ANOMALY DETECTION IN HETEROGENEOUS IOT SYSTEMS:
LEVERAGING SYMBOLIC ENCODING OF PERFORMANCE
METRICS FOR ANOMALY CLASSIFICATION

A Thesis

presented to

the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science

by
Maanav Patel
June 2024
COMMITTEE MEMBERSHIP

TITLE: Anomaly Detection in Heterogeneous IoT Systems: Leveraging Symbolic Encoding of Performance Metrics for Anomaly Classification

AUTHOR: Maanav Patel

DATE SUBMITTED: June 2024

COMMITTEE CHAIR: Joydeep Mukherjee, Ph.D.
Assistant Professor of Computer Science

COMMITTEE MEMBER: Sumona Mukhopadhyay, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Paul Anderson, Ph.D.
Professor of Computer Science
ABSTRACT

Anomaly Detection in Heterogeneous IoT Systems: Leveraging Symbolic Encoding of Performance Metrics for Anomaly Classification

Maanav Patel

Anomaly detection in Internet of Things (IoT) systems has become an increasingly popular field of research as the number of IoT devices proliferate year over year. Recent research often relies on machine learning algorithms to classify sensor readings directly. However, this approach leads to solutions being non-portable and unable to be applied to varying IoT platform infrastructure, as they are trained with sensor data specific to one configuration. Moreover, sensors generate varying amounts of non-standard data which complicates model training and limits generalization. This research focuses on addressing these problems in three ways a) the creation of an IoT Testbed which is configurable and parameterizable for dataset generation, b) the usage of system performance metrics as the dataset for training the anomaly classifier which ensures a fixed dataset size, and c) the application of Symbolic Aggregate Approximation (SAX) to encode patterns in system performance metrics which allows our trained Long Short-Term Memory (LSTM) model to classify anomalies agnostic to the underlying system configuration. Our devised IoT Testbed provides a lightweight setup for data generation which directly reflects some of the most pertinent components of Industry 4.0 pipelines including a MQTT Broker, Apache Kafka, and Apache Cassandra. Additionally, our proposed solution provides improved portability over state-of-the-art models while standardizing the required training data. Results demonstrate the effectiveness of utilizing symbolized performance metrics as we were able to achieve accuracies of 95.87%, 87.33%, and 87.47% for three different IoT system configurations. The latter two accuracies represent the model’s ability to be generalized to datasets generated from differing system configurations.
ACKNOWLEDGMENTS

My deepest appreciation and thanks are extended to everyone who has played a part in the successful completion of this master’s thesis. Their guidance, support, and encouragement have been instrumental in making this accomplishment possible.

Firstly, I owe a debt of gratitude to my advisor, Dr. Mukherjee, whose expertise and support has been invaluable throughout my journey. His insightful suggestions and constructive feedback have significantly shaped and refined this thesis.

I would also like to express my sincere gratitude to Dr. Mukhopadhyay for her invaluable guidance throughout the development of my thesis project. Her expertise in the field of Symbolic Encoding and Machine Learning has been instrumental in shaping the direction of my research and refining the methodologies employed.

Last but not least, I would also like to express my heartfelt thanks to my committee member, Dr. Anderson, for lending his time and expertise to my thesis committee.

My family and friends deserve special mention for their unwavering support and encouragement throughout this academic endeavor. Their faith in me, constant motivation, and understanding during the highs and lows of this journey have been a pillar of strength.

To all those mentioned above and to anyone else who has contributed in any way, I extend my sincere appreciation. Your support, encouragement, and belief in my abilities have been pivotal in bringing this thesis to completion.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem and Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Research Objectives and Questions</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Research Contributions</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Thesis Organization</td>
<td>9</td>
</tr>
<tr>
<td>2 Background and Related Work</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Types of Performance Anomalies</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1 Increased and Decreased Data Transmission Rates</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2 Inconsistent Data Transmission Rates</td>
<td>12</td>
</tr>
<tr>
<td>2.2 Machine Learning Techniques for Anomaly Detection</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1 Unsupervised Learning</td>
<td>14</td>
</tr>
<tr>
<td>2.2.2 Supervised Learning</td>
<td>15</td>
</tr>
<tr>
<td>2.3 IoT Data Generation, Streaming, and Storage</td>
<td>16</td>
</tr>
<tr>
<td>2.3.1 Physical Sensor Devices</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2 MQTT Broker</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3 Apache Kafka</td>
<td>19</td>
</tr>
<tr>
<td>2.3.4 Apache Cassandra</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Performance Metrics in IoT Systems</td>
<td>21</td>
</tr>
<tr>
<td>2.5 Symbolic Encoding</td>
<td>22</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Univariate LSTM Model</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Multivariate LSTM Model</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Training and Evaluation</td>
</tr>
<tr>
<td>4.4</td>
<td>Summary</td>
</tr>
<tr>
<td>5</td>
<td>Results</td>
</tr>
<tr>
<td>5.1</td>
<td>Evaluation Metrics</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Symbolization</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Anomaly Classification</td>
</tr>
<tr>
<td>5.2</td>
<td>Symbolization Results</td>
</tr>
<tr>
<td>5.3</td>
<td>Anomaly Classification Results</td>
</tr>
<tr>
<td>5.4</td>
<td>Summary</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and Future Work</td>
</tr>
<tr>
<td></td>
<td>BIBLIOGRAPHY</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Representative Table of Data Points After Applying SAX on the Collected Time-Series Data</td>
<td>51</td>
</tr>
<tr>
<td>4.2 Explored Hyperparameter Configurations for LSTM Models</td>
<td>56</td>
</tr>
<tr>
<td>5.1 Lookup Table Used for Dist Function for Four Symbols</td>
<td>60</td>
</tr>
<tr>
<td>5.2 MINDIST Comparison for 3 Symbols</td>
<td>63</td>
</tr>
<tr>
<td>5.3 MINDIST Comparison for 4 Symbols</td>
<td>64</td>
</tr>
<tr>
<td>5.4 MINDIST Comparison for 5 Symbols</td>
<td>64</td>
</tr>
<tr>
<td>5.5 Confusion Matrix for Univariate Model Applied to Medium IoT Testbed</td>
<td>69</td>
</tr>
<tr>
<td>5.6 Confusion Matrix for Multivariate Model Applied to Medium IoT Testbed</td>
<td>71</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example Application of Piecewise Aggregate Approximation.</td>
<td>26</td>
</tr>
<tr>
<td>2.2</td>
<td>Example Application of Symbolic Aggregate Approximation.</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Implemented IoT Pipeline.</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>Implemented IoT Testbed.</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Implemented Data Processing Pipeline.</td>
<td>47</td>
</tr>
<tr>
<td>4.2</td>
<td>Implemented Univariate LSTM Model.</td>
<td>54</td>
</tr>
<tr>
<td>4.3</td>
<td>Implemented Multivariate LSTM Model.</td>
<td>56</td>
</tr>
<tr>
<td>5.1</td>
<td>Computing Pairwise MINDIST Values</td>
<td>59</td>
</tr>
<tr>
<td>5.2</td>
<td>Pairwise Mindist Values for 3 Symbols.</td>
<td>64</td>
</tr>
<tr>
<td>5.3</td>
<td>Pairwise Mindist Values for 4 Symbols.</td>
<td>65</td>
</tr>
<tr>
<td>5.4</td>
<td>Pairwise Mindist Values for 5 Symbols.</td>
<td>65</td>
</tr>
<tr>
<td>5.5</td>
<td>Average Training Loss by Symbols and PAA Segment Numbers.</td>
<td>68</td>
</tr>
<tr>
<td>5.6</td>
<td>Univariate Model CV Accuracy on Medium IoT Testbed by Overlapping Window Length.</td>
<td>68</td>
</tr>
<tr>
<td>5.7</td>
<td>Multivariate Model CV Accuracy on Medium IoT Testbed by Overlapping Window Length.</td>
<td>69</td>
</tr>
<tr>
<td>5.8</td>
<td>Test Accuracies of Univariate and Multivariate Models when Generalized to Small and Large IoT Testbeds.</td>
<td>71</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

The Internet of Things (IoT) has become a cornerstone of the modern digital age, with its proliferation transforming various sectors, including healthcare, agriculture, and industry [55]. The International Data Corporation estimates there will be 55.7 billion billion connected IoT devices by 2025, generating almost 80B zettabytes of data [32]. IoT systems, which consist of interconnected devices that communicate and exchange data, have the potential to optimize processes, improve efficiency, and enable new services. IoT systems are characterized by their intrinsic diversity, encompassing a multitude of components that operate at a vast scale across interconnected networks. This inherent heterogeneity, coupled with the expansive nature and intricacies involved, renders the process of constructing a comprehensive, large-scale real-world IoT pipeline a difficult challenge.

An IoT pipeline is a comprehensive system that orchestrates the flow of data from various IoT devices and sensors through multiple stages of processing, storage, and analysis. It serves as the backbone for IoT applications, enabling the collection, transmission, and utilization of data generated by connected devices in real-time. Realistic IoT pipelines adhering to the Industry 4.0 recommendations are distributed across diverse hardware, software, and virtualization layers, reflecting the complexity and heterogeneity of modern industrial environments. These distributed pipelines leverage a combination of edge computing, cloud computing, and centralized data processing components to efficiently handle the high volume, velocity, and variety of data streams generated by IoT devices. By integrating different technologies and architectures, IoT pipelines facilitate seamless data ingestion, transformation, storage,
and analysis, ultimately enabling data-driven decision-making, process optimization, and intelligent automation in industrial and other IoT-enabled settings.

As IoT systems continue to expand and become more complex, they also become more susceptible to anomalies. Anomalies, or deviations from normal behavior, pose significant challenges in IoT systems. Performance anomalies in IoT pipelines can arise from various sources, including intermittent connectivity issues, network congestion, sensor malfunctions, environmental disturbances, hardware failures, and software bugs [70]. These anomalies can manifest as irregular transmission frequencies, data quality issues, data loss, corrupted data streams, or system crashes. Consequently, they disrupt the seamless delivery of sensor data, undermine the reliability of analytical insights, and hinder real-time monitoring and decision-making capabilities, ultimately impacting the overall performance and responsiveness of IoT deployments. Therefore, detecting and addressing these anomalies is crucial for maintaining the integrity and functionality of IoT systems.

1.1 Problem and Motivation

Previous research efforts in detecting performance anomalies in IoT systems have primarily relied on using sensor data directly as input for training anomaly detection models. However, this approach presents significant challenges due to the inherent heterogeneity of sensor data formats, diverse sensor types, and the vast variability in data collected from different IoT pipeline architectures [20, 1, 19]. The complexity of handling such diverse and variable datasets can make the training process computationally expensive and may hinder the generalization capability of the trained models across different IoT system configurations.
While using performance metrics collected directly from the IoT pipeline can potentially address the challenges associated with sensor data, this approach also faces its own set of difficulties. IoT pipelines are dynamic in nature, with the underlying architecture being scalable to accommodate varying volumes of sensor data and meet the Quality of Service requirements for applications running on the IoT platform. As a result, the performance metrics collected from these dynamic pipelines can exhibit different patterns and characteristics, making it challenging to develop anomaly detection models that can generalize effectively across diverse IoT pipeline configurations and scaling scenarios.

By leveraging symbolic representations of time-series data and system performance metrics, this research aims to address the challenges of variability and complexity in IoT datasets. The symbolic representation approach using Symbolic Aggregate Approximation (SAX) was chosen due to its ability to capture essential patterns and trends while enabling efficient similarity comparisons and anomaly detection. Symbolic encoding techniques have been successfully applied in various engineering domains for pattern recognition and analysis of complex dynamical systems [52, 51]. By transforming the performance metrics data into symbolic sequences, we aimed to extract high-level features associated with normal and anomalous system behavior, while abstracting away the specific hardware configurations and sensor data characteristics. Symbolic encoding is known to be resilient to noise and enables dimensionality reduction by mapping raw numeric measurements from the real domain to a discrete domain [36]. In the context of IoT systems, the variabilities due to changes in system dynamics can be analogous to noise, and symbolic encoding of raw data allows for reducing such variabilities while capturing the underlying system dynamics [55]. This approach aligned with the goal of developing a generalized and scalable anomaly detection framework that could leverage the symbolic nature of the encoded data for
efficient pattern matching, clustering, and frequent pattern mining, ultimately enabling the trained models to generalize across varying IoT deployments.

Additionally, utilizing system performance metrics as the primary input for anomaly detection eliminates the need for training models using collected sensor data, which can be highly variable and complex depending on the specific sensor types and configurations. By focusing on system-level metrics, such as CPU and network utilization, the proposed approach aims to capture the overall system behavior and health, potentially reducing the variability and complexity of the dataset required for model training. This system-centric approach could lead to more robust and generalizable anomaly detection models, as the performance metrics may exhibit more consistent patterns across different IoT deployments compared to the diverse and heterogeneous sensor data.

