AI-EMPOWERED FRAMEWORKS FOR FAULT DETECTION IN TRAFFIC SENSOR NETWORKS

by

YONGCAN HUANG

(Under the Direction of Jidong J. Yang)

ABSTRACT

In today's rapidly digitalizing world, the integration of diverse sensors has become increasingly prevalent. Within the transportation sector, state Departments of Transportation have extensively deployed traffic sensors to support a wide range of engineering applications. Accurate and reliable sensor data is crucial for efficiently monitoring and managing large-scale transportation networks. However, these sensors inevitably suffer from issues such as data loss, random noise, biases, and drift, often caused by sensor aging, defects, or environmental factors. Therefore, detecting these faults is imperative to maintain data integrity. This dissertation introduces two distinct deep learning frameworks designed to enhance traffic sensor data quality, with a focus on fault detection. The first framework evaluates data from individual sensor stations, while the second incorporates geospatial context by considering spatiotemporal correlations among neighboring stations, resulting in improved fault detection accuracy. In contrast, the first framework is context-insensitive, requiring less data as it analyzes data from individual sensors, whereas the second framework, which integrates contextual information, demands more data. Particularly the first framework leverages symmetric contrastive learning within a triplet network architecture, enhanced by a cross-attention loss function to improve fault detection. Continuous

Wavelet Transformation (CWT) is first applied to convert traffic data into time-frequency wavelet images, which are used to pretrain a triplet encoder. A novel symmetric contrastive sampling strategy is employed to improve training efficiency by using a normal day's data as an anchor, from which both positive and negative examples are generated based on domain knowledge. This approach strengthens contrastive signals, enabling faster and more stable training. The second framework leverages graph neural networks (GNNs) to capture spatial dependencies within clusters of sensor stations. These clusters are formed in a reduced-dimensional latent space, constructed using a dual-encoding attention graph auto-encoder (DAGAE) that embeds both node and edge features. A cluster-guided denoising graph auto-encoder (CG-DGAE) is then trained using subgraphs generated from these clusters to reconstruct traffic data from corrupted inputs. A fault score function is then applied to compare observed and reconstructed data sequences, identifying discrepancies indicative of sensor faults. Together, these frameworks provide intelligent and practical solutions for fault detection and data quality control. Their implementation has the potential to transform the maintenance and operation of transportation systems, contributing to more reliable and resilient infrastructure.

INDEX WORDS: Sensor Data Quality Control, Sensor Fault Detection, Contrastive Learning, Triplet Network, Data Imputation and Reconstruction, Graph Auto-Encoder, Graph Representation Learning, Intelligent Transportation Systems

AI-EMPOWERED FRAMEWORKS FOR FAULT DETECTION IN TRAFFIC SENSOR NETWORKS

by

YONGCAN HUANG

B.S., Wuhan Institute of Technology, China, 2017

M.S., Changsha University of Science and Technology, China, 2019

A Dissertation Submitted to the Graduate Faculty of the University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2024

AI-EMPOWERED FRAMEWORKS FOR FAULT DETECTION IN TRAFFIC SENSOR NETWORKS

by

YONGCAN HUANG

Major Professor:	Jidong J. Yang
Committee:	Sung-Hee Kim
	Stephan Durham
	Mi Geum Chorzepa

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
December 2024

DEDICATION

I dedicate this dissertation to the memory of my paternal grandmother, Mrs. Bangxiu Li, whose teachings of kindness and integrity have profoundly shaped me.

With deep gratitude, I also dedicate this work to my mother, Rong Dai, and my father, Debin Huang, who have been my unwavering support, no matter where in the world I've found myself.

To my extended family and friends: over the past thirty years, I have embarked on a journey of self-discovery, much like the protagonist in *The Little Prince*. Each of you has enriched my life, helping me understand the beauty of connection and shared experiences.

A special acknowledgment goes to my childhood friends, Yue Ma and Banghua Du, for their constant and steadfast spiritual support.

Lastly, I want to express my appreciation for JD Vance's *Hillbilly Elegy*, a book and film that planted a seed of the American dream within me when I first encountered it in Tokyo, Japan, in 2018.

ACKNOWLEDGEMENTS

The past four years have flown by, and it is now time to express my deep gratitude to those who have supported me throughout this journey toward earning a Doctor of Philosophy.

First and foremost, I would like to extend my heartfelt thanks to my advisor, Dr. Jidong Yang. I couldn't have asked for a better mentor. Thank you for making the decision four years ago to accept me into your group. Your patience and passion have provided me with invaluable guidance. Thanks to your advisement, I am now graduating with multiple job interviews lined up, publications in top journals, and most importantly, the realization of my dream of becoming a scholar. I will carry forward your spirit, qualities, and work ethic in my future endeavors as a part of your lasting legacy. I am deeply grateful to you.

I would like to express my sincere gratitude to my committee members, Dr. Stephan Durham, Dr. Sung-Hee Kim, and Dr. Mi Geum Chorzepa, for their invaluable advisement and guidance throughout my doctoral journey. Thank you for agreeing to serve on my committee and for your continuous support, insightful feedback, and encouragement. Your expertise has greatly contributed to the development of my dissertation, and your mentorship has played an essential role in shaping my academic growth. I truly appreciate the time and effort you have dedicated to helping me succeed.

I would also like to extend my gratitude to Hao Zhen, who has been an exceptional research partner. Our teamwork has been outstanding, and she has provided me with immense support both

academically and personally. Additionally, I would like to thank my lab mates—Shihan, Yunxiang, Jialun, and Penghao—for the great times and camaraderie we've shared.

A special thanks to my family, who have been my unwavering support, always there to catch me when I faltered. I would also like to acknowledge my exes for their companionship during this period, as well as my beloved pets: Bailey, the English Cream Golden Retriever, and Sandy, the American Shorthair cat, who brightened my days. Lastly, I cannot hide my affection for the University of Georgia, where I received the best education of my life. Go Dawgs!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Against the backdrop of escalating population growth, urbanization trends, and the pervasive influence of digitalization, the last few decades have witnessed a rapid proliferation of Intelligent Transportation Systems (ITS). These systems incorporate a plethora of technologies including fixed and mobile sensing, Internet of Things (IoT), and connected and autonomous vehicles (CAVs). Within the realm of transportation management, ITS stands as a formidable platform aimed at enhancing flow efficiency, alleviating congestion, proactively addressing incidents, and ameliorating environmental impacts. The voluminous data generated by ITS serves as the cornerstone for various system functionalities, heavily reliant on sophisticated data analytics.

At the state level, high-quality traffic data plays a pivotal role in infrastructure-related decision making and serves as the foundation for various planning and engineering practices at the state Department of Transportation (DOT). Traffic count and classification data are continuously collected at permanent continuous count stations (CCS) using inductive loop technology. This data source has been used for multiple purposes, including generating annual Highway Performance Monitoring System (HPMS) report, providing data support for internal departments (e.g., Office of Planning, Office of Roadway Design, Office of Bridge Design, Office of Materials and Testing, and Office of Traffic Operations) as well as to a large variety of external customers, and supplying

critical traffic data to the Georgia Emergency Management Agency and surrounding states during emergency evacuations in inclement weather.

However, collecting traffic sensor data in real world is prone to errors due to various factors, such as sensor malfunctions, harsh environmental conditions, inappropriate installation, and maintenance, among others [1]. Erroneous traffic data can result in misleading analytical outcomes, leading to significant socio-economic losses. To address this issue, many state Departments of Transportation (DOTs) have implemented automatic review systems with definitive quality control (QC) rules to eliminate obvious faulty data, such as nonreporting of particular classes of vehicles for an extended period. In case of uncertainty, manual review by a human is still necessary for further data screening. These QC rules typically rely on threshold-based plausibility tests, which are not sensitive enough to detect all faulty signals and are applied to individual count stations rather than considering them collectively as a group. This could easily lead to false-positive or false-negative decisions as part of the QC process. In this context, this dissertation aims to develop a robust automatic quality control system, specifically deep learning frameworks, designed to filter out incoming faulty traffic sequences and identify and rectify any faults when necessary.

1.1. Motivation

Traffic sensor data is inherently temporal with patterns varying by time of day, day of the week, seasons, and around special events. AI models are adept at recognizing these temporal patterns and can discern anomalies that deviate from these patterns. Several studies have focused on learning time-series data representations using different neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), autoencoders (AE),

generative adversarial networks (GANs) and deep belief networks (DBNs). On the other hand, time series imaging is an advanced analytical approach that captures a sequence of images or data points at successive times. This technique is typically used to monitor and track changes that occur in a given subject or area over a period. Leveraging the frequent capture of data and the specific characteristics of the monitored phenomena, numerous specialists have harnessed time series imaging for the detection of faults or anomalies in sensor data, particularly due to the enhanced capabilities provided by deep learning models. For instance, Xu et al. (2021) proposed a hybrid deep learning model that combines the feature extraction capabilities of CNNs and the superior performance of deep forest classifier. The model extracts feature from CWT images of bearing vibration signals using CNNs and trains the classifier with a cascade forest strategy [2]. Song et al. (2023) introduced an innovative technique for the detection of manufacturing faults by merging image-encoded time series data with advanced deep generative models. The study utilized three distinctive methods for converting time series into visual formats: the Gramian Angular Difference Field (GADF), the Markov Transition Field (MTF), and the Recurrence Plot (RP). To process and learn from this data, two types of neural network models were employed: the Variational Autoencoder-Reconstruction along Projection Pathway (VAE-RaPP) and the Fence Generative Adversarial Network (Fence GAN) [3]. Morris et al. (2022) introduced an approach based on variational autoencoder (VAE) and trained two VAEs to separately encode CWT images and recurrent plots derived from traffic sensor data. In the joint latent space, anomalies are identified when they deviate significantly from the constructed manifold [4]. In another study of achieving the sensor faults due to harsh operational conditions in the field of the internet of things (IoT), Hasan et al. (2023) presented an innovative digital twin (DT)-inspired fault detection methodology utilizing a generative adversarial network (GAN) trained on Gramian Angular Field (GAF)-

encoded images to maintain the time series data's temporal integrity, achieving an impressive 98.7% accuracy in identifying sensor anomalies [5]. Xin et al. (2022) utilized the training efficiency of Deep Belief Networks (DBNs) to create a modified Gaussian Convolutional Deep Belief Network (MGCDBN) for fault diagnosis in infrared thermal images. The network employs Gaussian units to model real-valued inputs using a Gaussian distribution in both its visible and hidden layers [6]. Employing time series imaging enhanced by deep learning techniques presents a promising avenue for the detection of faults within such data.

In the realm of traffic management, the intricate interplay of spatial and temporal dimensions within sensor data holds paramount importance. The spatial-temporal correlation acknowledges that traffic dynamics at a given location are not isolated events but are significantly influenced by concurrent traffic flows in adjoining areas. Similarly, the temporal aspect underlines the evolution of traffic patterns over time, a key to predicting future conditions. As urban populations burgeon and road networks become increasingly congested, the need to comprehend and harness these correlations has never been more critical. Leveraging the spatial-temporal correlation and analyzing the traffic data collectively would greatly benefit capturing traffic data features and data mining for traffic data fault detection. Graph Neural Networks (GNNs), as a key branch of deep learning family, have emerged as a powerful deep learning approach for handling non-Euclidean data through graph analysis techniques. In the realm of traffic research, GNNs have found extensive applications, including traffic flow/speed forecasting, traffic prediction, and traffic accident prediction [7], [8], [9], [10]. These applications leverage GNNs' ability to capture complex spatial-temporal relationships in traffic networks, which is crucial for tasks such as fault detection.

1.2. Purpose of the Study

In this dissertation, the primary objective is to propose an optimal automatic data quality review framework with fault detection focus for traffic sensor networks. Specifically, the study introduces two distinct approaches: one that uses traffic time-series image-based deep learning techniques to analyze data from individual sensor stations without considering spatial context, and another that leverages spatiotemporal correlations among neighboring sensors using Graph Neural Networks (GNNs). These frameworks aim to enhance the accuracy and robustness of fault detection, reduce erroneous data, improve the quality control process, and ultimately support more effective transportation management and planning. The research questions in this study are as follow:

1. How can deep learning models be utilized to detect and correct faults in traffic sensor data from individual stations without the need for spatial context?

- What types of traffic data anomalies can be identified using this framework?
- How does the proposed framework's fault detection accuracy compare to traditional threshold-based plausibility tests?

2. How can the integration of spatiotemporal correlations using GNNs improve fault detection in traffic sensor data across multiple stations?

- To what extent does incorporating geospatial context enhance fault detection performance?
- How effectively do GNN-based models detect faults in comparison to models that analyze sensors individually?

3. What are the trade-offs between the two proposed frameworks in terms of fault detection accuracy, data requirements, and computational efficiency?

- Which framework performs better in scenarios where data is limited or noisy?
- How do these frameworks perform across different traffic patterns, such as during special events or extreme weather conditions?

1.3. Open Challenges

Realizing a comprehensive automatic fault detection framework is laden with challenges that must be surmounted.

First, establishing a framework capable of accurately identifying faulty traffic sequences involves the fundamental data quality control task akin to a binary classification problem, differentiating between faulty and normal data. However, the main challenge lies in proper construction of feature space for the classification task. While deep neural networks (DNNs) have shown impressive accuracy in detecting faults on time-series data, many of these methods either require labeled data or generate their own supervision signals to capture underlying patterns. Additionally, these methods often assume that the feature distribution of the training and testing data are similar, which is not always the case in real-world scenarios where variations in sensor operating conditions, degradation state, or environmental noise can affect the data. Consequently, the distribution discrepancy between the training and testing data can significantly impair the performance of deep learning models in detecting faults in sensor data.

Second, refining a framework that not only pinpoints faults but also corrects them with valid replacements is further complicated by the intrinsic nature of road traffic and the voluminous aggregation of traffic data. The challenge is compounded by the following facts we outlined concerning the intrinsic characteristics of road traffic and massive accumulation of traffic flow data resources.

1) Random occurrence of faulty traffic data. Faulty in-road sensor data can be categorized into recurring and non-recurring types. Recurring faults, which may stem from external factors (e.g., connectivity issues, hardware malfunctions, low battery, environmental extremes, and clipping) depending on sensor types, exhibit short-term cyclic patterns. In contrast, non-recuring faults may arise from unexpected events such as sudden changes in operating conditions or electrical interference, occurring randomly and infrequently at any time of the day. Detecting the latter type is particularly challenging.

2) Lack of reported faulty traffic data. The faulty in-road sensor data is usually not fully documented or critical information is lacking when flagged due to the limitations of currently adopted QC rules. There is no reliable ground truth for measuring the accuracy and feasibility of these QC rules.

3) Heterogeneity of traffic data. There could be thousands of in-road sensors spreading over the large roadway network within a state. Each sensor operates independently, and the occurrence of faulty data is less temporally dependent, unlike traffic anomalies. For example, a road section undergoing construction may gradually decrease traffic flow, resulting in anomalies in traffic volume data. These anomalies may persist over the extended construction period. However, a communication failure of a sensor may result in continuous non-responses, lasting for varying durations depending on how quickly the failure is detected and repaired. Additionally, traffic data characteristics recorded at different sensors may vary significantly by their locations and operating conditions. Consequently, a significant challenge associated

with in-road sensor networks is establishing consistent fault detection rules across the entire network [11].

4) Spatial-temporal dependency of traffic data. Given a specific roadway network topology, traffic flow data at adjacent sensors are inherently correlated and thus present unique spatial-temporal patterns. Some examples of similar traffic patterns observed at neighboring continuous count station (CCS) sites are shown in **Figure 1.1**. Leveraging the spatial-temporal correlation and collectively analyzing the traffic data from the correlated sensors would help to capture comprehensive traffic patterns at the network level, leading to improved fault detection. However, obtaining sufficiently large common periods of traffic data from all sensors poses a challenge for real-world, large-scale road networks.

1.4. Dissertation Overview

In this chapter, the background of traffic sensor quality control, summarize existing technologies, and highlight the research gaps that motivate this study were introduced. At the same time, the open challenges were outlined. The remainder of this dissertation is structured as follows: Chapter 2 reviewed key technologies and topics, discussing the limitations of current research. Chapters 3 and 4 presented the design of the two proposed frameworks, along with related datasets, experiments, and an analysis of the results. Finally, Chapter 5 summarized the key findings, draws conclusions, and suggested potential future research directions. References were provided after Chapter 5. To have a better understanding of this dissertation path, an overview is presented to guide the reader into the main content of each chapter as shown in **Table 1.1**.

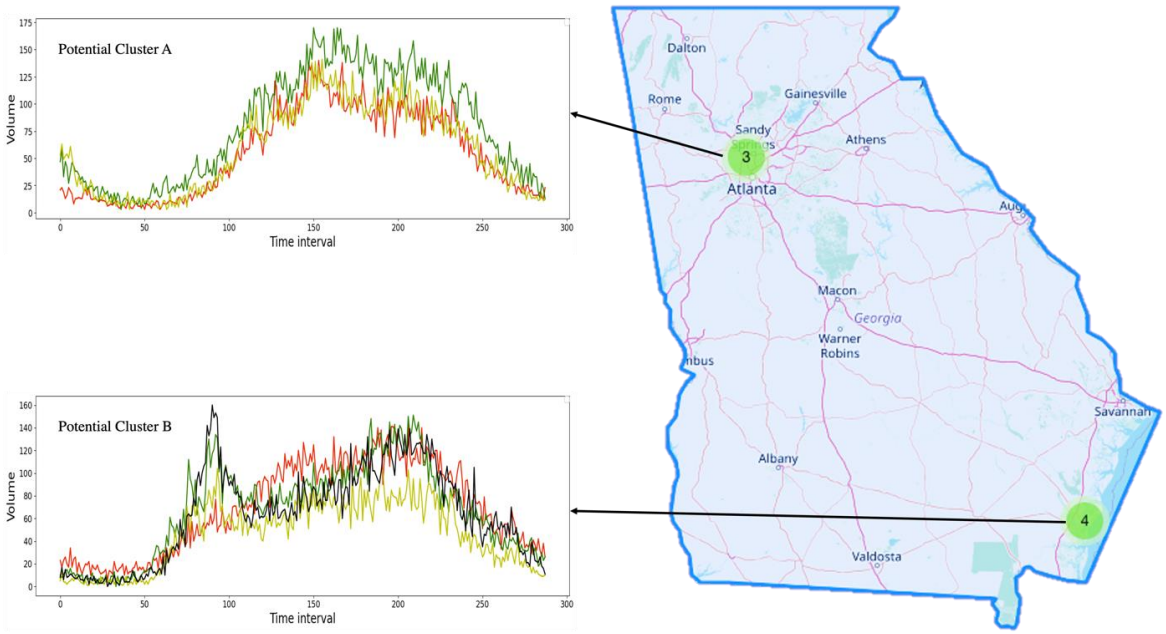

Figure 1.1:Similar traffic patterns observed at neighboring CCS sites.

Table 1.1: Overview of the chapters in this dissertation

Chapter	Content description
Chapter 1	Introduction to the background of traffic sensor data quality control, research motivation, purposes, research questions and open challenges of this dissertation.
Chapter 2	Review of literature and discussion of the current research gaps.
Chapter 3	Details of the symmetric contrastive learning-based framework for traffic sensor data fault detection at individual level, results, and conclusion.
Chapter 4	Details of the framework of cluster-guided denoising graph auto-encoder for enhanced traffic data imputation and fault detection at a cluster level, results, and conclusion.
Chapter 5	Summary of the findings and conclusions generated in this dissertation with a possible future direction related to this dissertation.
Reference	References that were used in this dissertation.
Appendix	Some algorithm and extensive experimental results.

CHAPTER 2

LITERATURE REVIEW

In this chapter, the existing works closely related to this dissertation were reviewed. First, the continuous wavelet transform (CWT) and its application in detecting faults in time-series data were introduced. Next, the literature of deep learning models for fault detection through the approaches of reconstruction and prediction were reviewed. Finally, the application of graph neural networks (GNNs) was reviewed, which leverages spatio-temporal correlations in traffic data.

2.1. CWT in Fault Detection

Continuous wavelets transform (CWT) is well known for decomposing time-frequency information, particularly, constructive to obtain salient features from dynamic time series data. CWT-based traffic data transformation has been shown to reveal unobvious patterns of traffic data in an efficient way [12].

Traditionally, CWT has been used to capture local changes, which are noisy and aperiodic. For example, Zheng et al. [13] demonstrated the utility of wavelet transform in analyzing important features associated with abnormal traffic conditions, such as bottleneck effects and traffic oscillation arising from congestion. The case study of three different scenarios of vehicle trajectories showed that the origins of deceleration waves could be detected by wavelet-based energies of a single vehicle, and the detected origins help to pinpoint possible causes. In another

study, Jiang et al. [14] developed a two-stage fault detection method for anomalous network traffic. In their methodology, CWT was applied to decompose the incoming signals into multiple continuous scales, followed by principal component analysis to extract the features of anomalous network traffic. Then, a new mapping function is constructed to detect the abnormal traffic.

Recently, CWT coupled with deep learning techniques offers a new approach for fault detection with time-series data. König et al. [15] proposed a deep learning-based method for anomaly detection and diagnosis on acoustic emission signals. With the acoustic emission signals being converted to CWT images, an autoencoder network was developed for anomaly detection in the latent space and GoogLeNet was adapted to the anomaly classification task. In another study, Jalayer et al. [16] developed a comprehensive deep learning-based fault detection and diagnosis model for rotating machinery by channeling up fast Fourier transform, CWT, and statistical features of raw signals. A convolutional long short-term memory was employed to classify the multi-channel input. Djaballah et al. [17] explored an innovative approach to bearing fault diagnosis by leveraging deep transfer learning methods. The study investigates various pre-trained convolutional neural networks (CNN), such as ResNet-50, GoogLeNet, and SqueezeNet, in combination with transfer learning to diagnose bearing faults from vibration signals transformed into time-frequency images using CWT. The methodology integrates fine-tuning strategies to optimize fault classification performance, achieving high accuracy, with their approach being validated on the CWRU dataset.

Based on the review of previous studies, CWT has been commonly used for processing time-series data. Generally, the methodology of converting signals into CWT images representations and further processing these image representations by deep-learning-based

methods to encode multiscale features has shown a great potential and improved performance in fault detection of time-series data.

2.2. Deep Learning-Based Anomaly Detection in Traffic Data

In the field of traffic sensor data anomaly detection, significant advancements have been accomplished by utilizing deep learning techniques, particularly in the analysis of traffic flow data. The primary approaches to anomaly detection include prediction-driven and reconstruction-driven methodologies.

