POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Efficacy of Detecting and Characterizing Anomalies Using Cortical Algorithms in Time Series Data

Supervisors

Prof. Paolo GARZA

Politecnico di Torino

Prof. Pulin AGRAWAL

Behrend College, Pennsylvania State University

Candidate

Fatemeh AHMADVAND

April 2025

Abstract

Detecting anomalies in time-series data is a critical challenge across numerous domains, including cybersecurity, healthcare, finance, and industrial monitoring. This thesis investigates the efficacy of cortical-inspired algorithms—specifically Hierarchical Temporal Memory (HTM) and Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT)—for univariate time-series anomaly detection. Traditional statistical and deep learning methods often struggle with data sparsity, non-stationary behavior, and computational overhead, especially in real-time scenarios. To address these limitations, HTM leverages biologically inspired mechanisms such as Sparse Distributed Representations (SDRs), spatial pooling, and temporal memory. These features enable continuous learning of evolving patterns and robust detection of irregular deviations in streaming data. However, its reliance on spatial correlations can reduce effectiveness for certain univariate tasks, highlighting a trade-off in settings where only a single feature is monitored. In parallel, SLADiT combines the representational power of autoencoders with the temporal modeling capabilities of recurrent LSTM layers. By enforcing sparsity in the latent space, SLADiT captures essential features of normal time-series dynamics, thus enhancing the ability to distinguish anomalies. Unlike HTM, SLADiT's batch-style training and reconstruction error–based detection can yield higher accuracy for subtle, point-based deviations in univariate data, but it is less suited for direct deployment in continuous monitoring without periodic retraining. A series of experiments on benchmark datasets from the UCR Anomaly Archive and NYC Taxi from NAB collection evaluate both models under controlled conditions, using metrics such as F1-score and Area Under the Curve (AUC) and Accuracy. Results indicate that HTM excels at capturing sequential dependencies and adapting to new patterns over time, benefiting real-time contexts. Conversely, SLADiT demonstrates strong reconstruction capabilities and a lower false-positive rate in univariate scenarios, though it requires re-initialization or online fine-tuning for non-stationary data streams. Overall, this thesis contributes a comparative perspective on cortical-inspired approaches to anomaly detection, illustrating how biological mechanisms and sparsity constraints can inform scalable solutions. These findings underscore the importance of aligning model selection with domain-specific constraints, including data dimensionality, real-time requirements, and tolerance for false alarms. The insights gleaned offer guidance for practitioners seeking to implement or refine anomaly detection strategies, paving the way for further exploration of brain-inspired architectures in diverse temporal analytics applications.

Summary

This thesis focuses on the challenge of detecting anomalies in univariate time-series data and equally emphasizes identifying anomalies alongside describing their severity, duration, and nature. The task is increasingly relevant in areas like finance, industrial maintenance, healthcare, and cybersecurity, where time-series signals guide critical decisions. Compared to static datasets, time-series data arrives sequentially, with each new observation influenced by preceding points, making small deviations potentially indicative of serious issues such as failures or fraud. This project, therefore, examines and contrasts two distinct methods grounded in different theoretical frameworks— **Hierarchical Temporal Memory (HTM)** which is inspired by cortical learning algorithms and excels at online sequence prediction, against **Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT)**, which imposes sparsity constraints on a deep autoencoder architecture to reconstruct normal patterns and flag deviations.

Time-series data introduces unique challenges because observations arrive in sequence and depend on prior values, requiring specialized methods that handle temporal correlations, as well as seasonality, trends, and abrupt changes. Traditional statistical techniques such as ARIMA-based forecasting or moving averages can struggle with sudden shifts or complex temporal patterns. Deep learning methods can capture non-linear relationships but often need extensive labeled data and high computational resources—prohibitive when anomalies are rare, labels are scarce, or rapid detection is critical. Consequently, this thesis explores HTM and SLADiT, which both offer distinct ways to learn from time-series data. A major challenge, however, is model tuning—HTM, for instance, has around 21 hyperparameters, and slight adjustments can significantly affect anomaly detection performance. HTM draws on cortical structures for continuous, unsupervised learning, while SLADiT leverages recurrent networks and sparsity constraints to build compact latent representations for reconstruction-based anomaly detection.

Hierarchical Temporal Memory (HTM) emulates how the neocortex processes sensory information, featuring two main components. The first, the Spatial Pooler (SP), transforms raw input into Sparse Distributed Representations (SDRs) by activating only those "mini-columns" that receive sufficient matching input bits,

thereby keeping a small fraction of columns active at any given time. The second, the Temporal Memory (TM), learns sequential transitions in data by forming dendritic segments that predict the next active set of neurons (or mini-columns). If the observed input differs from that prediction, the anomaly score increases. HTM naturally adapts to non-stationary environments because it continuously updates its learned representations in real time, making it particularly promising for streaming or industrial monitoring tasks where data patterns may shift. However, in univariate setups—where there is only a single variable evolving over time—the spatial correlations that HTM typically exploits can be less pronounced, sometimes diminishing its advantage.

However, in univariate contexts, HTM's reliance on spatial correlations can offer limited benefits, and it does not inherently learn seasonal or periodic structures. Experiments on UCR datasets showed that HTM struggles when data lacks time-based encoding, prompting the use of Fast Fourier Transform (FFT) to introduce time-based features. By concatenating synthetic sine and cosine dimensions, cyclical behaviors were captured even in datasets not explicitly time-stamped, preserving temporal structure and enhancing HTM's ability to detect subtle or non-temporal anomalies.

Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT) adds a sparsity constraint to the classic autoencoder design while leveraging Long Short-Term Memory (LSTM) layers to capture temporal dependencies. Its pipeline begins with an encoder composed of multiple stacked LSTM layers that compress a sequence of observations into a lower-dimensional latent vector. The latent space then imposes a sparsity constraint, often enforced through Kullback-Leibler divergence, which ensures that only a subset of latent neurons remain active and pushes the model to learn the most salient features of normal behavior. Finally, a decoder reconstructs the original time-series from this latent representation, and the reconstruction error signals how much an input deviates from the learned normal patterns, indicating an anomaly if the error exceeds a certain threshold. This approach generally excels at capturing subtle point anomalies in univariate time-series, aided by sparsity that reduces noise and overfitting. However, SLADiT often requires periodic retraining when patterns shift significantly, making continuous deployment more challenging in real-time applications.

This thesis employs the UCR Time-Series Anomaly Archive—widely used for benchmarking univariate anomaly detection—and the NYC Taxi dataset from the Numenta Anomaly Benchmark (NAB) for near real-time testing. Each dataset is split into training (mostly normal data) and testing (with known anomalies). Both HTM and SLADiT undergo hyperparameter tuning to maximize performance. For HTM, parameters include column count, cells per column, and permanence thresholds; for SLADiT, layer depth, hidden dimensions, and sparsity factors are refined. Evaluation relies on precision, recall, the F1-score (which balances false

positives and negatives), and the Area Under the ROC Curve (AUC), measuring the trade-off between correctly identifying anomalies and avoiding false alarms.

When balancing accuracy and real-time adaptability, HTM's online learning provides an edge for streaming applications, as it dispenses with a separate retraining phase. However, fine-tuning HTM is highly sensitive, and in purely univariate contexts its performance can sometimes lag behind SLADiT. Meanwhile, SLADiT offers robust reconstruction-based detection of point anomalies, often achieving high recall and lower false-positive rates—but its lack of built-in continuous adaptation can hamper real-time responsiveness. Both methods benefit from added data dimensionality: HTM draws on its Spatial Pooler more effectively with multiple correlated signals, while SLADiT can leverage deeper representations in both univariate and multivariate cases. In practical terms, SLADiT's minibatch gradient-based training often runs efficiently on GPUs but requires retraining. HTM can operate continuously on CPUs but grows computationally heavier if configured with many columns and cells.

Through systematic experiments on representative univariate time-series datasets, this thesis demonstrates that both HTM and SLADiT can excel at detecting anomalies, but their suitability depends on the application context. HTM's biologically inspired, continuous learning suits scenarios where data patterns evolve unpredictably, provided its numerous hyperparameters are properly tuned. SLADiT's reconstruction-based framework, enhanced by latent sparsity, yields strong detection performance in more stable conditions. The choice between these approaches depends on factors like tolerance for false alarms, adaptability to shifting processes, computational resources, and the nature of anomalies.

Future research could investigate hybrid or ensemble strategies that leverage the complementary strengths of HTM and SLADiT in real-time contexts, further enhancing the reliability and efficiency of anomaly detection in high-stakes environments. Transitioning from a multivariate, time-aware HTM model to a univariate implementation introduced significant challenges, as the model's effectiveness be- came heavily dependent on dataset characteristics. Low-noise datasets with clear periodic structures allowed for reasonable detection performance, whereas abrupt anomalies and non-stationary trends proved more problematic. The removal of contextual cues, particularly those derived from timestamps or external features, significantly reduced HTM's ability to infer meaningful predictions. Future improvements should focus on reintroducing time-based encodings, refining hyperparameters, and exploring hybrid methods that integrate HTM's continuous learning with more structured encoding strategies, thereby enhancing generalizability and adaptability for real-world univariate time-series anomaly detection.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Prof. Paolo Garza from Politecnico di Torino and Prof. Pulin Agrawal from Behrend College, Pennsylvania State University, for their unwavering guidance, support, and insightful feedback throughout this journey. Their expertise and encouragement have been instrumental in the completion of this dissertation.

I am also profoundly thankful to my family for their constant love, understanding, and support, which provided me with the strength to pursue my academic goals. Their encouragement was essential during the most challenging moments of my studies.

Finally, I extend my heartfelt thanks to my friends for their continuous moral support and for sharing in the ups and downs of this process. Their companionship made this academic journey both enriching and memorable. This work stands as a testament to the support of all those mentioned above, to whom I am sincerely indebted.

Fatemeh Ahmadvand

Table of Contents

List of Tables

List of Figures

Acronyms

HTM
Hierarchical Temporal Memory

SLADiT
Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series

SAE
Sparse AutoEncoder

AD
Anomaly Detection

CLAs
Cortical Learning Algorithms

RDSE
Random Distributed Scalar Encoder

SDRs
Sparse Distributed Representations

TM
Temporal Memory

SP
Spatial Pooler

LSTM
Long Short-Term Memory

GMMs

Gaussian Mixture Models

SVMs

Support Vector Machines

VAEs

Variational Autoencoders

KL

Kullback-Leibler

MSE

Mean Squared Error

ML

Machine Learning

AUC

Area Under Curve

ARIMA

AutoRegressive Integrated Moving Average

P

Percision

R

Recall

TP

True Positive

FN

False Negative

FP

False Positive

TN

True Negative

FFT

Fast Fourier Transform

ABP

Arterial Blood Pressure

SHAP

SHapley Additive exPlanations

LIME

Local Interpretable Model-agnostic Explanations

NAB

Numenta Anomaly Benchmark

UCR

UCR Anomaly Archive

t-SNE

t-distributed Stochastic Neighbor Embedding

STL

Seasonal and Trend decomposition using Loess

Chapter 1

Introduction

1.1 Background and Motivation

The task of detecting anomalies has been a subject of extensive research, driven by the need to identify rare, significant, and potentially dangerous deviations in data. In today's highly interconnected digital landscape, the sheer volume of data being generated and exchanged far surpasses human analytical capabilities, necessitating the development of automated anomaly detection methods. Anomaly detection, a critical aspect of data analysis, focuses on identifying data points that significantly deviate from the underlying distribution of a dataset, often indicating unusual, unexpected, or suspicious behaviors[1].

Anomaly detection has applications in a diverse range of domains, from cybersecurity and fraud detection to industrial monitoring and healthcare analytics [2]. While certain anomaly detection techniques are generalizable across multiple fields, many are custom-tailored to specific application areas.

There are multiple approaches to anomaly detection, including classification-based, nearest neighbor, clustering, statistical, spectral, information-theoretic, and graph-based techniques. Selecting the most suitable anomaly detection (AD) algorithm depends on several factors:

- **Nature of the input data**: Structured vs. unstructured, real-time vs. historical.
- **Type of anomalies**: Anomaly points, which can include extreme values, sudden shifts, or unexpected patterns.
- **Expected output**: Binary classification, anomaly scores, or ranking-based.
- **Domain knowledge**: Availability of expert insights for supervised or unsupervised learning methods[3].

1.2 Problem Statement

Anomaly detection in time-series data is a critical challenge across multiple domains, including cybersecurity, industrial monitoring, healthcare, IoT systems, and financial fraud detection. Identifying unexpected deviations or anomalies in a sequence of time-dependent values is essential for preventing failures, detecting fraud, and ensuring system reliability. However, existing anomaly detection methods face several key limitations, particularly when applied to univariate time-series data.

Traditional approaches such as statistical models (e.g., ARIMA, moving averages), clustering algorithms, and autoencoders have been widely used for anomaly detection. However, these methods often struggle with: A significant challenge in anomaly detection arises from two primary concerns: data sparsity and the need for real-time processing.

1. Data sparsity occurs in high-dimensional datasets, where only a small fraction of the data is relevant for identifying anomalies. This is prevalent in fields such as genomics, where complex interactions must be inferred from limited observations, making traditional machine learning approaches inefficient and computationally expensive[4].

2. Real-time processing is essential in applications such as autonomous vehicles, financial fraud detection, and industrial monitoring, where anomalies must be detected with minimal latency. Conventional deep learning models, despite their high accuracy, often suffer from high computational overhead, making them impractical for real-time applications[5].

To address these challenges, Cortical Learning Algorithms (CLAs) have emerged as a promising alternative. Inspired by the sparse and hierarchical structure of the neocortex, CLAs employ Sparse Distributed Representations (SDRs) to process information efficiently and robustly, even in sparse datasets. They leverage spatial and temporal sparsity for faster and more memory-efficient anomaly detection, incorporate predictive learning to dynamically model expected patterns, and offer energy efficiency, making them particularly suitable for low-power real-time applications[6].

The integration of CLAs for anomaly detection represents an exciting direction, particularly when compared to traditional methods like autoencoders, isolation forests, and statistical models. This thesis aims to explore the efficacy of Hierarchical Temporal Memory (HTM) in comparison to Sparse LSTM Autoencoder for Anomaly Detection in Time-Series (SLADiT), analyzing their ability to detect anomalies in univariate and multivariate time-series datasets while balancing accuracy, computational efficiency, and real-time applicability.

This thesis aims to explore the efficacy of Hierarchical Temporal Memory (HTM) in comparison to the Sparse LSTM Autoencoder for Anomaly Detection in

Time-Series (SLADiT). The study will analyze their ability to detect anomalies in univariate time-series data, while the multivariate analysis was included to demonstrate that cortical algorithms, such as HTM, perform more effectively on complex time-series data—where temporal behavioral patterns serve as an additional feature—compared to their performance on strictly univariate data. Specifically, the research will:

1. Assess the performance of CLAs in identifying anomaly points compared to SLADiT.
2. Investigate their adaptability to real-world applications by applying them to UCR and NAB datasets.
3. Determine the computational efficiency of CLAs for real-time streaming data applications.

1.3 Objectives and Scope of the Study

This study aims to investigate the effectiveness of Cortical Learning Algorithms (CLAs) for anomaly detection in time-series datasets, with a primary focus on univariate data. The multivariate analysis is included to demonstrate that cortical algorithms, such as HTM, perform more effectively on complex time-series data—where temporal behavioral patterns serve as an additional feature—compared to strictly univariate scenarios. Traditional anomaly detection methods often struggle with data sparsity, adaptability, and computational efficiency, particularly in real-time applications. This research explores whether CLAs can address these challenges and offer a viable alternative to conventional techniques.

The specific objectives of this study are as follows:

1. **Assess the effectiveness of CLAs for anomaly detection in univariate time series.**
   This study evaluates the ability of CLAs to detect anomaly points in univariate time-series datasets, identifying data points that significantly deviate from expected trends. The analysis will focus on issues such as data sparsity and robustness to noise. An extended evaluation on multivariate data is also conducted to illustrate the improved performance of cortical algorithms in capturing complex temporal behavioral patterns.
2. **Compare CLAs with existing anomaly detection techniques.**
   The research will benchmark CLAs against deep learning-based anomaly detection method. The comparison will assess key metrics such as detection accuracy, false positive rates, computational efficiency, and adaptability to unseen anomalies.

3. **Evaluate the adaptability of CLAs to real-world applications.**
   Since anomaly detection is widely used in domains such as cybersecurity, industrial monitoring, healthcare, IoT data monitoring, power consumption analysis, and fuel theft detection, this study will examine whether CLAs can meet the requirements for real-time anomaly detection.

By achieving these objectives, this study aims to provide insights into the practical advantages and limitations of CLAs for anomaly detection, determining their feasibility for real-world deployment in various industries.

1.4 Thesis Overview

This thesis is structured to provide a comprehensive analysis of Cortical Learning Algorithms (CLAs) for anomaly detection in univariate and multivariate time-series data. The thesis is organized to progressively establish a strong theoretical foundation, followed by a detailed methodological framework, experimental evaluation, and comprehensive analysis of findings.

- **Chapter 2: Fundamentals of Time-Series and Anomaly Detection**
  This chapter provides an overview of time-series analysis and anomaly detection techniques, discussing key concepts such as univariate versus multivariate time series, data sparsity, and common anomaly detection approaches. It also introduces the various categories of anomaly detection methods (supervised, semi-supervised, and unsupervised) and their applicability to real-world scenarios.

- **Chapter 3: Methodology**
  This chapter explores the application of cortical learning algorithms for anomaly detection, with a specific focus on Hierarchical Temporal Memory (HTM) and Sparse LSTM Autoencoder for Anomaly Detection in Time-Series (SLADiT). It details the underlying principles, architectural considerations, and computational efficiency of these models as well as their comparison with traditional techniques.

- **Chapter 4: Experimental Setup**
  This chapter details the datasets, preprocessing techniques, and evaluation metrics used in this study. The UCR and NAB time-series anomaly detection benchmarks are introduced as the primary datasets, along with a discussion on data labeling, feature engineering, and performance evaluation criteria.

- **Chapter 5: Results**
  This chapter presents the results of applying CLAs to anomaly detection tasks,

analyzing performance based on detection accuracy, false positive rates, and computational efficiency. It also includes visual representations of anomalies and model outputs to highlight key findings.

- **Chapter 6: Discussion and Comparative Analysis**
  This chapter provides a comparative discussion of CLAs versus traditional anomaly detection techniques, emphasizing strengths, limitations, and practical implications. The adaptability of CLAs to real-world applications such as IoT data monitoring, power consumption analysis, and fuel theft detection is also evaluated.

- **Chapter 7: Conclusion and Future Work**
  The final chapter summarizes the key findings of this study, discusses its limitations, and outlines directions for future research, including potential improvements in CLAs, alternative benchmarking datasets, and further optimization techniques for real-time applications.

Chapter 2

Fundamentals of Time-Series and Anomaly Detection

This chapter provides an overview of time-series data and the diverse methodologies developed for anomaly detection. It begins with the theoretical foundations of time-series analysis and proceeds to review various anomaly detection techniques. The discussion is enriched with insights from recent studies and benchmark efforts, setting the stage for this project on the efficacy of detecting and characterizing anomalies using cortical algorithms.

2.1 Theoretical Foundations of Time-Series Analysis

Time-series data consists of observations recorded in chronological order, capturing patterns and trends that evolve over time. Unlike traditional datasets, where instances are independent of one another, time-series data exhibits temporal dependencies, meaning that each observation is influenced by past values. This sequential nature requires specialized analytical techniques that account for correlations across time. Time-series analysis plays a crucial role in fields such as industrial process monitoring, healthcare analytics, meteorology, and financial forecasting. For example, in industrial settings, monitoring machine sensor data over time can help predict failures, while in healthcare, continuous patient monitoring can identify abnormal physiological signals[7].

Recent advancements in time-series analysis have emphasized the need for adaptive and real-time learning models due to the growing complexity of data streams. Traditional statistical models, such as ARIMA (AutoRegressive Integrated Moving Average) and GARCH (Generalized AutoRegressive Conditional

Heteroskedasticity), are widely used for forecasting but often struggle to handle nonlinear dependencies and evolving patterns[8]. Machine learning and deep learning approaches, including LSTMs (Long Short-Term Memory networks) and Hierarchical Temporal Memory (HTM), have emerged as powerful alternatives that learn complex temporal dependencies dynamically without requiring frequent retraining[9].

Time-series data can be broadly categorized based on its structure. Univariate time-series consists of a single variable evolving over time, such as electricity consumption records or heart rate measurements[1]. Multivariate time-series, on the other hand, captures multiple variables recorded simultaneously, such as temperature, humidity, and wind speed, where interactions between these variables are often important for predictive modeling[10]. The complexity of multivariate time-series analysis has led to the development of hybrid deep learning models that combine autoencoders and recurrent neural networks for enhanced feature extraction and anomaly detection[11].

Another critical aspect of time-series data is sampling frequency. Depending on the context, time-series data can be regularly spaced (e.g., stock prices recorded every second) or irregularly spaced (e.g., event-driven medical records). The irregularity in data sampling introduces challenges in model training and anomaly detection, requiring interpolation techniques or specialized methods such as cortical coding algorithms that can process unstructured temporal data efficiently[12].

Understanding these characteristics is essential for selecting appropriate modeling techniques, as many traditional statistical methods assume stationary patterns and uniform sampling intervals. However, real-world time-series data is often non-stationary, noisy, and incomplete, requiring robust feature extraction methods and anomaly detection techniques that can adapt to concept drift and missing values[8, 13]. The emergence of unsupervised learning approaches, such as Sparse Autoencoders and HTM-based models, provides a promising direction for handling the challenges posed by real-world time-series data[14].

2.1.1 Dimensionality of Time-Series Data

The dimensionality of time-series data refers to the number of attributes recorded at each time step Figure 2.2. Univariate time-series data consists of a single variable measured over time, making it simpler to analyze but limited in capturing complex dependencies. A common example is electricity consumption recorded at fixed intervals, where each data point represents the total power used during that period. Since only one feature is observed, anomaly detection in univariate time-series typically relies on statistical forecasting techniques such as AutoRegressive Integrated Moving Average (ARIMA) or Exponential Smoothing, which compare expected and actual values to detect deviations[15]. However, these methods often

assume stationarity and linearity, making them less effective for capturing nonlinear patterns and abrupt anomalies[11].

In contrast, multivariate time-series data contains multiple interdependent variables recorded over time, introducing additional complexity but also enabling richer pattern recognition. For instance, a smart grid monitoring system may collect voltage, current, and frequency readings simultaneously. Anomalies in such datasets are often detected by analyzing relationships between variables, rather than just tracking individual deviations. Machine learning models, including LSTM networks and Transformers, are commonly used to capture temporal dependencies and complex inter-variable interactions[16]. Recent studies have also demonstrated that unsupervised deep learning approaches, such as Sparse Autoencoders and Hierarchical Temporal Memory (HTM), provide robust anomaly detection by learning latent representations of multivariate time-series data without requiring labeled anomalies[14].

While multivariate analysis provides richer insights into system behavior, it also significantly increases computational complexity. High-dimensional datasets require models capable of efficiently handling relationships across multiple features, and traditional approaches often struggle with the curse of dimensionality Figure 2.1[17]. Recent advancements in cortical coding-based methods have shown promise in reducing dimensionality while preserving critical temporal structures, thus improving the efficiency of multivariate anomaly detection systems[12].

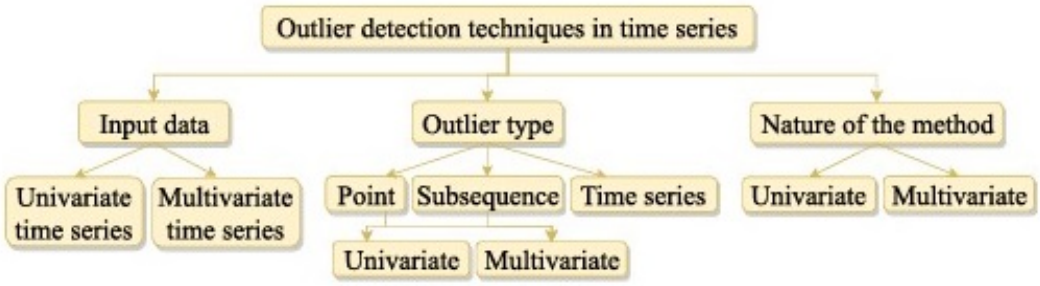

Figure 2.1: Proposed Taxonomy of outlier detection techniques in time series data

2.2 Theoretical Foundations of Anomaly Detection

Anomaly detection refers to the method of recognizing data points that substantially differ from anticipated trends within a dataset.[2]. In time-series data,

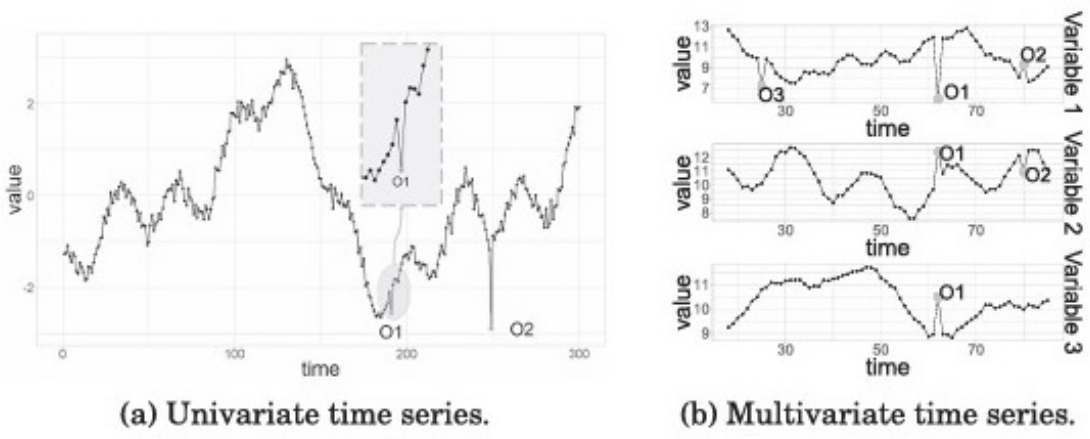

Figure 2.2: Point outliers in time series data

anomalies often signal critical events such as system failures, fraudulent transactions, network intrusions, or significant shifts in behavior. Detecting anomalies in time-series data presents unique challenges due to temporal dependencies, evolving patterns, and high-dimensional relationships, making it more complex than anomaly detection in static datasets.

