Dissertation submitted to the

UNIVERSITY OF MOHAMED BOUDIAF - M'SILA, ALGERIA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

in partial fulfillment of the requirements for the degree of

Master's in Computer Science

Specialization: Computer Science

Option: Networks and Information and Communication Technologies (NICT)

Presented by

Abdelhak Debbi

Entitled

An Empirical Study of GNN-Based Anomaly Detection in Smart Home Environments

Under the supervision of

Dr. Noureddine Amraoui

Jury Members

Dr. Nasereddine Amroune	University of M'sila	President
Dr. Noureddine Amraoui	University of M'sila	Reporter
Dr. Said Amri	University of M'sila	Examiner

June, 2026

الملخص

تقدم هذه الأطروحة دراسةً مقارنةً منهجيةً لمعماريات كشف الشذوذ المعتمدة على الشبكات العصبية البيانية Graph Neural Networks (GNNs) والمطبقة على بيانات السلاسل الزمنية متعددة المتغيرات الخاصة بالمنازل الذكية. تركز الدراسة على معماريتين ممثلتين في: شبكة انحراف الرسم البياني Graph Deviation Networks (GDN) ،وهي نموذج بياني متناثر قائم على التنبؤ، وMTAD-GAT، وهي معمارية هجينة كثيفة تعتمد على آلية الانتباه وتجمع بين هدفي التنبؤ وإعادة البناء. أُجريت التجارب باستخدام مجموعتي بيانات المنازل الذكية BRE وCU، بالإضافة إلى إعداد يتضمن دمج بيانات غير متجانسة، وذلك ضمن إطار تجريبي موحد يعتمد بروتوكولات متسقة للمعالجة المسبقة، والتدريب، وتحديد العتبات، والتقييم.

تُظهر النتائج أن احتساب درجات الشذوذ المعزز بإعادة البناء في نموذج MTAD-GAT يوفر متانة أكبر وقابلية فصل إحصائي أفضل في بيئات المنازل الذكية غير المتجانسة، في حين ينتج نموذج GDN استجابات أكثر حدةً ولكنها أكثر تمركزًا للشذوذ من خلال نمذجة التبعيات المتناثرة. كما يبيّن تحليل حساسية العتبات أن المعماريات الكثيفة المعتمدة على آليات الانتباه تحافظ على مجالات تشغيل أكثر استقرارًا عبر عتبات كشف مختلفة. بالإضافة إلى ذلك، تكشف تحليلات الرسوم البيانية وعلى مستوى الأجهزة عن سلوكيات تفسيرية متميزة بين البنى البيانية المتناثرة ذات الطابع الموضعي ونمذجة التبعيات الموزعة المعتمدة على آليات الانتباه.

الكلمات المفتاحية: الشبكات العصبية للرسوم البيانية، بيئات المنازل الذكية، كشف الشذوذ، السلاسل الزمنية متعددة المتغيرات

Abstract

This thesis presents a systematic comparative study of Graph Neural Network (GNN)-based anomaly detection architectures for smart home multivariate time-series data. The study focuses on two representative architectures: Graph Deviation Network (GDN), a sparse forecasting-based graph model, and MTAD-GAT, a dense attention-based hybrid architecture combining forecasting and reconstruction objectives. Experiments are conducted using the BRE and CU smart home datasets as well as a merged heterogeneous dataset configuration under a unified experimental framework with consistent preprocessing, training, thresholding, and evaluation protocols.

The results show that reconstruction-enhanced anomaly scoring in MTAD-GAT provides greater robustness and statistical separability in heterogeneous smart home environments, while GDN produces sharper but more localized anomaly responses through sparse dependency modeling. Threshold sensitivity analysis further demonstrates that dense attention-based architectures maintain more stable operating regions across varying detection thresh-

olds. In addition, graph and device-level analyses reveal distinct interpretability behaviors between sparse localized graph structures and distributed attention-based dependency modeling.

Keywords: Graph Neural Networks, Smart Home Environments, Anomaly Detection, Multivariate Time series

Dedication

“To my Mother and My grandMother Habiba”

AbdElhak

Acknowledgments

First and foremost, I would like to thank my thesis supervisor, Dr. Noureddine Amraoui, for his valuable guidance, encouragement, patience, and support in completing this work. His advice, motivation, and academic knowledge have contributed in a large way to the success of this thesis. I am grateful to him for the trust he showed in me and for the time he invested in supervising this research.

I would also like to thank all the professors and teachers of the Computer Science department who have taught me throughout my university studies. Their dedication, knowledge, and teaching efforts have equipped me with the necessary skills to complete this work.

Finally, I am eternally grateful to my family for their love, patience, sacrifices, and unwavering support. Their faith in me has always been my strongest pillar of motivation. This accomplishment would not be possible without their encouragement.

Contents

List of Tables

List of Figures

List of Symbols

α	Forecast score weight ($\alpha = 0.5$)
β	Reconstruction score weight ($\beta = 0.5$)
η_0	Initial learning rate
$\eta_{\min}$	Minimum learning rate
$\hat{x}_t^{(i)}$	Predicted value
$\mathbf{A}$	Adjacency matrix
$\mathbf{e}_i$	Node embedding vector
$\mathbf{W}^{(t)}$	Sliding window ending at time t
$\mathbf{X}$	Full multivariate time series
$\mathbf{X}^{\text{train}}$	Training data matrix
$\mathbf{x}_t$	Observation vector at time t
$\mathbf{y}_t$	Target vector at time t
$\mathcal{A}$	Set of anomalous time steps
$\mathcal{E}$	Set of edges
$\mathcal{G}$	Graph of device interactions
$\mathcal{L}$	Training loss function
$\mathcal{L}_{\text{for}}$	Forecasting head loss
$\mathcal{L}_{\text{rec}}$	Reconstruction head loss
$\mathcal{N}$	Set of normal time steps
$\mathcal{N}(i)$	Neighborhood of node i
$\mathcal{V}$	Set of nodes
Delay	Detection delay in number of windows

DR	Detection Rate
$\mathrm{Err}_i(t)$	Prediction error of device i at time t
FPR	False Positive Rate
ω	Smoothing window size
σ	Activation function
τ	Detection threshold
τ_{90}	90th percentile threshold
$\tilde{x}_t^{(i)}$	Reconstructed value
A_{ji}	Edge from node j to node i
B	Batch size
d	Embedding dimension
D_{KL}	Kullback--Leibler divergence
f	Trained model
$h_i^{(l)}$	Node embedding at layer l
k	Number of nearest neighbors
N	Number of device channels (nodes)
p	p-value of a statistical test
$S(t)$	Aggregated system-level anomaly score
$S_s(t)$	Smoothed anomaly score
S_{anomaly}	Actor 2 (anomalous) test scores
$S_{\mathrm{comb}}(t)$	Combined anomaly score (MTAD-GAT)
$S_{\mathrm{for}}(t)$	Forecasting anomaly score (MTAD-GAT)
S_{normal}	Actor 1-T2 (normal) test scores
$S_{\mathrm{rec}}(t)$	Reconstruction anomaly score (MTAD-GAT)
S_{train}	Training score series

S_{val}	Validation score series
T	Total number of time steps
t_{start}	Start time of the anomaly period
T_{train}	Number of training time steps
U	Mann--Whitney U statistic
W	Sliding window size
$W^{(l)}$	Layer weight matrix
$x_t^{(i)}$	Value of device i at time t

Introduction

Smart home environments generate continuous, high-dimensional multivariate time series through a dense network of heterogeneous devices like motion sensors, contact sensors, temperature and humidity probes, smart locks, and occupancy detectors, among others. Each device captures a distinct facet of residential activity, and together they produce a rich digital trace of human behaviour and environmental dynamics. This data is inherently *relational* where the state of a door contact sensor influences the expected reading of an adjacent motion sensor, the activation of a heating actuator changes the trajectory of temperature and humidity over the following minutes. These spatial dependencies, combined with the temporal evolution of daily routines, define the normal operating regime of a smart home system.

The practical value of this data extends well beyond comfort automation. Monitoring the device stream continuously and detecting deviations from the expected pattern can signal security breaches, equipment faults, or—as studied in this thesis—the presence of an occupant whose behaviour is inconsistent with the household's learned norms. Reliable anomaly detection in smart home multivariate time series is therefore both a technically challenging problem and a practically important one.

Problem

Traditional time series anomaly detection approaches often treat device streams independently or rely on implicit correlation modeling [1]. However, in real-world smart home environments, sensor data is inherently interdependent. Device readings exhibit spatial correlations (e.g., interactions between motion and contact sensors) as well as temporal dependencies driven by evolving activity patterns. Furthermore, anomalous events may propagate across multiple devices, making isolated analysis insufficient.

Motivation

Recent advances in Graph Neural Networks (GNNs) have shown strong potential for multivariate time series anomaly detection by explicitly modeling dependencies between devices and sensors. In smart home environments, where occupant behavior generates complex and interconnected activity patterns, graph-based approaches provide a natural framework for capturing relational behavioral anomalies that traditional sequential models may overlook.

Although several GNN-based anomaly detection methods have demonstrated promising results [2], [3], their behavior in smart home IoT environments remains insufficiently ex-

plored. Unlike industrial systems, smart home data is highly heterogeneous, sparse, asynchronous, and strongly influenced by human behavioral variability, making anomaly detection particularly challenging [4].

This motivates the need for a systematic comparative study of GNN-based anomaly detection architectures in smart home environments. This thesis investigates how graph structures, learning objectives, and anomaly scoring strategies influence detection performance, robustness, threshold stability, and interpretability across multiple smart home datasets using a unified experimental framework.

Research Objectives

The primary objective of this thesis is to systematically analyze and evaluate GNN-based anomaly detection models for smart home multivariate time series data, with a particular emphasis on architectural design choices, dataset and temporal parameter sensitivity, and device-level interpretability. This objective is pursued through a comprehensive empirical study designed to reveal how different modelling decisions affect both detection performance and the quality of the explanations the models provide to an operator.

Research Questions

The overarching research question guiding this thesis is: How do architectural and graph design choices affect anomaly detection performance and interpretability in smart home multivariate time series data?

This central question is decomposed into two operational sub-questions:

- Can graph-based models detect behavioral anomalies in smart home data without supervision?
- Which learning paradigm is more effective: forecasting or reconstruction?
- How does graph structure affect performance?
- How do models behave under increasing environmental heterogeneity?
- Which devices contribute most significantly to the anomaly score, and is this contribution consistent across models?

Thesis Contributions

This thesis presents the following contributions:

- **Controlled comparative evaluation on smart home IoT data.** We conduct the first systematic side-by-side comparison of GDN and MTAD-GAT on the IoT Garage BRE and CU datasets under a unified experimental protocol that controls for window size ($W = 10$), downsampling frequency (1 s vs. 3 s), random seed, and threshold strategy.
- **Quantitative anomaly score analysis.** Beyond standard F1, precision, and recall, we characterize the separation between normal and anomalous score distributions using KL divergence, and detection and false positive rates. This threshold-agnostic analysis provides a richer view of model behavior than binary classification metrics alone.
- **Multi-dataset and multi-temporal sensitivity study.** We quantify how detection quality changes across BRE, CU, and merged (BRE+CU) datasets, and across 1 s and 3 s downsampling frequencies, providing evidence-based guidance for deployment decisions in diverse residential environments.

Thesis Organization

The remainder of this thesis is structured as follows:

- **Chapter 1 — Related Work** reviews the literature on anomaly detection in multivariate time series, covering both classical statistical approaches and modern deep learning methods, with a focus on graph neural network (GNN)-based models. It also discusses graph construction strategies and interpretability techniques, and identifies the research gaps that motivate this work.
- **Chapter 2 — Problem Formulation and Experimental Framework** formalises the anomaly detection problem in multivariate time series, defines the graph-based modelling framework, and presents the unified experimental protocol used throughout the study. This includes sliding window construction, training and evaluation procedures, hyperparameter settings, and the complete anomaly scoring pipeline.
- **Chapter 3 — Dataset and Preprocessing** describes the IoT Garage BRE and CU smart home datasets, provides exploratory data analysis, and details the preprocessing pipeline, from raw event logs to normalised sliding window representations suitable for model training and evaluation.
- **Chapter 4 — Systematic Comparative Study** presents a comparative evaluation of GDN and MTAD-GAT. It analyzes training behavior, anomaly score distributions, temporal dynamics, threshold sensitivity, and robustness across dataset configurations. The chapter also studies the impact of different anomaly scoring strategies and evaluates model stability under varying operational conditions.

Chapter 1: Related Work

1.1 Introduction

Anomaly detection in multivariate time series (MTS) is a fundamental problem in many real-world applications, particularly in smart home environments where heterogeneous devices continuously monitor user activities. The objective is to identify abnormal patterns that deviate from expected system behavior. Unlike univariate anomaly detection, where anomalies manifest as extreme values in individual signals, smart home data is characterized by complex interdependencies among multiple devices. Consequently, anomalies often arise from irregular interactions between devices rather than isolated deviations, making the detection task significantly more challenging.

Traditional statistical methods have been extensively applied to this problem due to their simplicity, interpretability, and low computational cost. However, these approaches are inherently limited in their ability to capture nonlinear relationships and high-dimensional dependencies present in multivariate data. As a result, they often fail to detect anomalies that stem from subtle changes in the correlation structure between devices.

Recent advances in deep learning have substantially improved the modeling of complex temporal and multivariate patterns. In particular, Graph Neural Networks (GNNs) have emerged as a promising paradigm for MTS anomaly detection, as they explicitly represent devices as nodes and their interactions as edges within a graph structure. This formulation enables the model to capture both spatial dependencies (inter-device relationships) and temporal dynamics, capabilities that are especially important in smart home environments where anomalies frequently result from changes in user behavior affecting multiple interconnected devices.

This chapter provides a short review of existing approaches to MTS anomaly detection, discusses their limitations, and motivates the adoption of GNN-based models for capturing complex and dynamic device interactions in smart home settings. Section 1.2 covers the taxonomy of anomaly types and traces the evolution of detection methods. Section 1.3 reviews GNN-based detection approaches. Section 1.4 analyzes graph construction strategies. Section 1.5 consolidates the survey into a statement of the open gaps that motivate this thesis.

1.2 Anomaly Detection in Multivariate Time Series

Anomaly detection in MTS is the task of identifying observations whose behavior deviates from patterns established during normal operation. Wang *et al.* [1] distinguish three broad

categories of anomaly, which differ in how and where the deviation manifests.

Point-wise anomalies are isolated observations that sharply deviate from expectation at a single time-step in one or more channels. A temperature sensor suddenly reading an extreme value, or a contact device activating at an unusual hour, are typical examples. The deviation is contained in a single instant and is detectable by comparing each observation against its historical range.

Pattern-wise anomalies are sub-sequences that disrupt expected periodicities or trends over a contiguous interval even though no individual observation may be extreme in isolation. A device that normally activates once in the morning but activates repeatedly throughout the day exhibits a pattern-wise anomaly cause of the repeated activation is unusual not because any single event is out of range but because the overall temporal sequence deviates from the normal pattern.

Inter-metric anomalies manifest as a change in the correlation structure among variables, while each variable's marginal values remain within individually normal ranges. For example, two devices that normally activate together may stop co-activating when an unfamiliar person moves through the home differently. The individual readings of each device remain plausible, but their joint behavior reveals the anomaly.

The occupancy-change scenario studied in this thesis falls primarily into the inter-metric category rather than the pattern-wise one. While a pattern-wise anomaly would require an observable disruption in the overall temporal rhythm of device usage, the arrival of an external person does not necessarily break the daily cycle. Instead it alters how devices co-activate, which rooms are used, in what order, and in combination with which other devices. Individual readings remain within normal ranges, but the joint distribution shifts. A detector must therefore model cross-device dependencies explicitly rather than monitoring each device in isolation, which rules out large classes of classical methods and motivates graph structured architectures.

Existing anomaly detection approaches for multivariate time series data can generally be categorized into three major families: statistical methods, deep learning methods, and graph neural network (GNN)-based approaches. Each family differs in its ability to model temporal dynamics, nonlinear behavior, and inter-device dependencies. A comparative summary of these families is presented in Table 1.1.

1.2.1 Statistical Methods

The earliest principled approaches to MTS anomaly detection were grounded in classical time series statistics. Autoregressive Integrated Moving Average (ARIMA) and its seasonal extension (SARIMA) model each channel independently as a linear function of its own past values and a stochastic noise term large residuals are flagged as anomalies [1]. This univariate framing is computationally tractable and interpretable but it is inherently blind to cross

Table 1.1: Summary of Anomaly Detection Method Families

Family	**Representative Methods**	**Core Idea**	**Strengths**	**Limitations**
Statistical Methods	PCA, ARIMA, Isolation Forest, One-Class SVM	Detect deviations using statistical distributions and handcrafted patterns	Simple, interpretable, low computational cost	Limited modeling of nonlinear and complex temporal dependencies
Deep Learning Methods	LSTM, Autoencoders, VAE, GAN, Transformers	Learn temporal representations through forecasting or reconstruction	Capture nonlinear temporal dynamics and sequential behavior	Limited explicit modeling of inter-device relationships
GNN-Based Methods	GDN, MTAD-GAT, GLUE, SDGL, TopoGDN	Model sensor interactions using graph-structured dependency learning	Capture relational behavioral anomalies and device interactions	Sensitive to graph construction quality and computational complexity

channel correlations that characterize inter metric anomalies.

Vector Autoregression (VAR) [5] generalizes ARIMA to the multivariate case by representing each variable as a linear combination of all variables' past values. VAR therefore captures pairwise linear dependencies and can, in principle, detect anomalies in the correlation structure. In practice, however, the number of parameters grows quadratically with the number of channels, rendering VAR impractical for device networks with more than a few dozen variables. Furthermore, both ARIMA and VAR assume weak stationarity, a requirement that is routinely violated by smart home device streams exhibiting strong daily and weekly periodicities and structural shifts whenever the occupancy pattern changes.

