
K	The max size of the memory bank

C The pollution process

The noise pollution process $q(x^s|x^{s-1})$ is shown as

$$x^s = \sqrt{1-\beta^s}x^{s-1} + \sqrt{\beta^s}\epsilon^s, \ \forall s \in \{1, 2, \cdots, S\}, \tag{C.15}$$

where $\epsilon^s \sim N(0, I)$ are the independent Gaussian pollutions to be added in different steps, I is the identity matrix, $0 < \beta^s < 1$ are the hyper-parameters control the amount of pollution added in each step. Thus, we have:

$$
\begin{aligned}
x^s &= \sqrt{1-\beta^s}x^{s-1} + \sqrt{\beta^s}\epsilon^s \\
&= \sqrt{1-\beta^s}(\sqrt{1-\beta^{s-1}}x^{s-2} + \sqrt{\beta^{s-1}}\epsilon^{s-1}) + \sqrt{\beta^s}\epsilon^s \\
&= \cdots \\
&= (\sqrt{1-\beta^s}\sqrt{1-\beta^{s-1}}\cdots\sqrt{1-\beta^1})x^0 + \\
&\quad (\sqrt{1-\beta^s}\sqrt{1-\beta^{s-1}}\cdots\sqrt{1-\beta^2})\sqrt{\beta^1}\epsilon^1 + \\
&\quad (\sqrt{1-\beta^s}\sqrt{1-\beta^{s-1}}\cdots\sqrt{1-\beta^3})\sqrt{\beta^2}\epsilon^2 + \cdots + \\
&\quad \sqrt{1-\beta^s}\sqrt{\beta^{s-1}}\epsilon^{s-1} + \sqrt{\beta^s}\epsilon^s
\end{aligned}
\tag{C.16}
$$

For any independent Gaussian distributions $\epsilon_1 \sim N(\mu_1, \sigma_1^2)$ and $\epsilon_2 \sim N(\mu_2, \sigma_2^2)$, we have the following properties:

$$
a\epsilon_1 + b \sim N(a\mu_1 + b, a^2\sigma_1^2), \tag{C.17}
$$

$$
\epsilon_1 + \epsilon_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2). \tag{C.18}
$$

We also have:

$$
\begin{aligned}
1 = &\left(\sqrt{1-\beta^s}\sqrt{1-\beta^{s-1}}\cdots\sqrt{1-\beta^1}\right)^2 + \\
&\left(\sqrt{1-\beta^s}\sqrt{1-\beta^{s-1}}\cdots\sqrt{1-\beta^2}\right)^2\beta^1 + \\
&\left(\sqrt{1-\beta^s}\sqrt{1-\beta^{s-1}}\cdots\sqrt{1-\beta^3}\right)^2\beta^2 + \cdots + \\
&(1-\beta^s)\beta^{s-1} + \beta^s,
\end{aligned}
\tag{C.19}
$$

Thus, we have:

$$
x^s = \sqrt{\widetilde{\alpha}^s}x^0 + \sqrt{1-\widetilde{\alpha}^s}\epsilon, \tag{C.20}
$$

where $\epsilon \sim N(0, I)$ and $\sqrt{\widetilde{\alpha}^s} = \sqrt{1-\beta^s}\sqrt{1-\beta^{s-1}}\cdots\sqrt{1-\beta^1}$. Then, for enough S steps and proper $0 < \beta^s < 1$, $\sqrt{\widetilde{\alpha}^s}$ will be close to 0, and the output from the noise pollution process, *i.e.,* x^S, will be gradually corrupted into random Gaussian distribution.

Table C.9: Details of benchmark datasets.

Dataset	Domain	N	#Training (Unlabeled)	#Validation (Unlabeled)	#Test (Labeled)	Anomaly Rate(%)
SMD	server machine	38	566,724	141,681	708,420	4.16
PSM	application server	25	105,885	26,398	87,841	27.8
MSL	NASA space sensor	55	46,653	11,663	73,729	10.72
SMAP	NASA space sensor	25	108,146	27,036	427,617	13.13
SWAN	space solar weather	38	48,000	12,000	60,000	32.6
SWaT	water treatment	51	396,000	99,000	449,919	12.14
GECCO	water quality	9	55,408	13,852	69,261	1.1
UCR	various domains	1	1,790,679	447,670	6,143,541	0.6
Credit	finance	29	159,491	39,873	85,443	0.172

Table C.10: More experiment results of state-of-the-art methods, presented in percentages.