Traditional anomaly detection techniques rely on statistical models that assume stationarity and predefined data distributions. Gaussian Mixture Models (GMMs), moving average-based outlier detection, and z-score methods detect anomalies by flagging observations that fall outside predefined confidence intervals[18]. These approaches work well for simple, low-noise datasets but struggle with non-stationary time-series, dynamic trend shifts, and contextual anomalies[19].

To address these limitations, more recent approaches leverage machine learning and deep learning techniques. Unsupervised methods, such as Isolation Forest and One-Class SVM, can model normal behavior without requiring labeled anomalies, making them well-suited for real-time applications[16]. Supervised deep learning models, including Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, have shown remarkable success in detecting both point and contextual anomalies[17]. However, supervised models require labeled training data, which is often unavailable in real-world applications.

In contrast, unsupervised deep learning approaches such as Autoencoders, Variational Autoencoders (VAEs), and Hierarchical Temporal Memory (HTM) offer adaptive learning without the need for labeled data. Sparse Autoencoders, for instance, can automatically extract latent representations of normal patterns, flagging deviations as anomalies[14]. Additionally, HTM-based models mimic the brain's cortical learning mechanisms, enabling continuous, one-pass learning for real-time

anomaly detection[9].

Moreover, brain-inspired cortical coding methods have recently gained attention for their ability to handle non-stationary, high-dimensional time-series data with minimal computational overhead[12]. These approaches leverage self-organized hierarchical structuring and entropy maximization, allowing them to detect complex temporal anomalies more efficiently than many conventional methods. The integration of neuromorphic computing principles into anomaly detection is an emerging research direction that holds promise for improving real-time, low-power anomaly detection systems[20].

2.2.1 Anomaly Detection Techniques

The literature on anomaly detection is extensive. Agrawal and Agrawal (2015) [21] provide a survey on data mining techniques for anomaly detection, categorizing them into methods such as classification-based, clustering-based, and hybrid approaches. Their work emphasizes that the appropriate selection of an anomaly detection technique depends on factors like the nature of the input data, the type of anomalies, and the desired output (whether it be a probabilistic score or a binary label).

Toshniwal et al. (2020) [3] offer an overview of a wide range of anomaly detection methods in machine learning. They classify these methods into groups such as classification-based, nearest neighbor-based, clustering-based, statistical, information-theoretic, spectral, and graph-based techniques. Their survey underscores that the effectiveness of each method is highly dependent on the data characteristics and the specific anomaly types expected. This perspective highlights why some methods excel at detecting point anomalies, whereas others are better at spotting sequence-based anomalies.

Anomaly detection techniques can be broadly categorized into supervised, semi-supervised, and unsupervised methods, based on whether labeled data is available for training[2].

Supervised Anomaly Detection Supervised anomaly detection methods, such as Support Vector Machines (SVMs), Decision Trees, and Random Forests, rely on a fully labeled dataset where both normal and anomalous instances are explicitly marked. These models learn to distinguish between normal and anomalous patterns based on labeled training data. When labeled data is available, supervised learning achieves high detection accuracy, making it a preferred approach in applications such as network intrusion detection and credit card fraud detection.

However, a significant limitation of supervised methods is the difficulty of obtaining labeled anomaly data. In many real-world scenarios, anomalies are rare and unpredictable, making it expensive and often infeasible to manually label sufficient instances for training[22]. Moreover, supervised models struggle with

previously unseen anomalies, as they can only recognize patterns present in the training data. This makes them less effective for evolving systems where new types of anomalies frequently emerge.

Semi-Supervised Anomaly Detection Semi-supervised learning provides a middle ground between supervised and unsupervised techniques, leveraging a large set of unlabeled data along with a smaller set of labeled examples. The key assumption in semi-supervised anomaly detection is that normal behavior is well-represented in the training data, while anomalies are rare and often absent from the training set.

In semi-supervised approaches, models are trained only on normal data, learning its distribution and structure. Once trained, the model flags significant deviations from normal patterns as potential anomalies[23, 24]. This approach is particularly beneficial in scenarios where obtaining labeled anomalies is impractical or costly, such as fraud detection[25] and medical diagnosis, where normal physiological patterns are well-defined but anomalies (e.g., rare diseases) are difficult to label[24].

Recent research has demonstrated that deep learning architectures, including Autoencoders, Variational Autoencoders (VAEs), and LSTMs, significantly improve the performance of semi-supervised anomaly detection by capturing complex patterns and latent structures in time-series data. In particular, HTM-based models have been explored for their adaptive, self-learning capabilities, making them highly effective in environments with evolving patterns[9].

Unsupervised Anomaly Detection Unsupervised anomaly detection techniques, including clustering-based methods (DBSCAN, K-means) and probabilistic models (Gaussian Mixture Models, Hidden Markov Models), are the most widely applicable since they do not require labeled data. Instead, they infer normal patterns directly from the dataset and detect anomalies based on deviations from these patterns.

This approach is particularly useful when anomalies are not well-defined or vary over time, as it allows models to dynamically detect previously unseen anomalies. Unsupervised methods often rely on density-based clustering, statistical outlier detection, or deep learning-based feature extraction[26, 27].

Autoencoders are among the most commonly used unsupervised anomaly detection methods. These neural networks are trained to encode and reconstruct normal patterns, and anomalies are identified as instances that fail to reconstruct properly. Other deep learning techniques, such as Generative Adversarial Networks (GANs) and Hierarchical Temporal Memory (HTM), have demonstrated superior performance in detecting contextual and collective anomalies by modeling temporal dependencies in streaming data[14].

A major advantage of unsupervised methods is their ability to adapt to new data without requiring manual labeling. However, these models may also produce higher false positive rates, as they sometimes mistake normal variations in data for anomalies. Recent advancements in cortical coding methods have helped mitigate this issue by leveraging self-organizing hierarchical structures that improve robustness against noise and reduce false positives[12].

The differences between these three approaches are summarized in Table 2.1[25].

Supervised [28, 29]	**Unsupervised [30]**	**Semi-supervised [30]**
• Needs labeled normal and anomaly data • Learns patterns from existing labels • Fails to detect unseen anomalies	• No labeled data required • Detects anomalies as deviations • Identifies unseen anomalies	• Uses both labeled and unlabeled data • Requires few labeled samples • Learns normal behavior, flags deviations • Detects unseen anomalies better than supervised

Table 2.1: Comparison of anomaly detection methods based on data labeling requirements [25].

2.2.2 Anomaly Detection Types

Anomaly detection in time-series data is a specialized task that focuses on identifying unexpected deviations in temporal sequences. Two primary approaches to time-series anomaly detection are time-series forecasting[19] and time-series classification[31].

Forecasting-based models, such as LSTMs and ARIMA, predict future values based on historical data, flagging anomalies when observed values deviate significantly from predictions[32]. Recent advancements in deep learning optimization techniques, particularly Modified Sine Algorithm Dung Beetle Optimization (MSADBO), have further enhanced forecasting accuracy by reducing error margins and improving sensitivity to subtle anomalies[33].

Classification-based approaches assign entire time-series sequences to predefined categories, thereby helping to group different anomaly types for further analysis[13].

To develop an effective anomaly detection system, it is essential to understand the different types of anomalies that may appear in time-series data. These anomalies are typically classified into three main types:

- **Point anomalies:** A single data point that significantly deviates from the normal trend is classified as a point anomaly. In Figure 2.3(a), an artificial time series with a point anomaly is illustrated, where the anomaly's value falls beyond the expected range of normal observations[11].
- **Contextual anomalies:** These anomalies occur when a data point appears normal in a global context but anomalous in a specific local context. For

example, a sudden temperature drop may be normal in winter but anomalous in summer. Figure 2.3(b), illustrates a contextual anomaly, where the data point falls outside the expected distribution in a local window but remains within the overall expected range[11].

- **Collective anomalies:** A sequence of points that deviates from expected patterns rather than an individual outlier is known as a collective anomaly. Unlike point anomalies, these anomalies indicate irregular trends rather than isolated data points. Figure 2.3(c) provides an example of a synthetic collective anomaly[11].

These three categories are visually represented in Figure 2.3, illustrating the differences between point, contextual, and collective anomalies.

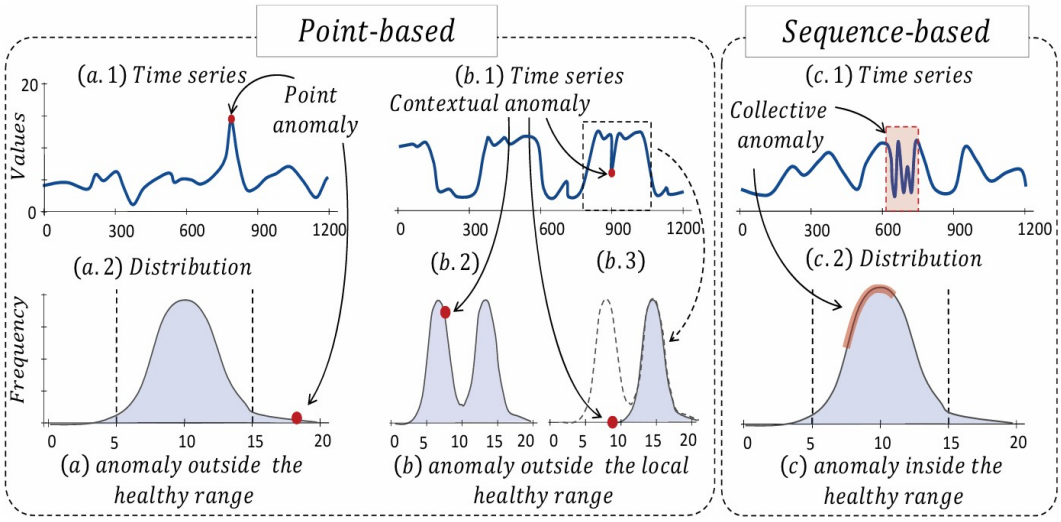

Figure 2.3: Illustration of time-series anomaly types: (a) point anomaly, (b) contextual anomaly, and (c) collective anomaly[11].

2.3 Challenges in Time-Series Anomaly Detection

Despite significant advancements in anomaly detection, several key challenges persist. One of the most pressing issues is data sparsity, particularly in applications such as rare event detection in industrial systems or fraudulent transactions in finance. Many anomaly detection models assume balanced datasets with sufficient anomaly samples, but in reality, anomalies often constitute only a tiny fraction of the

data. This extreme class imbalance leads to poor model generalization, as models trained on predominantly normal data struggle to recognize rare anomalies[23].

Another major challenge is real-time processing. Many deep learning-based models, while highly effective, require substantial computational resources, making them unsuitable for real-time streaming applications such as autonomous driving, financial fraud detection, and power grid monitoring[24]. To enable real-time anomaly detection, models must be optimized for low-latency inference while maintaining high accuracy and adaptability. Recent developments in neuromorphic computing and cortical-inspired models offer promising solutions by reducing computational overhead without sacrificing efficiency[12].

Concept drift, where normal behavior evolves over time, further complicates anomaly detection. Many traditional methods assume static patterns, leading to degraded performance when underlying distributions change. In dynamic environments such as cybersecurity, healthcare, and industrial monitoring, models must be continuously updated to reflect new trends[26]. Adaptive learning approaches, such as Hierarchical Temporal Memory (HTM) and streaming autoencoders, have been developed to address concept drift by learning patterns incrementally and adjusting their representations over time[9].

Effectively tackling these challenges requires a combination of adaptive learning algorithms, robust feature extraction, and scalable model architectures. The integration of self-organizing learning mechanisms, such as cortical coding and sparse distributed representations (SDR), has shown promise in handling data sparsity, reducing computational complexity, and adapting to evolving patterns in real-time[20].

2.4 State-of-the-Art in Anomaly Detection Methods

Modern anomaly detection methods encompass a diverse range of statistical, machine learning-based, and deep learning-based techniques. Each approach varies in its ability to handle complex temporal dependencies, computational efficiency, and adaptability to dynamic data environments.

Traditional methods, such as ARIMA (AutoRegressive Integrated Moving Average) and Exponential Smoothing, serve as baselines for time-series forecasting due to their simplicity, interpretability, and low computational cost[27]. These models rely on fixed mathematical formulations to identify trends and seasonality in time-series data. However, they struggle with non-linear patterns, high-dimensional dependencies, and evolving anomalies. They also require manual parameter tuning, making them less effective for large-scale or rapidly changing datasets.

Machine learning-based methods improve upon traditional techniques by leveraging data-driven pattern recognition. Common approaches include clustering

algorithms (e.g., K-means, DBSCAN), decision trees, and ensemble learning methods. These models automatically extract patterns from data, offering greater adaptability and robustness across different domains. However, their performance is sensitive to parameter tuning and often requires feature engineering to improve accuracy. While these methods reduce manual intervention compared to traditional statistical techniques, they can still be resource-intensive for real-time detection in high-frequency data streams.

Deep learning-based methods have emerged as the most advanced solutions for time-series anomaly detection, leveraging architectures such as Long Short-Term Memory (LSTM) networks, Variational Autoencoders (VAEs), and Generative Adversarial Networks (GANs). These models excel at capturing complex temporal dependencies and non-linear relationships, significantly improving anomaly detection accuracy in domains such as finance, healthcare, cybersecurity, and industrial monitoring[28]. However, deep learning approaches introduce several challenges:

- They often require large volumes of labeled training data, which is frequently unavailable in anomaly detection tasks.
- They incur high computational costs, making them impractical for resource-constrained environments.
- Many deep learning models operate as black-box architectures, reducing interpretability and limiting their adoption in high-stakes applications.

Recent advancements in unsupervised and neuromorphic computing approaches, such as Hierarchical Temporal Memory (HTM) and cortical coding methods, have provided promising alternatives. These methods mimic brain-inspired learning mechanisms to achieve real-time anomaly detection with minimal labeled data and lower computational complexity[12, 9]. By leveraging self-organizing learning structures and adaptive feature extraction, they bridge the gap between high accuracy and computational efficiency, making them suitable for streaming data applications.

2.5 Brain-Inspired Approaches: HTM & Sparse LSTM Auto-Encoders

Recent research has explored cortical learning algorithms inspired by the human brain, particularly Hierarchical Temporal Memory (HTM) and Sparse Autoencoders. These models draw on biological principles to improve pattern recognition, anomaly detection, and adaptive learning in time-series data.

HTM, developed by Numenta, mimics the hierarchical structure of the neocortex, using Sparse Distributed Representations (SDRs) to model sequential patterns and

temporal dependencies[34]. Unlike traditional machine learning methods, HTM performs continuous, one-pass learning, making it highly adaptive to evolving patterns in real-time streaming environments. This biologically inspired mechanism enables HTM to detect anomalies without retraining. This is a significant advantage over many deep learning models that require large labeled datasets[9]. However, HTM is sensitive to parameter tuning, and its performance varies depending on spatial and temporal pooling parameters.

Sparse autoencoders, on the other hand, impose sparsity constraints on neural networks to learn efficient feature representations of time-series data[30]. By forcing a network to activate only a small fraction of neurons per input, sparse autoencoders enhance anomaly detection performance in environments with limited labeled anomalies. They excel at capturing low-dimensional representations while preserving the essential structure of the original data, making them highly effective for unsupervised anomaly detection[14]. However, sparse autoencoders typically require substantial training data to optimize their latent representations, and performance depends on the selection of regularization constraints and activation functions.

While HTM provides strong adaptability and real-time learning capabilities, sparse autoencoders offer superior feature extraction for detecting subtle anomalies in high-dimensional time-series data. This study evaluates both methods in the context of univariate time-series anomaly detection, assessing their real-world applicability, computational efficiency, and robustness to evolving patterns.

2.6 Identified Gaps and Research Motivation

Despite significant progress in anomaly detection, several critical gaps remain. Many state-of-the-art models, particularly deep learning-based approaches, rely on large labeled datasets, limiting their effectiveness in sparse and real-time environments. In practical applications such as power consumption monitoring, industrial system diagnostics, and financial fraud detection, labeled anomalies are scarce and expensive to obtain. This reliance on labeled data reduces the generalizability and adaptability of existing models in dynamic, real-world settings[19].

Furthermore, while deep learning methods such as LSTMs and Autoencoders have demonstrated strong anomaly detection performance, few studies have conducted comparative evaluations of cortical learning algorithms (e.g., HTM) against deep learning approaches in real-time applications. The biologically inspired properties of HTM and Sparse Autoencoders, particularly their ability to learn patterns dynamically with minimal supervision, present an opportunity to develop more efficient and scalable anomaly detection models. However, the trade-offs between these approaches in terms of accuracy, adaptability, and computational efficiency

remain underexplored.

This research aims to fill these gaps by conducting a comparative evaluation of HTM and Sparse Autoencoders on the UCR and NAB time-series anomaly detection benchmarks. The study assesses their ability to detect anomalies in real-world streaming environments, highlighting their adaptability, computational feasibility, and overall performance. By exploring the strengths and limitations of cortical learning mechanisms, this research contributes to the development of scalable and biologically inspired anomaly detection frameworks for future applications.

Chapter 3

Methodology

Introduction

This chapter outlines the methodology employed to investigate and compare two distinct approaches for anomaly detection in time-series data: Hierarchical Temporal Memory (HTM) and Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT). Given the challenges associated with sequential data—such as data sparsity and evolving patterns—this chapter details the procedures for model selection, data preprocessing, hyperparameter tuning, and training strategies to build robust anomaly detection systems.

The methodological framework bridges theoretical principles with practical implementation. For each model, we describe the core theoretical background, model architecture, training process, and anomaly detection mechanism. While HTM is inspired by the neocortex and incorporates biologically plausible mechanisms like spatial pooling and temporal memory to continuously learn from streaming data, SLADiT leverages the capabilities of LSTM networks within an autoencoder framework, enhanced by sparsity constraints in the latent space to capture salient temporal features and flag anomalies through reconstruction errors.

This chapter is organized into two main sections, one for each model. For each approach, we discuss the underlying theory, detail the architecture and training process, and outline the anomaly detection strategy. By presenting both the biologically inspired (HTM) and deep learning-based (SLADiT) approaches, this study aims to offer a comprehensive evaluation of their trade-offs in terms of detection accuracy [12], computational efficiency, and robustness to noise [35]. A brief discussion of the comparative strengths and limitations of these models is provided, with further evaluation and benchmarking (including datasets and evaluation metrics) detailed in Chapter 4 and subsequent chapters.

The insights gained from this methodology will inform the experimental analysis and comparative performance assessment of HTM and SLADiT across multiple time-series datasets.

3.1 Hierarchical Temporal Memory (HTM)

3.1.1 Theoretical Background

The American company Numenta introduced HTMs, or hierarchical learning structures. They were proposed to simulate cortex-like processing of learned information because of their architecture, which draws inspiration from the cerebral cortex and more closely resembles its structure than typical artificial neural networks [36]. Although its full potential has not yet been shown, HTM is a relatively new technique that has shown effectiveness in solving some pattern recognition issues including temporal context [37]. Unlike conventional machine learning models that require extensive labeled data and batch training, HTM operates continuously in an unsupervised manner, dynamically adapting to new patterns in real-time. This ability makes HTM particularly effective for anomaly detection in streaming time-series data [36].

With a focus on spatial patterns and sequence learning, HTM is a framework that is modeled after the human neocortex that arranges neurons hierarchically into layers and columns. Its capacity to continuously learn from and adjust to new facts over time is its main strength.

Time series data, healthcare, and video surveillance are just a few of the domains in which HTM has been used. It has shown potential in real-time anomaly detection [38], especially in streaming datasets like ECG signals where it effectively detects cardiac irregularities [39]. Brain-inspired machine learning has a lot of potential to advance with the continuous development of HTM algorithms, especially in hardware optimization and new encoding techniques [39]

3.1.2 Model Architecture

HTM's architecture is designed for real-time learning. Unlike traditional deep learning models that require periodic retraining, HTM continuously adapts to evolving data patterns, making it particularly suited for applications such as network intrusion detection, financial fraud detection, and industrial system monitoring [37, 33].

In the overall HTM architecture figuring 3.1, the role of the Spatial Pooler, Temporal Memory, and the mechanisms of SDR are highlighted in regard to anomaly detection [40].

HTM is built upon the principles of Sparse Distributed Representations (SDRs), which provide robust and noise-tolerant encoding of input data. It also utilizes two main processing modules that work in tandem to learn and predict temporal patterns:

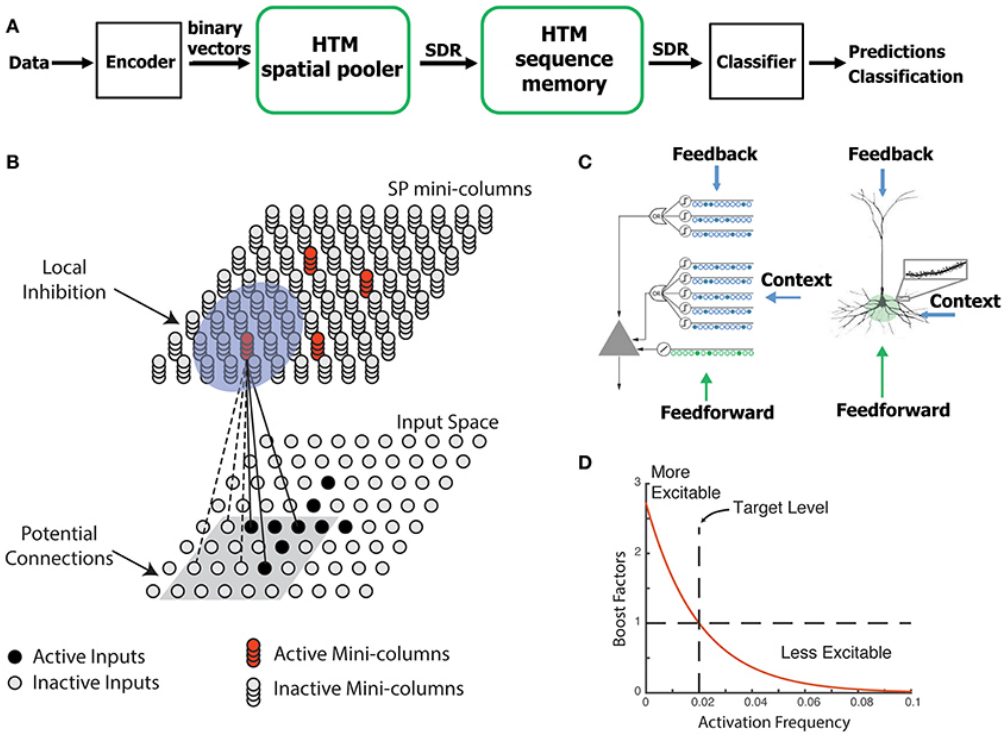

Figure 3.1: (A) An encoding interface, the HTM spatial pooler, the HTM temporal memory, and an SDR classifier, all of which function as modules within a comprehensive HTM system. (B) The HTM spatial pooler is responsible for converting inputs (at the bottom) into sparse distributed representations (SDRs) (at the top). Each mini-column of the spatial pooler can establish a localized group known as an initiation site, facilitated by synaptic connections that link it to various configurations within the input space (illustrated by the gray square, indicating potential connections). A local inhibition mechanism operates within a specified radius (illustrated by the shaded blue circle), allowing only a limited number of SP mini-columns that receive the majority of inputs to remain active. The synaptic permanences are modified in accordance with the Hebbian learning principle: for each active SP mini-column, the associated active inputs (represented by solid black lines) are reinforced, whereas the inactive inputs (indicated by dashed lines) are weakened. (C) An HTM neuron (on the left) contains three distinct dendritic integration zones that align with different regions of the dendritic tree of pyramidal neurons (on the right). The spatial pooler models the feedforward connections to the proximal dendrites. (D) The activation history of an SP mini-column affects its excitability.

Spatial Pooler (SP)

SP is the first stage in HTM's processing pipeline. This module transforms raw, binary-encoded input data into a sparse set of active mini-columns. By ensuring

that semantically similar inputs generate highly overlapping SDRs, the SP enhances robustness to noise and variability [40, 41].

Temporal Memory (TM)

Building on the SP's output, the TM learns sequential dependencies over time by forming distal synapses between neurons. It generates predictions based on prior observations, and when these predictions fail, an anomaly is signaled [42, 9].

A key feature of HTM is its ability to compute anomaly scores based on deviations from expected patterns. The anomaly score at a given time step is computed as:

$$A_t = 1 - \frac{|P_t \cap A_{t-1}|}{|A_{t-1}|} \tag{3.1}$$

where P_t represents the set of active mini-columns at time t, and A_{t-1} denotes the predicted active columns from the previous time step. A high anomaly score suggests a significant deviation from expected sequences, signaling the presence of an anomaly [43].

Sparse Distributed Representations (SDRs)

form the foundation of HTM's encoding mechanism. Unlike dense representations used in deep learning, SDRs activate only a small fraction (approximately 2%) of neurons at a time, making them highly noise-tolerant and efficient [20].

Neurons and Synapses in HTM

HTM neurons differ significantly from traditional artificial neurons in deep learning. Each HTM neuron consists of three functionally distinct dendritic regions:

- **Proximal Dendrites:** Process feedforward input from the Spatial Pooler.
- **Distal Dendrites:** Learn temporal dependencies by forming connections with prior neuron activations.
- **Apical Dendrites:** Receive top-down feedback, allowing hierarchical learning across different levels of abstraction [9].

