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: Artificial Intelligence

Presented by

Yaakoub Hadji

Entitled

Adversarial Vulnerability Assessment of GNN-Based Anomaly Detection in Smart Home Environments

Under the supervision of

Dr. Noureddine Amraoui

Jury Members

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

June, 2026

الملخص

يستكشف هذا العمل مدى متانة أنظمة كشف الشذوذ المعتمدة على الشبكات العصبية البيانية في بيئات المنازل الذكية أمام الهجمات التضليلية. فقد أصبحت الأساليب الحديثة لكشف الشذوذ تعتمد بشكل متزايد على التعلم البياني لنمذجة العلاقات الزمنية والترابطات بين أجهزة إنترنت الأشياء غير المتجانسة. وعلى الرغم من تحقيق هذه النماذج أداءً واعدًا في ظروف التشغيل العادية، فإن قدرتها على مقاومة التلاعب العدائي لا تزال غير مفهومة بشكل كافٍ. تركز هذه الدراسة على نموذجين بارزين لكشف الشذوذ المعتمد على الشبكات العصبية البيانية، وهما GDN وMTAD-GAT، باستخدام مجموعة بيانات Garage IoT الخاصة بالمنازل الذكية. وتم تطوير سلسلة من خطوات المعالجة المسبقة لتحويل البيانات متعددة المتغيرات الناتجة عن الحساسات إلى بيانات زمنية ذات بنية بيانية مناسبة لعمليات كشف الشذوذ. كما تم تقييم متانة النموذجين في عدة سيناريوهات هجومية تمثل مستويات مختلفة من معرفة المهاجم، تشمل الصندوق الأسود والصندوق الرمادي والصندوق الأبيض. وشملت التجارب استراتيجيات هجوم متعددة، مثل الهجمات العشوائية، والهجمات المعتمدة على الاستعلام، والهجمات الموجهة بالتفسير، والهجمات المعتمدة على التدرجات، وذلك تحت حدود مختلفة لشدة الاضطراب المطبق على البيانات.

أظهرت النتائج التجريبية أن الاضطرابات التضليلية قادرة على إحداث تدهور ملحوظ في أداء كشف الشذوذ، حتى عندما تكون التعديلات المدخلة على قراءات الحساسات محدودة. كما كشفت النتائج عن اختلافات في مستوى المتانة بين نموذجي GDN وMTAD-GAT، مما يبرز تأثير البنية المعمارية للنموذج وآليات تعلم العلاقات البيانية على درجة القابلية للتأثر بالهجمات. وبشكل عام، يزداد تدهور الأداء مع زيادة حجم الاضطراب، إلا أن فعالية الهجمات تختلف باختلاف نوع الهجوم ومستوى معرفة المهاجم.

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

Abstract

This thesis investigates the adversarial robustness of Graph Neural Network (GNN)-based anomaly detection systems in smart-home environments. Recent anomaly detection approaches increasingly leverage graph-based learning to model temporal and relational dependencies between heterogeneous Internet of Things (IoT) devices. While such models achieve promising detection performance under normal operating conditions, their resilience against adversarial manipulation remains insufficiently understood. The study focuses on two representative GNN-based anomaly detection models, Graph Deviation Network (GDN) and MTAD-GAT, evaluated using the IoT Garage smart-home dataset. A preprocessing pipeline was developed to transform multivariate sensor streams into graph-structured temporal data suitable for anomaly detection. The robustness of both models was assessed under multiple adversarial scenarios corresponding to black-box, grey-box, and white-box threat settings.

Several perturbation strategies, including random, query-based, explainability-guided, and gradient-based attacks, were applied under different perturbation budgets.

Experimental results demonstrate that adversarial perturbations can significantly degrade anomaly detection performance, even when modifications to sensor measurements remain limited. The findings further reveal differences in robustness between GDN and MTAD-GAT, highlighting the influence of architectural design and graph-based dependency modeling on adversarial vulnerability. Performance degradation generally increases with perturbation magnitude, although attack effectiveness varies across threat models and attack techniques.

Keywords: Graph Neural Networks, Adversarial Attacks, Smart Home Environments, Anomaly Detection, Multivariate Timeseries

Dedication

To my family and loved ones, for their encouragement, patience, and belief in me throughout this journey.

Yaakoub

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Noureddine Amraoui, for his continuous guidance, valuable advice, constructive feedback, and unwavering support throughout the development of this thesis. His expertise, encouragement, and dedication have greatly contributed to the completion of this work.

I would also like to extend my thanks to the members of the jury for accepting to evaluate this thesis and for the time and effort devoted to reviewing it.

My appreciation goes to the faculty members and staff of the Department of Computer Science for providing a stimulating academic environment and the knowledge that has shaped my education.

Finally, I am deeply grateful to my family and loved ones for their constant support, patience, encouragement, and belief in me throughout my studies. Their trust and motivation have been a source of strength during this journey.

Contents

List of Tables

List of Figures

List of Symbols

α	Per-step size in iterative attacks (PGD, BETA, NES)
$\alpha_{u,j}$	MTAD-GAT graph attention weight from node j to target node u
$\bar{\mathcal{V}}$	Set of influencer nodes selected by BETA
$\hat{\mathbf{g}}_k$	NES pseudo-gradient estimate at iteration k
$\hat{y}_t^{(i)}$	Predicted value for device i at time t
λ	Detection threshold (anomaly score decision boundary)
$\mathbf{A}$	Adjacency matrix
$\mathbf{e}_i$	Node embedding vector
$\mathbf{M}$	Binary perturbation mask restricting BETA updates to influencer nodes
$\mathbf{q}$	Axis-aligned candidate direction vector in SimBA
$\mathbf{u}_s$	Random unit-normal direction sampled in NES at pair s
$\mathbf{X}$	Full multivariate time series
$\mathbf{X}^{(t)}$	Sliding window ending at time t
$\mathbf{x}_t$	Observation vector at time t
$\mathbf{y}_t$	Target vector at time t
$\mathcal{E}$	Set of edges
$\mathcal{G}$	Graph of device interactions
$\mathcal{L}_{\text{adv}}^{(u)}$	Node-specific adversarial evasion loss at target node u (BETA)
$\mathcal{L}_{\text{adv}}$	Adversarial evasion loss (mean absolute forecast error over all nodes)
$\mathcal{N}(i)$	Neighborhood of node i
$\mathcal{V}$	Set of nodes
ASR	Attack Success Rate: absolute reduction in detection rate under attack

$\mathrm{DR}_{\mathrm{adv}}$ Adversarial detection rate (detection rate under attack)

$\mathrm{DR}_{\mathrm{clean}}$ Clean detection rate (detection rate with no attack)

$\odot$ Element-wise (Hadamard) multiplication

ω Smoothing window size

Π_{ε} Projection operator onto the ℓ_∞ ball of radius ε

σ NES smoothing radius

τ_{90} 90th-percentile threshold

τ_{99} 99th-percentile threshold

$\tilde{\mathbf{X}}^{(t)}$ Adversarially perturbed sliding window at time t

$\tilde{x}_t^{(i)}$ Reconstructed value for device i at time t

ε ℓ_∞ perturbation budget (maximum absolute perturbation per feature)

$\xi_{i,t}^{\mathrm{recon}}$ Per-device reconstruction error at node i, time t

$\xi_{i,t}$ Per-device forecast error at node i, time t

$A^{(t)}$ Aggregated anomaly score at time t

$A_s^{(t)}$ Smoothed anomaly score at time t

A_{actor2} Anomalous test score series (Actor 2 partition)

A_{train} Training anomaly score series

B Batch size (training) / influencer budget (BETA attack) --- context-dependent

d Embedding dimension

f Trained model

$h_i^{(l)}$ Node embedding at layer l

K Number of iterations in iterative attacks

k Number of nearest neighbors

N Number of device channels (nodes)

Q SimBA direction budget (maximum number of candidate directions tried)

S	Number of antithetic pairs per NES gradient estimation step
T	Total number of time steps
u	Target node (sensor) whose anomaly score BETA seeks to suppress
W	Sliding window size
$x_t^{(i)}$	Value of device i at time t

Introduction

Cyber-Physical Systems (CPS) increasingly rely on interconnected sensing and automation infrastructures to monitor physical environments, support autonomous decision-making, and ensure operational safety. Smart homes represent a prominent instance of such environments, integrating heterogeneous Internet of Things (IoT) devices including motion detectors, smart plugs, environmental sensors, lighting systems, contact sensors, and connected appliances into a continuously evolving digital ecosystem. These devices collectively generate large volumes of multivariate time-series data that reflect both user behavior and environmental dynamics.

As smart-home deployments become more complex and densely interconnected, manual monitoring of device behavior becomes impractical. Consequently, anomaly detection systems have emerged as a critical component for identifying abnormal activities, device malfunctions, cyber intrusions, and unexpected behavioral deviations. Traditional anomaly detection techniques based on statistical modeling or rule-based systems often struggle to capture the nonlinear temporal dependencies and cross-device interactions that characterize modern smart-home environments [8] [9]. In response to these limitations, deep learning approaches have become increasingly popular for modeling high-dimensional sequential data.

Among recent advances, Graph Neural Networks (GNNs) have attracted significant attention for multivariate time-series anomaly detection. Unlike conventional sequential models that treat each sensor independently, GNN-based approaches explicitly model inter-device relationships through graph representations, where sensors correspond to graph nodes and dependencies between devices are represented as edges. Architectures such as Graph Deviation Network (GDN) [1] and MTAD-GAT [2] combine temporal modeling with graph-based relational learning to capture complex interactions between heterogeneous devices. These models have demonstrated strong anomaly detection performance in cyber-physical environments by jointly exploiting temporal dynamics and structural dependencies.

Problem

Despite these advances, the growing integration of machine learning models into safety- and security-critical systems has raised important concerns regarding adversarial robustness. Adversarial Machine Learning (AML) studies how carefully crafted perturbations applied to model inputs can manipulate predictions and degrade performance. Prior research has shown that deep learning systems can remain highly vulnerable even when perturbations are small and difficult to distinguish from normal observations [15] [13]. In anomaly detection settings,

such perturbations may allow malicious activities or abnormal behaviors to evade detection entirely.

The adversarial vulnerability problem becomes particularly critical in graph-based anomaly detection systems. Since GNN-based models rely not only on temporal information but also on learned inter-device dependencies, perturbations affecting a small subset of devices may propagate through the graph structure and influence the anomaly scores of neighboring sensors [3]. This creates additional attack surfaces that are absent in conventional time-series models. While adversarial robustness has been widely studied in image classification and, more recently, in graph learning tasks such as node classification, relatively limited attention has been devoted to adversarial attacks against GNN-based anomaly detection systems operating on multivariate sensor streams.

Motivation

Existing GNN-based anomaly detection approaches in multivatiate time series present two main limitations. First, many graph-based anomaly detection models are evaluated exclusively under clean-data assumptions, where robustness against malicious perturbations is not considered as part of the evaluation process [1] [2]. Second, empirical robustness evaluations in realistic smart-home environments remain relatively scarce, especially for comparative studies involving multiple attack scenarios and multiple graph-based architectures [4] [7].

This thesis addresses these limitations through a systematic empirical study of adversarial vulnerability in GNN-based anomaly detection for smart-home multivariate time-series data. Rather than proposing a new anomaly detection model or a new adversarial attack method, the objective of this work is to evaluate how representative graph-based anomaly detectors behave under different adversarial conditions and perturbation strategies. The study focuses on two widely used architectures, namely GDN and MTAD-GAT, and investigates how their architectural characteristics influence robustness against adversarial manipulation.

The experimental framework is based on the IoT Garage dataset [19], a real-world smart-home dataset containing heterogeneous environmental and smart-device sensor streams collected over several weeks. The anomaly detection problem is formulated as an unsupervised behavioral deviation detection task in which normal behavior is learned exclusively from the primary resident (Actor 1), while the activities of a secondary actor (Actor 2) are treated as anomalous behavioral deviations. To support graph-based learning, the multivariate sensor streams are represented as learned dependency graphs processed through sliding temporal windows.

The adversarial evaluation framework considers multiple attacker knowledge levels corresponding to black-box, grey-box, and white-box settings. Several perturbation strategies are evaluated, including random-noise perturbations, query-based attacks, explainability-guided

attacks, and gradient-based attacks such as FGSM and PGD. All perturbations are constrained within bounded perturbation budgets in order to preserve realistic modifications to the sensor streams. The evaluation analyzes how perturbation magnitude, attack type, and model architecture affect anomaly detection reliability.

Objectives

The main objective of this thesis is therefore to assess the adversarial robustness of graph-based anomaly detection systems in smart-home environments and to better understand the vulnerabilities introduced by graph-based relational learning. More specifically, the study aims to:

- evaluate the robustness of representative GNN-based anomaly detectors under multiple adversarial attack scenarios;
- compare the vulnerability profiles of different GNN architectures under different attacker knowledge assumptions;
- analyze the effect of perturbation magnitude on anomaly detection performance;
- investigate how graph-based relational dependencies influence adversarial sensitivity;
- identify practical robustness limitations that should be considered before deploying graph-based anomaly detectors in cyber-physical environments.

Research Questions

To guide the empirical study, the thesis is organized around the following research questions:

- **RQ1:** How vulnerable are GNN-based anomaly detection models to adversarial perturbations in smart-home multivariate time-series data?
- **RQ2:** Which adversarial attack strategies are most effective in degrading anomaly detection performance?
- **RQ3:** Do different graph-based architectures exhibit different robustness characteristics under adversarial conditions?
- **RQ4:** To what extent do learned inter-device dependencies contribute to adversarial vulnerability?
- **RQ5:** How does perturbation magnitude influence the degradation of anomaly detection reliability?

Contributions

The contributions of this thesis can be summarized as follows:

- it provides a focused empirical robustness evaluation of graph-based anomaly detection models in smart-home environments;
- i establishes a comparative adversarial evaluation framework for two distinct GNN-based models, namely GDN and MTAD-GAT, across multiple attacker knowledge settings.
- it analyzes how graph-based relational modeling influences adversarial sensitivity in multivariate time-series anomaly detection;
- it highlights practical robustness limitations that should be considered when deploying GNN-based anomaly detectors in cyber-physical systems.

Thesis Organization

The remainder of this thesis is organized as follows. Chapter 1 reviews the literature on multivariate time-series anomaly detection, Graph Neural Networks, and adversarial machine learning. Chapter 2 formalizes the problem setting and defines the adversarial threat model adopted throughout the study. Chapter 3 presents the dataset characteristics and preprocessing pipeline. Chapter 4 introduces the baseline models and evaluates their clean detection performance. Chapter 5 presents the adversarial attack framework and analyzes the experimental robustness results. Finally, the thesis concludes with a summary of the main findings, limitations, and future research directions.

Chapter 1: Related Work

1.1 Overview

This chapter reviews the literature most relevant to this thesis at the intersection of anomaly detection, multivariate time-series modeling, graph neural networks, and adversarial machine learning. The goal is to situate the present work within the existing state of the art and to identify the specific gap addressed by the empirical study carried out in the following chapters.