The prediction-driven approach employs historical traffic data to train deep neural network models for forecasting future traffic states or conditions. Any significant deviation between the model's predictions and the actual observations triggers an anomaly detection, signaling a potential fault in the system. Predicting traffic flow has historically been and continues to be a formidable challenge. Traditional algorithms like Autoregressive Integrated Moving Average (ARIMA)-based method [18], probabilistic models, such as Bayesian Network [19], Markov Chain [20], and Markov Random Fields [21] and machine learning approaches like shallow Artificial Neural Networks, and Support Vector Regression (SVR) [22], have been studied for traffic flow estimation. However, these conventional methods can hardly capture the complex patterns underlying the data. Over the years, various approaches have been explored alongside the rapid development of various deep learning methods. Different deep learning models have been studied by researchers, such as Deep Belief Network (DBN) [23], Long short-term memory (LSTM) [24], Stacked Autoencoder (SAE) [25]. Nevertheless, the majority of these previous works treated traffic prediction as simple time series prediction problem focusing on isolated sensor data. Subsequently, some researchers have recognized the importance of leveraging the spatial-temporal

relationships inherent in multi-sensor data to enhance traffic prediction performance. This led to the development of techniques, such as Convolutional LSTM [26] and Graph Convolution Gated Recurrent Unit [27], which can naturally handle multi-sensor data and capture the spatial-temporal relationships among sensors [28]. One disadvantage of the prediction-driven approach is that it relies heavily on the accuracy of the forecasting model, which may struggle to capture sudden or rare events, leading to missed anomalies. Additionally, this method can be sensitive to changes in traffic patterns that do not necessarily indicate faults, resulting in false positives.

Missing data, often caused by sensor or system failures, is a prevalent issue that can detrimentally affect various traffic-related tasks. To mitigate the challenges posed by missing data, data reconstruction or imputation assumes critical importance. The reconstruction-driven approach involves compressing traffic data into a lower-dimensional vector space using methods like autoencoders, and subsequently reconstructing it to its original form. Consequently, substantial differences between the reconstructed data and the actual data indicate anomalies, suggesting faults or disruptions in traffic patterns. Traditional imputation methods include: ARIMA, KNN, principal component analysis (PCA)-based methods [29] and Bayesian imputation model [30]. Other state-of-the-art reconstruction methods include Recurrent Neural Networks (RNN)-based methods [31], deep sequential variational autoencoder based methods [25], generative adversarial networks (GAN)-based methods [32], and GNN-based methods [33].

On the other hand, researchers have pursued two primary approaches by examining individual sensors or collections of sensors. The distinction between these two approaches hinges on whether data from sensors exhibit any spatial and/or temporal dependence. Studies in the latter category often not only emphasize the temporal properties of isolated road sensor data but also underscore the importance of considering spatial correlations of data from different sensors.

Djenouri et al. (2019) classified existing anomaly detection techniques to three main categories: statistical, similarity-based, and pattern-based methods [34]. Many of these techniques are tailored for traffic anomaly analysis of individual sensors. Nevertheless, no matter is prediction-driven or reconstruction-driven approaches, a predominant trend in recent years is to leverage the spatial-temporal correlations among sensors to enhance the robustness for traffic data modeling [35], [36], [37], [38], which is also the focus and motivation of this study. With recent advancements in graph neural networks, researchers have found that GNN-based methods have proved excellent ability in capturing the explicitly spatial-temporal correlations [39]. Notably, for GNN-based spatial-temporal methods, dynamic subgraph-based GNN have recently been studied to empower the representation learning [28], which further improves the generalizability and performance of these models. Recently, Zhang et al. (2022) presented an automatic traffic anomaly detection method by leveraging spatial-temporal graph neural network for representation learning. They learned implicit graph features from multivariate time series of traffic flows and used a graph deviation score to detect traffic anomalies [40]. However, they do not consider the heterogeneity among multiple sensor data, which is crucial to consider especially in large sensor networks. Since sensor data patterns vary across different regions in terms of trend and magnitude, in our proposed traffic data fault detection framework at a cluster level, the sensors are grouped into smaller clusters with strong spatio-temporal correlations. This clustering approach guides the reconstruction model training, enhances the homogeneity within the embeddings, and helps mitigate the losses caused by heterogeneity, ultimately leading to significant improvements in performance.

2.3. Graph Neural Networks in Traffic Research

Graph Neural Networks (GNN), serving as an innovative deep learning approach tailored for handling non-Euclidean data through graph analysis techniques, have found extensive utility in a

range of data-driven applications within the realm of traffic research. These applications include but not limited to traffic flow forecasting [7], traffic speed prediction [8], traffic signal control [9] and traffic accident prediction [10].

Four groups of GNN have been widely applied in traffic research field, namely recurrent GNN (RecGNN), convolutional GNN (ConvGNN), graph autoencoders (GAE), and spatial-temporal GNN (ST-GNN) [41]. In the practice of RecGNN and ConvGNN, researchers typically use GNN to capture network-level spatial relations, along with RNN or CNN to extract temporal dependencies. To address the lack of flexibility in the local-feature extraction process in GNN, Cui et al. (2020) proposed a graph wavelet gated recurrent (GWGR) neural network to realize network-wide traffic forecasting with no need to specify the neighboring area in the graph topology, where graph wavelet is incorporated as a key component for extracting spatial features and a gated recurrent structure is employed to learn temporal dependencies in the sequence data [42]. As a response to the limitations of recurrent neural networks (RNN) in effectively capturing periodic temporal correlations, particularly in the context of gradient vanishing, Chen et al. (2019) combined the capabilities of graph convolutional networks, recurrent networks, and residual neural networks to jointly extract spatial-temporal features while considering external factors. The proposed Multiple Gated Recurrent Graph Neural Networks (MRes-RGNN) delivered the state-of-the-art results on traffic prediction at the time [43]. With a goal to forecast the origin-destination travel demand between regions, Wang et al. (2022) developed a GAE structured model that utilize the node representations in the latent space to capture the evolution of directed temporal networks. The innovative temporal graph autoencoder (TGAE) empowers the prediction of the link weight and direction based on the historical network snapshots [44].

In recent years, ST-GNN have garnered due attentions within traffic research area as they allow the concurrent modeling of spatial and temporal dependencies in dealing with a dynamic graph problem. In the realm of traffic data, these applications primarily tackle prediction and kriging challenges. For example, to address the limitations posed by incomplete adjacent connections that hinder the effective modeling of spatial-temporal dependencies in ST-GNN, Li and Zhu (2021) introduced a fusion graph module. This module operates on various temporal and spatial graphs concurrently for different time periods in parallel, allowing for the efficient learning of concealed spatial-temporal relationships for traffic flow prediction. The newly proposed spatial-temporal fusion graph neural networks (STFGNN or SFTGNN) exhibit the capability to handle long traffic flow sequences by harnessing stacked layers to learn more intricate spatial-temporal dependencies [45]. Kriging techniques have found extensive application in addressing traffic data imputation challenges. For instance, when dealing with the task of filling in distinct types of missing entries in spatial-temporal traffic data, Liang et al. (2022) incorporate both extracted spatial and temporal features as node representations. These representations are used as input for an architecture based on diffusion graph convolutional neural networks (DGCN) with a mask mechanism, facilitating the reconstruction of temporal node features. The proposed kriging model effectively fulfills all imputation requirements without the need to retrain the entire model [46].

Moreover, GNN training is renowned for encountering the neighbor explosion problem [47]. To alleviate this issue, subgraph-wise sampling has gained widespread acceptance in the field [48], [49], [50]. This approach aims to curtail the number of nodes involved in message passing, which, in turn, enhances computational efficiency, focuses on crucial nodes, addresses imbalanced data concerns, and mitigates noise. Nevertheless, hasty or ill-considered sampling can result in information loss or introduce biases during the training process. In previous works related to ST-

GNN in traffic research, some attention has been given to the concept of sampling tactics. Nonetheless, there is typically a lack of comprehensive logical instruction for the generation of subgraphs. For example, randomly sampled subgraphs might not accurately represent the full graph's structure and feature distribution, leading to a potential loss in model performance. Also, random walk sampling can introduce bias towards nodes with higher connectivity or those that are more frequently visited in walks, potentially overlooking less connected nodes. In this dissertation, we take a different approach by constructing subgraphs based on semantic clusters of neighboring traffic sensors formed in a low-dimensional vector space.

Additionally, the state-of-the-art graph generative models often use an auto-encoder framework where the encoder maps the input graph to a vector space, and the decoder reconstructs structures or node features from that space [51]. However, these models typically simply rebuild one modality, either the graph structure or node features, which limits the richness of the representation. The downstream task of spatial-temporal clustering thus often falls short in fully exploiting the graph structure or the interaction between the graph structure and node content. This limitation stems from their reliance on partial network information or superficial consideration of relationships between content and structure data, often applied directly to sparse original graphs [52]. Based on the review of existing literature, there are two primary hurdles: (1) the difficulty of simultaneously reconstructing both graph structures and node features, and (2) the limitations of current learning objectives, for example, using mean square error for node feature reconstruction and binary cross-entropy for link prediction [53]. To overcome these, in this study, we leverage both graph structure and node content and design a dual encoding scheme with a new loss function in graph representation learning task, attaining the optimal traffic sensor clusters.

CHAPTER 3

SYMMETRIC CONTRASTIVE LEARNING FOR TRAFFIC SENSOR FAULT DETECTION

In this chapter, our first framework was introduced, centering on a symmetric contrastive learning approach for data fault detection at the individual traffic sensor level. It employs a triplet network with an efficient sampling strategy, coupled with a novel cross-attention-boosted loss function for network training.

3.1. Research Overview

Although deep learning models perform well in fault detection for time-series data, they often rely on labeled data or generate their own supervision signals. These methods typically assume similar feature distributions between training and testing data, which isn't always true in real-world conditions where sensor variations, degradation, or environmental noise can alter the data. As a result, discrepancies between training and testing data can significantly reduce the models' fault detection performance. Contrastive learning, a self-supervised learning scheme initially proposed by [54], could be leveraged to mitigate this issue. The basic idea of contrastive learning is to learn a latent space where the similarity between the views under different augmentations of the same input data is maximized while minimizing the similarity between dissimilar data samples [55], [56]. This contrastive representation learning focuses on relationships between constructed pairs

of data samples and does not require explicit labels. Several studies have evaluated the effectiveness of contrastive learning for fault detection. For instance, Hojjati and Armanfard (2022) employed audio-specific augmentations and a contrastive learning framework with a revised loss function for acoustic anomaly detection, yielding promising experimental results [57]. In another study, Zhang et al. (2022) proposed a supervised fault detection approach based on contrastive learning. To address the discrepancy between the source and target domains, a cross-domain supervised contrastive loss is used with labeled information for domain adaptation [58].

However, current contrastive learning models are primarily focused on images and videos, with little attention paid to fault detection in traffic sensor data. The main challenge in detecting faults in traffic sensor data is to learn proper time-series representations. This involves identifying an appropriate feature extraction mechanism for the classification problem, as well as finding a suitable transformation of raw traffic data that highlights multiscale temporal features. Normal and faulty time-series can sometimes be quite similar depending on temporal scales, adding to the difficulty of the task. Triplet networks (TripletNet) [59] have been shown as a successful paradigm of contrastive learning and are widely applied to challenging tasks such as face recognition and image retrieval. In addition, TripletNet is effective in addressing the problem of distribution shift and can thus generalize well to new domains, as it is able to learn a distance metric that is insensitive to certain types of domain shifts. The triplet loss, which pulls similar instances closer and repels dissimilar ones, allows the network to target feature invariance with fine details by purposely constructing transformations. TripletNet has demonstrated tremendous potential in feature representation learning.

In this chapter, we presented a sample-efficient, symmetric contrastive learning method with triplet encoding for detecting faulty traffic data. Based on the previous work [60], The CWT

is first applied to convert the original time-series traffic sensor data into two-dimensional images in the time-frequency space with a desirable resolution. Next, we design a CNN encoder and use self-attention layers for selective feature pooling. Unlike traditional triplet networks that take a tuple of three elements as input, our proposed triplet network processes a tuple of seven instances, including an anchor, three positive examples generated by adding three types of domain-informed noises, and three negative examples produced by injecting three types of faults observed in historical traffic data, as illustrated in **Figure 3.1**. To guide the metric learning process explicitly, we introduce two cross-attention layers during training before computing the loss. We refer to this as cross-attention-boosted triplet loss, which is computed based on the nine permutations of the triplet, i.e., {anchor, one of the three positives, one of the three negatives}. For comparison, we evaluate, as baselines, two widely applied contrastive learning methods: the traditional triplet network and Siamese network (SiameseNet) [61], demonstrating superiority of our proposed method. In comparison to traditional triplet and Siamese networks, as well as a classic threshold-based method, our proposed approach shows superior performance in detecting faulty data sequences. The experimental results demonstrated an impressive accuracy of 97.6%, precision of 97.5%, recall of 97.7%, and an F1 score of 97.6%. The key contributions of this research study are summarized below:

(1) A sample-efficient symmetric contrastive sampling strategy was introduced to harness domain knowledge by perturbing the same normal traffic sequence to generate direct contrastive samples. This facilitates the learning of invariant features as well as contrastive signals.

(2) A novel cross-attention boosted loss function was introduced for training the triplet network, which significantly improved the cohesion within classes and the distinction between classes.

(3) A CNN encoder was designed with self-attention layers for selective feature pooling.

(4) Extensive experiments were conducted, demonstrating the effectiveness of the above-mentioned novel components.

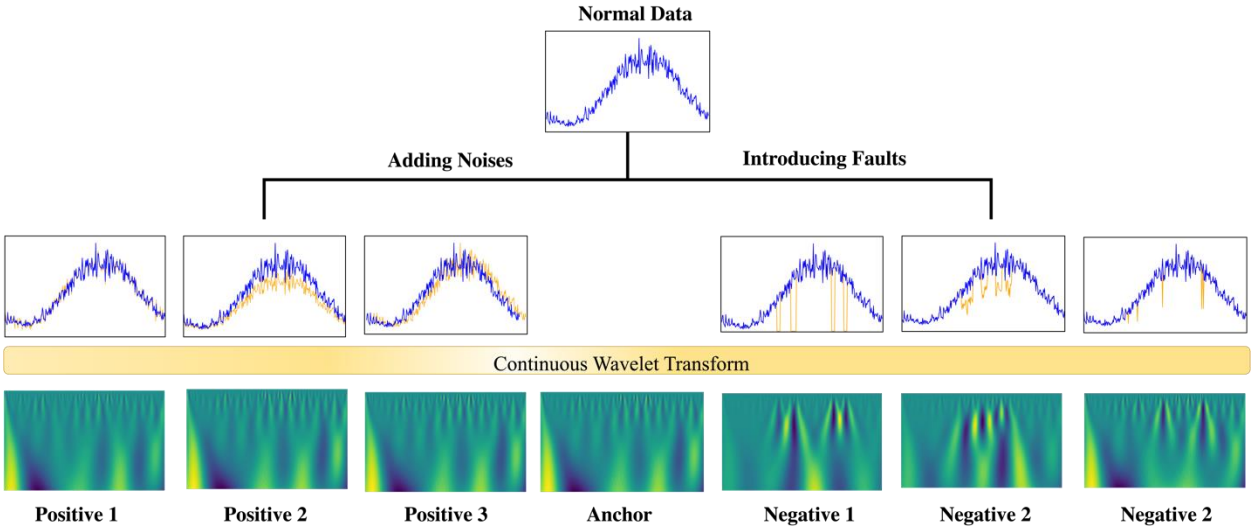

Figure 3.1: Domain-inspired triplet data generation.

3.2. Contrastive learning

Contrastive learning is a powerful technique within the field of metric learning that focuses on learning representations by contrasting positive and negative pairs of examples. Its similarity to human learning has led to its widespread recognition and adoption. In this section, we provide an overview of contrastive learning by highlighting two widely adopted networks: the Siamese network and the triplet network. These networks are trained in self-supervised learning settings to

acquire informative, meaningful embeddings, which can subsequently be employed for various downstream tasks, such as classification.

3.2.1. Siamese Network

SiameseNet [61] is a special type of neural network architectures that consists of two identical networks with shared parameters. During training, the parameters are updated by minimizing a contrastive loss function that is computed based on a distance metric in the latent embedding space [62]. In contrast to conventional learning systems where the loss function is a sum over sample batches, the contrastive loss for SiameseNet is computed over pairs of samples [63].

Let $\mathcal{F}_s(x)$ be the embeddings of an input x and let x_1 and x_2 denote the two paired inputs, where x_1 and x_2 can be either similar or dissimilar. A binary label y is assigned to the input pair to indicate whether the pair is similar or dissimilar, where $y = 0$ if x_1 and x_2 are similar and $y = 1$ if they are dissimilar. The contrastive loss $\mathcal{L}_s$ is computed using Eq. 3.1, where m is the margin between similar and dissimilar pairs, which is often treated as a hyperparameter. The distance metric, $D(\mathcal{F}_s(x_1), \mathcal{F}_s(x_2))$ is typically the Euclidean distance in the embedding space.

$$\mathcal{L}_s = (1-y)\frac{1}{2}(D(\mathcal{F}_s(x_1), \mathcal{F}_s(x_2)))^2 + (y)\frac{1}{2}\{max\ (0, m - (D(\mathcal{F}_s(x_1), \mathcal{F}_s(x_2)))\}^2. \qquad (3.1)$$

3.2.2. Triplet network

One limitation of the SiameseNet is that the margin parameter m only distinguishes between similar and dissimilar samples, without controlling the variability among samples within the same class. As a result, a large number of similar samples are required to effectively cluster intraclass samples in the feature space [64]. To address this limitation, the TripletNet [59] considers both positive and negative distances relative to an anchor in the loss function, resulting in improved

metric learning. The triplet loss is defined by Eq. 3.2, where minimizing the loss is equivalent to reducing the distance between the anchor and the positive (the first term in Eq. 3.2) while increasing the distance between the anchor and the negative (the second term in Eq. 3.2) such that the difference between the two distances is at least the margin value, which is a user-defined hyperparameter. This allows TripletNet to better handle outliers and varying levels of intraclass variance, offering more flexibility than SiameseNet in constructing the latent space.

$$\mathcal{L}_T = \max\{D(\mathcal{F}_T(a), \mathcal{F}_T(p))^2 - D(\mathcal{F}_T(a), \mathcal{F}_T(n))^2 + m, 0\} \quad (3.2)$$

where, a denotes an anchor input, p and n indicate positive and negative inputs, respectively; m is target margin; $\mathcal{F}_T(\cdot)$ is the encoder network.

3.3. Proposed method

The traditional TripletNet framework has limitations due to its reliance on selection and quality of training triplets, which can lead to ineffective and unstable feature representation learning if the triplets are not properly chosen, as well as high computational cost with a large number of triplets that could be constructed from a training dataset. To address these issues, we propose an adapted TripletNet framework that uses a symmetric sampling strategy, where same number of positive and negative samples are generated from the same anchor. Furthermore, we employ a cross-attention mechanism between positive and negative examples to boost loss signals. Our proposed method consists of three major components: (1) CWT imaging of time-series traffic data, (2) customized triplet network training to induce a latent embedding space endowed with discriminative power, and (3) a classifier that leverages embeddings for detection of faulty traffic sensor data. **Figure 3.2** illustrates the conceptional framework of the proposed method. The

symmetric sampling strategy and each of the major components are discussed in detail in the subsequent sections.

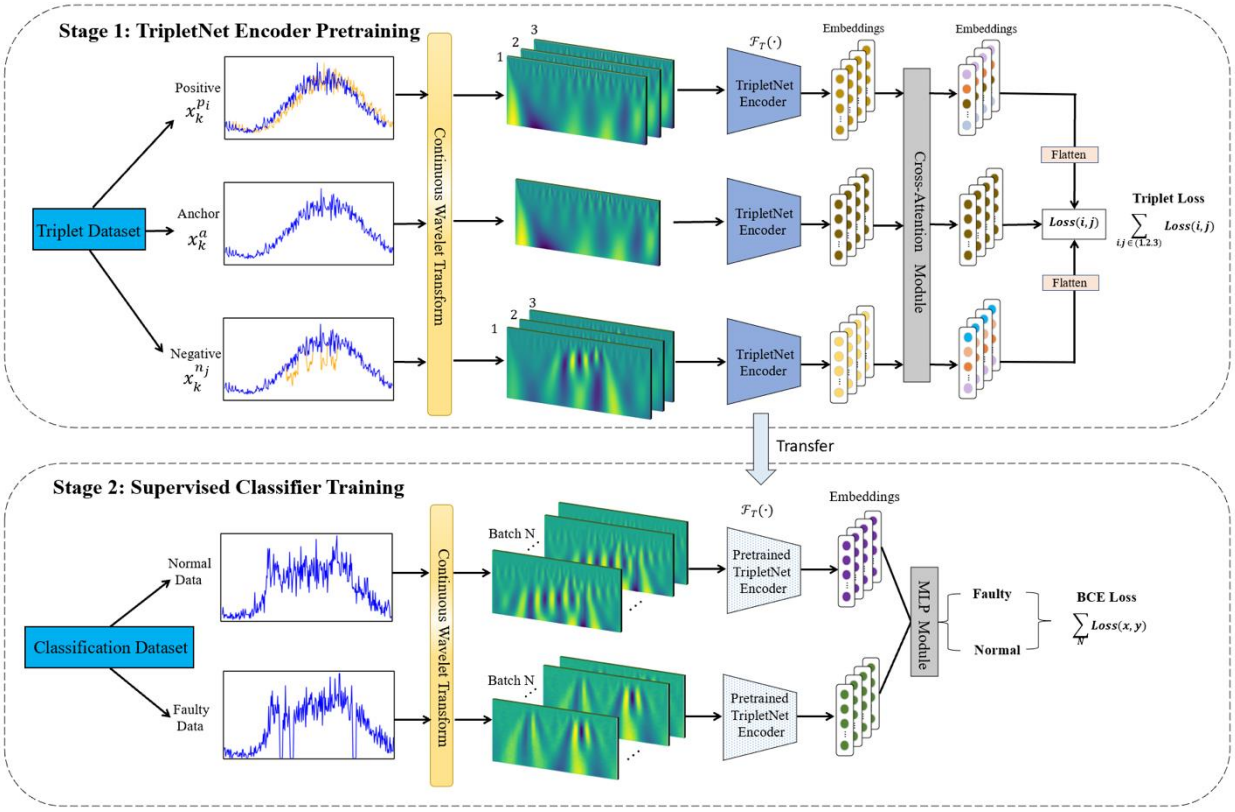

Figure 3.2: The conceptional framework of the proposed method.

3.3.1. Domain-Inspired Data Generation

The process of contrastive learning involves transforming original data to create multiple instances of each sample, which enables the learning of target invariant features from unlabeled data. These transformations must preserve the essential information of the original samples to enable the network to identify its distinctive features [65]. In this paper, we used domain-inspired augmentation strategies to generate positive and negative examples from same original data sample, referred to as anchor in the triplet network setting. Specifically, we used healthy time-

series traffic volumes in five-minute intervals over a day as the anchor, from which three positive and three negative instances are generated, leading to nine triplet pairings.

To simulate natural traffic variation, we created positive signals by applying three types of perturbations: (1) Gaussian noise (random noise), (2) temporal shifting, and (3) magnitude scaling. Gaussian noise captures the natural randomness present in traffic counts on a daily basis, temporal shifting simulates traffic shock wave that could be induced by bottleneck locations with reduced capacities, and magnitude scaling reflects significant traffic volume changes that could arise from special events, work zones, or incidents. **Figure 3.3** illustrates the effects of the three perturbation types, where the blue lines trace the original data trends, and the orange lines represent the perturbed data trends.