Density based methods assign anomaly scores based on the local density of observations in the feature space. Local Outlier Factor (LOF) [6] computes a score by comparing the density of a point's neighborhood with the density of its neighbors' neighborhoods points in anomalously low density regions receive high scores. One class Support Vector Machines (OCSVM) [7] learn a hypersphere in a kernel induced feature space enclosing the majority of normal data test points outside the sphere are declared anomalous. Both methods are effective for static low dimensional data but treat each observation as an exchangeable vector and discard temporal ordering entirely.

Isolation Forest [8] takes a different approach, positing that anomalous observations are those that can be isolated by fewer random axis-aligned binary splits. It is computationally efficient, requires no distributional assumption, and scales to moderate dimensionality. Nevertheless, like LOF and OCSVM, it treats each multivariate observation as an unordered

feature vector, ignoring the temporal evolution of device states and the lagged correlations between channels that encode behavioral routines.

In summary while statistical methods provide interpretable baselines and continue to be used in low data deployments they share two critical limitations in the smart home context the inability to capture non linear cross channel dependencies and the absence of an explicit temporal model that tracks how the joint distribution of device readings evolves over time.

1.2.2 Deep Learning Methods

Deep learning has significantly advanced multivariate time series (MTS) anomaly detection by enabling models to learn hierarchical and nonlinear representations directly from raw sequential data without manual feature engineering. Unlike traditional statistical or machine learning approaches, deep neural architectures can jointly capture temporal dependencies, inter-variable correlations, and complex distributional patterns in high-dimensional sensor streams. According to Wang *et al.* [1], deep learning methods for MTS anomaly detection can broadly be categorized into two main paradigms: Forecasting-based and Reconstruction-based methods.

Forecasting-based approaches. Forecasting-based methods learn the temporal evolution of a system by predicting future observations from a historical sliding window. The difference between the predicted and actual values constitutes the anomaly score. These methods are particularly effective when anomalous events produce deviations from learned temporal dynamics.

CNN-based forecasting models. Convolutional Neural Networks (CNNs) apply convolutional filters over temporal windows to capture local temporal patterns and short-term dependencies. CNN-based forecasting models are computationally efficient due to parallel convolution operations and are well suited for high-frequency data streams. Their localized receptive fields enable robust extraction of repetitive motifs and local trends. However, CNNs struggle to model long-range temporal dependencies unless very deep architectures or dilated convolutions are employed. In addition, CNNs do not explicitly capture relationships among variables in multivariate systems.

RNN/LSTM-based forecasting models. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks [9], are widely used for forecasting sequential data due to their ability to preserve temporal memory across long sequences. LSTM-NDT proposed by Hundman *et al.* [10] demonstrated the effectiveness of forecasting-based anomaly detection for spacecraft telemetry. The main advantage of LSTM forecasting models is their ability to capture long-term sequential dependencies and provide interpretable

channel-wise prediction errors. Nevertheless, sequential processing limits parallelization, increases training cost, and weakens performance on very long sequences due to memory bottlenecks.

GNN-based forecasting models. Graph Neural Networks (GNNs) extend forecasting models by explicitly learning inter-sensor dependencies through graph message passing. Instead of treating variables independently, GNNs represent sensors as nodes connected through learned or predefined edges. MTGNN proposed by Wu *et al.* [11] dynamically learns graph structures for multivariate forecasting, while AGCRN [12] adapts graph convolutional recurrent networks to evolving correlations. More recent methods such as MTAD-GAT [3], GDN [2], and TopoGDN [13] further improve anomaly detection by integrating graph attention mechanisms, adaptive topology learning, and structural graph regularization.

The major advantage of GNN-based forecasting approaches is their ability to explicitly model spatial and functional dependencies among devices, making them highly suitable for IoT environments and sensor networks. They can detect anomalies that only emerge through abnormal inter-variable relationships rather than individual channel deviations. However, GNN models require graph construction or graph learning mechanisms, which increases architectural complexity and computational cost. Their performance is also sensitive to noisy or incorrectly learned graph structures.

Transformer-based forecasting models. Transformer architectures employ self-attention mechanisms to model long-range temporal dependencies without recurrent processing. Anomaly Transformer [14] introduces an association discrepancy metric that measures the divergence between local and global attention distributions, while TranAD [15] uses dual Transformer encoders with adversarial optimization to enhance anomaly sensitivity.

Transformers provide excellent scalability and parallelization while capturing global temporal interactions more effectively than RNN-based methods. They are particularly effective for long sequences and complex temporal dynamics. However, self-attention computation scales quadratically with sequence length, leading to high memory consumption. Furthermore, standard Transformer architectures treat variables as a flat token set and do not explicitly model structured inter-device relationships unless combined with graph-based mechanisms.

Reconstruction-based approaches. Reconstruction-based methods learn a compact latent representation of normal behavior and attempt to reconstruct the original multivariate input from this representation. During inference, anomalous samples produce larger reconstruction errors because they deviate from the learned normal data manifold.

Autoencoder-based models. Autoencoders (AE) encode multivariate windows into a compressed latent space and reconstruct the original input signal. Malhotra *et al.* [16] proposed an LSTM Encoder Decoder architecture for anomaly detection in multi-sensor systems. Autoencoder models are relatively simple, computationally efficient, and capable of learning compact representations of normal system behavior. They are effective for denoising and dimensionality reduction. However, conventional autoencoders may reconstruct abnormal patterns too accurately when model capacity is large, reducing anomaly sensitivity.

Variational Autoencoder (VAE)-based models. Variational Autoencoders (VAE) [17] extend traditional autoencoders by learning a probabilistic latent distribution rather than deterministic embeddings. OmniAnomaly [18] combines stochastic recurrent neural networks with variational inference and normalizing flows to jointly capture temporal and probabilistic properties.

The probabilistic formulation of VAEs improves robustness to noise and uncertainty while enabling likelihood-based anomaly scoring. Additionally, VAEs often generalize better than deterministic autoencoders. Nevertheless, the imposed latent distribution may oversmooth learned representations, potentially reducing reconstruction fidelity and weakening sensitivity to subtle anomalies.

GAN-based models. Generative Adversarial Networks (GANs) formulate anomaly detection as an adversarial game between a generator and a discriminator. MAD-GAN proposed by Li *et al.* [19] generates realistic multivariate sequences while the discriminator evaluates whether the input belongs to the normal data distribution.

GAN-based models can learn highly complex data distributions and produce realistic synthetic temporal patterns. Their discriminator networks are often highly sensitive to distributional irregularities. However, GAN training is notoriously unstable due to adversarial optimization, requiring careful hyperparameter tuning and balanced generator-discriminator convergence.

Transformer-based reconstruction models. Transformers can also be employed in reconstruction-based settings by learning contextual embeddings and reconstructing temporal sequences through attention mechanisms. TranAD [15] combines Transformer reconstruction with adversarial objectives to amplify errors at anomalous timestamps.

Transformer-based reconstruction models effectively capture global contextual dependencies and complex temporal interactions while maintaining parallel computation efficiency. However, they remain computationally expensive for long sequences and may fail to exploit explicit graph structure among devices unless integrated with graph learning frameworks.

A comprehensive empirical evaluation by Garg *et al.* [4] demonstrated that no single deep learning architecture consistently outperforms all others across different datasets and

anomaly categories. Model effectiveness strongly depends on dataset properties including dimensionality, temporal resolution, anomaly duration, inter-variable dependency strength, and anomaly frequency. This observation directly motivates the comparative multi-model evaluation conducted in this thesis.

As shown in Table 1.1, statistical approaches are computationally efficient and interpretable but often struggle to capture complex nonlinear behavioral patterns. Deep learning methods improve temporal representation learning; however, many architectures still model sensor streams independently. In contrast, GNN-based methods explicitly model relational dependencies between devices, making them particularly suitable for smart home environments where anomalies often emerge from abnormal interaction patterns.

1.3 GNN-Based Approaches for MTS Anomaly Detection

The inadequacy of both channel-independent models and flat attention mechanisms for capturing the relational structure of device networks has motivated a class of architectures that couple GNNs with temporal modules. GNNs represent each device as a node in a graph and model inter-device dependencies as weighted edges, enabling information propagation across the network according to the learned topology [20].

Table 1.2 summarizes the main characteristics of representative GNN-based architectures which will be discussed in the following.

Table 1.2: Comparison of Representative GNN-Based Anomaly Detection Methods

Method	**Graph Type**	**Learning Objective**	**Main Characteristic**
GDN	Sparse learned graph	Forecasting	Localized dependency modeling with sparse graph connections
MTAD-GAT	Dense attention graph	Forecasting + Reconstruction	Distributed anomaly modeling using hybrid learning objectives
GLUE	Adaptive graph learning	Forecasting	Dynamic dependency adaptation between sensors
SDGL	Sparse dynamic graph	Hybrid learning	Structure-aware temporal graph learning
TopoGDN	Topology-enriched graph	Forecasting	Integrates structural graph constraints into dependency learning

Graph-based forecasting architectures. Forecasting-oriented GNN architectures model multivariate time series by predicting future sensor observations from historical graph-aware representations. In these methods, devices are represented as graph nodes while inter-device dependencies are encoded through learned or adaptive graph structures. The anomaly score is typically derived from the deviation between predicted and observed future values. Compared with channel-independent forecasting models, graph-based forecasting approaches explicitly capture relational dependencies among sensors, allowing them to detect anomalies arising from abnormal interaction patterns rather than isolated signal deviations.

Graph Deviation Network (GDN) [2] represents one of the most influential forecasting-based GNN anomaly detection frameworks. GDN learns a sparse adaptive graph directly from sensor embeddings and models normal behavior as stable graph propagation dynamics. By learning hidden inter-variable dependencies automatically, GDN avoids the need for manually predefined topologies and achieves strong performance in complex IoT environments. However, its learned graph structure remains relatively static during inference, which may limit its ability to capture rapidly evolving device interactions.

Dynamic graph learning methods further extend forecasting-based GNNs by allowing graph connectivity to evolve over time. GLUE [21] introduces a dynamic graph learning framework in which adjacency relationships are continuously updated according to temporal context, enabling the model to adapt to non-stationary environments where sensor dependencies vary across operational conditions. Similarly, SDGL [22] decomposes graph dependencies into static long-term correlations and dynamic short-term interactions fused through adaptive gating mechanisms. These approaches improve the modeling of evolving sensor relationships and temporal dependency shifts commonly observed in smart homes and industrial systems. Nevertheless, dynamically updating graph structures substantially increases computational complexity and training instability compared with static graph learning methods.

TopoGDN [13] further improves forecasting-based graph learning by integrating topological analysis into the graph construction process. Persistent homology features are used to regularize graph connectivity and preserve meaningful structural patterns while suppressing noisy or spurious edges. This improves the robustness and stability of learned graph representations, particularly in high-dimensional multivariate systems with complex dependency structures. The primary limitation of TopoGDN is the additional computational overhead introduced by topological feature extraction and persistent homology analysis.

Hybrid forecasting-reconstruction graph architectures. Hybrid graph-based architectures combine forecasting and reconstruction objectives within a unified anomaly detection framework. The motivation behind this design is that forecasting errors capture temporal deviations in future system evolution, whereas reconstruction errors measure deviations from

the learned manifold of normal multivariate behavior. Combining both objectives generally improves robustness against diverse anomaly types and enhances detection sensitivity in complex environments.

MTAD-GAT [3] represents a prominent hybrid GNN architecture that integrates graph attention mechanisms with both forecasting and reconstruction objectives. The model employs graph attention networks to dynamically estimate the importance of inter-device relationships while simultaneously predicting future sensor values and reconstructing historical observations. This dual optimization strategy improves the detection of both temporal irregularities and abnormal dependency structures among devices. Furthermore, the attention mechanism enhances interpretability by identifying the most influential neighboring sensors contributing to an anomaly decision. However, dense attention computation introduces significant computational cost for large-scale sensor systems and may reduce scalability in highly connected graphs.

Compared with purely forecasting-oriented GNNs such as GDN or SDGL, hybrid architectures provide a more comprehensive representation of normal system behavior by jointly modeling temporal evolution and latent relational structure. Nevertheless, optimizing multiple objectives simultaneously increases architectural complexity, hyperparameter sensitivity, and training time. In practice, the performance of hybrid graph models strongly depends on balancing the forecasting and reconstruction losses while maintaining stable graph learning dynamics.

Overall, GNN-based anomaly detection approaches provide substantial advantages over traditional sequence models by explicitly incorporating inter-device relationships into temporal learning. Their ability to capture graph propagation patterns, relational dependencies, and dynamic topology evolution makes them particularly effective for IoT, cyber-physical systems, and smart home environments. However, these advantages come at the cost of increased computational complexity, graph construction sensitivity, and scalability challenges in high-dimensional multivariate settings.

1.4 Graph Construction Strategies in Literature

The choice of how to construct the adjacency matrix $\mathbf{A}$ is among the most consequential design decisions in any GNN-based time series model. The literature spans a spectrum from fully pre-defined to fully dynamic strategies.

Pre-defined graphs. Early spatio temporal GNNs required domain knowledge to fix the graph before training road sensors connected if on the same road segment IoT devices connected if in the same room. Such graphs provide a strong inductive bias but are domain specific often incomplete and unavailable for commodity smart home deployments where

the physical arrangement of devices is not formally documented.

Correlation-based graphs. A natural data-driven alternative computes pairwise Pearson or cosine similarities between channel time series on the training set and thresholds the result to produce a binary adjacency matrix. This approach is simple and interpretable, but it conflates direct causal dependencies with correlations induced by common drivers, and cannot be updated during training.

Learned sparse graphs. Several methods derive the graph from trainable node embeddings the adjacency matrix is defined by embedding similarities with a top-k sparsification [2]. This makes the graph end-to-end differentiable and optimized for the detection objective. A key limitation is that the learned graph is fixed once training ends and cannot adapt to distribution shifts that occur at test time such as the arrival of a new occupant.

Learned dense graphs. Rather than enforcing sparsity through thresholding or top-k neighbor selection, some methods learn a dense weighted adjacency matrix directly during training. In these approaches, every node can potentially interact with every other node, while the model learns continuous edge strengths optimized for the forecasting or anomaly-detection objective. The graph is therefore fully differentiable and capable of capturing weak but globally distributed dependencies that sparse graphs may miss [23].

Dynamic graphs. SDGL [22] and Tian *et al.* [24] construct time-varying adjacency matrices that adapt to changing input statistics at each inference step. Dynamic graphs are in principle better suited to scenarios where the interaction structure shifts over time, at the cost of substantially higher computational complexity.

Topologically enriched graphs. TopoGDN [13] augments a learned sparse graph with persistent homology features derived from multi-scale graph filtration, adding a global structural perspective absent from purely local, attention-based aggregation.

The graph construction strategy interacts non trivially with the anomaly scoring mechanism. A static graph trained on one occupant's behavioral patterns will produce systematically elevated prediction errors when a new occupant's correlation structure departs from the learned graph which is precisely the anomaly signal this thesis attempts to detect.

1.5 Identified Gaps and Research Motivation

Based on our short review, we can identify four interconnected gaps in the literature that directly motivate the contributions of this thesis.

Lack of controlled comparisons on smart home data. The overwhelming majority of GNN based anomaly detection evaluations rely on industrial control system benchmarks SWaT WADI SMD that differ structurally from smart home IoT data. Industrial datasets feature high uniform sampling rates homogeneous device types short inter anomaly intervals and anomalies from deliberate physical attacks. Smart home datasets by contrast exhibit irregular sampling rates a small number of heterogeneous device types long inter anomaly intervals and anomalies driven by occupancy changes that unfold gradually. Garg *et al.* [4] confirmed empirically that method rankings are highly dataset dependent implying that conclusions drawn from industrial benchmarks cannot be extrapolated to the smart home setting. Yet no published work provides a direct controlled comparison of competing GNN based detectors on smart home data under a unified experimental protocol that controls window size sampling frequency random seed and threshold strategy.

Limited sensitivity analysis. The effect of temporal hyperparameters such as window size and sampling frequency on detection performance has not been systematically studied for GNN-based detectors on smart home data. Whether a higher sampling frequency better preserves behavioral context, or whether a wider window captures more informative temporal patterns which open an empirical questions whose answers have direct practical implications for deployment. Similarly, whether a model trained on one installation generalizes to another, or whether merging data from multiple environments improves or degrades detection quality, remains uninvestigated.

Inadequate evaluation metrics. Existing evaluations restrict reported metrics to precision, recall, and F1-score derived from threshold-based binary decisions. These metrics do not characterize the underlying score distributions or the degree to which normal and anomalous populations are separated. Threshold-agnostic measures such as KL divergence, Cohen's d effect size, and the Mann-Whitney U statistic provide a more complete picture of detection quality and have not been applied to GNN-based detectors on smart home data.

Insufficient interpretability analysis. GNN-based detectors offer natural interpretability through their learned graph structure and per-device attention weights, yet systematic analyses of which devices drive anomaly detection, how the learned dependency graph relates to physical device topology, and how attention patterns evolve during an anomaly episode remain rare in the literature.

1.6 Summary

This chapter has surveyed the landscape of MTS anomaly detection methods, from classical statistical approaches to modern GNN-based architectures. Statistical methods, while interpretable, fail to capture temporal dynamics and complex inter-device dependencies. Deep learning methods improve performance by learning richer representations through forecasting, reconstruction, or attention mechanisms, yet many still lack explicit modeling of device relationships. GNNs address this limitation by representing devices as interconnected nodes and learning their dependencies from data, making them well-suited for inter-metric anomaly detection in smart home environments.

The graph construction strategy plays a crucial role in performance, with learned and dynamic graphs offering greater flexibility in changing environments. Despite these advances, a persistent lack of controlled evaluations on smart home datasets, insufficient sensitivity analysis, inadequate evaluation metrics, and limited interpretability remain open gaps.