The review is organized around five themes. First, it summarizes the evolution of anomaly detection methods for multivariate time series. Second, it discusses graph-based and graph-neural approaches designed to capture inter-device dependencies explicitly. Third, it introduces the main concepts of adversarial machine learning that are relevant to anomaly detection. Fourth, it reviews studies on adversarial attacks and robustness in time-series and graph-based settings. Finally, it synthesizes the main research gaps and clarifies the positioning of this thesis.

1.2 Anomaly Detection in Multivariate Time Series

Multivariate time-series anomaly detection aims to identify observations or temporal events that deviate from the normal behavior of a system monitored by multiple sensors or features. Unlike univariate anomaly detection, the multivariate setting must account not only for the temporal dynamics of each signal, but also for the dependencies among signals. This is especially important in cyber-physical systems, where an anomaly may not correspond to an extreme value in one sensor, but rather to an inconsistent relationship between several sensors [1], [2]. Table 1.1 summarizes the main families of anomaly detection approaches discussed in the following sections.

Table 1.1: Summary of AD-MTS Method Families

Family	**Main Idea**	**Examples**	**Typical Applications**
Statistical	Use statistical assumptions to detect abnormal behavior	ARIMA, PCA	Finance, quality control, environmental monitoring
Deep Learning	Learn temporal and latent patterns from data	LSTM, AE/VAE, GAN, TranAD	IoT systems, healthcare monitoring, predictive maintenance
GNN-Based	Model sensor dependencies as graphs	GDN, MTAD-GAT, TopoGDN	Smart grids, traffic networks, industrial CPS

1.2.1 Statistical Methods

Early anomaly detection approaches in multivariate time series relied on statistical assumptions about normal behavior, such as linear correlation, stationarity, or distance from an estimated distribution. Reviews embedded in recent graph-based anomaly detection papers identify autoregressive models, ARIMA-like forecasting methods, PCA-style subspace modeling, and distance-based techniques among the most common classical baselines [8], [9]. These methods are often attractive because they are interpretable and computationally lightweight.

However, their usefulness decreases when the monitored system exhibits strong nonlinearity, heterogeneity across sensors, or rapidly changing operating conditions. In real sensor environments, relationships between variables may be dynamic and context-dependent. As a result, classical methods often fail to represent the complex interaction patterns that characterize anomalies in high-dimensional cyber-physical data [1].

1.2.2 Deep Learning Methods

Deep learning-based anomaly detection emerged as a response to those limitations. Forecasting-based approaches learn future sensor values from historical windows and use prediction errors as anomaly scores. Reconstruction-based approaches instead compress and reconstruct the observed sequence, flagging samples with high reconstruction error as abnormal. Hybrid approaches combine both ideas to exploit local predictive accuracy and global representation quality [2], [4].

Several model families have become common in this area. Recurrent models such as LSTM-based detectors capture sequential dependencies; autoencoders and variational autoencoders focus on latent representation learning; GAN-based approaches attempt to model normal data distributions more flexibly; and more recent transformer-based models, such as TranAD, improve temporal modeling through attention mechanisms and self-conditioning [10]. These methods substantially improved performance on benchmark datasets.

1.3 GNN-Based Anomaly Detection in Multivariate Time Series

1.3.1 Graph-Based Time Series Anomaly Detection

To address the limitations of traditional time series models, recent approaches incorporate graph neural networks (GNNs) to model dependencies among variables. In the context of multivariate time series, each sensor or device can be viewed as a node, while dependencies between sensors are represented as edges as illustrated in figure 1.1. This representation

allows anomaly detection models to reason about both temporal behavior and inter-sensor structure [1], [2].

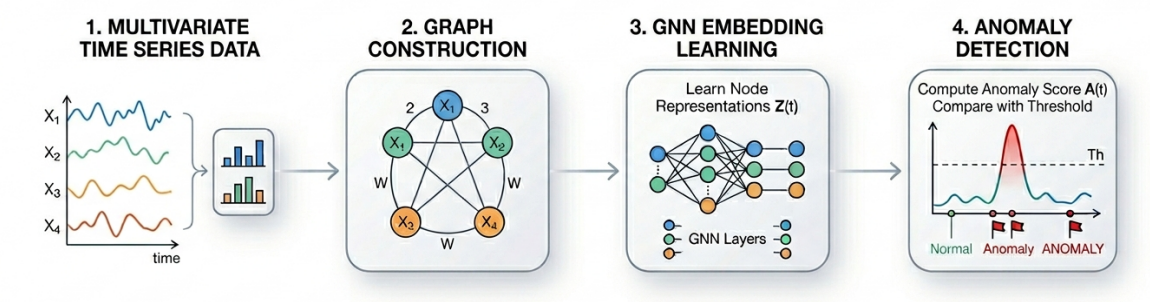

Figure 1.1: Generic Pipeline of GNN-based Anomaly Detection

1.3.2 Representative Architectures

Among the most influential models in this line of work is the Graph Deviation Network (GDN) [1]. GDN learns a sparse directed graph of sensor dependencies through trainable sensor embeddings, then uses graph attention to forecast future values and compute anomaly scores from normalized prediction errors. A key strength of GDN is that it does not require a manually defined graph and can provide some degree of explainability through learned relations and attention weights.

MTAD-GAT extends the graph-based perspective by combining feature-wise and time-wise graph attention with a recurrent temporal module and a hybrid anomaly score built from forecasting and reconstruction signals [2]. In contrast to GDN, MTAD-GAT assumes a denser relational structure and jointly models temporal and feature dependencies, which may offer richer representations but also introduces additional complexity.

Subsequent work refined these ideas in different ways. GLUE further explored graph-based multivariate anomaly detection with an emphasis on improving the quality of learned graph representations [11]. TopoGDN extends the original GDN framework by incorporating topological analysis and multi-scale temporal modeling, aiming to improve the representation of higher-order sensor interactions [12].

1.3.3 Limitations of Existing GNN-Based Approaches

Despite their advantages, existing graph-based anomaly detectors have several limitations relevant to this thesis. First, most models are designed and benchmarked under clean-data assumptions, with robustness considered only indirectly, if at all. Second, graph structure learning can itself create an attack surface: if sensor embeddings, attention coefficients, or

inter-node dependencies are manipulated, the resulting anomaly scores may become unreliable. Third, many methods are optimized primarily for clean accuracy, interpretability, or scalability, rather than for resistance to adversarial perturbations. Consequently, strong clean performance does not necessarily imply dependable behavior under malicious input manipulation [7].

These approaches have demonstrated strong performance on multivariate time-series anomaly detection tasks, particularly in smart-home and industrial monitoring environments. Nevertheless, the use of sliding windows introduces additional sensitivity to input perturbations, as predictions depend on short temporal contexts. This characteristic potentially increases the vulnerability of such models to adversarial manipulation.

1.4 Adversarial Machine Learning

Adversarial Machine Learning (AML) studies the robustness of machine learning models in the presence of maliciously crafted inputs. Unlike traditional machine learning settings where data is assumed to be independently and identically distributed (i.i.d.), AML considers adversaries that intentionally manipulate inputs to degrade model performance or induce incorrect predictions [13].

1.4.1 Definition and Motivation

Given a machine learning model $f : \mathcal{X} \to \mathcal{Y}$, an adversarial attack aims to construct an input $x' \in \mathcal{X}$ such that:

$$x' = x + \delta, \tag{1.1}$$

where δ is a carefully designed perturbation, satisfying:

$$f(x') \neq f(x), \quad \text{while} \quad \|\delta\| \leq \epsilon. \tag{1.2}$$

The key characteristic of adversarial examples is that the perturbation δ is typically small and often imperceptible to humans, yet it can significantly alter the model's prediction.

Adversarial vulnerabilities arise due to the high dimensionality of input spaces and the sensitivity of learned decision boundaries. These vulnerabilities pose serious risks in security-critical applications such as autonomous systems, healthcare, and smart environments.

Attack Stage: Evasion vs Poisoning

Evasion Attacks occur at inference time, where the adversary modifies input data to evade detection or mislead the model. These are the most common type of attacks and are particularly relevant in anomaly detection systems.

Poisoning Attacks occur during the training phase, where the adversary injects malicious samples into the training dataset to corrupt the learned model. This can lead to degraded performance or targeted backdoor behaviors.

Adversary Knowledge: White-box vs Black-box

White-box Attacks assume full access to the model, including its architecture, parameters, and gradients. This allows attackers to directly optimize perturbations using gradient-based methods.

Black-box Attacks assume no knowledge of the internal model. The attacker can only query the model and observe outputs. Such attacks often rely on transferability or query-based optimization techniques.

Attack Objectives: Targeted vs Untargeted

Untargeted Attacks aim to cause any incorrect prediction:

$$f(x') \neq y. \tag{1.3}$$

Targeted Attacks aim to force the model to produce a specific incorrect label y_t:

$$f(x') = y_t, \quad y_t \neq y. \tag{1.4}$$

1.4.2 Common Adversarial Attack Methods

Several methods have been proposed to generate adversarial examples, particularly in continuous domains such as images and time series.

Fast Gradient Sign Method (FGSM)

FGSM [13] is a single-step gradient-based attack that computes perturbations as:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f(x), y)), \tag{1.5}$$

where $\mathcal{L}$ is the loss function. FGSM is efficient and widely used as a baseline attack.

Projected Gradient Descent (PGD)

PGD [14] is an iterative extension of FGSM:

$$x^{t+1} = \Pi_{\mathcal{B}_\epsilon(x)} \left(x^t + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(f(x^t), y)) \right), \tag{1.6}$$

where Π denotes projection onto an ϵ-ball around the original input. PGD is considered one of the strongest first-order attacks.

1.4.3 Illustrative Examples of Adversarial Attacks

Image Classification

In image classification, small perturbations added to pixel values can cause a model to misclassify an image with high confidence, even though the image appears unchanged to human observers. This demonstrates the discrepancy between human perception and model decision boundaries [15].

Time Series Data

In time series applications, adversarial perturbations can be designed to preserve the overall shape of the signal while altering model predictions. For example, slight modifications to sensor readings in a smart home environment can hide anomalous behavior from detection systems [4].

Graph-Structured Data

In graph-based models, adversarial attacks can modify node features or graph structure (e.g., adding or removing edges) to influence message passing and alter predictions. These attacks are particularly relevant for graph neural networks used in relational and networked data [7].

1.4.4 Challenges in Adversarial Machine Learning

Adversarial machine learning presents several important challenges. One major issue is the high dimensionality of input spaces, where even small perturbations can accumulate across many features and significantly affect model predictions. Another challenge is the lack of robustness guarantees, as most deep learning models do not provide formal protection against adversarial perturbations. In addition, adversarial examples often exhibit transferability, meaning that attacks generated for one model may also succeed against other models, enabling effective black-box attacks. Finally, real-world adversarial settings introduce practical constraints, since perturbations must remain realistic and satisfy domain-specific limitations while still being capable of misleading the target model.

1.5 Adversarial Machine Learning on Graphs and Time Series

In graph-based and time-series settings, adversarial perturbations may affect input features, temporal values, learned dependencies, graph structure, or decision thresholds. Surveys of adversarial learning on graph data emphasize that the graph domain introduces unique challenges compared with images because the data are structured, often discrete, and governed by relational constraints [3]. Similarly, recent studies on anomaly detection stress that temporal continuity and physical plausibility matter when adversarial perturbations are applied to sensor streams [4].

Adversarial robustness of machine learning models has been extensively studied in recent years, particularly in domains such as computer vision and natural language processing [13], [14]. However, its investigation in the context of graph-based time series anomaly detection remains limited.

As summarized in Table 1.2, in this section, we review the literature along three main directions: (i) adversarial attacks on deep learning models, including graph neural networks, (ii) adversarial attacks on time series and anomaly detection models, and (iii) graph-based time series anomaly detection methods.

Table 1.2: Summary of Adversarial Attacks on Time-Series and Graph-Based Models

Family	**Adversarial Attack**	**Application / Target Domain**
Deep Learning Models	FGSM, PGD, optimization-based attacks	Image classification and general deep learning systems
Time-Series Models	Gradient-based temporal perturbations, evasion attacks	Time-series classification, IoT systems, cyber-physical systems
Graph-Based Time-Series Models	Feature perturbation, edge manipulation, BETA attack	Sensor networks, industrial monitoring, GNN-based anomaly detection

1.5.1 Adversarial Attacks on Deep Learning Models

Adversarial attacks were first extensively studied in the context of image classification, where imperceptible perturbations can lead to incorrect predictions. Methods such as the Fast Gradient Sign Method (FGSM) [13] and Projected Gradient Descent (PGD) [14] exploit gradi-

ents of the loss function with respect to the input to generate adversarial examples. These approaches highlight the vulnerability of deep neural networks to carefully crafted perturbations.

Subsequent research explored adversarial robustness across a wider range of deep learning models and application domains. Reinforcement learning-based approaches [16] and other optimization-driven attack strategies demonstrated that adversarial perturbations can effectively manipulate model predictions under different settings and constraints. These studies further emphasized the susceptibility of deep learning systems to both evasion and poisoning attacks.

Despite their effectiveness, many existing adversarial attack methods were designed for static data representations and fixed input structures, which limits their direct applicability to temporal or dynamically evolving systems.

1.5.2 Adversarial Attacks on Time Series and Anomaly Detection

Adversarial attacks on time series data have gained attention more recently. Existing works demonstrate that deep learning models for time series classification are highly vulnerable to small, smooth perturbations that preserve the overall shape of the signal while significantly altering model predictions [17]. These attacks typically adapt gradient-based methods such as FGSM and PGD to sequential data.

In the context of anomaly detection, recent studies have shown that models based on reconstruction or prediction errors can be easily deceived. By introducing carefully designed perturbations, attackers can reduce the discrepancy between observed and predicted values, thereby hiding anomalous behavior [2]. This is particularly critical in cyber-physical systems and IoT environments, where anomaly detection plays a key role in ensuring system security.

However, these works largely focus on univariate or multivariate time series without explicitly modeling inter-variable dependencies using graph structures. As a result, they do not capture the relational inductive biases leveraged by graph-based models.

1.5.3 Adversarial Attacks on Graph-Based Time Series Models

The broader adversarial GNN literature provides useful mechanisms and threat models. Survey work has catalogued a wide variety of graph attacks and defenses, including feature perturbation, edge manipulation, node injection, and transfer-based black-box attacks [3]. Figure 1.2 summarizes different attack surfaces. Practical attack studies have further shown that GNNs remain vulnerable even when attackers have limited node access and incomplete model knowledge [5]. Robustness studies at scale confirm that these vulnerabilities are not restricted to toy graphs or small benchmarks [6], [18].

