A POST-INFERENCE TRANSFORMER FRAMEWORK FOR ANOMALY RANGE DETECTION IN MULTIVARIATE TIME SERIES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

GÖKSU UZUNTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF DATA INFORMATICS

JUNE 2025

A Post-Inference Transformer Framework for Anomaly Range Detection in Multivariate Time Series

Date: 25.06.2025

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: **Göksu Uzuntürk**

Signature :

ABSTRACT

A POST-INFERENCE TRANSFORMER FRAMEWORK FOR ANOMALY RANGE DETECTION IN MULTIVARIATE TIME SERIES

Uzuntürk, Göksu

M.S., Department of Data Informatics

Supervisor: Prof. Dr. Tuğba Taşkaya Temizel

June 2025, 75 pages

Range-based anomaly detection in multivariate time series plays a pivotal role in domains such as healthcare, industrial monitoring, finance, and cloud systems, where temporally extended faults often provide more actionable insights than isolated point anomalies. This study presents a transformer-based framework specifically designed to address the challenges of highly imbalanced multiclass range-based anomaly detection. To enhance temporal and semantic consistency, two post-inference strategies are incorporated: majority voting over overlapping multi-step predictions and a domain-informed transition masking mechanism that enforces realistic class transitions. These strategies contribute to output stability and more reliable diagnostics in scenarios governed by known operational constraints. The proposed method is evaluated on the Exathlon benchmark, demonstrating a notable improvement with a 24% increase in F1 score of the weighted, binary anomaly detection evaluation framework.

Keywords: Anomaly interval detection, multivariate time series, multiclass classification, transformer networks, post-inference strategies

ÖZ

ÇOK DEĞİŞKENLİ ZAMAN SERİLERİNDE ANOMALİ ARALIĞI TESPİTİ İÇİN BİR ÇIKARIM SONRASI DÖNÜŞTÜRÜCÜ ÇERÇEVESİ

Uzuntürk, Göksu

Yüksek Lisans, Veri Bilişimi Bölümü

Tez Yöneticisi: Prof. Dr. Tuğba Taşkaya Temizel

Haziran 2025, 75 sayfa

Çok değişkenli zaman serilerinde aralık tabanlı anomali tespiti; sağlık, endüstriyel izleme, finans ve bulut sistemleri gibi alanlarda önemli bir rol oynamaktadır. Bu alanlarda, zamana yayılmış hatalar genellikle izole nokta anomalilerinden daha anlamlı ve eyleme geçirilebilir içgörüler sunar. Bu çalışma, yüksek derecede dengesiz ve çok sınıflı aralık tabanlı anomali tespitiyle ilişkili zorlukları ele almak üzere özel olarak tasarlanmış, dönüştürücü tabanlı bir çerçeve sunmaktadır. Zamansal ve anlamsal tutarlılığı artırmak amacıyla, çıkarım sonrası iki strateji entegre edilmiştir: çok adımlı ve çakışan tahminler üzerinde çoğunluk oyu uygulaması ve gerçekçi sınıf geçişlerini zorunlu kılan, alana özgü bir geçiş maskeleme mekanizması. Bu stratejiler, bilinen operasyonel kısıtlamalar tarafından yönetilen senaryolarda çıktı kararlılığına ve daha güvenilir teşhise katkıda bulunur. Önerilen yöntem, Exathlon veri seti üzerinde değerlendirilmiş ve ağırlıklı ikili anomali tespit değerlendirme çerçevesinde F1 skorunda %24'lük dikkat çekici bir artış elde etmiştir.

Anahtar Kelimeler: Anomali aralığı tespiti, çok değişkenli zaman serileri, çok sınıflı sınıflandırma, dönüştürücü ağlar, çıkarım sonrası stratejiler

To my family

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my thesis advisor, Prof. Dr. Tuğba Taşkaya Temizel, for her invaluable guidance, encouragement, and insightful feedback throughout the course of this research. Her support was instrumental in shaping both the direction and quality of this thesis.

I am also sincerely thankful to the esteemed professors at the Informatics Institute of Middle East Technical University, whose instruction and mentorship have enriched my academic journey and broadened my perspective in the field of data science.

Special thanks go to my family for their unwavering love, support, and constant encouragement throughout this journey.

To my mother, Bilgi Uzuntürk, who is not only a guiding light but also someone I can talk to about anything, with trust, openness, and warmth that never wavers.

To my father, Osman Uzuntürk, whose sense of humor brings lightness to every moment, and whose calm and reassuring presence makes me feel safe and at peace no matter the circumstances.

To my sister Cansu Uzuntürk, who inspires me every day and is by my side with her strength and joy of life; her endurance and energy remind me to move forward with courage and passion.

To our beloved cat, Bonnie, whose irresistible sweetness and quiet companionship brought joy and comfort during the most demanding times of this journey.

And to the love of my life and my best friend, Ahmet Yüksel, the person with whom I share the most laughter, adventures and unforgettable moments. From concerts to travels, from everyday simplicity to extraordinary experiences, he is my partner in life and my companion in discovering new places, ideas, and parts of myself.

I am deeply grateful to my dear friends and supportive colleagues for their motivation, and belief in me. I would like to extend my sincere thanks to my team leader, Dr. Kenan Ahıska, for his professional mentorship, understanding, and support throughout this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AE	Autoencoder
ADASYN	Adaptive Synthetic Sampling
AD	Anomaly Detection
BiGAN	Bidirectional Generative Adversarial Network
CNN	Convolutional Neural Network
DEG	Disruptive Event Generator
DWT	Discrete Wavelet Transform
EEI	Extended Effect Interval
EMD	Earth Mover's Distance
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
IQR	Interquartile Range
JSD	Jensen-Shannon Divergence
LSTM	Long Short-Term Memory
MAD	Mean Absolute Deviation
MTS	Multivariate Time Series
RCA	Root Cause Analysis
RCI	Root Cause Interval
RNN	Recurrent Neural Network
SMOTE	Synthetic Minority Over-sampling Technique
SVAE	Sequential Variational Autoencoder

STD	Standard Deviation
TIM	Transformer with Inference Mode
VAE	Variational Autoencoder

CHAPTER 1

INTRODUCTION

Anomaly detection in time series data is a foundational problem with applications across various domains, including finance, healthcare, manufacturing, and cybersecurity. The field encompasses a broad spectrum of dimensions, such as the structure of the time series (univariate vs. multivariate), the types of anomalies encountered (point-based vs. range-based, binary vs. multiclass), the benchmark datasets, the variety of detection methodologies, and the evaluation metrics employed. Several taxonomies and classification frameworks have been proposed to systematize these aspects, offering critical insights into algorithmic design choices and trade-offs in different detection settings [1, 2, 3, 4].

One fundamental distinction in anomaly detection is between univariate and multivariate time series. A univariate time series consists of observations of a single variable recorded over time. This type of data focuses on the temporal dynamics of one specific attribute, making it simpler to model and analyze. Common examples include daily temperature readings or stock prices. In contrast, a multivariate time series involves simultaneous observations of two or more variables over time. This approach captures the interdependencies and correlations between multiple attributes, providing a more comprehensive understanding of complex systems. For example, monitoring various economic indicators such as inflation rates, unemployment levels, and consumer spending can offer insights on economic health. Similarly, in healthcare, tracking multiple vital signs, such as heart rate, blood pressure, and oxygen saturation, allows a holistic assessment of the condition of a patient.

The categorization of anomalies further diversifies the problem space. In time series analysis, an anomaly refers to any observation or sequence that deviates significantly from the expected temporal pattern. These deviations may indicate critical events or faults, such as cyber intrusions, system failures, or medical emergencies. In general, anomalies in time series can be classified as point anomalies and range-based anomalies (or sequence/subsequence anomalies) [1, 2]. The point anomaly is a single observation that deviates significantly from others, without context. For example, a sudden spike in CPU usage or a data corruption event. On the other hand, the range anomaly (or sequence/subsequence anomaly) is a contiguous sequence of observations that together form an abnormal pattern, such as a prolonged temperature rise indicating engine overheating. These two categories differ not only in temporal structure but also in the complexity of labeling and metrics required for evaluation strategies [1, 5, 6, 7].

Another critical dimension in anomaly detection frameworks concerns the granularity of classification, namely, binary and multiclass approaches. Traditional binary detection models classify each observation as normal or anomalous, which is often sufficient for alert-driven applications such as intrusion detection or fault monitoring. However, this binary perspective often proves inadequate in real-world

scenarios that require diagnostic specificity or causal inference. In contrast, multiclass anomaly detection frameworks aim not only to detect the presence of an anomaly but also to determine its underlying type or cause, effectively functioning as a form of root cause analysis. In certain domains, this is also called fault diagnosis [8]. Such models categorize anomalies into predefined classes, enabling targeted interventions and more informed decision making. This level of granularity is particularly critical in medical diagnostics, where therapeutic responses depend on the specific nature of the detected condition [9], or in predictive maintenance, where repair strategies vary according to the type of identified fault [10].

Despite the increasing complexity and realism of real-world applications, the intersection of multivariate, range-based, and multiclass anomaly detection remains significantly underexplored. Practical scenarios, such as monitoring patient vitals, analyzing network traffic for cybersecurity threats, or detecting coordinated sensor failures in industrial systems, frequently involve these intertwined challenges. However, existing models largely focus on point-based binary settings in univariate data, limiting their applicability. Recent benchmarking studies have highlighted the urgent need for more expressive frameworks and evaluation metrics that can accommodate the full complexity of these dimensions of the problem [4, 11, 12].

1.1 Research Questions

This thesis investigates the design and evaluation of a novel transformer-based framework for range-based multiclass anomaly detection in multivariate time series. The following research questions guide the study:

Research Question 1: *How can a transformer-based framework be designed to effectively detect range-based anomalies in multivariate time series under highly imbalanced and multiclass conditions?*

Research Question 2: *What post-inference strategies can be applied to improve the temporal and semantic coherence of predictions?*

Research Question 3: *How does the proposed transformer-based framework with post-inference strategies perform compared to existing baseline models under a range-based evaluation setting on a real-world benchmark dataset?*

1.2 Contributions of the Study

This thesis proposes a transformer-based encoder-decoder framework for range-based multiclass anomaly detection in multivariate time series, publicly available on GitHub[1]. The model is designed to capture long-range temporal dependencies and contextual relationships within high-dimensional telemetry data, using the expressive capacity of transformer architectures.

In addition to addressing class imbalance through targeted oversampling during batch construction and adjustments at the loss level, the proposed framework incorporates domain-informed post-inference strategies to enhance prediction consistency. Specifically, two mechanisms are developed to enhance

[1] GitHub: https://github.com/goksu-uzunturk/TS-Anomaly-Detection.git

both temporal coherence and semantic plausibility of the predicted anomaly sequences: (i) a majority voting scheme that consolidates overlapping n-step-ahead predictions and (ii) a transition-aware masking mechanism that enforces realistic class transitions based on domain knowledge.

Although recent work has incorporated domain knowledge to improve anomaly detection accuracy by modeling feature-level relationships through graph structures [13], but modeling domain-informed transitions across anomaly classes remains underexplored. In many real-world applications, such as industrial fault diagnosis, it is often reasonable to assume that specific sequences of anomalies are unlikely; for example, a CPU overload would not immediately result in a disk failure without an intermediate degradation phase. Motivated by this gap and inspired by constrained decoding techniques in structured generation from the field of natural language processing [14, 15], this thesis introduces a transition-aware masking approach. In constrained decoding, invalid tokens are filtered by setting their probabilities to zero during generation, ensuring the adherence to structural rules. Analogously, the masking mechanism in this study restricts invalid anomaly class transitions by zeroing out their probabilities during post-inference, thereby improving the interpretability of the output anomaly sequences.

The framework is empirically evaluated on the Exathlon benchmark [16], a comprehensive dataset that incorporates all key dimensions, range-based labeling, multiclass categorization and multivariate inputs, within a realistic time series anomaly detection setting. As another contribution to this study, the original Exathlon evaluation pipeline is extended from binary to multiclass, allowing for type-specific, range-level assessments. In the standard binary evaluation setup, the proposed method achieves a 24% improvement in the F1 score of the weighted binary anomaly detection evaluation setting, significantly outperforming both forecasting-based models [17] and reconstruction-based approaches [18, 19] with classical thresholding techniques [20].

1.3 Organization of the Thesis

This thesis is organized into six main chapters.

Chapter 1 introduces the research problem, outlines the key research questions, and highlights the main contributions of the study. It also presents the structure of the thesis.

Chapter 2 reviews the related literature in three major areas: range-based anomaly detection in multivariate time series, multiclass anomaly detection, and evaluation methodologies for range-based detection. This contextual foundation informs the design and evaluation of the proposed approach.

Chapter 3 provides an overview of the dataset used in the study. It describes the data collection process and the dataset construction methodology.

Chapter 4 presents the proposed methodology in detail. It introduces the data formulation, model architecture, and four post-inference strategies (Modes 1 through 4) based on transformer-based sequence modeling.

Chapter 5 outlines the experimental design, including baseline comparisons, data partitioning, hyperparameter optimization, and evaluation metrics. Then it discusses the experimental results.

Chapter 6 concludes the thesis by summarizing the contributions, discussing limitations, and outlining potential directions for future research.

CHAPTER 2

RELATED WORK

In this chapter, related studies are given in detail.

2.1 Range-Based Anomaly Detection Models in Multivariate Time Series

Range-based anomaly detection in multivariate time series (MTS) focuses on identifying contiguous intervals where values deviate significantly across multiple correlated variables. These methods are essential in applications such as financial fraud detection, industrial monitoring, and clinical diagnostics. Unlike point anomalies, which occur at single time steps, range anomalies often span multiple time points and exhibit complex inter-variable dependencies, making their detection considerably more challenging.

Many existing models are not explicitly designed for range-based anomalies, but can still be evaluated in such contexts [1]. These models typically fall into two primary categories: forecasting-based and reconstruction-based approaches.

Forecasting-based models aim to predict future time points and identify significant deviations between predicted and observed values as anomalies. These models are typically trained exclusively on normal data, under the assumption that deviations from learned normal patterns indicate potential anomalies.

Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, are widely used due to their ability to model long-term dependencies. For example, Bontemps et al. [17] proposed a real-time LSTM-based model trained exclusively on normal data. Rather than evaluating prediction errors at individual time steps, their model aggregates errors over a sliding window, effectively capturing collective anomalies. Similarly, Alabugin et al. [21] applied LSTM models to monitor industrial control systems, demonstrating real-time detection of process anomalies and enhancing system stability. Wang et al. [22] introduced the Forecast-based Multi-aspect Unsupervised Anomaly Detection (FMUAD) framework, comprising three modules: a temporal module (LSTM-based) for capturing sequential dependencies, a spatial module (CNN-based) for detecting feature-level anomalies, and a correlation module (attention-based) for modeling inter-variable relationships. These components are integrated to compute a robust anomaly score across different datasets.

Reconstruction-based methods learn a model of normal data and detect anomalies through reconstruction errors, operating under the assumption that anomalous inputs will be poorly reconstructed. These models include classical autoencoders, transformer-based architectures, and generative models.

Traditional autoencoders, especially those based on recurrent architectures, have been effectively applied to industrial anomaly detection tasks. For example, Yan et al. [23] developed an LSTM-based autoencoder to detect anomalies in voltage transformers. Trained solely on normal operation data, the model reconstructs voltage time series and identifies anomalies by elevated reconstruction errors, demonstrating strong results in power system monitoring. To improve temporal and variable-wise modeling, Wang et al. [24] introduced the Multiscale Wavelet Graph Autoencoder (MEGA), which integrates Discrete Wavelet Transform (DWT) to decompose time series into multi-frequency components. It also employs a dynamic graph structure to capture inter-variable dependencies. This multiscale approach allows MEGA to localize anomalies that manifest in different temporal resolutions, outperforming traditional AE models.

Transformers have recently emerged as powerful tools for MTS anomaly detection. Kang et al.[25] proposed a transformer with inter-variable attention and explainable AI components, enabling interpretation of anomaly sources. Tuli et al. [26] introduced TranAD, a transformer-based anomaly detection framework that integrates several key components to enhance performance. First, it employs attention-based sequence encoders to effectively capture long-range temporal dependencies. Second, it incorporates dual adversarial training, which improves robustness and prevents overfitting. Third, it applies a focus score-based self-conditioning mechanism, which reinforces the learning of temporal patterns across time steps. Although TranAD assigns anomaly scores at each individual timestep, its architecture is inherently sensitive to multi-step trends, enabling it to detect not only isolated point anomalies but also extended range-based anomalies. The Anomaly Transformer proposed by Xu et al. [27] introduces a novel concept of association discrepancy, where it contrasts expected and observed attention patterns to flag anomalies. This model goes beyond the reconstruction error, using the attention behavior itself as an anomaly signal, enhancing both detection and interpretability.

Generative models provide another reconstruction-based paradigm. Su et al.[28] developed Omni-Anomaly, which combines the temporal modeling power of RNNs with the probabilistic generative capacity of Variational Autoencoders (VAEs). The model captures uncertainty in sequence generation, using probabilistic reconstruction errors for anomaly scoring. Miao et al. [29] explored Generative Adversarial Networks (GANs) for unsupervised anomaly detection in high-dimensional data. At its core, the model uses a transformer-based autoencoder to learn the structure of normal data, embedded within a GAN framework to enhance realism and robustness. GANs operate through a game-theoretic process: a generator tries to produce realistic samples that resemble normal data, while a discriminator tries to distinguish between the real data and the generated (fake) ones. The generator improves by learning to "fool" the discriminator, and this adversarial training enables it to better model complex data distributions. In this study, the authors also introduced contrastive loss in the discriminator, helping it focus on distinguishing real from fake in a more nuanced way. Once trained, the model detects anomalies based on reconstruction errors: if a data point cannot be well reconstructed (i.e., the error is large), it is likely anomalous.

Despite advances, many models rely on point-wise thresholding of anomaly scores [20], often resulting in fragmented or noisy outputs, particularly problematic when detecting range-based anomalies. To address these issues, recent research has introduced methods incorporating temporal smoothness, spatio-temporal constraints, and sequence-level refinement.

Li et al. [30] proposed the Sequential Variational Autoencoder (SVAE), which imposes a priori smoothness on latent representations using a recurrent structure. Unlike traditional approaches that enforce

smoothness directly on output scores, SVAE maintains consistency in latent dynamics, helping to prevent abrupt anomalies due to noise, and improving the overall temporal coherence of detection.

Ge et al. [31] integrated spatio-temporal constraints via graph contrastive learning. Temporal constraints enforce smooth intra-variable behavior over time (e.g., stable voltage patterns), while spatial constraints capture inter-variable relationships (e.g., consistent correlations between voltage and current in power systems). Their framework combines a temporal module with a graph-based spatial encoder, enabling the model to discern normal and abnormal interactions across both dimensions.

To go beyond point-wise constraints, Li et al. [32] introduced a sequence-level post-inference refinement framework to reduce false positives. Their approach identifies and clusters false-positive subsequences (FPS) from training (normal) data as reference exemplars. During inference, detected anomalies are compared to FPS patterns using time- and frequency-domain features. If similarity is high, the detection is suppressed, thus reducing noisy alerts in dynamic settings.

Additional frameworks have emerged to further support range-sensitive detection. GUARD (Multigranularity-based Unsupervised Anomaly Detection) [33] analyzes time series at multiple granularities. It segments time series using overlapping sliding windows of varying sizes to capture both short-term spikes and long-term drifts. For each window, GUARD computes two types of scores: trend-based (detecting individual variable deviations) and correlation-based (detecting abnormal inter-variable relationships). These scores are normalized and fused across all granularities to produce a robust anomaly profile. The anomalies are then flagged when the fused score exceeds a threshold. Using multigranularity and unsupervised scoring, GUARD effectively captures complex, range-spanning patterns in high-dimensional time series.

Series2Graph++ [34] is specifically designed for range-based anomaly detection, which means it identifies anomalous subsequences (time ranges) within multivariate time series rather than isolated point anomalies. Series2Graph++ operates by segmenting the time series into subsequences and embedding these into a graph structure. Each subsequence becomes a node in a graph, and transitions between them form edges. Anomalies are detected by identifying rare nodes or transitions, which correspond to entire segments of data that deviate from normal patterns.

Although recent models have significantly improved range-based anomaly detection, most remain limited to binary classification, differentiating between normal and anomalous ranges. There is a notable lack of methods that address multiclass range-based anomaly detection, where models must classify and localize distinct types of anomaly at the sequence level. This remains a critical open challenge in the field and represents a key direction for future research.

2.2 Multiclass Anomaly Detection Models in Multivariate Time Series

Multiclass anomaly detection in MTS data is an increasingly active area of research, particularly in domains such as cyber-physical systems, predictive maintenance, and healthcare diagnostics. Unlike binary detection tasks that only distinguish between normal and abnormal conditions, multiclass approaches aim to identify and categorize different types of anomalies, each with potentially distinct characteristics and causes. This introduces additional modeling challenges, as it necessitates not only the detection of anomalies but also their precise classification and, in some cases, diagnosis.

Root cause analysis (RCA) is frequently cast as a multiclass problem, especially in systems where different anomaly types originate from specific causal mechanisms. For example, the AERCA framework [35] incorporates Granger causal discovery to detect anomalies and identify the responsible variables, thus classifying the root causes. Similarly, the ICODE model [36] distinguishes between cyber and measurement anomalies by learning causality-aware propagation patterns, by formulating root cause localization as a structured classification task. These examples demonstrate how detection, diagnosis, and causality analysis are increasingly integrated into contemporary anomaly detection pipelines.

Deep learning techniques have shown strong performance in structured anomaly detection tasks that involve multivariate time series data. CNNs, for example, have been used effectively to extract hierarchical features from sensor data and differentiate between various anomaly types. Zhao et al. [11] applied CNNs to multiclass anomaly classification in structural health monitoring, achieving robust performance despite severe class imbalance.

RNNs, particularly LSTM networks, have also been applied due to their ability to model sequential dependencies. In clinical time series applications, Lipton et al. [37] used LSTM networks to classify multiple diagnostic categories, demonstrating their effectiveness in capturing temporal dynamics and supporting multiclass output.

More recently, transformer-based architectures have been adopted for their ability to model long-range and inter-variable dependencies through self-attention mechanisms. Wang et al. [38] proposed a two-stage framework combining graph attention and transformers to detect and classify anomalies in complex sensor networks.

Some models address anomaly detection and classification in separate stages. Detection typically refers to identifying the presence of an anomaly, while classification involves assigning a specific label or determining the underlying cause. For example, the AERCA model [35] initially detects anomalies through causal inference and then identifies the responsible variables, treating the diagnosis as a follow-up step. The ICODE framework [36] treats detection and root cause analysis as distinct but complementary tasks, embedding explainability throughout the pipeline. A comparable separation exists in the transformer-based model by Wang and Liu [38], where anomaly detection is carried out using a graph-enhanced transformer, followed by classification using a prototype network. In contrast, other models adopt an integrated approach. The CNN-based method by Zhao et al. [11] and the LSTM-based system by Lipton et al. [37] are trained end-to-end to detect and classify anomalies simultaneously. These unified strategies are particularly common in fields such as structural health monitoring and medical diagnosis.

Despite significant progress, class imbalance remains a major challenge in multiclass anomaly detection. In many applications, certain anomaly categories are significantly underrepresented, leading to biased model performance and limited generalization. Several strategies have been developed to address this issue, categorized into data-level and loss-level approaches.

These approaches focus on adjusting the training data to produce a more balanced distribution. Common techniques include oversampling minority classes, undersampling majority classes, and generating synthetic samples. Among these, the Synthetic Minority Over-sampling Technique (SMOTE) [39] and its time series adaptation [40] are widely used. SMOTE interpolates between existing minority class samples to produce synthetic instances that enhance class diversity.

An extension of SMOTE, known as Adaptive Synthetic Sampling (ADASYN) [41], concentrates synthetic sampling efforts in regions where minority class instances are more difficult to classify. Xue and Zhu [42] developed a hybrid approach that combines ADASYN-based oversampling with majority-class undersampling. Their method also incorporates a weighted majority voting ensemble, where the influence of each classifier on the final decision is determined by its performance, especially in underrepresented classes.

Another emerging technique involves GANs for oversampling. GAN-based methods train a generator to produce synthetic samples that closely resemble real minority class data, improving class balance in high-dimensional and structured domains such as time series and medical imaging [43].