Unlike standard artificial neurons that rely solely on weight updates through backpropagation, HTM neurons leverage three distinct dendritic integration mechanisms, enabling the system to continuously refine its representation of normal patterns in an unsupervised manner. This architectural difference enhances HTM's

ability to recognize temporal dependencies and detect anomalies as deviations from learned sequences without requiring periodic retraining.

Synapses in HTM strengthen or weaken dynamically, mimicking biological learning mechanisms. Unlike deep learning models that rely on backpropagation, HTM employs unsupervised Hebbian learning, where synaptic weights adjust based on local activity patterns [43].

3.1.3 Training and Anomaly Detection Process

Originally, HTM was structured to handle multivariate time-series data, where temporal encoding played a critical role in detecting patterns. However, for this study, the model was adapted to process univariate datasets from the UCR anomaly benchmark. This adaptation necessitated several modifications that impact its ability to infer temporal dependencies.

Data Preprocessing

The preprocessing stage transformed raw time-series data into a structured format compatible with HTM's encoding mechanisms. In its original implementation, HTM processed datasets such as `gymdata.csv` containing timestamped power consumption measurements. Adapting to the UCR dataset required reducing the input to a single univariate feature without explicit time-based information. The dataset was parsed, numerical values normalized, and structured for encoding. (Details on dataset-specific preprocessing are provided in Chapter 4.)

Data Encoding

HTM systems rely on converting raw input data into Sparse Distributed Representations (SDRs) through appropriate encoders. Different types of data require specialized encoding approaches:

- **Numeric Encoders:** A simple numeric encoder divides a fixed range of numbers into overlapping "buckets," ensuring that similar values activate overlapping bits. For continuous signals like ECG amplitudes or walking acceleration, a scalar numeric encoder is often used.

- **Categorical Encoders:** These encoders are used when data comes in distinct groups. Standard categorical encoders assign a unique bit pattern to each category, whereas cyclic encoders are designed for data with an inherent cycle (e.g., days of the week), wrapping the representation so that end-of-cycle values overlap with the beginning.

- **Geospatial Encoders:** These map geographic coordinates into SDRs. A simple geospatial encoder converts coordinates into grid cells and assigns bits via a hash function, while a flexible geospatial encoder can adjust its resolution based on context.

- **Natural Language Encoders:** These methods convert text into SDRs that capture semantic meaning, enabling the system to recognize linguistic patterns.

The key idea across these encoders is that similar input values yield SDRs with significant overlap, facilitating pattern recognition and noise tolerance.

One critical limitation arises when the chosen dataset, such as those from the UCR benchmark, lacks strong temporal dependencies. In such cases, a simple scalar numeric encoder may fail to capture all meaningful features of the data. It is important to assess the suitability of the encoder with respect to the data and consider parameter adjustments—such as modifying the total number of bits (n) or the number of active bits (w)—to enhance the representation. These considerations are discussed in [44] by Scott Purdy from Numenta, Inc.

HTM's biologically inspired framework provides a compelling approach for real-time anomaly detection, with unique advantages over conventional deep learning techniques. Encoding plays a crucial role in how HTM processes data. In the original multivariate implementation, the model employed both a DateEncoder for capturing time-of-day dependencies and a Random Distributed Scalar Encoder (RDSE) for numerical values. The DateEncoder facilitated the identification of recurring patterns, while RDSE transformed continuous variables into sparse representations. However, in the adapted univariate version, the DateEncoder was removed, leaving only the RDSE. This modification significantly affected HTM's ability to learn periodic dependencies, making context-dependent anomalies—such as seasonal variations—harder to detect and leading to increased false positives and negatives [40].

Spatial Pooler

HTM's Spatial Pooler (SP) transforms the encoded inputs into stable sparse representations that serve as the foundation for sequence learning in the Temporal Memory. In the original model, the SP processed diverse encoding inputs to form rich SDRs. However, with the shift to a single univariate feature, the variety of SDRs was significantly reduced. This reduction limits the model's ability to generalize across different data patterns, resulting in weaker spatial representations and a greater reliance on local feature variations rather than broader contextual patterns.

Temporal Memory

HTM's Temporal Memory (TM) is the core learning mechanism responsible for detecting sequential dependencies. In a multivariate context, TM leverages both numerical values and time-based features to form robust associations between past and present observations. In the univariate adaptation, TM must rely solely on past values, which compromises its ability to distinguish between expected variations and true anomalies—especially in non-stationary datasets [45]. The removal of time-based encoding also led to increased root mean squared (RMS) prediction errors, as the model struggled to maintain stable sequential learning.

Prediction and Anomaly Detection

HTM's anomaly detection mechanism derives from probability distributions generated by the active cells within the TM. An anomaly is flagged when the likelihood of an observation falls below a predefined threshold—typically set to the 95th percentile of training set likelihoods. However, in the UCR anomaly benchmark, variations in training and test set distributions reduced the reliability of this thresholding approach. Additionally, when learned patterns were unclear, the predictor exhibited increased uncertainty and detection inconsistency.

Performance Considerations

HTM's performance on the UCR datasets was influenced by multiple factors. One of the primary limitations was the absence of dataset-specific parameter tuning. Unlike deep learning models, which often undergo rigorous hyperparameter optimization, HTM's parameters—such as column count, activation thresholds, and synapse permanence values—were kept at default settings. This likely led to suboptimal performance, as the model was not tailored to the characteristics of individual datasets. Furthermore, univariate datasets with non-stationary patterns or high noise levels proved more difficult for HTM to analyze effectively. The lack of time-based encoding removed critical contextual information, making it harder to differentiate between normal fluctuations and true anomalies. Additionally, the RDSE encoder required careful calibration to avoid losing meaningful variations, as improper resolution settings could cause small anomalies to be overlooked.

Boosting Mechanism for Learning Stability

HTM incorporates a boosting mechanism to ensure that all mini-columns participate in learning. The excitability of a mini-column is inversely proportional to its

historical activation frequency. A boost factor is applied as follows:

$$B = \frac{1}{\sqrt{F + \epsilon}} \tag{3.2}$$

where F represents the mini-column's activation frequency, and ϵ is a small constant to prevent division by zero. Rarely active mini-columns receive higher boost values, encouraging their participation in learning [40].

3.1.4 HTM's Strengths and Limitations

HTM offers several advantages:

- **Continuous Learning:** HTM updates itself in real time without requiring retraining.
- **Noise Resilience:** SDRs and sparse activation patterns renders HTM robust to noisy inputs.
- **Unsupervised Learning:** HTM does not require labeled data, making it ideal for applications where anomaly definitions evolve over time [20].

However, HTM also has limitations:

- **Parameter Sensitivity:** Performance depends on fine-tuning hyperparameters such as synaptic permanence.
- **Computational Complexity:** The large number of synapses and neurons can make real-time execution computationally expensive.
- **Limited Benchmark Comparisons:** HTM's performance varies across datasets, making direct comparisons challenging [37].

HTM's biologically inspired framework provides a compelling approach for real-time anomaly detection, with unique advantages over conventional deep learning techniques.

3.2 Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT)

3.2.1 Theoretical Background

Anomaly detection in time-series data is a crucial task in applications such as cybersecurity, finance, healthcare, and industrial monitoring. Traditional statistical

and machine learning-based methods often fail to capture anomalies effectively, especially when the patterns are subtle or the dataset is highly sparse. In this regard, neural network-based architectures, particularly Sparse Autoencoders (SAEs) integrated with Long Short-Term Memory (LSTM) layers, have shown remarkable success in detecting anomalies by leveraging unsupervised learning principles [12].

Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT) is a specialized type of autoencoder that imposes sparsity constraints on its hidden layers. Unlike conventional autoencoders that simply learn to reconstruct inputs, SLADiT SLADiT forces the model to activate only a small subset of neurons at any given time, thereby thereby emphasizing the most salient features of the data [35]. When integrated with LSTM layers, SLADiT can capture temporal dependencies in time-series data, making them highly effective in modeling sequential patterns and detecting deviations that signify anomalies.

3.2.2 Model Architecture

The SLADiT architecture comprises three fundamental components: the encoder, the latent space (with sparsity enforcement), and the decoder. Each of these plays a critical role in learning meaningful representations of normal behavior and identifying deviations that indicate anomalies. Figure 3.2 illustrates the structure of the SLADiT model, including LSTM layers, fully connected layers, and sparsity enforcement mechanisms.

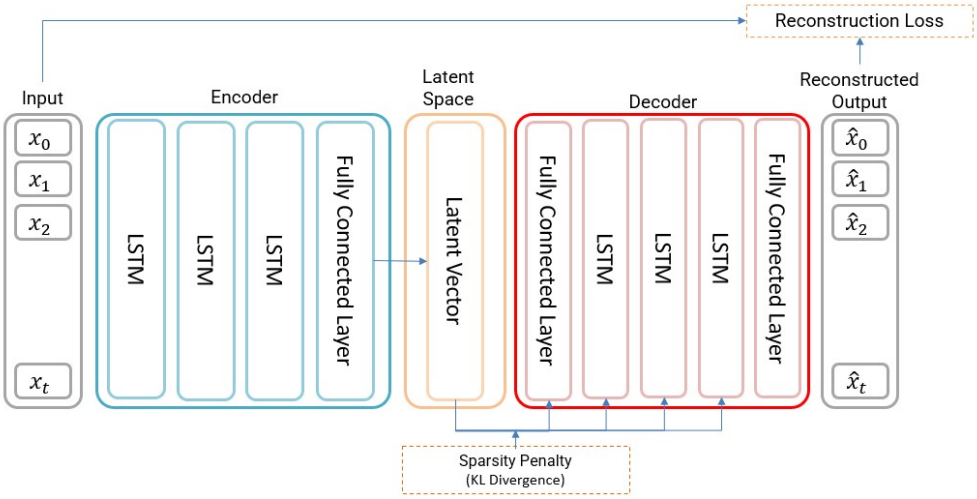

Figure 3.2: SLADiT architecture, highlighting its encoder, latent space, and decoder components.

Input Layer and Data Preprocessing

Before feeding time-series data into the SLADiT model, several preprocessing steps are necessary to ensure numerical stability and optimal learning performance. The data is first segmented into sliding windows, allowing the model to analyze localized temporal patterns. Additionally, normalization techniques are applied to standardize input values, preventing biased learning due to scale discrepancies.

Encoder

The encoder's primary function is to compress high-dimensional time-series input into a lower-dimensional latent representation while preserving essential temporal features. This is achieved through:

- **LSTM Layers:** The encoder consists of multiple stacked LSTM layers that process input sequences while maintaining long-term dependencies. Each layer refines the feature abstraction, enabling the capture of increasingly complex temporal relationships [46].
- **Fully Connected Layer:** The final LSTM layer is followed by a fully connected layer that maps the hidden state to a compact latent representation, setting the stage for sparsity constraints.

Latent Space and Sparsity Constraint

The latent space serves as a high-dimensional, meaningful representation of the input data, capturing its most salient features while discarding redundant information. This space is crucial for anomaly detection, as it ensures that only the most informative components of the data are preserved. A sparsity constraint is imposed to ensure that only the most relevant neurons remain active. This constraint is enforced through Kullback-Leibler (KL) divergence, which penalizes excessive activations:

$$\mathrm{KL}(p \parallel \hat{p}) = p \log \frac{p}{\hat{p}} + (1-p) \log \frac{1-p}{1-\hat{p}} \tag{3.3}$$

where p represents the desired average activation (a small target value), and $\hat{p}$ denotes the actual average activation of the hidden neurons. This regularization mechanism promotes sparsity and prevents the network from learning trivial patterns that could lead to overfitting.

Decoder

The decoder reconstructs the input data from the sparse latent representation by reversing the encoding process. It consists of:

- **Fully Connected Layer:** Converts the latent vector back into a format for sequence reconstruction.
- **LSTM Layers:** Reconstructs temporal sequences from the latent representation, ensuring that the output retains meaningful patterns.
- **Output Layer:** Projects the final reconstructed sequence back to its original input dimension, minimizing reconstruction error through Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{3.4}$$

  where x_i is the original input and $\hat{x}_i$ is the reconstructed output.

3.2.3 Training and Anomaly Detection Process

Data Preparation and Preprocessing

The training procedure for the LSTM-based Sparse Autoencoder begins with meticulous data preprocessing, which is essential to ensure robust learning and accurate anomaly detection. The time-series data is initially normalized to mitigate variations in scale, preventing biases in model training. Following normalization, the dataset is segmented into sliding windows, a technique that enables the model to capture localized temporal dependencies. By employing this approach, the model gains access to overlapping sequences, allowing it to generalize better across different time scales while preserving essential structural patterns. (Dataset-specific preprocessing details, including for the NYC Taxi dataset from the NAB collection and UCR benchmarks, are discussed in Chapter 4.)

Training Phase and Loss Function Optimization

During training, the SLADiT model is optimized to minimize a composite loss function, which combines reconstruction error and sparsity regularization. The primary objective is to ensure that the model accurately reconstructs normal time-series patterns while enforcing sparse activations in the latent space.The Mean Squared Error (MSE) is used to quantify reconstruction loss (3.4) where x_i is the original input and $\hat{x}_i$ is the reconstructed output.

In parallel, the sparsity constraint is enforced using Kullback-Leibler (KL) divergence (3.3), ensuring that only a subset of neurons in the latent space remains active. This is expressed as where p represents the desired average activation, and $\hat{p}$ denotes the actual average activation of the hidden neurons.

The final loss function used for model optimization is given by:

$$Loss = MSE + \lambda \cdot \mathrm{KL}(p \parallel \hat{p}) \tag{3.5}$$

where λ is a weighting factor that controls the trade-off between reconstruction accuracy and sparsity enforcement.

Anomaly Detection Process

Once the SLADiT model has been trained on predominantly normal time-series sequences, it is then applied to detect anomalies in unseen data by assessing reconstruction errors against a dynamically computed threshold. The detection pipeline follows these steps:

1. **Encoding:** Unseen data is passed through the trained encoder to extract a latent representation.
2. **Reconstruction:** The decoder reconstructs the input sequence from the latent space.
3. **Error Calculation:** MSE is computed to measure the deviation between the input and the reconstructed output.
4. **Threshold-Based Classification:** Anomalous instances are detected based on a dynamic threshold, computed as:

$$Threshold = \mu_{MSE} + k \cdot \sigma_{MSE} \tag{3.6}$$

   where μ_{MSE} is the mean reconstruction error on the training data, σ_{MSE} is its standard deviation, and k (typically between 2 and 3) is a constant that defines the sensitivity of anomaly detection.

Data points with reconstruction errors exceeding this threshold are classified as anomalies.

Evaluation Metrics and Model Performance Assessment

To assess the effectiveness of the SLADiT model, a suite of evaluation metrics is used, such as Precision, Recall, F1-Score, Accuracy and rea Under the Curve (AUC). we will discuss the metrics more in chapter 4. Additionally, the timeliness of anomaly detection is considered, ensuring that SLADiT is capable of detecting anomalies within a practical time frame for real-world applications.

While SLADiT demonstrates superior detection accuracy, its batch training approach makes it inherently less suitable for real-time detection scenarios. In applications requiring immediate anomaly response—such as fraud detection or industrial system monitoring—SLADiT would require frequent retraining or adaptation strategies (e.g., online fine-tuning), which could increase computational overhead.

A comparative evaluation of HTM and SLADiT's strengths and limitations, including their detection accuracy and computational efficiency, will be presented in Chapter 5 based on empirical results.

3.2.4 SLADiT's Strengths and Limitations

SLADiT offers several advantages as a deep learning-based approach to time-series anomaly detection:

- **Enhanced Feature Extraction:** By integrating LSTM layers with sparsity constraints, SLADiT is capable of capturing complex temporal dependencies and subtle variations in sequential data.
- **Robust Reconstruction:** The autoencoder framework allows SLADiT to learn an effective representation of normal behavior, making deviations—manifested as increased reconstruction errors—a reliable signal for anomalies.
- **Adaptability in Batch Settings:** When applied to datasets with consistent patterns, SLADiT demonstrates high detection accuracy and can effectively model the underlying distribution of normal data.

However, SLADiT also has some limitations:

- **Batch Training Constraints:** The model's reliance on batch training makes it less suitable for real-time anomaly detection scenarios, as it requires retraining or adaptation (e.g., online fine-tuning) to incorporate new patterns.
- **Computational Overhead:** Training multiple LSTM layers with sparsity constraints can be computationally intensive, particularly when dealing with large-scale or high-frequency time-series data.
- **Parameter Sensitivity:** The performance of SLADiT is sensitive to hyperparameter choices, including the number of LSTM layers, latent space dimensionality, and sparsity targets. Inadequate tuning may lead to suboptimal feature representations and reduced anomaly detection performance.

Overall, while SLADiT demonstrates strong capabilities in capturing temporal dynamics and detecting anomalies with high accuracy, its batch-based nature and computational demands pose challenges for real-time applications.

Chapter 4

Benchmark Datasets and Evaluation Metrics

In this chapter, we present the experimental framework used to evaluate the performance of anomaly detection models. We focus on the methodologies introduced in previous chapters, detailing both qualitative and quantitative assessments of their effectiveness.

We begin by categorizing time-series data based on its attributes, as the choice of anomaly detection techniques depends significantly on the nature of the input. Next, we describe the benchmark datasets selected from the UCR and NAB collections, followed by the evaluation metrics used to measure model performance. Finally, we discuss the challenges and limitations of benchmarking, particularly in the context of Cortical Learning Algorithms (CLAs) and Hierarchical Temporal Memory (HTM)-based models.

In subsequent sections, we provide detailed descriptions of data preprocessing—including feature engineering to capture cyclic patterns—and discuss model training nuances such as hyperparameter tuning, threshold selection, checkpointing, and logging, as well as computational considerations that influence model performance.

4.1 Types of Data in Time-Series Analysis

The foundation of any anomaly detection technique depends on the structure and nature of the input data. Time-series data can be categorical, numerical, or textual, and each dataset consists of individual observations (instances) characterized by one or more features, variables, or dimensions. The type of data not only influences the choice of detection methods but also determines the preprocessing and feature engineering steps that may be required to effectively capture underlying patterns[15].

4.1.1 Continuous vs. Categorical Attributes

- **Continuous attributes** (e.g., temperature, pressure, stock prices) require statistical or machine learning models that assume numerical relationships.Such data often benefits from scaling (e.g., normalization) and may require additional feature engineering—such as incorporating cyclic transformations (e.g., sine and cosine components) to capture periodicity.
- **Categorical attributes** (e.g., weather conditions: sunny, cloudy, rainy) often require probabilistic models, symbolic sequence mining, or encoding techniques to represent the data effectively.These attributes may be transformed into numerical representations (e.g., via one-hot encoding) before further analysis.

4.1.2 Univariate vs. Multivariate Time-Series Data

- **Univariate data** consists of a single variable evolving over time.
- **Multivariate data** contains multiple interdependent variables, requiring models capable of capturing complex dependencies.

The selection of an anomaly detection method depends largely on the data structure. For instance, Nearest-neighbor-based methods rely on distance measures, which differ for numerical and categorical data. Similarly, clustering-based methods require a well-defined feature space, which varies significantly between univariate and multivariate time series.

4.2 UCR and NAB Time-Series Anomaly Detection Benchmark

4.2.1 Description of UCR Datasets

The UCR Anomaly Archive [47] [48] is a widely recognized benchmark collection containing 250 univariate time-series datasets. These datasets span a diverse range of application domains, including human medicine, biological processes, meteorology, and industrial monitoring. Given its extensive coverage and structured labeling, the UCR archive serves as an essential resource for evaluating anomaly detection methodologies.

Initially introduced as part of an anomaly detection contest preceding the ACM SIGKDD conference in 2021, this dataset collection has since been extensively used in research to benchmark and compare different anomaly detection algorithms.

Each time series in the UCR Anomaly archive contains exactly one artificially injected anomaly in the test set, ensuring a controlled and consistent evaluation

framework. The training set contains only normal data, allowing models to learn typical patterns before encountering anomalies. Synthetic anomalies are introduced using Gaussian noise, wandering baselines, and other perturbation techniques, ensuring a structured test environment for assessing anomaly detection performance.

- **052_UCR_Anomaly_TkeepFifthMARS:** The dataset is a real-world time series derived from NASA spacecraft telemetry data, originally featured in a KDD 2018 paper. In this version, the train and test sets were merged, and the two original anomalies were carefully removed. Instead, a new anomaly was introduced by making one beat approximately 25% slower than the rest, creating a distinct deviation in the temporal pattern. The dataset provides a controlled yet realistic setting for anomaly detection research, allowing algorithms to distinguish true anomalies from natural variations in spacecraft telemetry.

- **054_UCR_Anomaly_WalkingAceleration5:** The dataset is derived from an activity recognition benchmark, specifically capturing acceleration data from a sensor worn by a subject while walking. Since the acceleration is originally recorded in three dimensions (x, y, and z axes), the dataset was created by averaging these values to produce a single time series representation of movement dynamics. To introduce an anomaly, a half-cycle of walking data was inverted, effectively flipping its pattern upside down. This manipulation disrupts the natural periodicity of the walking motion, providing a controlled yet challenging test case for anomaly detection algorithms in human activity recognition.

- **097_UCR_Anomaly_GP711MarkerLFM5z2:** The dataset originates from subject 7 in the GaitPhase Database [49]. The subject was required to walk on a split-belt treadmill at a walking speed of 1.1 m/s. This dataset represents the vertical displacement of a 3D marker's position, with the marker placed on the subject's left shoe above the second metatarsal head. It comprises a total of 55 complete gait cycles. To introduce a synthetic anomaly, a random gait cycle was modified by amplifying the second peak—making it almost as high as the main peak. This deliberate perturbation provides a controlled yet realistic challenge for anomaly detection algorithms in gait analysis.

- **123_UCR_Anomaly_ECG4:** The dataset is derived from an electrocardiogram (ECG) trace, designed to simulate real-world cardiac signal variations while incorporating a synthetic anomaly. Both the train and test sets exhibit fluctuating noise levels that naturally increase and decrease over time; however, these variations are not considered anomalies. The true anomaly in this dataset is artificially introduced by inverting approximately two beats at a

random location, effectively flipping the signal upside down by multiplying the subsequence by -1. This transformation creates a distinct yet subtle disruption in the ECG pattern, making the dataset a valuable resource for testing anomaly detection algorithms in biomedical signal processing.

- **142_UCR_Anomaly_InternalBleeding6:** The dataset is derived from an internal bleeding study, specifically focusing on arterial blood pressure measurements collected from pigs. It provides a critical physiological signal used for monitoring circulatory health. A synthetic anomaly was introduced by performing an upsampling operation on a short segment of normal arterial blood pressure data. Between time points 3393 and 3570, the frequency of the time series was doubled, with missing values filled using the previous time point's value. This modification creates an artificially smoothed yet temporally distorted region, challenging anomaly detection algorithms to recognize frequency-based anomalies within biomedical signals.

- **151_UCR_Anomaly_MesoplodonDensirostris:** This dataset originates from an accelerometer recording of a Blainville's beaked whale, capturing its natural swimming patterns. To introduce a controlled anomaly, one complete swim cycle was replaced with two human heartbeats, carefully scaled to match the original data's mean and variance. This subtle yet biologically distinct alteration creates an anomaly that blends into the overall signal while disrupting the expected rhythmic movement of the whale. The dataset provides a unique challenge for anomaly detection algorithms, requiring them to differentiate between natural aquatic locomotion and an artificially inserted human physiological pattern.

- **152_UCR_Anomaly_PowerDemand1:** The dataset is sourced from an Italian power demand record spanning from January 1, 1995, to May 31, 1998. It captures fluctuations in electricity consumption over time, reflecting real-world energy usage patterns. The anomaly in this dataset is artificially introduced by applying a moving average algorithm to a randomly selected two-week period of power supply data (from February 9, 1997, at 3:00 AM to February 23, 1997, at 3:00 AM). This transformation smooths out natural variations, creating an anomalous segment that deviates from the expected demand dynamics. The dataset serves as a valuable benchmark for detecting subtle yet impactful changes in power consumption patterns.

4.2.2 Description of NAB Dataset

In addition to the UCR Archive, this study also utilizes datasets from the Numenta Anomaly Benchmark (NAB)[50]. NAB provides a collection of real-world

and synthetic time-series datasets specifically designed for anomaly detection in streaming data. One of the most notable datasets from NAB is the **NYC_Taxi** dataset, which records taxi trip counts in New York City over time. This dataset is characterized by:

- **Real-World Complexity:** The NYC_Taxi dataset captures the inherent variability of urban mobility, including daily, weekly, and seasonal patterns.
- **Streaming Data Environment:** As part of the NAB collection, the dataset is structured to simulate real-time data streams, challenging models to adapt continuously and detect anomalies promptly.
- **Anomaly Characteristics:** Anomalies in the NYC_Taxi dataset may reflect unusual patterns such as unexpected surges or drops in taxi demand, which are critical for understanding urban dynamics and operational planning.

Benchmark Dataset Overview: Integrating datasets from both UCR and NAB enables a more comprehensive evaluation of anomaly detection models across controlled synthetic environments and complex real-world scenarios. The following table 4.1 summarizes the key characteristics of the selected datasets from the UCR Anomaly Archive and NAB collection used in this study.

Dataset Name	Records	Train	Test	Anomalies	Data Type
052_TkeepFifthMARS	11,308	3,500	7,808	99	Sensor
054_WalkingAceleration5	6,684	2,700	3,984	60	Sensor
097_GP711MarkerLFM5z3	12,000	5,000	7,000	46	Sensor
123_ECG4	30,000	5,000	25,000	301	ECG
142_InternalBleeding6	7,654	1,500	6,154	156	(ABP) Sensor
151_MesoplodonDensirostris	24,667	10,000	14,667	161	Accelerometer
152_PowerDemand1	29,931	9000	20,931	337	Sensor
NYC_Taxi (NAB)	10,320	5,000	4,420	5	Transportation

Table 4.1: Benchmark table for selected NAB and UCR datasets.