Dataset	PSM	SMAP	SWAN	SWaT	GECCO	SMD	MSL	UCR				Credit			
Metrics	AUC-ROC							P	R	F1	AUC-ROC	P	R	F1	AUC-ROC
DCdetector	50.91	43.98	51.00	65.57	50.79	50.44	30.34	12.59	10.90	11.68	48.65	37.50	12.96	19.26	64.34
PAD	51.02	50.13	50.85	61.35	49.71	49.66	48.28	29.92	20.35	24.22	51.67	31.49	16.01	21.23	68.76
ATransformer	54.91	53.70	51.95	75.63	50.39	59.30	50.21	32.07	20.86	25.28	51.68	17.58	26.85	21.25	68.55
CAE-Ensemble	63.68	58.76	65.41	81.71	52.94	67.99	51.26	34.15	21.47	26.37	58.26	18.16	45.98	26.04	82.16
D3R	63.93	59.61	66.13	80.62	53.09	66.83	51.83	39.10	20.77	27.13	60.11	17.85	50.17	26.33	80.68
IGCL	**65.75**	**61.92**	**68.92**	**82.39**	**54.02**	**69.97**	**55.89**	36.43	31.04	**33.52**	**69.82**	17.62	59.07	**27.15**	**84.00**

D Regularization loss

$$
\begin{aligned}
&KL\Big(N(\mu,\sigma^2)\Big\|N(\mu,I)\Big)\\
=&\int \frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/2\sigma^2}\left(\log\frac{e^{-(x-\mu)^2/2\sigma^2}/\sqrt{2\pi\sigma^2}}{e^{-(x-\mu)^2/2}/\sqrt{2\pi}}\right)dx\\
=&\int \frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/2\sigma^2}\log\left\{\frac{1}{\sqrt{\sigma^2}}\exp\left\{\frac{1}{2}(x-\mu)^2(1-\frac{1}{\sigma^2})\right\}\right\}dx \qquad \text{(C.21)}\\
=&\frac{1}{2}\int \frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/2\sigma^2}\Big[-\log\sigma^2+(x-\mu)^2(1-\frac{1}{\sigma^2})\Big]dx\\
=&\frac{1}{2}\Big(-\log\sigma^2+\sigma^2(1-\frac{1}{\sigma^2})\Big)=\frac{1}{2}\Big(-\log\sigma^2+\sigma^2-1\Big).
\end{aligned}
$$

E Experimental details

E.1 Datasets

- SMD [6] (Server Machine Dataset) is collected from a large computing cluster, where different server machines stack accessed traces of resource utilization with 38 dimensions.
- PSM [5] (Pooled Server Metrics) is a 25-dimension dataset collected by eBay that shows the performance of multiple application servers.
- MSL [35] (Mars Science Laboratory Rover) is from the spacecraft monitoring systems and collected by NASA. It shows the health check-up data of the sensors from the Mars rover.
- SMAP [35] (Soil Moisture Active Passive) is a 25-dimension dataset collected by NASA, which contains soil samples and telemetry information.
- SWAN [7] is extracted from solar photospheric vector magnetograms in the Spaceweather, which is the benchmark dataset used in NeurIPS Competition Track.
- SWaT [36] (Secure Water Treatment) is collected from sensors of the critical infrastructure systems under continuous operations for secure water treatment.
- GECCO [7] is a 9-dimension dataset collected from the cyber-physical systems showing the drinking water quality.
- UCR [34] is a 1-dimension dataset containing 250 sub-datasets from various domains.
- Credit [21] is from the finance scenario where the anomaly rate is only 0.172%.

These datasets are representative open benchmarks for multi-variable and uni-variable time series anomaly tasks, and thus we evaluate the unsupervised time series anomaly prediction problem on these datasets. More statistical information and details can be seen in Table C.9, where N is the number of variables observed at each timestamp. We follow the same training-validation-test splits as in the original papers [5–7, 22, 35, 36] shown in each column.