Loss-level strategies adjust the loss function during training to reduce the influence of class imbalance. Wang et al. [44] proposed a cost-sensitive hybrid model that combined CNN and GRU architectures with a custom loss function that assigns higher penalties for errors in minority classes. This approach improved the detection accuracy for rare anomaly types in time-series data. Similarly, Qi et al. (2023) introduced a dynamic cost-sensitive weighting mechanism that adapts class weights based on per-class performance metrics throughout training. Their model, which uses multi-head self-attention to capture temporal patterns, improved detection of minority-class anomalies.

Similar to integrated approaches that combine anomaly detection and classification within a single model [11, 37], the proposed framework addresses multiclass anomaly detection in an end-to-end manner. This unified design reduces overall system complexity and enables the learning of shared temporal representations that support both detection and classification. To handle the class imbalance problem, the framework incorporates strategies at both the data and loss levels. Specifically, minority-aware batch sampling ensures adequate representation of rare anomaly types during training, while a weighted loss function assigns greater importance to errors involving underrepresented classes.

As discussed in Section 2.1, most existing multiclass anomaly detection models primarily focus on point-wise anomalies. A significant gap remains in the literature regarding approaches capable of simultaneously addressing range-based and multiclass anomalies in multivariate time series data. This thesis aims to address both aspects of the problem within a unified framework.

2.3 Range-Based Anomaly Detection Evaluation

Traditional anomaly detection generally focuses on point-based anomalies that occur at isolated time steps. However, anomalies in real-world systems often unfold over extended temporal intervals, rendering point-wise evaluation insufficient. To address this limitation, Tatbul et al. [5] introduced a range-based evaluation framework that extends classical precision and recall metrics to support interval-based assessment. This framework incorporates several key components to align the evaluation with domain-specific priorities.

Size of Overlap: Larger intersections between predicted and ground-truth ranges are rewarded more heavily.

Positional Bias: Overlaps are weighted based on their position within the true anomaly range, allowing emphasis on early or late detection.

Cardinality of Overlaps: The score is adjusted based on the number of predicted ranges that intersect with a single ground-truth interval, discouraging redundant detections when appropriate.

This framework has become fundamental in the evaluation of time series anomaly detection systems. It has been adopted in a variety of recent empirical and benchmarking studies, including large-scale comparative evaluations [1], benchmarking toolkits [45], and deep learning-based anomaly detection frameworks [46]. In addition, recent survey work [4] has emphasized the growing complexity and diversity of evaluation metrics in the field, providing a comprehensive taxonomy and demonstrating that metric choice significantly influences performance interpretation. These efforts collectively reinforce the utility and generalizability of range-based metrics in assessing detection performance across practical applications and research benchmarks.

Building on this foundation, the Exathlon benchmark [16] formalizes detection evaluation through four levels of anomaly detection, namely AD-1, AD-2, AD-3 and AD-4, where each imposes progressively stricter criteria:
AD-1 (Anomaly Existence): Measures whether an anomaly is detected anywhere within the root cause interval (RCI) or the extended effect interval (EEI), capturing the presence of anomalous behavior.
AD-2 (Range Detection): Evaluates whether the predicted anomaly range aligns with the ground-truth interval, focusing on temporal accuracy.
AD-3 (Early Detection): Focuses on detecting anomalies as early as possible within the labeled interval, rewarding prompt identification.
AD-4 (Exactly-Once Detection): Enforces a strict detection policy by requiring anomalies to be reported exactly once, penalizing redundant or fragmented predictions.

As used in this thesis, the parameter configurations corresponding to the AD levels are summarized in Table 1 and further detailed in Equations 1–8.

Table 1: Parameter configurations for each anomaly detection (AD) level.

AD Level	α_R	ω_R	δ_R	γ_R	ω_P	δ_P	γ_P
AD-1 (Existence)	1	default	flat	dup	default	flat	dup
AD-2 (Range Detection)	0	default	flat	dup	default	flat	dup
AD-3 (Early Detection)	0	flat.normalized	front	dup	default	flat	dup
AD-4 (Exactly Once)	0	flat.normalized	front	no.dup	default	flat	no.dup

Let the ground-truth anomaly ranges and the predicted anomaly ranges be defined as follows, respectively:

$$T = \{T_1, T_2, \ldots, T_m\}, \quad P = \{P_1, P_2, \ldots, P_k\} \tag{1}$$

where each T_i and P_j is an interval representing the start and end positions of the anomaly. The evaluation involves calculating recall and precision based on range-level comparisons, incorporating detection characteristics through reward functions.

The recall score is computed over all ground-truth anomaly ranges $T_i \in T$ as:

$$\text{Recall} = \frac{1}{m} \sum_{T_i \in T} [\alpha_R \times \text{Existence Reward}(T_i, P) + (1 - \alpha_R) \times \text{Overlap Reward}(T_i, P, \omega_R, \delta_R, \gamma_R)] \tag{2}$$

where m is the number of true ranges and α_R is the weight of Existence Reward in the formulation.

The precision is computed over all predicted anomaly ranges $P_j \in P$ as:

$$\text{Precision} = \frac{1}{k} \sum_{P_j \in P} \text{Overlap Reward}(P_j, T, \omega_P, \delta_P, \gamma_P) \tag{3}$$

where k is the number of predicted ranges.

The Existence Reward is defined as:

$$\text{Existence Reward}(T_i, P) = \begin{cases} 1, & \text{if } \exists P_j \in P \text{ such that } T_i \cap P_j \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The Overlap Reward is defined for a target range R ($= T_i$ for recall and $= P_j$ for precision) and a set of comparison ranges S ($= P$ for recall and $= T$ for precision) as follows:

$$\text{Overlap Reward}(R, S, \omega, \delta, \gamma) = \gamma(n) \times \sum_{S_j \in S} \omega(R, R \cap S_j, \delta) \tag{5}$$

where $n = |\{S_j \in S : R \cap S_j \neq \emptyset\}|$ denotes the number of overlapping ranges in S with the target range R. The function $\gamma(n)$ applies a cardinality-based adjustment depending on the number of overlaps. The function $\omega(R, R \cap S_j, \delta)$ assigns a size-based reward to each overlap, modulated by a positional bias function δ.

The gamma functions $\gamma(n)$ apply a penalty or adjustment based on the number of overlapping ranges:

$$\gamma(n) = \begin{cases} 1, & \text{if } n \geq 1 \text{ (dup case in Table 1)} \\ 0, & \text{if } n \geq 1 \text{ (no.dup case in Table 1)} \\ 0, & \text{if } n = 0 \text{ (dup, no.dup cases in Table 1)} \end{cases} \tag{6}$$

The delta functions $\delta(i, L)$ define a normalized positional bias over a range of length L for the index $i \in \{0, 1, \ldots, L-1\}$. These functions are defined as:

$$\delta(i, L) = \begin{cases} \frac{1}{L}, & \text{(flat case in Table 1))} \\ \frac{2(L-1-i)}{L(L-1)}, & \text{(front case in Table 1))} \end{cases} \tag{7}$$

The omega functions $\omega(R, O, \delta)$ define the reward for an overlapping segment $O = [o_s, o_e]$ within a target range $R = [r_s, r_e]$, incorporating both the overlap size and positional bias. The reward is computed as:

$$\omega(R, O, \delta) = \begin{cases} \sum_{i=o_s-r_s}^{o_e-r_s} \delta(i, |R|), & \text{(default case in Table 1))} \\ \frac{\sum_{i=o_s-r_s}^{o_e-r_s} \delta(i,|R|)}{\max_{\substack{A \subseteq \{0,\ldots,|R|-1\} \\ |A|=|O|}} \sum_{i \in A} \delta(i,|R|)} \cdot \frac{|O|}{|R|}, & \text{(flat.normalized case in Table 1))} \end{cases} \tag{8}$$

where:

- $|R| = r_e - r_s + 1$ is the length of the target range.
- $|O| = o_e - o_s + 1$ is the length of the overlap.
- $\delta(i, |R|)$ is the weight of positional bias in relative index i within the range R.
- The set $A \subseteq \{0, 1, \ldots, |R| - 1\}$ contains the indices with the highest positional bias values, selected to match the overlap size such that $|A| = |O|$.

In the `default` variant, the reward is computed as the sum of the positional weights over the overlap, allowing bias functions to emphasize specific regions (e.g., early detection via `front` bias). In the `flat.normalized` variant, the reward is scaled so that it does not exceed the value assigned under a flat positional bias, i.e., $\frac{|O|}{|R|}$. This normalization ensures that the positional bias does not amplify the reward beyond the uniformly weighted case.

CHAPTER 3

DATASET

In this study, experiments were carried out using the Exathlon benchmark dataset [16], which represents the first publicly available benchmark for explainable anomaly detection in high-dimensional telemetry. The dataset provides labeled root-cause intervals within distributed stream processing systems, allowing both detection and diagnostic tasks. This chapter outlines the details of the dataset.

3.1 Spark Application Process

In the dataset, Apache Spark applications are deployed in a cluster to process large-scale data analytics, critical for business operations such as e-Commerce. These applications run on a four-node cluster equipped with high-performance processors and large memory and storage. The Spark system is extensively monitored during these applications. The application execution flow is as follows:

System Setup: Applications analyze streaming data, such as user click streams, through a controlled setup where data sender servers feed streams into the Spark cluster at a managed rate.

Application Execution: Spark initiates a Driver process that coordinates the execution across the cluster using a resource manager (YARN), which assigns executor processes to nodes. These executors handle tasks in parallel, and each node can manage multiple tasks simultaneously from different applications.

Metrics Collection: Spark applications are monitored via the Spark Monitoring and Instrumentation Interface and the operating system tools. A total of 2,283 metrics are collected to capture detailed behaviors at both the Spark and OS levels:

- **Spark UI Metrics:** 243 metrics are collected from the driver, including scheduling delays and statistics on received and processed data.
- **Executor Metrics:** Each of the five executors (three active and two backup) contributes 140 metrics covering timing, data sizes, network activity, memory usage, and I/O, totaling 700 metrics.
- **Operating System Metrics:** Each of the four cluster nodes collects 335 metrics using the `nmon` utility, covering CPU time, memory consumption and network throughput, resulting in 1,340 metrics in total.

Trace Collection and Analysis: The system records the output of each application run (called a trace) over a period of 2.5 months. These traces are analyzed to differentiate normal operations from

injected anomalies, enabling evaluation of the resilience of the system and the effectiveness of anomaly detection.

3.2 Dataset Construction

The dataset is constructed from 10 Spark streaming applications, which process user click streams from the 1998 World Cup website [47], using a batch interval of 5 seconds. These applications run in a four-node cluster for 2.5 months, generating data with a resolution of one second. The data collected from each run of a Spark streaming application is called a trace. Each of the 10 applications performs different operations involving group-by aggregations (counts or sums), join operations, filtering, user-defined functions (UDF) and varies in terms of window type (sliding vs. jumping), parameters (size and slide) and filtering conditions. The detailed specifications of these applications are provided in Appendix A.

The benchmark adopts a methodology inspired by chaos engineering, similar to the practices used by companies such as Netflix [48], to validate system reliability through stress tests such as workload surges and failures. Initially, the benchmark produces undisturbed traces that mimic standard Spark cluster operations, capturing routine system behaviors and inherent variations. This phase serves to establish a baseline for normal operation. Subsequently, disturbed traces are introduced, simulating six types of anomalies designed not to cause immediate crashes but to be detectable and traceable to their origins.These disturbances help evaluate the anomaly detection's efficacy in a controlled yet realistic setting, reflecting typical challenges in maintaining large-scale data analytics platforms. Each anomaly is annotated with a class label ranging from 0 (normal) to 7 (unknown), and is associated with two temporal intervals: a root cause interval (RCI) and an extended effect interval (EEI). Together, these intervals define the full duration of the anomaly, from the onset of the initial fault to its downstream impact, forming a single anomaly range. The anomaly types and corresponding EEI rules are as follows:

T1 – Bursty Input: Simulates sudden input rate spikes, for example, during special events, leading to surges in data that Spark struggles to process due to resource constraints. The system exhibits increased processing times and memory usage, causing data to queue up, which is evident from the extended scheduling delays. For simulation, the Disruptive Event Generator (DEG) increases the input load by 15 to 30 minutes, resulting in system overloads marked by increased memory usage, processing time, and scheduling delay. EEI ends when these metrics return to normal.

T2 – Bursty Input Until Crash: Extends T1 with a sustained high input rate that eventually exhausts system memory, leading to out-of-memory errors and executor crashes. No EEI is defined as the root cause ends with the application crash.

T3 – Stalled Input: Represents scenarios where input to the Spark application ceases, usually due to a data source failure. All data processing metrics, including the number of processed records and processing time, drop to zero, indicating inactivity despite the fact that the resources remain active and consume power. The DEG periodically sets the input rate to zero for approximately 15 minutes at a time. EEI ends when normal processing resumes.

T4 – CPU Contention: Consume CPU resources allocated to the Spark application. This contention slows down the application's data processing abilities. If severe enough, it can lead to processing

delays and potentially crash the driver node, disrupting the application's operation. This anomaly is created by the DEG, which runs Python scripts that aggressively consume CPU resources on a Spark node. EEI ends when metrics normalize or, in the case of a crash, typically one minute after application restart.

T5 – Driver Failure: Represents sudden failures in the Spark system's infrastructure, often caused by hardware faults or maintenance activities. Driver Failure is simulated by forcibly terminating the driver process, which immediately halts the Spark application since the driver manages task distribution and scheduling. As a result, system metrics reset to initial values, and the application enters a paused state. The system then automatically triggers a restart sequence, typically taking around 20 seconds, during which both the driver and all executors are reinitialized. This recovery introduces temporary but significant delays in processing. EEI ends when the application has successfully restarted and resumes normal operation.

T6 – Executor Failure: Represents sudden failures in the Spark system's infrastructure, often caused by hardware faults or maintenance activities, similar to T5. Executor Failure is mimicked by forcibly crashing the executor processes. This disrupts the data processing as executors are responsible for executing tasks. When an executor fails, it is automatically restarted by the driver after a short delay (about 10 seconds), but the effects on processing and scheduling may persist longer due to partial data processing and recovery overhead. EEI ends when scheduling delays normalize, which may occur after an executor is replaced or the application recovers from the disruption.

T7 – Unknown Anomalies: Manually labeled abnormal patterns from 11 traces that do not match predefined types. EEI is manually defined based on observed downstream effects.

In this study, the official preprocessing pipeline of the Exathlon benchmark [16] is adopted. The raw dataset consists of 2,283 high-dimensional features collected at each time step, capturing Spark internal metrics and system-level monitoring signals. These features are first subjected to systematic preprocessing, including cleaning, normalization, and dimensionality reduction.

To ensure temporal consistency, missing values, recorded only for inactive Spark executors, are imputed using forward filling, a technique in which the last valid observation is propagated forward in time. This method is appropriate in this context because the absence of data is semantically equivalent to an executor being inactive (i.e., not producing updates), which aligns with the system's behavior. In the raw data, these cases were previously represented as NaN or -1, both indicating a lack of update within the expected interval. Forward filling prevents artificial discontinuities in the time series while preserving the natural temporal dynamics of active metrics.

Following imputation, the data is resampled at 15-second intervals using mean aggregation to standardize the temporal resolution across all traces. In this study, resampling is performed to reduce the time complexity during both training and inference, given the extremely high temporal resolution and volume of the raw data. Each trace, corresponding to a single application run, lasts approximately 7 hours and is recorded at 1-second resolution, resulting in roughly 25,000 time steps per trace. With 59 undisturbed and 34 disturbed traces, the dataset contains millions of time points, which makes direct processing computationally intensive. After resampling, each trace is independently normalized via Z-score scaling: for each feature, the mean and standard deviation are computed over time and used for standardization.

A domain-informed feature selection strategy is applied to reduce redundancy and retain the most diagnostically relevant signals, yielding a curated set of 19 features. As detailed in Table 2, several features undergo specific transformations, such as first-order differencing to better capture changes over time. In particular, all executor-level features are first averaged across active Spark executors before differencing is applied. This approach reflects expert knowledge of Spark's operational semantics and system behavior. In particular, this domain-knowledge-guided feature selection significantly outperforms PCA-based alternatives, as demonstrated in the original Exathlon study [16], highlighting the value of interpretable and task-specific feature engineering in high-dimensional time series anomaly detection. Representative patterns of the selected features across different anomaly types are visualized in Appendix A.

Table 2: Summary of selected features in the Exathlon dataset.

Feature Category	**Description**	**Count**
Driver	Processing delay, scheduling delay, and total delay of the last completed batch	3
	First-order difference in the number of completed batches, received records, and processed records	3
	First-order difference in the number of records in the last received batch	1
	First-order difference in total memory used	1
	First-order difference in heap memory used by the JVM	1
Operating System	First-order difference in global CPU idle percentage for each of the four cluster nodes	4
Executor	Average number of HDFS write operations followed by first-order difference	1
	Average CPU time and runtime followed by first-order difference	2
	Average number of records read and written during shuffle operations followed by first-order difference	2
	Average heap memory used by the JVM followed by first-order difference	1

After preprocessing, Table 3 summarizes the distribution and temporal characteristics of the anomaly types in the Exathlon dataset. For each anomaly type, the table reports the number of traces in which the anomaly appears, the number of anomaly ranges, and the total number of point-level instances. Additionally, the duration of each anomaly range (RCI + EEI), is provided in minutes, showing the minimum, average, and maximum duration per range. The dataset includes both undisturbed and disturbed trace types. Normal behavior is present in all traces, corresponding to 96,571 instances in 59 undisturbed traces and 51,794 instances in 34 disturbed traces. However, anomalous behaviors (T1-T7) occur only in disturbed traces and show substantial variability in both frequency and temporal extent. It is important to note that some traces contain multiple anomaly types, for example, both T5 and T6 may appear within the same trace. However, anomaly ranges are strictly non-overlapping

within any single sequence. That is, there is no temporal intersection between different anomaly types within a given trace.

Table 3: Anomaly type range durations and distributions in the Exathlon dataset.

Trace Type	Anomaly Type	Range Duration (minutes) (min / avg / max)	# Traces	# Ranges	# Instances
Undisturbed	Normal	--- / --- / ---	59	–	96,571
Disturbed	Normal	--- / --- / ---	34	–	51,794
Disturbed	T1	15 / 22 / 33	6	29	2,573
Disturbed	T2	8 / 35 / 90	7	7	1,001
Disturbed	T3	14 / 16 / 16	4	16	1,031
Disturbed	T4	8 / 16 / 27	6	26	1,626
Disturbed	T5	1 / 1 / 1	8	9	44
Disturbed	T6	2 / 23 / 168	8	10	918
Disturbed	T7	1 / 5 / 11	11	12	257

CHAPTER 4

METHODOLOGY

This study proposes a transformer-based framework for addressing the challenges of highly imbalanced, multiclass, range-based anomaly detection in multivariate time series data. This chapter outlines the full methodology pipeline, including the organization of the dataset, model architecture, and the post-inference techniques used to improve temporal and semantic consistency of predictions.

4.1 Notations

The key notations used throughout this paper are summarized in Table 4:

Table 4: Notations

Symbol	Description
b	Batch size
w	Window size
s	Stride
d	Number of telemetry features
c	Number of classes
n	Number of future time steps predicted.
m	Number of minority classes
e	Embedding dimension in transformer-based architecture
X	Input to the model
P	Predicted class probability distributions for n future steps.
$\hat{\text{Y}}$	Predicted class labels for n future steps

4.2 Dataset Organization

To effectively train the transformer-based architecture on the dataset, three preprocessing stages are applied: (1) input-output sequence formulation for modeling temporal dynamics, (2) batch construction with minority class balancing strategies, and (3) data partition by cross-validation (presented in the Experimental Setup Section 5.2).

4.2.1 Input-Output Sequence Formulation for Modeling

The dataset consists of multivariate time series data, where multiple telemetry signals are recorded over time. To model temporal dependencies, a sliding window segmentation strategy is applied by transforming continuous time series into overlapping windows (or patches) of fixed length w. A stride parameter s controls the degree of overlap between adjacent windows, allowing the model to observe the shared temporal context between segments.

Each window is composed of two parts: $\mathbf{X}_{\text{ts}}$, representing the telemetry feature sequences, and $\mathbf{X}_{\text{c}}$, one-hot encoding of the corresponding anomaly type for each time step. These components are concatenated to form the final input tensor:

$$\mathbf{X} = [\mathbf{X}_{\text{ts}}, \mathbf{X}_{\text{c}}] \in \mathbb{R}^{b \times w \times (d+c)}, \tag{9}$$

The model learns a function f_θ that maps each input patch to a sequence of predicted class probability distributions over a horizon of length n, defined as:

$$\mathbf{P} = f_\theta(\mathbf{X}), \quad f_\theta : \mathbb{R}^{b \times w \times (d+c)} \to \mathbb{R}^{b \times n \times c}, \tag{10}$$

The final predicted class labels are obtained by taking the class with the highest probability at each time step:

$$\hat{\mathbf{Y}} = \arg \max_{j \in \{1,\ldots,c\}} \mathbf{P}[:, :, j], \quad \hat{\mathbf{Y}} \in \mathbb{R}^{b \times n}. \tag{11}$$

4.2.2 Batch Organization

A targeted batch construction strategy is adopted to address the severe class imbalance observed in the dataset, as detailed in Table 3. The designation of minority classes is guided by statistical considerations. Specifically, anomaly types whose number of instances falls below a defined empirical threshold are considered a minority. In this study, the threshold is set to 5% of the average number of instances in all types of anomalies, which is equivalent to approximately 53 instances. Based on this criterion, T5, with only 44 instances across 9 ranges, is identified as the sole minority class (m=1).

As depicted in Fig. 1, the training data windows are first partitioned into two groups: (i) minority class pools, containing at least one instance of the minority class, and (ii) the other pool, composed of remaining windows. Each training batch is initialized by randomly sampling one instance from the minority class pool to ensure that rare classes are consistently represented. This results in m minority sample per batch. The remaining $b - m$ slots in a batch are populated by randomly selecting windows from the other pool. Once selected, a sample is removed from the pool to prevent duplication. If the minority pool is exhausted, it is replenished, effectively implementing sampling with replacement. This batch generation process continues until the other pool is fully depleted. Finally, the constructed batches are randomly shuffled to avoid unintended sequential patterns, promoting generalization.

4.3 Model Architecture

In this study, transformer-based encoder-decoder architecture [49] is used for multiclass anomaly range detection, as illustrated in Fig. 2. The model processes multivariate time series data and predicts the

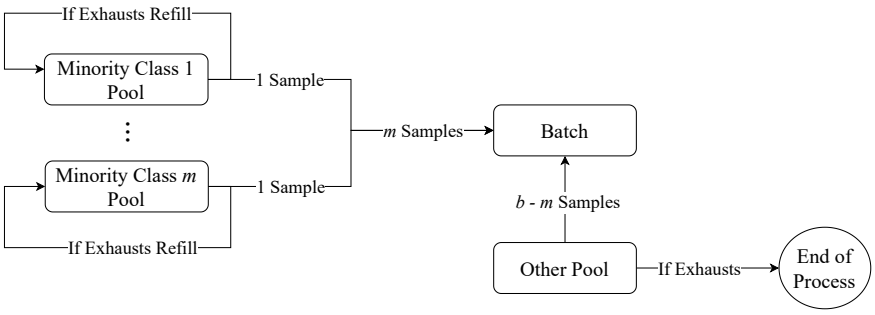


Figure 1: Batch construction process with minority class balancing.

class labels for the subsequent n time steps. In this setup, the encoder is designed to extract temporal patterns and anomaly-relevant features from the raw input sequence, while the decoder focuses on learning the structure of the anomaly ranges to generate accurate range-level class label predictions.

The input sequence $X \in \mathbb{R}^{b \times w \times (c+d)}$ is first mapped to a high-dimensional embedding space by a linear layer, followed by layer normalization and dropout. Sinusoidal positional encoding is added to incorporate temporal information, resulting in $X' \in \mathbb{R}^{b \times w \times e}$.

The enriched embeddings are then passed through a transformer encoder to capture temporal dependencies using self-attention mechanisms. The encoder output $Z \in \mathbb{R}^{b \times w \times e}$ is fed into a transformer decoder, which uses learnable query embeddings and causal masking to ensure auto-regressive prediction over the next n time steps. The decoder outputs $Z' \in \mathbb{R}^{b \times n \times e}$.