4.3 Preprocessing

The dataset used in this study is sourced from the UCR Anomaly Archive, which provides standardized benchmark time-series datasets for anomaly detection research, and the NYC Taxi dataset from the Numenta Anomaly Benchmark (NAB),

which captures real-world urban mobility patterns. Before applying anomaly detection models, the raw data undergoes preprocessing to ensure consistency and optimal learning conditions.

To achieve numerical stability and improve model performance, we apply z-score normalization (used in SLADiT) and min-max normalization (used in HTM), which transforms the dataset as follows:

$$X' = \frac{X - \mu}{\sigma} \tag{4.1}$$

where X represents the original data points, μ is the mean, and σ is the standard deviation. This transformation ensures that the data distribution has zero mean and unit variance, facilitating effective training and anomaly detection.

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{4.2}$$

In addition to scaling, we incorporate feature engineering steps to capture cyclic patterns inherent in our time-series datasets. Using a Fast Fourier Transform (FFT)[51]–based spectral analysis, the dominant cycle period is estimated. Subsequently, sinusoidal features are derived via the sine and cosine transformations:

$$\text{sin\_feature} = \sin\left(2\pi \frac{t}{T_{\text{est}}}\right), \quad \text{cos\_feature} = \cos\left(2\pi \frac{t}{T_{\text{est}}}\right) \tag{4.3}$$

where T_{est} is the estimated cycle period and t denotes the time index. These additional features help capture periodic behaviors, enhancing the model's ability to learn and detect deviations from normal cyclic patterns.

4.3.1 Anomaly Injection and Dataset Partitioning

The UCR datasets contain synthetically injected anomalies, ensuring a controlled evaluation environment for anomaly detection models. Each dataset's training set consists only of normal data, while the test set includes exactly one artificially injected anomaly. To ensure consistent evaluation, the dataset partitioning follows a standardized format. Each dataset in the UCR archive follows a structured naming convention. For example, the file:

`152_UCR_Anomaly_PowerDemand1_9000_18485_18821.txt`

provides key dataset attributes:

- Dataset number: 152
- Mnemonic name: PowerDemand1
- Training range: From indices 1–9000 (contains only normal data)
- Anomaly region: Begins at index 18,485 and ends at 18,821 in the test set.

NYC Taxi Anomalies from NAB

In addition to the UCR datasets, this study incorporates the NYC Taxi dataset from the Numenta Anomaly Benchmark (NAB), which provides a real-world scenario for anomaly detection. The NYC Taxi dataset aggregates the total number of taxi passengers in New York City into 30-minute intervals and captures the inherent variability of urban mobility. This dataset is particularly valuable for evaluating anomaly detection models in a streaming data environment, as it reflects both routine fluctuations and abrupt changes associated with significant events.

Anomalies in the NYC Taxi dataset correspond to notable events, including:

- **NYC Marathon (02.11.2014):** An anomaly is observed at `"2014-11-01 19:00:00"` with a passenger count of 28,398, reflecting unusual taxi demand related to the marathon.
- **Thanksgiving (27.11.2014):** At `"2014-11-27 15:30:00"`, the passenger count drops to 15,255, corresponding to altered travel patterns during the holiday.
- **Christmas (24–26.12.2014):** An anomaly is recorded at `"2014-12-25 15:00:00"` with a count of 12,039, capturing the impact of the Christmas season.
- **New Year's Day (01.01.2015):** A significant spike is observed at `"2015-01-01 01:00:00"` with a passenger count of 30,236, indicating heightened taxi activity on New Year's Day.
- **January 2015 North American Blizzard (26–27.01.2015):** At `"2015-01-27 00:00:00"`, the passenger count plummets to 109, reflecting a dramatic decrease in taxi demand due to severe weather.

The raw data for the NYC Taxi dataset is provided by the NYC Taxi and Limousine Commission, and the aggregation into 30-minute buckets facilitates the detection of both gradual and abrupt changes in urban mobility patterns.

Unlike traditional train-validation-test splits, no separate validation set was used due to the scarcity of anomalies. Instead, hyperparameter selection and threshold tuning were performed using metrics computed directly on the test set. While this approach does not strictly separate validation and test data, it was deemed sufficient for evaluating model performance on this specific dataset.

4.4 Evaluation Metrics

The effectiveness of anomaly detection models is assessed using well-established quantitative evaluation metrics. These metrics help determine the model's ability

to correctly classify anomalous and normal instances, particularly in imbalanced datasets where anomalies occur infrequently.

4.4.1 Anomaly Scoring Criteria

Anomaly detection models often differ in how they recognize the onset and offset of anomalies. Some algorithms detect the *leading edge*, while others identify the *trailing edge* of an anomaly window. To account for these variations, each data set is designed to have only one anomaly window and a predetermined training period that is free of anomalies. Our goal is to identify the data point with the highest anomaly score in each dataset. In UCR datasets, If this point falls within a specified detection range (see Figure 4.1), it is counted as a correctly detected anomaly. The overall performance score on the UCR anomaly benchmark is then calculated by taking the percentage of datasets where the anomaly was correctly detected [12].

Furthermore, the dynamic logic governing the adjustment of evaluation boundaries ensures robustness by constraining the anomaly detection within predefined limits. In cases where the predicted anomaly lies outside the extended detection range, the evaluation boundaries are fixed to these established limits, thus maintaining a controlled and consistent framework for anomaly detection. This method offers a balanced compromise between flexibility and control, enhancing the system's ability to detect true anomalies while mitigating the risk of excessive false alarms. Overall, this strategy provides a more forgiving yet rigorously bounded evaluation process, which is critical for optimizing the F1 score by carefully balancing the trade-offs between recall and precision in anomaly detection systems.

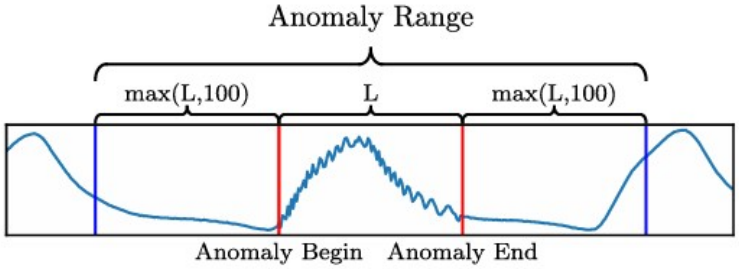

Figure 4.1: Anomaly detection range in the UCR anomaly benchmark data set.

Let the length of the anomaly be defined as:

$$L = \text{end} - \text{begin} + 1 \tag{4.4}$$

An integer prediction P is deemed correct if it lies within:

$$\min(\text{begin} - L,\ \text{begin} - 100) < P < \max(\text{end} + L,\ \text{end} + 100). \tag{4.5}$$

The constants L and 100 ensure that short anomalies (e.g., single-point anomalies) are not unfairly penalized due to minor misalignment. This scoring function, enhanced by the dynamic adjustment of evaluation boundaries, provides a balanced approach, preventing overly strict detection thresholds from distorting evaluation results while offering the flexibility to capture near-boundary deviations effectively.

4.4.2 Binary Classification Metrics

Evaluating anomaly detection models requires metrics that effectively capture the performance in imbalanced settings. In our experiments, we rely on a set of standard metrics derived from the confusion matrix, as well as threshold-independent measures.

For binary classification, the confusion matrix is defined as follows:

Actual / Predicted	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Table 4.2: Binary classification confusion matrix

Definition of Confusion Matrix Components: The following terminology is used to describe the values in the confusion matrix:

- **True Positive (TP)**: The number of positive examples correctly identified by the model.
- **False Negative (FN)**: The number of positive examples incorrectly classified as negative.
- **False Positive (FP)**: The number of negative examples incorrectly classified as positive.
- **True Negative (TN)**: The number of negative examples correctly identified by the model.

Using the confusion matrix, we calculate the following metrics:

- **Accuracy** measures the proportion of total correct predictions (both true positives and true negatives) out of all predictions. It is given by:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.6}$$

However, accuracy can be misleading in imbalanced datasets, where a model may classify most instances as normal (majority class) and still achieve high accuracy while failing to detect anomalies.

- **Precision** Precision, or positive predictive value, indicates the proportion of positive identifications that are indeed correct. In the context of anomaly detection, precision quantifies how many of the detected anomalies are truly anomalous:

$$\text{Precision, } p = \frac{TP}{TP + FP} \tag{4.7}$$

  A high precision score indicates a low false positive rate, meaning that detected anomalies are highly reliable.

- **Recall** (or sensitivity) measures the proportion of actual anomalies that are correctly detected by model:

$$\text{Recall, } r = \frac{TP}{TP + FN} \tag{4.8}$$

  A model with high recall ensures that most anomalies are detected, reducing the risk of missed anomalies.

To succinctly capture the balance between these two metrics, we employ the F1-score, defined as the harmonic mean of precision and recall:

$$\text{F1-score} = 2 \cdot \frac{r \cdot p}{r + p} \tag{4.9}$$

These evaluation strategies are consistent with those recommended in the literature (see [46] for further details). These metrics are particularly useful in anomaly detection due to the inherent imbalance between normal and anomalous samples. High precision indicates a low rate of false alarms, while high recall ensures that most true anomalies are detected. In many real-world applications, balancing these two metrics is essential, as both types of errors can have significant consequences.

Although accuracy may initially seem to be an appropriate measure for performance evaluation, it often fails in the presence of class imbalance—a common scenario in anomaly detection where anomalies are rare compared to normal instances. Relying solely on accuracy may lead to an overly optimistic assessment of a model's performance. Therefore, it is crucial to focus on class-specific metrics such as precision and recall. The F1-score is particularly useful in anomaly detection, where class imbalances make precision and recall equally important.

4.4.3 Area Under Curve (AUC)

Another vital metric for evaluating anomaly detection models is the Area Under Curve (AUC), derived from the Receiver Operating Characteristic (ROC) curve. The ROC curve is a graphical representation that illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at various threshold settings. This curve begins at the origin (0,0) and ends at (1,1), showing how the TPR increases relative to the FPR as the decision threshold is varied. The **Area Under Curve (AUC)** summarizes the ROC curve into a single value, representing the probability that the model will rank a randomly chosen anomalous instance higher than a randomly chosen normal instance. AUC is particularly valuable because it is threshold-independent and robust to class imbalance. To elaborate, let:

- P denote the total number of positive (anomalous) samples.
- N denote the total number of negative (normal) samples.
- TP denote the number of true positives, where anomalous timestamps are correctly identified.
- FP denote the number of false positives, where normal timestamps are erroneously classified as anomalous.

The true positive rate (TPR) is calculated as:

$$\text{TPR} = \frac{TP}{P} \tag{4.10}$$

Similarly, the false positive rate (FPR) is calculated as:

$$\text{FPR} = \frac{FP}{N} \tag{4.11}$$

The AUC is then computed as:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \tag{4.12}$$

A high AUC indicates that the model consistently distinguishes between anomalous and normal samples across different threshold levels.

By varying the threshold used for anomaly detection, we obtain different pairs of TPR and FPR, which together form the ROC curve. The AUC is then computed as the area under this ROC curve. In the context of anomaly detection, the AUC represents the probability that the algorithm will assign a higher anomaly score to a randomly selected anomalous point than to a randomly selected normal point. This

probabilistic interpretation makes the AUC particularly valuable when comparing different anomaly detection methods.

The AUC is widely regarded in the literature as a robust measure for model comparison, particularly in studies where class imbalance is a significant challenge. In our experiments, the AUC is a primary metric for performance evaluation, as it provides a clear and concise indication of the overall effectiveness of the anomaly detection methods.

Advantages of AUC in Anomaly Detection

- **Threshold-independent**: Unlike precision and recall, AUC does not require a fixed classification threshold.
- **Resistant to class imbalance**: AUC remains reliable even when anomalies are rare, making it ideal for anomaly detection tasks
- **Widely used for benchmarking**: AUC is one of the most commonly reported metrics in anomaly detection research.

Several recent studies [46] have employed AUC as a benchmark metric, reinforcing its effectiveness in model comparison.

4.5 Implementation Framework

4.5.1 HTM Model Implementation

This section outlines the technical implementation of the Hierarchical Temporal Memory (HTM) model, including data preprocessing, encoding, spatial pooling, temporal memory processing, and anomaly scoring mechanisms.

Data Preprocessing and Setup

we first normalize each time series using min-max normalization (Equation 4.2). For consistency, we feed the data as individual points to the HTM network in an online manner.

1. **Spatial Pooler (SP):** Learns spatial patterns and encodes input data into Sparse Distributed Representations (SDRs).
2. **Temporal Memory (TM):** Captures sequential dependencies and generates predictions for anomaly detection.

The final anomaly score is computed based on deviations from predicted sequences, and a dynamic thresholding mechanism adapts to the distribution of the input data.

- **Data Loading and Preprocessing:** The UCR anomaly dataset is loaded and preprocessed to extract the univariate time-series data. Ground truth labels are created to mark known anomaly regions, providing a basis for later evaluation.
- **Encoder Configuration:** We use an RDSE with parameters for resolution, size, and sparsity as outlined in our parameter dictionary. These parameters are chosen to ensure that the encoded SDRs effectively represent the input data while maintaining a low overlap between distinct inputs.
- **Spatial Pooler (SP) Setup:** The SP is configured with parameters such as `columnCount`, `localAreaDensity`, `potentialPct`, and synaptic permanence adjustments. These parameters allow the SP to form a robust set of active mini-columns that serve as a stable representation of the input.
- **Temporal Memory (TM) Setup:** The TM is configured to learn sequences with parameters including `cellsPerColumn`, `activationThreshold`, and `minThreshold`. It processes the output from the SP and generates predictions for future inputs. The prediction error is computed as the absolute difference between the predicted value and the actual input.
- **Anomaly Likelihood Calculation:** An anomaly likelihood is computed by combining the raw anomaly score from the TM with a dynamically determined threshold. A moving window average and a lambda multiplier are used to calculate this threshold, which is then applied to classify each data point as normal or anomalous.

CLA Parameters

HTMs can be configured using several meta-parameters, with the Numenta framework allowing adjustments to 14 different parameters. A random sweep of these parameters has shown that their values have a significant impact on the behavior of the algorithm. However, the relationship between parameter selection and model performance is not straightforward. As of the time of writing, there are few established heuristics to determine the optimal values for these parameters in advance.

Among these, the number of columns and neurons per column play a crucial role in performance, as they define the connectivity structure of the model's neural network and directly influence its capacity to learn complex patterns. However,

the impact of many other parameters is hard to isolate, as they interact with each other in ways that are difficult to predict.

In our implementation, the Spatial Pooler (SP) and Temporal Memory (TM) components utilize the following parameters:

Spatial Pooler (SP) Parameters:

- **columnCount:** Specifies the total number of columns in the SP, determining the resolution of the input encoding. A higher columnCount provides a more granular representation.
- **localAreaDensity:** Defines the desired proportion of active columns within a local neighborhood, influencing the sparsity and distribution of activations.
- **potentialPct:** The percentage of the input space that each column can potentially connect to, controlling the degree of overlap between columns.
- **globalInhibition:** Determines whether inhibition is computed globally across the entire SP or locally within (each active column inhibits its neighbors). Global inhibition selects the most active columns from the entire region, while local inhibition restricts competition to a defined area.
- **synPermActiveInc:** The increment value applied to the permanence of synapses that are active, reinforcing frequently active connections.
- **synPermInactiveDec:** The decrement value applied to the permanence of inactive synapses, facilitating the removal of seldom-used connections.
- **synPermConnected:** The permanence threshold a synapse must exceed to be considered connected, defining effective connectivity between columns and the input.
- **boostStrength:** A factor that increases the activation probability of under-active columns, ensuring that less active columns contribute to learning over time.

Temporal Memory (TM) Parameters:

- **cellsPerColumn:** The number of neurons allocated to each column, allowing each column to represent multiple temporal contexts.
- **activationThreshold:** The minimum number of active synapses required on a dendritic segment for that segment to be considered active, thus placing a requirement on the confidence of predictions.

- **initialPermanence:** The starting permanence value assigned to new dendritic synapses, establishing a baseline for subsequent learning.
- **connectedPermanence:** The permanence threshold that determines whether a dendritic synapse is considered connected; only synapses above this threshold contribute to activation.
- **minThreshold:** The minimum number of active synapses needed for a dendritic segment to be considered during prediction, preventing segments with insufficient evidence from influencing outcomes.
- **maxNewSynapseCount:** The maximum number of synapses that can be added to a newly formed dendritic segment, controlling the initial connectivity complexity.
- **permanenceIncrement:** The fixed amount by which the permanence value of an active synapse is increased during learning, reinforcing reliable connections.
- **permanenceDecrement:** The fixed amount by which the permanence value of an inactive synapse is decreased during learning, allowing the model to gradually forget less important connections.
- **maxSegmentsPerCell:** The maximum number of dendritic segments that each cell can form, limiting the cell's capacity for encoding different temporal contexts.
- **maxSynapsesPerSegment:** The maximum number of synapses allowed on a single dendritic segment, controlling the complexity of each segment's connectivity.
- **newSynapseCount:** The default number of synapses added when a new dendritic segment is created, which determines the initial connectivity of the segment.

Model Training and Optimization

During training, the model begins by normalizing the raw input data and deriving additional cyclical features using sine and cosine functions based on an estimated cycle period. These features are encoded into sparse distributed representations (SDRs) via multiple RDSE encoders and concatenated into a composite SDR, which is then fed into the Spatial Pooler (SP) that learns to extract stable, sparse patterns from the input. The resulting SP output is subsequently processed by the Temporal Memory (TM) component, which learns the temporal transitions and sequences by adjusting its synaptic connections according to the observed patterns.

Concurrently, a Predictor module is trained to forecast future input values based on the active cells from the TM, integrating spatial and temporal learning to build a robust representation of the data's dynamics.

Anomaly Scoring

The model computes its anomaly score by first using the Temporal Memory (TM) component, which compares the predicted active cells with the actually active ones at each time step to generate an instantaneous anomaly score—reflecting the degree of mismatch between the model's prediction and the observed input. This raw score is then passed to an AnomalyLikelihood module that, over a specified period, aggregates historical anomaly statistics to calculate a probabilistic anomaly likelihood, thereby smoothing out transient fluctuations and flagging statistically significant deviations as anomalies.

Threshold Selection and Hyperparameter Tuning

Threshold Selection: In the context of this model, threshold selection is a critical step to determine which anomaly likelihood scores are significant enough to be flagged as anomalies. After computing the instantaneous anomaly scores from the Temporal Memory (TM), these raw scores are processed by an AnomalyLikelihood module that smooths out short-term fluctuations by maintaining a history of anomaly scores. The model then calculates the mean and standard deviation of these likelihood scores over a specified period, and sets the anomaly threshold as the mean plus two times the standard deviation. This statistical approach assumes that most normal data points fall within a typical range, so scores exceeding this threshold are statistically rare and thus likely to represent true anomalies.

Hyperparameter tuning is the process of optimizing the key parameters that influence the model's performance across its different components—such as the encoders, Spatial Pooler (SP), Temporal Memory (TM), and anomaly detection modules. Given the extensive set of hyperparameters available in table 4.3, we initially employed a random search strategy to broadly explore the parameter space. This approach allowed us to quickly identify which parameters had the most significant impact on the model's performance by sampling a wide range of values, thus reducing the complexity of the search space before moving to a more focused tuning method.

After narrowing down the most influential hyperparameters, we transitioned to a grid search to systematically evaluate combinations of parameters, including encoder resolution, number of columns in the SP, anomaly likelihood period, activation thresholds, the number of cells per column, and the minimum threshold for TM segments. Each configuration was assessed using performance metrics like

Hyperparameter	**Original Value**	Model Value
sp.boostStrength	3.0	[2, 3, 4, 5, 7]
sp.columnCount	1638	[512, 1024, 1638, 2048, 4096]
sp.localAreaDensity	0.04395604395604396	[0.05]
sp.potentialPct	0.85	[0.80, 0.85, 0.90]
sp.synPermActiveInc	0.04	[0.40, 0.41]
sp.synPermConnected	0.13999999999999999	[0.13, 0.139999, 0.14]
sp.synPermInactiveDec	0.006	[0.006, 0.01]
tm.activationThreshold	17	[10, 12, 13, 15, 16, 17]
tm.cellsPerColumn	13	[10, 13, 32, 64, 128, 256]
tm.initialPerm	0.21	0.21 for all datasets
tm.maxSegmentsPerCell	128	[128, 256, 512] for all datasets
tm.maxSynapsesPerSegment	64	[32, 64]
tm.minThreshold	10	[5, 10]
tm.newSynapseCount	32	[32, 64]
tm.permanenceDec	0.1	[0.05, 0.1]
tm.permanenceInc	0.1	0.1 for all datasets
anomaly.period	1000	[50, 80, 100, 160, 250,500, 600]
enc.resolution	0.88	[0.25, 0.6, 0.75, 0.88, 0.9]
enc.size	700	700 for all datasets
enc.sparsity	0.02	0.02 for all datasets
predictor.sdrc_alpha	0.1	0.1 for all datasets

Table 4.3: Hyperparameter Configuration for the HTM Model

the F1 score, ROC AUC, precision, recall, and accuracy. By selecting the best configuration based on an optimal trade-off among these metrics, this two-stage approach ensures that the model is finely calibrated to learn stable representations, accurately predict future inputs, and robustly detect anomalies. It is important to note that key hyperparameters—including were found to be dataset-dependent, reflecting the diverse temporal characteristics and complexities of the different datasets.

Data Split

In the UCR dataset, the training and test ranges are pre-specified, with defined segments for training and anomaly labeling in the test data. However, since the model is highly sensitive to the choice of data range, we experimented with different splits to determine how varying the range affects performance. This sensitivity testing was similarly applied to the pure dataset, ensuring that our

evaluation captured the nuances of how range selection influences anomaly detection and prediction accuracy. Notably, this issue also occurred for the pure dataset, highlighting that the sensitivity to the selected range is a critical factor regardless of the dataset used.

HTM's Multi-Step Prediction

After splitting dataset into train and test set, during training the model learns temporal patterns and builds representations using a composite SDR that incorporates both the raw data and engineered time features, while the Predictor module is trained to forecast upcoming values. In the testing phase, the model generates multi-step predictions (e.g., 1-step and 5-step ahead) from the unseen test data, and these predictions are then shifted to align correctly with the corresponding input points, ensuring that the performance evaluation accurately reflects the model's ability to predict future values over multiple time steps.

4.5.2 SLADiT Model Implementation

SLADiT offer robust performance in reconstructing normal patterns and identifying deviations that signal anomalies. Unlike traditional anomaly detection methods, which rely on predefined statistical thresholds, SLADiT utilize self-learned latent space representations to adaptively detect deviations from expected behavior. This section provides a detailed explanation of the SLADiT model implementation, covering data preprocessing, model architecture, training process, and evaluation techniques.

Data Preprocessing and Preparation

The time-series dataset is first loaded and standardized to facilitate stable training. Normalization is applied by subtracting the mean and dividing by the standard deviation, ensuring that all values fall within a standardized range. In addition to normalization, and later for converting univariate data to multivariate, the dataset enriched with cyclic features— sine and cosine components derived from an FFT-based cycle estimation—to capture periodic patterns inherent in time-series signals.

Following normalization, the dataset is segmented into overlapping time windows to allow the model to capture temporal dependencies. A range of window sizes (including 85, 90, 100, 110, and 150 time steps) was considered, with the optimal size selected based on hyperparameter tuning. Each window is labeled as anomalous if any data point within it corresponds to a known anomaly region. This segmentation strategy facilitates the model's learning of normal sequential patterns and enhances its ability to detect deviations during testing.

Model Architecture

The Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT) model is designed to effectively capture temporal patterns and reconstruct normal sequences. Its architecture comprises an encoder-decoder LSTM network augmented with a KL-divergence sparsity constraint on the latent space to enforce feature selectivity. The components are as follows:

- Encoder: Three stacked LSTM layers process the input sequence and encode it into a lower-dimensional latent representation.
- Latent Space: A fully connected layer transforms the encoder's output into a compact latent representation, capturing essential features.
- Decoder: A mirrored LSTM-based decoder reconstructs the input time-series sequence from the latent space.
- Output Projection: The decoder's hidden states are mapped back to the original feature space through a fully connected layer.

The KL-divergence sparsity constraint is applied to the latent space to limit neuron activations. This regularization forces the network to learn only the most relevant features, which is critical for robust anomaly detection.

Model Training and Optimization

The model is trained using the AdamW optimizer, which facilitates efficient weight updates while mitigating overfitting through decoupled weight decay. A dynamic learning rate schedule is employed via ReduceLROnPlateau, which reduces the learning rate when the validation loss stagnates for a set number of epochs.

The overall loss function consists of two key components:

1. **Mean Squared Error (MSE) loss** to minimize the difference between the input and its reconstruction.
2. **KL-Divergence Regularization** to enforce sparsity in the latent space.

The total **loss function** is computed as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{KL}} \tag{4.13}$$

Training is performed exclusively on normal sequences so that the model learns the typical patterns of the data.

Anomaly Detection and Threshold Selection

Once trained, the model is applied to the test dataset, where it reconstructs input sequences and computes reconstruction errors. Higher errors indicate deviations from learned patterns, suggesting the presence of anomalies. A thresholding method is used to classify anomalies, computed as:

$$T = \mu_{\text{err}} + 2\sigma_{\text{err}} \tag{4.14}$$

where T is the threshold, μ_{err} is the mean reconstruction error, and σ_{err} is the standard deviation. A time window is classified as anomalous if its reconstruction error exceeds T.

Evaluation Metrics and Performance Analysis

The performance of the SLADiT model is evaluated using standard anomaly detection metrics that are especially suited for imbalanced datasets. Instead, we focus on precision, recall, F1-score and Accuracy, which together provide a balanced view of the model's ability to correctly identify anomalies without incurring an excessive rate of false alarms. In this context, precision measures the proportion of detected anomalies that are indeed true anomalies, while recall assesses the fraction of actual anomalies that the model successfully detects. The F1-score, being the harmonic mean of precision and recall, offers a single measure that encapsulates the trade-off between these two metrics. In addition, we compute the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) to capture the model's performance across a range of threshold values, thus providing a threshold-independent evaluation. A higher AUC indicates that the model is more effective at ranking anomalous instances higher than normal ones, a critical capability when anomalies are rare.