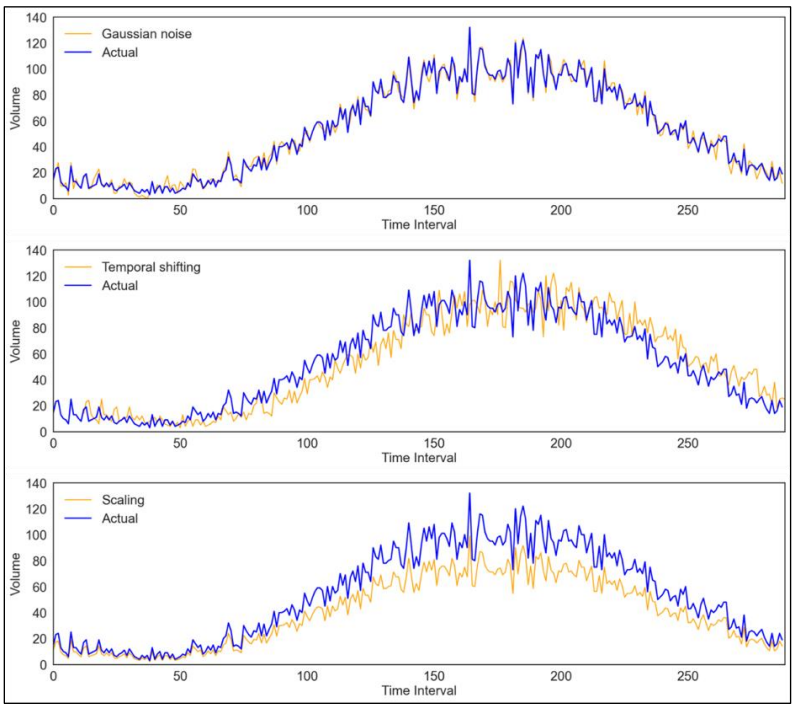

Figure 3.3: Visualization of three types of natural noises. The top plot simulates natural traffic variation by small perturbation of traffic volumes (i.e., injecting Gaussian noises); the middle plot indicates potential temporal shift of traffic; and the bottom plot demonstrates magnitude scaling to reflect potential increase or decrease of overall traffic due to certain events.

Three prevalent categories of faults observed in historical traffic sensor data, namely, nonresponsive faults, block faults, and point faults, serve as a basis for creating negative examples, contrasting with the positive ones [4]. Environmental factors such as weather-related damage, technical issues like power anomalies, and inappropriate setup can result in nonresponsive faults where traffic sensors fail to react. On the other hand, block faults, manifesting as undercounting of vehicles, may arise from a variety of causes, such as physical obstructions, misalignment of sensors, and calibration discrepancies. Moreover, point faults are attributed to a range of sporadic problems, which may include hardware malfunctions that occur intermittently, variable power supply, and transient blockages. The nonresponsive fault is generated by randomly selecting short time segments (e.g., 5 consecutive time intervals) and suppressing the traffic volumes to zero. Block faults are replicated in a similar fashion to nonresponsive faults but instead cause a significant reduction in traffic volume, for instance, a 40% decrease. Point faults, akin to block faults, sporadically lower the traffic count at random time points. **Figure 3.4** illustrates the effects of the three types of faulty signal injections, where the blue lines indicate normal data trends, and the orange lines indicate the contaminated signals.

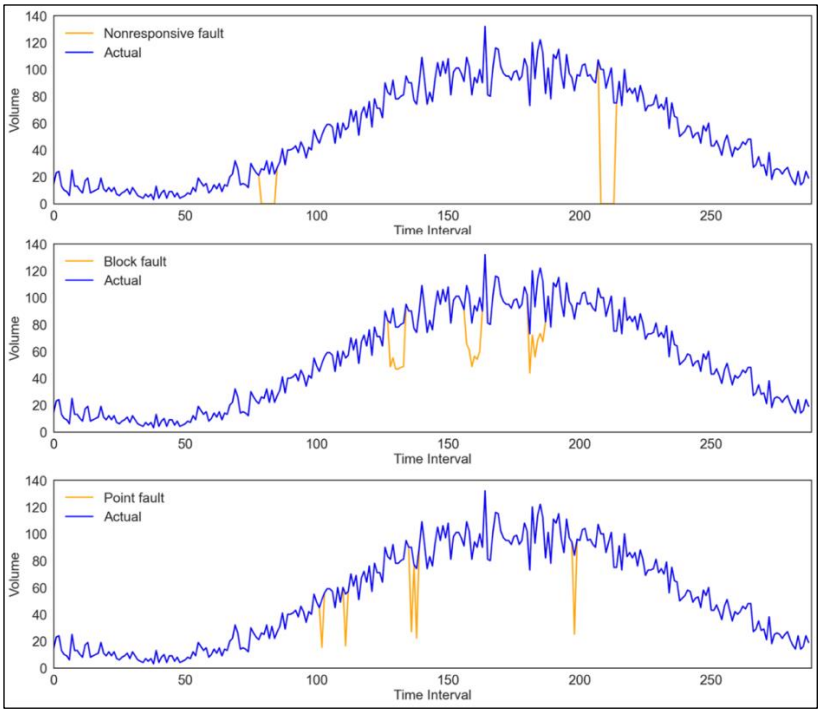

Figure 3.4: Visualization of three types of faulty signals; top: nonresponsive fault, middle: block fault, bottom: point fault.

3.3.2. Continuous Wavelet Transformation

CWT is employed as a preprocessing step in this study. Wavelets are formed by convoluting scaled and translated versions of the mother wavelet over time-sequence data. The mother function can be used to convert one dimensional data into scaled N-dimensional data. In this context, scaling refers to stretching or shrinking the signal in time by the scaling factor, which is inversely proportional to frequency. While shifting refers to moving the location of the wavelet imposed on the signal. Stretching a wavelet can help to capture slow changes in time series data, while shrinking improves the ability to detect abrupt changes. CWT results in a two-dimensional image that visually captures both slow and abrupt changes, inducing a powerful representation for time-series data. The CWT can be written in Eq. 3.3.

$$C(a,b;f(t),\psi(t)) = \int_{-\infty}^{\infty} f(t)\frac{1}{a}\psi * (\frac{t-b}{a})dt \quad (3.3)$$

where, the scale is represented by a, and the position by b. * denotes the complex conjugate. ψ is the mother wavelet function.

Figure 3. 5 displays some generated examples of the positive and negative instances, and their corresponding CWT images. The CWT image of the positive example appears nearly identical to that of the anchor example, while the negative example displays distinguishable characteristics in the higher frequency areas of the CWT image.

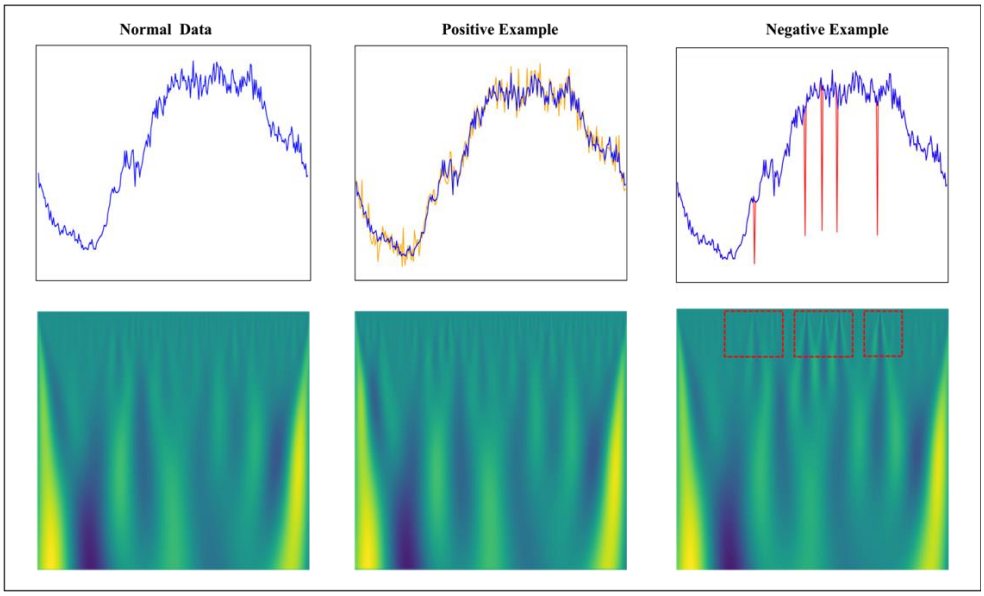

Figure 3.5: Sample wavelet transformations of time-series traffic volume data; left: normal data, middle: normal data with noises, right: faulty data. The red rectangles on the negative CWT image (right) highlight the distinguishable fault signals in contrast to the normal and positive CWT images (left).

3.3.3. Proposed Triplet network

The proposed TripletNet is composed of a CNN encoder and an MLP classifier. To train the encoder, we use a domain-inspired triplet loss function, which is based on an anchor and three positive, and three negative instances generated from the same anchor. This results in nine triplet pairings: $X_k = \{x_k^a, x_k^{p_i}, x_k^{n_j}\}$, where $i, j \epsilon \{1,2,3\}$ and x_k^a is anchor instance, $x_k^{p_i}$ denotes positive instance i and $x_k^{n_j}$ is negative instance j. We treat the nine triplets as a "minibatch" and feed them to the CNN encoder for training. Once the encoder is trained, its weights are fixed, and it functions as a feature extractor for training an MLP classifier. The subsequent subsections outline the architecture of the convolutional TripletNet encoder, followed by the cross-attention boosted triplet loss function and the MLP classifier designed for fault detection.

Architecture of the Encoder

The encoder architecture consists of four convolution blocks, each of which contains two 2D convolutional layers and a max pooling layer, resulting in a flattened vector with 2048 dimensions. Batch normalization and Rectified Linear Unit (ReLU) activation are applied after each convolutional layer. The inputs to the TripletNet encoder are CWT images with dimensions of 1×64×64.

The convolution operation is recognized for its data efficiency but can introduce high bias due to its localized structure. In contrast, the attention mechanism offers greater flexibility and has gained popularity following the success of the transformer model [66], finding extensive application in image classification tasks [67] . Drawing inspiration from the human visual attention mechanism, the attention module aggregates information by assigning different weights to different inputs based on their importance. As not all features obtained from the convolutional

block contribute equally to the classification task, a self-attention module [68] is utilized to perform selective feature pooling on the features obtained from each convolutional block. The features from each of the self-attention layers are then projected to a common-length vector. These vectors are concatenated to obtain an output of size 4×1×2048. The architecture of the proposed TripletNet encoder is illustrated in **Figure 3.6**.

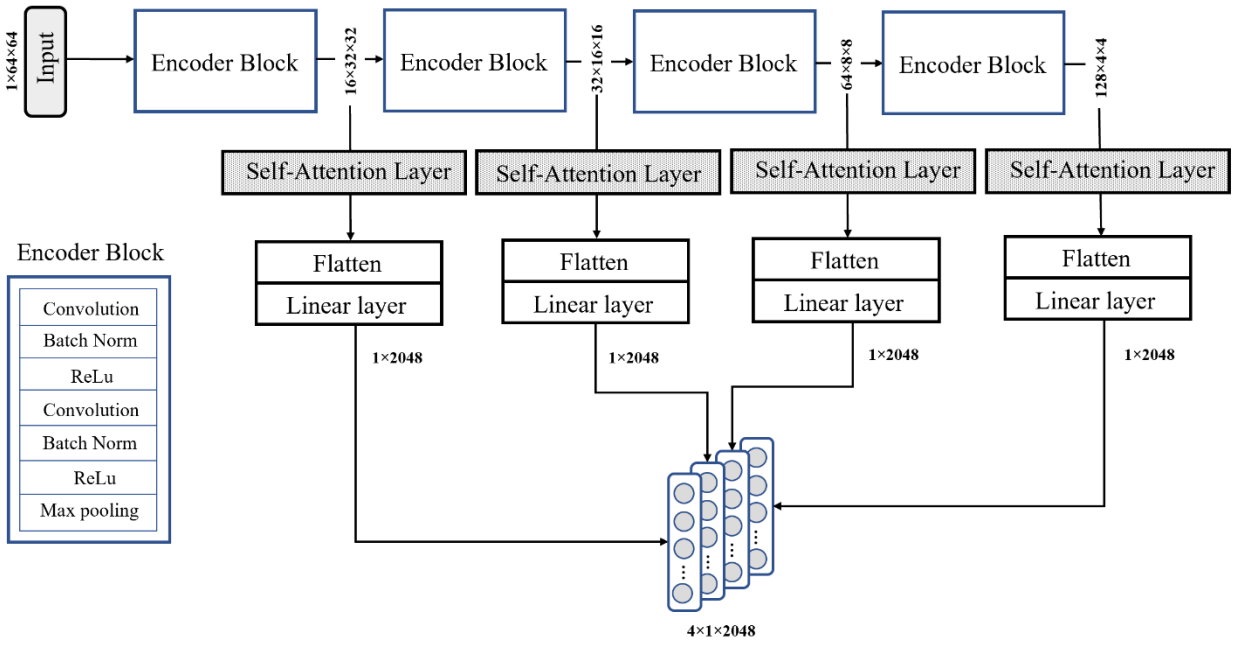

Figure 3.6: The proposed TripletNet encoder with self-attention layers.

Cross-Attention Boosted Loss Function

The effectiveness of the traditional triplet network heavily relies on the careful selection and quality of example triplets, and the triplet loss function is designed specifically to learn embeddings that aid in comparing the similarity between data points. However, in the case of traffic sensor data, the differences between normal and faulty time-series are relatively minor, making it challenging to enhance intraclass aggregation and interclass separation. To address this,

we propose a cross attention-boosted triplet loss function. The attention mechanism is used to identify states within a network that closely resembles a given state, allowing for the extraction of relevant information, and emphasizing significant local regions for extracting more distinctive features [23]. In the conventional triplet loss function, positive and negative instance embeddings are not able to access each other's information before being used in the calculation of the triplet loss. Considering this, we use two cross attention modules that connect and share information between positive and negative embeddings, highlighting relevant regions with distinctive features.

Assuming a batch size of N, let $X^a = \{x_1^a, x_2^a, \dots, x_N^a\}$ represents the anchor instances, $X^p = \{x_1^{p_i}, x_2^{p_i}, \dots, x_N^{p_i}\}$ represents corresponding positive instances, and $X^n = \{x_1^{n_j}, x_2^{n_j}, \dots, x_N^{n_j}\}$ represents corresponding negative instances. The embeddings obtained from the TripletNet encoder for the kth anchor, positive, and negative instances can be denoted as $\mathcal{F}_T(x_k^a)$, $\mathcal{F}_T(x_k^{p_i})$, and $\mathcal{F}_T(x_k^{n_j})$, respectively, where $\mathcal{F}_T(\cdot)$ is feature encoder. Our proposed cross-attention mechanism is illustrated in **Figure 3.7**. which strictly implements multi-head self-attention mechanism, where "cross" refers to the fact that attention is applied across different groups of embeddings (i.e., anchor, positive, and negative). In other words, the queries come from one group of embeddings while the keys and values are from a different group of embeddings. The scaled dot-product attention and multi-head attention are computed by Eqs. 3.4 and 3.5.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.4}$$

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^o \tag{3.5}$$

where, $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$; d_k denotes the dimension of queries and keys and d_v is the dimension of values; The parameter matrices are $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in$

$\mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^o \in \mathbb{R}^{hd_k \times d_{model}}$. In our study, the d_{model} is set to 2048 and $h = 8$.

For each anchor in a batch, three positive and three negative examples are generated from it and their respective embeddings (denoted as Embeddings (1) in **Figure 3.7**) are obtained from the pretrained TripletNet encoder. In the first cross-attention layer, cross-attention is applied between the anchor embeddings and the positive embeddings (Eq. 3.6) and between the anchor embeddings and the negative embeddings (Eq. 3.7). In the second cross-attention layer, cross-attention is applied (Eqs. 3.8 and 3.9) between the positive and negative embeddings (denoted as Embeddings (2) in **Figure 3.7**) to obtain the final embeddings (denoted as Embeddings (3) in **Figure 3.7**) for loss computation.

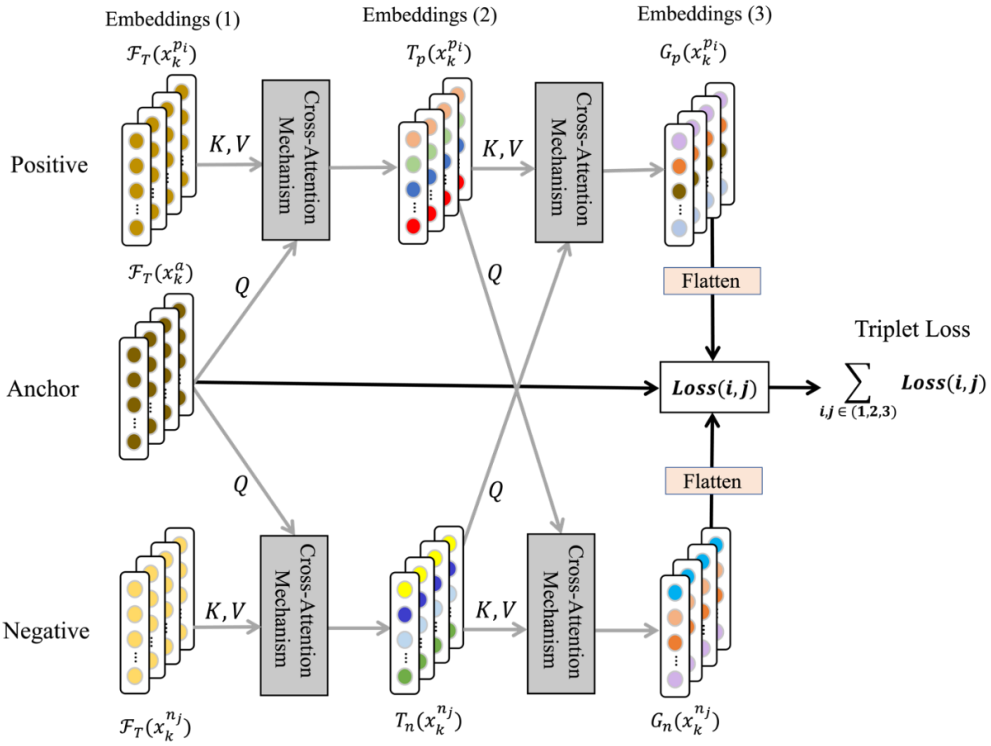

Figure 3.7: Cross-attention boosted contrastive loss.

$$T_p\left(x_k^{p_i}\right) = MultiHead_p\left(\mathcal{F}_T(x_k^a), \mathcal{F}_T\left(x_k^{p_i}\right), \mathcal{F}_T\left(x_k^{p_i}\right)\right) \quad (3.6)$$

$$T_n\left(x_k^{n_j}\right) = MultiHead_n\left(\mathcal{F}_T(x_k^a), \mathcal{F}_T\left(x_k^{n_j}\right), \mathcal{F}_T\left(x_k^{n_j}\right)\right) \quad (3.7)$$

$$G_p\left(x_k^{p_i}\right) = MultiHead_p\left(T_n\left(x_k^{n_j}\right), T_p\left(x_k^{p_i}\right), T_p\left(x_k^{p_i}\right)\right) \quad (3.8)$$

$$G_n\left(x_k^{n_j}\right) = MultiHead_n\left(T_p\left(x_k^{p_i}\right), T_n\left(x_k^{n_j}\right), T_n\left(x_k^{n_j}\right)\right) \quad (3.9)$$

The loss $\mathcal{L}$ per batch is computed by Eq. 10:

$$\mathcal{L} = \sum_{k=1}^{N} \sum_{i,j} \max\left\{L_2(\mathcal{F}_T(x_k^a), G_p\left(x_k^{p_i}\right)) - L_2(\mathcal{F}_T(x_k^a), G_n\left(x_k^{n_j}\right)) + m, 0\right\} \quad (3.10)$$

where $i, j \epsilon \{1,2,3\}$ and L_2 denotes Euclidean distance and m is margin.

Classifier

In the classification task, the multi-scale features retrieved from each block of the pretrained TripletNet encoder, are firstly projected to an 8,192-long vector and then fed to a multi-layer perceptron (MLP) [69]. The proposed MLP consists of two linear layers with ReLU nonlinearity, which can be mathematically represented by Eq. 3.11.

$$MLP(Z) = W_2\left(D\left(\sigma\left(\beta(W_1 Z)\right)\right)\right) \quad (3.11)$$

where W_1, W_2, are weight vectors for the first and the second layer of the MLP, Z is the input vector; σ denotes the ReLU activation function; β represents batch normalization; D indicates the dropout layer, with a drop rate of 0.5 being used in our experiments. Finally, softmax is applied to obtain the predicted class distribution. The classifier is trained using binary cross-entropy loss, denoted by Eq. 3.12.

$$L_{BCE} = -\frac{1}{M}\sum_{i=1}^{M}\left(y_i\, log\left(p(x_i)\right) + (1 - y_i)\, log\left(1 - p(x_i)\right)\right) \quad (3.12)$$

3.4. Dataset

Time series data of traffic volumes in 5-minute intervals are obtained from 254 active CCS sites across the state of Georgia over an 18-month period from August 2018 to July 2020. It should be noted that some sites have full-day missing records, which have been removed. The geospatial locations of the active CCS sites are depicted in **Figure 3.8**. The data is partitioned into two datasets: one is used for training the Triplet encoder, referred to as triplet dataset, and the other is used for training the classifier, referred to as classification dataset, as shown in Table 3.1.

Triplet dataset. The triplet dataset contains 81,443 normal daily time sequences of 5-minute traffic volumes, collected at 254 CCS sites from August 1, 2018, to July 31, 2019. Following the triplet network contrastive learning pathway, each normal daily time-series is used to generate three positive and three negative examples, as described previously.

Classification dataset. This dataset is used to train the classifier. It contains 20,015 daily time sequences of 5-minute normal traffic volume data from February 1, 2020, to April 30, 2020, and 20,599 daily time sequences of "faulty" data that were generated by injecting faulty signals to the normal traffic volume data collected in a different period from May 1, 2020, to July 31, 2020. This is done to prevent information leakage for the classifier training. It should be noted that the goals of contrastive learning and classifier training are different, as the former aims to learn invariant features from nuisance noises while the latter is intended for the classification task.