Finally, a linear classification layer maps the decoder output to class logits $L \in \mathbb{R}^{b \times n \times c}$ for each predicted time step. These logits are optionally passed through a $softmax$ function to obtain class probabilities $P \in \mathbb{R}^{b \times n \times c}$. The final predicted class labels are obtained by selecting the class with the highest probability at each time step using the $argmax$ function, resulting in $\hat{\mathbf{Y}} \in \mathbb{R}^{b \times n}$.

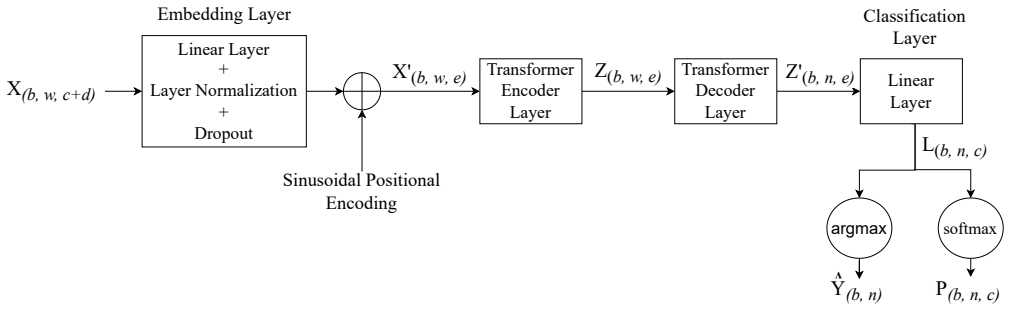


Figure 2: Transformer-based encoder-decoder architecture.

Training is performed using a weighted Cross-Entropy loss to address the class imbalance. The class weights are calculated inversely proportional to their frequency and scaled to be within $[0.3, 1]$ to avoid extreme weight values. The final loss function is:

$$\mathcal{L} = -\sum_{i=1}^{c} w_i \cdot y_i \cdot \log(\hat{y}_i), \tag{12}$$

where w_i represents the weight of class i, y_i and $\hat{y}_i$ denote the true and predicted class probabilities, respectively.

4.4 Post-Inference Techniques

As part of the proposed methodology, four post-inference strategies are introduced. In all cases, inference is performed auto-regressively with a stride of one, where previously predicted outputs are recursively fed back into the model as inputs. This sliding window mechanism produces multiple overlapping predictions for each time step.

Rather than limiting the model to one-step-ahead forecasting, each inference step generates predictions for n future time steps. This multi-step-ahead strategy improves temporal coherence, reduces cumulative prediction error, and provides multiple candidate labels for each time index, which enhances the stability of post-inference decisions.

The first two strategies, called Mode 1 and Mode 2, differ in how they consolidate overlapping predictions. Mode 1 selects the most recent prediction as the final label, while Mode 2 applies majority voting across overlapping outputs. Mode 3, introduced in this study, incorporates domain-informed transition constraints to enforce plausible transitions between anomaly types and to prevent abrupt class changes. Mode 4 combines the majority voting of Mode 2 with the transition rules of Mode 3, resulting in a hybrid approach that incorporates both statistical agreement and temporal constraints informed by domain knowledge.

4.4.1 Mode 1: Naive Auto-Regressive Inference

In Mode 1, the model performs auto-regressive forecasting by retaining only the first predicted class at each step and feeding it into the next input window. This baseline approach iteratively generates predictions over the entire trace using its own most recent output.

Let the window size be $w = 40$, and let the model predict the next $n = 10$ time steps. At each inference step t, the model receives an input tensor $\mathbf{X}(t)$, which is the concatenation of multivariate time series features $\mathbf{X}_{\text{ts}}(t) \in \mathbb{R}^{w \times d}$ and one-hot encoded class labels $\mathbf{X}_{\text{c}}(t) \in \mathbb{R}^{w \times c}$, such that:

$$\mathbf{X}(t=1) = [\mathbf{X}_{\text{ts}}(1),\ \mathbf{X}_{\text{c}}(1)] \tag{13}$$

$$\mathbf{X}_{\text{ts}}(1) = [x_1^{\text{ts}}, x_2^{\text{ts}}, \ldots, x_{40}^{\text{ts}}], \quad \mathbf{X}_{\text{c}}(1) = [x_1^{\text{c}}, x_2^{\text{c}}, \ldots, x_{40}^{\text{c}}] \tag{14}$$

The model then produces a predicted sequence of class labels for the next $n = 10$ time steps:

$$\hat{\mathbf{Y}}(t=1) = [\hat{y}_{41}(1), \hat{y}_{42}(1), \ldots, \hat{y}_{50}(1)] \tag{15}$$

Only the first prediction $\hat{y}_{41} = \hat{y}_{41}(1)$, corresponding to the time step $t = 41$, is retained and used to construct the next input window:

$$\mathbf{X}(t=2) = [\mathbf{X}_{\text{ts}}(2),\ \mathbf{X}_{\text{c}}(2)] \tag{16}$$

$$\mathbf{X}_{\text{ts}}(2) = [x_2^{\text{ts}}, x_3^{\text{ts}}, \ldots, x_{41}^{\text{ts}}], \quad \mathbf{X}_{\text{c}}(2) = [x_2^{\text{c}}, x_3^{\text{c}}, \ldots, \hat{y}_{41}] \tag{17}$$

The model then generates class label predictions for the next $n = 10$ time steps, based on the updated input that incorporates previously predicted labels.

$$\hat{\mathbf{Y}}(t=2) = [\hat{y}_{42}(2), \hat{y}_{43}(2), \ldots, \hat{y}_{51}(2)] \tag{18}$$

Only the first prediction $\hat{y}_{42} = \hat{y}_{42}(2)$, corresponding to the time step $t = 42$, is retained, and the process continues in an auto-regressive manner. As a result, the final sequence of predicted labels generated through this iterative procedure is:

$$\hat{\mathbf{Y}} = [\hat{y}_{41},\ \hat{y}_{42},\ \hat{y}_{43},\ \ldots] = [\hat{y}_{41}(1),\ \hat{y}_{42}(2),\ \hat{y}_{43}(3),\ \ldots] \tag{19}$$

4.4.2 Mode 2: Voting-Based Auto-Regressive Inference

In Mode 2, the model performs auto-regressive forecasting using majority voting to resolve overlapping predictions. This approach aggregates the predictions from the past v inference steps and selects the most frequent class label. The parameter v defines the size of the majority voting window.

Let the window size be $w = 40$ and the majority voting window size be $v = 3$. At each inference step t, the model receives input $\mathbf{X}(t) = [\mathbf{X}_{\text{ts}}(t), \mathbf{X}_{\text{c}}(t)]$, where:

$$\mathbf{X}_{\text{ts}}(t) = [x_t^{\text{ts}}, x_{t+1}^{\text{ts}}, \ldots, x_{t+w-2}^{\text{ts}}, x_{t+w-1}^{\text{ts}}] \tag{20}$$

$$\mathbf{X}_{\text{c}}(t) = [x_t^{\text{c}}, x_{t+1}^{\text{c}}, \ldots, \hat{y}_{t+w-2}, \hat{y}_{t+w-1}] \tag{21}$$

The model then outputs:

$$\hat{\mathbf{Y}}(t) = [\hat{y}_{t+w}(t), \hat{y}_{t+w+1}(t), \ldots, \hat{y}_{t+w+n-1}(t)] \tag{22}$$

Final predictions for time step $t + w$ are computed by majority voting over predictions from the last v steps:

$$\hat{y}_{t+w} = \text{mode}\left(\hat{y}_{t+w}(t-v+1), \ldots, \hat{y}_{t+w}(t)\right) \tag{23}$$

where $mode$ returns the most frequently predicted class label. In the case of a tie, that is, when two or more classes have the same frequency, the most recent prediction is selected as the final label.

4.4.3 Mode 3: Transition-Constrained Auto-Regressive Inference

Mode 3, proposed in this study for the first time, introduces a novel transition-aware post-inference strategy that integrates domain-specific class transition rules into the auto-regressive forecasting process. Mode 3 extends the inference process of Mode 1 by enforcing class transition constraints through a dynamic masking mechanism.

Based on domain knowledge of the temporal dynamics observed in the Exathlon anomaly types, transition rules are defined as priors to enforce semantically consistent and temporally plausible system behavior. As illustrated in Figure 3, the normal class is allowed to persist or transition to any anomaly class, reflecting the potential onset of anomalous behavior from a previously stable system state. Conversely, each anomaly class may either persist or transition back to the normal state, modeling typical fault recovery. These constraints are grounded in the operational characteristics of each anomaly type, as described in Table 5. Importantly, the Exathlon benchmark defines the end of each anomaly's Extended Effect Interval (EEI) as the point at which the system returns to normal. Therefore, direct transitions between different anomaly types are disallowed, as they would violate this recovery requirement and obscure root-cause attribution.

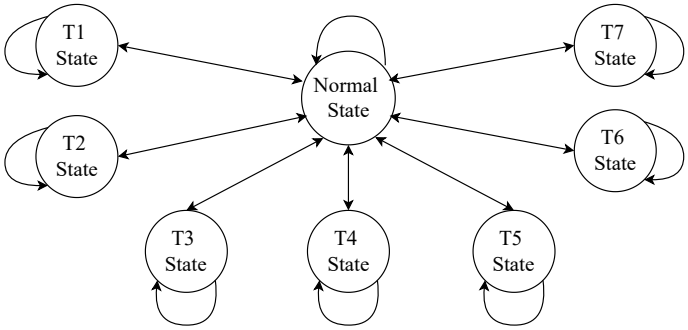


Figure 3: State transition diagram used for masked inference.

During inference, the transition structure is applied after the model produces the predicted class probability vector at each time step t. Specifically, the class probability vector corresponding to the record at time step $t+w$ in the output sequence $\hat{\mathbf{Y}}(t)$, as defined in Equation 22, is denoted by $\mathbf{p}_{t+w}(t) \in \mathbb{R}^c$.

To enforce valid transitions, a binary mask $\mathbf{m}_{t+w} \in \{0,1\}^c$ is constructed based on the previously predicted class $\hat{y}_{t+w-1}$. Each element of the mask is defined as:

$$\mathbf{m}_{t+w}[i] = \begin{cases} 1, & \text{if transition from } \hat{y}_{t+w-1} \text{ to class } i \text{ is allowed} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

The predicted probabilities are then masked using element-wise multiplication:

$$\tilde{\mathbf{p}}_{t+w}(t) = \mathbf{p}_{t+w}(t) \odot \mathbf{m}_{t+w}[i], \quad i = 1, 2, ..., c \tag{25}$$

Table 5: Class transition descriptions.

State	Transition Behavior and Justification
Normal	Unrestricted. Transitions to and from all states are permitted, as this state serves as both the origin and recovery point of system behavior. EEIs for all anomaly types are defined to end upon returning to this state.
T1 – Bursty Input	Reflects gradual system overload due to external load spikes. The state may persist or return to normal once elevated metrics (e.g., memory usage, scheduling delays) stabilize. Transitions to other anomaly states are not allowed, as EEI closure requires normalization.
T2 – Bursty Input Until Crash	Models system-wide failure caused by sustained overload. Only transitions back to the normal state are allowed post-recovery, as all operations halt during the crash.
T4 – CPU Contention	Captures performance degradation due to competing processes. This state may persist or resolve, returning to normal operation once contention subsides. Although, in rare cases, CPU contention may escalate into infrastructure-level failures (e.g., driver crashes), direct transitions to such anomalies are not permitted, allowing root-cause clarity and adherence to the EEI definition.
T5 – Driver Failure	Represents a critical system fault that halts application execution. Recovery involves restarting the driver and all executors. Transitions to other anomaly types are not allowed; the system must return to the normal state before entering any new anomaly phase.
T6 – Executor Failure	Similar to driver failure but limited to a subset of processing nodes. Execution is partially disrupted, and automatic recovery is triggered by the system. As with other failure states, the EEI ends upon resumption of normal processing, so transitions to other anomaly types are disallowed.
T7 – Unknown Anomalies	Covers manually labeled anomalies with irregular signatures. Transitions are limited to and from the normal state to reduce noise and enhance post-inference stability, consistent with other anomaly types.

Finally, the predicted class $\hat{y}_{t+w}(t)$ is selected as the class with the highest probability in $\tilde{\mathbf{p}}_{t+w}(t)$, ensuring that the output sequence adheres to the predefined transition rules and avoids invalid or abrupt class switches.

Furthermore, within the same output sequence $\hat{\mathbf{Y}}(t)$, as defined in Equation 22, each predicted class $\hat{y}_{t+w+i}(t)$ for $i = 1, \ldots, n-1$ conditions its transition on the previously predicted class $\hat{y}_{t+w+i-1}(t)$, so that masking is applied recursively across the prediction horizon. This masking enforces valid transitions between successive inference windows and within a single model output, promoting consistency and stability in the predicted class sequence.

4.4.4 Mode 4: Hybrid Auto-Regressive Inference

Mode 4 uses the strengths of Modes 2 and 3 by combining transition-based masking with majority voting within an auto-regressive forecasting framework. At each inference step, the model produces

overlapping predictions, to which transition constraints are applied via masking to enforce valid class transitions. The resulting masked probability vectors are then aggregated using majority voting to determine the final class labels.

CHAPTER 5

EXPERIMENTS AND RESULTS

This chapter presents the baseline models used for evaluation, followed by the experimental setup, performance metrics, and a comparative analysis of the results obtained on the Exathlon benchmark dataset, previously described in Chapter 3. The evaluation framework is designed to assess both binary and multiclass anomaly detection performance across a diverse set of anomaly types and temporal patterns. The results are analyzed with respect to detection accuracy and temporal consistency, and are benchmarked against established models based on forecasting and reconstruction provided by the Exathlon framework [16].

5.1 Baseline Models

The following procedures describe the baseline modeling strategy as defined in the original Exathlon benchmark framework. This approach serves as the basis for the comparison in the present study.

The anomaly detection phase in the benchmark involves learning normal system behavior from undisturbed traces to detect deviations indicative of anomalies. Specifically, the training data, comprising undisturbed traces, is segmented using stratified sampling with a sliding window approach to generate consecutive time-series sequences. Each trace is divided into three non-overlapping equal temporal segments, from which 70% of the samples are randomly selected for training (D_0), 15% for validation (D_1), and 15% for internal testing (D_2). Deep learning models are trained on D_0, with early stopping applied based on validation performance on D_1. Hyperparameter tuning is performed using Bayesian optimization to select configurations that maximize model performance in D_2. The optimized hyperparameter settings for each baseline model are detailed in the Appendix B.

The benchmark includes two main categories of anomaly detection models, namely forecasting-based and reconstruction-based approaches.
LSTM (Long Short-Term Memory) [17]: A forecasting model that predicts future time steps based on historical sequences. The outlier score for each point is calculated as the relative error between its true value and the prediction of the model. Larger forecast errors indicate a higher likelihood of anomalies as they reflect deviations from the expected temporal patterns.

AE (Autoencoder) [18]: A reconstruction-based model that encodes input data in a lower-dimensional latent representation and subsequently reconstructs it to match the original input as closely as possible. The outlier score for each window is computed using the mean squared reconstruction error (MSE).

To obtain a per-record anomaly score, the average of the MSE values from all windows in which the record appears is calculated, thereby capturing deviations from the model's learned normal behavior.

BiGAN (Bidirectional Generative Adversarial Network) [19]: A reconstruction-based generative model composed of an encoder, a generator, and a discriminator. The encoder maps the real input data X_{real} to a latent representation Z_{real}, while the generator maps random latent vectors Z_{fake} to synthetic inputs X_{fake} that resemble real data. The discriminator receives both real and synthetic data-latent pairs and attempts to distinguish between them. It operates in two input streams: the real path, which processes pairs $(X_{\text{real}}, Z_{\text{real}})$, and the fake path, which processes generated pairs $(X_{\text{fake}}, Z_{\text{fake}})$. The encoder and the generator are jointly trained to fool the discriminator, which encourages the model to produce realistic reconstructions. The discriminator, in turn, learns to differentiate real pairs from generated pairs, enabling effective anomaly scoring based on reconstruction quality and feature level consistency. Similarly to the AE model, anomaly scores in the BiGAN framework are first computed at the window level and then aggregated to the record level by averaging the anomaly scores of all windows containing the respective record. The anomaly score for each window is defined as the average of two components: the MSE computed by the encoder-generator pair, where the output of the encoder is used as input to the generator, and the feature loss derived from the discriminator. The feature loss evaluates the similarity between internal representations of real and generated samples as learned by the discriminator. It is calculated as the sum of the absolute differences between the real and fake feature vectors. Higher feature loss values indicate lower similarity, signaling a greater likelihood of anomalous behavior.

After anomaly scores are derived, the next step involves converting each score into a binary outcome. Specifically, scores equal to or exceeding a defined threshold are labeled as anomalies (1), while those below the threshold are labeled normal (0). The Exathlon benchmark adopts an unsupervised threshold selection strategy applied to the D_2 set. This approach combines a measure of central tendency (mean, median, or third quartile) with a measure of dispersion (standard deviation, median absolute deviation, or interquartile range) to establish a robust thresholding mechanism [20].The procedure is implemented in two iterative stages. First, an initial threshold threshold_i is calculated based on the distribution of available outlier scores. In the subsequent iteration, outlier scores exceeding the value removal factor $\times$ threshold_i are removed from the distribution. This process continues for a predefined number of iterations, and the final threshold from the last iteration is retained.

The following three thresholding strategies are implemented:
STD (Standard Deviation) Selector: $\text{threshold} = \mu + \alpha \times \sigma$,
where μ is the mean, σ is the standard deviation, and α is a thresholding factor.

MAD (Median Absolute Deviation) Selector: $\text{threshold} = \text{median} + \alpha \cdot \text{MAD}$,
where $\text{MAD} = 1.4826 \times \text{median}(|X - \text{median}(X)|)$

IQR (Interquartile Range): $\text{threshold} = Q3 + \alpha \cdot \text{IQR}$,
where $\text{IQR} = Q3 - Q1$, and $Q1$ and $Q3$ are the first and third quartiles of the score distribution.

5.2 Experimental Setup

This section outlines the experimental framework used to evaluate the proposed anomaly detection methodology in the dataset described in Chapter 3. The experiments are designed to ensure repro-

ducibility and fairness, adopting a supervised setup with a three-fold cross-validation strategy to allow direct comparison with existing unsupervised baselines. Subsequent subsections describe the data partitioning approach and the hyperparameter tuning procedure for the proposed transformer-based model.

5.2.1 Data Partition with Cross Validation

In the supervised setup, a three-fold cross-validation strategy is employed to allow fair and consistent comparison with the unsupervised baselines established in the Exathlon benchmark. The dataset is partitioned into three distinct folds to facilitate a comprehensive evaluation on multiple experimental runs. As illustrated in Figure 4, each fold is sequentially assigned to serve as the test set, while the remaining two folds are used for training and validation. This rotation ensures that all disturbed traces are eventually included in the test phase, maintaining full evaluation coverage, and adhering to the evaluation protocol defined for unsupervised benchmarking.

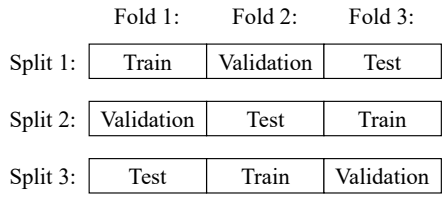


Figure 4: Cross validation splits.

To ensure strict separation between training and evaluation data, cross-validation splits are constructed to prevent any form of data leakage and to preserve temporal coherence. Each trace is assigned exclusively to a single fold, ensuring that there is no overlap of traces across folds. The assignment process begins with disturbed traces containing minority anomaly types, which are distributed first to guarantee their inclusion and balance across folds. Subsequently, the remaining traces are allocated in a manner that aims to balance both the number of traces and the number of anomaly instances per type across the three folds. The resulting distribution of anomaly types across folds is summarized in Table 6, with detailed trace-level allocations provided in Appendix C.

Undisturbed traces are excluded from the training and validation phases, as disturbed traces already contain a sufficient number of normal instances. This decision is empirically supported by a low Earth Mover's Distance (EMD = 0.26) [50] observed between normal instances originating from disturbed and undisturbed traces, indicating a high degree of distributional similarity. In contrast, comparisons between undisturbed normal data and each anomaly type show varying degrees of distributional divergence, with EMD values ranging approximately from 0.44 to 3.11. EMD quantifies the minimum effort required to transform one probability distribution into another and is particularly well suited for capturing differences in both mean and spread. This makes it more effective than Jensen-Shannon Divergence (JSD) [51], which mainly measures probabilistic similarity and is less sensitive to changes in distribution shape and dispersion characteristics that are evident in the Exathlon dataset.

Table 6: Summary of anomaly distribution in disturbed traces across cross-validation folds.

Anomaly Type	**# Traces**	**# Instances**
All	**Total:** 34 **Fold 1:** 12 (36%) **Fold 2:** 11 (32%) **Fold 3:** 11 (32%)	**Total:** Normal: 51794 Anomalous: 7450 **Fold 1:** Normal: 14476 (28%) Anomalous: 2536 (34%) **Fold 2:** Normal: 19971 (39%) Anomalous: 2359 (32%) **Fold 3:** Normal: 17347 (33%) Anomalous: 2555 (34%)
T1	**Total:** 6 **Fold 1:** 2 (33%) **Fold 2:** 2 (33%) **Fold 3:** 2 (33%)	**Total:** 2573 **Fold 1:** 593 (23%) **Fold 2:** 1286 (50%) **Fold 3:** 694 (27%)
T2	**Total:** 7 **Fold 1:** 3 (43%) **Fold 2:** 2 (29%) **Fold 3:** 2 (29%)	**Total:** 1001 **Fold 1:** 398 (40%) **Fold 2:** 221 (22%) **Fold 3:** 382 (38%)
T3	**Total:** 4 **Fold 1:** 1 (25%) **Fold 2:** 1 (25%) **Fold 3:** 2 (50%)	**Total:** 1031 **Fold 1:** 253 (25%) **Fold 2:** 257 (25%) **Fold 3:** 521 (50%)
T4	**Total:** 6 **Fold 1:** 2 (33%) **Fold 2:** 2 (33%) **Fold 3:** 2 (33%)	**Total:** 1626 **Fold 1:** 439 (27%) **Fold 2:** 429 (26%) **Fold 3:** 758 (47%)
T5	**Total:** 8 **Fold 1:** 3 (38%) **Fold 2:** 3 (38%) **Fold 3:** 2 (25%)	**Total:** 44 **Fold 1:** 19 (43%) **Fold 2:** 15 (34%) **Fold 3:** 10 (23%)
T6	**Total:** 8 **Fold 1:** 2 (25%) **Fold 2:** 2 (25%) **Fold 3:** 2 (25%)	**Total:** 918 **Fold 1:** 687 (75%) **Fold 2:** 89 (10%) **Fold 3:** 142 (15%)
T7	**Total:** 11 **Fold 1:** 4 (36%) **Fold 2:** 4 (36%) **Fold 3:** 3 (27%)	**Total:** 257 **Fold 1:** 147 (57%) **Fold 2:** 62 (24%) **Fold 3:** 48 (19%)

5.2.2 Hyperparameter Optimization

Several key parameters are fixed across all experiments to facilitate a focused feasibility analysis, with comprehensive optimization deferred to future work. A window size of $w = 40$ is adopted, consistent with the optimized configuration used in the Exathlon benchmark. For post-inference Modes 2 and 4, a majority voting window of $v = 3$ is employed. A prediction horizon of $n = 10$ is uniformly applied across all models.

Other hyperparameters are optimized independently for each cross-validation split using Bayesian optimization on the Weights & Biases platform[1]. The optimization objective is to minimize cross-entropy loss, using a learning rate of 10^{-5} with an early stopping criterion (maximum epochs = 50, patience = 1). The complete hyperparameter search space and the best performing configurations for each split are summarized in Table 7.

Table 7: Hyperparameter search space and best configurations per split.

Parameter	**Search Space**	**Split 1**	**Split 2**	**Split 3**
Batch Size	{16, 32, 64}	64	64	64
Dataset Stride	{1, 2}	1	1	2
Embedding Size	{64, 128, 256}	128	128	128
Number of Heads	{4, 8}	8	4	4
Number of Encoder/Decoder Layers	{1, 2, 3}	2	2	3
Dropout Rate	{0, 0.1, 0.3, 0.5}	0.1	0.5	0.3

5.3 Performance Metrics

Two evaluation strategies are employed to assess detection performance: binary evaluation and multiclass evaluation. Both strategies are aligned with the anomaly detection (AD) levels defined in the Exathlon benchmark: AD-1 (Anomaly Existence), AD-2 (Range Detection), AD-3 (Early Detection), and AD-4 (Exactly-Once Detection). These levels impose progressively stricter criteria for successful detection, as detailed in Section 2.3.