Visualization, Characterization, and Model Interpretation

Visualization plays a crucial role in understanding and interpreting the performance of the anomaly detection system, providing both qualitative insights and diagnostic feedback. A comprehensive set of visualization techniques is employed not only to assess the reconstruction error distribution and latent space separation but also to characterize the nature, duration, and severity of detected anomalies in detail.

One essential plot is the *Reconstruction Error Distribution*, which displays the distribution of reconstruction errors for both normal and anomalous samples. This plot helps us understand how the model differentiates between expected patterns and deviations. A clear separation between the error distributions of normal and anomalous data indicates that the model effectively captures the underlying patterns,

whereas significant overlap might suggest difficulties in distinguishing anomalies, prompting further investigation into threshold selection or model capacity.

Another critical visualization is the *Latent Space Visualization* achieved via t-SNE. This technique reduces the high-dimensional latent representations into a two-dimensional plot, where each point represents a compressed version of a time window. By coloring these points according to their true labels (normal or anomalous), we can assess the clustering behavior of the latent space. Ideally, normal samples should form distinct clusters separate from the anomalous ones, revealing how well the model has learned to compress and differentiate the underlying features of the time series. This visualization not only validates the feature extraction capability of the encoder but also helps identify any overlapping regions that may correspond to borderline cases.

The *Time-Series Overlay with Anomalies* plot superimposes the detected anomalies onto the original time-series data. This overlay provides an intuitive, visual means of verifying the alignment between the model's predictions and the actual anomaly occurrences. By highlighting the segments where the reconstruction error exceeds the established threshold, this plot allows us to visually inspect the consistency, timing, and duration of detected anomalies. Such an inspection is crucial for understanding whether the model captures isolated spikes or contiguous anomalous segments, and whether the predictions align well with known anomalies.

In addition to these plots, the *plot anomaly segments* function is used to visually highlight the segments of the time series where anomalies occur. This plot overlays the original time-series data with highlighted intervals corresponding to the detected anomalous segments, allowing for an intuitive comparison between the detected anomalies and the ground truth. It serves to reveal whether the model is capturing contiguous anomalous behavior or merely isolated spikes.

Similarly, the *plot anomaly durations* function presents a bar chart that displays the duration of each detected anomaly segment. This visualization is critical for understanding the temporal extent of anomalies, providing insights into whether the anomalies are short, abrupt deviations or prolonged periods of abnormal behavior.

The *plot anomaly magnitudes* function employs box plots to compare the magnitudes of anomalies across segments by illustrating the distribution of the maximum excess reconstruction error over the threshold within each anomaly.This characterization helps quantify the severity of anomalies, indicating how pronounced each anomaly is relative to the normal behavior.

Beyond these, additional plots further enrich the evaluation framework. The *ROC Curve* (Receiver Operating Characteristic) plot illustrates the trade-off between the true positive rate and false positive rate across various threshold settings. The *AUC (Area Under the ROC Curve)* quantifies the overall ability of the model to rank anomalous instances higher than normal ones, serving as a threshold-independent measure of performance. Complementing this, the *Precision-Recall*

(PR) Curve plot highlights the relationship between precision and recall, which is particularly important in imbalanced datasets where the number of anomalies is low. These curves help in selecting an appropriate threshold that balances the trade-offs between false positives and false negatives.

Finally, the *Confusion Matrix* is visualized as a heatmap that provides a detailed breakdown of the classification outcomes, showing the number of true positives, false negatives, false positives, and true negatives. This plot not only reinforces the numerical evaluation of the model's performance but also aids in identifying specific areas where the model might be misclassifying data, thereby guiding further refinements in the detection approach.

Together, these visualization techniques form a comprehensive characterization framework that not only supports quantitative evaluation but also deepens qualitative insights into the nature, duration, and severity of anomalies detected by the model.

Hyperparameter Tuning Strategy

A grid search was conducted over a range of hyperparameters to identify the best combination that maximized anomaly detection performance. The tuning process involved iteratively training the model with different hyperparameter values and evaluating its performance based on the F1-score and AUC on the test data. The key hyperparameters tuned included:

- `seq_length`: Defines the number of time steps in each input window. Using windows of an appropriate length is critical—too short a window may lose temporal dependencies, while too long a window increases computational overhead without significant performance gains.

- `hidden_dim`: Determines the number of units in each LSTM layer, which affects the model's capacity to capture complex temporal patterns. A balanced number of hidden units ensures sufficient representation power without causing overfitting or instability during training.

- `latent_dim`: Controls the size of the compressed feature representation in the latent space. This parameter is crucial for preserving essential features while minimizing redundancy; an appropriately sized latent space allows the model to focus on the most discriminative characteristics of the input data.

- `sparsity_lambda`: Adjusts the strength of the KL-divergence regularization applied to the latent space. A well-chosen sparsity weight (0.0001) encourages the network to activate only a small subset of neurons, reducing noise and preventing overfitting, thereby promoting the learning of only the most relevant features.

- `sparsity_target`: Specifies the desired level of neuron activation in the latent space. A target of 5% neuron activation has been found optimal, as it strikes a balance between sparsity and retaining sufficient information to reconstruct normal patterns accurately.
- `lr`: Governs the step size for weight updates during training. With a learning rate of 0.001 (using the AdamW optimizer), the model achieves efficient convergence; lower values can slow down training progress, whereas higher values may lead to unstable gradients.
- `batch_size`: Determines how many samples are processed before the model's parameters are updated. A batch size of 64 provides a good trade-off between computational efficiency and the stability of gradient updates—smaller batches may introduce high variance, and larger batches can impose excessive memory demands.
- `n_epochs`: Defines the total number of passes over the training data. Training for 100 epochs was necessary to ensure the model fully converged; using fewer epochs resulted in unstable reconstruction and sparsity objectives, leading to suboptimal anomaly detection performance.

Each hyperparameter combination in table 4.4 was evaluated by training the SLADiT model on normal data and computing the reconstruction error on the test set. The model was then tested using anomaly detection thresholds, and its classification performance was assessed using F1-score and AUC.

Hyperparameter Selection Justification

After systematically testing multiple hyperparameter combinations, the best-performing configuration was determined as follows:

To optimize the performance of the SLADiT model, we conducted a systematic grid search over a range of hyperparameters. This process involved iteratively training the model on normal data and evaluating its performance on the test set using metrics such as the F1-score and AUC. It is important to note that key hyperparameters—including the sequence length, hidden dimension, and latent dimension—were found to be dataset-dependent, reflecting the diverse temporal characteristics and complexities of the different datasets.

For instance, for the DISTORTEDTkeepThirdMARS dataset, the best results were achieved with a sequence length of 100, a hidden dimension of 128, and a latent dimension of 64. In contrast, for datasets such as PowerDemand1, NYC Taxi (from NAB), and InternalBleeding6 as well as NOISEGP711MarkerLFM5z3, the optimal configuration was a sequence length of 100, with both hidden and latent dimensions set to 128. Meanwhile, the ECG4 dataset required a slightly longer

sequence length of 112, while maintaining 128 units for both the hidden and latent layers. Furthermore, for DISTORTEDWalkingAceleration5, a longer sequence of 150 time steps and a higher hidden dimension of 256 (with a latent dimension of 128) were necessary to capture the complex temporal dependencies present in the data. Finally, for the MesoplodonDensirostris dataset, a sequence length of 110, combined with 128 units in both the hidden and latent layers, yielded the best performance.

Other hyperparameters such as the sparsity parameters (sparsity_lambda at 0.0001 and sparsity_target at 0.05), learning rate (0.001 using the AdamW optimizer with dynamic adjustment via ReduceLROnPlateau), batch size (64), number of epochs (100), number of LSTM layers (3), and dropout rate (0.2) were kept consistent across datasets. These settings were chosen to balance computational efficiency, training stability, and the model's ability to generalize.

This tuning strategy, which accounts for dataset-specific characteristics, ensures that the SLADiT model is tailored to capture the unique temporal patterns and anomaly structures of each dataset, thereby optimizing detection performance.

Hyperparameter	**Chosen Value**
Sequence Length (`seq_length`)	[85,100,110,150]
Number of Hidden Units (`hidden_dim`)	[64,128]
Latent Dimension (`latent_dim`)	[64,128]
Sparsity Weight (`sparsity_lambda`)	0.0001 for all datasets
Sparsity Target (`sparsity_target`)	0.05 for all datasets
Learning Rate (`lr`)	0.001 for all datasets
Number of Epochs (`n_epochs`)	100 for all datasets
Batch Size (`batch_size`)	64 for all datasets
Number of LSTM Layers (`num_layers`)	3 for all datasets
Dropout Rate (`dropout`)	0.2 for all datasets

Table 4.4: Hyperparameter Configuration for the SLADiT Model.

4.5.3 Data Split and Validation Considerations

Due to the scarcity of anomalies in the UCR and NYC Taxi datasets, no separate validation set was used in this study. Instead, hyperparameter selection and threshold tuning were performed using metrics computed directly on the test set. While this approach does not strictly separate validation and test data, it was deemed sufficient for evaluating model performance on these specific datasets. Given that the test set contains only one known anomaly region, we focus on performance metrics such as F1-score and AUC to assess the model's ability to distinguish

between normal and anomalous sequences. By following this approach, we ensure that hyperparameter optimization does not artificially inflate test performance, maintaining the integrity of our anomaly detection results.

4.6 Computational Feasibility and Challenges

Limitations of HTM in Univariate Data

Although HTM has demonstrated strong performance in detecting complex, multivariate anomalies, our experiments indicate performance limitations in univariate settings. The challenges observed include:

- Reduced effectiveness in detecting subtle anomalies due to limited spatial correlations in univariate time series.
- High sensitivity to hyperparameter tuning, making it challenging to generalize across different datasets.

In comparison, Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT) model has demonstrated greater robustness in univariate anomaly detection, likely due to their ability to capture feature-level dependencies in latent space representations [38] [39]. HTM, on the other hand, is inherently designed to capture spatial correlations, which are more pronounced in multivariate settings, leading to reduced effectiveness in univariate anomaly detection.

Despite these limitations, HTM remains a powerful model for real-time anomaly detection, particularly in scenarios where sequential pattern learning is critical. The architecture's emphasis on sparse, noise-resistant representations and dynamic adaptation presents a unique approach to time-series modeling. However, our experiments indicate that while HTM excels in multivariate anomaly detection, it underperforms in univariate scenarios compared to SLADiT model. This discrepancy is likely due to HTM's reliance on spatial correlations, which are less informative when analyzing single-variable time series [38].

Training and Evaluation Pipeline

For our experiments, both HTM and SLADiT models were trained on the UCR anomaly dataset and subsequently evaluated on a separate test set. The evaluation pipeline computed various performance metrics, including F1-score, AUC (Area Under the Curve), and precision-recall metrics, offering a comprehensive framework for performance assessment. While accuracy is a commonly used metric, it can be misleading in imbalanced datasets, where anomalies are significantly fewer than normal observations. Instead, precision, recall, F1-score, and AUC provide a more

nuanced assessment of how well each model distinguishes anomalies from normal patterns, particularly in real-world applications where false detections can lead to significant operational costs.

Challenges and Limitations in Benchmarking CLA and HTM Models

While the UCR dataset provides a strong benchmarking foundation, several challenges arise when evaluating Cortical Learning Algorithm (CLA) and Hierarchical Temporal Memory (HTM) models. These challenges primarily stem from variability in dataset characteristics, model sensitivity to hyperparameters, and differences in anomaly scoring mechanisms.

Performance Variability Across Datasets

Different datasets exhibit widely varying characteristics, such as noise levels, temporal dependencies, and anomaly distributions. These variations introduce inconsistencies in model performance, requiring extensive hyperparameter fine-tuning for each dataset.

- Inconsistent Performance Across Datasets: A model that achieves high anomaly detection accuracy on one dataset may struggle on another, particularly if the dataset exhibits different temporal structures or anomaly patterns differ.
- Generalization Challenges: The need to fine-tune hyperparameters separately for each dataset complicates the deployment of a universal, one-size-fits-all anomaly detection model. This limitation is particularly pronounced in HTM models, which rely heavily on the stability of learned temporal patterns.

Addressing these challenges requires the development of adaptive learning mechanisms that allow anomaly detection models to dynamically adjust to different datasets without requiring extensive manual tuning.

Benchmarking Issues in HTM Models

HTM models, particularly CLA-based implementations, face unique benchmarking challenges due to their biologically inspired learning mechanisms. Unlike traditional deep learning models, HTM algorithms do not use explicit backpropagation, complicating hyperparameter optimization. The following challenges were encountered:

Sensitivity to Hyperparameters

HTM models exhibit high sensitivity to hyperparameter configurations, significantly impacting detection accuracy. Unlike SLADiT models, where hyperparameter

tuning can be efficiently handled using grid search, HTM requires randomized search techniques due to its large parameter space. Key influential hyperparameters include:

- Synaptic Permanence Thresholds: These parameters control how quickly learned patterns are reinforced or forgotten. If tuned incorrectly, the model can become overly sensitive to noise or slow in adapting to new patterns.
- Boosting Factors: Designed to encourage neuron activation diversity, boosting requires careful balance—higher values increase learning efficiency but introduce instability in anomaly scoring.
- Prediction Confidence Thresholds: This parameter determines how anomaly scores are assigned. Lower thresholds lead to higher false positives, while overly restrictive thresholds may miss subtle anomalies.

Since HTM models dynamically update their representations in response to new data, their performance is highly dataset-dependent, requiring frequent re-optimization.

Opaque Scoring Functions and Interpretation Challenges

HTM models rely on biologically inspired prediction errors for anomaly scoring, which often lack clarity compared to explicit probability-based or reconstruction error-based scores. This results in:

- Different Implementations May Use Different Scoring Approaches: Some HTM implementations focus on raw anomaly scores, while others use anomaly likelihood functions computed over moving windows.
- Temporal Uncertainty in Anomaly Timing: Different HTM configurations may detect anomalies at different time offsets (e.g., leading edge vs. trailing edge anomalies), complicating the alignment of detection results with ground truth labels.

Addressing these issues requires developing standardized evaluation frameworks that ensure consistency in HTM benchmarking.

Computational Costs and Feasibility

Beyond accuracy, the feasibility of an anomaly detection model is determined by its computational efficiency—including the total training time and inference speed shown in table 4.5.

	Multivariate		**Univarite**	
	Total_Time		**Total_Time**	
Datasets|Metrics	SLADiT	HTM	SLADiT	HTM
3D Marker	1669.96	78	162.11	64
NASA	1504.85	67	1497.16	75
InternalBleeding6	1294	64	2868.97	71
NYC Taxi(NAB)	1562.29	92	3130.93	98
Walking	2662.25	120	2603.05	125
Whale	3185.23	121	3936.95	123
ECG4	4197.51	125	4292.61	127
PowerDemand1	4311.19	126	4046.82	128

Table 4.5: Models Comparison by Total Run Time

The HTM model and SLADiT model exhibited notable differences in computational demands:

- **Hierarchical Temporal Memory (HTM)**
  - Strengths: HTM operates in a continuous learning mode, adapting to real-time data streams without retraining.
  - Challenges: Computationally expensive due to large-scale synaptic updates and high-dimensional sparse representations. The memory requirements increase proportionally to sequence length, making large-scale deployment challenging.
- **Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT)**
  - Strengths: SLADiT leverage efficient batch processing, enabling faster training and inference times. The grid search for hyperparameter tuning was computationally feasible compared to the randomized search required for HTM.
  - Challenges: Despite its efficiency, SLADiT models still require considerable GPU resources for training, especially when using longer sequence lengths and larger batch sizes.

Both models present trade-offs in terms of real-time applicability. HTM models require continuous processing power, while SLADiT offer batch-based inference that can be optimized for efficiency with careful management of retraining intervals. Having established the dataset structure, evaluation framework, and computational considerations, the next chapter presents the empirical results obtained from

applying HTM and SLADiT to the UCR benchmark datasets. The following subsequent analyses will focus on anomaly detection accuracy, false positive rates, and reconstruction error distribution to assess the practical effectiveness of these models.

Chapter 5

Experimental Results

This chapter presents the empirical evaluation of the Sparse Autoencoder (SLADiT) and Hierarchical Temporal Memory (HTM) models on multiple datasets from the UCR Anomaly Archive and the NYC Taxi dataset from the NAB collection. We analyze their performance using standard anomaly detection metrics, examine the reconstruction error distributions, visualize latent space representations, and provide an intuitive understanding of the models' effectiveness through time-series anomaly detection visualizations.

For our experiments, we set up a benchmark pipeline in Python following the methods described in Sections 3.1.2 and 3.2.2. We used publicly available code from the HTM.core GitHub repository for some anomaly detection methods, and another model ,SLADiT which didn't have a Python version, we implemented them ourselves. We built the Sparse LSTM Autoencoder model using PyTorch.

To fine-tune the hyperparameters for each model, we ran 100 training epochs on seven selected time series from the UCR Anomaly Archive and the NYC Taxi dataset from the NAB collection, optimizing for the best F1 score. You can find a complete list of the hyperparameters from our search in Table 4.3 and Table 4.4. All the experiments were executed on an NVIDIA GeForce MX350, with memory allocation ranging roughly between 280MB and 570MB on SLADiT, depending on the dataset.

5.1 Anomaly Detection Performance

5.1.1 Quantitative Performance

The performance of the SLADiT and HTM models was evaluated using standard metrics—F1 score, AUC (Area Under the ROC Curve), and accuracy—across both univariate and multivariate time-series datasets from the UCR Archive and the

NYC Taxi dataset from NAB.

	SLADiT			**HTM**		
Datasets|Metrics	F1	AUC	Acc	F1	AUC	Acc
NASA[1]	0.97	0.99	0.99	0.04	0.45	0.64
Walking[2]	0.89	0.99	0.99	0.12	0.84	0.89
PowerDemand1[3]	0.88	0.96	0.99	0.01	0.85	0.88
InternalBleeding6[4]	0.85	0.98	0.98	0.06	0.67	0.84
3D Marker[5]	0.83	0.93	0.99	0.03	0.64	0.61
ECG4[6]	0.71	0.95	0.99	0.03	0.65	0.56
NYC Taxi(NAB)	0.70	0.97	0.93	0.05	0.54	0.94
Whale[7]	0.66	0.93	0.99	0.03	0.69	0.86

Table 5.1: Performance Metrics Comparison of SLADiT and HTM on Univariate Time Series.

	SLADiT			**HTM**		
Datasets|Metrics	F1	AUC	Acc	F1	AUC	Acc
NASA	0.97	0.99	0.99	0.23	0.81	0.94
Walking	0.88	0.98	0.99	0.32	0.66	0.95
PowerDemand1	0.83	0.97	0.99	0.12	0.70	0.89
ECG4	0.77	0.97	0.99	0.29	0.64	0.83
NYC Taxi(NAB)	0.76	0.97	0.94	0.20	0.65	0.95
3D Marker	0.49	0.90	0.98	0.20	0.44	0.94
Whale	0.48	0.98	0.97	0.21	0.93	0.94
InternalBleeding6	0.47	0.72	0.95	0.15	0.77	0.89

Table 5.2: Performance Metrics Comparison of SLADiT and HTM on Multivariate Time Series.

Univariate Results: As shown in Table 5.1, the SLADiT model consistently outperforms HTM in the univariate setting. SLADiT achieves F1 scores ranging

[1] 052_UCR_Anomaly_DISTORTEDTkeepThirdMARS_3500_4711_4809

[2] 054_UCR_Anomaly_DISTORTEDWalkingAceleration5_2700_5920_5979

[3] 152_UCR_Anomaly_PowerDemand1_9000_18485_18821

[4] 142_UCR_Anomaly_InternalBleeding6_1500_3474_3629

[5] 097_UCR_Anomaly_NOISEGP711MarkerLFM5z3_5000_5948_5993

[6] 123_UCR_Anomaly_ECG4_5000_16800_17100

[7] 151_UCR_Anomaly_MesoplodonDensirostris_10000_19280_19440

from 0.657 to 0.974, with AUC values consistently above 0.93 and accuracy levels close to 0.93. In contrast, HTM's performance is substantially lower, with F1 scores ranging between 0.010 and 0.120, and AUC values from 0.450 to 0.850. As detailed in Chapter 3, HTM was adapted to process univariate data using only a scalar numeric encoder, which removed the essential time-based cyclic encoding. The loss of cyclic features critical for distinguishing temporal patterns likely contributed to HTM's poor performance in these experiments.

Multivariate Results: Table 5.2 summarizes the performance on multivariate datasets. In this setting, both models show improved performance relative to their univariate results; however, SLADiT still maintains a significant edge over HTM. SLADiT's F1 scores in multivariate experiments range from 0.469 to 0.969, with AUC values consistently high (around 0.969–0.990) and accuracy nearly perfect. HTM's multivariate performance improves (F1 scores between 0.150 and 0.320), yet it remains substantially lower than SLADiT's results on most datasets. Even after incorporating cyclic features via FFT-based feature engineering—where sine and cosine components were added to capture periodic patterns—HTM's performance improved only marginally. This limited improvement is likely due to HTM's sensitivity to a large number of hyperparameters, making it difficult to fine-tune optimally for multivariate settings. For instance, on the 054_UCR_DISTORTEDWalkingAceleration5 dataset, SLADiT achieves an F1 score of 0.879 compared to HTM's 0.320, highlighting SLADiT's superior ability to capture complex interdependencies when multiple features are available(see Figures 5.1 and 5.2).

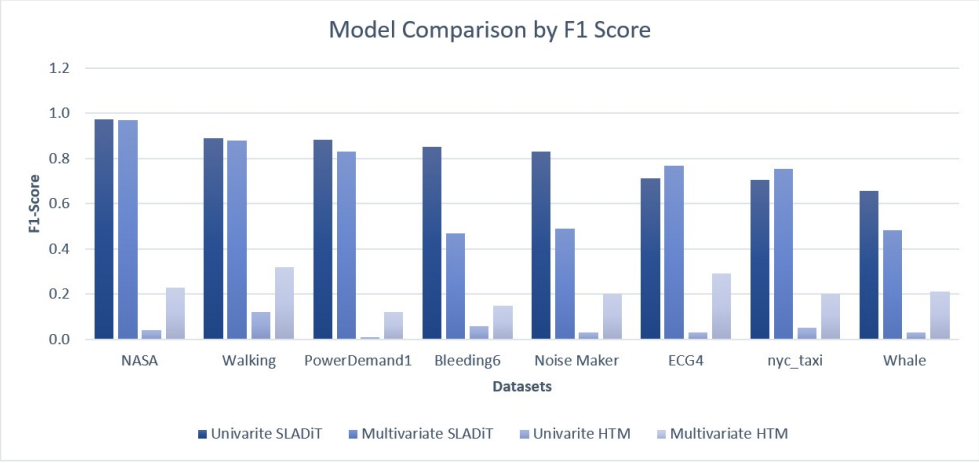

Figure 5.1: Model Comparison by F1 Score

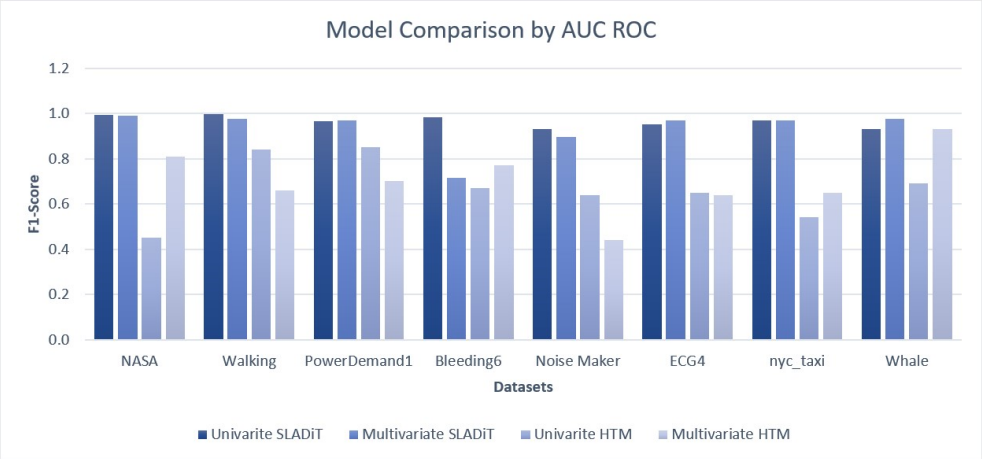

Figure 5.2: Model Comparison by AUC

Figure 5.3 shows the confusion matrix at the best-performing threshold, illustrating the distribution of true positives, false positives, false negatives, and true negatives.

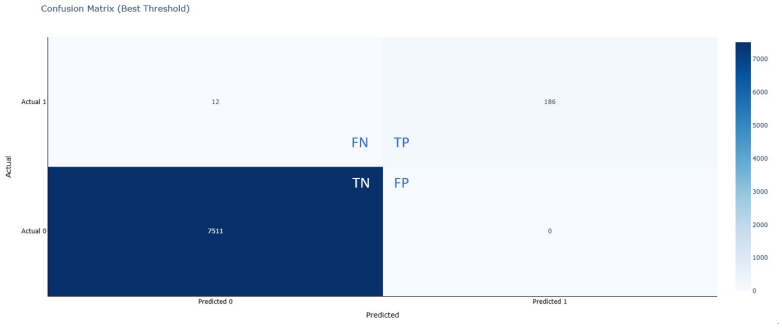

Figure 5.3: Confusion Matrix for DISTORTEDTkeepThirdMARS for SLADiT

Model Performance Curves

Figure 5.4 depicts the ROC curve (left) and the Precision-Recall curve (right) for DISTORTEDTkeepThirdMARS in SLADiT (multivariate), illustrating how the model's performance varies with different decision thresholds. The high AUC and steep precision-recall curve reflect robust anomaly detection capabilities.