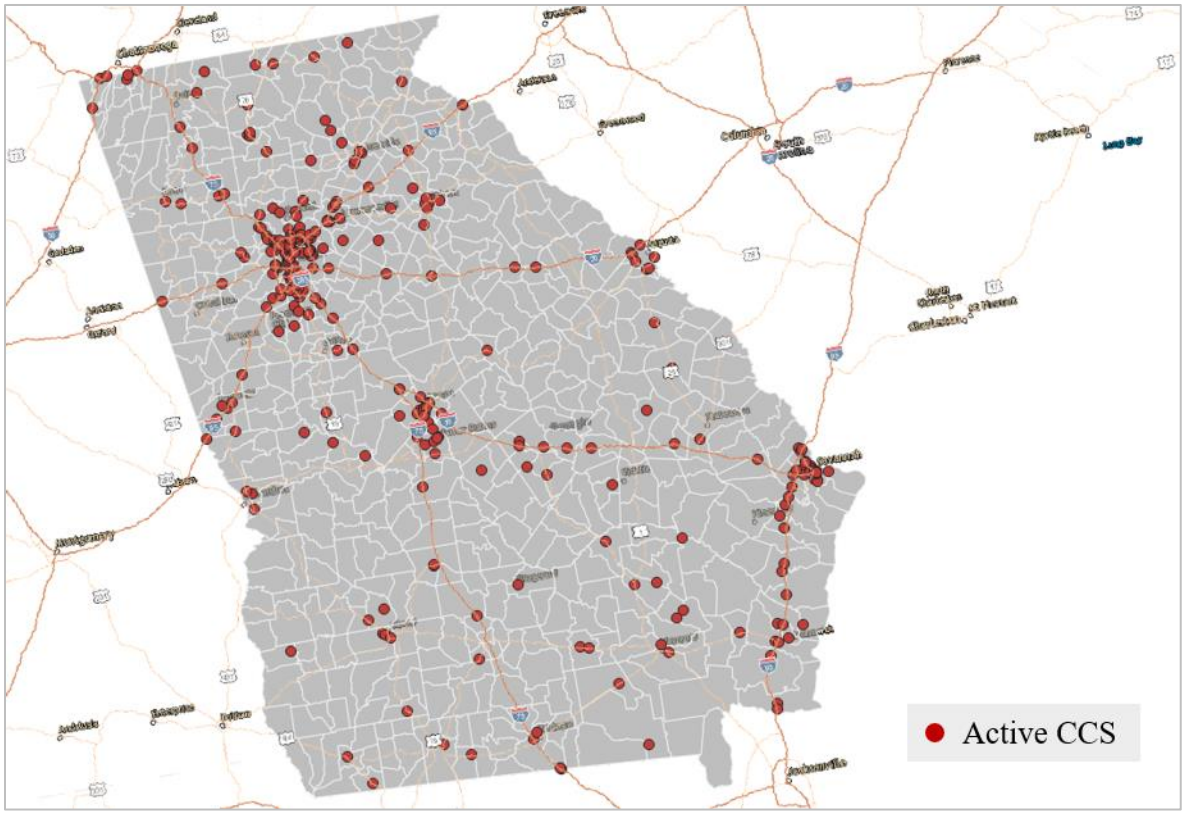

Figure 3.8: Locations of CCS in Georgia, USA [60].

Table 3.1: Summary of Datasets

Dataset	Time Window	# of CCS sites	Size (# of daily sequences)
Triplet dataset	8/1/2018-7/31/2019	254	81,443
Classification dataset	2/1/2020-7/31/2020	254	40,614 *

* Include 20,015 normal daily time sequences and 20,599 faulty daily time sequences generated from the 254 CCS sites.

3.5. Experiments

We present evidence that the TripletNet model sets a new benchmark for performance in contrastive learning applied to the detection of faults in daily traffic sequences, especially when dealing with images of time-series traffic data, utilizing a real-world continuous count station (CCS) dataset. The training process consists of two stages. In the first stage, the TripletNet and SiameseNet encoders are pretrained in self-supervised learning settings. In the second stage, these

pretrained encoders are used as backbones to train the classifiers in a supervised setting. We conducted various training scenarios to evaluate different encoder designs and loss functions, utilizing the dataset. This section also includes the pretraining of the TripletNet and SiameseNet encoders, the classifier training, the experimental settings, and results.

3.5.1. Model Architecture Design and Comparison

The TripletNet encoders are designed and trained in five different settings: (1) TripletNet A, where the encoder has no self-attention layers and the conventional triplet loss function is minimized based on nine triplet permutations, (2) TripletNet B, which adopts the same encoder as TripletNet A, but minimizes the cross-attention-boosted triplet loss, (3) TripletNet C, where self-attention layers are added in the encoder and the conventional triplet loss is minimized, (4) Proposed TripletNet, where the self-attention layers are added in the encoder and the cross-attention-boosted loss is minimized, and (5) TripletNet D, where only one triplet sample is processed per pass and there are no self-attention layers in the encoder, and the conventional triplet loss is minimized.

On the other hand, SiameseNet, which is widely used as a benchmark model in contrastive learning, was employed as a baseline for comparison. The SiameseNet employs the same CNN encoder architecture as TripletNet D. For pretraining the SiameseNet encoder, the data pairs were derived from the same triplet samples, where the anchor and positive examples were grouped as the normal class while the negative examples were treated as the faulty class. Furthermore, a classic threshold-based method using Z-score is also evaluated for comparison purposes. In this approach, faulty data points are identified based on the optimal threshold of Z score obtained experimentally.

3.5.2. Training of Classifiers

In the second stage, the pretrained TripletNet and SiameseNet encoders are frozen and simply used as feature extractors for training the multi-layer perceptron (MLP) classifier using the classification dataset. The encoders are kept frozen during classifier training. A k-nearest neighbor (KNN) algorithm [70] is also used as a baseline classifier for evaluating embedding quality. KNN is a non-parametric, supervised learning algorithm. There are no parameters to learn. It relies on majority vote of the k nearest neighbors to make class prediction. Since it does not impose any specific structures, the classification results reveal the quality of embeddings for class separation. Further details on the experimental settings and results are presented subsequently.

3.5.3. Experiment Settings

For traffic data imaging, the daily time-series traffic volume data is converted into CWT images (1×64×64) using the PyWavelets package [71]. In the stage of feature extractor pretraining, all models are trained using Adam [72] with initial learning rate of 0.0001 as optimizer. The batch size is set to 128. The triplet dataset is divided into a training set and a validation set with an 8:2 ratio. Each network is trained for 50 epochs. Margin value in loss function is set at 0.5. For the training of the MLP classifier, cross-entropy loss is used. The classification dataset is split into three sub-datasets: training set (60%), validation set (20%) and testing set (20%). Each classifier is trained for 50 epochs using Adam with a learning rate of 0.00001 and a batch size of 128.

3.5.4. Results and Discussions

For classification tasks, the model performance is commonly measured by the average accuracy, precision, F1-score, recall, and the area under the receiver operating characteristic curve (AUC).

These classification metrics for test data are summarized in **Table 3.2**, where the check mark (ü) indicates the specific design component adopted for each model. To evaluate the quality of embeddings from different encoders, a KNN classifier is applied, where a k value of 6 is selected based on experiments. Since there is no extra learning structure or parameters imposed by the KNN, the cluster results imply the embedding quality.

Table 3.2: Model Performance Evaluation

Architecture Component		**TripletNet**					**Siamese Network**	**Z-score Threshold Method**
		A	**B**	**C**	**D**	**Proposed**		
Number of samples processed per pass	9	✓	✓	✓		✓		
	1				✓		✓	
Encoder	CNN	✓	✓		✓			
	CNN + Self-Attn			✓		✓		
Loss Function	Triplet Loss	✓		✓	✓			
	Cross-Attn-Boosted Triplet Loss		✓			✓		
	Contrastive Loss						✓	
Classifier - MLP	Avg. Accuracy (%)	95.4	96.0	96.3	94.9	**97.6**	94.8	83.4
	Avg. Precision (%)	94.4	95.2	95.8	94.2	**97.5**	94.4	82.5
	Avg. Recall (%)	96.3	96.7	96.7	95.5	**97.7**	95.2	83.9
	Avg. F1-score (%)	95.4	95.9	96.3	94.9	**97.6**	94.8	83.2
	AUC	0.975	0.978	0.987	0.972	**0.992**	0.973	83.2
Classifier - KNN (k=6)	Avg. Accuracy (%)	92.4	92.9	93.3	91.6	93.9	91.9	83.4
	AUC	0.945	0.947	0.961	0.938	0.965	0.940	0.863

As is shown in **Table 3.2**, all deep-learning models have average accuracies in exceedance of 91%. Our proposed TripletNet, endowed with (1) self-attention layers at each CNN block, (2) nine triplet samples per pass for training, and (3) cross-attention-boosted loss function, achieved the best classification accuracy of 97.6% and AUC of 0.992 on the test dataset. All the TripletNet models outperformed the SiameseNet. Notably, all deep-learning models surpassed the conventional Z-score threshold-based method by a substantial margin of more than 10%. This could be explained by the fact that the faulty signals in daily time sequences contain rich features that are better captured by deep-learning methods than simple threshold-based methods. The latter performs point-wise detection and disregards any temporal features.

By comparison, the models (A, B, C, and Proposed) trained with nine triplet samples per pass have higher accuracies and AUCs than the one (D) trained with one triplet sample per pass. The superiority of model B over model A indicates the benefit of the cross-attention-boosted loss function. We argue that this is largely due to the enhanced contrast between positive and negative embeddings. Furthermore, upon comparing Model A and C, it became evident that the self-attention layers, which selectively gather information from different scales of embeddings, significantly enhanced the encoder's performance. To better understand the impact of the two proposed architectural design elements, i.e., self-attention layers and cross-attention-boosted loss, we plotted the training progress of the related models in **Figure 3.9**. The results show that Model B and the Proposed Model, which were optimized using the cross-attention boosted loss, converged much faster with improved stability as compared to Model A and Model C, respectively. Additionally, Model C and the Proposed Model, which are equipped with self-attention layers, also improved the training stability compared to Model A and Model B, which lack self-attention layers in their encoders.

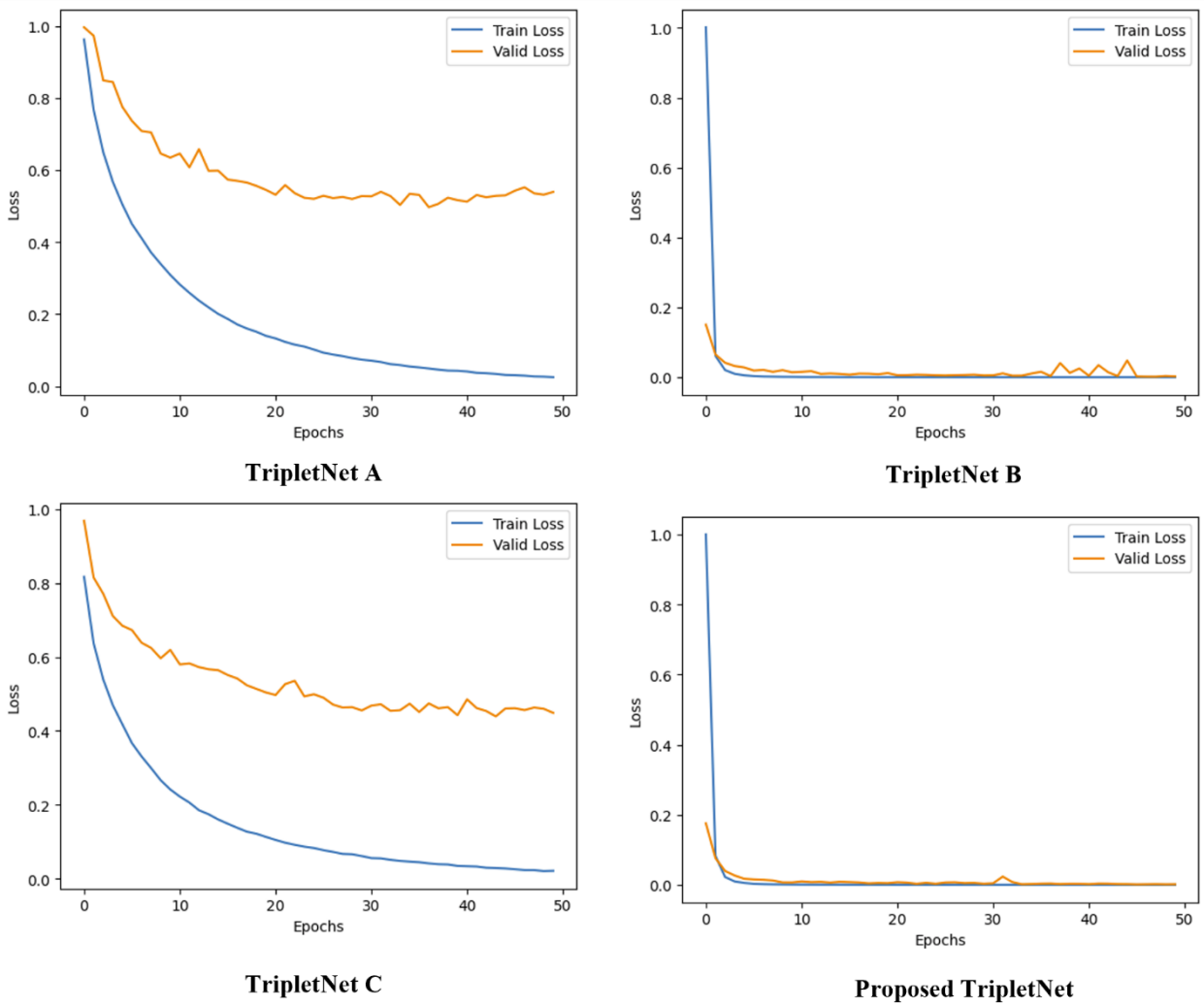

Figure 3.9: Progression of scaled training and validation losses.

To gain a better insight into the quality of the latent representation, t-SNE [73] is employed to embed 960 CCS CWT images, randomly sampled from the classification test dataset with equal numbers of faulty and normal samples. As shown in **Figure 3.10**, the pretrained proposed TripletNet encoder maps the faulty and normal data points to distinct clusters in the latent space. The KNN classification results in **Table 3.2** show the clear advantage of our proposed TripletNet encoder over the other encoders for the downstream classification task.

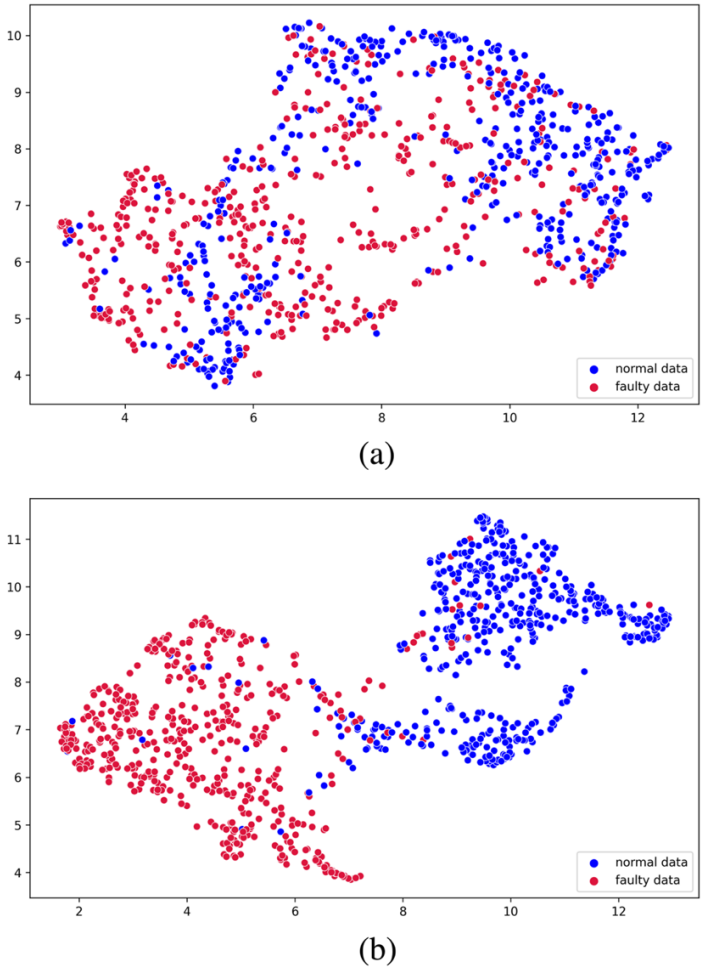

Figure 3.10: Features visualization by T-SNE embeddings for 960 sample data (480 positive instances + 480 negative instances); (a) original CWT image; (b) the proposed Triplet encoder.

3.6. Summary

In this chapter, we presented a sample-efficient, anchor-centered TripletNet framework for detecting faulty sensor data based on their unique temporal patterns. Our proposed model pipeline consists of three major sequential components: (1) CWT transformation of time-series traffic data, (2) a pretrained TripletNet encoder for separating faulty data from normal data in a multiscale embedding space, and (3) an MLP classifier for detecting faulty traffic data in the resultant

embedding space. Notably, we devised an anchor-centered data generation process for training the TripletNet encoder, whereby each normal day of time-series data is used as an anchor, from which three positive and three negative examples are generated based on the domain knowledge. This leads to nine permutations of triplet samples around the anchor that enables direct contrastive learning of faulty patterns. In addition, a cross-attention module is introduced during the training to enable the learning of a more nuanced embedding space for the subsequent classification task. Our experiments demonstrated the superiority of the proposed design and training strategy.

Nevertheless, this study can be extended to focus on the classification of different faults that may occur in time-series traffic data to provide valuable insights for sensor diagnosis. One limitation of our proposed method is the prerequisite of pre-training TripletNet. To facilitate continuous learning and adaptation to emerging features, it would be advantageous to explore meta-learning approaches and more suitable fine-tuning procedures.

Furthermore, it is important to note that the current study primarily concentrates on one-dimensional time-series data. To broaden the scope of applications, further research is needed to extend the existing framework to encompass high-dimensional correlated time-series data. For instance, our study focuses on individual traffic count stations, but the flow patterns of geographically proximate stations are inherently correlated and influenced by the network configuration. Therefore, adopting a more comprehensive approach that analyzes clusters and network topology of stations could enhance the reliability of anomaly detection and classification.

3.7. Publications

The work presented in this chapter has led to the following publications [74], [75]:

- Yongcan Huang, Jidong J. Yang. Symmetric contrastive learning for robust fault detection in time-series traffic sensor data[J]. International Journal of Data Science and Analytics (2024): 1-15.
- Yongcan Huang, Jidong J. Yang. Semi-supervised multiscale dual-encoding method for faulty traffic data detection[J]. Applied Computing and Intelligence, 2022, 2(2): 99-114.

CHAPTER 4

CLUSTER-GUIDED DENOISING GRAPH AUTO-ENCODER FOR TRAFFIC DATA IMPUTATION AND FAULT DETECTION

This chapter presented our second framework that leverages spatial contexts through clusters of sensors to improve fault detection. First, a traffic sensor clustering module was designed using a dual-encoding attention graph auto-encoder (DA-GAE) to identify clusters of traffic sensors. This module leverages a joint embedding of node and edge features in a low-dimensional vector space. Subsequently, utilizing the identified clusters, a cluster-guided denoising graph auto-encoder (CG-DGAE) was devised and trained for data reconstruction. The CG-DGAE employs a diffusion graph convolutional network (DGCN) and is trained with a cluster-wise sampling strategy. Extensive experiments were conducted using traffic data obtained from a real-world large sensor network, demonstrating superior performance of the CG-DGAE model in data reconstruction compared to various baseline methods. For fault detection, a score function is devised to discern potential faults by contrasting the sensor data sequence and the reconstructed data sequence.

4.1. Research Overview

Despite the advancements of deep learning models-based traffic data fault detection summarized in our literature review work, devising an optimal protocol for fault detection in traffic data remains a challenge, compounded by factors related to intrinsic characteristics of road traffic and the diversity of traffic data sources: These aforementioned factors collectively underscore the complexity and multifaceted nature of traffic data quality control, necessitating innovative approaches tailored to address the unique challenges posed by real-world traffic systems. In this chapter, we proposed a novel deep-learning-based approach consisting of three sequential phases: (1) traffic sensors clustering, (2) data reconstruction, and (3) fault detection.

(1) **Traffic sensors clustering:** a dual-encoding attention graph auto-encoder (DA-GAE) is designed to aggregate neighboring traffic sensors into coherent groups or clusters. It jointly encodes node features and network topology into a low-dimensional vector space. The established clusters are later utilized to enable a cluster-wise sampling strategy for robust traffic data reconstruction.

(2) **Data reconstruction:** a cluster-guided denoising graph auto-encoder (CG-DGAE) is proposed for traffic data reconstruction. The network employs a Diffusion Graph Convolutional Network (DGCN) to reconstruct healthy traffic data from contaminated one with faulty signals. Trained using subgraphs constructed from sensor clusters obtained previously, the CG-DGAE is able to recognize spatial-temporal dependency of sensors within each cluster and enhance the reconstruction and imputation accuracy.

(3) **Fault detection:** the pretrained CG-DGAE model is utilized to reconstruct sensor data sequences. By comparing the reconstructed data sequence with the input data sequence using a score function and a threshold, faults within the input data sequence can be identified.

Extensive experiments are conducted for each phase above using traffic datasets sourced from statewide continuous count stations (CCS) in Georgia. The results reveal that dual encoding empowered by the attention mechanism (i.e., DA-GAE) facilitates superior graph representation learning, leading to improved clusters. Our proposed CG-DGAE surpasses all baseline models in reconstructing traffic sensor data. We further validate the performance of fault detection using a manually annotated faulty traffic dataset. The major contributions of this work could be highlighted as below:

- A novel light framework for network level faulty detection is proposed. The framework features with the light dynamic cluster-guided subgraph sensor data, which is inherently well-suited not only for traffic faulty data but also for other domain data faulty detection problems.
- We designed a dual encoding scheme for graph representation learning that leverages both graph structure and node content, incorporating a new loss function to achieve optimal traffic sensor clustering and data reconstruction results.
- To optimize subgraph sampling in data reconstruction with GNNs, we developed a subgraph generation approach that clusters neighboring traffic sensors based on semantic similarity in a low-dimensional vector space.

- We developed a fault detection function based on a fault score derived from the cluster-level reconstruction sequence and observed sensor data. Testing on real-world faulty traffic sensor data demonstrated the method's feasibility and reliability.

4.2. Preliminaries

In this section, we first introduced graph representation for highway traffic sensor networks, followed by the formulation of the problem.

4.2.1. Graph Abstraction Representation of Highway Traffic Sensor Networks

In this work, we defined a highway traffic sensor network as a weighted directed graph $G = (V, E, A)$, where $V = \{v_1, v_2, \dots, v_n\}$ is a set of N nodes; E a set of edges indicating the connectivity between nodes; $A \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix capturing the dependency between the nodes. A physical traffic sensor in the highway network typically records the bidirectional flow data. In our case, the node is defined by direction, where the node features include directional sensor data. Existing approaches often define the adjacency matrix as a function of connectivity or distance. To get the connectivity in the graph, we first derive the distance matrix D from the geographic proximity. Then, the adjacency matrix A is derived based on the closest p nodes. For the highway traffic sensor network, the geographical proximity between two nodes or each element in D is computed by Eq. 4.1.

$$D_{ij} = \exp\left(-\frac{dist_{ij}}{max_{i,j}(dist_{ij})}\right) \tag{4.1}$$

where D_{ij} is the element value for geographical proximity between sensor i and j, $dist_{ij}$ is travel distance between i and j. The adjacency matrix A is defined as:

$$A_{ij} = \begin{cases} \frac{e^{D_{ij}}}{\sum_{j \in topP(i,p)} e^{D_{ij}}}, & j \in topP(i,p), \\ 0, & otherwise, \end{cases} \tag{4.2}$$

where $topP(i,p)$ represents a list of column indexes of the largest p element values in each row i of D.