In the binary evaluation setting, anomaly detection is framed as a type-agnostic classification task. The goal is to determine whether a time interval exhibits anomalous behavior, without regard to the specific anomaly type. As summarized in Table 8, micro precision and recall are used to evaluate overall detection quality across all types, while type-wise recall is computed to assess how well anomalies of each type are detected. Type-wise precision is not applicable in this setting, as predictions are unlabeled with respect to anomaly class. Consequently, type-wise F1 scores are also not computed.

The multiclass evaluation setting adopts a more fine-grained approach by requiring that each predicted anomaly interval be assigned a specific type. A prediction contributes to the performance score only if the correct type is identified. If an anomaly is detected but misclassified in terms of type, it is excluded from the recall of the true class and simultaneously penalizes the precision of the incorrectly predicted

[1] All model configurations are available on W&B: https://wandb.ai/ai-studies/TS-Anomaly-Detection/sweeps

class. This enables a class-sensitive evaluation of both detection and classification performance. As shown in Table 8, all evaluation metrics, including type-wise precision and type-wise F1 score, are applicable in the multiclass setting. This provides a comprehensive view of the model behavior across different types of anomalies.

For both binary and multiclass settings, a weighted anomaly detection score is computed as a linear combination of the four AD levels. Higher weights are assigned to more fundamental detection capabilities, reflecting the greater importance of simply detecting the presence of anomalies by accurately localizing them. The weighting scheme is: AD-1 = 0.4, AD-2 = 0.3, AD-3 = 0.2, and AD-4 = 0.1.

In addition, recall for the normal (non-anomalous) class is computed on undisturbed traces to assess the model's ability to correctly identify healthy system behavior. This ensures that the evaluation captures both the sensitivity to anomalies and the robustness to false positives.

Table 8: Computation and usage of evaluation metrics in binary and multiclass settings.

Metric	Computation	Binary Evaluation	Multiclass Evaluation
Micro Recall	Equation 2, with m as the number of all ground-truth anomaly ranges in disturbed traces.	Used	Used
Type-wise Recall	Equation 2, restricted to each anomaly type with m as the number of true ranges for that type.	Used	Used
Micro Precision	Equation 3, with k as the total number of predicted ranges in disturbed traces.	Used	Used
Type-wise Precision	Equation 3, restricted to predictions labeled as a specific type, with k as their count.	Not Applicable	Used
Micro F1 Score	Harmonic mean of global precision and global recall.	Used	Used
Type-wise F1 Score	Harmonic mean of type-wise precision and recall.	Not Applicable	Used

5.4 Results and Discussion

The proposed TIM models (Transformer with Inference Modes 1 to 4) are evaluated against three baseline methods, LSTM, AE, and BiGAN in the binary anomaly detection setting using disturbed traces only. The results for the TIM models are obtained from the test phase of each split in the cross-validation setup.

For the LSTM, AE, and BiGAN baselines, performance scores are derived from the Exathlon benchmark. These results represent the median performance across 24 thresholding configurations, including variations in thresholding method, thresholding factor, and number of iterations, while excluding the unknown anomaly type from evaluation. The specific parameter grid used for thresholding includes:

thresholding method = {STD, MAD, IQR}, thresholding factor ={1.5, 2.0, 2.5, 3.0}, number of iterations = {1, 2} and removal factor = {1.0}

The results for binary and multiclass classification settings in the four levels of anomaly detection: AD-1 (Existence), AD-2 (Range Detection), AD-3 (Early Detection) and AD-4 (Exactly-Once Detection) are summarized in Table 9 and Table 10, respectively. Appendix D contains the complete set of prediction visualizations for disturbed traces across all four inference modes.

AD-1 – Anomaly Existence Evaluation. This least restrictive evaluation requires identifying any anomaly presence within the detection window. All models perform best at this level. LSTM dominates in the binary setting with high recall (0.96) and F1 (0.77), while BiGAN shows high precision (0.90) but suffers from extremely low recall (0.19), indicating a conservative prediction strategy that frequently misses actual anomalies. The TIM models maintain a more balanced precision-recall trade-off, offering consistent results across all variants. In the multiclass setting, which requires accurate identification of anomaly types, all models experience performance degradation. Among TIM models, TIM-2 achieves the highest micro F1 score (0.47), aided by its voting-based inference that consolidates predictions into temporally and semantically stable sequences.

AD-2 – Range Detection Evaluation. This level evaluates whether the predicted anomaly intervals align with the true temporal ranges. In the binary setup, all TIM models outperform LSTM, AE, and BiGAN. LSTM shows a noticeable recall drop compared to AD-1, indicating difficulty in identifying the full span of anomalies. TIM-2 yields the highest F1 score (0.68), thanks to its post-inference voting scheme that helps stabilize noisy predictions. In the multiclass configuration, TIM-2 again leads (F1 = 0.41), suggesting that the voting mechanism supports range coherence even under classification constraints. TIM-4 achieves stronger precision, attributed to its hybrid structure, which combines voting and transition constraints. However, the added rigidity slightly reduces the recall, revealing a precision-recall trade-off.

AD-3 – Early Detection Evaluation. This level prioritizes early identification of anomalies. In binary evaluation, TIM-4 achieves the highest F1 score (0.63), balancing early detection and prediction consistency. TIM-3 achieves the highest recall (0.64), suggesting that transition-constrained inference helps identify anomaly onsets more quickly. In the multiclass case, TIM-4 again performs best (F1 = 0.39; precision = 0.47), confirming that its hybrid design effectively enforces early and type-consistent detection.

AD-4 – Exactly-Once Detection Evaluation. This most restrictive criterion requires each anomaly to be reported exactly once. In binary mode, TIM-4 attains the highest F1 score (0.59), demonstrating the strength to combine majority voting and transition masking to avoid redundant alerts. In the multiclass setting, it again leads in both F1 (0.39) and precision (0.47), indicating its effectiveness at maintaining class stability and limiting over-detection.

Weighted AD Level Evaluation. This composite score integrates the four AD levels with predefined weights to assess general robustness. In the binary setting, TIM-4 produces the highest F1 score (0.67), while TIM-3 offers the best recall (0.72). In the multiclass context, TIM-2 records the highest F1 (0.43)

and recall (0.44), while TIM-4 again delivers the strongest precision (0.47). These results confirm the efficacy of post-inference strategies.

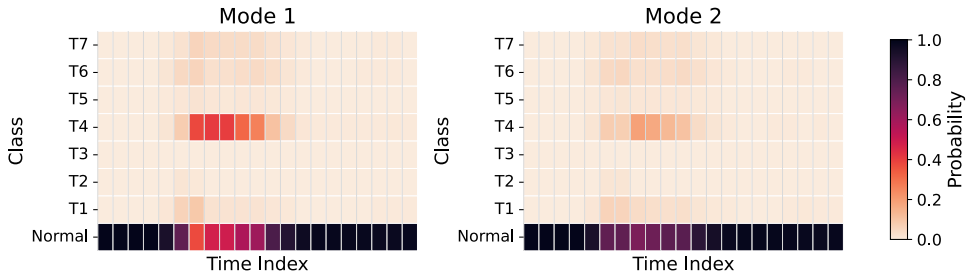
Table 9: Binary AD level evaluation results for disturbed traces. P: Precision, R: Recall

Model	Micro F1	Micro P	Micro R	T1 R	T2 R	T3 R	T4 R	T5 R	T6 R	T7 R
AD-1 Level										
LSTM	**0.77**	0.67	**0.96**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.67	–
AE	0.59	0.54	0.76	**1.00**	0.88	**1.00**	0.57	**1.00**	0.31	–
BiGAN	0.28	**0.90**	0.19	0.59	0.00	0.00	0.10	0.17	0.06	–
TIM-1	0.71	0.63	0.81	**1.00**	**1.00**	0.88	0.62	0.78	**0.70**	**0.70**
TIM-2	0.72	0.66	0.79	**1.00**	**1.00**	0.94	0.50	0.78	**0.70**	**0.70**
TIM-3	0.67	0.58	0.80	**1.00**	**1.00**	0.81	0.62	0.78	**0.70**	**0.70**
TIM-4	0.72	0.65	0.79	**1.00**	**1.00**	0.94	0.50	0.78	**0.70**	**0.70**
AD-2 Level										
LSTM	0.38	0.67	0.29	0.42	0.11	0.55	0.16	0.60	0.10	–
AE	0.52	0.54	0.60	**0.97**	0.15	0.67	0.40	**1.00**	0.15	–
BiGAN	0.17	**0.90**	0.10	0.30	0.00	0.00	0.06	0.17	0.00	–
TIM-1	0.67	0.63	**0.72**	0.96	**0.98**	0.71	**0.53**	0.64	**0.57**	0.55
TIM-2	**0.68**	0.66	0.69	0.96	0.96	**0.74**	0.40	0.60	0.55	**0.56**
TIM-3	0.64	0.58	0.70	0.96	0.89	0.68	0.51	0.64	0.57	0.54
TIM-4	0.66	0.65	0.67	0.96	0.89	0.72	0.39	0.60	0.55	0.54
AD-3 Level										
LSTM	0.29	0.67	0.20	0.27	0.04	0.37	0.10	0.58	0.08	–
AE	0.51	0.54	0.57	**0.96**	0.06	0.62	0.37	**1.00**	0.14	–
BiGAN	0.14	**0.90**	0.08	0.23	0.00	0.00	0.05	0.17	0.00	–
TIM-1	0.63	0.62	0.63	0.85	**0.84**	0.65	0.44	0.57	0.47	0.48
TIM-2	0.62	0.66	0.58	0.81	0.82	**0.66**	0.31	0.56	0.45	0.45
TIM-3	0.61	0.58	**0.64**	0.92	0.82	0.62	**0.44**	0.57	**0.50**	**0.49**
TIM-4	**0.63**	0.65	0.61	0.91	0.79	0.65	0.32	0.56	0.47	0.47
AD-4 Level										
LSTM	0.13	0.67	0.08	0.00	0.00	0.07	0.00	0.58	0.06	–
AE	0.49	0.52	0.56	**0.94**	0.06	0.58	0.37	**1.00**	0.14	–
BiGAN	0.14	**0.86**	0.08	0.23	0.00	0.00	0.05	0.17	0.00	–
TIM-1	0.52	0.61	0.45	0.58	0.38	0.47	0.36	0.57	**0.40**	0.32
TIM-2	0.52	0.64	0.43	0.56	0.37	0.59	0.25	0.56	0.38	0.29
TIM-3	0.57	0.55	**0.60**	0.87	**0.79**	0.57	**0.40**	0.57	**0.40**	**0.49**
TIM-4	**0.59**	0.62	0.57	0.86	0.76	**0.62**	0.29	0.56	0.38	0.42
Weighted AD Level										
LSTM	0.49	0.67	0.52	0.58	0.44	0.65	0.47	0.75	0.32	–
AE	0.54	0.54	0.65	**0.98**	0.42	0.78	0.46	**1.00**	0.21	–
BiGAN	0.21	**0.90**	0.13	0.40	0.00	0.00	0.07	0.17	0.02	–
TIM-1	0.66	0.63	0.71	0.92	0.90	0.74	0.53	0.67	0.59	0.57
TIM-2	0.67	0.66	0.68	0.91	0.89	**0.79**	0.41	0.66	0.57	0.57
TIM-3	0.64	0.58	**0.72**	0.96	**0.91**	0.71	**0.53**	0.67	**0.59**	**0.59**
TIM-4	**0.67**	0.65	0.70	0.96	0.90	0.78	0.41	0.66	0.58	0.58

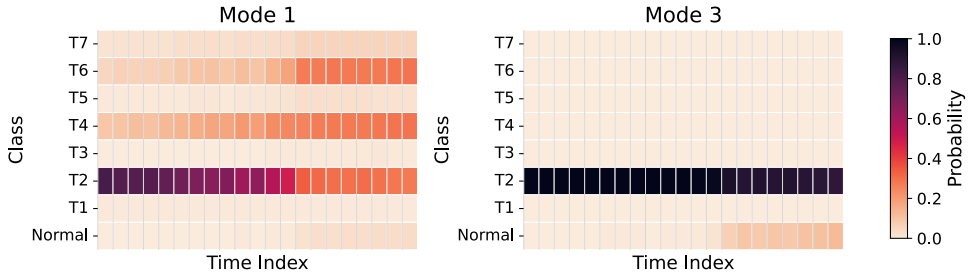
Table 10: Multiclass AD level evaluation results for disturbed traces. P: Precision, R: Recall

Model	Micro			T1			T2			T3			T4			T5			T6			T7		
	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
AD-1 Level																								
TIM-1	0.44	0.39	0.51	0.69	0.53	0.97	0.43	0.29	**0.86**	0.67	0.82	**0.56**	0.11	0.07	0.31	0.00	0.00	0.00	0.10	0.11	**0.10**	**0.41**	**0.67**	**0.30**
TIM-2	**0.47**	0.42	**0.53**	**0.73**	**0.58**	**1.00**	**0.43**	0.29	**0.86**	**0.72**	**1.00**	**0.56**	0.17	0.11	0.35	0.00	0.00	0.00	0.11	0.12	**0.10**	**0.41**	**0.67**	**0.30**
TIM-3	0.40	0.39	0.40	0.54	0.47	0.62	0.38	**0.29**	0.57	0.62	0.80	0.50	0.20	0.14	0.35	0.00	0.00	0.00	**0.13**	**0.19**	**0.10**	0.40	0.60	**0.30**
TIM-4	0.45	**0.47**	0.43	0.60	0.55	0.66	0.36	0.27	0.57	0.69	0.91	**0.56**	**0.29**	**0.24**	**0.38**	0.00	0.00	0.00	0.12	0.15	**0.10**	0.40	0.60	**0.30**
AD-2 Level																								
TIM-1	0.40	0.39	**0.41**	0.64	0.53	**0.79**	**0.41**	0.29	**0.71**	0.59	0.82	0.46	0.11	0.07	0.26	0.00	0.00	0.00	0.06	0.11	**0.04**	0.22	**0.67**	0.13
TIM-2	**0.41**	0.42	0.41	**0.67**	**0.58**	0.78	0.41	0.29	0.71	**0.63**	**1.00**	0.46	0.16	0.11	0.26	0.00	0.00	0.00	**0.06**	0.12	0.04	0.22	**0.67**	0.13
TIM-3	0.37	0.39	0.35	0.53	0.47	0.59	0.38	**0.29**	0.56	0.59	0.80	0.46	0.19	0.14	0.29	0.00	0.00	0.00	0.01	**0.19**	0.01	0.27	0.60	0.18
TIM-4	0.41	**0.47**	0.36	0.58	0.55	0.61	0.36	0.27	0.56	0.61	0.91	**0.46**	**0.26**	**0.24**	**0.29**	0.00	0.00	0.00	0.01	0.15	0.01	**0.29**	0.60	**0.19**
AD-3 Level																								
TIM-1	0.38	0.39	**0.37**	0.61	0.53	**0.72**	**0.39**	0.29	**0.63**	0.56	0.82	0.42	0.11	0.07	0.22	0.00	0.00	0.00	0.02	0.11	**0.01**	0.20	**0.67**	0.12
TIM-2	0.39	0.42	0.35	**0.63**	**0.58**	0.69	0.39	0.29	0.62	**0.58**	**1.00**	0.41	0.15	0.11	0.22	0.00	0.00	0.00	**0.02**	0.12	0.01	0.20	**0.67**	0.12
TIM-3	0.36	0.39	0.33	0.52	0.47	0.57	0.38	**0.29**	0.55	0.55	0.80	**0.42**	0.18	0.14	**0.25**	0.00	0.00	0.00	0.00	**0.19**	0.00	0.27	0.60	0.18
TIM-4	**0.39**	**0.47**	0.33	0.56	0.55	0.57	0.36	0.27	0.54	0.56	0.91	0.41	**0.24**	**0.24**	0.25	0.00	0.00	0.00	0.00	0.15	0.00	**0.29**	0.60	**0.19**
AD-4 Level																								
TIM-1	0.37	0.39	**0.36**	0.61	0.53	**0.70**	**0.39**	0.29	**0.63**	0.56	0.82	0.42	0.10	0.06	0.22	0.00	0.00	0.00	0.02	0.11	**0.01**	0.13	**0.67**	0.08
TIM-2	0.38	0.42	0.35	**0.63**	**0.58**	0.68	0.39	0.29	0.62	**0.58**	**1.00**	0.41	0.14	0.10	0.22	0.00	0.00	0.00	**0.02**	0.12	0.01	0.14	**0.67**	0.08
TIM-3	0.36	0.39	0.33	0.52	0.47	0.57	0.38	**0.29**	0.55	0.55	0.80	**0.42**	0.17	0.13	**0.25**	0.00	0.00	0.00	0.00	**0.19**	0.00	0.27	0.60	0.18
TIM-4	**0.39**	**0.47**	0.33	0.56	0.55	0.57	0.36	0.27	0.54	0.56	0.91	0.41	**0.23**	**0.21**	0.25	0.00	0.00	0.00	0.00	0.15	0.00	**0.29**	0.60	**0.19**
Weighted AD Level																								
TIM-1	0.41	0.39	0.44	0.65	0.53	0.84	**0.41**	0.29	**0.74**	0.61	0.82	**0.49**	0.11	0.07	0.27	0.00	0.00	0.00	0.07	0.11	**0.06**	0.28	**0.67**	0.19
TIM-2	**0.43**	0.42	**0.44**	**0.68**	**0.58**	**0.84**	0.41	0.29	0.74	**0.65**	**1.00**	0.49	0.16	0.11	0.28	0.00	0.00	0.00	**0.07**	0.12	0.06	0.29	**0.67**	0.19
TIM-3	0.38	0.39	0.37	0.53	0.47	0.60	0.38	**0.29**	0.56	0.59	0.80	0.47	0.19	0.14	0.30	0.00	0.00	0.00	0.06	**0.19**	0.04	0.32	0.60	0.23
TIM-4	0.42	**0.47**	0.38	0.58	0.55	0.62	0.36	0.27	0.56	0.63	0.91	0.49	**0.27**	**0.24**	**0.31**	0.00	0.00	0.00	0.05	0.15	0.04	**0.33**	0.60	**0.23**

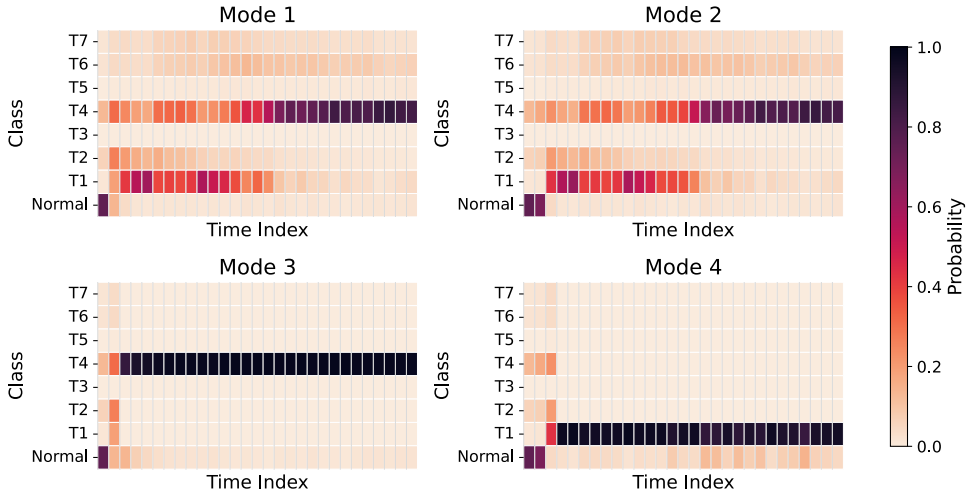
Probability Visualizations. To better understand the behavioral dynamics of the proposed TIM models under various inference modes, Figure 5 visualizes the class probability outputs after the four post-inference steps.



(a) Trace: 5_1_100000_64, Time Indices: 2070 - 2090, Actual Class: Normal



(b) Trace: 2_2_200000_69, Time Indices: 170 - 190, Actual Class: T2



(c) Trace: 5_1_500000_62, Time Indices: 2342 - 2370, Actual Class: T1

Figure 5: Predicted class probabilities across different inference modes for three representative traces. Trace name format: *app id_anomaly type_input rate_trace id*

As shown in Fig.1 (a), Mode 1 produces a false alarm by predicting class T4 with low confidence where the actual class is Normal. In contrast, Mode 2, which uses a majority voting mechanism, effectively consolidates predictions and prevents this false alarm. This reflects the main advantage of Mode 2: aggregating multiple predictions improves the precision of the model and leads to more confident decisions, which is also supported by the results in Table 9 and Table 10.

According to Fig.1 (b), the model fluctuates between T2, T4, and T6 under Mode 1, where the actual class is T2. However, Mode 3, which enforces temporal consistency by prohibiting abrupt transitions between anomaly states, produces a more stable prediction and stays in T2. This shows how transition rules improve prediction stability. However, such rules can also lead to errors when the model gets stuck in the wrong anomaly class, contributing to the lower precision and recall of Mode 3 in the multiclass setting.

In the case of Fig.1 (c), where the actual class is T2, the model under Mode 1 initially transitions from the Normal class to T4, then briefly to T1, and returns to T4, exhibiting unstable behavior. Mode 2 does not significantly improve the prediction in this case, as the model continues to fluctuate. Mode 3 introduces temporal consistency, but causes the model to become stuck in the incorrect class T4. On the other hand, when Mode 4 is applied, combining both majority voting and transition rules, the model can correctly identify the class as T2. This highlights the effectiveness of a hybrid inference strategy in resolving noisy prediction sequences.

Normal Behavior Evaluation. To assess the model's ability to correctly identify healthy system behavior, recall scores for the Normal class are extracted from undisturbed trace predictions across the three cross-validation splits. Table 11 reports the recall values per inference mode (Modes 1–4), calculated as the ratio of correctly predicted Normal instances to the total actual Normal instances. These results suggest that all inference modes perform reasonably well in preserving normal behavior detection, with Mode 2 achieving the highest average recall (0.78). This supports the effectiveness of majority voting in reducing false positives in normal segments.

Table 11: Recall scores for the normal class on undisturbed traces across cross-validation splits.

Model	**Split 1 - Recall**	**Split 2 - Recall**	**Split 3 - Recall**	**Average Recall**
TIM-1	0.92	0.58	0.78	0.76
TIM-2	0.96	**0.73**	0.79	**0.83**
TIM-3	0.93	0.61	**0.79**	0.78
TIM-4	**0.97**	0.61	0.75	0.78

Summary of Results. Table 9 demonstrates that the proposed TIM models consistently achieve a more balanced trade-off between precision and recall compared to the baseline models in binary evaluation settings. In Table 10, the benefits of post-inference strategies are evident in improving prediction robustness. However, configurations with stricter constraints (e.g., transition enforcement in TIM-3) occasionally result in degraded performance when the model becomes locked into incorrect anomaly classes, thereby lowering overall precision and recall.

As the anomaly detection (AD) level increases—from AD-1 (existence) to AD-4 (exactly-once detection)—a general decline in performance is observed across all models, which aligns with the increasing stringency of evaluation criteria. This degradation is more pronounced in the multiclass setting, where the requirement to correctly classify the anomaly type introduces additional complexity. Despite this, the TIM models consistently outperform the baseline methods at each AD level. Notably, their perfor-

mance advantage becomes increasingly evident at higher AD levels, where the benefits of the proposed post-inference strategies are more critical to maintaining detection stability and precision.

Across all anomaly detection (AD) levels, the TIM models demonstrate the strongest performance on the most prevalent anomaly types, namely T1–T3. In particular, T3 (Stalled Input) yields high precision due to its distinctive signature: a complete halt in incoming data while system processes continue running. This signature facilitates more reliable detection and classification. In contrast, T1 (Bursty Input) and T2 (Bursty Input Until Crash) share similar input spike and resource usage patterns, leading to confusion between these types and lower classification precision.

For the rarer anomaly types, especially T5 (Driver Failure), all TIM variants perform poorly, primarily due to severe class imbalance and limited training data. Specifically, T5 anomalies are frequently misclassified as T4 (CPU Contention), as shown in Figure 16 in Appendix D. T4 anomalies are caused by aggressive CPU resource consumption, which gradually degrades the Spark application's processing capabilities. When CPU contention becomes severe, it can cause processing delays and potentially crash the driver node, which closely resembles the effects observed in T5. This progression likely contributes to the frequent misclassification between T4 and T5.