Overall, the experimental results demonstrate that the SLADiT model significantly outperforms HTM in both univariate and multivariate anomaly detection tasks. The performance gap is especially pronounced in univariate settings, where

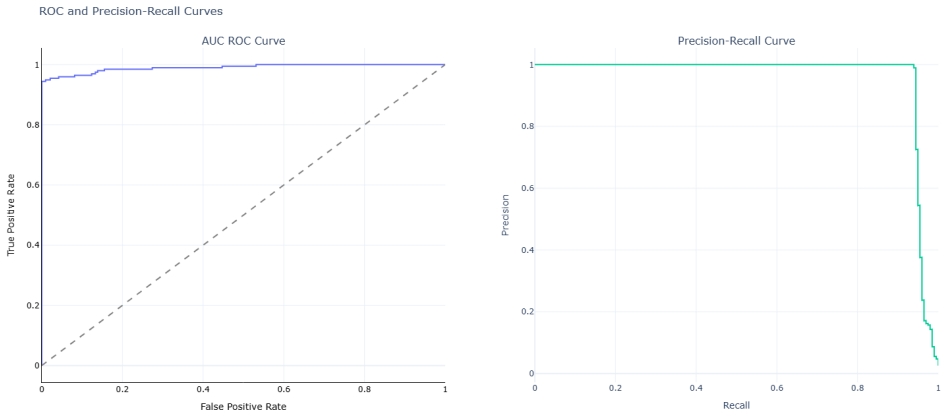

Figure 5.4: ROC and Precision-Recall Curves for DISTORTEDTkeepThirdMARS for SLADiT

SLADiT achieves high F1 scores, AUC values, and accuracy, while HTM's performance is markedly lower. In multivariate scenarios, although HTM's performance improves, SLADiT still delivers more robust and consistent anomaly detection. These findings underscore the advantage of the reconstruction-based, latent feature learning approach employed by SLADiT over the spatial correlation-dependent mechanism of HTM.

5.1.2 Comparison with Related Methods

To further contextualize the performance of our models on the NYC Taxi dataset from the NAB collection, Table 5.3 compares the results of our SLADiT and HTM models with several related methods, including SARIMA Only, LSTM Only, LSTM with STL, and STL. As shown in the table, the SLADiT model achieves an F1-Score of 0.755, with precision and recall values of 0.633 and 0.935, respectively. In contrast, the HTM model achieves an F1-Score of 0.200, indicating considerably lower performance.

These results highlight the robustness of the SLADiT model in accurately identifying anomalies in real-world urban mobility data. In particular, the high recall and F1-Score indicate that SLADiT effectively minimizes false negatives while maintaining a low false positive rate. In contrast, the poor performance of the HTM model suggests that its reliance on spatial correlations, especially when using only a scalar numeric encoder in univariate contexts, limits its efficacy for this dataset.

Overall, this comparison demonstrates that our SLADiT model outperforms traditional methods and the HTM approach on the NYC Taxi dataset, reinforcing

Models	Precision	Recall	F1-Score
SLADiT	**0.633**	**0.935**	**0.755**
HTM	0.200	0.200	0.200
SARIMA Only	0.000	0.000	0.000
LSTM Only	0.176	0.333	0.231
LSTM with STL	0.161	1.000	0.277
STL	0.533	0.889	0.667

Table 5.3: Comparison of SLADiT and HTM with related methods on NYC_Taxi (NAB) [52]

its suitability for time-series anomaly detection in complex, real-world environments.

5.2 Time-Series Visualization with Detected Anomalies

5.2.1 Comparison of Anomaly Detection Overlays

Explanation of the Subplots in UCR datasets: (a) Full Time Series (Train and Test Set) This subplot displays the complete time series. The train portion (red on the left) consists of normal data, while the test portion (green on the right) contains an injected anomaly. A blue circle pinpoints where the anomaly was introduced. This high-level view helps contextualize how much of the signal is used for training and where the model is expected to detect anomalies.

(b) Zoomed-In View of the Anomaly In this subplot, we illustrate a magnified section of the test portion where the anomaly is located. The blue line represents the original data before anomaly injection, while the green line shows the modified data after the synthetic anomaly was added. This direct Comparison depicts exactly how the time series was altered.

(c) SLADiT Detection in Univariate Mode Here, we show the results from the univariate SLADiT model. Typically, a shaded region (yellow in the example) shows the tart and end of anomaly point and red colored markers indicate the time steps that SLADiT has flagged as anomalous. This helps we visually confirm whether the model captures the injected anomaly in a single-variable context.

(d) SLADiT Detection in Multivariate Mode This subplot displays SLADiT's anomaly detection result when additional features or dimensions are included. Because SLADiT can learn correlations across multiple features, you might see more accurate or earlier detection compared to the univariate approach. Again, the anomalous region is usually highlighted or marked in a distinct color.

(e) HTM Detection in Multivariate Mode The final subplot shows how HTM detects anomalies when provided with multivariate input. Similar to (c) and (d), an overlay or shaded region can highlight the detected anomaly. If HTM struggles with univariate data (as indicated by your results), this subplot can demonstrate whether the multivariate setting helps HTM capture spatial correlations more effectively.

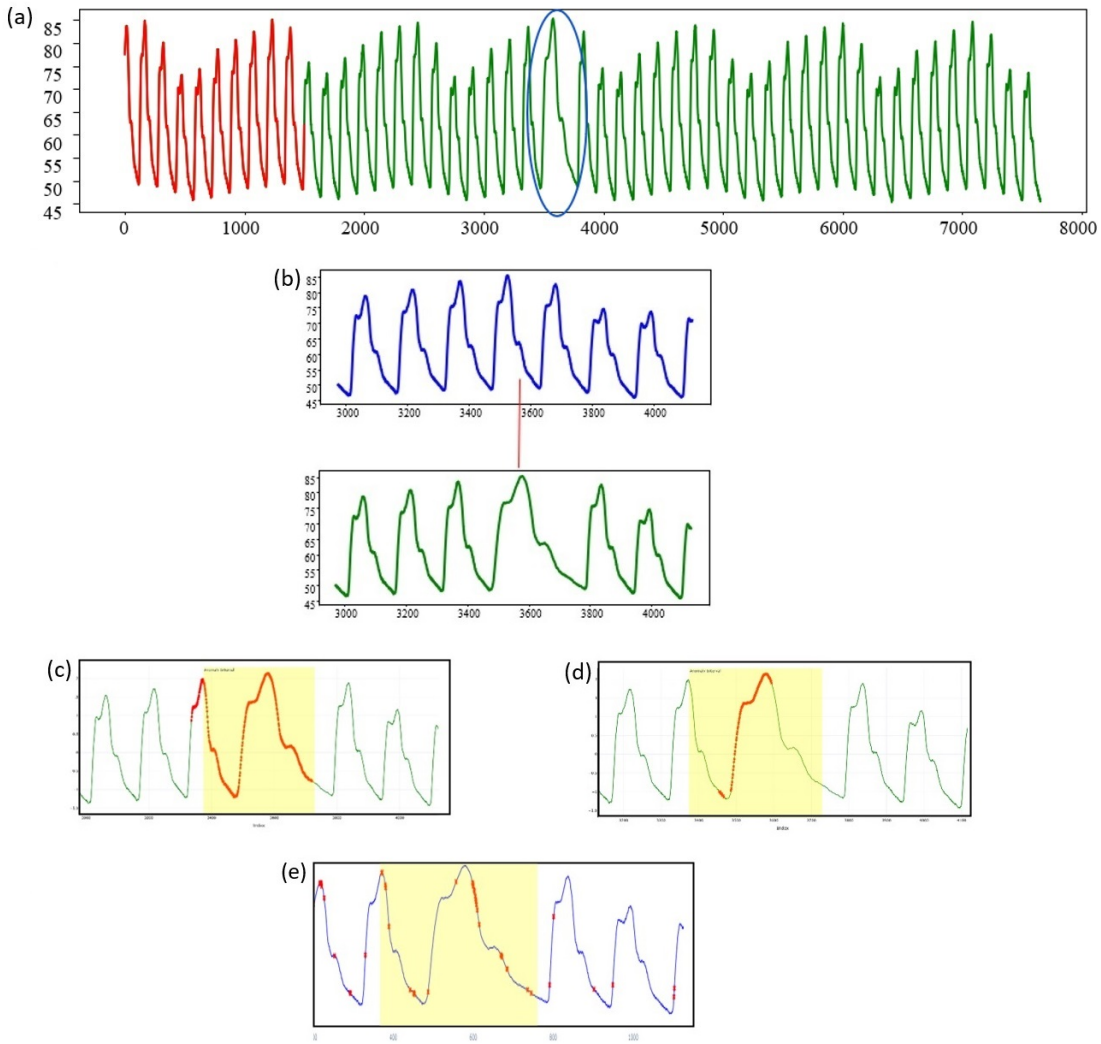

Figure 5.5: Time-Series Anomaly Detection on the InternalBleeding6 Dataset. (a) The entire time series, showing the training portion (left) and the test portion (right). The artificially introduced anomaly region is highlighted with a blue circle. (b) A zoomed-in view of the anomaly segment. The green signal represents the time series after anomaly injection, while the blue signal shows the original data for Comparison. (c) Univariate SLADiT detection result, with the anomalous region highlighted in yellow. (d) Multivariate SLADiT detection result, illustrating how additional features improve anomaly localization. (e) HTM detection result in multivariate mode, where the orange markers indicate points flagged as anomalies.

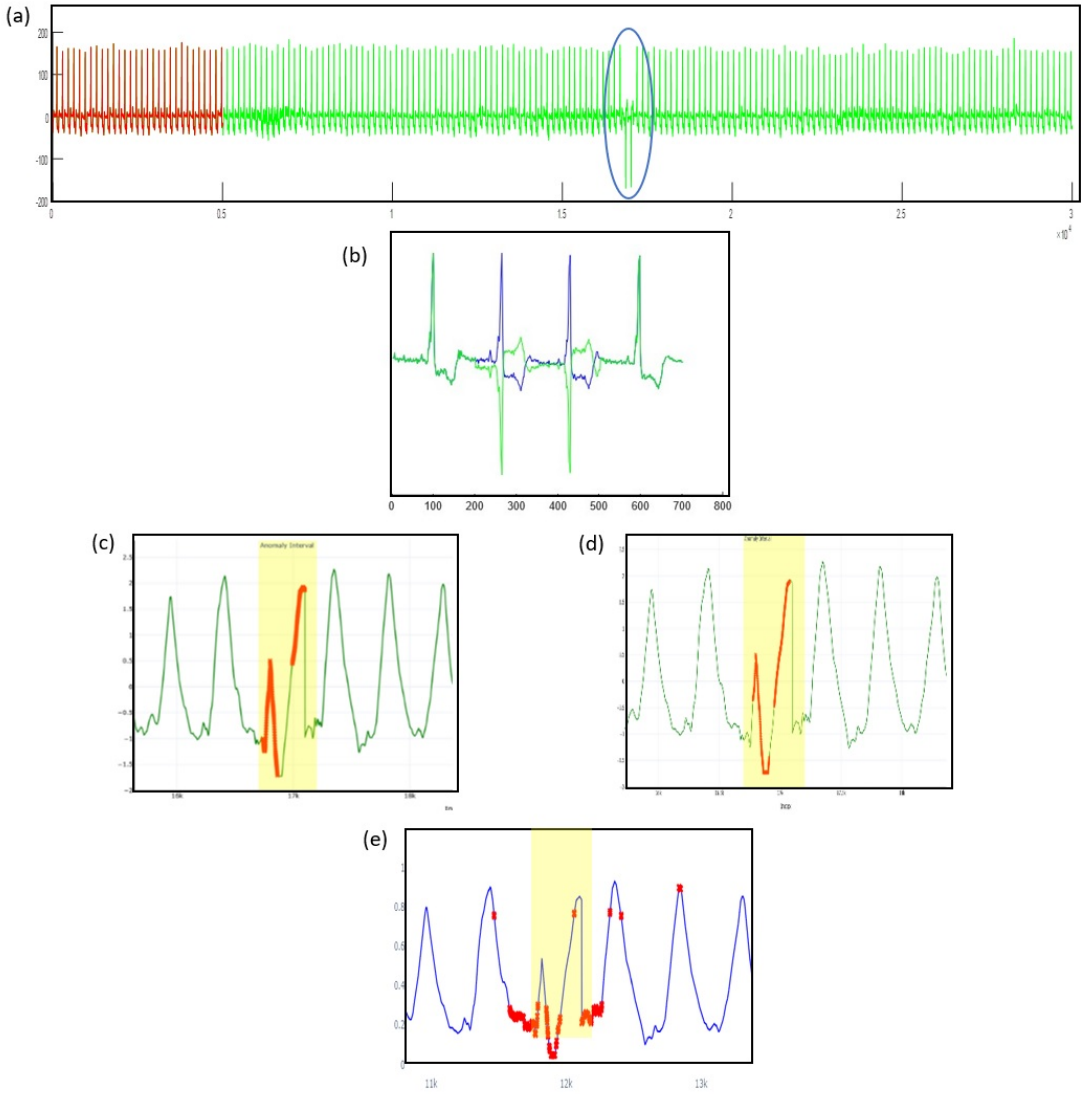

Figure 5.6: Time-Series Anomaly Detection on the ECG4 Dataset. (a) The entire time series, showing the training portion (left) and the test portion (right). The artificially introduced anomaly region is highlighted with a blue circle. (b) A zoomed-in view of the anomaly segment. The green signal represents the time series after anomaly injection, while the blue signal shows the original data for Comparison. (c) Univariate SLADiT detection result, with the anomalous region highlighted in yellow. (d) Multivariate SLADiT detection result, illustrating how additional features improve anomaly localization. (e) HTM detection result in multivariate mode, where the orange markers indicate points flagged as anomalies.

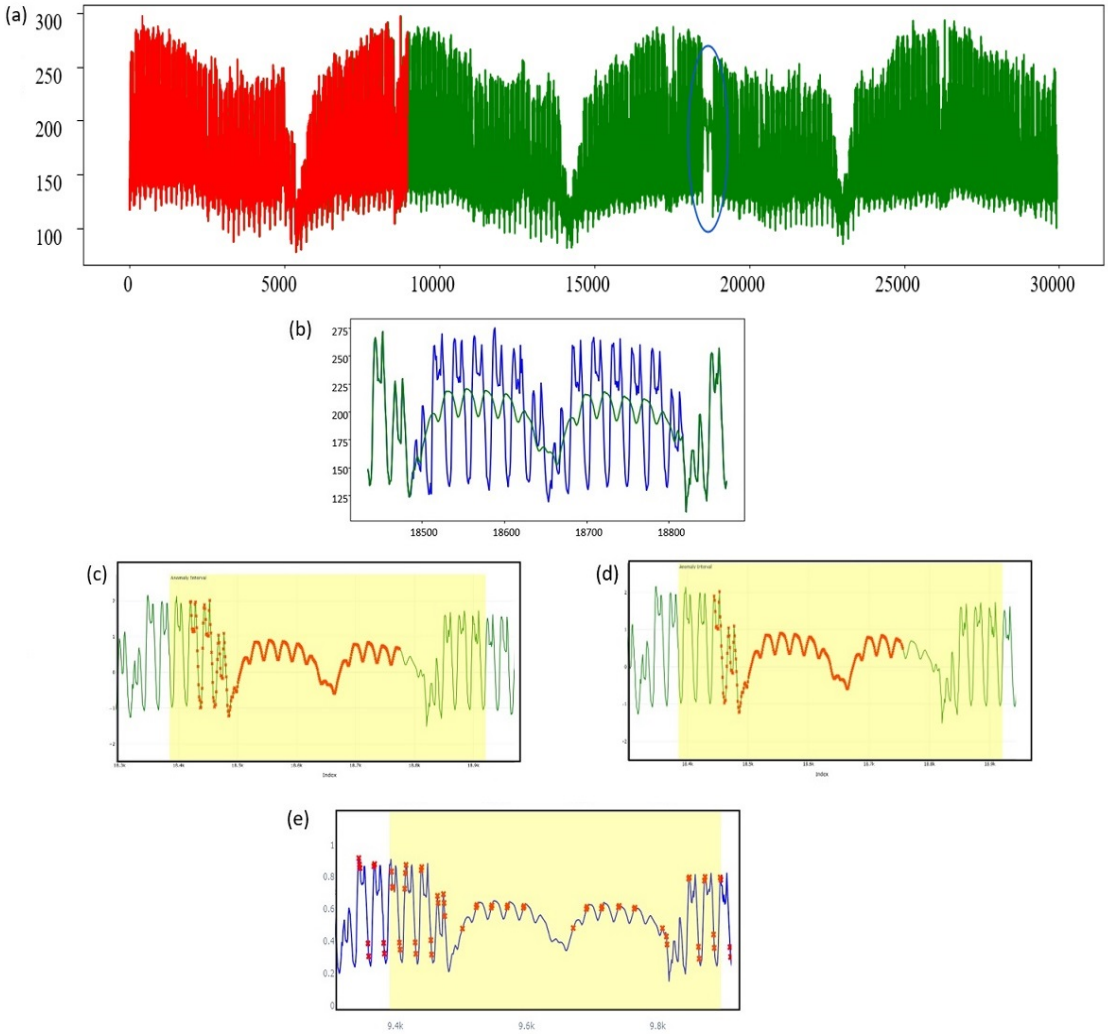

Figure 5.7: Time-Series Anomaly Detection on the PowerDemand1 Dataset. (a) The entire time series, showing the training portion (left) and the test portion (right). The artificially introduced anomaly region is highlighted with a blue circle. (b) A zoomed-in view of the anomaly segment. The green signal represents the time series after anomaly injection, while the blue signal shows the original data for Comparison. (c) Univariate SLADiT detection result, with the anomalous region highlighted in yellow. (d) Multivariate SLADiT detection result, illustrating how additional features improve anomaly localization. (e) HTM detection result in multivariate mode, where the orange markers indicate points flagged as anomalies.

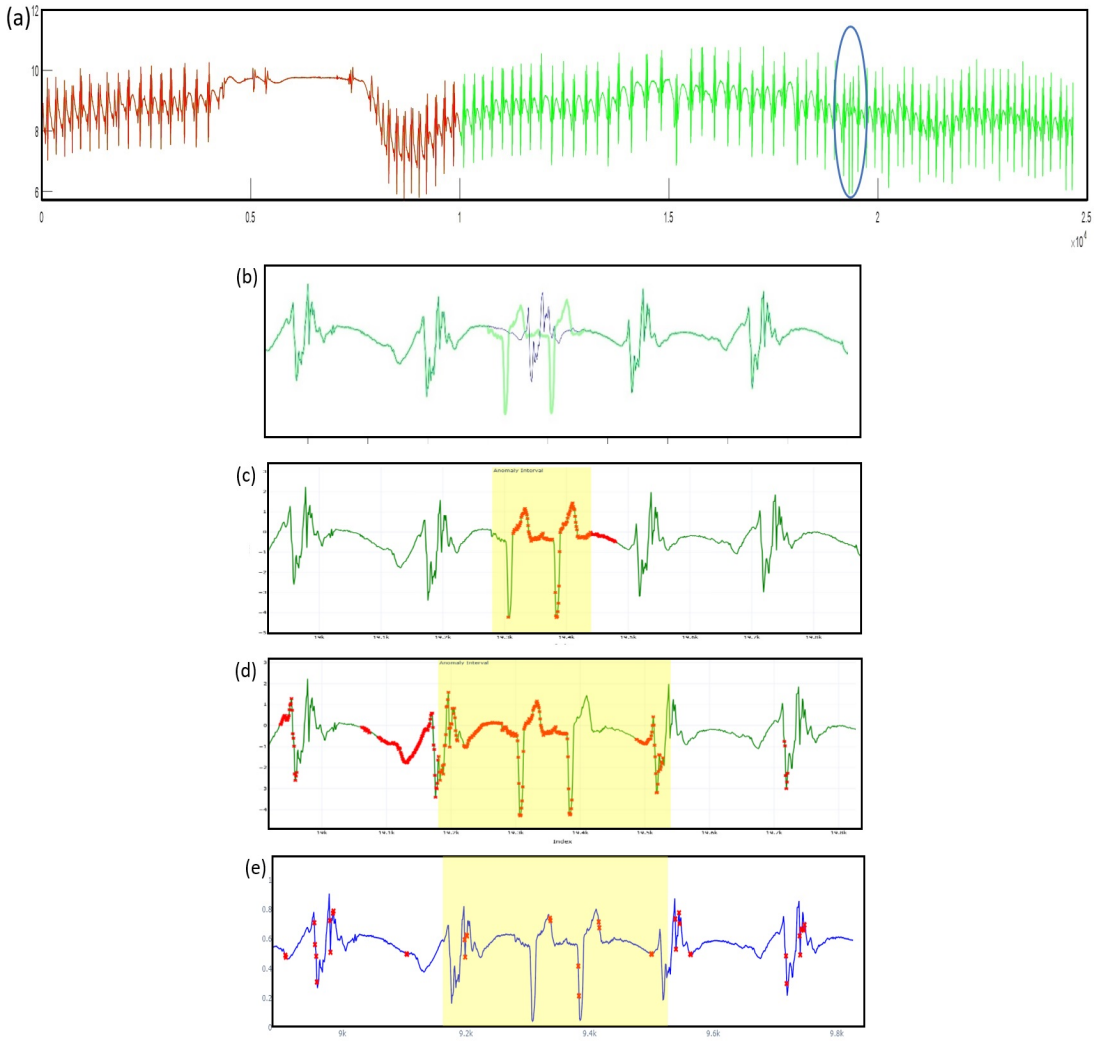

Figure 5.8: Time-Series Anomaly Detection on the MesoplodonDensirostris (Whale) Dataset. (a) The entire time series, showing the training portion (left) and the test portion (right). The artificially introduced anomaly region is highlighted with a blue circle. (b) A zoomed-in view of the anomaly segment. The green signal represents the time series after anomaly injection, while the blue signal shows the original data for Comparison. (c) Univariate SLADiT detection result, with the anomalous region highlighted in yellow. (d) Multivariate SLADiT detection result, illustrating how additional features improve anomaly localization. (e) HTM detection result in multivariate mode, where the orange markers indicate points flagged as anomalies.

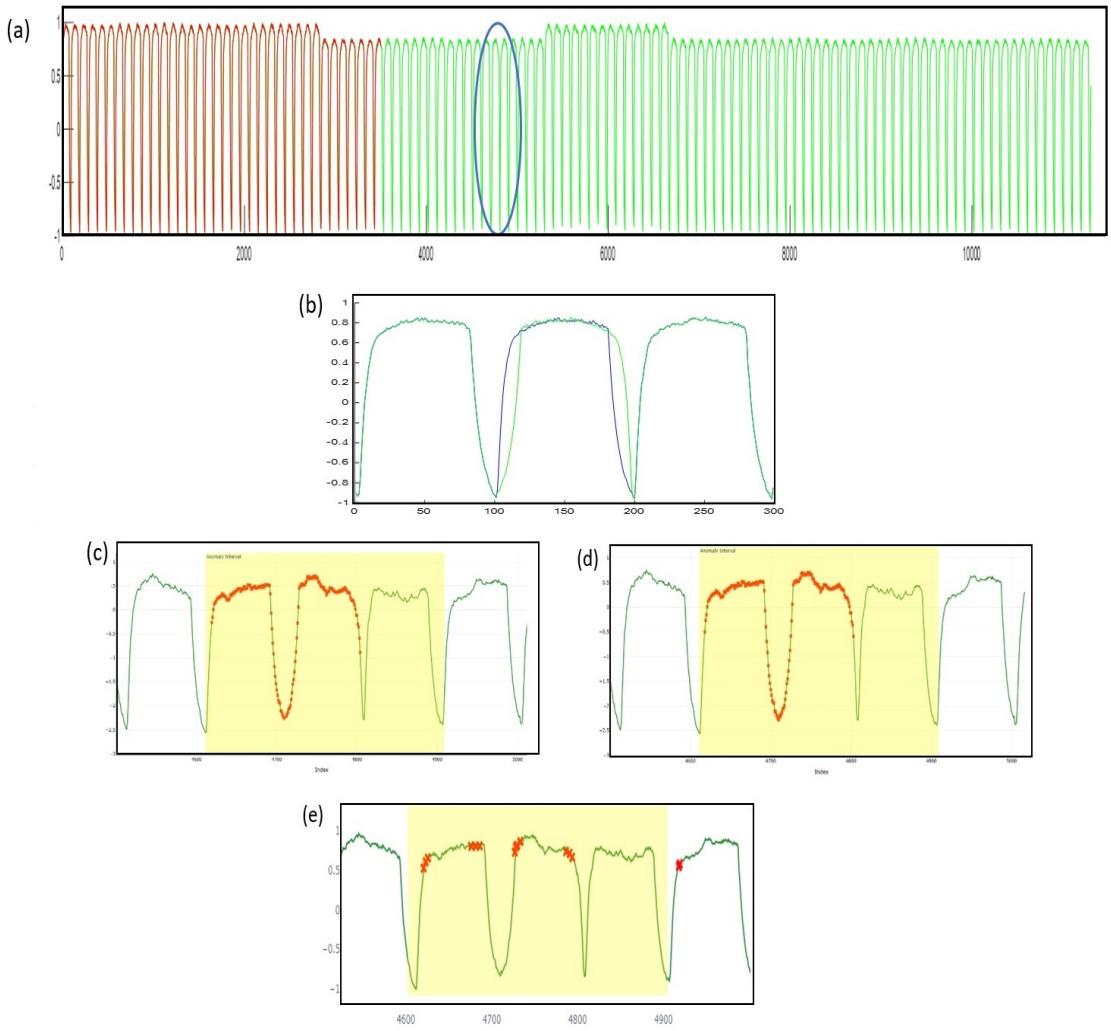

Figure 5.9: Time-Series Anomaly Detection on the DISTORTEDT-keepThirdMARS(NASA) Dataset. (a) The entire time series, showing the training portion (left) and the test portion (right). The artificially introduced anomaly region is highlighted with a blue circle. (b) A zoomed-in view of the anomaly segment. The green signal represents the time series after anomaly injection, while the blue signal shows the original data for Comparison. (c) Univariate SLADiT detection result, with the anomalous region highlighted in yellow. (d) Multivariate SLADiT detection result, illustrating how additional features improve anomaly localization. (e) HTM detection result in multivariate mode, where the orange markers indicate points flagged as anomalies.