4.2.2. Problem definition

In our setting, the objective of traffic sensor data fault detection is to identify anomalies present within the sensor data sequence. We assume that the traffic sensors can be organized into L distinct clusters, where the traffic data distribution within a cluster remains relatively stable and faults occur sporadically. For each node v in a cluster $\mathcal{g}$, the sensor data at a given timestamp t is denoted as $x_t^v \in \mathbb{R}$, and the sensor data recorded for the cluster $\mathcal{g}$ is represented as a graph signal $X_t = \{x_t^1, x_t^2, \dots, x_t^Z\}$, where $X_t \in \mathbb{R}^Z$ and Z represents the number of nodes in the cluster. Assuming a detection time frame of $[t-T, t]$, the crux of sensor data reconstruction lies in devising a function $F(.)$ capable of generating the fault-free traffic sensor data of each cluster $\hat{X}_{t-T:t} = F(X_{t-T:t})$, where $\hat{X}_{t-T:t}$; $X_{t-T:t} \in \mathbb{R}^{Z \times T}$ denoting the reconstructed data sequence and sensor data sequence, respectively, for the cluster.

To achieve the objective of fault detection, our proposed approach employs three sequential phases: (1) identifying optimal L clusters, (2) training cluster-wise traffic data reconstruction models, and (3) detecting faults by contrasting the reconstructed data with traffic sensor data. For clarity of presentation, a list of abbreviations is provided in Table 4.1, encompassing terms utilized in the methodology and experiments.

Table 4.1: Abbreviations

Abbreviation	Definition
AC	agglomerative clustering
BCE	binary cross entropy
Bi-LSTM	bi-directional LSTM
CCS	continuous count station
CG-DGAE	cluster-guided denoising graph auto-encoder
CHI	Calinski-Harabasz index
CNN	convolutional neural networks
DA-GAE	dual-encoding attention graph auto-encoder
DBI	Davies-Bouldin index
DGCN	diffusion graph convolutional neural networks
GAT	graph attention network
GCN	graph convolutional networks
GNN	graph neural networks
LSTM	long short-term memory
MAE	mean absolute error
Okriging	ordinary kriging
RS	random sampling
RMSE	root mean square error
RNN	recurrent neural network
RW	random walk sampling
SC	silhouette coefficient

4.3. Dataset

For this study, 5-minute traffic count data spanning six years from 2018 to 2023 were gathered from 221 active CCS sites embedded within Georgia's highway network. CCS sites use magnetic induction loops to capture traffic volumes every 5-minute in both directions. The distribution of the CCS sites is shown in **Figure 4.1**. The 5-minute traffic count data is further partitioned into three datasets, the **GC-dataset** (2018-2019) for graph clustering, the **GR-dataset** (2020-2022) for graph reconstruction and the **FD-dataset** (2023) for fault detection testing. This data split is designed to prevent potential information leakage between the three phases. To capture directional traffic flow, the dataset contains directional traffic volumes from each CCS, resulting in a traffic

sensor-informed graph with an effective node count of 442, which is twice the number of CCS sites.

Specifically, for the sensor clustering phase, we introduce a data aggregation trick (see section 4.4.3) to attain the median weekly traffic volume sequences as model input from GR-dataset to address the issue of data incompatibility on temporality at the cluster level. For the reconstruction phase, we further divide the GR-dataset into training data (2020-2021) and testing data (2022). The training of the proposed CG-DGAE model is based on the subgraphs generated by Algorithm 1 (see Section 4.4.4.). To test the performance of the reconstruction models, a uniform seven-day time window covering Monday to Sunday, common to all 221 CCS sites in both directions, is extracted from 2022. This test set is used for the CG-DGAE model as well as various baseline models.

For the FD-dataset, we selected five representative sequences containing actual faults to test our proposed CG-DGAE model in the fault detection phase. To better demonstrate the performance, we labeled every traffic count data point in these five faulty sequences with binary annotation, in which 0 denotes normal data points while 1 denotes faulty data points. The binary annotation is demonstrated in **Figure 4.2**. The binary annotation also allows efficient evaluation of our fault detection framework as a binary classification problem, which is detailed in the next subsection 4.4.4. The distribution of CCS sites where the selected faulty sequences are collected from is shown in **Figure 4.1**.

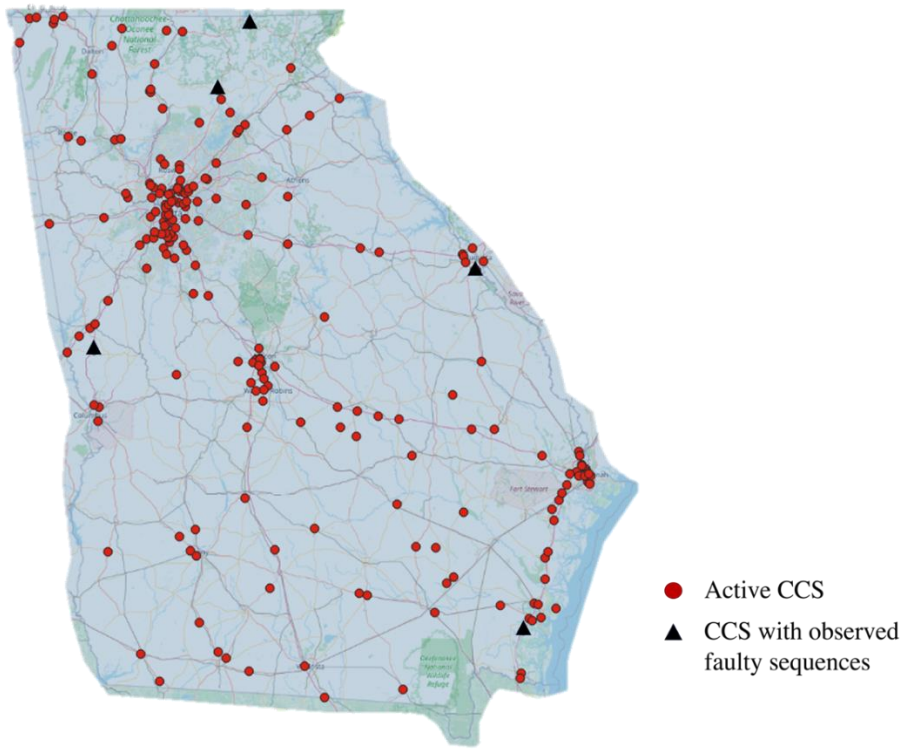

Figure 4.1: Locations of CCS in Georgia, USA. The black triangles denote the CCS sites where the five selected faulty sequences were collected.

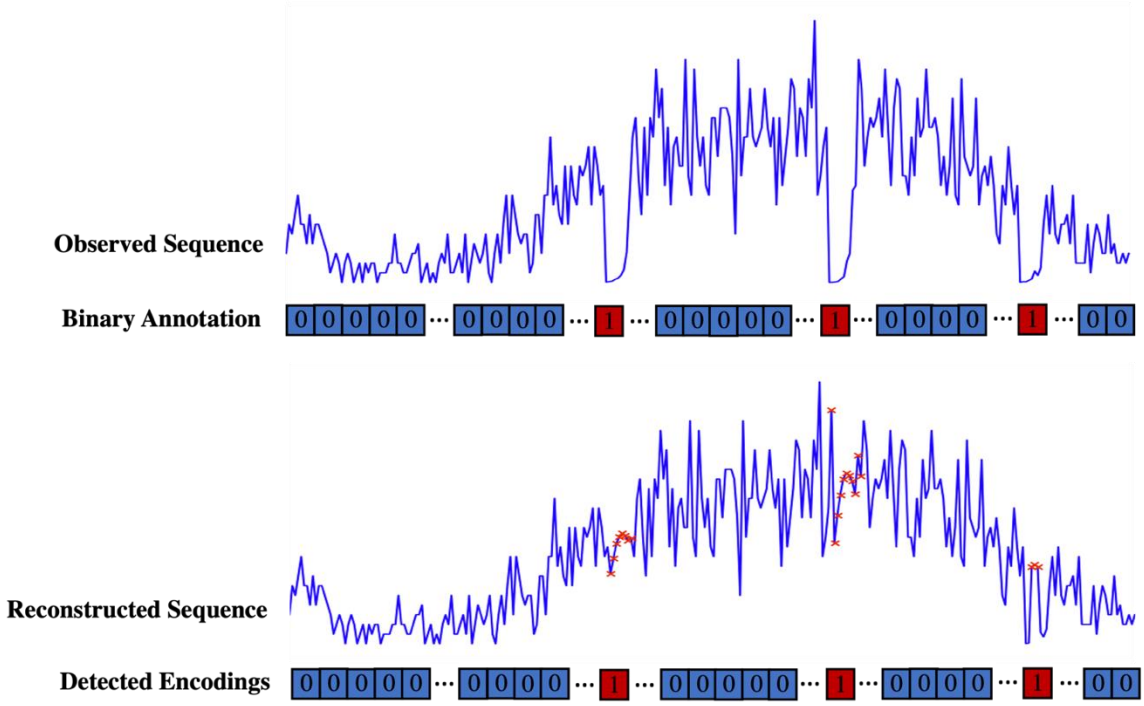

Figure 4.2: CCS sequence binary annotation (1 – faulty; 0 – normal).

4.4. Methodology

In this section, we presented our proposed GNN framework for traffic sensor data fault detection. For model development, as depicted in **Figure 4.3**, three sequential phases are undertaken: (1) Graph Clustering: traffic sensor clustering analysis is conducted in a joint node-edge embedding space, created by a dual encoding attention graph auto-encoder (DA-GAE), which takes into account the spatiotemporal dependency among adjacent sensors, (2) Graph Reconstruction: a cluster-guided denoising graph auto-encoder (CG-DGAE) is designed and trained to reconstruct the normal data from the corrupted ones, where faulty patterns are injected, and (3) Fault Detection: a score function is employed for fault detection by contrasting reconstructed data with input ones. In the following subsections, we first introduce two popular GNN architectures, GAT and DGCN, in Subsections 4.4.1 and 4.4.2, which serve as the building blocks for our proposed DA-GAE and CG-DGAE models. Subsection 4.4.3 covers the traffic sensors clustering task and

our proposed DA-GAE model. Subsection 4.4.4 present the traffic sensor data reconstruction task and our proposed CG-DGAE model. Finally, Subsection 4.4.5 discusses the fault detection task and the adopted score function.

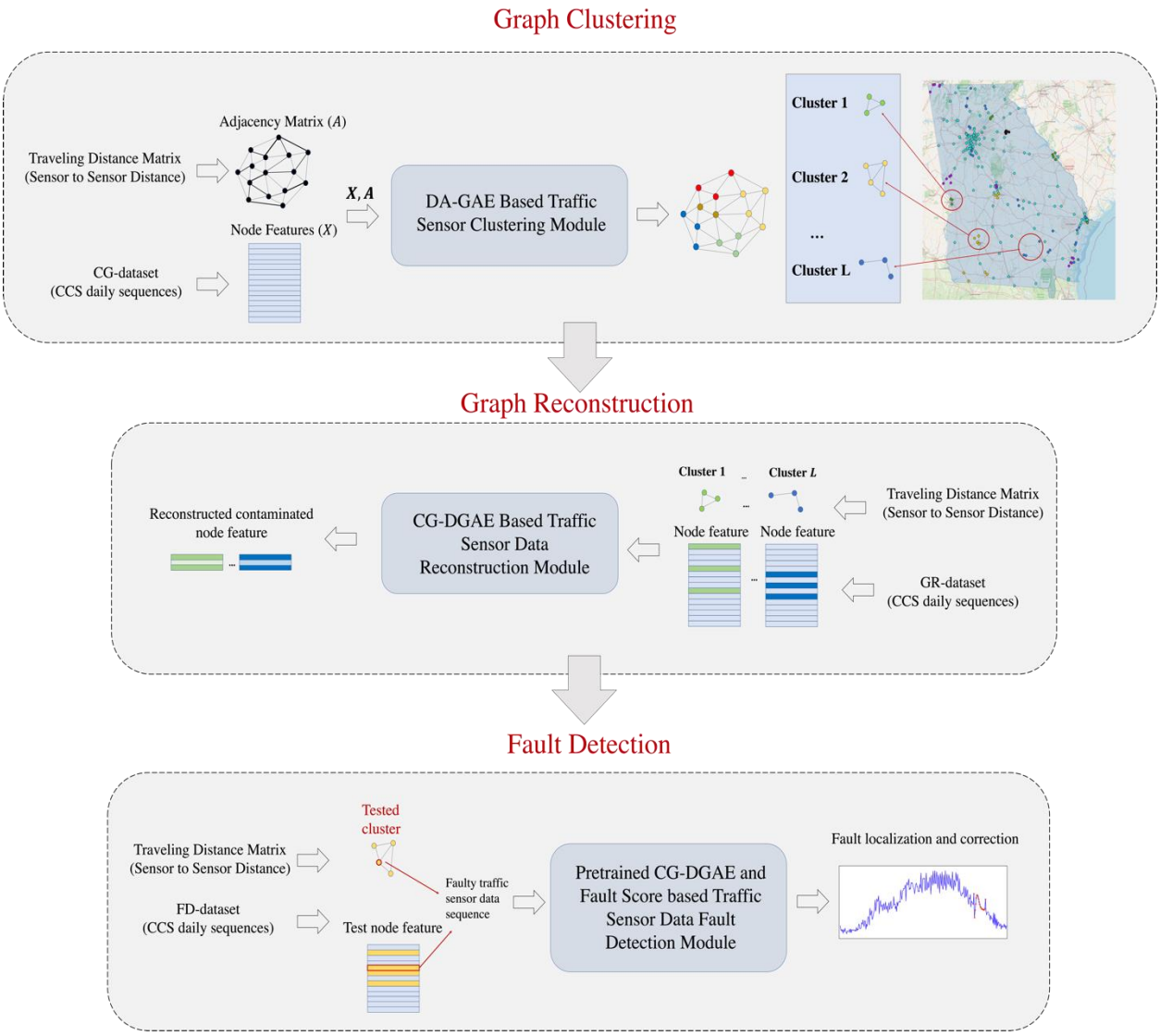

Figure 4.3: Three phases of the proposed framework for traffic sensor data fault detection.

4.4.1. Graph Attention Neural Network

The graph attention network (GAT) [51] is a specialized neural network tailored for processing graph-structured data. It harnesses attention mechanisms to facilitate the exchange of messages and aggregates information among neighboring nodes in the graph. However, traditional GAT considers only the 1-hop neighboring nodes for graph attention. To exploit the high-order neighbors, the revised GAT [52] is adopted in our paper.

Given a graph $G = (V, E, A)$ with n nodes, i.e., $\{v_1, v_2, \dots, v_n\}$, where $v_i \epsilon V$ is associated with a node feature vector x_{v_i}, and each edge $e_{ij} \in E$ connecting nodes v_i and v_j has a weight A_{ij}. The key idea behind GAT is to compute attention coefficients that capture the importance of node's state to another node. These coefficients are computed using a shared attentional mechanism for each pair of nodes. For nodes v_i and v_j, the attention score a_{ij} can be computed by the following Eqs. 4.3 and 4.4.

$$e'_{ij} = \delta A_{ij}(\vec{a}^T[Wx_{v_i} \parallel Wx_{v_j}]), \tag{4.3}$$

$$a_{ij} = \frac{\exp(e'_{ij})}{\sum_{k \in N_i} \exp(e'_{ik})}, \tag{4.4}$$

where, $\parallel$ denotes concatenation, $\vec{a} \in R^{2m}$ is a weight vector of learnable parameters where m is the dimension of the feature vector of each node, and δ is the nonlinear activation function, LeakyReLU. N_i indicates the neighboring nodes of i in A and W is a weight matrix. Consequently, a GAT layer computes the output of each node by Eq. 4.5.

$$GAT(X, A) = h'_{v_i} = \sigma(\sum_{k \in N_i} a_{ij} W x_{v_j}), \tag{4.5}$$

where h'_{v_i} denotes the output representation of node i, σ represents the sigmoid activation function; a_{ij} is the attention score which measures the contribution of node j to node i, where j is one of the adjacent nodes for node i.

4.4.2. Diffusion Graph Convolutional Network

Graph convolutional networks (GCN) have made significant strides in graph-based representation learning. However, traditional GCN predominantly operates within local neighborhoods, often overlooking the multi-hop relational structures. The diffusion graph convolutional network (DGCN) overcomes the constraint by integrating diffusion mechanisms. This allows for the extraction of information from remote nodes, resulting in a more global representation of graph structures in node embeddings. Considering that highway traffic sensor networks are characterized by directed links with an asymmetric distance matrix, the DGCN is utilized in this study to effectively capture the characteristics of spatial and directional dependencies.

For a directed graph $G = (V, E, A)$ comprising n nodes, denoted as $\{v_1, v_2, \dots, v_n\}$, each node $v_i \epsilon V$ is associated with a node feature vector x_{v_i}. Additionally, any edge $e_{ij} \in E$ forms a connection between nodes v_i and v_j. Here, A represents adjacency matrix. The layer-wise convolution in DGCN can be expressed by Eq. 4.6:

$$DGCN(H_l, A) = H_{l+1} = \sum_{r=1}^{R} T_r\left(\bar{A}_f\right) H_l \ominus_{b,l}^{r} + T_r(\bar{A}_b) H_l \ominus_{f,l}^{r}, \quad (4.6)$$

where $\bar{A}_f = A/rowsum(A)$ and $\bar{A}_b = A^T/rowsum(A^T)$ are the forward and backward transition matrices, respectively. R stands for the order (or steps) of diffusion convolution; The convolution process in DGCN leverages the Chebyshev polynomial for approximation and $T_r(E) = 2ET_{r-1}(E) - T_{r-2}(E)$, initialized with $T_0(E) = I$ and $T_1(E) = E$; $\ominus_{f,l}^{r}$ and $\ominus_{b,l}^{r}$ are the learnable parameters of the lth layer, dictating the information transformation among nodes; the

output from the lth layer is denoted as H_{l+1} with $H_0 = X$. Unlike conventional GNNs, which typically accept a fixed dimension of spatial inputs, DGCN exhibits versatility in accommodating diverse subgraph structures.

4.4.3. DA-GAE Based Traffic Sensor Clustering

The goal of cluster analysis is to identify traffic sensors with strong spatiotemporal correlations, thereby enhancing the accuracy of data fault detection in tightly knit groups within the broader graph structure. The outcomes of clustering serve as guidance for generating sampled subgraphs used in training GAE in the graph reconstruction and fault detection phases. To fully exploit the graph structure and node content, we adopt a novel self-supervised generative model, namely, dual-encoding attention graph auto-encoder (DA-GAE). This model innovatively maps both structure and node information to a joint latent space. Furthermore, a loss function that integrates the information loss from both modalities is devised, facilitating a more comprehensive and cohesive reconstruction of graphs. The proposed DA-GAE can inherently handle the spatial constraints of traffic data among neighboring traffic sensors, as depicted in **Figure 4.4**.

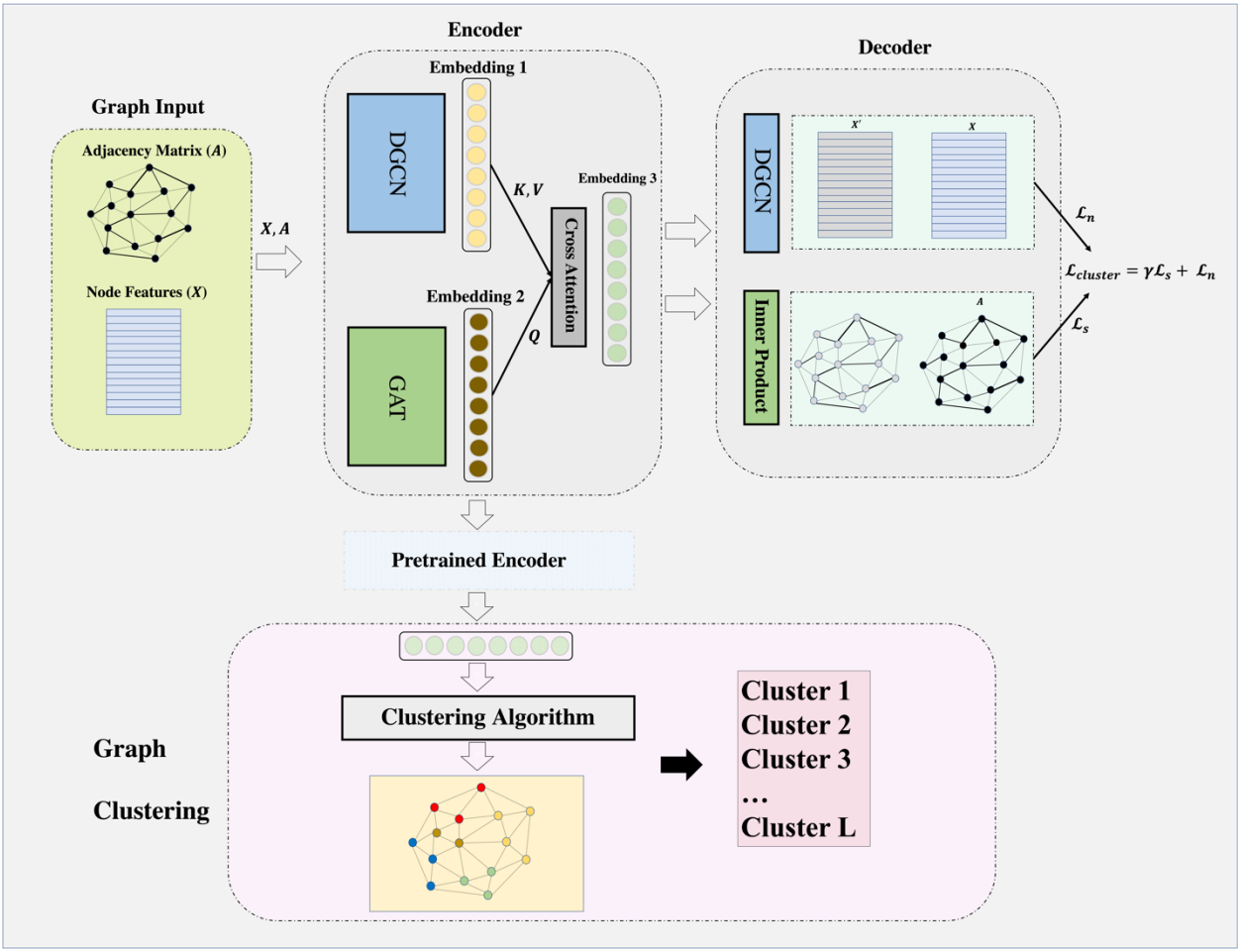

Figure 4.4:Architecture of graph autoencoder with dual encodings.