Table C.11: Anomaly prediction for irregular time series

Dataset	SWaT											
Dropping ratio	0%				30%				50%			
Metric	P	R	F1	AUC	P	R	F1	AUC	P	R	F1	AUC
PAD	74.83	59.09	66.04	61.35	74.17	57.80	64.97	60.71	69.29	50.94	58.71	56.09
IGCL+	84.98	63.54	**72.71**	**82.39**	84.02	59.97	**69.98**	**78.63**	70.81	53.77	**61.13**	**60.17**

E.2 Settings

We employ the official open-source implementations of baseline methods and have carefully tuned their hyper-parameters based on the recommendations from the original papers. The target label from the look-forward window is modified for the anomaly prediction task following PAD [10]. The anomaly score from the current look-back window is used for identifying the anomaly precursors to predict future anomalies also following PAD [10]. Our codes are at https://github.com/zhkai/IGCL. The experiments are conducted on an Ubuntu 18.04.5 LTS system server, with Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz and NVIDIA Quadro RTX 8000 GPU. All the deep learning-based models are executed with Pytorch 1.2.0. The size of the default look-forward window size f is 4, and we also evaluate with larger look-forward windows of $\{4, 8, 12\}$. We use Adam optimizer with the default parameter and the learning rate is default 0.0001. The batch size is tuned from $\{32, 64\}$. The look-back window size h and the number of sub-sequences b are tuned from $\{8, 16, 32, 64\}$. The memory bank size K is tuned from $\{16, 20, 24\}$ and we evaluate with more sizes in the parameter study. The number of positive samples P is tuned from $\{12, 16, 20\}$. The λ which controls the strength of regularization is tuned from $\{0.5, 1\}$. We use three parallel kernel sizes $\{2, 3, 5\}$ for our overlapping TCN. Following the existing work [21], we use the same strategy to choose the threshold for all methods.

E.3 More experiment results

Table C.10 shows more experiment results of most state-of-the-art methods on all datasets using more metrics.

E.4 Irregular time series

PAD uses interpolation and neural controlled differential equations (NCDE) to support irregular time series. The interpolation method can be applied before any models. Our TCN backbone can also be replaced with NCDE. We show the experimental results on the irregular series by dropping random points following PAD. We only add the interpolation method before our model, which is denoted as IGCL+. We can see that our method can be extended to irregular series easily and still perform well.

References

[1] X. Wu, D. Zhang, M. Zhang, C. Guo, B. Yang, and C. S. Jensen, “AutoCTS+: Joint neural architecture and hyperparameter search for correlated time series forecasting,” in *Proc. ACM Manag. Data*, 2023.

[2] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen, “LightTS: Lightweight time series classification with adaptive ensemble distillation,” *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 171:1–171:27, 2023.

[3] Q. Liu, P. Boniol, T. Palpanas, and J. Paparrizos, “Time-series anomaly detection: Overview and new trends,” *Proc. VLDB Endow.*, vol. 17, no. 12, pp. 4229–4232, 2024.

[4] A. Zhang, S. Deng, D. Cui, Y. Yuan, and G. Wang, “An experimental evaluation of anomaly detection in time series,” in *Proc. VLDB Endow.*, vol. 17, no. 3, 2023, pp. 483–496.

[5] A. Abdulaal, Z. Liu, and T. Lancewicki, “Practical approach to asynchronous multivariate time series anomaly detection and localization,” in *SIGKDD*, 2021, pp. 2485–2494.

[6] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, “Robust anomaly detection for multivariate time series through stochastic recurrent neural network,” in *SIGKDD*, 2019, pp. 2828–2837.

[7] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, and X. Hu, "Revisiting time series outlier detection: Definitions and benchmarks," in *NeurIPS*, 2021.

[8] R. Cirstea, C. Guo, and B. Yang, "Graph attention recurrent neural networks for correlated time series forecasting - full version," *CoRR*, vol. abs/2103.10760, 2021.

[9] R. Ghorbani, M. J. T. Reinders, and D. M. J. Tax, "PATE: proximity-aware time series anomaly evaluation," in *SIGKDD*, 2024, pp. 872–883.

[10] S. Y. Jhin, J. Lee, and N. Park, "Precursor-of-anomaly detection for irregular time series," in *SIGKDD*, 2023, pp. 917–929.