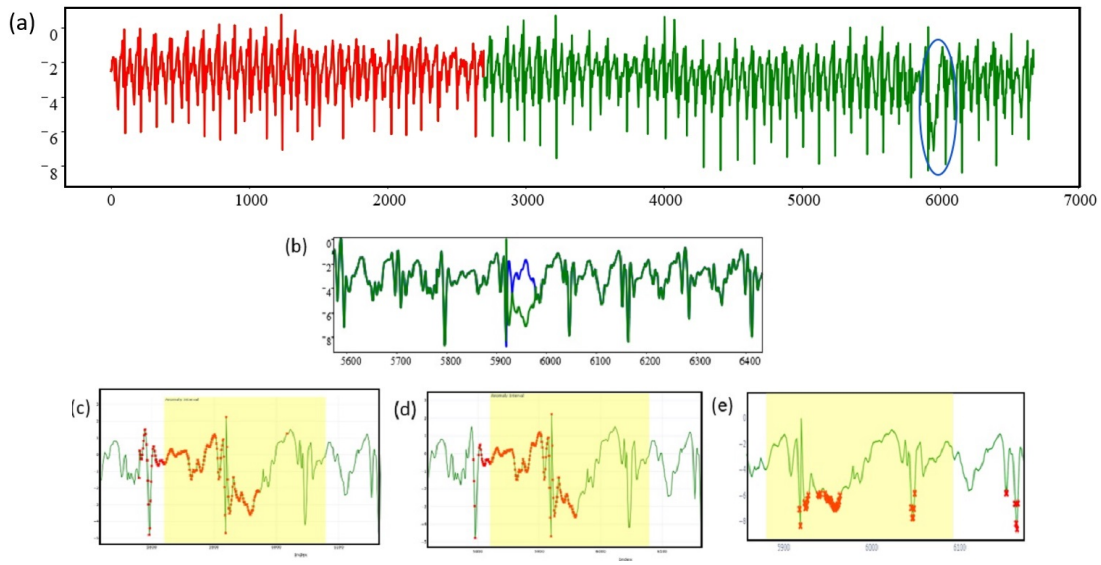

Figure 5.10: Time-Series Anomaly Detection on the DISTORTED-WalkingAceleration5 Dataset. (a) The entire time series, showing the training portion (left) and the test portion (right). The artificially introduced anomaly region is highlighted with a blue circle. (b) A zoomed-in view of the anomaly segment. The green signal represents the time series after anomaly injection, while the blue signal shows the original data for Comparison. (c) Univariate SLADiT detection result, with the anomalous region highlighted in yellow. (d) Multivariate SLADiT detection result, illustrating how additional features improve anomaly localization. (e) HTM detection result in multivariate mode, where the orange markers indicate points flagged as anomalies.

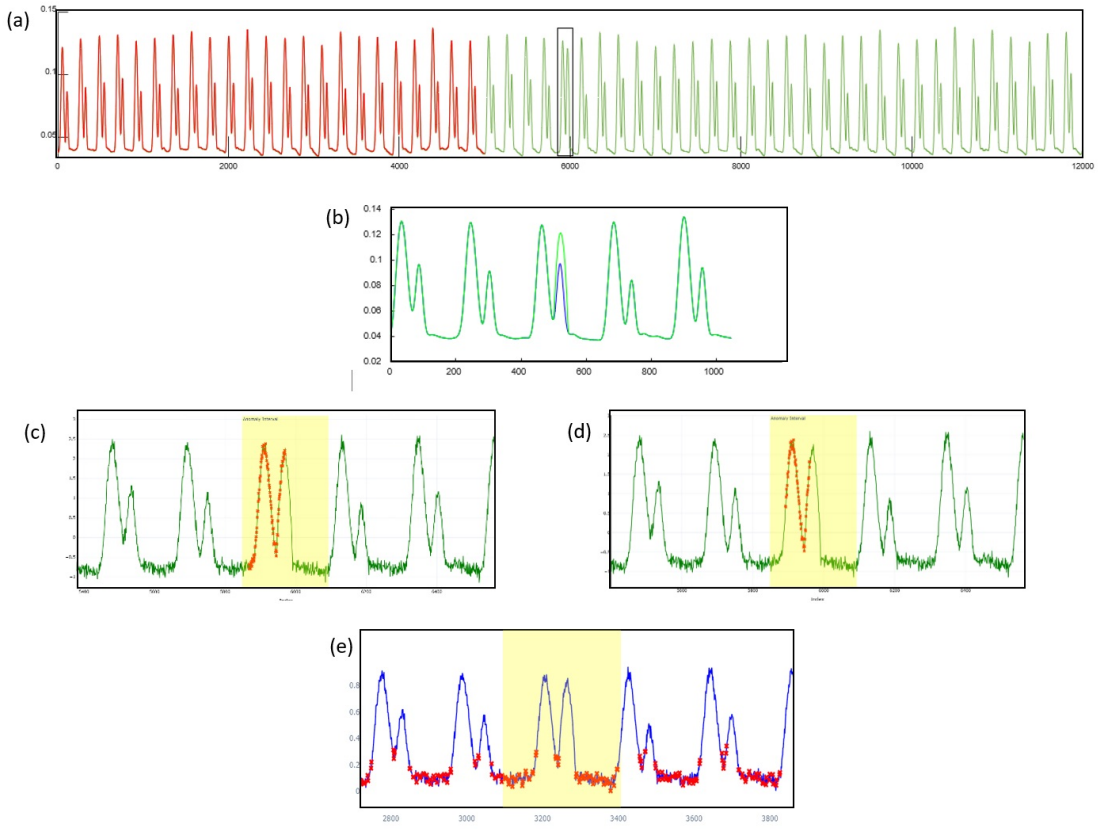

Figure 5.11: Time-Series Anomaly Detection on the NOISEGP711MarkerLFM5z3 Dataset. (a) The entire time series, showing the training portion (left) and the test portion (right). The artificially introduced anomaly region is highlighted with a blue circle. (b) A zoomed-in view of the anomaly segment. The green signal represents the time series after anomaly injection, while the blue signal shows the original data for Comparison. (c) Univariate SLADiT detection result, with the anomalous region highlighted in yellow. (d) Multivariate SLADiT detection result, illustrating how additional features improve anomaly localization. (e) HTM detection result in multivariate mode, where the orange markers indicate points flagged as anomalies.

NYC Taxi Anomaly Detection

Figure 5.12 shows the real-time graph of the NYC Taxi time series, where five significant time points—previously described in Section 4.3.1—are highlighted as anomaly intervals. These intervals correspond to notable events such as the NYC Marathon (02.11.2014), Thanksgiving (27.11.2014), Christmas (24–26.12.2014), New Year's Day (01.01.2015), and the January 2015 North American Blizzard

(26–27.01.2015). Each of these events caused abrupt increases or decreases in taxi demand, creating distinct anomaly windows in the time series.

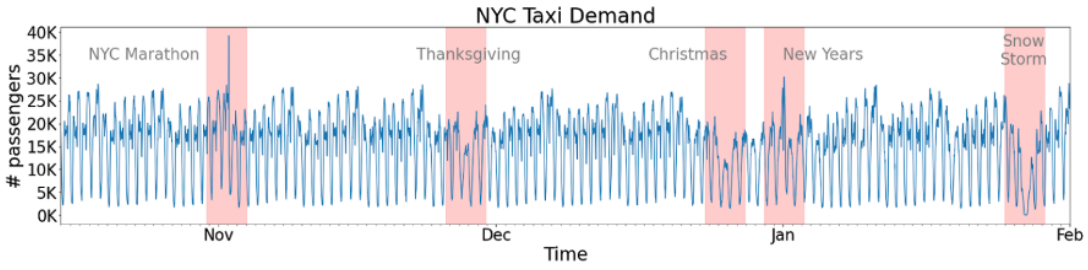

Figure 5.12: NYC Taxi demand with highlighted anomalies

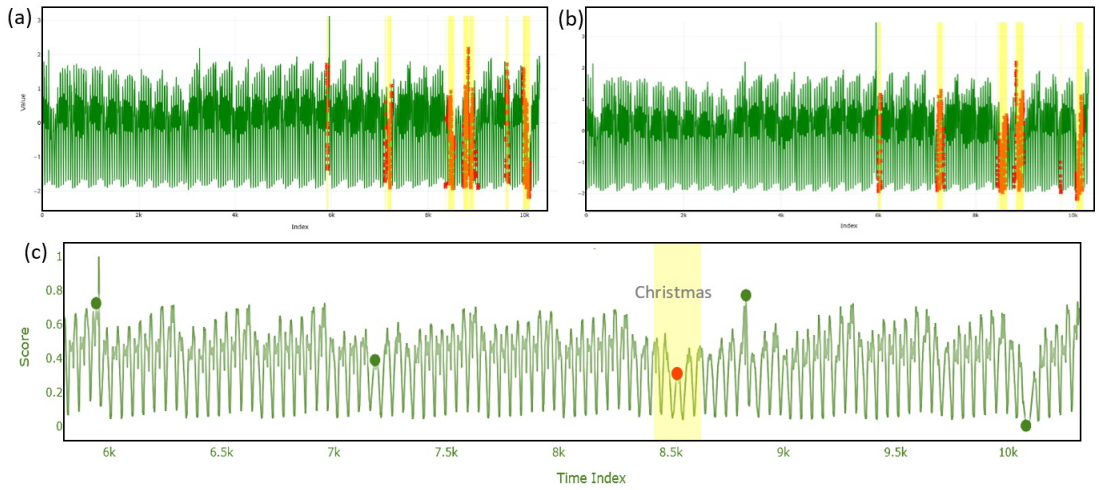

Figure 5.13: NYC Taxi demand with highlighted anomalies

Figure 5.13 provides a closer look at how SLADiT and HTM detect these anomalies:

- **(a) SLADiT in Univariate Mode:** SLADiT analyzes only the passenger count as a single feature, identifying large deviations in taxi demand around the known event dates. Although the univariate model flags most of the anomalies accurately, it occasionally produces minor false alarms.
- **(b) SLADiT in Multivariate Mode:** By incorporating additional features (cyclic transformations), SLADiT further refines anomaly detection, reducing false positives and capturing more subtle deviations. As shown, the multivariate approach offers improved localization of anomalies around the holiday

peaks and the blizzard-related drop in taxi demand.

- **(c) HTM Detection:** HTM operates in multivariate mode, but due to its high sensitivity to hyperparameters and reliance on spatial/temporal correlations, it only correctly identifies one of the five major anomalies. The model occasionally flags smaller fluctuations as anomalies while missing certain holiday-related spikes or dips.

The univariate SLADiT model performs reasonably well, but the multivariate version further reduces false positives by leveraging additional cyclic or contextual features. Meanwhile, HTM, despite its continuous-learning advantage, struggles to capture these specific event-driven anomalies consistently. The holiday peaks (Christmas and New Year's) and the sudden drop during the blizzard represent short-lived but high-impact deviations, making them challenging for HTM's anomaly likelihood mechanism to isolate without extensive hyperparameter tuning.

Overall, these findings align with the broader trend observed across the UCR datasets, where SLADiT consistently demonstrates higher F1 scores and better AUC values than HTM in both univariate and multivariate modes.

5.3 SLADiT Model Visualization and Anomaly Characterization: A Representative Example from the MARS Dataset

In this section, we present a detailed visualization of the SLADiT model's anomaly detection process using the DISTORTEDTkeepThirdMARS dataset as a representative example. The selected diagrams illustrate various aspects of the model's performance and provide insights into its detection capabilities. Although similar visualization patterns were observed across other datasets, this example serves as a comprehensive case study.

5.3.1 Reconstruction Error Analysis

Reconstruction Error Distribution

Figure 5.14 presents the distribution of reconstruction errors for SLADiT. The red dashed line represents the anomaly detection threshold. As observed, SLADiT produces a clear separation between the reconstruction errors of normal and anomalous samples, leading to fewer false positives and false negatives. In contrast, HTM's error distribution is broader, which contributes to higher false positive rates and lower recall.

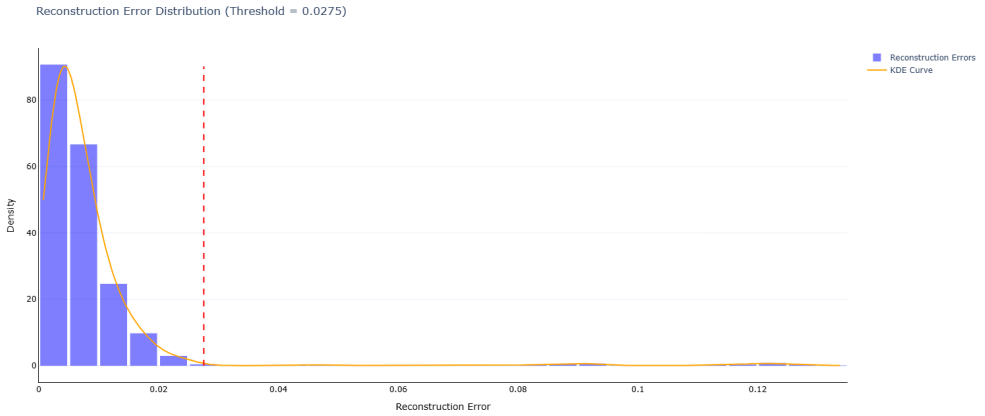

Figure 5.14: Reconstruction Error Distribution for SLADiT. The red dashed line indicates the anomaly detection threshold.

Figure 5.15 compares the reconstruction error distribution for normal vs. anomalous data. The anomalous points exhibit significantly higher errors, confirming that the model effectively separates the two groups.

Figure 5.15: Reconstruction Error Distribution for Normal vs. Anomalous Data.

5.3.2 Latent Space Visualization

t-SNE Projection of Latent Representations

To evaluate how effectively the SLADiT model encodes normal and anomalous data, we performed a 2D t-SNE projection of the latent vectors. As shown in Figure 5.16, normal samples form tight clusters, while anomalous points are distinctly separated.

This clear separation in the latent space confirms the model's robust feature extraction capabilities.

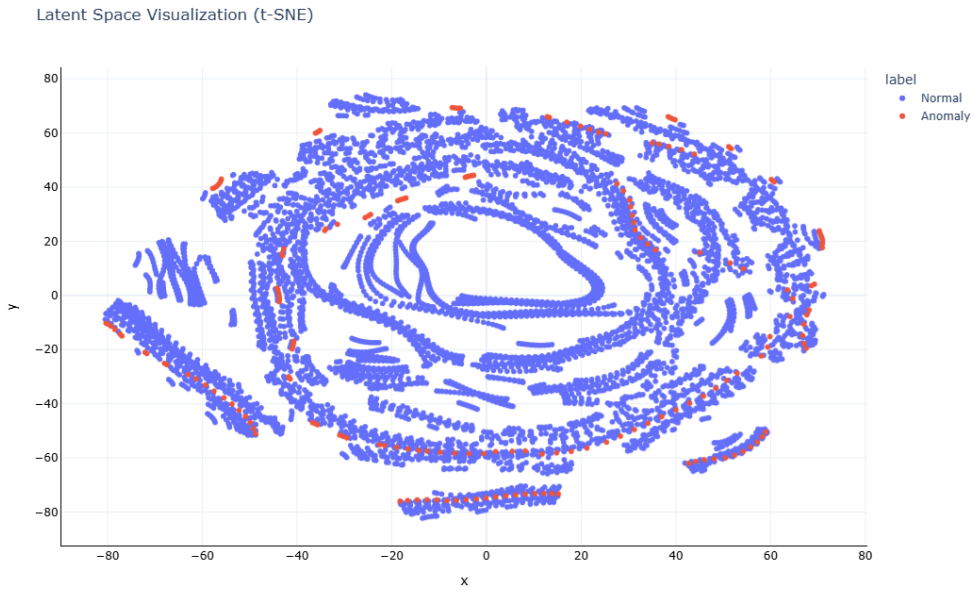

Figure 5.16: Latent Space Visualization (t-SNE) for SLADiT. Distinct clusters of normal data and anomalies are observed.

5.3.3 Anomaly Characterization

Beyond detection accuracy, it is crucial to characterize anomalies in terms of their duration, magnitude, and separability in the latent space. This analysis provides deeper insights into the severity and temporal extent of detected anomalies, informing practical decision-making.

Anomaly Duration: We analyze the duration of each detected anomaly by calculating the number of consecutive time steps during which the reconstruction error exceeds the threshold. As shown in Figure 5.17, longer durations indicating prolonged deviations from normal behavior and shorter durations corresponding to isolated spikes.

Anomaly Magnitude: The magnitude of an anomaly is quantified by measuring the excess reconstruction error over the threshold. Figure 5.18 illustrate the distribution of anomaly magnitudes across different segments. Higher magnitudes suggest more significant deviations from normal patterns, which can have critical implications in operational contexts.

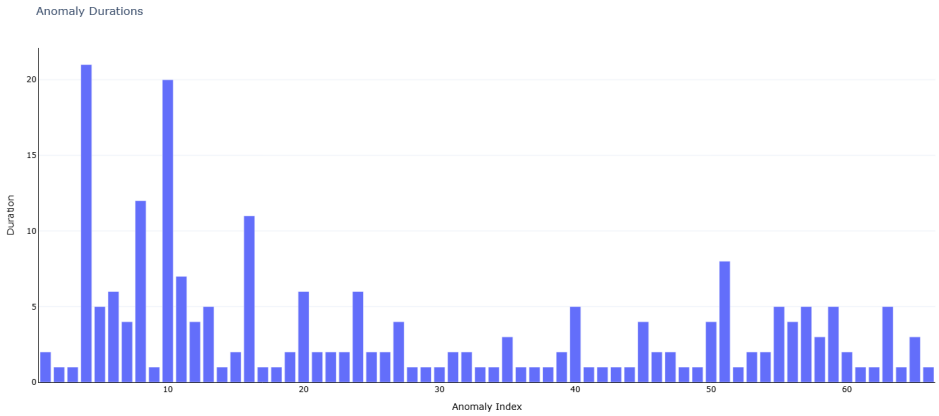

Figure 5.17: Duration of Each Detected Anomaly

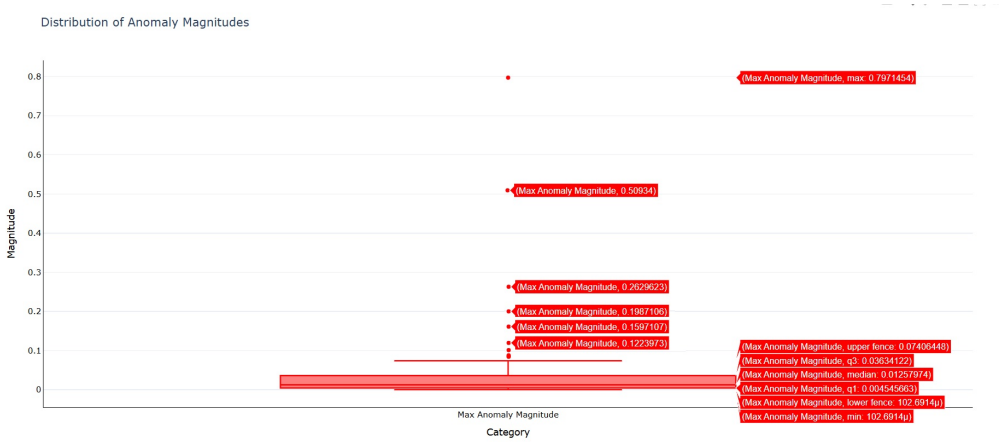

Figure 5.18: Distribution of Anomaly Magnitudes

Figure 5.18 shows the distribution of anomaly magnitudes across all detected anomalies. Higher values indicate more severe deviations from normal behavior.

Latent Space Separation: The effectiveness of the SLADiT model is further validated by its latent space separation. As shown in Figure 5.16, the t-SNE projection, a clear clustering of normal data alongside distinct regions for anomalies confirms that the model has learned a structured representation, facilitating robust anomaly detection.

Together, these characterization methods provide a comprehensive overview of the detected anomalies, complementing the quantitative performance metrics with qualitative insights.

5.3.4 Computational Performance Analysis

In addition to evaluating anomaly detection accuracy, we also assess the computational performance of both the SLADiT and HTM models. For each dataset, we measured total training time and inference speed, with results summarized in table 4.5 (refer to Chapter 4 for additional details on these metrics).

A line graph (Figure 5.19) visualizes the total run times (training plus testing times). Our experiments indicate that while SLADiT generally requires longer training and inference times—owing to its deep learning architecture and complex latent space computations—it also exhibits higher detection accuracy. In contrast, HTM, designed for continuous learning, has significantly lower computational requirements, making it attractive for real-time applications despite its lower anomaly detection performance in our experiments.

The memory usage for SLADiT ranged between approximately 280MB and 570MB on an NVIDIA GeForce MX350 GPU, depending on the dataset and the complexity of the input. This trade-off between computational cost and performance is crucial for determining the suitability of each model in different deployment scenarios.

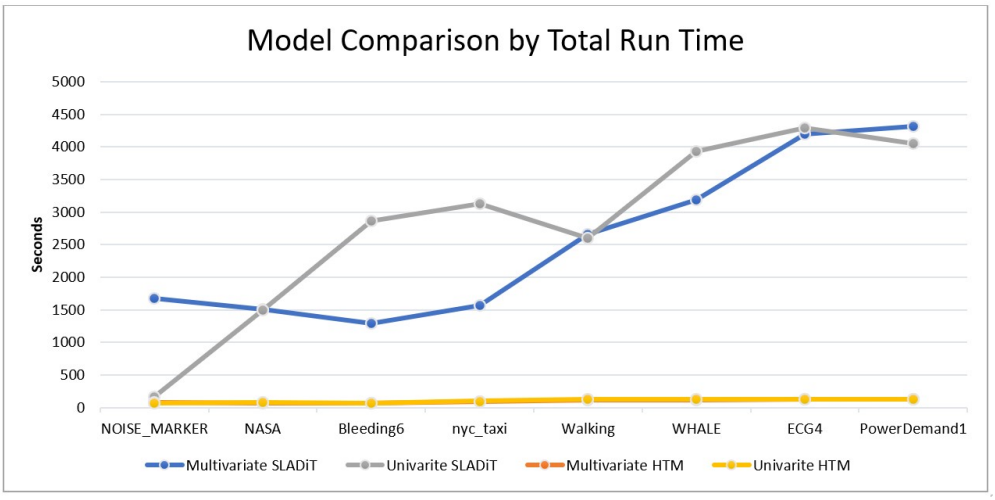

Figure 5.19: Line graph illustrating total run time for SLADiT and HTM model across various datasets.

Together, these computational performance analyses provide a comprehensive view of the resource demands of both models and help inform decisions regarding their practical deployment in real-world anomaly detection scenarios.

Conclusion

The experiments conducted in this study demonstrate that the SLADiT model consistently outperforms Hierarchical Temporal Memory (HTM) in both univariate and multivariate time-series anomaly detection tasks. In univariate scenarios, SLADiT achieved F1 scores as high as 0.97, whereas HTM's performance remained below 0.12. While HTM showed some improvement in multivariate settings (F1 scores up to 0.32), SLADiT maintained significantly higher performance across all evaluation metrics, including AUC and accuracy.

Even after incorporating cyclic features into HTM through FFT-based feature engineering, its performance gains were modest. This is largely attributed to its reliance on spatial correlations and sensitivity to a wide range of hyperparameters, which complicates tuning across diverse datasets. In contrast, SLADiT's reconstruction-based approach and self-learned latent representations provide robust generalization capabilities and stronger anomaly detection performance.

In terms of computational efficiency, SLADiT requires significantly more resources, including longer training times and higher memory usage. HTM, on the other hand, offers a lightweight and continuously adaptive architecture, making it more suitable for real-time and resource-constrained environments.

These findings suggest that while SLADiT is well-suited for high-accuracy, offline anomaly detection tasks, HTM may still be valuable in streaming scenarios. Future research could explore hybrid architectures that integrate the real-time adaptability of HTM with the representational power of SLADiT to enhance detection robustness and scalability.

Chapter 6

Discussion and Comparative Analysis

6.1 Summary and Conclusions

Both HTM and SLADiT exhibited strengths in anomaly detection, but their performance varied based on dataset characteristics and computational constraints. HTM's continuous learning approach allowed it to adapt dynamically, while SLADiT provided superior reconstruction-based detection accuracy. This section further examines these trade-offs.

The contrasting architectural paradigms of HTM and SLADiT reveal differing strengths—HTM's real-time adaptability aligns well with resource-constrained, streaming environments, while SLADiT's reconstruction-based approach excels in precision-critical, offline detection scenarios.

- SLADiT effectively captured temporal dependencies in structured time-series data but required significant computational resources.
- HTM's continuous learning mechanism provided real-time adaptability but struggled with univariate anomaly detection, as its performance heavily depended on hyperparameter tuning.
- Interpretability differed—HTM's anomaly scores were explainable due to its biologically inspired structure, whereas SLADiT relied on black-box latent space representations.

These findings emphasize the necessity of selecting models based on the specific application—real-time systems may benefit from HTM, whereas SLADiT excels in batch processing tasks requiring high accuracy.