Dual Encoding Attention GAE Encoder

GNN-related tasks usually require temporal consistency for node features. However, a significant number of traffic sensors struggle to achieve a sufficiently long common time window to meet the model training requirements. To address the issue of data incompatibility on temporality at a cluster level, the median daily traffic volume sequences of a week are represented as the graph's node features. We use "median", instead of "mean", to mitigate the effect of outliers. For a traffic sensor node v_i, the volume on day d of week w is given by $Value(v_i, d, w)$ of 288 datapoints,

where d index 7 days of a week. The median value for node v_i for day d is represented by $M(v_i, d)$ and computed as

$$M(v_i, d) = median\{Value(v_i, d, w)|w = 1,2, \dots, N\}, \quad (4.7)$$

where, N denote the complete counts of weeks recorded. Therefore, the node content $X(v_i)$ that summarizes the typical weekly traffic behavior for node v_i in the DA-GAE can be constructed as:

$$X(v_i) = [M(v_i, 1), M(v_i, 2), \dots, M(v_i, 7)]. \quad (4.8)$$

In vector $X(v_i)\ \epsilon\ \mathbb{R}^{288\times7}$, each component corresponds to a median value of a particular day of the week, starting from Monday to Sunday.

Simultaneously, topological features are represented by adjacency matrix A, which is derived from the distance measurements between sensors as described in section 4.2. To ascertain the dual graph representation by passing graph features X, A, two stacks of DGCN in GAE encoder extract graph $Embedding1$ and two layers of GAT generates graph $Embedding2$. The process is described by Eqs. 4.9-12.

$$H_1^D = DGCN_1(X, A), \quad (4.9)$$

$$Embedding1 = H_2^D = DGCN_2(H_1^D, A), \quad (4.10)$$

$$H_1^G = GAT_1(X, A), \quad (4.11)$$

$$Embedding2 = GAT_2(H_1^G, A), \quad (4.12)$$

where H_1^D is the intermediate output from the first layer of DGCN component and H_1^G represents the output of the first layer of GAT component. In the proposed GAE, the DGCN-structured module is tasked with reconstructing node content, and its embeddings are considered the primary encoding of graph representation. To improve communication between nodes and edge

embedding, a cross-attention mechanism is used to generate $Embedding3$ by referencing $Embedding2$ and $Embedding1$. The cross-attention [66] take queries from $Embedding2$ and keys and values from $Embedding1$, and compute $Embedding3$. The scaled dot-product attention and multi-head attention are adopted and computed by Eqs.4.13 and 14.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{dk}}\right)V, \tag{4.13}$$

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^0, \tag{4.14}$$

where, $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$; d_k denotes the dimension of queries and keys and d_v is the dimension of values; The parameter matrices are $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^o \in \mathbb{R}^{hd_k \times d_{model}}$.

Dual Encoding Attention GAE Decoder

The GAE decoder is responsible for reconstructing graph inputs from the latent representation. For node features, a DGCN layer transforms $Embedding3$ from the latent space to reconstructed node features, X':

$$X' = H_3^D = DGCN_3(Embedding3, A), \tag{4.15}$$

The loss item for node feature reconstruction is gauged using the mean squared error (MSE) loss:

$$\mathcal{L}_n = \frac{1}{N}\sum_{i=1}^{N}(X(v_i) - X'(v_i))^2, \tag{4.16}$$

where N is the number of data points. For the structure feature reconstruction by the decoder, a simple inner product layer [76] is adopted to reproduce the adjacency matrix A' for link prediction:

$$A'_{ij} = sigmoid(Embedding3_i^{\mathrm{T}} \cdot Embedding3_j), \tag{4.17}$$

the reconstruction loss for structure feature is computed as the binary cross entropy (BCE) loss:

$$\mathcal{L}_s = -\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}[(A_{i,j}\log(A'_{i,j}) + (1 - A_{i,j})log(1 - A'_{i,j})]. \quad (4.18)$$

The weights of GAE module are optimized by minimizing a linear combination of the two reconstruction losses:

$$\mathcal{L}_{cluster} = \varepsilon\mathcal{L}_s + \mathcal{L}_n, \quad (4.19)$$

where ε is a trade-off weight between the two losses. Additionally, a clustering evaluation is conducted using the dual-encoded embedding derived from passing the representative graph into the pre-trained GAE module.

Clustering Assessment

In the ablation experiments, our study compares the proposed DA-GAE model for graph representation learning with alternative methodologies that focus solely on node reconstruction or graph structure reconstruction. To demonstrate the advantages of using DGCN as a GNN base layer, a standard GCN is introduced as a comparison baseline for capturing node embedding. We investigate different clustering methods such as K-Means [77], agglomerative clustering [78], and OPTICS [79]. To evaluate the effectiveness of the embeddings, we apply three commonly used metrics, including the silhouette coefficient score, Calinski-Harabasz index, and Davies-Bouldin index.

- The Silhouette Coefficient (SC) quantifies how well each data point fits into its assigned cluster compared to other clusters, with values ranging from -1 to 1. A high SC value indicates better separation between clusters.

- The Calinski-Harabasz Index (CHI), or variance ratio criterion, assesses the clustering quality by calculating the ratio of the sum of between-cluster dispersion to the intra-cluster dispersion for all clusters. Higher CHI values denote superior clustering, reflecting dense, well-separated clusters.
- The Davies-Bouldin Index (DBI) evaluates clustering quality by measuring the average "similarity" between each cluster and its closest counterpart, with similarity defined as the ratio of within-cluster distances to between-cluster distances. Optimal clustering is indicated by lower DBI values, which suggest that clusters are compact and distinctly separated from each other.

An overview of the baseline models is presented below, including three versions of GAE and Node2Vec. Their differences are highlighted in contrast to our proposed method. The clustering results are presented in Section 4.5.2.

GAE (DGCN): Follows the architecture of DGCN-layered graph auto-encoder in the proposed DA-GAE. However, it does not include the GAT-layered component and only focuses on node content reconstruction.

GAE (GAT): Targets at the reconstruction of the graph structure using the architecture of the GAT-layered encoder and inner product decoder in our proposed DA-GAE. However, the embeddings in the latent space do not interact with DGCN-structured GAE.

GAE (GCN) [80]: The GCN model employs a layer-wise propagation technique that relies on a first-order approximation of spectral graph convolutions. This baseline is comparable to the proposed GAE except that it uses GCN instead of DGCN.

Node2Vec [81]: The essence of Node2Vec revolves round the idea of random walks. This concept is pivotal for deriving node embeddings in graph data. The algorithm conducts random walks on the graph, ensuring nodes are aware of their neighbors in a balanced way.

4.4.4. CG-DGAE based Traffic Sensor Data Reconstruction

Spatial-temporal graph neural networks (STGNNs) have shown their feasibility and robustness in reconstructing traffic sensor data with spatio-temporal dependencies. However, detecting faults in traffic data require consistent temporal information. The results from sensor clustering allow for the optimal utilization of shared temporal data within smaller groups. Additionally, denoising graph autoencoders have demonstrated their effectiveness in managing missing data, enhancing resistance to noise, and focusing the autoencoder's attention on specific nodes of a graph. This makes them well-suited for tasks that involve reconstructing data to detect faults. In this section, we present the CG-DGAE model, specifically designed and trained to reconstruct input traffic sequence data. Leveraging the nodes identified for each cluster in the previous section, subgraphs corrupted by artificial faults are generated for every cluster to facilitate DGAE training. By comparing the reconstructed sequence with the input sequence, data faults can be detected using a proper fault score function.

Fault Types

To improve the generalization and stability of the CG-DGAE model, faults are inserted to the node content during the training phase. We draw from three commonly observed fault categories found in historical continuous count station (CCS) data, nonresponsive faults, block faults, and point faults, as previously defined in [4]. Environmental conditions like weather-induced damage, along with technical issues such as power disruptions and incorrect sensor configurations, can lead to nonresponsive faults, which are characterized by sensor inactivity. Block faults, indicated by diminished traffic readings, may stem from physical sensor blockages, misalignments, and calibration errors. Point faults are random and sporadic, caused by hardware issues, fluctuating power supplies, or temporary obstructions. To simulate a nonresponsive fault, traffic volumes were set to zero for randomly selected short periods, each lasting between 5 and 20 consecutive 5-minute intervals. Block faults were generated similarly, but instead of setting the traffic volumes to zero, they were reduced by 40% to 60%. Point faults were introduced by reducing traffic volumes by 60% to 100% at randomly selected 5-minute intervals, with a sampling rate of 5 to 30 intervals per day. **Figure 4.5** demonstrates the impact of these three types of faults on signal integrity, with blue lines representing normal data sequence and orange lines showing the faulty signals.

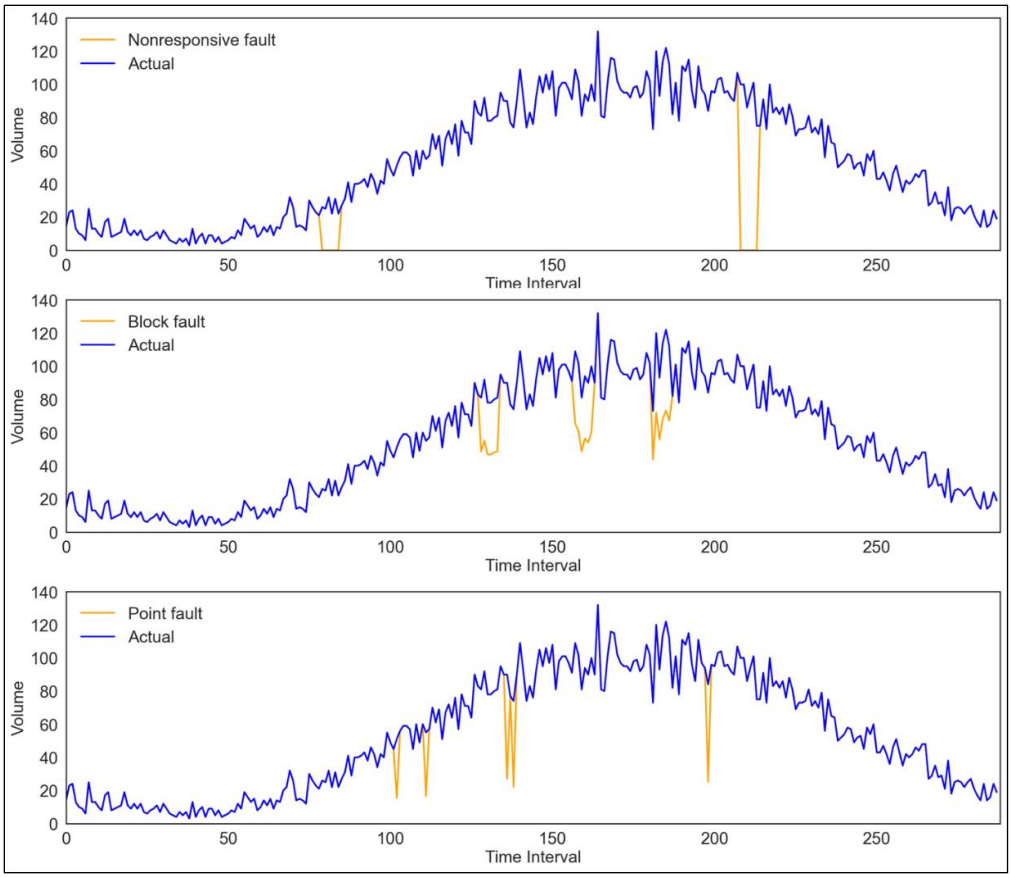

Figure 4.5: Visualization of three types of faulty signals; top: nonresponsive fault, middle: block fault, bottom: point fault.

Cluster-Guided Denoising Graph Auto-Encoder

Graph auto-encoder has been widely adopted for addressing traffic-related problems in spatial and/or temporal domains. In our setting, the GAE can leverage spatial dependency of neighboring nodes (i.e., continuous traffic count stations) and temporal traffic patterns at each node to identify potential data faults. Our methodology is established on the premise that data points that are spatially and temporally proximate tend to bear more resemblance to each other than those that are further apart. Different from traditional sampling strategies for GAE model training, we propose a cluster-guided sampling strategy, where all subgraphs are constructed from the clusters found in the previous section. To simulate faulty data scenarios, node contents are randomly selected for

contamination at batch level with three known fault types. The "contaminated" subgraphs are then passed to the GAE. The training objective is to reconstruct the original subgraphs from the contaminated subgraphs. The overall information flow of training CG-DGAE is illustrated in **Figure 4.6**.

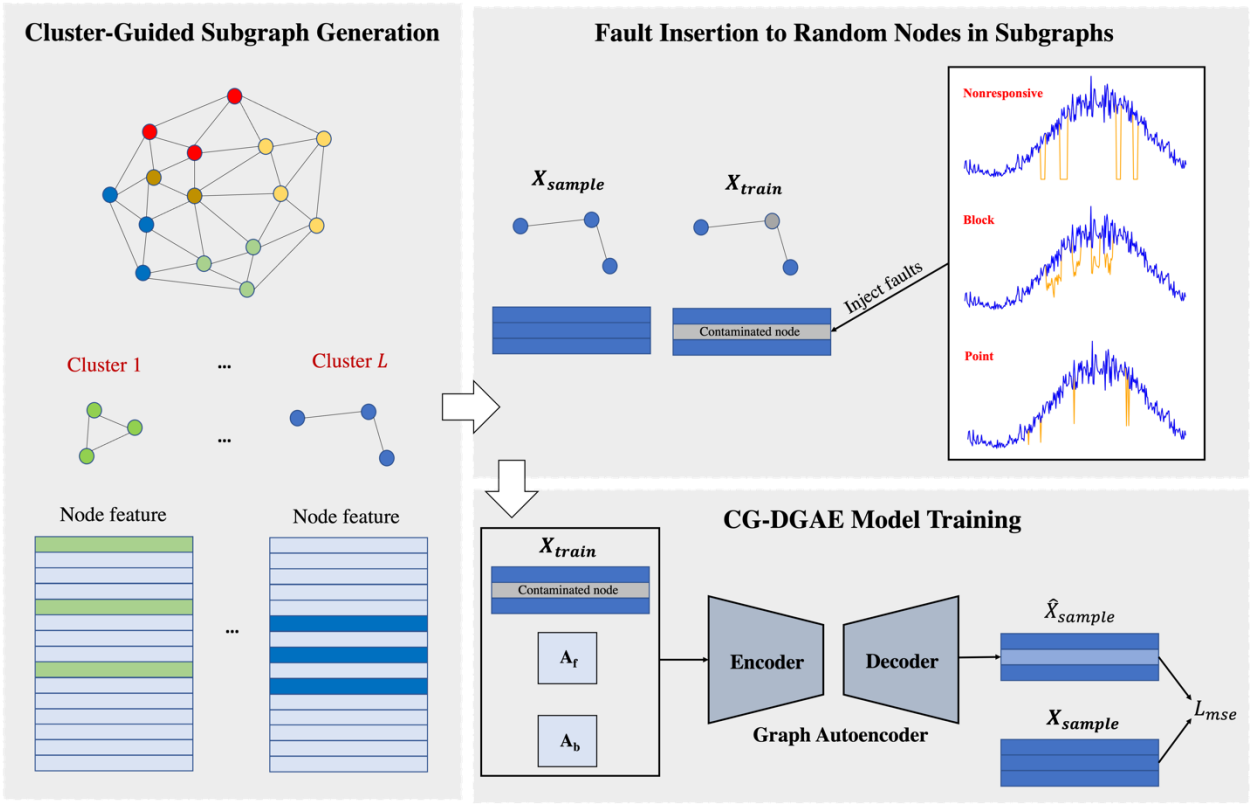

Figure 4.6: CG-DGAE training.

Subgraph Signals and Random Fault Injection: Considering the entire highway traffic sensor network, ensuring the availability of complete temporal information from all sensors presents a challenge. A pragmatic approach to address this data scarcity involves focusing on a cluster-level analysis to identify a common time window of sufficient length. To equip the CG-DGAE with the flexibility to adapt to varying patterns of sensor faults, three types of faults

identified in the CCS traffic data are introduced into the selected node signals, featuring random lengths and frequencies.

To assess the impact of the contamination proportion on node content, faulty signals are injected into the training subgraphs randomly selected within each training batch at a specified ratio γ. Within these selected subgraphs, a node is randomly chosen, and faults are introduced into its content. **Algorithm 1** below is used for generating training subgraphs.

Algorithm 1: Generating Subgraph Samples for Reconstruction Model Training

Algorithm 1: Generating subgraph samples for training

Input: Traffic data $X \in \mathbb{R}^{N \times P}$ over period $[0, P-1]$
Number of clusters = L,
Fault insertion function: $\eta(X, option)$, where $option \in \{nonresponsive,\ block,\ or\ point\}$,
Reconstruction window size: h,
Batch size: B,
Number of iterations: I,
Contamination ratio: γ.

for $i = 0 : I-1$ **do**
  for $\mathcal{g} = 0 : L-1$ **do**
    Find the common time window $X_T \in \mathbb{R}^{Z \times T}$ shared by Z sites in cluster $\mathcal{g}$.
    for $k = 0 : \big(T \parallel (h \times B)\big) - 1$ **do**
      Randomly sample distinct $\{b^0, \dots b^{B \times \gamma - 1}\}$ indexes within range $[0, B-1]$.
      Randomly sample distinct starting points $\{j^0, \dots j^{B-1}\}$ from $[0, T-h]$.
      for $j\ in\ \{j^0, \dots, j^{B-1}\}$ **do**
        Obtain submatrix $X^j_{sample} = X_T[:, h \times j : h \times (j+1)]$with the size of $Z \times h$.
        if $j\ in\ \{b^0, \dots b^{B \times \gamma - 1}\}$ **do**
          Inject faults to a randomly selected site m from Z: $\eta(X^j_{sample}[m, :], option)$.
        end if
      end for
      Return $\left\{X^{j^0 : j^{B-1}}_{sample}\right\}$ with size of $Z \times h \times B$ as minibatch for CG-DGAE training.
    end for
  end for
end for

GAE Architecture A CG-DGAE model is trained to reconstruct the full matrix X_{sample} on a subgraph given the contaminated signals X_{train}. In our work of reconstructing traffic sequences, we do not explicitly employ sequence-learning models to discern temporal dependencies. This approach is justified firstly by the relatively brief duration of the recovery

window h, leading us to treat all instances within this window as interrelated, effectively considering a signal of length h as having h features. Secondly, given the variable sizes of the input matrices, representing subgraphs, a standardized sequence-learning model is not feasible. The DGCN, detailed in section 5.2, is employed as the GNN base layer of the GAE architecture to capture the random nature of spatial and directional dependencies. **Figure 4.7** showcases the architecture of the GAE that includes an encoder of 3-DGCN base layers and a corresponding symmetrical 3-DGCN base layer decoder, where each layer of the encoder has a skip connection [82] to the corresponding layer of the decoder, enhancing the model's ability to capture and reconstruct complex patterns.

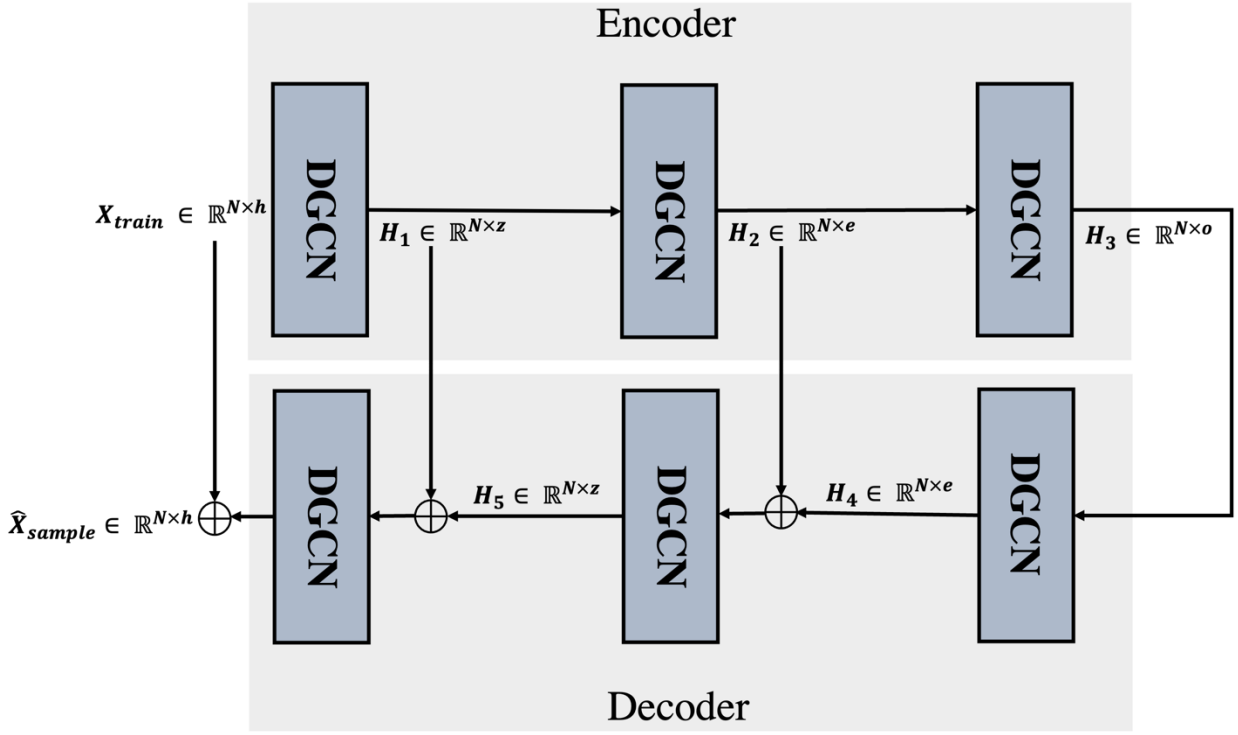

Figure 4.7: The graph autoencoder architecture in CG-DGAE.

Loss Function The goal of the CG-DGAE is to rebuild signals from sampled sensors within a cluster. Our loss function is simply the mean reconstruction error on all nodes within each cluster.

Reconstruction Assessment

To validate the efficacy of the proposed CG-DGAE, we compared it with different GNN or non-GNN baselines. The details are presented in **Table 4.2**.

GNN baselines include CG-DGAE, R-DGAE, and RW-DGAE, differing in the different sampling strategies, further tested with different base layer GCN or DGCN. The main purpose of GNN baselines is to demonstrate the superiority of our proposed CG-DGAE with DGCN base layer, in which cluster-guided information is crucial to the fault detection and better than other sampling strategies.

The non-GNN baselines include BiLSTM (Bi-directional Long Short-Term Memory), CNN-BiLSTM (Convolution-BiLSTM), and Okriging (Ordinary Kriging). These methods simply leverage either spatial or temporal dependency to regress or extrapolate sensor data. Each of these aspects are discussed in detail below.