In the following chapter, we will formalize the anomaly detection problem addressed in this thesis and establishes the experimental framework used to evaluate and compare the methods reviewed here.

Chapter 2: Problem Formulation and Experimental Framework

2.1 Introduction

This chapter presents our systematic experimental framework established for unsupervised anomaly detection in smart home multivariate time series, focusing on graph-based modeling of device interactions and self-supervised learning from normal occupancy behavior. The goal is to define a unified formulation that is independent of any specific architecture, while ensuring consistency in training, evaluation, and scoring across all experiments.

The problem is framed around learning normal behavioral patterns from heterogeneous IoT device streams and identifying deviations that emerge when these patterns are disrupted. In smart home environments, anomalies are rarely expressed as extreme values in individual devices. Instead, they typically appear as disruptions in the temporal coordination and statistical dependencies between multiple devices, making the problem inherently multivariate and relational.

A unified experimental design is adopted to ensure fair comparison across different models. This includes a consistent representation of the data, identical training and evaluation splits, and standardized anomaly scoring procedures. The framework is structured to isolate the impact of architectural differences while controlling for external factors such as preprocessing choices, optimization settings, and evaluation methodology.

The formulation is organized around several core components:

- **Multivariate time series representation:** Device readings are modeled as synchronized vectors capturing the state of all devices at each time step, forming a structured temporal sequence.
- **Self-supervised learning objective:** Models are trained exclusively on normal occupancy data using prediction-based or reconstruction-based objectives without access to anomaly labels.
- **Graph-based dependency modeling:** Inter-device relationships are represented through learned graph structures that encode functional and behavioral dependencies rather than relying on predefined physical layouts.
- **Anomaly scoring mechanism:** Prediction errors are transformed into normalized per-device signals and aggregated into a single system-level anomaly score used for detection.

- **Evaluation protocol:** Performance is assessed using both point-wise and distribution-based metrics to capture detection accuracy, robustness, and statistical separability between normal and anomalous behavior.

The framework is applied consistently across all datasets and experimental configurations, ensuring that differences in results can be attributed to model design choices rather than variations in data handling or evaluation methodology.

2.2 Problem Overview

The central problem addressed in this thesis is *unsupervised behavioral anomaly detection* in smart home multivariate time series. A smart home equipped with N heterogeneous IoT devices continuously generates a stream of readings that encodes the occupant's daily routine. Under single occupancy conditions these readings form a statistically regular correlated time series whose joint distribution reflects the behavioral patterns of a single resident referred to throughout as *Actor 1*. The anomaly of interest is the arrival of a second occupant (*Actor 2*) whose routine departs from Actor 1's devices are activated in unfamiliar combinations at atypical times relative to one another and in sequences that deviate from the learned joint distribution of device states.

Formally let $\mathbf{X}^{\text{train}} \in \mathbb{R}^{T_{\text{train}} \times N}$ denote the training data collected exclusively during Actor 1's normal occupancy. A model f is trained on $\mathbf{X}^{\text{train}}$ to minimize a self supervised prediction objective next step forecasting signal reconstruction or both without access to any anomaly labels. At inference time f produces a scalar anomaly score $s(t) \in \mathbb{R}$ for each time step t in the test set. A time step is declared anomalous if $s(t)$ exceeds a threshold τ derived from the validation split. The framework is unsupervised actor period labels Actor 1 or Actor 2 are used solely for evaluation and are never exposed to the model during training.

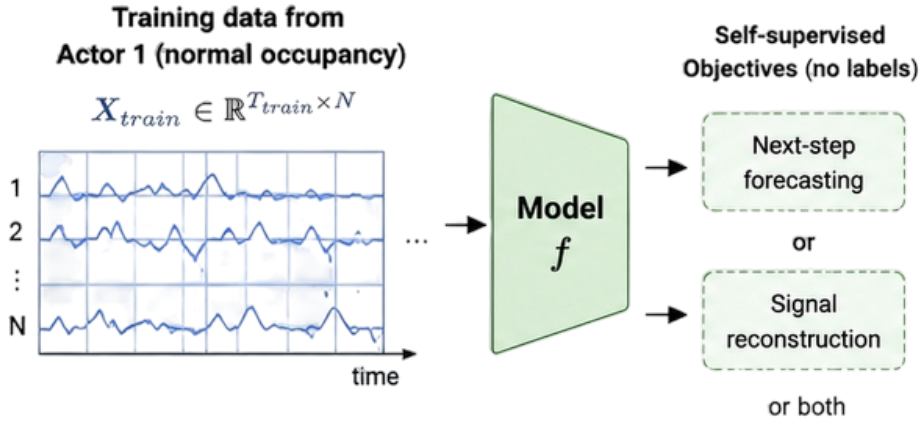

Figure 2.1: GNN Input/Output

2.3 Data Representation

Let N denote the number of device channels under study. At each discrete time-step t, the state of the smart home is represented by the observation vector:

$$\mathbf{x}_t = \left(x_t^{(1)},\, x_t^{(2)},\, \ldots,\, x_t^{(N)}\right)^\top \in \mathbb{R}^N, \tag{2.1}$$

where $x_t^{(i)}$ denotes the reading of device i at time t. The complete time series over a horizon of T discrete steps is:

$$\mathbf{X} = \left[\mathbf{x}_1,\ \mathbf{x}_2,\ \ldots,\ \mathbf{x}_T\right]^\top \in \mathbb{R}^{T\times N}. \tag{2.2}$$

To provide the model with temporal context for prediction, a causal sliding window of length W is applied to the time series. For each valid time index t, the input window and its prediction target are:

$$\mathbf{W}^{(t)} = \left[\mathbf{x}_{t-W},\ \mathbf{x}_{t-W+1},\ \ldots,\ \mathbf{x}_{t-1}\right]^\top \in \mathbb{R}^{W\times N}, \qquad \mathbf{y}_t = \mathbf{x}_t \in \mathbb{R}^N. \tag{2.3}$$

This construction defines a self-supervised learning problem: the model must predict the next device state from a causal window of past observations, without any anomaly labels. The window size W is a tunable hyperparameter that balances temporal context against computational cost.

2.4 Device Graph Modeling

The N devices are modeled as nodes in a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set encoding inter-device dependencies, and $\mathbf{A} \in \{0,1\}^{N\times N}$ is the binary adjacency matrix with $A_{ji} = 1$ indicating that device j contributes to the prediction of device i.

Rather than pre-specifying the graph from physical knowledge of the home layout, the adjacency matrix is *learned from data* through differentiable mechanisms. This is motivated by the observation that the most predictively relevant inter-device relationships are not always obvious from the physical arrangement of devices, and may correspond to behavioral co-occurrence patterns that can only be discovered from training data.

A GNN layer updates the embedding of node i at depth l by aggregating information from its neighborhood $\mathcal{N}(i)$ [20]:

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \alpha_{ij}\, W^{(l)} h_j^{(l)}\right), \tag{2.4}$$

where α_{ij} is the learned weight on the edge $(j \to i)$, $W^{(l)} \in \mathbb{R}^{d \times d}$ is a shared trainable weight matrix at layer l, d is the embedding dimension, and σ is a non-linear activation function.

Once trained, the graph is fixed and does not adapt at inference time. Any systematic shift in inter-device interaction patterns introduced by a new occupant must therefore manifest as elevated prediction error, which is captured by the anomaly scoring pipeline described in Section 2.5. The specific graph construction strategies adopted by each model under study are described in Chapter 4.

2.5 Anomaly Scoring Formulation

The anomaly score $S(t) \in \mathbb{R}$ is derived from the model's per-device prediction errors through a common four-stage pipeline [2].

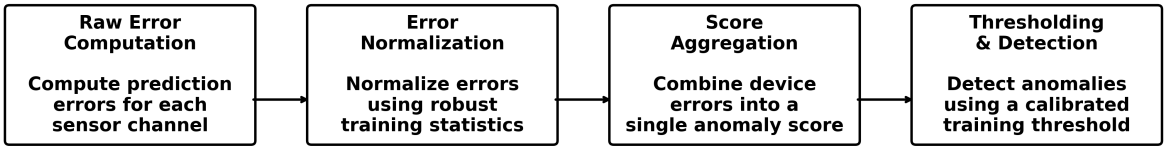

Figure 2.2: Anomaly scoring stages

Stage 1 — Raw error computation. For each device i and time-step t, the raw prediction error is the absolute deviation between the model's output and the observed value:

$$e_i(t) = \left| x_t^{(i)} - \hat{x}_t^{(i)} \right|, \tag{2.5}$$

where $\hat{x}_t^{(i)}$ is the model's prediction for device i at time t. Depending on the model architecture, this prediction may combine a direct next-step forecast, a reconstruction from a variational latent space, or a weighted sum of both. Raw errors are computed over all evaluation splits and stored separately from the scoring pipeline to support independent reconfiguration.

Stage 2 — Error normalization. Raw prediction errors vary in magnitude across device channels due to differences in physical scale. To produce a common, scale-invariant signal, each channel's error is normalized using robust statistics estimated *exclusively* on the training split:

$$a_i(t) = \frac{e_i(t) - \tilde{\mu}_i}{\tilde{\sigma}_i + \epsilon}, \qquad \epsilon = 10^{-5}, \tag{2.6}$$

where $\tilde{\mu}_i$ is the training-split median and $\tilde{\sigma}_i$ is the training-split inter-quartile range (IQR) of device i's error series. The median and IQR are used in place of the mean and standard deviation to provide robustness to occasional large errors arising from device transients or

non-stationarities in the training data. The same statistics are applied without modification to validation and test splits, ensuring no information leakage from anomalous periods.

Stage 3 — Score aggregation. The per-device normalized errors are reduced to a scalar system-level anomaly score by the maximum operator:

$$S(t) = \max_{i \in \{1,\ldots,N\}} a_i(t). \tag{2.7}$$

The maximum aggregation is preferred over averaging because occupancy anomalies typically affect a small, localized subset of devices simultaneously; averaging would dilute the anomaly signal from the most informative channels.

Stage 4 — Thresholding and detection. A time-step t is declared anomalous if $S(t)$ exceeds a threshold τ calibrated from the training split. The primary threshold uses the 90th percentile of the training scores:

$$\tau_{90} = \text{percentile}\big(S_{\text{train}},\ 90\big). \tag{2.8}$$

The threshold is derived solely from training data, ensuring no information leakage from the anomalous or normal test periods. The sensitivity of detection results to the choice of threshold is evaluated empirically in Chapter 4.

Optional smoothing. A simple moving average (SMA) of window ω is optionally applied to suppress high-frequency noise from brief device transients before thresholding:

$$S_s(t) = \frac{1}{\omega} \sum_{k=0}^{\omega-1} S(t-k). \tag{2.9}$$

The smoothing window represents a trade-off between noise suppression and temporal detection precision. Its empirical effect on performance is analyzed in Chapter 4.

2.6 Training Procedure

All models are trained under a fully unified procedure so that observed performance differences are attributable to architectural choices rather than to training configuration.

2.6.1 Hyperparameter Control

All hyperparameters that are not intrinsic to a specific architecture are held constant across all experiments, datasets, and configurations. These shared hyperparameters are listed in

Table 2.1. Architecture-specific hyperparameters are fixed to the values reported in the respective original publications and are not tuned on the datasets used in this thesis, preventing any dataset-specific advantage for any model.

Table 2.1: Shared training hyperparameters fixed across all experiments and configurations.

Hyperparameter	**Value**
Window size W	10
Batch size B	32
Optimizer	Adam
Initial learning rate η_0	1×10^{-4}
Maximum training epochs	100
Early stopping patience	10 epochs
LR scheduler reduction factor	0.5
LR scheduler patience	3 epochs
Minimum learning rate $\eta_{\min}$	1×10^{-6}
Validation ratio	10%

The window size $W = 10$ is selected as it balances temporal context against computational cost, and anomaly localization sensitivity. This size is sufficient to capture short-term behavioral transitions and inter-device dependencies while avoiding the excessive smoothing and computational overhead associated with larger temporal windows, it is treated as a fixed parameter in this study to isolate architectural differences. The batch size $B = 32$ is presented in random order during training to reduce gradient bias from temporal autocorrelation while inference proceeds sequentially to preserve the temporal alignment of anomaly scores.

2.6.2 Random Seed Control

Reproducibility is a prerequisite for controlled comparison. Any difference in detection performance must reflect a genuine architectural distinction rather than random variation across weight initializations or mini-batch orderings. To this end, a global random seed of 42 is set at the start of every experiment, applied identically to Python's `random` module, NumPy's random state, PyTorch's CPU random number generator, and all CUDA device generators when a GPU is available:

$$\text{seed} = 42 \;\longrightarrow\; \{\texttt{random},\ \texttt{numpy},\ \texttt{torch CPU},\ \texttt{torch CUDA}\}. \tag{2.10}$$

All results reported in this thesis correspond to a single training run per configuration

under this fixed seed, ensuring exact reproducibility of all reported figures and metrics.

2.6.3 Learning Rate Scheduler

The Adam optimizer is initialized with learning rate $\eta_0 = 10^{-4}$. A `ReduceLROnPlateau` scheduler halves the current learning rate whenever no improvement in validation loss is observed over 3 consecutive epochs:

$$\eta^{(e+1)} = \begin{cases} 0.5 \cdot \eta^{(e)}, & \text{if no improvement for 3 consecutive epochs} \\ \eta^{(e)}, & \text{otherwise,} \end{cases} \tag{2.11}$$

subject to $\eta^{(e)} \geq \eta_{\min} = 10^{-6}$. The scheduler patience of 3 epochs is strictly shorter than the early stopping patience of 10 epochs establishing a two stage response to stagnating validation loss the scheduler reduces the step size first giving the model an opportunity to escape the plateau early stopping then terminates training only if the loss remains stagnant for a further 7 epochs.

2.6.4 Early Stopping

Training terminates early if the validation loss does not improve for 10 consecutive epochs. A patience counter is incremented on stagnation and reset to zero on improvement:

$$\text{counter}^{(e+1)} = \begin{cases} 0, & \text{if } \mathcal{L}_{\text{val}}^{(e)} < \mathcal{L}_{\text{val}}^{*} \\ \text{counter}^{(e)} + 1, & \text{otherwise.} \end{cases} \tag{2.12}$$

Whenever an improvement is recorded, the current model weights are saved as the best checkpoint. Training halts when $\text{counter}^{(e)} \geq 10$, at which point the best checkpoint is restored for all subsequent evaluation. This mechanism prevents overfitting to the training period and ensures that the evaluated model corresponds to its best generalization on held-out normal data.

2.7 Hardware and Software Specifications

All experiments were conducted on a single workstation to ensure consistent runtime conditions across all configurations. Table 2.2 and Table 2.3 summarize the environment.

Configuration management is handled through a single `config.yaml` file specifying all hyperparameters, dataset paths, and evaluation settings, guaranteeing full reproducibility from stored checkpoints.

Table 2.2: Hardware environment used for all experiments.

Component	**Specification**
CPU	Intel Core i7 (64-bit)
RAM	16 GB DDR4
GPU	NVIDIA GPU with CUDA support
Operating System	Windows 11
Storage	NVMe SSD

Table 2.3: Software dependencies and their roles in the pipeline.

Library	**Version**	**Role**
Python	3.10	Runtime environment
PyTorch	2.x	Model definition, training, inference
PyTorch Geometric	2.x	Graph neural network layers
NumPy	1.24+	Numerical array operations
Pandas	2.x	Time series preprocessing
Scikit-learn	1.3+	Min-max normalization
SciPy	1.11+	Statistical tests
Matplotlib	3.7+	Result visualization
PyYAML	6.x	Configuration management

2.8 Summary

This chapter presented the formal problem formulation and unified experimental framework for anomaly detection in smart home multivariate time series. The task is defined as an unsupervised learning problem in which a model is trained exclusively on normal occupancy data and detects anomalies at inference time through elevated prediction error.

Devices are represented as nodes in a learned dependency graph enabling the system to capture inter device relationships not observable from individual channels. The anomaly scoring pipeline converts raw per device prediction errors into calibrated scalar scores through normalization maximum aggregation thresholding and optional smoothing. A unified training procedure controlling hyperparameters random seeds and optimization strategy ensures that performance differences between models reflect architectural properties rather than experimental artifacts.

In the following two chapters, we will first present the smart home datasets used and the

preprocessing pipeline in Chapter 3. Then, in Chapter 4, we will apply this framework to evaluate and compare GNN-based anomaly detection architectures on such data.

Chapter 3: Dataset and Preprocessing

3.1 Introduction

This chapter provides a structured description of the datasets and preprocessing pipeline used to transform raw smart home device logs into model-ready multivariate time series for unsupervised anomaly detection. The data originates from two instrumented smart home environments, the Building Research Establishment installation (BRE) and a second comparable laboratory environment (CU). A merged configuration is also constructed to evaluate model behavior under increased heterogeneity.

The preprocessing pipeline is designed to handle the intrinsic heterogeneity of smart home data, which includes differences in device modalities, sampling patterns, and value ranges. The main objective is to produce temporally aligned, normalized, and consistent multivariate arrays suitable for graph-based learning models.

The transformation process is structured around the following key components:

- **Dataset sources**: Two real-world smart home installations are used, BRE and CU, each containing dense IoT device deployments with different device ecosystems and levels of heterogeneity.
- **Occupancy structure**: Both datasets follow a three-phase structure consisting of a normal training period (Actor 1-T1), an anomalous occupancy period (Actor 2), and a return-to-normal period (Actor 1-T2).
- **Device heterogeneity**: The data includes multiple modalities such as binary event-driven sensors, continuous environmental measurements, and actuator or appliance states, each exhibiting distinct statistical behavior.
- **Data cleaning and filtering**: Raw device channels are filtered based on missingness, variance, and redundancy to retain only informative signals.
- **Temporal alignment and resampling**: Event-based and continuous signals are aligned onto a uniform temporal grid using type-aware aggregation rules that preserve short-duration events while smoothing continuous measurements.
- **Normalization and windowing**: Each channel is independently normalized using training-only statistics and converted into supervised learning samples through a causal sliding window formulation.