Instead of treating anomaly detection as a binary classification task, we explored a multi-class classification approach, considering four distinct labels: normal transmission frequency, increased transmission frequency, decreased transmission frequency, and inconsistent transmission frequency. This formulation allowed us to investigate the model’s performance in identifying various types of anomalies commonly encountered in IoT systems, providing insights into its capability to differentiate between diverse anomalous behaviors.

The proposed approach has the potential to enable more robust and scalable anomaly detection in IoT systems, enabling real-time monitoring and proactive mitigation of anomalies. By addressing the challenges of complexity, variability, and adaptability, this research aims to contribute to the development of more reliable and resilient IoT systems, ultimately enhancing the overall quality-of-service of these critical infrastructures.
1.2 Research Objectives and Questions

**RQ 1:** Can system performance metrics be effectively utilized to classify anomalies in IoT systems, thereby eliminating the need for training models using collected sensor data and decreasing the variability and complexity of a typical IoT dataset?

This question investigates the potential of using performance metrics collected from the underlying IoT pipeline as the input dataset to an ML model for anomaly classification. The objective is to evaluate if system performance metrics can be utilized as effective source for anomaly detection, eliminating the need for training models using collected sensor data, which can be more complex and variable.

**RQ 2:** Is it feasible to symbolically represent time-series data using Symbolic Aggregate Approximation (SAX), thereby capturing essential patterns and trends which enables the classification of anomalies independent of the underlying IoT system configuration?

This research question explores the viability of using SAX, a symbolic representation technique, to transform time-series data from IoT systems into symbolic representations. The goal is to determine whether these symbolic representations can effectively capture essential patterns and trends, enabling the classification of anomalies regardless of the specific IoT system configuration or sensor data characteristics.

**RQ 3:** How can a robust machine learning software architecture be designed to classify anomalies from symbolized IoT sequential time-series data, such that a trained model can evaluate anomalies from varying IoT hardware configurations?

This research question focuses on developing a robust machine learning software architecture capable of classifying anomalies from symbolized IoT sequential time-series
data. The goal is to design an architecture that enables a trained model to effectively evaluate anomalies across different IoT hardware configurations, ensuring generalizability and adaptability to varying system setups.

1.3 Research Contributions

This section describes the research contributions (RC) of this thesis in regards to the three research questions (RQ) mentioned in Section 1.2.

**RQ 1:** Can system performance metrics be effectively utilized to classify anomalies in IoT systems, thereby eliminating the need for training models using collected sensor data and decreasing the variability and complexity of a typical IoT dataset?

**RC 1:** Traditionally, anomaly detection in IoT systems has focused on leveraging sensor data as the primary input for training and inference. However, this approach can be problematic due to the inherent variability and complexity of sensor data, which often involves diverse data formats, varying dimensions, and inconsistent data quality. In our research, we propose a novel approach that utilizes system performance metrics, such as CPU and network utilization, as the sole features for anomaly detection.

In our research, we propose identifying anomalies in our IoT Pipeline from system performance metrics, specifically the CPU and network utilization, as opposed to the sensor-collected data. This approach allowed us to use a fixed dataset size across varying workloads applied to our testbed. Additionally, it standardized the input parameters for our proposed model, as opposed to sensor data, which can have varying numbers of data points, data widths, data types, and other changing characteristics. We found that our ML models trained on performance metrics yielded competitive
accuracies, demonstrating the efficacy of performance metrics as the sole features for an anomaly detection model.

**RQ 2:** Is it feasible to symbolically represent time-series data using Symbolic Aggregate Approximation (SAX), thereby capturing essential patterns and trends which enables the classification of anomalies independent of the underlying IoT system configuration?

**RC 2:** We propose transforming data collected from IoT systems using SAX to encode time-series patterns within the collected datasets. This approach allowed us to standardize collected metrics from differing IoT Testbeds for usage as inputs to our ML model. The application of SAX to varying hardware configurations demonstrated consistent and reliable pattern capture, indicating SAX’s capability to consistently capture patterns and anomalies in time-series data, regardless of the scale or complexity of the test environments.

Symbolic Aggregate Approximation (SAX) is a powerful technique for transforming time-series data into symbolic representations, enabling the extraction of essential patterns and trends. By applying SAX to the performance metrics collected from our IoT Testbeds, we were able to encode the time-series data into symbolic representations, which served as inputs to our ML model. This symbolic transformation effectively filtered out noise from the data, allowing the model to concentrate on capturing the most relevant patterns and anomalies.

One of the key advantages of using SAX is its ability to capture patterns and anomalies independent of the underlying IoT system configuration. Our experimental results across three different IoT Testbeds (small, medium, and large) demonstrated the consistency and robustness of this approach. The distance measurements among the symbolic representations derived from each testbed were within an acceptable range,
indicating that SAX effectively captured the essential patterns and trends, regardless of the scale or complexity of the IoT system. This uniformity in distance metrics across the test environments highlights SAX’s capability to consistently represent time-series data, enabling the classification of anomalies without being constrained by specific hardware configurations or sensor data characteristics.

By relying on performance metrics, we were able to employ a fixed dataset size across varying workloads applied to our IoT Testbeds. This standardization of input parameters simplified the data preprocessing and feature engineering steps, streamlining the overall anomaly detection process. Furthermore, performance metrics are generally more consistent and reliable than sensor data, reducing the challenges associated with handling diverse data types and formats. Our experimental results demonstrated that both univariate and multivariate models trained solely on performance metrics achieved competitive accuracies, highlighting the efficacy of this approach in accurately detecting anomalies in IoT systems.

**RQ 3:** How can a robust machine learning software architecture be designed to classify anomalies from symbolized IoT sequential time-series data, such that a trained model can evaluate anomalies from varying IoT hardware configurations?

**RC 3:** In this work, we developed a robust Long Short-Term Memory (LSTM) model designed to classify performance-based anomalies from our devised IoT Testbeds. This model was trained on collected performance metrics, which were subsequently collated, symbolized through SAX, and organized into overlapping windows. Our design utilized data from a baseline Medium IoT Testbed as the model’s training set. We then evaluated the model’s ability to generalize to unseen datasets from differing sources by utilizing two similarly configured IoT Testbeds, Small and Large, which employed the same IoT Pipeline but with varying hardware performance levels.
At the core of our anomaly detection approach is a robust Long Short-Term Memory (LSTM) model, a type of recurrent neural network well-suited for processing sequential data. The LSTM model was designed to classify performance-based anomalies in our IoT Testbeds by leveraging the symbolized performance metrics as input. To prepare the data for training and evaluation, we collated the collected performance metrics, transformed them into symbolic representations using SAX, and organized them into overlapping windows. This overlapping window approach allowed the model to capture temporal dependencies and patterns within the sequential data.

To assess the generalizability of our approach, we employed a strategic experimental setup. The LSTM model was trained on the symbolized performance metrics collected from a baseline IoT Testbed, serving as the primary training dataset. Subsequently, we evaluated the trained model’s performance on two additional IoT Testbeds, which shared the same IoT Pipeline architecture but differed in hardware performance. This setup enabled us to assess the model’s ability to generalize and accurately detect anomalies in unseen datasets originating from different hardware configurations.

1.4 Thesis Organization

This research is organized as follows.

Chapter 2 outlines the background, research, and related work to the field.

Chapter 3 discusses the devised IoT Pipeline at a high level, including each stage of the pipeline and its role in the overall data generation and processing procedure. This chapter also mentions the three implemented IoT Testbeds which were developed from our devised IoT Pipeline. These testbeds were comprised of similar components implemented on differing hardware levels. Additionally, unique performance anomali-
lies were systematically introduced to the system and three datasets were collected for training and evaluation.

Chapter 4 talks about the methodology employed in this work. The chapter explores the approach of data symbolization through SAX, collation, labeling, and additional preprocessing steps prior to classification. Additionally, the implementation of the devised LSTM models for anomaly classification are outlined and respective architectures are displayed.

Chapter 5 explores the methods and metrics utilized to evaluate our work, as well as the corresponding results obtained. Distances between symbolized datasets are compared. Additionally, the effectiveness of both a univariate and multivariate LSTM model architecture are discussed. Finally, the ability for models to be generalized between datasets is explored and presented.

Chapter 6 includes the conclusions derived from our research and respective results as well as possible future work which could expand our research and explore further contributions to this field.
2.1 Types of Performance Anomalies

In the context of Internet of Things (IoT) systems, performance anomalies can manifest in various forms, each presenting unique challenges and potential implications. Understanding the different types of performance anomalies is crucial for effective anomaly detection and mitigation strategies [50]. This section provides an overview of three common performance anomalies related to data transmission rates.

2.1.1 Increased and Decreased Data Transmission Rates

IoT systems often rely on efficient and reliable data transmission to facilitate communication between devices, gateways, and cloud services [24]. However, there may be instances where the data transmission rates unexpectedly increase or decrease, leading to potential performance issues.

Increased data transmission rates can be caused by various factors, such as network congestion, distributed denial-of-service (DDoS) attacks, or malfunctioning sensors generating excessive data [46]. Elevated data transmission rates can strain network resources, leading to latency issues, packet loss, and overall degradation of system performance. In industrial IoT (IIoT) environments, where real-time data processing is critical, such anomalies can disrupt production processes, potentially causing costly downtime or safety concerns [70]. Detecting and mitigating increased data transmission rate anomalies is essential for maintaining the reliability and efficiency
of IoT systems. Effective anomaly detection techniques can identify these anomalies early, enabling proactive measures to be taken, such as throttling data transmission, identifying and addressing the root cause, or implementing network load-balancing mechanisms [78].

On the other hand, IoT systems may also experience anomalies characterized by decreased data transmission rates. This situation can arise due to various factors, including network outages, hardware failures, or software bugs that affect data transmission functionality [14]. Decreased data transmission rates can lead to incomplete or delayed data delivery, potentially hampering the ability to monitor and control IoT devices effectively. In scenarios where real-time data analysis is crucial, such as predictive maintenance or remote monitoring applications, reduced data transmission rates can result in incomplete or outdated information, hindering decision-making processes.

Detecting and addressing decreased data transmission rate anomalies is vital for maintaining the integrity and reliability of IoT systems. Anomaly detection techniques can identify these issues promptly, enabling corrective actions to be taken, such as network troubleshooting, device replacement, or software updates to restore normal data transmission rates.

2.1.2 Inconsistent Data Transmission Rates

In addition to increased or decreased data transmission rates, IoT systems may also experience anomalies characterized by inconsistent or erratic data transmission patterns. These anomalies can manifest as fluctuations in data transmission rates, intermittent data delivery, or irregular bursts of data transmission [17].
Inconsistent data transmission rates can be caused by various factors, such as intermittent network connectivity issues, hardware or software instabilities, or interference from external sources [11]. These anomalies can introduce uncertainty and unpredictability into the system, making it challenging to establish reliable data transmission patterns and hampering the ability to make informed decisions based on real-time data analysis.

2.2 Machine Learning Techniques for Anomaly Detection

Machine learning techniques for anomaly detection can be broadly categorized into univariate and multivariate approaches [41]. Univariate techniques focus on analyzing individual features or time series independently, such as applying statistical methods, density estimation, or neural networks to identify anomalies in a single dimension [58]. Multivariate techniques, on the other hand, consider the relationships and dependencies between multiple features or time series, leveraging techniques like principal component analysis, clustering, or ensemble methods to detect anomalies in high-dimensional data spaces. The choice between univariate and multivariate approaches depends on the nature of the data, the complexity of the anomaly patterns, and the computational resources available.

Anomaly detection is a critical task in various domains, including Internet of Things (IoT) systems, where identifying and mitigating anomalous behavior is essential for ensuring system reliability, security, and optimal performance. Machine learning techniques have emerged as powerful tools for anomaly detection, offering data-driven approaches to identify patterns and deviations that may be difficult to detect through traditional rule-based methods [3, 1, 64, 19, 20, 30]. This section explores various ma-
chine learning techniques and their applications in the context of anomaly detection for IoT systems.

2.2.1 Unsupervised Learning

In many IoT scenarios, obtaining labeled data for anomaly detection can be challenging or impractical. Unsupervised learning techniques aim to identify anomalies without relying on labeled training data, making them well-suited for scenarios where anomalies are rare or unknown in advance [1]. Clustering techniques, such as k-means clustering or Gaussian mixture models, group similar instances together based on their proximity or similarity in the feature space. These techniques can be used for anomaly detection by identifying instances that do not belong to any of the learned clusters, as these are likely to be anomalies or outliers. Clustering algorithms have the advantage of being able to detect anomalies without requiring labeled data, but they may struggle with high-dimensional data or complex data distributions.