[11] V. Cerqueira, L. Torgo, and C. Soares, "Early anomaly detection in time series: a hierarchical approach for predicting critical health episodes," *Mach. Learn.*, vol. 112, no. 11, pp. 4409–4430, 2023.

[12] P. Boniol, M. Meftah, E. Remy, B. Didier, and T. Palpanas, "dcnn/dcam: anomaly precursors discovery in multivariate time series with deep convolutional neural networks," *Data-Centric Engineering*, vol. 4, 2023.

[13] H. Hojjati, M. Sadeghi, and N. Armanfard, "Multivariate time-series anomaly detection with temporal self-supervision and graphs: Application to vehicle failure prediction," in *PKDD*, vol. 14175, 2023.

[14] D. Xu, W. Cheng, J. Ni, D. Luo, M. Natsumeda, D. Song, B. Zong, H. Chen, and X. Zhang, "Deep multi-instance contrastive learning with dual attention for anomaly precursor detection," in *SIAM SDM*, 2021.

[15] Y. Ang, Q. Huang, A. Tung, and Z. Huang, "A stitch in time saves nine: Enabling early anomaly detection with correlation analysis," in *ICDE*, 2023, pp. 132–145.

[16] Y. Ang, Q. Huang, A. K. Tung, and Z. Huang, "Eads: An early anomaly detection system for sensor-based multivariate time series," in *ICDE*, 2024, pp. 5433–5436.

[17] A. Nina, "Analysis of VLF signal noise changes in the time domain and excitations/attenuations of short-period waves in the frequency domain as potential earthquake precursors," *Remote. Sens.*, vol. 16, no. 2, 2024.

[18] S. Schmidl, F. Naumann, and T. Papenbrock, "Autotsad: Unsupervised holistic anomaly detection for time series data," in *Proc. VLDB Endow.*, vol. 17, no. 4, 2024, pp. 766–779.

[19] R. Wang, X. Mou, R. Yang, K. Gao, P. Liu, C. Liu, T. Wo, and X. Liu, "Cutaddpaste: Time series anomaly detection by exploiting abnormal knowledge," in *SIGKDD*, 2024, pp. 3176 – 3187.

[20] X. He, Y. Li, J. Tan, B. Wu, and F. Li, "Oneshotstl: One-shot seasonal-trend decomposition for online time series anomaly detection and forecasting," in *Proc. VLDB Endow.*, vol. 16, no. 5, 2023, pp. 700–712.

[21] Y. Zhao, L. Deng, X. Chen, C. Guo, B. Yang, T. Kieu, F. Huang, T. B. Pedersen, K. Zheng, and C. S. Jensen, "A comparative study on unsupervised anomaly detection for time series: Experiments and analysis," *CoRR*, vol. abs/2209.04635, 2022.

[22] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen, "Unsupervised time series outlier detection with diversity-driven convolutional ensembles," in *Proc. VLDB Endow.*, vol. 15, no. 3, 2022, pp. 611–623.

[23] T. Kieu, B. Yang, C. Guo, C. S. Jensen, Y. Zhao, F. Huang, and K. Zheng, "Robust and explainable autoencoders for unsupervised time series outlier detection," in *ICDE*, 2022, pp. 3038–3050.

[24] Y. Li, W. Chen, B. Chen, D. Wang, L. Tian, and M. Zhou, "Prototype-oriented unsupervised anomaly detection for multivariate time series," in *ICML*, vol. 202, 2023, pp. 19 407–19 424.

[25] A. S. Iyer, J. S. Nath, and S. Sarawagi, "Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection," in *ICML*, vol. 32, 2014, pp. 530–538.

[26] L. Kohs, B. Alt, and H. Koeppl, “Markov chain monte carlo for continuous-time switching dynamical systems,” in *ICML*, 2022.

[27] N. Engelmann and H. Koeppl, “Forward-backward latent state inference for hidden continuous-time semi-markov chains,” in *NeurIPS*, 2022.

[28] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” *arXiv/1607.08022*, 2016.

[29] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, “Reversible instance normalization for accurate time-series forecasting against distribution shift,” in *ICLR*, 2021.