For anomaly detection in sensor networks, the most directly relevant recent work is BETA,

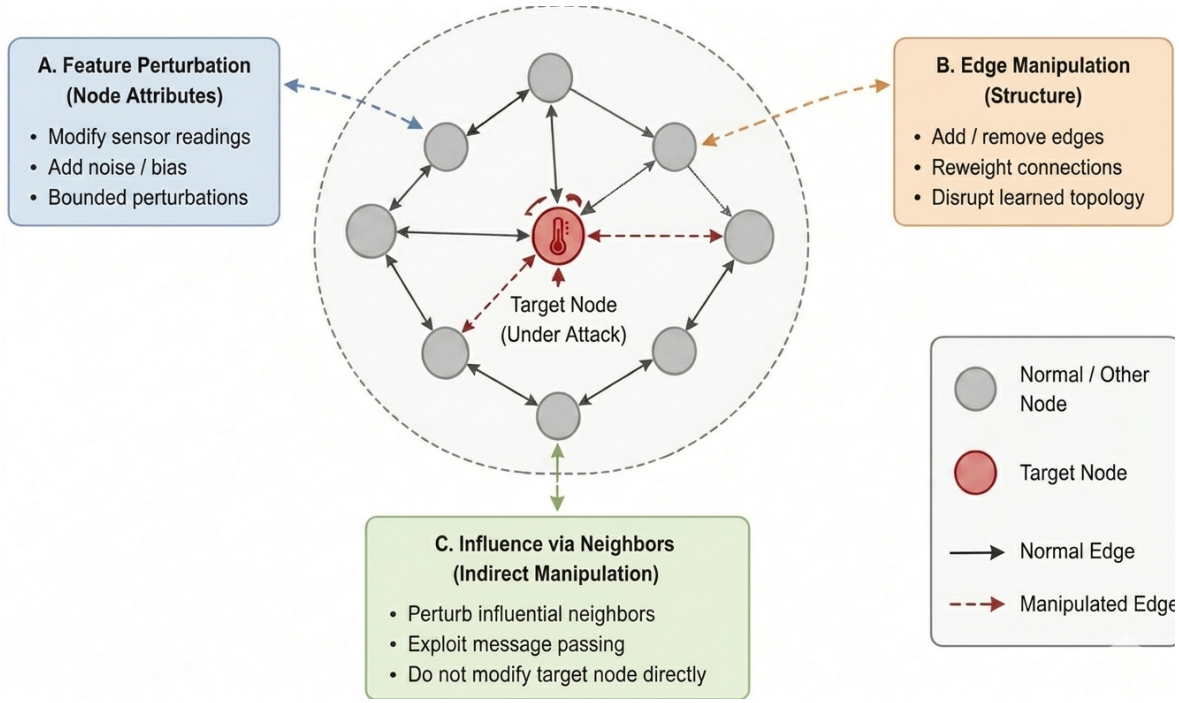

Figure 1.2: Attack Surfaces in GNN-based Anomaly Detection

a budgeted explainability-guided targeted attack against GNN-based anomaly detectors [7]. BETA is particularly important because it moves beyond generic graph attacks and focuses on realistic constraints: a safeguarded target node, a fixed attack budget, and perturbations applied to influential neighboring sensors. Its results show that state-of-the-art detectors such as GDN and TopoGDN can be degraded substantially even when the attacker does not perturb the target sensor directly.

Taken together, these studies suggest that graph-based anomaly detectors inherit vulnerabilities from both temporal forecasting models and graph representation learning systems. They are therefore exposed through multiple pathways: temporal perturbation, feature manipulation, and exploitation of learned inter-sensor dependencies.

1.6 Research Gaps and Positioning of This Work

Despite the rapid progress in multivariate time-series anomaly detection and the increasing adoption of graph neural networks (GNNs) for modeling inter-sensor dependencies, the literature reveals several critical limitations when these methods are considered in adversarial settings.

First, although many anomaly detection models report strong performance under clean conditions, robustness is rarely treated as a first-class evaluation criterion. Most existing works optimize for detection accuracy on benign datasets, implicitly assuming that training

and test data are free from malicious manipulation. As a result, reported performance metrics provide limited insight into how these models behave under adversarial perturbations, particularly in safety-critical environments such as smart homes.

Second, there is a lack of evaluation grounded in smart-home-specific contexts, where heterogeneous devices, localized interactions, and resource constraints introduce unique challenges. Many existing benchmarks are generic or industrial datasets that do not fully capture the characteristics of residential sensor networks. As a result, the applicability of prior findings to smart home environments remains uncertain.

These gaps collectively highlight the need for a systematic empirical assessment of adversarial vulnerability in GNN-based anomaly detection, focusing on comparative evaluation across models, realistic perturbation scenarios, and the interplay between temporal and structural components. Addressing these limitations is essential for understanding the reliability of current approaches and for guiding the development of more trustworthy anomaly detection systems in smart environments.

This thesis presents a comparative empirical study of adversarial vulnerability in GNN-based anomaly detection for smart-home multivariate time series. It does not propose a new model or attack method. Instead, it uses existing GNN models and adversarial techniques to examine how baseline models respond to different types of perturbations, which vary in intent, magnitude, and temporal realism. By applying these methods to smart home data, the study helps fill gaps in the literature and contributes to research on building more trustworthy AI systems for critical applications.

1.7 Summary

This chapter reviewed the state of the art in multivariate time-series anomaly detection, graph-based modeling, and adversarial machine learning. It showed that graph neural networks have become important tools for capturing inter-sensor dependencies, but that their robustness under adversarial conditions remains only partially understood. The literature also indicates that anomaly detectors can degrade sharply under small perturbations, while recent sensor-network attack studies confirm that graph-based detectors are not exempt from this risk.

These observations motivate the remainder of the thesis. The next chapter formalizes the problem setting and threat model used to evaluate the adversarial vulnerability of GNN-based anomaly detection in smart home environments.

Chapter 2: Problem Formulation and Threat Model

2.1 Problem Overview

This thesis investigates the adversarial vulnerability of Graph Neural Network (GNN)-based anomaly detection models applied to smart-home multivariate time-series data. The objective is to assess how adversarial perturbations applied to device measurements can degrade the ability of these models to detect anomalous behavior in cyber-physical environments.

The study is formulated as a controlled empirical evaluation, where both the data representation and the adversarial manipulation process are explicitly defined. The remainder of this chapter formalizes the multivariate time-series representation, the graph-based device model, the anomaly-detection task, the adversarial threat model, and the evaluation metrics used throughout the thesis.

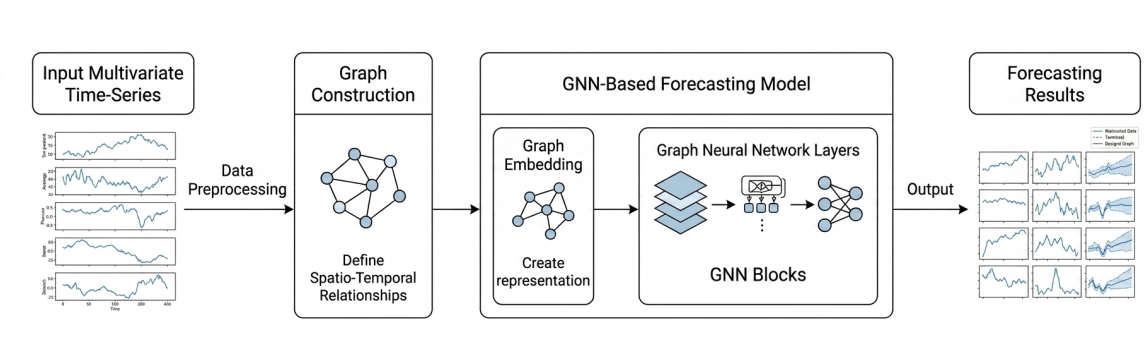

Figure 2.1: GNN-Based Multivariate Time-Series Forecasting Framework

2.2 Multivariate Time-Series Representation

Let N denote the number of devices and T the total number of time steps. The smart-home system is represented as a multivariate time series

$$\mathbf{X} = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}\right] \in \mathbb{R}^{T \times N} \tag{2.1}$$

where each row $\mathbf{x}^{(t)} \in \mathbb{R}^N$ contains the simultaneous readings of all N devices at time step t. In a smart home system, device channels include both binary event-driven signals (e.g.

motion triggers, contact-state changes, smart-plug activations) and continuous environmental measurements (e.g. temperature, humidity, illumination, air quality) [19].

Following standard practice in unsupervised time-series anomaly detection [1], [2], the model is trained exclusively on observations from the primary resident, which define the *normal reference distribution*. The test partition contains both normal observations and a contiguous anomalous segment produced by a secondary actor whose behavioral pattern differs from that of the primary resident [19].

2.3 Device Graph Modeling

To model inter-dependencies between devices, the system is represented as a directed weighted graph

$$\mathcal{G} = \big(\mathcal{V},\, \mathcal{E},\, \mathbf{A}\big) \tag{2.2}$$

where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N device nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed edges representing dependency relations, and $\mathbf{A} \in \{0,1\}^{N \times N}$ is the corresponding adjacency matrix.

In this work, the graph structure is *not* prescribed as prior knowledge but is instead *learned from data*, as is the case for both GDN and MTAD-GAT [1], [2].

2.4 Anomaly Detection Formulation

The anomaly detection task is formulated as an *unsupervised learning* problem, in which the model is trained exclusively on clean normal data and anomalies are identified at test time by elevated prediction or reconstruction errors [1], [2].

Sliding-window input. At each test time step t, the model receives a sliding window of W consecutive observations:

$$\mathbf{X}^{(t)} = \big[\mathbf{x}^{(t-W)}, \mathbf{x}^{(t-W+1)}, \ldots \mathbf{x}^{(t-1)}\big] \in \mathbb{R}^{W \times N} \tag{2.3}$$

and produces a prediction $\hat{\mathbf{y}}^{(t)} \in \mathbb{R}^N$ of the device values at the current step.

Per-device error. For each device i, the raw prediction error at time t is

$$\xi_{i,t} = \big|y_i^{(t)} - \hat{y}_i^{(t)}\big| \tag{2.4}$$

Forecasting-based anomaly score. The simplest approach to deriving a system-level score is to aggregate per-device forecasting errors at each time step. A large deviation between the model's one-step-ahead prediction and the observed value signals that the current observation is unexpected under the learned normal behavior. The system-level score can be formed by taking either the maximum or the mean of the normalized per-device errors across all N devices [1]:

$$A^{(t)} = \underset{i=1,\dots,N}{\mathrm{agg}} \tilde{\xi}_{i,t} \tag{2.5}$$

where $\tilde{\xi}_{i,t}$ is a normalized version of the raw error in Equation (2.4), and agg denotes the chosen aggregation function.

Reconstruction-based anomaly score. An alternative approach is to measure how well a model can reconstruct the input window from a compact latent representation. Rather than predicting future values, a reconstruction model encodes the sliding window $\mathbf{X}^{(t)}$ and decodes it back, with the per-device absolute reconstruction error[2]

$$\xi_{i,t}^{\mathrm{recon}} = \left| \hat{x}_{i,t}^{\mathrm{recon}} - x_i^{(t)} \right| \tag{2.6}$$

serving as the anomaly indicator, where $\hat{x}_{i,t}^{\mathrm{recon}}$ denotes the reconstructed value for device i at the last step of the input window. High reconstruction error signals that the observed pattern within the window is inconsistent with the distribution of normal behavior internalized during training. As with the forecasting case, the system-level score is obtained by aggregating the normalized per-device errors across all N devices:

$$A_{\mathrm{recon}}^{(t)} = \underset{i=1,\dots,N}{\mathrm{agg}} \tilde{\xi}_{i,t}^{\mathrm{recon}} \tag{2.7}$$

Hybrid anomaly score. Forecasting and reconstruction errors carry complementary information: the former captures local temporal surprises, while the latter reflects global distributional shift within the window. A hybrid score combines both signals into a single scalar by means of a weighted linear combination [2]:

$$A^{(t)} = \beta \cdot s_{\mathrm{forecast}}^{(t)} + (1 - \beta) \cdot s_{\mathrm{recon}}^{(t)} \tag{2.8}$$

where $\beta \in [0, 1]$ controls the relative contribution of each component. Setting $\beta = 1$ reduces to a pure forecasting detector, while $\beta = 0$ gives a pure reconstruction detector. The two models considered in this thesis differ in the type of signal they expose: one produces only a forecasting error, while the other exposes both forecasting and reconstruction errors, enabling the hybrid formulation above [2].

Error Normalization Raw errors differ substantially in scale across sensor channels due to the heterogeneous nature of the dataset. To allow fair aggregation into $\tilde{\xi}_{i,t}$ (as used in Equations (2.5) and (2.7)), each error type is normalized using statistics computed *exclusively on the training split*:

$$\tilde{\xi}_{i,t} = \left| \frac{\xi_{i,t} - \mathrm{median}(\xi_i^{\mathrm{train}})}{\mathrm{IQR}(\xi_i^{\mathrm{train}})} \right| \tag{2.9}$$

where the median and IQR are computed along the time axis for sensor i over the training split. To prevent division by zero, the IQR is clipped below at 1.0, and the normalized values are clipped to $[-5, 5]$ before the absolute value is taken. The same parameters are applied to all splits.

Score thresholding. Regardless of the scoring strategy, the final detection decision is obtained by comparing the anomaly score against a fixed threshold λ. A time step t is classified as anomalous if:

$$\hat{y}^{(t)} = \mathbf{1}\big[A^{(t)} > \lambda\big] \tag{2.10}$$

The threshold λ is calibrated on the training partition so that the false-positive rate on normal data is kept within an acceptable range. The specific threshold-selection strategy used in the experiments is discussed in Chapter chapter 4.

2.5 Adversarial Threat Model

This work considers an adversary whose goal is to suppress true anomaly detections - that is, to make an anomalous time segment appear normal to the detector. This corresponds to an *evasion attack* at test time: the model parameters are fixed, and only the test-time inputs are manipulated.

2.5.1 Attacker Objective

The attacker seeks to minimize the anomaly score produced by the detector during a known anomalous segment, so that the score falls below the detection threshold and the anomaly goes undetected. Formally, for a set of perturbed inputs $\tilde{\mathbf{X}}$, the attacker aims to induce

$$A_s^{(t)}\big(\tilde{\mathbf{X}}\big) \;\leq\; \lambda \quad \forall\, t \in \mathcal{T}_{\mathrm{anom}} \tag{2.11}$$

where $\mathcal{T}_{\mathrm{anom}}$ denotes the set of anomalous time steps. This is the most security-critical scenario in cyber-physical monitoring: an attacker conceals an intrusion or malfunction by manipulating device streams.

2.5.2 Attacker Knowledge

Three attack families are evaluated in this thesis, each corresponding to a different level of attacker knowledge. This taxonomy follows the general framework established in the adversarial machine learning literature [3], [5].

- **Black-box (no model access):** The attacker has no knowledge of model parameters or graph structure. Perturbations are drawn from a fixed statistical distribution (Gaussian or uniform noise) applied to all devices simultaneously. This represents the weakest but most realistic threat scenario [4].
- **Grey-box (graph structure known):** Following the threat model of BETA [7], the attacker knows the learned graph structure $\mathbf{A}$ but not the full model parameters. A surrogate model is used to guide gradient-based perturbations toward influential neighbouring nodes.
- **White-box (full access):** The attacker has complete access to model parameters and computes perturbations directly via gradient information, for example using FGSM or PGD [4]. This represents the strongest but least realistic threat and serves as an upper bound on adversarial vulnerability.