The merged BRE plus CU configuration is constructed by aligning both datasets on a shared timestamp axis, producing a higher-dimensional representation that introduces additional complexity due to cross-environment variability. This setup is specifically designed to evaluate robustness under distributional shift.

The preprocessing decisions are driven by both statistical properties of the data and modeling requirements. Binary devices require preservation of sparse activation events, while continuous devices require scaling to ensure comparability across heterogeneous physical units. The resulting dataset structure enables consistent evaluation of graph-based anomaly detection models across all experimental configurations.

3.2 Dataset Description

The experiments use data collected at two instrumented smart home testbeds the Building Research Establishment BRE installation and a second comparable laboratory designated CU. Both environments are furnished residential spaces monitored by a dense network of commercial off the shelf IoT devices communicating over local wireless protocols Zigbee Z Wave Wi Fi. Each device continuously logs state changes or periodic measurements to a central home automation hub and the full event stream is exported as a timestamped CSV file `BREMaster.csv` and `CUMaster.csv` for offline processing.

3.2.1 BRE Dataset

The BRE dataset spans the period from 18 October to 17 November 2022 and contains approximately 2.67 million raw event rows across 94 device channels sampled at native intervals of approximately one second. The physical installation comprises multiple rooms distributed across three floors, each equipped with a multi-sensor unit (Aqara A6 series) measuring temperature, humidity, motion, and illuminance, complemented by binary contact devices (Open/Close series, OC prefix) mounted on doors and windows.

The recording period is divided into three temporally contiguous occupancy phases:

- **Actor 1, Timeline 1 (Actor 1-T1):** 18 October – 07 November 2022. The regular resident follows their normal daily routine without interruption. This phase constitutes the sole source of training and validation data.
- **Actor 2:** 08 November – 10 November 2022. A second, unfamiliar occupant replaces the regular resident for three consecutive days. This period constitutes the *positive* (anomalous) evaluation split.
- **Actor 1, Timeline 2 (Actor 1-T2):** 11 November – 17 November 2022. The regular resident returns. This phase provides a held-out *negative* (normal) evaluation split for

false positive rate estimation.

After device filtering (Section 3.6.1), 61 device channels are retained from the original 94.

3.2.2 CU Dataset

The CU dataset covers the same calendar period as BRE and follows an identical three-phase occupancy structure, with Actor 2 boundaries aligned to the same dates (08–10 November 2022). The raw data contains 370 device channels sourced from a more heterogeneous and home-automation-rich ecosystem than BRE, including smart lighting, smart plugs, actuators, and specialized appliances in addition to environmental and binary contact devices. After filtering, 31 channels are retained from the initial 370.

3.2.3 Merged BRE+CU Dataset

A merged configuration is constructed by aligning the BRE and CU datasets on their shared UTC timestamp axis using a time-ordered nearest-neighbor join. Any channel present in only one dataset receives zeros in the rows contributed by the other, and the merged result is forward-filled then backward-filled to resolve residual missing values from the asynchronous sampling clocks. The merged dataset constitutes a single multivariate time series with $N_{\mathrm{BRE}} + N_{\mathrm{CU}} = 61 + 31 = 92$ channels and is used to evaluate whether joint training across two physically distinct residential environments improves or degrades detection quality.

Table 3.1 summarizes the principal statistics of the three configurations.

Table 3.1: Summary statistics for the three dataset configurations at 1 s resolution after device filtering.

Property	**BRE**	**CU**	**BRE+CU**
Channels (after filtering)	61	31	92
Actor 1-T1 duration	21 d	21 d	21 d
Actor 2 duration	3 d	3 d	3 d
Actor 1-T2 duration	7 d	7 d	7 d
Raw event rows (≈)	2.67M	2.67M	2.67M
Native sampling rate	≈1 s	≈1 s	≈1 s

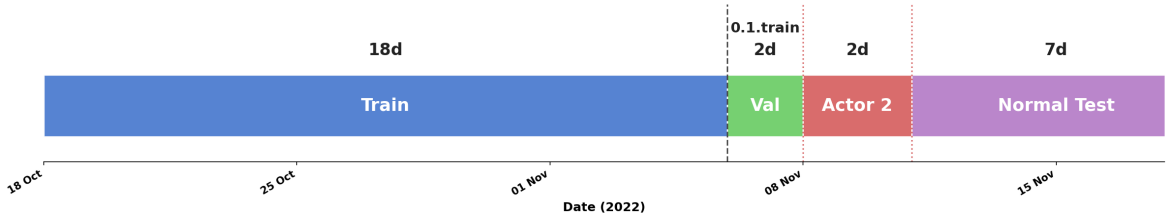

Figure 3.1: Chronological timeline of the four data splits for the BRE (top) and CU (bottom) installations.

3.3 Anomaly Definition and Experimental Assumptions

Anomaly definition. An anomaly is defined as any time-step during which the joint distribution of device readings deviates significantly from the patterns established during Actor 1's normal occupancy. Crucially, this definition does not require any individual device to produce an extreme or out-of-range value. Instead, as discussed in Section 1.2, the anomaly is of the *inter-metric* type: it manifests as a change in the co-activation patterns, timing relationships, and conditional dependencies among devices, even while each device's marginal distribution may remain broadly normal.

Concretely the anomaly window corresponds to the three day period during which Actor 2 occupies the home. Actor 2's behavioral routine which rooms are visited in what order at what times and in combination with which other devices differs from Actor 1's routine. This difference is not concentrated in a single device or a single time step but is distributed across the full multivariate state of the home.

Experimental assumptions. The experimental framework rests on the following assumptions:

1. **Unsupervised training.** The model is trained exclusively on data collected during Actor 1's normal occupancy (Actor 1-T1). No anomaly labels, no Actor 2 data, and no information about what constitutes an anomaly is available during training.

2. **Stationarity of normal behavior.** Actor 1's behavioral patterns during Actor 1-T1 (training) are assumed to be representative of Actor 1's patterns during Actor 1-T2 (normal test). Any performance degradation due to natural behavioral drift within Actor 1's routine is treated as a source of false positives and is quantified through the false positive rate on Actor 1-T2

3. **Ground truth labeling.** The entire Actor 2 period is treated as anomalous. Individual time steps within Actor 2's stay are not labeled.

4. **No adaptation at test time.** The trained model is applied to the test splits without any fine-tuning or threshold re-calibration. All scoring parameters are derived exclusively from training data.

3.4 Devices Characteristics and Data Representation

The devices retained in each installation are heterogeneous in type, physical measurement unit, native sampling behavior, and value range. Understanding this heterogeneity is essential for motivating the preprocessing decisions described in Section 3.6.

3.4.1 BRE Device Suite

The 61 retained BRE channels belong to two broad modality classes distributed across ten named zones ground floor hallway kitchen and living room first floor bedrooms and hallway second floor bedroom sitting room and hallway.

Continuous environmental channels come from Aqara multi-sensor units (prefix `A6`), each reporting up to four measurements: temperature (°C), relative humidity (%), illuminance (lux), and a binary motion flag. **Binary contact channels** (prefix `OC`) are mounted on interior doors, external doors, and windows, outputting 0 (closed) or 1 (open).

Table 3.2: BRE device channel composition after filtering.

Modality	**Count**	**Type**	**Typical range**
Temperature	22	Continuous	15—28 °C
Humidity	11	Continuous	25—75 %
Illuminance	12	Continuous	0—2000 lux
Motion	12	Binary $\{0, 1\}$	—
Contact	8	Binary $\{0, 1\}$	—
Total	**61**		

3.4.2 CU Device Suite

The 31 retained CU channels come from a more functionally diverse ecosystem. Unlike BRE, which is dominated by passive environmental devices, CU includes a substantial number of actuator and smart appliance channels whose state is directly controlled by occupant behavior.

Table 3.3: CU device channel composition after filtering.

Modality	**Count**	**Type**
Motion / occupancy	4	Binary $\{0, 1\}$
Contact	4	Binary $\{0, 1\}$
Vibration	3	Binary $\{0, 1\}$
Water leak	2	Binary $\{0, 1\}$
Wireless switch action	2	Binary $\{0, 1\}$
Smart plugs / switches	3	Binary $\{0, 1\}$
Smart lighting	5	Binary $\{0, 1\}$
Temperature / humidity	4	Continuous
$PM_{2.5}$ / air quality	1	Continuous
Climate / heating	1	Continuous
Appliances (kettle, vacuum, humidifier)	1	Binary $\{0, 1\}$
Total	**31**	

3.4.3 Data Modalities and Statistical Characteristics

The two modality classes exhibit fundamentally different statistical properties that directly inform preprocessing decisions.

Binary (event-driven) channels generate events only when their state changes, resulting in long quiescent periods of constant value interspersed with brief transitions. This sparsity makes them highly informative about occupancy events (a door opening, a light being switched on) but problematic for standard time series models that expect continuous variation. Their value range is fixed at $\{0, 1\}$.

Continuous (periodic) channels are logged at regular intervals and evolve smoothly over time. Temperature, humidity, and illuminance exhibit strong diurnal periodicity driven by heating schedules and natural light cycles. Their value ranges span multiple physical scales (e.g., 0–2000 lux vs. 15–28 °C), requiring normalization before any cross-channel comparison.

This heterogeneity both in data modality and in native sampling rate is the primary preprocessing challenge addressed in Section 3.6.

3.5 Exploratory Data Analysis

Exploratory data analysis is conducted prior to modeling to characterize the statistical properties of the device streams and to motivate the preprocessing and modeling decisions.

Missing values. After loading the raw CSV binary devices that have not changed state since the previous logging interval are not re logged resulting in NaN entries at every second during quiescent periods. The CU dataset additionally contains channels that are entirely null throughout the recording period devices registered in the hub but never commissioned. These are removed during the filtering step. For retained channels the fraction of missing values varies substantially binary event driven devices exhibit high sparsity whereas continuous environmental devices are logged at regular intervals with minimal missingness.

Diurnal activity patterns. During Actor 1-T1 periods, device activations follow a pronounced diurnal pattern consistent with single-occupancy residential routines. Motion and illuminance channels exhibit peaks between 07:00–09:00 and 18:00–23:00, corresponding to morning preparation and evening activity. Environmental channels evolve slowly and follow weekly periodicity partially reflecting heating schedules.

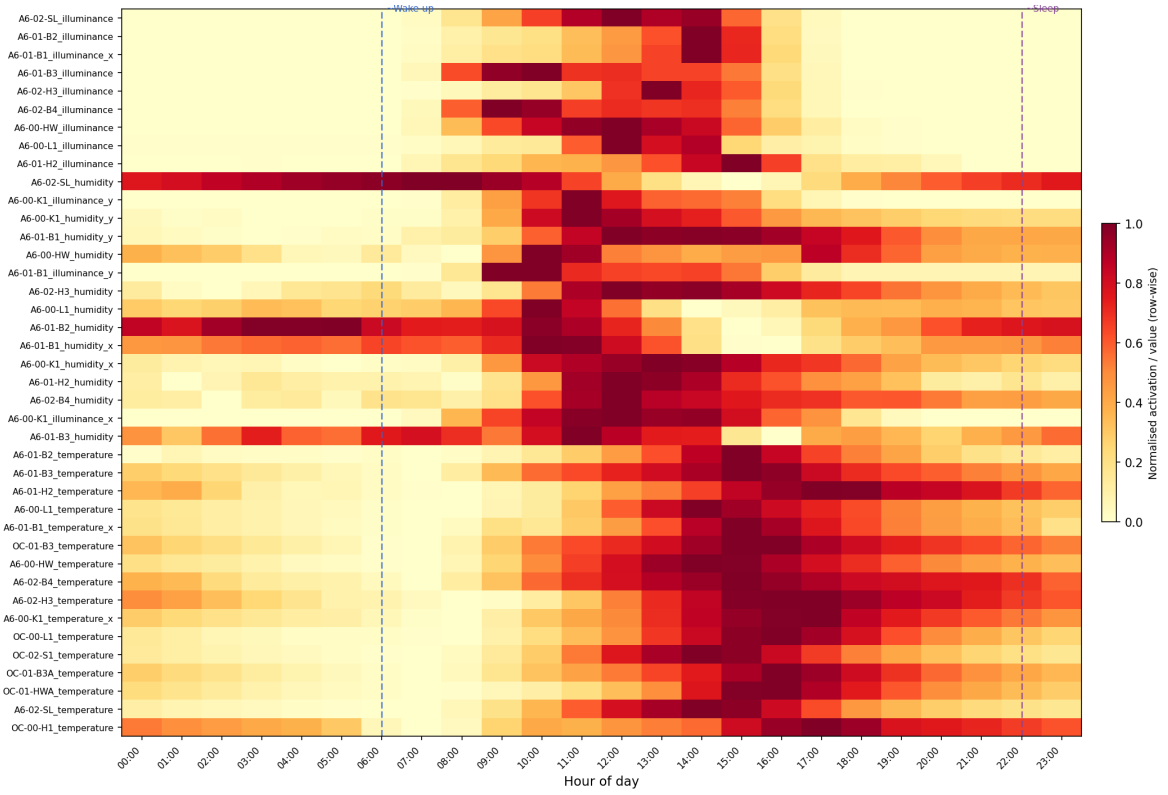

Figure 3.2: Diurnal activity heatmap for the BRE dataset.

Correlation structure. A pairwise Pearson correlation analysis of the BRE training split reveals a block diagonal structure that maps onto the physical floor plan devices within the same room form clusters of high positive correlation while inter floor correlations are near zero. This spatial structure motivates the graph based modeling approach devices that are co located and functionally related are natural candidates for edges in the learned device graph.

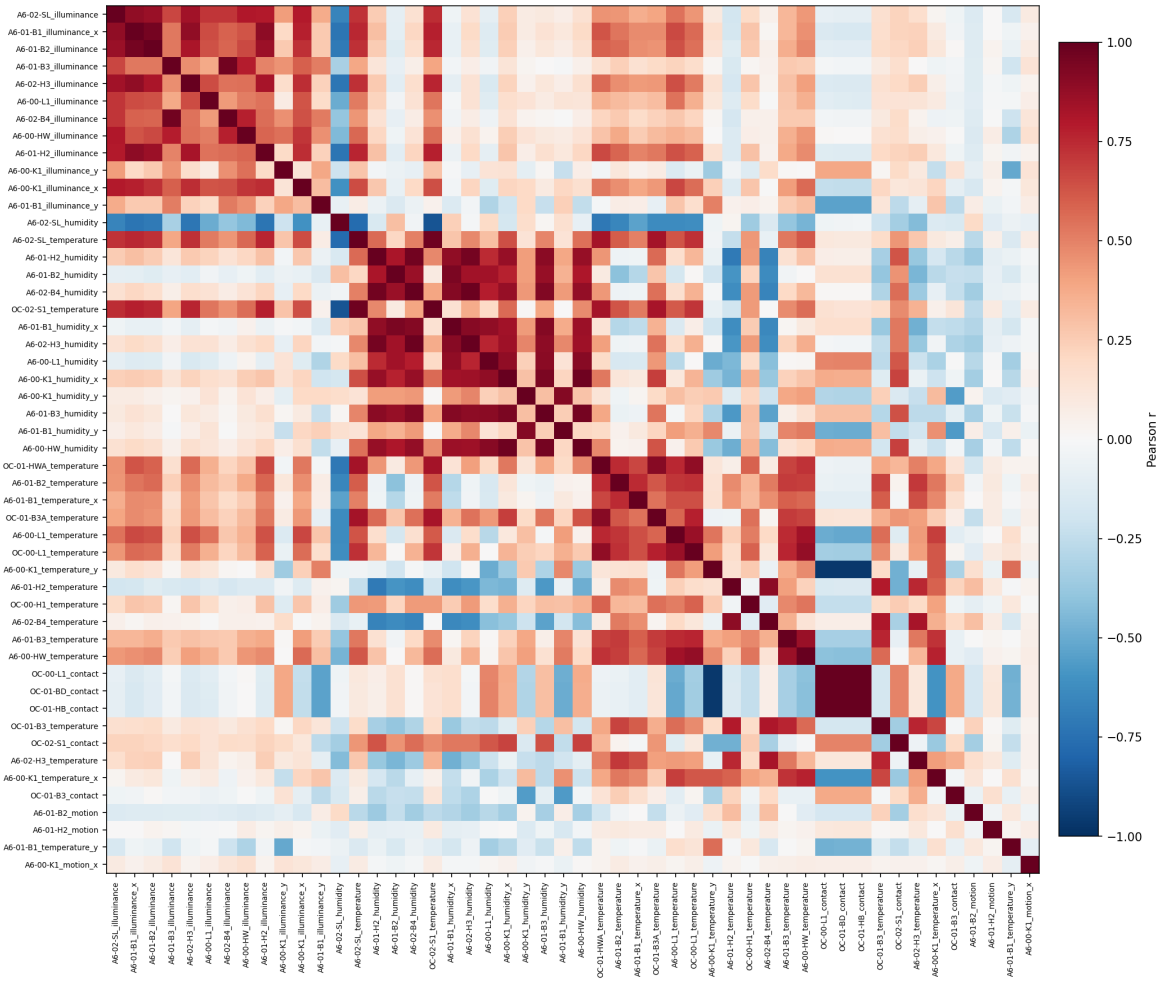

Figure 3.3: Pairwise Pearson correlation matrix of the 61 BRE device channels computed on the Actor 1-T1 training split.