An interesting and unexpected observation is that the T7 category (Unknown Anomaly) yields better performance than some known anomaly types, such as T4, T5, and T6, despite its undefined and irregular nature. One possible explanation is that T7 anomalies are present in the same disturbed traces as T4, T5, and T6, which may result in similar patterns in the feature space. These similarities may lead to misclassifications between T4, T5, T6, and T7, as the model struggles to distinguish subtle variations among them under limited supervision. Moreover, as shown in the model prediction visualizations in Appendix D, many of the correctly predicted T7 anomalies occur early in the sequence. During inference, the model is initially guided by a window of ground-truth labels before switching to its own predictions. Because several T7 anomalies begin before this transition, the ground-truth input may help reinforce the model's initial prediction, allowing it to remain in the correct class during subsequent steps. This mechanism may partially explain the relatively strong performance observed for the T7 class.

Regarding the evaluation of normal system behavior, all TIM variants achieve recall scores above 0.75 on undisturbed traces, highlighting their capability to accurately recognize non-anomalous patterns. Among them, TIM-2 achieves the highest recall, confirming its effectiveness in filtering out false positives through voting-based post-inference.

CHAPTER 6

CONCLUSION

This chapter summarizes the key contributions of the thesis and discusses the limitations of the study. It also outlines potential directions for future research.

6.1 Contributions

This thesis investigates the design and evaluation of a novel transformer-based framework for range-based, multiclass anomaly detection in multivariate time series. The study addresses several core challenges in this domain, including modeling temporal dependencies, handling severe class imbalance, and enforcing semantic consistency in predicted anomaly sequences.

To structure the study, a set of research questions was posed in Chapter 1, each targeting a different dimension of the anomaly detection pipeline, from the model architecture to the consistency and evaluation methodology after post-inference. In this section, these research questions are revisited, and the corresponding contributions are summarized.

Research Question 1: *How can a transformer-based framework be designed to effectively detect range-based anomalies in multivariate time series under highly imbalanced and multiclass conditions?*

To address this question, the thesis presents a transformer-based encoder-decoder model for anomaly detection in multivariate time series. The proposed architecture jointly models both the multivariate time series features and the corresponding anomaly label sequence. This dual-input design enables the model to learn temporal patterns from both the observed signals and the structure and evolution of anomalies over time. These two representations are integrated within the attention mechanism, which enables the model to dynamically assign importance to different time steps based on both time series features and anomaly context. As a result, the model can capture short-term fluctuations and long-range dependencies, supporting the detection of extended anomaly ranges across multiple variables and time steps.

To support multiclass detection, the model outputs a probability distribution over all anomaly classes at each time step, allowing fine-grained, time-resolved classification. This is especially important in diagnostic contexts, where different anomaly types require distinct responses.

To address class imbalance, which is a common issue in real-world datasets, two techniques are applied: (i) an oversampling strategy during batch construction to increase the presence of minority classes, and (ii) a weighted loss function to give higher importance to errors involving underrepresented classes. Although these methods improved performance for less frequent anomaly types, the results indicate that class imbalance remains a significant challenge. This suggests the need for future

research on more advanced solutions, such as data augmentation, generative models, or class-specific learning schemes.

Research Question 2: *What post-inference strategies can be applied to improve the temporal and semantic coherence of predictions?*

To enhance prediction consistency, the thesis introduces two post-inference strategies that operate after the model's initial output. These strategies aim to reduce noise and fragmentation in the predicted label sequences and enforce meaningful transitions between anomaly types.

The first strategy is a majority voting scheme applied across overlapping n-step-ahead predictions. Each time step may receive multiple predicted labels, and the most frequently predicted label is selected as the final output. This approach reduces instability in the output sequence and improves the detection of coherent anomaly ranges.

The second strategy is a transition-aware masking mechanism, which incorporates domain knowledge about plausible transitions between anomaly classes. For example, the model may be constrained to prevent direct transitions between unrelated faults without passing through a normal state. This mechanism sets the probabilities of invalid transitions to zero, ensuring the predicted class sequences align with expected system behavior.

Together, these strategies improve both the temporal and semantic coherence of the model outputs by smoothing predictions over time and enforcing valid transitions.

Research Question 3: *How does the proposed transformer-based framework with post-inference strategies perform compared to existing baseline models under a range-based evaluation setting on a real-world benchmark dataset?*

To evaluate performance, the proposed framework was tested on the Exathlon benchmark under both binary and multiclass settings. Evaluation was conducted across several levels of detection complexity, from anomaly existence to early detection and exact-range identification.

Compared to baseline models such as LSTM, Autoencoder, and BiGAN, the transformer-based framework with post-inference techniques consistently achieved more balanced and stable performance. In the binary setting, the proposed models performed well in both recall and precision, producing fewer false positives and more coherent anomaly ranges. In the multiclass configuration, where models must also identify the correct anomaly type, performance decreased across all methods due to increased task complexity.

Among the model variants, those incorporating both majority voting and transition-aware masking delivered the most reliable performance, especially in scenarios that demand type-specific predictions. While applying transition rules alone enhances prediction stability by eliminating unrealistic class switches, it can occasionally cause the model to become stuck in an incorrect class, as seen in some evaluations. In contrast, the hybrid post-inference strategy, which combines majority voting with a more flexible use of transition constraints, helps the model to recover from such situations and improves the correctness of predictions over time.

From a computational standpoint, the proposed transformer-based model is efficient to train, especially when compared to the BiGAN baseline, which involves computationally expensive adversarial training. However, the sequential application of post-inference strategies introduces some additional overhead during inference, which may be a consideration in real-time or resource-constrained deployments.

Overall, the results confirm that the proposed framework and post-inference methods significantly improve the clarity, reliability, and diagnostic usefulness of anomaly detection outputs, especially under complex, real-world temporal conditions.

6.2 Limitations and Future Work

While this study introduces a novel and effective framework for range-based, multiclass anomaly detection in multivariate time series, several limitations remain that point toward valuable directions for future research.

One of the most significant limitations is the persistent challenge of class imbalance, particularly for rare anomaly types. Although oversampling and weighted loss functions were applied to reduce its impact, the model's performance on underrepresented classes remained suboptimal. Future work could explore advanced strategies such as class-specific data augmentation, adversarial training, or generative modeling to address this issue more effectively.

The proposed transition-aware masking mechanism relies on predefined rules about allowable class transitions. While effective in domains with well-understood system behavior (e.g., industrial systems, process monitoring), this approach may not generalize to settings where such domain knowledge is unavailable or unreliable. An important direction for future research is the development of data-driven methods to learn transition patterns automatically or adaptively from annotated sequences.

The multiclass nature of the task assumes access to detailed, type-specific anomaly labels at the range level. In practice, such annotated datasets are expensive and time-consuming to obtain. As a result, supervised training becomes less feasible at scale. Future work could investigate semi-supervised or weakly supervised approaches that reduce dependence on exhaustive labeling while still enabling type-aware anomaly detection.

Another limitation lies in the lack of publicly available datasets that support multiclass, range-based evaluation in multivariate time series. Although the Exathlon benchmark partially addresses this gap, there is still a need for larger, more diverse datasets that reflect the complexity of real-world applications. Extending existing benchmarks or generating realistic synthetic datasets could facilitate broader adoption and more rigorous evaluation of anomaly detection models.

An additional avenue for future work involves exploring the model's ability to handle unseen or unknown anomaly classes. In this study, class label 7 is reserved for such unknown or ambiguous cases, yet the current evaluation does not explicitly test performance under open-set or partially labeled conditions. Incorporating mechanisms for assigning instances to the unknown class, particularly when an anomaly pattern does not match any known category, could improve the system's robustness in practical deployments, where not all fault types are known a priori. Supporting open-set recognition and evaluating generalization to novel anomalies remain important directions for future research.

In summary, addressing these limitations would not only improve the robustness and generalizability of the proposed framework but also help advance the broader field of anomaly detection toward more practical, scalable, and domain-adaptive solutions.

REFERENCES

[1] A. Zhang, S. Deng, D. Cui, Y. Yuan, and G. Wang, "An experimental evaluation of anomaly detection in time series," *Proc. VLDB Endow.*, vol. 17, p. 483–496, Nov. 2023.

[2] Z. Zamanzadeh Darban, G. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Computing Surveys*, vol. 57, Oct. 2024. Publisher Copyright: © 2024 Copyright held by the owner/author(s).

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, July 2009.

[4] S. Sørbø and M. Ruocco, "Navigating the metric maze: a taxonomy of evaluation metrics for anomaly detection in time series," *Data Min. Knowl. Discov.*, vol. 38, p. 1027–1068, Nov. 2023.

[5] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich, "Precision and recall for time series," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[6] M. Liu, Y. Wang, H. Xu, X. Zhou, B. Li, and Y. Wang, "Smoothing point adjustment-based evaluation of time series anomaly detection," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[7] J. Paparrizos, P. Boniol, T. Palpanas, R. S. Tsay, A. Elmore, and M. J. Franklin, "Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection," *Proc. VLDB Endow.*, vol. 15, p. 2774–2787, July 2022.

[8] Y. Zhao, T. Li, X. Zhang, and C. Zhang, "Artificial intelligence-based fault detection and diagnosis methods for building energy systems: Advantages, challenges and the future," *Renewable and Sustainable Energy Reviews*, vol. 109, pp. 85–101, 2019.

[9] S. Maleki Varnosfaderani and M. Forouzanfar, "The role of ai in hospitals and clinics: Transforming healthcare in the 21st century," *Bioengineering*, vol. 11, no. 4, 2024.

[10] M. Achouch, M. Dimitrova, K. Ziane, S. Sattarpanah Karganroudi, R. Dhouib, H. Ibrahim, and M. Adda, "On predictive maintenance in industry 4.0: Overview, models, and challenges," *Applied Sciences*, vol. 12, no. 16, 2022.

[11] M. Zhao, A. Sadhu, and M. Capretz, "Multiclass anomaly detection in imbalanced structural health monitoring data using convolutional neural network," *Journal of Infrastructure Preservation and Resilience*, vol. 3, 08 2022.

[12] D. Baumgartner, H. Langseth, H. Ramampiaro, and K. Engø-Monsen, "mtads: Multivariate time series anomaly detection benchmark suites," in *2023 IEEE International Conference on Big Data (BigData)*, pp. 588–597, 2023.

[13] Y. Li, Z. Zhou, S. Deng, X. Sun, X. Xue, S. Yangui, and W. Gaaloul, "Accurate anomaly detection leveraging knowledge-enhanced gat," in *2024 IEEE International Conference on Web Services (ICWS)*, pp. 568–577, 2024.

[14] D. Deutsch, S. Upadhyay, and D. Roth, "A general-purpose algorithm for constrained sequential inference," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)* (M. Bansal and A. Villavicencio, eds.), (Hong Kong, China), pp. 482–492, Association for Computational Linguistics, Nov. 2019.

[15] M. Kuchnik, V. Smith, and G. Amvrosiadis, "Validating large language models with relm," in *Proceedings of Machine Learning and Systems* (D. Song, M. Carbin, and T. Chen, eds.), vol. 5, pp. 457–476, Curan, 2023.

[16] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, "Exathlon: a benchmark for explainable anomaly detection over time series," *Proc. VLDB Endow.*, vol. 14, p. 2613–2626, July 2021.

[17] L. Bontemps, V. L. Cao, J. McDermott, and N.-A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," in *Future Data and Security Engineering* (T. K. Dang, R. Wagner, J. Küng, N. Thoai, M. Takizawa, and E. Neuhold, eds.), (Cham), pp. 141–152, Springer International Publishing, 2016.

[18] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science (New York, N.Y.)*, vol. 313, pp. 504–7, 08 2006.

[19] T. Schlegl, P. Seeböck, S. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Information Processing in Medical Imaging*, pp. 146–157, 03 2017.

[20] J. Yang, S. Rahardja, and P. Fränti, "Outlier detection: how to threshold outlier scores?," in *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*, AIIPCC '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[21] S. K. Alabugin and A. N. Sokolov, "Applying of recurrent neural networks for industrial processes anomaly detection," in *2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pp. 0467–0470, 2021.

[22] L. Wang, Y. Lin, Y. Wu, H. Chen, F. Wang, and H. Yang, " Forecast-based Multi-aspect Framework for Multivariate Time-series Anomaly Detection ," in *2021 IEEE International Conference on Big Data (Big Data)*, (Los Alamitos, CA, USA), pp. 938–947, IEEE Computer Society, Dec. 2021.

[23] X. Yan, C. He, J. Yin, C. Zhang, and H. Li, "Voltage transformer anomaly detection using lstm based autoencoder," in *2023 5th International Conference on Data-driven Optimization of Complex Systems (DOCS)*, pp. 1–6, 2023.

[24] J. Wang, S. Shao, Y. Bai, J. Deng, and Y. Lin, "Multiscale wavelet graph autoencoder for multivariate time-series anomaly detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–11, 2023.

[25] H. Kang and P. Kang, "Transformer-based multivariate time series anomaly detection using inter-variable attention mechanism," *Know.-Based Syst.*, vol. 290, Apr. 2024.

[26] S. Tuli, G. Casale, and N. R. Jennings, "Tranad: deep transformer networks for anomaly detection in multivariate time series data," *Proc. VLDB Endow.*, vol. 15, p. 1201–1214, Feb. 2022.

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

[28] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, (New York, NY, USA), p. 2828–2837, Association for Computing Machinery, 2019.

[29] J. Miao, H. Tao, H. Xie, J. Sun, and J. Cao, "Reconstruction-based anomaly detection for multivariate time series using contrastive generative adversarial networks," *Information Processing and Management*, vol. 61, no. 1, p. 103569, 2024.

[30] L. Li, J. Yan, H. Wang, and Y. Jin, "Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 1177–1191, 2021.

[31] D. Ge, Z. Dong, Y. Cheng, and Y. Wu, "An enhanced spatio-temporal constraints network for anomaly detection in multivariate time series," *Know.-Based Syst.*, vol. 283, Jan. 2024.

[32] T. Li, M. L. Comer, E. J. Delp, S. R. Desai, J. L. Mathieson, R. H. Foster, and M. W. Chan, "Anomaly scoring for prediction-based anomaly detection in time series," in *2020 IEEE Aerospace Conference*, pp. 1–7, 2020.

[33] F. Meng, Q. Yang, Z. He, S. Yang, and W. Tang, "Guard: Multigranularity-based unsupervised anomaly detection algorithm for multivariate time series," in *2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS)*, pp. 25–30, 2022.

[34] P. Wenig and T. Papenbrock, "Series2graph++: Distributed detection of correlation anomalies in multivariate time series," in *Big Data Analytics and Knowledge Discovery: 26th International Conference, DaWaK 2024, Naples, Italy, August 26–28, 2024, Proceedings*, (Berlin, Heidelberg), p. 225–230, Springer-Verlag, 2024.

[35] X. Han, S. Absar, L. Zhang, and S. Yuan, "Root cause analysis of anomalies in multivariate time series through granger causal discovery," in *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, OpenReview.net, 2025.

[36] Y. Sun, R. S. Blum, and P. Venkitasubramaniam, "Unifying explainable anomaly detection and root cause analysis in dynamical systems," *CoRR*, vol. abs/2502.12086, 2025.

[37] Z. C. Lipton, D. C. Kale, C. Elkan, and R. C. Wetzel, "Learning to diagnose with LSTM recurrent neural networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[38] C. Wang and G. Liu, "From anomaly detection to classification with graph attention and transformer for multivariate time series," *Advanced Engineering Informatics*, vol. 60, p. 102357, 2024.

[39] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, p. 321–357, June 2002.

[40] P. Zhao, C. Luo, B. Qiao, L. Wang, S. Rajmohan, Q. Lin, and D. Zhang, "T-smote: Temporal-oriented synthetic minority oversampling technique for imbalanced time series classification," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22* (L. D. Raedt, ed.), pp. 2406–2412, International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[41] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, 2008.

[42] L. Xue and T. Zhu, "Hybrid resampling and weighted majority voting for multi-class anomaly detection on imbalanced malware and network traffic data," *Engineering Applications of Artificial Intelligence*, vol. 128, p. 107568, 2024.

[43] G. Deng, C. Han, T. Dreossi, C. Lee, and D. S. Matteson, *IB-GAN: A Unified Approach for Multivariate Time Series Classification under Class Imbalance*, pp. 217–225.

[44] X. Wang, Y. Zhang, N. Bai, Q. Yu, and Q. Wang, "Class-imbalanced time series anomaly detection method based on cost-sensitive hybrid network," *Expert Syst. Appl.*, vol. 238, Mar. 2024.

[45] P. Wenig, S. Schmidl, and T. Papenbrock, "Timeeval: a benchmarking toolkit for time series anomaly detection algorithms," *Proc. VLDB Endow.*, vol. 15, p. 3678–3681, Aug. 2022.

[46] D. Wagner, T. Michels, F. C. Schulz, A. Nair, M. Rudolph, and M. Kloft, "TimeseAD: Benchmarking deep multivariate time-series anomaly detection," *Transactions on Machine Learning Research*, vol. 4, 2023.

[47] B. Li, E. Mazur, Y. Diao, A. McGregor, and P. Shenoy, "Scalla: A platform for scalable one-pass analytics using mapreduce," *ACM Trans. Database Syst.*, vol. 37, Dec. 2012.

[48] A. Basiri, N. Behnam, R. de Rooij, L. Hochstein, L. Kosewski, J. Reynolds, and C. Rosenthal, "Chaos engineering," *IEEE Softw.*, vol. 33, p. 35–41, May 2016.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.

[50] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover"s distance as a metric for image retrieval," tech. rep., Stanford, CA, USA, 1998.

[51] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

APPENDIX A

DATASET DETAILS

Table 12: Description of streaming applications in the Exathlon dataset

App ID	Description	Query Operations
1	Counts the number of clicks for each user within the last batch by saving the results to HDFS for the first batch only.	`GROUP BY + COUNT`
2	Selects URLs visited more than 1,000 times from the start of the application by saving the results to HDFS for the first batch only.	`GROUP BY + COUNT + FILTER`
3	Selects URLs visited more than 1,000 times within a 30-second sliding window by saving the results to HDFS for the first batch only.	`GROUP BY + COUNT + FILTER`
4	Groups information by user within a 30-second sliding window by saving the results to HDFS for the first batch only.	`GROUP BY`
5	Computes the sum of ranks for the URLs visited by each user within a 30-second sliding window by saving the results to HDFS for the first batch only.	`JOIN + GROUP BY + SUM`
6	Performs the same operation as app 5, except that all the results are saved to HDFS.	`JOIN + GROUP BY + SUM`
7	Performs the same operation as app 4, except for the addition of a "CPU pressure" to simulate a user-defined function (here a computation of pi) when processing batches.	`GROUP BY + Simulated UDF`
8	Selects users that have performed more than 300 clicks within a 10-second jumping window by saving all the results to HDFS.	`GROUP BY + COUNT + FILTER`
9	Groups information by user within a 10-second jumping window by saving all the results to HDFS.	`GROUP BY`
10	Selects URLs visited more than 500 times within a 10-second jumping window by saving all the results to HDFS.	`GROUP BY + COUNT + FILTER`

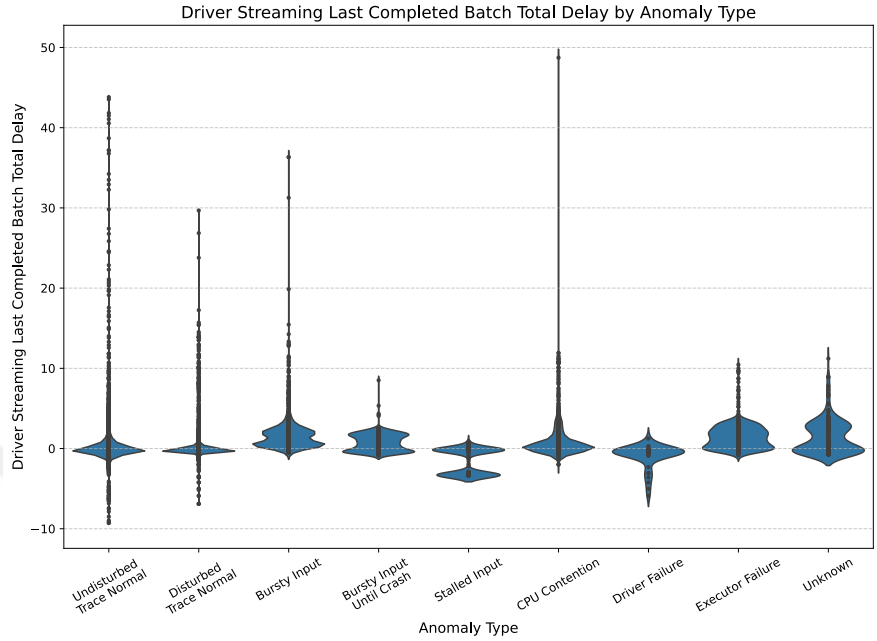


Figure 6: Driver streaming last completed batch processing total delay values by anomaly types.

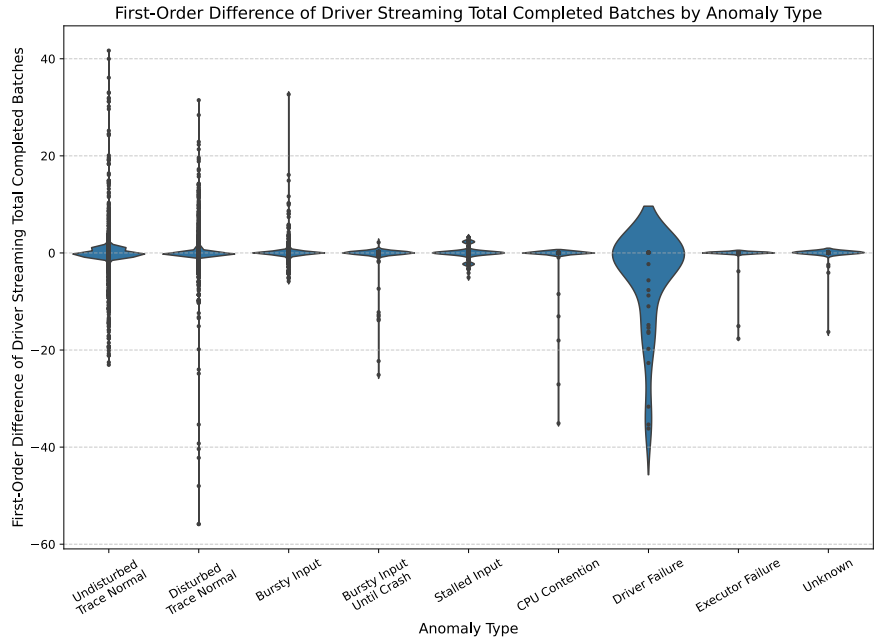


Figure 7: First-order difference of driver streaming total completed batches values by anomaly types.

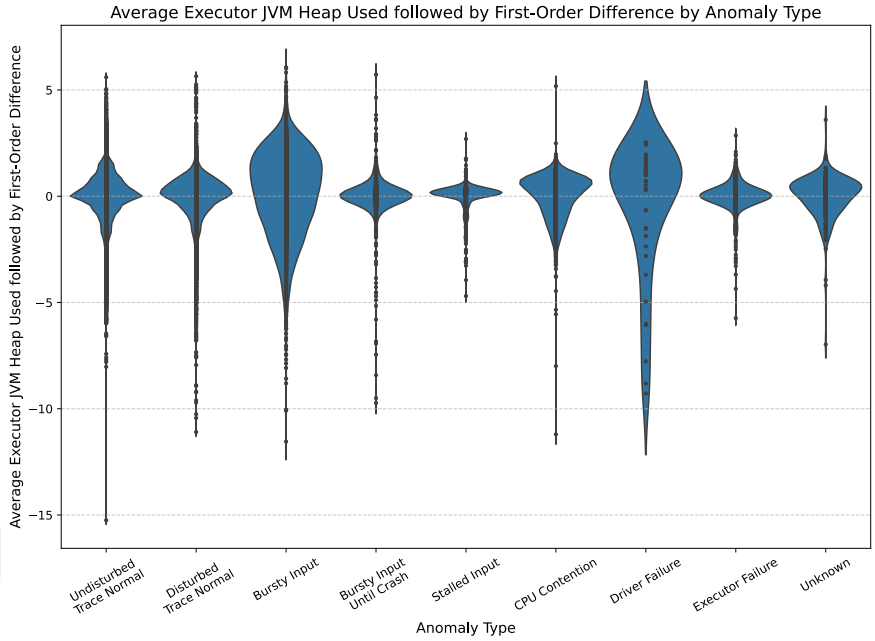


Figure 8: Average Executor JVM Heap Used followed by First-Order Difference values by anomaly types.