2.5.3 Perturbation Model

All perturbations are *additive*: the adversarial input for device i at time t is

$$\tilde{x}_i^{(t)} = x_i^{(t)} + \delta_i^{(t)} \tag{2.12}$$

where $\delta_i^{(t)}$ is the perturbation term. Perturbations are bounded by an ℓ_∞ budget ε:

$$\|\boldsymbol{\delta}\|_\infty = \max_{i,t} \left|\delta_i^{(t)}\right| \;\leq\; \varepsilon \tag{2.13}$$

so that perturbations remain small relative to the normalized input range. The budget ε is varied across experiments to produce degradation curves that reveal how robustness scales with perturbation magnitude [4]. All perturbations are applied in the normalized feature space used as model input, which means that the physical interpretation of ε depends on the original scale of each device channel.

2.5.4 Attack Surface

The attack surface is restricted to device feature values in the test partition. The following are explicitly out of scope:

- **Training-time (poisoning) attacks:** Modifying training data or model weights is not considered. The threat model focuses on evasion at inference time [3].
- **Graph-structure manipulation:** Edge injection or deletion is not evaluated. Only node-feature perturbations are applied, consistent with the realistic assumption that communication links are harder to forge than device readings.
- **Target-device safeguard:** For targeted perturbation attacks, the device whose anomaly score is to be suppressed is treated as safeguarded and is itself not perturbed, following the experimental setup of BETA [7].

2.6 Summary

This chapter formalized the problem of adversarial vulnerability evaluation for GNN-based anomaly detection in smart-home environments. The data representation defines a multivariate time series of N device nodes over T time steps, structured as a learned dependency graph and processed through sliding windows. Rather than relying on a predefined topology, the graph structure is learned end-to-end, enabling the model to capture latent statistical dependencies among devices during normal operation.

Three families of anomaly scoring strategies were identified: forecasting-based, reconstruction-based, and hybrid approaches that combine both mechanisms. In all cases, the model produces a scalar anomaly score $A^{(t)}$, which is compared against a fixed threshold λ to determine whether a given observation is classified as normal or anomalous.

The adversarial threat model considers three levels of attacker knowledge, with perturbations constrained by an ℓ_∞ budget ε and applied only within the test-time feature space. These definitions establish the formal foundation for the clean and adversarial experimental frameworks developed in chapter 4 and chapter 5.

Chapter 3: Data and Preprocessing

3.1 Dataset Description

The IoT Garage dataset introduced by Majib *et al.* [19] is used throughout this study. The dataset was collected in a real smart-home environment and combines multiple sensing modalities, including cyber signals, smart-device interactions, and environmental measurements aligned on a shared temporal axis. In its complete merged form, the dataset contains 463 features.

This thesis focuses on two complementary subsets: `BREMaster.csv`, which contains environmental sensing data with 94 device channels, and `CUMaster.csv`, which contains smart-device and IoT entity data with 369 channels. Both datasets span approximately four weeks with a timestamp resolution of one second. Their main characteristics and the temporal partitions used in the experiments are summarized in Table 3.1.

The timeline is organized around two actors with different behavioral profiles. *Actor 1*, representing the owner of the smart home, occupies the majority of the recording period and defines the normal behavioral baseline used for training and validation. During a shorter interval, *Actor 2*, representing a stranger, interacts with the environment and introduces behavioral deviations from the learned normal patterns. After this interval, the owner resumes normal activity. This temporal organization makes the dataset particularly suitable for behavioral anomaly detection in smart-home environments, where anomalies emerge from changes in activity patterns and device interactions rather than isolated abnormal sensor values [19].

Table 3.1: Dataset characteristics and experimental timeline partitions.

Property	**BREMaster.csv**	**CUMaster.csv**
Total rows	2,671,187	2,671,187
Device channels	94	369
Timestamp resolution	1-second	
Time range	2022-10-18 to 2022-11-17	
Experimental Timeline Partitions		
Actor 1 (Owner)	2022-10-18 -- 2022-11-07	
Actor 2 (Stranger)	2022-11-08 -- 2022-11-10	
Actor 1 (Owner is back)	2022-11-10 -- 2022-11-17	

3.2 Anomaly Definition and Experimental Assumptions

In this thesis, anomaly detection is treated as *behavioral deviation detection*. The central assumption is that Actor 1 defines the reference distribution of normal smart-home behavior, while Actor 2 introduces a distribution shift that may be interpreted as anomalous from the perspective of the learned baseline. This framing follows naturally from the motivation of the dataset itself, which was designed to support the detection of deviations in users' routines [19].

This choice is important because the problem is formulated in an unsupervised setting. The anomaly detector is trained without access to labeled anomaly annotations and instead learns regular temporal and cross-device patterns from normal behavior, identifying deviations through anomaly scoring..

3.3 Device Characteristics and Data Representation

The dataset combines three broad categories of information [19]. The first is **cyber data**, mainly network traffic and communication traces produced by connected devices. The second is **smart-device state data**, such as events emitted by consumer IoT devices, smart appliances, and home-automation components. The third is **environmental sensing**, including measurements related to temperature, humidity, air quality, motion, illumination, proximity, and other physical conditions of the home.

This heterogeneous composition is well suited to multivariate time-series anomaly detection. Some variables behave as binary or event-like indicators, for example contact changes, motion triggers, or device activations. Others evolve more continuously, such as humidity, temperature, pressure, and particulate matter measurements. As a result, the learned anomaly detector must account for both abrupt state transitions and slower environmental trends, as well as the dependencies between them.

Figure 3.1 illustrates this heterogeneity by comparing temperature readings from multiple devices placed in different locations of the same environment over a 48-hour window. Even under shared household conditions, the signals exhibit distinct local patterns and variability, highlighting the non-uniform nature of sensor behavior.

The dataset documentation also shows that many daily activities activate multiple modalities simultaneously. Actions such as entering the house, cooking, or watching television are associated with both environmental responses and smart-device state changes. This cross-modal coupling is one of the main reasons why a multivariate, graph-based representation is more informative than analyzing each source independently.

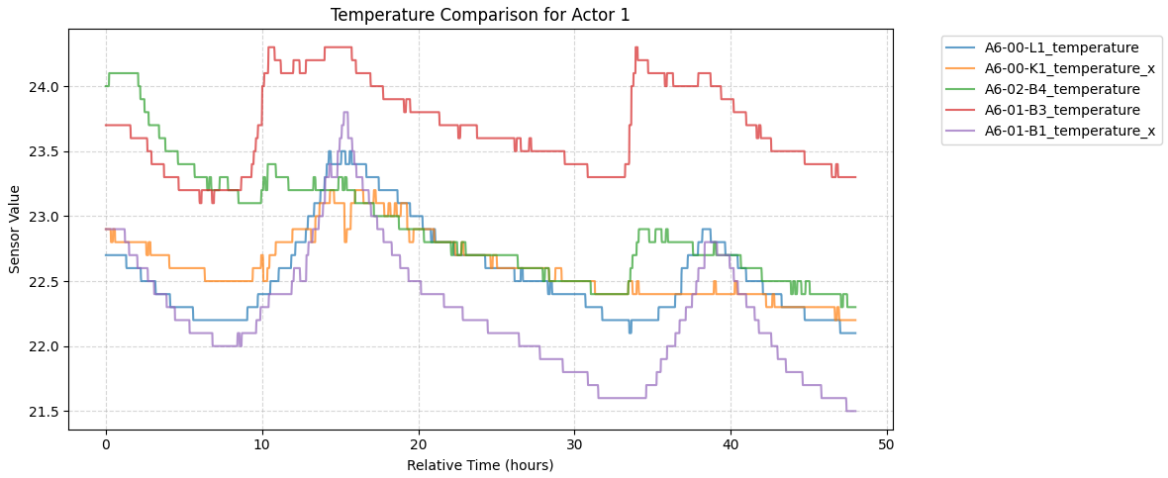

Figure 3.1: Temperature Readings from Five Devices Over a 48-Hour Window (Actor 1): Illustrating Per-Device Baseline and Response Variability

3.4 Preprocessing Pipeline

The preprocessing adopted in this thesis converts the raw smart-home sensor streams into a stable and comparable input representation for the anomaly detection models. The pipeline consists of six sequential steps, illustrated in Figure 3.2 and described below.

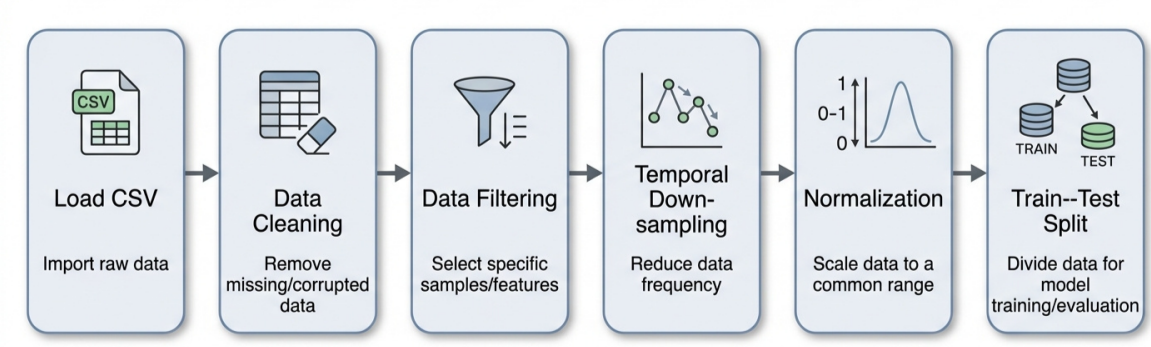

Figure 3.2: Overview of the six-step preprocessing pipeline.

3.4.1 Data Cleaning

The `CUMaster.csv` file contains a column whose name carries an extraneous trailing whitespace, as well as a column in which all values are `NaN` (a non-functional device channel). The cleaning step strips whitespace from all column names and removes columns that are entirely `NaN`, ensuring that no empty feature enters the model [19]. No rows are dropped by default, so the temporal alignment of the remaining columns is preserved. This step is applied to

`CUMaster.csv` only, as `BREMaster.csv` does not exhibit the same artefacts.

3.4.2 Data Filtering

Because the dataset contains up to 369 device channels, a data filtering step is necessary to retain only channels that carry meaningful behavioral information while keeping the anomaly detection problem computationally tractable.

For the CUMaster, the selected channels include motion sensors, contact sensors, vibration sensors, wireless switch event signals, temperature and humidity probes, water-leak sensors, smart-plug switches, lighting fixtures, an air-quality sensor, a climate controller, and connected appliances such as a kettle, a humidifier, and a robotic vacuum. Altogether, **32 channels** are retained.

For the BREMaster, the selected channels consist primarily of room-level environmental sensors covering motion, humidity, temperature, illuminance, and contact states across different areas of the house, including the hallway, bedrooms, bathroom, kitchen, landing, and sitting room. Contact sensors on the front door are also included. Altogether, **62 channels** are retained.

A summary of the retained channels by device type is provided in Table 3.2.

Table 3.2: Summary of selected device channels by type for each dataset file.

Device type	**BREMaster**	**CUMaster**
Motion / Occupancy	✓	✓
Temperature	✓	✓
Humidity	✓	✓
Illuminance	✓	--
Contact (door / window)	✓	✓
Vibration	--	✓
Wireless switch events	--	✓
Water leak	--	✓
Smart plugs / switches	--	✓
Lighting (smart bulbs)	--	✓
Air quality (PM2.5)	--	✓
Climate controller	--	✓
Connected appliances	--	✓
Total selected	**62**	**32**

This selection ensures that the input representation emphasises variables that reflect temporal behavior rather than static identifiers, descriptive metadata, or near-constant channels that carry no discriminative information [19].

3.4.3 Temporal Down-sampling

A key challenge in smart-home datasets is the heterogeneity of sampling frequencies across different sensing modalities. To address this issue, the dataset authors perform temporal alignment by projecting all sources onto a unified timeline and applying forward-filling of the most recently observed values for lower-frequency streams during the construction of the combined multivariate representation [19]. This synchronization procedure converts asynchronous device streams into a unified time-indexed representation. The resulting sequence is suitable for downstream modeling.

Following this alignment step, the data are resampled to a uniform target frequency. In this thesis, a sampling interval of **1-second** is adopted across all experiments. Several alternative downsampling rates, including 3-second and 5-second intervals, were also evaluated during preliminary experiments.

Down-sampling is performed using aggregation strategies that depend on the nature of the signal. For binary and event-driven channels, a *maximum* aggregation function is applied within each time window to ensure that short-lived activations are preserved. In contrast, continuous-valued signals are aggregated using the *mean*, resulting in a smoother representation that reduces high-frequency noise. This distinction is handled automatically within the preprocessing pipeline through heuristic-based detection of device type, leveraging statistical properties such as value range and cardinality.

3.4.4 Normalization

Because the retained variables come from very different sensing mechanisms, they naturally occupy different numeric ranges. Environmental variables may vary gradually over wide intervals, while device states may remain close to a small bounded set of values. Before training, continuous features are therefore scaled to comparable ranges using a **MinMax scaler** fitted exclusively on the Actor 1 training segment, so that no information from the anomalous test period leaks into the normalization parameters. This step is especially important in multivariate forecasting settings, where anomaly scores are derived from prediction errors across all channels.

The fitted scaler is saved alongside the processed arrays and is reloaded during evaluation so that the same transformation is applied consistently to all data splits.

3.4.5 Train-Test Split

The temporal order of the data is preserved throughout all experiments. Random shuffling would destroy the sequential dependencies that the models are designed to learn, and more importantly, would blur the distinction between the primary resident's habitual behavior and

the secondary actor's shifted behavior.

Accordingly, the Actor 1 data forms the basis of the normal reference regime used for model development. Within Actor 1's period, the last $10\,\%$ of time steps are held out as a **validation set**, while the remaining $90\,\%$ constitute the **training set**. The Actor 2 segment serves as the **anomalous test set**. This chronological partition respects the real-world deployment logic of anomaly detection: a model should learn normal patterns first and then be exposed later to a potentially abnormal behavior.

The approximate size of each split at a 1-second resolution is given in Table 3.3.

Table 3.3: Approximate proportion of the full timeline represented by each data split.

Split	**Source**	**% of full timeline**
Training	Actor 1	$\approx 61.1\%$
Validation	Actor 1	$\approx 6.8\%$
Anomalous test	Actor 2	$\approx 9.7\%$
Remaining data	Actor 1	$\approx 22.6\%$

3.4.6 Sliding Window Construction

To feed the anomaly detection models, the synchronized multivariate sequence is reorganised into overlapping sliding windows. Each window contains W consecutive observations and is paired with the device values at the immediately following time step, which serves as the prediction target. Formally, sample t in the dataset is represented as the pair $(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$ as defined in Equation (2.3).

The window size W is treated as a key hyperparameter and is selected empirically through hyperparameter search, with its impact analyzed in Chapter 4. This formulation allows the models to capture both short-term temporal dynamics and cross-feature dependencies, which are essential for effective anomaly detection in multivariate time series.

Sliding windows are constructed without shuffling, preserving the chronological order of samples within each split.

3.5 Summary

This chapter presented the IoT Garage dataset [19] and the preprocessing pipeline adopted to convert it into a form suitable for anomaly detection. The dataset captures synchronized cyber, smart-device, and environmental information from a real smart home over multiple weeks, across two actors with contrasting behavioral profiles. While the full dataset contains a large number of heterogeneous sensing channels, this thesis focuses on a filtered subset of behavioral features derived from the original recordings.