6.2 HTM vs. SLADiT: Performance Comparison

The following table provides a structured comparison of HTM and SLADiT based on critical performance factors:

Criterion	HTM	SLADiT
Learning Type	Online, continuous learning	Offline, batch training required
Detection Accuracy	Low; highly dependent on hyperparameters	High; captures complex temporal patterns
Interpretability	High; biologically inspired, explainable	Moderate; black-box latent representations
Computational Cost	Low for real-time inference	High due to deep learning layers
Robustness to Noise	Strong due to sparse representations	Susceptible to overfitting and noise

Table 6.1: Comparative Analysis of HTM and SLADiT in Anomaly Detection.

- HTM is highly suitable for real-time anomaly detection applications due to its continuous learning mechanism.
- SLADiT outperforms HTM in structured batch-learning tasks but requires significant computational resources.
- HTM is more interpretable, making it suitable for regulated industries, whereas SLADiT offers better anomaly detection accuracy at the cost of interpretability.

These insights suggest that the choice of model should be guided by the application's requirements—whether it prioritizes real-time adaptability or detection accuracy.

6.3 Model Interpretability

Interpretability is a crucial aspect of anomaly detection models, particularly in high-stakes domains like finance, healthcare, and cybersecurity. This section evaluates how HTM and SLADiT explain their anomaly detection results.

Aspect	HTM	SLADiT
Explainability	High; predictions based on SDR patterns	Moderate; relies on learned latent features
Feature Transparency	Each neuron activation is interpretable	Black-box latent representation
Decision Rationale	Anomalies detected via sequence violations	Anomalies detected via reconstruction errors

Table 6.2: Comparison of Model Interpretability.

- HTM's interpretability is a key advantage, as it allows users to trace back why an anomaly was detected.
- SLADiT, while more accurate, lacks transparency, making it difficult to explain why certain data points are flagged as anomalies.
- While HTM is favorable for real-time, resource-constrained environments, SLADiT is more suitable when accuracy is the primary concern.

6.4 Scalability and Computational Complexity

Efficient anomaly detection requires balancing detection accuracy, inference speed, and memory usage. The following is a comparison of the scalability of HTM and SLADiT:

Factor	HTM	SLADiT
Computational Demand	Low inference cost (continuous learning)	High, due to deep LSTM layers
Real-Time Capability	High real-time performance	Moderate; batch-based inference
Memory Usage	Lower, due to sparse representations	Higher, requires full network parameters
Training Overhead	Minimal (continuous updates)	Significant periodic retraining required

Table 6.3: Comparison of Scalability and Computational Complexity between HTM and SLADiT.

HTM is more scalable for edge computing and real-time applications, whereas SLADiT, despite its higher accuracy, requires substantial computational resources and is better suited for batch processing environments.

6.4.1 Challenges in HTM Prediction and Parameter Optimization

The performance of HTM models in both prediction and anomaly detection has not been widely validated on real-world datasets. In fact, as noted in previous research (Price, 2011; Perea et al., 2009), there is a scarcity of empirical studies evaluating HTM's performance, with reported classification accuracies ranging widely—from around 90% down to 55%—depending on the use case [53, 54].

One of the central challenges is assessing how well an HTM model can predict future values based on recognized patterns. Prediction is a fundamental capability for any intelligent system—whether it is a robot catching a ball or an automated trader analyzing stock markets. The quality of these predictions depends heavily on several factors, including the volume and quality of training data, the domain-specific structure of the input data, and the internal configuration of the HTM model. Specifically, the hierarchical organization of HTM and the precise settings of its numerous parameters (such as learning rate, synaptic thresholds, and SDR encoder bit densities) play a crucial role in determining prediction accuracy. Despite the importance of these factors, there is little publicly available data that quantifies HTM's prediction quality, making even a partial assessment valuable.

Comparing HTM's performance across studies is challenging due to differences in evaluation methods. For instance, one study reported that when a Markov model's top-3 predictions were considered a hit, the accuracy reached around 50% [53]; by contrast, using a similar evaluation method, HTM has been shown to achieve an accuracy of 77.8% in some settings. However, when applied to anomaly detection—where the model's performance is gauged by the spread between reconstruction errors for normal and abnormal data—the scores are typically much lower, often in the range of 0.2 to 0.3. This does not necessarily mean that HTM is ineffective; rather, it highlights the sensitivity of HTM's anomaly detection performance to its parameter settings.

Optimizing HTM parameters is itself a significant challenge. Unlike conventional deep learning models, HTM's architecture involves a large number of interconnected parameters, and a change in one parameter often affects the performance across many others. Numenta's approach to this problem involves a heuristic "swarming" algorithm that runs multiple instances in parallel to identify an optimal configuration. However, this process is computationally intensive and time-consuming, as the search space grows exponentially with every additional parameter. As observed in the master thesis by Galetzka (2014)[55], achieving optimal parameter settings for HTM models can require extensive experimentation and remains a complex, open research problem.

Furthermore, our experience with the NuPIC framework—a popular implementation of HTM—revealed that the available documentation is sparse, and many

parameters lack clear guidance on their value ranges. This often forces researchers into a trial-and-error process, which, if done exhaustively via swarming, would require days of continuous runs for each dataset. In our study, practical constraints limited the extent of this parameter exploration.

Overall, these challenges underscore the difficulty in reliably predicting the performance of HTM models, as well as the considerable effort needed to fine-tune them for specific applications. Despite these hurdles, HTM's ability to continuously learn and adapt in real-time remains an attractive feature, even if its prediction and anomaly detection performance may not always match that of more traditional reconstruction-based approaches like SLADiT.

6.5 Limitations of the Study

Every study has its constraints; the limitations of this work include:

- **Dataset Constraints:** The UCR dataset includes only synthetic univariate anomalies, which may limit generalization to real-world scenarios.
- **HTM's Hyperparameter Sensitivity:** Extensive tuning is required for HTM, affecting the reproducibility of results.
- **SLADiT's Computational Cost:** High GPU usage during training limits the feasibility of deploying SLADiT in real-time systems.

6.6 Future Directions

Future research should explore several avenues to further enhance the performance and applicability of anomaly detection models:

- **Multivariate Anomaly Detection:** While our study extended anomaly detection to multivariate settings using FFT-based feature engineering to capture cyclic patterns, the current configuration still leaves room for improvement. Future work should investigate alternative mathematical approaches and additional configurations to better capture the inherent cyclic and interdependent patterns in high-dimensional datasets. Such efforts could lead to more robust and reliable performance, particularly for HTM, which remains sensitive to hyperparameter tuning.
- **Improved Explainability for SLADiT:** Despite its strong anomaly detection performance, SLADiT functions largely as a black-box model due to its reliance on latent space representations. Future research should focus on integrating feature attribution techniques—such as SHAP or LIME—to

enhance the interpretability of SLADiT's predictions. This improved transparency would be valuable in high-stakes applications where understanding the rationale behind an anomaly detection decision is critical.

These directions not only aim to improve detection accuracy but also enhance the practical usability of the models in diverse real-world scenarios.

Chapter 7

Conclusion

This thesis presented a comparative evaluation of two distinct anomaly detection frameworks for univariate time-series data: **Hierarchical Temporal Memory (HTM)**—a biologically inspired, continuously learning system—and **Sparse LSTM AutoEncoder for Anomaly Detection in Time-Series (SLADiT)**—a deep learning–based method relying on reconstruction errors within a sparsity-constrained latent space. Across benchmark datasets from the UCR Anomaly Archive and the Numenta Anomaly Benchmark (NAB), the experiments revealed clear differences in accuracy, adaptability, and resource demands.

Key Observations

SLADiT consistently outperformed HTM by a significant margin (often exceeding a 0.65 difference in F1-score) in univariate settings, primarily because it can encode and reconstruct complex temporal dependencies in a latent space and thus achieve higher precision while reducing false alarms. HTM, conversely, performed less effectively with single-feature data, especially when it relied on simple scalar numeric encoders that do not capture vital cyclic elements. Nevertheless, HTM's sparse distributed representations helped maintain considerable resilience to noise. In real-time adaptability, HTM excels due to its continuous learning paradigm, making it highly suitable for streaming settings such as industrial monitoring or network intrusion detection, where ongoing incremental learning can proceed without retraining. SLADiT, by contrast, requires a batch, offline training process and considerably more computational overhead, yet its enhanced anomaly detection performance makes it apt for domains like finance or healthcare, in which accuracy and lower false-positive rates are paramount.

In terms of computational and interpretability considerations, HTM's low training overhead and biologically inspired anomaly-scoring process lend it clarity, though numerous hyperparameters (including synaptic thresholds) can complicate

cross-dataset consistency. SLADiT, while offering stronger detection metrics, has more stringent GPU demands and limited transparency due to the latent representations it employs, meaning that interpretability methods (for instance, LIME or SHAP) would be essential for domain experts. When the data dimensionality increased beyond a single feature, HTM improved because multiple correlated inputs enhanced its spatial correlation capabilities, whereas SLADiT continued to benefit from reconstruction-based techniques that were already robust for univariate and multivariate data alike. Beyond the question of detection alone, the thesis also addressed the capacity of each model to characterize anomalies in terms of duration, severity, and distinctiveness in the latent space. SLADiT's reconstruction error plots and t-SNE clusters offered an in-depth portrayal of how each detected anomaly departed from normal patterns, while HTM's anomaly-likelihood measure aided in pinpointing abrupt sequence violations in scenarios calling for ongoing adjustments.

Limitations and Future Directions

A primary limitation of this work lies in the use of a small number of UCR datasets—many of which contain synthetic anomalies—and a single real-world NAB dataset, NYC Taxi. This may constrain the generalizability of conclusions about the two models' performance in other domains. Future investigations should involve additional data sources, especially genuine streaming feeds from IoT systems or continuous sensor data, to assess how these models handle large-scale shifts and more intricate real-world noise. Another challenge concerns the extensive hyperparameter tuning that HTM demands; applying the so-called swarming algorithms or adopting adaptive parameter strategies to reduce manual trial and error could improve HTM's consistency across varied datasets. Regarding SLADiT, further research can integrate interpretability tools (such as LIME or SHAP) to clarify which aspects of the time-series signal drive reconstruction errors. This enhanced transparency would be particularly beneficial for practitioners in high-stakes sectors like healthcare or finance, where comprehensibility of model decisions is often as critical as the accuracy itself. Additional exploration into hybrid solutions is also compelling: combining the continuous updating of HTM with the advanced reconstruction-based detection of SLADiT may yield synergy that addresses both immediate adaptation and thorough offline analysis in a single system.

Concluding Remarks

By systematically evaluating HTM and SLADiT on recognized benchmarks, the thesis shows how these models embody contrasting paradigms for time-series anomaly detection. SLADiT delivers consistently stronger detection results and fewer false positives, making it an excellent fit for environments that tolerate

offline batch training and require high-precision detection. HTM, in contrast, offers an appealing trade-off for real-time or resource-limited applications, featuring straightforward continuous updates and less computational overhead. Ultimately, no single model completely dominates every scenario, and real-world success in anomaly detection depends on an appropriate alignment between the model's capabilities and the application's demands. These findings form a valuable basis for further refinement of both approaches, including prospective ensemble techniques that merge the adaptiveness of cortical learning algorithms with the representational power of deep neural networks.

Bibliography

[1] M. Braei and S. Wagner. «Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art». In: *arXiv* (Apr. 2020). URL: https://doi.org/10.48550/arXiv.2004.00433 (cit. on pp. 1, 7).

[2] V. Chandola, A. Banerjee, and V. Kumar. «Anomaly detection: A survey». In: *ACM Computing Surveys* 41 (July 2009), pp. 1–58. URL: https://doi.acm.org/10.1145/1541880.1541882 (cit. on pp. 1, 8, 10).

[3] A. Toshniwal, K. Mahesh, and R. Jayashree. «Overview of Anomaly Detection techniques in Machine Learning». In: *Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)* 41 (2020), pp. 808–815. URL: https://doi.org/10.1109/I-SMAC49090.2020.9243329 (cit. on pp. 1, 10).

[4] E. Capobianco. «Denoising and dimensionality reduction of genomic data». In: *SPIE Third International Symposium on Fluctuations and Noise* 5841 (May 2005), pp. 808–815. URL: https://doi.org/10.1117/12.609299 (cit. on p. 2).

[5] V. Wojcik and P. Comte. «Algorithms for speedy visual recognition and classification of patterns formed on rectangular imaging sensors». In: *Neurocomputing* 5841 (Dec. 2010), pp. 140–154. URL: https://doi.org/10.1016/j.neucom.2009.10.029 (cit. on p. 2).

[6] Emre O. Neftci. «Data and Power Efficient Intelligence with Neuromorphic Learning Machines». In: *iScience* 5 (2018), pp. 52–68. URL: https://doi.org/10.1016/j.isci.2018.06.010 (cit. on p. 2).

[7] S. Nousias, E.-V. Pikoulis, C. Mavrokefalidis, and A. S. Lalos. «Accelerating deep neural networks for efficient scene understanding in multi-modal automotive applications». In: *IEEE Access* 2023 (11), pp. 28208–28221. URL: https://doi.org/10.1109/ACCESS.2023.3258400 (cit. on p. 6).

[8] E. Ghaderpour, H. Dadkhah, H. Dabiri, F. Bozzano, G. Scarascia Mugnozza, and P. Mazzanti. «Precipitation time series analysis and forecasting for italian regions». In: *Engineering Proceedings* 39 (2023), p. 23. URL: https://www.mdpi.com/2673-4591/39/1/23 (cit. on p. 7).

[9] B. Anomadarshi, D. Muthirayan, P.Khargonekar, and M. Faruque. «Hierarchical Temporal Memory-Based One-Pass Learning for Real-Time Anomaly Detection and Simultaneous Data Prediction in Smart Grids». In: *IEEE Transactions on Dependable and Secure Computing* 19 (May 2022), pp. 1770–1782. URL: https://doi.org/10.1109/TDSC.2020.3037054 (cit. on pp. 7, 10, 11, 14–16, 21).

[10] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutt. «Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures». In: *Frontiers in big data* 3 (2020), p. 4. URL: https://doi.org/10.3389/fdata.2020.00004 (cit. on p. 7).

[11] J. Paparrizos, Y. Kang, P. Bonioland R. Tsay, T. Palpanas, and M. J. Franklin. «TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection». In: *Proc. VLDB Endow* 15 (Apr. 2022), pp. 1697–1711. URL: https://doi.org/10.14778/3529337.3529354 (cit. on pp. 7, 8, 12, 13).

[12] M. Yucel, A. Sertbas, and B. Ustundag. «High Performance Time Series Anomaly Detection Using Brain Inspired Cortical Coding Method». In: *IEEE Access* 11 (Jan. 2023), pp. 8345–8361. URL: https://doi.org/10.1109/ACCESS.2023.3239212 (cit. on pp. 7, 8, 10, 12, 14, 15, 18, 26, 38).

[13] S. Schmidl, P. Wenig, and T. Papenbrock. «Anomaly Detection in Time Series: A Comprehensive Evaluation». In: *Proceedings of the VLDB Endowment* 15 (May 2022), pp. 1779–1797. URL: https://doi.org/10.14778/3538598.3538602 (cit. on pp. 7, 12).

[14] D. Rozado, F.B. Rodriguez, and P. Varona. «Optimizing Hierarchical Temporal Memory for Multivariable Time Series». In: *Springer* 6353 (2010), pp. 506–518. URL: http://link.springer.com/10.1007/978-3-642-15822-3_62 (cit. on pp. 7–9, 11, 16).

[15] P. Gogoi, B. Borah, and D. K. Bhattacharyya. «Anomaly Detection Analysis of Intrusion Data using Supervised & Unsupervised Approach». In: *Convergence Information Technology* 5 (Feb. 2010). URL: 10.4156/jcit.vol5.issue1.11 (cit. on pp. 7, 31).

[16] R. Chalapathy and S. Chawla. «Deep learning for anomaly detection: A survey». In: *arXiv preprint* (2019) (cit. on pp. 8, 9).

[17] K. Choi, J. Yi, C. Park, and S. Yoon. «Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines». In: *IEEE Access* 9 (2021), pp. 120043–120065. URL: https://ieeexplore.ieee.org/document/9523565/ (cit. on pp. 8, 9).

[18] D. M. Hawkins. *Identification of outliers*. first. Dordrecht: Springer Netherlands, 1980 (cit. on p. 9).

[19] R. Hyndman and G. Athanasopoulos. «Forecasting». In: *OTexts* (2018) (cit. on pp. 9, 12, 16).

[20] K. Roy, A. R. Jaiswal, and P. Panda. «Towards spike-based machine intelligence with neuromorphic computing». In: *Nature* 575 (Nov. 2019), pp. 607–617. URL: https://doi.org/10.1038/s41586-019-1677-2 (cit. on pp. 10, 14, 21, 25).

[21] S. Agrawal and J. Agrawal. «Survey on Anomaly Detection using Data Mining Techniques». In: *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)* 60 (Sept. 2015), pp. 708–713. URL: https://doi.org/10.1016/j.procs.2015.08.220 (cit. on p. 10).

[22] L. Qiong and W. Ying. «Supervised Learning». In: *Encyclopedia of the Sciences of Learning*. Boston, MA: Springer US, 2012 (cit. on p. 10).

[23] A. Mohammad Iquebal and A. Mohammad Gulam. «Detecting telecommunication fraud using neural networks through data mining». In: *International Journal of Scientific and Engineering Research* 3 (Mar. 2012), pp. 601–606. URL: https://api.semanticscholar.org/CorpusID:6738874 (cit. on pp. 11, 14).

[24] Z. Xiaojin and G. Andrew B. «Introduction to semi-supervised learning». In: Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer Nature, 2022. URL: https://link.springer.com/10.1007/978-3-031-01548-9 (cit. on pp. 11, 14).

[25] B. H. Kam T. Pourhabibi K. L. Ong and Y. L. Boo. «Fraud detection: A systematic literature review of graph-based anomaly detection approaches». In: *Decision Support Systems* 133 (2020), p. 113303. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167923620300580 (cit. on pp. 11, 12).

[26] K. Noto, C. Brodley, and D. Slonim. «FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection». In: *Data Min Knowl Discov* 25 (Sept. 2012), pp. 109–133. URL: http://link.springer.com/10.1007/s10618-011-0234-x (cit. on pp. 11, 14).

[27] R. J. Bolton and D. J. Hand. «Unsupervised profiling methods for fraud detection». In: (2002). URL: https://api.semanticscholar.org/CorpusID:14365948 (cit. on pp. 11, 14).

[28] R. J. Bolton and D. J. Hand. «Statistical fraud detection: A review». In: *Statistical Science* 17 (Aug. 2002), pp. 235–255. URL: https://projecteuclid.org/journals/statistical-science/volume-17/issue-3/Statistical-Fraud-Detection-A-Review/10.1214/ss/1042727940.full (cit. on pp. 12, 15).

[29] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland. «Data mining for credit card fraud: A comparative study». In: *Decision Support Systems* 50 (Feb. 2011), pp. 602–613. URL: https://doi.org/10.1016/j.dss.2010.08.008 (cit. on p. 12).

[30] A. Aisha, M. Mohd Aizaini, and Z. Anazida. «Fraud detection system: A survey». In: *Journal of Network and Computer Applications* 68 (June 2016), pp. 90–113. URL: https://doi.org/10.1016/j.jnca.2016.04.007 (cit. on pp. 12, 16).

[31] . A. Maharaj, P. D'Urso, and J. Caiado. *Time Series Clustering and Classification*. first. Chapman and Hall/CRC, 2019 (cit. on p. 12).

[32] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. «Unsupervised real-time anomaly detection for streaming data». In: *Neurocomputing* 262 (Nov. 2017), pp. 134–147. URL: https://doi.org/10.1016/j.neucom.2017.04.070 (cit. on p. 12).

[33] F. Ahmadvand, M. Hosseini Imani, and H. Taheri Andaniand M. Taheri Andani. «Optimizing the LSTM Model for Electricity Price Forecasting using the MSADBO Algorithm». In: *Texas Power and Energy Conference (TPEC) 2025* (Feb. 2025) (cit. on pp. 12, 19).

[34] J. Hawkins. «HIERARCHICAL TEMPORAL MEMORY including HTM Cortical Learning Algorithms». In: *Numenta* (2011). URL: https://www.numenta.com/resources/research-publications/papers/hierarchical-temporal-memory-white-paper/ (cit. on p. 16).

[35] M. Yucel, S. Bagis, A. Sertbas, M. Sarikaya, and B. Üstündağ. «Brain Inspired Cortical Coding Method for Fast Clustering and Codebook Generation». In: *Entropy* 24 (Nov. 2022), p. 1678. URL: https://doi.org/10.3390/e24111678 (cit. on pp. 18, 26).

[36] J. Hawkins and D. George. «Hierarchical Temporal Memory Concepts , Theory , and Terminology». In: *Numenta* (2006). URL: https://api.semanticscholar.org/CorpusID:17363521 (cit. on p. 19).

[37] D. Maltoni. «Pattern Recognition by Hierarchical Temporal Memory». In: *Social Science Research Network* (2011). URL: https://doi.org/10.2139/ssrn.3076121 (cit. on pp. 19, 25).

[38] F. Shoeleh, M. Erfani, D. Le, and A. A. Ghorbani. «Ensemble of Hierarchical Temporal Memory for Anomaly Detection». In: *Proc. IEEE 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)* (Oct. 2020), pp. 50–59. URL: https://doi.org/10.1109/DSAA49011.2020.00017 (cit. on pp. 19, 55).

[39] T. Charrad, K. Nouira, and A. Ferchichi. «Use of Hierarchical Temporal Memory Algorithm in Heart Attack Detection». In: *World Academy of Science, Engineering and Technology, International Journal of Mechanical and Mechatronics Engineering* 13 (2019), pp. 312–315. URL: https://doi.org/10.5281/ZENODO.3298882 (cit. on pp. 19, 55).

[40] Y. Cui, S. Ahmad, and J. Hawkins. «The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding». In: *Front. Comput. Neurosci.* 11 (2017). URL: https://doi:%2010.3389/fncom.2017.00111/ (cit. on pp. 19, 21, 23, 25).

[41] R. W. Hiorns A. J. Rockel and T. P. Powell. «The basic uniformity in structure of the neocortex». In: *Brain* 103 (June 1980), pp. 221–244. URL: https://doi.org/10.1093/brain/103.2.221 (cit. on p. 21).

[42] M. Otahal and Š. Olga. «ANOMALY DETECTION WITH CORTICAL LEARNING ALGORITHM FOR SMART HOMES». In: *Smart homes* 20144 (Nov. 2014), p. 24. URL: https://doi.org/10.13140/2.1.2927.8405 (cit. on p. 21).

[43] J. Mnatzaganian, E. Fokoue, and D. Kudithipudi. «A Mathematical Formalization of Hierarchical Temporal Memory's Spatial Pooler». In: *Frontiers Robotics AI* 3 (Jan. 2017). URL: https://doi.org/10.3389/frobt.2016.00081 (cit. on pp. 21, 22).

[44] S. Purdy. «Encoding Data for HTM Systems». In: *Neural and Evolutionary Computing* (2016), pp. 708–713. URL: https://arxiv.org/abs/1602.05925 (cit. on p. 23).

[45] J. Hawkins and S. Ahmad. «Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex». In: *Frontiers in Neural Circuits* 10 (Mar. 2016). URL: http://dx.doi.org/10.3389/fncir.2016.00023 (cit. on p. 24).

[46] M. Wissal, Z. Fki, and M. BenAyed. «Online anomaly detection in ECG signal using Hierarchical Temporal Memory». In: *n 2019 Fifth International Conference on Advances in Biomedical Engineering (ICABME)* (2019), pp. 1–4. URL: https://ieeexplore.ieee.org/document/8940307/ (cit. on pp. 27, 40, 42).

[47] R. Wu and E. J. Keogh. «Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress». In: *IEEE Transactions on Knowledge and Data Engineering* 35 (Mar. 2023), pp. 2421–2429. URL: https://doi.org/10.1109/TKDE.2021.3112126 (cit. on p. 32).

[48] H. A. Dau, A. Bagnall, K.Kamgar, C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. «The UCR Time Series Archive». In: *arXiv* (Sept. 2019). URL: https://arxiv.org/abs/1810.07758 (cit. on p. 32).

[49] Machine Learning and Data Analytics (MaD) Lab Friedrich-Alexander-Universität. *Friedrich-Alexander-Universität(FAU)*. Accessed on March 16 2025. n.d. URL: https://www.mad.tf.fau.de/research/activitynet/gaitphase-database (cit. on p. 33).

[50] Numenta. *Numenta Anomaly Benchmark (NAB) Dataset*. https://github.com/numenta/NAB/tree/master/data/realKnownCause. Accessed on March 01 2025. n.d. (Cit. on p. 34).

[51] J. W. Cooley and J. W. Tukey. «An algorithm for the machine calculation of complex Fourier series». In: *Mathematics of Computation* 19 (Apr. 1965), pp. 297–301. URL: https://api.semanticscholar.org/CorpusID:121744946 (cit. on p. 36).

[52] J. -B. Kao and J. -R. Jiang. «Anomaly Detection for Univariate Time Series with Statistics and Deep Learning». In: *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)* (Oct. 2019), pp. 404–407. URL: http://dx.doi.org/10.1109/ECICE47484.2019.8942727 (cit. on p. 65).

[53] JE. Meroño SAJ. Perea and MJ. Aguilera. «Application of Numenta® Hierarchical Temporal Memory for land-use classification». In: *South African Journal of Science* 105 (Sept. 2009), pp. 370–375. URL: https://doi.org/10.4102/sajs.v105i9/10.114 (cit. on p. 84).

[54] R. W. Price. «Hierarchical temporal memory cortical learning algorithm for pattern recognition on multi-core architectures». In: *PhD thesis. Portland State University sity* (2011) (cit. on p. 84).

[55] M. Galetzka. «Intelligent Predictions: an empirical study of the Cortical Learning Algorithm». In: *Master Thesis. Department of Computer Science University of Applied Sciences Mannheim* (Oct. 2014) (cit. on p. 84).