As illustrated in Figure 6, the total delay, comprising both processing and scheduling components, exhibits marked increases under the Bursty Input, Bursty Input Until Crash, Executor Failure, and Unknown anomaly types. These anomalies introduce significant processing bottlenecks: Bursty Input and its extreme variant, Bursty Input Until Crash, overwhelm Spark's processing resources, leading to congestion and increased scheduling latency. Executor Failures further contribute to delay by disrupting task execution and triggering task rescheduling across available executors. Although Unknown anomalies are not explicitly categorized, their impact on delay suggests that they may involve similar underlying disruptions, such as resource contention or system-level failures. In contrast, the Stalled Input anomaly leads to a reduction in total delay. This inverse effect arises because the absence of incoming data eliminates the need for active processing and complex scheduling, resulting in lower measured delays. However, this reduction does not indicate improved system performance; rather, it reflects a state of inactivity where the Spark system remains operational but lacks data to process.

Figure 7 shows that the Driver Failure anomaly has a pronounced negative effect on batch completion, evidenced by a sharp decline in the number of completed batches. This behavior is consistent with the expected disruption caused by driver crashes, which temporarily halt task scheduling and execution until the system recovers.

According to Figure 8, heap memory usage among executors exhibits notable variability under Bursty Input anomalies, consistent with the expectation that sudden surges in input data lead to elevated memory consumption. In contrast, the Driver Failure anomaly is characterized by negative distribution tails, indicating substantial drops in memory usage. This behavior likely results from driver crashes, which terminate active tasks and release the associated memory resources.

APPENDIX B

BASELINE MODELS' HYPERPARAMETER SETTINGS

B.1 LSTM

Table 13: Details of LSTM anomaly detection model implemented in Exathlon benchmark.

Category	**Hyperparameters**
Dataset	Batch Size = 32, Window Size = 40, Stride = 1, Forecast Horizon = 1
Architecture	See Figure 9
Optimization	Optimizer = Adam, Learning Rate = 7.87×10^{-4}
Training	Epochs = 200, Early Stopping Patience = 10
Loss	Mean Square Error (MSE)

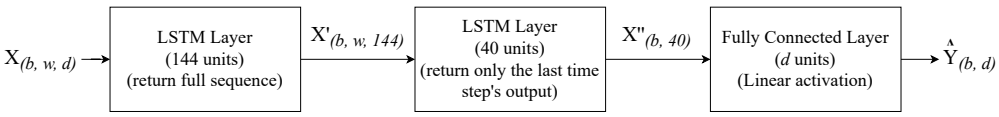


Figure 9: LSTM architecture in Exathlon benchmark.

B.2 Autoencoder

Table 14: Details of Autoencoder anomaly detection model implemented in Exathlon benchmark.

Category	**Hyperparameters**
Dataset	Batch Size = 32, Window Size = 40, Stride = 1
Architecture	See Figure 10
Optimization	Optimizer = Adam, Learning Rate = 3.60×10^{-4}
Training	Epochs = 200, Early Stopping Patience = 10
Loss	Mean Square Error (MSE)

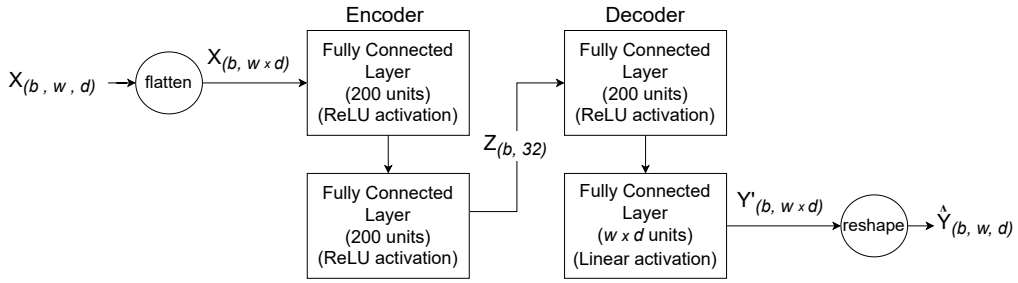


Figure 10: Autoencoder architecture in Exathlon benchmark.

B.3 BiGAN

Table 15: Details of BiGAN anomaly detection model implemented in Exathlon benchmark.

Category	**Hyperparameters**
Dataset	Batch Size = 32, Window Size = 40, Stride = 1
Architecture	See Figure 11
Optimization	Optimizer = Adam, Discriminator Learning Rate = 0.0004, Encoder/Generator Learning Rate = 0.0001
Training	Epochs = 200, Early Stopping Patience = 10
Loss	Binary Cross Entropy for both Discriminator and Encoder/Generator

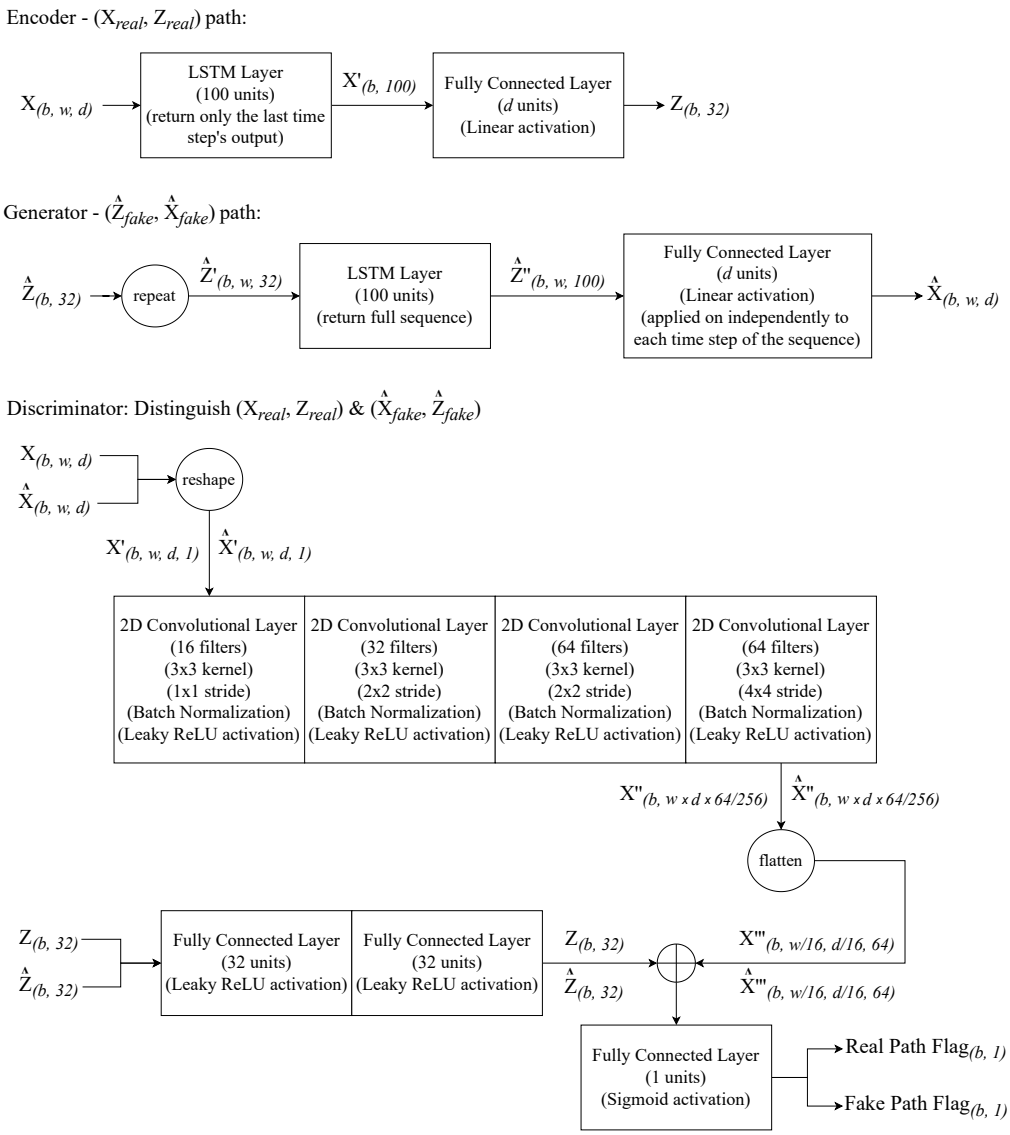


Figure 11: BiGAN architecture in Exathlon benchmark.

APPENDIX C

DATA PARTITION DETAILS

Table 16: Anomaly distribution in disturbed traces across cross-validation folds. Trace Name format: *app id_anomaly type_input rate_trace id*

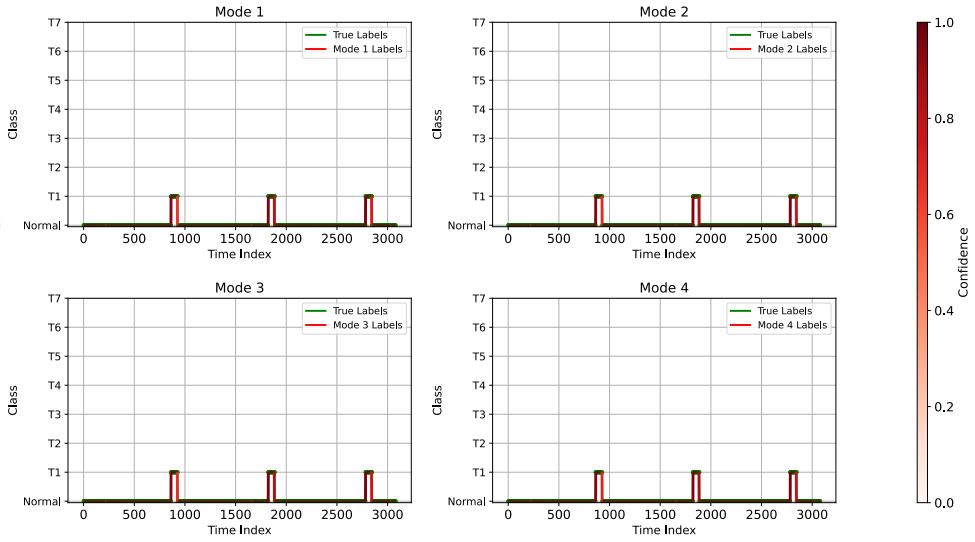
Trace Name	App Id	Trace Length	Anomaly Type	# Ranges	# Instances	Fold
Anomaly Type 1 Traces						
5_1_500000_62	5	3122	T1	6	394	1
2_1_100000_60	2	3121	T1	3	199	1
4_1_100000_61	4	8640	T1	9	1086	2
5_1_100000_64	5	3120	T1	3	200	2
6_1_500000_65	6	3121	T1	5	329	3
5_1_500000_63	5	2881	T1	3	365	3
Anomaly Type 2 Traces						
2_2_200000_69	2	194	T2	1	112	1
9_2_100000_66	9	502	T2	1	232	1
3_2_500000_70	3	175	T2	1	54	1
1_2_100000_68	1	197	T2	1	156	2
5_2_100000_72	5	167	T2	1	65	2
3_2_100000_71	3	167	T2	1	31	3
10_2_100000_67	10	687	T2	1	351	3
Anomaly Type 3 Traces						
6_3_200000_76	6	3120	T3	4	253	1
10_3_100000_75	10	3121	T3	4	257	2
8_3_200000_73	8	3120	T3	4	257	3
9_3_500000_74	9	3121	T3	4	264	3
Anomaly Type 4 Traces						
1_4_100000_80	1	2883	T4	6	375	1
5_4_100000_82	5	283	T4 T7	1 2	64 27	1

Trace Name	App Id	Trace Length	Anomaly Type	# Ranges	# Instances	Fold
3_4_100000_81	3	283	T4 T7	1 1	46 11	2
8_4_100000_77	8	2883	T4	6	383	2
9_4_100000_78	9	2885	T4	6	382	3
10_4_100000_79	10	2883	T4	6	376	3
Anomaly Type 5 and 6 Traces						
2_5_100000_87	2	2883	T5 T6	2 2	9 680	1
4_5_100000_90	4	243	T6 T7	1 1	7 36	1
5_5_100000_92	5	243	T5 T7	1 1	5 40	1
6_5_100000_93	6	243	T5 T7	1 1	5 44	1
1_5_100000_86	1	243	T6 T7	1 1	13 33	2
3_5_100000_89	3	397	T5 T6 T7	1 1 1	5 12 10	2
5_5_100000_91	5	2883	T5 T6	1 2	5 64	2
10_5_100000_85	10	396	T5 T7	1 1	5 8	2
2_5_100000_88	2	243	T6 T7	1 1	108 34	3
9_5_100000_84	9	397	T5 T6 T7	1 1 1	6 23 9	3
8_5_100000_83	8	397	T5 T6 T7	1 1 1	4 11 5	3

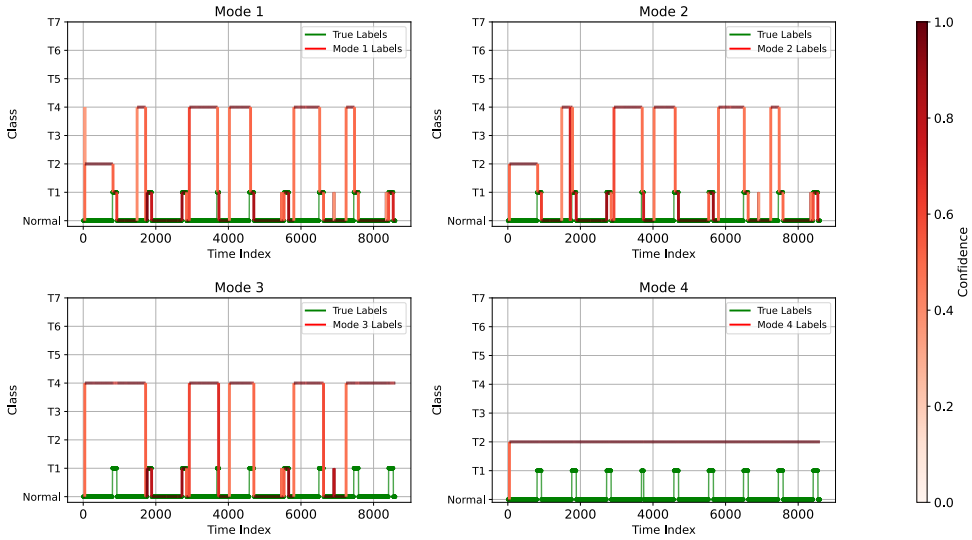
APPENDIX D

MODEL PREDICTION VISUALIZATIONS FOR DISTURBED TRACES

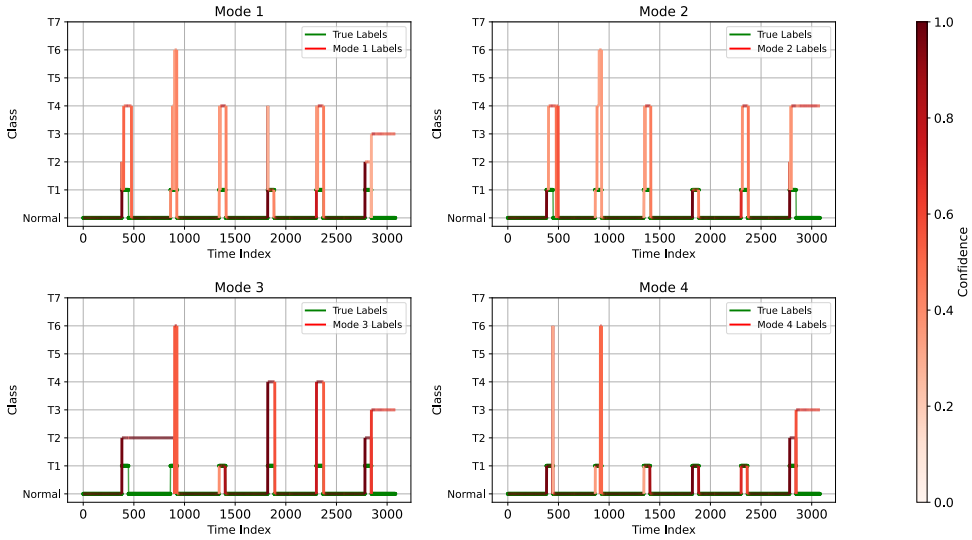
Figure 12: Prediction results for T1 anomalies.
Trace name format: *app id_anomaly type_input rate_trace id*.



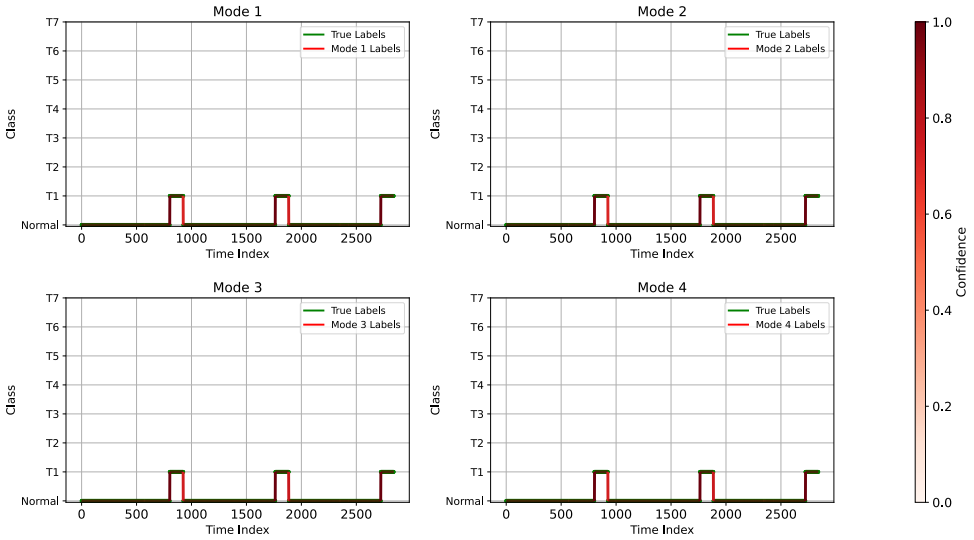
(a) Trace: 2_1_100000_60



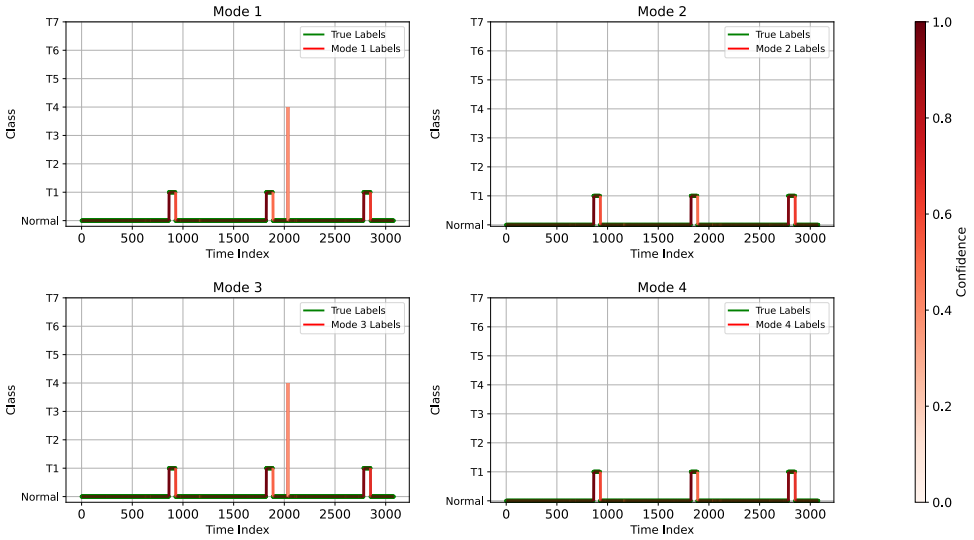
(b) Trace: 4_1_100000_61



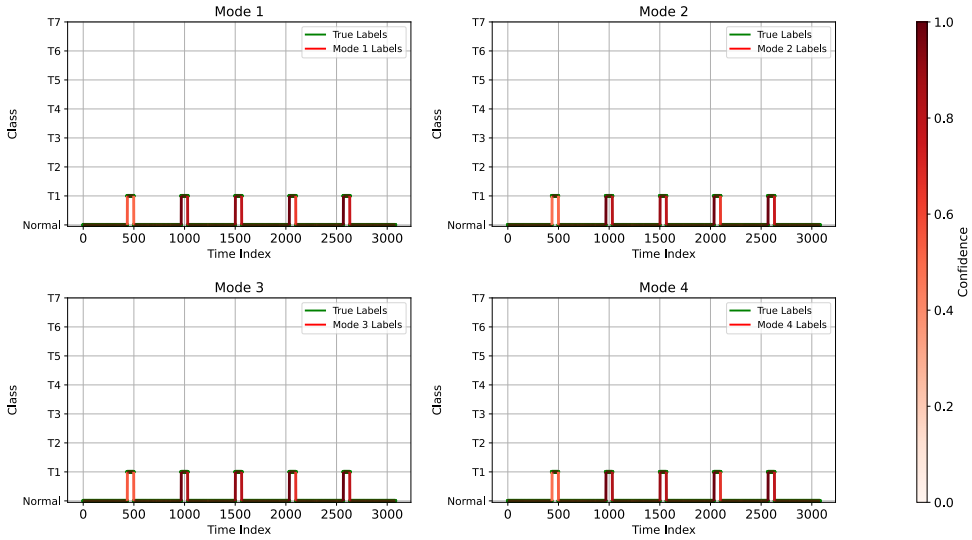
(c) Trace: 5_1_500000_62



(d) Trace: 5_1_500000_63

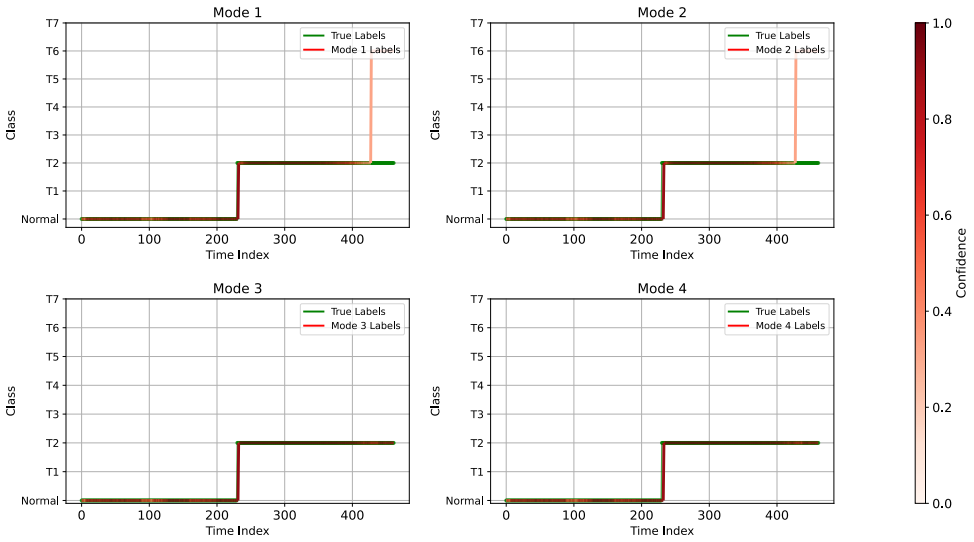


(e) Trace: 5_1_100000_64

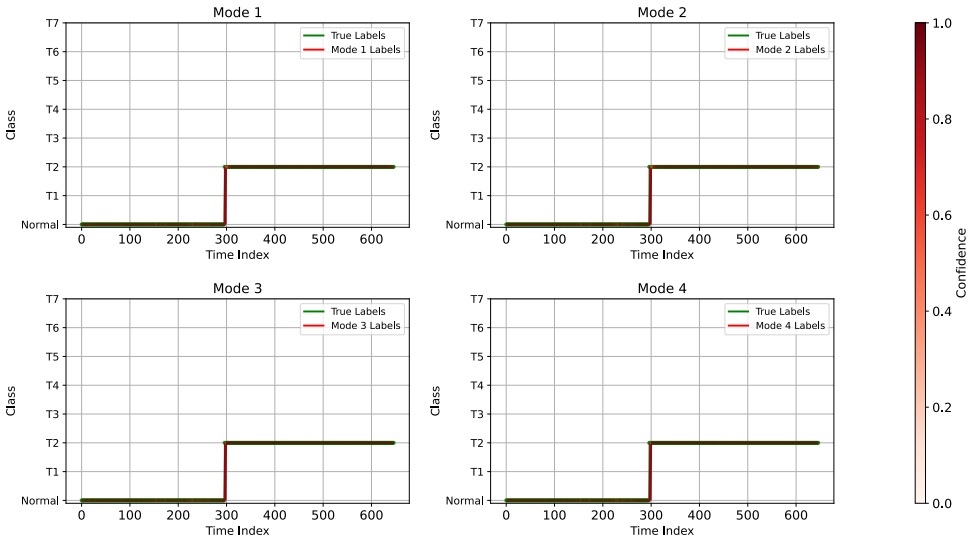


(f) Trace: 6_1_500000_65

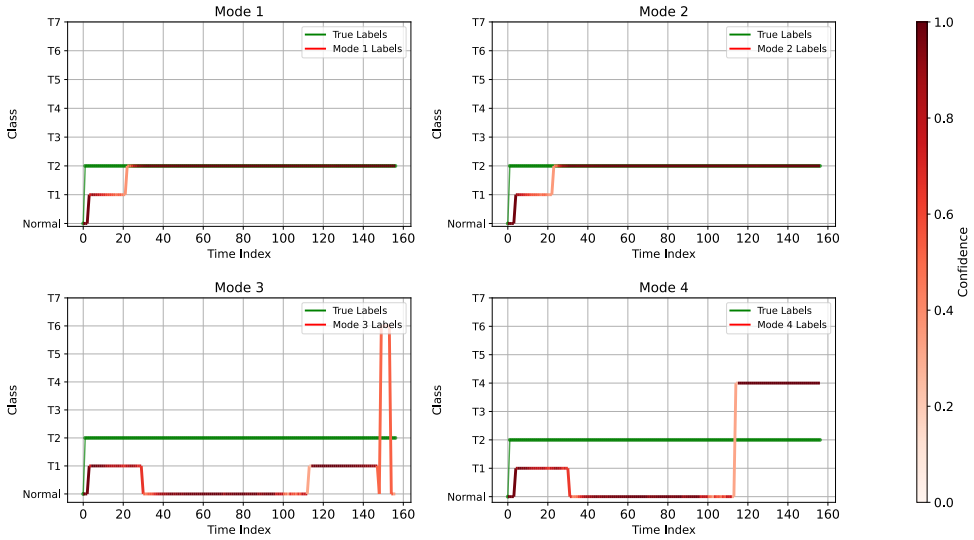
Figure 13: Prediction results for T2 anomalies.
Trace name format: *app id_anomaly type_input rate_trace id*.