Preprocessing is understood as the preparation of a coherent multivariate sequential view. This includes loading the raw data, cleaning invalid or missing entries, selecting relevant behavioral device channels, and aligning asynchronous streams onto a unified temporal grid to obtain a consistent time-indexed representation. The resulting time series is then partitioned chronologically into training, validation, and anomalous test splits according to the actor timeline. Feature scaling is performed using a MinMax normalization scheme fitted exclusively on the training portion of the benign data, preventing leakage of test-period statistics into the normalization parameters. Finally, fixed-length overlapping sliding windows are constructed to enable forecasting-based anomaly detection.

These design choices establish the data foundation for the baseline and adversarial experiments developed in the following chapters.

Chapter 4: Baseline Models and Clean Performance

4.1 Overview

This chapter presents the two Graph Neural Network (GNN)-based anomaly detection models evaluated in this thesis (Graph Deviation Network (GDN) [1] and MTAD-GAT [2]) together with the training protocol used to build them and the clean detection performance obtained before any adversarial perturbation is applied.

The goal of this chapter is twofold. First, it provides a precise description of each model's architecture and forward-pass logic so that the differences in their internal representations, which are later exploited or probed by adversarial attacks, are clearly established. Second, it reports the performance of each model on the anomaly detection task under normal (non-adversarial) conditions, thereby defining the reference baseline against which robustness degradation is measured in Chapter 5.

Both models follow the general anomaly detection formulation established in Chapter 2: they receive a sliding window $\mathbf{X}^{(t)}$ (Equation (2.3)) as input, produce one or more error signals, and compare an aggregated anomaly score $A^{(t)}$ against a fixed threshold λ (Equation (2.10)) to issue a binary detection decision.

4.2 Models Under Study

Two GNN-based anomaly detection architectures are selected for this study because they represent contrasting design philosophies that are relevant to adversarial robustness analysis. GDN [1] relies on a sparse, learned graph over sensor embeddings and adopts a pure forecasting objective, yielding an interpretable, efficient detector sensitive to sensor-level deviations. MTAD-GAT [2] operates on a dual-attention architecture with fully connected feature and temporal attention mechanisms and jointly optimizes forecasting and reconstruction objectives, producing a richer but more computationally demanding representation.

These two models have been applied to IoT and cyber-physical anomaly detection in related work [4], [7] and are well-suited for studying adversarial vulnerability because their graph-based representations make the propagation pathways of adversarial perturbations more interpretable [3].

4.2.1 Graph Deviation Network (GDN)

GDN was introduced by Deng and Hooi [1] for multivariate time-series anomaly detection in sensor networks. Its central innovation is a sensor embedding layer that assigns to every sensor node $v_i \in \mathcal{V}$ a trainable vector $\mathbf{e}_i \in \mathbb{R}^d$, where d is the embedding dimension. These embeddings are learned jointly with all other model parameters during training, allowing the model to internalize the statistical role of each sensor in the system.

Graph structure learning. The embeddings are used to construct the graph topology. The directed adjacency structure is determined by selecting, for each sensor v_i, the k sensors whose embeddings are most similar under cosine similarity:

$$\mathrm{sim}(v_i, v_j) = \frac{\mathbf{e}_i^\top \mathbf{e}_j}{\|\mathbf{e}_i\| \, \|\mathbf{e}_j\|} \tag{4.1}$$

The resulting adjacency matrix $\mathbf{A}$ is sparse, with only a limited number of incoming edges retained for each node. This Top-k sparsity makes GDN computationally efficient and restricts the immediate propagation of perturbations.

Embedding-conditioned graph attention. Once the graph is built, GDN performs message passing with an embedding-conditioned attention mechanism. For a target node v_i and a neighbor $v_j \in \mathcal{N}(i)$, the attention weight is computed from the concatenation of the transformed temporal feature and the node embedding. The aggregated neighborhood representation is then processed by a batch normalization layer, a ReLU activation, and a two-layer MLP output head that produces the per-sensor prediction $\hat{y}_t^{(i)}$.

Anomaly scoring. GDN uses the forecast-based anomaly score $A^{(t)}$ defined in Equation (2.5), with the aggregation operator set to maximum. The raw per-device errors $\xi_{i,t}$ (Equation (2.4)) are normalized per sensor using training-split statistics before aggregation, and a moving average is applied to the resulting score sequence before thresholding.

4.2.2 MTAD-GAT

MTAD-GAT was introduced by Zhao et al. [2] and differs from GDN in three fundamental respects: it operates on a fully connected graph (no Top-k sparsity), it employs dual attention mechanisms over both the feature dimension and the time dimension, and it jointly trains a forecasting head and a reconstruction head.

1-D convolutional preprocessing. Before graph attention is applied, the input window $\mathbf{X}^{(t)} \in \mathbb{R}^{N \times W}$ passes through a one-dimensional convolutional layer that operates indepen-

dently along the time axis for each feature channel. This layer extracts local temporal patterns and provides a richer feature representation to the subsequent attention layers.

Feature-oriented graph attention. The feature-oriented GAT layer models dependencies between different sensor channels. It operates on the full $N \times N$ adjacency matrix, so every pair of sensors can attend to each other. For each sensor v_i, the layer produces a context vector $h_i^{(l)}$ that summarizes the influence of all neighboring sensors.

Time-oriented graph attention. A symmetric attention layer is applied across the time dimension, treating each of the W time steps as a node and computing dependencies between them. This allows the model to identify which temporal positions in $\mathbf{X}^{(t)}$ are most informative for the current detection decision.

GRU integration. The outputs of the two attention layers and the original convolutional features are concatenated along the feature axis, yielding a combined representation of shape $(B, W, 3N)$. This tensor is fed into a Gated Recurrent Unit (GRU) [20], which compresses the temporal dimension into a fixed-size hidden state corresponding to the last time step.

Dual output heads. The GRU hidden state is passed to two independent output modules. An MLP *forecasting head* produces the per-sensor prediction $\hat{y}_t^{(i)}$ for the next time step. A GRU-based *reconstruction head* produces $\tilde{x}_t^{(i)}$, the reconstructed value for each sensor at the last step of the input window.

Anomaly scoring. At evaluation time, MTAD-GAT exposes both the forecast-based score $A^{(t)}$ (Equation (2.5)) and the reconstruction-based score $A_{\text{recon}}^{(t)}$ (Equation (2.7)), which are combined into the hybrid score (Equation (2.8)) with mixing coefficient $\beta = 0.5$. Score aggregation across sensor channels uses the maximum operator for both error types.

4.2.3 Architectural Considerations for Adversarial Analysis

Several architectural differences between GDN and MTAD-GAT are directly relevant to adversarial vulnerability analysis and motivate the comparative evaluation conducted in this thesis.

First, GDN's Top-k sparse graph restricts message passing to at most k neighbors per node, which limits the reach of localized perturbations. MTAD-GAT's fully connected feature graph, by contrast, allows a perturbation injected into any single sensor to influence all other sensors within one attention step, potentially amplifying its effect on the aggregate anomaly score.

Second, the dual objective of MTAD-GAT provides two complementary detection signals. An adversarial manipulation designed to minimize the forecast error $\xi_{i,t}$ (Equation (2.4)) might simultaneously increase the reconstruction error $\xi_{i,t}^{\text{recon}}$ (Equation (2.6)), making it harder to suppress the combined hybrid score. GDN, with its single forecasting objective, presents a simpler attack surface.

Third, the embedding-conditioned attention in GDN ties the graph topology $\mathbf{A}$ to learned representations of sensor identity $\mathbf{e}_i$ rather than to instantaneous feature values. The graph structure is therefore fixed at inference time and cannot adapt in response to perturbations, a characteristic that both constrains the defense and limits the scope of structural attacks.

These properties are analyzed empirically through the adversarial experiments in Chapter 5.

4.3 Training Procedure

4.3.1 Training Setup

Both models are trained on the CUMaster sensor channels described in Chapter 3, using the Actor 1 data as the source of normal behavior. The training split corresponds to the first $90\,\%$ of Actor 1's activity period (approximately $1{,}633{,}000$ time steps at 1-second resolution), and the remaining $10\,\%$ constitute the validation split used for early stopping and model selection.

All experiments use the hyperparameters summarized in Table 4.1. The window size W differs between models following the empirical search described in Section 4.4.

Table 4.1: Training hyperparameters for each model.

Hyperparameter	**GDN**	**MTAD-GAT**
Dataset	CUMaster	CUMaster
Window size W	30	30
Downsample rate	1s	1s
Batch size B	32	32
Maximum epochs	50	50
Learning rate	10^{-4}	10^{-4}
Early stopping patience	10	10
Random seed	42	42

Model-specific architectural hyperparameters follow the configuration used during model training: GDN uses a hidden dimension $d = 64$, $k = 15$ neighbors, and a single attention head. MTAD-GAT uses a convolutional kernel of size 5, a GRU hidden dimension of 100, and 2 attention heads.

4.3.2 Loss Functions

GDN is trained with a mean-squared error (MSE) loss between its predicted values $\hat{y}_t^{(i)}$ and the actual next-step observations $x_t^{(i)}$:

$$\mathcal{L}_{\text{GDN}} = \frac{1}{NB} \sum_{b=1}^{B} \sum_{i=1}^{N} \left(\hat{y}_{t_b}^{(i)} - x_{t_b}^{(i)} \right)^2 \tag{4.2}$$

where B is the batch size (Table 4.1).

MTAD-GAT jointly minimizes forecasting and reconstruction errors:

$$\mathcal{L}_{\text{MTAD}} = \mathcal{L}_{\text{forecast}} + \mathcal{L}_{\text{recon}} \tag{4.3}$$

where $\mathcal{L}_{\text{forecast}}$ is the MSE between $\hat{y}_t^{(i)}$ and the next-step targets $x_t^{(i)}$, and $\mathcal{L}_{\text{recon}}$ is the MSE between the reconstructed values $\tilde{x}_t^{(i)}$ and the original input window. Both terms receive equal weight, consistent with the $\beta = 0.5$ scoring policy applied at evaluation time.

4.3.3 Optimizer and Learning Rate Scheduler

Both models are optimized with the Adam algorithm at a fixed initial learning rate of 10^{-4} [20]. A `ReduceLROnPlateau` scheduler monitors the validation loss and halves the learning rate whenever no improvement is observed for three consecutive epochs, with a minimum floor of 10^{-6}. This prevents stagnation in later training stages while preserving initial convergence speed.

4.3.4 Early Stopping and Model Selection

Training is stopped if the validation loss does not improve for 10 consecutive epochs. At each epoch, two checkpoints are maintained: the *last* model, saved unconditionally, and the *best* model, saved only when a new minimum validation loss is achieved. The best checkpoint is used for all subsequent evaluation and adversarial experiments.

4.3.5 Reproducibility

All experiments use random seed 42, applied to Python's `random` module, NumPy, and PyTorch. Sliding windows are constructed in chronological order without shuffling, preserving the temporal structure of the training data.

4.4 Hyperparameter Search

Before the final model configurations were fixed, a systematic sweep over window size W, dataset, and downsampling rate was conducted to identify the best-performing setup for each model.

Dataset selection. Three dataset conditions were compared: the CUMaster alone, the BREMaster alone, and a merged version combining both. The CUMaster consistently yielded the lowest validation losses for both models (GDN: 0.000329; MTAD-GAT: 0.000096) compared with the BREMaster (GDN: 0.003694; MTAD-GAT: 0.00116). The merged condition degraded performance in all cases and produced unstable detection at larger window sizes. This outcome reflects the higher regularity and lower noise level of the CUMaster channels, which present clearer temporal patterns for the model to learn. All final experiments therefore use the CUMaster dataset exclusively.

Window size. Window sizes of 10, 20, and 30 time steps were evaluated. For GDN on the CUMaster, $W = 30$ minimized the validation loss. For MTAD-GAT on the CUMaster, detection performance was essentially window-independent, but $W = 30$ was selected as a moderate compromise. Large windows (40-50) degraded performance for both models.

Downsampling rate. Experiments at 1-second and 3-second resolution were conducted for both GDN and MTAD-GAT on the CUMaster and BREMaster. The 3-second rate produced lower variance and faster training but reduced temporal resolution and lowered peak anomaly scores. The 1-second rate was adopted for all final configurations.

The resulting best configurations are summarized in Table 4.2.

Table 4.2: Best hyperparameter configurations.

Model	**Dataset**	W	**Downsample**	**LR**
GDN	CU	30	1s	10^{-4}
MTAD-GAT	CU	30	1s	10^{-4}

4.5 Evaluation Metrics

To compute anomaly scores, we follow the same pipeline described in Section 2.4

4.5.1 Detection Metrics

Each time step t receives a binary label $\hat{y}^{(t)} \in \{0, 1\}$ via Equation (2.10). Two scalar summaries are computed:

- **Detection rate (DR):** The proportion of anomalous time steps that are correctly flagged by the model:

$$\mathrm{DR} = \frac{1}{|\mathcal{T}_{\mathrm{anom}}|} \sum_{t \in \mathcal{T}_{\mathrm{anom}}} \mathbf{1}\big[A^{(t)} > \lambda\big]$$

  A high detection rate indicates that the model successfully identifies the anomalous period. Under an evasion attack, suppressing this quantity is the primary attacker objective.

- **False-positive rate (FPR):** The proportion of normal time steps that are incorrectly flagged:

$$\mathrm{FPR} = \frac{1}{|\mathcal{T}_{\mathrm{normal}}|} \sum_{t \in \mathcal{T}_{\mathrm{normal}}} \mathbf{1}\big[A^{(t)} > \lambda\big]$$

  The threshold λ is selected to keep the FPR within an acceptable range, ensuring that the reported detection rate reflects genuine sensitivity rather than indiscriminate flagging.

4.6 Clean Detection Performance

The thresholds τ_{99} and τ_{90} were defined from the empirical distribution of anomaly scores on the normal training data A_{train}. Because the setting is unsupervised, anomalous labels are not available during threshold selection. Consequently, the threshold must be estimated solely from normal behavior statistics. Percentile-based thresholds are therefore used to define operating points with controlled false-positive behavior. The 99th-percentile threshold represents a conservative high-specificity regime in which only the largest $1\,\%$ of normal scores are classified as anomalous, while the 90th-percentile threshold provides a more sensitive operating regime with higher Detection Rate at the cost of increased false positives.

Table 4.3 reports the detection rate (DR) and false-positive rate (FPR) for both models under the two threshold settings. These values constitute the clean baselines against which adversarially perturbed performance is compared in Chapter 5.

Table 4.3: Clean detection performance for GDN and MTAD-GAT on the Actor 2 test set at two threshold levels.