Dimensionality reduction methods like Principal Component Analysis (PCA) or Autoencoders can project high-dimensional data into a lower-dimensional space, enabling the detection of anomalies as instances with high reconstruction errors or deviations from the normal subspace [76]. PCA is a linear dimensionality reduction technique that finds the directions of maximum variance in the data, while Autoencoders are a type of neural network that learns to reconstruct the input data in a lower-dimensional space [82]. Anomalies can be identified as instances with high reconstruction errors, indicating that they deviate significantly from the learned normal patterns [59].

One-class classifiers, such as One-Class Support Vector Machines (OC-SVMs) or Isolation Forests, learn to model the normal behavior of the system and identify instances that deviate from this learned distribution as anomalies [19]. OC-SVMs find a max-
imum margin boundary that separates the normal instances from the origin in the feature space, while Isolation Forests are an ensemble method that randomly partitions the data and identifies anomalies as instances that are isolated more quickly during the partitioning process. These techniques are particularly useful when dealing with imbalanced datasets, where anomalies are rare or scarce compared to normal instances.

\subsection{2.2.2 Supervised Learning}

Supervised learning algorithms are trained on labeled datasets, where examples of normal and anomalous behavior are provided. These algorithms learn to distinguish between multiple classes and can then be applied to new, unseen data for anomaly detection. Support Vector Machines (SVMs) are a popular supervised learning technique that can be employed for binary classification tasks, separating normal and anomalous instances in the feature space by finding the optimal hyperplane that maximizes the margin between the two classes \cite{39}. SVMs have been widely used in various anomaly detection applications due to their ability to handle high-dimensional data and their effectiveness in separating non-linearly separable classes through the use of kernel functions \cite{18}.

Decision trees and Random Forests are another class of supervised learning algorithms that have been successfully applied to anomaly detection tasks \cite{53}. Decision trees construct a tree-like model of decisions and their possible consequences, while Random Forests are an ensemble learning method that combines multiple decision trees, trained on different subsets of the data, to improve the overall prediction accuracy and robustness \cite{4}. These algorithms can effectively learn patterns and make predictions, enabling the identification of anomalous instances based on the learned decision rules.
Deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have gained significant attention in the field of anomaly detection due to their ability to learn complex patterns and representations from data [65]. CNNs are particularly effective for detecting anomalies in image or spatial data, while RNNs, including Long Short-Term Memory (LSTM) networks, are well-suited for anomaly detection in time-series data, which is commonly encountered in IoT systems [80]. LSTM networks are designed to capture long-term dependencies and patterns in sequential data, making them well-suited for modeling and anomaly detection in time-series scenarios [26].

In the context of anomaly detection, LSTM networks can be trained on historical time-series data representing normal behavior. During the training process, the LSTM learns to model the underlying patterns and dependencies in the data. Once trained, the LSTM can be used to make predictions on new, unseen data points. Deviations between the predicted values and the actual values can be flagged as potential anomalies, indicating a departure from the learned normal behavior.

2.3 IoT Data Generation, Streaming, and Storage

Effective data generation, streaming, and storage are critical components of any Internet of Things (IoT) system. This section explores the key elements involved in these processes, starting with the physical sensor devices that generate the raw data. The Texas Instruments CC2650 SensorTag is highlighted as a versatile and powerful tool for IoT data collection, with its array of sensors capable of capturing various environmental parameters [34]. The MQTT protocol and its associated brokers are then discussed, highlighting their importance in facilitating efficient and scalable machine-to-machine communication and message distribution within IoT systems [16]. Ad-
ditionally, the section delves into the roles of Apache Kafka and Apache Cassandra, which are widely adopted technologies for streaming data processing and storage, respectively [73]. These components form the backbone of modern IoT pipelines, enabling the ingestion, processing, and persistence of vast amounts of data generated by connected devices.

2.3.1 Physical Sensor Devices

The TI CC2650 SensorTag is a powerful tool in the realm of IoT, providing a wealth of data from its array of 10 low-power MEMS sensors [79]. These sensors collect data on ambient light, ambient temperature, air pressure, and humidity. This diverse range of data collection makes the SensorTag highly versatile, capable of being utilized in a variety of IoT applications.

The SensorTag’s data can be particularly useful in IoT systems due to its ability to connect to the cloud via Bluetooth low energy [34]. This allows for real-time data collection and analysis, which can be crucial in many IoT applications. For example, the SensorTag could be used in a home automation system, where its sensors could provide data on environmental conditions like temperature and humidity. This data could then be used to control heating and cooling systems, optimizing energy usage and improving comfort.

Moreover, the SensorTag is designed to be expandable with DevPacks, making it easy to add additional sensors or actuators [5]. This means that the SensorTag can be customized to collect the specific data needed for a particular IoT application. For instance, additional environmental sensors could be added for use in a weather monitoring system, or a motion sensor could be added for use in a security system.
2.3.2 MQTT Broker

MQTT (Message Queuing Telemetry Transport) is a lightweight protocol that supports the Internet of Things (IoT) and machine-to-machine (M2M) communication. It is designed to be highly efficient, reliable, and scalable, making it well-suited for IoT scenarios where devices may have limited resources and operate on unreliable or low-bandwidth networks [68]. MQTT follows a publish-subscribe messaging pattern, which decouples the message senders (publishers) from the message receivers (subscribers), allowing for asynchronous communication and efficient message distribution.

In the MQTT architecture, publishers and subscribers communicate through topics, which are hierarchical strings that represent different data streams or areas of interest [31]. Publishers send messages to specific topics, while subscribers express interest in one or more topics to receive messages published to those topics. This topic-based messaging model allows for flexible and scalable communication, as publishers and subscribers can dynamically join or leave topics without disrupting the overall system.

MQTT Brokers are central to the functioning of the MQTT protocol, acting as intermediary entities that enable MQTT clients (publishers and subscribers) to communicate. An MQTT Broker functions as a central hub, efficiently handling the flow of messages between devices and applications [8]. Specifically, an MQTT broker receives messages published by clients, filters the messages by topic, and distributes them to subscribers interested in those topics.

The role of MQTT Brokers in IoT systems is crucial. They facilitate communication between MQTT clients, ensuring efficient and accurate delivery of messages. MQTT Brokers can handle a large number of simultaneous connections, which is essential for IoT and M2M communication scenarios involving thousands or even millions of
connected devices. This ability to manage connections and messages enables the MQTT protocol to scale effectively. Additionally, MQTT Brokers provide security measures like authentication and encryption to ensure that the data transmitted between IoT devices and applications is secure.

### 2.3.3 Apache Kafka

Apache Kafka, an open-source distributed event and stream-processing platform, was developed to handle demanding real-time data feeds [25]. It was created by former LinkedIn data engineers and released to the open-source community in 2011 as a highly scalable messaging system. Today, it is part of the Confluent Stream Platform and handles trillions of events every day.

The key design principles of Kafka were formed based on the growing need for high-throughput architectures that are easily scalable and provide the ability to store, process, and reprocess streaming data. Its reliability, ease of use, and fault tolerance have led to its widespread adoption.

Kafka is designed to collect, process, store, and integrate data at scale. It supports numerous use cases, including distributed streaming, stream processing, data integration, and publish-subscribe (pub/sub) messaging [77].

Kafka is built around the concept of modeling events as key/value pairs, where the key represents a unique identifier for the event, and the value is the actual data payload. This event-centric approach aligns well with the nature of IoT systems, which generate a continuous stream of events from various sources [72]. Moreover, Kafka is based on the abstraction of a distributed commit log, which serves as an immutable, fault-tolerant sequence of records ordered by time. This design choice allows Kafka to scale out systems horizontally by splitting a log into multiple partitions, each of
which can be replicated across multiple brokers for fault tolerance and load balancing [74]. The partitioning mechanism enables Kafka to handle large volumes of data and high throughput workloads, making it a suitable choice for ingesting and processing the vast amounts of data generated by IoT pipelines.

2.3.4 Apache Cassandra

Apache Cassandra is a free and open-source, distributed, wide-column store, NoSQL database management system. It was initially developed at Facebook by Avinash Lakshman and Prashant Malik to power the Facebook inbox search feature. Cassandra was designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure [12].

Cassandra offers support for clusters spanning multiple data centers, with asynchronous masterless replication allowing low latency operations for all clients. It implements a combination of Amazon’s Dynamo distributed storage and replication techniques combined with Google’s Bigtable data and storage engine model [67, 10].

In July 2008, Facebook released Cassandra as an open-source project. In March 2009, it became an Apache Incubator project. Today, Cassandra is trusted by thousands of companies for scalability and high availability without compromising performance [22]. Its linear scalability, proven fault-tolerance on commodity hardware or cloud infrastructure, and no single point of failure make it an ideal platform for mission-critical data.
2.4 Performance Metrics in IoT Systems

Internet of Things (IoT) systems are complex and distributed systems that involve various components, including sensors, gateways, communication networks, and cloud infrastructure. Monitoring and analyzing performance metrics is crucial for ensuring the reliability, efficiency, and optimal operation of these systems. Performance metrics provide insights into the system’s behavior, resource utilization, and potential bottlenecks, enabling proactive measures to be taken to mitigate issues and maintain desired service levels.

In the context of IoT systems, performance metrics can be collected from various sources, including hardware components (e.g., sensors, gateways), communication networks, and cloud infrastructure [77, 75, 48]. These metrics can encompass various aspects of the system, such as data transmission rates, network utilization, processing power, memory usage, and storage performance. By analyzing these metrics, system administrators and developers can gain valuable insights into the overall health and performance of the IoT system, enabling them to identify and address potential anomalies or bottlenecks before they escalate into more significant issues.

CPU utilization is a critical performance metric that provides insights into the processing load on the system components. High CPU utilization can indicate resource contention, leading to performance degradation or system instability. Monitoring CPU utilization can help identify bottlenecks and optimize resource allocation [50].

Disk utilization metrics, including disk read/write rates and disk queue lengths, are essential for understanding the storage performance of IoT systems. Excessive disk activity or long disk queues can cause delays in data ingestion, processing, or retrieval, impacting the overall system performance [40].
Network utilization is another crucial metric, as IoT systems heavily rely on efficient and reliable data transmission between devices, gateways, and cloud services. Monitoring network throughput, packet loss, and latency can help detect network congestion or connectivity issues, enabling timely mitigation and ensuring seamless data flow within the system [7].

By combining these performance metrics with other system-level metrics and contextual information, system administrators and developers can gain a comprehensive understanding of the IoT system’s overall performance and identify potential correlations or root causes of performance issues.

2.5 Symbolic Encoding

Symbolic encoding techniques have emerged as powerful tools for representing and analyzing time-series data, particularly in scenarios where dimensionality reduction and noise filtering are essential. These techniques transform raw time-series data into symbolic representations, capturing essential patterns and trends while reducing the data’s complexity and computational overhead [45]. The symbolic representations can then be leveraged for various applications, including anomaly detection, pattern discovery, and data mining. By transforming complex time-series data into symbolic sequences, these techniques enable the application of powerful string-based algorithms and analysis methods, facilitating the identification of anomalous patterns, recurring motifs, and hidden insights within the data. Furthermore, the compact and interpretable nature of symbolic representations makes them suitable for efficient storage, transmission, and processing, which is particularly advantageous in resource-constrained environments common in IoT systems.
2.5.1 Piecewise Aggregate Approximation

Piecewise Aggregate Approximation (PAA) is a fundamental technique for time-series data representation and dimensionality reduction [29]. It involves dividing the time-series into equal-sized segments and representing each segment by its mean value. This process results in a reduced-dimensionality representation of the original time-series, effectively filtering out noise and capturing the overall shape and trends. PAA serves as a building block for more advanced symbolic encoding techniques and is often used as a preprocessing step.

The dimensionality reduction in PAA is achieved by replacing the original data points within each segment with a single mean value [57]. For example, if a time-series of length \( n \) is divided into \( w \) equal-sized segments, the resulting PAA representation will have a dimensionality of \( w \), where each dimension corresponds to the mean value of the respective segment, displayed in Figure 2.1. This reduction in dimensionality is particularly useful when dealing with high-resolution time-series data, as it can significantly reduce the computational complexity and storage requirements for subsequent processing steps, while still preserving the essential characteristics of the original data.