Distribution shifts between Actor 1 and Actor 2. For the majority of BRE channels the marginal empirical distributions during Actor 1 T1 and Actor 2 are visually similar temperature values span the same range contact devices fire at comparable rates and illuminance readings occupy the same quantile range. The anomaly is therefore not detectable by a univariate threshold on any single channel as shown in the Figure 3.2.

3.6 Preprocessing Pipeline

Raw device event logs are transformed into model-ready arrays through a sequential preprocessing pipeline. Figure 3.4 provides a schematic overview.

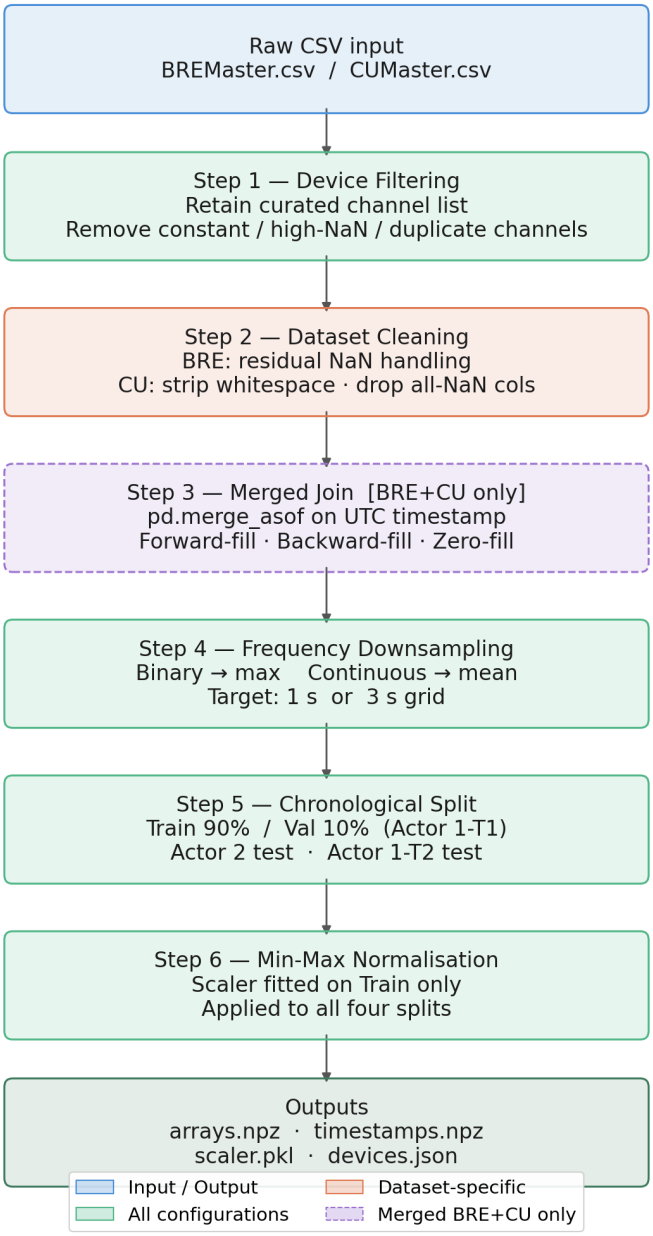

Figure 3.4: End-to-end preprocessing pipeline.

3.6.1 Data Filtering

The raw device set is reduced to a curated subset of channels likely to carry behavioral information relevant to occupancy detection. Retained channels are specified in two plain text configuration files `BRE_selected_devices.txt` that have 61 channels and `CU_selected_devices.txt` that have 31 channels.Channels are excluded based on three criteria:

- **Constant-value channels**: channels whose standard deviation across the entire Actor 1-T1 training period falls below 10^{-4} carry no temporal information.
- **High-missingness channels**: channels with a missing value rate exceeding 50% of time-steps across the training period are excluded, as forward-filling would introduce a flat, uninformative signal over the majority of time-steps.
- **Redundant channels**: duplicate readings from the same physical device reported under multiple identifiers are deduplicated, retaining the channel with the lowest missing value rate.

For the BRE dataset this step retains 61 channels from an initial set of 94. For CU 31 channels are retained from 370 a high rejection rate reflecting the large number of unresponsive or redundant channels in the CU hub.

3.6.2 Data Cleaning

After filtering, dataset-specific cleaning steps resolve data quality issues that differ between BRE and CU.

BRE cleaning. The BRE dataset is structurally well-formed and requires no cleaning beyond the filtering step. Residual missing values from quiescent binary devices are resolved in the downsampling step.

CU cleaning. Three targeted operations are applied. First, trailing whitespace is stripped from all column names. Second, columns that are entirely null throughout the recording period are removed. Third, when CU data is used in the merged configuration, CU cleaning is applied immediately after loading before the timestamp-aligned join.

Merged dataset cleaning. After the nearest neighbor timestamp join residual missing values are resolved in three passes forward fill backward fill then zero fill for any remaining NaNs.

3.6.3 Temporal Down-sampling

Following cleaning, the dataset is resampled to a uniform temporal grid at one of two target frequencies: **1 s** or **3 s**. The aggregation rule is device-type-dependent:

- **Binary devices** (contact, motion, vibration, occupancy) are aggregated by `max` over each resampling interval, so that any activation event is preserved. This is critical for short-duration events — such as a door contact transition — that would be erased under mean aggregation.
- **Continuous devices** (temperature, humidity, illuminance) are aggregated by `mean`, consistent with their slowly varying nature.

Table 3.4 compares the two frequency configurations.

Table 3.4: Comparison of downsampling frequency configurations for the BRE dataset.

Property	**1 s**	**3 s**
Window context ($W = 10$)	10 s	30 s
Training rows (Actor 1-T1, $\approx$)	1.63M	544K
Binary event preservation	Exact	`max` over 3 s bin
Anomaly localization resolution	High	Moderate
Computational cost per epoch	Higher	Lower

3.6.4 Train, Validation and Test Split

The downsampled dataset is partitioned into four non-overlapping, chronologically ordered splits by extracting rows whose UTC timestamp falls within the occupancy boundaries:

$$\begin{aligned} \text{Actor 1-T1} &: \texttt{2022-10-18} \rightarrow \texttt{2022-11-07} \\ \text{Actor 2} &: \texttt{2022-11-08} \rightarrow \texttt{2022-11-10} \\ \text{Actor 1-T2} &: \texttt{2022-11-11} \rightarrow \texttt{2022-11-17} \end{aligned}$$

The Actor 1-T1 segment is further subdivided into a training set (first 90%) and a validation set (last 10%):

$$\text{Train} = \text{Actor 1-T1}\big[: \lfloor 0.9 \times T_{\text{T1}} \rfloor\big], \quad \text{Val} = \text{Actor 1-T1}\big[\lfloor 0.9 \times T_{\text{T1}} \rfloor :\big]. \tag{3.1}$$

The split boundaries are defined by calendar dates and are identical across all model architectures, downsampling frequencies, and dataset configurations.

3.6.5 Normalization

After splitting, each of the N device channels is independently scaled to the interval $[0, 1]$ using a min-max scaler fitted *exclusively* on the training split:

$$\tilde{x}_t^{(i)} = \frac{x_t^{(i)} - \min_i^{\text{train}}}{\max_i^{\text{train}} - \min_i^{\text{train}} + \epsilon}, \qquad \epsilon = 10^{-8}. \tag{3.2}$$

The fitted scaler is then applied without modification to the validation and test sets, ensuring that no information leakage from future splits is introduced during preprocessing.

Min-max scaling is preferred over z-score standardization for two main reasons. First, several channels are binary or naturally bounded within $[0, 1]$, and min-max normalization preserves this physical interpretation directly. Second, z-score normalization is more sensitive to outliers in the training split, which can compress the effective dynamic range of other sensor channels and reduce discriminative signal strength.

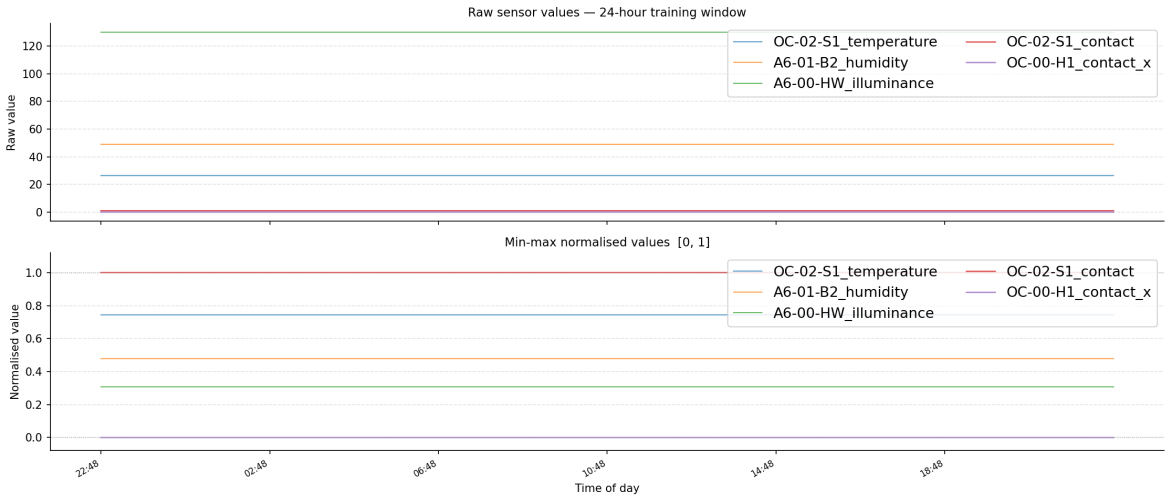

Figure 3.5: Effect of min-max normalization on a representative subset of BRE device channels over a 24-hour window from the Actor 1-T1 training split.

Figure 3.5 illustrates the effect of this transformation on representative sensor channels. The top panel shows the raw signals, where heterogeneous physical units lead to strong scale imbalance across devices. For instance, the illuminance channel operates at significantly higher numeric magnitude than humidity or temperature sensors, while binary contact sensors remain near zero throughout the observation window.

In contrast, the bottom panel shows the same signals after min-max normalization. All channels are mapped into a common bounded range $[0, 1]$, eliminating scale dominance from

high-magnitude sensors. This transformation preserves the temporal shape of each signal while making all device channels directly comparable during model training.

A key observation is that even though all signals appear visually flat over time (due to stable behavioral patterns in this window), their relative ordering changes after scaling. Binary contact signals become saturated at the upper and lower bounds, while continuous signals are redistributed proportionally within the normalized space. This highlights an important property: min-max normalization does not introduce temporal variation, but rather re-encodes inter-device magnitude relationships into a unified numerical domain.

Overall, this preprocessing step ensures stable optimization for both GDN and MTAD-GAT by preventing high-range sensors from dominating the learning process while preserving meaningful relative differences between device channels.

3.6.6 Sliding Window Construction

The final preprocessing step converts each normalized split array into a dataset of supervised input-target pairs through a stride-1 causal sliding window of length W. For a split of length T, the i-th sample is:

$$\big(\mathbf{x}^{(i)},\ \mathbf{y}^{(i)}\big) = \big(\mathbf{X}[i\ :\ i{+}W[,\ \ \mathbf{Y}[i{+}W]\big), \tag{3.3}$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^{W\times N}$ is the causal context window and $\mathbf{y}^{(i)} \in \mathbb{R}^{N}$ is the next-step prediction target.

The learning task is entirely self supervised no anomaly labels are required and the sole supervision signal is the model's ability to predict future device states from causal context. At 1 s resolution the 21 day Actor 1 T1 training period produces approximately 1.63 million training windows at 3 s resolution the same period yields approximately 544 000 windows.

Implications of data preprocessing on evaluation comparison. The preprocessing choices made in this chapter directly affect model behavior and the interpretation of results. The type-aware downsampling rule (`binary` $\rightarrow$ `max`, `continuous` $\rightarrow$ `mean`) ensures that short-duration activation events are preserved even at coarser temporal resolution making the 1,s and 3,s configurations genuinely comparable. The training-only normalization prevents leakage from anomalous periods into the feature representation ensuring that any score elevation during Actor 2's stay reflects genuine model uncertainty rather than a normalization artifact. The sliding window size W determines the temporal context available to the model too small a window may miss behavioral patterns that unfold over multiple device interactions while too large a window risks spanning multiple distinct activity episodes within a single input reducing the model's ability to detect transitions. These trade-offs are analyzed empirically in Chapter 4.

3.7 Summary

This chapter presented the smart home datasets used in this study and described the preprocessing pipeline applied to transform heterogeneous IoT sensor streams into graph-ready multivariate time series representations. The process included device selection, temporal alignment, resampling, normalization, and sliding-window generation. Particular attention was given to preserving meaningful behavioral information while ensuring consistency across datasets. The resulting standardized representations provide the foundation for the comparative evaluation of GNN-based anomaly detection architectures presented in the next chapter.

Chapter 4: Systematic Comparative Study

4.1 Introduction

This chapter presents the systematic comparative evaluation conducted on two graph-based architectures for unsupervised anomaly detection in smart home multivariate time series: the Graph Deviation Network (GDN) and the Multivariate Time series Anomaly Detection via Graph Attention Network (MTAD-GAT). Both models leverage graph neural network mechanisms to capture dependencies between heterogeneous device channels, yet they differ significantly in their architectural design, learning objectives, and anomaly scoring strategies.

The main objective is to understand how these design choices affect detection behavior under a unified experimental framework. Rather than focusing only on overall accuracy, the analysis emphasizes how robust the learned representations are across datasets, and how stable the resulting anomaly scores are under varying operational conditions.

The comparison is structured around the following key aspects

- **Learning objective**: GDN relies on forecasting based anomaly detection where deviations between predicted and observed values indicate anomalies, while MTAD-GAT integrates both forecasting and reconstruction objectives within a joint learning framework.
- **Graph construction**: GDN builds a sparse graph based on learned device embeddings and top k similarity, whereas MTAD-GAT relies on dense attention mechanisms that allow full pairwise interactions between nodes.
- **Temporal modeling**: GDN uses sliding window based forecasting, while MTAD-GAT combines convolutional feature extraction with recurrent modeling to capture both short term and long term dependencies.
- **Anomaly scoring strategies**: GDN produces a single forecasting error based score, while MTAD-GAT allows multiple scoring modes including forecasting, reconstruction, and their combination.
- **Robustness across environments**: The behavior of both models is analyzed under single installation datasets BRE and CU and a more challenging merged dataset combining heterogeneous environments.

The evaluation is carried out across three dataset configurations BRE, CU, and a merged BRE plus CU dataset designed to introduce higher variability and inter installation differ-

ences. This setup allows assessment of model generalization under progressively more complex conditions.

Performance is assessed through detailed metrics to capture fine-grained detection ability. Additionally, a deeper analysis is conducted on anomaly score distributions, threshold sensitivity, temporal stability of detections, and device-level interpretability. Together, these evaluations offer a comprehensive view of how each architecture behaves in practice beyond aggregate performance metrics.

4.2 Models Under Study

The Graph Deviation Network (GDN) and the Multivariate Time series Anomaly Detection via Graph Attention Network (MTAD-GAT) are selected for their strong performance in prior work and their fundamentally different approaches to the problem, enabling a meaningful analysis of how architectural choices influence detection behavior in smart home environments.

Comparative Perspective

The two models embody distinct detection paradigms that make them well suited to a controlled comparison.

GDN follows a **forecasting-based** paradigm. The model learns to predict the next device state from a sliding window of past observations and detects anomalies as large deviations between predictions and observations. Its graph is sparse and constructed from trainable node embeddings, focusing attention on the strongest inter-device dependencies.

MTAD-GAT follows a **joint forecasting and reconstruction** paradigm. The model simultaneously learns to forecast future device states and to reconstruct the full input window from a compact latent representation. Anomaly scores can be derived from the forecasting error alone, the reconstruction error alone, or a weighted combination of both. This design allows the contribution of each learning objective to be isolated and compared empirically.

Both models rely on graph-based representations of device relationships, providing a consistent structural framework that isolates the impact of their differing learning objectives.

4.3 Model Architectures

4.3.1 Graph Deviation Network

Deng and Hooi [2] introduced GDN, framing MTS anomaly detection as a graph deviation problem where normal behavior corresponds to regular prediction patterns across a learned

device dependency graph and anomalies correspond to deviations from those patterns. GDN is built from four components.

Device embeddings. Each of the N devices receives a trainable embedding vector $\mathbf{e}_i \in \mathbb{R}^d$ representing its long-term behavioral identity. These embeddings drive graph construction and condition the attention mechanism to allow heterogeneous aggregation across devices with different physical roles.

Graph structure learning. GDN constructs a sparse directed graph by retaining, for each device i, the k nearest neighbors in the embedding space under cosine similarity:

$$e_{ji} = \frac{\mathbf{e}_i^\top \mathbf{e}_j}{\|\mathbf{e}_i\| \, \|\mathbf{e}_j\|}, \qquad A_{ji} = \begin{cases} 1, & j \in \mathrm{TopK}_k\big(\{e_{\ell i}\}\big) \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

This sparsification focuses the model on the most informative inter-device correlations.

Graph attention forecasting. A graph attention layer aggregates neighborhood embeddings to produce a contextual representation for each device, which is then passed through fully connected layers to generate a next-step prediction.

Anomaly scoring. The raw prediction error $\mathrm{Err}_i(t) = |x_t^{(i)} - \hat{x}_t^{(i)}|$ is normalized by the training-split median and IQR, and the system-level score is $S(t) = \max_i a_i(t)$.

4.3.2 MTAD-GAT

Zhao *et al.* [3] proposed MTAD-GAT, a self-supervised framework that simultaneously models dependencies in the feature dimension (inter-device) and the temporal dimension (across time-steps).