Model	Threshold	DR	FPR
MTAD-GAT	τ_{99} (99th percentile)	0.37	0.01
MTAD-GAT	τ_{90} (90th percentile)	0.79	0.10
GDN	τ_{99} (99th percentile)	0.59	0.01
GDN	τ_{90} (90th percentile)	0.79	0.10

Several observations can be drawn from these results. At the conservative threshold τ_{99}, GDN achieves a considerably higher detection rate than MTAD-GAT (0.59 vs. 0.37) for the

same false-positive rate of $1\,\%$. This result is somewhat unexpected given MTAD-GAT's lower validation loss and richer architecture. It suggests that, at very high specificity, GDN's maximum aggregation of $\tilde{\xi}_{i,t}$ produces anomaly scores that are better separated from the tail of the normal distribution A_{train}. MTAD-GAT's reconstruction component may amplify score variance in the normal regime, producing a heavier normal tail that pushes τ_{99} to a level that is less discriminative for the anomalous period.

At the sensitive threshold τ_{90}, both models converge to the same detection rate of 0.79 at a false-positive rate of $10\,\%$, indicating that the Actor 2 anomaly signal A_{actor2} is strong enough for both models to detect approximately $79\,\%$ of anomalous time steps once the threshold is relaxed.

The clean performance results establish several properties that are consequential for the adversarial analysis.

At the strictest operating point (τ_{99}), GDN is the stronger detector (DR $= 0.59$) despite MTAD-GAT's lower training loss. This suggests that MTAD-GAT's score distribution A_{train} has a heavier tail (produced by the reconstruction component amplifying score variance in the normal regime) which requires a less aggressive threshold to separate the anomalous period from normal behavior.

Both models reach the same DR of 0.79 at τ_{90}, setting a common clean-performance baseline for robustness comparisons: an attack that suppresses one model's DR below this ceiling by more than the other's reveals a genuine vulnerability specific to that model's architecture or scoring strategy.

The DR values (0.37 and 0.79) might appear modest in absolute terms. However, the anomalous period corresponds to a *behavioral distribution shift* (a different actor's routine) rather than isolated fault events or injected synthetic anomalies. The detection rate reflects how densely $A^{(t)}$ exceeds the threshold within the anomalous period.

4.7 Summary

This chapter described the two GNN-based anomaly detection models evaluated in this thesis and reported their performance under clean conditions. Both models are trained on the CUMaster dataset with a standardized protocol using the Adam optimizer, early stopping, and a learning rate scheduler.

The clean detection results, summarized in Table 4.3, show that at the conservative threshold τ_{99}, GDN achieves a higher detection rate (0.59) than MTAD-GAT (0.37) for the same 1% false-positive rate. At the sensitive threshold τ_{90}, both models converge to a detection rate of 0.79 at a 10% false-positive rate.

These baseline characteristics define the reference against which adversarial vulnerability is assessed in Chapter 5.

Chapter 5: Adversarial Attack Framework and Evaluation

5.1 Overview

This chapter describes the adversarial attack framework developed and applied in this thesis, and reports the empirical results obtained by evaluating it against the two GNN-based anomaly detectors established in Chapter 4. The attack families are organized by attacker knowledge, spanning three threat levels. Black-box attacks assume no access to model internals and typically provide a conservative estimate of achievable degradation. Grey-box attacks exploit knowledge of the learned graph structure to concentrate perturbations on the most influential nodes. White-box attacks have full access to model parameters and gradients, representing the strongest adversarial threat and and are generally expected to produce the highest attack effectiveness. For each threat level, the corresponding attacks are formally described, their parameter grids defined, and their effectiveness evaluated against both GDN and MTAD-GAT. The chapter concludes with a comparative analysis of the two architectures' robustness profiles across all attack families.

5.2 Attack Taxonomy

The adversarial attacks evaluated in this thesis are classified by the level of information available to the attacker, consistent with the threat model formalized in Chapter 2 and with the classification adopted in the adversarial machine learning literature [3], [5]. Three knowledge levels are considered, each corresponding to a different assumption about what the adversary can observe or access.

5.2.1 Black-Box Attacks

Natural Evolution Strategies (NES)

NES, introduced by Ilyas et al. [21] as a score-based black-box adversarial attack, estimates a pseudo-gradient of the anomaly score from a small number of paired positive/negative score queries and then takes a signed descent step --- achieving gradient-like efficiency with no access to model parameters or architecture. At each iteration k, a set of S random unit-normal directions $\mathbf{u}_s \in \mathbb{R}^{W \times N}$ is sampled. Each direction is queried at $\tilde{\mathbf{X}}_k^{(t)} \pm \sigma \mathbf{u}_s$ (antithetic pairs), and the NES gradient estimate is:

$$\hat{\mathbf{g}}_k = \frac{1}{2S\sigma} \sum_{s=1}^{S} \left[\mathcal{L}_{\text{adv}}\left(\tilde{\mathbf{X}}_k^{(t)} + \sigma \mathbf{u}_s\right) - \mathcal{L}_{\text{adv}}\left(\tilde{\mathbf{X}}_k^{(t)} - \sigma \mathbf{u}_s\right) \right] \mathbf{u}_s \tag{5.1}$$

where σ is the smoothing radius and S is the number of antithetic pairs per step (total queries per step: $2S$). The adversarial input is then updated using the same signed-step rule as PGD, followed by projection back onto the ℓ_∞ ball:

$$\tilde{\mathbf{X}}_{k+1}^{(t)} = \Pi_\varepsilon\left(\tilde{\mathbf{X}}_k^{(t)} - \alpha \cdot \text{sign}(\hat{\mathbf{g}}_k)\right) \tag{5.2}$$

where $\hat{\mathbf{g}}_k$ is the NES gradient estimate from Equation (5.1) and Π_ε is the projection operator defined in Equation (5.8). The best adversarial input seen across all K steps is returned as the final perturbation.

In this implementation, $S = 10$ antithetic pairs are used per step, giving a total query cost of $2SK = 400$ model calls per batch.

Algorithm 1: Natural Evolution Strategies (NES) — Black-Box Attack

Input: Input window X, model f, budget ε, step size α, iterations K, number of random pairs S

Output: Best adversarial window found

Start from a random perturbation inside the allowed budget
Record the current anomaly score as the best seen so far
for *each iteration* $k = 1$ **to** K **do**
    `// Estimate a gradient without backpropagation`
    Initialize gradient estimate to zero
    for *each random pair* $s = 1$ **to** S **do**
        Sample a random direction **u**
        Query the model at current input + small noise in direction **u**
        Query the model at current input − small noise in direction **u**
        Add the score difference scaled by **u** to the gradient estimate
    end
    Normalize the gradient estimate
    Take one step using the estimated gradient
    Clip the perturbation so it never exceeds the budget ε
    if *this step produced a lower anomaly score than the best so far* **then**
        Save this as the new best result
    end
end
return the best adversarial window found

Simple Black-Box Attack (SimBA)

SimBA, introduced by Guo et al. [22], is a purely random-search black-box attack that requires no gradient estimation. At each step, a random direction affecting only a single feature dimension $\mathbf{q} \in \mathbb{R}^{W \times N}$ is proposed (a unit vector with a single non-zero entry at position (w, n)) and the anomaly score is queried at $\tilde{\mathbf{X}}^{(t)} + \varepsilon\mathbf{q}$. If the score decreases, the positive step is accepted, otherwise the negative direction $-\varepsilon\mathbf{q}$ is tried. If neither reduces the score, the direction is discarded. Formally, the update rule is:

$$\tilde{\mathbf{X}}^{(t)} \leftarrow \begin{cases} \Pi_\varepsilon\left(\tilde{\mathbf{X}}^{(t)} + \varepsilon\mathbf{q}\right) & \text{if } \mathcal{L}_{\text{adv}}\left(\tilde{\mathbf{X}}^{(t)} + \varepsilon\mathbf{q}\right) < \mathcal{L}_{\text{adv}}\left(\tilde{\mathbf{X}}^{(t)}\right) \\ \Pi_\varepsilon\left(\tilde{\mathbf{X}}^{(t)} - \varepsilon\mathbf{q}\right) & \text{else if } \mathcal{L}_{\text{adv}}\left(\tilde{\mathbf{X}}^{(t)} - \varepsilon\mathbf{q}\right) < \mathcal{L}_{\text{adv}}\left(\tilde{\mathbf{X}}^{(t)}\right) \\ \tilde{\mathbf{X}}^{(t)} & \text{otherwise} \end{cases} \tag{5.3}$$

The process is repeated for a budget of Q direction draws, with an actual query cost of between Q and $2Q$ model calls depending on how many directions are accepted on the first try.

SimBA introduces no hyper-parameters beyond ε and Q, and is not sensitive to smoothing radius tuning. It is considered the a standard random-search black-box baseline [22] and complements NES in the evaluation: where NES is efficient in high-dimensional spaces by aggregating information across multiple directions, SimBA is more transparent and better suited to diagnosing which individual sensor channels are easiest to exploit.

Algorithm 2: Simple Black-Box Attack (SimBA)

Input: Input window X, model f, budget ε, direction budget Q
Output: Adversarial window $\tilde{X}$

```
Start with no perturbation
Query the model to get the current anomaly score
Shuffle the list of all (time step, sensor) positions
for each position (w, n) in the shuffled list, up to Q positions do
    Try adding +ε to feature (w, n)
    if the anomaly score decreases then
        Keep this change
    else
        Try adding −ε to feature (w, n)
        if the anomaly score decreases then
            Keep this change
        end
    end
end
return the perturbed window
```

5.2.2 Grey-Box Attack

Budgeted Evasion via Targeted Node Selection and Constrained PGD (BETA)

BETA was introduced by Xaviar and Ardakanian [7] specifically for graph-based anomaly detection in sensor networks. Whereas FGSM and PGD perturb all device channels simultaneously using full gradient access, BETA operates under a grey-box assumption: the attacker knows the learned graph structure but uses a model-specific structural information to identify the most influential neighbors of a target node u, the sensor whose anomaly score is to be suppressed. The target node itself is never perturbed, following the safeguard assumption of the original paper [7]. The attack proceeds in two stages.

Stage 1 --- Influencer selection. The first stage identifies a set of B influencer nodes $\bar{\mathcal{V}} \subset \mathcal{V} \setminus \{u\}$ whose perturbation is expected to have the greatest effect on node u's anomaly score. The selection mechanism is model-native and differs between the two architectures:

- **For GDN:** The knn adjacency graph $\mathbf{A}$ is constructed from GDN's trainable sensor embeddings $\mathbf{e}_i$ (Section 4.2.1). Eigenvector centrality is computed on this graph using power iteration, and the B neighbors of u with the highest centrality scores are selected as influencers.

- **For MTAD-GAT:** The feature-oriented attention matrix is extracted from MTAD-GAT's graph attention layer (Section 4.2.2) on a representative input window. A combined influence score is formed for each candidate node $j \neq u$ as the product of the attention weight $\alpha_{u,j}$ (measuring how strongly node u attends to node j) and the eigenvector centrality of j computed on the absolute attention matrix. The B nodes with the highest combined score are selected as influencers.

Influencer selection is performed once per target node before the batch loop, so its computational cost does not scale with the number of attacked windows.

Stage 2 --- Constrained PGD. Once the influencer set $\bar{\mathcal{V}}$ has been identified, BETA runs a constrained variant of PGD in which gradients are masked so that only the influencer nodes are perturbed. Let $\mathbf{M} \in \{0,1\}^{W \times N}$ be a binary mask with $M_{t',i} = 1$ if and only if $i \in \bar{\mathcal{V}}$. The masked update rule is:

$$\tilde{\mathbf{X}}_{k+1}^{(t)} = \mathbf{X}^{(t)} + \mathrm{clip}\Big(\tilde{\mathbf{X}}_{k}^{(t)} - \alpha \cdot \mathrm{sign}\Big(\nabla_{\tilde{\mathbf{X}}_{k}^{(t)}} \mathcal{L}_{\mathrm{adv}}^{(u)}\Big) \odot \mathbf{M},\ -\varepsilon,\ +\varepsilon\Big) \tag{5.4}$$

where $\odot$ denotes element-wise multiplication and the node-specific loss $\mathcal{L}_{\mathrm{adv}}^{(u)}$ is computed only at the target node:

$$\mathcal{L}_{\mathrm{adv}}^{(u)}\big(\tilde{\mathbf{X}}^{(t)}\big) = \big|\hat{y}_t^{(u)} - x_t^{(u)}\big| \tag{5.5}$$

By concentrating the perturbation budget on the most graph-central neighbors of the target node, BETA can achieve score suppression comparable to global PGD while modifying far fewer device channels --- a more realistic and stealthy attack profile that better represents the capabilities of a knowledgeable but resource-constrained adversary [7].

Algorithm 3: BETA — Budgeted Evasion via Targeted Node Selection

Input: Input windows, model f, target sensor u, influencer budget B, perturbation budget ε, iterations K

Output: Adversarially perturbed windows

`// Stage 1 - Select which sensors to perturb (done once)`

if *model is GDN* **then**
  Build the sensor dependency graph from the learned embeddings
  Rank all neighbor sensors of u by their graph centrality score
else
  Extract the attention weights that u uses to attend to other sensors
  Rank all other sensors by their attention weight multiplied by their centrality
end
Select the top B sensors as influencers
Build a mask that allows perturbations only on those B sensors

`// Stage 2 - Perturb each window using constrained PGD`

foreach *input window* **do**
  Start from a random perturbation applied only to the influencer sensors
  for *each iteration* $k = 1$ **to** K **do**
    Run the model and compute the prediction error at target sensor u
    Compute the gradient of that error
    Take one step in the direction that reduces the error, **masked to influencers only**
    Clip the perturbation so it stays within budget ε
  end
end
return all perturbed windows

5.2.3 White-Box Attacks

Fast Gradient Sign Method (FGSM)

FGSM was introduced by Goodfellow et al. [13] as a single-step adversarial attack that linearizes the loss surface around the clean input and takes one step in the sign of the gradient direction. In the evasion setting, the step is taken in the *negative* gradient direction to minimize the loss:

$$\tilde{\mathbf{X}}^{(t)} = \mathbf{X}^{(t)} - \varepsilon \cdot \operatorname{sign}(\nabla_{\mathbf{X}^{(t)}} \mathcal{L}_{\text{adv}}) \tag{5.6}$$

where ε is the ℓ_∞ perturbation budget (Equation (2.13)) and $\nabla_{\mathbf{X}^{(t)}}\mathcal{L}_{\text{adv}}$ is the gradient of the adversarial loss with respect to the clean input window. The sign function ensures that every element of the perturbation takes value in $\{-\varepsilon, +\varepsilon\}$, so that the ℓ_∞ constraint is saturated in a single step.

FGSM requires only one forward pass and one backward pass through the model, making it computationally inexpensive. However, because it relies on a single linear approximation of the loss landscape, it may not find the most effective perturbation when the loss surface is highly curved --- a known limitation that motivates the iterative extension described below [14]. In this implementation, gradients are computed with the model placed in training mode to ensure that batch-normalization layers propagate gradients correctly.