The PAA algorithm is computationally efficient and can be applied to both univariate and multivariate time-series data. By reducing the dimensionality of the data, PAA not only facilitates more efficient storage and transmission but also simplifies subsequent data mining and analysis tasks. Although PAA achieves dimensionality reduction, the resulting representation remains in the numerical domain, making it suitable for further processing by symbolic encoding techniques like SAX.
2.5.2 Symbolic Aggregate Approximation

Symbolic Aggregate Approximation (SAX) is a widely adopted symbolic representation technique that builds upon PAA [43]. After applying PAA to the time-series data, SAX transforms the reduced-dimensionality representation into a string of symbols. This transformation is achieved by discretizing the PAA representation into a predefined alphabet, typically using a Gaussian distribution or other statistical techniques, as shown in Figure 2.2. Discretizing involves mapping the continuous numerical values of the PAA representation to a finite set of discrete symbols or characters from an alphabet. The resulting symbolic representation captures the essential patterns and trends in the time-series while providing a compact and interpretable representation suitable for various data mining tasks.

One of the key advantages of SAX is its ability to enable efficient similarity comparisons and anomaly detection through the use of simple operations on symbolic strings. By representing time-series data as symbolic strings, SAX allows for the application of string-based algorithms and techniques, such as pattern matching, clustering, and frequent pattern mining. This symbolic representation also facilitates efficient storage and transmission, making SAX particularly well-suited for applications in resource-constrained environments, such as IoT systems.

In this research, we leverage the advantages of SAX for symbolically encoding the performance metrics data collected from an IoT pipeline. By transforming the time-series performance data into symbolic strings, SAX enables the generation of consistent and uniform symbolic sequences that represent normal and anomalous events. This symbolic representation is particularly beneficial in the context of IoT systems, where the underlying hardware and configurations can vary significantly. By decoupling the anomaly detection process from the specific IoT pipeline configuration, the
use of SAX allows a trained machine learning model to generalize across different setups without the need for retraining on vast amounts of IoT data every time the configuration changes. This approach not only improves the efficiency and scalability of the anomaly detection process but also reduces the variability and complexity typically associated with raw IoT datasets. Furthermore, the symbolic nature of SAX facilitates the application of string-based algorithms and techniques, enabling efficient similarity comparisons, pattern matching, and anomaly detection, even in resource-constrained IoT environments.

2.6 Related Work

Anomaly detection in Internet of Things (IoT) systems has gained significant attention due to the growing complexity and scale of these distributed systems. Researchers have explored various techniques and approaches to address the challenges posed by the heterogeneous nature of IoT data, the need for real-time monitoring, and the scalability requirements of large-scale deployments. Symbolic representations, particularly SAX, have emerged as powerful tools for dimensionality reduction and pattern extraction in time-series data. Additionally, machine learning solutions have shown promise in developing data-driven anomaly detection models capable of adapting to the dynamic nature of IoT environments. This section provides an overview of relevant studies and methodologies proposed in the literature, highlighting their contributions and limitations in the context of anomaly detection for IoT systems, with a focus on symbolization techniques and machine learning approaches.
Figure 2.1: Example Application of Piecewise Aggregate Approximation.

Figure 2.2: Example Application of Symbolic Aggregate Approximation.
2.6.1 IoT Architecture for Data Processing

Designing efficient and scalable architectures for data processing in IoT environments is a critical challenge that has garnered significant attention. One prominent solution is the utilization of Apache Kafka, a distributed streaming platform renowned for its scalability, fault tolerance, and real-time data processing capabilities. [74] presents a replicated logging system built upon Apache Kafka, demonstrating its applicability in handling massive volumes of data with high availability. Moreover, [66] highlights Kafka as a modern platform for data management and analysis in the big data domain, emphasizing its suitability for building resilient and high-throughput data pipelines in IoT and other data-intensive applications.

Complementing the data ingestion and streaming capabilities of Kafka, Apache Cassandra has emerged as a prominent distributed NoSQL database system for managing large-scale, decentralized datasets generated by IoT devices and sensors. [22] proposes an IoT sensor data acquisition and storage system that leverages Raspberry Pi and Apache Cassandra, showcasing the integration of these technologies for efficient data collection and storage in IoT deployments. Furthermore, [12] presents a big data modeling methodology specifically tailored for Apache Cassandra, providing insights into designing and optimizing data models for handling the diverse and high-volume data streams characteristic of IoT environments.

These studies collectively highlight the adoption of Apache Kafka and Apache Cassandra as key components in building scalable and fault-tolerant data processing architectures for IoT systems. Kafka’s distributed streaming capabilities and Cassandra’s decentralized storage and data management features complement each other, enabling the construction of robust and high-performance data pipelines essential for handling the massive volumes of data generated by IoT devices and sensors.
By leveraging these technologies, researchers and practitioners aim to address the challenges of data ingestion, storage, and processing in IoT environments, enabling real-time analysis, anomaly detection, and decision-making based on the insights derived from the vast amounts of data generated by interconnected devices and systems.

2.6.2 Performance Anomalies in IoT Systems

Ensuring optimal performance and detecting anomalies in IoT systems is crucial for maintaining reliable and efficient operations. Several studies have explored this domain, providing valuable insights and methodologies. [70] presents a performance analysis of an IoT-based sensor, big data processing, and machine learning model for real-time monitoring in automotive manufacturing. This work highlights the importance of evaluating system performance in the context of IoT deployments, addressing challenges such as data processing, analytics, and anomaly detection.

Extending the focus to anomaly detection in IoT datasets, [56] conducts a performance analysis of different anomaly detection techniques using cloud microservices. This study sheds light on the impact of cloud-based architectures and microservices on the performance of anomaly detection algorithms, a critical aspect in ensuring the timely identification and mitigation of anomalies in IoT environments.

As messaging brokers play a pivotal role in IoT communication infrastructures, [48] offers a comparative analysis of performance measurements for stress-testing MQTT brokers. This research explores the performance characteristics of various MQTT brokers under varying load conditions, providing insights into their scalability and reliability in handling high-volume data streams typical of IoT systems.

Complementing the analysis of messaging brokers, [77] focuses on performance prediction for the Apache Kafka messaging system. Given the widespread adoption of
Kafka in IoT data pipelines, this study contributes to the understanding of Kafka’s performance characteristics and enables proactive capacity planning and resource allocation, ultimately enhancing the reliability and efficiency of IoT data processing architectures.

These works collectively highlight the importance of performance analysis and anomaly detection in IoT systems, spanning various components such as sensors, data processing pipelines, machine learning models, messaging brokers, and cloud infrastructures. By addressing performance bottlenecks, optimizing resource utilization, and detecting anomalies in a timely manner, researchers and practitioners aim to enhance the overall reliability, responsiveness, and scalability of IoT deployments, enabling real-time monitoring, predictive maintenance, and data-driven decision-making in diverse applications.

2.6.3 ML for IoT Anomaly Detection

The application of machine learning (ML) techniques for anomaly detection in Internet of Things (IoT) systems has garnered significant attention from researchers and practitioners. [11] presents a comprehensive survey of IoT anomaly detection methods and applications, providing a broad overview of the current state-of-the-art and highlighting the potential of ML in this domain.

Several studies have explored the use of advanced neural network architectures for anomaly detection in IoT environments. [78] investigates the application of Graph Neural Networks (GNNs) for anomaly detection in the Industrial Internet of Things (IIoT), leveraging the ability of GNNs to capture complex relationships and dependencies within IoT data. Similarly, [53] proposes an enhanced network anomaly de-
tection approach based on deep neural networks, demonstrating the effectiveness of these models in identifying anomalous patterns in IoT network traffic.

Recognizing the importance of secure and reliable IoT systems, [30] reviews the application of machine learning techniques for attack and anomaly detection in IoT networks. This work highlights the potential of ML in identifying malicious activities, intrusions, and anomalous behaviors that could compromise the integrity and performance of IoT deployments.

Extending the scope to IoT data analysis, [1] presents a comprehensive review of machine learning and deep learning techniques for anomaly detection in IoT data. This study covers a wide range of algorithms and methodologies, providing insights into their applicability, strengths, and limitations in the context of IoT anomaly detection tasks.

Complementing these works, [61] offers a survey of outlier detection techniques specifically tailored for IoT environments. This research provides a classification and comparative analysis of various outlier detection algorithms, enabling practitioners to select appropriate techniques based on the characteristics of their IoT data and specific use cases.

These studies collectively demonstrate the growing interest and potential of machine learning techniques in addressing the challenges of anomaly detection in IoT systems. By leveraging advanced algorithms, neural network architectures, and data-driven approaches, researchers aim to enhance the reliability, security, and performance of IoT deployments, enabling real-time monitoring, predictive maintenance, and proactive decision-making based on the insights derived from IoT data streams.
2.6.4 SAX for Anomaly Detection and Classification

The application of symbolic representation techniques, particularly Symbolic Aggregate Approximation (SAX), has gained significant traction in the domain of anomaly detection and classification for time series data across various domains. One notable contribution is the work presented in [35], where the authors introduced the HOT SAX algorithm, an efficient approach for finding the most unusual time series subsequence by leveraging SAX and a novel data structure called the Hierarchical Ordering Tiling (HOT).

Several studies have explored the use of SAX for anomaly detection in symbolic time series representations of reduced dimensionality [9, 37]. These works demonstrated that by transforming raw time series data into symbolic representations using SAX, it is possible to achieve dimensionality reduction while preserving essential patterns and trends, facilitating efficient anomaly detection and classification, particularly in scenarios where high-dimensional data poses computational challenges.

The effectiveness of SAX in capturing patterns and anomalies across various time series datasets has been extensively evaluated and compared to alternative methods [13]. These comparative studies reinforced the suitability of SAX for anomaly detection tasks, outperforming other techniques in terms of accuracy and computational efficiency, particularly in scenarios involving large-scale time series data.

The application of SAX has also been explored in specific domains, such as biomedical signal analysis [81, 55]. [81] proposed a novel approach for anomaly detection in electrocardiogram (ECG) signals using a variant of SAX called Trend Symbolic Aggregate Approximation (TSAX), incorporating trend information into the symbolic representation for improved detection accuracy. Similarly, [55] explored the applica-
tion of SAX for the identification of heart disorders from ECG signals, showcasing the potential of SAX in biomedical signal processing and disease diagnosis.

To address scenarios involving multiple correlated data streams, such as in IoT and sensor network applications, [6] introduced MSAX, the Multivariate Symbolic Aggregate Approximation algorithm. This technique enables the symbolic representation of multidimensional time series, facilitating anomaly detection and classification in multivariate settings.

Furthermore, [36] demonstrated the efficacy of symbolic time-series analysis using SAX for anomaly detection in mechanical systems. By transforming vibration and sensor data into symbolic representations, the authors were able to identify anomalous patterns indicative of potential mechanical failures, enabling predictive maintenance and condition monitoring.

These studies collectively highlight the versatility and effectiveness of SAX and symbolic representation techniques for anomaly detection and classification across diverse domains, including IoT, biomedical, mechanical systems, and time series analysis applications.

2.7 Novelty of This Work

The research presented in this thesis addresses several limitations and gaps identified in the existing literature on anomaly detection in IoT systems. While previous studies have explored various aspects of IoT architectures, performance analysis, machine learning techniques, and symbolic representations, our work presents a novel and integrated approach to tackle the challenges of generalized anomaly detection in IoT environments.
Existing studies on IoT data processing architectures have primarily focused on individual components, such as Apache Kafka for data streaming [74] or Apache Cassandra for distributed storage [22]. However, our research proposes an end-to-end IoT Pipeline that seamlessly integrates virtual sensor simulators, an MQTT broker, Apache Kafka, and Apache Cassandra, enabling the generation, ingestion, and storage of realistic IoT data streams.

Furthermore, our work addresses the need for performance analysis and anomaly detection in IoT systems by leveraging system-level performance metrics from the Kafka and Cassandra instances [70]. By utilizing these performance metrics as input features, we eliminate the reliance on collected sensor data, thereby reducing the variability and complexity typically associated with raw IoT datasets.

Recognizing the potential of machine learning for anomaly detection in IoT [78, 53, 20], our research introduces a novel approach by combining symbolic representations, specifically SAX, with LSTM neural networks. While previous studies have explored the use of SAX for anomaly detection in various domains [81, 13, 9], our work is the first to apply SAX for symbolically encoding system performance metrics in IoT environments, enabling the development of generalized anomaly detection models.

By leveraging the properties of SAX, such as dimensionality reduction, noise resilience, and pattern extraction capabilities [52, 35, 6], our approach aims to capture the essential characteristics of system behavior while abstracting away the underlying hardware configurations and sensor data characteristics. This symbolic representation facilitates the training of LSTM models that can generalize across diverse IoT deployments, addressing the limitations of existing techniques that rely on specific system configurations or sensor data profiles.
Moreover, our research explores both univariate and multivariate approaches, evaluating the performance of LSTM models trained on symbolized representations of individual performance metrics (univariate) as well as combinations of multiple metrics (multivariate). This comprehensive analysis contributes to a deeper understanding of the trade-offs and considerations involved in selecting appropriate input features for anomaly detection in IoT systems.