Dual graph attention layers. Following a 1-D convolution for local feature extraction, the model passes representations through two parallel GAT layers. The feature-oriented GAT treats the N device channels as nodes in a complete graph and learns attention weights over all N^2 inter-device pairs. The temporal GAT treats the W time-steps as nodes and learns temporal attention weights, acting as a learnable positional encoding. The outputs of both GAT layers and the convolution are concatenated and fed to a GRU for long-range temporal modeling.

Joint optimization. MTAD-GAT trains two prediction heads simultaneously. A forecasting head is minimized by MSE on the next-step prediction, and a reconstruction head based

on a VAE is minimized by the ELBO. The joint loss $\mathcal{L} = \mathcal{L}_{\text{for}} + \mathcal{L}_{\text{rec}}$ is optimized in a single backward pass.

Anomaly scoring. MTAD-GAT produces three anomaly scores that are evaluated separately in this study. The forecast score $S_{\text{for}}(t)$ is derived from the next-step prediction error only. The reconstruction score $S_{\text{rec}}(t)$ is derived from the VAE reconstruction error only. The combined score $S_{\text{comb}}(t) = \alpha \cdot S_{\text{for}}(t) + \beta \cdot S_{\text{rec}}(t)$ uses a weighted sum with $\alpha = \beta = 0.5$.

4.3.3 Architectural Comparison

Table 4.1 summarizes the key architectural differences between the two models. The most consequential distinction is the graph construction strategy. GDN's sparse learned graph focuses attention on the strongest pairwise dependencies, while MTAD-GAT's dense complete graph captures all pairwise correlations simultaneously. The availability of three distinct scoring modes in MTAD-GAT enables a principled decomposition of the contribution of each learning objective to detection performance.

Table 4.1: Architectural comparison of GDN and MTAD-GAT.

Dimension	**GDN**	**MTAD-GAT**
Detection paradigm	Forecasting only	Forecasting + Reconstruction
Graph construction	Sparse top-k learned embeddings	Dense pairwise attention
Graph topology	Sparse, directed, static after training	Fully connected, bidirectional
Temporal encoding	Direct sliding window	Temporal Conv + GRU
Attention mechanism	Embedding-driven GAT	Scaled dot-product (feature and time)
Anomaly score	Max normalized forecast error	Forecast / Recon / Combined
Key hyperparameters	d=64, k=15, heads=1	kernel=7, GRU$_h$=150, α=β=0.5

4.4 Experimental Setup

All experiments follow the standardized preprocessing, training, and evaluation pipeline introduced in Chapter 2. The same data preparation strategy, window generation procedure, normalization pipeline, training schedule, and thresholding protocol are applied across all models and datasets to ensure a fair and controlled comparison. The hardware and software environment is specified in Section 2.7.

The experiments are conducted on three dataset configurations:

- **BRE**: a single-installation smart-home dataset.
- **CU**: a second smart-home installation with a different device topology and occupant activity profile.
- **Merged BRE+CU**: a heterogeneous multi-installation configuration obtained by combining both environments into a single multivariate representation.

Four anomaly scoring configurations are evaluated:

- GDN forecasting score.
- MTAD-GAT forecasting score.
- MTAD-GAT reconstruction score.
- MTAD-GAT combined forecasting-reconstruction score.

For MTAD-GAT, all three scoring modes are derived from the same trained model weights and differ only in the computation of the final anomaly score.

4.5 Quantitative Performance Comparison

4.5.1 Evaluation Metrics

The comparison focuses on point-wise anomaly detection behavior. Three metrics are used throughout the evaluation.

Detection Rate (DR). The Detection Rate measures the proportion of anomalous windows correctly identified during the Actor 2 period:

$$\mathrm{DR} = \frac{\sum_{t \in \mathcal{A}} \mathbf{1}(S(t) > \tau)}{|\mathcal{A}|} \tag{4.2}$$

where $\mathcal{A}$ denotes the set of anomalous time-steps, $S(t)$ is the anomaly score at time-step t, and τ is the decision threshold.

False Positive Rate (FPR). The False Positive Rate measures the proportion of normal windows incorrectly classified as anomalous during the Actor 1-T2 period:

$$\text{FPR} = \frac{\sum_{t \in \mathcal{N}} \mathbf{1}(S(t) > \tau)}{|\mathcal{N}|} \tag{4.3}$$

where $\mathcal{N}$ represents the set of normal time-steps.

Detection Delay. Detection delay measures the elapsed time steps between the beginning of the Actor 2 period and the first detected anomaly:

$$\text{Delay} = \min\{t \in \mathcal{A} : S(t) > \tau\} - t_{\text{start}} \tag{4.4}$$

where t_{start} is the first time-step of the Actor 2 period.

4.5.2 Overall Performance Comparison

Table 4.2 summarizes the quantitative results across all datasets and scoring configurations.

Table 4.2: Point-wise anomaly detection performance across all experimental configurations using a threshold at the 90th percentile of training scores.

Dataset	Scoring	DR	FPR	Delay
BRE	GDN (forecast)	9.39%	10.00%	4,980
	MTAD-GAT (forecast)	32.73%	10.00%	4,980
	MTAD-GAT (recon)	99.67%	10.00%	0
	MTAD-GAT (combined)	60.94%	10.00%	1
CU	GDN (forecast)	7.61%	10.00%	810
	MTAD-GAT (forecast)	24.34%	10.00%	810
	MTAD-GAT (recon)	39.86%	10.00%	0
	MTAD-GAT (combined)	25.96%	9.99%	810
Merged	GDN (forecast)	0.00%	9.89%	133,902
	MTAD-GAT (forecast)	41.71%	10.00%	810
	MTAD-GAT (recon)	93.62%	10.00%	811
	MTAD-GAT (combined)	63.40%	10.00%	810

The results reveal substantial behavioral differences between the two architectures. GDN

consistently behaves as a conservative forecasting-based detector, producing relatively low detection rates across all datasets while maintaining stable false positive behavior. In contrast, MTAD-GAT exhibits significantly stronger sensitivity to behavioral deviations, particularly when reconstruction-based scoring is used.

The reconstruction scoring mode of MTAD-GAT consistently achieves the highest detection rates across all datasets. This behavior indicates that reconstruction error captures deviations in global behavioral structure more effectively than forecasting error alone. The forecasting-only configuration remains substantially more stable but less sensitive to localized deviations.

The merged BRE+CU configuration produces the largest divergence between architectures. GDN exhibits severe degradation under the heterogeneous multi-installation setting, whereas MTAD-GAT maintains high detection performance. This suggests that dense graph attention and joint reconstruction-based learning provide greater robustness to inter-installation variability and device heterogeneity.

4.5.3 Performance Across Dataset Configurations

BRE dataset. BRE produces the strongest overall detection performance for all configurations. MTAD-GAT reconstruction achieves near-complete coverage of anomalous windows with immediate detection at the start of the Actor 2 period. The forecasting configurations exhibit substantially lower sensitivity and require longer periods of behavioral deviation before the anomaly score crosses the threshold.

The strong reconstruction performance suggests that Actor 2 induces a global deviation in device activation structure that is captured effectively by latent reconstruction error. GDN, however, only responds to a limited subset of highly distinguishable temporal patterns, resulting in sparse detections.

CU dataset. All configurations experience reduced detection rates on CU compared to BRE. The reduction indicates that behavioral differences between Actor 1 and Actor 2 are less separable in the CU environment.

The reconstruction mode remains the strongest MTAD-GAT configuration, though the gap between reconstruction and forecasting is smaller than in BRE. This suggests that CU contains weaker global distributional shifts and that part of the anomaly signal is carried through localized forecasting deviations rather than large-scale reconstruction inconsistencies.

GDN again behaves conservatively and detects only a limited fraction of anomalous windows.

Merged BRE+CU dataset. The merged configuration introduces substantial inter-installation heterogeneity due to the combination of two distinct smart-home layouts and device ecosys-

tems.

GDN experiences a near-collapse in detection capability under this setting. The sparse embedding-based graph learned during training is unable to generalize effectively across the enlarged and diverse device space.

MTAD-GAT remains comparatively robust. The dense graph attention mechanism enables flexible pairwise interactions between channels, while the reconstruction objective preserves sensitivity to broader behavioral inconsistencies even under highly heterogeneous conditions.

4.5.4 Threshold Sensitivity Analysis

The anomaly threshold directly controls the balance between detection rate (DR) and false positive rate (FPR). To analyze the robustness of each architecture under different operating conditions, the percentile threshold is progressively varied from the 50th to the 100th percentile of the training anomaly score distribution.

The resulting DR—FPR curves provide a threshold-sensitive view of model behavior in the unsupervised anomaly detection setting. Unlike ROC or Precision—Recall analysis, which are primarily designed for fully supervised classification tasks, the DR—FPR formulation more directly reflects the operational trade-off between anomaly sensitivity and false alarm suppression.

Architectural Comparison: GDN vs MTAD-GAT Combined

Figure 4.1 compares the DR—FPR behavior of GDN and the MTAD-GAT combined scoring strategy.

The two architectures exhibit markedly different threshold behaviors. GDN demonstrates a relatively limited operational range. As the threshold increases, the detection rate decreases steadily before undergoing a sharp collapse in the high-percentile region. This behavior indicates that only a limited subset of anomalous windows produces scores substantially higher than the normal distribution. Consequently, small increases in the threshold rapidly eliminate a large fraction of true detections.

In contrast, MTAD-GAT combined scoring maintains substantially higher detection performance across a much wider threshold interval. The detection rate remains stable over a broad percentile range before declining more gradually at higher thresholds. This indicates a stronger separation between normal and anomalous score distributions, allowing the model to preserve high sensitivity even under stricter threshold conditions.

The broader stable operating region of MTAD-GAT reflects the advantages of its dense graph attention mechanism and dual-objective learning framework. By combining forecasting and reconstruction signals, the architecture captures both local temporal deviations and

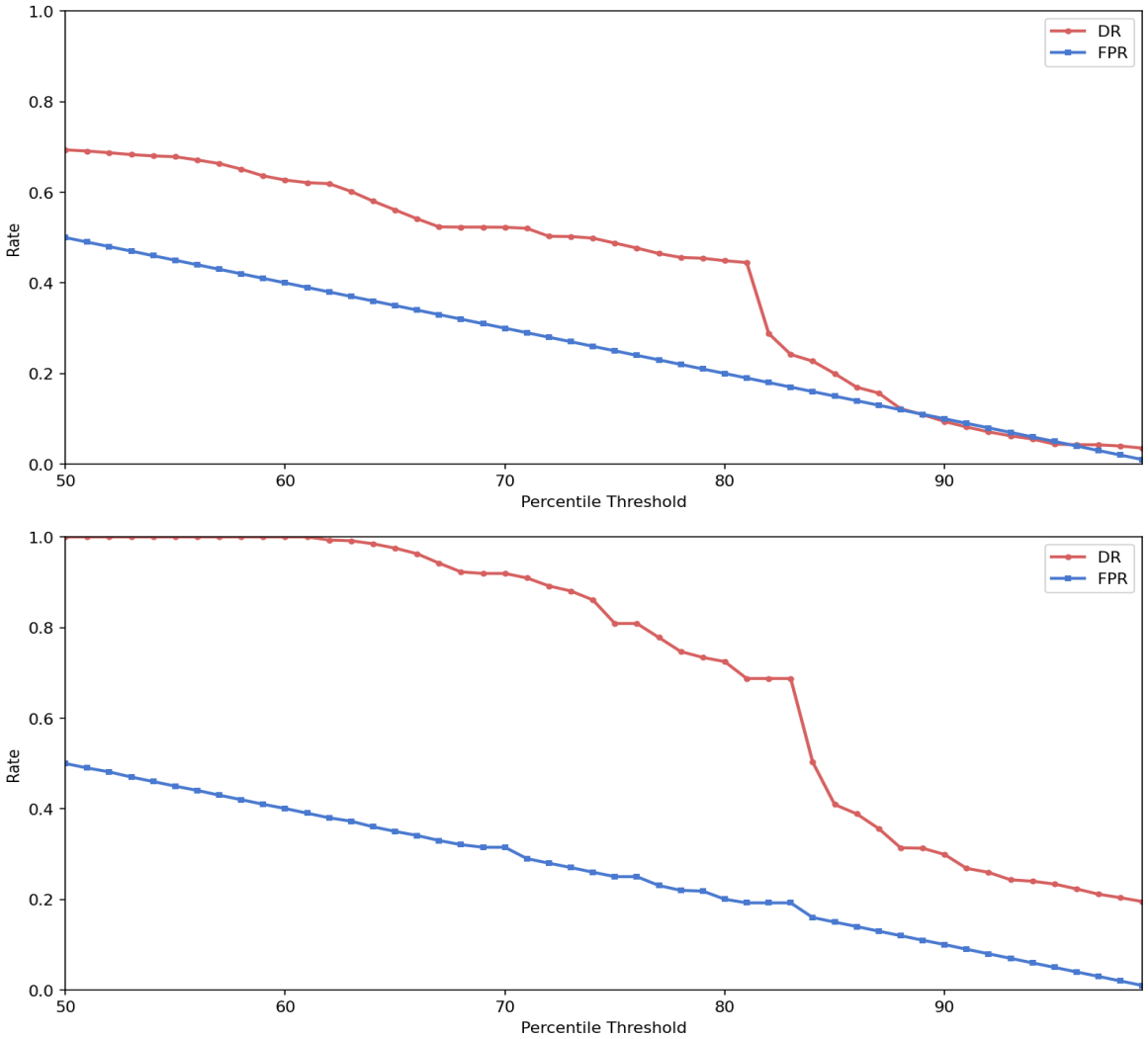

Figure 4.1: Detection Rate (DR) and False Positive Rate (FPR) across percentile thresholds for GDN (TOP) and MTAD-GAT combined scoring (BOTTOM).

global distributional inconsistencies across multiple correlated device channels. GDN, in comparison, relies primarily on sparse forecasting relationships and therefore reacts only to more localized or strongly pronounced deviations.

Overall, the DR—FPR comparison confirms that MTAD-GAT provides greater threshold robustness, stronger anomaly separability, and more stable detection behavior than GDN under varying operational conditions.

Internal Analysis of MTAD-GAT Scoring Strategies

To better understand the contribution of each learning objective, Figure 4.2 compares the forecasting-based and reconstruction-based scoring modes of MTAD-GAT.

The forecasting-based scoring mode exhibits a smooth and progressive decay in detection performance as the threshold increases. The absence of abrupt transitions indicates relatively stable score distributions between normal and anomalous windows. Forecasting errors there-

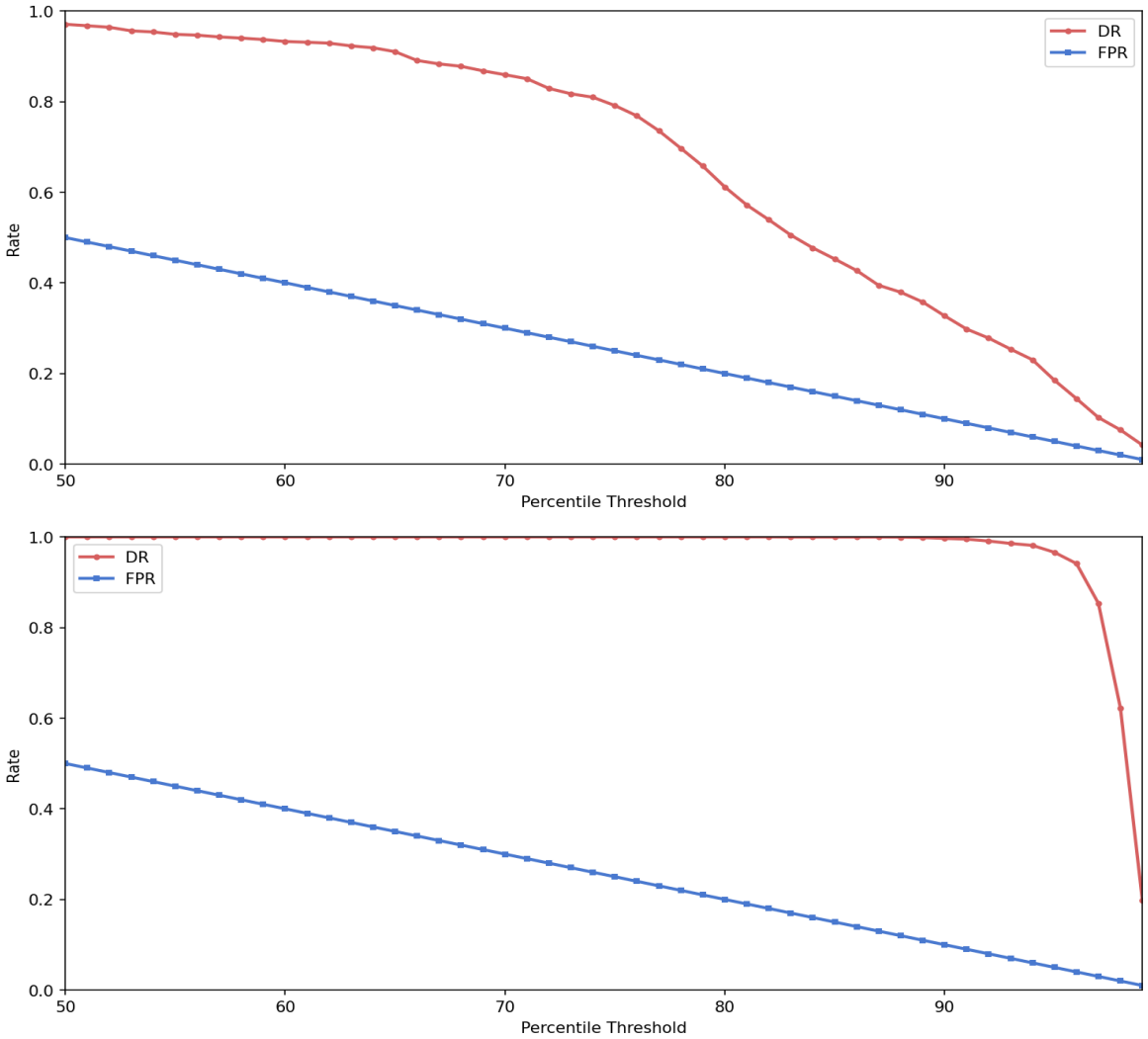

Figure 4.2: Detection Rate (DR) and False Positive Rate (FPR) across percentile thresholds for MTAD-GAT forecasting (TOP) and reconstruction (BOTTOM) scoring modes.

fore provide a moderate but consistent anomaly signal across the full threshold range.