Algorithm 4: Fast Gradient Sign Method (FGSM)

Input: Input window X, model f, perturbation budget ε

Output: Adversarial window $\tilde{X}$

1 Run the model on X to get predictions
2 Compute the prediction error (loss)
3 Compute the gradient of the loss with respect to X
4 Perturb every input feature by ε in the direction that reduces the loss
5 **return** perturbed window $\tilde{X}$

Projected Gradient Descent (PGD)

PGD, introduced by Madry et al. [14], is an iterative multi-step extension of FGSM that applies K successive gradient steps, each of size α, followed by a projection back onto the ℓ_∞ ball of radius ε centred at the clean input. The update rule at iteration k is:

$$\tilde{\mathbf{X}}^{(t)}_{k+1} = \Pi_\varepsilon\Big(\tilde{\mathbf{X}}^{(t)}_{k} - \alpha \cdot \operatorname{sign}\Big(\nabla_{\tilde{\mathbf{X}}^{(t)}_{k}}\mathcal{L}_{\text{adv}}\Big)\Big) \tag{5.7}$$

where α is the per-step size and Π_ε is the projection operator

$$\Pi_\varepsilon\Big(\tilde{\mathbf{X}}^{(t)}\Big) = \mathbf{X}^{(t)} + \operatorname{clip}\Big(\tilde{\mathbf{X}}^{(t)} - \mathbf{X}^{(t)},\ -\varepsilon,\ +\varepsilon\Big) \tag{5.8}$$

that ensures the perturbed input never exceeds the ℓ_∞ budget. The per-step size is set to $\alpha = 2\varepsilon/K$, which allows the cumulative drift to explore the full ε-ball while remaining numerically stable. PGD is initialized from a random starting point drawn uniformly inside the ε-ball to reduce sensitivity to the local geometry of the loss surface [14].

Algorithm 5: Projected Gradient Descent (PGD)

Input: Input window X, model f, budget ε, step size α, number of iterations K
Output: Best adversarial window found

Start from a random perturbation inside the allowed budget
Record the best result seen so far
for *each iteration* $k = 1$ **to** K **do**
    Run the model on the current perturbed window
    Compute the prediction error and its gradient
    Take one step in the direction that reduces the error
    Clip the perturbation so it never exceeds the budget ε
    if *this step produced a lower anomaly score than the best so far* **then**
        Save this as the new best result
    end
end
return the best adversarial window found across all iterations

5.3 Shared Evasion Objective

Regardless of the knowledge level, all attacks in this thesis share the same evasion goal: drive the model's forecast error on the perturbed input toward zero so that the anomaly score falls below the detection threshold. Formally, the attacks minimize the adversarial evasion loss

$$\mathcal{L}_{\text{adv}}\big(\tilde{\mathbf{X}}^{(t)}\big) = \frac{1}{N}\sum_{i=1}^{N}\big|\hat{y}_t^{(i)} - x_t^{(i)}\big| \tag{5.9}$$

where $\tilde{\mathbf{X}}^{(t)}$ is the perturbed sliding window (Equation (2.3)), $\hat{y}_t^{(i)}$ is the model's one-step-ahead prediction for device i, and $x_t^{(i)}$ is the ground-truth observation. Driving $\mathcal{L}_{\text{adv}} \to 0$ forces the model's predictions to match the actual sensor values, reducing the per-device forecast error $\xi_{i,t}$ (Equation (2.4)) and, consequently, the aggregated anomaly score $A^{(t)}$ (Equation (2.5)) below the threshold λ (Equation (2.10)). For MTAD-GAT, perturbing the input also implicitly affects the reconstruction term of the hybrid score (Equation (2.8)), so targeting the forecast loss alone is sufficient to suppress the combined score. The black-box attacks (NES and SimBA) optimize the same objective but access it only through scalar score queries rather than gradients.

5.4 Attack Budget and Experimental Parameter Grid

All attacks are constrained by the ℓ_∞ budget ε defined in Equation (2.13), which limits the maximum absolute perturbation applied to any device channel at any time step. Perturbations are applied in the normalized feature space used as model input, so ε is dimensionless and directly comparable across all sensor channels regardless of their original physical scale. The full parameter grids used in the experiments are summarized in Table 5.1.

Table 5.1: Attack parameters used in the adversarial evaluation.

Attack	**Parameter**	**Symbol**	**Values**
FGSM	Perturbation budget	ε	$\{0.001, 0.002, 0.005, 0.010, 0.020\}$
PGD	Perturbation budget	ε	$\{0.001, 0.002, 0.005, 0.010, 0.020\}$
	Step size	α	$2\varepsilon/K$
	Iterations	K	10
	Random start	---	Yes
BETA	Perturbation budget	ε	$\{0.002, 0.005, 0.020\}$
	Step size	α	$2\varepsilon/K$
	Iterations	K	10
	Influencer budget	B	$\{3, 5\}$
	Random start	---	Yes
NES	Perturbation budget	ε	$\{0.001, 0.005\}$
	Step size	α	0.01
	Iterations	K	20
	Antithetic pairs	S	10
	Smoothing radius	σ	0.001
	Random start	---	Yes
SimBA	Perturbation budget	ε	$\{0.001, 0.005\}$
	Direction budget	Q	200

The ε range is kept deliberately conservative (0.001--0.020 in normalized space) to ensure that perturbations remain imperceptible in physical terms while still probing a meaningful vulnerability gradient. For BETA, the target node u is chosen as the sensor with the highest clean anomaly score in the Actor 2 test set, and the influencer budget B is varied to measure the trade-off between stealthiness and attack effectiveness.

5.5 Attack Evaluation Metrics

Evaluating adversarial attacks on an anomaly detector requires metrics that capture both the detector's residual effectiveness and the attacker's net gain relative to a clean baseline. Accordingly, two complementary metrics are used throughout this chapter.

5.5.1 Adversarial Detection Rate

The adversarial detection rate $\mathrm{DR}_{\mathrm{adv}}$ is the same quantity computed after the attack has been applied to the test inputs:

$$\mathrm{DR}_{\mathrm{adv}} = \frac{\left|\{t \in \mathcal{T}_{\mathrm{anom}} : A^{(t)}(\tilde{\mathbf{X}}^{(t)}) > \lambda\}\right|}{|\mathcal{T}_{\mathrm{anom}}|} \tag{5.10}$$

where $A^{(t)}(\tilde{\mathbf{X}}^{(t)})$ denotes the anomaly score recomputed on the perturbed input $\tilde{\mathbf{X}}^{(t)}$ with the same fixed threshold λ. A lower $\mathrm{DR}_{\mathrm{adv}}$ indicates that the attack has successfully caused the model to miss more anomalies. The metric is computed point-by-point across all anomalous windows in the Actor 2 test partition and then averaged.

5.5.2 Attack Success Rate

The attack success rate ASR quantifies the attacker's net gain as the absolute reduction in detection rate caused by the attack:

$$\mathrm{ASR} = \mathrm{DR}_{\mathrm{clean}} - \mathrm{DR}_{\mathrm{adv}} \tag{5.11}$$

$\mathrm{ASR} = 0$ means the attack had no effect; $\mathrm{ASR} = \mathrm{DR}_{\mathrm{clean}}$ means every previously detected anomaly is now missed. Using an absolute difference rather than a relative ratio makes results directly comparable across attack families and budget levels, since $\mathrm{DR}_{\mathrm{clean}}$ is identical for both models. A higher ASR indicates a more effective attack. All results in this chapter are reported at the τ_{90} threshold.

5.6 Experimental Setup

5.6.1 Dataset Usage

All adversarial attacks are applied exclusively to the Actor 2 test partition of the CUMaster dataset (see Table 3.3 in Chapter 3). The training and validation partitions are never modified; they are used only to load the pre-trained model checkpoints and to recover the threshold λ calibrated in Chapter 4. This design ensures that the threat model is strictly evasion-based, as formalized in Chapter 2: model weights and detection thresholds are fixed, and only the test-time inputs are perturbed.

5.6.2 Evaluation Protocol

For each attack configuration, the perturbed test set $\tilde{\mathbf{X}}$ is constructed by applying the corresponding attack independently to each sliding window in the Actor 2 partition. The anomaly

score $A^{(t)}$ is then recomputed on the perturbed inputs using the same scoring pipeline described in Chapter 4: raw errors are normalized with training-split statistics, aggregated by the maximum operator, and smoothed with a moving average of window $\omega = 5$. The three metrics defined in Section 5.5 ($\mathrm{DR}_{\mathrm{clean}}$, $\mathrm{DR}_{\mathrm{adv}}$, and ASR) are then computed over all anomalous time steps in the test partition.

5.7 Black-Box Attack Results

Table 5.2 reports results for NES and SimBA across both models and two ε values.

Table 5.2: NES and SimBA results for GDN and MTAD-GAT. $\mathrm{DR}_{\mathrm{clean}} = 79.00\%$ for all configurations.

Model	**Attack**	ε	$\mathrm{DR}_{\mathrm{adv}}$ **(%)**	**ASR (%)**
GDN	NES	0.001	77.53	1.47
		0.005	73.10	5.90
GDN	SimBA	0.001	77.67	1.23
		0.005	74.85	4.15
MTAD-GAT	NES	0.001	77.48	1.42
		0.005	73.87	5.03
MTAD-GAT	SimBA	0.001	77.74	1.26
		0.005	75.11	3.89

5.7.1 NES

NES achieves an ASR of 5.90% against GDN and 5.03% against MTAD-GAT at $\varepsilon = 0.005$. These figures are notable for a black-box attack: NES with 400 model queries per batch achieves performance comparable to white-box FGSM at the same budget (GDN: 5.80% ASR at $\varepsilon = 0.005$). This indicates that the loss surface of both models is sufficiently smooth that antithetic finite-difference estimation recovers a useful descent direction even without true gradients. GDN is again slightly more vulnerable than MTAD-GAT (5.90% vs. 5.03% at $\varepsilon = 0.005$), consistent with the white-box results.

5.7.2 SimBA

SimBA achieves a lower ASR than NES at both ε levels for both models. On GDN, SimBA reaches 4.15% at $\varepsilon = 0.005$ versus NES's 5.90%. On MTAD-GAT, SimBA achieves 3.89% versus NES's 5.03% at the same budget. The difference reflects SimBA's purely axis-aligned random search: with a direction budget of $Q = 200$ in a $W \times N$-dimensional space, SimBA explores a small fraction of the perturbation space per batch, and many proposed directions are rejected. MTAD-GAT's dual-stream architecture further reduces SimBA's effectiveness

because the temporal convolution branch compensates for perturbations in individual sensor channels, making axis-aligned steps less impactful than the coordinated multi-channel perturbations that NES naturally produces.

5.8 Grey-Box Attack Results

5.8.1 BETA

Table 5.3 reports BETA results for both models across three ε values and two influencer budgets $B \in \{3, 5\}$.

Table 5.3: BETA results for GDN and MTAD-GAT. $\mathrm{DR}_{\mathrm{clean}} = 79.00\%$ for all configurations.

Model	B	ε	$\mathrm{DR}_{\mathrm{adv}}$ **(%)**	**ASR (%)**
GDN	5	0.002	76.90	2.10
		0.005	74.20	4.80
		0.020	63.60	15.40
GDN	3	0.002	77.65	1.35
		0.005	75.80	3.20
		0.020	67.45	11.55
MTAD-GAT	5	0.002	76.20	2.80
		0.005	72.90	6.10
		0.020	60.20	18.80
MTAD-GAT	3	0.002	76.60	2.40
		0.005	73.85	5.15
		0.020	62.10	16.90

Several observations stand out. First, MTAD-GAT is notably *more* vulnerable to BETA than GDN at the same budget. With $B = 5$ and $\varepsilon = 0.020$, BETA achieves an ASR of 18.80% against MTAD-GAT versus 15.40% against GDN --- a reversal of the pattern observed under FGSM and PGD. This reversal is explained by the influencer selection mechanism: for MTAD-GAT, influencers are chosen using the feature attention weights $\alpha_{u,j}$, which directly reflect the model's relational dependencies. Perturbing nodes that the target node attends to most strongly is a precisely targeted intervention that proves more effective than perturbing all nodes simultaneously at a smaller effective per-node budget.

Second, the influencer budget B has a meaningful effect on GDN but a smaller one on MTAD-GAT. Increasing B from 3 to 5 raises the ASR on GDN by 3.85 points at $\varepsilon = 0.020$ (11.55% $\rightarrow$ 15.40%), but only by 1.90 points on MTAD-GAT (16.90% $\rightarrow$ 18.80%). This suggests that MTAD-GAT's attention structure concentrates influence on a small number of highly weighted neighbors, so the marginal value of adding more influencers diminishes quickly.

Third, comparing BETA to PGD at $\varepsilon = 0.020$: BETA with $B = 5$ reaches 15.40% ASR on GDN versus PGD's 23.60%, and 18.80% on MTAD-GAT versus PGD's 20.10%. The gap is large for GDN but narrow for MTAD-GAT, confirming that BETA's targeted strategy is near-optimal when the graph structure reliably identifies the most influential neighbors.

5.9 White-Box Attack Results

Table 5.4 reports FGSM and PGD results for GDN and MTAD-GAT across the five ε values. Both models show a consistent and monotonic degradation in $\mathrm{DR}_{\mathrm{adv}}$ as ε increases, confirming that FGSM effectively exploits gradient information even in a single step.

Table 5.4: FGSM and PGD results for GDN and MTAD-GAT. $\mathrm{DR}_{\mathrm{clean}} = 79.00\%$ for all configurations.

Model	**Attack**	ε	$\mathrm{DR}_{\mathrm{adv}}$ **(%)**	**ASR (%)**
GDN	FGSM	0.001	77.85	1.15
		0.002	76.40	2.60
		0.005	73.20	5.80
		0.010	68.55	10.45
		0.020	62.12	16.88
GDN	PGD	0.001	77.10	1.90
		0.002	75.25	3.75
		0.005	70.80	8.20
		0.010	64.15	14.85
		0.020	55.40	23.60
MTAD-GAT	FGSM	0.001	78.10	0.90
		0.002	76.95	2.05
		0.005	74.30	4.70
		0.010	70.15	8.85
		0.020	63.80	15.20
MTAD-GAT	PGD	0.001	77.45	1.55
		0.002	75.80	3.20
		0.005	72.15	6.85
		0.010	66.40	12.60
		0.020	58.90	20.10

5.9.1 FGSM

For GDN, FGSM achieves an ASR of 16.88% at $\varepsilon = 0.020$, reducing detection from 79.00% to 62.12%. MTAD-GAT is marginally more resilient at the same budget, retaining a $\mathrm{DR}_{\mathrm{adv}}$ of 63.80% (ASR 15.20%). The difference is consistent across all ε values and reflects MTAD-GAT's dual-stream architecture: the temporal convolution branch provides a secondary representation of the input that FGSM's single-step perturbation does not fully suppress.

5.9.2 PGD

PGD consistently outperforms FGSM at every ε level for both models, as expected from the iterative nature of the attack. At $\varepsilon = 0.020$, PGD achieves an ASR of 23.60% against GDN ($\mathrm{DR}_{\mathrm{adv}}$ = 55.40%) and 20.10% against MTAD-GAT ($\mathrm{DR}_{\mathrm{adv}}$ = 58.90%). The gap between FGSM and PGD widens with ε: at $\varepsilon = 0.001$ the difference is less than 1 percentage point, but at $\varepsilon = 0.020$ PGD exceeds FGSM by 6.72 points on GDN and 4.90 points on MTAD-GAT. This pattern is characteristic of curved loss surfaces where multi-step optimization provides a meaningful advantage over the single-step linear approximation.