By integrating diverse components, leveraging system performance metrics, and employing a novel combination of symbolic representations and machine learning techniques, our work paves the way for developing generalized and scalable anomaly detection solutions for IoT deployments, addressing the limitations of existing approaches and contributing to the advancement of reliable and efficient IoT systems.

2.8 Summary

This chapter covers key concepts for anomaly detection in IoT systems, including performance metrics like network utilization and types of performance anomalies. It explores machine learning techniques for anomaly classification, both supervised (SVMs, Decision Trees, Neural Networks) and unsupervised (clustering, dimensionality reduction, novelty detection). Additionally, symbolic encoding methods like PAA, SAX, MSAX, and HOT SAX are discussed for representing and analyzing time-series data efficiently while preserving patterns.
In this research, we devise a theoretical IoT Pipeline comprising various components, including physical sensor devices, an MQTT broker, Apache Kafka for data streaming, and Apache Cassandra for data storage. This theoretical pipeline serves as a blueprint for our experimental setup. To validate our approach and collect relevant data, we implement three distinct IoT Testbeds – small, medium, and large – which are specifically configured instances of the theoretical IoT Pipeline. These Testbeds are designed to simulate real-world IoT systems at different scales, allowing us to generate and collect performance metrics data under varying conditions. By analyzing the data from these Testbeds, we can evaluate the effectiveness of our proposed anomaly detection method and its ability to generalize across different IoT configurations.

The IoT pipeline implemented in this research comprises several key stages, as illustrated in Fig. 3.1. The pipeline begins with data generation, where virtual sensors emulate the behavior of real-world IoT devices, explored in Section 2.3.1, which represent realistic IoT pipeline data generation. producing time-series data streams representing various operational scenarios, including both normal and anomalous conditions. The generated data is then ingested into the pipeline through an MQTT broker, which facilitates seamless communication between the virtual sensors and downstream components. Apache Kafka, a distributed streaming platform, serves as the next stage, enabling real-time data ingestion, buffering, and parallel processing. Subsequently, the data is persisted in a scalable database system, Apache Cassan-
dra, ensuring reliable storage and retrieval capabilities for downstream analytics and anomaly detection workflows.

This research leverages a comprehensive IoT Testbed to simulate and evaluate the proposed anomaly detection framework. The testbed, shown in Fig. 3.2, is an implementation of the generic IoT Pipeline displayed in Fig. 3.1. The testbed emulates a realistic Industry 4.0 pipeline, facilitating the generation, processing, and analysis of IoT data streams under various operational conditions.

The IoT Testbed is designed to simulate an end-to-end IoT pipeline, encompassing data generation, transmission, storage, and analysis components. Virtual sensors are employed to generate time-series data emulating real-world scenarios, including both normal and anomalous behavior patterns. This data is then transmitted through a messaging broker and ingested into a distributed streaming platform, mimicking the data flow in an actual IoT deployment. Subsequently, the data is persisted in a scalable database system, enabling downstream analysis and anomaly detection processes.

![Figure 3.1: Implemented IoT Pipeline.](image-url)
3.1 Components of the IoT Pipeline

The proposed IoT Pipeline consists of several key components that work together to facilitate data generation, streaming, and storage within an IoT system. These components include physical sensor devices responsible for collecting raw data, a messaging protocol and broker for efficient machine-to-machine communication, a streaming platform for processing and distributing data in real-time, and a distributed database for storing and retrieving the collected data. The following subsections provide an overview of each component, highlighting their roles and significance within the IoT Pipeline architecture.

Figure 3.2: Implemented IoT Testbed.

3.1.1 Sensors

IoT sensors represent a fundamental component of modern interconnected systems, facilitating the acquisition of real-world data for a myriad of applications. Tracing their origins to the convergence of sensor technology and internet connectivity, the evolution of IoT sensors can be attributed to the emergence of ubiquitous computing in the early 2000s [71]. Since then, advancements in miniaturization, energy efficiency, and wireless communication have propelled the proliferation of IoT sensors across
various domains, ranging from smart homes and industrial automation to healthcare and environmental monitoring. These sensors are designed to capture a diverse array of physical phenomena, including temperature, humidity, motion, light, and more, enabling the generation of actionable insights and informed decision-making within IoT ecosystems [79].

In the context of this research project, IoT sensors play a pivotal role in generating data which serves as a workload for our pipeline. Initial development and testing of our devised IoT Pipeline was completed using TI SensorTag CC2650’s, as described in Section 2.3.1 which are popular industry-grade devices intended for developer usage [5]. Subsequently, our research employs virtual sensor simulators which emulate the sensor data generation process, enabling the simulation of diverse environmental conditions and scenarios. The pipeline achieves a comprehensive and configurable data generation framework, a crucial prerequisite for the robust evaluation and validation of anomaly detection algorithms within the simulated IoT pipeline.

For our IoT Testbed, we created Python scripts¹ to represent our virtual sensor module, which can emulate the behavior of N physical sensors, where N is a configurable parameter based on the IoT pipeline infrastructure. In our work, we used a maximum of N = 75. The virtual sensors generate data including battery percentage, temperature, light, humidity, pressure, and time readings. These virtual sensors transmit data at a configurable frequency, with a normal transmission rate of six seconds. To introduce anomalous patterns, we implement three different scenarios: increased transmission frequency (four seconds), decreased transmission frequency (eight seconds), and inconsistent transmission periods ranging randomly between four and eight seconds. These values were selected for our specific testbeds as they generated a steady workload, resulting in a baseline CPU utilization of approximately 30%. As

¹Github: https://github.com/mpatel6262/ThesisProject
shown in Fig. 3.2, the generated data is published to a dedicated topic hosted on our MQTT Broker, emulating a continuous data stream. While hosted on an AWS EC2 instance for convenience, the virtual sensor module is not inherently tied to a specific infrastructure, as long as the system can support generating data from N sensors concurrently.

3.1.2 MQTT Broker

MQTT brokers operate on a publish-subscribe messaging paradigm, where IoT devices (publishers) communicate data to the broker, which then distributes this data to interested subscribers [68]. This asynchronous messaging model minimizes network overhead and ensures timely delivery of messages, making MQTT brokers a cornerstone of scalable and responsive IoT architectures [31]. Over the years, MQTT has gained widespread adoption across various industries, driving interoperability and standardization efforts within the IoT landscape. In the context of this research project, MQTT brokers play a central role in facilitating seamless communication and data transmission within the simulated Industry 4.0 pipeline [48].

To facilitate seamless communication between the virtual sensors and the downstream data processing components, the testbed employs HiveMQ [16], a reliable and scalable MQTT broker. HiveMQ serves as the intermediary, hosting the ”VirtualSensorData” topic and enabling the virtual sensors to stream data to the Apache Kafka instance, as shown in Figure 3.2. This integration of MQTT and Kafka technologies allows for efficient and fault-tolerant data ingestion, ensuring reliable data delivery from the simulated sensors to the data persistence layer.
3.1.3 Apache Kafka

At its core, Kafka employs a distributed commit log architecture, wherein data is stored durably and replicated across multiple nodes within a cluster [66]. This architecture enables Kafka to handle massive volumes of data while ensuring low-latency message delivery and fault tolerance.

As sensor data is generated by the virtual sensors within the IoT Testbench, Kafka provides a robust and scalable platform for collecting, buffering, and distributing this data to downstream processing systems [74]. By leveraging Kafka’s distributed nature and fault-tolerant design, the project ensures reliable and low-latency data transmission, a crucial prerequisite for the accurate evaluation of anomaly detection algorithms. Furthermore, Kafka’s support for stream processing frameworks such as Apache Flink and Apache Spark enables seamless integration with advanced analytics and machine learning pipelines, facilitating real-time data analysis and anomaly detection [72]. By harnessing the power of Apache Kafka, the project establishes a resilient and efficient data pipeline, capable of handling the demanding requirements of Industry 4.0 environments and enabling actionable insights from IoT data streams.

At the core of our testbed’s data ingestion and streaming layer lies an instance of Apache Kafka, as depicted in Figure 3.2, Apache Kafka is hosted on an AWS EC2 instance. Kafka is configured with one topic partitioned into five partitions, ensuring parallel data processing and fault tolerance. The MQTT Broker topic is subscribed to, and data is pulled from the MQTT broker and buffered on the Apache Kafka server. Network utilization data is collected from this instance using the collectl utility, providing valuable insights into the system’s performance during data ingestion. Other resource utilization metrics, such as CPU and disk utilization, were discarded due to their inconsistency and inaccuracy in identifying performance anomalies com-
pared to network utilization, which proved to be a more reliable indicator of system performance in our testbed setup.

3.1.4 Kafka Consumers

Apache Kafka is a distributed streaming platform that allows for the reliable and scalable transfer of data between applications and systems. In a Kafka architecture, producers publish data to topics, which are partitioned and replicated across multiple brokers for fault tolerance and scalability. Consumers, on the other hand, are responsible for reading and processing data from these topics.

To enable efficient data processing and prevent bottlenecks, the testbed employs five Apache Kafka consumers hosted on an AWS EC2 instance. Consumers are applications or processes that subscribe to one or more topics and continuously consume (read) data from the partitions they are assigned to. In this case, the consumers stream data in parallel from the Apache Kafka instance to the Apache Cassandra database, leveraging the partitioned topic structure for parallelism. By having multiple consumers, the workload of consuming and processing data can be distributed, improving overall throughput and reducing the risk of bottlenecks.

CPU utilization data is collected from this instance using the sar utility, allowing for comprehensive monitoring of the system’s performance during data consumption and processing [23]. Monitoring the CPU utilization is crucial as it provides insights into the resource consumption and potential bottlenecks that may arise during the data processing pipeline.
3.1.5 Apache Cassandra

Cassandra’s tunable consistency levels and eventual consistency model render it particularly well-suited for use cases that demand high throughput and low-latency data access, such as IoT data management, time-series data, and real-time analytics [54]. Apache Cassandra’s ability to scale linearly and handle high write throughput makes it an ideal choice for ingesting and storing the massive volumes of data generated by IoT devices and sensors. Its decentralized, masterless architecture ensures that there is no single point of failure, enhancing the system’s fault tolerance and availability [12].

Cassandra’s flexible data model, which supports column-oriented storage and wide rows, is well-suited for storing and querying time-series data, a common data structure in IoT applications. Furthermore, Cassandra’s tunable consistency levels allow for a trade-off between consistency and availability, enabling applications to prioritize either strong consistency or high availability based on their specific requirements [22]. This flexibility is particularly valuable in real-time analytics scenarios, where low-latency data access is crucial for timely decision-making and incident response. Additionally, Cassandra’s support for geographically distributed clusters and data replication facilitates the deployment of IoT systems across multiple regions, ensuring data locality and minimizing latency for globally distributed applications.

Leveraging these capabilities, the testbed’s data persistence layer is powered by an instance of Apache Cassandra, hosted on the AstraDB managed service [15]. Cassandra serves as the final destination for the data generated by the virtual sensors, providing a scalable and fault-tolerant storage solution capable of handling the testbed’s workload and data throughput requirements. Additionally, Cassandra’s linear scalability allows for seamless expansion of the cluster as the testbed’s data storage needs grow,
making it an ideal choice for handling the ever-increasing volume of data generated by IoT systems. The AstraDB managed service further simplifies the deployment and management of the Cassandra instance, allowing the testbed to focus on its core functionality while benefiting from a fully-managed and optimized Cassandra environment.

3.2 Induced IoT Anomalies

IoT systems fall vulnerable to anomalies that can significantly impact the transmission frequency of data emanating from sensors, leading to disruptions in the seamless flow of information and potential delays in delivering critical insights. These anomalies often manifest as irregularities in the timing and frequency of data packets transmitted by sensors to central processing units or cloud-based servers [11]. As mentioned previously in Section 2.1, we introduced three forms of performance-based anomalies:

- Increased data frequency anomalies displaying when sensors transmit data at unexpectedly high rates, overwhelming system processing capabilities.
- Decreased data frequency anomalies which manifest when sensors fail to transmit data at the expected rate, leading to gaps or delays in data delivery.
- Inconsistent data frequency anomalies which represent unpredictable variations in transmission rates, causing irregularities in the flow of information.