[30] C. Xiao, Z. Gou, W. Tai, K. Zhang, and F. Zhou, “Imputation-based time-series anomaly detection with conditional weight-incremental diffusion models,” in *SIGKDD*, 2023, pp. 2742–2751.

[31] C. Wang, Z. Zhuang, Q. Qi, J. Wang, X. Wang, H. Sun, and J. Liao, “Drift doesn’t matter: Dynamic decomposition with diffusion reconstruction for unstable multivariate time series anomaly detection,” in *NeurIPS*, 2023.

[32] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *ICLR*, 2014.

[33] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.

[34] R. Wu and E. J. Keogh, “Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress,” in *ICDE*, 2022, pp. 1479–1480.

[35] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding,” in *SIGKDD*, 2018, pp. 387–395.

[36] J. Xu, H. Wu, J. Wang, and M. Long, “Anomaly transformer: Time series anomaly detection with association discrepancy,” in *ICLR*, 2022.

[37] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, “Deep autoencoding gaussian mixture model for unsupervised anomaly detection,” in *ICLR*, 2018.

[38] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al, “K-means-based isolation forest,” *Knowledge-based systems*, vol. 195, p. 105659, 2020.

[39] L. Ruff, R. A. Vandermeulen, N. G"ornitz, L. Deecke, S. A. Siddiqui, A. Binder, E. M"uller, and M. Kloft, “Deep one-class classification,” in *ICML*, vol. 80, 2018, pp. 4393–4402.

[40] L. Shen, Z. Li, and J. Kwok, “Timeseries anomaly detection using temporal hierarchical one-class network,” *NeurIPS*, 2020.

[41] Y. Yang, C. Zhang, T. Zhou, Q. Wen, and L. Sun, “Dcdetector: Dual attention contrastive representation learning for time series anomaly detection,” in *SIGKDD*, 2023, pp. 3033–3045.

[42] T. Kieu, B. Yang, C. Guo, R. Cirstea, Y. Zhao, Y. Song, and C. S. Jensen, “Anomaly detection in time series with robust variational quasi-recurrent autoencoders,” in *ICDE*, 2022, pp. 1342–1354.

[43] B. Du, X. Sun, J. Ye, K. Cheng, J. Wang, and L. Sun, “Gan-based anomaly detection for multivariate time series using polluted training set,” *Trans. Knowl. Data Eng.*, vol. 35, no. 12, 2023.

[44] J. Paparrizos, P. Boniol, T. Palpanas, R. Tsay, A. J. Elmore, and M. J. Franklin, “Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection,” *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2774–2787, 2022.

[45] A. Huet, J. M. Navarro, and D. Rossi, “Local evaluation of time series anomaly detection algorithms,” in *SIGKDD*, 2022, pp. 635–645.

[46] Y. Sun, G. Pang, G. Ye, T. Chen, X. Hu, and H. Yin, “Unraveling the ’anomaly’ in time series anomaly detection: A self-supervised tri-domain solution,” in *ICDE*, 2024, pp. 981–994.

[47] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: identifying density-based local outliers,” in *SIGMOD*, 2000.

[48] Y. Chen, C. Zhang, M. Ma, Y. Liu, R. Ding, B. Li, S. He, S. Rajmohan, Q. Lin, and D. Zhang, "Imdiffusion: Imputed diffusion models for multivariate time series anomaly detection," in *Proc. VLDB Endow.*, vol. 16, no. 4, 2023, pp. 766–779.

[49] Y. Shin, S. Lee, S. Tariq, M. S. Lee, O. Jung, D. Chung, and S. S. Woo, "ITAD: integrative tensor-based anomaly detection system for reducing false positives of satellite systems," in *CIKM*, 2020.

[50] Y. Fang, J. Xie, Y. Zhao, L. Chen, Y. Gao, and K. Zheng, "Temporal-frequency masked autoencoders for time series anomaly detection," in *ICDE*, 2024, pp. 1228–1241.

[51] X. Hao, Y. Chen, C. Yang, Z. Du, C. Ma, C. Wu, and X. Meng, "From chaos to clarity: Time series anomaly detection in astronomical observations," in *ICDE*, 2024, pp. 570–583.

[52] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, and D. Pei, "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *SIGKDD*, 2021, pp. 3220–3230.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[54] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, vol. 119, 2020.