GNN baselines:

1) Sampling strategies: Sampling techniques are pivotal in enhancing the efficiency and the scalability of GNN training. Two prevalent node-wise sampling techniques including both random sampling (RS) and random walk sampling (RW) are used as benchmarks to compare with our proposed DA-GAE, which uses cluster guided sampling strategy.

a) Random sampling (RS): This RS sampling, proposed in GraphSAGE [83], relies on a random sampling of nodes to generate the subgraphs for training GNNs.

b) Random walk sampling (RW): RW-sampling was initially employed in GraphSAINT [84]. It depends on a stochastic process that begins at a random node and explores its neighbors in a sequence of steps, forming a trajectory known as a random walk. The sampled subgraphs obtained through these walks are representative of the larger graph's topology and are used to construct subgraphs for training GNN.

2) GNN base layer: To demonstrate the superiority of the DGCN in our proposed CG-DGAE, we considered the original GCN [80] as a GNN structure baseline.

Non-GNN baselines:

1) Temporal method: The RNN has achieved notable success in capturing time-series features. Modules such as Long Short-Term Memory (LSTM) [85], and its derivative BiLSTM [86] are known for good performance with temporal modeling. In this study, we evaluated the efficacy of BiLSTM and CNN-BiLSTM [87] for comparison with GNNs regarding the temporal modeling, trained in an end-to-end manner at a sensor level.

2) Spatial method: To examine the effectiveness of reconstructing time series using only spatial correlations among traffic sensors, a kriging approach is chosen as a baseline: ordinary kriging (OKriging) [88]. OKriging applies a weighted average of observed values from nearby locations, where the weights are derived from the spatial autocorrelation function. In this experiment, we utilize the geographical locations of test CCS sites to determine the neighboring sensors for kriging purposes.

Table 4.2: Our proposed CG-DGAE and baseline models for comparison

Model	GCN base layer	Sampling strategy	Spatial	Temporal
CG-DGAE	DGCN	Cluster-guided	✓	✓
	GCN	Cluster-guided	✓	✓
R-DGAE	DGCN	RS	✓	✓
	GCN	RS	✓	✓
RW-DGAE	DGCN	RW	✓	✓
	GCN	RW	✓	✓
BiLSTM	-	-	-	✓
CNN-BiLSTM	-	-	-	✓
OKriging	-	-	✓	-

For the phase of graph reconstruction, the performance of CCS sequences is measured with two metrics: mean absolute error (MAE) and root mean square error (RMSE) using test data from **GR-dataset**.

4.4.5. Traffic Sensor Data Fault Detection

The goal of traffic fault detection is to identify faults within the sensor traffic data sequence. The pretrained CG-DGAE reconstruction model described in Subsection 4.4.4 is used to generate a reconstructed sequence, which is then compared with the input sequence to detect faults. A fault

score and a pre-defined threshold are employed for this purpose. The following subsections introduce the fault score and the assessment metrics for fault detection.

Fault Score

A fault score is employed to locate faults within a traffic sequence. This score is derived by computing the difference between the sensor data sequence and the sequence reconstructed by the pre-trained CG-DGAE. Through our analysis of difference distribution, anomalies indicative of faults exhibits a distinct pattern. While the Z-score is a standard measure for identifying outliers in one-dimensional data, a modified Z-score has been introduced for more accurate evaluation of anomalies in the difference of two sequences [89], [90]. Given a sensor data sequence $X_{t-T:t} = \{X_{t-T}, X_{t-T+1}, \dots, X_t\}$ and the reconstructed sequence $\hat{X}_{t-T:t} = \{\hat{X}_{t-T}, \hat{X}_{t-T+1}, \dots, \hat{X}_t\}$, for a data point X_t, the fault score can be written as follows:

$$score(t) = \frac{\beta \times (RD_t - median(RD))}{MAD(RD)} \tag{4.20}$$

where $RD_t = \frac{X_t - \hat{X}_t}{\hat{X}_t}$ is the relative difference at the timestamp t, and $median(RD)$ is the median of the relative difference sequence RD; $MAD(RD)$ is the median absolute deviation of RD, defined as $MAD(RD) = median(|RD - median(RD)|)$; the value of consistency correction β depends on the underlying distribution of the data. In our work, $\beta = 0.6745$ is adopted. The score threshold was determined experimentally by evaluating a range of values. It is important to calibrate this threshold using local data to account for region-specific data distribution.

Fault Detection Assessment

For the phase of fault detection, by referencing the selected natural faulty data from **FD-dataset**, each data point in a daily CCS sequence is assigned a binary label to indicate its faulty status. As such, the detection of faulty data points becomes a binary classification problem. The effectiveness of fault detection across all CCS sequences in the **FD-dataset** is then quantified using accuracy, recall, F1 score, precision, and AUC_ROC. The concept of CCS sequence binary annotation is illustrated in **Figure 4.2**.

4.5. Experiments

To assess the performance of our proposed approach for traffic sensor clustering, traffic data reconstruction, and fault detection, this section provides a comparative analysis for the graph clustering model DA-GAE and the graph reconstruction model CG-DGAE. The comparison involves different GNN base layers and non-graph baselines. Details on the experimental settings and results are presented subsequently.

4.5.1. Experiment Settings

All experimental comparisons are performed on a system equipped with an AMD Ryzen Threadripper PRO 5955WX 16-Core CPU operating at 1.794 GHz base and 4.0 GHz max frequencies, and 32 GB of L3 cache. To ensure the fairness of comparison, the comparative GNN models in clustering and reconstruction phases are trained and tested with the same hyperparameters depicted in **Table 4.3** for DA-GAE and CG-DGAE, respectively.

Table 4.3: Hyperparameter settings for DA-GAE and CG-DGAE

DA-GAE Hyperparameter	Value	CG-DGAE Hyperparameter	Value
Input time window length	288	Input time window length	288
Order of Chebyshev approximation R	2	Order of Chebyshev approximation R	2
DGCN hidden dimension 1	32	DGCN hidden dimension 1	144
DGCN hidden dimension 2	16	DGCN hidden dimension 2	72
GAT hidden dimension	16	DGCN hidden dimension 3	16
Batch size	4	Batch size	64
Learning rate	0.00001	Learning rate	0.0001
Trade-off weight ε	0.08	Optimizer	'Adam'
Optimizer	'Adam'	Maximum number of iterations	100
Maximum number of iterations	1500	Fault score reference threshold	28

4.5.2. Results

Sensor Clustering

In this section, we compared the proposed dual-encoding attention graph auto-encoder (DA-GAE) based embeddings with other embedding baselines. In terms of clustering results in the embedding space. Three different clustering algorithms, including K-Means, agglomerative clustering (AC), and OPTICS, are adopted from this assessment. The findings are summarized in **Table 4.4**.

The results show that all graph embedding methods under K-Means and AC yield comparable cluster counts and cluster scores. Our DA-GAE, particularly when paired with K-Means, achieved the highest SC value at 0.9786, the highest Calinski-Harabasz index (CHI) at 5.0747e+11, and the lowest Davies-Bouldin index (DBI) at 0.0001, resulting in a total of 169 clusters. DA-GAE consistently scores highest across the three chosen clustering algorithms, outperforming GAT-based GAE, DGCN-based GAE, GCN-based GAE, and Node2Vec. This superiority indicates that the fusion of graph embeddings through dual-encoding attention

mechanisms enhances graph representation as compared to the methods focusing solely on node reconstruction or graph structure reconstruction.

The comparison between DGCN-based GAE and GCN-based GAE, suggests that DGCN is more effective at capturing graph features for node content reconstruction for our dataset as compared to GCN. However, GAT-based GAE shows marginally superior clustering metrics compared to DGCN-based GAE, indicating that within our experimental framework, embeddings derived from graph structure reconstruction offer a better representation than those from node content reconstruction. Node2Vec yields the lowest clustering scores, highlighting the benefits of GAE-based graph embedding methods over topology-based approaches. The cluster size distribution is presented in **Figure 4.8**, showing most clusters contain bi-directional nodes, typically corresponding to two directions of same CCS site, which is intuitively expected.

Table 4.4: Clustering performance (SC/CHI/DBI) comparison of different embedding methods.

Graph embedding	GNN base layer	Clustering algorithms	# Of clusters	SC	CHI	DBI
DA-GAE	GAT & DGCN	K-Means	**169**	**0.9786**	**5.0747e+11**	**0.0001**
		AC	169	0.9776	5.0020e+11	0.0002
		OPTICS	41	0.3411	49.6106	0.8465
GAE	DGCN	K-Means	210	0.8533	1.9270e+07	0.0078
		AC	210	0.8521	1.9249e+07	0.0078
		OPTICS	39	0.0928	25.4319	1.1748
	GAT	K-Means	166	0.9387	3.2871e+07	0.0051
		AC	166	0.9397	3.2743e+07	0.0048
		OPTICS	41	0.3289	26.5911	1.1219
	GCN	K-Means	213	0.6002	2272.9946	0.2848
		AC	213	0.6046	2420.7861	0.2880
		OPTICS	29	0.1828	14.6042	1.7310
Node2Vec	-	K-Means	161	0.0714	3.5652	1.2428
		AC	161	0.1234	4.2118	1.2819
		OPTICS	4	0.0558	3.3534	2.3142

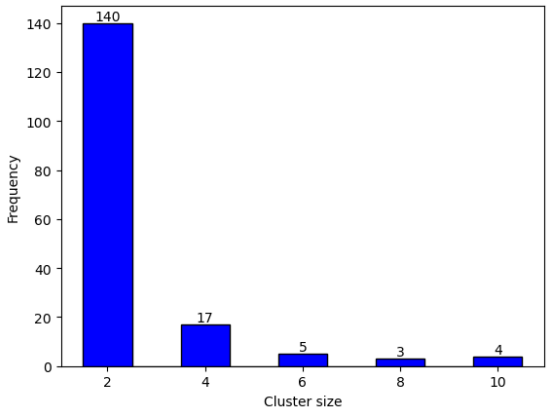

Figure 4.8: The distribution of clusters by size.

Data Reconstruction

In this section, the robustness of our proposed CG-DGAE with DGCN as a base layer is evaluated against baseline models under different batch-wise contamination ratios, ranging from 0.20 to 0.80. The testing data from **GR-dataset** is firstly subjected to three summarized types of faults (nonresponsive, block, and point) as the input to the models. The original normal daily sequences serve as label for reconstruction. The comparative reconstruction performance of the different models is quantitatively evaluated and presented in **Table 4.5** and **Figure 4.9**. This evaluation is based on the computation of mean MAE and RMSE across all site data. To account for variability, all results presented are the averages of three separate experimental runs. The findings reveal that the CG-DGAE model with DGCN as a base layer consistently outperforms other models across all contamination ratios in aspects such as sampling strategies, GCN base layers, and spatial or temporal dependencies, with the lowest mean MAE (6.00) and mean RMSE (16.28) values. Its performance peaks at a 50% contamination ratio, suggesting that the proposed model is robust to noise and can maintain accuracy even with relatively large data contamination.

Generally, GNN models exhibit superior performance over non-GNN models like BiLSTM, CNN-BiLSTM, and Okriging, which focus solely on either topology or temporality. Within GNN models, the base layer structured with DGCN consistently achieve lower MSE and RMSE across all contamination ratios, compared to GCN models, indicating DGCN's enhanced ability to leverage spatial-temporal information. For GNN models using the same base layer, the cluster-guided sampling strategy consistently yields the best results, followed by random walk and random sampling for each contamination ratio. This highlights the effectiveness of our previously derived clustering results in guiding subgraph sampling, enabling the GNN model to better aggregate hidden correlations in traffic data with diverse spatial features than random or random-walk sampling. Regarding the contamination ratio, for most models, as the contamination ratio increases, there is a noticeable deterioration in performance, as indicated by the rising MAE and RMSE values. This trend highlights that higher levels of noise in the data have a detrimental effect on the models' ability to reconstruct the sequences. Nevertheless, a dynamic variation in performance was observed across noise ratios. The performance of most baseline models reaches their nadir at a 40% contamination ratio. In contrast, the CG-DGAE with DGCN base layer achieves its minimal values at a contamination ratio of 50% during training. Relative to other baseline models, the CG-DGAE exhibits enhanced resilience to noise or contamination.

Moreover, the BiLSTM and CNN-BiLSTM models exhibit considerably higher MAE and RMSE values at all levels of contamination as compared to the CG-DGAE with a DGCN base layer revealing temporal models' relative inefficiency in handling contaminated data for this specific task. In contrast, the Okriging model only leverage spatial information, resulting in much higher MAE and RMSE. Conclusively, the baseline models do not match the performance of CG-DGAE with DGCN as a base layer for any contamination ratio in reconstruction of the CCS data

sequences. The experimental results underscore the success of the cluster-based sampling strategy during the graph training phase. The DGCN demonstrates its superiority over GCN as a base layer. This confirms the strength and effectiveness of our proposed CG-DGAE (DGCN) and training strategy in capturing the spatial-temporal relationships within network-level traffic data.

Table 4.5: Reconstruction performance (MAE and RMSE) comparison of various models under different contamination ratios on test set. Results averaged over 3 independent runs.

Model	Base layer	Metrics	Contamination ratios γ						
			0.20	0.30	0.40	0.50	0.60	0.70	0.80
CG-DGAE(DGCN)	DGCN	MAE	6.25	6.08	6.04	**6.00**	6.06	6.28	7.20
		RMSE	17.58	16.79	16.35	**16.28**	16.33	16.34	16.42
R-DGAE(DGCN)	DGCN	MAE	21.45	21.01	**20.23**	24.58	23.77	24.28	25.69
		RMSE	58.76	55.32	**52.98**	54.16	55.10	55.38	57.18
RW-DGAE(DGCN)	DGCN	MAE	20.99	**20.89**	23.48	22.63	22.72	24.75	22.47
		RMSE	51.87	**51.12**	54.49	52.56	53.00	57.85	52.04
CG-DGAE(GCN)	GCN	MAE	15.14	15.24	**15.09**	15.30	15.22	15.30	15.23
		RMSE	21.82	22.00	**21.77**	22.16	21.99	22.12	21.96
R-DGAE(GCN)	GCN	MAE	**29.59**	38.82	46.02	62.94	35.05	43.93	37.31
		RMSE	**60.88**	66.78	67.54	81.73	63.84	67.19	64.80
RW-DGAE(GCN)	GCN	MAE	**24.65**	26.95	27.01	33.64	53.40	51.39	35.55
		RMSE	**56.41**	57.74	58.42	60.92	72.75	72.78	65.28
BiLSTM	-	MAE	40.90	41.88	**37.66**	38.78	38.05	39.44	40.59
		RMSE	47.30	48.21	**44.28**	45.08	44.41	45.98	46.81
CNN-BiLSTM	-	MAE	38.90	39.35	**35.37**	36.90	36.23	37.77	38.93
		RMSE	45.66	46.58	**42.12**	43.77	43.20	44.02	45.10

*Okriging model enables the reconstruction of traffic sequences at a specific CCS site without requiring site specific data input, thereby eliminating the need for contamination. The approach yields a mean RMSE of 146.91 and a mean MAE of 126.82.

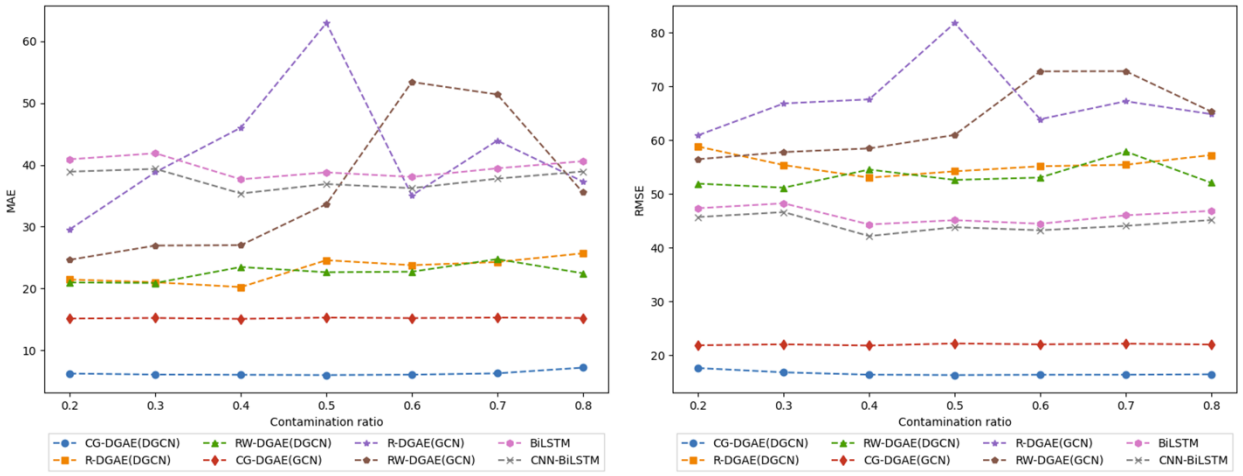

Figure 4.9: Left: mean MAE comparison for different contamination ratios on testing set of GR-dataset; right: mean RMSE comparison for different contamination ratios on testing set of GR-dataset. Results averaged over 3 independent runs.

Fault Detection

To demonstrate the effectiveness of the fault detection results of the pretrained CG-DGCN with DGCN as the base layer, we compare the fault detection performances between GNN-based models utilizing two distinct base layers, DGCN and GCN, across three subgraph sampling strategies, namely cluster-guided, random, and random walk. We manually labeled five natural sequences containing faulty data points in the **FD-dataset**. These sequences were specifically annotated with point-wise binary labels. As a result, the faulty data point detection is evaluated in the form of binary classification. The results are presented in **Table 4.6**.

The models with the DGCN base layer contain significantly more parameters, 243,000 compared to 49,000 for those with the GCN base layer. Despite this disparity, both models are still considered lightweight and can be effectively deployed on edge computing devices. The CG-DGAE model with DGCN base layer achieved an impressive overall accuracy of 99.09%, with a precision of 99.13%, a recall of 99.53%, and a F1 score of 99.53%, underscoring its superior fault

detection capability even being trained at a higher contamination ratio of 0.50. The models employing the RS and RW sampling strategies with a DGCN base layer, namely RS-DGAE and RW-DGAE, showed a slightly decreased performance.

In comparison to the DGCN, the GCN-based models, despite their impeccable Recall, underperformed in identifying faulty points, which is a critical aspect of the fault detection task. The diminished accuracies for the GCN models indicate that the DGCN offers a stronger discriminative capacity for time-series data. Furthermore, these results highlight the critical role that the base layer and sampling strategy play in a model's competency to handle contaminated datasets and accurately reconstruct the intrinsic data structure for data imputation and fault detection. **Figure 4.10** depicts the observed faulty sequences from various cluster sizes within **FD-dataset** alongside their reconstructions by the pretrained CG-DGAE with DGCN base layer. The model's robustness is evident in its reconstruction of time series data, particularly its adeptness at reflecting fine patterns as well as overarching trends.

Table 4.6: Faulty detection performance of GNN-based models

Model	Base layer	Number of Parameters (in 1,000s)	Contamination ratio*	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
CG-DGAE	DGCN	243	0.50	**99.09**	**99.13**	**99.92**	**99.53**
RS-DGAE	DGCN	243	0.40	97.56	98.20	99.27	98.74
RW-DGAE	DGCN	243	0.30	97.57	98.06	99.41	98.73
CG-DGAE	GCN	49	0.40	95.69	95.68	100.00	97.80
RS-DGAE	GCN	49	0.20	95.63	95.62	100.00	97.76
RW-DGAE	GCN	49	0.20	95.63	95.62	100.00	97.96

* Data contamination ratios used for model training that result in the best reconstruction performance.

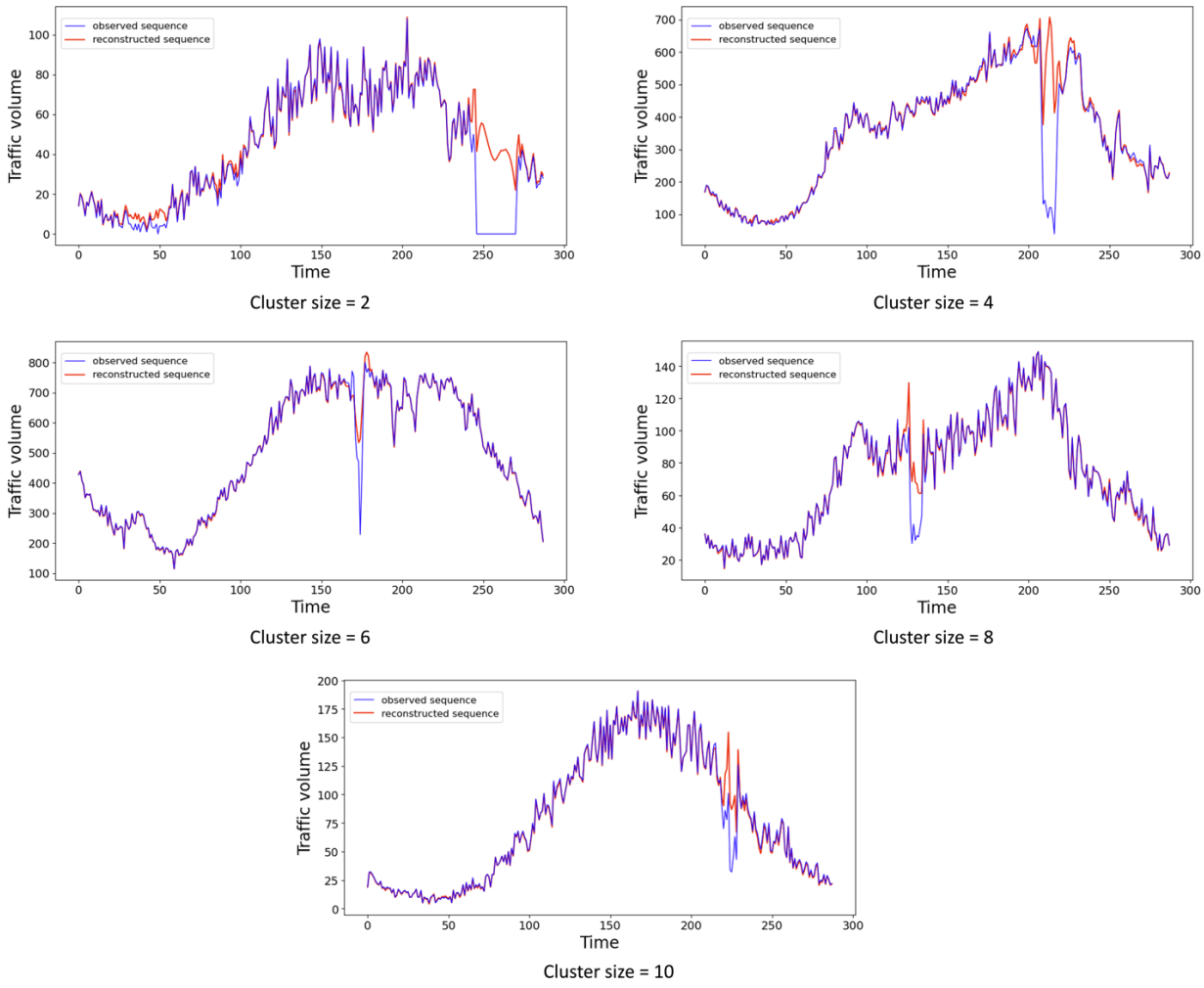

Figure 4.10: Reconstruction visualization of five natural faulty sequences from different-sized cluster by the pretrained CG-DGAE with DGCN base layer.