Detecting and mitigating these anomalies that impact data transmission frequency is paramount for ensuring the reliability, responsiveness, and efficient performance of IoT systems [17]. By focusing on these specific anomalies, this research endeavor aims to develop targeted anomaly detection techniques tailored to identifying and addressing disruptions in data transmission.
3.2.1 Increased Data Frequency

Anomalies can manifest as unexpected increases in the transmission frequency of data between sensors and central processing units, disrupting the normal cadence of information flow. In our testbed, these anomalies were introduced by modifying the configuration scripts that controlled the virtual sensor simulators. Specifically, we reduced the transmission interval for the simulated sensor data, causing the sensors to transmit data more frequently than the normal operating conditions [44]. This change resulted in an increased volume of data ingested by the Kafka broker and subsequently processed by the Cassandra database, leading to higher resource utilization and potential bottlenecks in the system. However, the increase in transmission frequency was calibrated to a level that did not cause severe performance degradation, making these anomalies challenging to detect using simple threshold-based methods. By monitoring the performance metrics collected from the Kafka and Cassandra instances, we verified the manifestation of these anomalies through elevated network utilization and intermittent spikes in CPU usage.

3.2.2 Decreased Data Frequency

Anomalies can manifest as unexpected decreases in the transmission frequency of data between sensors and central processing units, disrupting the normal flow of information. In our testbed, we injected these anomalies by modifying the configuration scripts to increase the transmission interval for the simulated sensor data [14]. This change resulted in a reduced volume of data ingested by the Kafka broker and processed by the Cassandra database, leading to underutilization of system resources. Again, the decrease in transmission frequency was controlled to avoid severe performance degradation, making these anomalies challenging to detect using sim-
ple threshold-based methods. By analyzing the performance metrics collected from the Kafka and Cassandra instances, we verified the manifestation of these anomalies through decreased network utilization and periods of low CPU usage.

### 3.2.3 Inconsistent Data Frequency

The intricate nature of IoT systems renders them susceptible to anomalies that can manifest as variations in the transmission frequency of data between sensors and central processing units. In our testbed, we introduced these anomalies by incorporating randomness into the configuration scripts that controlled the virtual sensor simulators. Specifically, we implemented a random transmission interval within a specified range, causing the sensors to transmit data at varying frequencies [24]. This change resulted in fluctuations in the volume of data ingested by the Kafka broker and processed by the Cassandra database, leading to intermittent spikes and dips in resource utilization. By monitoring the performance metrics collected from the Kafka and Cassandra instances, we verified the manifestation of these anomalies through irregular patterns in network utilization and CPU usage, reflecting the inconsistent data transmission patterns.

### 3.3 Summary

This chapter talked about the configuration of an IoT Pipeline including the individual components and respective roles in data processing as well as the implemented IoT Testbeds which were developed from the aforementioned IoT Pipeline. Additionally, there were also discussion on the specific choices for where each stage of the pipeline was hosted. Finally, methods for emulating IoT sensors and three unique performance anomalies are outlined.
Chapter 4

METHODOLOGY

This research proposes a comprehensive methodology for anomaly detection and classification in IoT pipelines, detailed in Figure 4.1. It involves generating simulated IoT data exhibiting anomalous transmission frequency patterns using virtual sensor frameworks and inducing factors like sensor malfunctions, network issues, and environmental disturbances. The collected data undergoes preprocessing steps such as cleaning, normalization, and feature extraction. Symbolic Aggregate Approximation (SAX) is then applied to obtain symbolic representations that capture intrinsic patterns while reducing dimensionality [43]. These symbolic sequences are fed into machine learning models utilizing Long Short-Term Memory (LSTM) networks, which are trained to accurately identify and categorize anomalies in transmission frequency. By integrating virtual simulations, symbolic encoding techniques, and advanced machine learning algorithms, this multifaceted approach aims to advance anomaly detection methods for enhancing reliability and resilience in Industry 4.0 IoT systems.

The research employs three different IoT testbeds, categorized as small, medium, and large, to evaluate the proposed approach’s effectiveness and generalizability across varying system configurations. The small testbed represents a minimal setup with a lower data throughput rate. Conversely, the large testbed simulates a more complex and resource-intensive scenario with a higher data throughput rate. The medium testbed falls between these two extremes, representing a moderately sized IoT deployment. These three testbeds are designed to mimic real-world IoT systems with varying scales and complexities, allowing for a comprehensive evaluation of the proposed anomaly detection approach. By training the machine learning model on the medium
testbed data and evaluating its performance on all three testbeds, the methodology aims to assess the model’s ability to generalize and accurately classify anomalies in unseen IoT system configurations, thereby demonstrating the robustness and versatility of the proposed solution.

4.1 Dataset Generation and Preprocessing

Virtual sensors were employed to emulate the behavior of TI SensorTag CC2650 devices, generating time-series data representing various scenarios of interest. These virtual sensors were configured to transmit data either at a normal frequency, mimicking regular operations, or at anomalous frequencies, simulating potential performance issues.

The anomalous data transmission periods were categorized into three distinct classes: increased transmission frequency, decreased transmission frequency, and variable transmission frequency. These categories were designed to capture a wide range of anomalous behaviors that could arise in real-world IoT deployments, such as network congestion, sensor malfunctions, or environmental disturbances [64].
In addition to the simulated sensor data, system-level performance metrics were collected from the Apache Kafka and Apache Cassandra instances, which served as the message broker and data storage components, respectively, within the IoT pipeline. These instances were hosted on AWS EC2. Performance metrics were collected at a frequency of 1 second from the Kafka instance, as discussed in Section 3.1.3 and the Cassandra instance, as discussed in Section 3.1.5.

Specifically, the sar (System Activity Report) and collectl (Collect System Data) utilities were utilized to monitor and record CPU utilization and network utilization data from these critical components [23].

The generated sensor data and collected system performance metrics were subsequently collated and labeled according to the corresponding period’s behavior: normal or anomalous. This labeling process facilitated the creation of a comprehensive dataset, encompassing both regular and anomalous scenarios, essential for training and evaluating the anomaly detection algorithms.

By combining simulated sensor data with real-world system performance metrics, this data generation and processing framework aimed to accurately represent the complexities and challenges inherent in IoT pipelines [61]. The resulting dataset captured the intricate interplay between sensor data transmission patterns and system-level performance indicators, enabling the development and assessment of robust anomaly detection techniques tailored to the unique characteristics of IoT environments.

Each data point in the dataset can be represented as a set $S = \{ \text{Feature 1, Feature 2, ..., Feature N} \}$, where the features are collected from the resource utilizations of the underlying IoT Testbed. While our experiments focused on models trained on one or two performance metrics (CPU and network utilization), our design allows for
the inclusion of any number of features (N) to capture a comprehensive view of the system’s behavior.

The testbeds were run for a total of six hours, with data points collected every second. During this period, the testbed cycled through four operational states – one representing normal operation, one with higher transmission frequency, one with lower transmission frequency, and one with inconsistent transmission frequencies – every six minutes. This process resulted in the collection of 21,600 data points from each of the three testbeds (small, medium, and large), providing a diverse and representative dataset for training and evaluating our anomaly detection models.

4.2 Symbolic Encoding

After data generation, the time series data underwent a preprocessing step before symbolic transformation. Preprocessing involved collating the collected data points from various sources (sensor data, Kafka metrics, Cassandra metrics) and labeling each data point based on the operational state of the testbed at that time. Specifically, the labels indicated whether the data point corresponded to normal operation, higher transmission frequency, lower transmission frequency, or inconsistent transmission frequencies. This labeling process was essential for supervised learning, as it provided the ground truth information for the anomaly detection models to learn the patterns associated with each operational state.

Once the data was collated and labeled, it underwent a symbolic transformation process using the Symbolic Aggregate Approximation (SAX) technique. The SAX algorithm characterizes each data point into a discrete symbol based on its value relative to the entire dataset, ensuring an equal distribution of symbols across the data
To explore the impact of different symbolic encoding parameters on anomaly detection performance, multiple symbolic representations were generated.

The number of unique symbols used in the SAX representation was varied from 3 to 5. This variation allowed us to assess the impact of different levels of granularity in the symbolic encoding on the performance of our anomaly detection models. The bin boundaries, or in other words, the boundaries between symbols are set by mapping an equal amount of data points to each symbol. For example, if we used an alphabet size of 5, 20% of data points would be mapped to each of the five symbols. A higher number of symbols (e.g., 5) provides a more fine-grained representation, potentially capturing subtle patterns and variations in the data. Conversely, a lower number of symbols (e.g., 3) results in a coarser representation, potentially missing out on some nuances but potentially being more robust to noise [35].

Table 4.1 illustrates the application of SAX on a sample of twenty data points representing measured CPU utilization percentages. The table demonstrates the transformation of the original numerical time-series data into symbolic representations using different alphabet sizes (3, 4, and 5 symbols).

The first column displays the raw CPU utilization values, where the subsequent columns show the corresponding symbolic representations obtained after applying SAX with alphabet sizes of 3, 4, and 5 symbols, respectively.

In the SAX representation, each numerical value is mapped to a symbolic character from the chosen alphabet. For instance, when using an alphabet size of 3 symbols (e.g., 'A', 'B', and 'C'), the CPU utilization values are discretized into three equal-sized regions, and each value is assigned the corresponding symbol based on the region it falls into.
The table highlights how the symbolic representations change as the alphabet size increases, providing varying levels of granularity in capturing the patterns and trends within the time-series data. With a larger alphabet size, the symbolic representation becomes more detailed, potentially preserving more nuanced information from the original data.

Table 4.1: Representative Table of Data Points After Applying SAX on the Collected Time-Series Data

<table>
<thead>
<tr>
<th>Measured CPU Utilization (%)</th>
<th>SAX with 3 Symbols</th>
<th>SAX with 4 Symbols</th>
<th>SAX with 5 Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>28.65</td>
<td>B</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>30.52</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>31.25</td>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>25.37</td>
<td>A</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>32.83</td>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>24.80</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>29.23</td>
<td>B</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>33.95</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>23.87</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>25.26</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>27.15</td>
<td>B</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>25.11</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>26.15</td>
<td>A</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>29.83</td>
<td>B</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>24.84</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>35.71</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>26.95</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>37.35</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>37.46</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>33.54</td>
<td>C</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

Additionally, we systematically varied the number of data points aggregated into a single symbol, with values ranging from 1 to 10. This parameter is formally known as the Piecewise Aggregate Approximation (PAA) segment size. A smaller segment size (e.g., 1) means that each symbol represents a single data point, resulting in a highly granular but potentially noisy representation. On the other hand, a larger segment size (e.g., 10) means that each symbol represents an aggregation of multiple data points, leading to a smoother and potentially more robust representation, but potentially missing out on some fine-grained details.
The specific range of 1 to 10 for the segment size was chosen to explore a broad spectrum of granularity levels, from the most granular (1) to a reasonably coarse representation (10). This range allowed us to investigate the trade-off between capturing fine-grained patterns and noise robustness, and to identify the optimal segment size for our anomaly detection task.

The number of PAA segments was set to the total number of data points divided by the chosen segment size. For example, if we had 100 data points and a segment size of 5, the number of PAA segments would be 100 / 5 = 20. This means that the time series would be divided into 20 segments, and each segment would be represented by a single symbol based on the average value of the data points within that segment.

By varying both the number of symbols and the segment size, we aimed to find the optimal combination that balances the trade-off between pattern recognition and noise robustness, ultimately enhancing the performance of our anomaly detection models on the symbolized time series data.

This multi-parametric approach allowed for a comprehensive exploration of the symbolic representation space, facilitating the identification of optimal parameter configurations that yield the most accurate characterization of the system’s performance and underlying anomalies.

By transforming the time series data into symbolic sequences, the algorithm captured the essential patterns and trends while reducing the dimensionality of the dataset. This symbolic representation not only facilitated efficient analysis and classification but also enabled the direct interpretation of patterns and irregularities by humans [55].

The symbolic data representations were then utilized as inputs for the subsequent anomaly detection and classification phases, enabling the evaluation of machine learn-
ing models’ performance across a diverse range of symbolic encodings. This systematic exploration of the symbolic representation space aimed to uncover the optimal balance between precision and interpretability, ultimately enhancing the robustness and effectiveness of anomaly detection within the simulated IoT pipeline [62].

### 4.3 Long Short-Term Memory Classification Models

The symbolic data representations obtained from the SAX transformation were subsequently utilized as inputs for training Long Short-Term Memory (LSTM) classifier models. Two distinct algorithms were developed: a univariate approach and a multivariate approach, tailored to explore the effectiveness of leveraging different combinations of input features. The performance of both of these models are reported and compared in Section 5.3.