GDN is more vulnerable to PGD than MTAD-GAT across all budgets. At $\varepsilon = 0.020$, the gap is 3.50 ASR points (23.60% vs. 20.10%). This is consistent with GDN's simpler, single-stream architecture: once the graph attention mechanism is misled by a sufficiently large perturbation, there is no secondary representational pathway to fall back on.

5.10 Comparative Vulnerability Analysis

5.10.1 Attack Effectiveness Across Knowledge Levels

Table 5.5 consolidates the peak ASR achieved by each attack at its highest tested ε value to facilitate direct comparison across knowledge levels.

Table 5.5: Peak ASR for each attack and model at the highest tested ε.

Attack	Level	ε	GDN ASR (%)	MTAD-GAT ASR (%)
NES	Black-box	0.005	5.90	5.03
SimBA	Black-box	0.005	4.15	3.89
BETA ($B = 5$)	Grey-box	0.020	15.40	18.80
FGSM	White-box	0.020	16.88	15.20
PGD	White-box	0.020	23.60	20.10

The results follow the expected ordering across knowledge levels: white-box attacks are the most effective, black-box attacks are the least effective, and BETA occupies an intermediate position.

BETA challenges the white-box upper bound for MTAD-GAT. At $\varepsilon = 0.020$, BETA with $B = 5$ achieves 18.80% ASR against MTAD-GAT, compared to FGSM's 15.20%. BETA is a grey-box attack with access to only $B = 5$ nodes, yet it outperforms a white-box attack that perturbs all N nodes simultaneously. This is explained by the attention-guided influencer selection: BETA concentrates its budget precisely where the model is most sensitive, achieving a higher achieving more concentrated and effective perturbations than FGSM, which distributes perturbations uniformly across all channels. This finding has an important implication: for attention-based architectures, targeted grey-box attacks that exploit the learned relational structure can be more dangerous than unrestricted white-box attacks, because the structure itself reveals the attack surface.

NES is a meaningful threat despite zero model knowledge. At $\varepsilon = 0.005$, NES reaches 5.90% ASR on GDN with 400 forward-pass queries per batch, matching the performance of white-box FGSM at the same budget. This confirms that the anomaly scoring functions of both GDN and MTAD-GAT are smooth enough for finite-difference gradient estimation to be viable, and that the models cannot be considered secure against a patient query-based adversary.

SimBA reveals the limits of axis-aligned perturbations. SimBA is less effective than NES at both ε levels for both models, achieving 4.15% ASR on GDN and 3.89% on MTAD-GAT at $\varepsilon = 0.005$. Since SimBA perturbs one sensor channel at a time, its limited effectiveness reflects the model's cross-sensor and temporal redundancy: a single-channel perturbation is partially absorbed, and the anomaly score does not drop as readily as it does under NES's coordinated multi-channel perturbations.

5.10.2 GDN vs. MTAD-GAT Robustness

Taken together, the results reveal a consistent but nuanced robustness profile for the two architectures. Under white-box attacks, GDN is more vulnerable than MTAD-GAT at every ε level, with PGD at $\varepsilon = 0.020$ producing a 3.50 ASR-point gap (23.60% vs. 20.10%). Under grey-box attacks, the ranking reverses: MTAD-GAT is more vulnerable to BETA because its attention weights directly expose its relational dependencies to the influencer selection mechanism. Under black-box attacks, GDN is again slightly more vulnerable to NES (5.90% vs. 5.03%) and to SimBA (4.15% vs. 3.89%).

This pattern suggests that MTAD-GAT architecture provides genuine robustness against unstructured or single-channel perturbations, but introduces a structural vulnerability when the attacker can exploit the graph attention mechanism. GDN, by contrast, is uniformly more susceptible to gradient-based attacks but does not expose a clear structural weakness to attention-based targeting. From a deployment perspective, the threat model matters greatly: against a white-box or random black-box adversary, MTAD-GAT is the more robust choice, against a grey-box adversary with knowledge of the graph topology, GDN may in fact be the

safer option.

Neither model achieves what would be considered strong adversarial robustness: even the most robust evaluated configuration (MTAD-GAT under SimBA at $\varepsilon = 0.005$) shows an ASR of 3.89%, meaning that a purely random black-box attacker with no model knowledge can suppress detection for nearly one in twenty-five anomalous windows using a very small perturbation budget. This underscores the need for adversarial training or certified defense mechanisms in any deployment of GNN-based anomaly detection for smart home security.

Conclusion

Summary of the Study and Contributions

This thesis presented a comparative empirical study of adversarial vulnerability in two GNN-based anomaly detection models (Graph Deviation Network (GDN) and MTAD-GAT) applied to multivariate time-series data from a real smart home environment. The study was motivated by a gap in the literature: while GNN-based detectors have shown strong performance under clean conditions, their behavior under adversarial manipulation in cyber-physical settings had not been systematically characterized, especially across multiple attacker knowledge levels.

The investigation was organized around three threat levels. At the white-box level, FGSM and PGD were applied with full gradient access. At the grey-box level, BETA exploited the learned graph structure to concentrate perturbations on the most influential neighbor nodes of a designated target sensor. At the black-box level, NES estimated pseudo-gradients from antithetic score queries, while SimBA performed greedy axis-aligned random search using no model knowledge beyond scalar score observations. All five attacks shared a common evasion objective, and were evaluated using attack success rate as the primary metric.

The principal findings are summarized as follows. Both GDN and MTAD-GAT are vulnerable to adversarial perturbations even under small perturbation budgets, indicating that neither architecture provides inherent robustness against evasion attacks. Across white-box attacks, GDN is consistently more vulnerable than MTAD-GAT, PGD achieving roughly 24% attack success against GDN versus 20% against MTAD-GAT at $\varepsilon = 0.020$. However, this ranking reverses under grey-box attacks: BETA with $B = 5$ influencers achieves 18.80% against MTAD-GAT but only 15.40% against GDN at the same budget, outperforming FGSM on MTAD-GAT. Among black-box attacks, NES proves to be a serious threat despite requiring no model internals, reaching 5.90% ASR against GDN and 5.03% against MTAD-GAT at $\varepsilon = 0.005$ --- matching the performance of white-box FGSM at the same budget. SimBA, the most restricted attacker, is less effective than NES at both budget levels, achieving 4.15% ASR on GDN and 3.89% on MTAD-GAT at $\varepsilon = 0.005$, an asymmetry attributable to MTAD-GAT's temporal convolution branch absorbing single-channel perturbations.

These results make several contributions to the literature. First, they establish a comparative adversarial robustness benchmark for GDN and MTAD-GAT on a real smart home dataset, providing reference numbers for future defense work. Second, they demonstrate empirically that grey-box attacks exploiting learned graph topology can be more dangerous than unrestricted white-box attacks against attention-based architectures, a finding with direct im-

plications for threat modeling. Third, they show that the two models' architectural differences produce meaningfully different vulnerability profiles rather than uniform fragility, and that which model is considered more robust depends on the assumed threat level.

Direct Answers to Research Questions

RQ1 - How vulnerable are GNN-based anomaly detection models to adversarial perturbations in smart-home multivariate time-series data?

Both models are vulnerable. Even at the smallest tested budget ($\varepsilon = 0.001$), every attack reduces the detection rate of at least one model by more than one percentage point. At $\varepsilon = 0.020$, PGD suppresses GDN's detection rate from 79% to 55.40%, a reduction of nearly a quarter of the clean performance. Neither model can be considered adversarially robust under the tested conditions.

RQ2 - Which adversarial attack strategies are most effective in degrading anomaly detection performance?

PGD is the most effective overall, consistent with its status as the strongest first-order white-box attack. However, BETA is the most efficient attack relative to its information budget: it achieves near-PGD performance against MTAD-GAT while accessing only $B = 5$ nodes and requiring no full gradient computation over all channels. Among black-box attacks, NES consistently outperforms SimBA across both models.

RQ3 - Do different graph-based architectures exhibit different robustness characteristics under adversarial conditions?

Yes, and the difference is systematic rather than incidental. GDN is more vulnerable to gradient-based white-box and black-box attacks. MTAD-GAT is more vulnerable to graph-targeted grey-box attacks. This pattern is explained by their architectures: GDN's sparse single-objective design provides less redundancy against gradient descent, while MTAD-GAT's feature attention mechanism directly exposes its relational dependencies to the BETA influencer selection procedure.

RQ4 - To what extent do learned inter-device dependencies contribute to adversarial vulnerability?

For MTAD-GAT, the sensors selected as influencers by BETA account for a disproportionate share of the model's vulnerability. Perturbing only 5 out of all sensors achieves attack success rates close to perturbing all sensors simultaneously. For GDN, the eigenvector-central neighbors of the target node in the knn embedding graph play a similar concentrating role, though the effect is less pronounced because the embedding-based adjacency is less tightly coupled to instantaneous model sensitivity.

RQ5 - How does perturbation magnitude influence the degradation of anomaly detection reliability?

Degradation is monotonic in ε for all attacks and both models. The rate of degradation increases with ε for iterative attacks (PGD, BETA, NES) but saturates sooner for FGSM and SimBA. For PGD, the gap between GDN and MTAD-GAT grows with ε, suggesting that GDN's loss surface is more globally exploitable as the budget increases.

Limitations

Several limitations of this study should be acknowledged when interpreting the results.

Single dataset. All experiments use a single smart home dataset (IoT Garage, CUMaster partition). While this dataset is real, multi-week, and covers a diverse range of device types, the results may not generalize to smart home environments with different device compositions, behavioral rhythms, or anomaly types. Validation on additional datasets would strengthen the generalizability of the findings.

Evasion-only threat model. The study considers only test-time evasion attacks. Poisoning attacks, which corrupt the training data or the learned graph structure, are out of scope. In practice, an adversary with long-term access to the smart home network might pursue poisoning strategies that are more disruptive and harder to detect.

No adaptive defense. No adversarial training or certified defense was applied to either model. The reported results therefore represent the vulnerability of undefended detectors and do not assess whether standard defense techniques would be effective. This is intentional but it limits what can be said about how the models could be hardened.

Fixed target node for BETA. BETA was evaluated with a single target node chosen randomly. The choice of target node affects influencer selection and consequently, attack effectiveness. A more thorough evaluation would test BETA across multiple target nodes and report average and worst-case performance.

Limited ε range for NES and SimBA. The black-box attacks were evaluated at only two ε values due to the high query cost per batch. The degradation curves for these attacks are therefore less complete than those for the gradient-based methods, making it harder to assess whether their effectiveness plateaus or continues to grow at larger budgets.

Future Directions

Several directions emerge naturally from the findings and limitations of this thesis.

Adversarial training for GNN-based detectors. The most direct extension is to apply adversarial training (augmenting the training set with PGD or BETA adversarial examples) and to measure whether this improves robustness without significantly degrading clean detection performance. Given the results showing that MTAD-GAT's dual objective provides partial natural robustness against single-channel perturbations, it is worth investigating whether

adversarial training interacts differently with the two architectures.

Certified robustness. Randomized smoothing and interval bound propagation have been used to certify robustness of deep classifiers against ℓ_∞ perturbations. Extending these techniques to the anomaly scoring setting (where the output is a scalar score rather than a class label) is a non-trivial but tractable research direction.

Graph structure as an attack surface. This thesis restricted perturbations to node feature values. An important open question is how vulnerable the learned graph structure itself is: could an adversary manipulate device readings during the training phase to corrupt the sensor embeddings in GDN or the attention weights in MTAD-GAT, and would the resulting structural corruption be detectable? This bridges the evasion and poisoning threat models.

Broader attack coverage. The five attacks evaluated here cover the main families of evasion strategies, but several variants were not tested: transferability-based black-box attacks (using a surrogate model), patch-based attacks that corrupt a small contiguous time segment, and physically constrained attacks that respect the causal dynamics of specific device types. Each of these would probe a different aspect of the models' robustness.

Defense by graph topology hardening. The BETA result suggests that the learned graph structure is itself a vulnerability surface for attention-based models. One promising defense direction is to regularize the attention weights during training (for example, by penalizing high-concentration attention distributions) to reduce the advantage that an attacker gains from identifying high-attention neighbors. Whether such regularization degrades clean detection performance is an open empirical question.

References

[1] 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

[2] 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

[3] L. Sun et al., ``Adversarial attack and defense on graph data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[4] S. Tariq, B. M. Le, and S. S. Woo, ``Towards an awareness of time series anomaly detection models' adversarial vulnerability," *ACM Transactions on Privacy and Security*, 2022. [Online]. Available: https://arxiv.org/abs/2208.11264

[5] J. Ma, S. Ding, and Q. Mei, ``Practical adversarial attacks on graph neural networks," in *Proceedings of the 37th International Conference on Machine Learning*, ser. PMLR, 2020.

[6] S. Geisler, T. Schmidt, H. Sirin, D. Zügner, A. Bojchevski, and S. Günnemann, ``Robustness of graph neural networks at scale," in *Advances in Neural Information Processing Systems*, 2021. [Online]. Available: https://arxiv.org/abs/2110.14038

[7] S. Xaviar and O. Ardakanian, *Budgeted adversarial attack against graph-based anomaly detection in sensor networks*, 2025. arXiv: 2509.17987 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2509.17987

[8] V. Medel, O. Chernov, and T. Kurniawati, ``Anomaly detection using forecasting methods arima and hwds," in *2017 IEEE 15th Student Conference on Research and Development (SCOReD)*, IEEE, 2017, pp. 6–10. [Online]. Available: https://ieeexplore.ieee.org/document/7814437

[9] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, ``A novel anomaly detection scheme based on principal component classifier," Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL, USA, Tech. Rep., 2003.

[10] 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

[11] 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

[12] 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

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, 2015. arXiv: 1412.6572 [cs.LG].

[14] A. Madry et al., *Towards deep learning models resistant to adversarial attacks*, 2018. arXiv: 1706.06083 [cs.LG].

[15] C. Szegedy et al., ``Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013. [Online]. Available: https://arxiv.org/abs/1312.6199

[16] H. Dai et al., *Adversarial attack on graph structured data*, 2018. arXiv: 1806.02371 [cs.LG].

[17] F. Karim et al., *Adversarial attacks on time series*, 2019. arXiv: 1902.10755 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1902.10755

[18] T. Wu et al., ``Understanding the robustness of graph neural networks against adversarial attacks," *arXiv preprint arXiv:2406.13920*, 2025. [Online]. Available: https://arxiv.org/abs/2406.13920

[19] Y. Majib, M. Alosaimi, A. Asaturyan, and C. Perera, *Dataset for cyber-physical anomaly detection in smart homes*, 2024.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT Press, 2016.

[21] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, ``Black-box adversarial attacks with limited queries and information," *arXiv preprint arXiv:1804.08598*, 2018. [Online]. Available: https://arxiv.org/abs/1804.08598

[22] C. Guo, J. R. Gardner, Y. You, A. G. Wilson, and K. Q. Weinberger, ``Simple black-box adversarial attacks," *arXiv preprint arXiv:1905.07121*, 2019. [Online]. Available: https://arxiv.org/abs/1905.07121