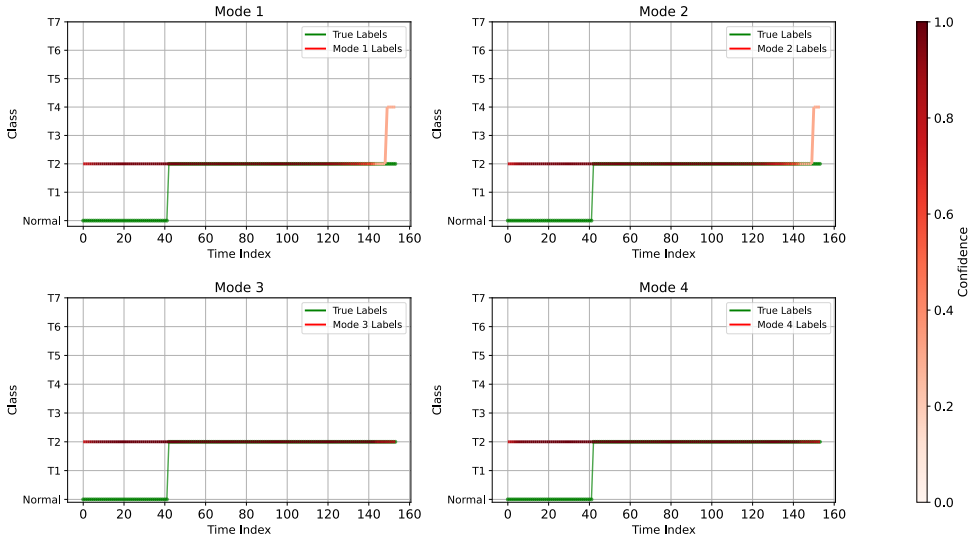
(a) Trace: 9_2_100000_66



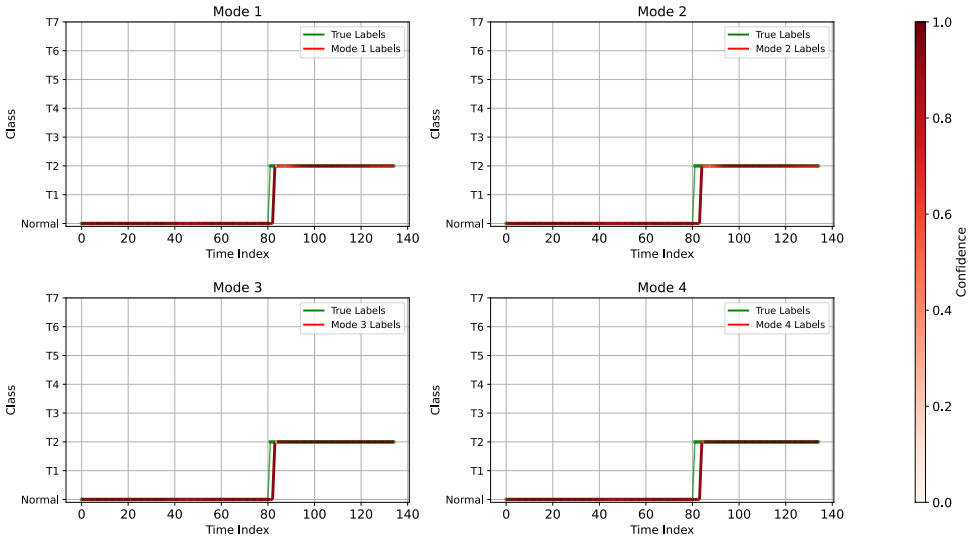
(b) Trace: 10_2_100000_67



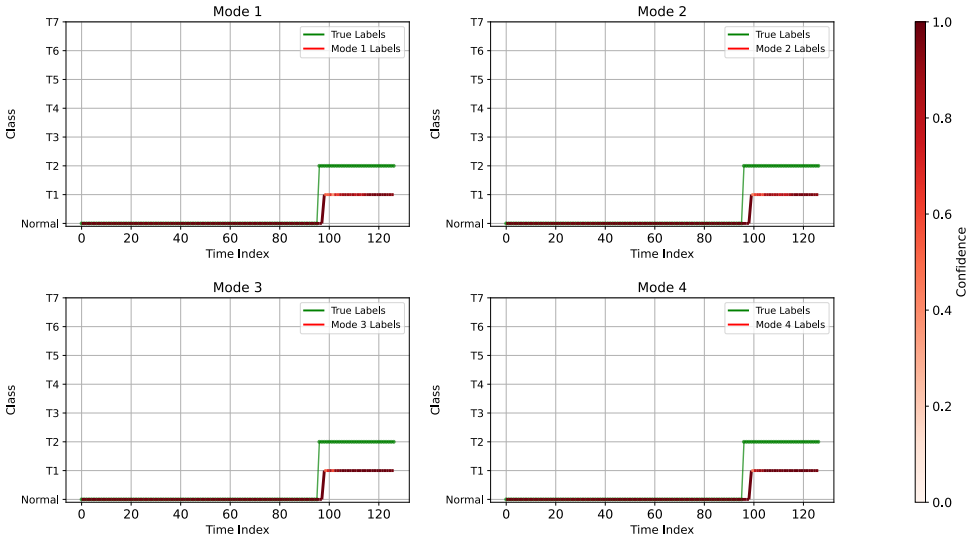
(c) Trace: 1_2_100000_68



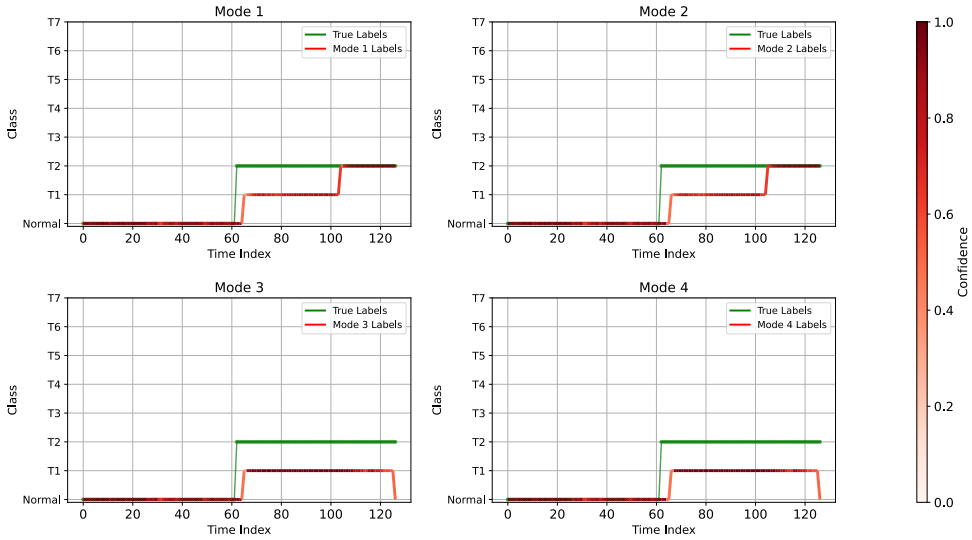
(d) Trace: 2_2_200000_69



(e) Trace: 3_2_500000_70

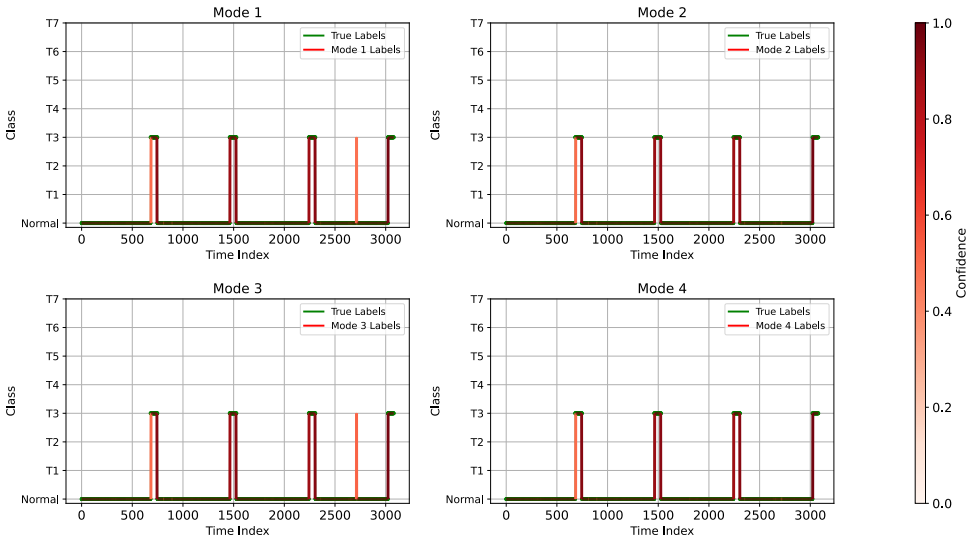


(f) Trace: 3_2_100000_71

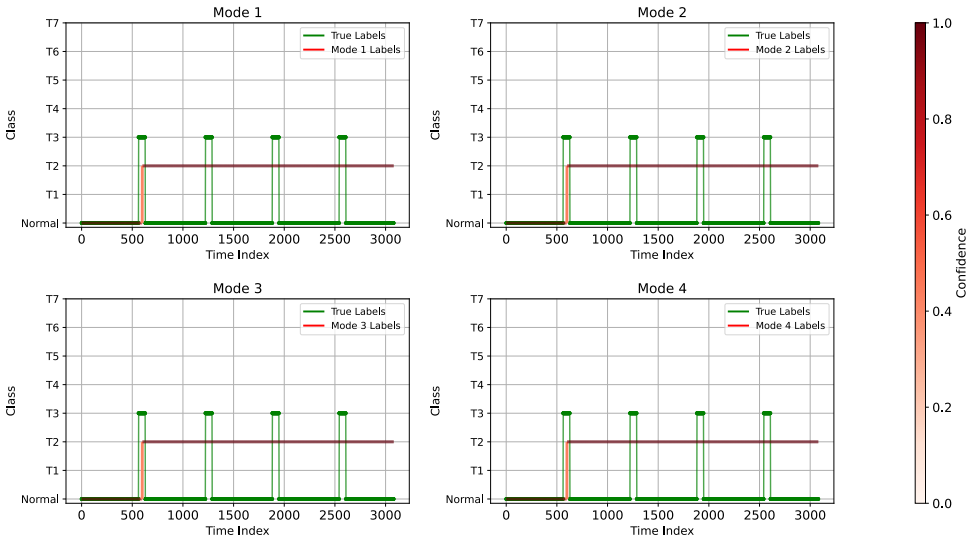


(g) Trace: 5_2_100000_72

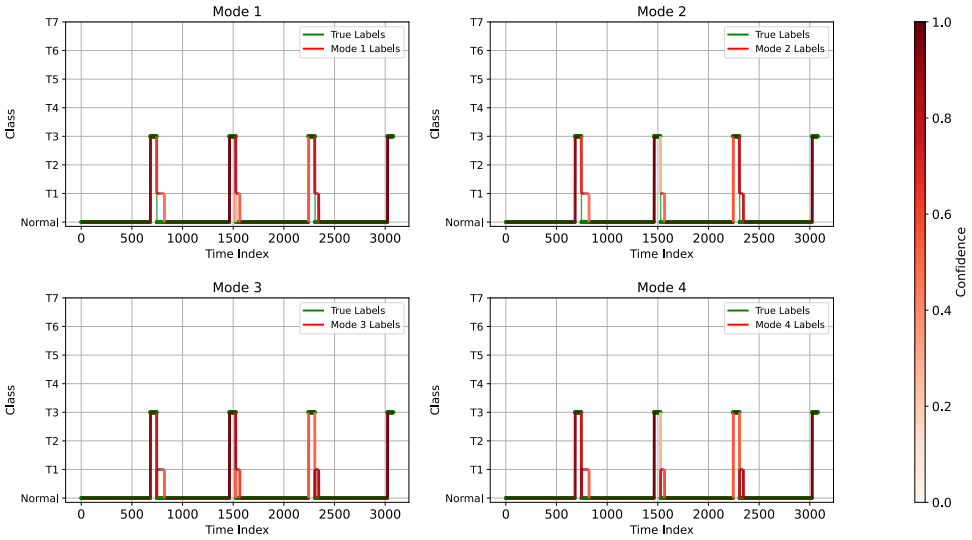
Figure 14: Prediction results for T3 anomalies.
Trace name format: *app id_anomaly type_input rate_trace id*.



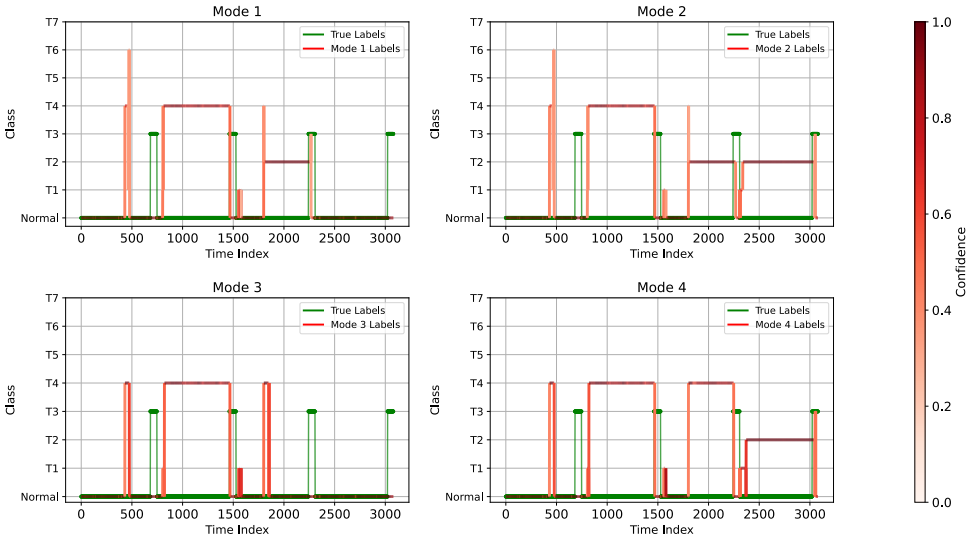
(a) Trace: 8_3_200000_73



(b) Trace: 9_3_500000_74

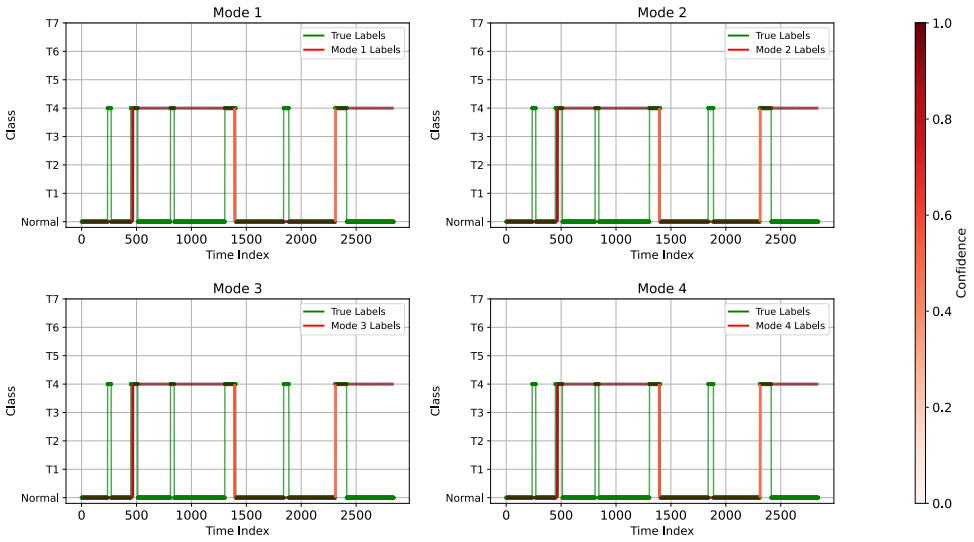


(c) Trace: 10_3_100000_75

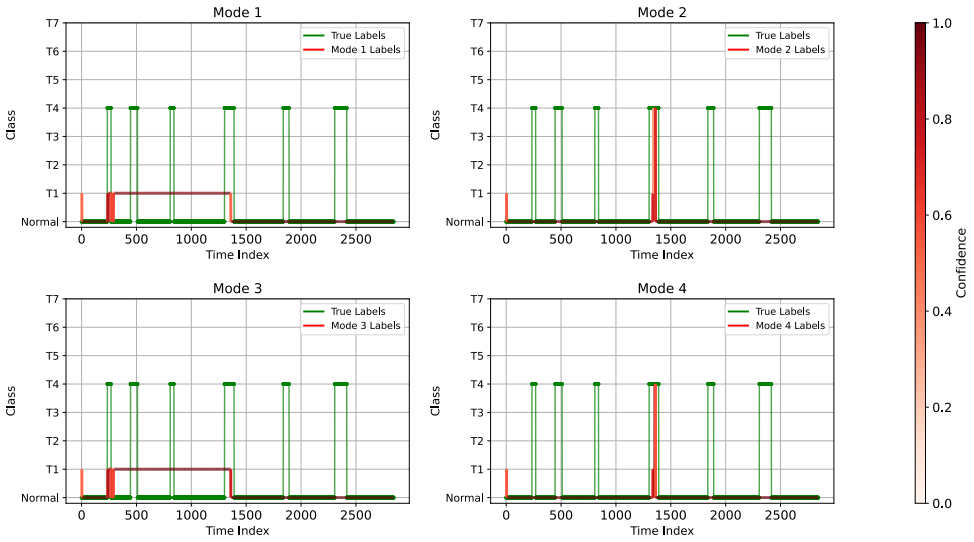


(d) Trace: 6_3_200000_76

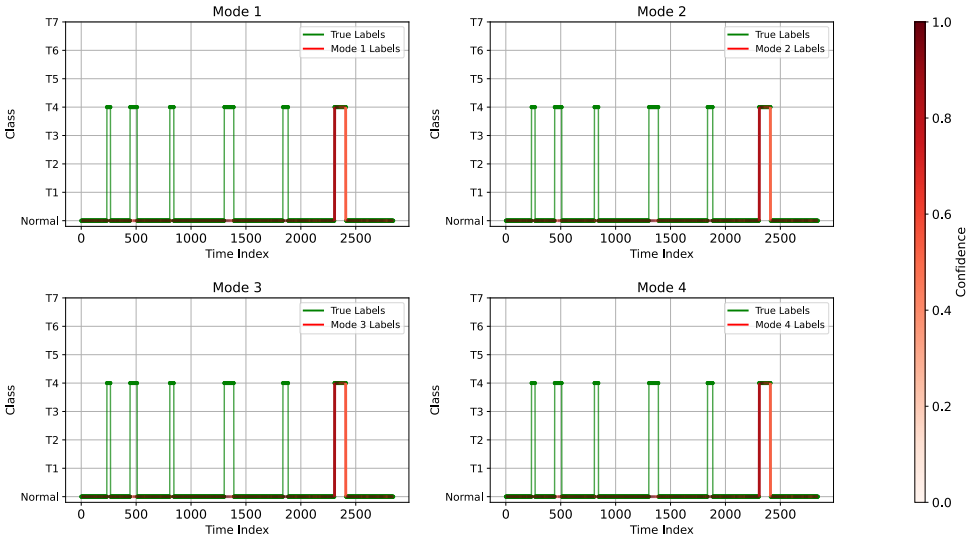
Figure 15: Prediction results for T4 anomalies.
Trace name format: *app id_anomaly type_input rate_trace id*.