#### 4.3.1 Univariate LSTM Model

In the univariate algorithm, the symbolic sequences derived from the CPU utilization data of the IoT Pipeline were considered. CPU utilization was selected as the univariate metric because it provided the most accurate representation of the testbed’s performance with a single metric. Instead of treating each symbol as an isolated instance, overlapping windows of varying lengths were constructed from the symbolic representations, providing context for the time-series dependencies in CPU utilization. Overlapping windows capture sequential segments of the symbolic data, with the window length determining the amount of temporal context considered. The chosen range allows the exploration of different levels of context, from individual symbols (window length 1) to lengthier sequences. This approach aims to identify the optimal amount of temporal information required for effective anomaly detection, as shorter
windows may suffice for simple patterns, while longer windows may be necessary to capture more complex anomalies manifesting over extended periods.

This approach recognizes that individual symbols in the symbolized data do not exist in isolation but are part of a larger sequence representing the system’s behavior over time. By creating overlapping windows, the model can capture the temporal relationships and patterns within the symbolized CPU utilization data.

For each row of training data, instead of providing a single symbol, overlapping windows were constructed by considering a symbol along with its preceding and succeeding symbols within a specified window length. For example, with a window length of 5, the input to the model would be a sequence of 5 consecutive symbols. This approach, which is outlined in Fig. 4.2, aimed to capture the dynamic patterns within the symbolic data, enhancing the model’s ability to identify and classify anomalies effectively [26]. The model itself comprises an embedding layer, which converts the symbolic sequences into vector representations, followed by an LSTM layer that captures the temporal dependencies, and a dense layer that classifies the data into four classes: 0 for normal operation, 1 for increased transmission frequency, 2 for decreased transmission frequency, and 3 for inconsistent transmission frequency.

Figure 4.2: Implemented Univariate LSTM Model.
4.3.2 Multivariate LSTM Model

The multivariate algorithm extended the input features by incorporating symbolic representations from both the CPU utilization and network utilization of the IoT Pipeline. Similar to the univariate model, overlapping windows were constructed, providing context for the time-series dependencies within each symbolic sequence. However, in the multivariate case, these overlapping windows were created for multiple symbol streams simultaneously.

The model architecture followed a similar structure as the univariate model, with an embedding layer, LSTM layer, and dense layer for classification. However, the embedding layer handled multiple input sequences, one for each performance metric, before concatenating the embedded vectors and feeding them into the LSTM layer. This approach allowed the model to learn the temporal dependencies and patterns within each symbolic sequence while capturing the interactions between different performance metrics.

By combining these complementary performance metrics, the multivariate model aimed to leverage the synergistic effects of multiple data streams, potentially improving the detection and classification of anomalies within the IoT pipeline [80]. The approach, visualized in Fig. 4.3, provided a more comprehensive representation of the system’s behavior, enabling more accurate anomaly detection and classification.

4.3.3 Training and Evaluation

Both LSTM models were trained using the data collected from the medium IoT testbed, which provided a representative sample of the system’s behavior under various conditions. Hyperparameter tuning was performed to optimize the model per-
formance, with parameters such as the number of units, dropout rates, recurrent dropout rates, batch size, and overlapping window length being systematically varied and evaluated [65]. The different hyperparameters which were modified to optimize for performance are outlined in Table 4.2.

Table 4.2: Explored Hyperparameter Configurations for LSTM Models

<table>
<thead>
<tr>
<th>Parameter Names</th>
<th>Values Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlapping Window Length</td>
<td>1, 3, 5, 7, 9, 11, 13, 15</td>
</tr>
<tr>
<td>Batch Size</td>
<td>32, 64, 128</td>
</tr>
<tr>
<td>Dropout Rate</td>
<td>0.15, 0.2, 0.25, 0.3</td>
</tr>
<tr>
<td>Recurrent Dropout Rate</td>
<td>0.15, 0.2, 0.25, 0.3</td>
</tr>
<tr>
<td>LSTM Units</td>
<td>32, 64, 128</td>
</tr>
</tbody>
</table>

The models were initially trained using 5-fold cross-validation on the dataset collected from the medium IoT testbed, which served as a baseline configuration for training the anomaly detection models. The use of 5-fold cross-validation during the training phase allowed for a comprehensive evaluation of the models’ performance and generalization capabilities on the medium testbed dataset. By splitting the dataset into 5 distinct folds, the models were trained and evaluated multiple times, with each fold

Figure 4.3: Implemented Multivariate LSTM Model.
serving as the test set once. This approach helped to mitigate issues such as overfitting and provided a reliable estimate of the models’ performance on unseen data from the medium testbed configuration.

To further assess the generalization capabilities of the trained models, evaluations were conducted using datasets collected from the small and large IoT testbeds, respectively. This approach ensured a comprehensive assessment of the models’ performance across diverse scenarios, ranging from small-scale deployments to large, complex IoT pipelines. By exploring both univariate and multivariate LSTM architectures, this stage of the methodology aimed to uncover the optimal combination of input features and model configurations for accurate anomaly detection and classification within the simulated IoT environment. The systematic evaluation across multiple testbeds further ensured the robustness and scalability of the proposed anomaly detection framework, paving the way for practical applications in real-world IoT deployments.

4.4 Summary

This chapter explained the methodology of our research including dataset generation, preprocessing, symbolic encoding through SAX, and anomaly classification utilizing a LSTM model. Additional details regarding the training and evaluation of our LSTM models are also provided.
Chapter 5

RESULTS

5.1 Evaluation Metrics

To comprehensively evaluate the performance of the proposed anomaly detection approach and assess its efficacy in addressing the stated research questions, several evaluation metrics were employed. These metrics provide quantitative measures to analyze the model’s ability to accurately classify anomalies in IoT systems while considering different aspects of its performance.

The evaluation results obtained using these metrics will be presented and analyzed in the subsequent sections, facilitating a thorough examination of the research questions and the efficacy of the proposed solution in achieving robust and generalizable anomaly detection in IoT systems.

5.1.1 Symbolization

To evaluate the effectiveness of SAX in capturing time series patterns and anomalies consistently across the small, medium, and large IoT testbeds, we employed the Mindist metric which is explored in detail in Equation (5.1). Mindist calculates the distance between symbolic sequences generated by SAX, quantifying the similarity between their representations. The MINDIST between two SAX representations of a time series is calculated by first computing the distance between a pair of symbols, an example lookup table is shown in Table 5.1. Subsequently, those distances are squared, summed, square rooted, and ultimately multiplied by the compression rate’s
square root. By comparing the Mindist values for the symbolic sequences derived from each testbed, we could assess the degree to which SAX consistently encoded the underlying patterns and anomalies present in the time series data, regardless of the testbed’s scale or complexity. Lower Mindist values between the symbolic representations of different testbeds would indicate a higher degree of consistency, suggesting that SAX could effectively capture the essential characteristics of the time series data, enabling the application of a single classification model trained on one testbed to accurately generalize and perform anomaly detection on the other testbeds.

Figure 5.1: Computing Pairwise MINDIST Values

\[
\text{MINDIST}(\hat{Q}, \hat{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (\text{dist}(\hat{q}_i, \hat{c}_i))^2}
\]  

\(5.1\)

- \(\hat{Q}\) = Symbolic representation of first time series
- \(\hat{C}\) = Symbolic representation of second time series
- \(n\) = Number of data points in each time series
- \(w\) = Number of PAA segments in each symbolic representation
- \(\hat{q}_i\) = Single symbol from first time series
• \( \hat{c}_i = \) Single Symbol from second time series

• \( \text{dist}(\hat{q}_i, \hat{c}_i) = \) character distance(\(\hat{q}_i, \hat{c}_i\)) * 4 / a, where a represents the number of unique symbols. For example, dist('a', 'c') where the alphabet consists of three symbols = 2 * 4 / 3 = 8/3

Table 5.1: Lookup Table Used for Dist Function for Four Symbols

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

5.1.2 Anomaly Classification

To evaluate the effectiveness of our proposed approach and investigate the generalization capabilities of the trained models, we conducted a series of experiments. Initially, we trained an LSTM classifier using the symbolized performance metrics data collected from the medium IoT testbed. This testbed represented a typical configuration of our IoT Pipeline, serving as a baseline for training the anomaly detection model.

The performance of the trained LSTM model was evaluated using cross-validation (CV) accuracy and F1-score metrics, defined in Equations (5.2) and (5.5) respectively, on the medium testbed data, which it was directly trained on. These metrics provided insights into the model’s ability to accurately classify anomalies in the symbolized data streams from the medium testbed configuration.

Subsequently, we ran the trained LSTM classifier on symbolized data collected from varying underlying IoT configurations, specifically the small and large testbeds. The generalized test accuracies and F1-scores on these testbeds were used to determine
the model’s ability to generalize and detect anomalies in symbolized data streams originating from different IoT setups, without the need for retraining on new datasets each time the underlying configuration changed.

To establish a baseline for comparison, we also trained a separate LSTM model directly on the raw, non-symbolized performance metrics data collected from the medium testbed. The performance of this baseline model was evaluated using the same cross-validation and generalized test metrics as our proposed approach.

By comparing the results obtained from our symbolized data approach with the baseline LSTM model trained on raw data, we aimed to assess the benefits and trade-offs of using symbolic representations generated through SAX for enabling effective anomaly detection and generalization across diverse IoT testbed configurations.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.2)
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (5.3)
\]

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (5.4)
\]

\[
F1\text{Score} = \frac{2 * TP}{2 * TP + FP + FN} \quad (5.5)
\]

- TP = The number of correctly predicted positive class values
- TN = The number of correctly predicted negative class values
• FP = The number of incorrectly predicted positive class values

• FN = The number of incorrectly predicted negative class values

5.2 Symbolization Results

Through the analysis of Mindist values, we were able to determine that SAX could generate similar distance values for each of the three pairwise comparisons between the symbolic sequences derived from the small, medium, and large IoT testbeds. Notably, tabulated in Tables 5.2, 5.3, and 5.4, each individual distance value fell within 10% of the average for the given number of symbols and PAA segments. This similarity in distance values across the testbeds demonstrated SAX’s ability to consistently encode the patterns and anomalies present in the time series data, regardless of the testbed’s scale or complexity. Notably, we observed that the consistency across the three symbolic encoding streams increased as the size of the Piecewise Aggregate Approximation (PAA) segments used in SAX increased. The bolded MINDIST values represent the best performing PAA segment length values. Larger PAA segments facilitated a more consistent symbolic representation by aggregating multiple data points into a single symbol, effectively smoothing out local fluctuations and noise. This smoothing effect helped SAX capture the essential characteristics and underlying patterns of the time series data more effectively, while mitigating the impact of transient variations or outliers that could potentially obscure the broader trends. By representing the time series using larger PAA segments, we enabled SAX to provide a more stable and robust symbolic encoding, enhancing its ability to accurately characterize the system’s behavior and enable effective anomaly detection in our IoT application.
RQ 2: Our research demonstrates the feasibility of symbolically representing time-series data using Symbolic Aggregate Approximation (SAX), as we found MINDIST values within 10% of the average across testbeds for a given workload.

During our experiments, we observed that the number of symbols used for symbolic representation through SAX had a significant impact on the training loss and model effectiveness as shown in Figure 5.5. Models trained on symbolized data with a higher number of symbols, such as 5 symbols, exhibited lower training losses compared to models trained on data with fewer symbols (3 or 4 symbols).

This behavior can be attributed to the increased precision and granularity provided by a larger symbol alphabet. With more symbols available, the symbolic representation can capture finer details and nuances in the patterns present within the time-series data, enabling the LSTM model to better distinguish and learn the intricate patterns associated with normal and anomalous behaviors.

However, it is important to balance the number of symbols with the computational complexity and memory requirements of the model, as using a larger number of symbols may increase the computational overhead and memory footprint, which can be a concern in resource-constrained IoT environments.