4.6. Summary

Traffic sensor data is crucial for transportation planning, design practices, and the development of modern intelligent transportation systems (ITS). Ensuring the quality of traffic sensor data is essential, yet current methods relying on simple heuristic rules are insufficient. This chapter introduces a novel approach to enhance the quality and reliability of traffic data through a dual-encoding attention graph auto-encoder (DA-GAE) model for traffic sensor clustering and a cluster-guided denoising graph auto-encoder (CG-DGAE) model for traffic data reconstruction. The DA-

GAE model is designed for advanced graph representation learning, efficiently clustering traffic sensors and facilitating a reliable subgraph sampling strategy for GNN training. The robustness of CG-DGAE model is attributable to the strategic amalgamation of elements, including GNN base layer, training strategy, and contamination ratio. Our experimental evaluation, using real-world datasets, benchmarks the performance of the proposed models against baseline methods, focusing on clustering efficacy, data reconstruction fidelity, and fault detection accuracy.

Based on the experimental results, several conclusions can be drawn:

- The DA-GAE model demonstrates significant effectiveness in clustering traffic sensors. Our innovative approach of integrating joint embeddings into the graph's latent space from both node and structure reconstruction advances graph representation learning.
- The proposed CG-DGAE methodology, utilizing diffusion graph convolutional networks (DGCN), excels in capturing the hidden characteristics of traffic data and effectively distinguishes between normal and faulty data. This bolsters the performance of both data imputation and fault detection.
- The implementation of a cluster-guided sampling strategy enhances the GNN's ability to discern spatial-temporal dependencies among traffic sensors, which is largely due to the cluster-wise spatial-temporal context leveraged by the model.

Beyond these areas mentioned above, future research could explore further architectural enhancements, such as transformer-based models, to improve performance. Additionally, enriching node features by incorporating a broader spectrum of traffic data types and sources could enhance model accuracy. While this study is based on fixed location sensors, future work should

consider dynamic graphs with probe sensors, particularly in light of emerging connected and autonomous vehicle technologies. These advancements are expected to further improve both data imputation and fault detection performance.

4.7. Publications

The work presented in this chapter has led to the following publications [91]:

- Huang, Yongcan, Hao Zhen, and Jidong J. Yang. Cluster-guided denoising graph auto-encoder for enhanced traffic data imputation and fault detection[J]. Expert Systems with Applications (2024): 125531.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Given the increasing need for quality control in handling the growing volume of traffic sensor data, this dissertation focuses on developing Artificial Intelligence (AI)-powered fault detection frameworks to ensure the integrity of traffic sensor data. The dissertation begins with a comprehensive review of existing studies, examining key technologies in the field, assessing their feasibility, and outlining current challenges. Two novel approaches are introduced subsequently for handling data from Continuous Count Stations (CCS). The first framework addresses fault detection at the individual sensor level using time-series data imaging, computer vision algorithms, and a contrastive learning scheme to analyze daily CCS sequences. In contrast, the second framework targets data fault detection at the cluster level, employing Graph Neural Networks (GNNs) to leverage the spatial-temporal correlations among traffic sensors within each cluster. Experimental results demonstrate the effectiveness of these proposed methodologies, highlighting their potential to enhance data quality and improve decision making in transportation management. The proposed frameworks are quite generic and can be extended to sensor data beyond the traffic domain.

5.1. Limitations and Outlook

This dissertation fulfills the proposed research goal of automatic quality control system for traffic sensor network with the focus of fault detection. However, we recognize areas for potential improvement that can be addressed in future work:

- Fault Type Evaluation: For both Framework 1 and 2, our approach currently frames fault detection as a binary classification task without evaluating performance across different fault types. Future work will focus on quantifying the system's effectiveness in detecting specific types of faults in traffic sensor data.
- Test Set Limitations: In the Framework 2, the current test set for fault detection includes only five daily CCS site sequences from clusters of varying sizes, all derived from real-world data. Expanding the number of natural faulty sequences would further validate the effectiveness of our three-phase fault detection system.
- Clustering Sensitivity: In Framework 2, while our study centers on a cluster-guided graph autoencoder reconstruction-based fault detection system that leverages spatio-temporal correlations in traffic sensor data, we have not yet explored the sensitivity of fault detection to the clustering results. Future research will assess how the quality of clustering impact's fault detection performance.

5.2. Future Work

As outlined in Section 5.1.2, future work will build on the limitations discussed by further deepening and broadening the study of the data quality control system for traffic sensor networks. First, as more data becomes available to project traffic snapshots, we plan to expand our dataset to

further validate the feasibility of the proposed frameworks. Second, while the current data primarily relies on Continuous Count Stations (CCS) records, additional information such as roadway profiles, weather conditions, and crash data could be incorporated to enrich the model's understanding and improve its ability to capture variations in inputs and outputs. To illustrate some of the improvements, we conducted a preliminary experiment on sensor clustering by incorporating roadway and regional information, and distinguishing truck from non-truck traffic. With the updated clustering framework, based on the dual-encoding attention graph auto-encoder (DA-GAE), the results are presented in Table 5.1. By including urban/rural data, lane counts, functional classification, AADT, and separating truck and non-truck traffic series, along with reconstructing these two series, the clustering results have shown improvements compared to those in Section 4.5.2. Consequently, future work will focus on node feature sensitivity analysis and explore the use of multi-modal sensor data, alongside advanced deep learning models, to develop an optimal quality control system for multi-modal traffic sensor data. Third, we aim to collect more naturally occurring faulty data and classify each instance into specific fault types, which will enhance the model’s ability to reason about different types of faults. Most existing literature does not clearly differentiate between anomalies and faults in traffic data. Faults refer to unusable data that require detection and correction, while anomalies cover a broader range of issues, including special events and incidents that cause irregularities in the data. Future work will focus on identifying such types of anomalies by integrating additional data sources. This approach will not only improve detection accuracy but also offer valuable insights for incident detection and traffic management.

Table 5.1: Clustering performance (SC/CHI/DBI) comparison of different embedding methods under the updated DA-GAE

Graph Embedding	GNN Base Layer	Clustering Algorithms	# Of clusters	SC	CHI	DBI
DA-GAE	GAT & DGCN	K-Means	**175**	**0.9872**	**5.7013E+11**	**0.0001**
		AC	175	0.9863	5.6013E+11	0.0001
		OPTICS	35	0.3551	53.6296	0.8376
	DGCN	K-Means	212	0.8672	2.0201E+07	0.0075
		AC	232	0.8612	2.0321E+07	0.0072
		OPTICS	24	0.0954	33.1181	1.0324
GAE	GAT	K-Means	166	0.9387	3.2902E+07	0.0051
		AC	166	0.9397	3.2710E+07	0.0048
		OPTICS	41	0.3289	26.5911	1.1219
	GCN	K-Means	231	0.8577	2.1256E+04	0.1223
		AC	232	0.8581	2.6410E+04	0.1171
		OPTICS	120	0.6744	634.3424	0.5388
Node2Vec	-	K-Means	183	0.3807	67.7533	0.7377
		AC	178	0.3912	69.5368	0.7459
		OPTICS	25	0.0641	18.4239	1.2873

BIBLIOGRAPHY

[1] K. Ni et al., “Sensor network data fault types,” ACM Transactions on Sensor Networks (TOSN), vol. 5, no. 3, pp. 1–29, 2009.

[2] Y. Xu, Z. Li, S. Wang, W. Li, T. Sarkodie-Gyan, and S. Feng, “A hybrid deep-learning model for fault diagnosis of rolling bearings,” Measurement, vol. 169, p. 108502, 2021.

[3] J. Song, Y. C. Lee, and J. Lee, “Deep generative model with time series-image encoding for manufacturing fault detection in die casting process,” J Intell Manuf, vol. 34, no. 7, pp. 3001–3014, 2023.

[4] C. Morris, J. J. Yang, M. G. Chorzepa, S. S. Kim, and S. A. Durham, “Self-Supervised Deep Learning Framework for Anomaly Detection in Traffic Data,” J Transp Eng A Syst, vol. 148, no. 5, p. 04022020, 2022.

[5] M. N. Hasan, S. U. Jan, and I. Koo, “Wasserstein GAN-based Digital Twin Inspired Model for Early Drift Fault Detection in Wireless Sensor Networks,” IEEE Sens J, 2023.

[6] L. Xin, S. Haidong, J. Hongkai, and X. Jiawei, “Modified Gaussian convolutional deep belief network and infrared thermal imaging for intelligent fault diagnosis of rotor-bearing system under time-varying speeds,” Struct Health Monit, vol. 21, no. 2, pp. 339–353, 2022.

[7] Z. Fang, Q. Long, G. Song, and K. Xie, “Spatial-temporal graph ode networks for traffic flow forecasting,” in Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 364–373.

[8] Z. Lu et al., “Graph sequence neural network with an attention mechanism for traffic speed prediction,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 13, no. 2, pp. 1–24, 2022.

[9] S. Yang and B. Yang, “An inductive heterogeneous graph attention-based multi-agent deep graph infomax algorithm for adaptive traffic signal control,” Information fusion, vol. 88, pp. 249–262, 2022.

[10] L. Yu, B. Du, X. Hu, L. Sun, L. Han, and W. Lv, “Deep spatio-temporal graph convolutional network for traffic accident prediction,” Neurocomputing, vol. 423, pp. 135–147, 2021.

[11] J. R. Taylor and H. L. Loescher, “Automated quality control methods for sensor data: a novel observatory approach,” Biogeosciences, vol. 10, no. 7, pp. 4957–4971, 2013.

[12] B. Gunay and G. Erdemir, “Using wavelet transforms for better interpretation of traffic simulation,” Traffic Engineering and Control, vol. 50, no. 10, pp. 450–453, 2009.

[13] Z. Zheng, S. Ahn, D. Chen, and J. Laval, “Applications of wavelet transform for analysis of freeway traffic: Bottlenecks, transient traffic, and traffic oscillations,” Transportation Research Part B: Methodological, vol. 45, no. 2, pp. 372–384, 2011.

[14] D. Jiang, C. Yao, Z. Xu, and W. Qin, “Multi‐scale anomaly detection for high‐speed network traffic,” Transactions on Emerging Telecommunications Technologies, vol. 26, no. 3, pp. 308‐317, 2015.

[15] F. König, C. Sous, A. O. Chaib, and G. Jacobs, “Machine learning based anomaly detection and classification of acoustic emission events for wear monitoring in sliding bearing systems,” Tribol Int, vol. 155, p. 106811, 2021.

[16] M. Jalayer, C. Orsenigo, and C. Vercellis, “Fault detection and diagnosis for rotating machinery: A model based on convolutional LSTM, Fast Fourier and continuous wavelet transforms,” Comput Ind, vol. 125, p. 103378, 2021.

[17] S. Djaballah, K. Meftah, K. Khelil, and M. Sayadi, “Deep transfer learning for bearing fault diagnosis using CWT time–frequency images and convolutional neural networks,” Journal of Failure Analysis and Prevention, vol. 23, no. 3, pp. 1046–1058, 2023.

[18] C. Chen, J. Hu, Q. Meng, and Y. Zhang, “Short-time traffic flow prediction with ARIMA-GARCH model,” in 2011 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2011, pp. 607–612.

[19] Z. Li, S. Jiang, L. Li, and Y. Li, “Building sparse models for traffic flow prediction: An empirical comparison between statistical heuristics and geometric heuristics for Bayesian network approaches,” Transportmetrica B: Transport Dynamics, 2017.

[20] D. Huang, Z. Deng, L. Zhao, and B. Mi, “A short-term traffic flow forecasting method based on Markov chain and grey Verhulst model,” in 2017 6th Data Driven Control and Learning Systems (DDCLS), IEEE, 2017, pp. 606–610.

[21] M. Lippi, M. Bertini, and P. Frasconi, “Collective traffic forecasting,” in Joint European conference on machine learning and knowledge discovery in databases, Springer, 2010, pp. 259–273.

[22] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, “Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions,” Expert Syst Appl, vol. 36, no. 3, pp. 6164–6173, 2009.

[23] H.-F. Yang, T. S. Dillon, and Y.-P. P. Chen, “Optimized structure of the traffic flow forecasting model with a deep learning approach,” IEEE Trans Neural Netw Learn Syst, vol. 28, no. 10, pp. 2371–2381, 2016.

[24] R. L. Abduljabbar, H. Dia, and P.-W. Tsai, “Unidirectional and bidirectional LSTM models for short-term traffic prediction,” J Adv Transp, vol. 2021, pp. 1–16, 2021.

[25] V. Fortuin, G. Rätsch, and S. Mandt, “Multivariate time series imputation with variational autoencoders,” arXiv preprint arXiv:1907.04155, vol. 67, 2019.

[26] X. Cheng, R. Zhang, J. Zhou, and W. Xu, “Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting,” in 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, 2018, pp. 1–8.

[27] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” arXiv preprint arXiv:1707.01926, 2017.

[28] Y. Wu, D. Zhuang, A. Labbe, and L. Sun, “Inductive graph neural networks for spatiotemporal kriging,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 4478–4485.

[29] Y. Li, Z. Li, and L. Li, “Missing traffic data: comparison of imputation methods,” IET Intelligent Transport Systems, vol. 8, no. 1, pp. 51–57, 2014.

[30] X. Chen, Z. He, and L. Sun, “A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation,” Transp Res Part C Emerg Technol, vol. 98, pp. 73–84, 2019.

[31] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, “Brits: Bidirectional recurrent imputation for time series,” Adv Neural Inf Process Syst, vol. 31, 2018.

[32] J. Yoon, J. Jordon, and M. Schaar, “Gain: Missing data imputation using generative adversarial nets,” in International conference on machine learning, PMLR, 2018, pp. 5689–5698.

[33] X. Kong, W. Zhou, G. Shen, W. Zhang, N. Liu, and Y. Yang, “Dynamic graph convolutional recurrent imputation network for spatiotemporal traffic missing data,” Knowl Based Syst, vol. 261, p. 110188, 2023.

[34] Y. Djenouri, A. Belhadi, J. C.-W. Lin, D. Djenouri, and A. Cano, “A survey on urban traffic anomalies detection algorithms,” IEEE Access, vol. 7, pp. 12192–12205, 2019.

[35] Y. Yuan, Y. Zhang, B. Wang, Y. Peng, Y. Hu, and B. Yin, “STGAN: Spatio-temporal generative adversarial network for traffic data imputation,” IEEE Trans Big Data, vol. 9, no. 1, pp. 200–211, 2022.

[36] Y. Liang, Z. Zhao, and L. Sun, “Dynamic spatiotemporal graph convolutional neural networks for traffic data imputation with complex missing patterns,” arXiv preprint arXiv:2109.08357, 2021.

[37] Y. Wölker, C. Beth, M. Renz, and A. Biastoch, “SUSTeR: Sparse Unstructured Spatio Temporal Reconstruction on Traffic Prediction,” in Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, 2023, pp. 1–10.

[38] M. Liu, T. Zhu, J. Ye, Q. Meng, L. Sun, and B. Du, “Spatio-temporal autoencoder for traffic flow prediction,” IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 5, pp. 5516–5526, 2023.

[39] H. Kim, B. S. Lee, W.-Y. Shin, and S. Lim, “Graph anomaly detection with graph neural networks: Current status and challenges,” IEEE Access, 2022.

[40] H. Zhang, S. Zhao, R. Liu, W. Wang, Y. Hong, and R. Hu, “Automatic traffic anomaly detection on the road network with spatial-temporal graph neural network representation learning,” Wirel Commun Mob Comput, vol. 2022, pp. 1–12, 2022.

[41] J. Zhou et al., “Graph neural networks: A review of methods and applications,” AI open, vol. 1, pp. 57–81, 2020.

[42] Z. Cui, R. Ke, Z. Pu, X. Ma, and Y. Wang, “Learning traffic as a graph: A gated graph wavelet recurrent neural network for network-scale traffic prediction,” Transp Res Part C Emerg Technol, vol. 115, p. 102620, 2020.

[43] C. Chen et al., “Gated residual recurrent graph neural networks for traffic prediction,” in Proceedings of the AAAI conference on artificial intelligence, 2019, pp. 485–492.

[44] Q. Wang, H. Jiang, M. Qiu, Y. Liu, and D. Ye, “TGAE: Temporal Graph Autoencoder for Travel Forecasting,” IEEE Transactions on Intelligent Transportation Systems, 2022.

[45] M. Li and Z. Zhu, “Spatial-temporal fusion graph neural networks for traffic flow forecasting,” in Proceedings of the AAAI conference on artificial intelligence, 2021, pp. 4189–4196.

[46] W. Liang et al., “Spatial-temporal aware inductive graph neural network for C-ITS data recovery,” IEEE Transactions on Intelligent Transportation Systems, 2022.

[47] Z. Shi, X. Liang, and J. Wang, “LMC: Fast Training of GNNs via Subgraph Sampling with Provable Convergence,” arXiv preprint arXiv:2302.00924, 2023.

[48] H. Zeng et al., “Decoupling the depth and scope of graph neural networks,” Adv Neural Inf Process Syst, vol. 34, pp. 19665–19679, 2021.

[49] Z. Zhang, P. Cui, and W. Zhu, “Deep learning on graphs: A survey,” IEEE Trans Knowl Data Eng, vol. 34, no. 1, pp. 249–270, 2020.

[50] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, “Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks,” in Proceedings of the 25th

ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 257–266.

[51] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” arXiv preprint arXiv:1710.10903, 2017.

[52] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, “Attributed graph clustering: A deep attentional embedding approach,” arXiv preprint arXiv:1906.06532, 2019.

[53] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 16000–16009.

[54] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature verification using a" siamese" time delay neural network,” Adv Neural Inf Process Syst, vol. 6, 1993.

[55] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in International conference on machine learning, PMLR, 2020, pp. 1597–1607.

[56] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 9729–9738.

[57] H. Hojjati and N. Armanfard, “Self-supervised acoustic anomaly detection via contrastive learning,” in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 3253–3257.

[58] Y. Zhang, Z. Ren, S. Zhou, K. Feng, K. Yu, and Z. Liu, “Supervised contrastive learning-based domain adaptation network for intelligent unsupervised fault diagnosis of rolling bearing,” IEEE/ASME Transactions on Mechatronics, vol. 27, no. 6, pp. 5371–5380, 2022.

[59] E. Hoffer and N. Ailon, “Deep metric learning using triplet network,” in International workshop on similarity-based pattern recognition, Springer, 2015, pp. 84–92.

[60] Y. Huang and J. J. Yang, “Semi-supervised multiscale dual-encoding method for faulty traffic data detection,” Applied Computing and Intelligence, vol. 2, no. 2, pp. 99–114, 2022.

[61] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in ICML deep learning workshop, Lille, 2015, p. 0.

[62] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), IEEE, 2005, pp. 539–546.

[63] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), IEEE, 2006, pp. 1735–1742.

[64] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, “Sampling matters in deep embedding learning,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2840–2848.

[65] C.-Y. Chuang, J. Robinson, Y.-C. Lin, A. Torralba, and S. Jegelka, “Debiased contrastive learning,” Adv Neural Inf Process Syst, vol. 33, pp. 8765–8775, 2020.

[66] A. Vaswani et al., “Attention is all you need,” Adv Neural Inf Process Syst, vol. 30, 2017.

[67] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” arXiv preprint arXiv:2010.11929, 2020.

[68] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in International conference on machine learning, PMLR, 2019, pp. 7354–7363.

[69] H. Ramchoun, Y. Ghanou, M. Ettaouil, and M. A. Janati Idrissi, “Multilayer perceptron: Architecture optimization and training,” 2016.

[70] J. M. Keller, M. R. Gray, and J. A. Givens, “A fuzzy k-nearest neighbor algorithm,” IEEE Trans Syst Man Cybern, no. 4, pp. 580–585, 1985.

[71] G. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. O’Leary, “PyWavelets: A Python package for wavelet analysis,” J Open Source Softw, vol. 4, no. 36, p. 1237, 2019.

[72] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[73] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE.,” Journal of machine learning research, vol. 9, no. 11, 2008.

[74] Y. Huang and J. J. Yang, “Semi-supervised multiscale dual-encoding method for faulty traffic data detection,” arXiv preprint arXiv:2212.13596, 2022.

[75] Y. Huang and J. J. Yang, “Symmetric contrastive learning for robust fault detection in time-series traffic sensor data,” Int J Data Sci Anal, pp. 1–15, 2024.

[76] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” arXiv preprint arXiv:1611.07308, 2016.

[77] M. Ahmed, R. Seraj, and S. M. S. Islam, “The k-means algorithm: A comprehensive survey and performance evaluation,” Electronics (Basel), vol. 9, no. 8, p. 1295, 2020.

[78] D. Müllner, “Modern hierarchical, agglomerative clustering algorithms,” arXiv preprint arXiv:1109.2378, 2011.

[79] M. Ankerst, M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander, “Ordering points to identify the clustering structure,” in Proc. ACM SIGMOD, 2008.

[80] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” arXiv preprint arXiv:1609.02907, 2016.

[81] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[82] D. Wu, Y. Wang, S.-T. Xia, J. Bailey, and X. Ma, “Skip connections matter: On the transferability of adversarial examples generated with resnets,” arXiv preprint arXiv:2002.05990, 2020.

[83] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” Adv Neural Inf Process Syst, vol. 30, 2017.

[84] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, “Graphsaint: Graph sampling based inductive learning method,” arXiv preprint arXiv:1907.04931, 2019.

[85] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput, vol. 9, no. 8, pp. 1735–1780, 1997.

[86] Z. Huang, W. Xu, and K. Yu, “Bidirectional LSTM-CRF models for sequence tagging,” arXiv preprint arXiv:1508.01991, 2015.

[87] J. P. C. Chiu and E. Nichols, “Named entity recognition with bidirectional LSTM-CNNs,” Trans Assoc Comput Linguist, vol. 4, pp. 357–370, 2016.

[88] H. Wackernagel and H. Wackernagel, “Ordinary kriging,” Multivariate geostatistics: an introduction with applications, pp. 79–88, 2003.

[89] P. J. Rousseeuw and C. Croux, “Alternatives to the median absolute deviation,” J Am Stat Assoc, vol. 88, no. 424, pp. 1273–1283, 1993.

[90] L. Chiaramonte, H. Liu, F. Poli, and M. Zhou, “How accurately can Z‐score predict bank failure?,” Financial markets, institutions & instruments, vol. 25, no. 5, pp. 333‐360, 2016.

[91] Y. Huang, H. Zhen, and J. J. Yang, “Cluster-guided denoising graph auto-encoder for enhanced traffic data imputation and fault detection,” Expert Syst Appl, p. 125531, 2024.