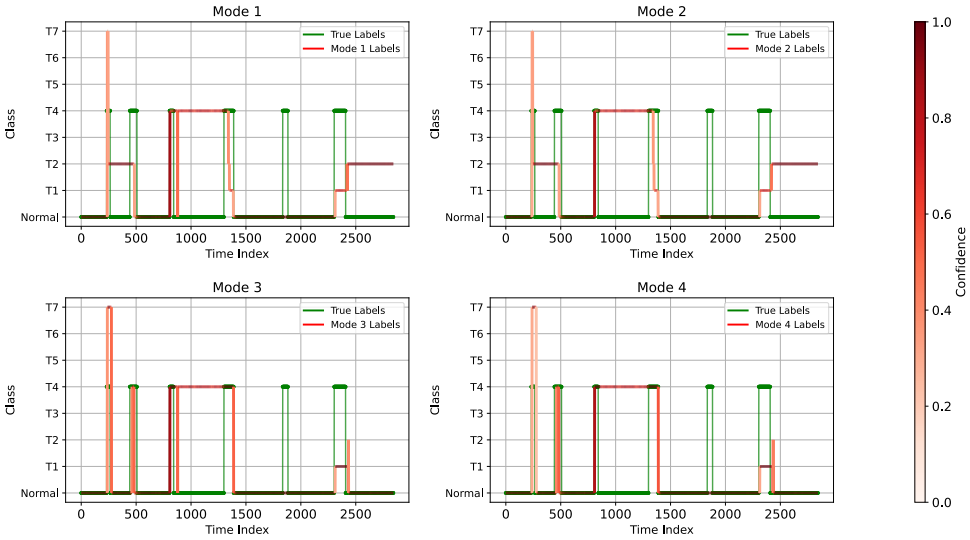
(a) Trace: 8_4_100000_77



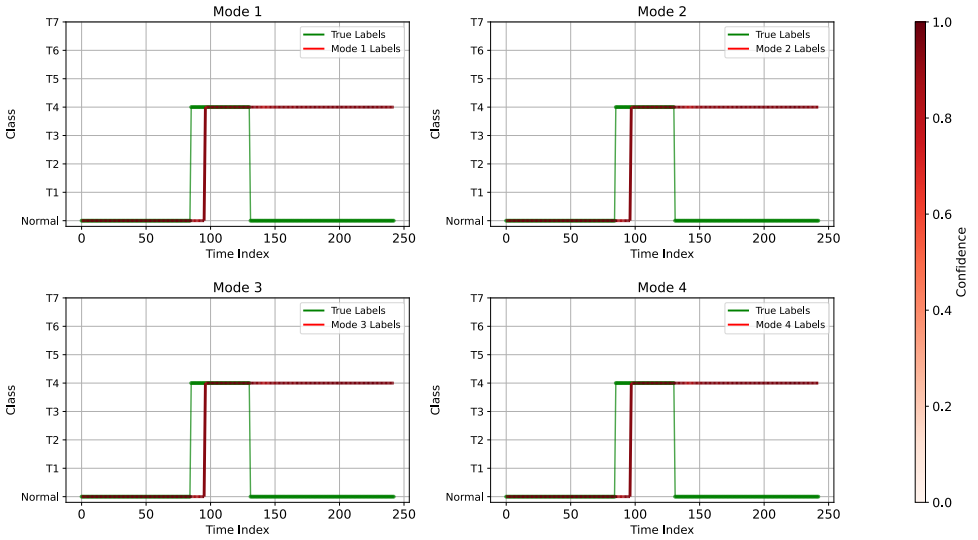
(b) Trace: 9_4_100000_78



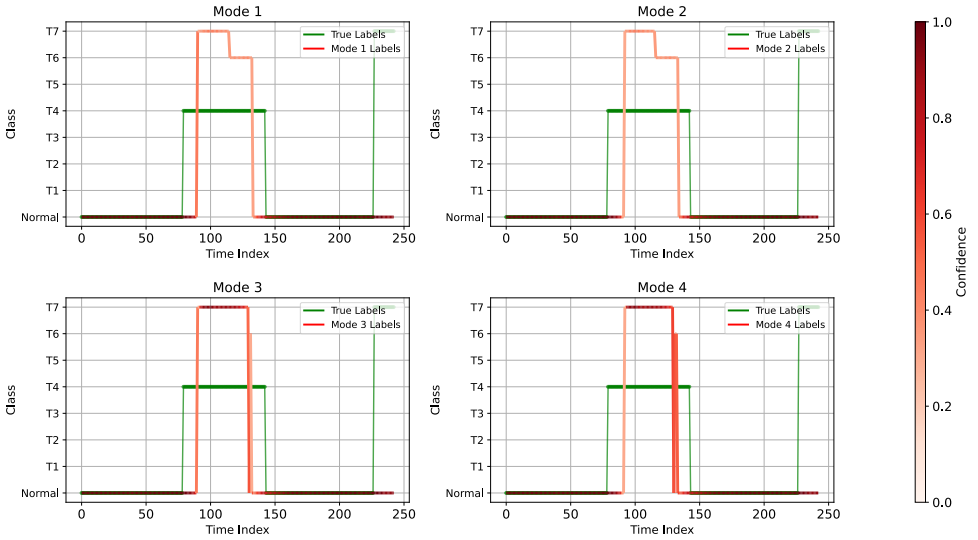
(c) Trace: 10_4_100000_79



(d) Trace: 1_4_100000_80

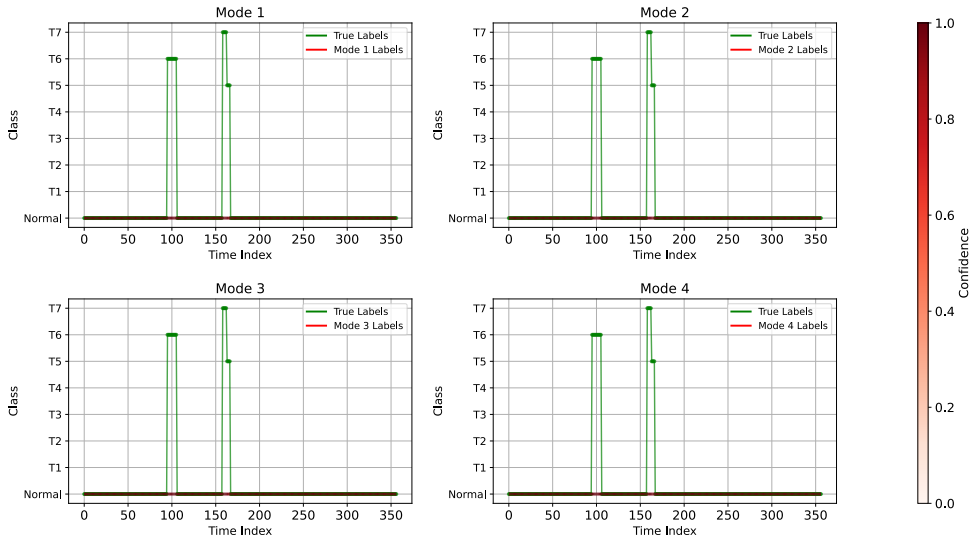


(e) Trace: 3_4_100000_81

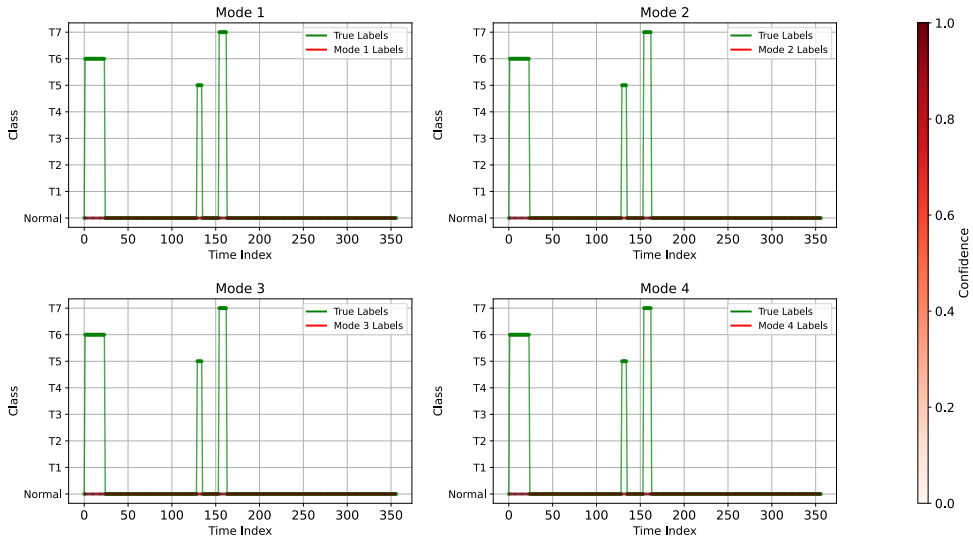


(f) Trace: 5_4_100000_82

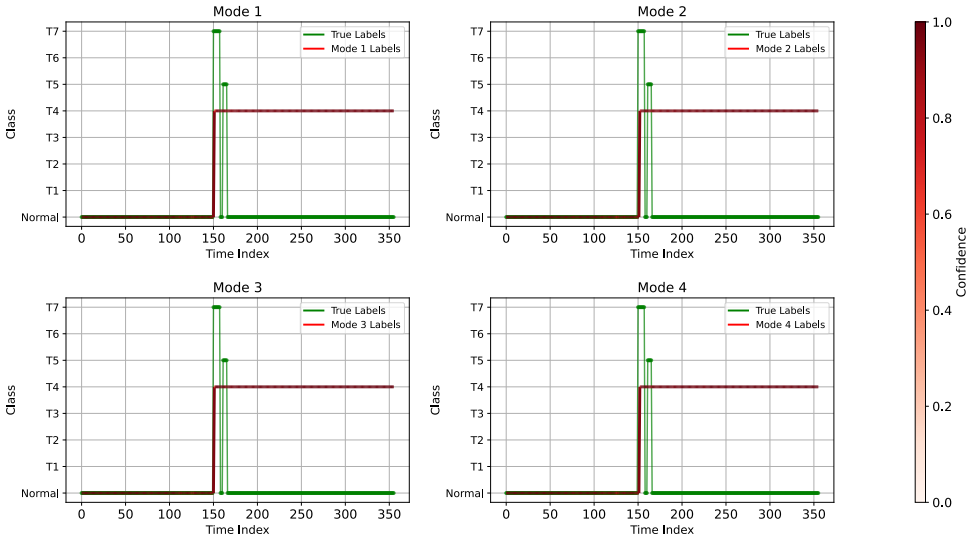
Figure 16: Prediction results for T5 and T6 anomalies.
Trace name format: *app id_anomaly type_input rate_trace id.*



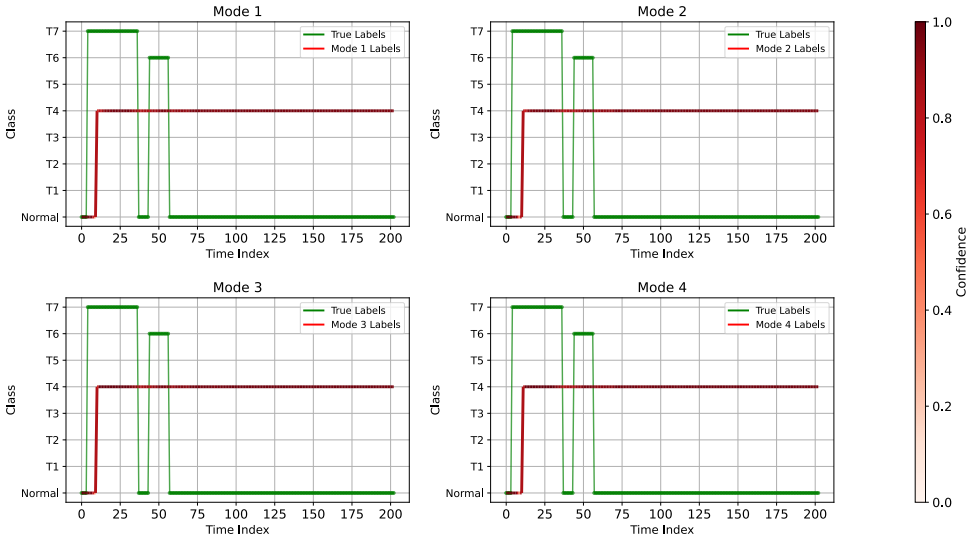
(a) Trace: 8_5_100000_83



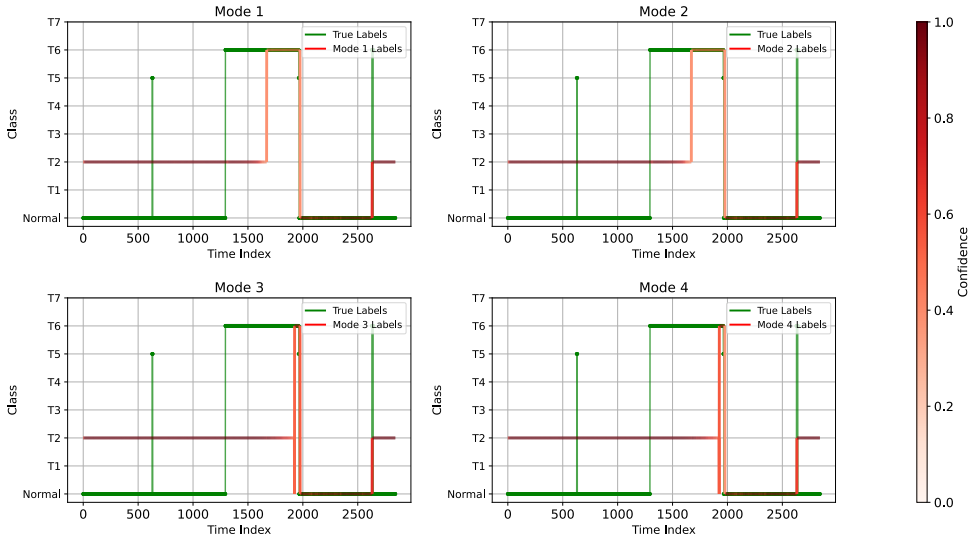
(b) Trace: 9_5_100000_84



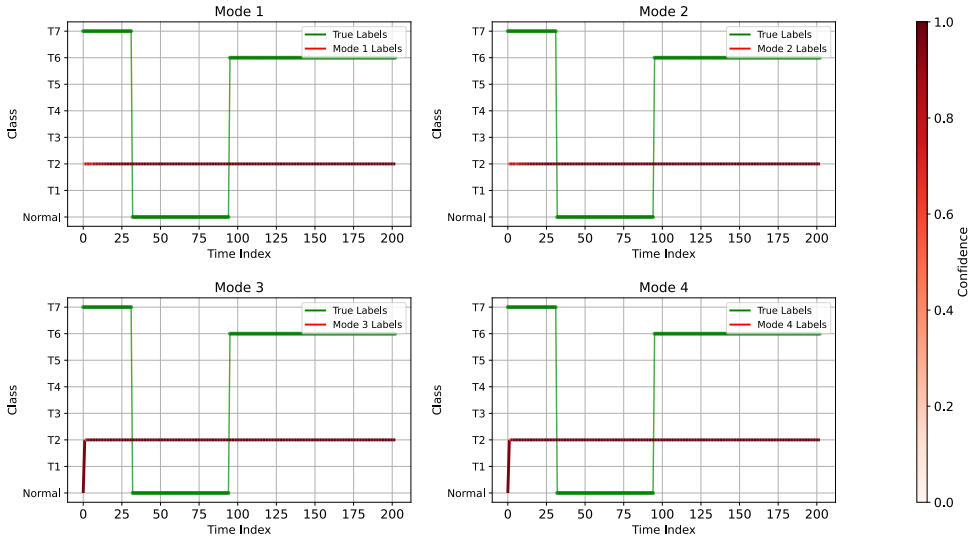
(c) Trace: 10_5_100000_85



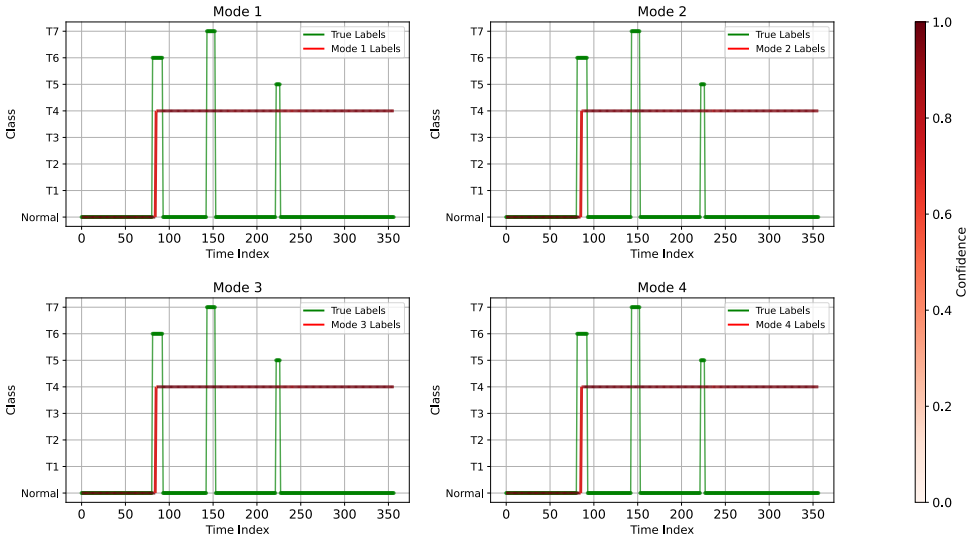
(d) Trace: 1_5_100000_86



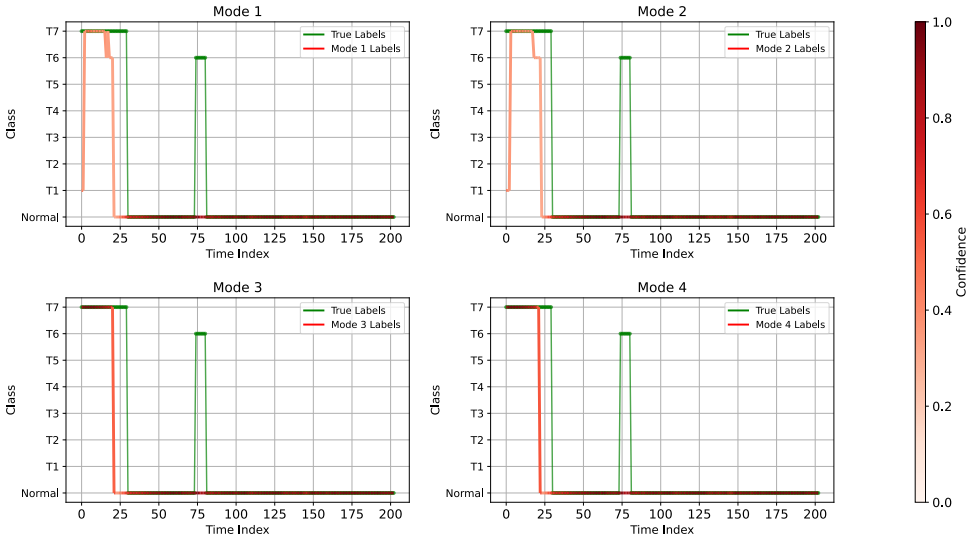
(e) Trace: 2_5_100000_87



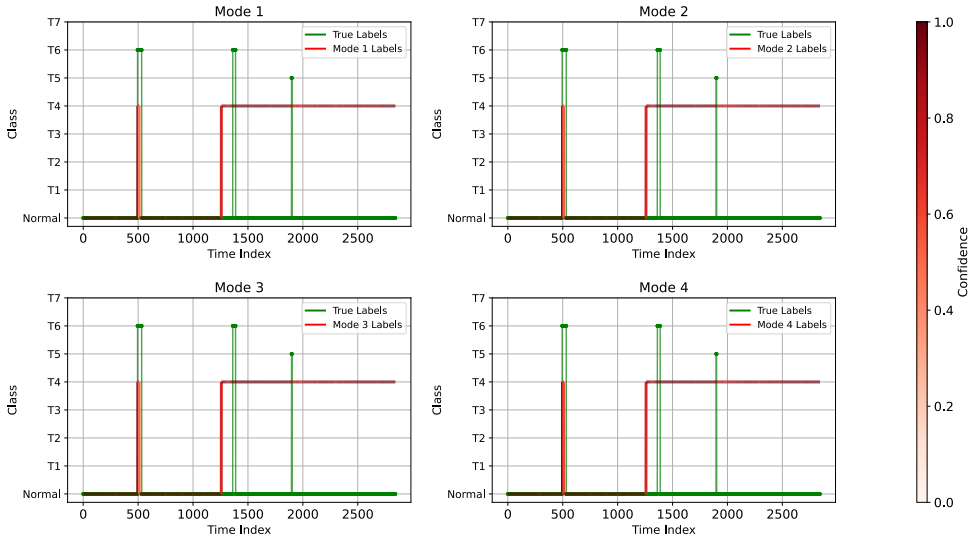
(f) Trace: 2_5_100000_88



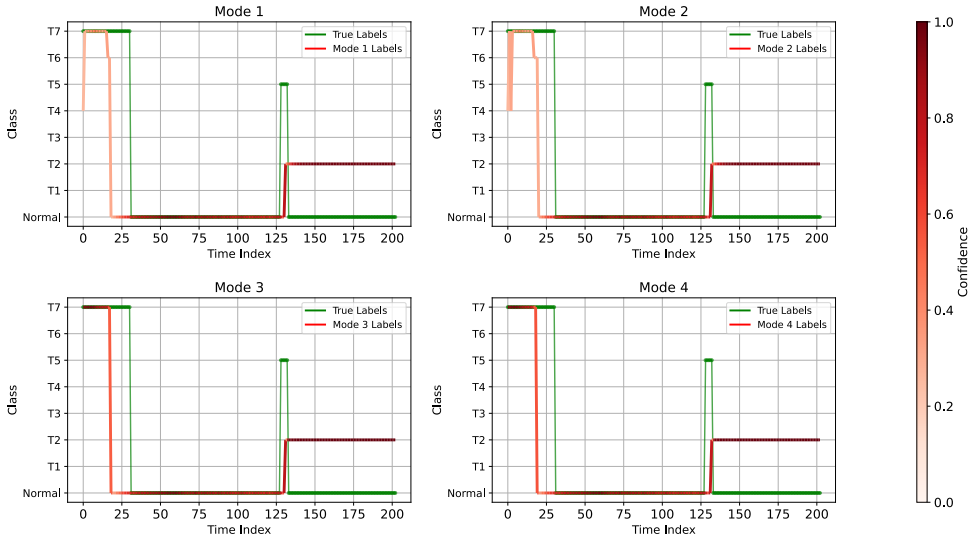
(g) Trace: 3_5_100000_89



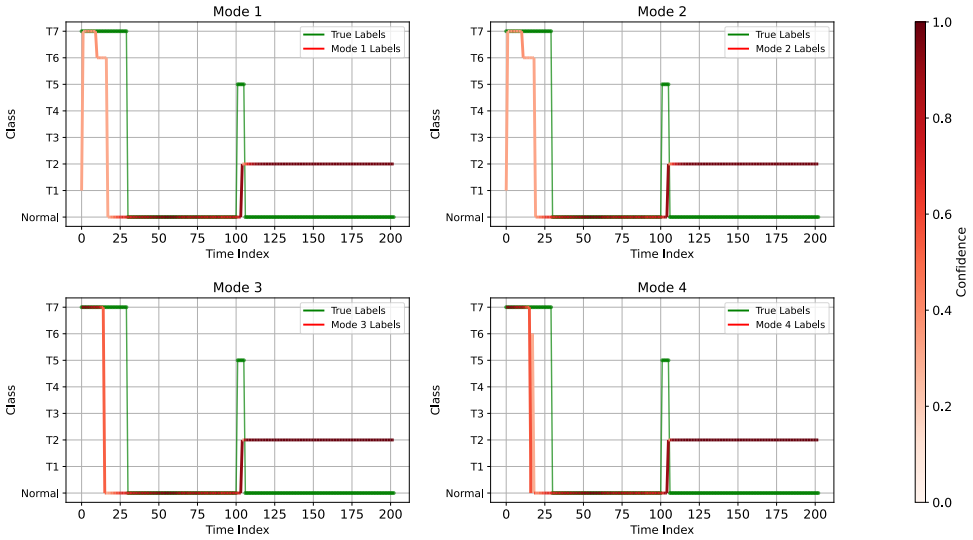
(h) Trace: 4_5_100000_90



(i) Trace: 5_5_100000_91



(j) Trace: 5_5_100000_92



(k) Trace: 6_5_100000_93

TEZ İZİN FORMU/ THESIS PERMISSION FORM

ENSTİTÜ / INSTITUTE

Fen Bilimleri Enstitüsü/ Graduate School of Natural and Applied Sciences

Sosyal Bilimler Enstitüsü/ Graduate School of Social Sciences

Uygulamalı Matematik Enstitüsü/ Graduate School of Applied Mathematics

Enformatik Enstitüsü/ Graduate School of Informatics

Deniz Bilimleri Enstitüsü/ Graduate School of Marine Sciences

YAZARIN/ AUTHOR

Soyadı/ Surname : Uzuntürk .....
Adı/ Name : Göksu .....
Bölümü / Department : Veri Bilimi .....

TEZİN ADI/ TITLE OF THE THESIS **(İngilizce/** English) A Post-Inference Transformer Framework Far Anomaly Range Detection in Multivariate Time Series

TEZİN TÜRÜ/ DEGREE: **Yüksek Lisans/** Master **Doktora/** PhD

1. **Tezin tamamı dünya çapında erişime açılacaktır./** Release the entire work immediately for access worldwide.

2. **Tez iki yıl süreyle erişime kapalı olacaktır./** Secure the entire work for patent and/or proprietary purposes for a period of **two year.**

3. **Tez altı ay süreyle erişime kapalı olacaktır./** Secure the entire work far period of **six months.**

4. **Tezim ile ilgili bir ambargo kararı bulunmamakla birlikte, tezimin OpenMETU'de erişime açılmasının mezuniyet tarihimden itibaren 1 (bir) yıl süreyle ertelenmesini talep ederim (Bu seçenek tezin sadece** OpenMETU'de erişimini sınırlandırır)./ Although there is no embargo decision **regarding my thesis, 1request** that the public access to my thesis on OpenMETU be postponed for a **period of one (1) year following my** graduation date (This option only restricts access to the thesis on **OpenMETU)**

Yazarın imzası/ Signature