The reconstruction-based scoring mode demonstrates substantially stronger sensitivity. The detection rate remains near-perfect across almost the entire percentile interval before collapsing only at very high thresholds. This behavior indicates a much clearer separation between Actor 1 and Actor 2 score distributions. Most anomalous windows consistently produce reconstruction errors significantly above normal operating behavior.

These observations confirm that the reconstruction objective is the dominant anomaly detection component within MTAD-GAT. The VAE-based reconstruction mechanism captures deviations in the overall behavioral distribution more effectively than forecasting alone, particularly when anomalies manifest as subtle multi-device interaction changes rather than isolated short-term temporal deviations.

However, the increased sensitivity of the reconstruction score also introduces an important trade-off. Extremely sensitive reconstruction behavior may amplify transient fluctuations and

low-intensity behavioral variations that are not necessarily operationally relevant. As a result, while reconstruction scoring achieves superior detection coverage, excessive sensitivity can also increase susceptibility to false alarms or overreaction to minor behavioral perturbations.

This trade-off explains why the combined scoring strategy provides a useful compromise between the stability of forecasting and the strong sensitivity of reconstruction. The forecasting component contributes additional robustness, while the reconstruction component preserves high anomaly separability, resulting in a more balanced operational behavior overall.

4.6 Distribution and Temporal Analysis of Anomaly Scores

4.6.1 Temporal Evolution of Anomaly Scores

Figure Figure 4.3 presents the hourly distribution of mean anomaly scores across a 24-hour cycle for both GDN and MTAD-GAT. In both figures, the **blue curves correspond to normal behavior (Actor 1)** while the **red curves represent anomalous behavior (Actor 2)**.

For both architectures, the blue (Actor 1) distributions exhibit stable low-score nighttime baselines followed by gradual increases during daytime activity periods. This reflects the inherent structure of occupant behavior, where device usage intensifies after morning activation and progressively decreases toward the evening.

The red (Actor 2) distributions reveal clearer behavioral deviations from the normal cycle. Both models consistently identify a strong transition phase starting shortly after 07:00, which marks the onset of daily activity. In this region, anomaly scores rise sharply and reach their most significant values during two key periods: the morning transition window (approximately 08:00—09:00) and the midday activity window (approximately 12:00—13:00).

GDN exhibits higher absolute anomaly magnitudes, with pronounced and sharp peaks in the red curve during these transition intervals. The most significant response occurs around 08:00—09:00, where scores form a sustained high-intensity plateau, followed by a secondary rise at midday. This indicates that GDN reacts strongly to concentrated bursts of deviation but remains relatively insensitive outside these peaks, leading to partial overlap between blue (normal) and red (anomalous) regimes in non-peak regions.

MTAD-GAT, in contrast, produces lower absolute score magnitudes but a more structured and temporally consistent separation between blue and red curves. Instead of isolating only sharp deviations, MTAD-GAT maintains elevated anomaly scores across broader temporal windows, particularly during active behavioral periods. This suggests that the model captures distributed deviations across multiple devices rather than relying solely on isolated high-error events.

A key structural difference is also observed in the shape of the anomaly response. GDN

Figure 4.3: Temporal distribution of mean anomaly scores across the 24-hour daily cycle for GDN (top) and MTAD-GAT (bottom). Blue curves correspond to normal behavior (Actor 1), while red curves represent anomalous behavior (Actor 2).

tends to produce sharper, more localized spikes in the red curve, whereas MTAD-GAT exhibits smoother and more continuous transitions between normal (blue) and anomalous (red) regimes. This aligns with the earlier findings, where MTAD-GAT demonstrated stronger performance under reconstruction-based scoring due to its ability to integrate weaker signals across time and devices.

Overall, the temporal analysis confirms that the most significant behavioral deviations are concentrated during morning and midday activity transitions. These periods correspond

to high interaction density in the smart-home environment, where even subtle changes in occupant behavior become amplified in the learned representations of both models.

4.6.2 Distribution of Anomaly Scores

Figure 4.4 compares the anomaly score distributions generated by GDN and MTAD-GAT reconstruction on the BRE dataset.

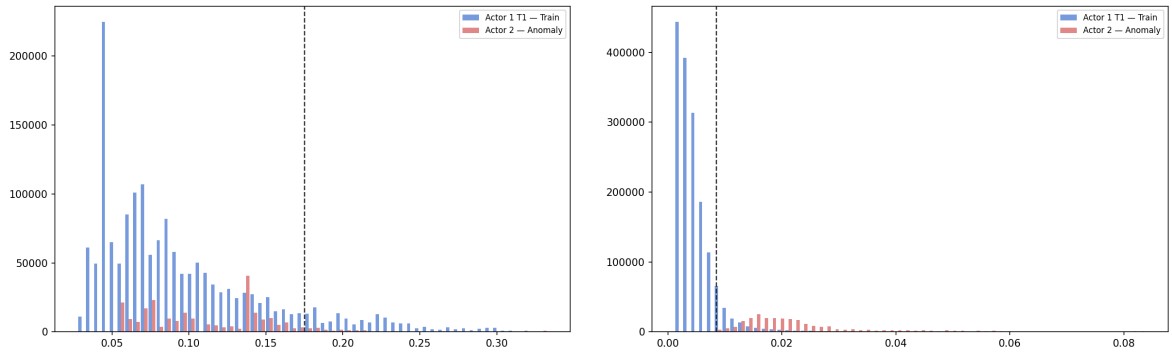

Figure 4.4: Distribution of anomaly scores for GDN (left) and MTAD-GAT reconstruction (right) on BRE.

The GDN distributions exhibit substantial overlap between Actor 1 and Actor 2 scores. Only a relatively small tail of anomalous windows extends beyond the threshold region. This overlap explains the strong threshold sensitivity and low point-wise detection rates observed earlier.

The MTAD-GAT reconstruction distributions exhibit considerably stronger separation. Although overlap remains between the two populations, anomalous windows display a clear tendency toward higher reconstruction errors. The broader displacement of the anomalous distribution produces a larger stable operating region and substantially higher detection coverage.

The comparison demonstrates that reconstruction-based scoring produces more discriminative anomaly score distributions than forecasting-only strategies under occupant behavioral anomalies.

4.7 Interpretability and Device-Level Analysis

4.7.1 Device-Level Anomaly Contributions

The system-level anomaly score is determined by the device producing the highest normalized anomaly contribution at each time-step. Analyzing the frequency with which devices dominate the anomaly score provides insight into which behavioral patterns are most informative for distinguishing Actor 2 from the normal activity profile.

To enable a direct architectural comparison, the BRE and CU datasets are analyzed separately. For each dataset, the device contribution profiles of GDN and MTAD-GAT are placed side-by-side to highlight how the two architectures distribute anomaly sensitivity across device channels.

BRE Dataset

Figure 4.5 compares the device-level contributions obtained from GDN and MTAD-GAT on the BRE dataset.

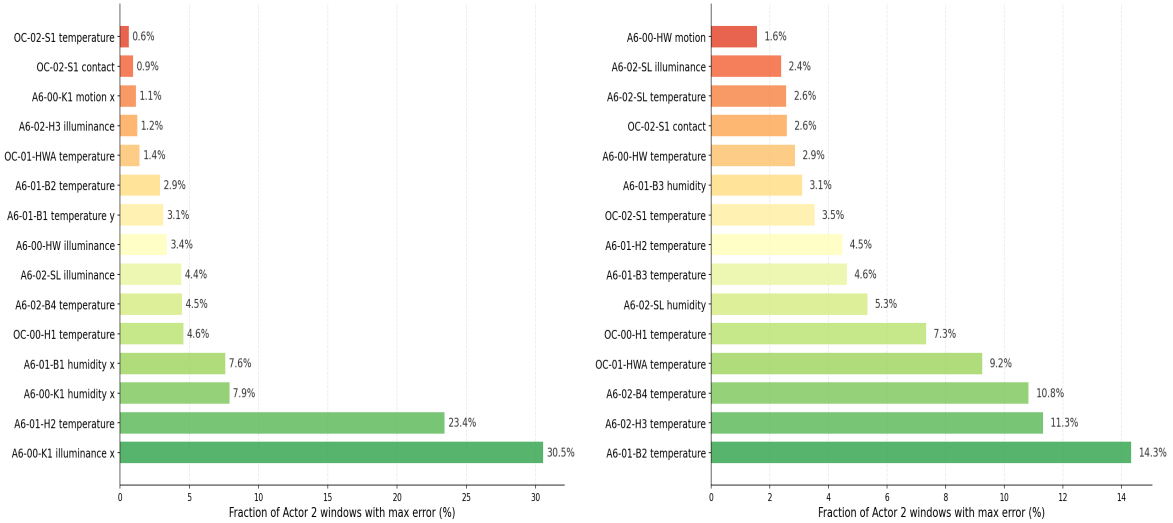

Figure 4.5: Comparison of top device anomaly contributions on the BRE dataset for GDN (left) and MTAD-GAT (right).

The two architectures exhibit substantially different contribution distributions. GDN produces a highly concentrated error profile dominated primarily by a small subset of sensors. The largest anomaly contributions originate from the illuminance and temperature sensors, while the remaining devices contribute comparatively little to the global anomaly score. This concentration indicates that GDN relies on a limited number of highly discriminative device relationships when detecting deviations from the learned behavioral routine.

In contrast, MTAD-GAT distributes anomaly contributions more evenly across multiple devices. Rather than relying on one or two dominant channels, the model captures a broader collection of moderate contributions originating from temperature, contact, and environmental sensors distributed throughout the installation. This behavior is consistent with the dense graph attention mechanism of MTAD-GAT, which allows information propagation across all pairwise device interactions.

The difference between the two distributions reflects the underlying architectural assumptions of the models. GDN's sparse graph structure focuses attention on the strongest learned dependencies, leading to localized anomaly attribution. MTAD-GAT's dense attention struc-

ture instead aggregates information from a larger set of correlated channels, producing a more spatially distributed anomaly profile.

The broader contribution distribution observed in MTAD-GAT also helps explain its substantially higher point-wise detection rates on BRE. Behavioral deviations affecting multiple correlated sensors can be captured simultaneously, whereas GDN reacts primarily when deviations occur in the limited subset of dominant devices selected by the sparse graph construction process.

CU Dataset

Figure 4.6 presents the corresponding comparison for the CU dataset.

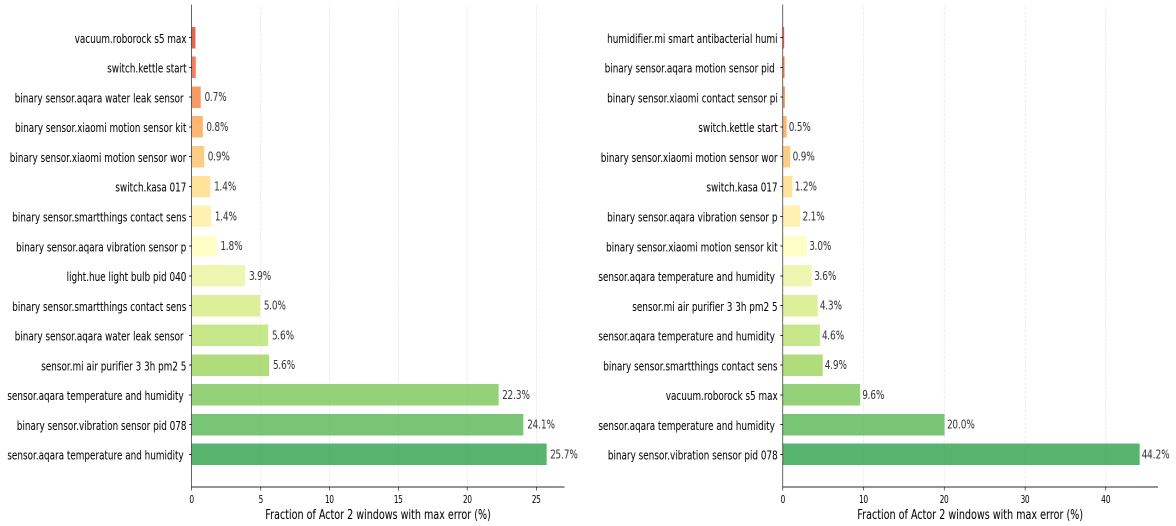

Figure 4.6: Comparison of top device anomaly contributions on the CU dataset for GDN (left) and MTAD-GAT (right).

The CU dataset produces a different contribution structure from BRE. For GDN, the anomaly score remains concentrated in a small number of devices, with temperature, humidity, vibration, and air quality sensors dominating the detection process. The rapid decline in contribution magnitude after the leading devices indicates that the model continues to rely on a highly localized anomaly representation.

MTAD-GAT on CU exhibits an even stronger concentration effect than on BRE, with a single vibration sensor accounting for a dominant portion of anomaly detections. Secondary contributions originate from temperature and robotic vacuum activity sensors, while the remaining devices contribute only marginally.

This result suggests that the behavioral anomaly in CU is strongly associated with a limited set of highly informative activity-driven devices. Unlike BRE, where anomaly evidence is distributed across many correlated channels, the CU anomaly appears to be concentrated around specific motion and interaction patterns. MTAD-GAT amplifies these highly discrim-

inative signals through its dense attention mechanism, resulting in a dominant peak associated with the vibration sensor.

Overall, the device contribution analysis demonstrates that anomaly localization differs substantially between datasets and architectures. GDN consistently produces sparse and localized attribution patterns, while MTAD-GAT captures broader cross-device relationships when such relationships exist in the data. These findings are consistent with the earlier quantitative results showing that MTAD-GAT achieves greater sensitivity to subtle behavioral deviations across diverse device environments.

4.7.2 Learned Graph Structures

The learned graph structures provide an interpretable view of how each architecture models inter-device relationships and propagates behavioral information throughout the smart-home network. Although both models rely on graph-based representations, the resulting topologies differ significantly in sparsity, connectivity profile, and information flow organization.

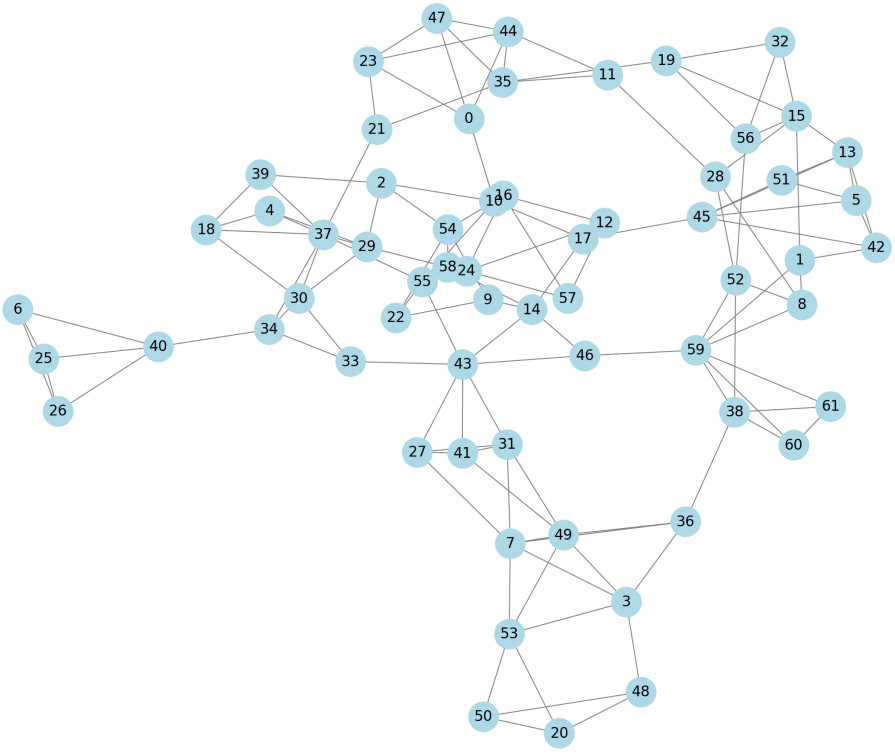

Figure 4.7: Learned sparse graph structure generated by GDN.

GDN learned graph structure. GDN constructs its graph using cosine similarity between trainable device embeddings followed by top-k sparsification. Consequently, only the strongest

inter-device relationships are preserved in the final topology.

Figure 4.7 reveals a highly decentralized mesh-like topology composed of multiple localized communities and bridge connections. Instead of relying on a single dominant hub, the graph naturally organizes into several spatially separated subgraphs linked through articulation nodes.

A clearly isolated cluster appears on the far left of the graph, where nodes 6, 25, and 26 form a compact local community connected to the remaining topology primarily through node 40. Additional localized structures emerge throughout the graph, including dense triangular and loop-based communities such as the 27—41—31 region and the 0—23—47—44—35 cluster near the upper portion of the network. Another distinct local community appears around nodes 3, 20, 48, 50, and 53 in the lower section.

Node 43 acts as a major intermediary bridge linking several otherwise separated regions. The resulting topology therefore resembles a sparse network mesh in which information propagation depends heavily on multi-hop communication between local clusters rather than direct global connectivity.

This structure directly reflects the sparsification strategy used by GDN. Since edges are retained only for highly similar embedding pairs, weaker interactions are intentionally removed. The model therefore emphasizes dominant local dependencies while suppressing broader distributed correlations.