**Table 5.2: MINDIST Comparison for 3 Symbols**

<table>
<thead>
<tr>
<th>PAA Segment Length</th>
<th>Small &amp; Medium</th>
<th>Medium &amp; Large</th>
<th>Small &amp; Large</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>199.87</td>
<td>197.19</td>
<td>208.71</td>
<td>201.93</td>
</tr>
<tr>
<td>2</td>
<td>164.78</td>
<td>164.98</td>
<td>177.18</td>
<td>168.98</td>
</tr>
<tr>
<td>3</td>
<td>147.32</td>
<td>147.05</td>
<td>159.95</td>
<td>151.44</td>
</tr>
<tr>
<td>4</td>
<td>138.47</td>
<td>138.56</td>
<td>149.61</td>
<td>142.21</td>
</tr>
<tr>
<td>5</td>
<td>130.15</td>
<td>130.43</td>
<td>143.95</td>
<td>134.85</td>
</tr>
<tr>
<td>6</td>
<td>122.01</td>
<td>123.99</td>
<td>135.01</td>
<td>127.00</td>
</tr>
<tr>
<td>10</td>
<td>106.75</td>
<td>108.66</td>
<td>114.59</td>
<td><strong>110.00</strong></td>
</tr>
</tbody>
</table>
Table 5.3: MINDIST Comparison for 4 Symbols

<table>
<thead>
<tr>
<th>PAA Segment Length</th>
<th>Small &amp; Medium</th>
<th>Medium &amp; Large</th>
<th>Small &amp; Large</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>183.19</td>
<td>182.20</td>
<td>189.15</td>
<td>184.85</td>
</tr>
<tr>
<td>2</td>
<td>155.66</td>
<td>153.90</td>
<td>163.26</td>
<td>157.61</td>
</tr>
<tr>
<td>3</td>
<td>143.33</td>
<td>141.50</td>
<td>152.38</td>
<td>145.74</td>
</tr>
<tr>
<td>4</td>
<td>133.74</td>
<td>135.65</td>
<td>143.49</td>
<td>137.63</td>
</tr>
<tr>
<td>5</td>
<td>128.09</td>
<td>128.33</td>
<td>138.07</td>
<td>131.50</td>
</tr>
<tr>
<td>6</td>
<td>121.41</td>
<td>123.77</td>
<td>126.74</td>
<td>123.97</td>
</tr>
<tr>
<td>10</td>
<td>109.11</td>
<td>109.19</td>
<td>111.38</td>
<td>109.89</td>
</tr>
</tbody>
</table>

Table 5.4: MINDIST Comparison for 5 Symbols

<table>
<thead>
<tr>
<th>PAA Segment Length</th>
<th>Small &amp; Medium</th>
<th>Medium &amp; Large</th>
<th>Small &amp; Large</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>176.83</td>
<td>176.58</td>
<td>182.12</td>
<td>178.51</td>
</tr>
<tr>
<td>2</td>
<td>147.70</td>
<td>147.90</td>
<td>151.14</td>
<td>148.91</td>
</tr>
<tr>
<td>3</td>
<td>132.20</td>
<td>133.30</td>
<td>136.69</td>
<td>134.06</td>
</tr>
<tr>
<td>4</td>
<td>125.06</td>
<td>126.08</td>
<td>130.11</td>
<td>127.08</td>
</tr>
<tr>
<td>5</td>
<td>118.61</td>
<td>119.26</td>
<td>123.78</td>
<td>120.55</td>
</tr>
<tr>
<td>6</td>
<td>118.48</td>
<td>118.71</td>
<td>119.99</td>
<td>119.06</td>
</tr>
<tr>
<td>10</td>
<td>103.70</td>
<td>106.15</td>
<td>107.22</td>
<td>105.69</td>
</tr>
</tbody>
</table>

Pairwise Mindist Values for 3 Symbols

![Pairwise Mindist Values for 3 Symbols](image)

Figure 5.2: Pairwise Mindist Values for 3 Symbols.
Pairwise Mindist Values for 4 Symbols

![Graph showing pairwise Mindist values for 4 symbols.]

Figure 5.3: Pairwise Mindist Values for 4 Symbols.

Pairwise Mindist Values for 5 Symbols

![Graph showing pairwise Mindist values for 5 symbols.]

Figure 5.4: Pairwise Mindist Values for 5 Symbols.
5.3 Anomaly Classification Results

The univariate and multivariate machine learning models were trained by using five-fold cross-validation for the model training on the medium IoT testbed. Testing was measured on the small, medium, and large IoT testbeds. This approach allowed for a comprehensive assessment of the models’ generalization capabilities across varying scales and configurations. The overlapping window length for the input sequences was varied from 1 to 15, while also modifying other hyperparameters, including batch size, dropout, and recurrent dropout rates, as outlined in Table 4.2. This systematic exploration of the hyperparameter space aimed to identify the optimal configurations for accurate anomaly detection and classification.

The results demonstrated in Figures 5.6 and 5.7 highlight the ability of the proposed models to accurately identify anomalies in the IoT system, relying solely on the performance metrics collected from the system, without requiring direct sensor data. Additionally, we were able to show that our approach does not sacrifice significant performance for the dataset it was trained on when compared to our baseline models trained on unsymbolized data. We achieved univariate and multivariate CV accuracies of 93.14% and 95.87%, which both fall within 2% of our baseline models’ accuracies of 94.14% and 97.64%, respectively. This approach not only simplifies the data acquisition and preprocessing steps but also reduces the variability and complexity inherent in typical IoT datasets, which often comprise heterogeneous sensor data streams. By leveraging system performance metrics as the primary input, the models can effectively capture the manifestations of anomalies within the system, enabling reliable anomaly detection and classification.
**RQ 1:** The incorporation of system performance metrics proved effective in training an LSTM model for classifying anomalies in IoT systems, yielding CV accuracies of 93.14% and 95.87% for our Medium IoT Testbed.

The confusion matrices displayed in Tables 5.5 and 5.6 compare the model’s performance in classifying anomalies across the four classes: normal transmission frequency (0), increased transmission frequency (1), decreased transmission frequency (2), and inconsistent transmission frequency (3). The confusion matrix reveals that the model performed best in classifying instances of normal operation, with the highest true positive rate of 5,237 and 5,313 out of 5,400 instances correctly identified for the univariate and multivariate models, respectively. The model exhibited diminished performance for the increased and decreased transmission frequency classes; however, a substantial portion of instances were still accurately classified in these categories. However, the model struggled the most with accurately detecting instances of inconsistent transmission frequency anomalies, as evident from the relatively higher misclassifications for this class (4,818 and 5,069 true positives out of 5,400 instances).

The superior performance in classifying normal operation can be attributed to the model’s ability to learn and recognize the consistent patterns associated with regular system behavior effectively. The increased and decreased transmission frequency classes, while representing anomalous behavior, may exhibit more distinguishable and consistent patterns compared to the inconsistent transmission frequency class, allowing the model to generalize and classify them with reasonable accuracy. In contrast, inconsistent transmission frequency anomalies may manifest in diverse and irregular ways, making it challenging for the model to capture and generalize the underlying patterns reliably, leading to higher misclassifications for this class.

Furthermore, the results also demonstrate the ability of the models to generalize to similar IoT testbeds with differing underlying hardware configurations. This general-
Figure 5.5: Average Training Loss by Symbols and PAA Segment Numbers.

Figure 5.6: Univariate Model CV Accuracy on Medium IoT Testbed by Overlapping Window Length.
Figure 5.7: Multivariate Model CV Accuracy on Medium IoT Testbed by Overlapping Window Length.

Table 5.5: Confusion Matrix for Univariate Model Applied to Medium IoT Testbed

<table>
<thead>
<tr>
<th></th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Label 0</td>
</tr>
<tr>
<td>Predicted</td>
<td></td>
</tr>
<tr>
<td>Label 0</td>
<td>5237</td>
</tr>
<tr>
<td>Label 1</td>
<td>27</td>
</tr>
<tr>
<td>Label 2</td>
<td>33</td>
</tr>
<tr>
<td>Label 3</td>
<td>103</td>
</tr>
</tbody>
</table>
ization capability underscores the benefits of symbolization as a method for enabling the portability of anomaly detection models across diverse IoT systems. By transforming the time-series data into symbolic representations using SAX, the models can learn to recognize patterns and anomalies independent of the specific hardware or system configurations. Consequently, this approach alleviates the need for training multiple individual models tailored to each unique IoT system, thereby enhancing the scalability and efficiency of the proposed anomaly detection framework.

**RQ 3:** Our work produced a trained LSTM model which yielded accuracies of 87.33% and 87.47% on datasets generated from our Small and Large IoT Testbeds, respectively, demonstrating the ability for models to be generalized to differing system configurations with similar workloads.

The results in Figure 5.8 demonstrate the superior performance of our symbolized data approach over the baseline models when generalizing to the small and large IoT testbeds. By leveraging symbolic representations generated through SAX, our LSTM models were able to effectively classify anomalies in symbolized data streams originating from different IoT configurations, without the need for retraining on new datasets. This ability to generalize across varying testbed setups highlights the strength of our approach in decoupling the anomaly detection process from the specific IoT pipeline configuration. The symbolic nature of the data allowed the models to capture essential patterns and trends, enabling them to adapt to diverse operational scenarios and workloads while maintaining high accuracy in anomaly detection.

The results depicted in Figures 5.6 and 5.7 showcase the multivariate models outperformed the univariate models in accurately identifying anomalies within the IoT system, specifically, the multivariate model’s accuracies of 87.33% and 87.47% compared to the univariate model’s accuracies of 82.69% and 82.22% for the small and large IoT testbeds, respectively. By considering multiple variables, the multivariate
Table 5.6: Confusion Matrix for Multivariate Model Applied to Medium IoT Testbed

<table>
<thead>
<tr>
<th>Actual Label</th>
<th>Predicted Label 0</th>
<th>Predicted Label 1</th>
<th>Predicted Label 2</th>
<th>Predicted Label 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label 0</td>
<td>5313</td>
<td>87</td>
<td>81</td>
<td>103</td>
</tr>
<tr>
<td>Label 1</td>
<td>23</td>
<td>5183</td>
<td>8</td>
<td>112</td>
</tr>
<tr>
<td>Label 2</td>
<td>15</td>
<td>11</td>
<td>5177</td>
<td>116</td>
</tr>
<tr>
<td>Label 3</td>
<td>49</td>
<td>119</td>
<td>134</td>
<td>5069</td>
</tr>
</tbody>
</table>

Figure 5.8: Test Accuracies of Univariate and Multivariate Models when Generalized to Small and Large IoT Testbeds.
models gained a more comprehensive understanding of the system dynamics, enabling them to capture complex interdependencies and subtle patterns that may not be discernible when considering only a single metric. This holistic view of the system’s behavior allowed the multivariate models to better differentiate between normal and anomalous states, leading to enhanced anomaly detection accuracy. Additionally, leveraging multiple variables mitigated the impact of outliers or noise in individual metrics, as the models could rely on the collective information from all variables to make more informed decisions.

5.4 Summary

This chapter discussed the evaluation metrics used to contrast the performance of our developed symbolic encoding and anomaly classification models. Results were presented on the distances found between SAX-encoded data streams as well as the accuracies achieved by our univariate and multivariate LSTM models. To conclude, our models demonstrated the ability for generalization across testbeds.
This research endeavor has made significant strides in addressing the critical challenge of anomaly detection within Internet of Things (IoT) pipelines. By leveraging a multifaceted approach that combines virtual sensor simulations, symbolic data representations, and advanced machine learning techniques, we have developed a robust and scalable framework for identifying and classifying anomalies in IoT data streams.

Through the strategic utilization of the IoT Testbench and the integration of virtual sensors, we successfully emulated realistic sensor data patterns, encompassing both normal and anomalous scenarios. The application of Symbolic Aggregate Approximation (SAX) facilitated the transformation of time-series data into symbolic representations, capturing essential patterns while reducing dimensionality, thereby enabling efficient analysis and interpretability.

The development of univariate and multivariate LSTM classifier models demonstrated the effectiveness of leveraging symbolic data representations for anomaly detection and classification. By systematically exploring different input feature combinations and model configurations, we identified optimal architectures tailored to the unique characteristics of IoT environments.

Through rigorous evaluation across multiple testbeds, ranging from small to large scale IoT pipelines, we have demonstrated the generalization capabilities and scalability of our proposed anomaly detection framework. The results obtained highlight the potential for practical applications in diverse Industry 4.0 settings, contributing to enhanced reliability, responsiveness, and overall performance of IoT deployments.
While this research has made substantial contributions, exciting opportunities exist for further advancement. Incorporating diverse data sources, such as operational logs, network traffic, and environmental sensors, could potentially enhance the robustness and comprehensiveness of the anomaly detection framework. Conducting pilot studies and real-world deployments would provide invaluable insights into practical applicability, enabling further refinements and optimizations. Developing mechanisms for continuous learning and adaptation could enable the models to evolve dynamically, adapting to changing patterns and incorporating new data, enhancing resilience and responsiveness.

Leveraging anomaly detection capabilities in conjunction with predictive maintenance techniques could enable proactive maintenance scheduling, minimizing downtime and maximizing operational efficiency. Exploring interpretable machine learning models and explainable AI techniques could provide valuable insights into decision-making processes, fostering trust and facilitating human-in-the-loop decision support systems. By pursuing these future research directions, we can solidify the foundation established, driving innovative solutions that contribute to the realization of resilient, efficient, and secure IoT ecosystems, propelling Industry 4.0 and enabling new frontiers in data-driven decision-making.
BIBLIOGRAPHY


[37] V. Kumar, A. Banerjee, and V. Chandola. Anomaly detection for symbolic sequences and time series data. 2009.