The sparse organization provides improved interpretability because the discovered communities often correspond to coherent behavioral or spatial device groups. However, the same sparsity also limits the model's ability to capture subtle distributed anomalies spanning many heterogeneous devices simultaneously. This behavior is consistent with the earlier quantitative results, where GDN achieved low false positive rates but substantially lower detection coverage, particularly on the merged multi-installation dataset.

MTAD-GAT learned graph structure. Unlike GDN, MTAD-GAT relies on dense graph attention mechanisms that allow all device channels to interact through learned attention weights. This produces a substantially different graph organization.

Figure 4.8 reveals a highly centralized hub-and-spoke topology dominated by a dense core of high-degree hub nodes. Nodes 58, 59, 60, and 61 form the primary central region of the graph, while intermediate routing nodes such as 32 and 53 reinforce the connectivity of this dense core.

Most peripheral nodes arranged around the outer circular structure connect directly inward toward these dominant hubs. In contrast to GDN, very little localized neighbor-to-neighbor clustering is visible among peripheral regions. Instead, communication occurs primarily through the central hub structure, producing a globally interconnected and structurally shallow topology.

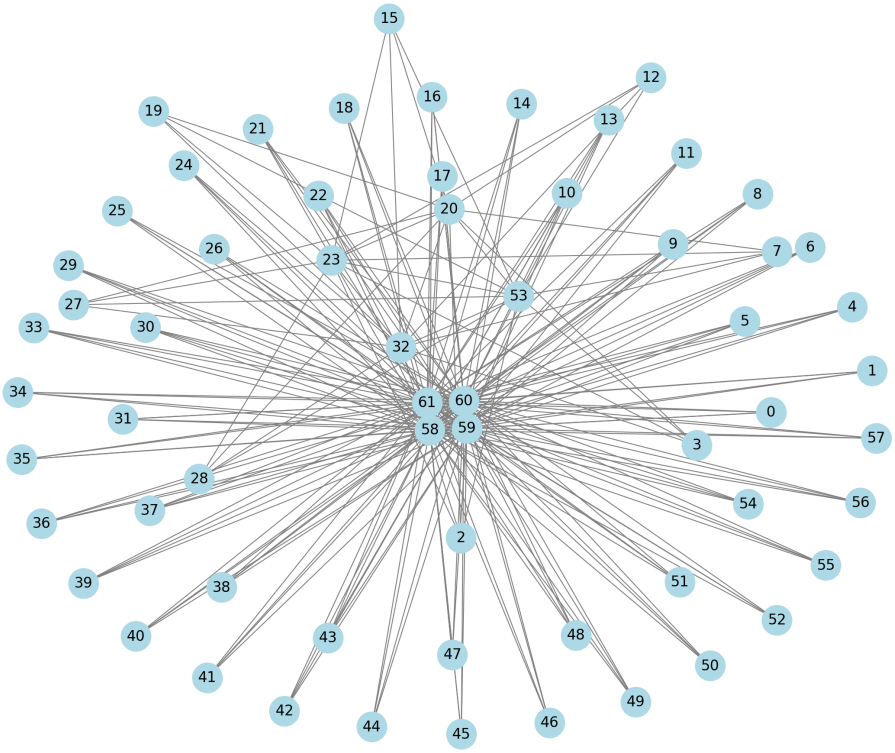

Figure 4.8: Learned dense graph structure generated by MTAD-GAT.

This dense organization reflects the architectural design of MTAD-GAT. Because the model preserves broad pairwise interactions through graph attention rather than sparse neighborhood selection, the learned representation captures distributed behavioral correlations across the entire device set simultaneously.

The centralized topology also explains the stronger robustness of MTAD-GAT observed on heterogeneous datasets. Global information aggregation through dominant hub nodes allows the model to maintain coherent behavioral representations even when device relationships become highly variable across installations.

Compared with GDN, MTAD-GAT therefore behaves as a distributed behavioral detector rather than a localized deviation detector. Instead of relying on a small subset of dominant local relationships, the architecture integrates weaker correlations across many sensors simultaneously. This broader aggregation capability contributes directly to the substantially higher detection rates achieved by MTAD-GAT throughout the experimental evaluation.

Overall, the learned graph visualizations provide a structural interpretation of the quantitative results presented earlier in this chapter. Sparse decentralized topology favors specificity and stability, whereas dense globally connected topology favors sensitivity and robustness to complex heterogeneous behavioral patterns.

4.7.3 Interpretability Discussion

The device-level analysis reveals that occupant behavioral anomalies are not uniformly distributed across all smart-home devices. Instead, a relatively small subset of activity-sensitive sensors accounts for the majority of detection events.

Devices associated with movement transitions, environmental changes, and occupancy-related activity contribute most strongly to anomaly scores. These devices encode temporal patterns that differ between occupants and therefore become primary indicators of behavioral change.

The comparison also highlights a fundamental architectural difference. GDN behaves as a localized detector focused on sparse high-amplitude deviations, whereas MTAD-GAT behaves as a distributed behavioral detector integrating weaker signals across many channels.

This distinction explains why MTAD-GAT generalizes more effectively to heterogeneous multi-installation environments. Distributed graph attention and reconstruction-based learning allow the model to capture broader behavioral structure rather than relying on a small set of dominant sensor correlations.

4.8 Summary of Findings

This chapter presented a systematic comparative evaluation of GDN and MTAD-GAT across the BRE, CU, and merged smart-home datasets under a unified experimental framework. Multiple scoring configurations were investigated in order to analyse the influence of forecasting, reconstruction, and combined anomaly scoring strategies on overall detection behaviour.

MTAD-GAT reconstruction scoring achieves the strongest overall performance. Across all datasets, the reconstruction-based scoring strategy consistently produces the highest detection capability. On the BRE dataset, MTAD-GAT reconstruction achieves a detection rate of 99.67% with zero detection delay, indicating an almost immediate and highly stable response to behavioural deviations. Strong performance is also maintained on the merged dataset with a detection rate of 93.62%, demonstrating that reconstruction learning generalises well even under highly heterogeneous multi-installation environments. On the CU dataset, reconstruction scoring again surpasses both forecasting and combined scoring modes.

Forecasting-only scoring is substantially weaker than reconstruction-based scoring. Both GDN and MTAD-GAT forecasting configurations exhibit noticeably lower detection capability across all experiments. Forecasting errors alone are less effective at capturing subtle behavioural deviations because future sensor states remain partially predictable even during anomalous periods. This effect is especially visible on the BRE dataset, where MTAD-GAT forecasting achieves only 32.73% detection compared to 99.67% for reconstruction

scoring.

GDN behaves as a conservative anomaly detector. GDN consistently produces the lowest detection rates among all evaluated configurations. Although its sparse embedding-based graph provides stable behaviour and controlled false positive levels, the model reacts primarily to isolated high-amplitude deviations and therefore fails to capture broader distributed behavioural inconsistencies. The performance degradation becomes particularly severe on the merged dataset, where GDN fails to effectively distinguish anomalous behaviour from the expanded heterogeneous normal distribution.

MTAD-GAT generalises more effectively across heterogeneous environments. Unlike GDN, MTAD-GAT maintains relatively stable performance across BRE, CU, and the merged configuration. Its dense graph attention mechanism enables the model to aggregate information from many device channels simultaneously, allowing behavioural deviations to be represented as distributed correlation changes rather than isolated local spikes. This property becomes especially important in the merged dataset containing devices from multiple independent smart-home installations.

The scoring strategy strongly influences detection behaviour. The experiments demonstrate that detection performance depends not only on the selected architecture but also on the anomaly scoring mechanism itself. Reconstruction scoring produces the highest sensitivity to occupancy and behavioural changes, while forecasting scoring yields smoother but substantially weaker anomaly separation. The combined strategy offers an intermediate compromise between both approaches by integrating predictive and reconstruction information into a unified anomaly score.

Overall, the experimental results indicate that MTAD-GAT with reconstruction-based scoring provides the most reliable and robust framework for behavioural anomaly detection in multivariate smart-home time series, particularly under heterogeneous and large-scale deployment conditions.

Conclusion

This thesis presented a systematic comparative study of Graph Neural Network (GNN)-based anomaly detection architectures for smart home multivariate time-series data. The study investigated how graph dependency structures, learning objectives, and anomaly scoring strategies influence detection behavior, robustness, threshold sensitivity, and interpretability in heterogeneous smart home environments.

To address these objectives, two representative GNN architectures were evaluated under a unified experimental framework: Graph Deviation Network (GDN), a sparse forecasting-based graph architecture, and MTAD-GAT, a dense attention-based hybrid model combining forecasting and reconstruction objectives.

Experiments were conducted using the BRE and CU smart home datasets, as well as a merged heterogeneous dataset configuration designed to evaluate cross-environment robustness. All models were compared using identical preprocessing, training, thresholding, and evaluation procedures to ensure fair architectural comparison.

Beyond conventional anomaly detection metrics, the study performed detailed analyses of threshold sensitivity and operating stability, anomaly score distributions and statistical separability, temporal evolution of anomaly scores, device-level anomaly attribution, and learned graph dependency structures.

Integrated Discussion

The comparative analysis highlights that anomaly detection performance in smart homes is strongly influenced by the choice of learning objective and graph structure design. Although both models operate within a graph neural network framework, their behavior differs significantly in terms of sensitivity, robustness, and stability of anomaly scores.

The results indicate that anomalies in smart home environments are not driven by isolated device failures or extreme sensor values, but rather by distributed changes in inter-device relationships and temporal activation patterns. This makes multivariate modeling essential for reliable detection.

Reconstruction vs forecasting insights

A key observation is the difference between forecasting-based and reconstruction-based anomaly scoring strategies. Forecasting-based approaches tend to produce more conservative detection behavior, requiring sustained deviations in temporal patterns before triggering anomaly

signals.

In contrast, reconstruction-based objectives are more sensitive to global inconsistencies in multivariate input structure, making them more responsive to subtle behavioral changes. However, this increased sensitivity may also lead to reduced specificity under normal behavioral variability.

Overall, reconstruction-based scoring proves more effective at capturing distributed behavioral deviations typical of smart home anomalies, while forecasting-based methods provide more stable but less sensitive detection behavior.

Interpretability vs performance trade-offs

The study also reveals a clear trade-off between interpretability and detection capability.

The sparse graph structure used in GDN provides clearer interpretability, as device relationships can be more directly associated with physical or functional proximity. This allows better understanding of which device interactions contribute to anomaly decisions.

However, this structured sparsity limits the ability to capture complex and heterogeneous dependencies across devices, especially when environments become more diverse.

In contrast, MTAD-GAT uses dense attention mechanisms that allow richer modeling of inter-device relationships and improve robustness across different environments. This leads to stronger detection capability but reduces transparency in terms of interpreting individual device contributions and learned dependencies.

Deployment implications in real smart homes

The results highlight important considerations for deploying anomaly detection systems in real-world smart home environments.

First, unsupervised graph-based models are effective for detecting behavioral changes without requiring labeled anomaly data, making them suitable for practical residential applications.

Second, model design choices significantly affect operational behavior. Architectures optimized for sensitivity may detect more subtle anomalies but can also increase the risk of false alarms, while more conservative models reduce false detections at the cost of missed events.

Third, robustness to heterogeneous environments is critical for real-world deployment, as smart homes differ significantly in device types, configurations, and usage patterns. Models relying on sparse dependency assumptions may struggle in such settings compared to more flexible attention-based approaches.

Direct Answers to Research Questions

- **Can graph-based models detect behavioral anomalies in smart home data without supervision?**
  Yes. Both architectures successfully learn normal behavioral patterns and identify deviations without requiring labeled anomaly data.

- **Which learning paradigm is more effective: forecasting or reconstruction?**
  Reconstruction-based approaches are generally more effective at capturing distributed multivariate deviations, while forecasting-based approaches are more conservative and less sensitive.

- **How does graph structure affect performance?**
  Sparse graph structures improve interpretability but reduce flexibility, while dense attention-based graphs improve robustness and expressive power at the cost of interpretability.

- **How do models behave under increasing environmental heterogeneity?**
  Models with dense attention and hybrid learning objectives are more robust to heterogeneous environments, while sparse graph-based models degrade when faced with cross-environment variability.

- **Which devices contribute most significantly to the anomaly score, and is this contribution consistent across models?**
  The anomaly score is driven primarily by continuous-valued environmental and activity-sensitive sensors rather than binary-state devices. Sensors measuring temperature, humidity, illuminance, air quality, and vibration contribute most strongly to anomaly detection because they capture gradual behavioural and environmental changes over time. Binary devices such as contact or on/off sensors contribute less consistently, although they may still provide useful contextual signals during specific behavioural transitions.

Limitations

This study is limited by the scope of the datasets, which represent a restricted set of smart home environments and occupancy scenarios. The anomaly definition is also limited to occupant substitution scenarios and does not include other realistic anomaly types such as device malfunction or adversarial behavior.

In addition, the evaluation is performed in an offline setting, without considering real-time adaptation or continuous learning. Finally, while interpretability was analyzed through

structural and statistical methods, deep graph-based models remain inherently difficult to fully explain in terms of causal decision mechanisms.

Future Directions

Adversarial Robustness

Future work should investigate the robustness of graph-based anomaly detection models against intentional or adversarial manipulations of sensor data, particularly in security-sensitive smart home environments.

Online Anomaly Detection

Extending these models to online and streaming settings would enable real-time detection and adaptation to evolving occupant behavior patterns while maintaining stable detection performance.

Cross-house Transfer Learning

Future research should explore transfer learning and domain adaptation techniques to improve generalization across different smart home installations and reduce the need for retraining on each new environment.

Explainable Graph Learning

Improving interpretability remains an open challenge. Future work should focus on developing explainable graph learning methods capable of identifying the precise temporal and structural factors that drive anomaly detection decisions.

References

[1] F. Wang, Y. Jiang, R. Zhang, A. Wei, J. Xie, and X. Pang, ``A survey of deep anomaly detection in multivariate time series: Taxonomy, applications, and directions," *Sensors (Basel, Switzerland)*, vol. 25, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:275218971

[2] A. Deng and B. Hooi, ``Graph neural network-based anomaly detection in multivariate time series," *arXiv preprint arXiv:2106.06947*, 2021. [Online]. Available: https://arxiv.org/abs/2106.06947

[3] H. Zhao et al., ``Multivariate time-series anomaly detection via graph attention network," in *IEEE International Conference on Data Mining (ICDM)*, 2020, pp. 841–850.

[4] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, ``An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2508–2517, 2021.

[5] E. Zivot and J. Wang, ``Vector autoregressive models for multivariate time series," in *Modeling Financial Time Series with S-PLUS*, New York, NY: Springer, 2003, pp. 369–413. DOI: 10.1007/978-0-387-21763-5_11 [Online]. Available: https://link.springer.com/chapter/10.1007/978-0-387-21763-5_11

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, ``Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00, Association for Computing Machinery, 2000, pp. 93–104. DOI: 10.1145/342009.335388 [Online]. Available: https://dl.acm.org/doi/10.1145/342009.335388

[7] K. Yang, S. Kpotufe, and N. Feamster, ``An efficient one-class svm for anomaly detection in the internet of things," *arXiv preprint arXiv:2104.11146*, 2021. [Online]. Available: https://arxiv.org/abs/2104.11146

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou, ``Isolation forest," in *IEEE International Conference on Data Mining (ICDM)*, 2008, pp. 413–422.

[9] S. Hochreiter and J. Schmidhuber, ``Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, ``Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," *Proceedings of the 24th ACM SIGKDD*, pp. 387–395, 2018.

[11] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, ``Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 753–763.

[12] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, ``Adaptive graph convolutional recurrent network for traffic forecasting," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[13] Z. Liu, X. Huang, J. Zhang, Z. Hao, L. Sun, and H. Peng, ``Multivariate time-series anomaly detection based on enhancing graph attention networks with topological analysis," *arXiv preprint arXiv:2408.13082*, 2024. [Online]. Available: https://arxiv.org/abs/2408.13082

[14] J. Xu, H. Wu, J. Wang, and M. Long, ``Anomaly transformer: Time series anomaly detection with association discrepancy," in *International Conference on Learning Representations (ICLR)*, 2022.

[15] S. Tuli, G. Casale, and N. R. Jennings, ``Tranad: Deep transformer networks for anomaly detection in multivariate time series data," *arXiv preprint arXiv:2201.07284*, 2022. [Online]. Available: https://arxiv.org/abs/2201.07284

[16] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, ``LSTM-based encoder-decoder for multi-sensor anomaly detection," in *ICML Anomaly Detection Workshop*, 2016.

[17] J. An and S. Cho, ``Variational autoencoder based anomaly detection using reconstruction probability," in *ICDM Workshop on Irregularities and Anomalies in Data*, 2015.

[18] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, ``Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD*, 2019, pp. 2828–2837.

[19] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, ``MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *International Conference on Artificial Neural Networks*, 2019, pp. 703–716.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, ``Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017. [Online]. Available: https://arxiv.org/abs/1710.10903

[21] S. Ray, S. Lakdawala, M. Goswami, and C. Gao, ``Learning graph neural networks for multivariate time series anomaly detection," *arXiv preprint arXiv:2111.08082*, 2021. [Online]. Available: https://arxiv.org/abs/2111.08082

[22] Z. L. Li, G. W. Zhang, J. Yu, and L. Y. Xu, ``Dynamic graph structure learning for multivariate time series forecasting," *Pattern Recognition*, vol. 138, p. 109 423, 2023.

[23] H. Zhao et al., ``Multivariate time-series anomaly detection via graph attention network,'' *arXiv preprint arXiv:2009.02040*, 2020. [Online]. Available: https://arxiv.org/abs/2009.02040

[24] . e. a. Tian, ``Dynamic graph learning for multivariate time series anomaly detection,'' *[Journal/Conference]*, 2023.